Exhibit B19 - Invalidity of 12,211,095 in view of CAM

CAM anticipates and/or renders obvious (alone or in combination with other references), at least under USAA's apparent infringement theories, all asserted claims of U.S. Patent No. 12,211,095 ("the '095 patent"). CAM was known, publicly used, and/or on sale in the United States at least as of October 23, 2005 and is therefore prior art to the '095 patent. Example evidence of the design and function of CAM includes, but is not limited to:

- Tapan S. Parikh, "Using Mobile Phones for Secure, Distributed Document Processing in the Developing World," IEEE Pervasive Computing Magazine, Vol. 4, No. 2 (Apr. 2005), available at http://tap2k.org/papers/074-081.pdf ("Pervasive Computing") (**REGIONS_USAA-00005716**);
- Tapan S. Parikh, "CAM: A Mobile Paper-based Information Services Architecture for Remote Rural Areas in the Developing World," IEEE Symposium on Visual Languages and Human-Centric Computing (Sept. 20–24, 2005), available at http://tap2k.org/papers/24430341.pdf ("Parikh 2005a") (**REGIONS_USAA-00005725**);
- Tapan S. Parikh, "CAM: A Mobile Interaction Framework for Digitizing Paper Processes in the Developing World," Doctoral Symposium, ACM Symposium on User Interface Software and Technology (UIST) (Seattle, WA Oct. 23–26, 2005), available at http://tap2k.org/papers/UIST05PhD-parikh.pdf ("Parikh 2005b") (**REGIONS_USAA-00005712**);
- Tapan S. Parikh & Paul Javid, "CAM: A Mobile Interaction Framework for Digitizing Paper Processes in the Developing World," Demonstration Video (Seattle, WA Oct. 23–26, 2005), https://uist.acm.org/archive/adjunct/2005/videos/demos/d5-parikh.mov ("Video") (**REGIONS_USAA-00005357**); and
- Tapan S. Parikh, Designing an Architecture for Delivering Mobile Information Services to the Rural Developing World, Ph.D. Dissertation, Univ. of Wash. (2007), available at http://tap2k.org/papers/parikh-thesis.pdf ("Parikh 2007") (**REGIONS_USAA-00005535**).

Regions has charted the portions of the reference that correspond to each element, including at least under USAA's apparent interpretation of the claims as reflected in USAA's Complaint and Infringement Contentions. Where the chart below states that the reference "discloses," "teaches," or "suggests" a claim limitation, such disclosure, teaching, or suggestion may be express, implied, or inherent. The cited evidence is merely illustrative, and Regions reserves the right to cite alternative or additional evidence.

No part of this exemplary chart construes, or is intended to construe, the specification, file history, or claims of the asserted patents. In addition, nothing stated in this chart shall be treated as an admission or suggestion that Regions agrees with USAA regarding either the scope of any of the asserted claims or Regions' apparent interpretation of the scope of the asserted claims, or that Regions' accused products meet any limitations of the claims.

EXHIBIT 1084
REGIONS v. USAA
IPR2025-01356

ls that CAM does not disclose one or more limitations of the asserted claims, it would have been obvious to combine
ıe knowledge of one of ordinary skill in the art, and/or (2) the teachings of any of the prior art set forth in Regions'
pleading, to show all the limitations of the asserted claims, including for the reasons set forth in Regions' Invalidity
ich are incorporated herein.

vs that it would have been obvious to combine CAM with the prior art set forth in Regions' other invalidity charts
by showing similarities with the other prior art and/or elements that could be improved by combination,
ı in view of the other prior art.

Exhibit B19 - Invalidity of 12,211,095 in view of CAM

CAM anticipates and/or renders obvious (alone or in combination with other references), at least under USAA's apparent infringement theories, all asserted claims of U.S. Patent No. 12,211,095 ("the '095 patent"). CAM was known, publicly used, and/or on sale in the United States at least as of October 23, 2005 and is therefore prior art to the '095 patent. Example evidence of the design and function of CAM includes, but is not limited to:

- Tapan S. Parikh, "Using Mobile Phones for Secure, Distributed Document Processing in the Developing World," IEEE Pervasive Computing Magazine, Vol. 4, No. 2 (Apr. 2005), available at http://tap2k.org/papers/074-081.pdf ("Pervasive Computing") (**REGIONS_USAA-00005716**);
- Tapan S. Parikh, "CAM: A Mobile Paper-based Information Services Architecture for Remote Rural Areas in the Developing World," IEEE Symposium on Visual Languages and Human-Centric Computing (Sept. 20–24, 2005), available at http://tap2k.org/papers/24430341.pdf ("Parikh 2005a") (**REGIONS_USAA-00005725**);
- Tapan S. Parikh, "CAM: A Mobile Interaction Framework for Digitizing Paper Processes in the Developing World," Doctoral Symposium, ACM Symposium on User Interface Software and Technology (UIST) (Seattle, WA Oct. 23–26, 2005), available at http://tap2k.org/papers/UIST05PhD-parikh.pdf ("Parikh 2005b") (**REGIONS_USAA-00005712**);
- Tapan S. Parikh & Paul Javid, "CAM: A Mobile Interaction Framework for Digitizing Paper Processes in the Developing World," Demonstration Video (Seattle, WA Oct. 23–26, 2005), https://uist.acm.org/archive/adjunct/2005/videos/demos/d5-parikh.mov ("Video") (**REGIONS_USAA-00005357**); and
- Tapan S. Parikh, Designing an Architecture for Delivering Mobile Information Services to the Rural Developing World, Ph.D. Dissertation, Univ. of Wash. (2007), available at http://tap2k.org/papers/parikh-thesis.pdf ("Parikh 2007") (**REGIONS_USAA-00005535**).

Regions has charted the portions of the reference that correspond to each element, including at least under USAA's apparent interpretation of the claims as reflected in USAA's Complaint and Infringement Contentions. Where the chart below states that the reference "discloses," "teaches," or "suggests" a claim limitation, such disclosure, teaching, or suggestion may be express, implied, or inherent. The cited evidence is merely illustrative, and Regions reserves the right to cite alternative or additional evidence.

No part of this exemplary chart construes, or is intended to construe, the specification, file history, or claims of the asserted patents. In addition, nothing stated in this chart shall be treated as an admission or suggestion that Regions agrees with USAA regarding either the scope of any of the asserted claims or Regions' apparent interpretation of the scope of the asserted claims, or that Regions' accused products meet any limitations of the claims.

To the extent that USAA contends that CAM does not disclose one or more limitations of the asserted claims, it would have been obvious to combine the teachings of CAM with (1) the knowledge of one of ordinary skill in the art, and/or (2) the teachings of any of the prior art set forth in Regions' other invalidity charts and cover pleading, to show all the limitations of the asserted claims, including for the reasons set forth in Regions' Invalidity Contentions Cover Pleading, which are incorporated herein.

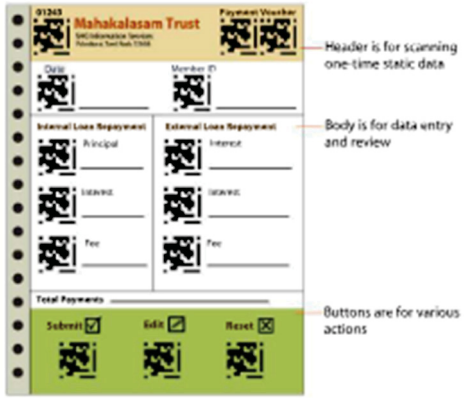
The cited disclosure further shows that it would have been obvious to combine CAM with the prior art set forth in Regions' other invalidity charts and cover pleading, for example by showing similarities with the other prior art and/or elements that could be improved by combination, replacement, and/or modification in view of the other prior art.

Because USAA has yet to identify any limitation of the asserted claims that it contends is not fully disclosed by CAM, either alone or in combination with other prior art cited by Regions, Regions expressly reserves the right to rebut any such contention, including by identifying additional § 103 combinations, if any such contention is made by USAA.

Claims	**Prior Art**
1[pre]. A system enabling mobile check deposit auto-capture functionality via video frame processing, the system comprising:	To the extent the preamble is limiting, CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[a] system enabling mobile check deposit auto-capture functionality via video frame processing, the system." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images

Claims	Prior Art
	taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world.

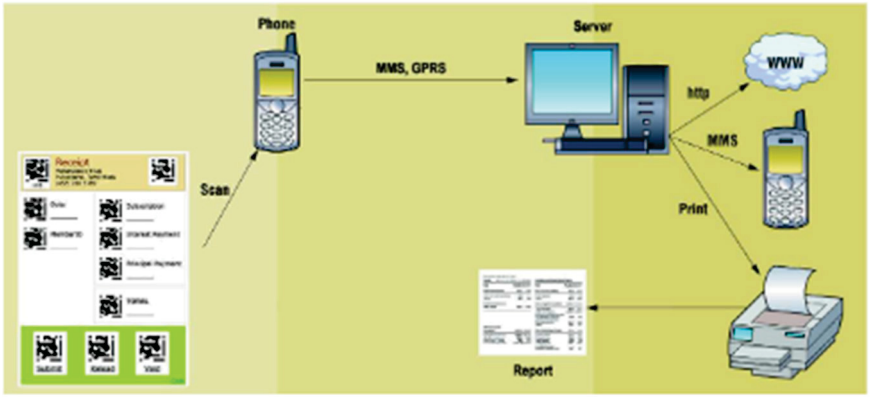
Claims	Prior Art
	CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 In contrast to other tangible UI programming toolkits [2, 9], CAM is not an API for software developers to build different physical user interfaces and applications. CAM is more like the World Wide Web, in that it unites a user interface, a programming language and a delivery mechanism for accessing networked information services. Like the web, our goal is to allow many different services to be developed and deployed using a common infrastructure. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell is the programming language that is used to define interaction with visual codes embedded in the form. There is one visual code in every CamForm that serves as a form identifier. This code has a 0 as the lowest-order bit. The next 4 bits identify the form to the CamBrowser application. This information is used to load the form description schema, an XML file containing metadata about the form and the underlying data elements (including any default or pre-existing values). The remaining bits specify the

Claims	Prior Art
	protocol that should be used to load the schema file (currently HTTP, bluetooth or the local filesystem), and the required information to find the schema (a network address or a file name). CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people

Claims	Prior Art
	living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID.

Claims	Prior Art
	CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 54–55 The same application can be accessed by entering a numeric string using the phone keypad. This is useful for mobile phones without a camera, or for applications that do not have an associated paper representation. The numeric string is based on a phone number, providing a familiar metaphor for rural users. An application is "dialed" by entering an arbitrary length phone number (or 12-digit IP address) for the server, a '*' delimiter (two '*'s for an IP address), and then a server-specific application ID. If the address or phone [Table 4.1] number is omitted, it is treated as a local application that may have been downloaded to the phone previously (the equivalent of accessing localhost in a web browser). CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value.

Claims	Prior Art
	CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the

Claims	Prior Art
	entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 55 (Table 4.1) Table 4.1: Barcode and numeric inputs for accessing CAM applications and functions. *Application* Barcode: 00-Protocol(3 bits)-Addr(38)-AppID(24) Numeric: [Addr-*]-AppID-[FuncID[#Param]] *Function* Barcode: 11-FuncID(8 bits)-Param(66) Numeric: FuncID[#Param] CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2)

Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images.

input_int, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text*	Prompts asking the user to enter different kinds of values
record_audio, *take_picture*	Ask the user to record an audio clip, or take a picture of a form or other object
message_note, *conf_note*	Send a message to the user, or display a confirmation dialog
get, *put*	Retrieve or submit application data to the server using using http, sms or mms
encrypt, *decrypt*	Encrypt or decrypt application data with a specified key
sms, *mms*, *email*	Send arbitrary messages
phonecall, *browser*	Make a phone call or launch a web browser
log	Write a string to the application log

CAM (Parikh 2007) at 60 (Figure 4.5)

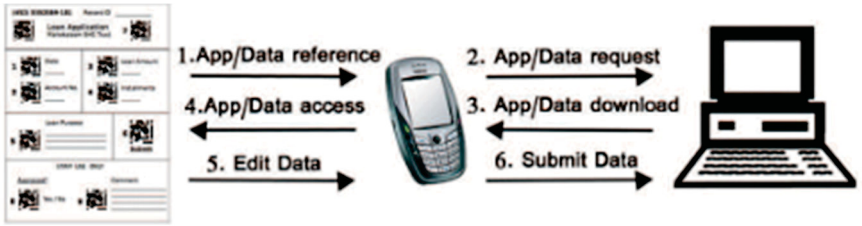
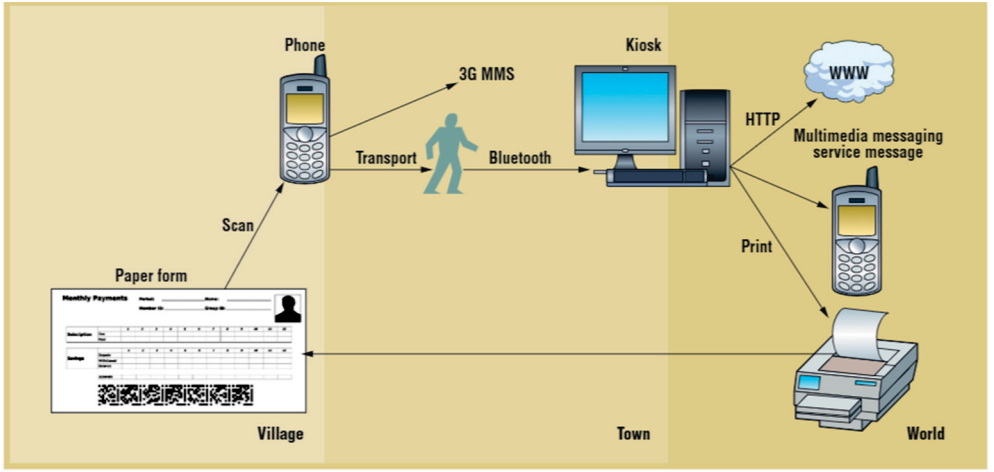


Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas.

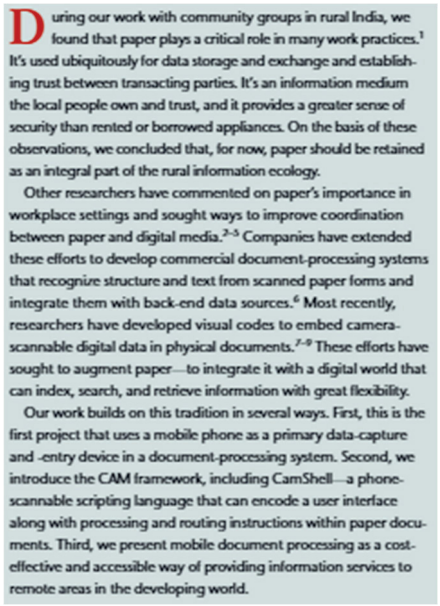
CAM (Pervasive Computing) at 74

> Even as currently posed, rural and pervasive computing share many infrastructural, interface, and application-level themes. The basic problem is to introduce computing seamlessly into a physical, natural world populated by nonexpert users who can't, don't, or would rather not spend their time in front of computers. These users aren't knowledge workers, don't use modern information technology as part of

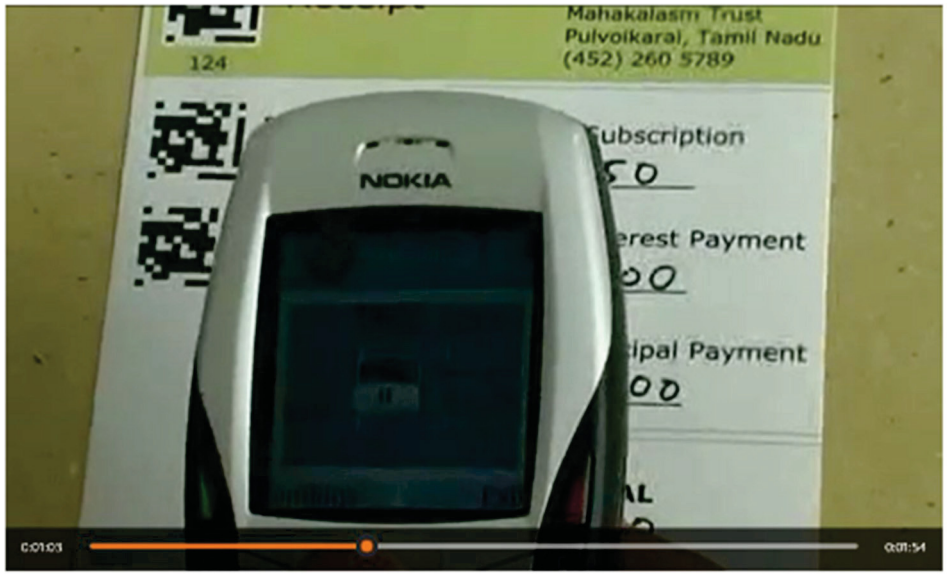
Claims	Prior Art
	their daily lives, and would like maximal utility from computing devices with minimal engagement. At a systems level, both rural and pervasive computing face environmental constraints in power, network connectivity, and cost-bearing capabilities. Moreover, applications in both domains must build value chains around technological innovations that can have a real, beneficial impact on people's lives. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing.

Claims	Prior Art
	CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Pervasive Computing) at 75 In designing CAM services, we found it convenient to refer to sets of augmented paper documents as CamForm applications. Each document can represent a distinct query or data entry action, one step in an action sequence (as in the case of a multipage form), or a previous action's result. Taken together, CamForms give a view of the application's current state and provide various interaction options with a distant server, not unlike a traditional client.

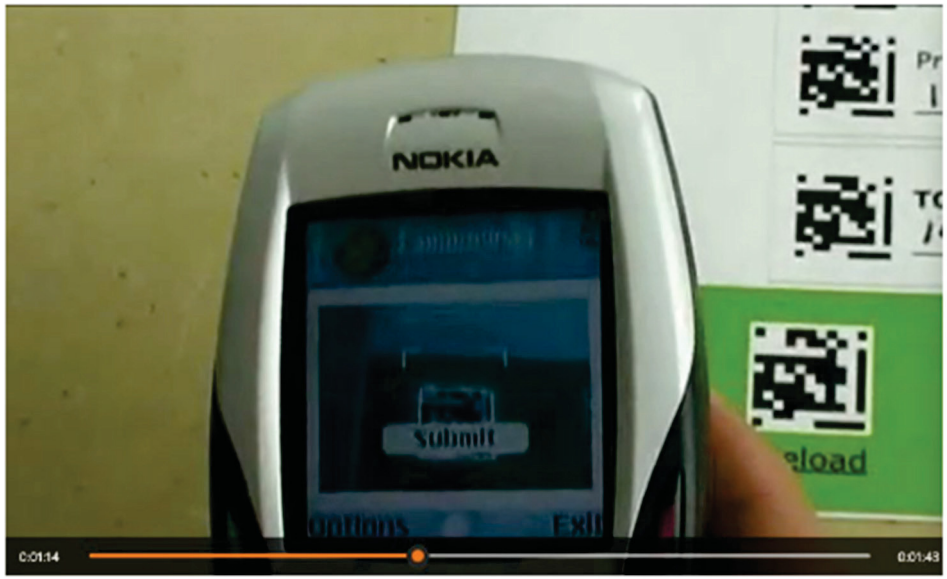
Claims	Prior Art
	CAM (Pervasive Computing) at 76  CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-

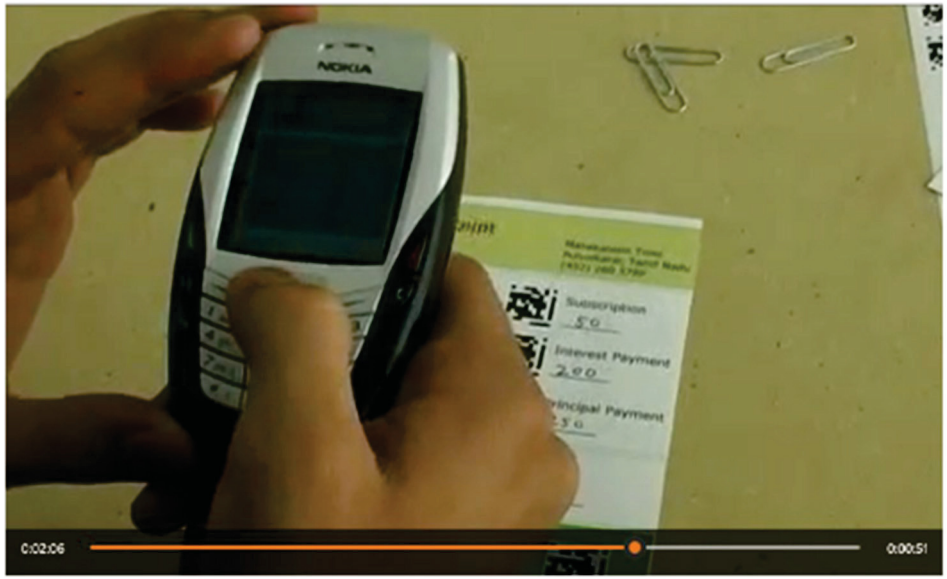
Claims	Prior Art
	carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example,

Claims	Prior Art
	when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus

Claims	Prior Art
	combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1a] a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the processor, the memory storing:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the processor, the memory storing." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious

Claims	Prior Art
	system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking

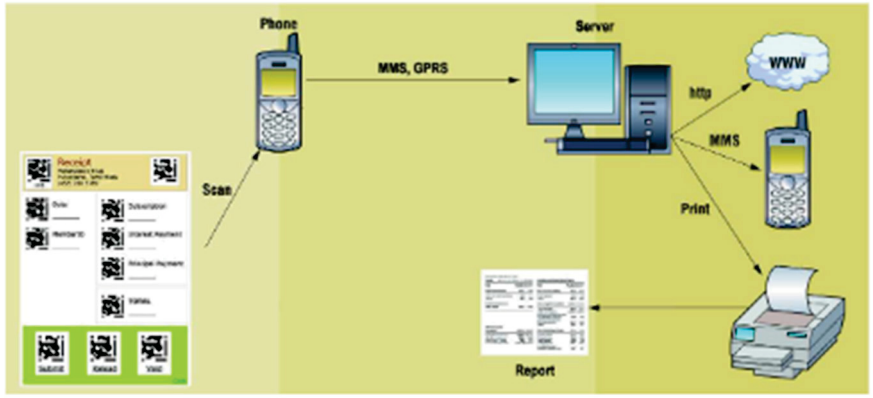
Claims	Prior Art
	- Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 In contrast to other tangible UI programming toolkits [2, 9], CAM is not an API for software developers to build different physical user interfaces and applications. CAM is more like the World Wide Web, in that it unites a user interface, a programming language and a delivery mechanism for accessing networked information services. Like the web, our goal is to allow many different services to be developed and deployed using a common infrastructure. CAM (Parikh 2005b) at 2 CamBrowser

Claims	Prior Art
	The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell is the programming language that is used to define interaction with visual codes embedded in the form. There is one visual code in every CamForm that serves as a form identifier. This code has a 0 as the lowest-order bit. The next 4 bits identify the form to the CamBrowser application. This information is used to load the form description schema, an XML file containing metadata about the form and the underlying data elements (including any default or pre-existing values). The remaining bits specify the protocol that should be used to load the schema file (currently HTTP, bluetooth or the local filesystem), and the required information to find the schema (a network address or a file name). CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number.

Claims	Prior Art
	• Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone’s user interface, networking and multimedia input

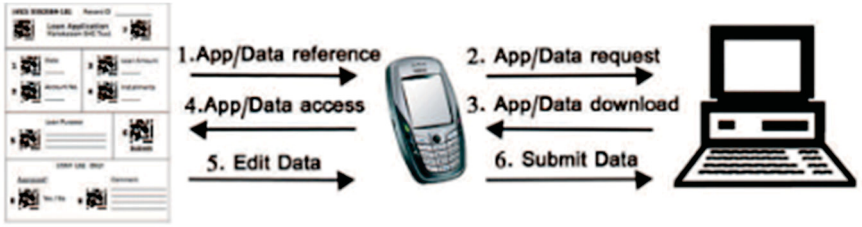
Claims	Prior Art
	and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 54–55 The same application can be accessed by entering a numeric string using the phone keypad. This is useful for mobile phones without a camera, or for applications that do not have an associated paper representation. The numeric string is based on a phone number, providing a familiar metaphor for rural users. An application is "dialed" by entering an arbitrary length phone number (or 12-digit IP address) for the server, a '*' delimiter (two '*'s for an IP address), and then a server-specific application ID. If the address or phone [Table 4.1] number is omitted, it is treated as a local application that may have been downloaded to the phone previously (the equivalent of accessing localhost in a web browser). CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-

Claims	Prior Art
	based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed

Claims	Prior Art
	barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

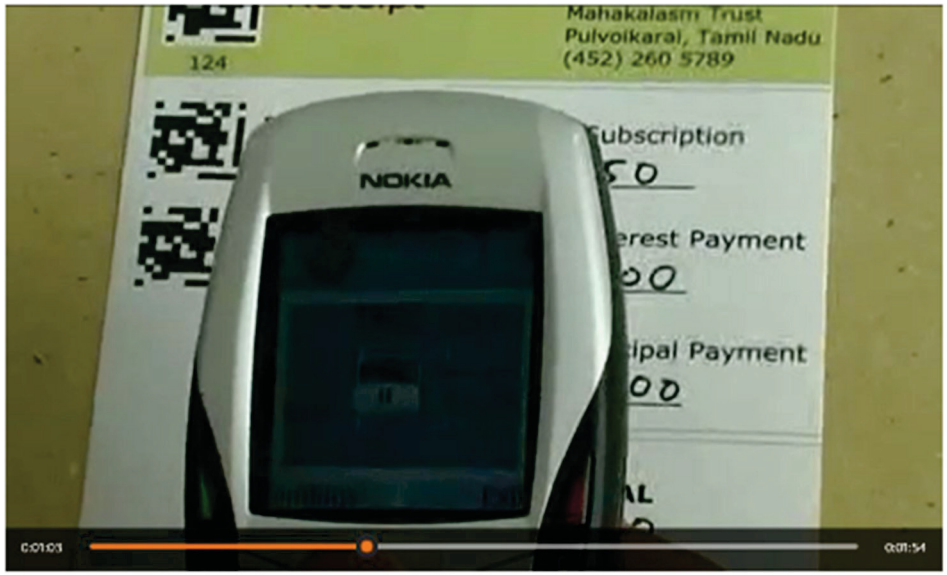
Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  **Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.** CAM (Parikh 2007) at 55 (Table 4.1) **Table 4.1: Barcode and numeric inputs for accessing CAM applications and functions.** *Application* Barcode: 00-Protocol(3 bits)-Addr(38)-AppID(24) Numeric: [Addr-*]-AppID-[FuncID[#Param]] *Function* Barcode: 11-FuncID(8 bits)-Param(66) Numeric: FuncID[#Param]

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

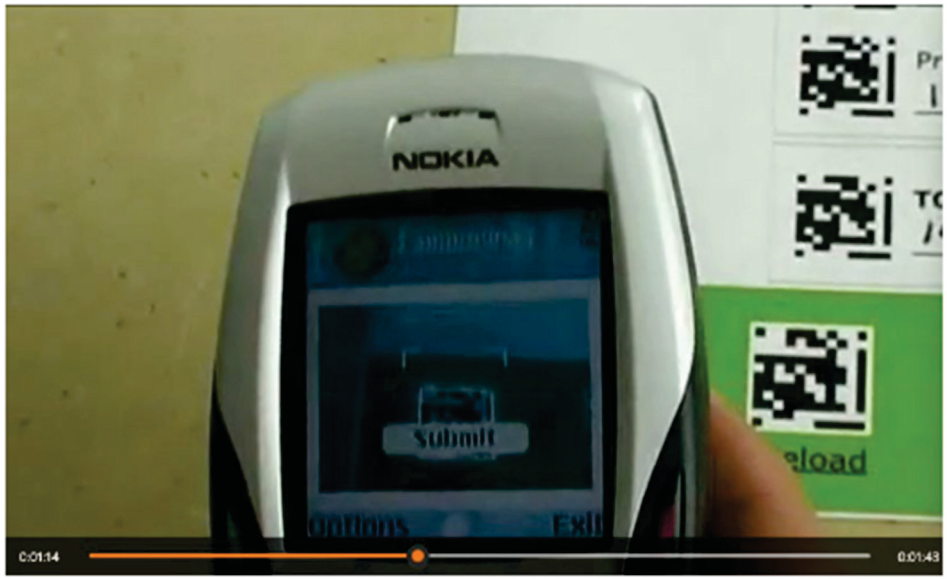
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74 Even as currently posed, rural and pervasive computing share many infrastructural, interface, and application-level themes. The basic problem is to introduce computing seamlessly into a physical, natural world populated by nonexpert users who can't, don't, or would rather not spend their time in front of computers. These users aren't knowledge workers, don't use modern information technology as part of their daily lives, and would like maximal utility from computing devices with minimal engagement. At a systems level, both rural and pervasive computing face environmental constraints in power, network connectivity, and cost-bearing capabilities. Moreover, applications in both domains must build value chains around technological innovations that can have a real, beneficial impact on people's lives. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions

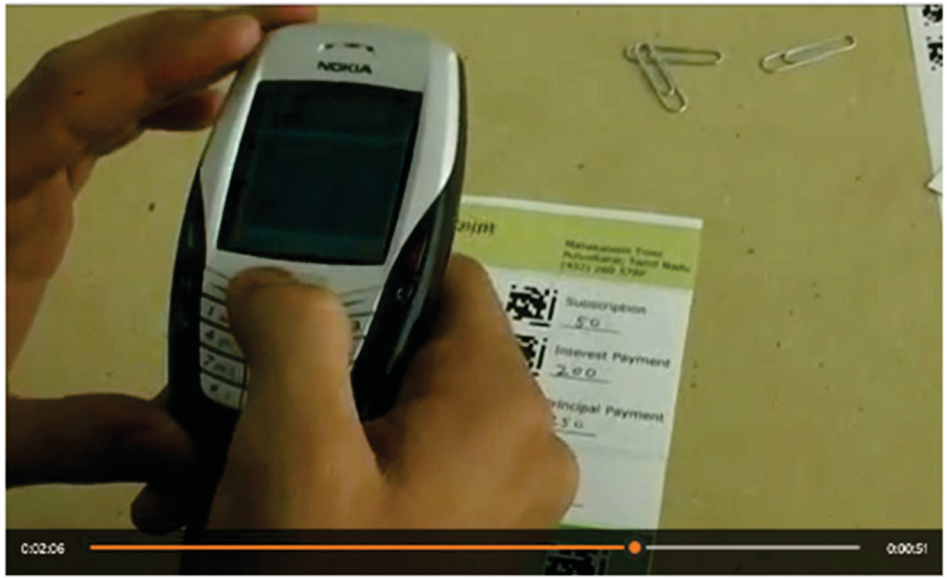
Claims	Prior Art
	in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75 In designing CAM services, we found it convenient to refer to sets of augmented paper documents as CamForm applications. Each document can represent a distinct query or data entry action, one step in an action sequence (as in the case of a multipage form), or a previous action's result. Taken together, CamForms give a view of the application's current state and provide various interaction options with a distant server, not unlike a traditional client. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a

Claims	Prior Art
	simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1b] a software abstraction layer for controlling the camera, the software abstraction layer comprising Instructions that, when executed by the processor, control the camera; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a software abstraction layer for controlling the camera, the software abstraction layer comprising Instructions that, when executed by the processor, control the camera." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the

Claims	Prior Art
	CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 In contrast to other tangible UI programming toolkits [2, 9], CAM is not an API for software developers to build different physical user interfaces and applications. CAM is more like the World Wide Web, in that it unites a user interface, a programming language and a delivery mechanism for accessing networked information services. Like the web, our goal is to allow many different services to be developed and deployed using a common infrastructure.

