ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/221073651

The vSLAM Algorithm for Robust Localization and Mapping.

Conference Paper · January 2005

DOI: 10.1109/ROBOT.2005.1570091 · Source: DBLP

CITATIONS
318

READS
14,427

6 authors, including:

Niklas Karlsson
Amazon
46 PUBLICATIONS **837** CITATIONS

SEE PROFILE

Jim Ostrowski
Blue River Technology
101 PUBLICATIONS **9,005** CITATIONS

SEE PROFILE

Luis Goncalves
California Institute of Technology
37 PUBLICATIONS **1,346** CITATIONS

SEE PROFILE

Mario Munich
Embodied
57 PUBLICATIONS **1,485** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Visual SLAM View project

Project Control and Optimization in Online Advertising View project

All content following this page was uploaded by Luis Goncalves on 01 June 2014.

The user has requested enhancement of the downloaded file.

Published in Proc. of Int. Conf. on Robotics and Automation (ICRA) 2005

The vSLAM Algorithm for Robust Localization and Mapping

Niklas Karlsson, Enrico Di Bernardo, Jim Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E. Munich
Evolution Robotics, Inc.
Pasadena, California, USA
Email: {niklas,enrico,jim,luis,paolo,mario}@evolution.com

***Abstract*— This paper presents the *Visual Simultaneous Localization and Mapping* (vSLAM™) algorithm, a novel algorithm for *simultaneous localization and mapping* (SLAM). The algorithm is vision- and odometry-based, and enables low-cost navigation in cluttered and populated environments. No initial map is required, and it satisfactorily handles dynamic changes in the environment, for example, lighting changes, moving objects and/or people. Typically, vSLAM recovers quickly from dramatic disturbances, such as "kidnapping".**

Index Terms - SLAM, vision, Particle filter, Kalman filter, Mixed proposal distribution.

I. Introduction

Simultaneous localization and mapping (SLAM) is one of the most fundamental, yet most challenging problems in mobile robotics. To achieve full autonomy a robot must possess the ability to explore its environment without user-intervention, build a reliable map, and localize itself in the map. In particular, if *global positioning sensor* (GPS) data and external beacons are unavailable, the robot must somehow, by itself, determine what are appropriate reference points, on which to build a map.

Only within the last 4-8 years has the research community made significant progress towards solutions to the SLAM problem. The progress is enabled by the availability of better sensors, as well as, better algorithms. In particular, the SICK™ *Laser Range Finder* (LRF) was a milestone for autonomous robotics and navigation, and the discovery of probabilistic robotics [9] was a breakthrough.

Initially, the most promising algorithm for solving the SLAM problem was based on *Expectation Maximization* (EM) techniques [12]. The idea behind EM algorithms is to start with an initial guess of robot pose, followed by computing the most likely map. Next, the estimate of where the robot really is located is improved – based on the computed map. Using the improved estimate of robot pose, the previously computed map is made better, and so forth. Unfortunately, since the EM algorithm is not recursive, it is in most practical situations useless. The excellent survey [11] describes other proposed SLAM solutions, as well as, the state-of-art of SLAM in year 2002.

The recently most promising approach to SLAM is published in [7]. It is there realized that if the localization and mapping problem is described as a problem of path estimation rather than pose estimation, it can be separated via a clever factorization.

The SICK LRF provides high resolution 2D distance scans, which makes it possible to reliably detect and localize corners and other edge features in the environment. However, there are at least three problems in using the SICK LRF: 1) It is difficult to reliably associate detected features with previously detected features; 2) the total number of features to maintain in a map is large, potentially causing a prohibitive computational overhead; and 3) it is difficult to extract and recognize corners and edge features in cluttered and highly populated environments.

One way to address the above problems is to use a camera and computer vision based techniques to select appropriate reference points. This was until recently prohibitive because of the price of the camera, the big computational requirement, and the lack of appropriate algorithms. Today, however, cameras have become a commodity and, thanks to Moore's law, high-performing computers are available at low cost. Also, very powerful methods in computer vision [6] and probabilistic robotics now offer the necessary tools for vision-based SLAM algorithms.

An early attempt of using vision in navigation of mobile robots was presented in [13]. However, this attempt addressed only localization, and not mapping, hence, does not qualify as a SLAM algorithm.

