

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.,
Petitioner

IPR2024-00695
U.S. Patent No. 8,649,426

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

PETITIONER’S EXHIBIT LIST	5
I. INTRODUCTION	6
II. GROUNDS FOR STANDING.....	7
III. NOTE.....	7
IV. TECHNOLOGY BACKGROUND.....	7
A. Video Coding	7
B. Known Challenges	8
C. Known Solutions	10
V. SUMMARY OF THE ’426 PATENT	12
A. Overview of the ’426 Patent.....	12
B. Prosecution History of the ’426 Patent.	15
VI. LEVEL OF ORDINARY SKILL IN THE ART.....	16
VII. CLAIM CONSTRUCTION	16
VIII. RELIEF REQUESTED AND REASONS FOR THE REQUESTED RELIEF	17
IX. DISCRETIONARY DENIAL WOULD BE INAPPROPRIATE.....	17
A. Discretionary denial under the <i>Fintiv</i> factors is not appropriate.....	17
1. No evidence regarding a stay	17
2. Parallel proceeding trial date	18
3. Investment in the parallel proceeding.....	18

4.	Overlapping issues with the parallel proceeding	19
5.	Identity of parties	20
6.	Other circumstances	20
B.	Discretionary denial under 35 U.S.C. § 325(d) is not appropriate	24
C.	Discretionary denial under <i>General Plastic</i> is not appropriate.....	24
X.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE....	25
A.	Challenged Claims and Statutory Grounds for Challenge	25
B.	Ground 1: Claims 1-5 and 7-13 are obvious under 35 U.S.C. § 103(a) over Perlman.	25
1.	Summary of Perlman.	25
2.	Claim 1	27
3.	Claim 2	54
4.	Claim 3	55
5.	Claim 4	56
6.	Claim 5	57
7.	Claim 7	59
8.	Claim 8	62
9.	Claim 9	65
10.	Claim 10	66
11.	Claim 11	67
12.	Claim 12	67

13.	Claim 13	76
C.	Ground 2: Claim 6 is obvious in view of Perlman and Oguz.	78
1.	Summary of Oguz	78
2.	Reasons to Combine Oguz with Perlman	79
3.	Claim 6.....	81
XI.	CONCLUSION.....	82
XII.	MANDATORY NOTICES	84
A.	Real Party-in-Interest	84
B.	Related Matters.....	84
C.	Lead and Back-up Counsel and Service Information	84
	CERTIFICATE OF WORD COUNT.....	86
	CERTIFICATE OF SERVICE	87

PETITIONER’S EXHIBIT LIST

Ex.1001	U.S. Patent No. 8,649,426 to White et al.
Ex.1002	Prosecution History of U.S. 8,649,426.
Ex.1003	Declaration of Dr. Immanuel Freedman under 37 C.F.R. § 1.68.
Ex.1004	<i>Curriculum Vitae</i> of Dr. Freedman.
Ex.1005	U.S. Patent No. 5,745,181 to Wilkinson (“Wilkinson”).
Ex.1006	U.S. Patent Publication No. 2009/0118017 to Perlman et al. (“Perlman”).
Ex.1007	U.S. Patent No. 5,729,279 to Fuller (“Fuller”).
Ex.1008	U.S. Patent Publication No. 2008/0151997 to Oguz (“Oguz”)
Ex.1009	IEEE 100: The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition, 2000.
Ex.1010	Original Complaint - <i>Video Solutions Pte. Ltd. v. Cisco Systems, Inc.</i> EDTX-2-23-cv-00222.
Ex.1011	Amended Complaint - <i>Video Solutions Pte. Ltd. v. Cisco Systems, Inc.</i> EDTX-2-23-cv-00222.
Ex.1012	Docket Control Order - <i>Video Solutions Pte. Ltd. v. Cisco Systems, Inc.</i> EDTX-2-23-cv-00222.
Ex.1013	Federal Court Trial Statistics – Dec. 31, 2023.
Ex.1014	U.S. Patent No. 5,553,073 to Barraclough (“Barraclough”)
Ex.1015	U.S. Patent No. 6,665,728 to Graumann (“Graumann”)
Ex.1016	U.S. Patent No. 6,466,248 to Spann (“Spann”)

I. INTRODUCTION

Pursuant to 35 U.S.C. §§ 311, 314(a), and 37 C.F.R. § 42.100, Cisco Systems, Inc. (“Petitioner”) respectfully requests that the Board review and cancel as unpatentable under (pre-AIA) 35 U.S.C. §103(a) claims 1-13 (the “Challenged Claims”) of U.S. Patent No. 8,649,426 (“’426 patent,” Ex.1001).

The prior art presented in this Petition describes the same solution to the same problem that is described in the ’426 patent. The purported novelty of the ’426 patent is a video compression technique that “divides each frame of the video signal into a plurality of segments.” Ex.1001, 5:53-55. “Each segment is encoded independently using a separate encoder.” Ex.1001, 5:55-56. The ’426 patent explains that “[e]ncoding the different segments in parallel reduces latency.” Ex.1001, 5:66.

As demonstrated in this Petition, however, the Perlman reference similarly describes “low-latency compression logic” that “subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168]. As shown below and confirmed in the Declaration of Dr. Freedman (Ex.1003), the claimed concept of separately encoding different segments of a video image was already known. *See generally* Ex.1003. The references presented in this Petition render obvious the Challenged Claims, which should be canceled for unpatentability.

II. GROUNDS FOR STANDING

Petitioner certifies that the '426 Patent is eligible for IPR and that Petitioner is not barred or estopped from requesting IPR challenging the patent claims. 37 C.F.R. § 42.104(a).

III. NOTE

Petitioner cites to exhibits' original page numbers. Emphasis in quoted material has been added.

IV. TECHNOLOGY BACKGROUND

A. Video Coding

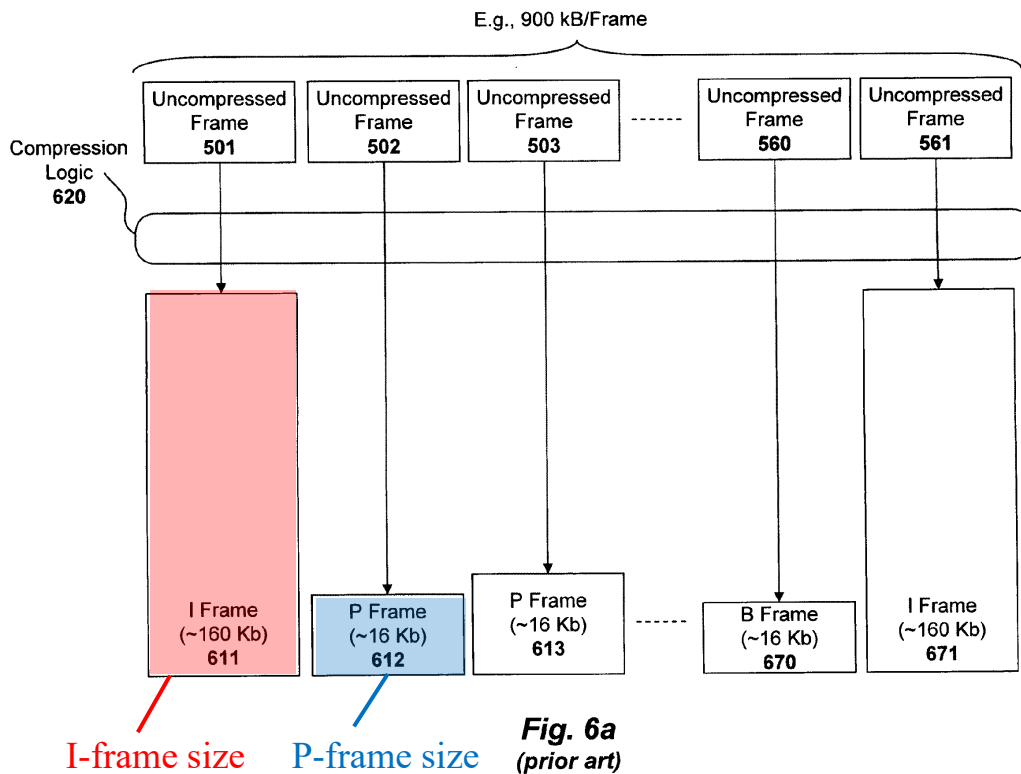
Streaming video data over the Internet was known long before the filing of the '426 patent. *See e.g.*, Ex.1006, [0009]-[0010]. Before video data is transmitted over a network, it is typically compressed using a standard codec. Various “types of compression, such as H.264, or Windows Media VC9, MPEG2 and MPEG4 are all more efficient at compressing a video stream because they exploit the similarities between successive frames.” Ex.1006, [0156]. Ex.1003, ¶ 27.

A video stream comprises video data that represents a series of frames that make up a video sequence. To encode a video stream, video codecs will designate different frames in different ways. Frames may be designated as reference frames (also referred to as I-frames) or non-reference frames (also referred to as P-frames). In MPEG coding, for example, includes “I-frames which are coded

without reference to other frames” and “P-frames which are predictive frames encoded using motion compensation from the previous I-frame.” Ex.1007, 19:46-50. In other words, a reference frame, or I-frame, includes all the data needed to produce a full image. *See* Ex.1006, [0062], [0157]. A P-frame, however, includes data for only the parts of the frame that have changed since a previous frame. *See* Ex.1006, [0062], [0157]. In this manner, the data representing the P-frames is much smaller than the data representing the I-frames. I-frames and P-frames were “well understood and documented.” Ex.1005, 1:26; Ex.1003, ¶ 28.

B. Known Challenges

While standard compression techniques can significantly reduce the amount of data required to transmit a video stream, they “typically come[] at the cost of higher latency.” Ex.1006, [0062]. This is because an “I frame is typically much larger than a ... P frame (often 10 times larger), and as a result, it takes proportionately longer to transmit at a given data rate.” Ex.1006, [0062]. Fig. 6a illustrates the size of I-frames compared to P-frames in kilobytes (Kb). Ex.1003, ¶ 29.



Ex.1006, Fig. 6a (annotated); Ex.1003, ¶ 29.

As a result, the data rate required to transmit a video stream will spike when transmitting the I-frames. “One problem with the foregoing types of compression is that although the average data rate is relatively low (e.g., 1.1 Mbps), a single I frame may take several frame times to transmit.” Ex.1006, [0159]. As illustrated in the figure below, “[t]he video stream data rate 624 is below the available max data rate 621 except for the I frames, where the peak data rate required for the I frames 623 far exceeds the available max data rate 622.” Ex.1006, [0161]. Ex.1003, ¶ 30.

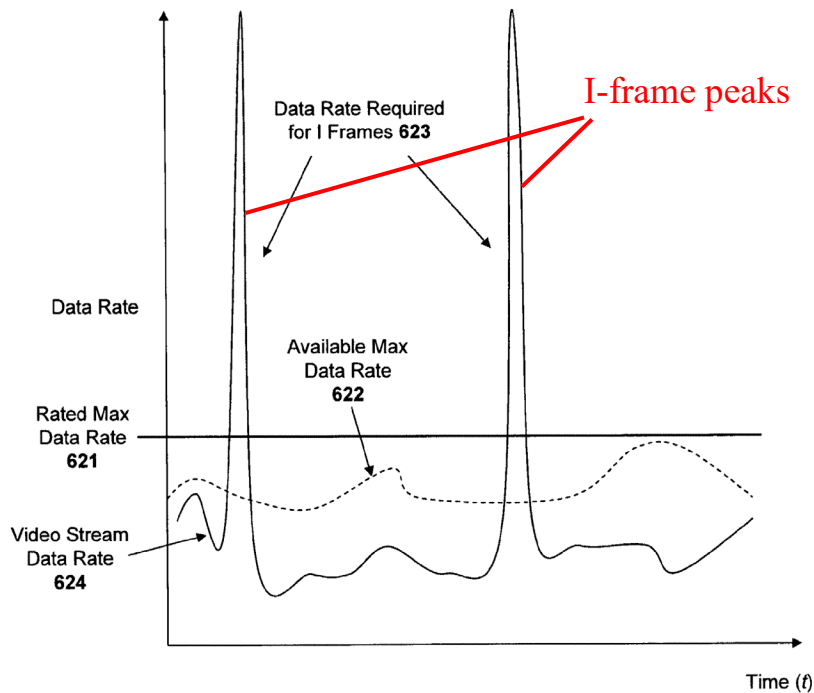
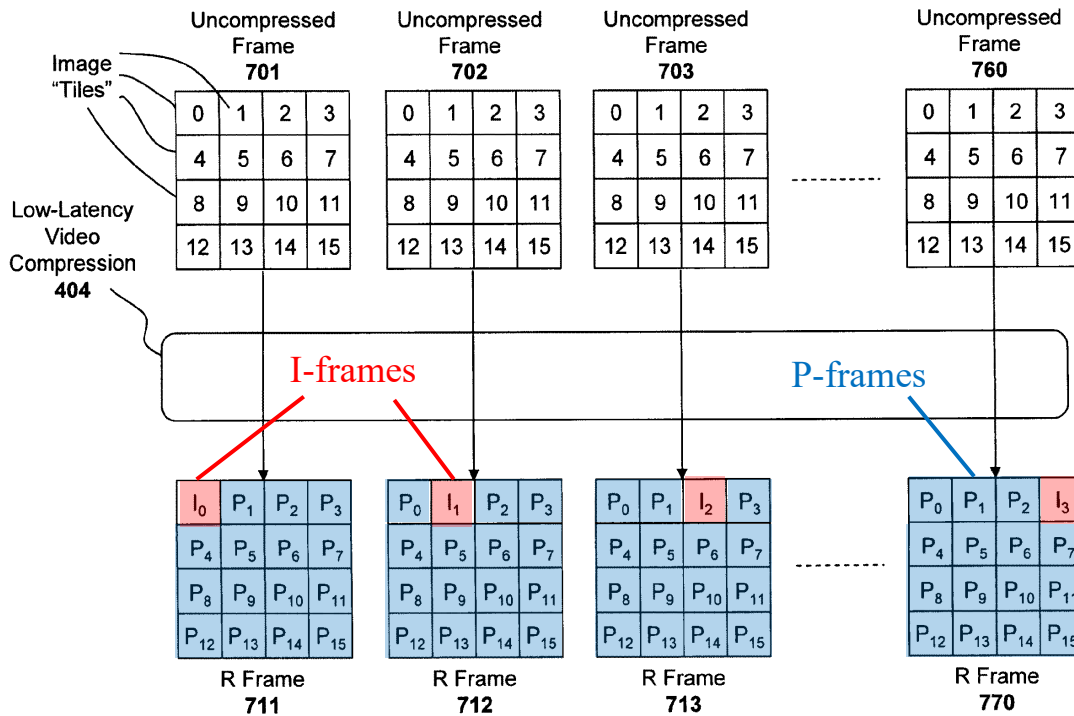


Fig. 6b

Ex.1006, Fig. 6b (annotated); Ex.1003, ¶ 30.

C. Known Solutions

One known solution to this problem was to independently encode different segments of a video frame and distribute the I-frames for each segment in time. For example, Perlman describes a technique that “subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168]. Then, “the I tiles and P tiles for tiles are progressively interleaved in time over successive frames.” Ex.1006, [0169]. This is illustrated below. Ex.1003, ¶ 31.



Ex.1006, Fig. 7a (annotated); Ex.1003, ¶ 31.

As shown in the figure above, an uncompressed image frame (e.g., 701) is divided into a four-by-four grid of 16 tiles. Each tile is then encoded as either an I-frame or a P-frame to produce an overall compressed frame (e.g., 711). In successive compressed frames, different tiles are designated as the I-frame, as shown from Frame 711 to 712 and 712 to 713. Only one tile in each overall compressed frame is designated to be an I-frame. Thus, the I-frames are distributed in time. Ex.1003, ¶ 32.

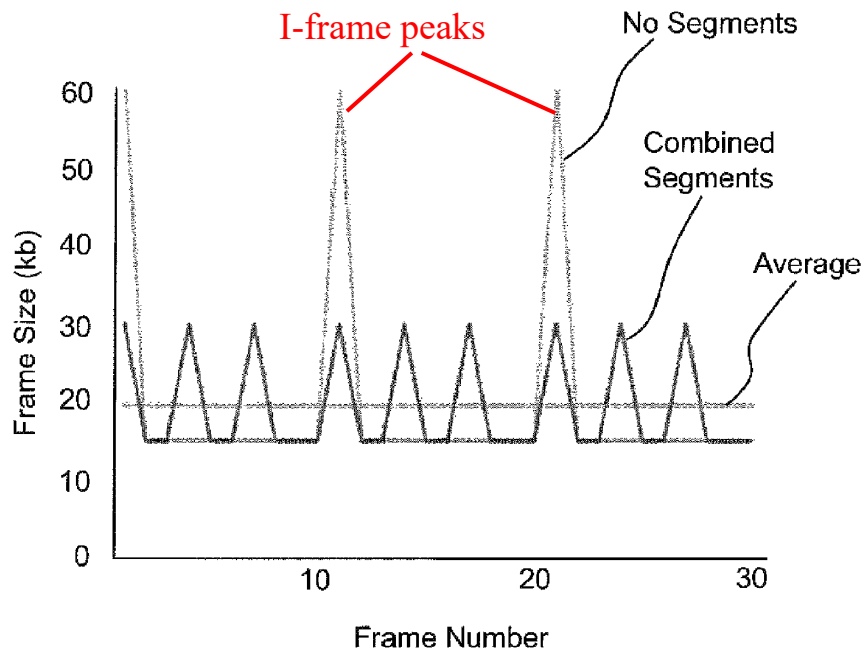
In summary, it was already known in the art—before the '426 patent's filing date—to encode separate segments of a video frame independently and interleave

the I-frames in time. Doing so was simply a known solution to the known problem of the I-frames causing spikes in the bandwidth required to transmit video data, thus leading to higher latency. Ex.1003, ¶ 33.

V. SUMMARY OF THE '426 PATENT

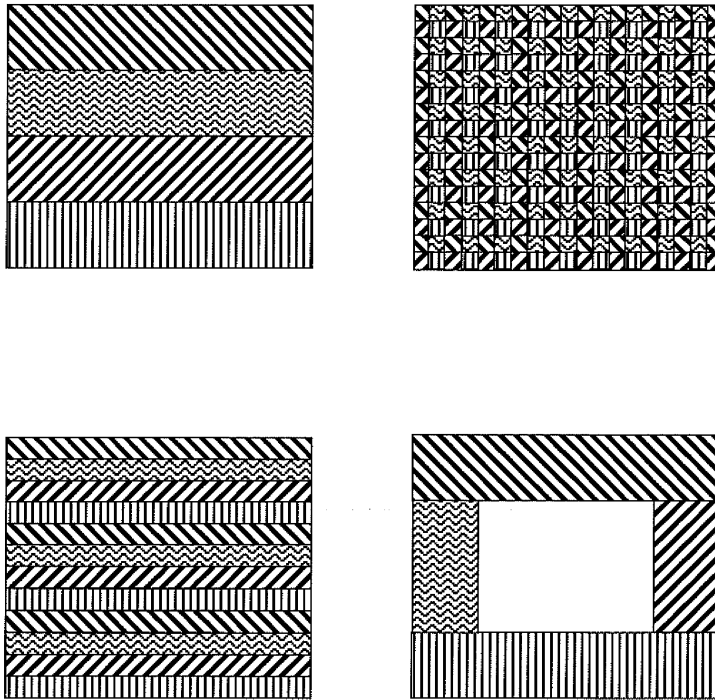
A. Overview of the '426 Patent

The '426 patent describes and claims nothing more than the known solutions to the known problems described above. The '426 patent explains that one “problem that is associated with modern video encoding techniques is the high peak/average data bursts caused by sending the I-frame via the communication network.” Ex.1001, 3:29-32. The '426 patent illustrates the same I-frame peaks as was known in the art: “Referring now to FIG. 2, it is apparent that for the prior art approach to video encoding the peak to average bit rate ratio of the video bitstream is very high, due to large I-frames, which causes large bursts of traffic on the network and often results in packet loss.” Ex.1001, 5:11-16. “[T]hus, a significant latency results from the databursts 21 unless the available bandwidth is sufficiently high.” Ex.1001, 5:22-24.



