

EXHIBIT E-1

U.S. Patent No. 7,620,109B2 (“Srinivasan”) in view of

U.S. Patent No 9,161,057 (“Karczewicz”) and/or U.S. Patent No. 8,606,028 to Noda et al. (“Noda”)

RENDERS OBVIOUS U.S. PATENT NO. 11,805,267

As described in the following claim chart, claims 1-36 of U.S. Patent No. 11,805,267 (the “’267 patent”) are invalid in view of U.S. Patent No. 7,620,109 (“Srinivasan”), entitled Sub-pixel interpolation in motion estimation and compensation, published on October 30, 2003. Srinivasan was filed on April 10, 2002, and is prior art under at least pre-AIA-35 U.S.C. § 102 (a) and/or 103. Claims 1-36 of the ’267 patent are rendered obvious in view of Srinivasan in combination with U.S. Patent No 9,161,057 (“Karczewicz”) and/or U.S. Patent No. 8,606,028 to Noda et al. (“Noda”). Karczewicz is prior art to the ’267 patent under at least pre-AIA U.S.C. § 102 (a) and (e) and/or 103. Karczewicz was filed on July 9, 2009 and published on January 13, 2011. Noda is prior art to the ’267 patent under at least pre-AIA U.S.C. § 102 (a) and/or 103. Noda was published on April 2, 2009.

To the extent Complainant alleges that Srinivasan does not disclose any particular limitation of the asserted claims of the ’267 patent, either expressly or inherently, it would have been obvious to a person of ordinary skill in the art as of the priority date of ’267 patent to modify Srinivasan and/or to combine the teachings of Srinivasan with: (1) the knowledge of one of ordinary skill in the art to show all the limitations of the claim; (2) the teachings of the prior art references set forth in Respondent’s other ’267 patent invalidity charts with respect to the one or more relevant limitations; and/or (3) the teachings of the prior art references cited and discussed in Respondent’s interrogatory responses accompanying its invalidity contentions.

To the extent that these Invalidity Contentions rely on or otherwise embody particular constructions of terms or phrases in the Asserted Claims, Respondents are not proposing any such constructions as proper constructions of those terms or phrases. Various positions put forth in this document are predicated on Complainants’ incorrect and overly broad interpretation of its claims. Those positions are not intended to and do not necessarily reflect Respondents’ interpretation of the true and proper scope of Complainants’ claims, and Respondents reserve the right to adopt claim construction positions that differ from or even conflict with various positions put forth in this document.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
1[preamble] A method for encoding a block of pixels, the method comprising:	Srinivasan, as evidenced by the example citations below, discloses a method for encoding a block of pixels. <i>See, e.g.</i>, Srinivasan at 1:20-26. TECHNICAL FIELD Techniques and tools for sub-pixel shifting in motion estimation/compensation in video coding/decoding applications are described. For example, a video encoder and decoder defer bit shifting when performing multi-stage interpolation for pixel values at sub-pixel positions in a reference frame. <i>See, e.g.</i>, Srinivasan at 6:1-2. FIG. 4 is a block diagram of a generalized video encoder system used in several described embodiments. <i>See, e.g.</i>, Srinivasan at 8:47-10:46. II. Generalized Video Encoder and Decoder FIG. 4 is a block diagram of a generalized video encoder (400) and FIG. 5 is a block diagram of a generalized video decoder (500). The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, FIGS. 4 and 5 usually do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information

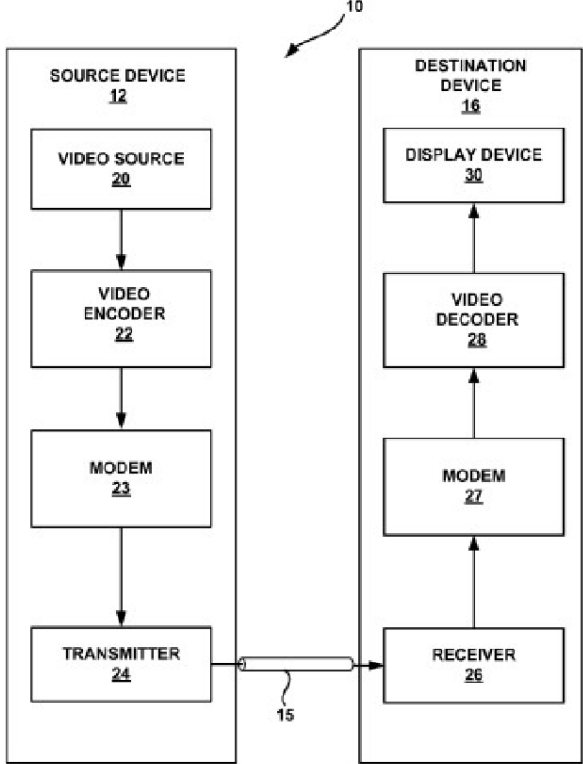
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	is sent in the output bitstream, typically after entropy encoding of the side information. The format of the output bitstream can be a Windows Media Video format or another format. The encoder (400) and decoder (500) are block-based and use a 4:2:0 macroblock format with each macroblock including 4 luminance 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks (e.g., one U block, one V block). Alternatively, the encoder (400) and decoder (500) are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration than 8×8 blocks and 16×16 macroblocks. Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques. A. Video Encoder FIG. 4 is a block diagram of a general video encoder system (400). The encoder system (400) receives a sequence of video frames including a current frame (405), and produces compressed video information (495) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (400). The encoder system (400) compresses predicted frames and key frames. For the sake of presentation, FIG. 4 shows a path for key frames through the encoder system (400) and a path for forward-predicted frames. Many of the components of the encoder system (400) are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A predicted frame [also called p-frame, b-frame for bidirectional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other frames. A prediction

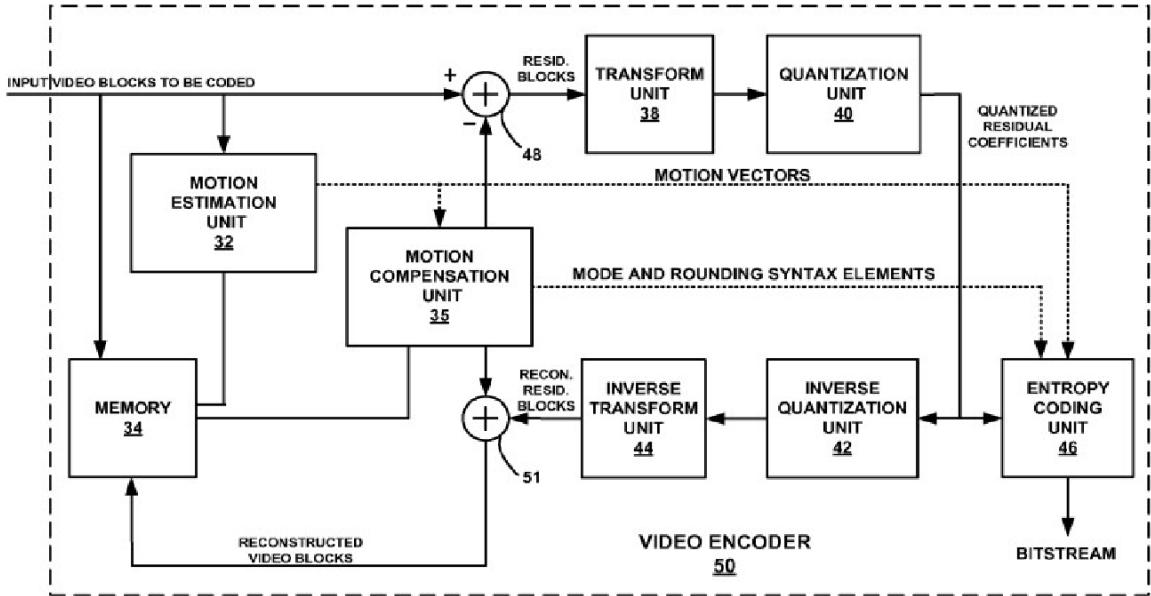
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	residual is the difference between what was predicted and the original frame. In contrast, a key frame [also called i-frame, intra-coded frame] is compressed without reference to other frames. If the current frame (405) is a forward-predicted frame, a motion estimator (410) estimates motion of macroblocks or other sets of pixels of the current frame (405) with respect to a reference frame, which is the reconstructed previous frame (425) buffered in the frame store (420). In alternative embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted. The motion estimator (410) outputs as side information motion information (415) such as motion vectors. A motion compensator (430) applies the motion information (415) to the reconstructed previous frame (425) to form a motion-compensated current frame (435). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (435) and the original current frame (405) is the prediction residual (445). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation. A frequency transformer (460) converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer (460) applies a discrete cosine transform [“DCT”] or variant of DCT to blocks of the motion prediction residual data, producing blocks of DCT coefficients. Alternatively, the frequency transformer (460) applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. In some embodiments, the frequency transformer (460) applies a frequency transform to blocks of spatial prediction residuals for key frames. The frequency transformer (460) can apply an 8×8, 8×4, 4×8, or other size frequency transforms. A quantizer (470) then quantizes the blocks of spectral data coefficients. The quantizer applies uniform, scalar quantization to the spectral data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the quantizer applies another type of quantization to the spectral data coefficients, for example, a non-uniform, vector, or non-adaptive quantization, or directly quantizes spatial domain data in an encoder system that does not use frequency transformations. In addition to

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	adaptive quantization, the encoder (400) can use frame dropping, adaptive filtering, or other techniques for rate control. When a reconstructed current frame is needed for subsequent motion estimation/compensation, an inverse quantizer (476) performs inverse quantization on the quantized spectral data coefficients. An inverse frequency transformer (466) then performs the inverse of the operations of the frequency transformer (460), producing a reconstructed prediction residual (for a predicted frame) or a reconstructed key frame. If the current frame (405) was a key frame, the reconstructed key frame is taken as the reconstructed current frame (not shown). If the current frame (405) was a predicted frame, the reconstructed prediction residual is added to the motion-compensated current frame (435) to form the reconstructed current frame. The frame store (420) buffers the reconstructed current frame for use in predicting the next frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame. The entropy coder (480) compresses the output of the quantizer (470) as well as certain side information (e.g., motion information (415), quantization step size). Typical entropy coding techniques include arithmetic coding, differential coding, Huffman coding, run length coding, LZ coding, dictionary coding, and combinations of the above. The entropy coder (480) typically uses different coding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular coding technique. The entropy coder (480) puts compressed video information (495) in the buffer (490). A buffer level indicator is fed back to bitrate adaptive modules. The compressed video information (495) is depleted from the buffer (490) at a constant or relatively constant bitrate and stored for subsequent streaming at that bitrate. Alternatively, the encoder system (400) streams compressed video information immediately following compression.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Before or after the buffer (490), the compressed video information (495) can be channel coded for transmission over the network. The channel coding can apply error detection and correction data to the compressed video information (495). See, e.g., Srinivasan at FIG. 4. Figure 4 The diagram illustrates two video compression paths. The Intraframe Path (top) starts with a Current frame 405 entering a Frequency transformer 460, followed by a Quantizer 470, an Entropy coder 480, and finally a Buffer 490 which outputs Compressed video info 495. The Interframe Path (bottom) starts with a Current frame 405 and a Reference frame 425. The current frame is processed by a Motion estimator 410 and a Motion compensator 430. The reference frame is stored in a Frame store 420 and also fed into the motion compensator. The motion compensator outputs a residual (435) which is subtracted from the current frame at a subtraction node (445). The resulting Residual 445 is then processed by a Frequency transformer 460, a Quantizer 470, an Entropy coder 480, and a Buffer 490 to produce Compressed video info 495. This compressed information is then passed through an Inverse quantizer 476 and an Inverse frequency tran. 466 to a summation node (415). The output of the summation node is fed back into the motion compensator (430) to refine the motion compensation process.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	This element would have further been known in the art and otherwise disclosed in the other references charted in Respondents’ Invalidity Contentions, at least for the reasons set forth in these Contentions. For example, Karczewicz discusses a method for encoding a block of pixels. <i>See, e.g.</i>, Karczewicz at 1:20-34. Digital multimedia capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless communication devices, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, digital cameras, digital recording devices, video gaming devices, video game consoles, cellular or satellite radio telephones, digital media players, and the like. Digital multimedia devices may implement video coding techniques, such as MPEG-2, ITU-H.263, MPEG-4, or ITU-H.264/MPEG-4 Part 10, Advanced Video Coding (AVC), to transmit and receive or store and retrieve digital video data more efficiently. Video encoding techniques may perform video compression via spatial and temporal prediction to reduce or remove redundancy inherent in video sequences. <i>See, e.g.</i>, Karczewicz at 4:37-42. FIG. 1 is a block diagram illustrating one exemplary video encoding and decoding system that may implement techniques of this disclosure. FIG. 2 is a block diagram illustrating an example of a video encoder that may perform offset techniques consistent with this disclosure. <i>See, e.g.</i>, Karczewicz at FIG. 1.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	 FIG. 1 <i>See, e.g., Karczewicz at 7:38-51.</i> Video encoder 22 and video decoder 28 may operate according to a video compression standard, such as the ITU-T H.264 standard, alternatively described as MPEG-4, Part 10, Advanced Video Coding (AVC). The techniques of this disclosure, however, are not limited to any particular coding

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	standard. Although not shown in FIG. 1, in some aspects, video encoder 22 and video decoder 28 may each be integrated with an audio encoder and decoder, and may include appropriate MUX-DEMUX units, or other hardware and software, to handle encoding of both audio and video in a common data stream or separate data streams. If applicable, MUX-DEMUX units may conform to the ITU H.223 multiplexer protocol, or other protocols such as the user datagram protocol (UDP). See, e.g., Karczewicz at FIG. 2.  FIG. 2 See, e.g., Karczewicz at 8:37-42.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	A video sequence typically includes a series of video frames. Video encoder 22 and video decoder 28 may operate on video blocks within individual video frames in order to encode and decode the video data. The video blocks may have fixed or varying sizes, and may differ in size according to a specified coding standard. Each video frame may include a series of slices or other independently decodable units. Each slice may include a series of macroblocks, which may be arranged into sub-blocks. As an example, the ITU-T H.264 standard supports intra prediction in various block sizes, such as 16 by 16, 8 by 8, or 4 by 4 for luma components, and 8×8 for chroma components, as well as inter prediction in various block sizes, such as 16 by 16, 16 by 8, 8 by 16, 8 by 8, 8 by 4, 4 by 8 and 4 by 4 for luma components and corresponding scaled sizes for chroma components. Video blocks may comprise blocks of pixel data, or blocks of transformation coefficients, e.g., following a transformation process such as discrete cosine transform or a conceptually similar transformation process.
1[a] 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;	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See, e.g., Srinivasan at 2:13-35.</i> For a macroblock of a predicted frame, the WMV8 encoder performs motion estimation. The motion estimation approximates the motion of a macroblock in a predicted frame by searching for and matching the macroblock in the predicted frame with a macroblock from a reference frame. In FIG. 1, for instance, the WMV8 encoder computes a motion vector for a macroblock (115) in the predicted frame (110). To compute the motion vector, the encoder searches in a search area (135) of a reference frame (130). Within the search area (135), the encoder compares the luminance values of the macroblock (115) from the predicted frame (110) to the luminance values of various candidate blocks from the reference frame (130) in order to find a good match. The WMV8 encoder may switch motion vector accuracy, and may use a search range and motion vectors with integer, half, or quarter-pixel

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	horizontal resolution and integer or half-pixel vertical resolution. With sub-pixel accurate motion vectors, the WMV8 encoder can approximate sub-pixel motion in a video sequence. During motion estimation or compensation, when a motion vector has subpixel accuracy (i.e., half-pixel or quarter-pixel), the WMV8 encoder or decoder must determine pixel values at sub-pixel positions in the reference frame. <i>See, e.g.</i>, Srinivasan at 6:1-2. FIG. 4 is a block diagram of a generalized video encoder system used in several described embodiments. <i>See, e.g.</i>, Srinivasan at 8:47-10:45. II. Generalized Video Encoder and Decoder FIG. 4 is a block diagram of a generalized video encoder (400) and FIG. 5 is a block diagram of a generalized video decoder (500). The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, FIGS. 4 and 5 usually do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bitstream, typically after entropy encoding of the side information. The format of the output bitstream can be a Windows Media Video format or another format. The encoder (400) and decoder (500) are block-based and use a 4:2:0 macroblock format with each macroblock including 4 luminance 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks (e.g., one U block, one V block). Alternatively, the encoder (400) and decoder (500) are object-based, use a different macroblock or block format, or perform

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	operations on sets of pixels of different size or configuration than 8×8 blocks and 16×16 macroblocks. Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques. A. Video Encoder FIG. 4 is a block diagram of a general video encoder system (400). The encoder system (400) receives a sequence of video frames including a current frame (405), and produces compressed video information (495) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (400). The encoder system (400) compresses predicted frames and key frames. For the sake of presentation, FIG. 4 shows a path for key frames through the encoder system (400) and a path for forward-predicted frames. Many of the components of the encoder system (400) are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A predicted frame [also called p-frame, b-frame for bidirectional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other frames. A prediction residual is the difference between what was predicted and the original frame. In contrast, a key frame [also called i-frame, intra-coded frame] is compressed without reference to other frames. If the current frame (405) is a forward-predicted frame, a motion estimator (410) estimates motion of macroblocks or other sets of pixels of the current frame (405) with respect to a reference frame, which is the reconstructed previous frame (425) buffered in the frame store (420). In alternative embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	The motion estimator (410) outputs as side information motion information (415) such as motion vectors. A motion compensator (430) applies the motion information (415) to the reconstructed previous frame (425) to form a motion-compensated current frame (435). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (435) and the original current frame (405) is the prediction residual (445). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation. A frequency transformer (460) converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer (460) applies a discrete cosine transform [“DCT”] or variant of DCT to blocks of the motion prediction residual data, producing blocks of DCT coefficients. Alternatively, the frequency transformer (460) applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. In some embodiments, the frequency transformer (460) applies a frequency transform to blocks of spatial prediction residuals for key frames. The frequency transformer (460) can apply an 8×8, 8×4, 4×8, or other size frequency transforms. A quantizer (470) then quantizes the blocks of spectral data coefficients. The quantizer applies uniform, scalar quantization to the spectral data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the quantizer applies another type of quantization to the spectral data coefficients, for example, a non-uniform, vector, or non-adaptive quantization, or directly quantizes spatial domain data in an encoder system that does not use frequency transformations. In addition to adaptive quantization, the encoder (400) can use frame dropping, adaptive filtering, or other techniques for rate control. When a reconstructed current frame is needed for subsequent motion estimation/compensation, an inverse quantizer (476) performs inverse quantization on the quantized spectral data coefficients. An inverse frequency transformer (466) then performs the inverse of the operations of the frequency transformer (460), producing a reconstructed prediction residual (for a predicted frame) or a reconstructed key frame. If the current frame (405) was a key frame, the reconstructed key frame is taken as the reconstructed current frame (not shown). If the current frame (405) was a predicted