The vSLAM algorithm, presented in this paper, utilizes the powerful object recognition algorithm in [6], and the factorization in [7] to build and maintain a map of virtually unique visual landmarks. By also using a localization scheme with a particle filter [2] and an adaptive mixed proposal distribution [3], [5], vSLAM offers a breakthrough SLAM algorithm that allows navigation with good accuracy in various real-world environments. The adaptive mixed proposal distribution also enables vSLAM to recover from "kidnapping" scenarios; i.e., situations where the robot is lifted up and moved without being notified.

The paper is organized as follows. Next section states the problem solved by vSLAM. Section III first gives an overview of the vSLAM system, then each sub-system is described separately. While the system contains a visual front-end, the focus in this paper is on the SLAM subsystem. The front-end is described in a separate paper. Section IV consists of two examples that illustrate the performance and some of the characteristics of vSLAM. Finally, in Section V we draw some conclusions and make some final remarks.

II. Problem Formulation

Consider a mobile robot that must localize itself in a previously unknown environment. The objective is to chose a sensor configuration and an algorithm to process the sensor data, which accurately and robustly accomplish the localization in real-world environments. Assume it is not accepted to alter the environment by installing beacons or other external equipment. That is, the design choices only apply to the robot itself, and the robot must determine its location based only on data collected by the selected on-board sensors. Also, realize that since the environment is unknown, the robots must also map the environment.

Now, suppose the mobile robot is equipped with an odometry sensor providing dead reckoning data, and a single camera providing images of the environment. The odometry sensor may consist of standard wheel encoders attached to the driving wheels of a differential drive system.

The objective of the vSLAM system is to fuse image and odometry data in a way that enables robust map-building and localization. Being "robust" is key since the sensor data acquired from the mobile robot (odometry and images) contains plenty of difficult-to-model noise.

The odometry data is incremental, and therefore it will accumulate error over time. Understand first of all that the odometry sensor can never be perfectly calibrated. But, since the robot may slip or may be lifted and moved, the odometry is also exposed to discrete events of dramatic errors.

The image data is challenging to process because of the existence of image blur, occlusions, limited image resolution, imperfect camera calibration, variable lighting conditions, limited processing power, etc. For example, if the robot is operating in a highly populated environment, the field of view of the camera is frequently filled by non-stationary objects, such as moving people, that can not be used as reference points in a map. Also, the lighting conditions may change from very dark to light, even saturating the captured image.

The remainder of this article describes the vSLAM algorithm, which is a system that addresses the above challenges and provides an accurate and robust way of doing localization and mapping in real-world environments.

III. The vSLAM System

From a system point of view, vSLAM can be described as in Figure 1. The inputs to the system are odometry and images, and the output is the robot pose and an abstract vSLAM map.

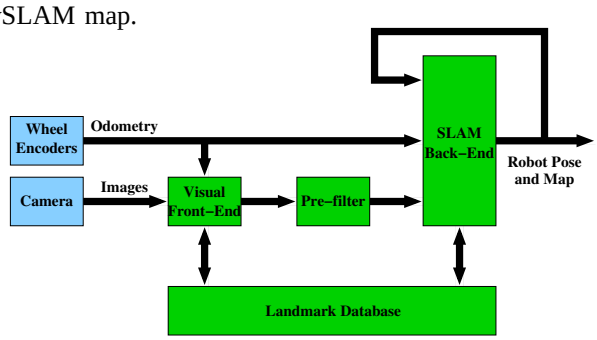

Fig. 1. Block diagram of vSLAM.

The basic component of the vSLAM map are the visual landmarks that are "created" along the path of the robot. A *visual landmark* is a collection of unique features extracted from an image. Each landmark is associated with a *landmark pose*, defined as the robot pose (x, y, and θ) when the landmark was created.

In the first module of vSLAM, the *Visual Front-end*, the images are processed and compared with previously created landmarks. If a recognition is made, an estimate is computed on where the robot is located relative to where it was located when the landmark was created. Such a relative estimate is throughout the paper called *visual measurement*. If a recognition is not made, the module attempts to create a new visual landmark. See more in Section III-A on landmarks, landmark creation, etc. If a landmark is created, the image corresponding to the landmark is saved for later recognition (it is added to the *Landmark Database*).