Ex.1001, Fig. 2 (annotated); Ex.1003, ¶ 34.

The '426 patent also describes the same solution that was known in the prior art: encoding different portions of a video frame separately, with only one portion encoded as an I-frame at a time. The '426 patent describes and claims “dividing each frame into a plurality of segments” and “encoding the segments independently of one another.” Ex.1001, 3:58-4:2. The '426 patent provides various examples of how to divide the frame into multiple segments, as shown below.



Ex.1001, Fig. 5 (partial).

The '426 patent also describes distributing the I-frames for each of the different segments in time: “This is shown diagrammatically in FIG. 4, where the I-frames for segment (0), segment (1) and segment (2) are offset one from another.” Ex.1001, 6:12-14. Ex.1003, ¶¶ 34-36.

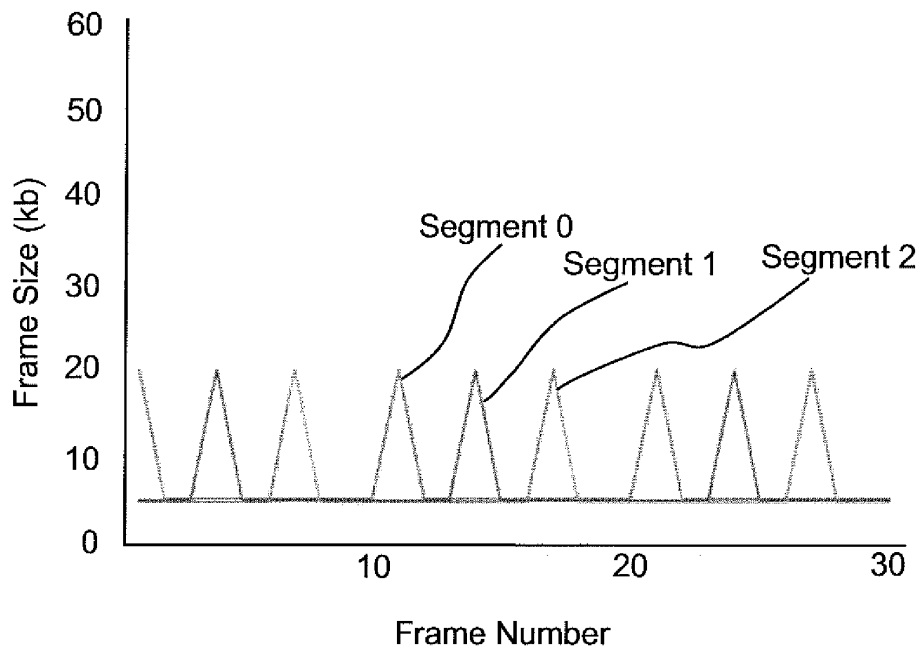


Figure 4
Ex.1001, Fig. 4.

This concept of encoding different segments of a frame independently and distributing the I-frames in time was already known at the time the '426 patent was filed. Ex.1003, ¶ 37.

B. Prosecution History of the '426 Patent.

The '426 patent was filed September 18, 2008. In a first Office Action, the claims were rejected as being anticipated by EP 1,227,684 to Bourne et al. ("Bourne"). Ex.1002, 142-68. The Applicant then amended the claims to add the concept that the I-frames for each segment are offset from one another. Ex.1002, 127, 134. The Office then rejected the claims as being obvious in view of Bourne

and U.S. Patent Publication 2008/0152014 to Schreier (“Schreier”). Ex.1002, 88-109. The Applicant then amended the claims to further recite details regarding the parallel encoding of the segments. Ex.1002, 51.

The Examiner then initiated an interview and allowed the claims after amending the claims to recite “*wherein during transmission of I-frame segment instances of one of the frame segments, P-frame segment instances of all other frame segments are transmitted.*” Ex.1002, 11-24. The patent then issued on February 11, 2024.

VI. LEVEL OF ORDINARY SKILL IN THE ART

A Person of Ordinary Skill in The Art (“POSITA”) in September of 2008 would have been someone knowledgeable and familiar with the video codec arts that are pertinent to the ’426 patent. A POSITA would have been familiar with the field of video codecs, including the encoding and transmission of video data, including MPEG and ITU standards that existed as of the late 2000s. A POSITA would have had a bachelor’s degree in electrical engineering, computer science, physics, or equivalent training, and approximately two years of experience working in the field of video codecs. Lack of professional experience can be remedied by additional education, and vice versa. Ex.1003, ¶¶ 20-22.

VII. CLAIM CONSTRUCTION

Claim terms in IPR are construed according to their “ordinary and customary

meaning” to those of skill in the art. 37 C.F.R. § 42.100(b). *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (en banc). Petitioner submits that, for the purposes of this proceeding and the grounds presented herein, no claim term requires express construction. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017). Ex.1003, ¶ 38.

VIII. RELIEF REQUESTED AND REASONS FOR THE REQUESTED RELIEF

Petitioner asks that the Board institute a trial for *inter partes* review and cancel the Challenged Claims in view of the analysis below.

IX. DISCRETIONARY DENIAL WOULD BE INAPPROPRIATE

A. Discretionary denial under the *Fintiv* factors is not appropriate.

The six factors considered for § 314 discretionary denial strongly favor institution given the compelling merits of this case. *See Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020) (precedential).

1. No evidence regarding a stay

No motion to stay has been filed, so the Board should not infer the outcome of such a motion. *Sand Revolution II LLC v. Continental Intermodal Group – Trucking LLC*, IPR2019-01393, Paper 24 at 7 (PTAB June 16, 2020) (informative); *see also Dish Network L.L.C. v. Broadband iTV, Inc.*, IPR2020-01359, Paper 15 (Feb. 12, 2021) (“It would be improper to speculate, at this stage,

what the Texas court might do regarding a motion to stay...”). Thus, this factor is neutral on discretionary denial.

2. Parallel proceeding trial date

The co-pending district court case was filed in the Eastern District of Texas on May 19, 2023, although the original complaint did not include the ’426 patent. *See* Ex.1010. The ’426 patent was added to the district court case in an amended complaint filed November 16, 2023. *See* Ex.1011.

Trial is currently scheduled for February 10, 2025. Ex.1012, 1. The most recent statistics for the Eastern District of Texas show a median time-to-trial of 21.4 months, consistent with the scheduled trial date. Ex.1013, 35. The Board’s final written decision is expected in September of 2025. While this is approximately seven months after the scheduled trial date, this factor is outweighed by both the compelling merits of this case (factor 6) and Petitioner’s stipulation (factor 4). The Board’s evaluation of the currently scheduled trial date should also be tempered by Petitioner’s diligence in quickly seeking inter partes review of the ’426 patent after it was first asserted six months into the pending litigation.

3. Investment in the parallel proceeding

The co-pending litigation is in its early stages. The Markman hearing is not scheduled until August of 2024. Ex.1012, 4. Moreover, the ’426 patent was only

recently added to the district court case. *See* Ex.1011. Petitioner has acted with diligence to prepare this Petition less than four months after the '426 patent was first asserted in the litigation. Under *Fintiv*, Petitioner's prompt filing "weigh[s] against exercising the authority to deny institution." *Fintiv*, Paper 11 at 11 ("If the evidence shows that the petitioner filed the petition expeditiously, such as promptly after becoming aware of the claims being asserted, this fact has weighed against exercising the authority to deny institution under NHK").

Regardless of whether this factor favors discretionary denial, it too is outweighed by the compelling merits of this case (factor 6) and Petitioner's stipulation (factor 4).

4. Overlapping issues with the parallel proceeding

There is no present overlap of prior art issues in the district court litigation. Moreover, Petitioner offers the following stipulation: Petitioner stipulates that if trial is instituted, it will not pursue the same grounds in the district court litigation. *See Sand Revolution II, LLC v. Cont'l Intermodal Group-Trucking LLC*, IPR2019-01393, Paper 24 at 12 (PTAB June 16, 2020). Consequently, this factor favors institution.

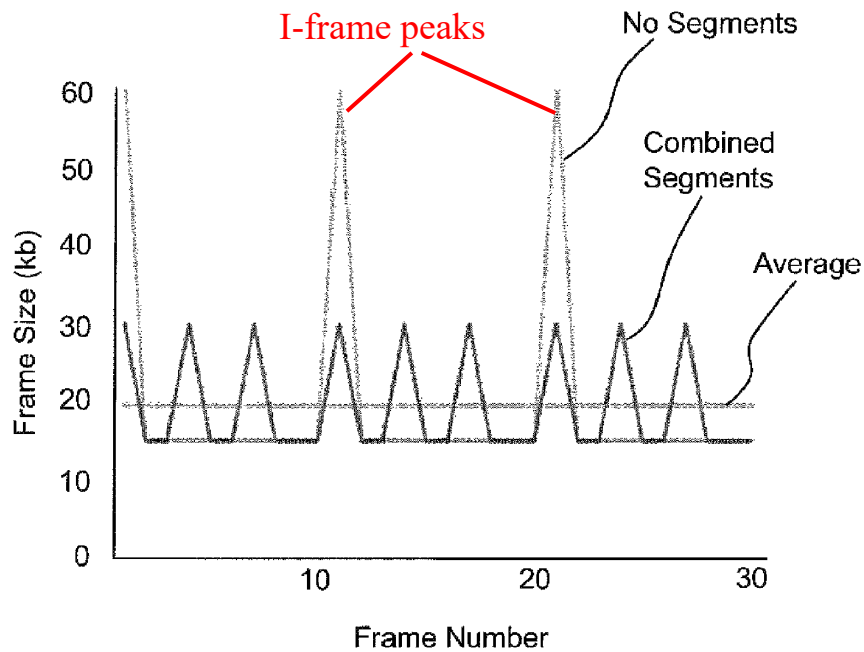
5. Identity of parties

Petitioner is a defendant in the litigation. That is true of most Petitioners in IPR proceedings. Accordingly, this factor should not be a basis for denying institution.

6. Other circumstances

The prior art presented in this Petition presents a particularly compelling case on the merits. The single reference, Perlman, renders obvious most of the challenged claims. Perlman identifies the same problem and the same solution described in the '426 patent.

The '426 patent describes and claims no more than a known solution to a known problem in encoding and transmitting video data. The '426 patent notes that one “problem that is associated with modern video encoding techniques is the high peak/average data bursts caused by sending the I-frame via the communication network.” Ex.1001, 3:29-31.



Ex.1001, Fig. 2 (annotated); Ex.1003, ¶ 34.

The solution to this problem—and purported novelty of the '426 patent—is a technique that “divides each frame of the video signal into a plurality of segments.” Ex.1001, 5:54-55. “Each segment is encoded independently using a separate encoder.” Ex.1001, 5:55-56. This allows the I-frames for each of the individual segments to be “offset one from another.” Ex.1001, 6:14.

However, POSITAs had already identified the same problem (I-frame spikes) that is described in the '426 patent. For example, Perlman explains that using standard compression techniques “typically comes at the cost of higher latency.” Ex.1006, [0062]. This is because an “I frame is typically much larger than a B or P frame (often 10 times larger), and as a result, it takes proportionately

longer to transmit at a given data rate.” Ex.1006, [0062]. This concept is illustrated below.

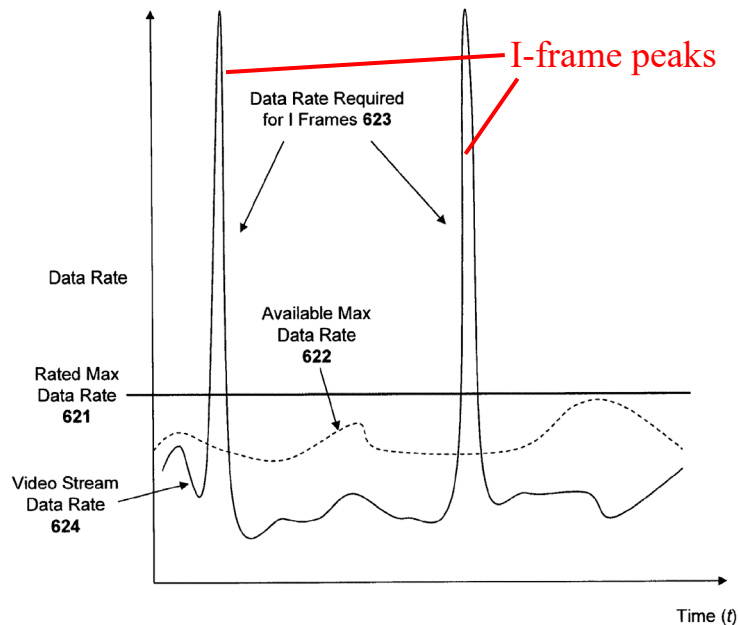


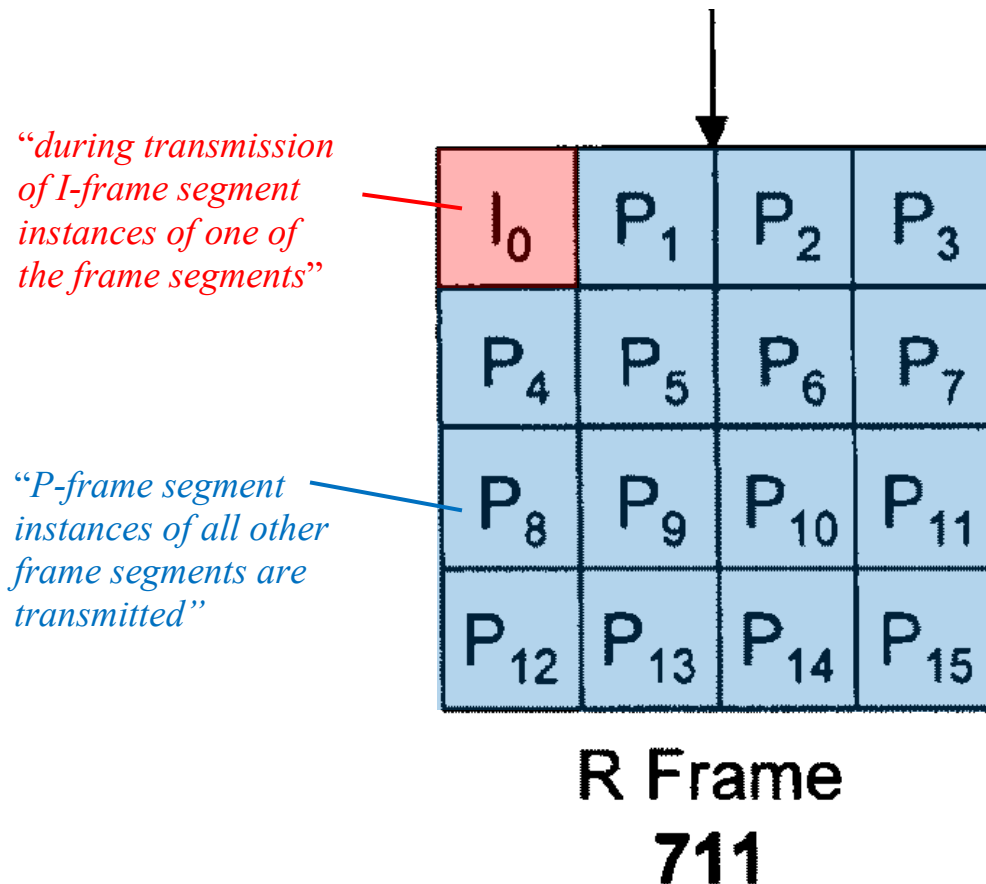
Fig. 6b

Ex.1006, Fig. 6b (annotated); Ex.1003, ¶ 30.

Perlman also describes the same solution as claimed in the '426 patent. Like the '426 patent, Perlman describes a technique that “subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168]. Then, “the I tiles and P tiles for tiles are progressively interleaved in time over successive frames.” Ex.1006, [0169].

Perlman’s technique also includes the same concept that resulted in allowance of the '426 patent. The claims of the '426 patent were allowed after adding the limitation “*wherein during transmission of I-frame segment instances of*

one of the frame segments, *P*-frame segment instances of all other frame segments are transmitted.” Ex.1002, 11-24. As shown in the annotated figure below, this limitation is taught by Perlman.



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 88.

The Board’s current policy is to “not deny institution of an IPR or PGR under Fintiv (i) when a petition presents compelling evidence of unpatentability.” Memorandum dated June 21, 2022 (“Memo”), 9. “Compelling, meritorious challenges are those in which the evidence, if unrebutted in trial, would plainly lead to a conclusion that one or more claims are unpatentable by a preponderance

of the evidence.” Memo, 4. Here, the petition plainly shows that the concept of separately encoding different portions of a video image and distributing the I-frames among the segments in time to reduce latency was already known, as evidenced by Perlman. Perlman describes (1) the same problem described in the ’426 patent, (2) the same solution described in the ’426 patent, and (3) the same concepts that were added to the claims to gain allowance of the ’426 patent. Notably, the prosecution examiner allowed the claims without evaluating their unpatentability in view of Perlman’s disclosure.

Thus, the evidence of unpatentability is particularly compelling here, and the PTAB should not deny institution under *Fintiv*.

B. Discretionary denial under 35 U.S.C. § 325(d) is not appropriate.

Denial under § 325(d) is not warranted because the challenges presented in this petition are neither cumulative nor redundant to the prosecution of the ’426 patent. The Examiner did not consider any of the references relied upon in this petition. Moreover, the challenges in this petition are non-cumulative because the prior art presented describes the claim limitations that were added to gain allowance. *See* Ex.1002, 11-24; Ex.1006, [0168]-[0169], Fig. 7a. Accordingly, the Board should not exercise its discretion to deny this petition under § 325(d).

C. Discretionary denial under *General Plastic* is not appropriate.

The ’426 patent has not been challenged in any prior IPR petition, so none of

General Plastic discretionary institution factors apply to this Petition. *See General Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 16 (PTAB Sept. 6, 2016) (Section II.B.4.i. precedential).

X. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A. Challenged Claims and Statutory Grounds for Challenge

Grounds	Claims	Basis
#1	1-5 and 7-13	35 U.S.C. § 103 (Pre-AIA) over Perlman
#2	6	35 U.S.C. § 103 (Pre-AIA) over Perlman and Oguz

U.S. Patent Publication No. 2009/0118017 to Perlman et al. (“Perlman”) was filed on December 5, 2007, and published on May 7, 2009. Perlman is thus prior art to the ’426 patent under (pre-AIA) 35 U.S.C. 102(e).

U.S. Patent Publication No. 2008/0151997 to Oguz (“Oguz”) was filed on December 20, 2007, and published on June 26, 2008. Oguz is thus prior art to the ’426 patent under (pre-AIA) 35 U.S.C. 102(e).¹

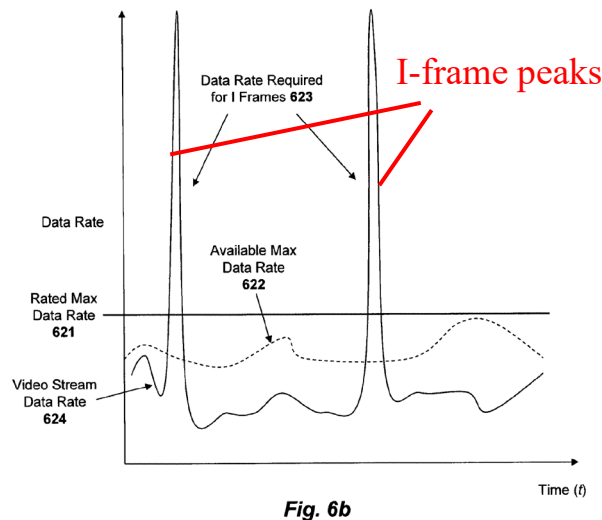
B. Ground 1: Claims 1-5 and 7-13 are obvious under 35 U.S.C. § 103(a) over Perlman.