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	frame, the reconstructed prediction residual is added to the motion-compensated current frame (435) to form the reconstructed current frame. The frame store (420) buffers the reconstructed current frame for use in predicting the next frame. In some embodiments, the encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame. The entropy coder (480) compresses the output of the quantizer (470) as well as certain side information (e.g., motion information (415), quantization step size). Typical entropy coding techniques include arithmetic coding, differential coding, Huffman coding, run length coding, LZ coding, dictionary coding, and combinations of the above. The entropy coder (480) typically uses different coding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular coding technique. The entropy coder (480) puts compressed video information (495) in the buffer (490). A buffer level indicator is fed back to bitrate adaptive modules. The compressed video information (495) is depleted from the buffer (490) at a constant or relatively constant bitrate and stored for subsequent streaming at that bitrate. Alternatively, the encoder system (400) streams compressed video information immediately following compression. Before or after the buffer (490), the compressed video information (495) can be channel coded for transmission over the network. The channel coding can apply error detection and correction data to the compressed video information (495). <i>See, e.g., Srinivasan at 21:22-52.</i> FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. <i>See, e.g., Srinivasan at FIG. 14.</i>

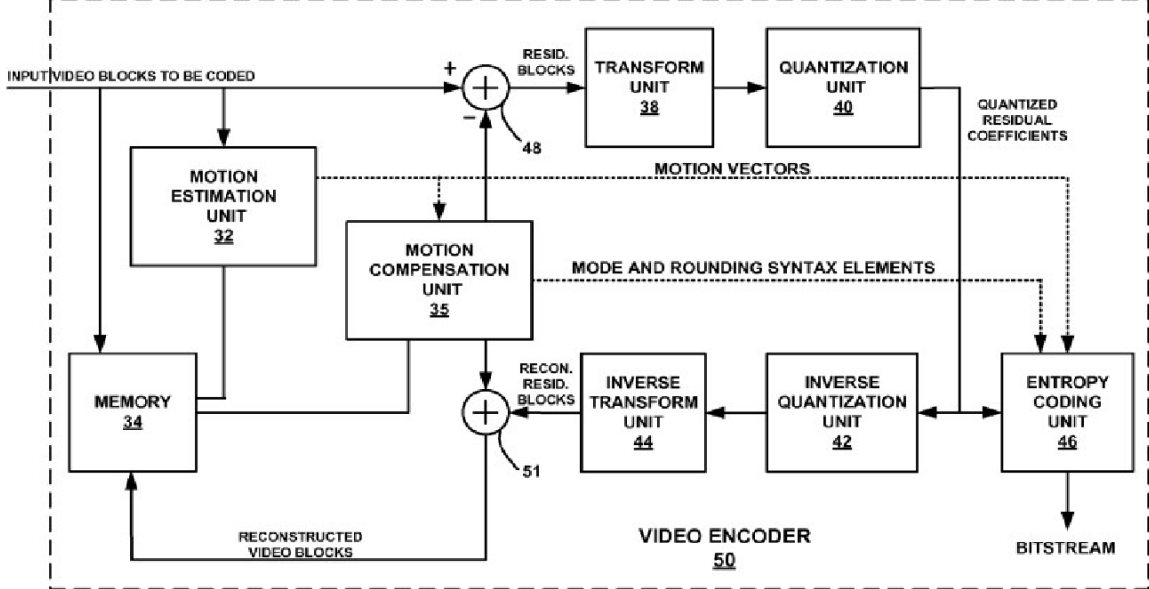
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 14 FIRST STAGE 1410 x-bit range input value(s) 1411 → 1412 Apply first filter F_1 → 1414 Perform shortened shift → 1400 y-bit range intermediate value(s) 1420, where $y > x$ ----- • INTERMEDIATE STAGES 1422 • ----- LAST STAGE 1430 z-bit range intermediate value(s) 1429, where $z > x$ → 1432 Apply last filter F_L → 1434 Perform lengthened shift → output value 1436 This element would have further been known in the art and otherwise disclosed in the other references charted in Respondents’ Invalidity Contentions, at least for the reasons set forth in these Contentions. For example, JVT-R201 (Joint Draft 5: Scalable Video Coding (in integrated form with ITU-T Rec. H.264 ISO/IEC 14996-10, published on January 2006) and Noda disclose that the pixels of the current block, the first reference block, and the second reference block have values with a first precision. See e.g., JVT-R201 at 66 (“bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array and the value of the luma quantisation parameter range offset $QpBdOffset_y$”), 67

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	(“bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantisation parameter range offset QpBdOffsetc”), 161 (“8.3.2.1 Reference picture selection process”); Noda at 24:51-56 (“There will now be described an operation of the image encoding apparatus with reference to flowchart of FIG.20B. A video signal wherein the bit depth of each pixel is, for example, N bit depth is input to this image encoding apparatus as an input image signal 100 in units of a frame, for example (S121).”), 25: 15-18 (“The local decoded picture signal 112 of the N bit depth stored in the frame memory 114 is input to the pixel bit depth increase unit 1001, to increase a value of each pixel by Mbits (S126)”). A person of ordinary skill in the art would have found it obvious to incorporate these teachings into Srinivasan and would have had a reasonable expectation of success in doing so, at least because it was well known in the art to have the same precision for pixels of the current block and reference block. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (the same precision for pixels of the current block and reference block) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance). In addition, the following disclosures of Karczewicz would satisfy or render obvious 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. <i>See, e.g., Karczewicz at 1:12-34.</i> TECHNICAL FIELD The disclosure relates to video encoding and, more particularly, video encoding techniques that use bidirectional prediction.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	BACKGROUND Digital multimedia capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless communication devices, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, digital cameras, digital recording devices, video gaming devices, video game consoles, cellular or satellite radio telephones, digital media players, and the like. Digital multimedia devices may implement video coding techniques, such as MPEG-2, ITU-H.263, MPEG-4, or ITU-H.264/MPEG-4 Part 10, Advanced Video Coding (AVC), to transmit and receive or store and retrieve digital video data more efficiently. Video encoding techniques may perform video compression via spatial and temporal prediction to reduce or remove redundancy inherent in video sequences. <i>See, e.g., Karczewicz at 1:49-2:28.</i> For inter-coding, a video encoder performs motion estimation to track the movement of matching video blocks between two or more adjacent frames or other coded units, such as slices of frames. Inter-coded frames may include predictive frames (“P-frames”), which may include blocks predicted from a previous frame, and bidirectional predictive frames (“B-frames”), which may include blocks predicted from a previous frame and a Subsequent frame of a video sequence. The terms P-frames and B-frames are somewhat historic in the sense that early coding techniques limited prediction in specific directions. Newer coding formats and standards may not limit the prediction direction of P-frames or B-frames. Thus, the term “bi-directional” now refers to prediction based on two or more lists of reference data regardless of the temporal relationship of such reference data relative to the data being coded. Consistent with newer video standards such as ITU H.264, for example, bi-directional prediction may be based on two different lists which do not necessarily need to have data that resides temporally before and after the current video block. In other words, B-video blocks may be predicted from two lists of data, which may correspond to data from two previous frames, two Subsequent frames, or one previous frame and one Subsequent frame. In contrast, P-Video blocks are predicted based on

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	one list, i.e., one data structure, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in Smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-Video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units. <i>See, e.g., Karczewicz at 4:37-42.</i> FIG. 1 is a block diagram illustrating one exemplary video encoding and decoding system that may implement techniques of this disclosure. FIG. 2 is a block diagram illustrating an example of a video encoder that may perform offset techniques consistent with this disclosure. <i>See, e.g., Karczewicz at 9:4-19.</i> The techniques of this disclosure are specifically applicable to weighted bi-directional prediction. As mentioned above, bi-directional prediction is prediction of so-called “B-video blocks” based on two different lists of data. B-video blocks may be predicted from two lists of data from two previous frames, two lists of data from Subsequent frames, or one list of data from a previous frame and one

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	from a Subsequent frame. In contrast, P-Video blocks are predicted based on one list, which may correspond to one predictive frame, e.g., one previous frame or one Subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in Smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-Video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. <i>See, e.g., Karczewicz at 12:32-43.</i> B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list 0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. <i>See, e.g., Karczewicz at FIG. 2.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	 The diagram illustrates a video encoder (50) and its reconstruction path. The process begins with 'INPUT VIDEO BLOCKS TO BE CODED' entering a summing junction (adder 48). The other input to adder 48 is a feedback signal from adder 51. The output of adder 48 is 'RESID. BLOCKS', which pass through a 'TRANSFORM UNIT 38' and a 'QUANTIZATION UNIT 40' to produce 'QUANTIZED RESIDUAL COEFFICIENTS'. These coefficients are then processed by an 'ENTROPY CODING UNIT 46' to generate a 'BITSTREAM'. For reconstruction, the 'BITSTREAM' is decoded by an 'INVERSE QUANTIZATION UNIT 42' and an 'INVERSE TRANSFORM UNIT 44' to produce 'RECON. RESID. BLOCKS'. These are added at adder 51 to the 'RECONSTRUCTED VIDEO BLOCKS' from memory 34. The output of adder 51 is fed back to adder 48. The encoder also includes a 'MOTION ESTIMATION UNIT 32' and a 'MOTION COMPENSATION UNIT 35'. Motion estimation unit 32 takes the input blocks and provides 'MOTION VECTORS' to motion compensation unit 35. Motion compensation unit 35 also receives 'MODE AND ROUNDING SYNTAX ELEMENTS' from the entropy coding unit 46. The output of motion compensation unit 35 is added to the input at adder 48. A 'MEMORY 34' stores the 'RECONSTRUCTED VIDEO BLOCKS' for feedback. FIG. 2 See, e.g., Karczewicz at 10:64-11:52. As shown in FIG. 2, video encoder 50 receives a current video block within a video frame or slice to be encoded. In the example of FIG.2, video encoder 50 includes motion estimation unit 32, a motion compensation unit 35, a memory 34, an adder 48, a transform unit 38, a quantization unit 40, and an entropy coding unit 46. For video block reconstruction, video encoder 50 also includes an inverse quantization unit 42, an inverse transform unit 44, and an adder 51. Video encoder 50 may also include a deblocking filter (not shown) to filter block 5 boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of adder 51.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	During the encoding process, video encoder 50 receives a Video block to be coded, and motion estimation unit 32 and 10 motion compensation unit 35 perform inter-predictive coding. Motion estimation unit 32 and motion compensation unit 35 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation is typically considered the process of generating motion vectors, which estimate 15 motion for video blocks. A motion vector, for example, may indicate the displacement of a predictive block within a predictive frame (or other coded unit) relative to the current block being coded within the current frame (or other coded unit). Motion compensation is typically considered the process of fetching or generating the predictive block based on the motion vector determined by motion estimation. Again, motion estimation unit 32 and motion compensation unit 35 may be functionally integrated. For demonstrative purposes, the techniques described in this disclosure are described as 25 being performed by motion compensation unit 35. Motion estimation unit 32 selects the appropriate motion vector for the video block to be coded by comparing the video block to video blocks of one or more predictive coded units (e.g., a previous and/or future frame in terms of time or temporally). Motion estimation unit 32 may, as an example, select a motion vector for a B-frame in a number of ways. In one way, motion estimation unit 32 may select a previous or future frame from a first set of frames (referred to as list 0) and determine a motion vector using only this previous or future frame from list 0. Alternatively, motion estimation unit 32 may select a previous or future frame from a second set of frames (referred to as list 1) and determine a motion vector using only this previous or future frame from list 1. In yet another way, motion estimation unit 32 may select a first frame from list 0 and a second frame from list 1 and select one or more motion vectors from the first frame of list 0 and the second frame of list 1. This third form of prediction may be referred to as bi-predictive motion estimation. Techniques of this disclosure may be implemented so as to efficiently select a motion-compensated bi-prediction mode. The selected motion vector for any given list may point to a predictive video block that is most similar to the video block being coded, e.g., as defined by a metric such as sum of absolute difference (SAD) or sum of squared difference (SSD) of pixel values of the predictive block relative to pixel values of the block being coded.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Srinivasan and Karczewicz are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz discloses a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. A person of ordinary skill in the art would have been motivated to incorporate these teachings of Karczewicz into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
1[b] using said first reference block to obtain a first prediction, said first prediction having a second	Srinivasan, as evidenced by the example citations below, discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
precision, which is higher than said first precision;	<i>See, e.g.</i>, Srinivasan at 5:12-28. SUMMARY In summary, the detailed description is directed to various techniques and tools for motion estimation and compensation. In video coding and decoding applications, for example, multi-stage interpolation may be used to compute the value at a particular sub-sample position. The intermediate values computed during a first or intermediate stage have a dynamic range greater than the dynamic range of the final value. Clamping during the first or intermediate stages may be skipped. Bit shifting performed during the first or intermediate stages may also be deferred until a final stage. The various techniques and tools, which can be used independently or in combination, can improve the accuracy with which the final value at the sub-sample position is computed. This increased accuracy improves the overall quality of compressed video sequence. <i>See, e.g.</i>, Srinivasan at 5:46-50. In another aspect, a component such as a video encoder or decoder computes a value of a particular fractional sample position using multiple stages of interpolation. The computing includes deferring bit shifting from a first stage of the multiple stages to a second stage of the multiple stages to increase accuracy of the value. <i>See, e.g.</i>, Srinivasan at 20:33-38. A component such as the encoder or decoder described with reference to FIGS. 4 and 5, respectively, may perform the multi-stage interpolation (1200, 1300, 1400, 1500). Alternatively, another encoder or decoder, or another type of component, may perform the multi-stage interpolation (1200,1300, 1400, 1500). <i>See, e.g.</i>, Srinivasan at 21:22-52.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. <i>See, e.g., Srinivasan at FIG. 14.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 14 Furthermore, it was well-known in the art to use a first reference block to obtain a first prediction, with said first prediction having a second precision, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses obtaining a first prediction, with said first prediction having a second precision. See e.g., JVT-R201 at 161 (“8.4.2.2. Fractional sample interpolation process”).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious said first prediction having a second precision, which is higher than said first precision. <i>See, e.g.</i>, Karczewicz at 1:12-34. TECHNICAL FIELD The disclosure relates to video encoding and, more particularly, video encoding techniques that use bidirectional prediction. BACKGROUND Digital multimedia capabilities can be incorporated into a wide range of devices, including digital televisions, digital direct broadcast systems, wireless communication devices, wireless broadcast systems, personal digital assistants (PDAs), laptop or desktop computers, digital cameras, digital recording devices, video gaming devices, video game consoles, cellular or satellite radio telephones, digital media players, and the like. Digital multimedia devices may implement video coding techniques, such as MPEG-2, ITU-H.263, MPEG-4, or ITU-H.264/MPEG-4 Part 10, Advanced Video Coding (AVC), to transmit and receive or store and retrieve digital video data more efficiently. Video encoding techniques may perform video compression via spatial and temporal prediction to reduce or remove redundancy inherent in video sequences. <i>See, e.g.</i>, Karczewicz at 1:49-2:28. For inter-coding, a video encoder performs motion estimation to track the movement of matching video blocks between two or more adjacent frames or other coded units, such as slices of frames. Inter-coded frames may include predictive frames (“P-frames”), which may include blocks predicted

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	from a previous frame, and bidirectional predictive frames (“B-frames”), which may include blocks predicted from a previous frame and a subsequent frame of a video sequence. The terms P-frames and B-frames are somewhat historic in the sense that early coding techniques limited prediction in specific directions. Newer coding formats and standards may not limit the prediction direction of P-frames or B-frames. Thus, the term “bi-directional” now refers to prediction based on two or more lists of reference data regardless of the temporal relationship of such reference data relative to the data being coded. Consistent with newer video standards such as ITU H.264, for example, bi-directional prediction may be based on two different lists which do not necessarily need to have data that resides temporally before and after the current video block. In other words, B-video blocks may be predicted from two lists of data, which may correspond to data from two previous frames, two subsequent frames, or one previous frame and one subsequent frame. In contrast, P-video blocks are predicted based on one list, i.e., one data structure, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>See, e.g.</i>, Karczewicz at 9:4-19. The techniques of this disclosure are specifically applicable to weighted bi-directional prediction. As mentioned above, bi-directional prediction is prediction of so-called “B-video blocks” based on two different lists of data. B-video blocks may be predicted from two lists of data from two previous frames, two lists of data from subsequent frames, or one list of data from a previous frame and one from a subsequent frame. In contrast, P-video blocks are predicted based on one list, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. <i>See, e.g.</i>, Karczewicz at 12:32-43. B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list 0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. <i>See, e.g.</i>, Karczewicz at 13:25-54. Such frequent and biased rounding operations can reduce precision of prediction. Furthermore, in the bidirectional prediction of the explicit weighted prediction, there are actually 2 roundings, one for the reference pictures and the other for the offsets. Therefore, in this case, the rounding error can

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	accumulate. In accordance with one aspect of this disclosure, instead of doing 2 separate roundings, the video encoder can add the offsets to the weighted prediction before right shift as the following: $\text{pred}(i,j)=(\text{pred0}(i,j)*w0+\text{pred1}(i,j)*w1+((o1+o2)\ll r)+2r)\gg(r+1),$ where $\text{pred}(i,j)$ is the weighted prediction data associated with rounding, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, $w0$ and $w1$ are weight factors, $o1$ and $o2$ are offsets, and r and $2r$ are rounding terms that provide the rounding in conjunction with a right shift operation “$\gg$” by $(r+1)$ bits. This may provide better prediction accuracy. In this case, a new syntax element may also be defined to allow for combination of two different offsets ($o1$ and $o2$) into one offset. Furthermore, in this case, a rounding value may comprise the rounding adjustment described above (e.g., $2r$ prior to a right shift operation by $(r+1)$ bits) as well as another a rounding value (“r”) associated with the offset. The equation above may also be modified slightly to provide higher precision for the offsets. If higher precision for the offsets are desired, the offsets may be multiplied by factors (such as by 2) and then rounded to an integer numbers. The left shift may also be changed to account for this added precision to the offset. In Karczewicz, VCEG-AI33 is incorporated by reference in its entirety. VCEG-AI33 discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision. <i>See, e.g., Karczewicz at 7:66-8:12.</i> Work to advance the H.264/MPEG-4 AVC standard takes place in various forums of the ITU-T, such as the Key Technologies Area (KTA) forum. The KTA forum seeks, in part, coding technologies that exhibit higher coding efficiency than that exhibited by the H.264/AVC standard. The techniques described in this disclosure may provide for coding improvements relative to the H.264/AVC standard. Recently, the KTA forum received a document detailing techniques that may be considered relevant or related to the techniques described herein, as a submission numbered VCEG-AI33, titled “High Precision Interpolation and Prediction,” by Yan Ye, Peisong Chen, and Marta Srinivasan and