If a visual landmark was recognized in the Visual Front-end and a visual measurement is computed, then the second module, the *Pre-filter*, assess the reliability of the measurement. If the measurement is considered unreliable, it is rejected and thrown away. It will thereafter not be used in any further processing. However, if it is accepted, it is used as an input to the SLAM module.

The SLAM module is a feedback system taking odometry data and relative pose measurements as inputs, to be fused and compared to a current map. First the robot pose is estimated based on the most up-to-date map and the two inputs. Then, the map is updated to reflect the new information.

A. Visual Frontend

The Visual Front-end is described in detail in [4], and we shall here consider it merely a black box.

The inputs to the Visual Front-end are images with time stamps plus time-stamped odometry. The output depends on whether

1) The front-end created a new landmark
2) The front-end computed a visual measurement
3) The front-end neither computed a visual measurement, nor created a landmark

If a landmark was created, a landmark ID is provided to be used as a reference for future landmark recognitions. Note that no landmark pose is provided by the front-end. The location of individual landmarks in a global map is estimated in the SLAM module.

If a visual measurement was computed, the output is a relative pose $y = [\Delta_x, \Delta_y, \Delta_\theta]^T$ and a landmark ID n. The output indicates where landmark n is estimated to be located relative the current robot pose. This relative pose is described in the standard, local robot coordinate system. The origin of this coordinate system is at the floor level, directly below the drive wheels' wheelbase. The positive x-axis extends forward in the direction the robot is facing. The positive y-axis extends to the left assuming you stand straight up and face along the positive x-axis.

To allow the Pre-filter to assess the reliability of the measurement, the following three statistics are also computed and attached to each measurement:

1) The *Root Mean Square projection error* (ProjErrRMS) of the estimation. For a perfectly consistent measurement, this error should be zero.
2) The *number of feature points* (NPoints) that have been recognized between the current image and the image corresponding to a landmark. For a reliable estimation this number should be as large as possible.
3) The estimated *Slope* of the robot, which under the assumption that the robot is operating on a leveled floor, should be approximately zero.

Remark: If multiple landmarks are recognized in an individual image, they may for simplicity be treated as separate measurements and be treated one after another. This is how they are treated in this paper. However, for improved performance it is possible to process the measurements corresponding to one image simultaneously.

B. Pre-filter

The Pre-filter is considered a separate entity to emphasize that different applications may have different criteria for what is a reliable measurement. The basic rule is that ProjErrRMS and Slope should be very small and NPoints should be large for a measurement to be considered reliable. A simple scheme is hence to define a region over these three variables, and to reject any measurement that falls outside that region. If the boundaries of the region are defined by some threshold values, and these are set conservatively (the region is too small), a big portion of all measurements are rejected resulting in very few updates of the map. On the other hand, if too many of the measurements are accepted (the region is too large), the error variance of the visual measurements may be higher than is acceptable.

Experimentally, we have determined that a good compromise between accurate yet few rejected measurements is obtained if the following criteria are used:

$$\begin{cases} \text{if ProjErrRMS} > 3.0, & \text{then reject the measurement;} \\ \text{if NPoints} < 10, & \text{then reject the measurement;} \\ \text{if Slope} > 0.1, & \text{then reject the measurement.} \end{cases}$$

Note that a measurement is rejected as soon as one of the above three inequalities is satisfied.

The concept of a Pre-filter allows the user to select their own acceptance region. E.g., in some applications the environment may be poor on texture, which is needed to create and recognize landmarks. If, in the same application, the over-all requirement on accuracy is fairly low, the robustness of the localization is improved by enlarging the acceptance region. More visual measurements are then accepted making sure that frequent feedback is achieved in the SLAM module for map update and localization. This comes with the price of measurements that on average are somewhat less accurate.

Based on experiments, it is determined that the measurements remaining after the above filtering satisfactorily can be described by the true relative pose plus a stochastic, uncorrelated error vector $[\epsilon_x, \epsilon_x, \epsilon_\theta]^T$ which belongs to the Gaussian distribution of mean zero and covariance Σ^{vis}, where

$$\Sigma^{vis} = \begin{bmatrix} 81 \text{ cm}^2 & 0 & 0 \\ 0 & 81 \text{ cm}^2 & 0 \\ 0 & 0 & 0.0019 \text{ deg}^2 \end{bmatrix}. \quad (1)$$

In reality, the error vectors do not belong to any Gaussian distribution, but to some heavier-tailed distribution. But satisfactory results are obtained when we use a Gaussian approximation.