1. Summary of Perlman.

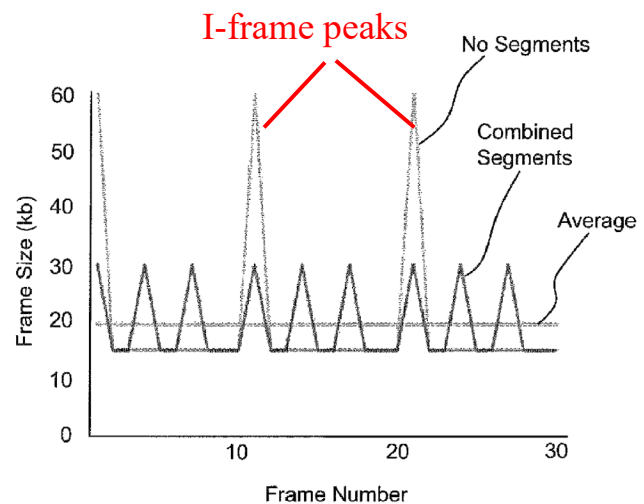
Perlman, like the ’426 patent, describes a technique for handling the problem

¹ If Patent Owner argues that the ’426 patent is an AIA patent, Perlman and Oguz would still qualify as prior art under post-AIA 35 U.S.C. §102(a).

of high latency due to I-frame data bursts. Perlman explains that using standard compression techniques “typically comes at the cost of higher latency.” Ex.1006, [0062]. This is because an “I frame is typically much larger than a B or P frame (often 10 times larger), and as a result, it takes proportionately longer to transmit at a given data rate.” Ex.1006, [0062]. This concept is illustrated below, along with a similar diagram from the ’426 patent.



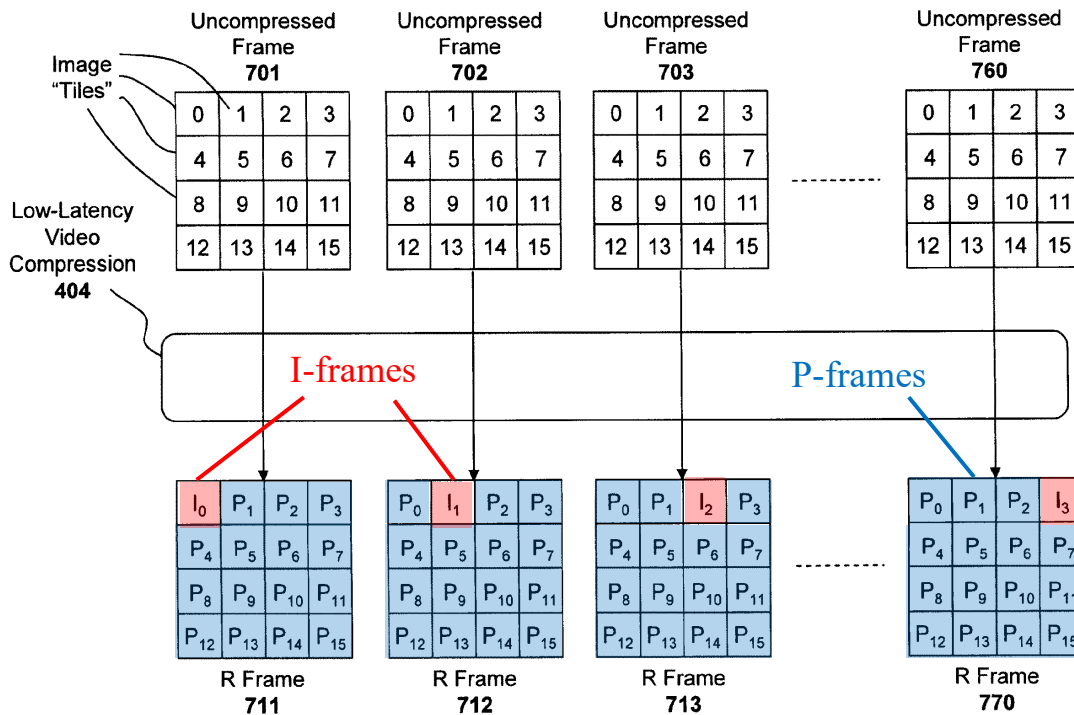
**Ex.1006, Fig. 6b (annotated);
Ex.1003, ¶ 42.**



**Ex.1001, Fig. 2 (annotated);
Ex.1003, ¶ 42.**

Like the ’426 patent, Perlman describes a technique that “subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168]. Then, “the I tiles and P tiles for tiles are progressively interleaved in time over successive frames.” Ex.1006, [0169]. As shown in the figure below, an uncompressed image frame 701 is divided into a four-by-four grid of 16 tiles. Each tile is then encoded as either an

I-frame or a P-frame. In different frames, different tiles are designated as the I-frame. In other words, the I-frames are distributed in time. Ex.1003, ¶ 43. This is illustrated below.



Perlman, Fig. 7a (annotated); Ex.1003, ¶ 43.

Thus, Perlman also provides evidence that it was known to encode separate segments of a video frame independently and distribute the I-frames in time. Again, doing so was simply a known solution to the known problem of the I-frames causing spikes in the bandwidth required to transmit video data, thus leading to higher latency. Ex.1003, ¶¶ 42-44.

2. Claim 1

[1.0] A method of encoding in real time a video signal that is associated with a

videoconference or telepresence session, the method comprising:

First, Perlman describes a method for compressing and transmitting video streams in real time (“*encoding in real time a video signal*”). Perlman describes “a new approach to video compression which decreases the latency and the peak bandwidth requirements for transmitting video.” Ex.1006, [0153]. Perlman also refers to its video compression as “real-time video compression.” Ex.1006, [0258].

A POSITA would have understood that compressing video data using a codec is “*encoding*” the video data as claimed. Ex.1003, ¶ 46. Indeed, Perlman expressly correlates the two words “encodes” and “compresses” with one another: “In one embodiment, the low-latency compression logic 404 divides up the video frame into a number of tiles, and **encodes (i.e., compresses)** one tile from each frame as an I frame.” Ex.1006, [0169].

Second, Perlman suggests using the disclosed encoding technique for video conferencing. While much of Perlman’s disclosure is in the context of video game streaming applications, Perlman explains that its techniques for video compression also apply to video conference streaming (“*a video signal that is associated with a videoconference or telepresence session*”):

Another feature of the hosting service 210 is the ability [] for **multiple users to collaborate while viewing live video**, even if using widely disparate devices for viewing. This is useful both when playing games and when using applications....

Such cameras and/or microphones, combined with local video/audio compression capability (particularly employing the low latency video compression techniques described herein) will enable a user to transmit video and/or audio from the user premises 211 to the hosting service 210, together with the input device control data. When such techniques are employed, then a capability illustrated in FIG. 19 is achievable: a user can have his video and audio 1900 appear on the screen within another user's game or application. This example is a multiplayer game, where teammates collaborate in a car race. **A user's video/audio could be selectively viewable/hearable only by their teammates.**

Such collaboration can be an immensely powerful tool for applications ... Each party will hear the voice of the other parties (**the conference calling** will be handled by any of many widely available **conference calling** software package in the app/game server(s) 1521-1525.... Still, for most of the purposes of the meeting, where the investor is watching the architects control the building fly-by or for **talking of video teleconferencing, even cellular latency should be acceptable.**

Ex.1006, [0257], [0259], [0261]; *see also* [0262]-[0263].

Perlman illustrates in Figure 19 a videogame screen that includes a videoconference view of the user "Player MASTERDAVE." The videogame screen itself is "*a video signal that is associated with a videoconference*":



Videoconference video of user Player MASTERDAVE

Perlman, Fig. 19 (annotated); Ex.1003, ¶ 48.

Perlman further explains that video conferencing involves “**real-time compression**” either in software or using one of many hardware compression devices to compress the video.” Ex.1006, [0258]. “And, since there would be effectively no latency, using the techniques described above the players would be able to talk or make motions to each other in real-time without perceptible delay.” Ex.1006, [0259].

Perlman thus explicitly states that video conferencing systems, whether in a gaming or business context, would benefit from Perlman’s compression techniques, “particularly employing the low latency video compression techniques described herein.” Ex.1006, [0259]; *see also* Ex.1006, [0261]- [0263]. In addition,

a POSITA would have found it obvious to apply Perlman’s low latency compression techniques to video conferencing because reducing latency improves personal interactions in a real-time collaboration. *See e.g.*, Ex.1014, 2:7-10 (low latency is desired “in order for a natural two way conversation to be feasible”); Ex.1016, 1:32-33 (“During real time videoconferencing, longer latency is considered a negative”); Ex.1003, ¶ 49. Indeed, if the latency is high, then “it may be difficult for parties to interrupt another in a manner that resembles a normal, in-person conference.” Ex.1015, 1:53-59; Ex.1003, ¶ 49.

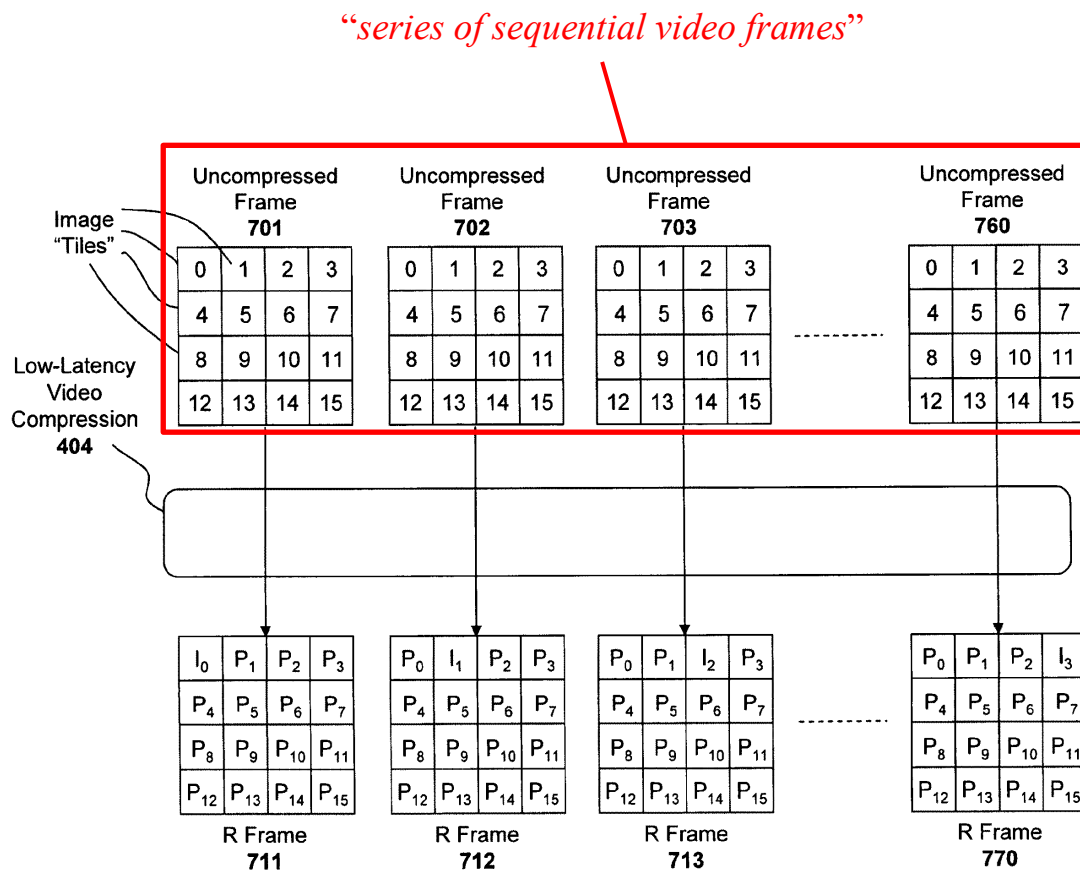
Thus, because Perlman describes a method for compressing video data in real time, and that video stream would be used for real time video conferencing, Perlman renders obvious a “*method of encoding in real time a video signal that is associated with a videoconference or telepresence session*” as recited in the claim. Ex.1003, ¶¶ 45-50.

[1.1] capturing a stream of video data comprising a series of sequential video frames;

Perlman describes a process by which video compression logic 404 receives (i.e., “*capture[s]*”) graphical and audio output from application software (“*stream of video data*”). “The graphical and audio output of the executed game or application software—i.e., new **sequences of video images—are provided to the low-latency video compression logic 404** which compresses the sequences of

video images according to low-latency video compression techniques, such as those described herein and transmits a compressed video stream.” Ex.1006, [0117].

Second, consistent with how video streams work, Perlman describes that the received video stream comprises a sequence of uncompressed frames 701, 702, 703 (“a series of sequential video frames”). Ex.1006, [0169]. Fig. 7a of Perlman illustrates a sequence of frames, as shown below.



Ex.1006, Fig. 7a (annotated); Ex.1003, ¶ 52.

Perlman’s technique uses “low-latency compression logic 404 [that] subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and

individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168].

“In the specific example shown in FIG. 7 a, each uncompressed frame is subdivided into a 4×4 matrix of 16 tiles. However, these underlying principles are not limited to any particular subdivision scheme.” Ex.1006, [0168].

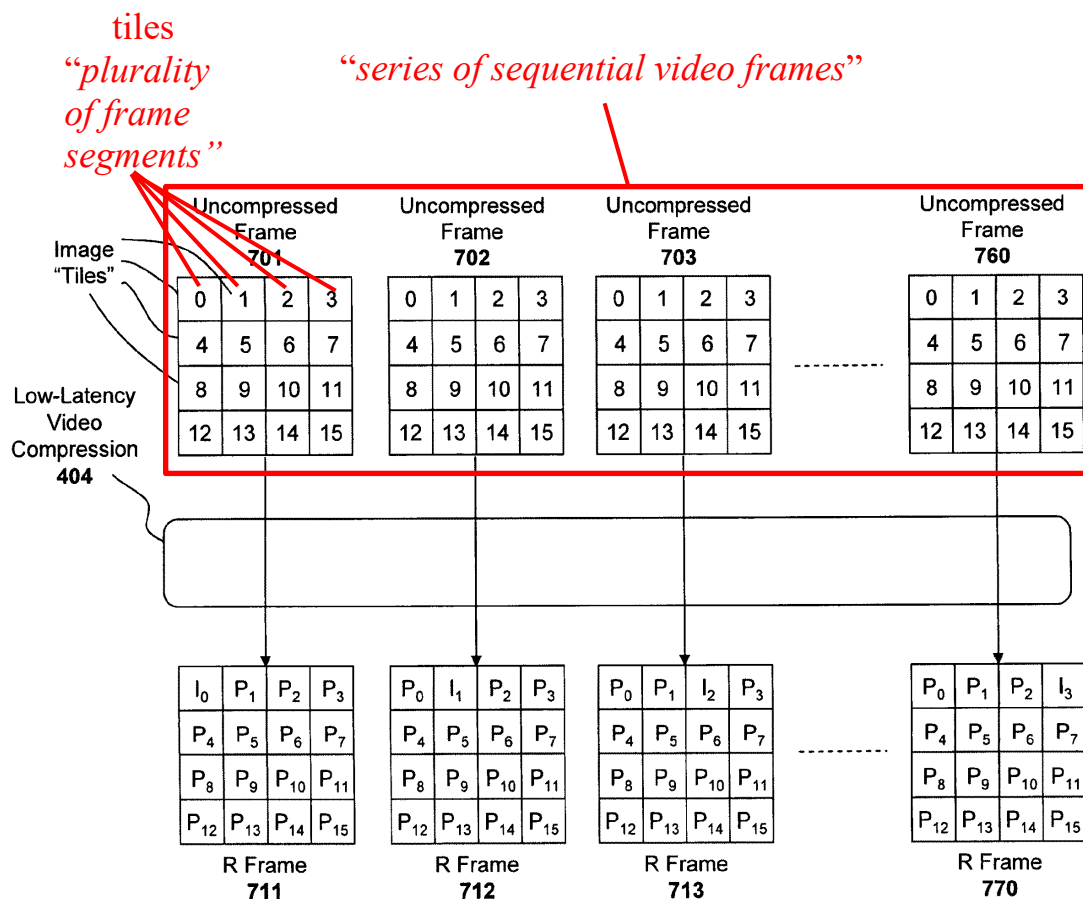
Thus, because Perlman describes receiving a stream of video data that includes a sequence of video images, Perlman renders obvious “*capturing a stream of video data comprising a series of sequential video frames*” as claimed. Ex.1003, ¶¶ 51-53.

[1.2] *separating the video frames into a plurality of frame segments, wherein each frame segment represents a part of a complete video frame;*

First, Perlman’s technique divides the video frames into a plurality of tiles (“*separating the video frames into a plurality of frame segments*”). Perlman’s technique uses “low-latency compression logic 404 [that] subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168]. “In the specific example shown in FIG. 7a, each uncompressed frame is subdivided into a 4×4 matrix of 16 tiles. However, these underlying principles are not limited to any particular subdivision scheme.” Ex.1006, [0168].

Second, Perlman also explains that each tile represents 1/16th of the full image (“*each frame segment represents a part of a complete video frame*”). In the

example of Fig. 7a, “the low-latency compression logic 404 divides up the video frame into a number of tiles, and encodes (i.e., compresses) one tile from each frame as an I frame (i.e., the tile is compressed as if it is a separate video frame of 1/16th the size of the full image), and the compression used for this ‘mini’ frame is I frame compression).” Ex.1006, [0169]. “[T]he remaining tiles [are compressed] as P frames (i.e., the compression used for each ‘mini’ 1/16th frame is P frame compression).” Ex.1006, [0169].



Ex.1006, Fig. 7a (annotated); Ex.1003, ¶ 55.

Thus, because Perlman describes dividing the frames into a plurality of tiles,

each tile representing a part of a full image, Perlman renders obvious “*separating the video frames into a plurality of frame segments, wherein each frame segment represents a part of a complete video frame*” as claimed. Ex.1003, ¶¶ 54-56.

[1.3] *encoding each frame segment independently and in parallel to generate parallel encoded data streams, one for each of the plurality of frame segments,*

First, Perlman discloses encoding each tile individually and in parallel using independent processes (“*independently and in parallel*”). As explained above at [1.2], “the low-latency compression logic 404 subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168].

Perlman refers to the collection of I-frame and P-frame tiles corresponding to an input video image as an “R frame.” Ex.1006, [0168]. Perlman explains that each of the tiles within an R frame are encoded “*independently and in parallel.*” To compress the tiles in parallel, Perlman provides an example in which “a multi-core and/or multi-processor 1200 is used to compress 8 tiles **in parallel**.” Ex.1006, [0212]. Compressing each tile may be done using multi-core processors, FPGAs, or other components. *See* Ex.1006, [0212]. As an example, Perlman explains that a multi-core processor may be used such that each core implements the coding as “an **independent** process.” Ex.1006, [212]. “[S]ince each of the 8 cores is responsible for just one tile, it operates largely independently from the other cores,

each running a separate instantiation of x264 [open-source software library for compressing].” Ex.1006, [213], [0156].

Accordingly, Perlman’s technique may use various hardware or software configurations for encoding each of the tiles “*independently and in parallel*” as claimed. A POSITA would have found it obvious to implement Perlman’s various techniques—including the techniques described with respect to Figure 7a—using parallel processors (or other suitable hardware) to encode different tiles in parallel as Perlman describes with respect to Fig. 12. Ex.1003, ¶ 59. A POSITA would have found this beneficial because encoding the separate tiles in parallel—as opposed to in series—maintains Perlman’s goal of reducing latency. *See* Ex.1006, [0169]; Ex.1003, ¶ 59.

Second, the output of Perlman’s parallel processors (or other type of encoders) produce “*parallel encoded data streams, one for each of the plurality of frame segments*” as illustrated in Fig. 12 below.