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	presented at the 35th Meeting in Berlin, Germany, occurring on Jul. 16-18, 2008, which is hereby incorporated by reference in its entirety. <i>See, e.g.,</i> VCEG-AI33 at page 1. Introduction In the interpolation process defined in H.264, pixels at 1/4-pixel positions are interpolated by using pixels at half pixel positions after they have been rounded and clipped. Moreover, during bilinear filtering, which is used to obtain the 1/4-pixel values, upward rounding is always used. Such frequent and biased rounding effectively reduces precision of the interpolation filtering. In [1] it was proposed to postpone rounding and clipping until the last step not to compromise the final precision. In this contribution we demonstrate that after this modification the filtering can still be performed using 16 bit arithmetic. In addition we address rounding accumulation in case of bidirectional prediction. High Precision Interpolation When deriving pixels at positions pixels at 1/4-pixel positions labeled as a, c, d, n, f, i, k, and q (Figure 1) we postpone the shift, rounding and clipping till the last step: $a = (G \ll 5 + b1 + 32) \gg 6$ $c = (H \ll 5 + b1 + 32) \gg 6$ $d = (G \ll 5 + h1 + 32) \gg 6$ $n = (M \ll 5 + h1 + 32) \gg 6$ $f = (j1 \gg 5 + b1 + 32) \gg 6$ $i = (j1 \gg 5 + h1 + 32) \gg 6$ $k = (j1 \gg 5 + m1 + 32) \gg 6$ $l = (j1 \gg 5 + s1 + 32) \gg 6$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	where the intermediated values denoted as b1, h1, m1 and s1 are derived by applying the 6-tap filter vertically and horizontally, e.g.: $b1 = E - 5 * F + 20 * G + 20 * H - 5 * I + J.$ $h1 = A - 5 * C + 20 * G + 20 * M - 5 * R + T.$ The pixels at 1/4-pixel positions labeled as e, g, p and r are calculated by averaging with upward rounding of the two nearest pixels at half pixel positions in the diagonal direction by using $e = (b1 + h1 + 32) >> 6$ $g = (b1 + m1 + 32) >> 6$ $p = (h1 + s1 + 32) >> 6$ $r = (m1 + s1 + 32) >> 6$ In addition, the following disclosures of Noda would satisfy or render obvious said first prediction having a second precision, which is higher than said first precision. <i>See, e.g., Noda at 1:5-31.</i> 2. Description of the Related Art H.264 which is a representative video encoding standard system is a lossy compression system which performs orthogonal transform such as discrete cosine transform (DCT) on a prediction error signal between an input signal and a prediction image signal generated by intraframe prediction or motion compensation, and subjects the transform coefficients to compression processing based on quantization and encoding after the prediction error signal is transformed on such a frequency axis to acquire an encoded image. When the prediction image signal is generated, an interpolation filter for intra-frame prediction or motion compensation performs rounding processing with the same precision as the pixel bit depth of the input image after multiplying a plurality of pixels by a filter coefficient and adding them.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>See, e.g.</i>, Noda at 1:66-2:4. It is an object of the present invention to provide an image encoding apparatus and an image encoding method and an image decoding apparatus and an image decoding method to improve an encoding efficiency by enhancing a precision of intra-prediction or motion compensation enough. <i>See, e.g.</i>, Noda at 13:5-31. As described above, when the input image signal is 8 bits (represented by square plots in FIG. 42) and the prediction image signal is formed of 8 bits identical to the input image signal (represented by rhombus plots in FIG. 42), the pixel value of the pixel interpolated at the half pixel position occurs a rounding error. In contrast, when a prediction image of, for example, 12 bits is created by performing bit depth increase as described in the present embodiment (represented by circular plots in FIG. 42), the rounding error of the pixel value of the pixel interpolated at the half pixel position decreases. When bit depth increase is done, the bit depth of residual is increased to 12 bits in comparison with one before bit depth increase. Therefore, the absolute value of residual increases in comparison with the case obtaining the residual at 8 bits without the bit depth increase. However, since the rounding error decreases, the error occurring in quantizing the image with the quantization step size expressed by the equation (16) and having substantially the same number of bits decreases, resulting in making it possible to improve encoding efficient. In the present embodiment, as shown in FIG. 43, it can be done with bit depth higher than the bit depth of the input image by M bits, and the prediction image signal is created with precision higher than the input image signal by M bits. Therefore, the precision of a filter for motion compensation, loop filter, intra picture prediction and the like is improved, resulting in making it possible to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g.</i>, Noda at 28:12-39.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 24 shows a configuration example of the inter-frame predictor B103. The inter-frame predictor B103 creates the inter-frame prediction image signal 102 from the motion vector, the local decoded picture signal and the adaptive filter coefficient information. The integer pixel weight/offset correction unit B301 executes processing for performing so-called weighted prediction of integer pixel value. For a concrete example, the prediction image signal 102 of integer pixel is created according to the following equation (7). $Y=(W \times X+(1 \ll (L-1)) \gg L)+O \quad (7)$ where W is a weighting factor, L is a shift coefficient, O is offset coefficient, and these are contained in the adaptive filter coefficient information. The pixel value X is processed based on the equation (7) to be converted to a Y value, whereby correction of luminance or correction of color becomes possible. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1+W_2 \times X_2+(1 \ll (L-1)) \gg L)+((O_1+O_2) \gg 1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. See, e.g., Noda at FIG. 42.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 42 Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g.</i>, Noda at 1:43-48 (“Because, in H.264, a rounding error occurs in a process of rounding processing when a prediction signal is generated, it cannot be processed with enough precision in an interpolation filter

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	for intra-prediction or motion compensation, resulting in increasing a prediction error signal and decreasing the encoding efficiency.”). A person of ordinary skill in the art would have been motivated to incorporate these teachings into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
1[c] using said second reference block to obtain a second prediction, said second prediction having the second precision;	Srinivasan, as evidenced by the example citations below, discloses using the second reference block to obtain a second prediction, the second prediction having the second precision. <i>See, e.g., Srinivasan at 5:12-28.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	SUMMARY In summary, the detailed description is directed to various techniques and tools for motion estimation and compensation. In video coding and decoding applications, for example, multi-stage interpolation may be used to compute the value at a particular sub-sample position. The intermediate values computed during a first or intermediate stage have a dynamic range greater than the dynamic range of the final value. Clamping during the first or intermediate stages may be skipped. Bit shifting performed during the first or intermediate stages may also be deferred until a final stage. The various techniques and tools, which can be used independently or in combination, can improve the accuracy with which the final value at the sub-sample position is computed. This increased accuracy improves the overall quality of compressed video sequence. <i>See, e.g., Srinivasan at 5:46-50.</i> In another aspect, a component such as a video encoder or decoder computes a value of a particular fractional sample position using multiple stages of interpolation. The computing includes deferring bit shifting from a first stage of the multiple stages to a second stage of the multiple stages to increase accuracy of the value. <i>See, e.g., Srinivasan at 21:22-52.</i> FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. <i>See, e.g., Srinivasan at FIG. 14</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 14 Figure 14 illustrates a multi-stage process. The process begins with the FIRST STAGE 1410, which takes an x-bit range input value(s) 1411 and applies a first filter F_1 (1412) to produce a y-bit range intermediate value(s) 1420, where $y > x$ (1414). This is followed by INTERMEDIATE STAGES 1422 (indicated by a dashed line and three dots). The process concludes with the LAST STAGE 1430, which takes a z-bit range intermediate value(s) 1429, where $z > x$ (1432) and applies a last filter F_L to produce an output value 1436 (1434). Furthermore, it was well-known in the art to use a second reference block to obtain a second prediction, with said second prediction having the second precision, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses obtaining a second prediction, with said second prediction having the second precision. See e.g., JVT-R201 at 161 (“8.4.2.2. Fractional sample interpolation process”). Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious using said second reference block to obtain a second prediction, said second prediction having the second precision. <i>See, e.g., Karczewicz at 1:49-2:28.</i> For inter-coding, a video encoder performs motion estimation to track the movement of matching video blocks between two or more adjacent frames or other coded units, such as slices of frames. Inter-coded frames may include predictive frames (“P-frames”), which may include blocks predicted from a previous frame, and bidirectional predictive frames (“B-frames”), which may include blocks predicted from a previous frame and a subsequent frame of a video sequence. The terms P-frames and B-frames are somewhat historic in the sense that early coding techniques limited prediction in specific directions. Newer coding formats and standards may not limit the prediction direction of P-frames or B-frames. Thus, the term “bi-directional” now refers to prediction based on two or more lists of reference data regardless of the temporal relationship of such reference data relative to the data being coded. Consistent with newer video standards such as ITU H.264, for example, bi-directional prediction may be based on two different lists which do not necessarily need to have data that resides temporally before and after the current video block. In other words, B-video blocks may be predicted from two lists of data, which may correspond to data from two previous frames, two subsequent frames, or one previous frame and one subsequent frame. In contrast, P-video blocks are predicted based on one list, i.e., one data structure, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P- video blocks or I-video blocks. I-units may include only I-video blocks. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units. <i>See, e.g., Karczewicz at 9:4-19.</i> The techniques of this disclosure are specifically applicable to weighted bi-directional prediction. As mentioned above, bi-directional prediction is prediction of so-called “B-video blocks” based on two different lists of data. B-video blocks may be predicted from two lists of data from two previous frames, two lists of data from subsequent frames, or one list of data from a previous frame and one from a subsequent frame. In contrast, P-video blocks are predicted based on one list, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. <i>See, e.g., Karczewicz at 12:32-43.</i> B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list

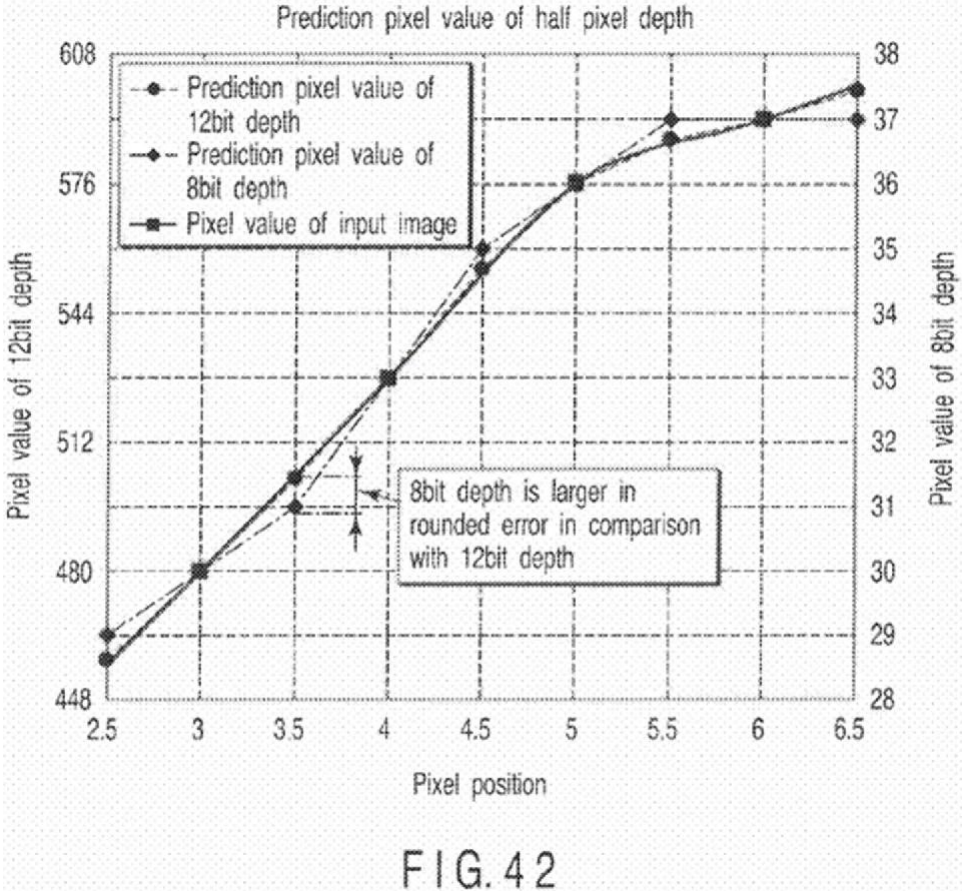
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. <i>See, e.g., Karczewicz at 13:25-54.</i> Such frequent and biased rounding operations can reduce precision of prediction. Furthermore, in the bidirectional prediction of the explicit weighted prediction, there are actually 2 roundings, one for the reference pictures and the other for the offsets. Therefore, in this case, the rounding error can accumulate. In accordance with one aspect of this disclosure, instead of doing 2 separate roundings, the video encoder can add the offsets to the weighted prediction before right shift as the following: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + ((o1 + o2) << r) + 2r) >> (r+1),$ where $\text{pred}(i,j)$ is the weighted prediction data associated with rounding, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, $w0$ and $w1$ are weight factors, $o1$ and $o2$ are offsets, and r and $2r$ are rounding terms that provide the rounding in conjunction with a right shift operation “>>” by $(r+1)$ bits. This may provide better prediction accuracy. In this case, a new syntax element may also be defined to allow for combination of two different offsets ($o1$ and $o2$) into one offset. Furthermore, in this case, a rounding value may comprise the rounding adjustment described above (e.g., $2r$ prior to a right shift operation by $(r+1)$ bits) as well as another a rounding value (“r”) associated with the offset. The equation above may also be modified slightly to provide higher precision for the offsets. If higher precision for the offsets are desired, the offsets may be multiplied by factors (such as by 2) and then rounded to an integer numbers. The left shift may also be changed to account for this added precision to the offset. In Karczewicz, VCEG-AI33 is incorporated by reference in its entirety. VCEG-AI33 discloses using the second reference block to obtain a second prediction, said the second prediction having the second precision.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>See, e.g.</i>, Karczewicz at 7:66-8:12. Work to advance the H.264/MPEG-4 AVC standard takes place in various forums of the ITU-T, such as the Key Technologies Area (KTA) forum. The KTA forum seeks, in part, coding technologies that exhibit higher coding efficiency than that exhibited by the H.264/AVC standard. The techniques described in this disclosure may provide for coding improvements relative to the H.264/AVC standard. Recently, the KTA forum received a document detailing techniques that may be considered relevant or related to the techniques described herein, as a submission numbered VCEG-AI33, titled “High Precision Interpolation and Prediction,” by Yan Ye, Peisong Chen, and Marta Srinivasan and presented at the 35th Meeting in Berlin, Germany, occurring on Jul. 16-18, 2008, which is hereby incorporated by reference in its entirety. <i>See, e.g.</i>, VCEG-AI33 at page 1. Introduction In the interpolation process defined in H.264, pixels at 1/4-pixel positions are interpolated by using pixels at half pixel positions after they have been rounded and clipped. Moreover, during bilinear filtering, which is used to obtain the 1/4-pixel values, upward rounding is always used. Such frequent and biased rounding effectively reduces precision of the interpolation filtering. In [1] it was proposed to postpone rounding and clipping until the last step not to compromise the final precision. In this contribution we demonstrate that after this modification the filtering can still be performed using 16 bit arithmetic. In addition we address rounding accumulation in case of bidirectional prediction. High Precision Interpolation When deriving pixels at positions pixels at 1/4-pixel positions labeled as a, c, d, n, f, i, k, and q (Figure 1) we postpone the shift, rounding and clipping till the last step: $a = (G \ll 5 + b1 + 32) \gg 6$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$c = (H \ll 5 + b1 + 32) \gg 6$ $d = (G \ll 5 + h1 + 32) \gg 6$ $n = (M \ll 5 + h1 + 32) \gg 6$ $f = (j1 \gg 5 + b1 + 32) \gg 6$ $i = (j1 \gg 5 + h1 + 32) \gg 6$ $k = (j1 \gg 5 + m1 + 32) \gg 6$ $l = (j1 \gg 5 + s1 + 32) \gg 6$ where the intermediated values denoted as b1, h1, m1 and s1 are derived by applying the 6-tap filter vertically and horizontally, e.g.: $b1 = E - 5 * F + 20 * G + 20 * H - 5 * I + J.$ $h1 = A - 5 * C + 20 * G + 20 * M - 5 * R + T.$ The pixels at 1/4-pixel positions labeled as e, g, p and r are calculated by averaging with upward rounding of the two nearest pixels at half pixel positions in the diagonal direction by using $e = (b1 + h1 + 32) \gg 6$ $g = (b1 + m1 + 32) \gg 6$ $p = (h1 + s1 + 32) \gg 6$ $r = (m1 + s1 + 32) \gg 6$ In addition, the following disclosures of Noda would satisfy or render obvious using said second reference block to obtain a second prediction, said second prediction having the second precision. <i>See, e.g., Noda at 1:5-31.</i> 2. Description of the Related Art H.264 which is a representative video encoding standard system is a lossy compression system which performs orthogonal transform such as discrete cosine transform (DCT) on a prediction error signal

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	between an input signal and a prediction image signal generated by intraframe prediction or motion compensation, and subjects the transform coefficients to compression processing based on quantization and encoding after the prediction error signal is transformed on such a frequency axis to acquire an encoded image. When the prediction image signal is generated, an interpolation filter for intra-frame prediction or motion compensation performs rounding processing with the same precision as the pixel bit depth of the input image after multiplying a plurality of pixels by a filter coefficient and adding them. See, e.g., Noda at 1:66-2:4. It is an object of the present invention to provide an image encoding apparatus and an image encoding method and an image decoding apparatus and an image decoding method to improve an encoding efficiency by enhancing a precision of intra-prediction or motion compensation enough. See, e.g., Noda at 13:5-31. As described above, when the input image signal is 8 bits (represented by square plots in FIG. 42) and the prediction image signal is formed of 8 bits identical to the input image signal (represented by rhombus plots in FIG. 42), the pixel value of the pixel interpolated at the half pixel position occurs a rounding error. In contrast, when a prediction image of, for example, 12 bits is created by performing bit depth increase as described in the present embodiment (represented by circular plots in FIG. 42), the rounding error of the pixel value of the pixel interpolated at the half pixel position decreases. When bit depth increase is done, the bit depth of residual is increased to 12 bits in comparison with one before bit depth increase. Therefore, the absolute value of residual increases in comparison with the case obtaining the residual at 8 bits without the bit depth increase. However, since the rounding error decreases, the error occurring in quantizing the image with the quantization step size expressed by the equation (16) and having substantially the same number of bits decreases, resulting in making it possible to improve encoding efficient.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	In the present embodiment, as shown in FIG. 43, it can be done with bit depth higher than the bit depth of the input image by M bits, and the prediction image signal is created with precision higher than the input image signal by M bits. Therefore, the precision of a filter for motion compensation, loop filter, intra picture prediction and the like is improved, resulting in making it possible to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g., Noda at 28:12-39.</i> FIG. 24 shows a configuration example of the inter-frame predictor B103. The inter-frame predictor B103 creates the inter-frame prediction image signal 102 from the motion vector, the local decoded picture signal and the adaptive filter coefficient information. The integer pixel weight/offset correction unit B301 executes processing for performing so-called weighted prediction of integer pixel value. For a concrete example, the prediction image signal 102 of integer pixel is created according to the following equation (7). $Y=(W \times X+(1 \ll (L-1)) \gg L)+O \quad (7)$ where W is a weighting factor, L is a shift coefficient, O is offset coefficient, and these are contained in the adaptive filter coefficient information. The pixel value X is processed based on the equation (7) to be converted to a Y value, whereby correction of luminance or correction of color becomes possible. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1+W_2 \times X_2+(1 \ll (L-1)) \gg L)+((O_1+O_2) \gg 1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. <i>See, e.g., Noda at FIG. 42.</i>  FIG. 42