C. SLAM

The SLAM module maintains a map of all landmarks and estimates the location of the robot within this map. In particular, it estimates the global location of each landmark and the robot. Note that the global landmark location is completely unknown to the Visual Front-end.

The inputs to the SLAM module consist of 1) visual measurements passed on from the Pre-filter, 2) dead reckoning data from the odometry sensor, and 3) the previous estimate of map and robot location. The outputs consist of a refined estimate of the map and an updated robot location.

To describe the vSLAM algorithm, let us denote the map by Φ. The map consists of references to the landmarks created in the Visual Front-end, each of which will be denoted ϕ_n. The total number of landmarks at a given time is denoted N. This variable is initially set to zero, but will over time increase as landmarks are created and added to the map. The robot pose is denoted s_t, where t is a discrete number corresponding to the current time. The *pose* is defined by the 3×1 vector $[x_t, y_t, \theta_t]^T$, where x_t and y_t is the current location, and θ_t the heading.

To implement vSLAM, one needs to know two stochastic equations. The first equation is the *motion model*, which is a state-space model

$$s_{t+1} = f(s_t, u_t) + w_t, \quad (2)$$

where u_t denotes the odometry collected from time t to $t+1$ and w_t is a noise process representing the error in odometry.

The specific motion model depends on the robot's odometry, which on the other hand depends on the robot's kinematics and of the floor surface. In this paper, we use a two-parameterized motion model derived under the assumption of a holonomic drive system; however, the precise model is irrelevant for the development of the algorithm. Our model leads to

$$E(w_t) = 0, \qquad E(w_t w_t^T) = g(\theta_t, u_t, \sigma_T, \sigma_R),$$

where σ_T and σ_R are parameters characterizing the translational and rotational odometry uncertainty.

The second model is the *measurement model*

$$y_t = h(s_t, \phi_{n_t}) + v_t, \quad (3)$$

where n_t is the observed landmark at time t and v_t is a noise process representing the error in the relative

visual measurements. If the true robot pose is s_t, then the "error-free" visual measurement is given by $h(s_t, \phi_{n_t})$; i.e., $E(v_t) = 0$ and $E(v_t v_t^T) = \Sigma^{vis}$ (see also (1)).

The sequence $s^t = s_1, s_2, \ldots, s_t$ denotes the path of the robot up to time t. While s_t denotes a single robot pose, s^t denotes a sequence of poses from time 1 to time t.

The ultimate goal is to estimate the robot pose s_t based on 1) an estimate of s_{t-1}, 2) measurements u_{t-1} and y_t, and 3) the landmark pose ϕ_{n_t}. However, also ϕ_{n_t} is uncertain and must be estimated. Therefore, let $\phi_{n_t,t}$ denote the estimate of ϕ_{n_t} at time t. The idea proposed in [7] to estimate s^t and Φ (rather than s_t and Φ) is then adopted. In particular, we consider the posterior distribution $p(s^t, \Phi | n^t, y^t, u^t)$.

Now, think of the SLAM problem as a Bayesian network and note that the robot path *d*-separates [1] the landmark estimation (knowing the robot path makes the individual landmark estimation problems independent of each other). Use Bayesian calculus to rewrite the posterior distribution as

$$p(s^t, \Phi | n^t, y^t, u^t) \quad = \quad p(s^t | n^t, y^t, u^t) \prod_{i=1}^{N} p(\phi_i | s^t, n^t, y^t). \tag{4}$$

The factor $p(s^t | n^t, y^t, u^t)$ can be computed recursively from $p(s^{t-1}, \Phi | n^{t-1}, y^{t-1}, u^{t-1})$ and the most recent sensor readings y_t and u_t. For reasons that are explained in the sequel, we recommend to implement this recursive update using a particle filter, but other implementations are possible.

Note that the above factorization decomposes the problem of estimating $N+1$ poses, and their cross-correlations, simultaneously; into estimating one robot path and N landmark poses (separately).

The vSLAM update now proceeds by first updating the robot path, and then the map of visual landmarks.