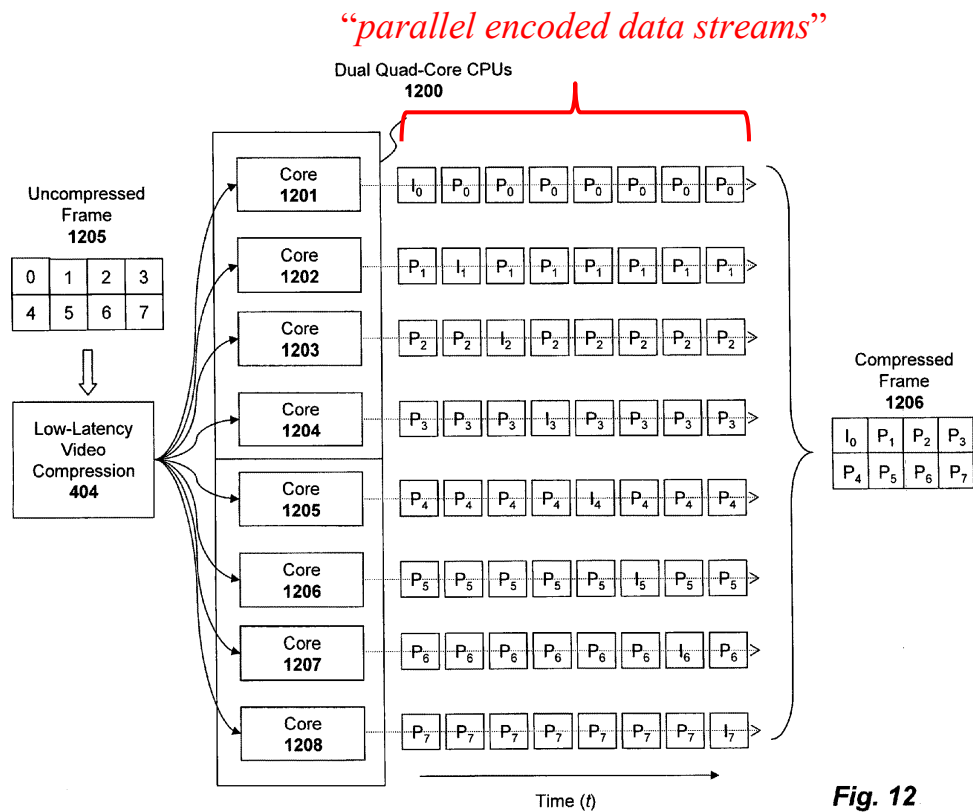
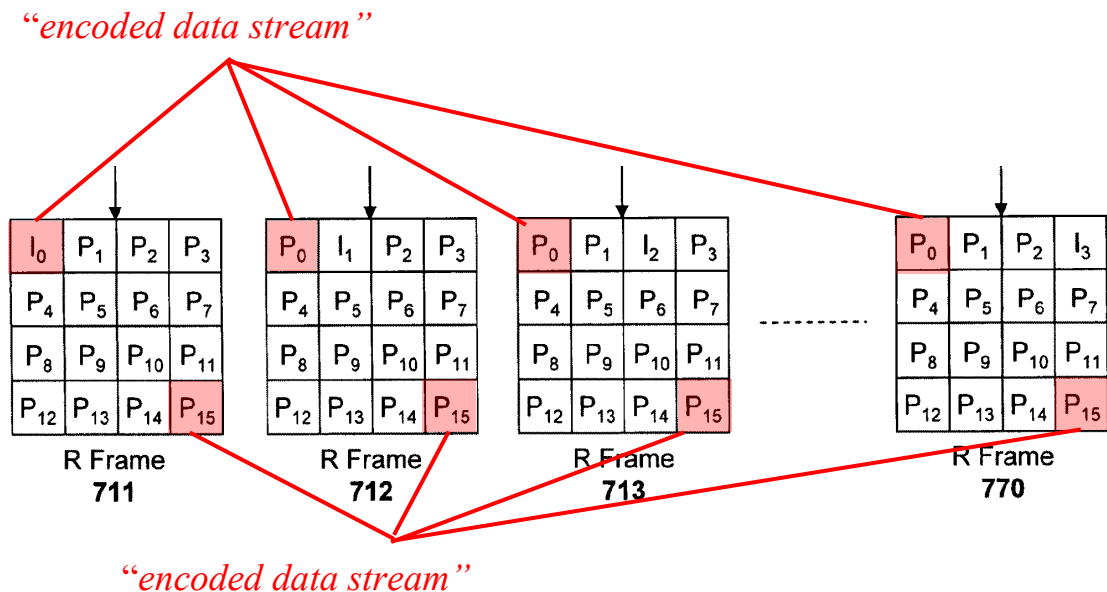


Fig. 12

Ex.1006, Fig. 12 (annotated); Ex.1003, ¶ 60.

Referring to the example in Fig. 7a (annotated below), one encoded data stream is represented by the tiles at the top left position (position “0”). Another encoded data stream is represented by the tiles at the bottom right position (position “15”). The tiles at each position thus represent “*parallel encoded data streams*” as claimed.



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 61.

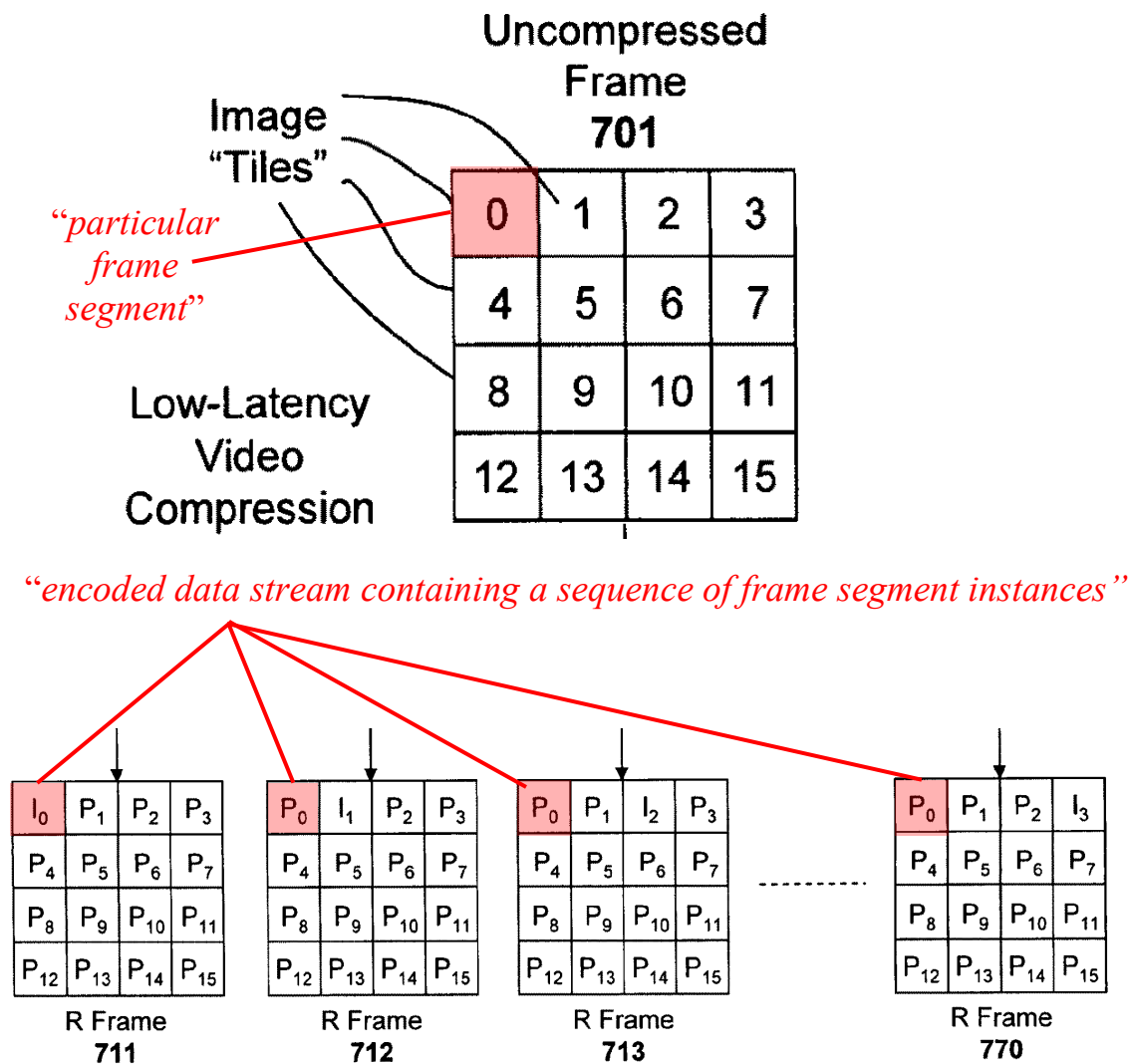
Thus, because Perlman describes individually encoding each tile in parallel as an independent process to produce parallel encoded streams, Perlman renders obvious “encoding each frame segment independently and in parallel to generate parallel encoded data streams, one for each of the plurality of frame segments” as claimed. Ex.1003, ¶¶ 57-62.

[1.4] each encoded data stream containing a sequence of frame segment instances each corresponding to a particular frame segment in the video frames, and

First, as explained above at [1.3], Perlman explains that each tile is encoded separately and in parallel to form “parallel encoded data streams.”

Second, Perlman discloses that each encoded data stream corresponds to a series of tile instances. This concept is further illustrated in Fig. 7a. As shown in

the annotated figure below, the sequence of tile instances for the top left tile (“a particular frame segment”) represent an “encoded data stream containing a sequence of frame segment instances.”



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 64.

Fig. 12 also illustrates the sequences of frame segment instances for the encoded data streams.

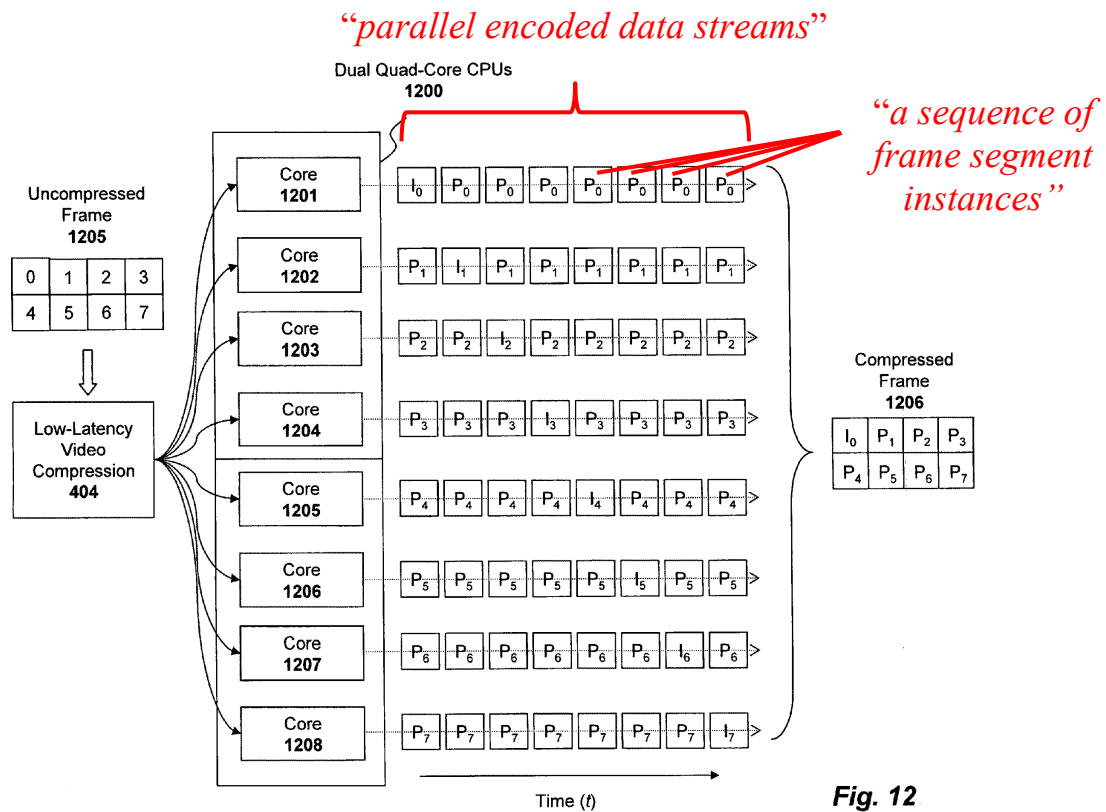


Fig. 12

Ex.1006, Fig. 12 (annotated); Ex.1003, ¶ 65.

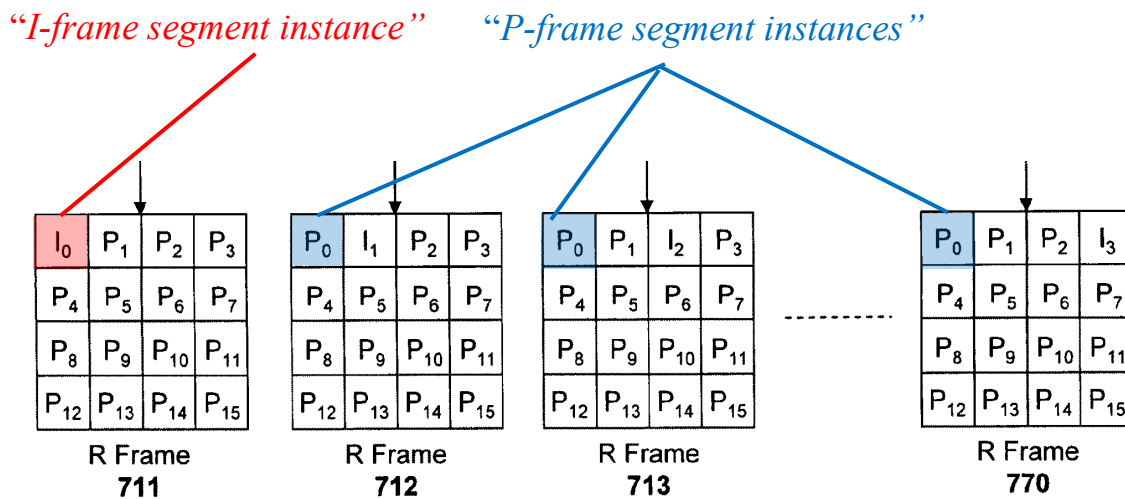
Thus, because Perlman describes dividing a frame into tiles, each tile including a series of tile instances, Perlman renders obvious “*each encoded data stream containing a sequence of frame segment instances each corresponding to a particular frame segment in the video frames*” as claimed. Ex.1003, ¶¶ 63-66.

[1.5] each encoded data stream containing I-frame segment instances that carry complete information about a particular frame segment and P-frame segment instances that carry partial information about the particular frame segment, the partial information representing information that has changed relative to a previous frame segment instance in that encoded data stream, and

First, as explained above at [1.4], Perlman teaches separately encoding each tile in parallel to form “*parallel encoded data streams*” as claimed.

Second, Perlman explains that each of the encoded data streams includes both I-frames and P-frames. Perlman’s “low-latency compression logic 404 subdivides each uncompressed frame 701-760 into a series of ‘tiles’ and individually encodes each tile as either an I frame or a P frame.” Ex.1006, [0168].

As shown in the figure below, the tiles at a particular position within the grid in each frame correspond to “*segment instances*” for a particular encoded data stream. For example, the tiles at the top left are all “*segment instances*” for the encoded data stream corresponding to the top left tile. Some of these “*segment instances*” are “*I-frame segment instances*” and some are “*P-frame segment instances*.”



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 69.

Perlman explains that the terms I-frame and P-frame have known meaning in the art. In video codec terminology, I-frames are reference frames that include all

the information that is used to produce a frame. P-frames are much smaller in size as they represent only the changes from one frame to the next. “Briefly, an I frame 611 is a DCT-based compression of a complete uncompressed frame 501 (similar to a compressed JPEG image as described above).” Ex.1006, [0157]. “P frames 612-613 generally are significantly smaller in size than I frames 611 because they take advantage of the data in the previous I frame or P frame; that is, they contain data indicating the changes between the previous I frame or P frame.” Ex.1006, [0157]; *see also* Ex.1006, [0062].

Accordingly, a POSITA would have understood that Perlman uses the terms I-frame and P-frame consistent with their generally accepted meaning within the field of video encoding. Ex.1003, ¶ 71. Thus, when Perlman uses the term I-frame, it refers to a frame that “*that carr[ies] complete information about a particular frame segment.*” When Perlman uses the term P-frame, it refers to a frame that “*carr[ies] partial information about the particular frame segment, the partial information representing information that has changed relative to a previous frame segment instance in that encoded data stream.*”

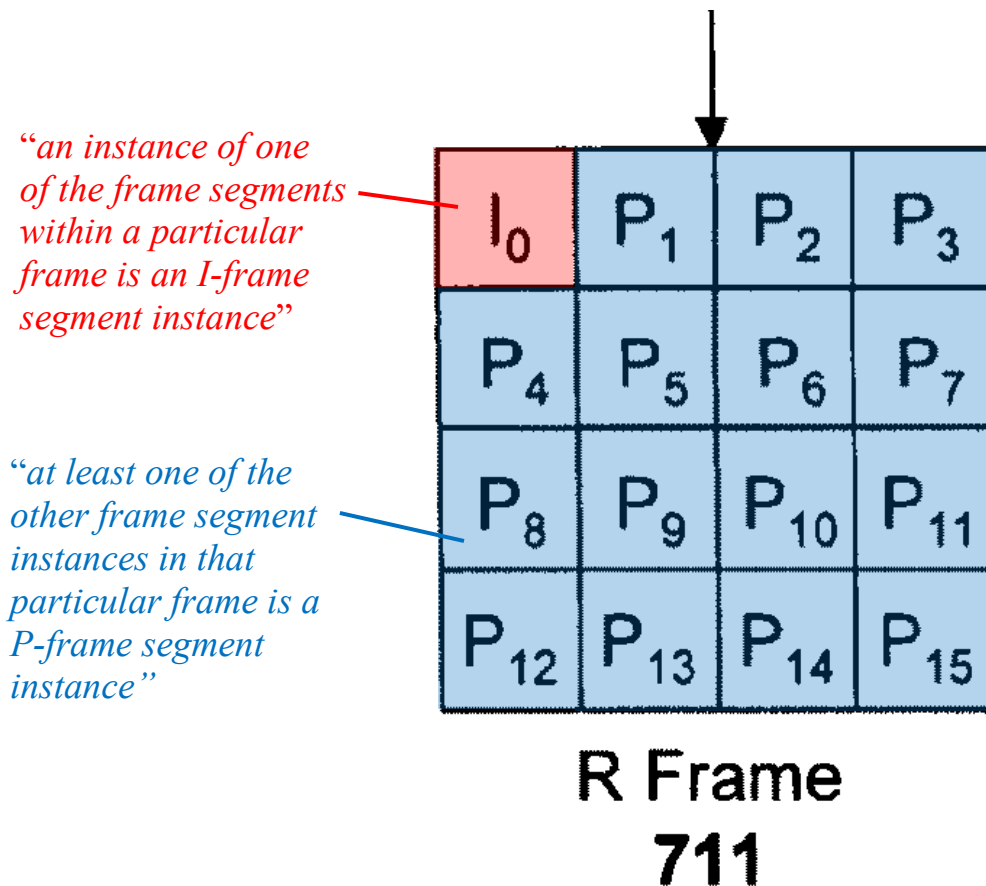
Thus, because Perlman’s technique uses I-frames and P-frames for each of the streams associated with each of the tiles, and a POSITA would have understood the meaning of I-frames and P-frames, Perlman renders obvious “*each encoded data stream containing I-frame segment instances that carry complete*

information about a particular frame segment and P-frame segment instances that carry partial information about the particular frame segment, the partial information representing information that has changed relative to a previous frame segment instance in that encoded data stream” as claimed. Ex.1003, ¶¶ 67-72.