Claims	Prior Art
	CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell is the programming language that is used to define interaction with visual codes embedded in the form. There is one visual code in every CamForm that serves as a form identifier. This code has a 0 as the lowest-order bit. The next 4 bits identify the form to the CamBrowser application. This information is used to load the form description schema, an XML file containing metadata about the form and the underlying data elements (including any default or pre-existing values). The remaining bits specify the protocol that should be used to load the schema file (currently HTTP, bluetooth or the local filesystem), and the required information to find the schema (a network address or a file name). CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached.

Claims	Prior Art
	• Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145].

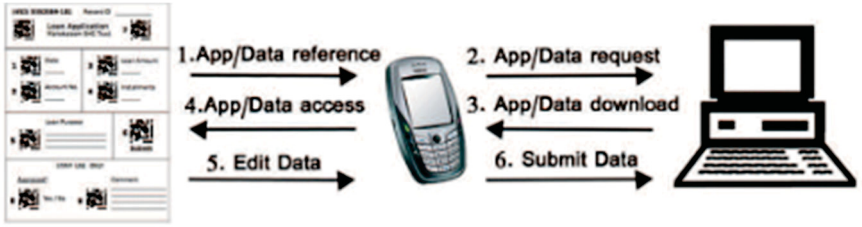
Claims	Prior Art
	CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 54–55 The same application can be accessed by entering a numeric string using the phone keypad. This is useful for mobile phones without a camera, or for applications that do not have an associated paper representation. The numeric string is based on a phone number, providing a familiar metaphor for rural users. An application is "dialed" by entering an arbitrary length phone number (or 12-digit IP address) for the server, a '*' delimiter (two '*'s for an IP address), and then a server-specific application ID. If the address or phone [Table 4.1] number is omitted, it is treated as a local application that may have been downloaded to the phone previously (the equivalent of accessing localhost in a web browser). CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the

Claims	Prior Art
	SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data

Claims	Prior Art
	collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

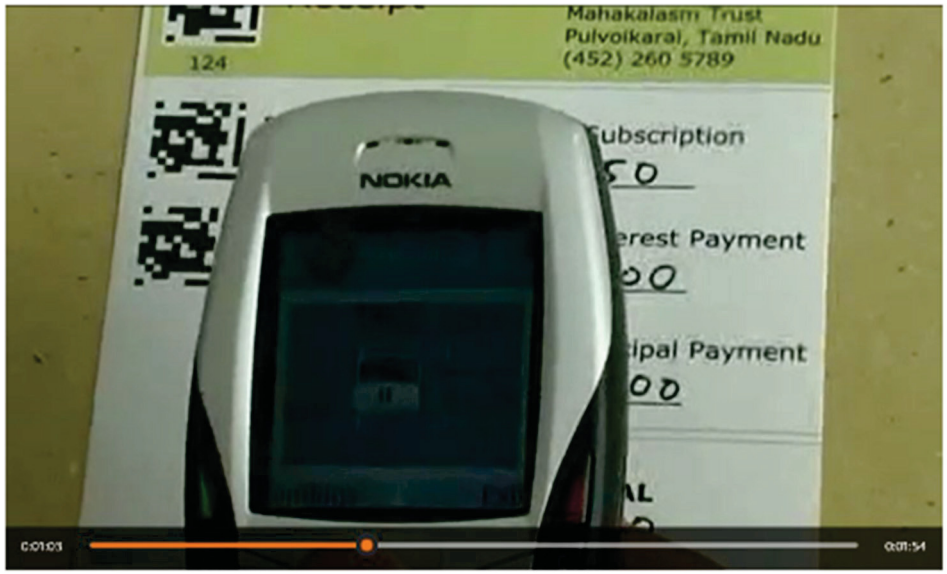
Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 55 (Table 4.1) Table 4.1: Barcode and numeric inputs for accessing CAM applications and functions. *Application* Barcode: 00-Protocol(3 bits)-Addr(38)-AppID(24) Numeric: [Addr-*]-AppID-[FuncID[#Param]] *Function* Barcode: 11-FuncID(8 bits)-Param(66) Numeric: FuncID[#Param]

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

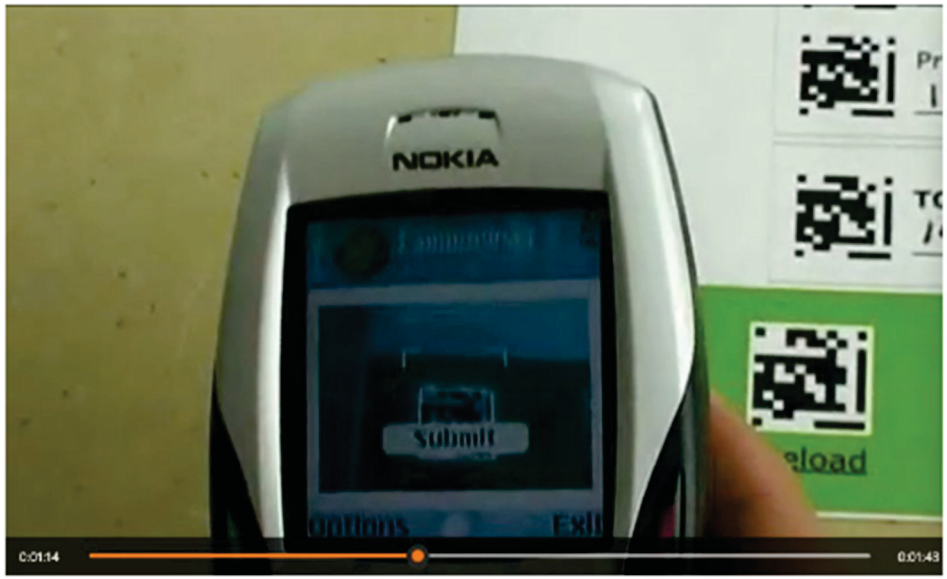
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74 Even as currently posed, rural and pervasive computing share many infrastructural, interface, and application-level themes. The basic problem is to introduce computing seamlessly into a physical, natural world populated by nonexpert users who can't, don't, or would rather not spend their time in front of computers. These users aren't knowledge workers, don't use modern information technology as part of their daily lives, and would like maximal utility from computing devices with minimal engagement. At a systems level, both rural and pervasive computing face environmental constraints in power, network connectivity, and cost-bearing capabilities. Moreover, applications in both domains must build value chains around technological innovations that can have a real, beneficial impact on people's lives. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions

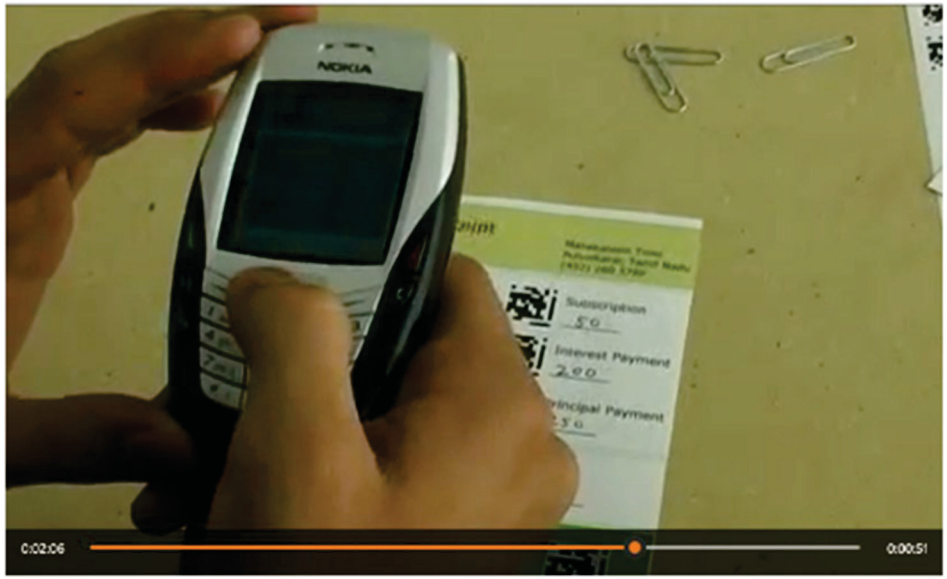
Claims	Prior Art
	in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75 In designing CAM services, we found it convenient to refer to sets of augmented paper documents as CamForm applications. Each document can represent a distinct query or data entry action, one step in an action sequence (as in the case of a multipage form), or a previous action's result. Taken together, CamForms give a view of the application's current state and provide various interaction options with a distant server, not unlike a traditional client. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user.

Claims	Prior Art
	• control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity.

Claims	Prior Art
	Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre], 1[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1c] a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera through the software abstraction layer, the downloaded app comprising Instructions that, when executed by the processor, cause the user's mobile device to perform operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera through the software abstraction layer, the downloaded app comprising Instructions that, when executed by the processor, cause the user's mobile device to perform operations including." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that

Claims	Prior Art
	contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met

Claims	Prior Art
	banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 In contrast to other tangible UI programming toolkits [2, 9], CAM is not an API for software developers to build different physical user interfaces and applications. CAM is more like the World Wide Web, in that it unites a user interface, a programming language and a delivery mechanism for accessing networked information services. Like the web, our goal is to allow many different services to be developed and

Claims	Prior Art
	deployed using a common infrastructure. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell is the programming language that is used to define interaction with visual codes embedded in the form. There is one visual code in every CamForm that serves as a form identifier. This code has a 0 as the lowest-order bit. The next 4 bits identify the form to the CamBrowser application. This information is used to load the form description schema, an XML file containing metadata about the form and the underlying data elements (including any default or pre-existing values). The remaining bits specify the protocol that should be used to load the schema file (currently HTTP, bluetooth or the local filesystem), and the required information to find the schema (a network address or a file name). CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity.

Claims	Prior Art
	• SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB.

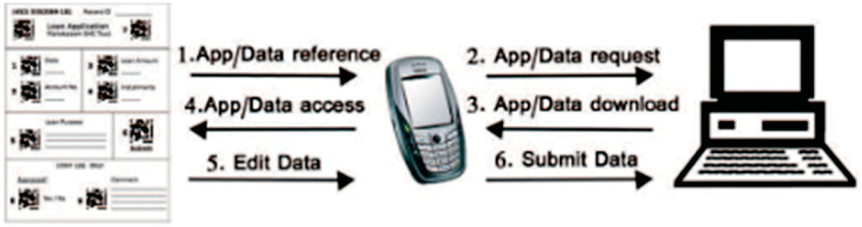
Claims	Prior Art
	CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 54–55 The same application can be accessed by entering a numeric string using the phone keypad. This is useful for mobile phones without a camera, or for applications that do not have an associated paper representation. The numeric string is based on a phone number, providing a familiar metaphor for rural users. An application is "dialed" by entering an arbitrary length phone number (or 12-digit IP address) for the server, a '*' delimiter (two '*'s for an IP address), and then a server-specific application ID. If the address or phone [Table 4.1] number is omitted, it is treated as a local application that may have been downloaded to the phone previously (the equivalent of accessing localhost in a web browser). CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2.

Claims	Prior Art
	CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the

Claims	Prior Art
	mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

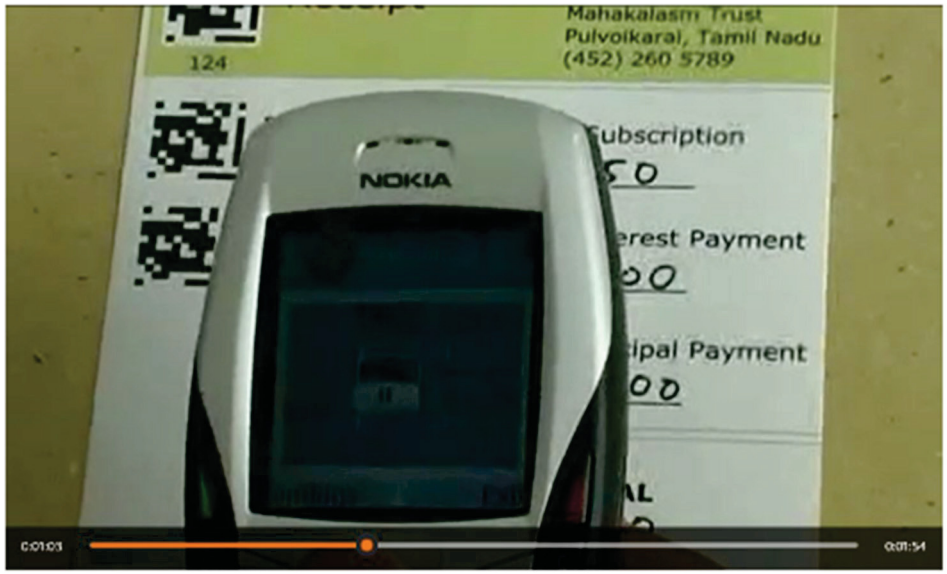
Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 55 (Table 4.1) Table 4.1: Barcode and numeric inputs for accessing CAM applications and functions. *Application* Barcode: 00-Protocol(3 bits)-Addr(38)-AppID(24) Numeric: [Addr-*]-AppID-[FuncID[#Param]] *Function* Barcode: 11-FuncID(8 bits)-Param(66) Numeric: FuncID[#Param]

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

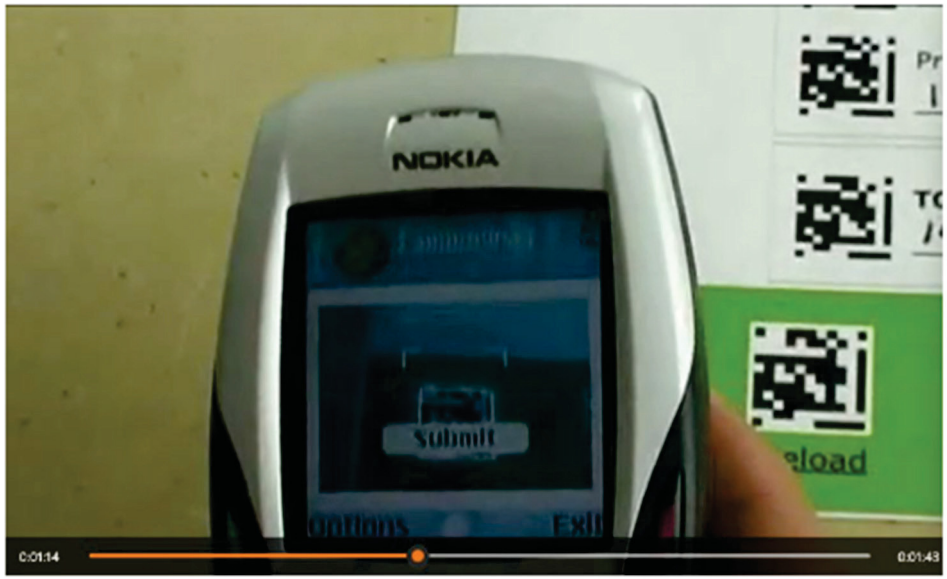
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74 Even as currently posed, rural and pervasive computing share many infrastructural, interface, and application-level themes. The basic problem is to introduce computing seamlessly into a physical, natural world populated by nonexpert users who can't, don't, or would rather not spend their time in front of computers. These users aren't knowledge workers, don't use modern information technology as part of their daily lives, and would like maximal utility from computing devices with minimal engagement. At a systems level, both rural and pervasive computing face environmental constraints in power, network connectivity, and cost-bearing capabilities. Moreover, applications in both domains must build value chains around technological innovations that can have a real, beneficial impact on people's lives. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions

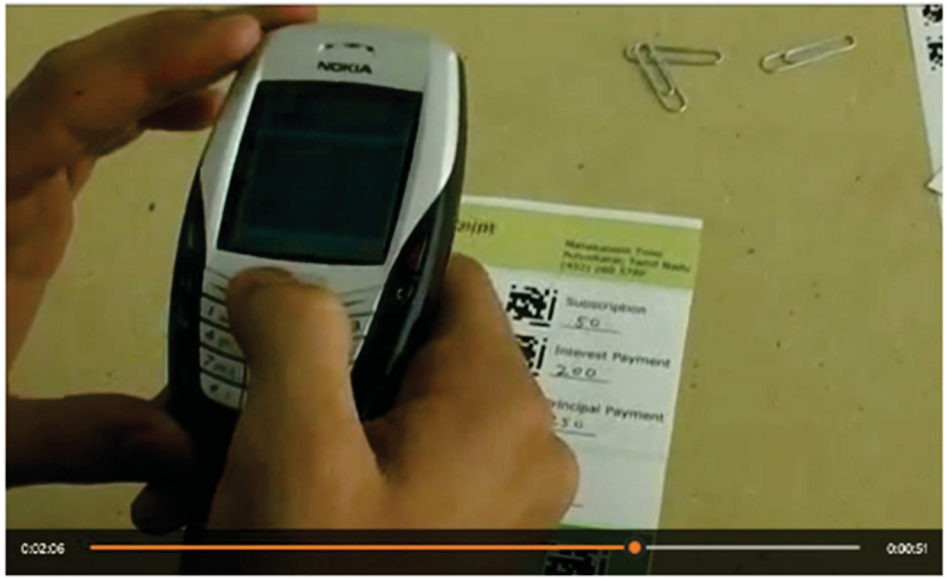
Claims	Prior Art
	in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75 In designing CAM services, we found it convenient to refer to sets of augmented paper documents as CamForm applications. Each document can represent a distinct query or data entry action, one step in an action sequence (as in the case of a multipage form), or a previous action's result. Taken together, CamForms give a view of the application's current state and provide various interaction options with a distant server, not unlike a traditional client. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user.

Claims	Prior Art
	• control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity.

Claims	Prior Art
	Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

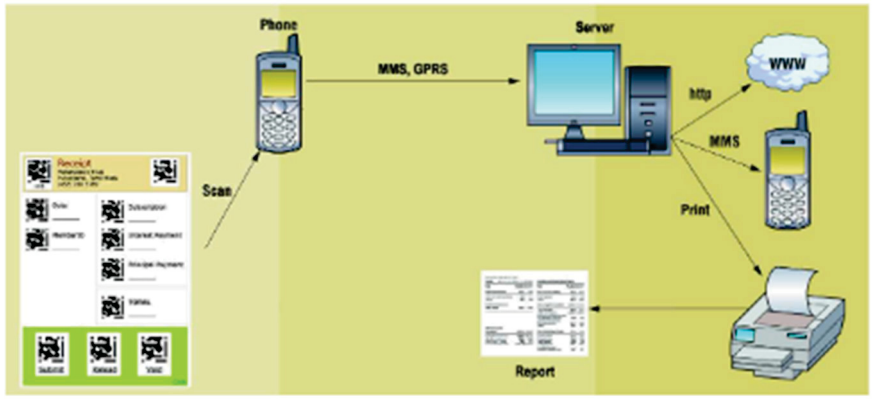
Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1d] providing Instructions to the user for capturing an image of a check;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "providing Instructions to the user for capturing an image of a check." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous

Claims	Prior Art
	connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards

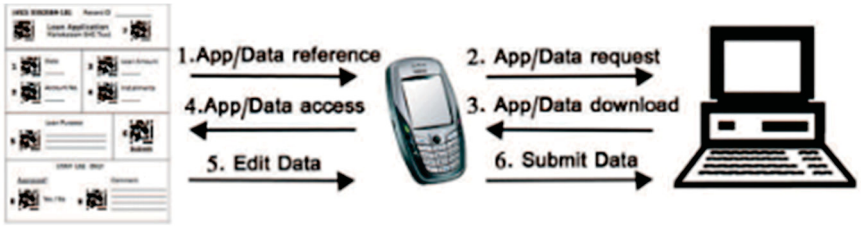
Claims	Prior Art
	CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user

Claims	Prior Art
	interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 46 Guided Interaction

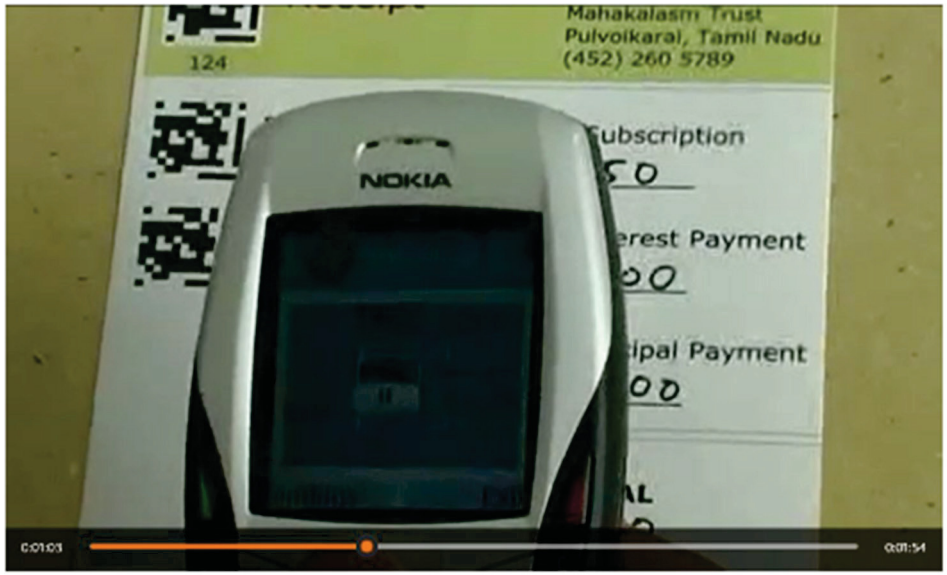
Claims	Prior Art
	With the users' limited means of identifying distinct screen elements, we found that minimizing the number of tasks possible at one time made it easier for users to comprehend a screen's overall purpose and structure. We therefore followed a design pattern of guided interaction, where users were guided from task to task through a sequence of discrete but consistent screens, each containing a number of well-ordered sub-tasks. For example, during a SHG Meeting, a number of different types of transactions (savings deposits, loan repayments, loan disbursements) are conducted for each member. We found that limiting each screen to one type of transaction proved more successful than a screen from which it was possible to record several different transactions for each member at one time. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 56–57 The prompts are displayed in sequence rather than laid out spatially like in a web-based entry form. Time-multiplexing is better suited to the small screen of the mobile device, where it is difficult to display or enter several pieces of information at one time. I call this wizard interaction, as it resembles the task wizards used for installing or configuring software applications. Users are guided step-wise through the process, reducing navigation requirements. This method avoids additional cognitive decision points, where the user has to [Figure 4.4] pause after each action to decide what to do next (and how to do it). This pause can create unnecessary confusion for novice users. Instead, in CAM, users just choose a high-level task and then follow the prompts. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is

Claims	Prior Art
	incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

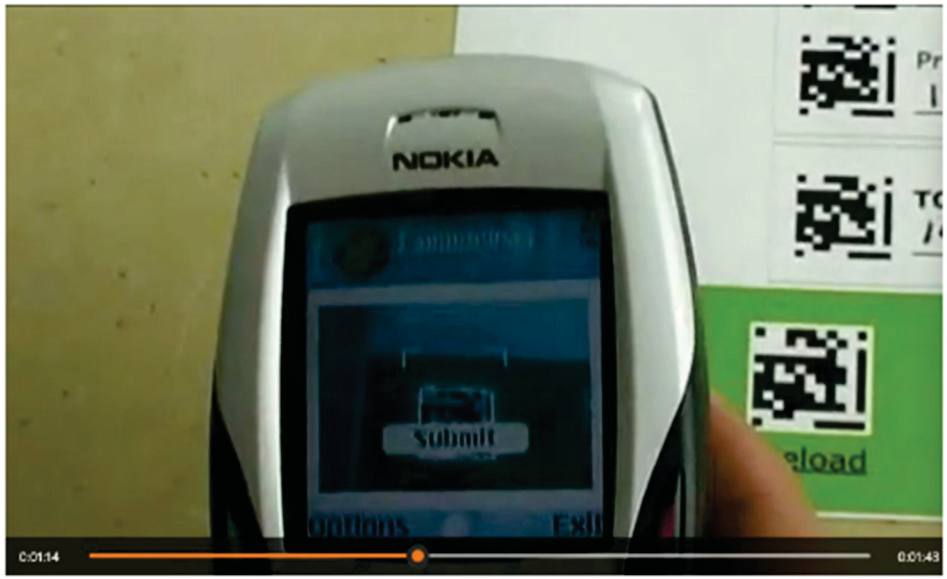
Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

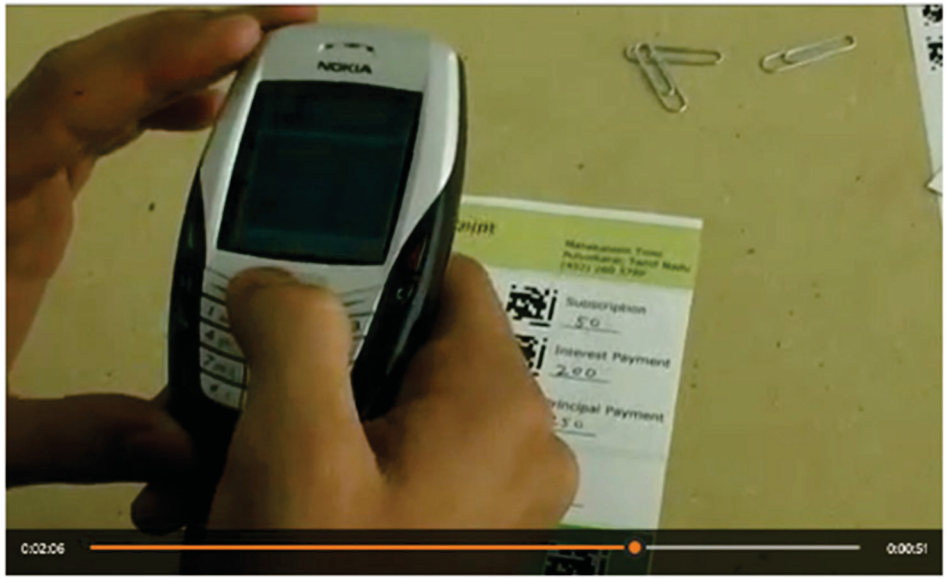
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on),

Claims	Prior Art
	letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request.

Claims	Prior Art
	SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1e] generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera through the software abstraction layer;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera through the software abstraction layer." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect

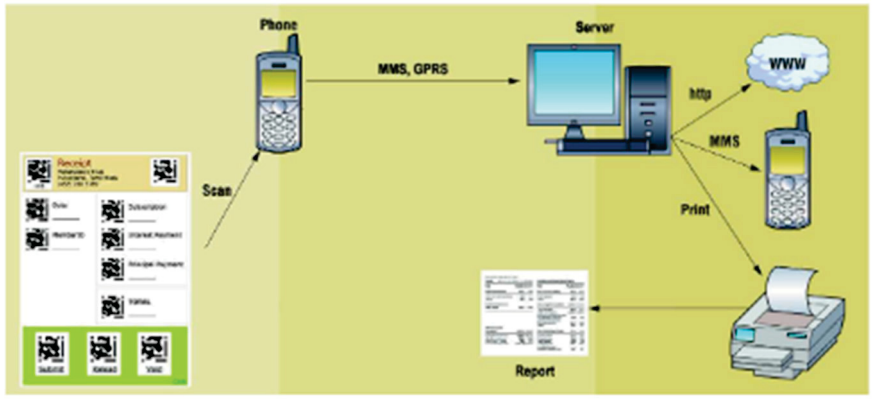
Claims	Prior Art
	bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 In contrast to other tangible UI programming toolkits [2, 9], CAM is not an API for software developers to build different physical user interfaces and applications. CAM is more like the World Wide Web, in that it unites a user interface, a programming language and a delivery mechanism for accessing networked information services. Like the web, our goal is to allow many different services to be developed and deployed using a common infrastructure. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone’s camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in

Claims	Prior Art
	real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell is the programming language that is used to define interaction with visual codes embedded in the form. There is one visual code in every CamForm that serves as a form identifier. This code has a 0 as the lowest-order bit. The next 4 bits identify the form to the CamBrowser application. This information is used to load the form description schema, an XML file containing metadata about the form and the underlying data elements (including any default or pre-existing values). The remaining bits specify the protocol that should be used to load the schema file (currently HTTP, bluetooth or the local filesystem), and the required information to find the schema (a network address or a file name). CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM

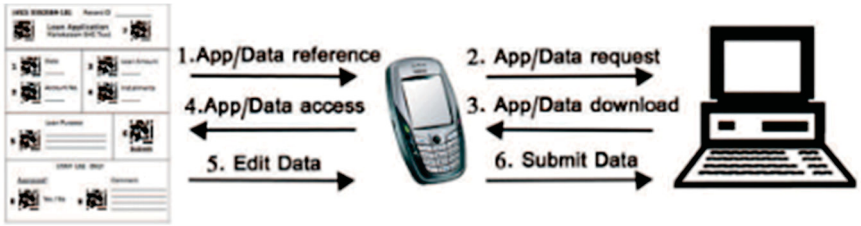
Claims	Prior Art
	— a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several

Claims	Prior Art
	barcodes can be detected from a single camera image. CAM (Parikh 2007) at 54–55 The same application can be accessed by entering a numeric string using the phone keypad. This is useful for mobile phones without a camera, or for applications that do not have an associated paper representation. The numeric string is based on a phone number, providing a familiar metaphor for rural users. An application is "dialed" by entering an arbitrary length phone number (or 12-digit IP address) for the server, a '*' delimiter (two '*'s for an IP address), and then a server-specific application ID. If the address or phone [Table 4.1] number is omitted, it is treated as a local application that may have been downloaded to the phone previously (the equivalent of accessing localhost in a web browser). CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is

Claims	Prior Art
	sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users.