Similar to [8], vSLAM implements the robot path update in (4) as a particle filter. Some advantages of this choice for the robot pose update is that the particle filter does not assume the pose distribution to be neither Gaussian, nor uni-modal. The landmark pose updates are implemented as Kalman filters because of the simple dynamic of the landmark poses (they are typically static).

The basic particle filter (as used in [7], [8]) is implemented by first computing some number, e.g. 200, of hypothetical paths (so called *particles*) based on the motion model (2), odometry data, and the odometry noise model, [10]. These initial particles represent a prior distribution of the robot path. Thereafter, so called *importance factors* are computed based on the prior distribution and the visual measurements. By resampling, with replacement, from the particles with probabilities equal to the importance factors, it can be shown that the distribution of the resulting particles (representing the posterior distribution) indeed approximates the true distribution $p(s^t | n^t, y^t, u^t)$.

As recognized by many researchers, the basic particle filter performs poorly if the motion model generates too few hypothetical paths in regions where $p(s^t | n^t, y^t, u^t)$ is large. This flaw of the particle filter is in vSLAM addressed by using a *mixture proposal distribution* instead of the motion model to compute the prior distribution [3], [5]. In particular, a subset of the particles, say 180, in the prior distribution are generated based on the motion model, while the remainder of them, say 20, are generated based on the measurement model (3). The particles generated according to the basic filter are called *nominal particles*, while the ones generated based on the measurement model are called *dual particles*.

The importance factors for each nominal particle is in principle computed based on the measurement model, while the importance factors for each dual particle in principle is computed based on the motion model. However, great care must be exercised to ensure that the importance factors of the nominal and dual particles are consistent and can be compared in the resampling process.

A very useful feature in vSLAM enabled by the novel use of the mixture proposal distribution, is that the ratio of nominal versus dual particles can be changed to deal with extreme situation, such as mal-functioning sensors, or kidnapping. More specifically, if a kidnapping scenario is detected or suspected, then one should generate only dual particles, and compute the importance factors from a uniform distribution. Note that simply modifying the odometry noise model to accommodate for the large odometry error typically will not work if a particle filter is employed since, in a normal-sized environment, it would require a very large number of particles to approximate a uniform distribution. Also, if no prior particle is in some neighborhood of the true pose, then all importance factors are likely to become less than the machine precision of the computer; i.e., all importance factors will become zero making it impossible to make any inference from the measurements.

Now, proceed to the landmark pose estimation. It is interesting to realize that since each Kalman filter update of a landmark pose must obey $p(\phi_i | s^t, n^t, y^t)$, it is necessary to associate a separate bank of Kalman filters (one filter per landmark) to each robot path hypothesis s^t (each particle). In other words, if there are 200 particles and 50 landmarks, then the total number of Kalman filters is 200 times 50; i.e., 10,000. On the other hand, each Kalman filter contains only three state variables (the recursively estimated landmark pose), and three measurement variables (the current landmark pose estimate); and only the Kalman filters associated to the measured landmark is updated in any given iteration of the vSLAM filter. Consequently, the update scheme becomes computationally very efficient.

IV. Experimental Results

To illustrate the performance of vSLAM, we shall investigate its localization capability as it is integrated on a robot operating in a typical home environment.

A. Systematic SLAM in Home Environment

First, consider Figure 2, which shows the result of vSLAM after the robot has traveled around along a *reference*

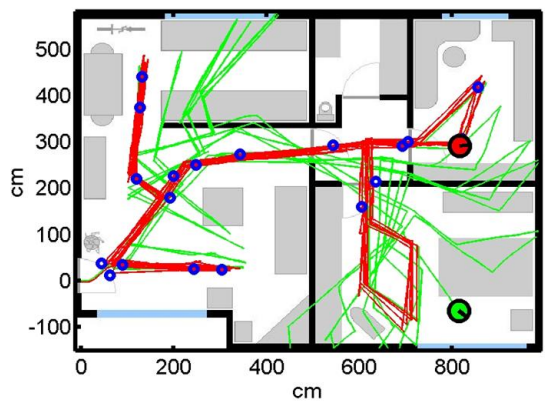

Fig. 2. Example result of SLAM using vSLAM. Red path (darker gray): vSLAM estimate of robot trajectory. Green path (lighter gray): odometry estimate of robot trajectory. Blue circles: vSLAM landmarks created during operations.

path in a two-bedroom apartment. The reference path, on which the robot drove, was unknown to the vSLAM algorithm.