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g.</i>, Noda at 1:43-48 (“Because, in H.264, a rounding error occurs in a process of rounding processing when a prediction signal is generated, it cannot be processed with enough precision in an interpolation filter for intra-prediction or motion compensation, resulting in increasing a prediction error signal and decreasing the encoding efficiency.”). A person of ordinary skill in the art would have been motivated to incorporate these teachings into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
1[d] obtaining a combined prediction based at least partly upon said first prediction and said second prediction;	Srinivasan, as evidenced by the example citations below, discloses obtaining a combined prediction based at least partly upon the first prediction and the second prediction. <i>See, e.g., Srinivasan at 2:13-35.</i> For a macroblock of a predicted frame, the WMV8 encoder performs motion estimation. The motion estimation approximates the motion of a macroblock in a predicted frame by searching for and matching the macroblock in the predicted frame with a macroblock from a reference frame. In FIG. 1, for instance, the WMV8 encoder computes a motion vector for a macroblock (115) in the predicted frame (110). To compute the motion vector, the encoder searches in a search area (135) of a reference frame (130). Within the search area (135), the encoder compares the luminance values of the macroblock (115) from the predicted frame (110) to the luminance values of various candidate blocks from the reference frame (130) in order to find a good match. The WMV8 encoder may switch motion vector accuracy, and may use a search range and motion vectors with integer, half, or quarter-pixel horizontal resolution and integer or half-pixel vertical resolution. With sub-pixel accurate motion vectors, the WMV8 encoder can approximate sub-pixel motion in a video sequence. During motion estimation or compensation, when a motion vector has subpixel accuracy (i.e., half-pixel or quarter-pixel), the WMV8 encoder or decoder must determine pixel values at sub-pixel positions in the reference frame. <i>See, e.g., Srinivasan at 6:1-2.</i> FIG. 4 is a block diagram of a generalized video encoder system used in several described embodiments.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	See, e.g., Srinivasan at 8:47-9:48. II. Generalized Video Encoder and Decoder FIG. 4 is a block diagram of a generalized video encoder (400) and FIG. 5 is a block diagram of a generalized video decoder (500). The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, FIGS. 4 and 5 usually do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bitstream, typically after entropy encoding of the side information. The format of the output bitstream can be a Windows Media Video format or another format. The encoder (400) and decoder (500) are block-based and use a 4:2:0 macroblock format with each macroblock including 4 luminance 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks (e.g., one U block, one V block). Alternatively, the encoder (400) and decoder (500) are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration than 8×8 blocks and 16×16 macroblocks. Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques. A. Video Encoder FIG. 4 is a block diagram of a general video encoder system (400). The encoder system (400) receives a sequence of video frames including a current frame (405), and produces compressed video

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	information (495) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (400). The encoder system (400) compresses predicted frames and key frames. For the sake of presentation, FIG. 4 shows a path for key frames through the encoder system (400) and a path for forward-predicted frames. Many of the components of the encoder system (400) are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A predicted frame [also called p-frame, b-frame for bidirectional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other frames. A prediction residual is the difference between what was predicted and the original frame. In contrast, a key frame [also called i-frame, intra-coded frame] is compressed without reference to other frames. If the current frame (405) is a forward-predicted frame, a motion estimator (410) estimates motion of macroblocks or other sets of pixels of the current frame (405) with respect to a reference frame, which is the reconstructed previous frame (425) buffered in the frame store (420). In alternative embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted. The motion estimator (410) outputs as side information motion information (415) such as motion vectors. A motion compensator (430) applies the motion information (415) to the reconstructed previous frame (425) to form a motion-compensated current frame (435). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (435) and the original current frame (405) is the prediction residual (445). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation. It was well-known in the art to obtain a combined prediction based at least partly upon said first prediction and said second prediction, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses obtaining a combined prediction based at least partly upon said first

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	prediction and said second prediction. See e.g., JVT-R201 at 167 (“8.4.2.3. Weighted sample prediction process”). Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious obtaining a combined prediction based at least partly upon said first prediction and said second prediction. <i>See, e.g.,</i> Karczewicz at 11:53-13:54. According to the ITU-T H.264/AVC standard, three motion-compensated bi-predictive algorithms or modes may be used to predict a B-frame or portions thereof, such as video blocks, macroblocks or any other discreet and/or contiguous portion of a B-frame. A first motion-compensated bi-predictive algorithm or mode, which is commonly referred to as default weighted prediction, may involve applying default weights to each identified video block of the first frame of list 0 and the second frame of list 1. The default weights may be programmed according to the standard, and are often selected to be equal for default weighted prediction. The weighted blocks of the first and second frames are then added together and divided by the total number of frames used to predict the B-frame, e.g., two in this instance. Often, this division is accomplished by adding 1 to the addition of the weighted blocks of the first and second frames and then shifting the result to the right by one bit. The addition of 1 is a rounding adjustment. In accordance with one aspect of this disclosure, the addition of 1 (the rounding adjustment) prior to the right shift by one may be avoided, thus eliminating upward biased rounding. Motion compensation unit 35 may generate both a weighted block with rounding and a weighted block without rounding, and may select the block that achieves the best coding efficiency. More generally, weighted prediction may be given by:

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 2r) \gg (r+1)$ where $\text{pred}(i,j)$ is data associated with the weighed prediction block, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, $w0$ and $w1$ are the weight factors, $2r$ is the rounding term, and $\gg$ is a right shift operation by $(r+1)$ bits. Consistent with this disclosure, two different version of $\text{pred}(i,j)$ may be generated and considered by motion compensation unit 35. The first is consistent with the equation above, and the second is consistent with the equation above without rounding, i.e., with the term “$2r$” removed from the equation. Eliminating this rounding may achieve better weighted predictive data in some cases, which can improve coding efficiency. Motion compensation unit 35 may generate one or more syntax elements to define whether or not rounding was used for a given video block or set of video blocks. Both the bi-directional prediction mode and the one or more syntax elements indicating whether rounding was used may be output from motion compensation unit 35 to entropy coding unit 46 for inclusion in the encoded bitstream. B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list 0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. In this disclosure, the term “B pictures” will be used to refer generally to any types of B units, which may be B frames, B slices, or possibly other video units that include at least some B video blocks. As mentioned, B pictures may allow 3 types of weighted prediction. For simplicity, only forward prediction in unidirectional prediction is shown below, although backwards prediction could also be used.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Default weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = \text{pred0}(i,j)$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) + \text{pred1}(i,j) + 1) \gg 1$ where $\text{pred0}(i,j)$ and $\text{pred1}(i,j)$ are prediction data from list 0 and list 1. Implicit weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = \text{pred0}(i,j)$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 32) \gg 6$ In this case, each prediction is scaled by a weighting factor $w0$ or $w1$, where $w0$ and $w1$ are calculated based on the relative temporal position of the list 0 and list 1 reference pictures. Explicit weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + 2^{r-1}) \gg r + o1$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 2^r) \gg (r+1) + ((o1 + o2 + 1) \gg 1)$ In this case, the weighting factors are determined by the encoder and transmitted in the slice header, and $o1$ and $o2$ are picture offsets for list 0 and list 1 reference pictures respectively. Conventionally, rounding adjustments are always used in bidirectional prediction. According to the equations above, a rounding adjustment of 1 is used in the default weighted prediction prior to a right shift by one, and a rounding adjustment of 32 is used in the implicit weighted prediction prior to a

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	right shift by six. Generally, a rounding adjustment of 2^{r-1} is commonly used prior to a right shift by r, where r represents a positive integer. Such frequent and biased rounding operations can reduce precision of prediction. Furthermore, in the bidirectional prediction of the explicit weighted prediction, there are actually 2 roundings, one for the reference pictures and the other for the offsets. Therefore, in this case, the rounding error can accumulate. In accordance with one aspect of this disclosure, instead of doing 2 separate roundings, the video encoder can add the offsets to the weighted prediction before right shift as the following: $\text{pred}(i,j) = (\text{pred0}(i,j) * w_0 + \text{pred1}(i,j) * w_1 + ((o_1 + o_2) \ll r) \gg (r+1)),$ where $\text{pred}(i,j)$ is the weighted prediction data associated with rounding, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, w_0 and w_1 are weight factors, o_1 and o_2 are offsets, and r and 2^r are rounding terms that provide the rounding in conjunction with a right shift operation “$\gg$” by $(r+1)$ bits. This may provide better prediction accuracy. In this case, a new syntax element may also be defined to allow for combination of two different offsets (o_1 and o_2) into one offset. Furthermore, in this case, a rounding value may comprise the rounding adjustment described above (e.g., 2^r prior to a right shift operation by $(r+1)$ bits) as well as another a rounding value (“r”) associated with the offset. The equation above may also be modified slightly to provide higher precision for the offsets. If higher precision for the offsets are desired, the offsets may be multiplied by factors (such as by 2) and then rounded to an integer numbers. The left shift may also be changed to account for this added precision to the offset. In addition, the following disclosures of Noda would satisfy or render obvious obtaining a combined prediction based at least partly upon said first prediction and said second prediction. <i>See, e.g., Noda at 28:12-39.</i> FIG. 24 shows a configuration example of the inter-frame predictor B103. The inter-frame predictor B103 creates the inter-frame prediction image signal 102 from the motion vector, the local decoded

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	picture signal and the adaptive filter coefficient information. The integer pixel weight/offset correction unit B301 executes processing for performing so-called weighted prediction of integer pixel value. For a concrete example, the prediction image signal 102 of integer pixel is created according to the following equation (7). $Y=(W \times X+(1 \ll (L-1)) \gg L)+O \quad (7)$ where W is a weighting factor, L is a shift coefficient, O is offset coefficient, and these are contained in the adaptive filter coefficient information. The pixel value X is processed based on the equation (7) to be converted to a Y value, whereby correction of luminance or correction of color becomes possible. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1+W_2 \times X_2+(1 \ll (L-1)) \gg L)+((O_1+O_2) \gg 1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. A person of ordinary skill in the art would have been motivated to incorporate these teachings of Noda into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs when the prediction is created. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
1[e] decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and	Srinivasan, as evidenced by the example citations below, discloses decreasing a precision of the combined prediction by shifting bits of the combined prediction to the right. <i>See, e.g., Srinivasan at 2:13-35.</i> For a macroblock of a predicted frame, the WMV8 encoder performs motion estimation. The motion estimation approximates the motion of a macroblock in a predicted frame by searching for and matching the macroblock in the predicted frame with a macroblock from a reference frame. In FIG. 1, for instance, the WMV8 encoder computes a motion vector for a macroblock (115) in the predicted frame (110). To compute the motion vector, the encoder searches in a search area (135) of a reference frame (130). Within the search area (135), the encoder compares the luminance values of the macroblock (115) from the predicted frame (110) to the luminance values of various candidate blocks

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	from the reference frame (130) in order to find a good match. The WMV8 encoder may switch motion vector accuracy, and may use a search range and motion vectors with integer, half, or quarter-pixel horizontal resolution and integer or half-pixel vertical resolution. With sub-pixel accurate motion vectors, the WMV8 encoder can approximate sub-pixel motion in a video sequence. During motion estimation or compensation, when a motion vector has subpixel accuracy (i.e., half-pixel or quarter-pixel), the WMV8 encoder or decoder must determine pixel values at sub-pixel positions in the reference frame. <i>See, e.g., Srinivasan at 6:1-2.</i> FIG. 4 is a block diagram of a generalized video encoder system used in several described embodiments. <i>See, e.g., Srinivasan at 8:47-9:48.</i> II. Generalized Video Encoder and Decoder FIG. 4 is a block diagram of a generalized video encoder (400) and FIG. 5 is a block diagram of a generalized video decoder (500). The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, FIGS. 4 and 5 usually do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bitstream, typically after entropy encoding of the side information. The format of the output bitstream can be a Windows Media Video format or another format. The encoder (400) and decoder (500) are block-based and use a 4:2:0 macroblock format with each macroblock including 4 luminance 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks (e.g., one U block, one V block). Alternatively, the encoder (400)

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	and decoder (500) are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration than 8×8 blocks and 16×16 macroblocks. Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques. A. Video Encoder FIG. 4 is a block diagram of a general video encoder system (400). The encoder system (400) receives a sequence of video frames including a current frame (405), and produces compressed video information (495) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (400). The encoder system (400) compresses predicted frames and key frames. For the sake of presentation, FIG. 4 shows a path for key frames through the encoder system (400) and a path for forward-predicted frames. Many of the components of the encoder system (400) are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A predicted frame [also called p-frame, b-frame for bidirectional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other frames. A prediction residual is the difference between what was predicted and the original frame. In contrast, a key frame [also called i-frame, intra-coded frame] is compressed without reference to other frames. If the current frame (405) is a forward-predicted frame, a motion estimator (410) estimates motion of macroblocks or other sets of pixels of the current frame (405) with respect to a reference frame, which is the reconstructed previous frame (425) buffered in the frame store (420). In alternative

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted. The motion estimator (410) outputs as side information motion information (415) such as motion vectors. A motion compensator (430) applies the motion information (415) to the reconstructed previous frame (425) to form a motion-compensated current frame (435). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (435) and the original current frame (405) is the prediction residual (445). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation. <i>See, e.g., Srinivasan at 16:21-17:17.</i> C. Multi-Dimensional Interpolation In some embodiments, interpolation is performed for sub-pixel positions that are offset in two dimensions. In FIG. 6, for instance, P3, P4, P6, and P7 are positions for which interpolation occurs in both the horizontal and vertical dimensions. In one embodiment, which corresponds to the interpolation method (800) shown in FIG. 8, two-dimensional sub-pixel locations are interpolated first along the vertical direction and then along the horizontal direction. As more fully described below, interpolation is performed using one or more of the filters F1, F2, or F3 specified above in equations (16) to (18). In the embodiment illustrated in FIG. 8, rounding is applied after both vertical filtering and horizontal filtering. The bit shifting in the rounding rule ensures retention of precision permitted by 16-bit arithmetic in the intermediate results. In FIG. 8, vertical filtering is performed first, followed by horizontal filtering. Starting with vertical filtering improves performance in some architectures. In other embodiments, the order of the filtering is different. For instance, interpolation is performed in the horizontal direction before the vertical direction. Or, various other combinations of interpolation filters are used (e.g., multiple horizontal and/or multiple vertical filters, etc.).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	The input and output pixel values (811, 838) have a bit depth of 8 bits, and have a dynamic range with 256 values. The intermediate values (820) have a bit depth of 16 bits, for a dynamic range with 65,536 values. In alternative embodiments, the input, output, and intermediate values have different (e.g., higher) bit depth. In a first stage (810), a suitable vertical filter (FV) is applied (812) to 8-bit input pixel value(s) (811). The vertical filter applied will depend on whether the selected sub-pixel position is shifted by a quarter pixel, a half pixel, or a three-quarter pixel and may take the form of one of the bicubic filters described above. The rounding rule after vertical filtering is defined as: $(S + R \cdot V) \gg \text{shiftV} \quad (25),$ where S is the vertically filtered result and $RV = 2^{\text{shiftV}} - 1 + R$. R is a rounding control value that alternates between 0 and 1 on a frame-by-frame basis. Thus, the rounding rule includes rounding with stage-alternating rounding control (813) and a bit shift (814). Right shifting potentially causes loss in resolution, so at least some of the right shifting is deferred to later stages of interpolation. The right shift value for shiftV depends on the sub-pixel position being interpolated. Specifically, $\text{shiftV} = \{5, 3, 3, 1\}$ for P3, P4, P6, and P7. The amount of shifting is less than that required to compensate for expansion due to the first stage filter coefficient values (e.g., the shift is less than 6-bits for the approximate bicubic filter), but enough to guarantee that the intermediate results of subsequent filtering stay within the dynamic range for intermediate values (e.g., 65,536 possible values for 16-bit words). Compared to full shifting, this shortened shift retains precision of the intermediate pixel values (820) after the first stage (810) of interpolation. The intermediate pixel values (820) have a dynamic range of y bits, where y is greater than 8 bits. The amount of shifting performed in the first stage may depend on the available bit depth and the

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	coefficients of the interpolation filters. For instance, in the exemplary implementation described herein, intermediate values are limited to a word limit of 16 bits. It was well-known in the art to decrease a precision of said combined prediction by shifting bits of the combined prediction to the right, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right. See e.g., JVT-R201 at 167 (“8.4.2.3. Weighted sample prediction process”). Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right. <i>See, e.g., Karczewicz at 11:53-13:54.</i> According to the ITU-T H.264/AVC standard, three motion-compensated bi-predictive algorithms or modes may be used to predict a B-frame or portions thereof, such as video blocks, macroblocks or any other discreet and/or contiguous portion of a B-frame. A first motion-compensated bi-predictive algorithm or mode, which is commonly referred to as default weighted prediction, may involve applying default weights to each identified video block of the first frame of list 0 and the second frame of list 1. The default weights may be programmed according to the standard, and are often selected to be equal for default weighted prediction. The weighted blocks of the first and second frames are then added together and divided by the total number of frames used to predict the B-frame, e.g., two in this instance. Often, this division is accomplished by adding 1 to the addition of the weighted blocks of the first and second frames and then shifting the result to the right by one bit. The addition of 1 is a rounding adjustment.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	In accordance with one aspect of this disclosure, the addition of 1 (the rounding adjustment) prior to the right shift by one may be avoided, thus eliminating upward biased rounding. Motion compensation unit 35 may generate both a weighted block with rounding and a weighted block without rounding, and may select the block that achieves the best coding efficiency. More generally, weighted prediction may be given by: $\text{pred}(i,j) = (\text{pred0}(i,j) * w_0 + \text{pred1}(i,j) * w_1 + 2r) \gg (r+1)$ where $\text{pred}(i,j)$ is data associated with the weighed prediction block, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, w_0 and w_1 are the weight factors, $2r$ is the rounding term, and $\gg$ is a right shift operation by $(r+1)$ bits. Consistent with this disclosure, two different version of $\text{pred}(i,j)$ may be generated and considered by motion compensation unit 35. The first is consistent with the equation above, and the second is consistent with the equation above without rounding, i.e., with the term “$2r$” removed from the equation. Eliminating this rounding may achieve better weighted predictive data in some cases, which can improve coding efficiency. Motion compensation unit 35 may generate one or more syntax elements to define whether or not rounding was used for a given video block or set of video blocks. Both the bi-directional prediction mode and the one or more syntax elements indicating whether rounding was used may be output from motion compensation unit 35 to entropy coding unit 46 for inclusion in the encoded bitstream. B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list 0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	In this disclosure, the term “B pictures” will be used to refer generally to any types of B units, which may be B frames, B slices, or possibly other video units that include at least some B video blocks. As mentioned, B pictures may allow 3 types of weighted prediction. For simplicity, only forward prediction in unidirectional prediction is shown below, although backwards prediction could also be used. Default weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = \text{pred0}(i,j)$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) + \text{pred1}(i,j) + 1) \gg 1$ where $\text{pred0}(i,j)$ and $\text{pred1}(i,j)$ are prediction data from list 0 and list 1. Implicit weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = \text{pred0}(i,j)$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 32) \gg 6$ In this case, each prediction is scaled by a weighting factor $w0$ or $w1$, where $w0$ and $w1$ are calculated based on the relative temporal position of the list 0 and list 1 reference pictures. Explicit weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + 2^{r-1}) \gg r + o1$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 2^r) \gg (r+1) + ((o1 + o2 + 1) \gg 1)$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	In this case, the weighting factors are determined by the encoder and transmitted in the slice header, and o1 and o2 are picture offsets for list 0 and list 1 reference pictures respectively. Conventionally, rounding adjustments are always used in bidirectional prediction. According to the equations above, a rounding adjustment of 1 is used in the default weighted prediction prior to a right shift by one, and a rounding adjustment of 32 is used in the implicit weighted prediction prior to a right shift by six. Generally, a rounding adjustment of 2^{r-1} is commonly used prior to a right shift by r, where r represents a positive integer. Such frequent and biased rounding operations can reduce precision of prediction. Furthermore, in the bidirectional prediction of the explicit weighted prediction, there are actually 2 roundings, one for the reference pictures and the other for the offsets. Therefore, in this case, the rounding error can accumulate. In accordance with one aspect of this disclosure, instead of doing 2 separate roundings, the video encoder can add the offsets to the weighted prediction before right shift as the following: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + ((o1 + o2) \ll r) \gg (r+1)),$ where $\text{pred}(i,j)$ is the weighted prediction data associated with rounding, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, $w0$ and $w1$ are weight factors, $o1$ and $o2$ are offsets, and r and 2^r are rounding terms that provide the rounding in conjunction with a right shift operation “$\gg$” by $(r+1)$ bits. This may provide better prediction accuracy. In this case, a new syntax element may also be defined to allow for combination of two different offsets ($o1$ and $o2$) into one offset. Furthermore, in this case, a rounding value may comprise the rounding adjustment described above (e.g., 2^r prior to a right shift operation by $(r+1)$ bits) as well as another a rounding value (“r”) associated with the offset. The equation above may also be modified slightly to provide higher precision for the offsets. If higher precision for the offsets are desired, the offsets may be multiplied by factors (such as by 2) and then rounded to an integer numbers. The left shift may also be changed to account for this added precision to the offset.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Further, Noda includes the following description that would satisfy or render obvious decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right. <i>See, e.g.</i>, Noda at 25: 45-50. The bit depth increased input image signal 1009 bit-increased as described above is led to the image encoder 10. The number of bits M to be increased, the bit depth N of the input image signal, and the bit depth conversion information 1010 such as gamma value, the pixel maximum value, the minimum value, histogram and the like which are necessary for conversion are generated (S14), and led to the multiplexer 12 as bit depth increase information 1003 with the bit depth conversion controller (bit depth conversion controller) 1002. <i>See, e.g.</i>, Noda at 9:27-32. For example, when the bit depth of a certain pixel of the bit depth increased decoded image signal 203 is decreased by M bits, the pixel value k' after decrease of M bits is calculated according to the following equation (2). $K'=(K+offset)>>M$ $offset=(1<<(M-1)) \quad (2)$ <i>See, e.g.</i>, Noda at 28:29-48. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1 + W_2 \times X_2 + (1<<(L-1))>>L) + ((O_1 + O_2)>>1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. When weight/offset correction is not needed by the adaptive filter information 116, the switch B301 is switched to generate a decimal pixel adaptive interpolation picture. In the present invention, since the bit depth in encoding and decoding is set to a value larger than the pixel bit depth of the input image signal, calculation precision for the weight/offset correction processing can be ensured, resulting in making it possible to perform weight/offset correction of higher precision. Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. A person of ordinary skill in the art would have been motivated to incorporate these teachings of Noda into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs when the prediction is created. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
1[f] 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.	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See, e.g., Srinivasan at 9:1-35.</i> A. Video Encoder FIG. 4 is a block diagram of a general video encoder system (400). The encoder system (400) receives a sequence of video frames including a current frame (405), and produces compressed video information (495) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (400). The encoder system (400) compresses predicted frames and key frames. For the sake of presentation, FIG. 4 shows a path for key frames through the encoder system (400) and a path for forward-predicted frames. Many of the components of the encoder system (400) are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A predicted frame [also called p-frame, b-frame for bidirectional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other frames. A prediction