[1.6] wherein the encoding of the frame segments for each frame is performed such that when an instance of one of the frame segments within a particular frame is an I-frame segment instance, at least one of the other frame segment instances in that particular frame is a P-frame segment instance,

Perlman provides an example in which, for a particular frame (711), one of the tiles (top left) is an I-frame instance (“*when an instance of one of the frame segments within a particular frame is an I-frame segment instance*”), and all other tiles are P-frame instances (“*at least one of the other frame segment instances in that particular frame is a P-frame segment instance*”). This is shown in Fig. 7a below.²

² While R Frame 711 is provided as an example, the other illustrated frames (712-713) similarly show only one I-frame tile while the rest of the tiles are P-frames.



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 73.

Perlman explains that “in a given frame time, only one tile of the tiles in the video frame is an I tile, and the remainder of the tiles are P tiles.” Ex.1006, [0169]. For example, in one frame, “tile 0 ... is encoded as I tile I₀ and the remaining 1-15 tiles are encoded as P tiles P₁ through P₁₅ to produce R frame 711.” Ex.1006, [0169]. In the next frame, “tile 1 ... is encoded as I tile I₁ and the remaining tiles 0 and 2 through 15 are encoded as P tiles, P₀ and P₂ through P₁₅.” Ex.1006, [0169]. “Thus, the I tiles and P tiles for tiles are progressively interleaved in time over successive frames.” Ex.1006, [0169].

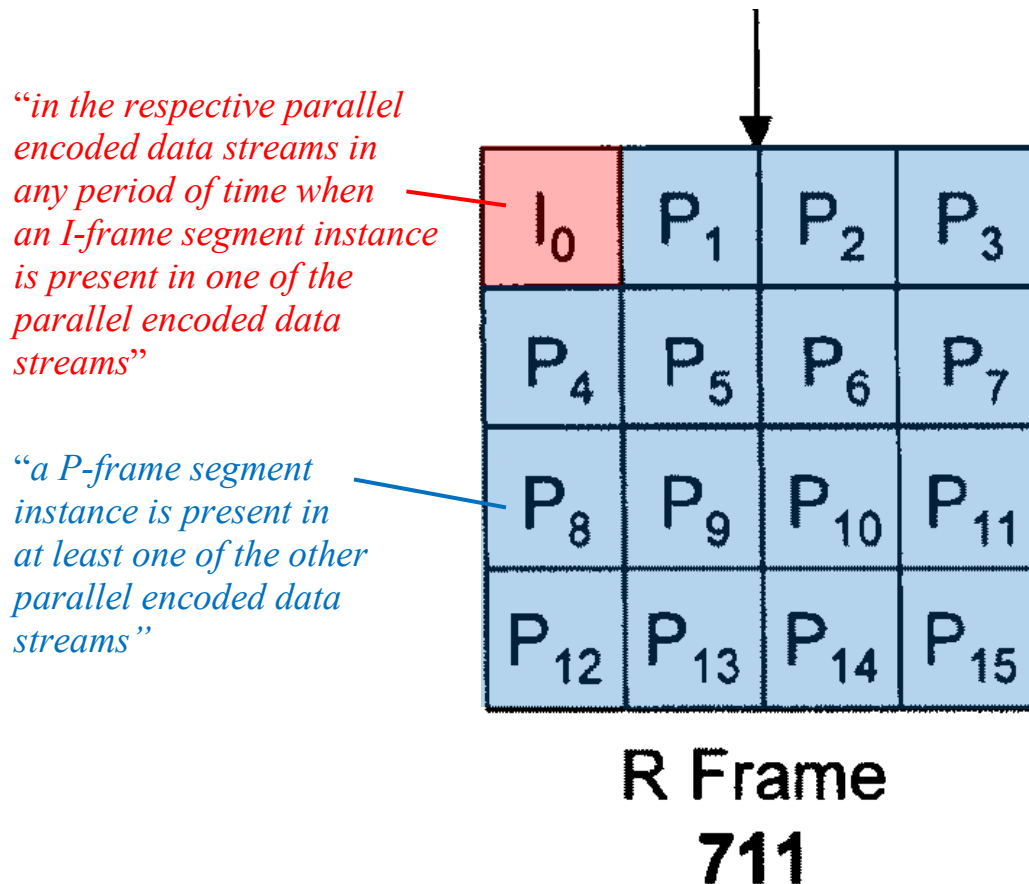
Thus, because Perlman provides an example where one tile is an I-frame and all other tiles are P-frames, Perlman renders obvious “*wherein the encoding of the frame segments for each frame is performed such that when an instance of one of the frame segments within a particular frame is an I-frame segment instance, at least one of the other frame segment instances in that particular frame is a P-frame segment instance*” as claimed. Ex.1003, ¶¶ 73-74.

[1.7] *whereby in the respective parallel encoded data streams in any period of time when an I-frame segment instance is present in one of the parallel encoded data streams, a P-frame segment instance is present in at least one of the other parallel encoded data streams;*

Perlman explains that “in a given frame time, **only one tile of the tiles in the video frame is an I tile, and the remainder of the tiles are P tiles.**” Ex.1006, [0169]. For example, in one frame, “tile 0 ... is encoded as I tile I₀ and the remaining 1-15 tiles are encoded as P tiles P₁ through P₁₅ to produce R frame 711.” Ex.1006, [0169]. In the next frame, “tile 1 ... is encoded as I tile I₁ and the remaining tiles 0 and 2 through 15 are encoded as P tiles, P₀ and P₂ through P₁₅.” Ex.1006, [0169]. “Thus, the I tiles and P tiles for tiles **are progressively interleaved in time over successive frames.**” Ex.1006, [0169].

The annotated figure below illustrates a given frame time (“*period of time*”). Each of the different tiles corresponds to a different parallel encoded data stream. For a given frame time, there is one I-frame tile (“*when an I-frame segment*

instance is present in one of the parallel encoded data streams”) and all other tiles are P-frames (“a P-frame segment instance is present in at least one of the other parallel encoded data streams”).



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 76.

Fig. 12 also illustrates that in a particular “period of time” there is only one I-frame and all other tiles are P-frames.

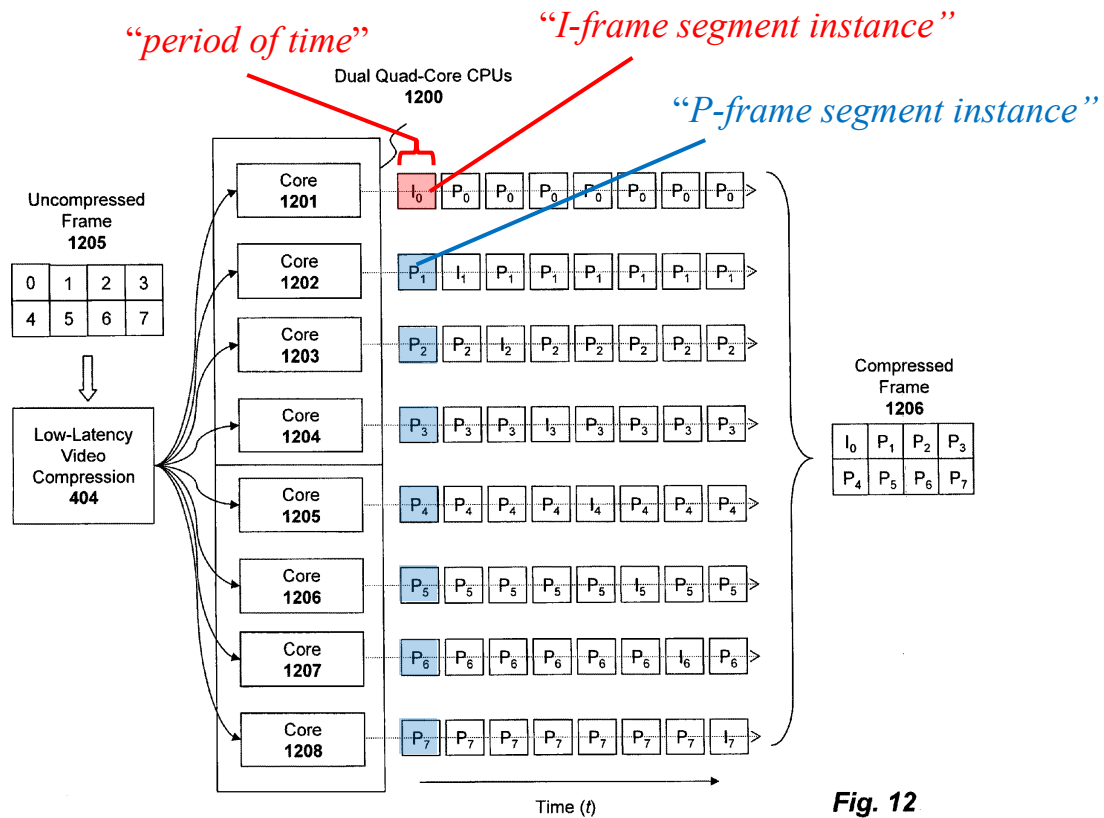


Fig. 12

Ex.1006, Fig. 12 (annotated); Ex.1003, ¶ 77.

Thus, because Perlman describes distributing the I-frames in time such that for a given frame time, one tile is an I-frame and all other tiles are P-frames, Perlman renders obvious “whereby in the respective parallel encoded data streams in any period of time when an I-frame segment instance is present in one of the parallel encoded data streams, a P-frame segment instance is present in at least one of the other parallel encoded data streams” as claimed. Ex.1003, ¶¶ 75-78.

[1.8] transmitting the parallel encoded data streams independently over a network to a receiver to permit the receiver to decode each frame segment and recombine the decoded frame segments from the different encoded data streams to reproduce an image;

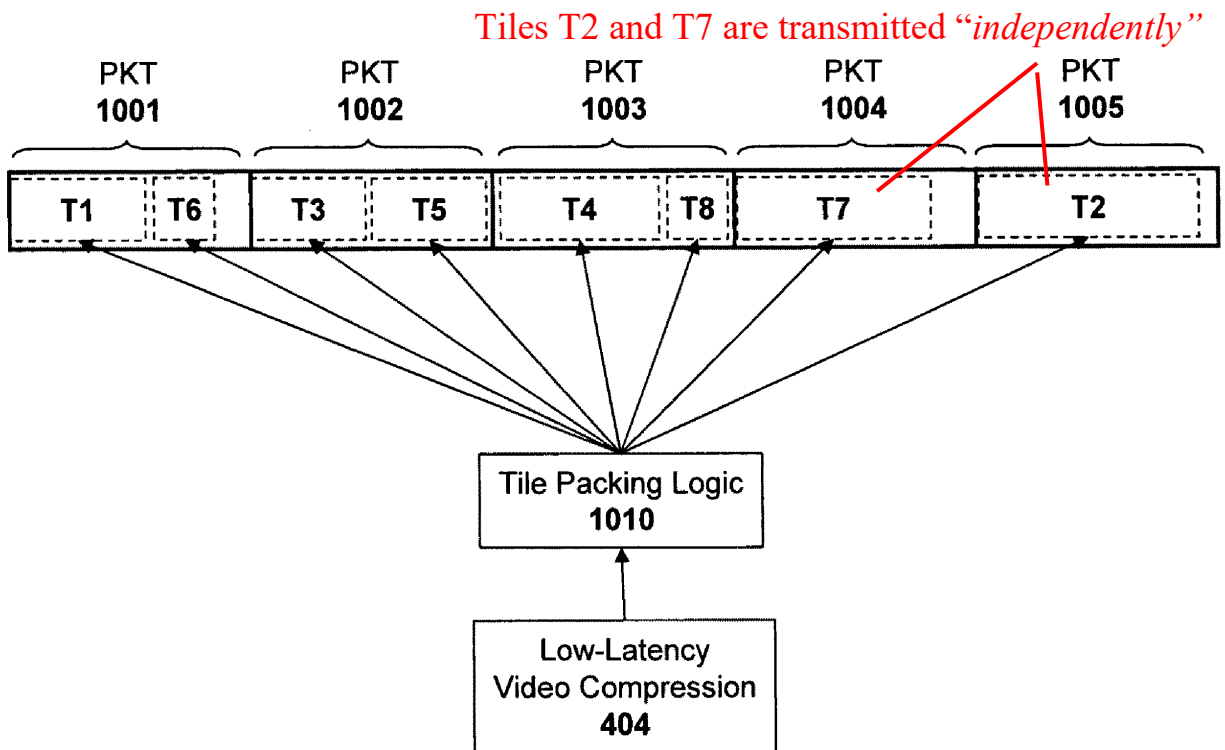
First, Perlman explains that the compressed tiles (“*parallel encoded data streams*”) are transmitted to a destination client over the Internet (“*over a network*”). “[C]ompressed tiles are transmitted to a destination client 415.” Ex.1006, [0214]. “The graphical and audio output of the executed game or application software—i.e., new sequences of video images—are provided to the low-latency video compression logic 404 which compresses the sequences of video images according to low-latency video compression techniques ... and transmits a compressed video stream ... back to the client 415 over the Internet 410.” Ex.1006, [0117].

Second, Perlman’s streams are “*independently*” transmitted because they have to be recombined on the receiving end. While Perlman describes combining the compressed tiles into a packet stream, the compressed tiles maintain their independence through transmission. Perlman decompresses each tile separately and then combines the tiles for rendering the full image. Indeed, similar to the parallel encoding process, Perlman describes a parallel **decoding** process as well. “The video decompression logic 412 running on the client 415 decompresses each tile as if it is a separate video sequence of small I and P frames, and then renders each tile to the frame buffer driving display device 422.” Ex.1006, [0170]. “For example, I_0 and P_0 from R frames 711 to 770 are used to decompress and render tile 0 of the video image.” Ex.1006, [0170]. “As mentioned above, decompression of I frames

and P frames is well known in the art, and decompression of I tiles and P tiles can be accomplished by having a **multiple instances of a video decompressor** running in the client 415.” Ex.1006, [0170].

Accordingly, because Perlman’s technique decompresses each tile separately and reconstructs the video on the receiving end, Perlman’s compressed tiles maintain their independence through transmission and are thus transmitted “*independently over a network to a receiver.*”

Alternatively, Perlman’s tiles are transmitted “*independently*” because different tiles—belonging to different data streams—are transmitted in separate packets. Perlman describes tile packing logic in which tile T7 is in packet 1004 and tile T2 is in packet 1005. *See* Ex.1006, [0198]. This is shown below in Fig. 10b.



Ex.1006, Fig. 10b (annotated); Ex.1003, ¶ 82.

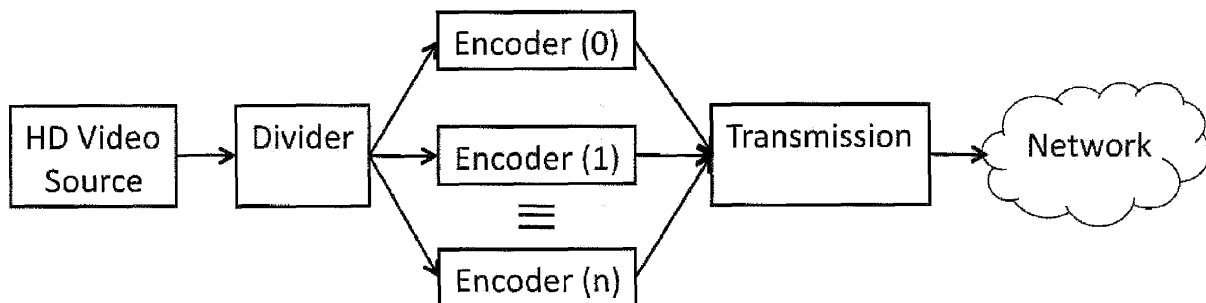
Accordingly, because each tile is part of a different data stream, and at least some of the tiles are transmitted in separate packets, at least some of the encoded streams are transmitted “*independently*” from each other. Moreover, it would have been obvious that in some instances, each of the different tiles from the different streams would be transmitted in different packets. Ex.1003, ¶ 83.

Thus, because a POSITA would have found it obvious for each tile to be transmitted in a separate packet, Perlman renders obvious “*transmitting the parallel encoded data streams **independently** over a network ...*” as claimed.

Ex.1003, ¶ 83.

Third, because Perlman’s technique transmits the encoded data streams independently for decompression and recombination on the receiving end. “The video decompression logic 412 running on the client 415 decompresses each tile as if it is a separate video sequence of small I and P frames, and then renders each tile to the frame buffer driving display device 422.” See Ex.1006, [0170]. Perlman’s technique “*permit[s] the receiver to decode each frame segment and recombine the decoded frame segments from the different encoded data streams to reproduce an image.*”

Perlman describes combining the independently encoded streams for transmission similar to how transmission is described in the ’426 patent. In the ’426 patent, the independently encoded streams are illustrated as being combined for transmission as shown in Fig. 3 below.



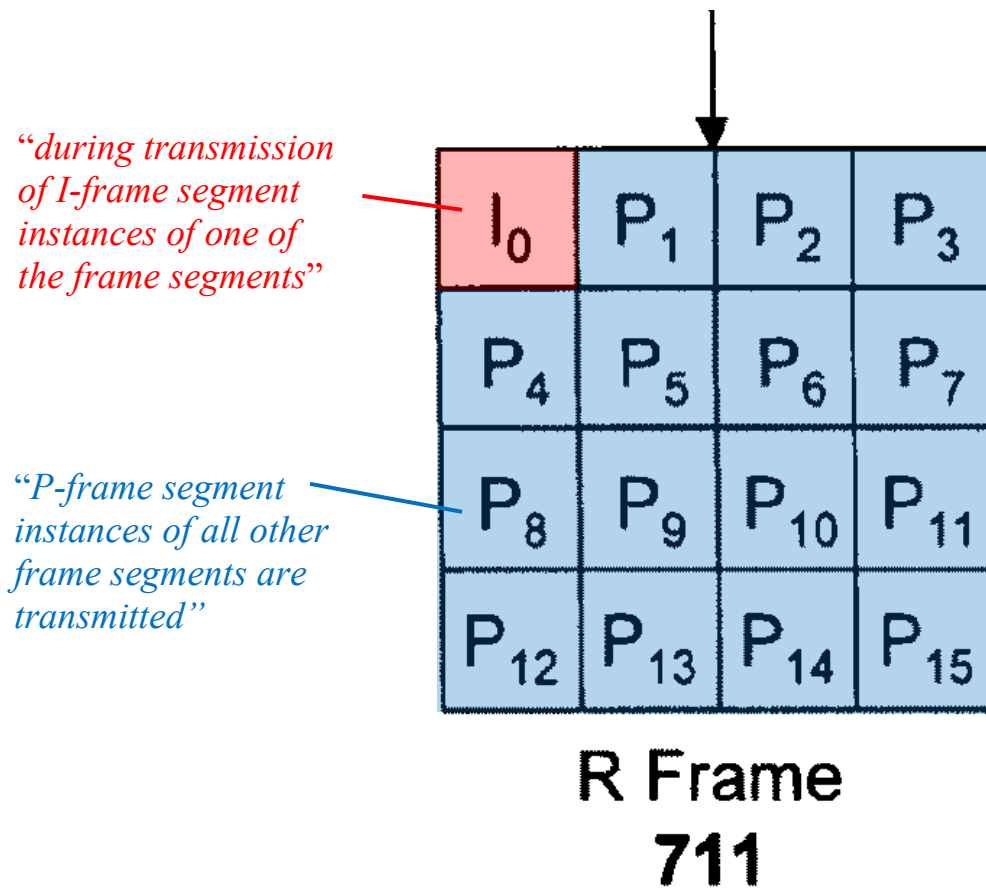
Ex.1001, Fig. 3.

Thus, because Perlman describes transmitting separately encoded data

streams such that the receiver would reassemble the streams to form the video image, Perlman renders obvious “*transmitting the parallel encoded data streams independently over a network to a receiver to permit the receiver to decode each frame segment and recombine the decoded frame segments from the different encoded data streams to reproduce an image*” as claimed. Ex.1003, ¶¶ 79-87.