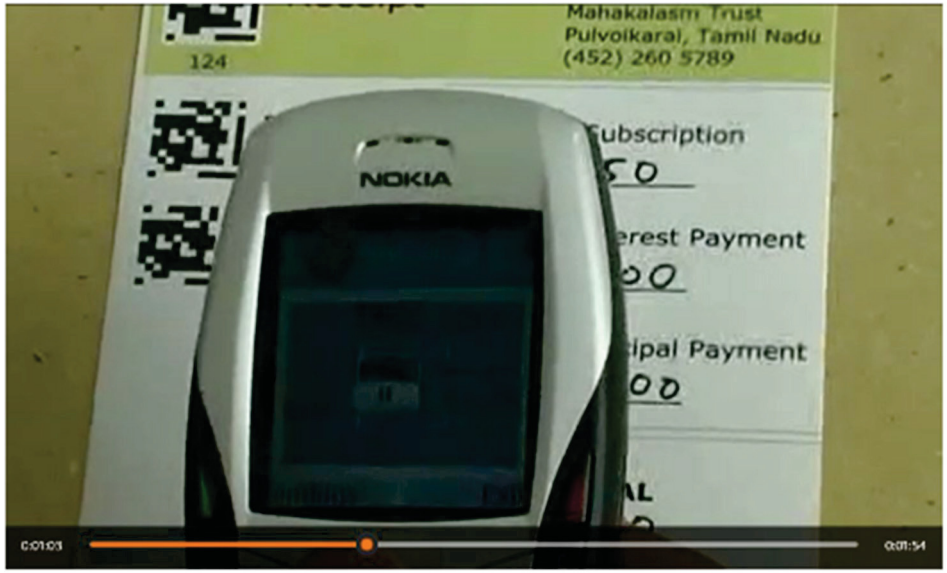
Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 55 (Table 4.1) Table 4.1: Barcode and numeric inputs for accessing CAM applications and functions. *Application* Barcode: 00-Protocol(3 bits)-Addr(38)-AppID(24) Numeric: [Addr-*]-AppID-[FuncID[#Param]] *Function* Barcode: 11-FuncID(8 bits)-Param(66) Numeric: FuncID[#Param] CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

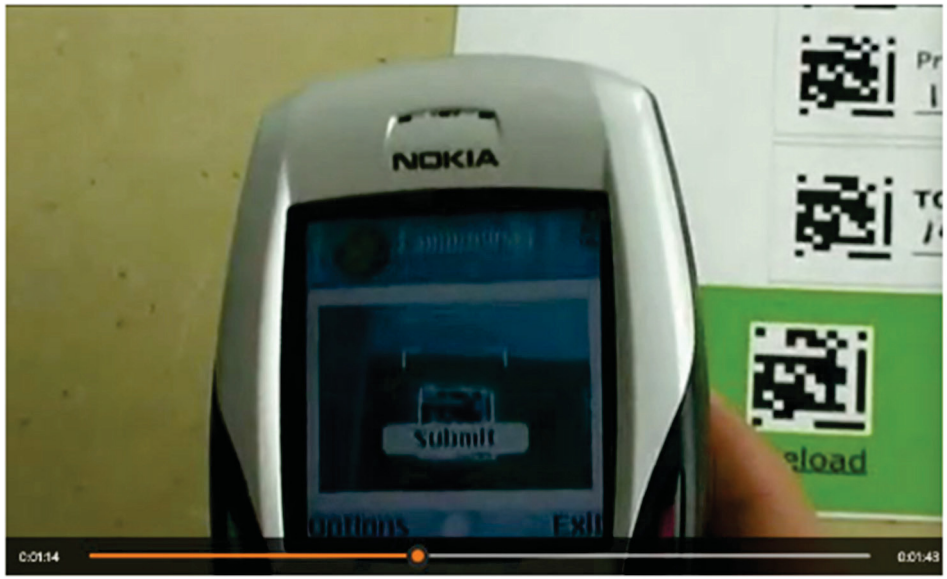
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int, input_date, input_password, input_pin, input_phone_number, input_time, input_number, input_text* — Prompts asking the user to enter different kinds of values *record_audio, take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note, conf_note* — Send a message to the user, or display a confirmation dialog *get, put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt, decrypt* — Encrypt or decrypt application data with a specified key *sms, mms, email* — Send arbitrary messages *phonecall, browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74 Even as currently posed, rural and pervasive computing share many infrastructural, interface, and application-level themes. The basic problem is to introduce computing seamlessly into a physical, natural world populated by nonexpert users who can't, don't, or would rather not spend their time in front of computers. These users aren't knowledge workers, don't use modern information technology as part of

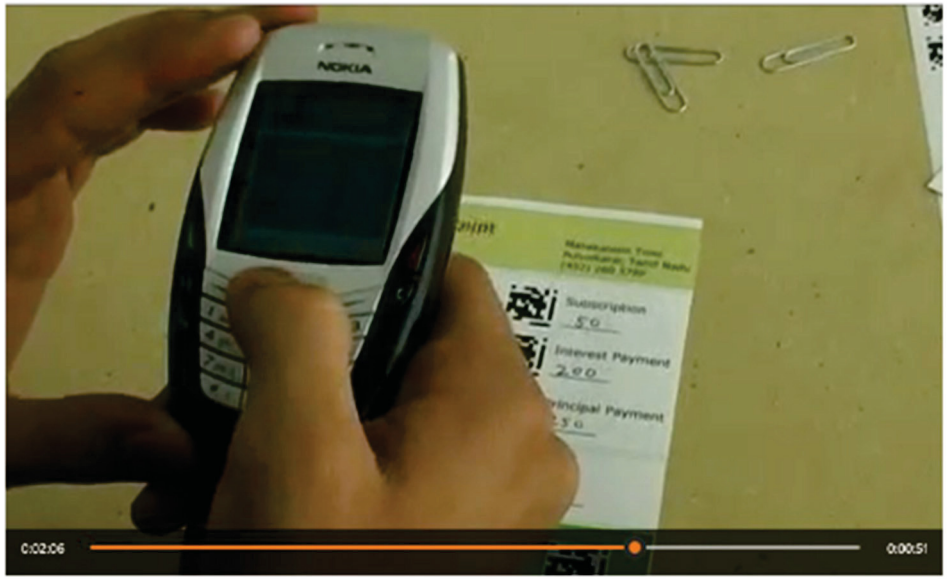
Claims	Prior Art
	their daily lives, and would like maximal utility from computing devices with minimal engagement. At a systems level, both rural and pervasive computing face environmental constraints in power, network connectivity, and cost-bearing capabilities. Moreover, applications in both domains must build value chains around technological innovations that can have a real, beneficial impact on people's lives. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75 In designing CAM services, we found it convenient to refer to sets of augmented paper documents as CamForm applications. Each document can represent a distinct query or data entry action, one step in an action sequence (as in the case of a multipage form), or a previous action's result. Taken together, CamForms give a view of the application's current state and provide various interaction options with a distant server, not unlike a traditional client. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat

Claims	Prior Art
	Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes

Claims	Prior Art
	either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[c]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1f] overlaying an alignment guide on the live video of the check in the field of view of the camera;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "overlaying an alignment guide on the live video of the check in the field of view of the camera." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous

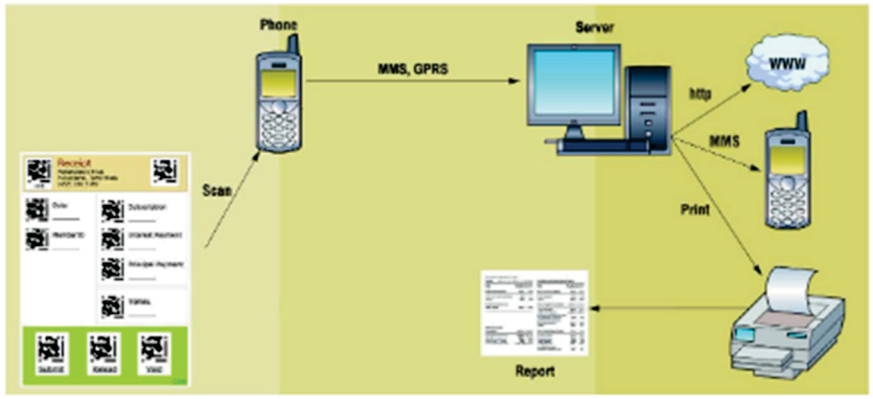
Claims	Prior Art
	connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone

Claims	Prior Art
	is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users.

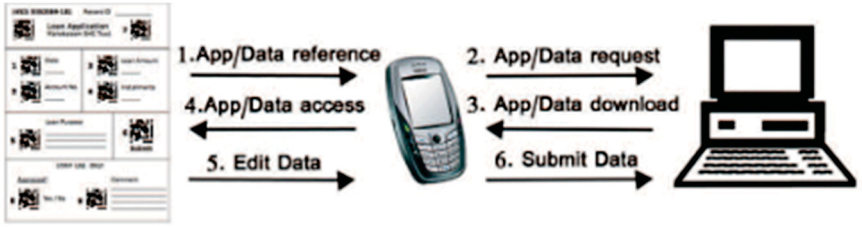
Claims	Prior Art
	• HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the

Claims	Prior Art
	phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value.

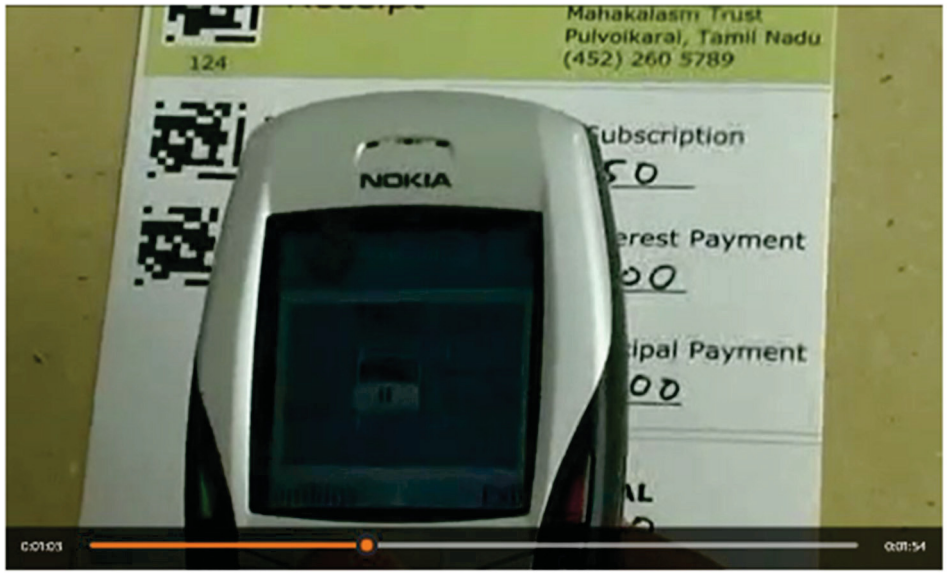
Claims	Prior Art
	CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users.

Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

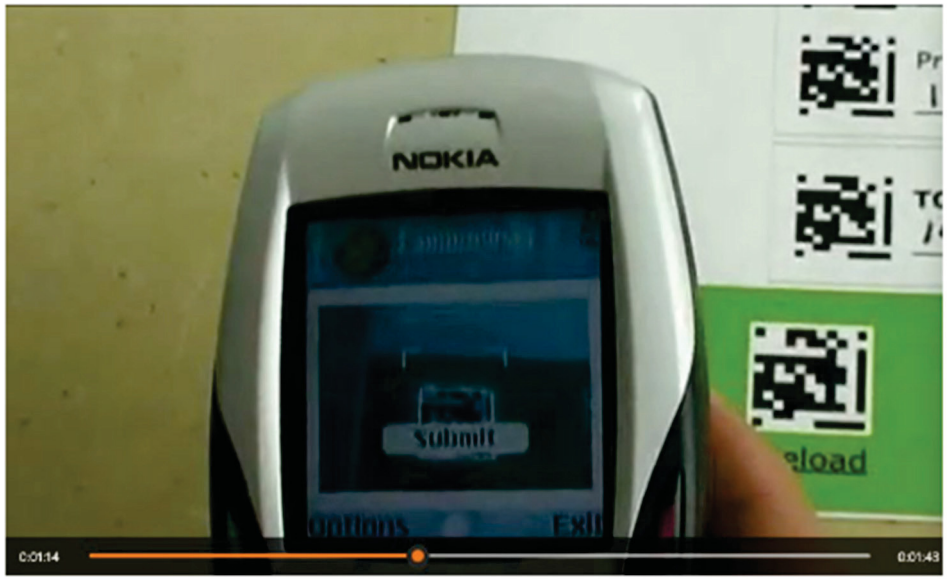
Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

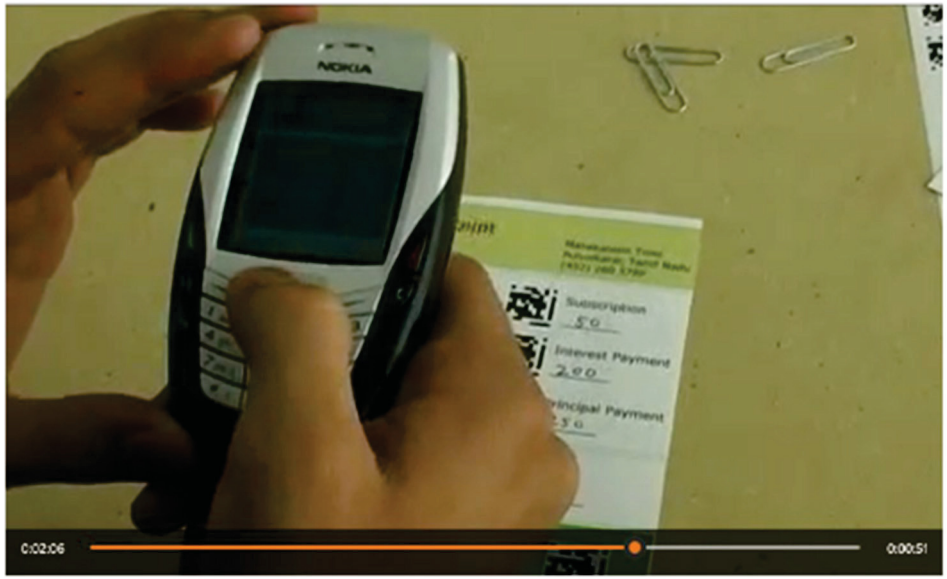
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of

Claims	Prior Art
	error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode).

Claims	Prior Art
	We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

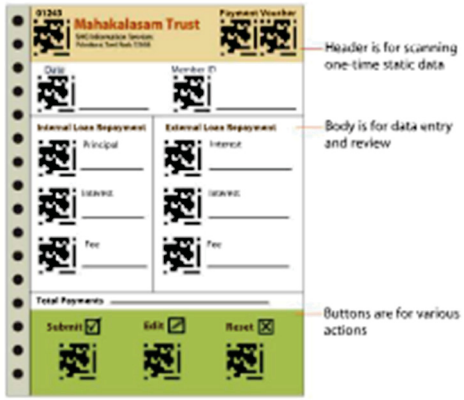
Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1g] monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect

Claims	Prior Art
	bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback

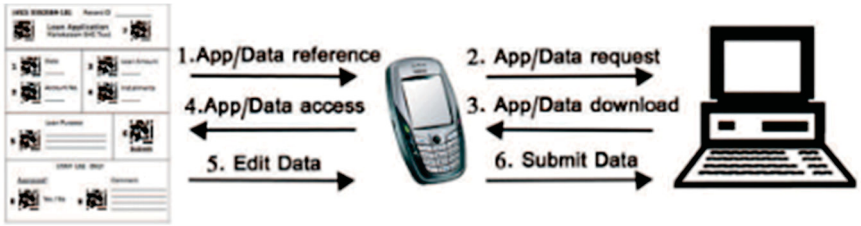
Claims	Prior Art
	has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context.

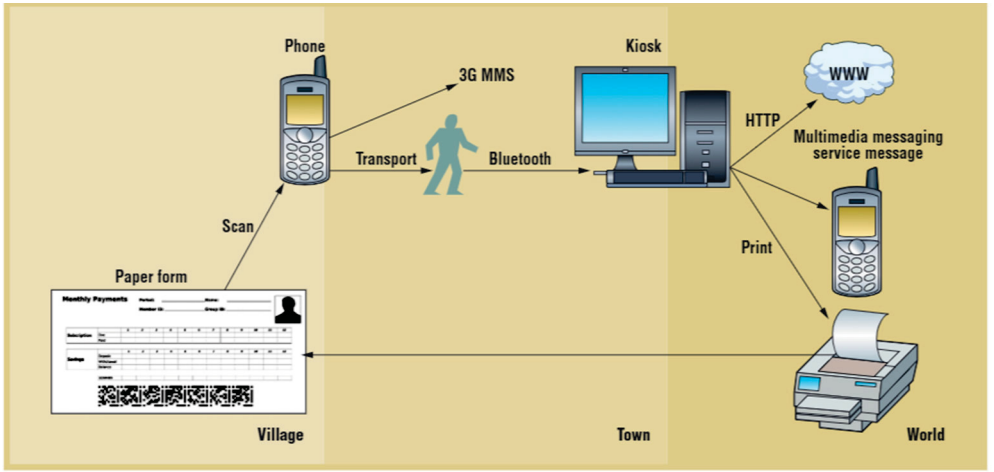
Claims	Prior Art
	Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image.

Claims	Prior Art
	CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities.

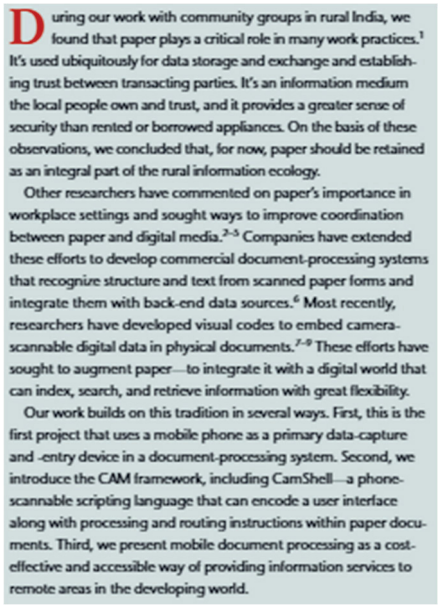
Claims	Prior Art
	CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

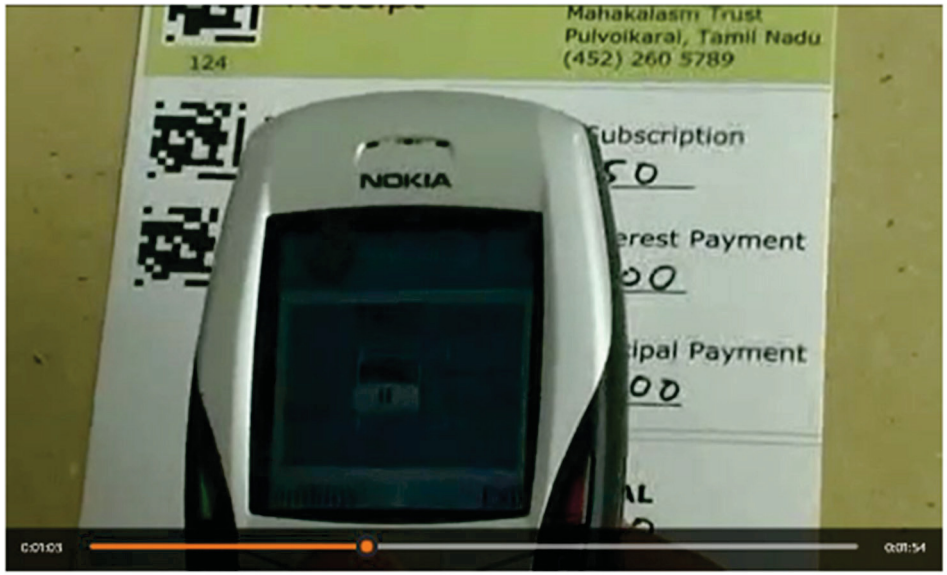
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural

Claims	Prior Art
	information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

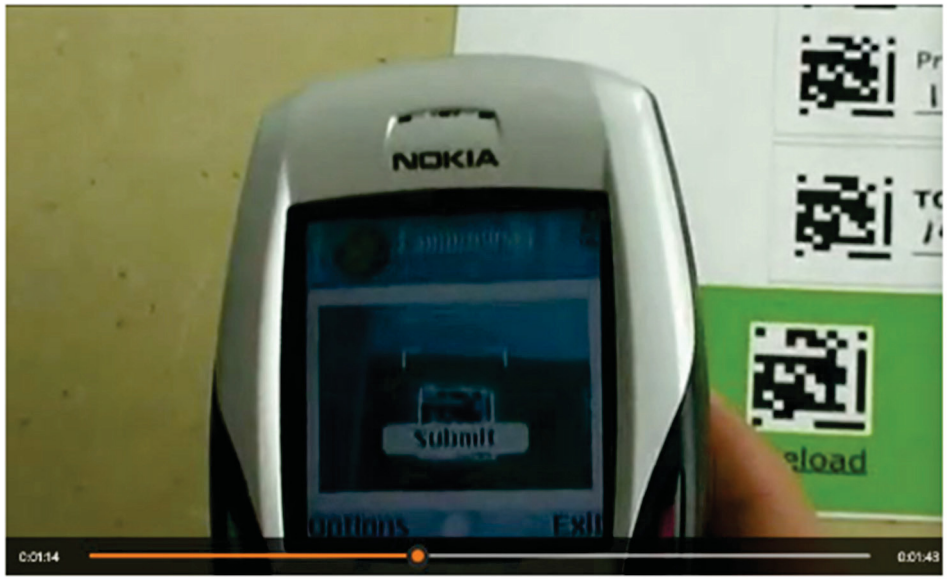
Claims	Prior Art
	CAM (Pervasive Computing) at 76  CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-

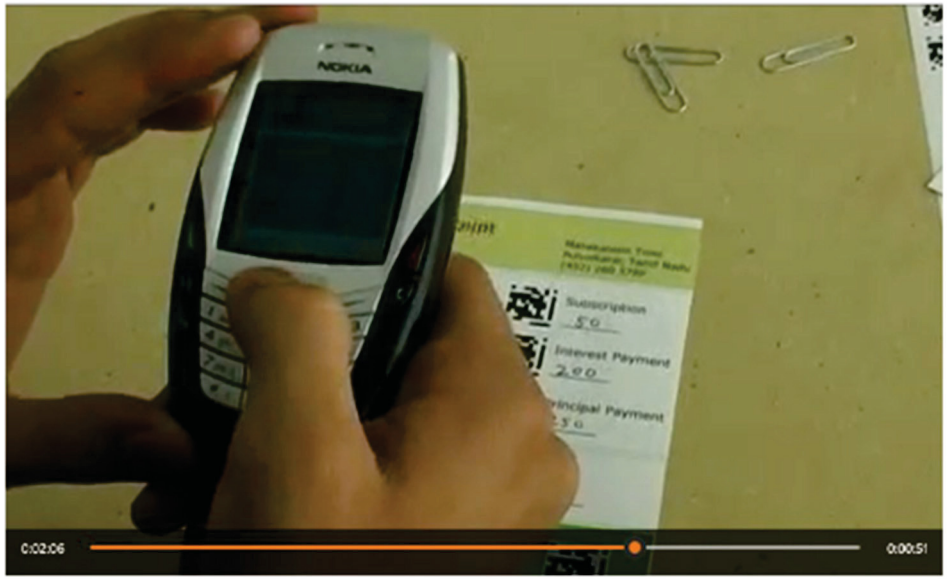
Claims	Prior Art
	carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example,

Claims	Prior Art
	when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

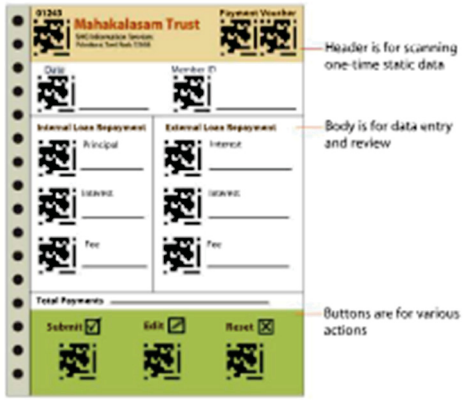
Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[b], 1[e]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1h] determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect

Claims	Prior Art
	bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback

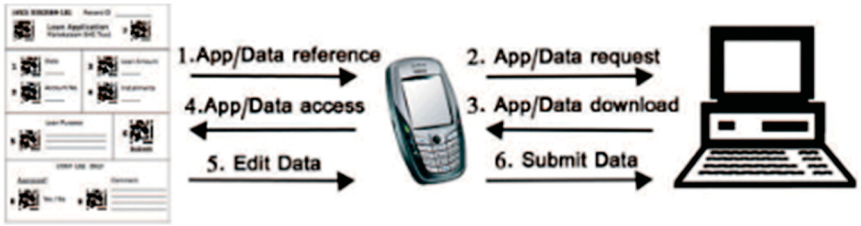
Claims	Prior Art
	has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context.

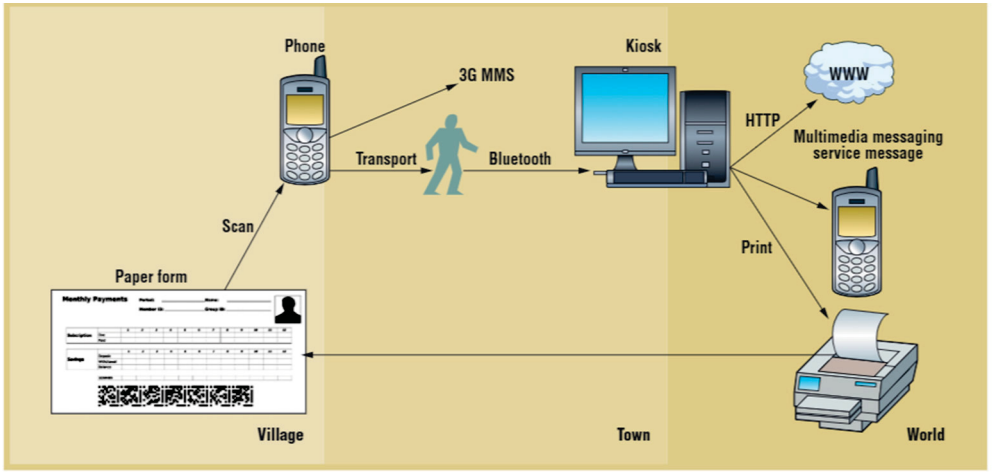
Claims	Prior Art
	Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image.

Claims	Prior Art
	CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities.

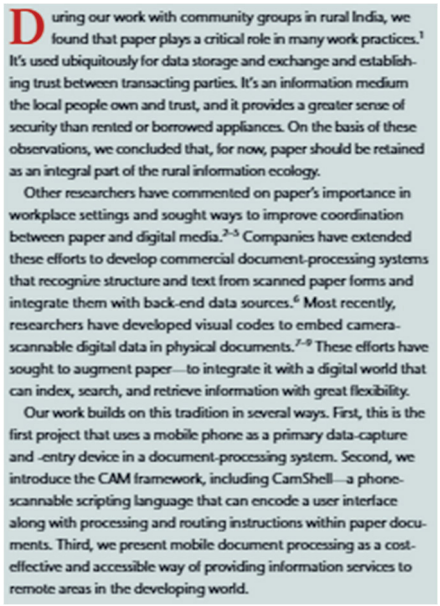
Claims	Prior Art
	CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

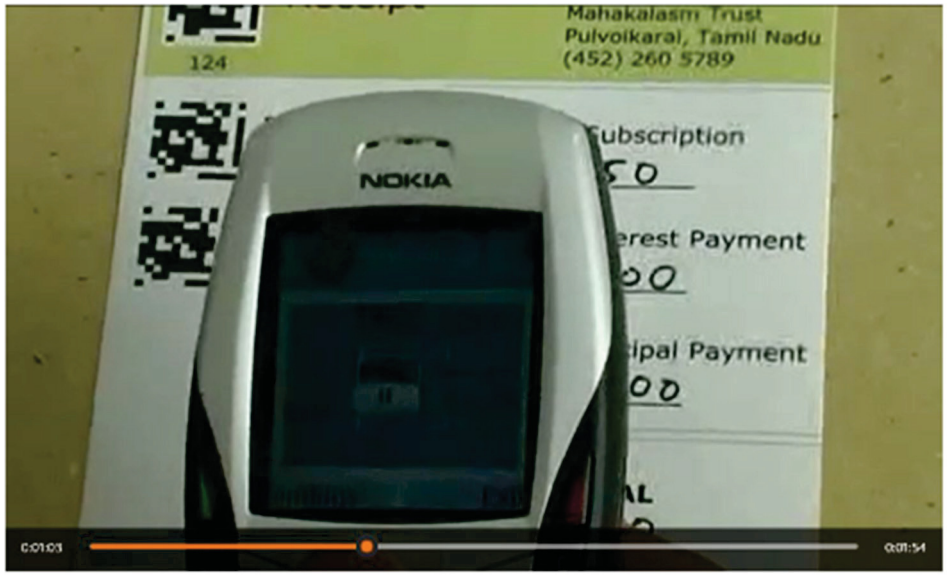
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural

Claims	Prior Art
	information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

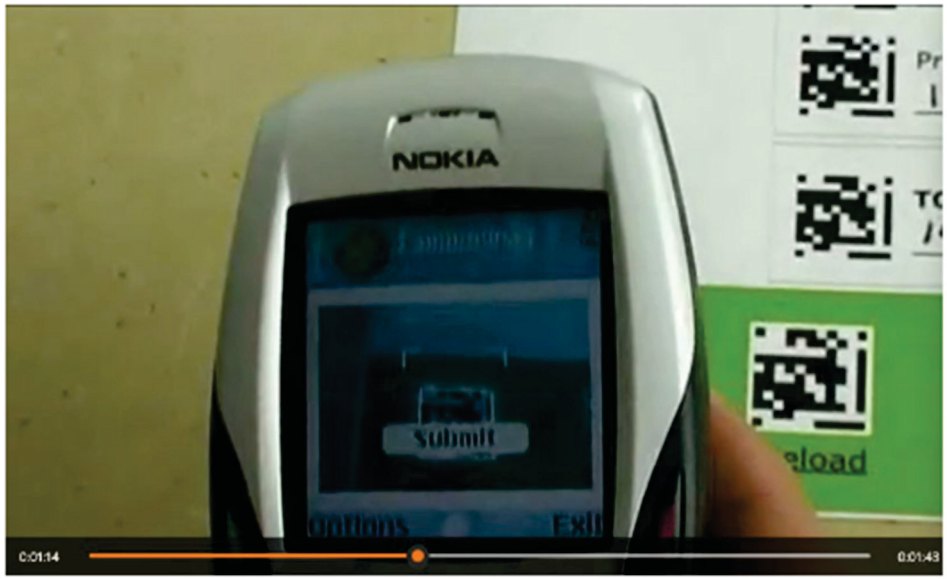
Claims	Prior Art
	CAM (Pervasive Computing) at 76  CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-

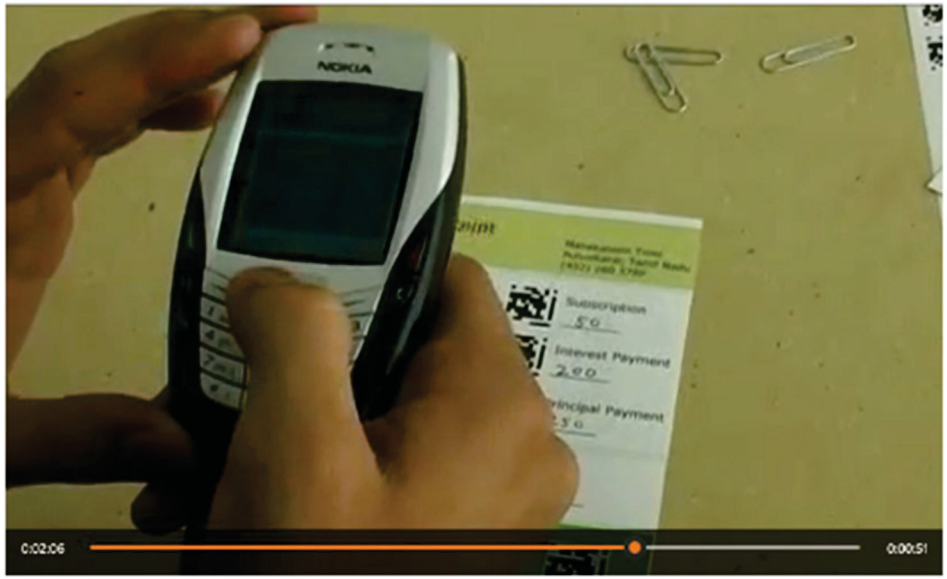
Claims	Prior Art
	carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example,

Claims	Prior Art
	when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

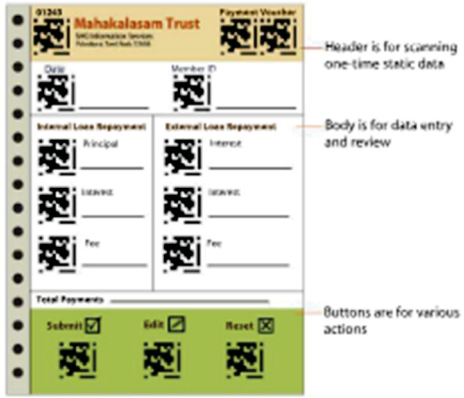
Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[b], 1[e], 1[g]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1i] when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least one monitoring criterion, providing feedback advising the user that the check has been captured;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least one monitoring criterion, providing feedback advising the user that the check has been captured." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that