To help interpret the result, the layout of the apartment is superimposed in the figure. This layout was neither generated, nor used, by vSLAM.

The vSLAM algorithm builds a map consisting of landmarks, which are marked as circles in the figure. The green path (odometry *only*) is obviously incorrect, since, according to this path, the robot is traversing through walls and furniture. The red path (the vSLAM corrected path), on the other hand, is consistently following the reference path. The vSLAM path, which uses a combination of visual measurements and odometry, provides a robust and accurate position determination for the robot.

Figure 3 shows that also a motion blurred image can

Fig. 3. The current view from the camera (left), which is successfully matched with the landmark image (right).

be used by vSLAM. Indeed, the landmark on the right is recognized in the blurred image in spite it is only partly visible in the current view (left).

Average localization error using vSLAM is in this example about 10-15 cm, while the median error is 9 cm.

B. Random SLAM in Home Environment

In many applications, it is not reasonable to assume that a robot travels on a reference path. Let us therefore now investigate vSLAM localization in a scenario where the robot travels in an arbitrary path (unknown to vSLAM). The experiment took place in a large living room of 7m x 4m on hardwood floor. As a reference, we acquired ground truth pose data with a Nav200 SICK laser. The speed of travel was about 15cm/sec.

The localization error as a function of time is shown in Figure 4. As a comparison, also the pure odometry results is displayed.

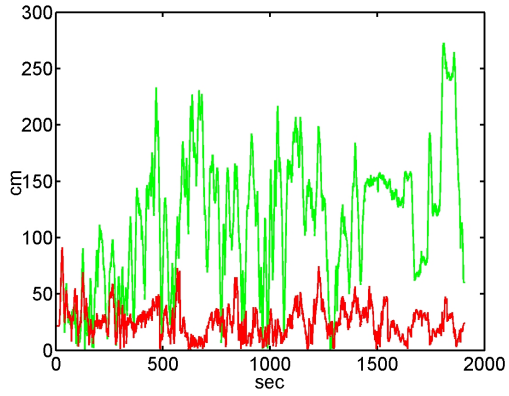

Fig. 4. Position localization error using only odometry (green curve) and using vSLAM (red path).

The localization accuracy using vSLAM is in this particular example somewhat lower than in the previous example. In particular, the average error is about 20 to 25 cm, while the median is about 14 cm.

It is also of interest to investigate the distribution of visual measurements among the landmarks. As seen in Figure 5, it turns out that a small portion of all landmarks result in the big majority of all visual measurement. There

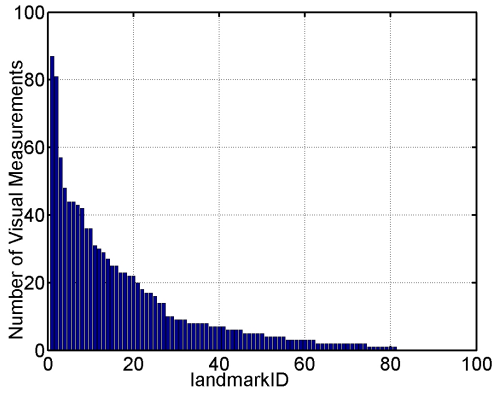

Fig. 5. Number of visual measurements per landmark.

are two reasons to this behavior. First of all, some landmarks were created late in the run, and did not have time to result in many measurements. But, a second important reason is that landmarks differ in quality. Some landmarks correspond to scenes with plenty of texture and many unique features. Other landmarks correspond to scenes for which the visual front-end barely managed to create a landmark. Poor landmarks, are difficult to recognize and generate visual measurements from.

V. Conclusions and Future Work

We have described vSLAM, a novel vision- and odometry-based SLAM algorithm that enables low-cost and

robust navigation in cluttered and populated environments. The vSLAM algorithm does not require an initial map, and the algorithm is typically good at detecting and correcting for slippage and collisions.

The key characteristics of vSLAM are

1) The low cost, which is enabled by basing the algorithm on a low-cost camera and an odometer.
2) The visual landmarks that are created and recognized in the front-end. They are essentially unique, which virtually eliminates the data association problem present in other landmark schemes.
3) The update scheme for robot pose and map. It is based on a particle filter and a Kalman filter bank and enables an efficient real-time implementation.
4) The mixed proposal distribution and the dynamic mixture ratio, which dramatically improves localization recovery from kidnapping and other large disturbances.