[1.9] wherein during transmission of I-frame segment instances of one of the frame segments, P-frame segment instances of all other frame segments are transmitted.

Perlman provides an example in which “each successive video frame, the tile to be encoded as an I tile is changed. **Thus, in a given frame time, only one tile of the tiles in the video frame is an I tile, and the remainder of the tiles are P tiles.**” Ex.1006, [0169]. “Thus, the I tiles and P tiles for tiles are progressively interleaved in time over successive frames.” Ex.1006, [0169]. As shown in the example below, when the top left tile (I_0) is an I-tile (“*during transmission of I-frame segment instances of one of the frame segments*”), the remaining tiles P_1 through P_{15} are P tiles (“*P-frame segment instances of all other frame segments are transmitted*”).



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 88.

Fig. 12 also illustrates that during transmission of an I-frame tile, all other tiles are P-frames.

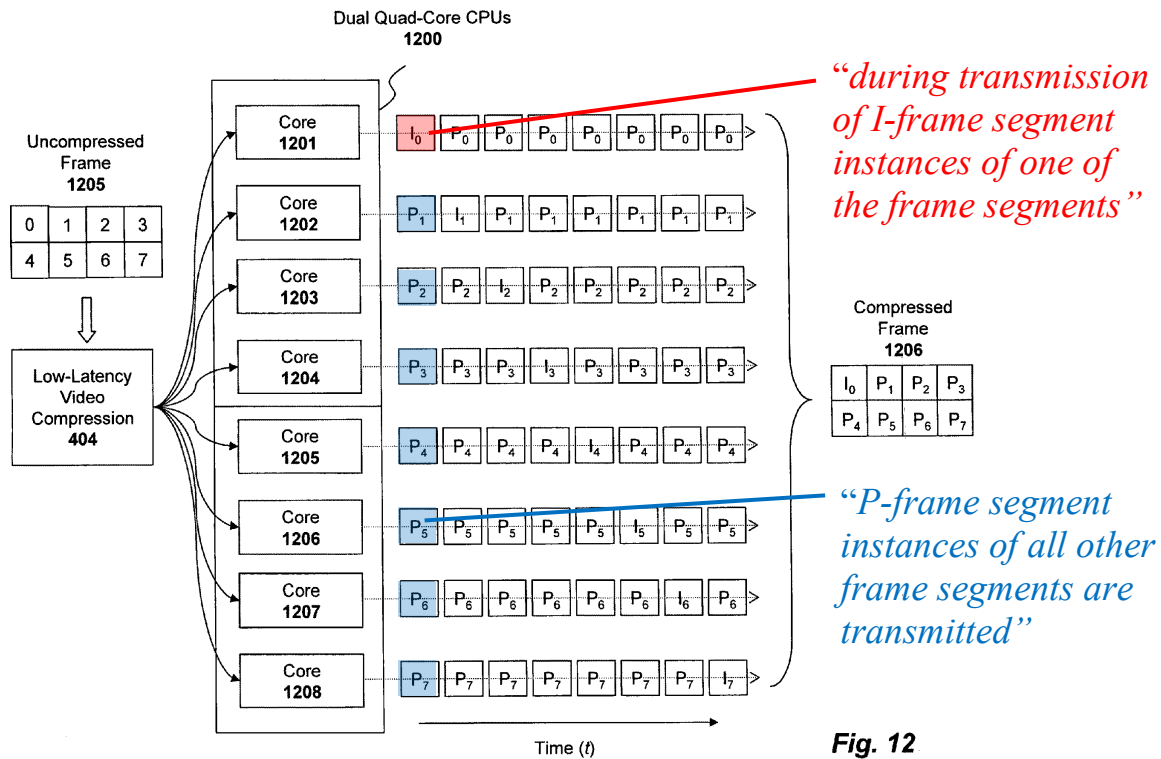


Fig. 12

Ex.1006, Fig. 12 (annotated); Ex.1003, ¶ 89.

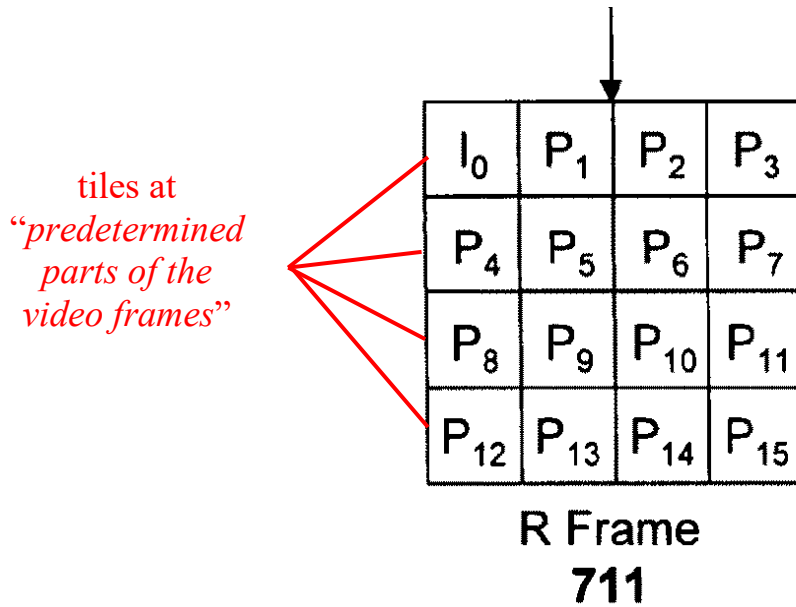
Thus, because Perlman provides an example where one tile is an I-frame and all other tiles are P-frames, Perlman renders obvious “wherein during transmission of I-frame segment instances of one of the frame segments, P-frame segment instances of all other frame segments are transmitted” as claimed. Ex.1003, ¶¶ 88-90.

3. Claim 2

[2.1] A method according to claim 1, wherein the frame segments correspond to predetermined parts of the video frames.

Perlman illustrates a “specific example shown in FIG. 7a [in which] each uncompressed frame is subdivided into a 4×4 matrix of 16 tiles.” Ex.1006, [0168].

The four-by-four grid is a predetermined pattern of equally-sized tiles, each sized at “1/16th the size of the full image.” Ex.1006, [0169].



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 92.

Accordingly, by using the compression logic to divide the frame into a four-by-four tile grid of equally sized tiles, the tiles “*correspond to predetermined parts of the video frames*” as claimed.

Thus, because Perlman describes dividing the tiles in a predetermined grid pattern, Perlman renders obvious “*wherein the frame segments correspond to predetermined parts of the video frames*” as claimed. Ex.1003, ¶¶ 91-94.

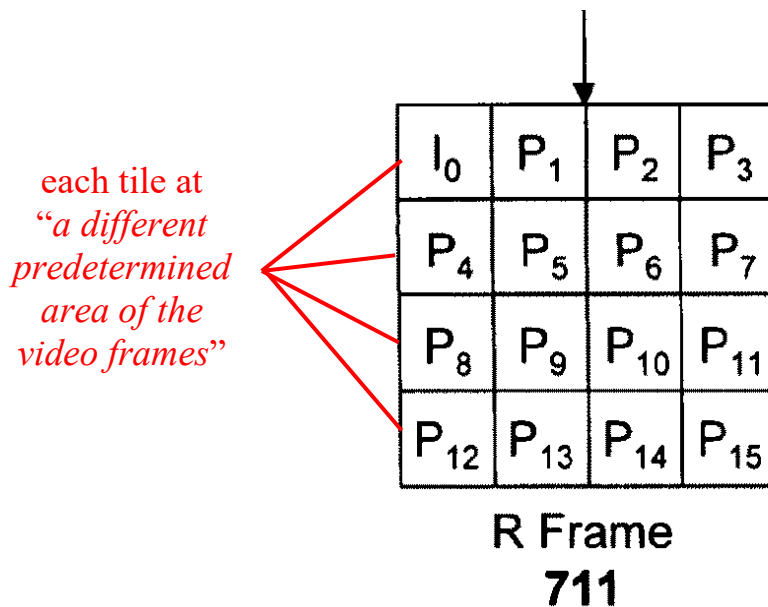
4. Claim 3

[3.1] *A method according to claim 1, wherein each frame segment corresponds to a different predetermined area of the video frames.*

First, as discussed above at [2.1], Perlman describes a method to divide the

frame into 16 equally-sized tiles.

Second, in the example shown in Fig. 1 there are no overlapping tiles and thus each tile “*corresponds to a different predetermined area of the video frames.*”



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 96.

Thus, because Perlman describes dividing the frame so as to produce non-overlapping tiles, Perlman renders obvious “*wherein each frame segment corresponds to a different predetermined area of the video frames*” as claimed.

Ex.1003, ¶¶ 95-97.

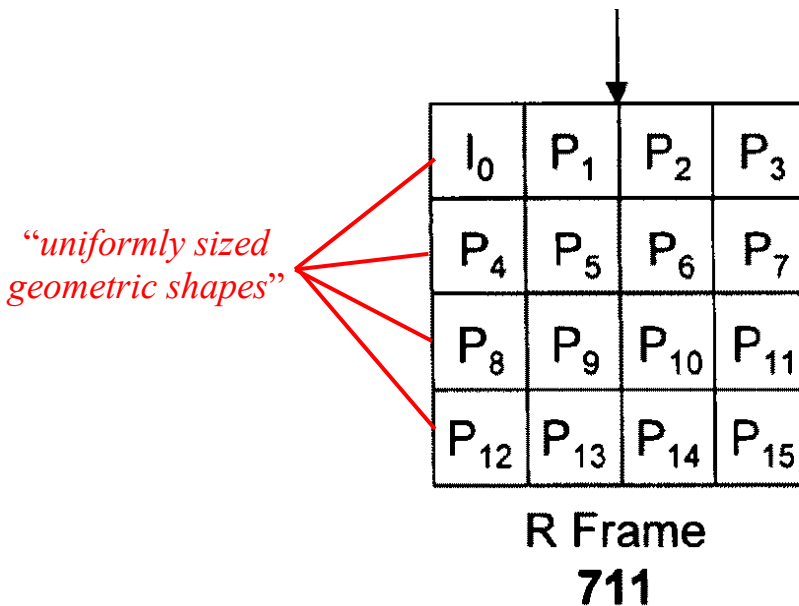
5. Claim 4

[4.1] A method according to claim 1, wherein the frame segments are uniformly sized geometric shapes.

Perlman both illustrates and describes an example in which the tiles are “*uniformly sized*” at 1/16th of the total image. “In one embodiment, the low-latency

compression logic 404 divides up the video frame into a number of tiles, and encodes (i.e., compresses) one tile from each frame as an I frame (i.e., the tile is compressed as if it is a separate video frame of “1/16th the size of the full image.” Ex.1006, [0169].

As illustrated in Fig. 7a below, the tiles are divided into equal-sized squares (“uniformly sized geometric shapes”).



Ex.1006, Fig. 7a (partial, annotated); Ex.1003, ¶ 99.

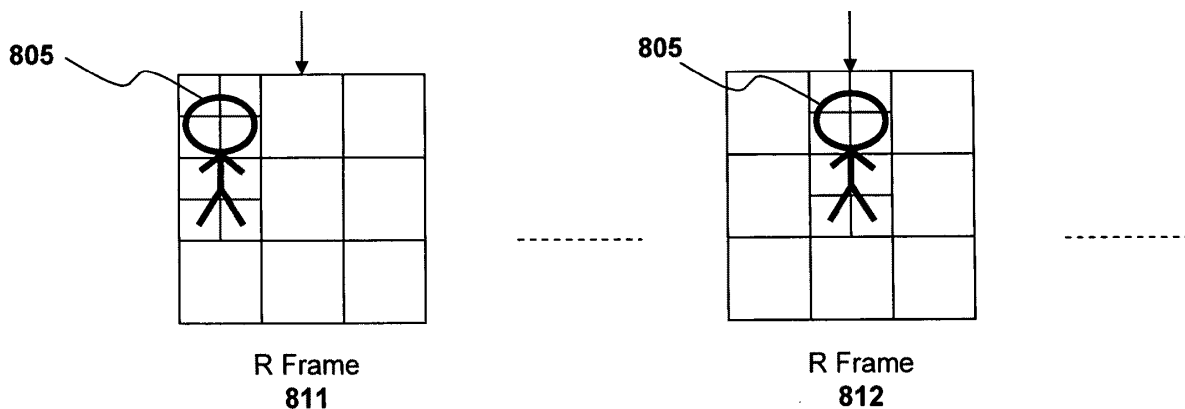
Thus, because Perlman describes an example in which the tiles are equal-sized squares, Perlman renders obvious “wherein the frame segments are uniformly sized geometric shapes” as claimed. Ex.1003, ¶¶ 98-100.

6. Claim 5

[5.1] *A method according to claim 1, wherein at least one of frame segment size and shape is different for one of the frame segments compared to another of the*

frame segments.

While Perlman provides an example in which the tiles are equal-sized, Perlman discloses that the tiles may have various sizes and shapes. “Note that the tiles need not be square, nor must the video frame be square, or even rectangular. The tiles can be broken up into whatever shape best suits the video stream and the application used.” Ex.1006, [0174]. “As mentioned above, various different sizes and shapes may be employed as ‘tiles’ while still complying with these underlying principles.” Ex.1006, [0178]. Perlman further provides an example in which different tiles are different sizes, as shown in Fig. 8 below.



Ex.1006, Fig. 8 (partial).

Perlman explains that “to improve resolution of certain regions of the video stream, the video compression logic 404 uses smaller tiles to encode areas of the video stream with relatively more scene complexity and/or motion than areas of the video stream with relatively less scene complexity and/or motion.” Ex.1006, [0178]. Indeed, it was known in the art to use variously sized and shaped slices.

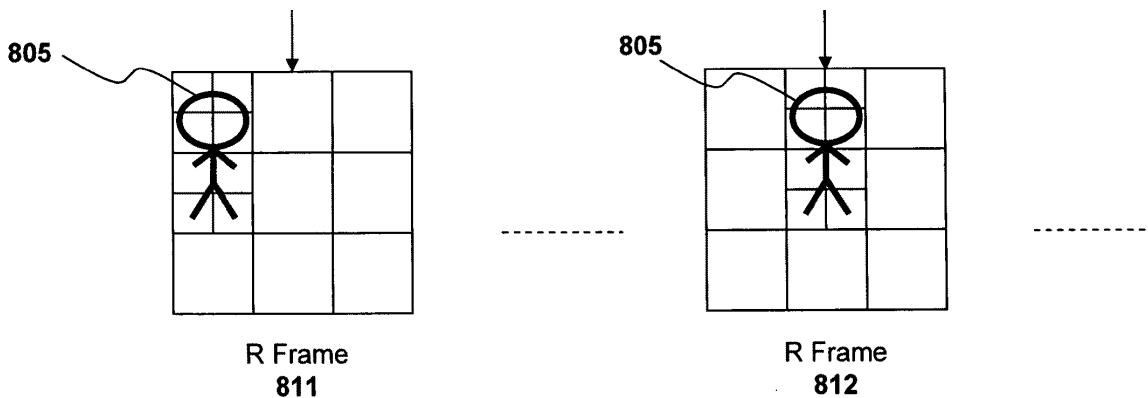
See e.g. Ex.1008, Fig. 3A, 3B; Ex.1003, ¶ 102.

Thus, because Perlman provides examples of using various shapes and sizes as known in the art, Perlman renders obvious “*wherein at least one of frame segment size and shape is different for one of the frame segments compared to another of the frame segments*” as claimed. Ex.1003, ¶¶ 101-03.

7. Claim 7

[7.1] A method according to claim 1, wherein the frame segments are determined in dependence upon external application data and indicative of the amount of motion that is expected in different areas of the video frames.

First, Perlman discloses that an uncompressed frame may be divided into tiles based on where there is expected to be more scene complexity (“*amount of motion that is expected in different areas of the video frames*”). Perlman provides an example in which “the video compression logic 404 uses smaller tiles to encode areas of the video stream with relatively more scene complexity and/or motion than areas of the video stream with relatively less scene complexity and/or motion.” Ex.1006, [0178]. “For example, as illustrated in FIG. 8, smaller tiles are employed around a moving character 805 in one area of one R frame 811 (potentially followed by a series of R frames with the same tile sizes (not shown)).” Ex.1006, [0178]. “Then, when the character 805 moves to a new area of the image, smaller tiles are used around this new area within another R frame 812, as illustrated.” Ex.1006, [0178].



Ex.1006, Fig. 8 (partial).

Perlman also gives an example in which the frames are encoded based on where there is expected to be more or less motion using pre-configured settings. “[T]he low-latency video compression logic 404 in FIGS. 7a and 7b controls the allocation of bits to the various tiles in the R frames either by **being pre-configured by settings, based on known characteristics of the video sequence to be compressed**, or automatically, based upon an ongoing analysis of the image quality in each tile.” Ex.1006, [0177]. Perlman provides a specific example related to a racing game. “[I]n some racing video games, the front of the player's car (which is relatively motionless in the scene) takes up a large part of the lower half of the screen, whereas the upper half of the screen is entirely filled with the oncoming roadway, buildings and scenery, which is almost always in motion.” Ex.1006, [0177]. A POSITA would have found it obvious that in the context of video conferencing (which Perlman describes as discussed at [1.0]), there would be

more motion around a user's face and less motion in the user's background.

Ex.1003, ¶ 106.

Second, Perlman describes dividing the frame into tiles and encoding the tiles ("*the frame segments are determined*") based on pre-configured settings ("*external application data*"). Data from a game, teleconference, or movie application that indicates where a moving character is currently at within the frame is "*application data*" that is "*external*" to Perlman's compression logic. This data may include for example, "pre-configured by settings, based on known characteristics of the video sequence to be compressed." Ex.1006, [0177].

Perlman describes dividing frames based on expected motion similar to how the '426 patent describes dividing frames based on external application data: "Optionally, **the segments are defined based upon knowledge that is provided for**, or determined on the basis of **usage of a particular videoconferencing or telepresence system**." Ex.1001, 7:30-44.

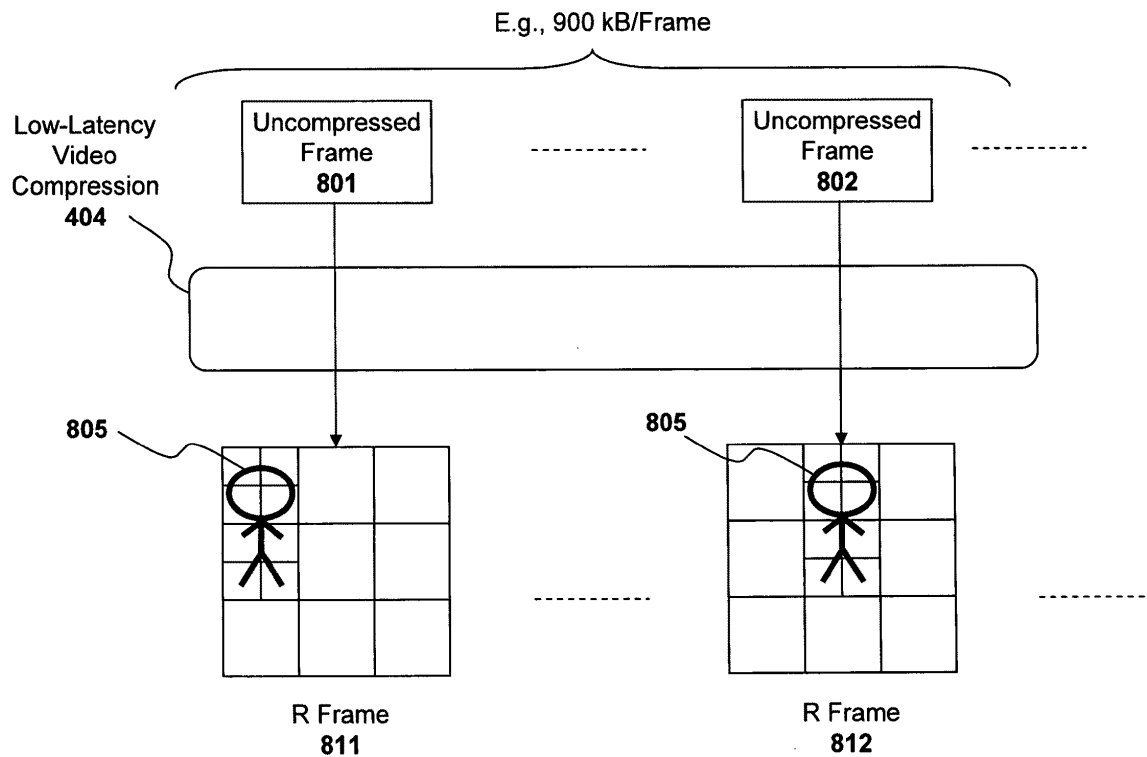
Thus, because Perlman describes dividing and encoding frame tiles based on pre-configured settings of known characteristics (or character movement) that indicates where expected motion will take place, Perlman renders obvious "*wherein the frame segments are determined in dependence upon external application data and indicative of the amount of motion that is expected in different areas of the video frames*" as claimed. Ex.1003, ¶¶ 104-09.

