

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMAZON.COM, INC., AMAZON.COM SERVICES LLC,

Petitioner,

v.

NOKIA TECHNOLOGIES OY,

Patent Owner.

IPR2024-00626
U.S. Patent No. 11,805,267

PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 11,805,267

TABLE OF CONTENTS

	<u>Page</u>
EXHIBIT LIST	III
I. BACKGROUND	1
A. Motion Prediction.....	1
B. Precision and Bit Shifting	5
II. THE '267 PATENT	9
A. Overview	9
B. Prosecution History	11
C. Priority Date	11
III. LEVEL OF ORDINARY SKILL IN THE ART.....	11
IV. CLAIM CONSTRUCTION	12
A. “precision”	12
V. IDENTIFICATION OF CHALLENGE	12
A. Ground 1	13
1. Walker (Ex-1004)	13
2. Independent Claims 1, 7, and 13.....	15
3. Dependent Claims 2, 8, and 14	27
4. Dependent Claims 3, 9, and 15	29
5. Dependent Claims 4, 10, and 16	31
6. Dependent Claims 5, 11, and 17	33
7. Dependent Claims 6, 12, and 18	36
B. Ground 2.....	37
1. Karczewicz-I (Ex-1005).....	37
2. Karczewicz-II (Ex-1006)	37
3. Motivation to Combine	38
4. Independent Claims 1, 7, and 13.....	53
5. Dependent Claims 2, 8, and 14	75
6. Dependent Claims 3, 9, and 15	79

7.	Dependent Claims 4, 10, and 16	81
8.	Dependent Claims 5, 11, and 17	83
9.	Dependent Claims 6, 12, and 18	85
VI.	DISCRETIONARY ANALYSIS FOR REVIEW.....	86
A.	<i>Fintiv</i>	86
B.	<i>Advanced Bionics</i>	86
VII.	STANDING	87
VIII.	MANDATORY NOTICES AND FEES	87

EXHIBIT LIST

Exhibit No.	Description
1001	U.S. Patent No. 11,805,267 to Ugur et al. (“the ’267 Patent”)
1002	Prosecution File History for the ’267 Patent
1003	Declaration of Immanuel Freedman
1004	U.S. Patent Application Publication No. 2005/0281334 (“Walker”)
1005	U.S. Patent Application Publication No. 2011/0007799 (“Karczewicz-I”)
1006	U.S. Patent Application Publication No. 2009/0257499 (“Karczewicz-II”)
1007	Prosecution History for U.S. Patent No. 9,432,693
1008	U.S. Patent No. 9,344,744 (“Kirchhoffer-744”)
1009	Srinivasan, An Overview of VC-1
1010	U.S. Patent Application Publication No. 2003/0112864 (“Karczewicz-864”)
1011	Wiegand, Overview of the H.264/AVC Video Coding Standard
1012	Richardson, The H.264 Advanced Video Compression Standard
1013	U.S. Patent Application Publication No. 2008/0198935 (“Srinivasan-935”)
1014	H.264 Advanced Video Coding for Generic Audiovisual Services (March 2009)
1015	U.S. Patent Application Publication No. 2013/0034158 (“Kirchhoffer-158”)
1016	U.S. Patent No. 8,594,188 (“Demos”)

The '267 patent applies a well-known concept—using higher precision for intermediate calculations—to motion prediction and pixel interpolation for video coding. But this concept was already known and taught by Walker (Ground 1). Additionally, the Karczewicz references (Ground 2) taught pixel interpolation with the same concept—using higher precision for intermediate calculations—for the same calculations, years before the '267 patent.

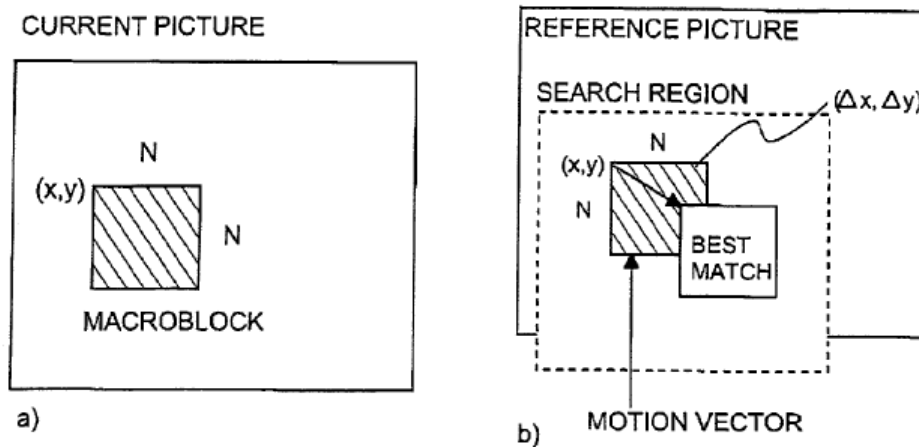
I. BACKGROUND

Video encoding exploited data redundancy to reduce the size of video. Since the 1990s, major video standards employed the same architecture, which divided video frames into blocks of pixels (e.g., squares of 16x16 or 8x8 pixels) and applied motion prediction to those blocks. Subsequent steps produced a video bitstream, and the decoder performed the inverse steps to decode the video. Introduced in 2003, the H.264 standard became a ubiquitous video format employing these techniques. Ex-1003, ¶¶3-4.

A. Motion Prediction

Video blocks were encoded with references to one or more other blocks using predictive coding. Inter-frame prediction used motion estimation and compensation to encode blocks with reference to blocks in other frames, taking advantage of similarities between frames. For example, when an object appeared in successive frames, inter-frame prediction encoded and transmitted information for

a first frame, and encoded subsequent frames by reference to reference blocks in the first frame. A motion vector signaled motion estimation by indicating the displacement of a current (target) block with respect to a reference block, for example indicating that the reference block moved to the right 5 pixels and moved down 3 pixels between frames. Ex-1003, ¶5; Ex-1010, ¶18, Fig. 4:



There were two types of inter-coded blocks: “P” and “B.” P-blocks allowed *unidirectional* prediction based on a single block in another frame, while B-blocks allowed *bidirectional* prediction, meaning blocks could be predicted using two other frames. Ex-1011, 000002, 000007; Ex-1012, 000198-200; Ex-1003, ¶6. For bidirectional prediction, the encoder generated two motion vectors, each pointing to a different block in a different reference frame. Ex-1012, 000194-195, 000200, 000062, 000090; Ex-1003, ¶7. Each pixel of the bi-predicted block was an average (or weighted average) of the two corresponding pixels in the two reference blocks obtained from reference frames. Ex-1003, ¶¶7-8; Ex-1011, 000011; Ex-1012,

000195 (averaging 16x16 reference blocks from List0 and List1 into bi-prediction block):

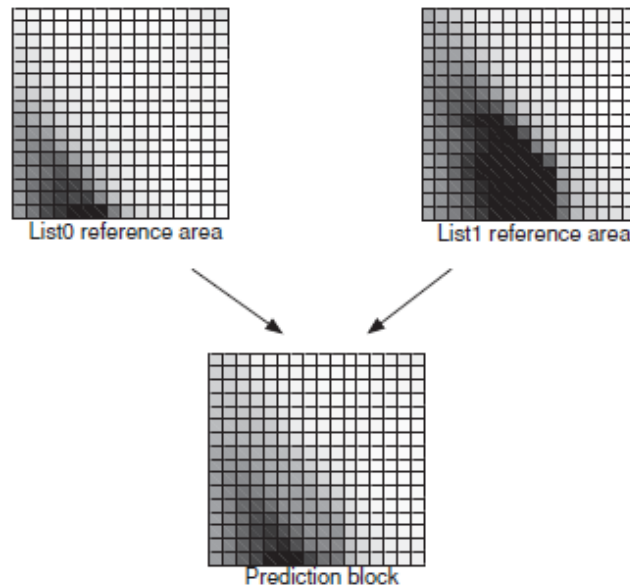


Figure 6.35 Bi-prediction example

The predicted block might not precisely match the target block. Therefore, an encoder calculated the per-pixel difference between the target block and the corresponding pixels in the bidirectionally-predicted block to obtain a block of residual differences. Ex-1012, 000062-63, 000117; Ex-1003, ¶9. To represent a target block, the encoder merely needed to encode and transmit the motion vectors and residual block to the decoder, which used those items to perform the inverse process and reconstruct the target block. *E.g.*, Ex-1011, 000007; Ex-1012, 000059-63, 000074, 000084, 000102, 000123; Ex-1003, ¶¶9-11.

When a motion vector pointed to an integer pixel position, the values of the block at that position were used to generate a predicted block. But H.264 and other

standards allowed motion vectors to have subpixel resolution, where the best match for a target block was not located at an exact integer pixel position, but was instead located at a fractional (subpixel) position, such as when an object moved a half or quarter-pixel between frames. Ex-1011, 000002; Ex-1012, 000184; Ex-1003, ¶12. Integer positions did not require interpolation (because one could just use the integer value). Subpixel positions were interpolated to generate the predicted block. For example, in half-pixel interpolation, the value at a half-pixel position was often calculated as the average of six adjacent integer pixels. Ex-1003, ¶13; Ex-1011, 000010; Ex-1012, 000184-189:

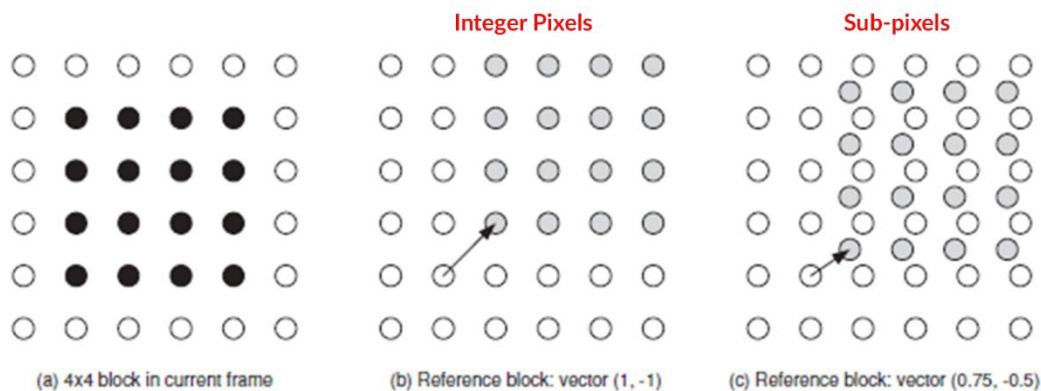


Figure 6.18 Example of integer and sub-pixel prediction

Since bi-prediction used two motion vectors—one for each reference block—it often involved interpolated predictions, based on whether one or both motion vectors point to integer or sub-pixel positions. *See id.*

B. Precision and Bit Shifting

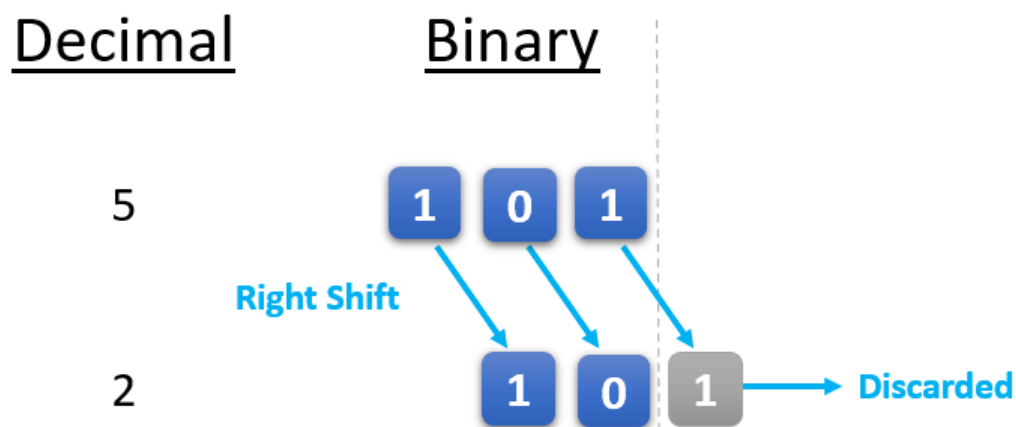
Pixel and prediction values were represented in computers as binary numbers, meaning each digit (called a “bit”) is 0 or 1. The precision of a variable was determined by the number of bits needed to represent its possible values. For example, the value of a pixel on a black & white display can be represented with a single bit having two possible values: 0 (black) and 1 (white). A pixel with 2 bits of precision has $2^2=4$ possible values from 0-3, a pixel with 3 bits of precision has $2^3=8$ possible values from 0-7, and so on. Pixels today may have 8 bits of precision representing $2^8=256$ possible values (from 0-255).

Precision could be changed by mathematical operations. Multiplication and addition can increase the precision of a variable because more bits are required to represent the result. For example, when a 2-bit variable (with possible values from 0-3) is multiplied by two, the possible values of the result could range from 0-6 and thus require 3 bits of precision to account for the largest possible value 6 (110 in binary). Conversely, division and subtraction can reduce the precision of a variable because fewer bits may be required to represent the result. Ex-1003, ¶14.

A widely used way to adjust the precision of a variable represented in binary was by shifting bits to the left or right. A left shift operation added one or more 0-value bits to the right side of the binary number, thereby shifting the existing bits to the left. Ex-1008, 20:42-47; Ex-1013, ¶91, ¶116, ¶131; Ex-1003, ¶16. For

example, the 2-bit binary number 10 (decimal 2) can be left-shifted by one place to produce the 3-bit binary number 100 (decimal 4). A left shift operation thus increases the precision of the variable by adding an extra digit. Left-shifting a binary number is equivalent to multiplying the number by 2 for each bit shifted. Ex-1003, ¶16.

A right-shift operation, in contrast, truncated the variable by essentially chopping off one or more bits on the right side of the binary number (the least significant bits). Ex-1004, ¶46; Ex-1005, ¶57; Ex-1003, ¶15. Right-shifting is mathematically equivalent to dividing the value of the variable by 2 for each bit shifted, and discarding the remainder (because binary numbers are whole integers). For example, the 3-bit binary number 101 (decimal 5) can be right-shifted by one bit to produce the 2-bit binary number 10 (decimal 2):



The right-most shifted bit is simply discarded as shown in the figure. Thus, a right-shift can reduce the precision of a variable from 3 bits to 2 bits. Ex-1003, ¶15.

Right-shifting a binary number rounds the result because the remainder of the division operation is discarded. As a result, a right shift can introduce a rounding error into a calculation. In the example above, the right shift operation applied to the binary number representing 5 produced the binary number representing 2 ($5 \div 2 = 2$). Mathematically, of course, $5 \div 2 = 2.5$, meaning the division by right-shift produced a rounding error of 0.5. As another example, averaging 2 and 3 would normally result in 2.5, but with binary numbers can result in $\frac{2+3}{2} = \frac{5}{2} = 2$ when the division is performed using a right shift. Ex-1003, ¶17.

Sequential calculations involving right-shifted variables could produce erroneous results with compounded rounding errors. For example, if the above average of 2 and 3 is calculated twice and then added together, the expected result would be $2.5 + 2.5 = 5$. Recall that the binary result of averaging 2 and 3 is 2 and the resulting rounding error is 0.5. Accordingly, the binary result of calculating the average twice and adding them together is $2 + 2 = 4$, meaning the rounding error has increased from the 0.5 to 1. Ex-1003, ¶17.

The prior art devised techniques to account for rounding errors produced by right-shifted binary numbers. One technique involved adding a rounding offset to a binary value before rounding. Ex-1011, 000010; Ex-1005, ¶55; Ex-1003, ¶18.

Another technique involved maintaining higher precision intermediate calculations, e.g., by delaying rounding (division/right-shift) operations. *E.g.*, Ex-

1008, 7:4-19; Ex-1003, ¶19. Using the example from above, the division in the averaging operation can be delayed until the end, preventing the rounding error:

$$\frac{(2+3)}{2} + \frac{(2+3)}{2} = \frac{(2+3)+(2+3)}{2} = \frac{5+5}{2} = \frac{10}{2} = 5$$

Ex-1003, ¶17.

Motion prediction and sub-pixel interpolation included multiplication and addition, which increased the number of bits needed to represent intermediate results. For example, when two 8-bit pixels were averaged, they were first added together, resulting in a sum that may require 9 bits to represent. The sum was next divided by two, resulting in an 8-bit number. This division operation could cause rounding errors, which were then compounded when multiple division operations were performed in sequence. Ex-1003, ¶14, ¶17.

Key to this Petition, it was well known in the prior art that the potential problem of rounding errors in mathematical operations involved in video coding could be addressed by maintaining values for intermediate calculations at a higher precision and delaying rounding operations until later. *E.g.*, Ex-1008, 7:4-19 (“An advantage ... is that by predicting and reconstructing in a higher precision than the picture is defined, a more precise prediction and reconstruction can be obtained[.]”), 16:22-17:36; Ex-1009, 000005; Ex-1010, ¶¶61-62; Ex-1003, ¶19. By doing so, the accumulation of rounding errors was minimized, as intermediate

operations had a finer granularity and could represent values more accurately. *Id.*; Ex-1003, ¶¶19-20.