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	residual is the difference between what was predicted and the original frame. In contrast, a key frame [also called i-frame, intra-coded frame] is compressed without reference to other frames. If the current frame (405) is a forward-predicted frame, a motion estimator (410) estimates motion of macroblocks or other sets of pixels of the current frame (405) with respect to a reference frame, which is the reconstructed previous frame (425) buffered in the frame store (420). In alternative embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted. The motion estimator (410) outputs as side information motion information (415) such as motion vectors. A motion compensator (430) applies the motion information (415) to the reconstructed previous frame (425) to form a motion-compensated current frame (435). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (435) and the original current frame (405) is the prediction residual (445). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation. A frequency transformer (460) converts the spatial domain video information into frequency domain (i.e., spectral) data. For block-based video frames, the frequency transformer (460) applies a discrete cosine transform [“DCT”] or variant of DCT to blocks of the motion prediction residual data, producing blocks of DCT coefficients. Alternatively, the frequency transformer (460) applies another conventional frequency transform such as a Fourier transform or uses wavelet or subband analysis. In some embodiments, the frequency transformer (460) applies a frequency transform to blocks of spatial prediction residuals for key frames. The frequency transformer (460) can apply an 8×8, 8×4, 4×8, or other size frequency transforms. It was well-known in the art to encode residual data in a bitstream, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses encoding residual data in a bitstream. <i>See</i> e.g., JVT-R201 at 5(“3.14 bitstream”), 11 (“3.127 residual: The decoded difference between a <i>prediction</i> of a sample or data element and its decoded value”), 99 (“7.4.5.3 residual data semantics”), 170 (“8.5. transform coefficient decoding process and picture construction process prior to deblocking filter process”).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render encoding residual data in a bitstream. <i>See, e.g.,</i> Karczewicz at 4:37-42. FIG. 1 is a block diagram illustrating one exemplary video encoding and decoding system that may implement techniques of this disclosure. FIG. 2 is a block diagram illustrating an example of a video encoder that may perform offset techniques consistent with this disclosure. <i>See, e.g.,</i> Karczewicz at FIG. 2.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 2 See, e.g., Karczewicz at 2:15-28. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units. <i>See, e.g., Karczewicz at 10:64-11:52.</i> As shown in FIG. 2, video encoder 50 receives a current video block within a video frame or slice to be encoded. In the example of FIG.2, video encoder 50 includes motion estimation unit 32, a motion compensation unit 35, a memory 34, an adder 48, a transform unit 38, a quantization unit 40, and an entropy coding unit 46. For video block reconstruction, video encoder 50 also includes an inverse quantization unit 42, an inverse transform unit 44, and an adder 51. Video encoder 50 may also include a deblocking filter (not shown) to filter block 5 boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of adder 51. During the encoding process, video encoder 50 receives a Video block to be coded, and motion estimation unit 32 and 10 motion compensation unit 35 perform inter-predictive coding. Motion estimation unit 32 and motion compensation unit 35 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation is typically considered the process of generating motion vectors, which estimate 15 motion for video blocks. A motion vector, for example, may indicate the displacement of a predictive block within a predictive frame (or other coded unit) relative to the current block being coded within the current frame (or other coded unit). Motion compensation is typically considered the process of fetching or generating the predictive block based on the motion vector determined by motion estimation. Again, motion estimation unit 32 and motion compensation unit 35 may be functionally integrated. For demonstrative purposes, the techniques described in this disclosure are described as 25 being performed by motion compensation unit 35. Motion estimation unit 32 selects the appropriate motion vector for the video block to be coded by comparing the video block to video blocks of one or more predictive coded units (e.g., a previous and/or future frame in terms of time or temporally). Motion estimation unit 32 may, as an example, select a motion vector for a B-frame in a number of ways. In one way, motion estimation unit 32 may

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	select a previous or future frame from a first set of frames (referred to as list 0) and determine a motion vector using only this previous or future frame from list 0. Alternatively, motion estimation unit 32 may select a previous or future frame from a second set of frames (referred to as list 1) and determine a motion vector using only this previous or future frame from list 1. In yet another way, motion estimation unit 32 may select a first frame from list 0 and a second frame from list 1 and select one or more motion vectors from the first frame of list 0 and the second frame of list 1. This third form of prediction may be referred to as bi-predictive motion estimation. Techniques of this disclosure may be implemented so as to efficiently select a motion-compensated bi-prediction mode. The selected motion vector for any given list may point to a predictive video block that is most similar to the video block being coded, e.g., as defined by a metric such as sum of absolute difference (SAD) or sum of squared difference (SSD) of pixel values of the predictive block relative to pixel values of the block being coded. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the first motion vector points to a subpixel, the first prediction is obtained by interpolation using pixel values of the first reference block. <i>See, e.g., Srinivasan at 12:29-59.</i> Motion vector components are usually specified in terms of pixel displacements, often with sub-pixel accuracy. Sub-pixel displacements are realized by filtering the reference frame using appropriately defined motion compensation filters. For the case of rectilinear sub-pixel motion compensation, the X and Y components are represented as fixed-point numbers. The integer part of these numbers is called the full-pixel shift and the fractional part is called the sub-pixel shift. When the sub-pixel shift

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	is zero, the motion is an integer number of pixels. Most often, this is implemented as a block copy from the reference frame to generate the predictor (although, in theory, some form of filtering could potentially be applied). On the other hand, when the sub-pixel shift is non-zero, the predictor is generated by applying one or more filters corresponding to the sub-pixel shift to integer-pixel locations in the reference frame. Therefore, the motion compensation filter is determined by the sub-pixel shift. To implement sub-pixel shifts as filtering operations, motion compensation filters interpolate data points at fractional pixel locations based on reference values at integer-pixel locations. In general, the quality of interpolation increases with the support of the filter. Some embodiments use separable 2-tap and 4-tap (in each direction) filters, which correspond to bilinear and bicubic interpolators. In some embodiments, the motion compensation filters use integer arithmetic and division implemented as a bit shift. The rounding control parameter R takes on the value of 0 or 1, and determines the direction of rounding for these divisions. This rounding control parameter may be set to a constant, signaled externally, or may be implicitly derived from past coded information. <i>See, e.g., Srinivasan at 16:22-50.</i> C. Multi-Dimensional Interpolation In some embodiments, interpolation is performed for sub-pixel positions that are offset in two dimensions. In FIG. 6, for instance, P3, P4, P6, and P7 are positions for which interpolation occurs in both the horizontal and vertical dimensions. In one embodiment, which corresponds to the interpolation method (800) shown in FIG. 8, two-dimensional sub-pixel locations are interpolated first along the vertical direction and then along the horizontal direction. As more fully described below, interpolation is performed using one or more of the filters F1, F2, or F3 specified above in equations (16) to (18). In the embodiment illustrated in

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 8, rounding is applied after both vertical filtering and horizontal filtering. The bit shifting in the rounding rule ensures retention of precision permitted by 16-bit arithmetic in the intermediate results. In FIG. 8, vertical filtering is performed first, followed by horizontal filtering. Starting with vertical filtering improves performance in some architectures. In other embodiments, the order of the filtering is different. For instance, interpolation is performed in the horizontal direction before the vertical direction. Or, various other combinations of interpolation filters are used (e.g., multiple horizontal and/or multiple vertical filters, etc.). The input and output pixel values (811, 838) have a bit depth of 8 bits, and have a dynamic range with 256 values. The intermediate values (820) have a bit depth of 16 bits, for a dynamic range with 65,536 values. In alternative embodiments, the input, output, and intermediate values have different (e.g., higher) bit depth. In a first stage (810), a suitable vertical filter (FV) is applied (812) to 8-bit input pixel value(s) (811). The vertical filter applied will depend on whether the selected sub-pixel position is shifted by a quarter pixel, a half pixel, or a three-quarter pixel and may take the form of one of the bicubic filters described above. The rounding rule after vertical filtering is defined as: $(S+R \gg \text{shiftV}) \gg \text{shiftV} \quad (25),$ where S is the vertically filtered result and $R = 2^{\text{shiftV}-1} - 1 + R$. R is a rounding control value that alternates between 0 and 1 on a frame-by-frame basis. Thus, the rounding rule includes rounding with stage-alternating rounding control (813) and a bit shift (814). Right shifting potentially causes loss in resolution, so at least some of the right shifting is deferred to later stages of interpolation. The right shift value for shiftV depends on the sub-pixel position being interpolated. Specifically, $\text{shiftV} = \{5, 3, 3, 1\}$ for P3, P4, P6, and P7. The amount of shifting is

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	less than that required to compensate for expansion due to the first stage filter coefficient values (e.g., the shift is less than 6-bits for the approximate bicubic filter), but enough to guarantee that the intermediate results of subsequent filtering stay within the dynamic range for intermediate values (e.g., 65,536 possible values for 16-bit words). Compared to full shifting, this shortened shift retains precision of the intermediate pixel values (820) after the first stage (810) of interpolation. The intermediate pixel values (820) have a dynamic range of y bits, where y is greater than 8 bits. The amount of shifting performed in the first stage may depend on the available bit depth and the coefficients of the interpolation filters. For instance, in the exemplary implementation described herein, intermediate values are limited to a word limit of 16 bits. <i>See, e.g., Srinivasan at 21:22-52.</i> FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. See, e.g., Srinivasan at FIG. 14 Figure 14 <pre> graph LR subgraph FIRST_STAGE [FIRST STAGE 1410] 1411[x-bit range input value(s) 1411] --> 1412[Apply first filter F₁] 1412 --> 1414[Perform shortened shift] 1414 --> 1420[y-bit range intermediate value(s) 1420, where y > x] end 1420 -.-> 1422[INTERMEDIATE STAGES 1422] 1422 -.-> 1430[LAST STAGE 1430] subgraph LAST_STAGE [LAST STAGE 1430] 1429[z-bit range intermediate value(s) 1429, where z > x] --> 1432[Apply last filter F_L] 1432 --> 1434[Perform lengthened shift] 1434 --> 1436[output value 1436] end style 1400 fill:none,stroke:none 1400 --> 1420 </pre> It was well-known in the art 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, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses obtaining

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	a combined prediction based at least partly upon said first prediction and said second prediction. <i>See e.g.</i>, JVT-R201 at 161 (“8.4.2.2. Fractional sample interpolation process”). Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious 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. <i>See, e.g.</i>, Karczewicz at 1:49-2:28. For inter-coding, a video encoder performs motion estimation to track the movement of matching video blocks between two or more adjacent frames or other coded units, such as slices of frames. Inter-coded frames may include predictive frames (“P-frames”), which may include blocks predicted from a previous frame, and bidirectional predictive frames (“B-frames”), which may include blocks predicted from a previous frame and a subsequent frame of a video sequence. The terms P-frames and B-frames are somewhat historic in the sense that early coding techniques limited prediction in specific directions. Newer coding formats and standards may not limit the prediction direction of P-frames or B-frames. Thus, the term “bi-directional” now refers to prediction based on two or more lists of reference data regardless of the temporal relationship of such reference data relative to the data being coded. Consistent with newer video standards such as ITU H.264, for example, bi-directional prediction may be based on two different lists which do not necessarily need to have data that resides temporally before and after the current video block. In other words, B-video blocks may be predicted from two lists of data, which may correspond to data from two previous frames, two subsequent frames, or one previous frame and one subsequent frame. In contrast, P-video blocks are predicted based on one list, i.e., one data structure, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units. <i>See, e.g., Karczewicz at 9:4-19.</i> The techniques of this disclosure are specifically applicable to weighted bi-directional prediction. As mentioned above, bi-directional prediction is prediction of so-called “B-video blocks” based on two different lists of data. B-video blocks may be predicted from two lists of data from two previous frames, two lists of data from subsequent frames, or one list of data from a previous frame and one from a subsequent frame. In contrast, P-video blocks are predicted based on one list, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. <i>See, e.g., Karczewicz at 12:32-43.</i> B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list 0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. In Karczewicz, VCEG-AI33 is incorporated by reference in its entirety. VCEG-AI33 discloses 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. <i>See, e.g., Karczewicz at 7:66-8:12.</i> Work to advance the H.264/MPEG-4 AVC standard takes place in various forums of the ITU-T, such as the Key Technologies Area (KTA) forum. The KTA forum seeks, in part, coding technologies that exhibit higher coding efficiency than that exhibited by the H.264/AVC standard. The techniques described in this disclosure may provide for coding improvements relative to the H.264/AVC standard. Recently, the KTA forum received a document detailing techniques that may be considered relevant or related to the techniques described herein, as a submission numbered VCEG-AI33, titled “High Precision Interpolation and Prediction,” by Yan Ye, Peisong Chen, and Marta Srinivasan and presented at the 35th Meeting in Berlin, Germany, occurring on Jul. 16-18, 2008, which is hereby incorporated by reference in its entirety. <i>See, e.g., VCEG-AI33 at page 1.</i> Introduction In the interpolation process defined in H.264, pixels at 1/4-pixel positions are interpolated by using pixels at half pixel positions after they have been rounded and clipped. Moreover, during bilinear filtering, which is used to obtain the 1/4-pixel values, upward rounding is always used. Such frequent and biased rounding effectively reduces precision of the interpolation filtering. In [1] it was proposed

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	to postpone rounding and clipping until the last step not to compromise the final precision. In this contribution we demonstrate that after this modification the filtering can still be performed using 16 bit arithmetic. In addition we address rounding accumulation in case of bidirectional prediction. High Precision Interpolation When deriving pixels at positions pixels at 1/4-pixel positions labeled as a, c, d, n, f, i, k, and q (Figure 1) we postpone the shift, rounding and clipping till the last step: $a = (G \ll 5 + b1 + 32) \gg 6$ $c = (H \ll 5 + b1 + 32) \gg 6$ $d = (G \ll 5 + h1 + 32) \gg 6$ $n = (M \ll 5 + h1 + 32) \gg 6$ $f = (j1 \gg 5 + b1 + 32) \gg 6$ $i = (j1 \gg 5 + h1 + 32) \gg 6$ $k = (j1 \gg 5 + m1 + 32) \gg 6$ $l = (j1 \gg 5 + s1 + 32) \gg 6$ where the intermediated values denoted as b1, h1, m1 and s1 are derived by applying the 6-tap filter vertically and horizontally, e.g.: $b1 = E - 5 * F + 20 * G + 20 * H - 5 * I + J.$ $h1 = A - 5 * C + 20 * G + 20 * M - 5 * R + T.$ The pixels at 1/4-pixel positions labeled as e, g, p and r are calculated by averaging with upward rounding of the two nearest pixels at half pixel positions in the diagonal direction by using $e = (b1 + h1 + 32) \gg 6$ $g = (b1 + m1 + 32) \gg 6$ $p = (h1 + s1 + 32) \gg 6$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$r = (m1+s1+32)>>6$ In addition, the following disclosures of Noda would satisfy or render obvious 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. <i>See, e.g.</i>, Noda at 1:5-31. 2. Description of the Related Art H.264 which is a representative video encoding standard system is a lossy compression system which performs orthogonal transform such as discrete cosine transform (DCT) on a prediction error signal between an input signal and a prediction image signal generated by intraframe prediction or motion compensation, and subjects the transform coefficients to compression processing based on quantization and encoding after the prediction error signal is transformed on such a frequency axis to acquire an encoded image. When the prediction image signal is generated, an interpolation filter for intra-frame prediction or motion compensation performs rounding processing with the same precision as the pixel bit depth of the input image after multiplying a plurality of pixels by a filter coefficient and adding them. <i>See, e.g.</i>, Noda at 1:66-2:4. It is an object of the present invention to provide an image encoding apparatus and an image encoding method and an image decoding apparatus and an image decoding method to improve an encoding efficiency by enhancing a precision of intra-prediction or motion compensation enough. <i>See, e.g.</i>, Noda at 13:5-31. As described above, when the input image signal is 8 bits (represented by square plots in FIG. 42) and the prediction image signal is formed of 8 bits identical to the input image signal (represented by rhombus plots in FIG. 42), the pixel value of the pixel interpolated at the half pixel position occurs

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	a rounding error. In contrast, when a prediction image of, for example, 12 bits is created by performing bit depth increase as described in the present embodiment (represented by circular plots in FIG. 42), the rounding error of the pixel value of the pixel interpolated at the half pixel position decreases. When bit depth increase is done, the bit depth of residual is increased to 12 bits in comparison with one before bit depth increase. Therefore, the absolute value of residual increases in comparison with the case obtaining the residual at 8 bits without the bit depth increase. However, since the rounding error decreases, the error occurring in quantizing the image with the quantization step size expressed by the equation (16) and having substantially the same number of bits decreases, resulting in making it possible to improve encoding efficient. In the present embodiment, as shown in FIG. 43, it can be done with bit depth higher than the bit depth of the input image by M bits, and the prediction image signal is created with precision higher than the input image signal by M bits. Therefore, the precision of a filter for motion compensation, loop filter, intra picture prediction and the like is improved, resulting in making it possible to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g., Noda at 28:12-39.</i> FIG. 24 shows a configuration example of the inter-frame predictor B103. The inter-frame predictor B103 creates the inter-frame prediction image signal 102 from the motion vector, the local decoded picture signal and the adaptive filter coefficient information. The integer pixel weight/offset correction unit B301 executes processing for performing so-called weighted prediction of integer pixel value. For a concrete example, the prediction image signal 102 of integer pixel is created according to the following equation (7). $Y=(W \times X+(1 \ll (L-1)) \gg L)+O \quad (7)$ where W is a weighting factor, L is a shift coefficient, O is offset coefficient, and these are contained in the adaptive filter coefficient information. The pixel value X is processed based on the equation