Claims	Prior Art
	contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we

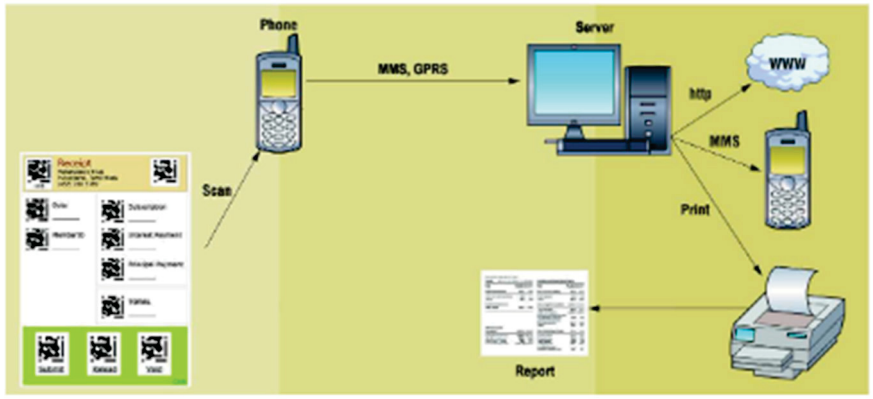
Claims	Prior Art
	need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey

Claims	Prior Art
	a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of

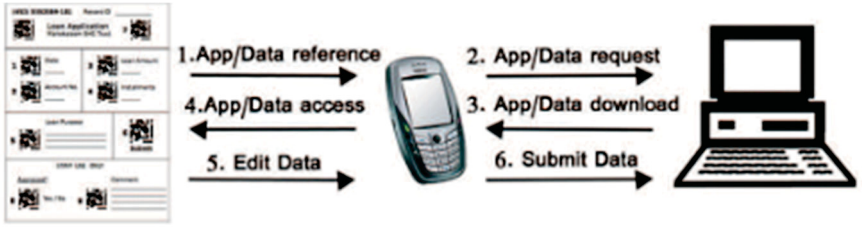
Claims	Prior Art
	education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 46 Guided Interaction With the users' limited means of identifying distinct screen elements, we found that minimizing the number of tasks possible at one time made it easier for users to comprehend a screen's overall purpose and structure. We therefore followed a design pattern of guided interaction, where users were guided from task to task through a sequence of discrete but consistent screens, each containing a number of well-ordered sub-tasks. For example, during a SHG Meeting, a number of different types of transactions (savings deposits, loan repayments, loan disbursements) are conducted for each member. We found that limiting each screen to one type of transaction proved more successful than a screen from which it was possible to record several different transactions for each member at one time. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated

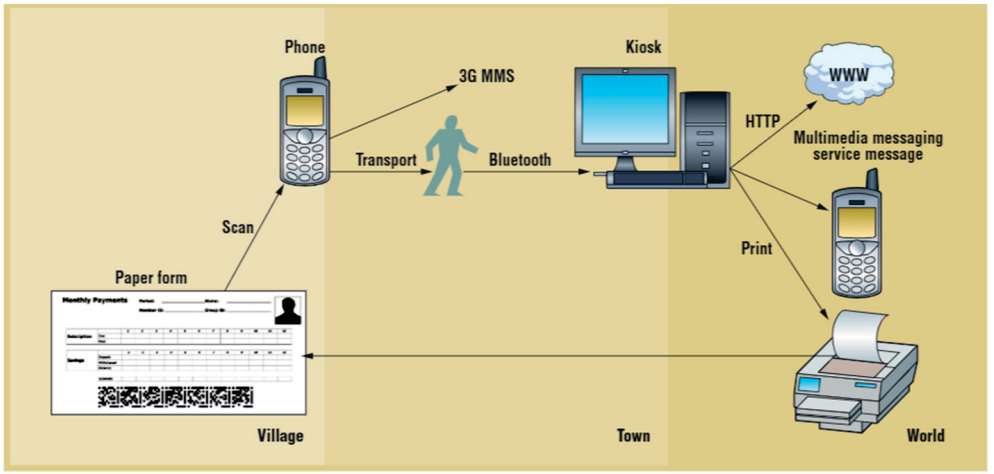
Claims	Prior Art
	files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 56–57 The prompts are displayed in sequence rather than laid out spatially like in a web-based entry form. Time-multiplexing is better suited to the small screen of the mobile device, where it is difficult to display or enter several pieces of information at one time. I call this wizard interaction, as it resembles the task wizards used for installing or configuring software applications. Users are guided step-wise through the process, reducing navigation requirements. This method avoids additional cognitive decision points, where the user has to [Figure 4.4] pause after each action to decide what to do next (and how to do it). This pause can create unnecessary confusion for novice users. Instead, in CAM, users just choose a high-level task and then follow the prompts. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is

Claims	Prior Art
	incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users.

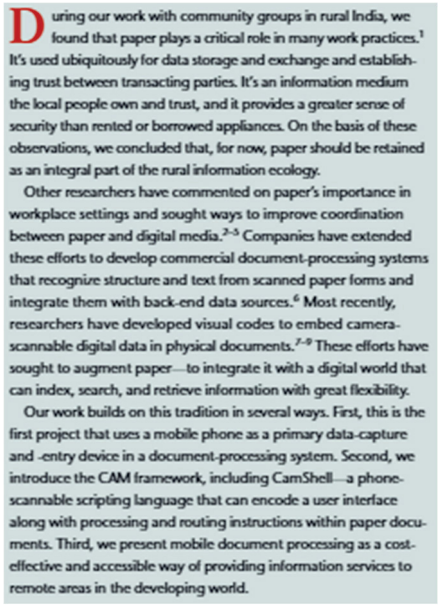
Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

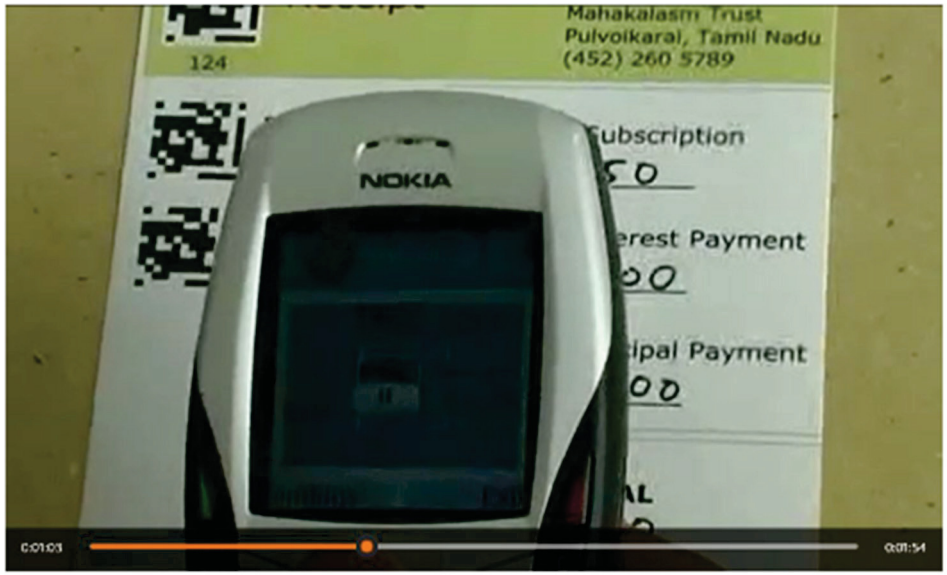
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing.

Claims	Prior Art
	CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Pervasive Computing) at 76 

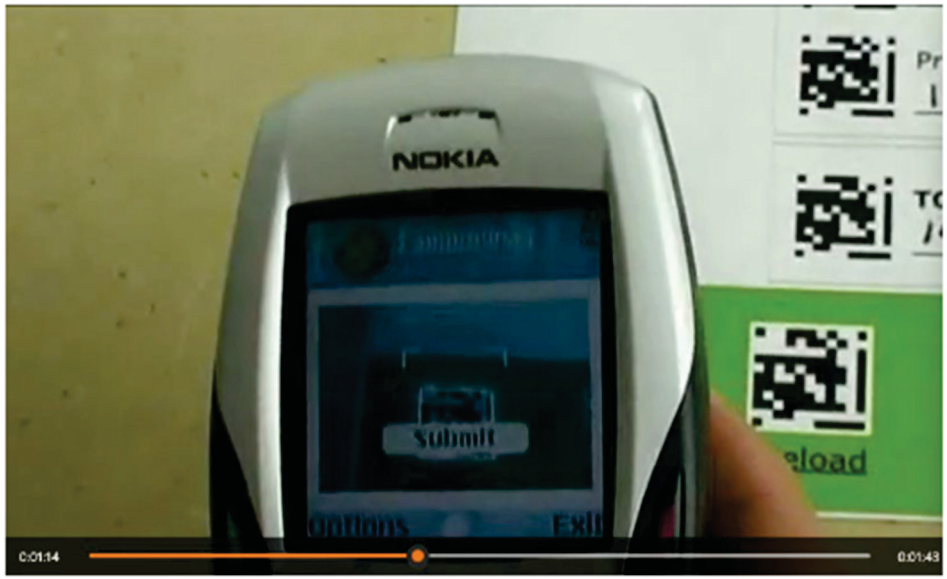
Claims	Prior Art
	CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning.

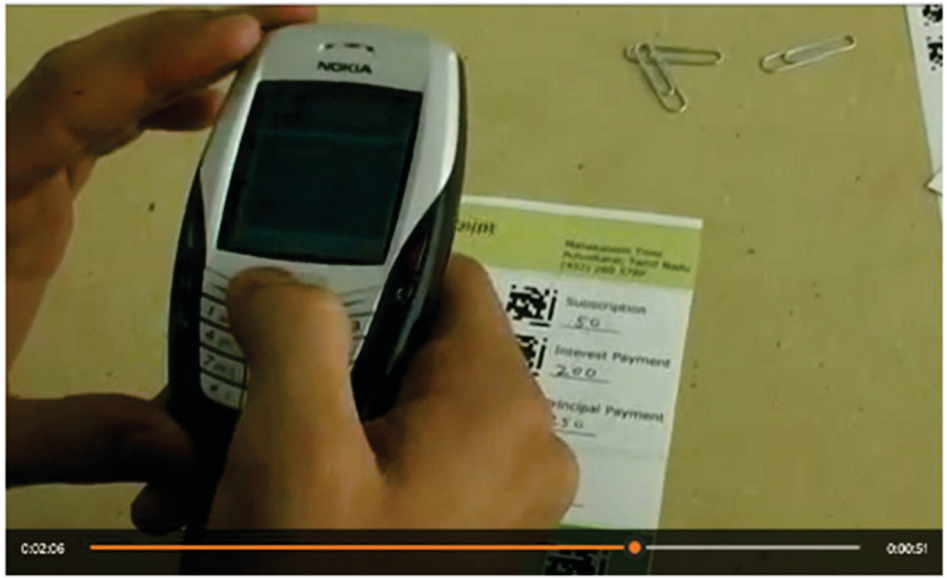
Claims	Prior Art
	CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current

Claims	Prior Art
	implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

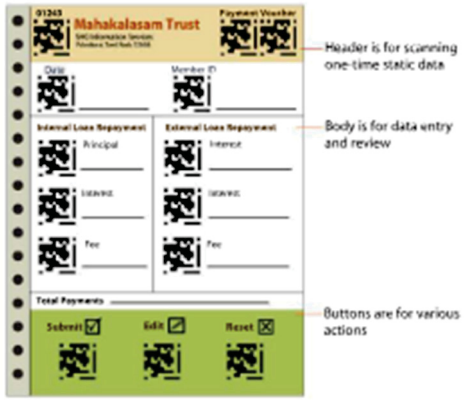
Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[b], 1[e], 1[g], 1[h]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1j] using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, wherein the digital image is the captured video frame or is based on the captured video frame; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, wherein the digital image is the captured video frame or is based on the captured video frame." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that

Claims	Prior Art
	contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met

Claims	Prior Art
	banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms.

Claims	Prior Art
	Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser.

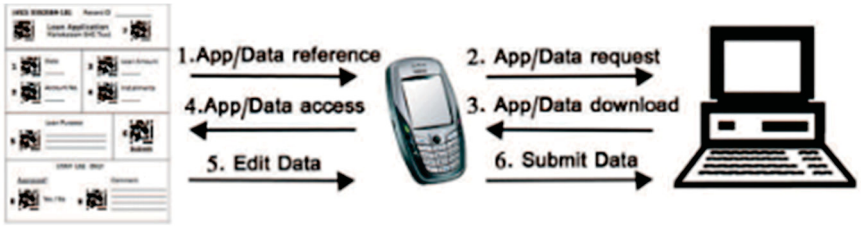
Claims	Prior Art
	CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach.

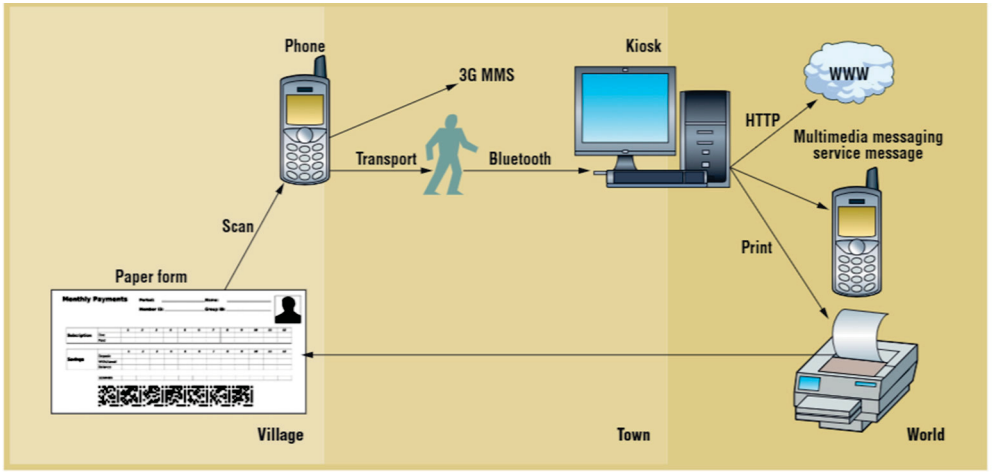
Claims	Prior Art
	CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing

Claims	Prior Art
	the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically. CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities.

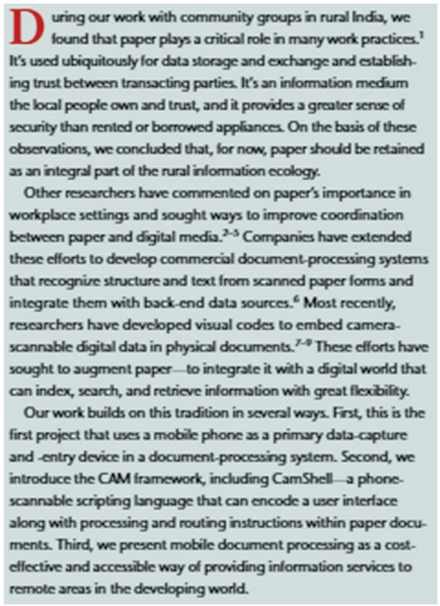
Claims	Prior Art
	CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural

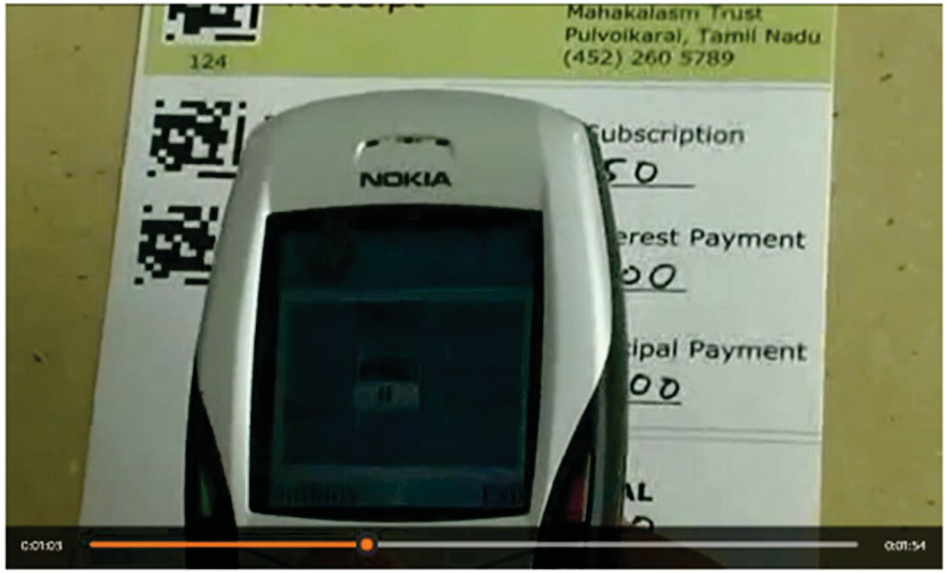
Claims	Prior Art
	information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Pervasive Computing) at 76  CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-

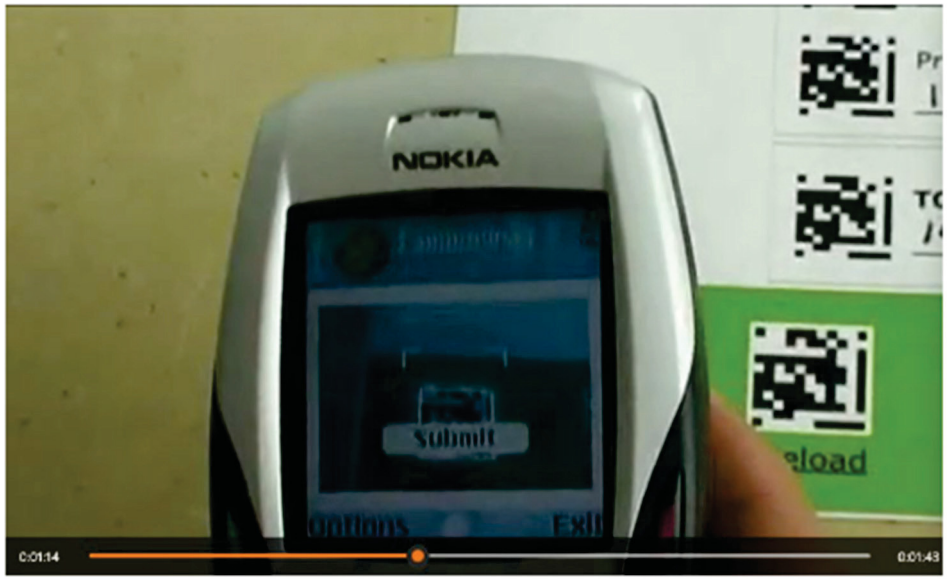
Claims	Prior Art
	carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example,

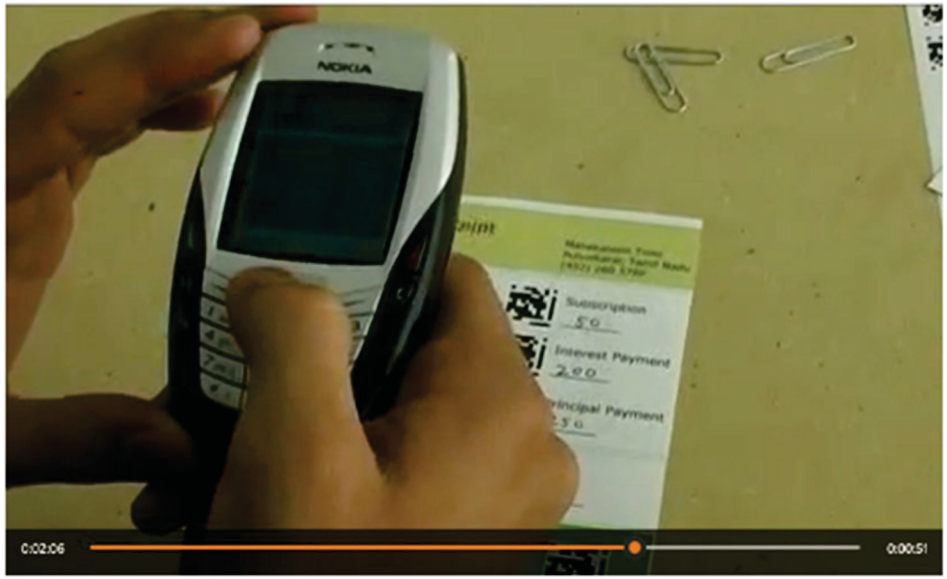
Claims	Prior Art
	when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76–77 The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content. In the case of a URL, the phone can operate in one of three network modes: [Sidebar] • online, in which the phone executes HTTP requests over an Internet connection. • offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection. • proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests. These network modes let users choose where and how to incur communications costs. If they live in an area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location.

Claims	Prior Art
	CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input. CAM (Pervasive Computing) at 77 The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to

Claims	Prior Art
	keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

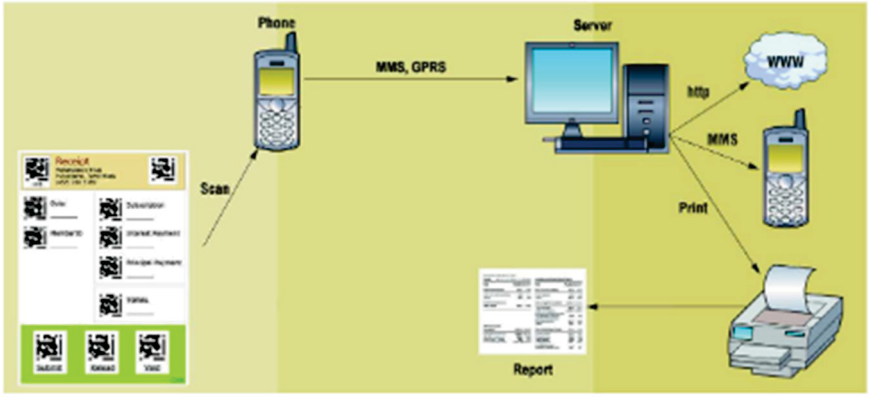
Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre], 1[e], 1[i]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1k] confirming that the mobile check deposit can go forward after optical character recognition is performed on the check in the digital image, a magnetic ink character recognition (MICR) line on the check in the digital image is recognized, and an amount payable on the check in the digital image is validated;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "confirming that the mobile check deposit can go forward after optical character recognition is performed on the check in the digital image, a magnetic ink character recognition (MICR) line on the check in the digital image is recognized, and an amount payable on the check in the digital image is validated." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that

Claims	Prior Art
	contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 The body contains input codes that are like HTML input tags. These are co-located with form fields, and are used to transcribe data from the document using the phone’s keypad. To edit a form field the user

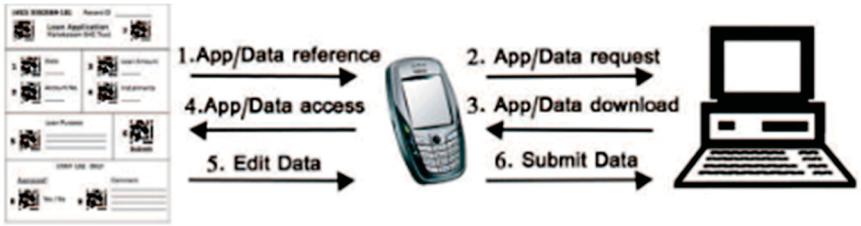
Claims	Prior Art
	must click on the input code, which brings up an editable dialog window. When a set of input codes is clicked together in one image, their respective dialogs are displayed in sequence. When the CamBrowser scans a visual code, the value returned by the associated scan callback function is shown on the screen. This is usually the value associated with the particular field. CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking

Claims	Prior Art
	instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 Each callback function can contain any number of sequential actions, some of which may be executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems

Claims	Prior Art
	inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 57 We have avoided using optical character recognition because of its high error rate and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad. In the future, I am considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream machines. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the

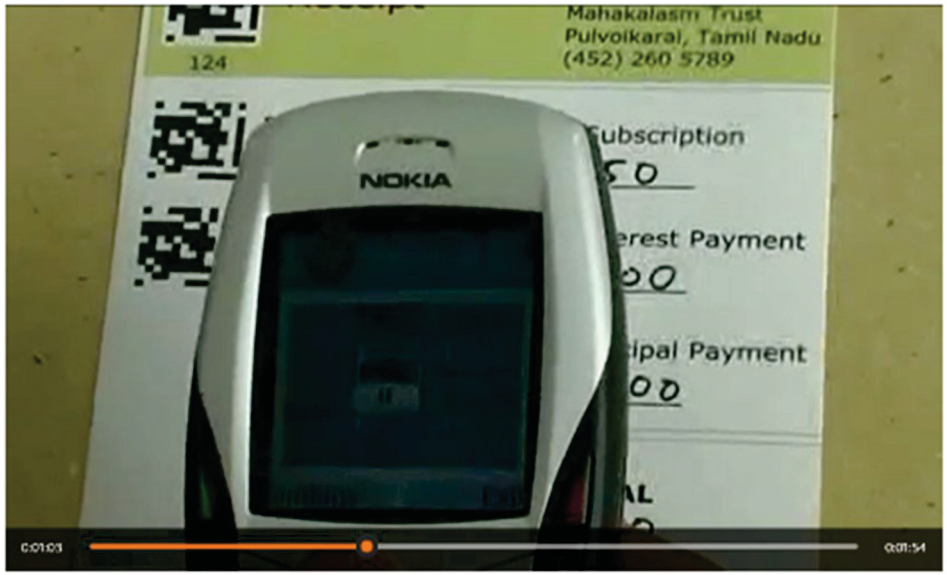
Claims	Prior Art
	SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

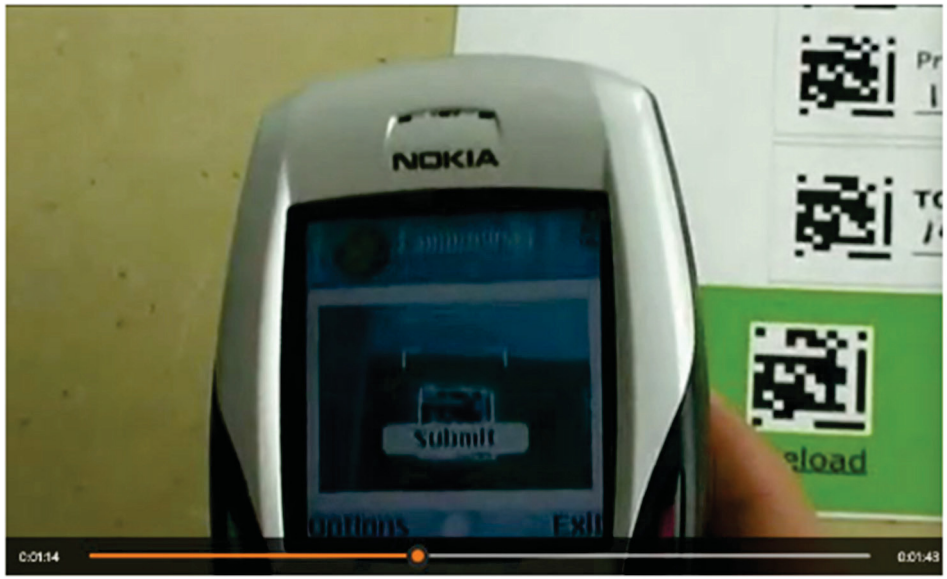
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document.

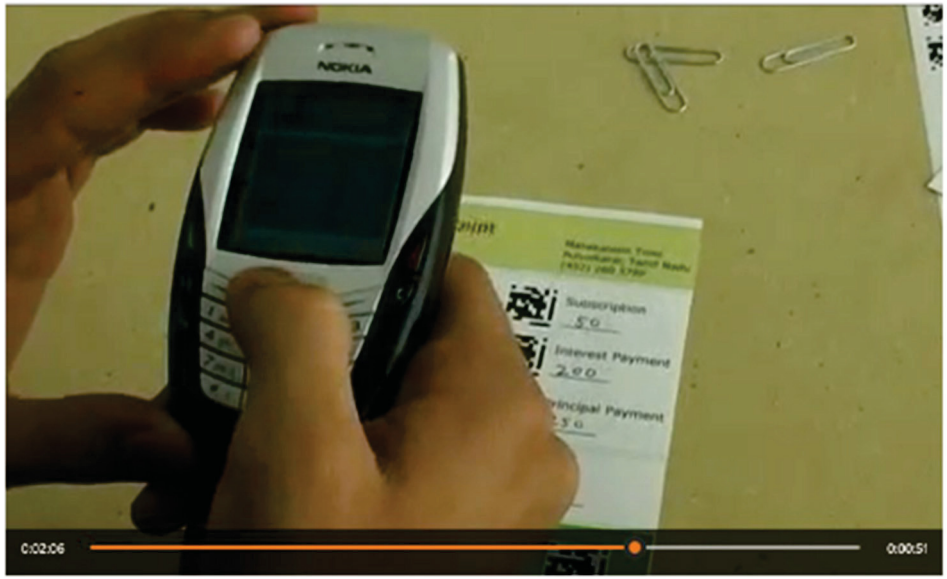
Claims	Prior Art
	We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream servers. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or

Claims	Prior Art
	recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

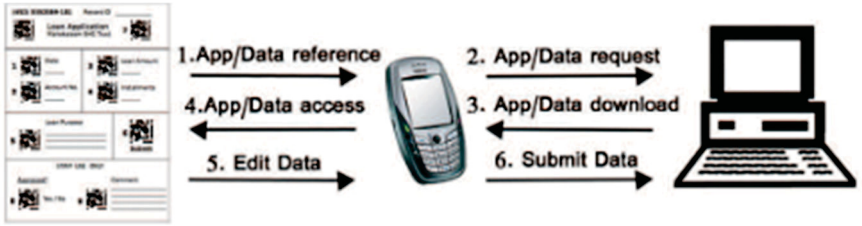
Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1l] a server, remote from the user's mobile device, comprising a processor coupled to a memory storing Instructions that, when executed by the processor, cause the server to perform operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a server, remote from the user's mobile device, comprising a processor coupled to a memory storing Instructions that, when executed by the processor, cause the server to perform operations including." For example: CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users.

Claims	Prior Art
	• HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 76–77 The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content. In the case of a URL, the phone can operate in one of three network modes: [Sidebar] • online, in which the phone executes HTTP requests over an Internet connection.