II. THE '267 PATENT

A. Overview

The '267 patent is directed to “utilizing motion prediction in video coding.” Ex-1001, Abstract. The patent discusses an “existing” process for generating a bi-directional prediction for a current block. *See, e.g.*, Ex-1001, 13:43-55, Fig. 10:

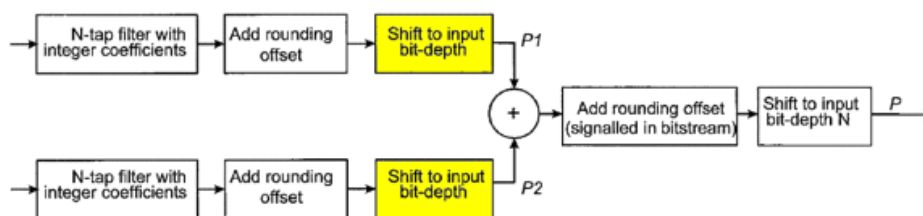


Fig. 10

The '267 patent does not purport to invent this conventional process, admitting that the “Background” art includes video coding standards, e.g., H.264 (Ex-1001, 1:34-46), and “bi-directional prediction” (*id.*, 3:49-55), with reference blocks determined by motion vectors (*id.*, 2:20-34, 3:12-18), predictions determined based on reference blocks (*id.*, 1:34-46), bi-directional prediction obtained by combining two predictions (*id.*, 3:49-55, 3:66-4:20), and residual data calculated as a difference between the prediction and current block (*id.*, 1:52-59, 3:25-30, 2:1-12).

The patent further admits that the prior art included various techniques for addressing rounding errors when averaging two blocks. Ex-1001, 3:66-4:20.

According to the '267 patent, “these methods increase somewhat the complexity” of encoders. *Id.*, 4:21-25.

The patent purports to “reduc[e] the effect of rounding errors in bi-directional ... prediction” by maintaining prediction signals “in a higher precision during the prediction calculation” and rounding the prediction down to a lower precision “after the two or more prediction signals have been combined with each other.” Ex-1001, 4:29-43. In particular, it describes pixels having a first precision (e.g., N) and predictions having a second precision (e.g., M) that is higher than the first. Ex-1001, 12:41-13:55, Fig. 11:

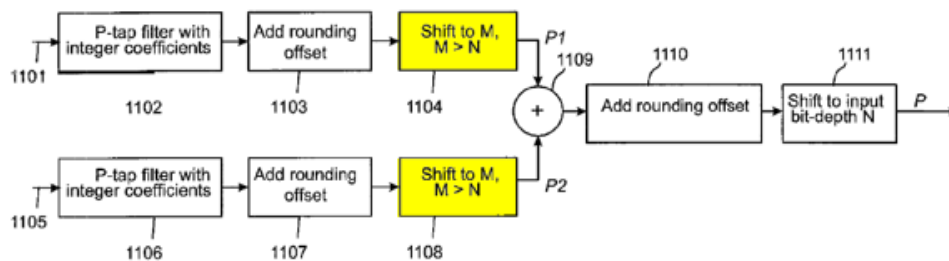


Fig. 11

The patent only purports to change the number of right-shifted bits for predictions before they are combined.

However, this approach was already known and used in video coding. *Supra* §I.A.

B. Prosecution History

During prosecution of a parent application, the Examiner issued §103 rejections based on U.S. 2013/0142262 (“Ye”). Ex-1003, ¶¶26-27. The Examiner explained that Ye’s weighted-prediction teachings “increase[d] the precision of $P_0(x,y)$ ” by multiplying it by a weight (“ $w \cdot P_0(x,y)$ ”), which suggests a higher second precision. Ex-1007, 000249-51. However, the Examiner acknowledged that Ye did not explicitly state “the number of bits needed to represent values of said first prediction and values of said second prediction,” which is “higher than said first precision[.]” Ex-1007, 000252.

In response, the Applicant did not dispute that the weighted prediction (e.g., “ $w \cdot P_0(x,y)$ ”) teaches the claimed first or second prediction, and instead argued that Ye lacks teachings about increased precision. Ex-1007, 000280-82; Ex-1003, ¶¶28-30.

C. Priority Date

The ’267 patent cites a provisional application filed January 7, 2011. This Petition analyzes obviousness as of that date, and all relied-upon art predates January 7, 2011.

III. LEVEL OF ORDINARY SKILL IN THE ART

The filed of the patent is video encoding/decoding. A POSITA at the time of the alleged invention of the ’267 patent would have had a (1) Bachelor’s degree in electrical engineering, computer engineering, computer science, or a comparable

field of study such as physics, and (2) approximately two to three years of practical experience with video encoding/decoding. Additional experience can substitute for the level of education, and vice-versa. Ex-1003, ¶¶33-36.

IV. CLAIM CONSTRUCTION

A. “precision”

A POSITA would have understood “precision” is satisfied by, but not necessarily limited to, “a number of bits needed to represent possible values.” Ex-1003, ¶¶39-42.

Dependent claims 6, 12, and 18 specify that “*precision indicates a number of bits needed to represent the values* of the pixels” and predictions. Likewise, the specification uses “precision” to refer to a number of bits representing possible values for pixels/predictions. Ex. 1001, 12:41-13:55, 14:4-10. During prosecution, the Applicant stated, “*precision indicates the number of bits needed to represent values of said pixels*” and “prediction[s.]” Ex-1007, 000280.

V. IDENTIFICATION OF CHALLENGE

Petitioner requests *inter partes* review and cancelation of the challenged claims on the following grounds:

Grounds	Claims	Statutory Basis	Prior Art ¹
1	1-18	§ 103	Walker

¹ Pre-AIA 35 U.S.C. §102 applies. MPEP §2159.02.

2	1-18	§ 103	Karczewicz-I in view of Karczewicz-II
---	------	-------	--

The Grounds render the challenged claims obvious because any differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a POSITA to which the subject matter pertains. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007).

A. Ground 1

1. Walker (Ex-1004)

Walker is U.S. Patent Application Publication 2005/0281334, published December 22, 2005; it is prior art under §§102(a), (b), and (e). Ex-1003, ¶47.

Walker teaches “weighted prediction methods” for video encoding under H.264 that obtain a combined prediction based on a weighted combination of predictions of two reference blocks. Ex-1004, ¶3, ¶¶58-70. For decoding, Walker proposes “a universal formula” to decode weighted prediction frames[.]” Ex-1004, ¶72, ¶92; Ex-1003, ¶48.

Moreover, Walker teaches the limitation the Examiner found to be missing in the prior art (*supra* §II.B), with explicit teachings regarding the numbers of bits needed to represent its pixel/prediction values:

TABLE 2

Op- eration No.	Operation	Bitwidths Involved	Bitwidth of Operation Result
1	(pred0)w0,	8 bits * 8 bits	16
2	(pred1)w1 2^{LWD-1}	logWD has a maximum value of 7, therefore LWD - 1 has a maximum value of 7 for case 3	7
3	(pred0)w0 + (pred1)w1 + 2^{LWD-1}	16 bits + 16 bits + 7 bits	18
4	((pred0)w0 + (pred1)w1 + $2^{\log WD}$) >> (LWD + 1))	(18 bits) >> (8 bits)* *maximum value of LWD + 1	10
5	(o ₀ + o ₁ + 1) >> (1)	(8 bits + 8 bits + 1) >> (1)	8
6	Clip1[Op. 4 + Op. 5] (for Clip1, see Eq. (11))	Clip1[10 bits + 8 bits]	8

Ex-1004, ¶84; Ex-1003, ¶56.

Walker includes multiple figures and corresponding teachings directed to interrelated aspects of video encoding/decoding that are used together. Figure 1 shows “a general communications system for encoding and decoding” that includes bi-prediction under H.264. Ex-1004, Fig. 1, ¶25, ¶¶45-46, ¶72. Figures 5, 8-9, along with their corresponding descriptions, provide additional details on bi-prediction and interpolation processes carried out by video encoders/decoders, likewise in the H.264 context. Ex-1004, ¶29, ¶¶34-35, ¶¶110-111, ¶114. Walker applies these teachings to the encoder/decoder of Figure 1. Ex-1004, ¶29, ¶¶110-111, ¶114. Therefore, a POSITA would have been motivated to combine Walker’s teachings, as explained for its embodiments. Ex-1003, ¶¶49-54.

Walker is analogous art in the same field as the ’267 patent (*supra* §III) because it is directed to video encoding/decoding and motion prediction. Ex-1004, ¶3, ¶¶46-49, ¶¶58-72, ¶112; Ex-1003, ¶55.

2. Independent Claims 1, 7, and 13

[1a]. A method for encoding a block of pixels, the method comprising:

Walker teaches the preamble. Ex-1003, ¶¶57-60.

Walker teaches “weighted prediction *methods* [] used for *encoding* the video data.” Ex-1004, ¶3.² Walker teaches “a general communications system for encoding and decoding streaming pictures.” Ex-1004, ¶25, Fig. 1:

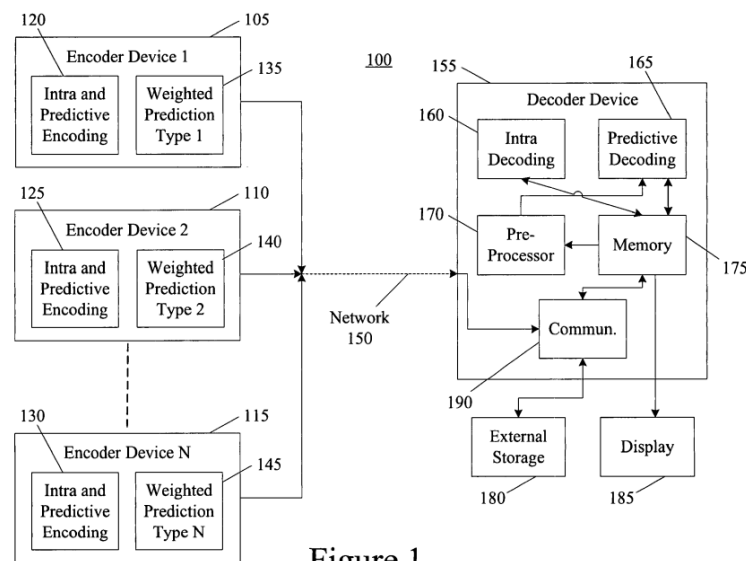


Figure 1

Walker teaches “perform[ing] the various types of encoding,” including “weighted bi-directional prediction.” Ex-1004, ¶25; Ex-1003, ¶58.

Walker encodes a block of pixels, including with bi-directional motion prediction on macroblocks. Ex-1004, ¶34, ¶60, ¶111. A macroblock is a block of pixels. Ex-1004, ¶30; Ex-1003, ¶59.

² All annotations/emphasis added unless otherwise noted.

Walker teaches a method for encoding a block of pixels, comprising the operations explained below for limitations [1b]-[1g]. *Supra* §V.A.1; Ex-1003, ¶60.

[7a]. An apparatus for encoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:

Walker teaches [7a], including an encoder for encoding a block of pixels, as explained above. *Supra* §V.A.2[1a]; §V.A.1; Ex-1003, ¶¶61-64. Walker implements its encoder teachings using a processor and memory that includes computer program code (software module). Ex-1004, ¶123; Ex-1003, ¶63.

A POSITA would have understood that, in a conventional computing device, the processor executes computer program code in the memory which is configured to cause the device to carry out its functionalities. Therefore, Walker teaches an encoder device as recited in [7a] that performs operations as described for limitations [7b]-[7g]. Ex-1003, ¶64.

[13a]. A computer program product for encoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

Walker teaches [13a], including a device with computer program code for encoding a block of pixels, as explained above. *Supra* §V.A.2[1a]-[7a]; Ex-1003, ¶¶65-67. Walker implements its teachings using program code, executable by a

processor, residing on computer-readable storage media, including memory types, hard disks, and CD-ROMs, which were forms of non-transitory media as opposed to transitory signals. *See* Ex-1004, ¶123. Software modules obviously comprise executable program code portions including program code instructions. Walker applies its teachings to a computer-program product. The operations taught by Walker, performed by the software module, teach the limitations of claim 13. *Infra* §§V.A.2[13b-13g]; *supra* §V.A.1. Additionally, a POSITA would have found it obvious to have a computer program product that includes a storage medium storing the software module, since computer program products have almost universally stored their software program code in non-transitory mediums for decades. *See* Ex-1004, ¶123. Therefore, Walker teaches a computer program product as recited in [13a] that performs the operations recited in claim 13. Ex-1003, ¶¶66-67.

[1b]/[7b]/[13b] [determining/determine], for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

Walker teaches limitations [1b], [7b], and [13b]. Ex-1003, ¶¶68-74. First, Walker teaches **determining, for a current block** (e.g., Walker’s “current macroblock” 515), **a first reference block** (e.g., “best matching macroblock” 520 in previous reference picture 510) **based on a first motion vector** (e.g., motion

vector 525) **and a second reference block** (e.g., “best matching macroblock” 580 in subsequent reference picture 575) **based on a second motion vector** (e.g., motion vector 585). Ex-1004, ¶29, ¶34, Fig. 5:

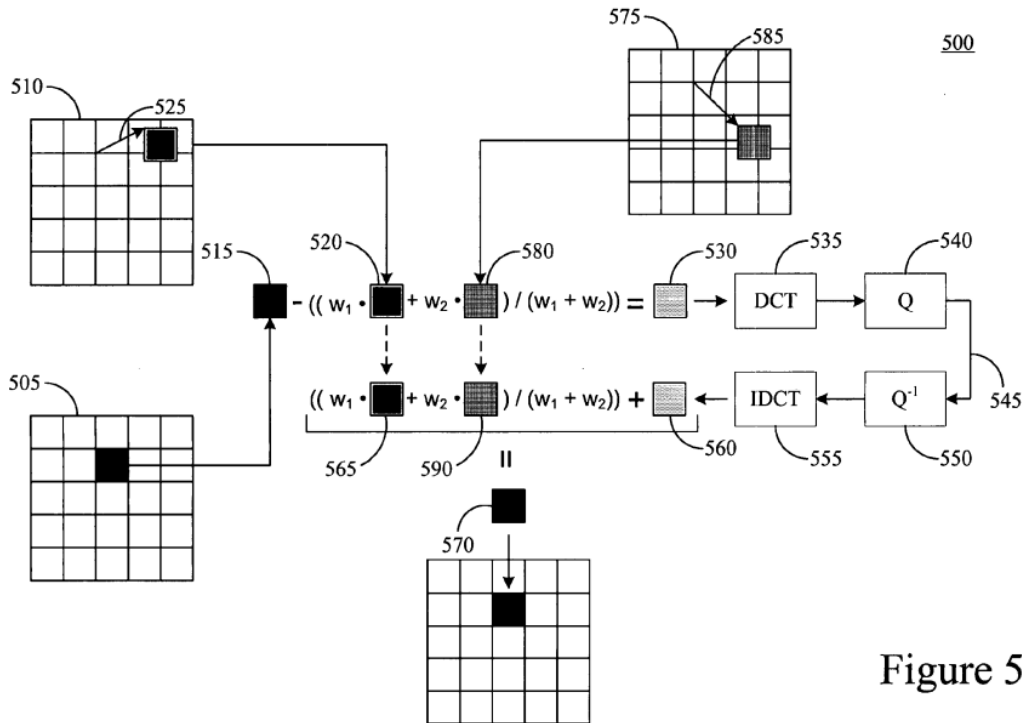


Figure 5

Ex-1003, ¶68.

For each macroblock of a B frame, Walker teaches determining two “best matching” reference blocks based on two motion vectors. Ex-1004, ¶28, ¶32, ¶34; Ex-1003, ¶69.

Walker determines a *first*, best-matching reference block for the current macroblock based on a first, *backward*-pointing motion vector. Ex-1004, ¶34, ¶32; Ex-1003, ¶70. Walker determines a *second*, best-matching reference block for the

current macroblock based on a second, *forward*-pointing motion vector. Ex-1004, ¶34, ¶32; Ex-1003, ¶71. The determination of which block to use as a reference block is based on the motion vector that identifies the location of that block.³ See Ex-1004, ¶32, ¶34; Ex-1003, ¶72.

Second, Walker teaches that **the pixels of the current block, the first reference block, and the second reference block have values with a first precision** (e.g., 8 bits). Walker teaches that the pixels in its macroblocks have 8-bit luminance and chrominance values. Ex-1004, ¶30, ¶60. Therefore, Walker teaches a “first precision,” where 8 bits are needed to represent the possible values of the pixels of the current block, the first reference block, and the second reference block. Ex-1003, ¶¶73-74; *supra* §IV.A.