The space constraint did not permit us to include evidence of all of vSLAM's robustness. However, it can be shown that vSLAM is highly robust to kidnapping and dynamic changes in the environment, for example, changes in lighting, moving objects and/or people. vSLAM recovers quickly from kidnapping once the map has become dense with landmarks. Early in the mapping procedure, is it important to avoid kidnapping and disturbances in general.

A problem with the current implementation of vSLAM is the landmark data base, which in large environments may become prohibitively large. In particular, an individual landmark occupies anywhere between 40 kB and 500 kB depending on the number of visual features associated with the landmark, and the number of particles used in the SLAM module. For most practical implementations, it is therefore essential to integrate a *Landmark Database Manager* with the basic vSLAM system (see Figure 6). Such a database manager typically includes a *pruning mechanism*, which removes poor landmarks. But it also includes a *segmentation scheme* that partition the full vSLAM map into sub regions to limit the required primary memory for the map.

In some applications, it may also be appropriate to use more than one camera (see Figure 6). For example, one forward-pointing and one backward-pointing camera. Note that these are independent sensors, and images acquired by each one of them are processed separately adding landmarks to a common database, but generating visual measurements independently. The system as described in this paper already accommodates for the use of multiple cameras. An example of when multiple cameras might be appropriate is when the environment is poor on texture and where lowest possible price is not crucial.

It is useful to understand that the use of multiple cameras does not necessarily increase the computational requirements since an alternating image acquisition scheme can be employed to optimize the use of visual information.

Finally, note that if the environment does not contain a sufficient amount of visual features, then landmarks will not be created and visual measurements will not be computed. vSLAM will still produce pose estimates, but the estimates will be exactly what is obtained from wheel odometry. This scenario is rarely experienced, but is most likely to happen in environments that are free from furniture, decorations, and texture.

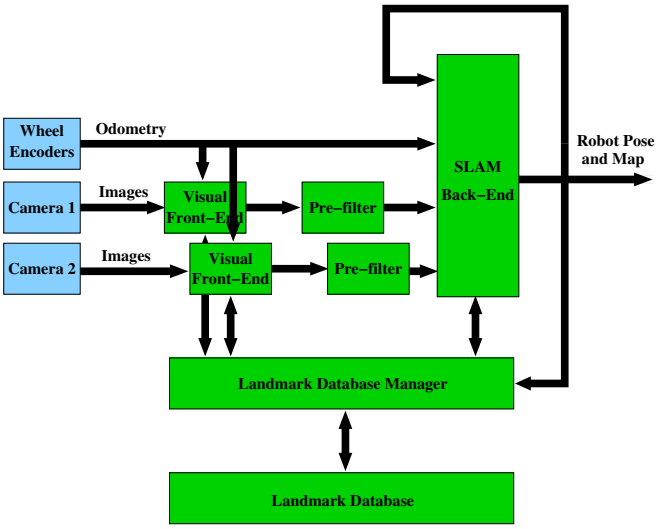

Fig. 6. Block diagram of a proposed extended vSLAM system.

References

[1] G. R. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York, 1999.

[2] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[3] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 19. Springer-Verlag, New York, 2001.

[4] L. Goncalves, E. Di Bernardo, D. Benson, M. Svedman, J. Ostrowski, N. Karlsson, and P. Pirjanian. A visual frontend for simultaneous localization and mapping. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2005.

[5] D.V. Hinkley. *Bootstrap Methods and their Application*. Cambridge Series in Statistical and Probabilistic Methematics, 1997.

[6] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *2002 IEEE ICRA, Workshop W4 Notes*, 2002.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[9] S. Thrun. Probabilistic algorithms in robotics. *Artificial Intelligence*, (4):93–109, 2000.

[10] S. Thrun. Probabilistic algorithms in robotics. Technical Report CMU-CS-00-126, Carnegie Mellon University, April 2000.

[11] S. Thrun. Mapping: A survey. Technical Report CMU-CS-02-111, Carnegie Mellon University, February 2002.

[12] S. Thrun, D. Fox., and W. Burgard. A probabilitic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[13] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrievel system based on invariant features. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, 2002.

View publication stats