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	(7) to be converted to a Y value, whereby correction of luminance or correction of color becomes possible. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1 + W_2 \times X_2 + (1 \ll (L-1)) \gg L) + ((O_1 + O_2) \gg 1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. <i>See, e.g., Noda at FIG. 42.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 42 Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. See, e.g., Noda at 1:43-48 (“Because, in H.264, a rounding error occurs in a process of rounding processing when a prediction signal is generated, it cannot be processed with enough precision in an interpolation filter

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	for intra-prediction or motion compensation, resulting in increasing a prediction error signal and decreasing the encoding efficiency.”). A person of ordinary skill in the art would have been motivated to incorporate these teachings into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
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	Srinivasan, as evidenced by the example citations below, discloses that the first prediction is obtained by interpolation using values of the first reference block by right shifting a sum of a P-tap filter using values of the first reference block. <i>See, e.g., Srinivasan at 2:49-4:15.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
filter using values of said first reference block.	During motion estimation or compensation, when a motion vector has subpixel accuracy (i.e., half-pixel or quarter-pixel), the WMV8 encoder or decoder must determine pixel values at sub-pixel positions in the reference frame. The WMV8 encoder or decoder generates values for sub-pixel positions using interpolation filters. FIG. 2 is a chart (200) that shows sub-pixel sample positions H0, H1, H2, which have values computed by interpolation of integer-pixel values a, b, c, . . . , p. When operating with half-pixel motion vector accuracy, the interpolation filters used for luminance pixel values at the three distinct half-pixel positions H0, H1, H2 are: $H_0 = (f + g + R_2) \gg 1 \quad (1),$ $H_1 = (f + j + R_2) \gg 1 \quad (2),$ and $H_2 = (f + g + j + k + R_1) \gg 2 \quad (3),$ where R1 and R2 are rounding control values that are controlled by a one-bit rounding-control flag that indicates the rounding mode for a particular frame. If the rounding-control flag is set to 0, then R1=2 and R2=1. If the rounding-control flag is set to 1, then R1=R2=0. The value of the rounding-control flag alternates between 1 and 0 for each P-frame. At each I frame, the value of the rounding-control flag is reset to 0. Thus, the rounding control operates on a frame-by-frame basis. Equations 1, 2, and 3 are examples of bilinear interpolation. Bilinear interpolation is fast and tends to smooth pixel values. The smoothing may have desirable effects (such as decreasing perceptibility of quantization noise), but it can also lead to loss of valid pixel information. For quarter-pixel motion vector resolution, the WMV8 encoder or decoder first employs bicubic filters to interpolate luminance pixel values at half-pixel positions. Bicubic interpolation is slower

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	than bilinear interpolation, but tends to preserve edge values and result in less loss of valid pixel information. The bicubic filters for the three distinct half-pixel positions H0, H1, H2 are: $H_0 = (-e + 9f + 9g - h + 8) \gg 4 \quad (4),$ $H_1 = (-b + 9f + 9j - n + 8) \gg 4 \quad (5),$ and $H_2 = (-t_0 + 9t_1 + 9t_2 - t_3 + 8) \gg 4 \quad (6),$ where t0, t1, t2, t3 are computed as follows: $t_0 = (-a + 9b + 9c - d + 8) \gg 4 \quad (7),$ $t_1 = (-e + 9f + 9g - h + 8) \gg 4 \quad (8),$ $t_2 = (-i + 9j + 9k - l + 8) \gg 4 \quad (9),$ and $t_3 = (-m + 9n + 9o - p + 8) \gg 4 \quad (10).$ Equations (4)-(10) can result in output outside of the range of input values. For example, for 8-bit input (range 0 . . . 255), the series of values 0 255 255 0 produces an output value of 287 in any of equations (4)-(10). So, the WMV8 encoder or decoder clamps (or, “clips”) the output value of any of equations (4)-(10) to be within the valid range. For example, for 8-bit output values, values less than 0 are changed to 0, and values greater than 255 are changed to 255. Clamping addresses the range problem, but slows down computation. In addition, clamping results in loss of precision.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	The WMV8 encoder or decoder then computes pixel values at certain quarter-pixel positions in a subsequent stage of interpolation. These quarter-pixel locations are situated horizontally in between either two half-pixel locations or an integer-pixel location and a half-pixel location. For these quarter-pixel locations, the WMV8 encoder or decoder uses bilinear interpolation (i.e., $(x+y+1) \gg 1$) using the two horizontally neighboring half-pixel/integer-pixel locations without rounding control. Once luminance motion vectors are computed, the WMV8 encoder or decoder derives co-located chrominance motion vectors. Because a chrominance plane in WMV8 is half as large as a luminance plane both horizontally and vertically, luminance motion vector values must be scaled into appropriate chrominance motion vector values. In WMV8, this conversion process includes halving the luminance motion vectors and rounding the resulting chrominance motion vectors to half-pixel accuracy. Thus, luminance motion vectors having half-pixel accuracy are not converted to chrominance motion vectors having quarter-pixel accuracy. Moreover, chrominance rounding in WMV8 operates in a single mode that cannot be modified or selected by the user. In WMV8, the pixel values at sub-pixel positions in a reference frame may exhibit underflow or overflow in some circumstances. For example, the luminance pixel value at a quarter-pixel position may be 271 (which is outside the range of 0 . . . 255) if the neighboring integer-pixel position value is 255 and the neighboring half-pixel position value is 287 ($0+9*255+9*255-0+8 \gg 4=287$) ($255+287+1 \gg 1=271$). To address this problem, after adding the residual blocks to the predictor for a macroblock, the WMV8 encoder and decoder clamp reconstructed values for the macroblock to be within the range of 0 . . . 255, if necessary. <i>See, e.g., Srinivasan at 12:29-59.</i> Motion vector components are usually specified in terms of pixel displacements, often with sub-pixel accuracy. Sub-pixel displacements are realized by filtering the reference frame using appropriately defined motion compensation filters. For the case of rectilinear sub-pixel motion compensation, the X and Y components are represented as fixed-point numbers. The integer part of these numbers is called the full-pixel shift and the fractional part is called the sub-pixel shift. When the sub-pixel shift

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	is zero, the motion is an integer number of pixels. Most often, this is implemented as a block copy from the reference frame to generate the predictor (although, in theory, some form of filtering could potentially be applied). On the other hand, when the sub-pixel shift is non-zero, the predictor is generated by applying one or more filters corresponding to the sub-pixel shift to integer-pixel locations in the reference frame. Therefore, the motion compensation filter is determined by the sub-pixel shift. To implement sub-pixel shifts as filtering operations, motion compensation filters interpolate data points at fractional pixel locations based on reference values at integer-pixel locations. In general, the quality of interpolation increases with the support of the filter. Some embodiments use separable 2-tap and 4-tap (in each direction) filters, which correspond to bilinear and bicubic interpolators. In some embodiments, the motion compensation filters use integer arithmetic and division implemented as a bit shift. The rounding control parameter R takes on the value of 0 or 1, and determines the direction of rounding for these divisions. This rounding control parameter may be set to a constant, signaled externally, or may be implicitly derived from past coded information. <i>See, e.g., Srinivasan at 21:22-52.</i> FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. <i>See, e.g., Srinivasan at FIG. 14</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 14 Furthermore, it was well-known in the art that 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, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses that 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. See e.g., JVT-R201 at 161 (“8.4.2.2. Fractional sample interpolation process”). Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have

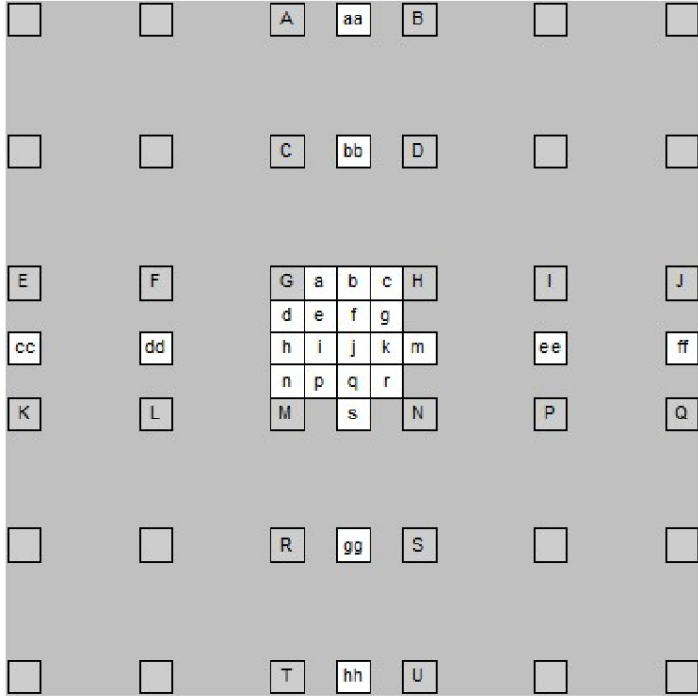
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious that said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block. <i>See, e.g., Karczewicz at 1:49-2:28.</i> For inter-coding, a video encoder performs motion estimation to track the movement of matching video blocks between two or more adjacent frames or other coded units, such as slices of frames. Inter-coded frames may include predictive frames (“P-frames”), which may include blocks predicted from a previous frame, and bidirectional predictive frames (“B-frames”), which may include blocks predicted from a previous frame and a subsequent frame of a video sequence. The terms P-frames and B-frames are somewhat historic in the sense that early coding techniques limited prediction in specific directions. Newer coding formats and standards may not limit the prediction direction of P-frames or B-frames. Thus, the term “bi-directional” now refers to prediction based on two or more lists of reference data regardless of the temporal relationship of such reference data relative to the data being coded. Consistent with newer video standards such as ITU H.264, for example, bi-directional prediction may be based on two different lists which do not necessarily need to have data that resides temporally before and after the current video block. In other words, B-video blocks may be predicted from two lists of data, which may correspond to data from two previous frames, two subsequent frames, or one previous frame and one subsequent frame. In contrast, P-video blocks are predicted based on one list, i.e., one data structure, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P- video blocks or I-video blocks. I-units may include only I-video blocks. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units. <i>See, e.g., Karczewicz at 9:4-19.</i> The techniques of this disclosure are specifically applicable to weighted bi-directional prediction. As mentioned above, bi-directional prediction is prediction of so-called “B-video blocks” based on two different lists of data. B-video blocks may be predicted from two lists of data from two previous frames, two lists of data from subsequent frames, or one list of data from a previous frame and one from a subsequent frame. In contrast, P-video blocks are predicted based on one list, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. <i>See, e.g., Karczewicz at 12:32-43.</i> B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction from the combination of both list 0 and list 1 reference pictures. To get the combination of both list

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. <i>See, e.g., Karczewicz at 13:25-54.</i> Such frequent and biased rounding operations can reduce precision of prediction. Furthermore, in the bidirectional prediction of the explicit weighted prediction, there are actually 2 roundings, one for the reference pictures and the other for the offsets. Therefore, in this case, the rounding error can accumulate. In accordance with one aspect of this disclosure, instead of doing 2 separate roundings, the video encoder can add the offsets to the weighted prediction before right shift as the following: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + ((o1 + o2) \ll r) + 2r) \gg (r+1),$ where $\text{pred}(i,j)$ is the weighted prediction data associated with rounding, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, $w0$ and $w1$ are weight factors, $o1$ and $o2$ are offsets, and r and $2r$ are rounding terms that provide the rounding in conjunction with a right shift operation “$\gg$” by $(r+1)$ bits. This may provide better prediction accuracy. In this case, a new syntax element may also be defined to allow for combination of two different offsets ($o1$ and $o2$) into one offset. Furthermore, in this case, a rounding value may comprise the rounding adjustment described above (e.g., $2r$ prior to a right shift operation by $(r+1)$ bits) as well as another a rounding value (“r”) associated with the offset. The equation above may also be modified slightly to provide higher precision for the offsets. If higher precision for the offsets are desired, the offsets may be multiplied by factors (such as by 2) and then rounded to an integer numbers. The left shift may also be changed to account for this added precision to the offset. In Karczewicz, VCEG-AI33 is incorporated by reference in its entirety. VCEG-AI33 discloses that 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.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>See, e.g.</i>, Karczewicz at 7:66-8:12. Work to advance the H.264/MPEG-4 AVC standard takes place in various forums of the ITU-T, such as the Key Technologies Area (KTA) forum. The KTA forum seeks, in part, coding technologies that exhibit higher coding efficiency than that exhibited by the H.264/AVC standard. The techniques described in this disclosure may provide for coding improvements relative to the H.264/AVC standard. Recently, the KTA forum received a document detailing techniques that may be considered relevant or related to the techniques described herein, as a submission numbered VCEG-AI33, titled “High Precision Interpolation and Prediction,” by Yan Ye, Peisong Chen, and Marta Srinivasan and presented at the 35th Meeting in Berlin, Germany, occurring on Jul. 16-18, 2008, which is hereby incorporated by reference in its entirety. <i>See, e.g.</i>, VCEG-AI33 at page 1. Introduction In the interpolation process defined in H.264, pixels at 1/4-pixel positions are interpolated by using pixels at half pixel positions after they have been rounded and clipped. Moreover, during bilinear filtering, which is used to obtain the 1/4-pixel values, upward rounding is always used. Such frequent and biased rounding effectively reduces precision of the interpolation filtering. In [1] it was proposed to postpone rounding and clipping until the last step not to compromise the final precision. In this contribution we demonstrate that after this modification the filtering can still be performed using 16 bit arithmetic. In addition we address rounding accumulation in case of bidirectional prediction. High Precision Interpolation When deriving pixels at positions pixels at 1/4-pixel positions labeled as a, c, d, n, f, i, k, and q (Figure 1) we postpone the shift, rounding and clipping till the last step: $a = (G \ll 5 + b1 + 32) \gg 6$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$c = (H \ll 5 + b1 + 32) \gg 6$ $d = (G \ll 5 + h1 + 32) \gg 6$ $n = (M \ll 5 + h1 + 32) \gg 6$ $f = (j1 \gg 5 + b1 + 32) \gg 6$ $i = (j1 \gg 5 + h1 + 32) \gg 6$ $k = (j1 \gg 5 + m1 + 32) \gg 6$ $l = (j1 \gg 5 + s1 + 32) \gg 6$ where the intermediated values denoted as b1, h1, m1 and s1 are derived by applying the 6-tap filter vertically and horizontally, e.g.: $b1 = E - 5 * F + 20 * G + 20 * H - 5 * I + J.$ $h1 = A - 5 * C + 20 * G + 20 * M - 5 * R + T.$ The pixels at 1/4-pixel positions labeled as e, g, p and r are calculated by averaging with upward rounding of the two nearest pixels at half pixel positions in the diagonal direction by using $e = (b1 + h1 + 32) \gg 6$ $g = (b1 + m1 + 32) \gg 6$ $p = (h1 + s1 + 32) \gg 6$ $r = (m1 + s1 + 32) \gg 6$ See, e.g., VCEG-AI33 at page 2.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	 Figure 1: Integer pixels (shaded blocks with upper-case letters) and fractional pixel positions (white blocks with lower-case letters). In addition, the following disclosures of Noda would satisfy or render obvious that 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. <i>See, e.g.,</i> Noda at 1:5-31. 2. Description of the Related Art

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	H.264 which is a representative video encoding standard system is a lossy compression system which performs orthogonal transform such as discrete cosine transform (DCT) on a prediction error signal between an input signal and a prediction image signal generated by intraframe prediction or motion compensation, and subjects the transform coefficients to compression processing based on quantization and encoding after the prediction error signal is transformed on such a frequency axis to acquire an encoded image. When the prediction image signal is generated, an interpolation filter for intra-frame prediction or motion compensation performs rounding processing with the same precision as the pixel bit depth of the input image after multiplying a plurality of pixels by a filter coefficient and adding them. <i>See, e.g., Noda at 1:66-2:4.</i> It is an object of the present invention to provide an image encoding apparatus and an image encoding method and an image decoding apparatus and an image decoding method to improve an encoding efficiency by enhancing a precision of intra-prediction or motion compensation enough. <i>See, e.g., Noda at 13:5-31.</i> As described above, when the input image signal is 8 bits (represented by square plots in FIG. 42) and the prediction image signal is formed of 8 bits identical to the input image signal (represented by rhombus plots in FIG. 42), the pixel value of the pixel interpolated at the half pixel position occurs a rounding error. In contrast, when a prediction image of, for example, 12 bits is created by performing bit depth increase as described in the present embodiment (represented by circular plots in FIG. 42), the rounding error of the pixel value of the pixel interpolated at the half pixel position decreases. When bit depth increase is done, the bit depth of residual is increased to 12 bits in comparison with one before bit depth increase. Therefore, the absolute value of residual increases in comparison with the case obtaining the residual at 8 bits without the bit depth increase. However, since the rounding error decreases, the error occurring in quantizing the image with the quantization

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	step size expressed by the equation (16) and having substantially the same number of bits decreases, resulting in making it possible to improve encoding efficient. In the present embodiment, as shown in FIG. 43, it can be done with bit depth higher than the bit depth of the input image by M bits, and the prediction image signal is created with precision higher than the input image signal by M bits. Therefore, the precision of a filter for motion compensation, loop filter, intra picture prediction and the like is improved, resulting in making it possible to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g., Noda at 28:12-39.</i> FIG. 24 shows a configuration example of the inter-frame predictor B103. The inter-frame predictor B103 creates the inter-frame prediction image signal 102 from the motion vector, the local decoded picture signal and the adaptive filter coefficient information. The integer pixel weight/offset correction unit B301 executes processing for performing so-called weighted prediction of integer pixel value. For a concrete example, the prediction image signal 102 of integer pixel is created according to the following equation (7). $Y=(W \times X+(1 \ll (L-1)) \gg L)+O \quad (7)$ where W is a weighting factor, L is a shift coefficient, O is offset coefficient, and these are contained in the adaptive filter coefficient information. The pixel value X is processed based on the equation (7) to be converted to a Y value, whereby correction of luminance or correction of color becomes possible. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1+W_2 \times X_2+(1 \ll (L-1)) \gg L)+((O_1+O_2) \gg 1) \quad (8)$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”																																								
	The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. See, e.g., Noda at FIG. 42. Prediction pixel value of half pixel depth <table><caption>Approximate data points from FIG. 42</caption><thead><tr><th>Pixel position</th><th>Input Image (12bit)</th><th>12bit Prediction</th><th>8bit Prediction</th></tr></thead><tbody><tr><td>2.5</td><td>450</td><td>450</td><td>29</td></tr><tr><td>3.0</td><td>480</td><td>480</td><td>30</td></tr><tr><td>3.5</td><td>510</td><td>510</td><td>31</td></tr><tr><td>4.0</td><td>540</td><td>540</td><td>32</td></tr><tr><td>4.5</td><td>570</td><td>570</td><td>33</td></tr><tr><td>5.0</td><td>590</td><td>590</td><td>34</td></tr><tr><td>5.5</td><td>600</td><td>600</td><td>35</td></tr><tr><td>6.0</td><td>605</td><td>605</td><td>36</td></tr><tr><td>6.5</td><td>608</td><td>608</td><td>37</td></tr></tbody></table> FIG. 42 Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video	Pixel position	Input Image (12bit)	12bit Prediction	8bit Prediction	2.5	450	450	29	3.0	480	480	30	3.5	510	510	31	4.0	540	540	32	4.5	570	570	33	5.0	590	590	34	5.5	600	600	35	6.0	605	605	36	6.5	608	608	37
Pixel position	Input Image (12bit)	12bit Prediction	8bit Prediction																																						
2.5	450	450	29																																						
3.0	480	480	30																																						
3.5	510	510	31																																						
4.0	540	540	32																																						
4.5	570	570	33																																						
5.0	590	590	34																																						
5.5	600	600	35																																						
6.0	605	605	36																																						
6.5	608	608	37																																						