[1c]/[7c]/[13c] [using/use] said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;
--

Walker teaches limitations [1c], [7c], and [13c]. Ex-1003, ¶¶75-84; *supra* §V.A.1 (explaining that Figure 5’s process is applicable to Walker’s encoder).

Walker teaches **using said first reference block** (e.g., a “prediction block” from a past frame) **to obtain a first prediction** (e.g., “(pred0)w0”). “Weighted prediction is invoked in H.264 by setting one or both variables ‘predFlagL0’ and

³ The ’267 patent admits that determining reference blocks based on motion vectors was known in the art. See Ex-1001, 2:20-34, 3:12-18.

‘predFlagL1’ equal to 1.” Ex-1004, ¶59. Equation 13 governs weighted predictions (e.g., Case 3, where “both of the two reference partitions are to be weighted”):

$$Final_pred = Clip1[(((pred0)w0 + (pred1)w1 + 2^{\log WD}) >> (\log WD + 1)) + ((o_0 + o_1 + 1) >> 1)] \quad (13)$$

Ex-1004, ¶¶67-68. Ex-1003, ¶76.

Here, the value “(pred0)w0” is the weighted prediction from the first reference block: “pred0 and pred1, are ... samples from prediction blocks from the two reference frames (one past, one future)[.]” Ex-1004, ¶60. The prediction blocks refer to best-matching macroblocks satisfying the recited “first reference block” and “second reference block.” *Supra* §V.A.2[1b]/[7b]/[13b]; Ex-1004, ¶34; *supra* §I.A; Ex-1003, ¶77.

The coefficients and constants in this equation are explained in Table 1. Ex-1004, ¶69; Ex-1003, ¶78.

Walker teaches executing a series of computational operations based on Equation 13 when both reference values are weighted. Ex-1004, ¶¶67-68. The operations are outlined in Table 2:⁴

⁴ Walker Table 2 and Equation 14 represent a general formulation that includes Equation 13. Ex-1004, ¶¶61-69, ¶74, ¶84. Equation 14’s 2^{LWD-1} and LWD equals Equation 13’s $2^{\log WD}$ and $\log WD + 1$ in Case 3, where Equation 13 applies. Ex-1004, ¶82, *compare* Ex-1004, ¶68 *with* ¶74. A POSITA would have understood that Table 2 describes the computational operations related to Equation 13. Ex-1003, ¶79, n.4.

TABLE 2

Op- eration No.	Operation	Bitwidths Involved	Bitwidth of Operation Result
1	(pred0)w0,	8 bits * 8 bits	16
2	(pred1)w1 2^{LWD-1}	logWD has a maximum value of 7, therefore LWD - 1 has a maximum value of 7 for case 3	7
3	(pred0)w0 + (pred1)w1 + 2^{LWD-1}	16 bits + 16 bits + 7 bits	18
4	((((pred0)w0 + (pred1)w1 + $2^{\log WD}$) >> (LWD + 1))	(18 bits) >> (8 bits)* *maximum value of LWD + 1	10
5	(o ₀ + o ₁ + 1) >> (1)	(8 bits + 8 bits + 1) >> (1)	8
6	Clip1[Op. 4 + Op. 5] (for Clip1, see Eq. (11))	Clip1[10 bits + 8 bits]	8

Ex-1004, ¶84; Ex-1003, ¶79.

The weighted prediction value “(pred0)w0” is the product of a sample value (e.g., pred0) from a first reference block (e.g., from a past frame) multiplied by a weight (e.g., w0), and is thus a first prediction obtained using the first reference block.⁵ Ex-1004, ¶60, ¶¶68-69; Ex-1003, ¶¶80-81.

Walker teaches that **said first prediction** (e.g., the value “(pred0)w0”) **having a second precision** (e.g., 16 bits), **which is higher than said first precision** (e.g., 8 bits). In Walker’s Table 2, the column named “Bitwidth of

⁵ This is consistent with the usage of “prediction” in the ’267 specification, which includes embodiments where predictions are calculated by mathematical operations including multiplying pixel values in reference blocks with weights. *See, e.g.*, Ex-1001, 12:41-13:18, 13:19-42, 14:4-22; *supra* §I.B; *infra* §V.A.5. During prosecution, the Applicant did not dispute that weighted predictions teach first and second predictions. Ex-1007, 000280-81; *supra* §II.B; Ex-1003, ¶81, n.5.

Operation Result” includes the number of bits needed to store the result of each operation based on the possible values of the result. Ex-1004, ¶84. The row corresponding to Operation No. 1 indicates that 16 bits are needed to represent the possible values of (pred0)w0. *Id.*; Ex-1003, ¶82. Thus, the first prediction has a second precision of 16 bits, which is higher than the first precision of 8 bits. *Supra* §IV.A; *supra* §V.A.2[1b]/[7b]/[13b]; Ex-1003, ¶83. Walker applies the above-described teachings to all the pixels of the block. *E.g.*, Ex-1004, ¶34; Ex-1003, ¶84.

[1d]/[7d]/[13d] [using/use] said second reference block to obtain a second prediction, said second prediction having the second precision;
--

Walker teaches limitations [1d], [7d], and [13d]. Ex-1003, ¶¶85-87. The analysis for limitations [1c], [7c], and [13c] similarly applies here. *Supra* §V.A.2[1c]/[7c]/[13c]. Walker teaches **using said second reference block** (e.g., a “prediction block” from a future frame) **to obtain a second prediction** (e.g., “(pred1)w1”), **said second prediction having the second precision** (e.g., 16 bits). Ex-1003, ¶85; Ex-1004, ¶¶67-68.

The value “(pred1)w1” is the product of a sample value (e.g., pred1) from a second reference block (e.g., from a future frame) multiplied by a weight (e.g., w1), and is thus is thus a second prediction obtained using the second reference block; the second prediction has the second precision of 16 bits. Ex-1004, ¶60, ¶69,

¶84; *supra* §V.A.2[1c]/[7c]/[13c]; Ex-1003, ¶86. Walker applies the above-described teachings to all the pixels of the block. *E.g.*, Ex-1004, ¶34; Ex-1003, ¶87.

[1e]/[7e]/[13e] [obtaining/obtain] a combined prediction based at least partly upon said first prediction and said second prediction;

Walker teaches limitations [1e], [7e], and [13e]. Ex-1003, ¶¶88-91. Equation 13 and Table 2 govern weighted predictions:

$$\text{Final_pred} = \text{Clip1}[\left(\frac{((\text{pred0})w0 + (\text{pred1})w1 + 2^{\log \text{WD}})}{2^{\log \text{WD} + 1}}\right) + \left(\frac{o_0 + o_1 + 1}{2^{\log \text{WD} + 1}}\right)] \quad (13)$$

Ex-1004, ¶¶67-68;

TABLE 2

Op- eration No.	Operation	Bitwidths Involved	Bitwidth of Operation Result
1	(pred0)w0,	8 bits * 8 bits	16
2	(pred1)w1 $2^{\text{LWD}-1}$	logWD has a maximum value of 7, therefore LWD - 1 has a maximum value of 7 for case 3	7
3	(pred0)w0 + (pred1)w1 + $2^{\text{LWD}-1}$	16 bits + 16 bits + 7 bits	18
4	$\left(\frac{((\text{pred0})w0 + (\text{pred1})w1 + 2^{\log \text{WD}})}{2^{\log \text{WD} + 1}}\right)$	(18 bits) >> (8 bits)* *maximum value of LWD + 1	10
5	$(o_0 + o_1 + 1) >> (1)$	(8 bits + 8 bits + 1) >> (1)	8
6	Clip1[Op. 4 + Op. 5] (for Clip1, see Eq. (11))	Clip1[10 bits + 8 bits]	8

Ex-1004, ¶84; *supra* §§V.A.2[1c-1d]/[7c-7d]/[13c-13d]; Ex-1003, ¶88.

Walker teaches **obtaining a combined prediction based at least partly upon said first prediction** (e.g., “(pred0)w0”) **and said second prediction** (e.g., “(pred1)w1”). As explained above, Operation No. 1 calculates the first and second predictions (e.g., “(pred0)w0” and “(pred1)w1”). *Supra* §§V.A.2[1c-1d]/[7c-7d]/[13c-13d]. Operation No. 2 calculates a rounding offset (e.g., $2^{\text{LWD}-1}$ or $2^{\log \text{WD}}$)⁶. *Infra* §V.A.6. Operation No. 3 calculates a sum by adding up the first prediction (e.g., “(pred0)w0”), the second prediction (e.g., “(pred1)w1”), and the rounding offset (e.g., $2^{\text{LWD}-1}$ or $2^{\log \text{WD}}$). Ex-1003, ¶89. This sum of Operation No. 3 (e.g., “(pred0)w0+(pred1)w1+ $2^{\log \text{WD}}$ ”) is a combined prediction obtained based *at least partly* upon said first prediction and said second prediction. Ex-1003, ¶90. Walker applies the above-described teachings to all the pixels of the block. *E.g.*, Ex-1004, ¶34; Ex-1003, ¶91.

[1f]/[7f]/[13f] [decreasing/decrease] a precision of said combined prediction by shifting bits of the combined prediction to the right; and

Walker teaches limitations [1f], [7f], and [13f]. Ex-1003, ¶¶92-96.

Equation 13 and Table 2 govern weighted predictions:

$$\text{Final_pred} = \text{Clip1}[\left(\left(\left(\text{pred0}\right)w0 + \left(\text{pred1}\right)w1 + 2^{\log \text{WD}}\right) \gg (\log \text{WD} + 1)\right) + ((o_0 + o_1 + 1) \gg 1)] \quad (13)$$

Ex-1004, ¶¶67-68;

⁶ $2^{\text{LWD}-1}$ and $2^{\log \text{WD}}$ are equivalent in the case where Equation 13 applies. *Supra* §§V.A.2[1c-1d]/[7c-7d]/[13c-13d], n.4; Ex-1004, ¶82; Ex-1003, ¶79, n.4.

TABLE 2

Op- eration No.	Operation	Bitwidths Involved	Bitwidth of Operation Result
1	(pred0)w0,	8 bits * 8 bits	16
2	(pred1)w1 2^{LWD-1}	logWD has a maximum value of 7, therefore LWD - 1 has a maximum value of 7 for case 3	7
3	(pred0)w0 + (pred1)w1 + 2^{LWD-1}	16 bits + 16 bits + 7 bits	18
4	((((pred0)w0 + (pred1)w1 + $2^{\log WD}$) >> (LWD + 1))	(18 bits) >> (8 bits)* *maximum value of LWD + 1	10
5	(o ₀ + o ₁ + 1) >> (1)	(8 bits + 8 bits + 1) >> (1)	8
6	Clip1[Op. 4 + Op. 5] (for Clip1, see Eq. (11))	Clip1[10 bits + 8 bits]	8

Ex-1004, ¶84; *supra* §§V.A.2[1c-1e]/[7c-7e]/[13c-13e]; Ex-1003, ¶93.

Walker teaches **shifting bits of the combined prediction to the right** (e.g., “>>(logWD+1)”). Operation No. 4 teaches shifting bits of the sum to the right as emphasized above. Ex-1004, ¶68, ¶84. The symbol >> means right bit-shift. Ex-1004, ¶46. (logWD+1) indicates the number of bits shifted. Ex-1003, ¶94.

Walker teaches that shifting bits to the right **decreases a precision of said combined prediction**⁷ (e.g., from 18 bits to 10 bits). As shown in Table 2,

⁷ Walker teaches performing additional mathematical operations on the combined prediction. Ex-1004, ¶¶66-69, ¶84. These additional operations are allowed by the claims 1, 7, and 13 because each claim recites the transitional term “comprising.” Ex-1003, ¶95, n.9.

Operation No. 4 decreases the number of bits needed to represent possible values of the combined prediction from 18 bits to 10 bits. Ex-1004, ¶84. Walker thus teaches decreasing a precision of said combined prediction. *Supra* §IV.A; Ex-1003, ¶95. Walker applies the above-described teachings to all the pixels of the block. *E.g.*, Ex-1004, ¶34; Ex-1003, ¶96.

[1g]/[7g]/[13g] [encoding/encode] residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.
--

Walker teaches limitations [1g], [7g], and [13g]. Ex-1003, ¶¶97-100.

First, Walker teaches **determining residual data** (e.g., “residual error”) **based upon a difference between the combined prediction** (e.g., “weighted combined bi-directional macroblock”) **and the block of pixels** (e.g., “current macroblock”). Walker teaches obtaining a combined prediction for the pixels of the current block. *Supra* §§V.A.2[1e-1f]/[7e-7f]/[13e-13f]; *see* Ex-1004, ¶30, ¶60, ¶34; Ex-1003, ¶98.

The collective combined prediction values are referred to as “a weighted combined bi-directional macroblock” in Walker, which “is subtracted from current macroblock 515 resulting in residual error 530.” Ex-1004, ¶34. Thus, the residual error (i.e., residual data) is determined based upon a difference between the combined prediction and the block of pixels. *See* Ex-1004, ¶32, ¶34; Ex-1003, ¶99.

Second, Walker teaches **encoding the residual data** (e.g., “residual error”) **in a bitstream** (e.g., a data stream that includes B frames). Ex-1004, ¶34 (“Residual error 530 is encoded with DCT 535 ...”), ¶28 (“FIG. 2B is a diagram illustrating a conventional encoded data stream ... [including] residual errors ...”), Fig. 2B; Ex-1003, ¶100.

3. Dependent Claims 2, 8, and 14

2. The method according to claim 1,

8. The apparatus according to claim 7,

14. The computer program product according to claim 13,

wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

Walker teaches claims 1, 7, and 13 and the limitations of claims 2, 8, and 14. *Supra* §V.A.2; Ex-1003, ¶¶101-107.

First, Walker teaches **an instance in which said first motion vector points to a subpixel** (e.g., an “interpolated pixel”). Walker teaches an encoder⁸ using a motion vector to point to either a “pixel or interpolated pixel.” Ex-1004, ¶111, Fig. 8. These interpolated pixels are subpixels. *Compare* Ex-1004, ¶114, Fig. 9 *with* Ex-1001, 12:56-63, Fig. 12; Ex-1003, ¶102, ¶105.

⁸ Walker teaches the process of Figure 8 being used to decode weighted bi-directional predicted video data generated by the encoder devices of Figure 1. *Supra* §V.A.1.

Second, Walker teaches **said first prediction is obtained by interpolation using pixel values of said first reference block**. Ex-1004, ¶114, ¶111; Ex-1003, ¶103. Walker teaches obtaining a prediction for a subpixel by interpolation. Ex-1004, ¶114 (“FIG. 9 is an illustration of an example of half-pixel interpolation for use in motion compensation.”), Fig. 9; Ex-1003, ¶104.

Walker teaches an example of performing interpolation with a 2-tap FIR filter using pixel values of integer pixels neighboring a half pixel, which are in the reference block. Ex-1004, ¶114, Fig. 9. Thus, Walker teaches obtaining said first prediction by interpolation using pixel values of said first reference block. Ex-1003, ¶106.

As explained for claim 1, Walker teaches using the first motion vector and the first prediction in a bi-directional weighted motion prediction process (*supra* §V.A.2). By teaching sub-pixel interpolation, Walker teaches or at least suggests that the interpolation results are used in determining a weighted prediction for motion prediction.⁹ Ex-1004, ¶111, ¶114. This confirms that it would have been

⁹ While Walker’s paragraph ¶111 discusses weighted prediction as part of a decoding process, a POSITA would have understood that this also teaches, or at least suggests, that a matching weighted prediction process is used for encoding. *Supra* §I.A.

obvious for the first motion vector to point to a subpixel and for the first prediction to be obtained by interpolation.¹⁰ Ex-1003, ¶107.

4. Dependent Claims 3, 9, and 15

3. The method according to claim 2,

9. The apparatus according to claim 8,

15. The computer program product according to claim 14,

wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.

Walker teaches claims 2, 8, and 14 and the limitations of claims 3, 9, and 15.

Supra §V.A.3; Ex-1003, ¶¶108-112.

Walker teaches performing interpolation using values of said first reference block (*supra* §V.A.3) for a half pixel using a “2-tap FIR filter,” which is a “P-tap filter,” where P is 2. Ex-1004, ¶114, Fig. 9. As was known in the art, the popular H.264 standard used 6-tap filters for interpolation, where P is 6.¹¹ Ex-1003, ¶109; Ex-1004, ¶114.

Walker teaches using “a 2-tap FIR filter with weights [0.5 0.5].” Ex-1004, ¶114, Fig. 9. This performs an averaging function using weights of 0.5 to

¹⁰ Moreover, the '267 patent admits that the limitations of claims 2, 8, and 14 were known in the prior art. Ex-1001, 2:60-3:11; Ex-1003, ¶107, n.12.