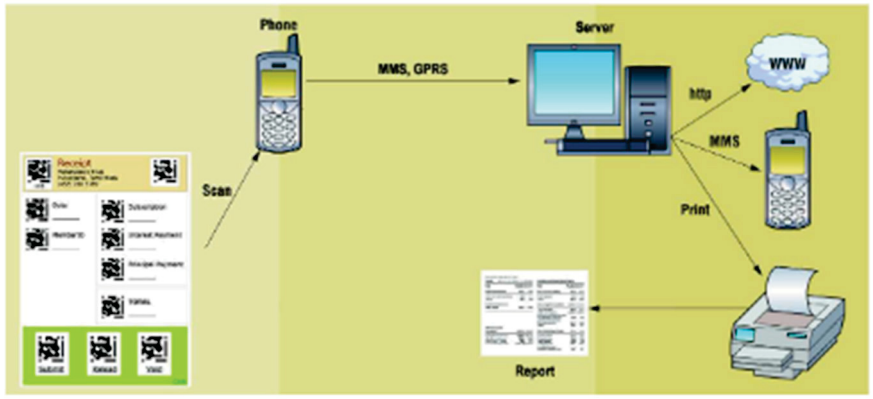
Claims	Prior Art
	• offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection. • proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests. These network modes let users choose where and how to incur communications costs. If they live in an area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location. CAM (Pervasive Computing) at 77 The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. *See also* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1m] receiving the digital image of the check transmitted by the user's mobile device, and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "receiving the digital image of the check transmitted by the user's mobile device." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that

Claims	Prior Art
	contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register

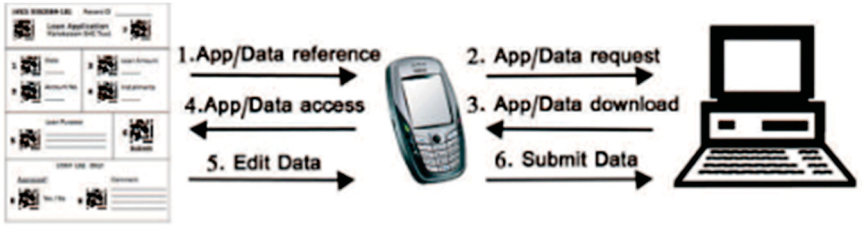
Claims	Prior Art
	- Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone’s camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in

Claims	Prior Art
	real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that

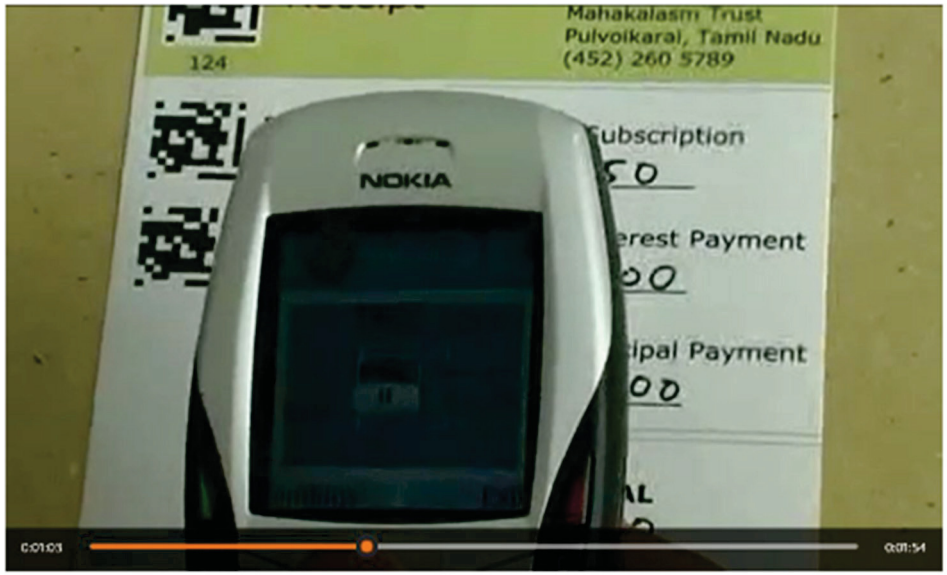
Claims	Prior Art
	could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection.

Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

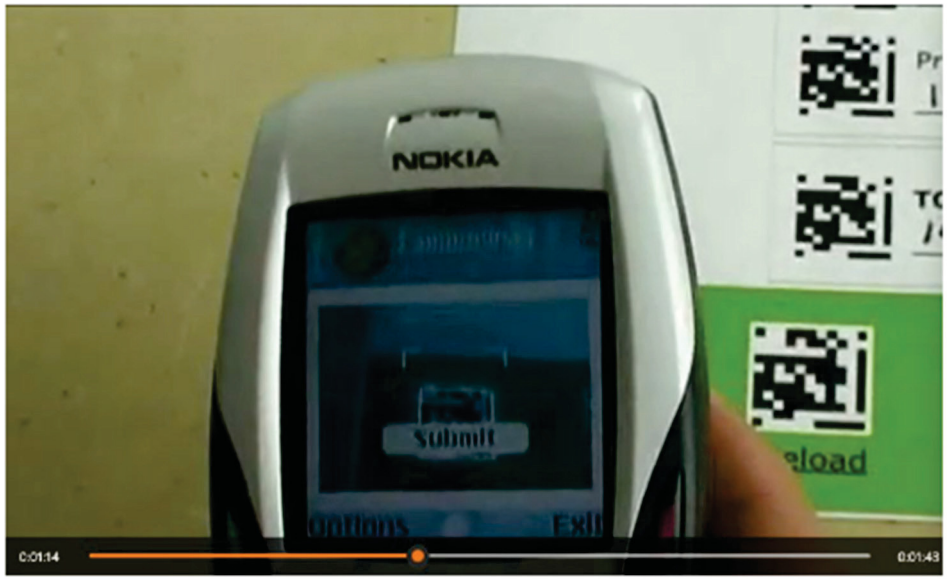
Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

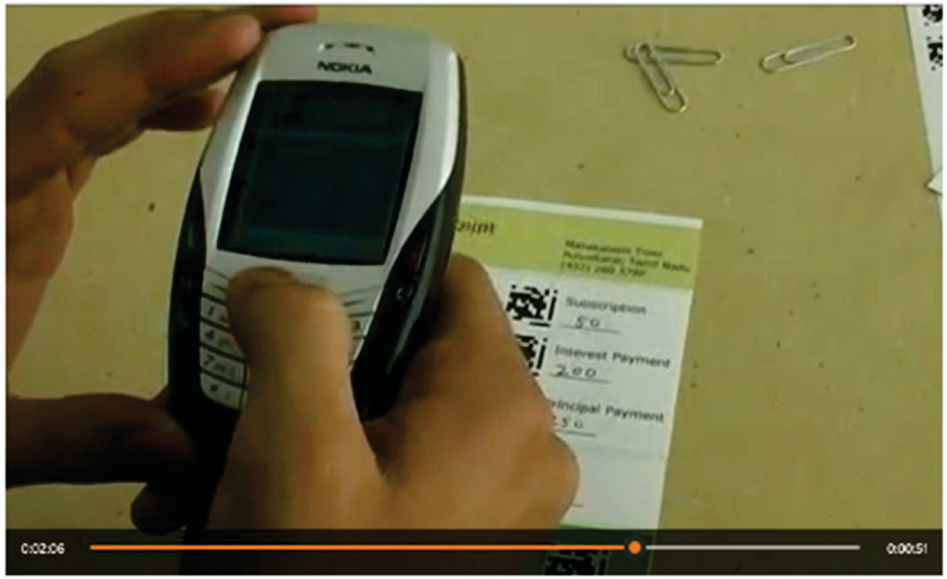
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76–77 The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content. In the case of a URL, the phone can operate in one of three network modes: [Sidebar] • online, in which the phone executes HTTP requests over an Internet connection. • offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection. • proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests. These network modes let users choose where and how to incur communications costs. If they live in an

Claims	Prior Art
	area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location. CAM (Pervasive Computing) at 77 The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

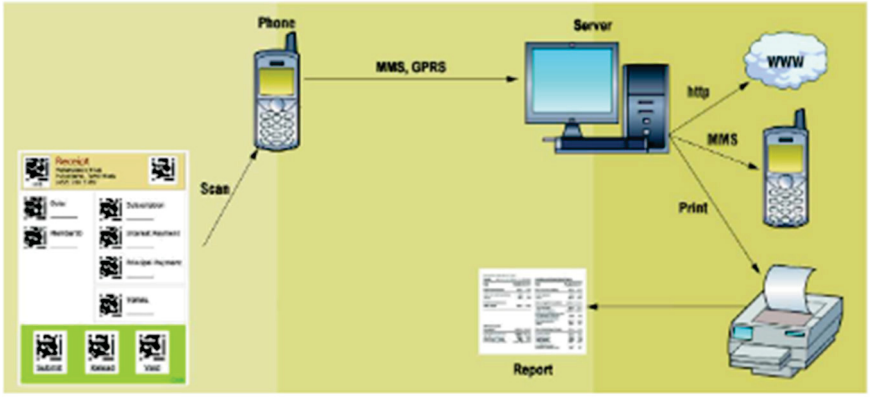
Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre], 1[j], 1[l]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1n] processing the digital image of the check including performing optical character recognition on the check in the digital image and verifying that the digital image of the check is acceptable for mobile check deposit, the user's mobile device receiving an Indication of a result of the server's processing before confirming that the mobile check deposit can go forward; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "processing the digital image of the check including performing optical character recognition on the check in the digital image and verifying that the digital image of the check is acceptable for mobile check deposit, the user's mobile device receiving an Indication of a result of the server's processing before confirming that the mobile check deposit can go forward." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile

Claims	Prior Art
	phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 The body contains input codes that are like HTML input tags. These are co-located with form fields, and

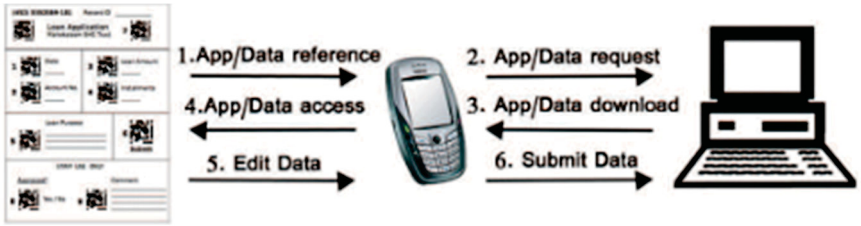
Claims	Prior Art
	are used to transcribe data from the document using the phone's keypad. To edit a form field the user must click on the input code, which brings up an editable dialog window. When a set of input codes is clicked together in one image, their respective dialogs are displayed in sequence. When the CamBrowser scans a visual code, the value returned by the associated scan callback function is shown on the screen. This is usually the value associated with the particular field. CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking

Claims	Prior Art
	instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 Each callback function can contain any number of sequential actions, some of which may be executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems

Claims	Prior Art
	inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 57 We have avoided using optical character recognition because of its high error rate and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad. In the future, I am considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream machines. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the

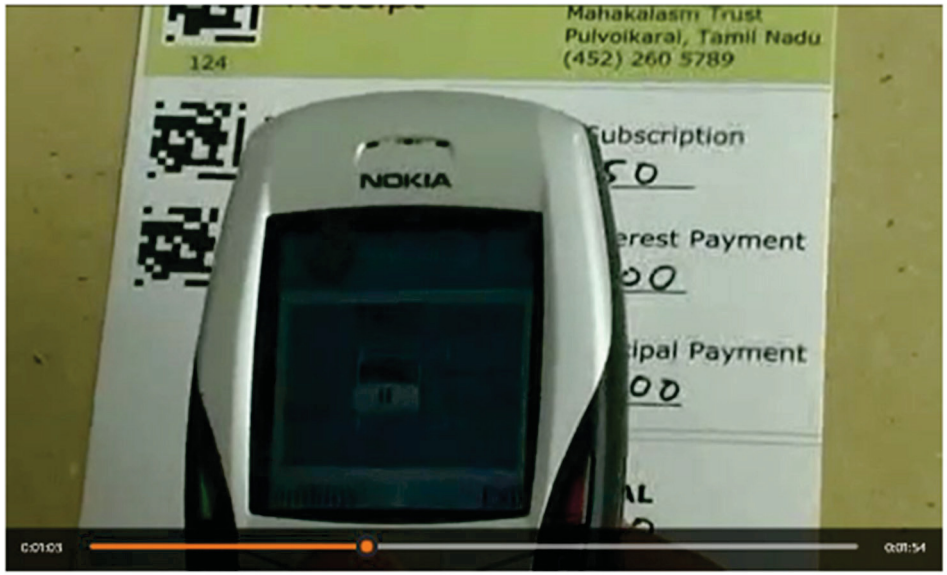
Claims	Prior Art
	SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

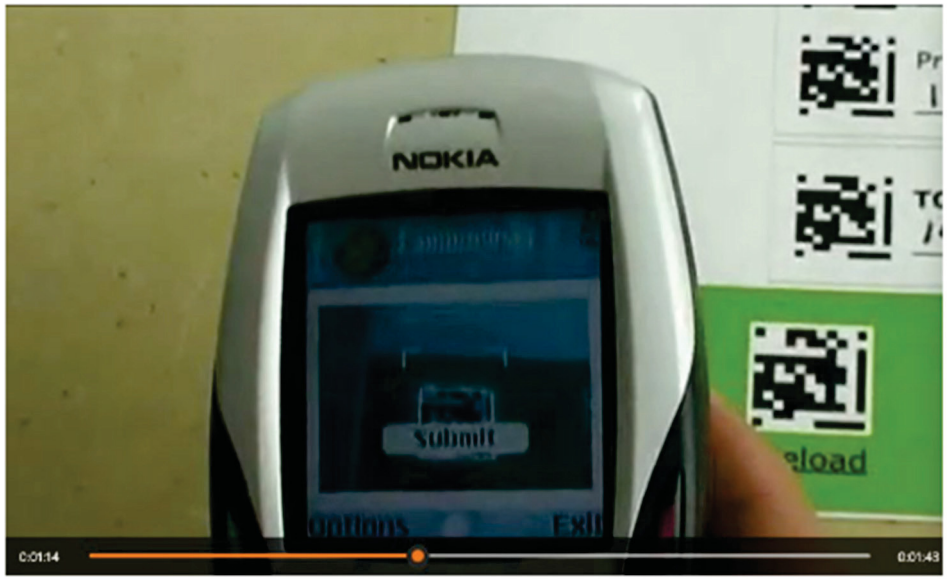
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document.

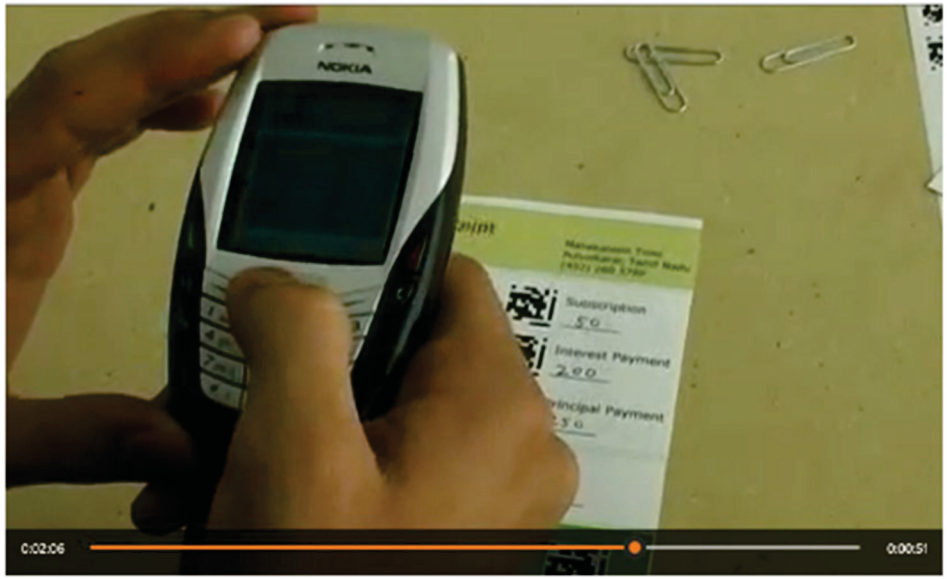
Claims	Prior Art
	We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream servers. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or

Claims	Prior Art
	recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre], 1[k], 1[l]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[1o] a computer comprising a processor coupled to a memory storing Instructions that, when executed by the processor, cause the computer to update a balance of an account at a financial institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the submitting the	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a computer comprising a processor coupled to a memory storing Instructions that, when executed by the processor, cause the computer to update a balance of an account at a financial institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the submitting the check for mobile check deposit in the bank, the confirming that the mobile check deposit can go forward, and the verifying that the digital image of the check is acceptable for mobile check deposit." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link

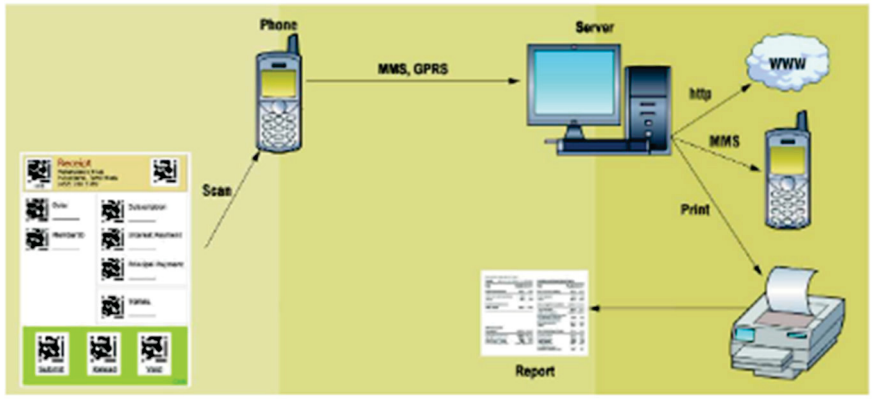
Claims	Prior Art
check for mobile check deposit in the bank, the confirming that the mobile check deposit can go forward, and the verifying that the digital image of the check is acceptable for mobile check deposit.	paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing]

Claims	Prior Art
	CAM (Parikh 2005a) at 2 The body contains input codes that are like HTML input tags. These are co-located with form fields, and are used to transcribe data from the document using the phone's keypad. To edit a form field the user must click on the input code, which brings up an editable dialog window. When a set of input codes is clicked together in one image, their respective dialogs are displayed in sequence. When the CamBrowser scans a visual code, the value returned by the associated scan callback function is shown on the screen. This is usually the value associated with the particular field. CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data

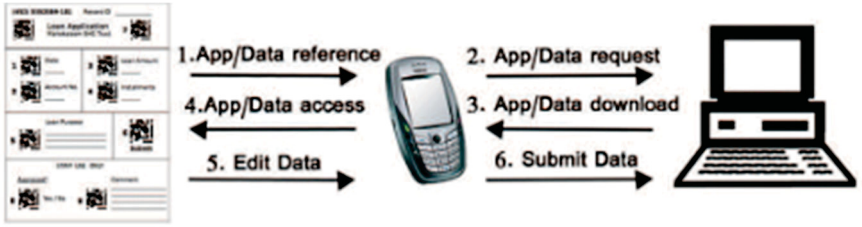
Claims	Prior Art
	entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 2 Each callback function can contain any number of sequential actions, some of which may be executed

Claims	Prior Art
	conditionally. Conditionals are useful when performing form validation. An example callback function is given below. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a

Claims	Prior Art
	barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 57 We have avoided using optical character recognition because of its high error rate and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad. In the future, I am considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream machines. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection.

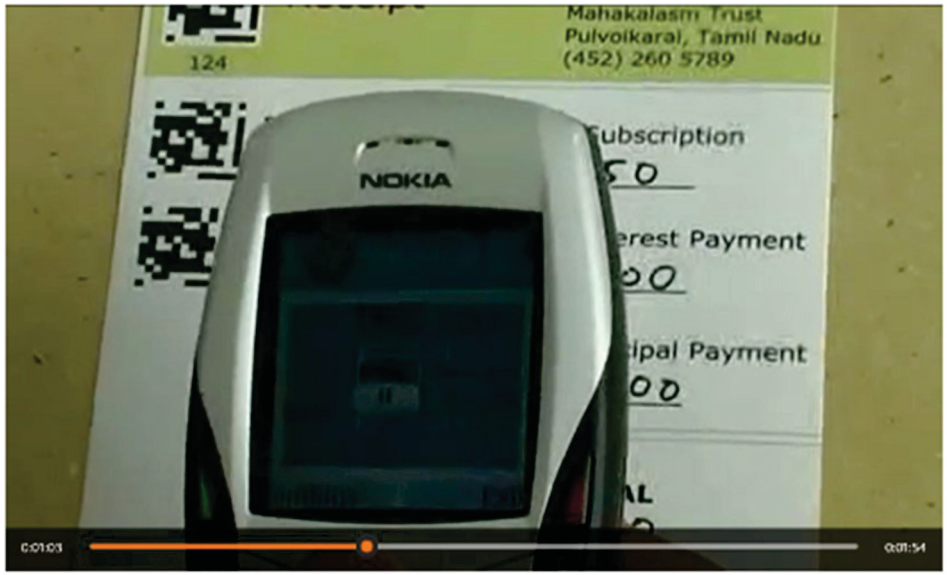
Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

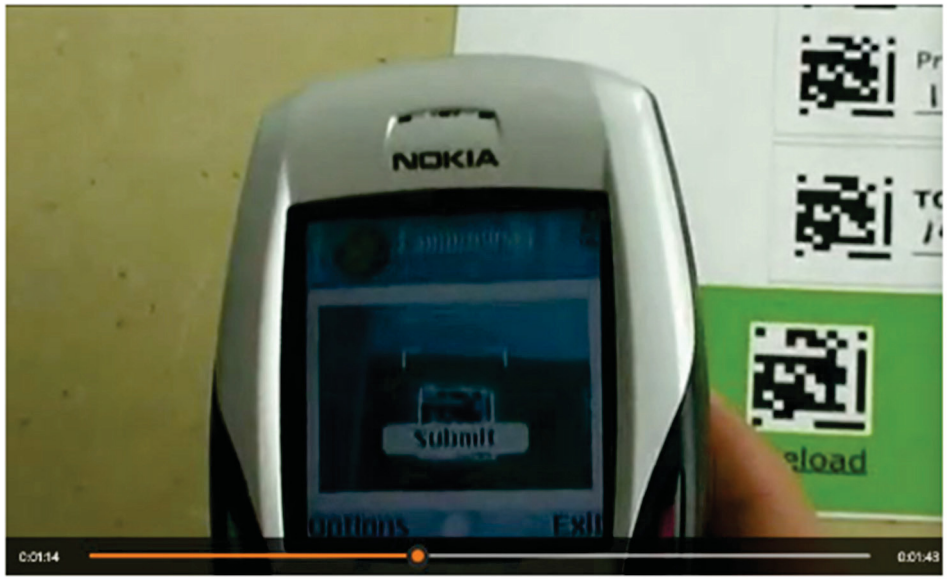
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode).

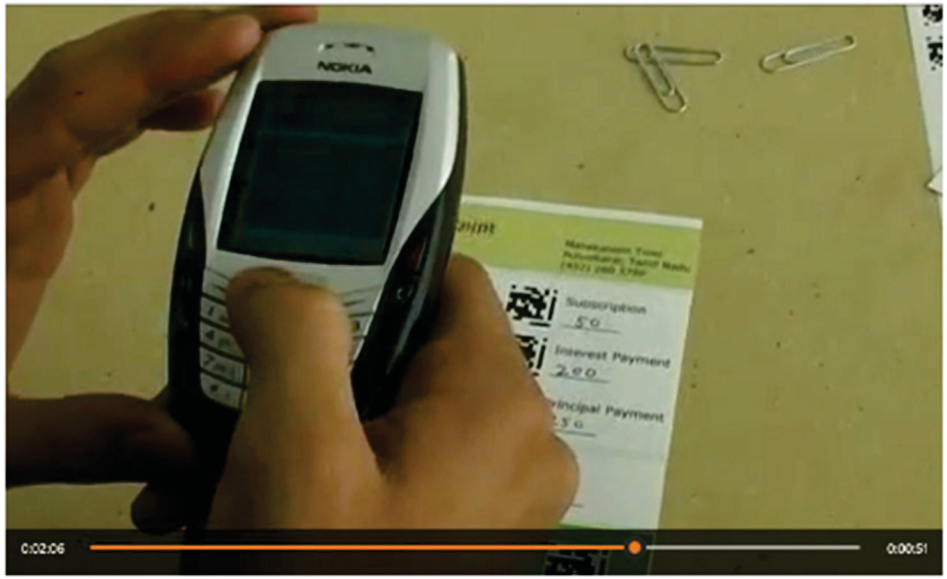
Claims	Prior Art
	We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream servers. CAM (Pervasive Computing) at 76–77 The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content. In the case of a URL, the phone can operate in one of three network modes: [Sidebar] • online, in which the phone executes HTTP requests over an Internet connection. • offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection. • proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests. These network modes let users choose where and how to incur communications costs. If they live in an area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location. CAM (Pervasive Computing) at 77 The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the

Claims	Prior Art
	CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

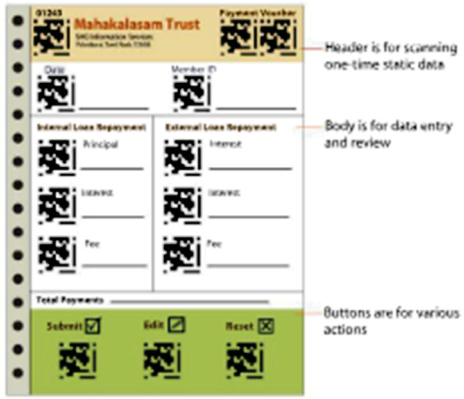
Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre], 1[kj]–1[l], 1[n]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
2[pre]. The system of claim 1, wherein the downloaded app causes the user's mobile device to perform additional operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 1, wherein the downloaded app causes the user's mobile device to perform additional operations including." For example: *See* limitations 1[pre], 1[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[2a] before determining that the video frame of the live video of the	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion,

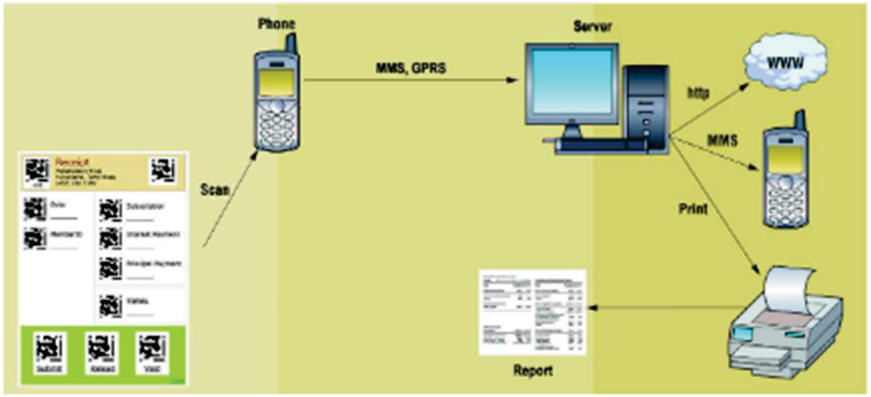
Claims	Prior Art
check satisfies the at least one monitoring criterion, determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion; and	determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion." For example: *See* limitations 1[pre]-1[b], 1[e], 1[g], 1[h], 2[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[2b] after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user." For example: *See* limitations 1[pre]-1[b], 1[e], 1[g]-1[i], 2[pre], 2[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
3. The system of claim 1, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 1, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people.

Claims	Prior Art
	CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register

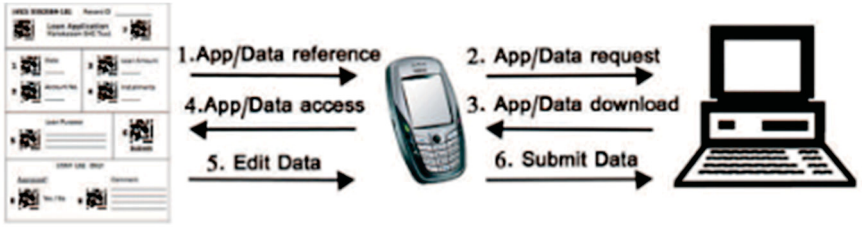
Claims	Prior Art
	- Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone’s camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in

Claims	Prior Art
	real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing

Claims	Prior Art
	world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the

Claims	Prior Art
	entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing.

Claims	Prior Art
	CAM (Pervasive Computing) at 75

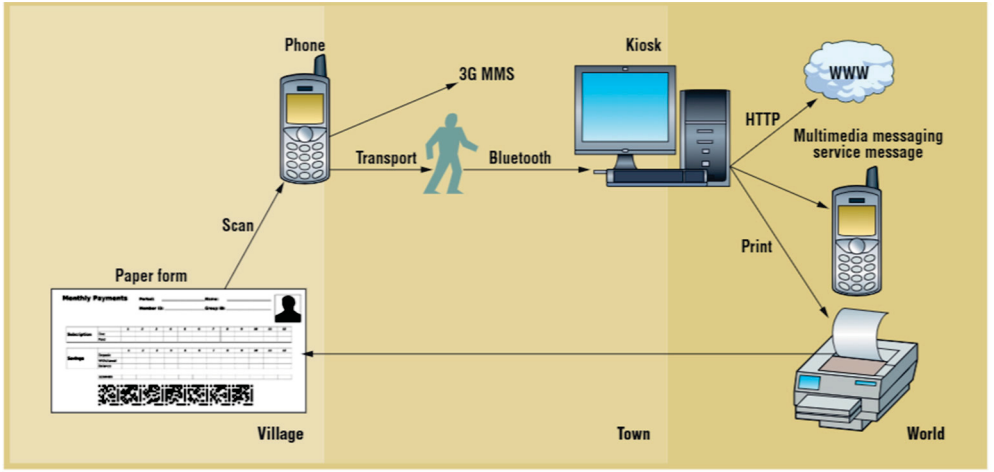
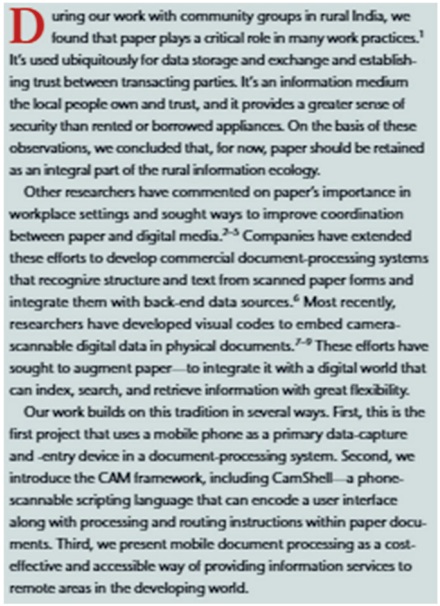


Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

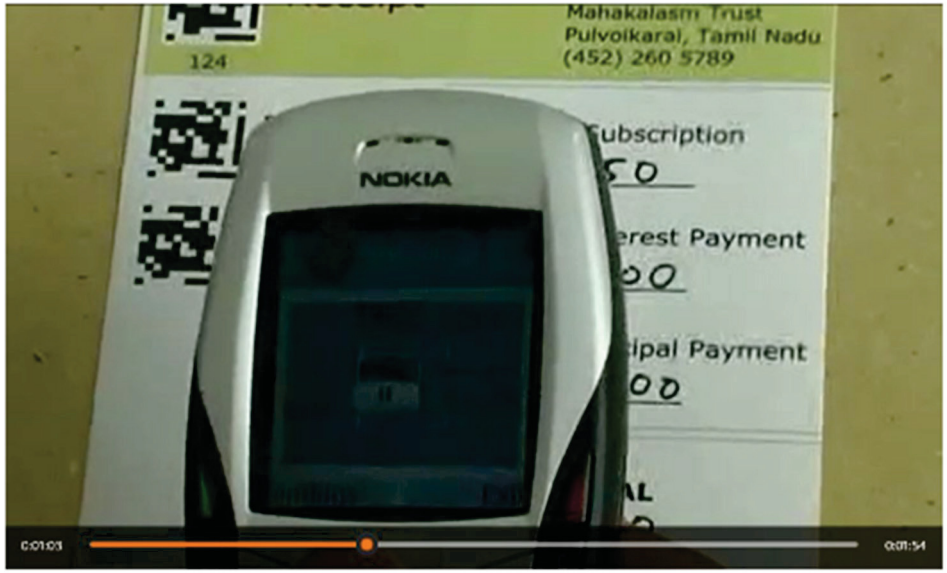
CAM (Pervasive Computing) at 76



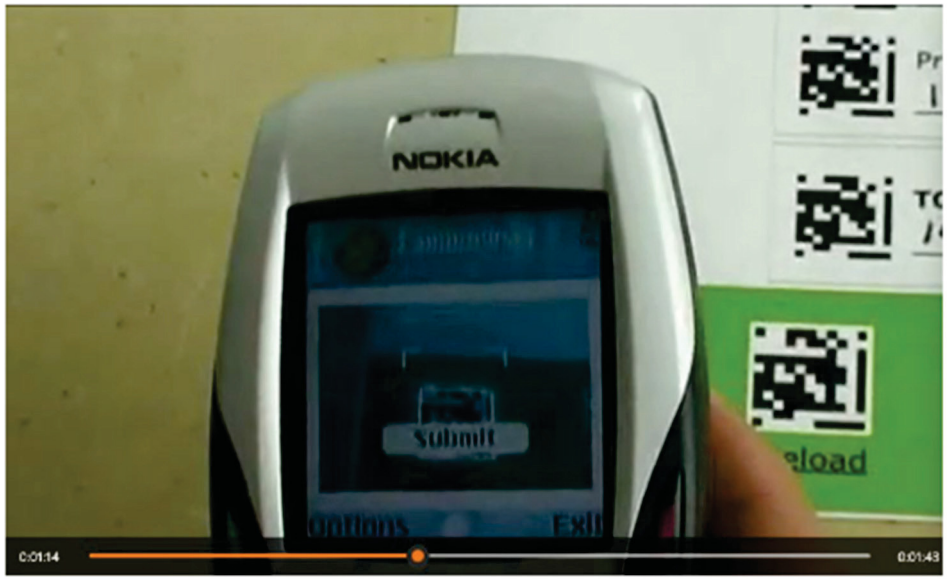
Claims	Prior Art
	CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning.