8. Claim 8

[8.1] *A method according to claim 1, wherein the frame segments are determined in dependence upon external prior use data indicative of the amount of motion previously determined in different areas of the video frames.*

First, as discussed above at [7.1], a POSITA would have found it obvious to divide a frame into multiple tiles based on where there is expected to be more scene complexity and motion (“*data indicative of the amount of motion*”).

Ex.1003, ¶ 110. Perlman describes a technique in which a frame is divided up based on the motion within the video sequence. In Perlman’s example, “the video compression logic 404 uses smaller tiles to encode areas of the video stream with **relatively more scene complexity and/or motion** than areas of the video stream with **relatively less scene complexity and/or motion**.” Ex.1006, [0178]. “For example, as illustrated in FIG. 8, smaller tiles are employed around a moving character 805 in one area of one R frame 811 (potentially followed by a series of R frames with the same tile sizes (not shown)).” Ex.1006, [0178]. “Then, when the character 805 moves to a new area of the image, smaller tiles are used around this new area within another R frame 812, as illustrated.” Ex.1006, [0178].



Ex.1006, Fig. 8.

Accordingly, by tracking motion within the frame and using that tracking to divide the frame into tiles, Perlman teaches dividing frames based on prior use data. A POSITA would have found it obvious that to sense motion within the frame, an application would determine where the object has been (prior use) to compare with where the object is now. Ex.1003, ¶ 111. In other words, changes in motion involve both past and present frames and thus include “*prior use data indicative of the amount of motion previously determined in different areas of the video frames.*” Again, data from a game, teleconference, or movie application that indicates where a moving character is currently at within the frame is “*application*

data” that is “*external*” to Perlman’s compression logic.

Additionally, as described above at [7.1], Perlman describes dividing and encoding the tiles (“*frame segments are determined*”) based on pre-configured settings (“*prior use data indicative of the amount of motion previously determined in different areas of the video frames*”). For example, Perlman explains that the encoding of tiles may be based on “pre-configured by settings, based on known characteristics of the video sequence to be compressed.” Ex.1006, [0177]. As an example of known characteristics, Perlman describes a racing game in which “the front of the player’s car (which is relatively motionless in the scene) takes up a large part of the lower half of the screen, whereas the upper half of the screen is entirely filled with the oncoming roadway, buildings and scenery, which is almost always in motion.” Ex.1006, [0177]. The data from the game that indicates where motion is expected (based on known characteristics) is “*external*” to the compression logic. Accordingly, data associated with a game, teleconference, or movie application that indicate where there is more scene complexity or motion is “*application data*” that is “*external*” to Perlman’s compression logic. Ex.1003, ¶ 112.

Thus, because Perlman describes using various techniques to divide a frame into tiles based on where motion is detected (comparing past and current position of an object) and encoding the tiles based on pre-configured settings, Perlman

renders obvious “*wherein the frame segments are determined in dependence upon external prior use data indicative of the amount of motion previously determined in different areas of the video frames*” as claimed. Ex.1003, ¶¶ 110-13.

9. Claim 9

[9.1] A method according to claim 1, wherein a period between regularly spaced I-frame segment instances for one of the frame segments is different than a period between regularly spaced I-frame segment instances for another of the frame segments.

Perlman explains that an I-frame occurs for each tile at regular intervals (“*a period between regularly spaced I-frame segment instances*”), for example, once every sixteen frames: “the 16 tiles that make up an I frame cycle through in 16 frames times,” so for each tile “an I frame is introduced every 16 frame times.” Ex.1006, [0172]. Perlman also contemplates other regular intervals, such as “an I tile generated only every other frame.” Ex.1006, [0175].

Perlman explains that certain tiles may have I-frames generated more frequently, for example in image areas involving more movement or scene changes. “[C]ertain tiles may be sequenced with I tiles more often than other tiles if, for example, certain areas of the screen are known to have more motion requiring from frequent I tiles, while others are more static (e.g., showing a score for a game) requiring less frequent I tiles.” Ex.1006, [0175]. Accordingly, if some tiles produce I-frames more frequently due to greater motion, then some tiles will

have different periods between the I-frames than others.

Thus, because Perlman describes a dividing technique in which some tiles have more frequent I-frames than others, Perlman renders obvious “*wherein a period between regularly spaced I-frame segment instances for one of the frame segments is different than a period between regularly spaced I-frame segment instances for another of the frame segments*” as claimed. Ex.1003, ¶¶ 114-17.

10. Claim 10

[10.1] *A method according to claim 1, wherein the frame segments are encoded in encoders having different encoding characteristics.*

Perlman teaches that different tiles may be compressed at different bit rates (“*different encoding characteristics*”). In Perlman’s example, “the low-latency video compression logic 404 in FIGS. 7a and 7b controls the allocation of bits to the various tiles in the R frames either by being pre-configured by settings, based on known characteristics of the video sequence to be compressed, or automatically....” Ex.1006, [0177]. Perlman provides a specific example related to a racing game. “[I]n some racing video games, the front of the player’s car (which is relatively motionless in the scene) takes up a large part of the lower half of the screen, whereas the upper half of the screen is entirely filled with the oncoming roadway, buildings and scenery, which is almost always in motion.” Ex.1006, [0177]. In such a scenario, Perlman suggests configuring the video compression

logic 404 “to allocate more bits to the tiles in the top of the screen than to tiles at the bottom of the screen.” Ex.1006, [0177]. Accordingly, because some tiles have more allocated bits, they are encoded at a higher bit rate (“*different encoding characteristics*”).

Perlman also indicates that the compression logic may use different processors or other hardware components (“*encoders*”). *See supra*, [1.3].

Thus, because Perlman describes using encoders to allocate more bits to the tiles at the top of the screen than those at the bottom, Perlman renders obvious “*wherein the frame segments are encoded in encoders having different encoding characteristics*” as claimed. Ex.1003, ¶¶ 118-20.

11. Claim 11

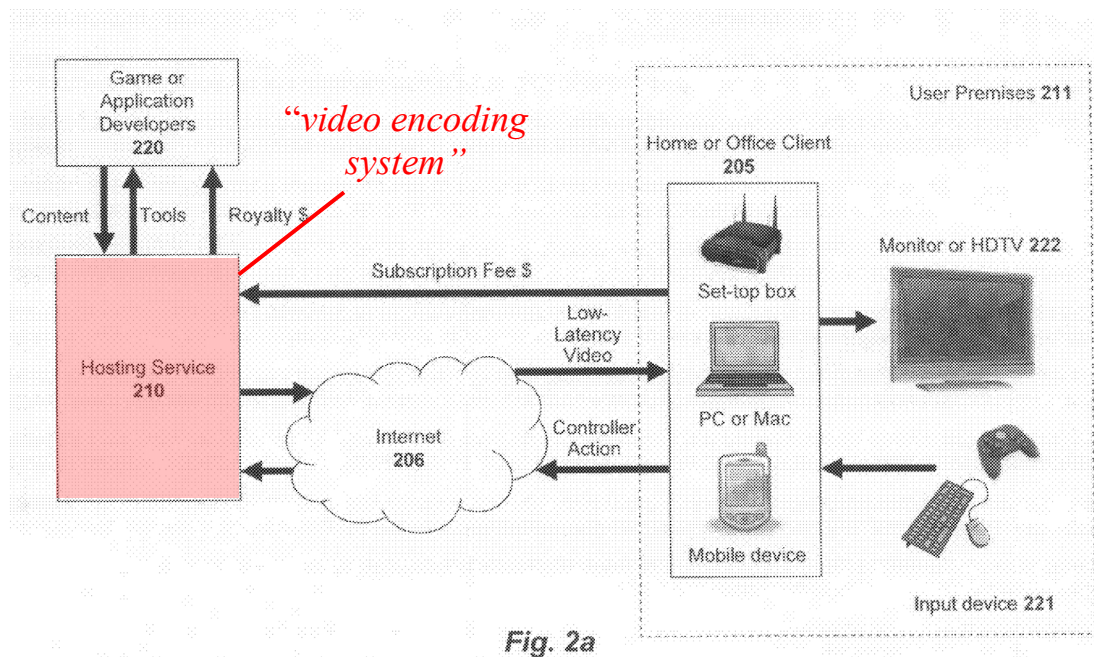
[11.1] *A method according to claim 10, wherein at least one of the encoders has a bitrate that is greater than a bitrate of the other encoders.*

As described above at [10.1], Perlman describes a technique in which some tiles are allocated more bits, and are thus encoded at a higher bitrate. Thus, Perlman renders obvious “*wherein at least one of the encoders has a bitrate that is greater than a bitrate of the other encoders*” as claimed. Ex.1003, ¶ 121.

12. Claim 12

[12.0] *A video encoding system for encoding in real time a video signal that is associated with one of a videoconference and a telepresence session, the system comprising:*

First, Perlman describes a hosting service 210 for compressing real-time video signals (“*video encoding system for encoding in real time a video signal*”). The hosting service 210 “executes the gaming program code in response and generates successive frames of video output (a sequence of video images) for the game or **application software**.” Ex.1006, [0103]. “This sequence of video images is then compressed using a low-latency video compressor, and the hosting service 210 then transmits the low-latency video stream through the Internet 206.” Ex.1006, [0103].



Ex.1006, Fig. 2a (annotated); Ex.1003, ¶ 122.

As explained above at [1.0], compressing video data is “*encoding*” video data as claimed. As also explained above at [1.0], Perlman’s video streams are “*real time*” streams as claimed.

Second, for the reasons explained above at [1.0], Perlman explains that its techniques performed by the encoding system would be used for video conferencing or as part of a video game system that includes video conferencing (“*a video signal that is associated with a videoconference or telepresence session*”). See, e.g., Ex.1006, [0259].

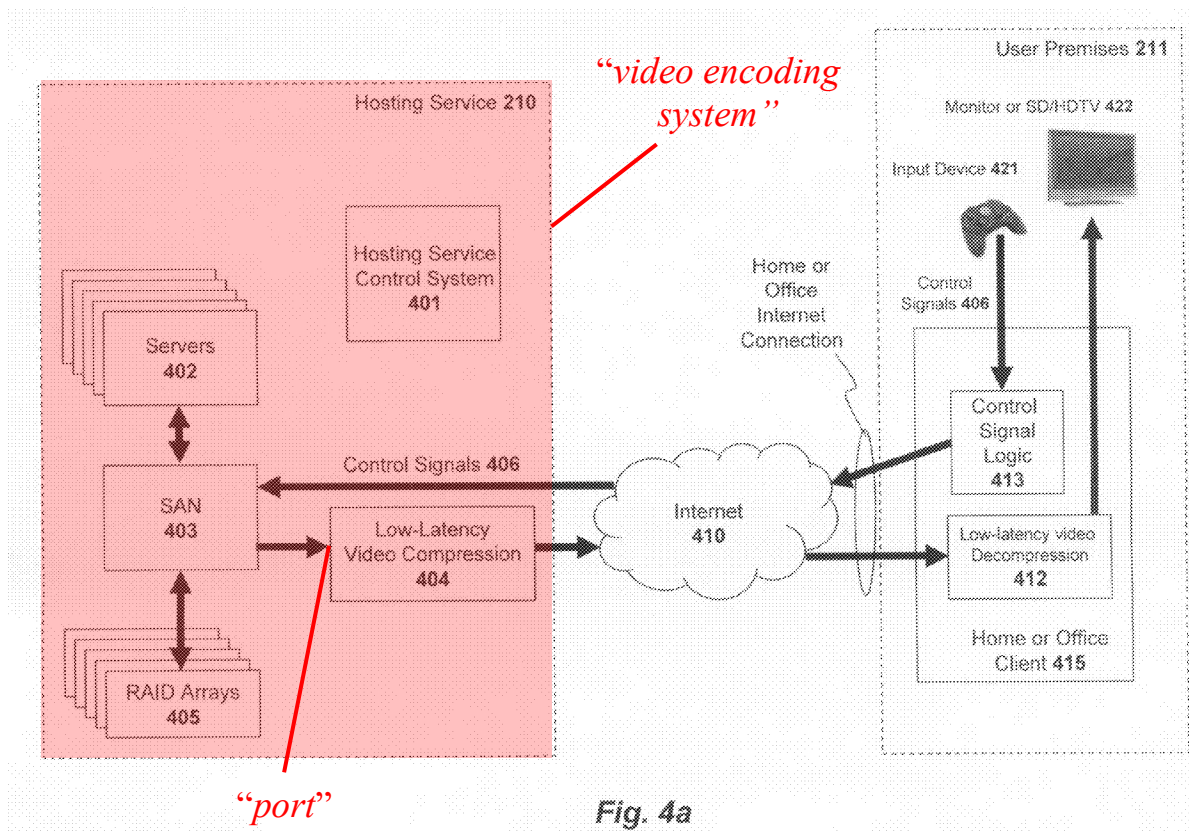
Thus, because Perlman describes a system for compressing video data using a codec, and that video stream would be used for video conferencing, Perlman renders obvious a “*video encoding system for encoding in real time a video signal that is associated with one of a videoconference and a telepresence session*” as recited in the claim. Ex.1003, ¶¶ 122-25.

[12.1] a port for receiving a stream of video data comprising a series of sequential video frames;

First, Perlman’s hosting service 210 (“*system*”) includes compression logic 404 that receives the graphical and audio output of the executed game or application software (“*a stream of video data*”). See *supra*, [1.1]. “[T]he hosting service 210 employs low-latency video compression logic 404 described in detail below.” Ex.1006, [0116]. The compression logic 404 receives “graphical and audio output” and “compresses the sequences of video images according to low-latency video compression techniques, such as those described herein.” Ex.1006, [0117]. For the reasons explained above at [1.1], Perlman’s video stream “*compris[es] a*

series of sequential video frames.”

Second, a POSITA would have understood that the physical component that receives the input video is a “port” within the commonly understood meaning of the term. *See e.g.*, Ex.1009, 844 (“Port ... An interface point connecting a communications channel and a device.”); Ex.1003, ¶ 128.



Ex.1006, Fig. 4a (annotated); Ex.1003, ¶ 128.

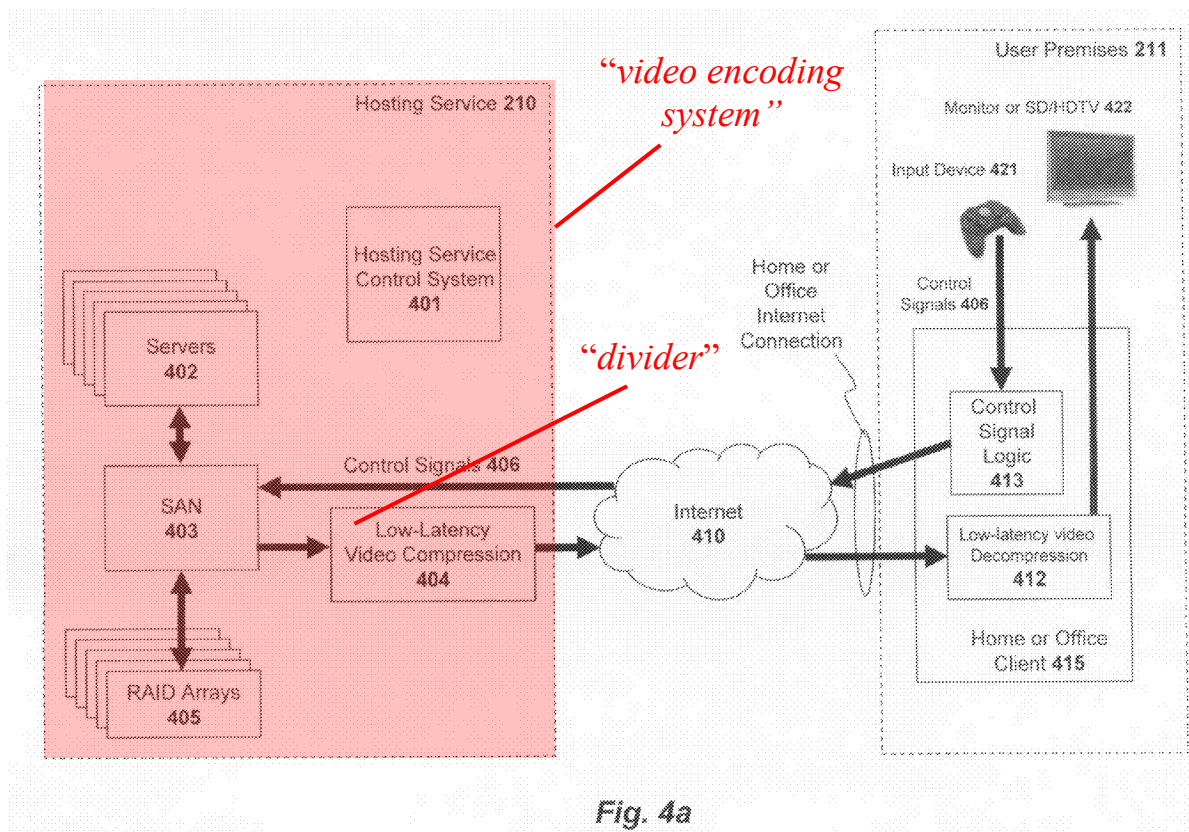
Thus, because Perlman describes a system that includes compression logic that receives audio and video output from an application, Perlman renders obvious “a port for receiving a stream of video data comprising a series of sequential video

frames” as claimed. Ex.1003, ¶¶ 126-29.

[12.2] a divider for dividing the stream of video data into a series of substreams of data corresponding to frame segments, wherein each frame segment represents a part of a complete video frame;

First, for the reasons explained above at [1.2]-[1.4], Perlman’s compression logic 404 divides the frame into a sequence of separate tiles (“*series of substreams of data corresponding to frame segments*”) and encodes each substream to form parallel encoded streams.

Second, a POSITA would have understood that the hardware/software functionality of compression logic 404 that divides the frames into tiles teaches a “*divider*” as claimed. Ex.1003, ¶131. The output of the divider—and input to the encoder—would be the sequences of separate tiles that correspond to the “*series of substreams of data corresponding to frame segments.*” See, e.g., Ex.1005, Fig.12.



Ex.1006, Fig. 4a (annotated); Ex.1003, ¶131.

Thus, because Perlman’s system includes compression hardware and software that divides a video stream into separately encoded streams, Perlman renders obvious “a divider for dividing the stream of video data into a series of substreams of data corresponding to frame segments, wherein each frame segment represents a part of a complete video frame” as claimed. Ex.1003, ¶¶ 130-32.