¹¹ This was admitted by the '267 patent. Ex-1001, 2:60-3:11.

effectively divide values by two (multiply by $1/2$) when averaging them. *See id.*; Ex-1003, ¶110. Walker further teaches an optimization where bit-shifting to the right is used instead of division. Ex-1004, ¶37 (“Another way of normalizing without a division operation is by use of bit shifting.”), ¶46. When applied to Walker’s interpolation teachings (Ex-1004, ¶114), it would have been obvious to avoid multiplication and division by using an FIR filter with equal weights [1 1] and right-shifting the results by 1 bit (e.g., dividing the sum by 2). This achieves the same result while reducing computational complexity and avoiding multiplication and division as Walker also teaches (Ex-1004, ¶37). Ex-1003, ¶111.

A POSITA would have been motivated to apply Walker’s teachings in this manner because, in computing, multiplication and division were known to be far more computationally complex than addition and bit-shifting, which is why calculations were often optimized in the art by bit-shifting rather than multiplying or dividing values. *See id.*; Ex-1003, ¶112.

5. Dependent Claims 4, 10, and 16

4. The method according to claim 2,

10. The apparatus according to claim 8,

16. The computer program product according to claim 14,

wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

Walker teaches claims 2, 8, and 14 and the limitations of claims 4, 10, and 16. *Supra* §V.A.3; Ex-1003, ¶¶113-116.

Walker teaches **instances in which said second motion vector points to an integer sample**. As explained above, Walker teaches that the motion vector may point to an integer pixel or a subpixel and therefore Walker's teachings encompass instances in which said second motion vector points to an integer sample.¹² *Supra* §V.A.3; Ex-1004, ¶111 ("pixel or interpolated pixel"), ¶114, Figs. 8-9.

Walker further teaches **said second prediction is obtained by shifting values of said second reference block to the left**. For both integer and sub-pixels, Walker teaches obtaining the second prediction (e.g., "(pred1)w1") as the product of a sample value (e.g., pred1) from a second reference block (e.g., from a future frame) multiplied by a weight (e.g., "w1"). *Supra* §V.A.2[1d]/[7d]/[13d]; Ex-1004,

¹² The '267 patent admits that integer ("full") pixels were known in the art. Ex-1001, 2:60-3:11.

¶¶67-68; Ex-1003, ¶115. This calculation is applied to all the pixels of the block.

E.g., Ex-1004, ¶34; *supra* §V.A.2[1d]/[7d]/[13d].

Walker teaches examples where the weight takes a value that is a power of 2 (e.g., 4). Ex-1004, ¶37 (“w₂ could be equal to 4”). Walker further teaches that multiplication and bit-shifting are equivalent mathematical operations. Ex-1004, ¶46 (“[T]he bit shifting operations could be accomplished by other methods such as, for example, applying a scaling factor through multiplication or division.”). Therefore, it would have been obvious to implement Walker’s teachings, where the sample pixel value from the reference block is multiplied by 4, by left-shifting the pixel value to the left by 2 bits. A POSITA would have found it obvious and been motivated to do so because, in computing terms, left-shifting is a simpler operation than multiplication and can be performed without the need for a multiplication unit. Ex-1003, ¶115.

Furthermore, left-shifting to multiply a value by a coefficient and increase the precision of intermediate values in calculations was well known in the art. Ex-1013, ¶91, ¶116, ¶124, ¶129, ¶131; Ex-1015, ¶67, ¶¶84-85, ¶103. This confirms that it would have been obvious to shift values of the second reference block to the left. Ex-1003, ¶116.

6. Dependent Claims 5, 11, and 17

- 5. The method according to claim 1, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises:**
- 11. The apparatus according to claim 7, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by:**
- 17. The computer program product according to claim 13, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to:**
- [inserting/insert] a rounding offset to the combined prediction before said decreasing.**

Walker teaches claims 1, 7, and 13 and the limitations of claims 5, 11, and 17. *Supra* §V.A.2; Ex-1003, ¶¶117-122.

Walker teaches **inserting a rounding offset** (e.g., 2^{LWD-1} or $2^{\log WD}$) **to the combined prediction** as part of Operation No. 3 **before said decreasing** of the precision in Operation No. 4. As explained above, Walker teaches obtaining a combined prediction via its disclosure of calculating a sum by adding up the first prediction (e.g., “(pred0)w0”), the second prediction (e.g., “(pred1)w1”), and the rounding offset (e.g., 2^{LWD-1} , which equals $2^{\log WD}$ for Case 3, *supra* §V.A.2[1c]/[7c]/[13c], n.4) in Operation No. 3 of Table 2. *Supra* §V.A.2[1e]/[7e]/[13e]; Ex-1004, ¶68, ¶84. Therefore, Walker inserts a rounding

offset to the combined prediction before its next step, in Operation No. 4 of Table 2, where Walker decreases a precision of said combined prediction by shifting bits of the combined prediction to the right. *Supra* §V.A.2[1f]/[7f]/[13f]; Ex-1004, ¶68, ¶84:

TABLE 2

Op- eration No.	Operation	Bitwidths Involved	Bitwidth of Operation Result
1	(pred0)w0,	8 bits * 8 bits	16
2	(pred1)w1 2^{LWD-1}	logWD has a maximum value of 7, therefore LWD - 1 has a maximum value of 7 for case 3	7
3	(pred0)w0 + (pred1)w1 + 2^{LWD-1}	16 bits + 16 bits + 7 bits	18
4	((((pred0)w0 + (pred1)w1 + $2^{\log WD}$) >> (LWD + 1))	(18 bits) >> (8 bits)* *maximum value of LWD + 1	10
5	(o ₀ + o ₁ + 1) >> (1)	(8 bits + 8 bits + 1) >> (1)	8
6	Clip1[Op. 4 + Op. 5] (for Clip1, see Eq. (11))	Clip1[10 bits + 8 bits]	8

Ex-1003, ¶118.

Walker explains that 2^{LWD-1} is a **rounding offset** because it is added to sample values, thereby providing an offset for the sample value. Ex-1004, ¶74 (“offset variables”). Moreover, the offset variable 2^{LWD-1} is inserted into the combined prediction before a right-shift operation, which causes values to round to

the closest integer after the right shift, rather than always rounding down which would be the result absent the offset. Thus, the offset variable 2^{LWD-1} is a rounding offset. Ex-1003, ¶119.

Walker teaches this rounding offset as part of its rounding process, which decreases the precision as recited by the claims. Ex-1004, ¶68, ¶84. Additionally, it was obvious for said decreasing to include the rounding offset (claim 5) because the insertion of the rounding offset is performed right before and in conjunction with the right-shifting to affect the direction of the rounding. Walker includes a rounding offset to control rounding error resulting from the right-shift operation that decreases precision. This was common in the art. *Supra* §I.B; Ex-1003, ¶120.

Because Walker teaches applying its operations to a computer implementation, it teaches at least one memory and computer program code being configured to cause the apparatus to perform the operations of claim 11. *Supra* §V.A.2[7a]; Ex-1003, ¶121.

Likewise, because Walker applies its teachings to a computer implementation, it teaches the program code instructions configured to perform the operations of claim 17. *Supra* §V.A.2[13a]; Ex-1003, ¶122.

7. Dependent Claims 6, 12, and 18

6. The method according to claim 1,

12. The apparatus according to claim 7,

18. The computer program product according to claim 13,

wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

Walker teaches claims 1, 7, and 13 and the additional limitations of claims 6, 12, and 18. *Supra* §V.A.2; Ex-1003, ¶¶123-125.

As explained above, Walker teaches that the pixels of the current block, the first reference block, and the second reference block have values with a first precision because 8 bits are needed to represent the possible pixel values of these blocks. *Supra* §V.A.2[1b]/[7b]/[13b]. Here, **the first precision indicates a number of bits needed to represent the values of the pixels.** Ex-1003, ¶124.

As explained above, Walker teaches said first and second predictions having a second precision because 16 bits are needed to represent the possible values of the first and second predictions. *Supra* §§V.A.2[1c-1d]/[7c-7d]/[13c-13d]. Here, **the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.** Ex-1003, ¶125.

B. Ground 2

1. Karczewicz-I (Ex-1005)

Karczewicz-I is U.S. Patent Application Publication 2011/0007799, filed July 9, 2009; it is prior art under §102(e). Ex-1003, ¶128.

Karczewicz-I teaches block-based techniques for motion prediction, including bi-directional predictions calculated by averaging predictions based on two reference blocks, which may be calculated using sub-pixel interpolation. *E.g.*, Ex-1005, ¶35, ¶¶41-44, ¶¶55-60; Ex-1003, ¶129.

Karczewicz-I is analogous art in the same field as the '267 patent (*supra* §III) because it is directed to video encoding/decoding, motion prediction, and H.264. *E.g.*, Ex-1005, ¶53, ¶¶35-37; Ex-1003, ¶130.

2. Karczewicz-II (Ex-1006)

Karczewicz-II is U.S. Patent Application Publication 2009/0257499, published October 2009; it is prior art under §§102(a), (b) and (e). Ex-1003, ¶132.

Karczewicz-II reduced rounding inaccuracies with interpolated pixels by maintaining higher precision for intermediate values while delaying rounding until later in the process. *E.g.*, Ex-1006, ¶10, ¶39, ¶53, ¶59, ¶¶93-106. Ex-1003, ¶134. For example, Karczewicz-II “generates half-pixel values ... stores the half-pixel values as *non-rounded* versions” and combines them “based on the *non-rounded versions* of the half-pixel values and the integer pixel values. Ex-1006, ¶17, ¶¶96-108; Ex-1003, ¶¶133-134.

These are the same calculations as embodiments of the '267 patent, which likewise perform two half-pixel interpolations (one for each reference block) and then average the higher-precision half-pixel predictions together. Ex-1001, 12:41-13:55.

Karczewicz-II is analogous art in the same field as the '267 patent (*supra* §III) because it is directed to video encoding/decoding and H.264. *E.g.*, Ex-1006, ¶2, ¶¶46-47; Ex-1003, ¶135.

3. Motivation to Combine

Karczewicz-I and Karczewicz-II are Qualcomm patent applications by the same inventors. Both apply their teachings to similar video-coding architectures. Ex-1005, ¶2, ¶¶29-50, Fig. 1; Ex-1006, ¶2, ¶¶40-53, Fig. 1. Both are directed to block-based H.264 motion prediction. Ex-1005, ¶35, ¶44, ¶¶55-60; Ex-1006, ¶8, ¶¶35-36, ¶46, ¶54. Both teach a video encoder that performs motion estimation and compensation for inter-predictive coding:

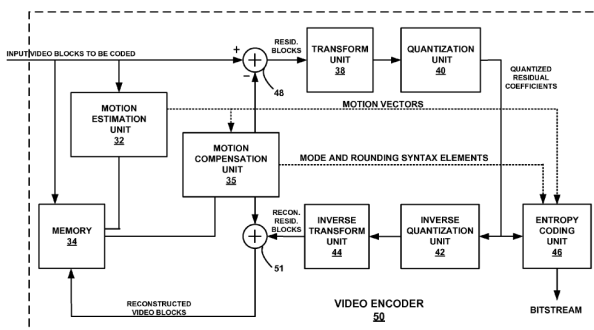


FIG. 2

Ex-1005, Fig. 2, ¶53.

Ex-1003, ¶¶136-137.

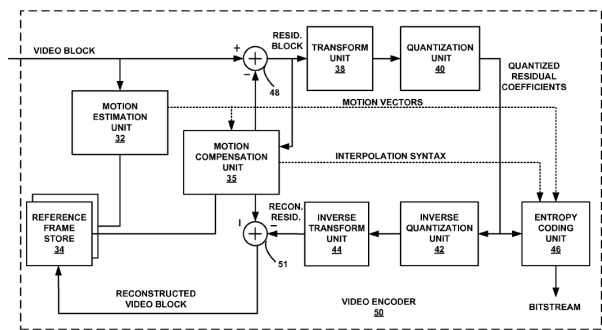


FIG. 2

Ex-1006, Fig. 2, ¶56.

These similarities would have motivated a POSITA to implement teachings from Karczewicz-I and Karczewicz-II using their common architecture, combining prior art elements according to known methods. Karczewicz-I teaches bi-prediction techniques for H.264 with two motion vectors pointing to two blocks of pixels that are averaged together. Ex-1005, ¶¶53, ¶35, ¶44, ¶60. Beyond integer pixels, Karczewicz-II teaches that H.264 motion vectors can also point to fractional sub-pixels (Ex-1006, ¶¶56-58, ¶¶93-102),¹³ and Karczewicz-II teaches improved calculations for averaging interpolated pixel values, where rounding is delayed until later in the process, thereby maintaining higher precision for intermediate calculations (Ex-1006, ¶10, ¶¶20-24, ¶39, ¶¶99-106). Therefore, Karczewicz-II provides complementary teachings that improve Karczewicz-I by reducing rounding inaccuracies. *See, e.g.*, Ex-1006, ¶102, ¶39, ¶53, ¶59; Ex-1003, ¶¶138-140.

Karczewicz-I and Karczewicz-II both encompass motion prediction calculations that average interpolated pixel values. The calculations taught by Karczewicz-II correspond to the calculations used for Karczewicz-I's bi-predicted pixel values when applied to integer and sub-pixel values. *Id.* A POSITA would have been motivated to use Karczewicz-II's known techniques to improve similar

¹³ This was known in the art, as admitted in the '267 patent's "Background Information" section. *Supra* §I.A; Ex-1001, 2:60-3:11.

devices/methods, as taught by Karczewicz-I, in the same way to improve prediction accuracy. Ex-1006, ¶102, ¶10. And it would have had predictable results because it applies teachings from Karczewicz-II to corresponding mathematical calculations used for Karczewicz-I, using similar video codec architectures. Ex-1003, ¶141.

A POSITA would have recognized the applicability of Karczewicz-II to the corresponding calculations in Karczewicz-I, which was a simple matter given the level of ordinary skill. Notably, a POSITA would have understood fundamental computer logic concepts (e.g., binary arithmetic, bit shifting, precision control, rounding and offsetting, and error analysis) and mathematical calculations for known motion estimation and compensation techniques (e.g., bi-directional prediction, determination and use of motion vectors, interpolation for fractional pixels) because they are integral to working with video codecs. Ex-1003, ¶141.

Pixel Interpolation. H.264 includes interpolated fractional pixel positions. Karczewicz-II explains that H.264 calculates predictions for half-pixel positions using 6-tap filters that interpolate the sub-pixel based on nearby pixels in the same row (e.g., half-pixel “b”) or column (e.g., half-pixel “h”). Ex-1006, ¶74, ¶¶93-94, Fig. 4A-4B:

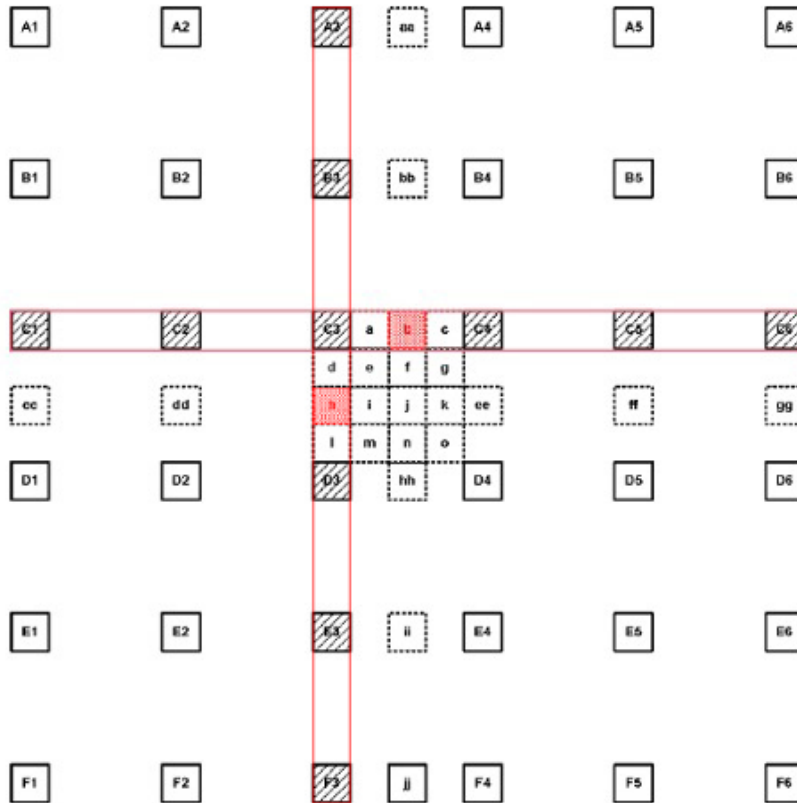


FIG. 4B

Ex-1003, ¶144.