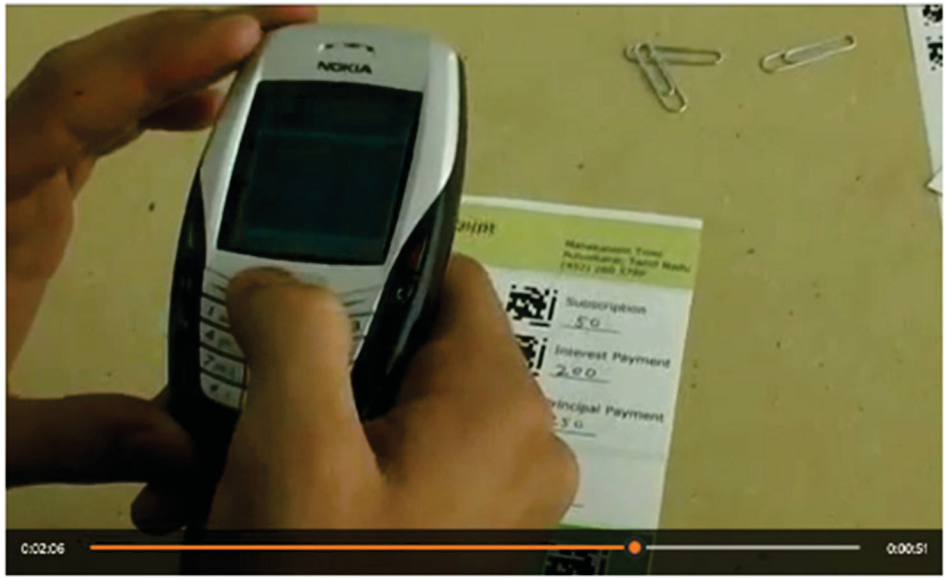
Claims	Prior Art
	CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful

Claims	Prior Art
	upstream servers. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker

Claims	Prior Art
	then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

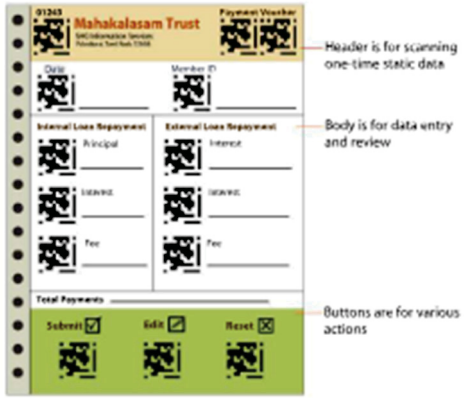
Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[pre]–1[b], 1[e], 1[g]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
4. The system of claim 3, wherein the plurality of monitoring criteria Includes a criterion based on magnetic ink character recognition (MICR) line detection.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 3, wherein the plurality of monitoring criteria Includes a criterion based on magnetic ink character recognition (MICR) line detection." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the

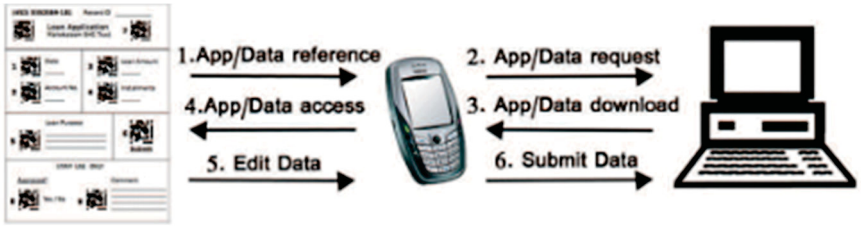
Claims	Prior Art
	CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and

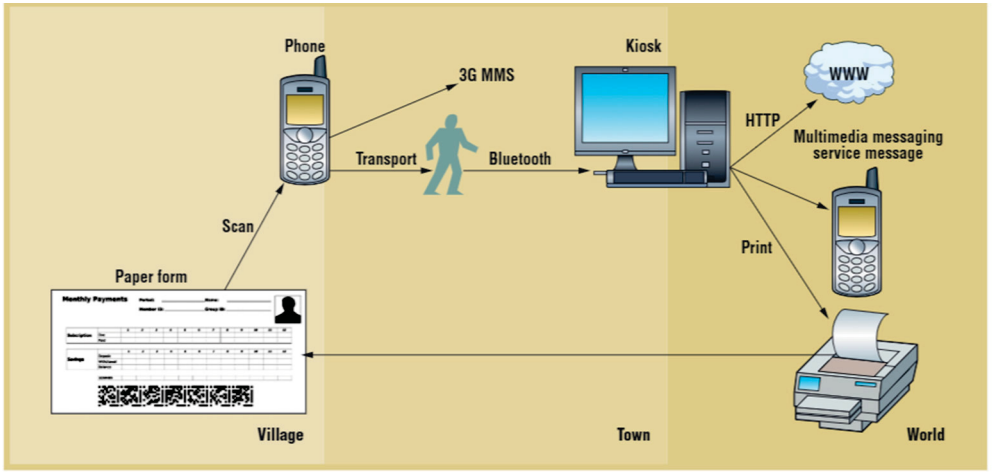
Claims	Prior Art
	translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device

Claims	Prior Art
	better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained

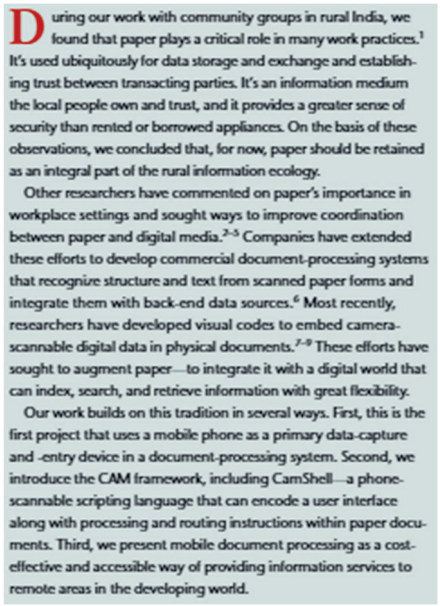
Claims	Prior Art
	contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

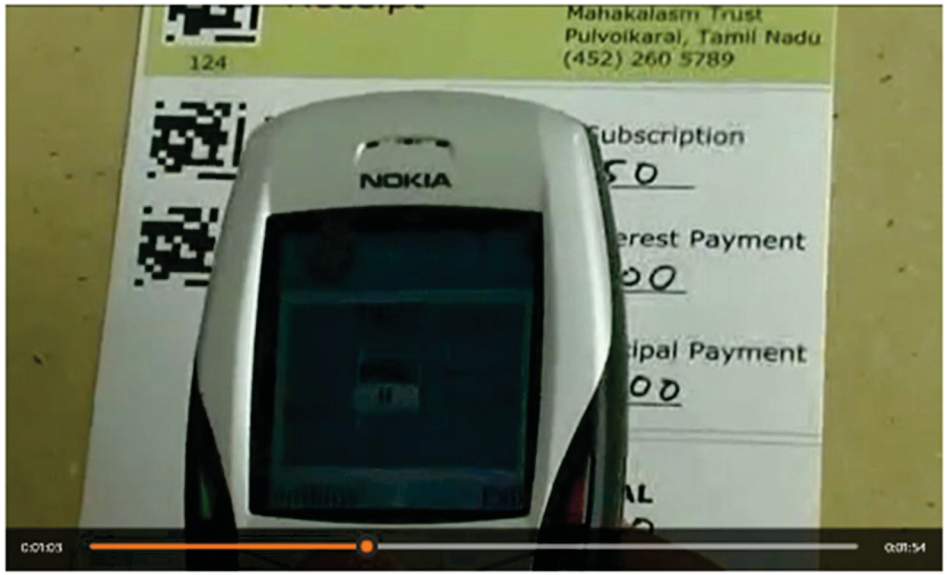
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural

Claims	Prior Art
	information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing. CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

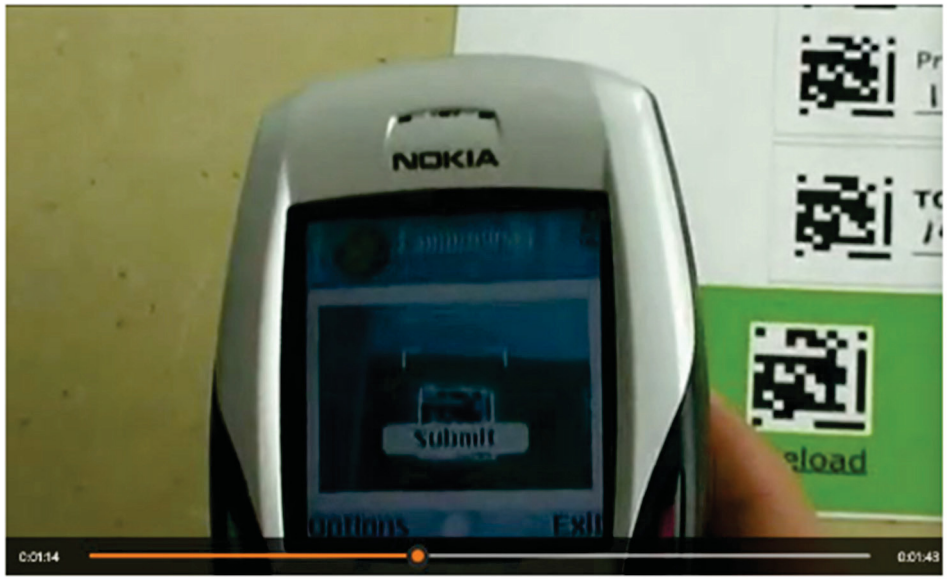
Claims	Prior Art
	CAM (Pervasive Computing) at 76  CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-

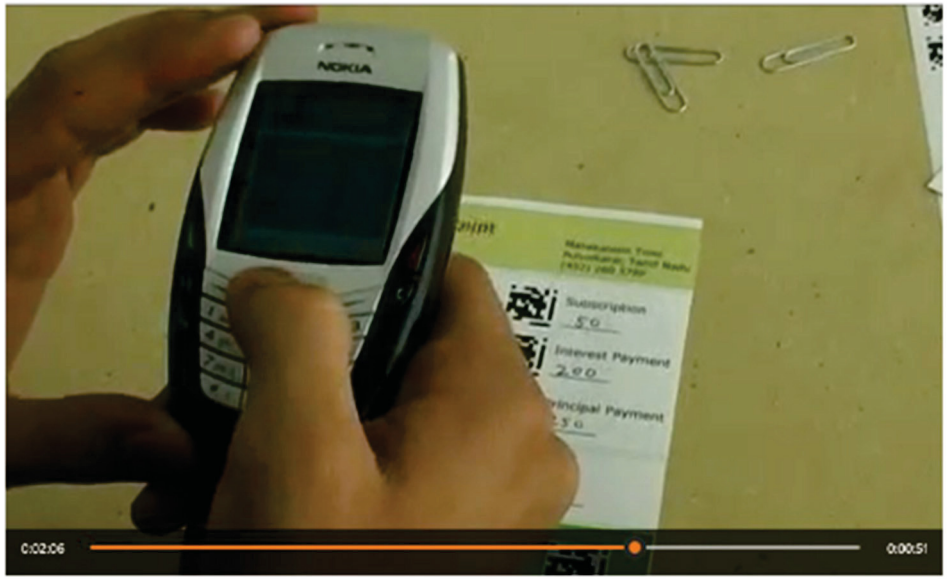
Claims	Prior Art
	carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more

Claims	Prior Art
	effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream servers. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered.

Claims	Prior Art
	CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

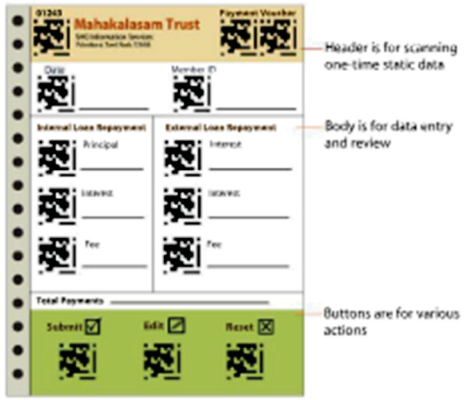
Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

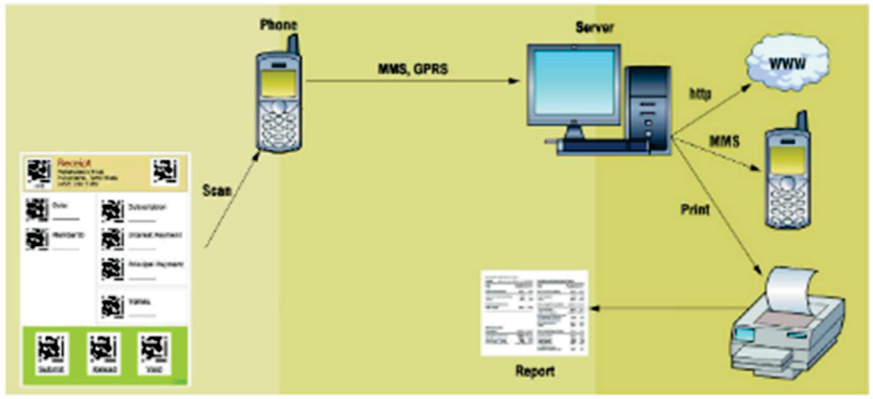
Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* claim 3. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
5. The system of claim 3, wherein the plurality of monitoring criteria Includes criteria based on light brightness, skewing, and warping.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 3, wherein the plurality of monitoring criteria Includes criteria based on light brightness, skewing, and warping." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of

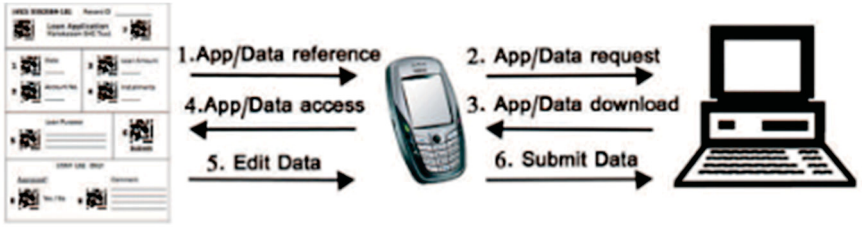
Claims	Prior Art
	users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions.

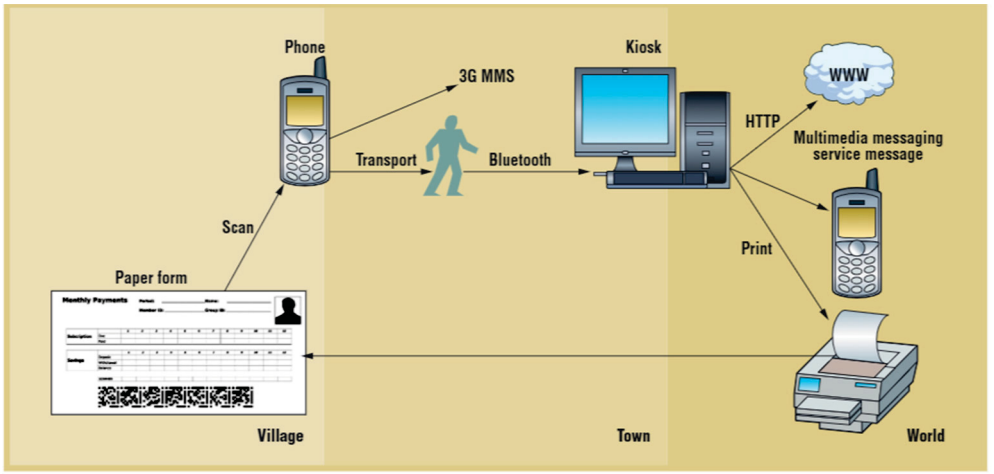
Claims	Prior Art
	CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach.

Claims	Prior Art
	CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users.

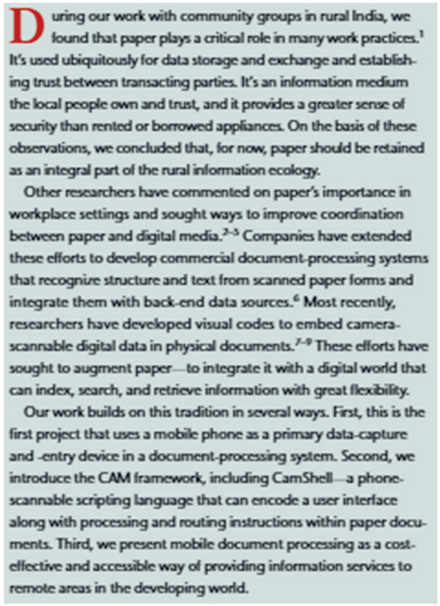
Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

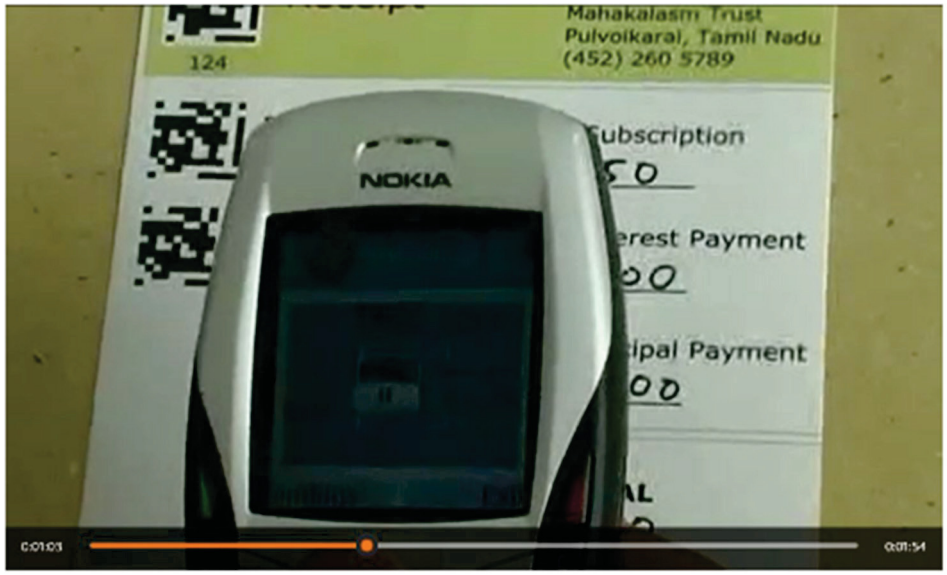
Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing.

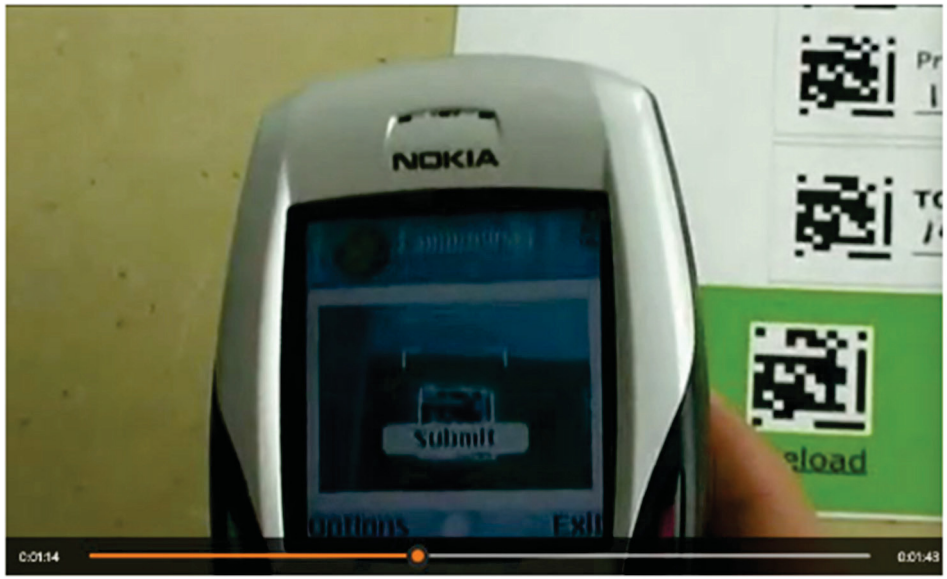
Claims	Prior Art
	CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Pervasive Computing) at 76 

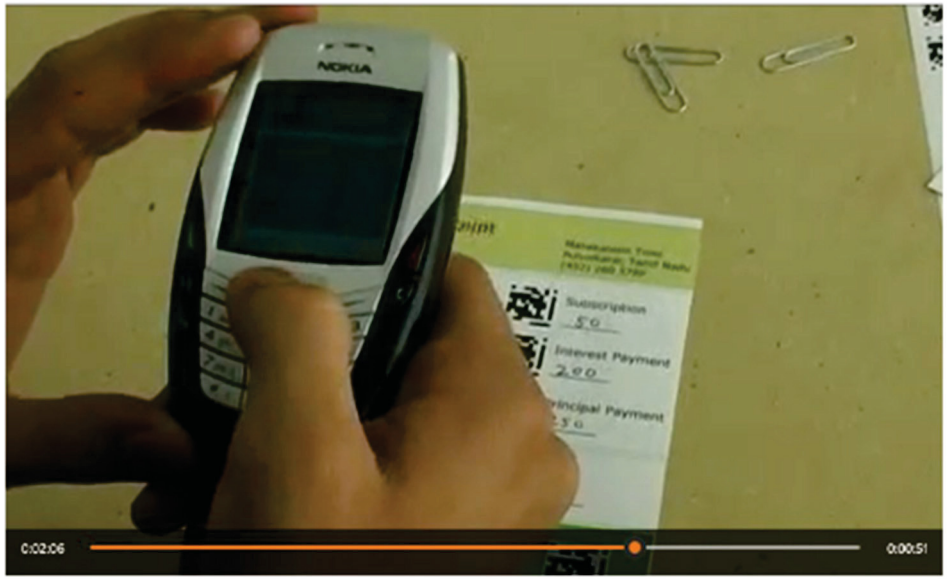
Claims	Prior Art
	CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning.

Claims	Prior Art
	CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like

Claims	Prior Art
	wizards that construct new CamForms based on prior input. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* claim 3. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
6. The system of claim 1, wherein the confirming that the mobile check deposit can go forward is performed after the transmitting the digital image of the check from the user's mobile device.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 1, wherein the confirming that the mobile check deposit can go forward is performed after the transmitting the digital image of the check from the user's mobile device." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By

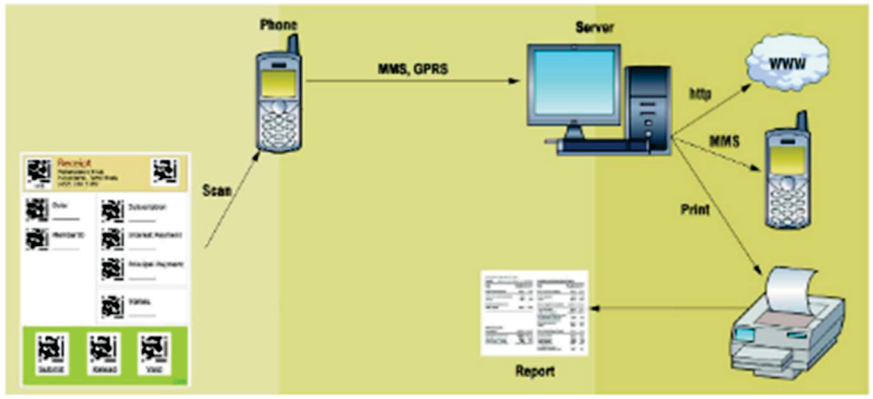
Claims	Prior Art
	combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 The body contains input codes that are like HTML input tags. These are co-located with form fields, and are used to transcribe data from the document using the phone’s keypad. To edit a form field the user must click on the input code, which brings up an editable dialog window. When a set of input codes is

Claims	Prior Art
	clicked together in one image, their respective dialogs are displayed in sequence. When the CamBrowser scans a visual code, the value returned by the associated scan callback function is shown on the screen. This is usually the value associated with the particular field. CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking

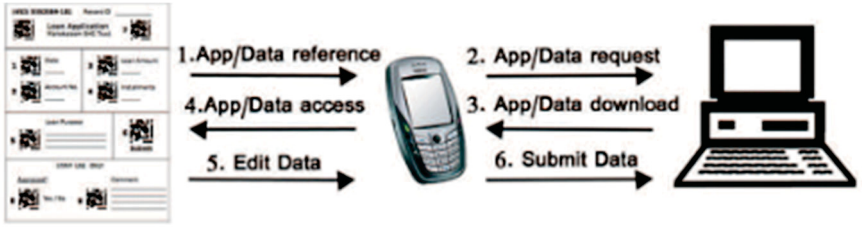
Claims	Prior Art
	instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 2 Each callback function can contain any number of sequential actions, some of which may be executed conditionally. Conditionals are useful when performing form validation. An example callback function is

Claims	Prior Art
	given below. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83

Claims	Prior Art
	bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 57 We have avoided using optical character recognition because of its high error rate and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad. In the future, I am considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream machines. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection.

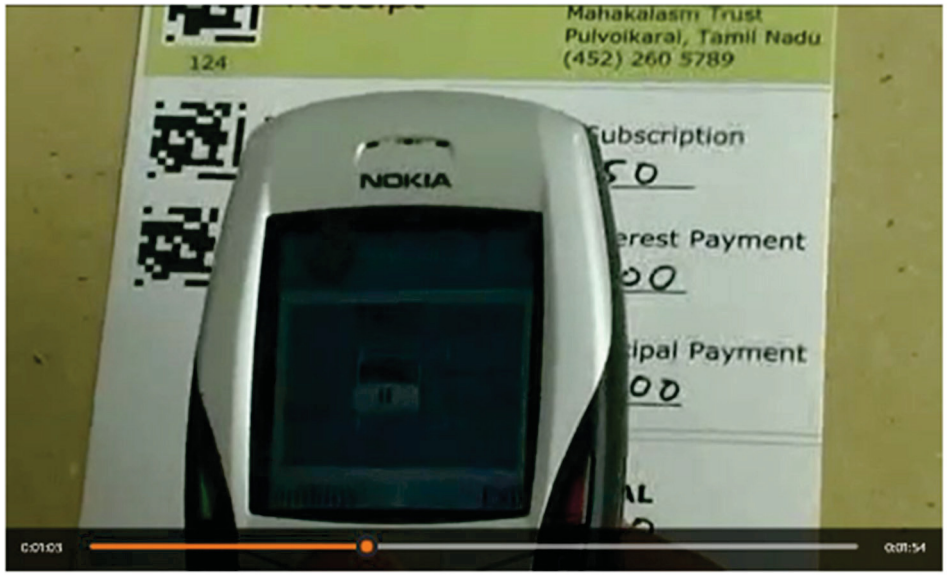
Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

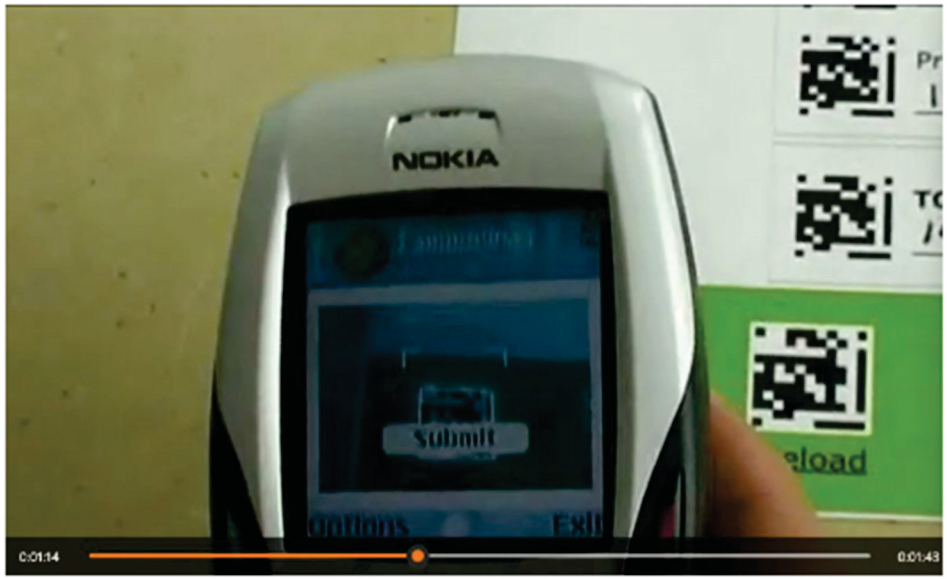
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode).