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g.</i>, Noda at 1:43-48 (“Because, in H.264, a rounding error occurs in a process of rounding processing when a prediction signal is generated, it cannot be processed with enough precision in an interpolation filter for intra-prediction or motion compensation, resulting in increasing a prediction error signal and decreasing the encoding efficiency.”). A person of ordinary skill in the art would have been motivated to incorporate these teachings into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
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.	Srinivasan, as evidenced by the example citations below, discloses and/or renders obvious that in an instance in which the second motion vector points to an integer sample, the second prediction is obtained by shifting values of the second reference block to the left. <i>See, e.g., Srinivasan at 12:29-59.</i> Motion vector components are usually specified in terms of pixel displacements, often with sub-pixel accuracy. Sub-pixel displacements are realized by filtering the reference frame using appropriately defined motion compensation filters. For the case of rectilinear sub-pixel motion compensation, the X and Y components are represented as fixed-point numbers. The integer part of these numbers is called the full-pixel shift and the fractional part is called the sub-pixel shift. When the sub-pixel shift is zero, the motion is an integer number of pixels. Most often, this is implemented as a block copy from the reference frame to generate the predictor (although, in theory, some form of filtering could potentially be applied). On the other hand, when the sub-pixel shift is non-zero, the predictor is generated by applying one or more filters corresponding to the sub-pixel shift to integer-pixel locations in the reference frame. Therefore, the motion compensation filter is determined by the sub-pixel shift. To implement sub-pixel shifts as filtering operations, motion compensation filters interpolate data points at fractional pixel locations based on reference values at integer-pixel locations. In general, the quality of interpolation increases with the support of the filter. Some embodiments use separable 2-tap and 4-tap (in each direction) filters, which correspond to bilinear and bicubic interpolators. In some embodiments, the motion compensation filters use integer arithmetic and division implemented as a bit shift. The rounding control parameter R takes on the value of 0 or 1, and determines the direction of rounding for these divisions. This rounding control parameter may be set to a constant, signaled externally, or may be implicitly derived from past coded information.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>See, e.g.</i>, Srinivasan at 21:22-52. FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. <i>See, e.g.</i>, Srinivasan at FIG. 14.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 14 <pre> graph LR subgraph FIRST_STAGE_1410 [FIRST STAGE 1410] 1411[x-bit range input value(s) 1411] --> 1412[Apply first filter F₁] 1412 --> 1414[Perform shortened shift] 1414 --> 1420[y-bit range intermediate value(s) 1420, where y > x] end subgraph INTERMEDIATE_STAGES_1422 [INTERMEDIATE STAGES 1422] 1422[...] end subgraph LAST_STAGE_1430 [LAST STAGE 1430] 1429[z-bit range intermediate value(s) 1429, where z > x] --> 1432[Apply last filter F_L] 1432 --> 1434[Perform lengthened shift] 1434 --> 1436[output value 1436] end 1400[1400] -- points to 1410 </pre> Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious that 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. <i>See, e.g., Karczewicz at 7:66-8:12.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Work to advance the H.264/MPEG-4 AVC standard takes place in various forums of the ITU-T, such as the Key Technologies Area (KTA) forum. The KTA forum seeks, in part, coding technologies that exhibit higher coding efficiency than that exhibited by the H.264/AVC standard. The techniques described in this disclosure may provide for coding improvements relative to the H.264/AVC standard. Recently, the KTA forum received a document detailing techniques that may be considered relevant or related to the techniques described herein, as a submission numbered VCEG-AI33, titled “High Precision Interpolation and Prediction,” by Yan Ye, Peisong Chen, and Marta Srinivasan and presented at the 35th Meeting in Berlin, Germany, occurring on Jul. 16-18, 2008, which is hereby incorporated by reference in its entirety. <i>See, e.g., VCEG-AI33 at page 1.</i> Introduction In the interpolation process defined in H.264, pixels at $\frac{1}{4}$-pixel positions are interpolated by using pixels at half pixel positions after they have been rounded and clipped. Moreover, during bilinear filtering, which is used to obtain the $\frac{1}{4}$-pixel values, upward rounding is always used. Such frequent and biased rounding effectively reduces precision of the interpolation filtering. In [1] it was proposed to postpone rounding and clipping until the last step not to compromise the final precision. In this contribution we demonstrate that after this modification the filtering can still be performed using 16 bit arithmetic. In addition we address rounding accumulation in case of bidirectional prediction. High Precision Interpolation When deriving pixels at positions pixels at $\frac{1}{4}$-pixel positions labeled as a, c, d, n, f, l, k, and q (Figure 1) we postpone the shift, rounding and clipping till the last step: $a = (G \ll 5 + b1 + 32) \gg 6$ $c = (H \ll 5 + b1 + 32) \gg 6$ $d = (G \ll 5 + h1 + 32) \gg 6$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$n = (M \ll 5 + h1 + 32) \gg 6$ $f = (j1 \gg 5 + b1 + 32) \gg 6$ $I = (j1 \gg 5 + h1 + 32) \gg 6$ $k = (j1 \gg 5 + m1 + 32) \gg 6$ $l = (j1 \gg 5 + s1 + 32) \gg 6$ where the intermediated values denoted as b1, h1, m1 and s1 are derived by applying the 6-tap filter vertically and horizontally, e.g.: $b1 = E - 5 * F + 20 * G + 20 * H - 5 * I + J.$ $H1 = A - 5 * C + 20 * G + 20 * M - 5 * R + T.$ The pixels at $1/4$-pixel positions labeled as e, g, p and r are calculated by averaging with upward rounding of the two nearest pixels at half pixel positions in the diagonal direction by using $e = (b1 + h1 + 32) \gg 6$ $g = (b1 + m1 + 32) \gg 6$ $p = (h1 + s1 + 32) \gg 6$ $r = (m1 + s1 + 32) \gg 6$ <i>See, e.g., VCEG-AI33 at page 2.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 1: Integer pixels (shaded blocks with upper-case letters) and fractional pixel positions (white blocks with lower-case letters). Furthermore, Noda includes the following description that would satisfy or render obvious that 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. <i>See, e.g.,</i> Noda at 5:38-44. For example, when the bit depth of a certain pixel of the input image signal is K, the pixel value K' after M bits increase of the bit depth of the pixel value K is calculated by the following equation (1).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$K' = K \ll M$ (1) <i>See, e.g.,</i> Noda at 1:5-31. 2. Description of the Related Art H.264 which is a representative video encoding standard system is a lossy compression system which performs orthogonal transform such as discrete cosine transform (DCT) on a prediction error signal between an input signal and a prediction image signal generated by intraframe prediction or motion compensation, and subjects the transform coefficients to compression processing based on quantization and encoding after the prediction error signal is transformed on such a frequency axis to acquire an encoded image. When the prediction image signal is generated, an interpolation filter for intra-frame prediction or motion compensation performs rounding processing with the same precision as the pixel bit depth of the input image after multiplying a plurality of pixels by a filter coefficient and adding them. <i>See, e.g.,</i> Noda at 1:66-2:4. It is an object of the present invention to provide an image encoding apparatus and an image encoding method and an image decoding apparatus and an image decoding method to improve an encoding efficiency by enhancing a precision of intra-prediction or motion compensation enough. <i>See, e.g.,</i> Noda at 28:12-39. FIG. 24 shows a configuration example of the inter-frame predictor B103. The inter-frame predictor B103 creates the inter-frame prediction image signal 102 from the motion vector, the local decoded picture signal and the adaptive filter coefficient information. The integer pixel weight/offset correction unit B301 executes processing for performing so-called weighted prediction of integer pixel value. For a concrete example, the prediction image signal 102 of integer pixel is created according to the following equation (7).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	$Y=(W \times X+(1 \ll (L-1)) \gg L)+O \quad (7)$ where W is a weighting factor, L is a shift coefficient, O is offset coefficient, and these are contained in the adaptive filter coefficient information. The pixel value X is processed based on the equation (7) to be converted to a Y value, whereby correction of luminance or correction of color becomes possible. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1+W_2 \times X_2+(1 \ll (L-1)) \gg L)+((O_1+O_2) \gg 1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. <i>See, e.g., Noda at 28:49-59.</i> When the motion vector indicates a decimal pixel, the decimal pixel adaptive interpolation picture generator B303 generates an interpolated pixel value of the decimal pixel from an integer pixel value, using the interpolation filter coefficient in the adaptive filter information, and generates the prediction image signal 102. About this processing, in the present invention, since the bit depth in encoding and decoding is set to a value larger than the pixel bit depth of the input image signal, calculation precision of interpolation filtering processing can be ensured, resulting in making it possible to obtain a better decimal pixel value. <i>See, e.g., Noda at FIG. 42.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	FIG. 42 Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. <i>See, e.g.</i>, Noda at 1:43-48 (“Because, in H.264, a rounding error occurs in a process of rounding processing when a prediction signal is generated, it cannot be processed with enough precision in an interpolation filter for intra-prediction or motion compensation, resulting in increasing a prediction error signal and

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	decreasing the encoding efficiency.”). A person of ordinary skill in the art would have been motivated to incorporate these teachings into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
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	Srinivasan, as evidenced by the example citations below, discloses that the decreasing the precision of the combined prediction by shifting bits of the combined prediction to the right, further comprises inserting a rounding offset to the combined prediction before the decreasing. <i>See, e.g., Srinivasan at 2:49-4:15.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
rounding offset to the combined prediction before said decreasing.	During motion estimation or compensation, when a motion vector has subpixel accuracy (i.e., half-pixel or quarter-pixel), the WMV8 encoder or decoder must determine pixel values at sub-pixel positions in the reference frame. The WMV8 encoder or decoder generates values for sub-pixel positions using interpolation filters. FIG. 2 is a chart (200) that shows sub-pixel sample positions H0, H1, H2, which have values computed by interpolation of integer-pixel values a, b, c, . . . , p. When operating with half-pixel motion vector accuracy, the interpolation filters used for luminance pixel values at the three distinct half-pixel positions H0, H1, H2 are: $H_0 = (f + g + R_2) \gg 1 \quad (1),$ $H_1 = (f + j + R_2) \gg 1 \quad (2),$ and $H_2 = (f + g + j + k + R_1) \gg 2 \quad (3),$ where R1 and R2 are rounding control values that are controlled by a one-bit rounding-control flag that indicates the rounding mode for a particular frame. If the rounding-control flag is set to 0, then R1=2 and R2=1. If the rounding-control flag is set to 1, then R1=R2=0. The value of the rounding-control flag alternates between 1 and 0 for each P-frame. At each I frame, the value of the rounding-control flag is reset to 0. Thus, the rounding control operates on a frame-by-frame basis. Equations 1, 2, and 3 are examples of bilinear interpolation. Bilinear interpolation is fast and tends to smooth pixel values. The smoothing may have desirable effects (such as decreasing perceptibility of quantization noise), but it can also lead to loss of valid pixel information. For quarter-pixel motion vector resolution, the WMV8 encoder or decoder first employs bicubic filters to interpolate luminance pixel values at half-pixel positions. Bicubic interpolation is slower

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	than bilinear interpolation, but tends to preserve edge values and result in less loss of valid pixel information. The bicubic filters for the three distinct half-pixel positions H0, H1, H2 are: $H_0 = (-e + 9f + 9g - h + 8) \gg 4 \quad (4),$ $H_1 = (-b + 9f + 9j - n + 8) \gg 4 \quad (5),$ and $H_2 = (-t_0 + 9t_1 + 9t_2 - t_3 + 8) \gg 4 \quad (6),$ where t0, t1, t2, t3 are computed as follows: $t_0 = (-a + 9b + 9c - d + 8) \gg 4 \quad (7),$ $t_1 = (-e + 9f + 9g - h + 8) \gg 4 \quad (8),$ $t_2 = (-i + 9j + 9k - l + 8) \gg 4 \quad (9),$ and $t_3 = (-m + 9n + 9o - p + 8) \gg 4 \quad (10).$ Equations (4)-(10) can result in output outside of the range of input values. For example, for 8-bit input (range 0 . . . 255), the series of values 0 255 255 0 produces an output value of 287 in any of equations (4)-(10). So, the WMV8 encoder or decoder clamps (or, “clips”) the output value of any of equations (4)-(10) to be within the valid range. For example, for 8-bit output values, values less than 0 are changed to 0, and values greater than 255 are changed to 255. Clamping addresses the range problem, but slows down computation. In addition, clamping results in loss of precision.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	The WMV8 encoder or decoder then computes pixel values at certain quarter-pixel positions in a subsequent stage of interpolation. These quarter-pixel locations are situated horizontally in between either two half-pixel locations or an integer-pixel location and a half-pixel location. For these quarter-pixel locations, the WMV8 encoder or decoder uses bilinear interpolation (i.e., $(x+y+1) \gg 1$) using the two horizontally neighboring half-pixel/integer-pixel locations without rounding control. Once luminance motion vectors are computed, the WMV8 encoder or decoder derives co-located chrominance motion vectors. Because a chrominance plane in WMV8 is half as large as a luminance plane both horizontally and vertically, luminance motion vector values must be scaled into appropriate chrominance motion vector values. In WMV8, this conversion process includes halving the luminance motion vectors and rounding the resulting chrominance motion vectors to half-pixel accuracy. Thus, luminance motion vectors having half-pixel accuracy are not converted to chrominance motion vectors having quarter-pixel accuracy. Moreover, chrominance rounding in WMV8 operates in a single mode that cannot be modified or selected by the user. In WMV8, the pixel values at sub-pixel positions in a reference frame may exhibit underflow or overflow in some circumstances. For example, the luminance pixel value at a quarter-pixel position may be 271 (which is outside the range of 0 . . . 255) if the neighboring integer-pixel position value is 255 and the neighboring half-pixel position value is 287 ($0+9*255+9*255-0+8 \gg 4=287$) ($255+287+1 \gg 1=271$). To address this problem, after adding the residual blocks to the predictor for a macroblock, the WMV8 encoder and decoder clamp reconstructed values for the macroblock to be within the range of 0 . . . 255, if necessary. <i>See, e.g., Srinivasan at 6:1-2.</i> FIG. 4 is a block diagram of a generalized video encoder system used in several described embodiments. <i>See, e.g., Srinivasan at 8:47-9:48.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	II. Generalized Video Encoder and Decoder FIG. 4 is a block diagram of a generalized video encoder (400) and FIG. 5 is a block diagram of a generalized video decoder (500). The relationships shown between modules within the encoder and decoder indicate the main flow of information in the encoder and decoder; other relationships are not shown for the sake of simplicity. In particular, FIGS. 4 and 5 usually do not show side information indicating the encoder settings, modes, tables, etc. used for a video sequence, frame, macroblock, block, etc. Such side information is sent in the output bitstream, typically after entropy encoding of the side information. The format of the output bitstream can be a Windows Media Video format or another format. The encoder (400) and decoder (500) are block-based and use a 4:2:0 macroblock format with each macroblock including 4 luminance 8×8 luminance blocks (at times treated as one 16×16 macroblock) and two 8×8 chrominance blocks (e.g., one U block, one V block). Alternatively, the encoder (400) and decoder (500) are object-based, use a different macroblock or block format, or perform operations on sets of pixels of different size or configuration than 8×8 blocks and 16×16 macroblocks. Depending on implementation and the type of compression desired, modules of the encoder or decoder can be added, omitted, split into multiple modules, combined with other modules, and/or replaced with like modules. In alternative embodiments, encoder or decoders with different modules and/or other configurations of modules perform one or more of the described techniques. A. Video Encoder FIG. 4 is a block diagram of a general video encoder system (400). The encoder system (400) receives a sequence of video frames including a current frame (405), and produces compressed video

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	information (495) as output. Particular embodiments of video encoders typically use a variation or supplemented version of the generalized encoder (400). The encoder system (400) compresses predicted frames and key frames. For the sake of presentation, FIG. 4 shows a path for key frames through the encoder system (400) and a path for forward-predicted frames. Many of the components of the encoder system (400) are used for compressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A predicted frame [also called p-frame, b-frame for bidirectional prediction, or inter-coded frame] is represented in terms of prediction (or difference) from one or more other frames. A prediction residual is the difference between what was predicted and the original frame. In contrast, a key frame [also called i-frame, intra-coded frame] is compressed without reference to other frames. If the current frame (405) is a forward-predicted frame, a motion estimator (410) estimates motion of macroblocks or other sets of pixels of the current frame (405) with respect to a reference frame, which is the reconstructed previous frame (425) buffered in the frame store (420). In alternative embodiments, the reference frame is a later frame or the current frame is bi-directionally predicted. The motion estimator (410) outputs as side information motion information (415) such as motion vectors. A motion compensator (430) applies the motion information (415) to the reconstructed previous frame (425) to form a motion-compensated current frame (435). The prediction is rarely perfect, however, and the difference between the motion-compensated current frame (435) and the original current frame (405) is the prediction residual (445). Alternatively, a motion estimator and motion compensator apply another type of motion estimation/compensation. It was well-known in the art to insert a rounding offset to the combined prediction before said decreasing, as evidenced by various versions of the H.264 standards. For example, JVT-R201 discloses inserting a rounding offset to the combined prediction before said decreasing. <i>See e.g.</i>, JVT-R201 at 167 (“8.4.2.3. Weighted sample prediction process”).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious inserting a rounding offset to the combined prediction before said decreasing. <i>See, e.g.</i>, Karczewicz at 12:51-13:54. Default weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = \text{pred0}(i,j)$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) + \text{pred1}(i,j) + 1) \gg 1$ where $\text{pred0}(i,j)$ and $\text{pred1}(i,j)$ are prediction data from list 0 and list 1. Implicit weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = \text{pred0}(i,j)$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 32) \gg 6$ In this case, each prediction is scaled by a weighting factor $w0$ or $w1$, where $w0$ and $w1$ are calculated based on the relative temporal position of the list 0 and list 1 reference pictures. Explicit weighted prediction may be defined by the following equations for unidirectional prediction and bidirectional prediction, respectively. Unidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + 2^{r-1}) \gg r + \underline{0}1$ Bidirectional prediction: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + 2^r) \gg (r+1) + ((o1 + o2 + 1) \gg 1)$