[12.3] a plurality of video encoders arranged in parallel for independently encoding the respective frame segments to generate parallel encoded data streams, one for each of the plurality of frame segments,

For the reasons described above at [1.3] Perlman describes encoding the tiles

in parallel, which may be done using several processors, FPGAs, or other hardware or software components (“*plurality of video encoders*”). For example, Perlman describes using multiple parallel encoders like Cores 1201 through 1208 shown in Figure 12, below. Ex.1006, [0212]-[0213].

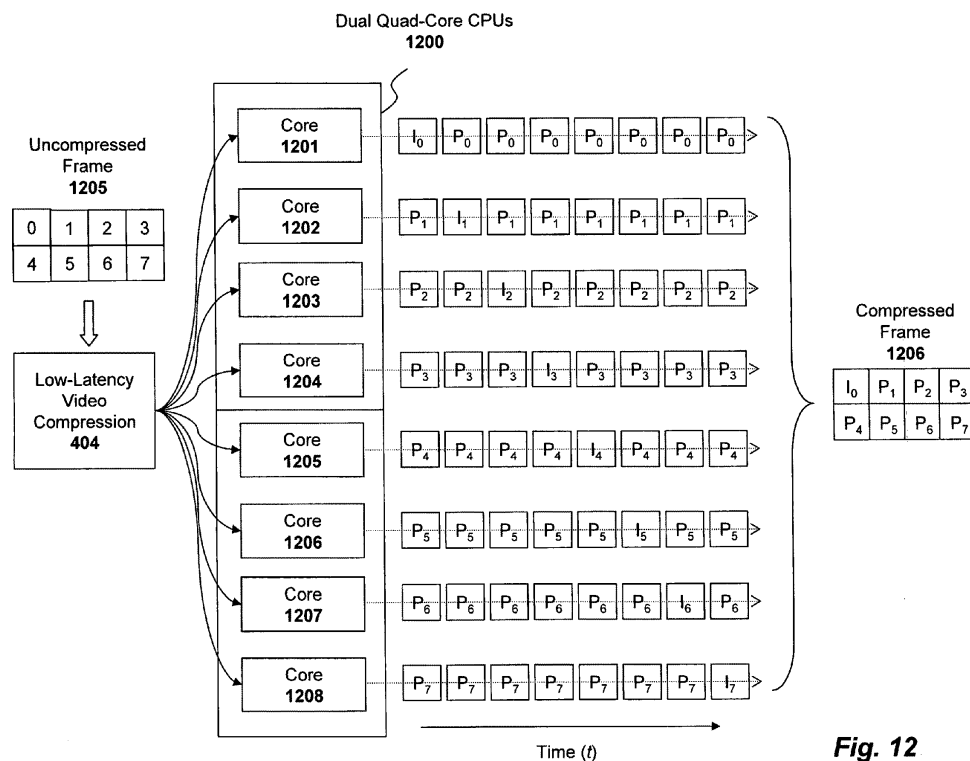


Fig. 12

Ex.1006, Fig. 12.

Thus, because Perlman describes encoding each tile in parallel to create parallel encoded streams, one for each tile, Perlman renders obvious “a *plurality of video encoders arranged in parallel for independently encoding the respective frame segments to generate parallel encoded data streams, one for each of the plurality of frame segments*” as claimed. Ex.1003, ¶¶ 133-34.

[12.4] each encoded data stream containing I-frame segment instances that carry complete information about a particular frame segment and P-frame segment instances that carry partial information about a particular frame segment, the partial information representing information that has changed relative to a previous frame segment instance in that encoded data stream, and

This limitation is substantially similar to [1.5]. Thus, for the reasons explained above at [1.5], Perlman renders this limitation obvious. Ex.1003, ¶ 135.

[12.5] the video encoders being arranged to encode each frame segment such that when an instance of one of the frame segments within a particular frame is an I-frame segment instance, at least one of the other frame segment instances in that particular frame is a P-frame segment instance,

This limitation is substantially similar to that of [1.6]. As discussed above at [12.4], Perlman’s system includes video encoders. Thus, for the reasons explained above at [12.4] and [1.6], Perlman renders obvious “*the video encoders being arranged to encode each frame segment such that when an instance of one of the frame segments within a particular frame is an I-frame segment instance, at least one of the other frame segment instances in that particular frame is a P-frame segment instance*” as claimed. Ex.1003, ¶ 136.

[12.6] whereby in the respective parallel encoded data streams in any period of time when an I-frame segment instance is present in one of the parallel encoded data streams, a P-frame segment instance is present in at least one of the other parallel encoded data streams; and

This limitation is substantially similar to [1.7]. Thus, for the reasons explained above at [1.7], the prior art renders this limitation obvious. Ex.1003, ¶ 137.

[12.7] *a transceiver for transmitting the parallel encoded data streams independently over a network to a receiver to permit the receiver to decode each frame segment and recombine the decoded frame segments to reproduce an image;*

First, for the reasons explained above at [1.8], Perlman describes transmitting the encoded video data over a network. Perlman’s system “transmits a compressed video stream, typically with compressed or uncompressed audio, back to the client 415 over the Internet 410 .” Ex.1006, [0117]. A POSITA would have found it obvious to transmit the data using a “*transceiver*.” Ex.1003, ¶ 139. Perlman provides an explicit example in which a network interface “may be a Cable or DSL modem, a WiMax transceiver, a Fiber transceiver.” Ex.1006, [0143]. Moreover, a POSITA would have found it obvious to use network interface devices that both receive and transmit data because videoconferencing involves two-way communication. Ex.1003, ¶ 139.

Second, for the reasons explained above at [1.8], Perlman’s transmitter is “*for transmitting the parallel encoded data streams independently over a network to a receiver to permit the receiver to decode each frame segment and recombine the decoded frame segments to reproduce an image.*”

Thus, because Perlman describes a network interface such as a transceiver to transmit the independently encoded streams over a network for recombination by the receiver, Perlman renders obvious “*a transceiver for transmitting the parallel*

encoded data streams independently over a network to a receiver to permit the receiver to decode each frame segment and recombine the decoded frame segments to reproduce an image” as claimed. Ex.1003, ¶¶ 138-40.

[12.8] *wherein during transmission of I-frame segment instances of one of the frame segments, P-frame segment instances of all other frame segments are transmitted.*

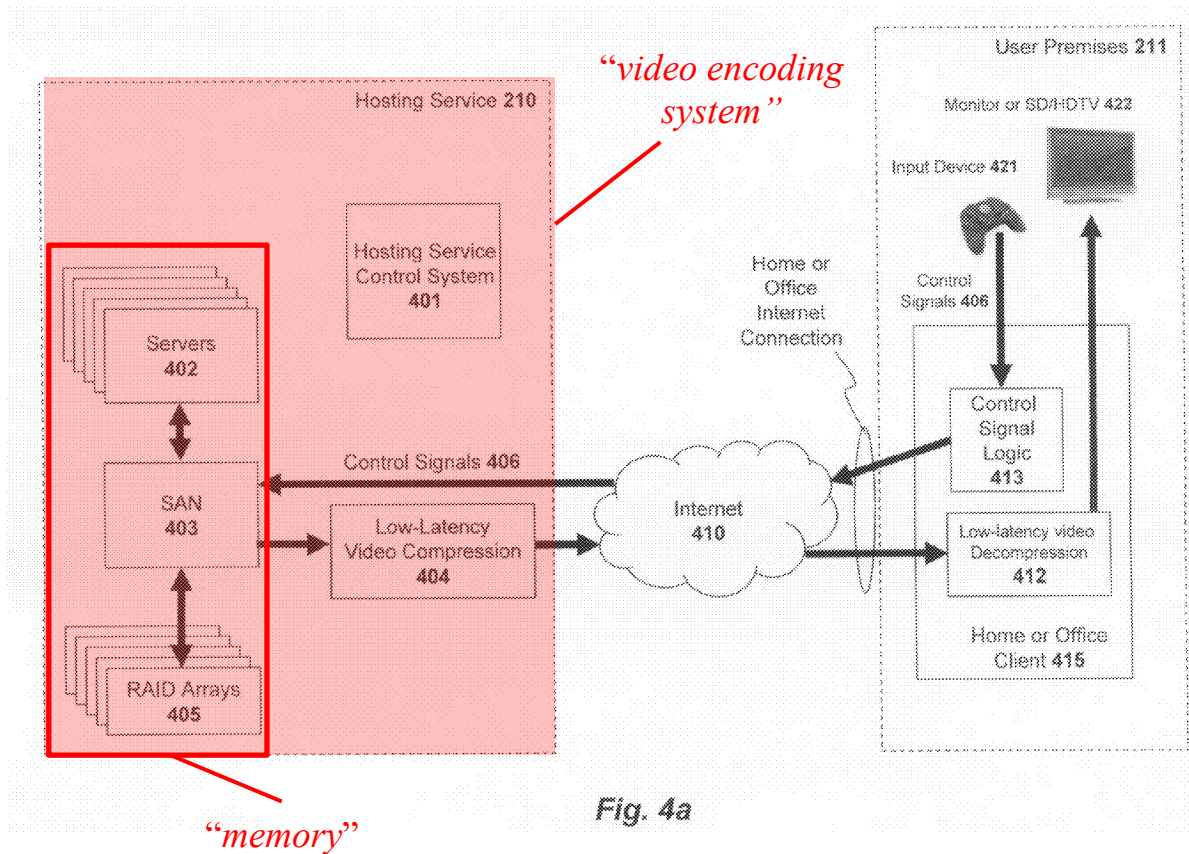
This limitation is substantially similar to [1.9]. Thus, for the reasons explained above at [1.9], the prior art renders this limitation obvious. Ex.1003, ¶ 141.

13. Claim 13

[13.1] *A video encoding system according to claim 12 comprising: a memory for storing predetermined data defining the frame segments, which correspond to predetermined parts of the video frames.*

First, for the reasons explained above at [2.1] and [3.1], Perlman describes dividing a frame into a pre-determined arrangement (i.e., a four-by-four grid).

Second, Perlman’s hosting service 110 (“*system*”) includes various types of memory, including “Flash memory,” “Storage Area Network (SAN) 403 connected to a series of Redundant Arrays of Independent Disks (RAID) 405,” and “RAM.” Ex.1006, [0115].



Ex.1006, Fig. 4a (annotated); Ex.1003, ¶ 143.

Third, a POSITA would have found it obvious to store any data used by the compression logic in such memory. Indeed, Perlman explains that the techniques it discloses may be embodied on machine readable media. Ex.1003, ¶ 144. “Elements of the disclosed subject matter may also be provided as a machine-readable medium for storing the machine-executable instructions.” Ex.1006, [0358].

Thus, because Perlman’s system includes a memory which may be used to stored data and machine-readable instructions related to dividing a frame into a predetermined four-by-four tile grid, Perlman renders obvious “a memory for

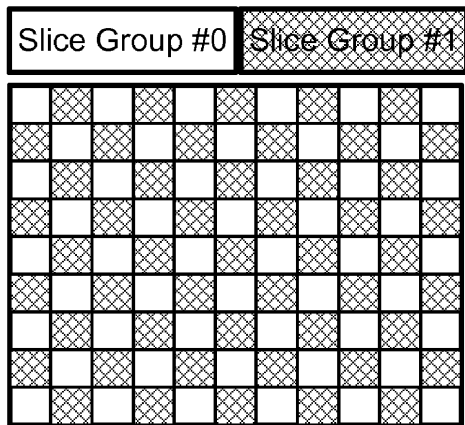
storing predetermined data defining the frame segments, which correspond to predetermined parts of the video frames.” Ex.1003, ¶¶ 142-45.

C. Ground 2: Claim 6 is obvious in view of Perlman and Oguz.

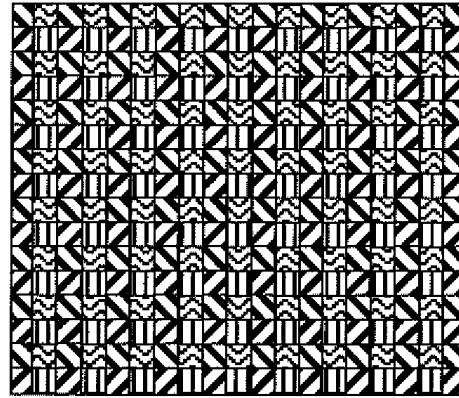
1. Summary of Oguz

Oguz “relates generally to video encoding and, more specifically, to techniques for content adaptive video frame slicing and non-uniform access unit coding for improved coding efficiency.” Ex.1008, [0003]. Oguz explains that “[m]ultimedia processing systems, such as video encoders, may encode multimedia data using encoding methods based on international standards such as Moving Picture Experts Group (MPEG)-1, -2 and -4 standards, the International Telecommunication Union (ITU)-T H.263 standard, and the ITU-T H.264 standard and its counterpart, ISO/IEC MPEG-4, Part 10, i.e., Advanced Video Coding (AVC).” Ex.1008, [0042].

Like both Perlman and the ’426 patent, Oguz explains that “each frame or field may further include one or more slices, or sub-portions of the frame or field.” Ex.1008, [0043]. Oguz illustrates various “slice partitioning samples.” For example, Oguz illustrates an example where the frame slices are in a checkerboard pattern, as shown below in comparison with a similar checkerboard example from the ’426 patent.



Ex.1008, Fig. 4B.



Ex.1001, Fig. 5 (partial)

Oguz illustrates “[i]n FIG. 4B, the frame 300 includes a checker-board pattern for improved error resilience and concealment.” Ex.1008, [0065]. “Thus, the checker-board pattern essentially provides scattered slices.” Ex.1008, [0065]. Oguz explains that this technique is applicable to the H.264 codec. *See* Ex.1008, [0066].

Thus, Oguz provides evidence that it was known to divide a frame into a checkerboard pattern in which a particular slice group (“*segment*”) represents a non-contiguous area. Ex.1003, ¶¶ 146-48.

2. Reasons to Combine Oguz with Perlman

A POSITA implementing Perlman’s frame dividing techniques would have found it obvious to do so in a variety of manners such as those described by either Perlman or Oguz. For example, a POSITA would have found it obvious to implement Perlman’s frame slicing method so that particular slices are non-

contiguous as taught by Oguz. Ex.1003, ¶ 149.

While Perlman provides an example in which a video frame is divided into a four-by-four grid of 16 tiles, Perlman explains that other slice geometries may be used. For example, Perlman states that “various different sizes and shapes may be used for the tiles while still complying with these underlying principles.” Ex.1006, [0169]. “Note that the tiles need not be square, nor must the video frame be square, or even rectangular. The tiles can be broken up into whatever shape best suits the video stream and the application used.” Ex.1006, [0174]. Ex.1003, ¶ 150.

Accordingly, a POSITA would have found it obvious to divide a frame according to any known, suitable alternatives to the four-by-four grid described by Perlman. Oguz provides one example of such an alternative by placing the tiles in a checkerboard pattern: “the frame 300 includes a checker-board pattern for improved error resilience and concealment.” Ex.1008, [0065]. Ex.1003, ¶ 151.

A POSITA would have had a reasonable expectation of success in combining the teachings of Oguz with those of Perlman. Oguz explains that the checkerboard pattern is useful for the H.264 codec. Ex.1008, [0066]. Perlman lists the H.264 codec as one example of a codec that may be used with its teachings. Ex.1006, [0156] (listing “types of compression, **such as H.264**, or Windows Media VC9, MPEG2 and MPEG4.”). Accordingly, when using the H.264 codec, a POSITA would have had expected success in dividing a frame into two segments

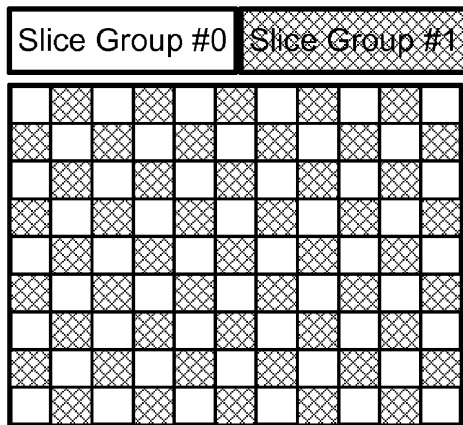
positioned in a checkerboard pattern. Ex.1003, ¶ 152.

Thus, the combination of Oguz with Perlman is an example of using a known technique (checkerboard pattern) to improve similar methods (Perlman's frame dividing technique) in the same way (using the H.264 codec). Ex.1003, ¶ 153.

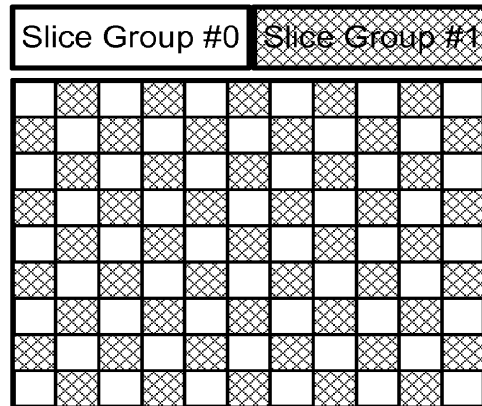
3. Claim 6

[6.1] *A method according to claim 1, wherein at least one of the frame segments comprises a plurality of predetermined non-contiguous areas of the video frames.*

A POSITA would have found it obvious to divide the video frame into two slices put into a checkerboard pattern as evidenced by Oguz. Oguz illustrates “[i]n FIG. 4B, the frame 300 includes a checker-board pattern for improved error resilience and concealment.” Ex.1008, [0065]. “Thus, the checker-board pattern essentially provides scattered slices.” Ex.1008, [0065]. Fig. 4B is illustrated below in comparison with a similar pattern from the '426 patent.



Ex.1008, Fig. 4B.



Ex.1001, Fig. 5 (partial)

When the frame is divided into such a pattern, the two different “tiles” comprise “*a plurality of predetermined non-contiguous areas of the video frames.*” For the reasons explained above at X.C.2, a POSITA would have found it obvious to implement Perlman’s techniques in a variety of ways, including a checkerboard pattern as described by Oguz. Ex.1003, ¶ 155.

Thus, because Perlman describes dividing a frame into a variety of sizes and shapes, and Oguz provides an example of dividing a frame into a checkerboard pattern, Perlman and Oguz render obvious “*wherein at least one of the frame segments comprises a plurality of predetermined non-contiguous areas of the video frames*” as claimed. Ex.1003, ¶¶ 154-56.

XI. CONCLUSION

Accordingly, Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

Respectfully submitted,

Dated: March 15, 2024
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, Texas 75201
Customer No. 27683

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

XII. MANDATORY NOTICES

A. Real Party-in-Interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner certifies that the real party-in-interest is Cisco Systems, Inc.

B. Related Matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of the Petitioner, the '426 patent is involved in the following case:

Case Heading	Number	Court	Date
<i>Video Solutions Pte. Ltd. v. Cisco Systems, Inc.</i>	EDTX-2-23-cv-00222	EDTX	May 19, 2023

C. Lead and Back-up Counsel and Service Information

Lead Counsel

Theodore M. Foster
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (303) 382-6205
Fax: (214) 200-0853
ipr.theo.foster@haynesboone.com
USPTO Reg. No. 57,456

Back-up Counsel

David L. McCombs
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (214) 651-5533
Fax: (214) 200-0853
david.mccombs.ipr@haynesboone.com
USPTO Reg. No. 32,271

Calmann J. Clements
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (972) 739-8638
Fax: (214) 200-0853
calmann.clements.ipr@haynesboone.com
USPTO Reg. No. 66,910

Please address all correspondence to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and in reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 12,510.

Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: March 15, 2024

/Theodore M. Foster/

Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105, service was made on Patent Owner as detailed below.

Date of service March 15, 2024

Manner of service FEDERAL EXPRESS

Documents served Petition for *Inter Partes* Review Under 35 U.S.C. § 312 and 37 C.F.R. § 42.104 of U.S. 8,649,426; Petitioner's Exhibit List; All Exhibits; Petitioner's Power of Attorney.

Persons served Marks & Clerk
180 Kent Street, Suite 1900
Ottawa, ON K1P 0B6
CANADA

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456