This process involved two steps. First, the filter multiplied the six pixels in the row (or column) by filter values and added the products together to produce a non-rounded prediction (e.g., b1):

$$b1 = C1 - 5 * C2 + 20 * C3 + 20 * C4 - 5 * C5 + C6$$

Second, the result was rounded, e.g., right-shifted (“>>”) producing rounded prediction b:

$$b = \max(0, \min(255, (b1 + 16) >> 5))$$

Ex-1006, ¶¶93-94, Fig. 4B; Ex-1003, ¶145.

For half-pixel positions in the center of four integer pixels (e.g., j), interpolation was applied in two rounds: first to interpolate a middle row of half-pixels, second to interpolate that middle row into the center-pixel, applying the using the same 6-tap interpolation filter and two-step process described above to the interpolated middle row. Ex-1006, ¶95, Fig. 4C; Ex-1003, ¶146.

For quarter-pixel predictions, Karczewicz-II averaged the two nearest integer or half-pixel predictions as explained above. Ex-1006, ¶¶96-97. Karczewicz-II teaches an improvement by “keep[ing] the highest possible precision through the intermediate steps” and avoiding “any shifting, rounding and clipping operations[] until the very last step of the interpolation process.” Ex-1006, ¶99, ¶102, ¶10, ¶39, ¶53, ¶59; Ex-1003, ¶147.

Calculation Scenarios. Karczewicz-I teaches weighted bi-directional prediction that averages two predictions based on two reference blocks (e.g., pred0(i,j), pred1(i,j)) with an offset (e.g., +1), including a default mode (Ex-1005, ¶55) with equal weights that performs a simple average. Ex-1005, ¶60:

$$\text{pred}(i,j)=(\text{pred0}(i,j)+\text{pred1}(i,j)+1)>>1$$

Ex-1005, ¶¶58-60. Because H.264 allows motion vectors to point to integer pixels or fractional pixels/subpixels (Ex-1005, ¶41; Ex-1006, ¶¶93-102, ¶¶56-58; *supra* §I.A), the predictions pred0(i,j) and pred1(i,j) encompass scenarios that include

integer pixel prediction, a half-pixel prediction, or a center-pixel prediction, and Karczewicz-I calculates an average of these pixel values. Ex-1003, ¶¶142-143, ¶¶148-149.

Consistent with H.264, Karczewicz-I teaches that the inter-predictive coding process includes interpolation, providing express teaching, suggestion, and motivation (“TSM”) to combine with known interpolation teachings. Ex-1005, ¶41. Karczewicz-II teaches optimizations for interpolating and averaging integer, half-, and center-pixels, which a POSITA would have been motivated to apply to at least three scenarios for Karczewicz-I’s teachings. Ex-1003, ¶150.

Scenario 1 (with the first motion vector pointing to a half-pixel position and the second motion vector pointing to an integer pixel position). Beyond integer pixels, Karczewicz-II explains how, for H.264, motion vectors can also point to half-pixel positions (Ex-1006, ¶¶93-102, ¶¶56-58), which therefore teaches or suggests the default weighted prediction calculated as an average of a half-pixel and integer pixel. *See* Ex-1005, ¶60, ¶55; Ex-1003, ¶151.

Karczewicz-II teaches an improved calculation for averaging a half-pixel value with an integer pixel value. *See* Ex-1006, ¶96, Fig. 4D. Karczewicz-II first explains the conventional calculation for averaging two numbers. Ex-1006, Fig. 4D, Table 1:

TABLE 1

$a = (C3 + b + 1) \gg 1$
$c = (C4 + b + 1) \gg 1$
$d = (C3 + h + 1) \gg 1$
$l = (D3 + h + 1) \gg 1$
$f = (j + b + 1) \gg 1$
$i = (j + h + 1) \gg 1$
$k = (j + ee + 1) \gg 1$
$n = (j + hh + 1) \gg 1$

Ex-1003, ¶152.

Karczewicz-II improves this conventional approach by replacing the equations in Table 1 with those in Table 3, where the pixel values are combined at a higher precision. Ex-1006, 99. The integer pixel value (e.g., C3) is multiplied by 32 (left-shifted 5 bits), taking its precision from 8 to 13 bits. Ex-1006, Table 5. Instead of using a rounded 8-bit half-pixel (e.g., b), Karczewicz-II delays the rounding step and instead uses a *non-rounded* half-pixel prediction (e.g., b1) that is 15 bits. *Id.* These pixel values are combined, along with a rounding offset of 32, before rounding to reduce the precision at the end, e.g., shifting 6 bits to the right (“ $\gg 6$ ”). *Id.*, Table 3:

TABLE 3

a =	(C3 << 5 + b1 + 32) >> 6
c =	(C4 << 5 + b1 + 32) >> 6
d =	(C3 << 5 + h1 + 32) >> 6
l =	(D3 << 5 + h1 + 32) >> 6
f =	(j1 >> 5 + b1 + 32) >> 6
i =	(j1 >> 5 + h1 + 32) >> 6
k =	(j1 >> 5 + ee1 + 32) >> 6
n =	(j1 >> 5 + hh1 + 32) >> 6

Ex-1003, ¶153.

The operations for this improved approach are shown in Table 5 of
Karczewicz-II. Ex-1006, ¶103, Table 5:

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D				
Operation	Comment	Min value	Max value	Register size
$r1 = x$	$r1$ is integer pixel x	0	255	8u
$r1 = r1 \ll 5$	$r1$ is $32 * x$	0	8160	13u
$r2 = y0$	$r2$ is $y0$ ($y0$ is a one-dimensional (1-D) half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r1 = r1 + r2$	$r1$ is $32 * x + y0$	-2550	18870	16s
$r1 = r1 + 32$	$r1$ is $32 * x + y0 + 32$	-2518	18902	16s
$r1 = r1 \gg 6$	$r1$ is $(32 * x + y0 + 32) \gg 6$	-39	295	11s
$r1 = \max(0, r1)$	clip $r1$ on the low side	0	295	10u
$r1 = \min(255, r1)$	clip $r1$ on the high side	0	255	8u

Ex-1003, ¶154.

Since Karczewicz-II teaches this optimization for averaging an interpolated half-pixel and an integer pixel, a POSITA would have been motivated to apply that improved calculation to Karczewicz-I's default weighted prediction equation, which likewise calculates an average, with the half-pixel prediction (e.g., $\text{pred0}(i,j)$) kept at a higher, non-rounded precision and the integer pixel prediction

(e.g., $\text{pred1}(i,j)$) left-shifted. *See* Ex-1005, ¶60, ¶55; Ex-1006, ¶¶93-94, ¶99, Table 3. As Karczewicz-II teaches, these values are combined with a rounding offset of 32 and then right-shifted 6 bits to reduce the precision. This combination results in the following equation:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{pred1}(i,j) \ll 5 + 32) \gg 6$$

See Ex-1006, ¶96, ¶99, Tables 1 and 3; Ex-1005, ¶60; Ex-1003, ¶¶155-156.

Scenario 2 (with the first motion vector pointing to a center-pixel position and the second motion vector pointing to a half-pixel position). The default weighted prediction is calculated as an average of a center-pixel and a half-pixel. *See* Ex-1005, ¶60, ¶55; Ex-1003, ¶157.

Karczewicz-II teaches an improved calculation for averaging a center- and half-pixel. Ex-1006, Fig. 4D. Karczewicz-II first explains the conventional calculation for averaging two numbers. Ex-1006, ¶96, Fig. 4D, Table 1:

TABLE 1

```

a = (C3 + b + 1) >> 1
c = (C4 + b + 1) >> 1
d = (C3 + h + 1) >> 1
l = (D3 + h + 1) >> 1
f = (j + b + 1) >> 1
i = (j + h + 1) >> 1
k = (j + ee + 1) >> 1
n = (j + hh + 1) >> 1

```

Ex-1003, ¶158.

Karczewicz-II improves this conventional approach by replacing the equations in Table 1 with those in Table 3, where the pixel values are combined at a higher precision. Ex-1006, ¶99. Whereas Table 1 used a fully-rounded center-pixel (e.g., “j”), Table 3 uses a partially-rounded pixel (e.g., “j1>>5”) that remains 5 bits longer than the rounded value in Table 1.¹⁴ *Infra* §V.B.6. Likewise, Table 3 replaces the rounded half-pixel (e.g., “b”) with a non-rounded half-pixel (e.g., “b1”) that is 15 bits. These pixel values are combined, along with a rounding offset of 32, before rounding to reduce the precision at the end, e.g., shifting 6 bits to the right (“>>6”). Ex-1006, Table 3:

TABLE 3

a =	(C3 << 5 + b1 + 32) >> 6
c =	(C4 << 5 + b1 + 32) >> 6
d =	(C3 << 5 + h1 + 32) >> 6
l =	(D3 << 5 + h1 + 32) >> 6
f =	(j1 >> 5 + b1 + 32) >> 6
i =	(j1 >> 5 + h1 + 32) >> 6
k =	(j1 >> 5 + ee1 + 32) >> 6
n =	(j1 >> 5 + hh1 + 32) >> 6

Ex-1003, ¶159.

¹⁴ Here, the partially-rounded prediction is obtained by shifting the non-rounded prediction j1 to the right by 5 bits. This is fewer bits than the 10 bits shifted for calculating the fully rounded prediction j. Ex-1006, ¶95.

The operations for this improved approach are shown in Table 8 of Karczewicz-II. Ex-1006, ¶105, Table 8:

TABLE 8				
<u>positions {f, i, k, n} of FIGS. 4A-4D</u>				
Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	-2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	-9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	-4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	-7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	-7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	-235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

Ex-1003, ¶160.

Since Karczewicz-II teaches this optimization for averaging interpolated center- and half-pixels, a POSITA would have been motivated to apply that improved calculation to Karczewicz-I's default weighted prediction equation, which likewise calculates an average, with the center-pixel prediction partially rounded and the half-pixel prediction non-rounded. *See* Ex-1005, ¶60, ¶55; *e.g.*, Ex-1006, ¶10, ¶20, ¶¶23-24, ¶39, ¶74, ¶93, ¶¶99-103. This combination results in the following equation:

$$\text{pred}(i,j)=(\text{non-rounded}(\text{pred0}(i,j))\gg 5+\text{non-rounded}(\text{pred1}(i,j))+32)\gg 6$$

See Ex-1006, ¶96, ¶99, Tables 1 and 3; Ex-1005, ¶60; Ex-1003, ¶¶161-162.

Scenario 3 (with both motion vectors pointing to half-pixel positions). The default weighted prediction is calculated as an average of two half-pixels. See Ex-1005, ¶60, ¶55; Ex-1003, ¶163.

Karczewicz-II teaches an improved calculation for averaging two half-pixel values. Ex-1006, Fig. 4D. Karczewicz-II explains the conventional calculation for averaging two numbers. Ex-1006, ¶97, Fig. 4D, Table 2:

TABLE 2

$$\begin{aligned} e &= (b + h + 1) \gg 1 \\ g &= (b + ee + 1) \gg 1 \\ m &= (h + hh + 1) \gg 1 \\ o &= (ee + hh + 1) \gg 1 \end{aligned}$$

Ex-1003, ¶164.

Karczewicz-II improves this conventional approach replacing the equations in Table 2 with those in Table 4, where the pixel values are combined at a higher precision. Ex-1006, ¶101. Instead of using a rounded 8-bit half-pixel (e.g., “b” or “h”), Karczewicz-II delays the rounding step and instead uses *non-rounded* half-pixels (e.g., “b1” and “h1”) that are 15 bits each. *Id.* These pixel values are combined, along with a rounding offset of 32, before rounding to reduce the precision at the end, e.g., shifting 6 bits to the right (“>>6”). *Id.*, Table 4:

TABLE 4

$$e = (b1 + h1 + 32) \gg 6$$

$$g = (b1 + ee1 + 32) \gg 6$$

$$m = (h1 + hh1 + 32) \gg 6$$

$$o = (ee1 + hh1 + 32) \gg 6$$

Ex-1003, ¶165.

The operations for this improved approach are shown in Table 6 of Karczewicz-II. Ex-1006, ¶103, Table 6:

TABLE 6

<u>positions {e, g, m, o} of FIGS. 4A-4D</u>				
Operation	Comment	Min value	Max value	Register size
$r1 = y0$	$r1$ is $y0$ ($y0$ is a 1-D half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r2 = y1$	$r2$ is $y1$ ($y1$ is a 1-D half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r1 = r1 + r2$	$r1$ is $y0 + y1$	-5100	21420	16s
$r1 = r1 + 32$	$r1$ is $y0 + y1 + 32$	-5068	21452	16s
$r1 = r1 \gg 6$	$r1$ is $(y0 + y1 + 32) \gg 6$	-79	335	11s
$r1 = \max(0, r1)$	clip $r1$ on the low side	0	335	10u
$r1 = \min(255, r1)$	clip $r1$ on the high side	0	255	8u

Ex-1003, ¶166.

Since Karczewicz-II teaches this optimization for averaging two interpolated half-pixels, a POSITA would have been motivated to apply that improved

calculation to Karczewicz-I's default weighted prediction equation, which likewise calculates an average, with the half-pixel predictions non-rounded. *See* Ex-1005, ¶60, ¶55; Ex-1006, ¶¶97-101. This combination results in the following equation:

$$\text{pred}(i,j)=(\text{non-rounded}(\text{pred0}(i,j))+\text{non-rounded}(\text{pred1}(i,j)) +32)>>6$$

See Ex-1006, ¶97, ¶101, Tables 2 and 4; Ex-1005, ¶60; Ex-1003, ¶¶167-168.

Express TSM in the Art. It was well-known to use higher-precision intermediate values for video-coding computations; the prior art provides express TSM to apply such teachings from Karczewicz-II to related techniques in Karczewicz-I to achieve a higher accuracy. *See, e.g.,* Ex-1016, Abstract; Ex-1003, ¶169.

Preserving higher-precision intermediate values was known to benefit motion compensation. Ex-1008, 7:4-19 (“a prediction and a reconstruction for a block of a picture to be predicted ... *in a higher precision.*”), 16:22-17:36. This approach was known to result in smaller residual information. *Id.*; Ex-1003, ¶170. It was also known that preserving higher-precision intermediate values improved the accuracy of interpolation. Ex-1009, 000005; Ex-1010, ¶¶61-62; Ex-1003, ¶171.

Compatible Teachings. The combination would not have changed the principle of operation of either reference, but merely includes the use Karczewicz-II's known techniques to improve Karczewicz-I's similar devices or methods in the same way. Ex-1005, Figs. 1-2, ¶¶29-50, ¶53; Ex-1006, Figs. 1-2, ¶¶40-53, ¶56.

Given the similarities between Karczewicz-I and Karczewicz-II, a POSITA would have understood that Karczewicz-II's techniques are readily applicable to Karczewicz-I. Ex-1003, ¶172.

Moreover, the combination merely changes when rounding occurs for calculations that already included rounding. *See* Ex-1005, ¶¶60, ¶55; Ex-1006, ¶¶96-106, Tables 1-8. This minor implementation detail would not have changed the principle of operation of Karczewicz-I or Karczewicz-II. Ex-1003, ¶173.

Reasonable Expectation of Success. A POSITA would have had a reasonable expectation of success because the combination applies the math behind Karczewicz-II's improved calculations to Karczewicz-I's scenarios without further modification. Ex-1003, ¶174.

Furthermore, a POSITA would have been more than capable of applying Karczewicz-II's teachings because Karczewicz-II's calculations involve basic mathematic and logical operations (e.g., addition and bit-shifting) that were well known or taught in high-school or undergraduate-level courses, and basic video codec operations that were a core part of industry work in video codecs, as discussed above. *Supra* §I.B; Ex-1003, ¶175.

4. Independent Claims 1, 7, and 13

[1a]. A method for encoding a block of pixels, the method comprising:
--

Ground 2 teaches the preamble. Ex-1003, ¶¶176-182.

Karczewicz-I “relates to ... video encoding techniques that use bi-directional prediction” including for encoders. Ex-1005, ¶2, ¶¶8-9, ¶22, ¶¶29-30, ¶32, Fig. 1. The encoder “perform[s] ... inter-coding of blocks within video frames” that includes motion estimation and compensation operations. Ex-1005, ¶¶51-76, Fig. 2; Ex-1003, ¶177. Karczewicz-I encodes “blocks of pixel data[.]” *See, e.g.*, Ex-1005, ¶¶39-40; Ex-1003, ¶178.

Karczewicz-II “describes various interpolation techniques performed by an encoder and a decoder during the motion compensation process of video coding.” Ex-1006, Abstract, ¶2, ¶12, ¶4, ¶¶40-41, ¶43, ¶¶54-62, Figs. 1-2. Karczewicz-II encodes a block of pixels. Ex-1006, ¶49, ¶50; Ex-1003, ¶¶179-180.