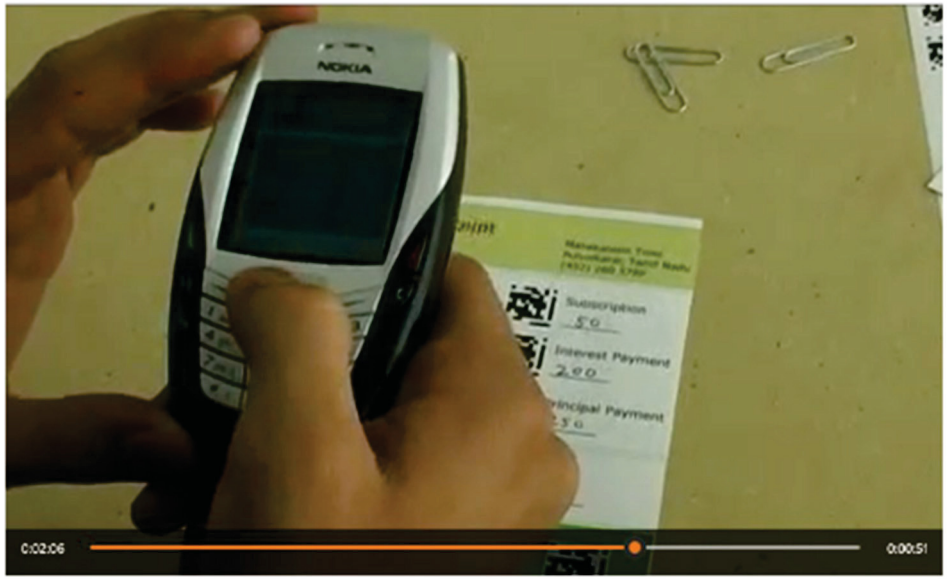
Claims	Prior Art
	We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful upstream servers. CAM (Pervasive Computing) at 76–77 The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content. In the case of a URL, the phone can operate in one of three network modes: [Sidebar] • online, in which the phone executes HTTP requests over an Internet connection. • offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection. • proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests. These network modes let users choose where and how to incur communications costs. If they live in an area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location. CAM (Pervasive Computing) at 77 The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the

Claims	Prior Art
	CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

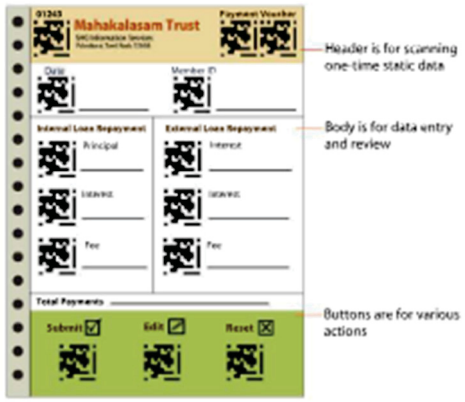
Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[i], 1[k]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
7. The system of claim 1, wherein the digital image is a modified version of the captured video frame.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 1, wherein the digital image is a modified version of the captured video frame." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous

Claims	Prior Art
	connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring:

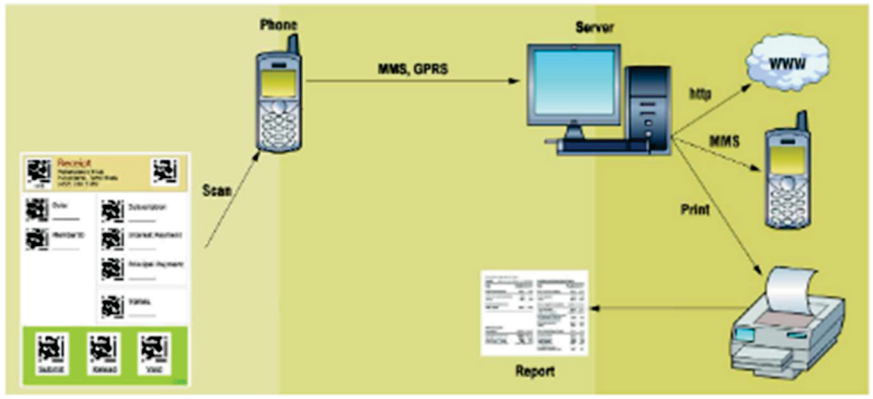
Claims	Prior Art
	- Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone’s camera relative to an individual visual code [19]. However it also has been

Claims	Prior Art
	observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number. • Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with.

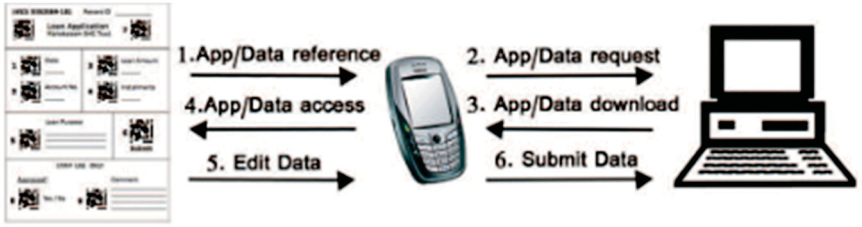
Claims	Prior Art
	CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 50 Developing custom applications for mobile phones is becoming more common. However, this requires knowledge of new APIs (some of which require licensing). Distributing applications is cumbersome. Either providers must push content to users, or users must navigate to web sites and download the software. Due to the difficulty of using the mobile web and the latency in the mobile data network, the easiest way to install an application is by using a PC to download the installer and transfering it to the phone using bluetooth or USB. CAM (Parikh 2007) at 52 The CamBrowser downloads and executes applications written in Simkin, an XML-based scripting language which provides support for function calls, control flow, arithmetic and basic datatypes [145]. CAM provides an API for accessing the mobile phone's user interface, networking and multimedia input and output capabilities. Applications are downloaded on demand from an online source, either via the web or a multimedia message (MMS). The XML is cached locally on the phone along with any associated

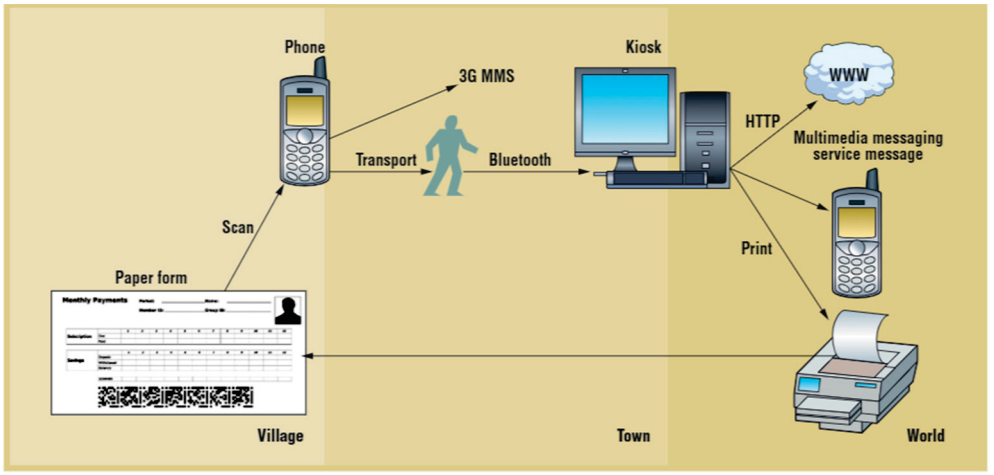
Claims	Prior Art
	files, in a directory structure organized according to the data server and application ID. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 56 Figure 4.2 shows a CAM-enabled loan application designed for a microfinance institution. The barcode in the top right, when clicked, activates a function that displays a sequence of prompts for the user to enter each of the values from the form. A section of the code for this function is shown below. input_date, conf_note, recordaudio and put are all CAM API functions. A full list of API functions is given in Table 4.2. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 60 Eventually, some applications may have to be flushed from the cache to make room for others. By implementing an intelligent caching scheme, we can ensure that the most frequently used applications are retained. This has not been a requirement thus far due to the small number of CAM applications that have been developed. All of these fit within the available flash memory of the phone (32 MB, expandable to 1 GB). The user can also manually request a flush of the local cache, or force a server refresh of a particular application. The CAM API also supports sending a refresh request to the server programmatically.

Claims	Prior Art
	CAM (Parikh 2007) at 69 Because there is a negligible legacy content or application base in the rural developing world, CAM is not constrained by the limitations of current platforms or protocols. Instead of using menus, CAM applications are navigated with barcodes and numbers. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page-by-page. CAM also takes full advantage of the mobile phone's multimedia input and output capabilities. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users.

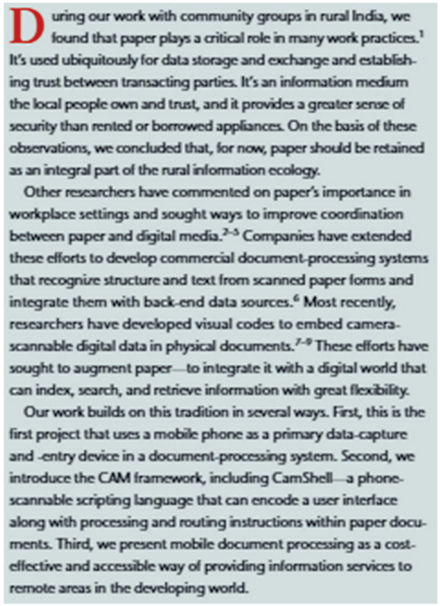
Claims	Prior Art
	CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

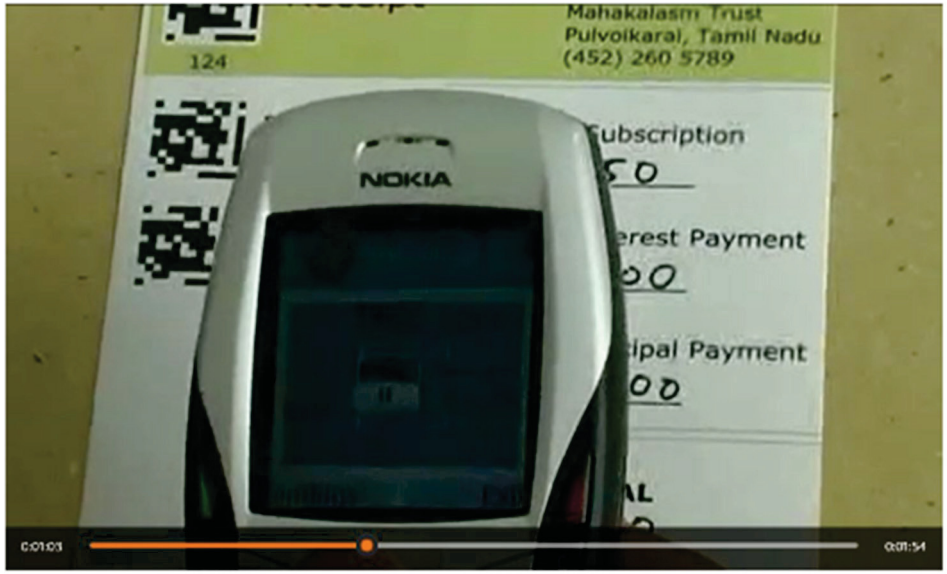
Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing.

Claims	Prior Art
	CAM (Pervasive Computing) at 75  Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Pervasive Computing) at 76 

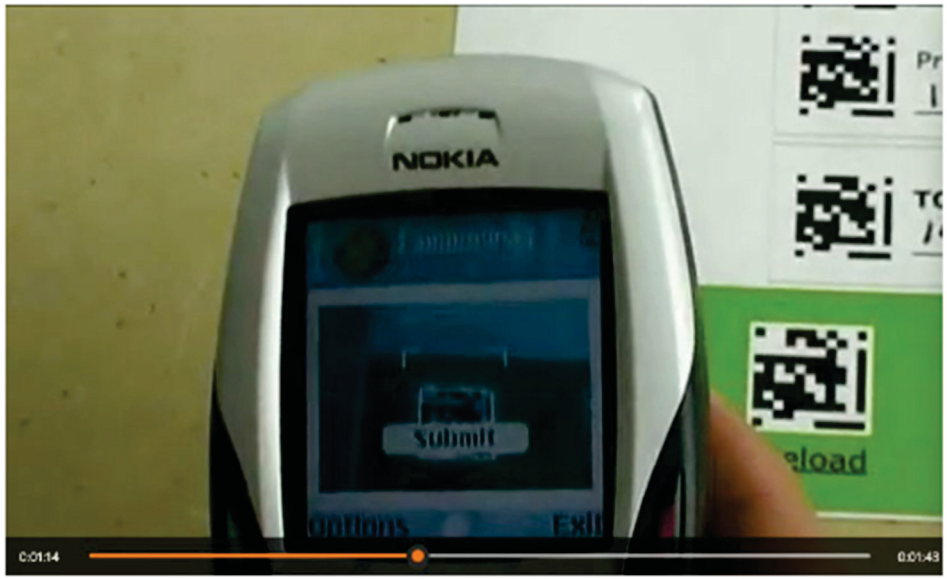
Claims	Prior Art
	CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning.

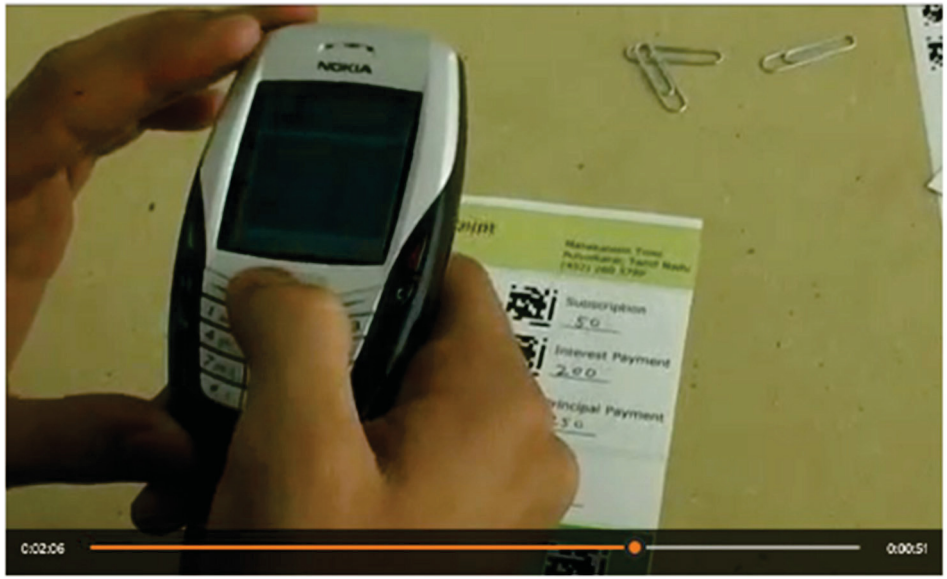
Claims	Prior Art
	CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 The main instruction classes are • assignment instructions, which assign a 32-bit value to a 32-bit identifier. They can be used to encode hidden values in documents. • dialog instructions, which launch different types of dialogs (number, currency, date, text, and so on), letting the user transcribe data from the document. The user-entered value is assigned to a 64-bit identifier, which serves as the dialog prompt. This instruction can also be used to send notification messages to the user. • control and arithmetic instructions, which are used for simple arithmetic and branches. They're useful for encoding loops—for example, when reading data from a table. • application-launching instructions, which open other applications in separate threads (such as Web and Wireless Application Protocol browsers), sending the current variables as form parameters. Other options include making a phone call or sending a multimedia messaging service (MMS) message. • multimedia instructions, which open predefined audio and image files. These provide for audio and graphical prompts, which we've found useful for interacting with semiliterate users. In our current

Claims	Prior Art
	implementation, prerecorded voice prompts accompany all dialogs if so desired. • networking and security instructions, which provide for secure communications with CamServer. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input. CAM (Pervasive Computing) at 78 The CAM architecture's distributed nature makes security an important concern. CamShell provides instructions for performing encryption, digital signing, or both (signcryption). These instructions include digital keys and can be embedded either directly in CamForms or within special security tokens, called CamTokens. When the CamBrowser processes these instructions, the software automatically signs and/or encrypts the collected data before transmission. CAM application developers can use these instructions to create secure, signed communications channels between a CamForm client and the CamServer. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee.

Claims	Prior Art
	When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitations 1[j], 1[m], 1[o]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
8. The system of claim 1, wherein the wireless network is a cellular network.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 1, wherein the wireless network is a cellular network." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous

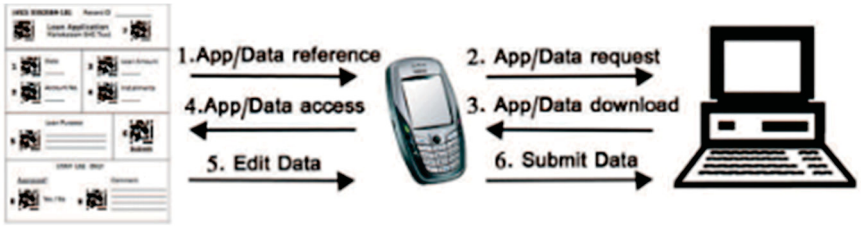
Claims	Prior Art
	connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world.

Claims	Prior Art
	CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions. CAM (Parikh 2005b) at 2 CamShell provides an API for accessing the mobile phone's functionality - including the phone's user interface, networking and telephony features. Listed below are the main functions currently included in this API: • User Dialogs - These instructions launch various user dialogs to collect data, get confirmation or convey a message to the user. Each dialog can be associated with audio and graphical prompts. Audio feedback has been found very useful by potential users. • HTTP Post, Get - These instructions are used to make HTTP requests using the phone's built-in connectivity. • SMS, MMS, Email - These instructions transmit asynchronous network messages. In the case of MMS and Email the current form image can be attached. • Phone Call - This instruction is used to make a phone call to a particular number.

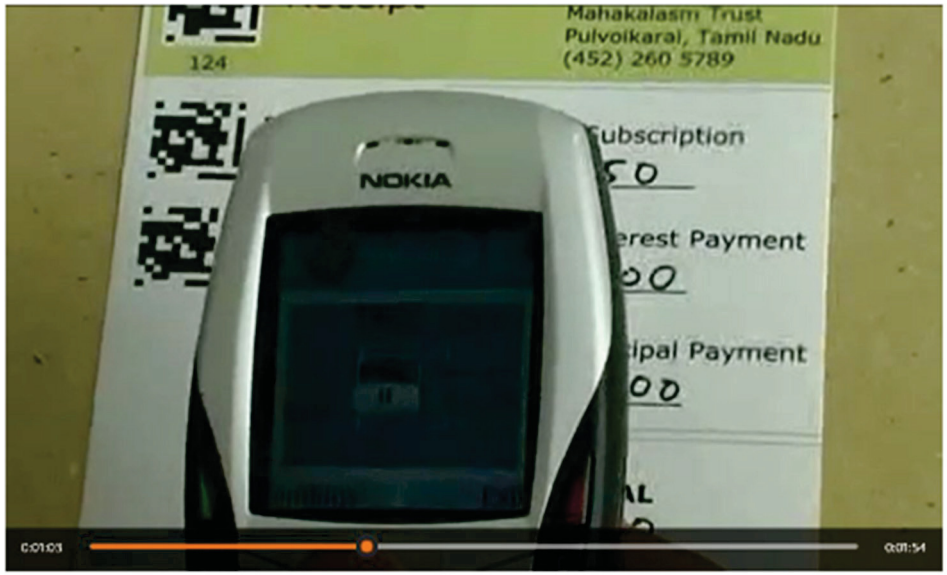
Claims	Prior Art
	• Applications - These instructions are used to launch other applications, such as the Web or WAP browser. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context. Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-

Claims	Prior Art
	based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 59 4.1.4 Networking The first time CamBrowser encounters an application, it attempts to download the code and data from the specified server. The barcode or numeric string specifies the protocol to submit the request. Currently, the options are sending a HTTP request to a web server, or a SMS to a phone number. In the first case, the response is received immediately as XML over HTTP. In the second case, the application code and data is sent by the server as a MMS message with a SIS attachment [2]. Local applications can be pre-loaded over a bluetooth OBEX connection. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

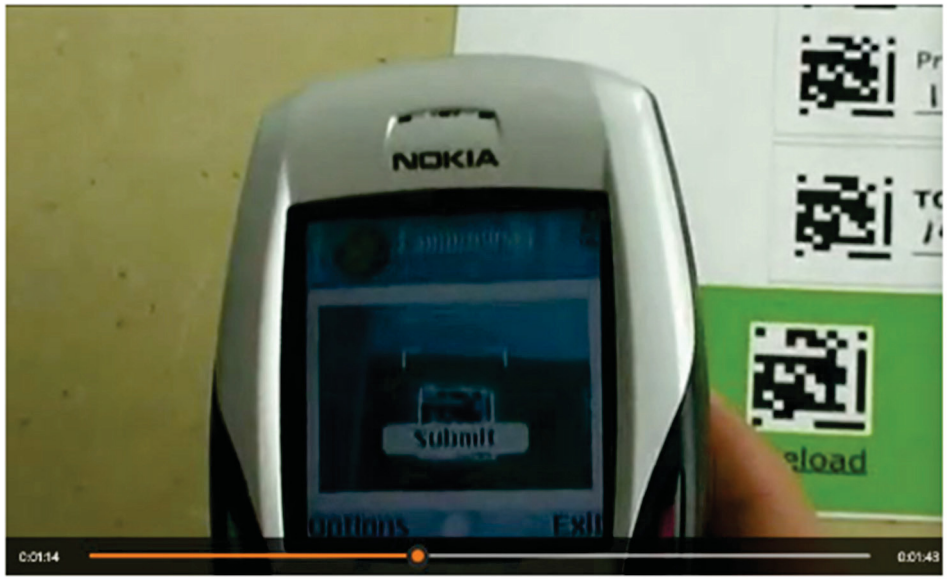
Claims	Prior Art
	CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code. CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played.

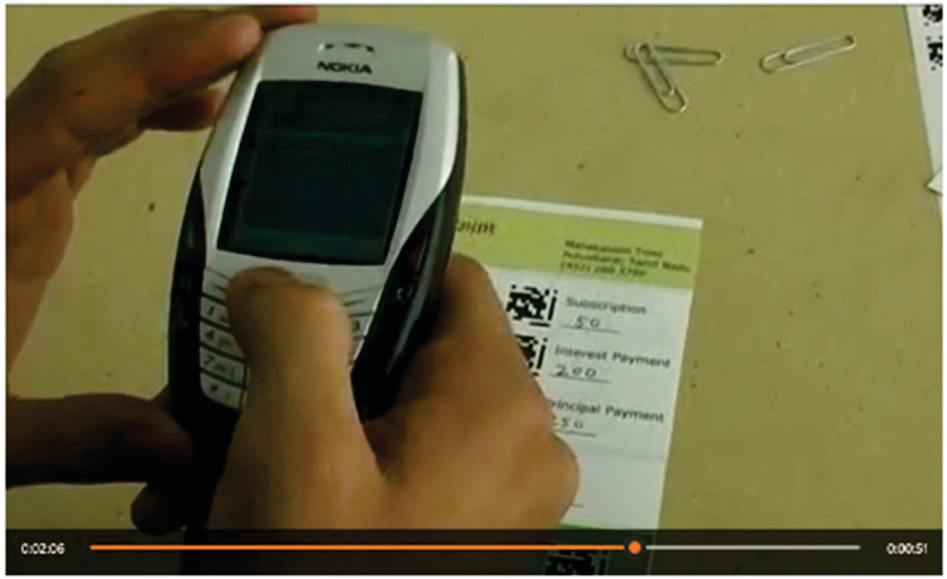
Claims	Prior Art
	CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a

Claims	Prior Art
	simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76–77 The CamBrowser application executes user instructions until it encounters a network or application request. The request can be one of three types: a URL, a phone call, or an MMS message. The interpreter executes the request using the appropriate application, with the local variables as the request content. In the case of a URL, the phone can operate in one of three network modes: [Sidebar] • online, in which the phone executes HTTP requests over an Internet connection. • offline, in which the phone caches requests locally and executes them later when it has connectivity, either via a short-range (Bluetooth) or long-range (Internet) connection. • proxy, in which the phone acts as a proxy interface for a nearby PC, forwarding requests over a Bluetooth connection. The PC's Web browser processes and displays these requests. These network modes let users choose where and how to incur communications costs. If they live in an area where the cellular network isn't good, or if service costs are disproportionately high, they can physically carry the phone to a better connected location. CAM (Pervasive Computing) at 77 The CamBrowser connects to the server via HTTP requests to standard Web applications, which can be written in any scripting language, such as Perl or PHP. The CamBrowser also uploads captured images to the server for a visual record or further upstream processing. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered.

Claims	Prior Art
	CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See also* limitation 1[j]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
9[pre]. A system enabling mobile check deposit auto-capture functionality via video frame processing, the system comprising:	To the extent the preamble is limiting, CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[a] system enabling mobile check deposit auto-capture functionality via video frame processing, the system." For example: *See* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9a] a mobile device of a user, the user's mobile device including a	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the

Claims	Prior Art
camera, a processor, and a memory coupled to the processor, the memory storing a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera, the system being configured to perform operations including:	processor, the memory storing a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera, the system being configured to perform operations including." For example: *See* limitations 1[a], 1[c]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9b] providing Instructions to the user for capturing an image of a check;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "providing Instructions to the user for capturing an image of a check." For example: *See* limitation 1[d]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9c] generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera." For example: *See* limitation 1[e]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9d] overlaying an alignment guide on the live video of the check in the field of view of the camera;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "overlaying an alignment guide on the live video of the check in the field of view of the camera." For example: *See* limitation 1[f]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
[9e] monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion." For example: *See* limitation 1[g]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9f] determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame." For example: *See* limitation 1[h]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9g] when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least one monitoring criterion, providing feedback advising the user that the check has been captured;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least one monitoring criterion, providing feedback advising the user that the check has been captured." For example: *See* limitation 1[i]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9h] using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, the digital image	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, the digital image of the check being the captured video frame or being based on the captured video frame." For example: *See* limitation 1[j].

Claims	Prior Art
of the check being the captured video frame or being based on the captured video frame;	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9i] processing the digital image of the check Including performing optical character recognition and verifying that the digital image of the check Is acceptable for mobile check deposit; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "processing the digital image of the check Including performing optical character recognition and verifying that the digital image of the check Is acceptable for mobile check deposit." For example: *See* limitation 1[l], 1[n]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9j] confirming that the mobile check deposit can go forward after optical character recognition is performed on the check in the digital image, a magnetic ink character recognition (MICR) line on the check in the digital image is recognized, and an amount payable on the check in the digital image is validated; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "confirming that the mobile check deposit can go forward after optical character recognition is performed on the check in the digital image, a magnetic ink character recognition (MICR) line on the check in the digital image is recognized, and an amount payable on the check in the digital image is validated." For example: *See* limitation 1[k]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[9k] a bank computer programmed to update a balance of an account at a financial institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the check is submitted for mobile check deposit in the bank, confirming that the mobile check deposit can go forward, and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a bank computer programmed to update a balance of an account at a financial institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the check is submitted for mobile check deposit in the bank, confirming that the mobile check deposit can go forward, and verifying that the digital image of the check is acceptable for mobile check deposit." For example: *See* limitation 1[o]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
verifying that the digital image of the check is acceptable for mobile check deposit.	
10[pre]. The system of claim 9, wherein the system is configured to perform additional operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 9, wherein the system is configured to perform additional operations including." For example: *See* limitation 2[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[10a] before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion, determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion, determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion." For example: *See* limitation 2[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[10b] after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user." For example: *See* limitation 2[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
11. The system of claim 9, further comprising one or more computers programmed to perform operations comprising the monitoring, the	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 9, further comprising one or more computers programmed to perform operations comprising the monitoring, the determining, or both the monitoring and the determining." For example: *See* limitations 9[a], 9[e], 9[f].

Claims	Prior Art
determining, or both the monitoring and the determining.	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
12. The system of claim 9, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 9, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection." For example: *See* claim 3. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
13. The system of claim 12, wherein the plurality of monitoring criteria Includes criteria based on light brightness, skewing, and warping.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 12, wherein the plurality of monitoring criteria Includes criteria based on light brightness, skewing, and warping." For example: *See* claim 5. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
14. The system of claim 9, wherein the confirming that the mobile check deposit can go forward is performed after the transmitting the digital image of the check from the user's mobile device.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 9, wherein the confirming that the mobile check deposit can go forward is performed after the transmitting the digital image of the check from the user's mobile device." For example: *See* claim 6. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
15. The system of claim 14, wherein the user's mobile device receives an Indication of a result of the processing the digital image of the check before the confirming step.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 14, wherein the user's mobile device receives an Indication of a result of the processing the digital image of the check before the confirming step." For example: *See* limitations 1[n], 9[i], 9[j]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
16. The system of claim 11, wherein the one or more computers include the bank computer.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 11, wherein the one or more computers include the bank computer." For example: *See* limitations 9[a], 9[e], 9f], claim 11. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
17. The system of claim 9, wherein the wireless network is a cellular network.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 9, wherein the wireless network is a cellular network." For example: *See* limitation 9[h]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
18[pre]. A system enabling mobile check deposit auto-capture functionality via video frame processing, the system comprising:	To the extent the preamble is limiting, CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[a] system enabling mobile check deposit auto-capture functionality via video frame processing, the system." For example: *See* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
[18a] a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the processor, the memory storing:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the processor, the memory storing." For example: *See* limitation 1[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18b] a software abstraction layer, the software abstraction layer comprising Instructions that, when executed by the processor, control the camera; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a software abstraction layer, the software abstraction layer comprising Instructions that, when executed by the processor, control the camera." For example: *See* limitation 1[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18c] a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera through the software abstraction layer, the downloaded app comprising Instructions that, when executed by the processor, cause the user's mobile device to perform operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera through the software abstraction layer, the downloaded app comprising Instructions that, when executed by the processor, cause the user's mobile device to perform operations including." For example: *See* limitation 1[c]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18d] providing Instructions to the user for capturing an image of a check;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "providing Instructions to the user for capturing an image of a check." For example: *See* limitation 1[d].

Claims	Prior Art
	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18e] generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera through the software abstraction layer;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera through the software abstraction layer." For example: *See* limitation 1[e]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18f] monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion." For example: *See* limitation 1[g]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18g] determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame." For example: *See* limitation 1[h]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18h] when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least one monitoring criterion, providing feedback advising the user that the check has been captured." For example:

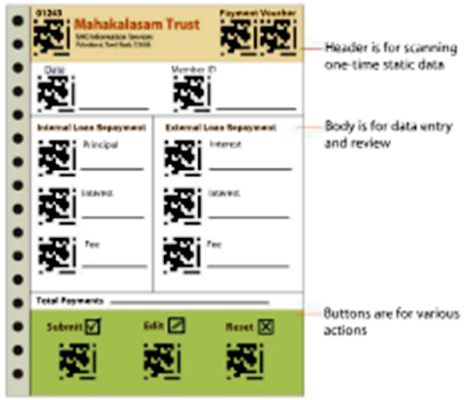
Claims	Prior Art
one monitoring criterion, providing feedback advising the user that the check has been captured;	*See* limitation 1[i]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18i] using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, wherein the digital image is the captured video frame or is based on the captured video frame; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, wherein the digital image is the captured video frame or is based on the captured video frame." For example: *See* limitation 1[j]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18j] confirming that the mobile check deposit can go forward after optical character recognition is performed on the check, a magnetic ink character recognition (MICR) line on the check is recognized, and an amount payable on the check is validated; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "confirming that the mobile check deposit can go forward after optical character recognition is performed on the check, a magnetic ink character recognition (MICR) line on the check is recognized, and an amount payable on the check is validated." For example: *See* limitation 1[k]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[18k] a computer comprising a processor coupled to a memory storing Instructions that, when executed by the processor, cause the computer to update a balance of an account at a financial institution of the user to reflect an amount of funds associated with the check submitted for mobile	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a computer comprising a processor coupled to a memory storing Instructions that, when executed by the processor, cause the computer to update a balance of an account at a financial institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the submitting the check for mobile check deposit and the confirming that the mobile check deposit can go forward." For example: *See* limitation 1[o].