It was obvious to modify Karczewicz-I’s encoding method, based on Karczewicz-II’s interpolation techniques. *Supra* §V.B.3 (explaining combination). The combination teaches a method for encoding a block of pixels, comprising the operations explained below for [1b]-[1g]. Ex-1003, ¶¶181-182.

[7a]. An apparatus for encoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:
--

Ground 2 teaches [7a]. Ex-1003, ¶¶183-190.

Ground 2 teaches a video encoder for encoding a block of pixels that performs operations described below for [7b]-[7g]. *Supra* §V.B.4[1a] (explaining encoder); Ex-1005, ¶¶29-30, ¶32, Fig. 1, ¶¶51-76, Fig. 2:

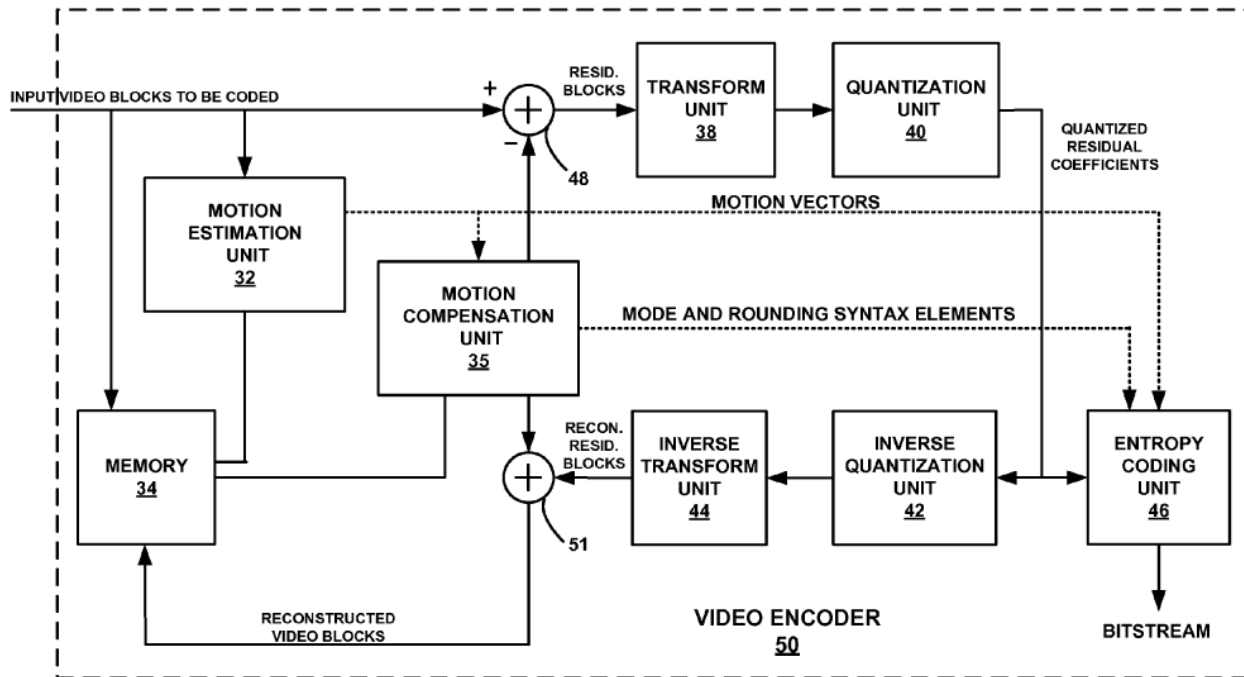


FIG. 2

Ex-1006, ¶¶40-41, ¶43, Fig. 1, ¶¶54-62, Fig. 2; Ex-1003, ¶¶184-185.

The video encoder encodes a block of pixels. Ex-1005, ¶¶39-40; Ex-1006, ¶¶49-50. Karczewicz-I and Karczewicz-II implement their encoder teachings in various types of devices (Ex-1005, ¶3; Ex-1006, ¶3) using a processor and memory (computer readable medium) that includes software, which comprises computer program code. Ex-1005, ¶12, ¶38, ¶¶98-100; Ex-1006, ¶¶25-27, ¶¶118-120. The medium comprises well-known memory types (e.g., RAM, ROM). Ex-1005, ¶99; Ex-1006, ¶119; Ex-1003, ¶¶186-188.

Karczewicz-I and Karczewicz-II teach the processor executing computer program code in memory to carry out their teachings. Ex-1005, ¶¶99-100; Ex-1006, ¶48, ¶¶119-120. The apparatus further performs operations described below for [7b]-[7g]. Ex-1003, ¶¶189-190.

[13a]. A computer program product for encoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

Ground 2 teaches the preamble. Ex-1003, ¶¶191-193.

As explained above, Karczewicz-I and Karczewicz-II implement their video-encoder teachings and perform operations for encoding blocks of pixels using software stored in computer-readable storage media and executed by a processor. *Supra* §V.B.4[7a]; Ex-1005, ¶12, ¶38, ¶¶98-100; Ex-1006, ¶¶25-26, ¶48, ¶¶119-120. These types of media (e.g., RAM, ROM) are non-transitory. Ex-1005, ¶99; Ex-1006, ¶119. The software includes computer-executable program code portions that comprise program code instructions. Ex-1005, ¶100; Ex-1006, ¶26, ¶120; Ex-1003, ¶192.

The software performs operations taught by Ground 2, including those described below for [13b-13g]. The combination further teaches forming “a computer program product” using the software media. Ex-1005, ¶99; Ex-1006,

¶119. Therefore, Ground 2 teaches a computer program product as recited in [13a] that performs the operations described below for [13b-13g]. Ex-1003, ¶193.

[1b]/[7b]/[13b] [determining/determine], for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

Ground 2 teaches [1b], [7b], and [13b]. Ex-1003, ¶¶194-202. Karczewicz-I and Karczewicz-II perform motion estimation and compensation. Ex-1005, ¶4, ¶51, ¶53, Fig. 2; Ex-1006, ¶4, ¶6, ¶54, ¶56, Fig. 2; Ex-1003, ¶¶194-195.

Motion estimation “generates motion vectors” that point to reference blocks (predictive video blocks) **for a current block** and “indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units.” Ex-1005, ¶7, ¶¶53-54; Ex-1006, ¶4, ¶56; Ex-1003, ¶196.

For motion compensation, the encoder determines reference blocks for further calculations and ascertains values of reference blocks, thereby **determining reference blocks** (e.g., predictive video blocks), **based on motion vectors**. Ex-1005, ¶7, ¶53; Ex-1006, ¶73, ¶4, ¶56, ¶58; Ex-1003, ¶197.

Karczewicz-I teaches video encoding using bi-directional prediction “from two lists of data[.]” Ex-1005, ¶6, ¶5, ¶22, ¶42, ¶54; Ex-1003, ¶198. The reference data from the two lists include data in two reference blocks and when the current

block is a B block having two motion vectors, two reference blocks are determined based on these two motion vectors. Ex-1005, ¶7, ¶¶53-54. Therefore, the combination teaches **determining, for a current block, a first reference block based on a first motion vector and second reference block based on a second motion vector**. Ex-1003, ¶199.

Karczewicz-II teaches **that the pixels of the current block, the first reference block, and the second reference block have values with a first precision (8 bits)**. “Precision” is at least satisfied by “a number of bits needed to represent possible values.” *Supra* §IV.A. Karczewicz-II teaches this precision (Ex-1006, ¶89; *supra* §IV.A), with 8 bits needed to represent possible pixel values for the two reference blocks. Table 5 shows that integer pixels (e.g., integer pixel x) include possible values between 0-255, and 8-bit unsigned numbers (“8u”) are needed to represent these possible values. Ex-1006, ¶103, Table 5:

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D				
Operation	Comment	Min value	Max value	Register size
$r1 = x$	$r1$ is integer pixel x	0	255	8u
$r1 = r1 \ll 5$	$r1$ is $32 * x$	0	8160	13u
$r2 = y0$	$r2$ is $y0$ ($y0$ is a one-dimensional (1-D) half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r1 = r1 + r2$	$r1$ is $32 * x + y0$	-2550	18870	16s
$r1 = r1 + 32$	$r1$ is $32 * x + y0 + 32$	-2518	18902	16s
$r1 = r1 \gg 6$	$r1$ is $(32 * x + y0 + 32) \gg 6$	-39	295	11s
$r1 = \max$ (0, $r1$)	clip $r1$ on the low side	0	295	10u
$r1 = \min$ (255, $r1$)	clip $r1$ on the high side	0	255	8u

Ex-1003, ¶¶200-201.

Therefore, the combination teaches or suggests that the pixels of the current block, the first reference block, and the second reference block have values with a first precision (8 bits). Ex-1003, ¶202.

[1c]/[7c]/[13c] [using/use] said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

Ground 2 teaches limitations [1c], [7c], and [13c]. Ex-1003, ¶¶203-217.

Karczewicz-I teaches calculating a bi-directional prediction using a default weighted prediction mode, with equal weights assigned to two reference blocks (Ex-1005, ¶¶55, ¶24, ¶44, ¶48), calculated as an average of two predictions (pred0(i,j) and pred1(i,j)). Ex-1005, ¶60:

Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) + \text{pred1}(i,j) + 1) >> 1$

Ex-1003, ¶¶204-205.

Here, pred0(i,j) is a first prediction based on a “motion compensated reference area[] ... obtained from list 0 ... reference picture.” Ex-1005, ¶¶57-58. Likewise, pred1(i,j) is obtained from list 1. *Id.* The motion compensated reference area refers to a reference block since Karczewicz-I teaches bi-directional prediction based on reference blocks. *Supra* §V.B.4[1b]/[7b]/[13b]; Ex-1005, ¶7, ¶¶53-54. The combination of Karczewicz-I and Karczewicz-II does not change the reference blocks on which Karczewicz-I’s predictions are based. *Supra* §V.B.3 (explaining combination). Thus, the combination teaches **using said first reference block (block from list 0) to obtain a first prediction.** Ex-1003, ¶¶206-207.

Karczewicz-I uses the reference block to obtain predictions in three scenarios (among others). *Supra* §V.B.3 (subsections explaining “Scenarios 1-3”). In each, the combination teaches said first prediction having a second precision higher than the first. Ex-1003, ¶208.

Scenario 1. For bi-directional prediction averaging a half-pixel prediction and an integer pixel prediction, the combination of Karczewicz-II to Karczewicz-I uses a non-rounded half-pixel prediction (non-rounded(pred0(i,j))) and a left-shifted integer pixel prediction (pred1(i,j)<<5). *Supra* §V.B.3 (explaining combined teachings for Scenario 1):

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{pred1}(i,j) \ll 5 + 32) \gg 6$$

Ex-1003, ¶209.

Karczewicz-II teaches that the non-rounded half-pixel prediction (e.g., b1) has possible values from -2550 to 10710 and that a 15-bit signed number (“15s”) is needed to represent these possible values. Karczewicz-II further teaches that the left-shifted integer pixel prediction (r1<<5) has possible values from 0 to 8160 and that a 13-bit unsigned number (“13u”) is needed to represent these possible values.

Ex-1006, ¶103, Table 5:

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D				
Operation	Comment	Min value	Max value	Register size
$r1 = x$	$r1$ is integer pixel x	0	255	8u
$r1 = r1 \ll 5$	$r1$ is $32 * x$	0	8160	13u
$r2 = y0$	$r2$ is $y0$ ($y0$ is a one-dimensional (1-D) half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r1 = r1 + r2$	$r1$ is $32 * x + y0$	-2550	18870	16s
$r1 = r1 + 32$	$r1$ is $32 * x + y0 + 32$	-2518	18902	16s
$r1 = r1 \gg 6$	$r1$ is $(32 * x + y0 + 32) \gg 6$	-39	295	11s
$r1 = \max(0, r1)$	clip $r1$ on the low side	0	295	10u
$r1 = \min(255, r1)$	clip $r1$ on the high side	0	255	8u

Ex-1003, ¶210.

The first and second predictions each need at least 13 bits to represent their possible values, which is higher than the 8-bit precision for the pixels of the current block, the first reference block, and the second reference block. *Supra* §V.B.4[1b]/[7b]/[13b]. Thus, the combination teaches the first prediction having a second precision (13 bits), which is higher than the first (8 bits) under Scenario 1. *Supra* §IV.A; Ex-1003, ¶211.

Scenario 2. For a center-pixel prediction and a half-pixel prediction, the combination of Karczewicz-II to Karczewicz-I uses a partially-rounded center-

pixel prediction ($\text{non-rounded}(\text{pred0}(i,j)) \gg 5$) and a non-rounded half-pixel prediction ($\text{non-rounded}(\text{pred1}(i,j))$). *Supra* §V.B.3 (subsection “Scenario 2” explaining calculations and combined teachings):

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) \gg 5 + \text{non-rounded}(\text{pred1}(i,j)) + 32) \gg 6$$

Ex-1003, ¶212.

Karczewicz-II teaches that the partially-rounded center-pixel prediction ($j1 \gg 1$)¹⁵ has possible values from -4957 to 13116 and that a 15-bit signed number (“15s”) is needed to represent these possible values. The non-rounded half-pixel prediction (e.g., b1) has possible values from -2550 to 10710, and a 15-bit signed integer (“15s”) is needed to represent these possible values. Ex-1006, ¶105, Table 8:

¹⁵ The equations in Table 3 of Karczewicz-II show that j1 is shifted to the right by 5 bits. Table 8 accomplishes this in two steps. The first step (“r2 = j1”) is to slightly round the value of j1 to fit in a 16-bit register, which shaves off 4 bits from the right side (the least significant bits). Ex-1006, ¶59, ¶10, ¶39, ¶53; *supra* §V.B.3. The second step (“r2 = r2 >> 1”) right shifts one more bit, bringing the total right-shift amount to 5 bits.

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D				
Operation	Comment	Min value	Max value	Register size
$r1 = y0$	$r1$ is $y0$ (1-D half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r2 = j1$	$r2$ is $j1$ (2-D half-pixel $j1$ before shifting down)	-9914	26232	16s
$r2 = r2 \gg 1$	$r2$ is $j1 \gg 1$	-4957	13116	15s
$r1 = r1 + r2$	$r1$ is $y0 + (j1 \gg 1)$	-7507	23826	16s
$r1 = r1 + 32$	$r1$ is $y0 + (j1 \gg 1) + 32$	-7491	23842	16s
$r1 = r1 \gg 6$	$r1$ is $(y0 + (j1 \gg 1) + 32) \gg 6$	-235	745	11s
$r1 = \max(0, r1)$	clip $r1$ on the low side	0	745	10u
$r1 = \min(255, r1)$	clip $r1$ on the high side	0	255	8u

Ex-1003, ¶213.

Therefore, for Scenario 2, the first and second predictions each have a second precision of 15 bits, which is higher than the first precision (8 bits) for, e.g., pixels of the current block. *Supra* §V.B.4[1b]/[7b]/[13b]. *Supra* §IV.A; Ex-1003, ¶214.

Scenario 3. For two half-pixel predictions, the combination of Karczewicz-II to Karczewicz-I uses non-rounded half-pixel predictions. *Supra* §V.B.3 (subsection “Scenario 3” explaining calculations and combined teachings):

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{non-rounded}(\text{pred1}(i,j)) + 32) \gg 6$$

Ex-1003, ¶215.

Karczewicz-II teaches that the non-rounded half-pixel predictions (e.g., b1, h1) each has values from -2550 to 10710, and a 15-bit signed number (“15s”) is needed to represent these possible values. Ex-1006, ¶103, Table 6:

TABLE 6

<u>positions {e, g, m, o} of FIGS. 4A-4D</u>				
Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hhl before shifting down)	-2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hhl before shifting down)	-2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	-5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	-5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	-79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

Ex-1003, ¶216.

Thus, for Scenario 3, the first and second predictions each have a second precision of 15 bits, which is higher than the first precision (8 bits) for, e.g., pixels of the current block. *Supra* §V.B.4[1b]/[7b]/[13b]; *supra* §IV.A; Ex-1003, ¶217.

[1d]/[7d]/[13d] [using/use] said second reference block to obtain a second prediction, said second prediction having the second precision;

Ground 2 teaches [1d], [7d], and [13d]. Ex-1003, ¶¶218-224.

As explained above, Karczewicz-I teaches calculating a bi-directional prediction as an average of two predictions, and $\text{pred1}(i,j)$ is a second prediction obtained based on a reference block. *Supra* §V.B.4[1c]/[7c]/[13c]; Ex-1005, ¶¶57-58, ¶60. The combination of Karczewicz-I and Karczewicz-II does not change the reference blocks on which Karczewicz-I's predictions are based. *Supra* §V.B.3 (explaining combination). Thus, the combination teaches **using said second reference block** (block from list 1) **to obtain a second prediction** in three scenarios. *Supra* §V.B.3 (subsections explaining Scenarios 1-3); Ex-1003, ¶¶219-220.