Claims	Prior Art
check deposit by the user's mobile device, after the submitting the check for mobile check deposit and the confirming that the mobile check deposit can go forward.	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
19[pre]. The system of claim 18, wherein the downloaded app causes the user's mobile device to perform additional operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 18, wherein the downloaded app causes the user's mobile device to perform additional operations including." For example: *See* limitation 2[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[19a] before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion, determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion, determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion." For example: *See* limitation 2[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[19b] after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user." For example: *See* limitation 2[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
20. The system of claim 18, wherein the at least one monitoring	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 18, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or

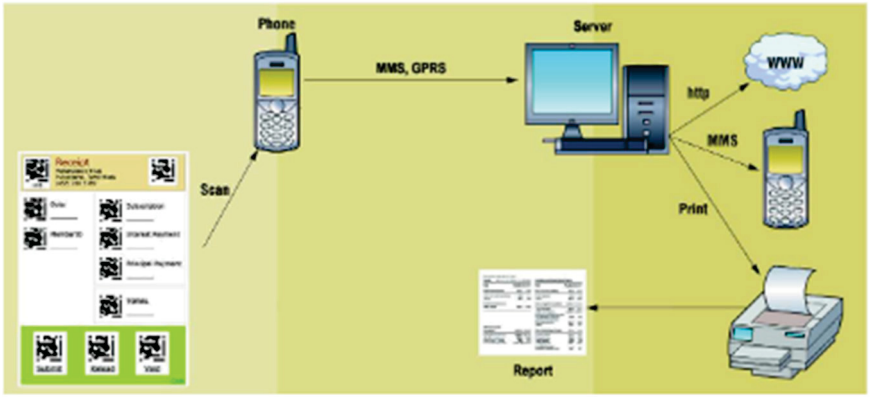
Claims	Prior Art
criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection.	more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection." For example: *See* claim 3. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
21. The system of claim 20, wherein the plurality of monitoring criteria being based on magnetic ink character recognition (MICR) line detection and one or both of corner detection and edge detection.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 20, wherein the plurality of monitoring criteria being based on magnetic ink character recognition (MICR) line detection and one or both of corner detection and edge detection." For example: CAM (Parikh 2005a) at Abstract During our work with microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera-equipped mobile phone is used as an image capture and data entry device. Our system is able to process paper forms that contain CAMShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized five years of design experience into a usable and locally harmonious system that has the potential to impact billions of people. CAM (Parikh 2005a) at 1 In this paper we introduce mobile document processing as an accessible and locally harmonious way to provide information services to the developing world [2]. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a narrative document-embedded programming language that can allow developers to add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents.

Claims	Prior Art
	CAM (Parikh 2005a) at 1 2.1. CAMBrowser The CAM client application is called the CAMBrowser, and has currently been implemented for Series 60 camera-equipped mobile phones. CAMBrowser is designed to process specially designed CAMForm documents. CAMForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [3]. Visual codes serve as references to interactive content embedded within CAMForms. CAM interaction consists of two primitives. The user can either scan codes from low-resolution images taken by the viewfinder in real time, or click codes by taking a high-resolution image using the joystick button. Our focus is on designing a simple and intuitive interaction model, with well-defined affordances between actions. CAM (Parikh 2005a) at 2 CAMShell also provides an API for accessing the mobile phone's functionality - including functions to access the phone's user interface, networking and telephony. Each callback function can contain any number of sequential actions, some of which may be only executed conditionally. Conditionals are useful when performing form validation. An example callback function is given below. [Listing] CAM (Parikh 2005a) at 2 3. Applications We believe that CAM is suitable for a rich set of applications in rural India and elsewhere. We have met banks, private companies, NGOs and governmental agencies that have expressed interest in using the CAM framework. Below are some of the applications that we are currently exploring: - Sales and inventory tracking - Health data collection - Transaction processing - Mobile cash register - Accounting for microfinance groups - Paper / digital postcards CAM (Parikh 2005b) at Abstract During our work with community microfinance groups in rural India, we found that paper plays a crucial role in many local information practices. However, paper-based record-keeping can be inefficient, so we need to link paper with the flexibility of modern information tools. A mobile phone can be the perfect bridging device. Here we present the CAM mobile document processing system, in which a camera phone

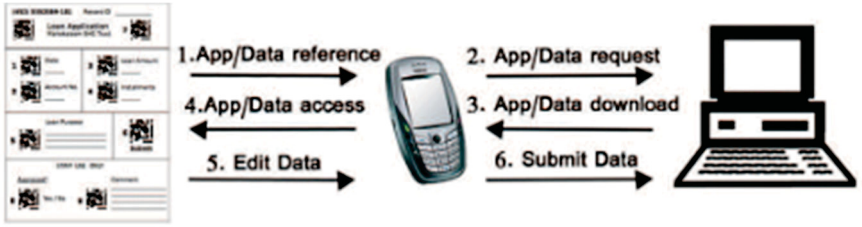
Claims	Prior Art
	is used as an image capture and data entry device. Our system is able to process paper forms containing CamShell programs - embedded instructions that are decoded from an electronic image. By combining 1) paper, 2) audio, 3) numeric data entry, 4) narrative scripted execution and 5) asynchronous connectivity, we have synthesized our experience into a system that we believe is well-suited for an important set of users and applications in the developing world. CAM (Parikh 2005b) at 2 In this paper we introduce mobile document processing as a way to provide information services to the developing world. In our system a camera-equipped mobile phone is used for image capture and data entry. Document interaction is specified using CamShell - a document-embedded programming language that allows developersto add interactive audio, data entry, validation, processing, and networking instructions to otherwise inert paper documents. CAM (Parikh 2005b) at 2 CamBrowser The CAM client application is called the CamBrowser, and has currently been implemented for Nokia Series 60 camera phones. CamBrowser is designed to process specially designed CamForm documents (see Figure 2). CamForms contain visual codes - two-dimensional data glyphs containing up to 76-bits of data that can be decoded from a camera image [18]. Visual codes serve as references to the interactive content embedded within CamForms. Other researchers have presented a rich set of interactions made possible by tilting, rotating and translating the phone's camera relative to an individual visual code [19]. However it also has been observed that most of these interactions are not accessible to novice users [22]. In CAM we have simplified the interaction to two primitives. The user can scan codes from low-resolution images taken in real time, or click codes by taking a high-resolution image using the joystick button. Our focus was on designing a simple and intuitive interaction model with well-defined affordances between actions.

Claims	Prior Art
	CAM (Parikh 2005b) at 3  Figure 2: An example CamForm, showing the three sections the user can interact with. CAM (Parikh 2005b) at 3 Other researchers have commented on the importance of paper in workplace settings [20, 12], and sought to improve coordination between paper and digital media [6, 4, 8, 3]. Companies have extended these efforts to develop commercial document processing systems that recognize structure and text from scanned paper forms and integrate with back end data sources [5]. None of these systems have used a mobile phone as the primary image capture and data entry device, and none provide a general purpose networked programming environment for specifying user interaction and client processing. CAM (Parikh 2007) at Abstract Populations in the rural developing world have just as many, if not more, information needs as people living in more affluent areas. But their constraints — intermittent power, intermittent connectivity, limited education, literacy and capital — make first-world approaches to accessing information systems inapplicable. Mobile phones are on the cusp of spurring an information revolution in such regions. Long battery life, wireless connectivity, solid-state memory, low price and immediate utility make this device better suited to rural conditions than a PC. However, current software on mobile phones makes them hard to use and to program. In this dissertation, I present the design, implementation and evaluation of CAM — a mobile application framework designed to address the information needs of the rural developing world. Beginning with a two-month participatory design study, including users with varying levels of education, I propose some general guidelines for user interface and system design in this context.

Claims	Prior Art
	Motivated by these guidelines, I present CAM. CAM applications are accessed by capturing barcodes on paper forms using the mobile phone camera, or entering numeric strings with the keypad. Supporting one-step navigation, direct linkage to paper practices and offline multi-media interaction, CAM is uniquely adapted to rural device, user and infrastructure constraints. To demonstrate the usability of this framework, I implement and evaluate several distinct CAM-based applications (one of which has already been commercially deployed). I also provide preliminary motivation for fourteen other applications that could be implemented with the same, or similar, approach. CAM (Parikh 2007) at 53–54 For barcode recognition and generation, I am using the open source toolkit developed by Rohs and Gfeller [128]. This includes a 2D barcode recognition library for Nokia Series 60 phones. A user can capture a barcode by taking a picture of it using the mobile phone [Figure 4.3] camera. Each barcode contains 83 bits of data, 7 of which are being used for error correction (leaving 76 bits of data capacity). Several barcodes can be detected from a single camera image. CAM (Parikh 2007) at 58–59 After data has been entered, the user can review values before submitting them to the server. By focusing the mobile phone camera on the appropriate barcode, the currently entered data is displayed on the screen, usually next to the value entered on paper (see figure 4.4). This allows both the record keeper and the SHG member to confirm that data was entered properly (mimicking the transparency of the earlier paper-based process). By focusing on the loan purpose field, the recorded audio clip is played. If a value is incorrect, the user can capture an image of just that barcode, displaying a prompt to edit the value. CAM (Parikh 2007) at 129 9.1.3 CAM: A Framework for Distributed Data Collection using Mobile Phones Based on this understanding, I designed and implemented CAM, a software toolkit for distributed data collection using mobile phones. Instead of using menus, CAM applications are navigated with printed barcodes and numbers. The rich indexing capabilities of embedded physical tokens provides fine-grained contextual access to applications and content. Rather than optimize spatial layout of pages, CAM displays prompts in sequence to reduce screen requirements and user decisions. Instead of being written declaratively, CAM applications are scripted, generating sequences of actions. To support offline use, the

Claims	Prior Art
	entire application is downloaded at once, rather than page by page. CAM also provides multimedia input and output capabilities, providing rich interaction modes for rural users. CAM (Parikh 2007) at 51 (Figure 4.1)  Figure 4.1: Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper- based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways. CAM (Parikh 2007) at 54 (Figure 4.3)  Figure 4.3: The CamBrowser application. The current value for the field is displayed beside the visual code.

Claims	Prior Art
	CAM (Parikh 2007) at 57 (Figure 4.4)  Figure 4.4: On the left, reviewing an entered value by focusing on a barcode. On the right, clicking brings up a prompt for editing the value. An audio prompt is also played. CAM (Parikh 2007) at 58 (Table 4.2) Table 4.2: Functions provided by the CAM API. All prompts and messages can be accompanied by arbitrary audio and/or images. *input_int*, *input_date*, *input_password*, *input_pin*, *input_phone_number*, *input_time*, *input_number*, *input_text* — Prompts asking the user to enter different kinds of values *record_audio*, *take_picture* — Ask the user to record an audio clip, or take a picture of a form or other object *message_note*, *conf_note* — Send a message to the user, or display a confirmation dialog *get*, *put* — Retrieve or submit application data to the server using using http, sms or mms *encrypt*, *decrypt* — Encrypt or decrypt application data with a specified key *sms*, *mms*, *email* — Send arbitrary messages *phonecall*, *browser* — Make a phone call or launch a web browser *log* — Write a string to the application log

Claims	Prior Art
	CAM (Parikh 2007) at 60 (Figure 4.5)  Figure 4.5: An example set of steps in accessing and using a CAM application. Except for steps 4-5, there can be arbitrary time and distance between each step, supporting an offline, asynchronous interaction model appropriate for rural areas. CAM (Pervasive Computing) at 74–75 The CAM framework We've developed an information services architecture that uses a smart phone equipped with a built-in digital camera to process augmented paper documents. CAM, so called because the phone's camera plays a key role in the user interface, is a three-tiered, document-based architecture for providing remote rural information services. The user tier comprises a set of paper forms [Figure 1] and artifacts that people use to record information, perform queries, and conduct transactions. The server tier is a standard Web application server, which can reside locally, in a nearby town, or virtually in the Internet ether. The mobile phone acts as a roving middleware, playing the role of scanner, user interface, network, cache, and preprocessor. Figure 1 presents the CAM architecture's overall structure. The CAM framework comprises four components. Users initially interact with CamForms, which are augmented paper applications. The CamShell scripting language is used to embed processing instructions in these forms, which the CamBrowsermobile phone application scans and processes. CamBrowser connects to a local or remote CamServer, which receives and organizes CamForm submissions and supports additional data entry and document processing.

Claims	Prior Art
	CAM (Pervasive Computing) at 75

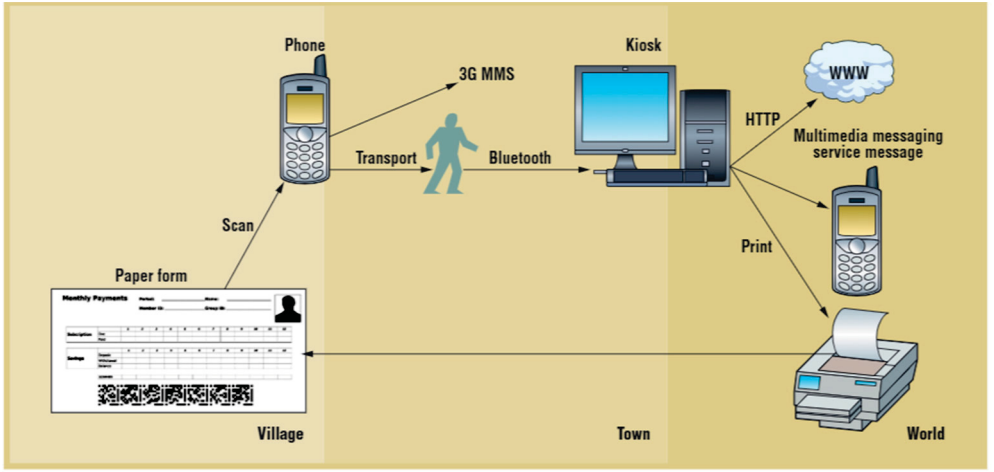
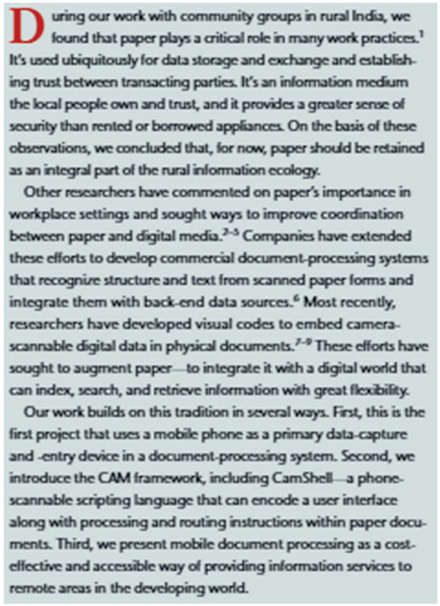


Figure 1. Three-tiered CAM framework. CAM provides the appropriate information medium for each context. Rural users transfer paper-based information to roving smart-phone middleware, which uploads the data to remote storage for distribution in various ways.

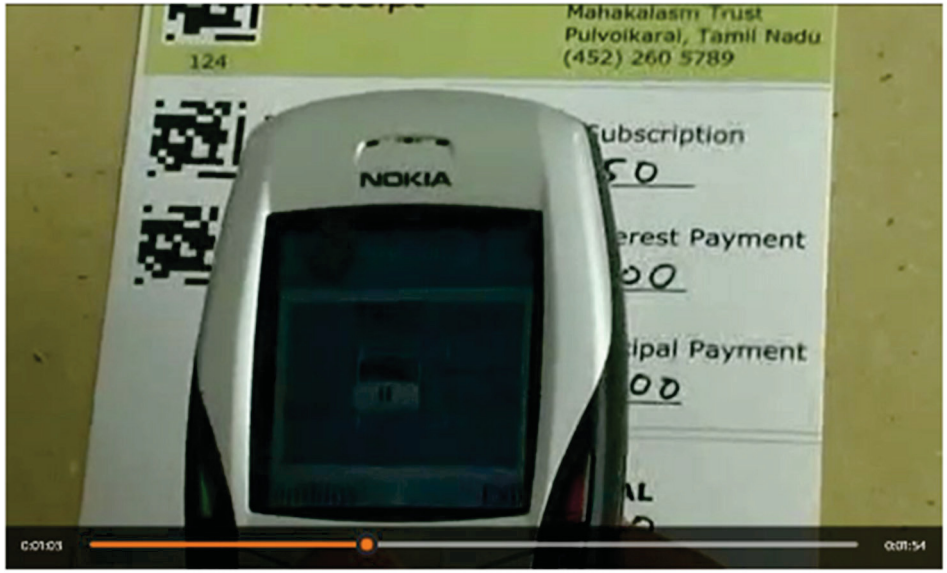
CAM (Pervasive Computing) at 76



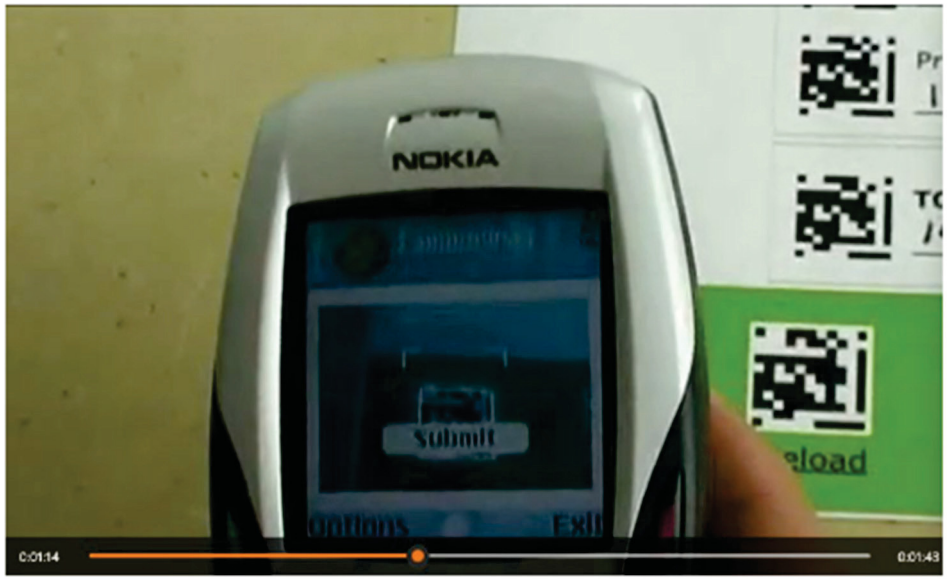
Claims	Prior Art
	CAM (Pervasive Computing) at 75 However, paper is notoriously bad for other uses. You can't easily search, index, or archive it. Using it to perform calculations and create consolidated reports is difficult. In our work with community microfinance groups around the world, we found that data consolidation and reporting using manual, ledger-based systems is virtually impossible. We needed a way to digitally capture information from paper and let computing systems do the things they're good at. So, we had to find a way to link the physical world of paper with the digital world of information. CAM (Pervasive Computing) at 75–76 CamShell We use CamShell to embed processing instructions in CamForm documents. A camera-equipped mobile phone running the CamBrowser application interprets these instructions, which use the phone's user interface and networking libraries to process and route the document. We embed data in our forms using the 2D visual-code-recognition library that Michael Rohs and Beat Gfeller developed.[1] This package lets the phone's camera scan 83-bit 2D codes (including seven bits of error correction). Because of this substrate size, CamShell is composed of fixed-length, 76-bit instructions. When mobile phones with higher-resolution cameras are more widely available, the data-carrying capacity of this substrate should increase accordingly, and we'll be able to make CamShell more expressive. In the meantime, we've [Figure 2] designed a compact but powerful programming language that's sufficient for many uses. Each instruction encodes its sequence number in the first four bits. Code recognition isn't exact, so the phone might not recognize all the codes embedded on a form in one scan. In a sense, it must reassemble the program from the individual codes after scanning.

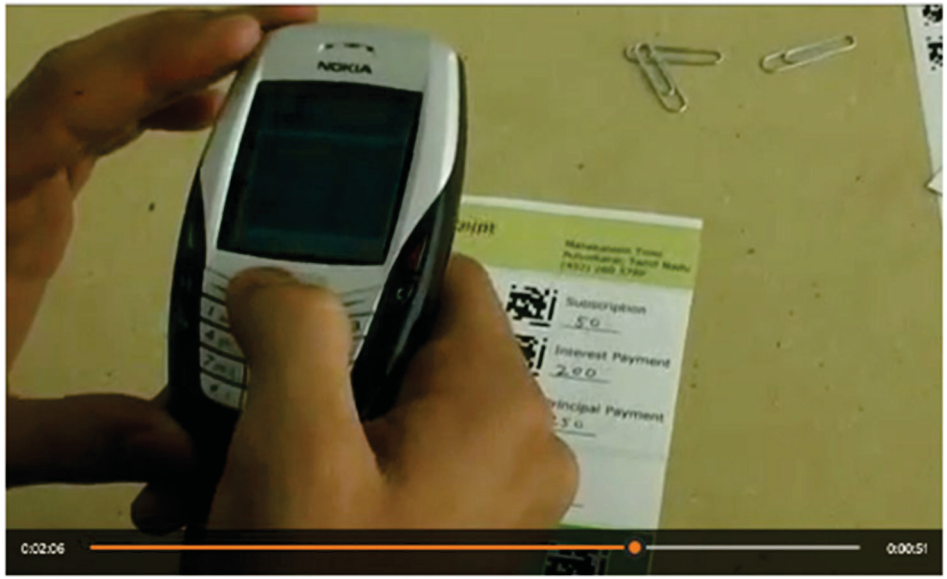
Claims	Prior Art
	CAM (Pervasive Computing) at 77  During our work with community groups in rural India, we found that paper plays a critical role in many work practices.[1] It's used ubiquitously for data storage and exchange and establishing trust between transacting parties. It's an information medium the local people own and trust, and it provides a greater sense of security than rented or borrowed appliances. On the basis of these observations, we concluded that, for now, paper should be retained as an integral part of the rural information ecology. Other researchers have commented on paper's importance in workplace settings and sought ways to improve coordination between paper and digital media.[2–5] Companies have extended these efforts to develop commercial document-processing systems that recognize structure and text from scanned paper forms and integrate them with back-end data sources.[6] Most recently, researchers have developed visual codes to embed camera-scannable digital data in physical documents.[7–9] These efforts have sought to augment paper—to integrate it with a digital world that can index, search, and retrieve information with great flexibility. Our work builds on this tradition in several ways. First, this is the first project that uses a mobile phone as a primary data-capture and -entry device in a document-processing system. Second, we introduce the CAM framework, including CamShell—a phone-scannable scripting language that can encode a user interface along with processing and routing instructions within paper documents. Third, we present mobile document processing as a cost-effective and accessible way of providing information services to remote areas in the developing world. CAM (Pervasive Computing) at 76 CamBrowser The CamBrowser application's primary purpose is to capture a visual image of the form and decode and interpret the embedded CamShell program, which executes the user interface that collects information from the user. CamBrowser executes instructions in sequence, starting with instruction zero. Internally, a simple virtual machine keeps track of the current program counter and local variables. You can scan codes either by pointing the camera at the CamForm (scan mode) or by explicitly taking a picture (click mode). We've found the former to be more suitable for quick, directed user interaction, while the latter is more effective for capturing an entire CamShell program and entering large amounts of data—for example, in a loop. CAM (Pervasive Computing) at 76 We've avoided using optical character recognition because of its high error rates and associated usability concerns. Users must manually transcribe all the CamForm data using the phone's alphanumeric keypad (see Figure 2). In the future, we're considering adding simple recognition features, such as automatic population of checkboxes or selection ovals, and delegating some recognition tasks to more powerful

Claims	Prior Art
	upstream servers. CAM (Pervasive Computing) at 77 Users can view the response document on the phone's screen, on a nearby PC, or printed from a connected printer. Response screens can include further CamShell scripts, which users can scan from the screen or from a printed copy. The scripts can include scannable links to further content or can act like wizards that construct new CamForms based on prior input. CAM (Pervasive Computing) at 77–78 We're refining our model using XML to provide generalized document services. In the new model, users scan and then package CamForms into an XML file containing the image, the data they entered, and the CamShell program. This package is sent over the network, where the CamServer receives it. The CamServer organizes CamForm submissions and supports further browser-based data entry and document processing. This lets phone users delegate certain tasks upstream, allowing more arduous data entry or recognition tasks [Figure 3] to be done on a PC. In this way, the system provides flexibility in the trade-off between recognition and local and remote data entry. CAM (Pervasive Computing) at 78–79 First, the system can transmit the form's scanned image to the CamServer to provide an online digital representation of the form as the user entered it. Second, users can retain the physical CamForms as records of their transactions. Third, the service provider can send a printed summary from the CamServer to the user acknowledging receipt of the transaction and its content. [Figure 4] Finally, we're implementing audio feedback using text-to-speech to let users hear the data as it's being entered. CAM (Pervasive Computing) at 80 We're working on SHG-Checkbook as a way to enable secure, human-mediated electronic transactions in remote rural areas. SHGs would conduct transactions with a registered banking proxy equipped with a mobile phone running CamBrowser. These proxies could be traders, merchants, or other locally known persons. Later, the bank or NGO would settle client accounts with the proxy in exchange for a predetermined service fee. When processing an SHG-check, the proxy banker first scans the member's CamID, as shown in Figure 4. This populates the member ID and group ID fields and provides the user's private key. The proxy banker

Claims	Prior Art
	then scans the check, which provides the remaining information for the transaction. The banker thus combines the CamID and SHG-check documents into a single CAM request. SHG-checks carry both the bank's public key and a one-time shared key. The system encrypts the request using the bank's public key and signs it with both the user's private key and the one-time shared key, ensuring the message's confidentiality and authenticity. Using SHG-Checkbook, SHGs can write checks to members to disburse loans and use deposit slips to keep unused funds in interest-bearing bank accounts. This electronic currency would let SHGs maintain local liquidity without wasteful and dangerous physical handling of money. CAM (Video) at 0:57–1:04 

Claims	Prior Art
	CAM (Video) at 1:30–1:51 

Claims	Prior Art
	CAM (Video) at 1:11–1:14 

Claims	Prior Art
	CAM (Video) at 2:02–2:22 

Claims	Prior Art
	CAM (Video) at 1:44–1:53  *See* claims 3, 4, 20. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
22. The system of claim 20, wherein the plurality of monitoring criteria Includes criteria based on light brightness, skewing, and warping.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 20, wherein the plurality of monitoring criteria Includes criteria based on light brightness, skewing, and warping." For example: *See* claim 5. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
23. The system of claim 18, wherein the confirming that the mobile check deposit can go forward is performed after the transmitting the digital image of the check from the user's mobile device.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 18, wherein the confirming that the mobile check deposit can go forward is performed after the transmitting the digital image of the check from the user's mobile device." For example: *See* claim 6. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
24. The system of claim 18, wherein the digital image is a modified version of the captured video frame.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 18, wherein the digital image is a modified version of the captured video frame." For example: *See* claim 7. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
25. The system of claim 18, wherein the wireless network is a cellular network.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 18, wherein the wireless network is a cellular network." For example: *See* claim 8. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
26[pre]. A system enabling mobile check deposit auto-capture functionality via video frame processing, the system comprising:	To the extent the preamble is limiting, CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[a] system enabling mobile check deposit auto-capture functionality via video frame processing, the system." For example: *See* limitation 1[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
[26a] a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the processor, the memory storing:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a mobile device of a user, the user's mobile device including a camera, a processor, and a memory coupled to the processor, the memory storing." For example: *See* limitation 1[a]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26b] a software abstraction layer, the software abstraction layer comprising Instructions that, when executed by the processor, control the camera; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a software abstraction layer, the software abstraction layer comprising Instructions that, when executed by the processor, control the camera." For example: *See* limitation 1[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26c] a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera through the software abstraction layer, the downloaded app comprising Instructions that, when executed by the processor, cause the user's mobile device to perform operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a downloaded app, the downloaded app associated with a bank to control check deposit by controlling the camera through the software abstraction layer, the downloaded app comprising Instructions that, when executed by the processor, cause the user's mobile device to perform operations including." For example: *See* limitation 1[c]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26d] providing Instructions to the user for capturing a check;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "providing Instructions to the user for capturing a check." For example: *See* limitation 1[d].

Claims	Prior Art
	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26e] generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera through the software abstraction layer;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "generating a live video of the check in a field of view of the camera, the downloaded app causing the live video to be generated by controlling the camera through the software abstraction layer." For example: *See* limitation 1[e]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26f] monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "monitoring the live video of the check in the field of view of the camera with respect to at least one monitoring criterion." For example: *See* limitation 1[g]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26g] determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "determining that a video frame of the live video of the check satisfies the at least one monitoring criterion and then automatically capturing the video frame." For example: *See* limitation 1[h]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26h] when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "when the video frame of the live video of the check in the field of view of the camera is determined to satisfy the at least one monitoring criterion, providing feedback advising the user that the check has been captured." For example:

Claims	Prior Art
one monitoring criterion, providing feedback advising the user that the check has been captured;	*See* limitation 1[i]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26i] using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, wherein the digital image is the captured video frame or is based on the captured video frame; and	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "using a wireless network, transmitting a digital image of the check from the user's mobile device and submitting the check for mobile check deposit in the bank after automatically capturing the video frame, wherein the digital image is the captured video frame or is based on the captured video frame." For example: *See* limitation 1[j]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26j] confirming that the mobile check deposit can go forward after optical character recognition is performed on the check, a magnetic ink character recognition (MICR) line on the check is recognized, and an amount payable on the check is validated;	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "confirming that the mobile check deposit can go forward after optical character recognition is performed on the check, a magnetic ink character recognition (MICR) line on the check is recognized, and an amount payable on the check is validated." For example: *See* limitation 18[j]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26k] the system configured to perform image processing operations on an image of the check, the image processing operations including performing optical character recognition and verifying that the image of the check is acceptable for mobile check deposit, wherein the image	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "the system configured to perform image processing operations on an image of the check, the image processing operations including performing optical character recognition and verifying that the image of the check is acceptable for mobile check deposit, wherein the image of the check is the digital image transmitted by the user's mobile device or is based on the digital image transmitted by the user's mobile device." For example: *See* limitations 1[j], 1[n].

Claims	Prior Art
of the check is the digital image transmitted by the user's mobile device or is based on the digital image transmitted by the user's mobile device; and	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[26l] a computer programmed to update a balance of an account at a financial Institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the submitting the check for mobile check deposit in the bank, the confirming that the mobile check deposit can go forward, and the verifying that the image of the check is acceptable for mobile check deposit.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "a computer programmed to update a balance of an account at a financial Institution of the user to reflect an amount of funds associated with the check submitted for mobile check deposit by the user's mobile device, after the submitting the check for mobile check deposit in the bank, the confirming that the mobile check deposit can go forward, and the verifying that the image of the check is acceptable for mobile check deposit." For example: *See* limitation 1[o]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
27[pre]. The system of claim 26, wherein the downloaded app causes the user's mobile device to perform additional operations including:	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 26, wherein the downloaded app causes the user's mobile device to perform additional operations including." For example: *See* limitation 2[pre]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[27a] before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion, determining that a different video frame of the	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "before determining that the video frame of the live video of the check satisfies the at least one monitoring criterion, determining that a different video frame of the live video of the check does not satisfy the at least one monitoring criterion." For example: *See* limitation 2[a].

Claims	Prior Art
live video of the check does not satisfy the at least one monitoring criterion; and	This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
[27b] after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "after determining that the different video frame of the live video of the check does not satisfy the at least one monitoring criterion, providing feedback to the user." For example: *See* limitation 2[b]. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
28. The system of claim 26, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 26, wherein the at least one monitoring criterion is a plurality of monitoring criteria based on two or more selected from a group consisting of light contrast, light brightness, check positioning, check dimensions, character spacing, skewing, warping, corner detection, edge detection, and magnetic ink character recognition (MICR) line detection." For example: *See* claim 3. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.
29. The system of claim 28, wherein the plurality of monitoring criteria Includes a criterion based on magnetic ink character recognition (MICR) line detection.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 28, wherein the plurality of monitoring criteria Includes a criterion based on magnetic ink character recognition (MICR) line detection." For example: *See* claim 4. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.

Claims	Prior Art
30. The system of claim 26, wherein the wireless network is a cellular network.	CAM discloses and/or renders obvious, at least under USAA's apparent infringement theories, "[t]he system of claim 26, wherein the wireless network is a cellular network." For example: *See* claim 8. This limitation also would have been obvious to a person of ordinary skill in the art based on (1) CAM alone; (2) CAM in combination with the general knowledge of a person of ordinary skill in the art; and/or (3) Regions' other invalidity charts and cover pleading.