As explained above, in Scenario 1, the combination teaches **said second prediction** (left-shifted integer pixel prediction $\text{pred1}(i,j) \ll 5$) **having the second precision** (13 bits). *Supra* §V.B.4[1c]/[7c]/[13c]; Ex-1003, ¶222.

In Scenario 2, the combination teaches **said second prediction** (non-rounded half-pixel prediction $\text{non-rounded}(\text{pred1}(i,j))$) **having the second precision** (15 bits). *Supra* §V.B.4[1c]/[7c]/[13c]; Ex-1003, ¶223.

In Scenario 3, the combination teaches **said second prediction** (non-rounded half-pixel prediction $\text{non-rounded}(\text{pred1}(i,j))$) **having the second precision** (15 bits). *Supra* §V.B.4[1c]/[7c]/[13c]; Ex-1003, ¶224.

[1e]/[7e]/[13e] [obtaining/obtain] a combined prediction based at least partly upon said first prediction and said second prediction;

Ground 2 teaches [1e], [7e], and [13e]. Ex-1003, ¶¶225-231.

Karczewicz-I teaches obtaining a bi-directional prediction by averaging the first and second predictions. Ex-1005, ¶60 (adding predictions and dividing by two with “>>1”):

Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) + \text{pred1}(i,j) + 1) >> 1$

Supra §§V.B.4[1c-1d]/[7c-7d]/[13c-13d]. Ground 2 combines these bi-prediction teachings with Karczewicz-II. *Supra* §V.B.3 (explaining MTC); Ex-1003, ¶¶226-227.

Under each of the three scenarios discussed above (*supra* §V.B.3), the combination teaches **obtaining a combined prediction** (a sum of the first prediction, the second prediction, and a rounding offset) **based at least partly upon said first prediction and said second prediction**. Ex-1003, ¶228.

For Scenario 1, it was obvious to calculate the bi-directional prediction based on a non-rounded half-pixel prediction and a left-shifted integer pixel prediction:

$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{pred1}(i,j) << 5 + 32) >> 6$

Supra §V.B.3 (subsection “Scenario 1” explaining calculations). The combination calculates a sum of the first prediction (e.g., $\text{non-rounded}(\text{pred0}(i,j))$), the second

prediction (e.g., $\text{pred1}(i,j) \ll 5$) and a rounding offset (32). *See* Ex-1006, ¶103, Table 5; Ex-1003, ¶229.

Scenario 2, it was obvious to calculate the bi-directional prediction based on a partially-rounded center-pixel prediction and a non-rounded half-pixel prediction:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) \gg 5 + \text{non-rounded}(\text{pred1}(i,j)) + 32) \gg 6$$

Supra §V.B.3 (subsection “Scenario 2” explaining calculations). The combination calculates a sum of the first prediction (e.g., $\text{non-rounded}(\text{pred0}(i,j)) \gg 5$), the second prediction (e.g., $\text{non-rounded}(\text{pred1}(i,j))$), and a rounding offset (32). *See* Ex-1006, ¶105, Table 8; Ex-1003, ¶230.

Scenario 3, it was obvious to calculate the bi-directional prediction based on two non-rounded half-pixel predictions:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{non-rounded}(\text{pred1}(i,j)) + 32) \gg 6$$

Supra §V.B.3 (subsection “Scenario 3” explaining calculations). The combination calculates a sum of the first prediction (e.g., $\text{non-rounded}(\text{pred0}(i,j))$), the second prediction (e.g., $\text{non-rounded}(\text{pred1}(i,j))$), and a rounding offset (32). *See* Ex-1006, ¶103, Table 6; Ex-1003, ¶231.

[1f]/[7f]/[13f] [decreasing/decrease] a precision of said combined prediction by shifting bits of the combined prediction to the right; and

Ground 2 teaches [1f], [7f], and [13f]. Ex-1003, ¶¶232-239.

As explained above, Ground 2 teaches the combined prediction as a sum of the first prediction, the second prediction, and a rounding offset. *Supra* §V.B.4[1e]/[7e]/[13e]. Karczewicz-I and Karczewicz-II teach the operation “>>” for shifting bits to the right. Ex-1005, ¶57 (“>> is a right shift operation[.]”), ¶55, ¶60; Ex-1006, ¶94. The combination teaches **decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right** for each of the three scenarios discussed above. Ex-1003, ¶233.

Scenario 1. Here, the combination calculates bi-directional prediction based on a non-rounded half-pixel prediction and a left-shifted integer pixel prediction:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{pred1}(i,j) \ll 5 + 32) \gg 6$$

Supra §V.B.3 (subsection “Scenario 1” explaining calculations). Bits of the combined prediction $(\text{non-rounded}(\text{pred0}(i,j)) + \text{pred1}(i,j) \ll 5 + 32)$ are shifted to the right by 6 bits (e.g., “>>6”). Ex-1003, ¶234.

The shifting decreases a precision of said combined prediction. Ex-1006, ¶103, Table 5:

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D				
Operation	Comment	Min value	Max value	Register size
$r1 = x$	$r1$ is integer pixel x	0	255	8u
$r1 = r1 \ll 5$	$r1$ is $32 * x$	0	8160	13u
$r2 = y0$	$r2$ is $y0$ ($y0$ is a one-dimensional (1-D) half-pixel such as $b1$, $h1$, $ee1$ and $hh1$ before shifting down)	-2550	10710	15s
$r1 = r1 + r2$	$r1$ is $32 * x + y0$	-2550	18870	16s
$r1 = r1 + 32$	$r1$ is $32 * x + y0 + 32$	-2518	18902	16s
$r1 = r1 \gg 6$	$r1$ is $(32 * x + y0 + 32) \gg 6$	-39	295	11s
$r1 = \max(0, r1)$	clip $r1$ on the low side	0	295	10u
$r1 = \min(255, r1)$	clip $r1$ on the high side	0	255	8u

In this table, the operation “ $r1=r1+32$ ” calculates the sum of a rounding offset (32) with $r1$, which in turn is the sum of a non-rounded half-pixel prediction ($y0$ which can take the value of half-pixel prediction $b1$) and an integer pixel prediction that was left-shifted by 5 bits ($32*x$). Karczewicz-II teaches that this sum has possible values between -2518 and 18902 and that a 16-bit signed number (“16s”) is needed to represent these possible values. Next, the operation “ $r1=r1>>6$ ” shifts the sum 6 bits to the right. The result has possible values between -39 and 295; an 11-bit signed number (“11s”) is needed to represent these possible values. Thus, the

combination decreases a precision of said combined prediction from 16 bits to 11 bits. *Supra* §IV.A; Ex-1003, ¶235.

Scenario 2. Here, the combination calculates bi-directional prediction based on a partially-rounded center-pixel prediction ($\text{non-rounded}(\text{pred0}(i,j)) \gg 5$) and a non-rounded half-pixel prediction:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) \gg 5 + \text{non-rounded}(\text{pred1}(i,j)) + 32) \gg 6$$

Supra §V.B.3 (subsection “Scenario 2” explaining calculations). Bits of the combined prediction ($\text{non-rounded}(\text{pred0}(i,j)) \gg 5 + \text{non-rounded}(\text{pred1}(i,j)) + 32$) are shifted to the right (“ $\gg 6$ ”). Ex-1003, ¶236.

The shifting decreases a precision of said combined prediction. Ex-1006, ¶105, Table 8:

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D				
Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	-2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	-9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	-4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	-7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	-7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	-235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

The operation “r1=r1+32” calculates the sum of a partially-rounded center-pixel prediction (“j1>>1”), a non-rounded half-pixel prediction (y0, which may take the value of b1), and a rounding offset (32). Karczewicz-II teaches that this sum has possible values between -7491 and 23842 and that a 16-bit signed number (“16s”) is needed to represent these possible values. Next, the operation “r1=r1>>6” shifts the sum 6 bits to the right. The result has possible values between -235 and 745; an 11-bit signed number (“11s”) is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 bits to 11 bits. *Supra* §IV.A; Ex-1003, ¶237.

Scenario 3. Here, the combination calculates bi-directional prediction based on two non-rounded half-pixel predictions:

$$\text{pred}(i,j)=(\text{non-rounded}(\text{pred0}(i,j))+\text{non-rounded}(\text{pred1}(i,j)) +32)>>6$$

Supra §V.B.3 (subsection “Scenario 3” explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))+non-rounded(pred1(i,j)) +32) are shifted to the right (“>>6”). Ex-1003, ¶238.

The shifting decreases a precision of said combined prediction. Ex-1006, ¶103, Table 6:

TABLE 6				
<u>positions {e, g, m, o} of FIGS. 4A-4D</u>				
Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	-2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	-2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	-5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	-5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	-79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

The operation “r1=r1+32” calculates the sum of two non-rounded half-pixel predictions (y0 and y1, which may take the values of b1 and h1), and a rounding offset (32). Karczewicz-II teaches that this sum has possible values from -5068 to

21452, and a 16-bit signed number (“16s”) is needed to represent these possible values. Next, the operation “ $r1=r1>>6$ ” shifts the sum 6 bits to the right. The result has possible values from -79 to 335; an 11-bit signed number (“11s”) is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 to 11 bits. *Supra* §IV.A; Ex-1003, ¶239.

[1g]/[7g]/[13g] [encoding/encode] residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

Ground 2 teaches [1g], [7g], and [13g]. Ex-1003, ¶¶240-242.

The combination teaches determining the combined prediction in the motion compensation process. *Supra* §V.B.4[1b-1f]/[7b-7f]/[13b-13f]. The Karczewicz references teach **determining residual data** (e.g., a residual video block) **based upon a difference between the combined prediction** (e.g., prediction data) **and the block of pixels** (e.g., original video block). Ex-1005, ¶73 (“Video encoder 50 forms a residual video block by subtracting the prediction data from the original video block being coded.”), ¶7, ¶89; Ex-1006, ¶4, ¶6, ¶35, ¶50, ¶58, ¶60, ¶73, Fig. 2. Ex-1003, ¶241. The combination further teaches **encoding** (e.g., transforming, quantizing, and entropy coding) **residual data in a bitstream**. Ex-1005, ¶¶73-75, Fig. 2; Ex-1006, ¶5, ¶35, ¶50, ¶¶60-61, ¶110, Fig. 2; Ex-1003, ¶242.

5. Dependent Claims 2, 8, and 14

2. The method according to claim 1,

8. The apparatus according to claim 7,

14. The computer program product according to claim 13,

wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

Ground 2 teaches claims 1, 7, and 13, and the limitations of claims 2, 8, and 14. *Supra* §§V.B.3-4; Ex-1003, ¶¶243-251.

Karczewicz-II teaches obtaining predictions by interpolation using pixel values of reference blocks when motion vectors point to subpixels. Ex-1006, ¶42 (“The interpolation techniques of this disclosure may be performed by any encoding device that supports motion compensated interpolation to sub-pixel resolution.”), ¶7, ¶66, ¶¶68-72, Figs. 4A-4D; Ex-1003, ¶244.

Ground 2 satisfies this claim limitation for any of the three scenarios discussed above. Ex-1003, ¶245.

Scenario 1 is **an instance in which said first motion vector points to a subpixel** (a half-pixel position). *Supra* §V.B.3 (subsection “Scenario 1” explaining interpolation teachings). The half-pixel position refers to a subpixel. Ex-1006, ¶74, Figs. 4A-4D, ¶10; Ex-1003, ¶246.

Karczewicz-II teaches **obtaining the first prediction** (the non-rounded half-pixel prediction) **by interpolation using pixel values of said first reference block**. Ex-1006, ¶93 (H.264 defines one interpolation process for sub-pixels ...). The interpolation is performed using pixel values of “six closest integer pixels that surround” the subpixel, which are located in the reference block that the first motion vector points to. *See* Ex-1006, Fig. 4B:

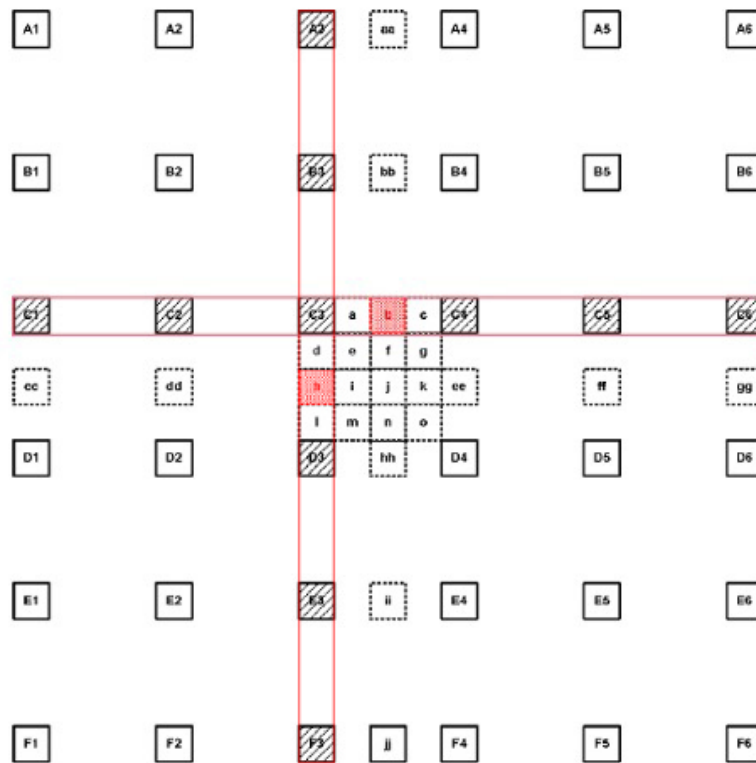


FIG. 4B

Thus, the interpolation is performed using pixel values of said first reference block. Ex-1003, ¶247.

Scenario 2 is **an instance in which said first motion vector points to a subpixel** (a center-pixel position). *Supra* §V.B.3 (subsection “Scenario 2” explaining interpolation teachings). The center-pixel position refers to a subpixel. Ex-1006, ¶74, Figs. 4A-4D, ¶10; Ex-1003, ¶248.

Karczewicz-II teaches **obtaining the first prediction** (the partially-rounded center-pixel prediction) **by interpolation using pixel values of said first reference block**. Ex-1006, ¶95 (“To interpolate sub-pixel ‘j,’ ...”). The interpolation is performed using non-rounded half-pixel predictions, which are located in the reference block that the first motion vector points to. *See* Ex-1006, Fig. 4C:

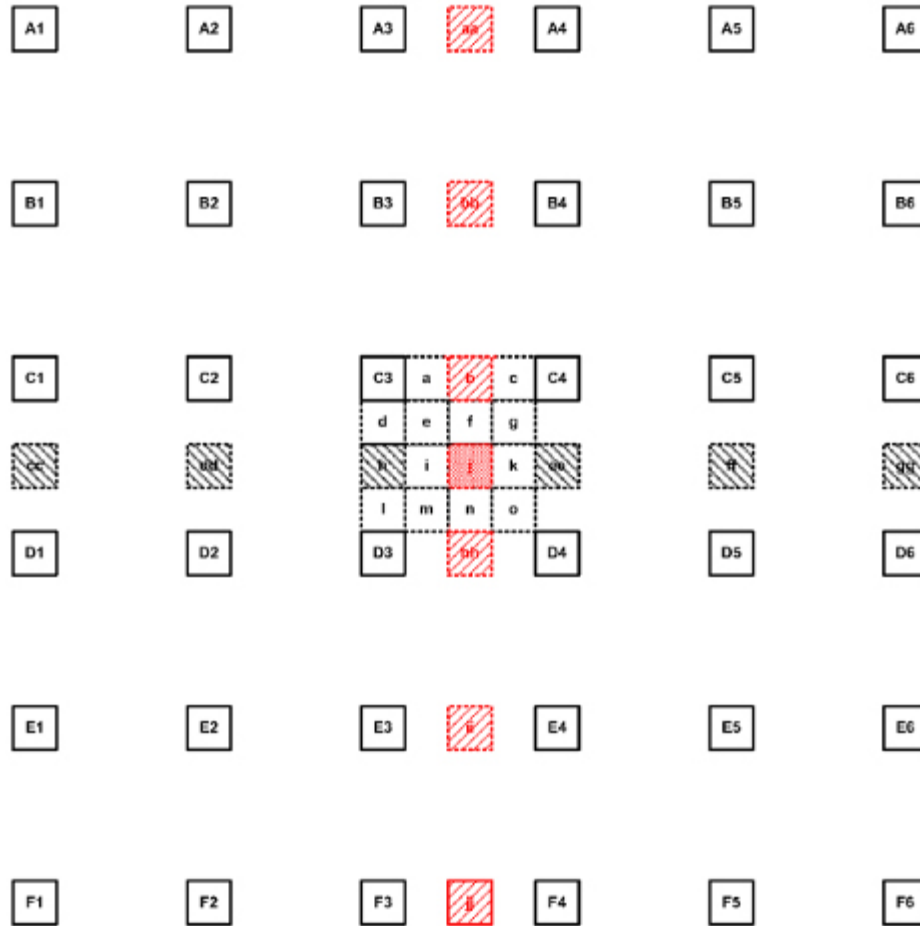


FIG. 4C

Thus, the interpolation is performed using pixel values of said first reference block.

Ex-1003, ¶249.

Scenario 3. Here, the first motion vector points to a half-pixel position.

Supra §V.B.3 (subsection “Scenario 3” explaining interpolation teachings). For the same reasons as explained for Scenario 1, the combination teaches that, in an instance in which said first motion vector points to a subpixel, said first prediction

is obtained by interpolation using pixel values of said first reference block. Ex-1003, ¶250.

Additionally, the '267 patent confirms that the limitations of claims 2, 8, and 14 were known in the prior art by describing it in the “Background Information” section. Ex-1001, 2:60-3:11; Ex-1003, ¶251.

6. Dependent Claims 3, 9, and 15

3. The method according to claim 2,

9. The apparatus according to claim 8,

15. The computer program product according to claim 14,

wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.

Ground 2 teaches claims 2, 8, and 14, and the limitations of claims 3, 9, and 15 under Scenario 2. *Supra* §V.B.5; Ex-1003, ¶¶252-257.

In Scenario 2, the combination teaches the following equation for bi-directional prediction:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) \gg 5 + \text{non-rounded}(\text{pred1}(i,j)) + 32) \gg 6$$

Supra §V.B.3 (subsection “Scenario 2” explaining prediction teachings). The first prediction “non-rounded(pred0(i,j)) >> 5” is a partially-rounded center-pixel prediction. *Id.*; *supra* §V.B.4[1c]/[7c]/[13c]; Ex-1003, ¶253.

Karczewicz-II teaches calculating the partially-rounded center-pixel prediction ($j1 \gg 5$) by first determining a sum of a 6-tap filter ($j1$). Ex-1006, ¶95:

$$j1 = aa1 - 5 * bb1 + 20 * b1 + 20 * hh1 - 5 * ii1 + jj1$$

Supra §V.B.5. This sum is a non-rounded version of the center-pixel prediction.

Ex-1003, ¶254.

Next, the sum is right-shifted by 5 bits to obtain the partially-rounded center-pixel prediction ($j1 \gg 5$). *Supra* §V.B.3; Ex-1006, ¶99, ¶105, Tables 8, 3:

TABLE 3

$$\begin{aligned} a &= (C3 \ll 5 + b1 + 32) \gg 6 \\ c &= (C4 \ll 5 + b1 + 32) \gg 6 \\ d &= (C3 \ll 5 + h1 + 32) \gg 6 \\ l &= (D3 \ll 5 + h1 + 32) \gg 6 \\ f &= (j1 \gg 5 + b1 + 32) \gg 6 \\ i &= (j1 \gg 5 + h1 + 32) \gg 6 \\ k &= (j1 \gg 5 + ee1 + 32) \gg 6 \\ n &= (j1 \gg 5 + hh1 + 32) \gg 6 \end{aligned}$$

Ex-1003, ¶255.

Therefore, the combination teaches that **said first prediction** (partially-rounded center-pixel prediction) **is obtained by interpolation using values of said first reference block by: right shifting (“ $\gg 5$ ”) a sum of a P-tap filter** (non-rounded center-pixel prediction “non-rounded(pred0(i,j))”) **using values of said first reference block.** Ex-1003, ¶256.

7. Dependent Claims 4, 10, and 16

4. The method according to claim 2,

10. The apparatus according to claim 8,

16. The computer program product according to claim 14,

wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

Ground 2 teaches claims 2, 8, and 14, and the limitations of claims 4, 10, and 16 under Scenario 1. *Supra* §V.B.5; Ex-1003, ¶¶258-263.

Scenario 1 is **an instance in which said second motion vector points to an integer sample**. *Supra* §V.B.3 (subsection “Scenario 1” explaining motion vector pointing to integer pixel position). Ex-1003, ¶259.

The combination calculates bi-directional prediction:

$$\text{pred}(i,j) = (\text{non-rounded}(\text{pred0}(i,j)) + \text{pred1}(i,j) \ll 5 + 32) \gg 6$$

Supra §V.B.3 (subsection “Scenario 1”). The second prediction “ $\text{pred1}(i,j) \ll 5$ ” is a left-shifted integer pixel prediction. *Supra* §V.B.3; §§V.B.4[1c-1d]/[7c-7d]/[13c-13d]; Ex-1003, ¶260.

Karczewicz-II teaches calculating the left-shifted integer pixel prediction by shifting the pixel value to the left (e.g., $C3 \ll 5$). *Supra* §V.B.3; Ex-1006, ¶99, Table 3:

TABLE 3

$a = (C3 \ll 5 + b1 + 32) \gg 6$
$c = (C4 \ll 5 + b1 + 32) \gg 6$
$d = (C3 \ll 5 + h1 + 32) \gg 6$
$l = (D3 \ll 5 + h1 + 32) \gg 6$
$f = (j1 \gg 5 + b1 + 32) \gg 6$
$i = (j1 \gg 5 + h1 + 32) \gg 6$
$k = (j1 \gg 5 + ee1 + 32) \gg 6$
$n = (j1 \gg 5 + hh1 + 32) \gg 6$

The left-shifted value is of an integer pixel sample (e.g., C3) that is part of the second reference block. Ex-1006, ¶74 (“integer-pixel sample ‘C3’”), ¶93; Ex-1003, ¶261.

Karczewicz-I and Karczewicz-II teach that motion compensation is performed on a block basis. *Supra* §§V.B.4[1b, 1g]/[7b, 7g]/[13b, 13g]; Ex-1005, ¶7; Ex-1006, ¶4, ¶73. Thus, the left-shift operation is performed for the pixels of the current block based on corresponding pixels of the reference block. *Supra* §V.B.4[1b]/[7b]/[13b]; Ex-1005, ¶7, ¶¶53-54; Ex-1006, ¶4, ¶56; Ex-1003, ¶262.

Therefore, the combination teaches that **in an instance in which said second motion vector points to an integer sample, said second prediction (left-shifted integer pixel prediction) is obtained by shifting values of said second reference block (values of integer pixels) to the left (“<<5”).** Ex-1003, ¶263.

8. Dependent Claims 5, 11, and 17

- 5. The method according to claim 1, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises:**
- 11. The apparatus according to claim 7, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by:**
- 17. The computer program product according to claim 13, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to:**
- [inserting/insert] a rounding offset to the combined prediction before said decreasing.**

Ground 2 teaches claims 1, 7, and 13, and the limitations of claims 5, 11, and 17. *Supra* §V.B.4; Ex-1003, ¶¶264-269.

As explained above, the combination teaches obtaining a combined prediction by summing the first prediction, the second prediction, and the rounding offset (e.g., 32) and decreasing a precision of the combined prediction. *Supra* §§V.B.4[1e-1f]/[7e-7f]/[13e-13f]. The combination thus teaches **inserting a rounding offset (32) to the combined prediction, before said decreasing** of the precision. Ex-1003, ¶265.

The value added to the sum of the first and second predictions is a rounding offset. Karczewicz-I refers to the term that is added to the weighted sum of the first

and second predictions as a “rounding adjustment.” Ex-1005, ¶¶63, ¶55.¹⁶ Because “rounding adjustment” was used interchangeably with “rounding offset” and the value (32) is inserted to the combined prediction right before the rounding operation, this value is a rounding offset. Ex-1003, ¶266.

Karczewicz-I and Karczewicz-II teach this rounding offset as part of its rounding process, which decreases the precision as recited by the claims. Ex-1005, ¶¶55, ¶63; Ex-1006, ¶¶96-101, Tables 1-4. Additionally, it was obvious for said decreasing to include the rounding offset because the insertion of the rounding offset is performed immediately before the right-shifting to affect the direction of the rounding. The combination includes a rounding offset to control rounding error resulting from the right-shift operation that decreases precision. This was common in the art. *Supra* I.B; Ex-1003, ¶267.

Because the combination applies its operations to a computer implementation, it teaches 1) at least one memory and computer program code being configured to cause the apparatus to perform the operations of claim 11 (*supra* §V.B.4[7a]); and 2) program code instructions configured to perform the operations of claim 17 (*supra* §V.B.4[13a]). Ex-1003, ¶¶268-269.

¹⁶ Karczewicz-I teaches a rounding adjustment of 2^{r-1} prior to right-shifting r bits, consistent with Ground 2’s modified equation for calculating bi-directional prediction under each scenario (*supra* §V.B.3), which teaches right-shifting 6 bits and a rounding offset of $2^{6-1}=2^5=32$.

9. Dependent Claims 6, 12, and 18

6. The method according to claim 1,

12. The apparatus according to claim 7,

18. The computer program product according to claim 13,

wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

Ground 2 teaches claims 1, 7, and 13, and the limitations of claims 6, 12, and 18. *Supra* §V.B.4; Ex-1003, ¶¶270-272.

The first precision indicates a number of bits needed to represent the values of the pixels. The combination teaches that the pixels of the current block, the first reference block, and the second reference block have values with a first precision because 8 bits are needed to represent the possible pixel values of these blocks. *Supra* §V.B.4[1b]/[7b]/[13b]; Ex-1003, ¶271.

The second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction. The combination teaches said first prediction and second prediction having a second precision that is higher than said first precision because more bits are needed to represent the possible values of the predictions under each of Scenarios 1, 2, and 3. *Supra* §§V.B.4[1c-1d]/[7c-7d]/[13c-13d]; Ex-1003, ¶272.

VI. DISCRETIONARY ANALYSIS FOR REVIEW

A. *Fintiv*

Discretionary denial is not warranted. Since the '267 patent was asserted in a parallel ITC proceeding, the district court proceedings are subject to mandatory stays. 28 U.S. Code § 1659. “The PTAB will not discretionarily deny petitions based on applying *Fintiv* to a parallel ITC proceeding.” June 21, 2022 Memorandum, 5-7. Even under *Fintiv*, this Petition’s merits are compelling, which “alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*.” *See id.*, 3-5. No case management conference has occurred. No case schedule has issued.

B. *Advanced Bionics*

Denial is not warranted under §325(d). Walker and Karczewicz-I were not cited or considered during prosecution. Ex-1003, ¶46, ¶127. Karczewicz-II was cited, but in an Information Disclosure Statement along with 76 other references (Ex-1002, 000074-78; Ex-1003, ¶131) and no substantive analysis in the file history. *See* Ex-1002. Grounds based on references cited in an IDS but not considered at any length during prosecution should not serve as basis for denial under §325(d). *Becton*, IPR2017-01586, Paper 8 at 17-18 (PTAB Dec. 15, 2017) (Informative).

VII. STANDING

Petitioner certifies that Petitioner is not barred or estopped from requesting this *inter partes* review and the '267 patent is IPR-eligible.

VIII. MANDATORY NOTICES AND FEES

1. **Real Party In Interest:** The real parties-in-interest here are Amazon.com, Inc. and Amazon.com Services LLC. No other parties directed, controlled, or funded this *Inter Partes* Review proceeding (IPR).

2. **Related Matters:**

- *Nokia Corporation and Nokia Technologies Oy v. Amazon.com, Inc. and Amazon.com Services LLC*, 1-23-cv-01232 (D. Del.)
- *Nokia Technologies Oy v. Amazon.com, Inc. et al*, 1-23-cv-01236 (D. Del.)
- *Nokia Technologies Oy v. HP, Inc.*, 1:23-cv-01237 (D. Del.)
- Certain Video Capable Electronic Devices, Including Computers, Streaming Devices, Televisions, and Components and Modules Thereof; Inv. No. 337-TA-1380 (ITC)

3. **Lead and Backup Counsel:** Pursuant to 37 C.F.R. §42.8(b)(3) and 42.10(a), Petitioner designates Harper Batts (Reg. No. 56,160) as lead counsel and Chris Ponder (Reg. No. 77,167) and Jeffrey Liang (Reg. No. 69,043) as back-up counsel, each of Sheppard, Mullin, Richter & Hampton LLP.

4. **Service Information:** Petitioner consents to service by electronic mail addressed to:

HBatts@sheppardmullin.com;
CPonder@sheppardmullin.com; and

JLiang@sheppardmullin.com

5. Petitioner's counsel may also be served by mail or hand delivery at Sheppard, Mullin, Richter & Hampton LLP, 1540 El Camino Real Suite 120, Menlo Park, CA 94025. Petitioner's counsel may be reached by telephone at (650) 815-2673 and by facsimile at (650) 815-4668.

6. **Fees:** The Office is authorized to charge fees for this Petition to Deposit Account 50-4561, Ref. 16AG-382973.

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/

Harper Batts (Reg. No. 56,160)
Attorneys for Petitioner

CLAIM APPENDIX

[1a] A method for encoding a block of pixels, the method comprising:

[1b] determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

[1c] using said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

[1d] using said second reference block to obtain a second prediction, said second prediction having the second precision;

[1e] obtaining a combined prediction based at least partly upon said first prediction and said second prediction;

[1f] decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and

[1g] encoding residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

2. The method according to claim 1, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

3. The method according to claim 2, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

4. The method according to claim 2, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

5. The method according to claim 1, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises:

inserting a rounding offset to the combined prediction before said decreasing.

6. The method according to claim 1, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

[7a] An apparatus for encoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the

at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:

[7b] determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

[7c] use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

[7d] use said second reference block to obtain a second prediction, said second prediction having the second precision;

[7e] obtain a combined prediction based at least partly upon said first prediction and said second prediction;

[7f] decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and

[7g] encode residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

8. The apparatus according to claim 7, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

9. The apparatus according to claim 8, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

10. The apparatus according to claim 8, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

11. The apparatus according to claim 7, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by:

inserting a rounding offset to the combined prediction before said decreasing.

12. The apparatus according to claim 7, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

[13a] A computer program product for encoding a block of pixels, the computer program product comprising at least one non-transitory computer readable

storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

[13b] determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

[13c] use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

[13d] use said second reference block to obtain a second prediction, said second prediction having the second precision;

[13e] obtain a combined prediction based at least partly upon said first prediction and said second prediction;

[13f] decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and

[13g] encode residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

14. The computer program product according to claim 13, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

15. The computer program product according to claim 14, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

16. The computer program product according to claim 14, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

17. The computer program product according to claim 13, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to:

insert a rounding offset to the combined prediction before said decreasing.

18. The computer program product according to claim 13, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that on March 15, 2024, true and correct copies of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 11,805,267 and Exhibits 1001-1016 were served in entirety on the Patent Owner at the following addresses of record as listed on Patent Center, via Priority Mail Express®:

Nokia Corporation and Alston & Bird LLP
Vantage South End
1120 South Tryon Street, Suite 300
Charlotte, NC 28203-6818
Official Correspondence Address

The undersigned certifies that true and correct copies of the foregoing Petition for *Inter Partes* Review of U.S. Patent No. 11,805,267 and Exhibits 1001-1016 were also sent via electronic mail to the attorneys of record for Plaintiff in the concurrent litigation matter(s):

Brian E. Farnan
bfarnan@farnanlaw.com
Michael J. Farnan
mfarnan@farnanlaw.com
John D Haynes
john.haynes@alston.com
Nicholas T. Tsui
nick.tsui@alston.com
Shawn P. Gannon
shawn.gannon@alston.com
Theodore Stevenson, III
ted.stevenson@alston.com
Stephen R. Lareau
stephen.lareau@alston.com

Warren H. Lipschitz
wlipschitz@mckoolsmith.com
Alexandra F. Easley
aeasley@mckoolsmith.com
Joshua Budwin
jbudwin@mckoolsmith.com
R. Mitch Verboncoeur
mverboncoeur@mckoolsmith.com
Joshua Newcomer
jnewcomer@mckoolsmith.com
Kevin Burgess
kburgess@mckoolsmith.com

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/

Harper Batts (Reg. No. 56,160)
Attorneys for Petitioner

Date: March 15, 2024

1540 El Camino Real Suite 120
Menlo Park, CA 94025
Tel: (650) 815-2673

**CERTIFICATE OF COMPLIANCE WITH
TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS,
AND TYPE STYLE REQUIREMENTS**

1. This Petition complies with the type-volume limitation of 14,000 words, comprising 13,382 words, as counted using the Microsoft Word software that was used to prepare this paper, excluding the parts exempted by 37 C.F.R. § 42.24(a).

2. This Petition complies with the general format requirements of 37 C.F.R. § 42.6(a) and has been prepared using Microsoft® Word in 14-point Times New Roman.

Respectfully submitted,

SHEPPARD, MULLIN,
RICHTER & HAMPTON LLP

/Harper Batts/
Harper Batts (Reg. No. 56,160)
Attorney for Petitioner

Date: March 15, 2024

1540 El Camino Real Suite 120
Menlo Park, CA 94025
Tel: (650) 815-2673