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	In this case, the weighting factors are determined by the encoder and transmitted in the slice header, and o1 and o2 are picture offsets for list 0 and list 1 reference pictures respectively. Conventionally, rounding adjustments are always used in bidirectional prediction. According to the equations above, a rounding adjustment of 1 is used in the default weighted prediction prior to a right shift by one, and a rounding adjustment of 32 is used in the implicit weighted prediction prior to a right shift by six. Generally, a rounding adjustment of 2^{r-1} is commonly used prior to a right shift by r, where r represents a positive integer. Such frequent and biased rounding operations can reduce precision of prediction. Furthermore, in the bidirectional prediction of the explicit weighted prediction, there are actually 2 roundings, one for the reference pictures and the other for the offsets. Therefore, in this case, the rounding error can accumulate. In accordance with one aspect of this disclosure, instead of doing 2 separate roundings, the video encoder can add the offsets to the weighted prediction before right shift as the following: $\text{pred}(i,j) = (\text{pred0}(i,j) * w0 + \text{pred1}(i,j) * w1 + ((o1 + o2) \ll r) \gg (r+1)),$ where $\text{pred}(i,j)$ is the weighted prediction data associated with rounding, $\text{pred0}(i,j)$ is data from list 0, $\text{pred1}(i,j)$ is data from list 1, $w0$ and $w1$ are weight factors, $o1$ and $o2$ are offsets, and r and 2^r are rounding terms that provide the rounding in conjunction with a right shift operation “$\gg$” by $(r+1)$ bits. This may provide better prediction accuracy. In this case, a new syntax element may also be defined to allow for combination of two different offsets ($o1$ and $o2$) into one offset. Furthermore, in this case, a rounding value may comprise the rounding adjustment described above (e.g., 2^r prior to a right shift operation by $(r+1)$ bits) as well as another a rounding value (“r”) associated with the offset. The equation above may also be modified slightly to provide higher precision for the offsets. If higher precision for the offsets are desired, the offsets may be multiplied by factors (such as by 2) and then rounded to an integer numbers. The left shift may also be changed to account for this added precision to the offset.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Furthermore, Noda includes the following description that would satisfy or render obvious inserting a rounding offset to the combined prediction before said decreasing. <i>See, e.g.</i>, Noda at 28:29-48. The equation (8) is an example of weight/offset processing when bidirectional prediction is performed. $Y=(W_1 \times X_1 + W_2 \times X_2 + (1 \ll (L-1)) \gg L) + ((O_1 + O_2) \gg 1) \quad (8)$ The weighting factor W1 for the pixel value X1, the offset coefficient O1, the weighting factor W2 for the pixel value X2, the offset coefficient O2, and the shift coefficient L are included in the adaptive filter coefficient information. Luminance correction and color correction become possible by processing the values X1, X2 according to the equation B2 to convert it into a Y value. When weight/offset correction is not needed by the adaptive filter information 116, the switch B301 is switched to generate a decimal pixel adaptive interpolation picture. In the present invention, since the bit depth in encoding and decoding is set to a value larger than the pixel bit depth of the input image signal, calculation precision for the weight/offset correction processing can be ensured, resulting in making it possible to perform weight/offset correction of higher precision. Srinivasan, Karczewicz and Noda are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz and FIG. 20A of Noda disclose a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. A person of ordinary skill in the art would have been motivated to incorporate these teachings of Noda into Srinivasan to

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs when the prediction is created. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz and/or Noda in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
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.	Srinivasan, as evidenced by the example citations below, discloses that 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 the first prediction and values of the second prediction. <i>See, e.g., Srinivasan at 21:22-52.</i> FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters. <i>See, e.g., Srinivasan at FIG. 14</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 14 Although Srinivasan discloses and/or renders obvious this element on its own (including in combination with the knowledge of one of ordinary skill in the art), this element would also have been obvious in view of Karczewicz. The following disclosures of Karczewicz would satisfy or render obvious that 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. <i>See, e.g., Karczewicz at 1:49-2:28.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	For inter-coding, a video encoder performs motion estimation to track the movement of matching video blocks between two or more adjacent frames or other coded units, such as slices of frames. Inter-coded frames may include predictive frames (“P-frames”), which may include blocks predicted from a previous frame, and bidirectional predictive frames (“B-frames”), which may include blocks predicted from a previous frame and a subsequent frame of a video sequence. The terms P-frames and B-frames are somewhat historic in the sense that early coding techniques limited prediction in specific directions. Newer coding formats and standards may not limit the prediction direction of P-frames or B-frames. Thus, the term “bi-directional” now refers to prediction based on two or more lists of reference data regardless of the temporal relationship of such reference data relative to the data being coded. Consistent with newer video standards such as ITU H.264, for example, bi-directional prediction may be based on two different lists which do not necessarily need to have data that resides temporally before and after the current video block. In other words, B-video blocks may be predicted from two lists of data, which may correspond to data from two previous frames, two subsequent frames, or one previous frame and one subsequent frame. In contrast, P-video blocks are predicted based on one list, i.e., one data structure, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. For P- and B-video blocks, motion estimation generates motion vectors, which indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units. Motion compensation uses the motion vectors to generate prediction video blocks from the predictive reference frame(s) or other coded units. After motion compensation, a residual video block is formed by subtracting the prediction video block from the original video block to be coded. The video encoder usually applies transform, quantization and entropy coding processes to further reduce the bit rate associated with communication of the residual

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	block. I- and P-units are commonly used to define reference blocks for the inter-coding of P- and B-units. <i>See, e.g., Karczewicz at 4:37-42.</i> FIG. 1 is a block diagram illustrating one exemplary video encoding and decoding system that may implement techniques of this disclosure. FIG. 2 is a block diagram illustrating an example of a video encoder that may perform offset techniques consistent with this disclosure. <i>See, e.g., Karczewicz at 9:4-19.</i> The techniques of this disclosure are specifically applicable to weighted bi-directional prediction. As mentioned above, bi-directional prediction is prediction of so-called “B-video blocks” based on two different lists of data. B-video blocks may be predicted from two lists of data from two previous frames, two lists of data from subsequent frames, or one list of data from a previous frame and one from a subsequent frame. In contrast, P-video blocks are predicted based on one list, which may correspond to one predictive frame, e.g., one previous frame or one subsequent frame. B-frames and P-frames may be more generally referred to as P-units and B-units. P-units and B-units may also be realized in smaller coded units, such as slices of frames or portions of frames. B-units may include B-video blocks, P-video blocks or I-video blocks. P-units may include P-video blocks or I-video blocks. I-units may include only I-video blocks. <i>See, e.g., Karczewicz at 12:32-43.</i> B pictures use two lists of previously-coded reference pictures, list 0 and list 1. These two lists can each contain past and/or future coded pictures in temporal order. Blocks in a B picture may be predicted in one of several ways: motion-compensated prediction from a list 0 reference picture, motion-compensated prediction from a list 1 reference picture, or motion-compensated prediction

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	from the combination of both list 0 and list 1 reference pictures. To get the combination of both list 0 and list 1 reference pictures, two motion compensated reference areas are obtained from list 0 and list 1 reference picture respectively. Their combination will be used to predict the current block. See, e.g., Karczewicz at FIG. 2. FIG. 2 See, e.g., Karczewicz at 10:64-11:52. As shown in FIG. 2, video encoder 50 receives a current video block within a video frame or slice to be encoded. In the example of FIG.2, video encoder 50 includes motion estimation unit 32, a motion compensation unit 35, a memory 34, an adder 48, a transform unit 38, a quantization unit 40, and an entropy coding unit 46. For video block reconstruction, video encoder 50 also includes an inverse

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	quantization unit 42, an inverse transform unit 44, and an adder 51. Video encoder 50 may also include a deblocking filter (not shown) to filter block 5 boundaries to remove blockiness artifacts from reconstructed video. If desired, the deblocking filter would typically filter the output of adder 51. During the encoding process, video encoder 50 receives a Video block to be coded, and motion estimation unit 32 and 10 motion compensation unit 35 perform inter-predictive coding. Motion estimation unit 32 and motion compensation unit 35 may be highly integrated, but are illustrated separately for conceptual purposes. Motion estimation is typically considered the process of generating motion vectors, which estimate 15 motion for video blocks. A motion vector, for example, may indicate the displacement of a predictive block within a predictive frame (or other coded unit) relative to the current block being coded within the current frame (or other coded unit). Motion compensation is typically considered the process of fetching or generating the predictive block based on the motion vector determined by motion estimation. Again, motion estimation unit 32 and motion compensation unit 35 may be functionally integrated. For demonstrative purposes, the techniques described in this disclosure are described as 25 being performed by motion compensation unit 35. Motion estimation unit 32 selects the appropriate motion vector for the video block to be coded by comparing the video block to video blocks of one or more predictive coded units (e.g., a previous and/or future frame in terms of time or temporally). Motion estimation unit 32 may, as an example, select a motion vector for a B-frame in a number of ways. In one way, motion estimation unit 32 may select a previous or future frame from a first set of frames (referred to as list 0) and determine a motion vector using only this previous or future frame from list 0. Alternatively, motion estimation unit 32 may select a previous or future frame from a second set of frames (referred to as list 1) and determine a motion vector using only this previous or future frame from list 1. In yet another way, motion estimation unit 32 may select a first frame from list 0 and a second frame from list 1 and select one or more motion vectors from the first frame of list 0 and the second frame of list 1. This third form of prediction may be referred to as bi-predictive motion estimation. Techniques of this disclosure may be implemented so as to efficiently select a motion-compensated bi-prediction mode. The selected motion vector for any given list may point to a predictive video block that is most similar

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	to the video block being coded, e.g., as defined by a metric such as sum of absolute difference (SAD) or sum of squared difference (SSD) of pixel values of the predictive block relative to pixel values of the block being coded. Srinivasan and Karczewicz are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz discloses a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. A person of ordinary skill in the art would have been motivated to incorporate these teachings of Karczewicz into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz in order to make modifications, e.g., to the manner in which the video processing would operate. This element would have further been known in the art and otherwise disclosed in the other references charted in Respondents’ Invalidity Contentions, at least for the reasons set forth in these Contentions. For example, JVT-R201 (Joint Draft 5: Scalable Video Coding (in integrated form with ITU-T Rec. H.264 ISO/IEC 14996-10, published on January 2006) and Noda disclose that 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

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	second prediction. See e.g., JVT-R201 at 66 (“bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array and the value of the luma quantisation parameter range offset QpBdOffset_Y”), 67 (“bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantisation parameter range offset QpBdOffset_C”), Noda at 24: 51-56 (“There will now be described an operation of the image encoding apparatus with reference to flowchart of FIG.20B. A video signal wherein the bit depth of each pixel is, for example, N bit depth is input to this image encoding apparatus as an input image signal 100 in units of a frame, for example (S121).”), 25: 15-18 (“The local decoded picture signal 112 of the N bit depth stored in the frame memory 114 is input to the pixel bit depth increase unit 1001, to increase a value of each pixel by Mbits (S126)”). <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
7[preamble] An apparatus for encoding a block of pixels, the apparatus comprising:	Srinivasan, as evidenced by the example citations below, discloses an apparatus for encoding a block of pixels. <i>See limitation 1[preamble].</i>
7[a] 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	Srinivasan, as evidenced by the example citations below, discloses at least one processor and at least one memory including computer program code. <i>See limitation 1[a].</i> <i>See, e.g., Srinivasan at 7:24-8:39.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
one processor, cause the apparatus to:	I. Computing Environment FIG. 3 illustrates a generalized example of a suitable computing environment (300) in which several of the described embodiments may be implemented. The computing environment (300) is not intended to suggest any limitation as to scope of use or functionality, as the techniques and tools may be implemented in diverse general-purpose or special-purpose computing environments. With reference to FIG. 3, the computing environment (300) includes at least one processing unit (310) and memory (320). In FIG. 3, this most basic configuration (330) is included within a dashed line. The processing unit (310) executes computer-executable instructions and may be a real or a virtual processor. In a multiprocessing system, multiple processing units execute computer-executable instructions to increase processing power. The memory (320) may be volatile memory (e.g., registers, cache, RAM), non-volatile memory (e.g., ROM, EEPROM, flash memory, etc.), or some combination of the two. The memory (320) stores software (380) implementing sub-pixel interpolation techniques in an encoder and/or decoder, such as a video encoder and/or decoder. A computing environment may have additional features. For example, the computing environment (300) includes storage (340), one or more input devices (350), one or more output devices (360), and one or more communication connections (370). An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing environment (300). Typically, operating system software (not shown) provides an operating environment for other software executing in the computing environment (300), and coordinates activities of the components of the computing environment (300). The storage (340) may be removable or non-removable, and includes magnetic disks, magnetic tapes or cassettes, CD-ROMs, DVDs, or any other medium which can be used to store information and which can be accessed within the computing environment (300). The storage (340) stores instructions for the software (380) implementing sub-pixel interpolation techniques.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	The input device(s) (350) may be a touch input device such as a keyboard, mouse, pen, or trackball, a voice input device, a scanning device, or another device that provides input to the computing environment (300). For audio or video encoding, the input device(s) (350) may be a sound card, video card, TV tuner card, or similar device that accepts audio or video input in analog or digital form, or a CD-ROM or CD-RW that reads audio or video samples into the computing environment (300). The output device(s) (360) may be a display, printer, speaker, CD-writer, or another device that provides output from the computing environment (300). The communication connection(s) (370) enable communication over a communication medium to another computing entity. The communication medium conveys information such as computer-executable instructions, audio or video input or output, or other data in a modulated data signal. A modulated data signal is a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired or wireless techniques implemented with an electrical, optical; RF, infrared, acoustic, or other carrier. The techniques and tools can be described in the general context of computer-readable media. Computer-readable media are any available media that can be accessed within a computing environment. By way of example, and not limitation, with the computing environment (300), computer-readable media include memory (320), storage (340), communication media, and combinations of any of the above. The techniques and tools can be described in the general context of computer-executable instructions, such as those included in program modules, being executed in a computing environment on a target real or virtual processor. Generally, program modules include routines, programs, libraries, objects, classes, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The functionality of the program modules may be combined or split between program modules as desired in various embodiments. Computer-executable instructions for program modules may be executed within a local or distributed computing environment.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	This element would have further been known in the art and otherwise disclosed in the other references charted in Respondents’ Invalidity Contentions, at least for the reasons set forth in these Contentions. For example, Karczewicz disclose at least one processor and at least one memory including computer program code, the at least one memory. <i>See, e.g., Karczewicz at 3:63-4:4.</i> The techniques described in this disclosure may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the software may be executed in one or more processors, such as a microprocessor, application specific integrated circuit (ASIC), field programmable gate array (FPGA), or digital signal processor (DSP). The software that executes the techniques may be initially stored in a computer-readable medium and loaded and executed in the processor. <i>See, e.g., Karczewicz at 8:13-22.</i> Video encoder 22 and video decoder 28 each may be implemented as one or more microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), field programmable gate arrays (FPGAs), discrete logic, software, hardware, firmware or any combinations thereof. Each of video encoder 22 and video decoder 28 may be included in one or more encoders or decoders, either of which may be integrated as part of a combined encoder/decoder (CODEC) in a respective mobile device, subscriber device, broadcast device, server, or the like. <i>See, e.g., Karczewicz at 20:63-21:39.</i> The techniques of this disclosure may be implemented in a wide variety of devices or apparatuses, including a wireless handset, and integrated circuit (IC) or a set of ICs (i.e., a chip set). Any components, modules or units have been described provided to emphasize functional aspects and does not necessarily require realization by different hardware units. The techniques described herein may also be implemented in hardware, software, firmware, or any combination thereof. Any features

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	described as modules, units or components may be implemented together in an integrated logic device or separately as discrete but interoperable logic devices. In some cases, various features may be implemented as an integrated circuit device, such as an integrated circuit chip or chipset. If implemented in software, the techniques may be realized at least in part by a computer-readable medium comprising instructions that, when executed in a processor, performs one or more of the methods described above. The computer-readable medium may comprise a computer-readable storage medium and may form part of a computer program product, which may include packaging materials. The computer-readable storage medium may comprise random access memory (RAM) such as synchronous dynamic random access memory (SDRAM), read-only memory (ROM), non-volatile random access memory (NVRAM), electrically erasable programmable read-only memory (EEPROM), FLASH memory, magnetic or optical data storage media, and the like. The techniques additionally, or alternatively, may be realized at least in part by a computer-readable communication medium that carries or communicates code in the form of instructions or data structures and that can be accessed, read, and/or executed by a computer. The code or instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, an application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the term “processor,” as used herein may refer to any of the foregoing structure or any other structure suitable for implementation of the techniques described herein. In addition, in some aspects, the functionality described herein may be provided within dedicated software modules or hardware modules configured for encoding and decoding, or incorporated in a combined video codec. Also, the techniques could be fully implemented in one or more circuits or logic elements. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
7[b] 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;	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 1[a].</i>
7[c] use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;	Srinivasan, as evidenced by the example citations below, discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision. <i>See limitation 1[b].</i>
7[d] use said second reference block to obtain a second prediction, said second prediction having the second precision;	Srinivasan, as evidenced by the example citations below, discloses using the second reference block to obtain a second prediction, the second prediction having the second precision. <i>See limitation 1[c].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
7[e] obtain a combined prediction based at least partly upon said first prediction and said second prediction;	Srinivasan, as evidenced by the example citations below, discloses obtaining a combined prediction based at least partly upon the first prediction and the second prediction. <i>See limitation 1[d].</i>
7[f] decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and	Srinivasan, as evidenced by the example citations below, discloses decreasing a precision of the combined prediction by shifting bits of the combined prediction to the right. <i>See limitation 1[e].</i>
7[g] 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.	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 1[f].</i>
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	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the first motion vector points to a subpixel, the first prediction is obtained by interpolation using pixel values of the first reference block. <i>See claim 2.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
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.	Srinivasan, as evidenced by the example citations below, discloses that the first prediction is obtained by interpolation using values of the first reference block by right shifting a sum of a P-tap filter using values of the first reference block. <i>See claim 3.</i>
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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the second motion vector points to an integer sample, the second prediction is obtained by shifting values of the second reference block to the left. <i>See claim 4.</i>
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	Srinivasan, as evidenced by the example citations below, discloses that the at least one memory and computer code are configured to cause the apparatus to decrease the precision of the combined prediction by shifting bits of the combined prediction to the right, by inserting a rounding offset to the combined prediction before the decreasing. <i>See claim 5.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
combined prediction by shifting bits of the combined prediction to the right, by: inserting a rounding offset to the combined prediction before said decreasing.	<i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
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.	Srinivasan, as evidenced by the example citations below, discloses that 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 the first prediction and values of the second prediction. <i>See claim 6.</i>
13[preamble] 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	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitations 1[preamble], 7[preamble], and 7[a].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
program code portions comprising program code instructions configured to:	
13[a] 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;	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 1[a].</i>
13[b] use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;	Srinivasan, as evidenced by the example citations below, discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision. <i>See limitation 1[b].</i>
13[c] use said second reference block to obtain a second prediction, said	Srinivasan, as evidenced by the example citations below, discloses using the second reference block to obtain a second prediction, the second prediction having the second precision. <i>See limitation 1[c].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
second prediction having the second precision;	
13[d] obtain a combined prediction based at least partly upon said first prediction and said second prediction;	Srinivasan, as evidenced by the example citations below, discloses obtaining a combined prediction based at least partly upon the first prediction and the second prediction. <i>See limitation 1[d].</i>
13[e] decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and	Srinivasan, as evidenced by the example citations below, discloses decreasing a precision of the combined prediction by shifting bits of the combined prediction to the right. <i>See limitation 1[e].</i>
13[f] 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.	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 1[f].</i>
14. The computer program product according to claim 13, wherein in an instance in which said first motion vector	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the first motion vector points to a subpixel, the first prediction is obtained by interpolation using pixel values of the first reference block.

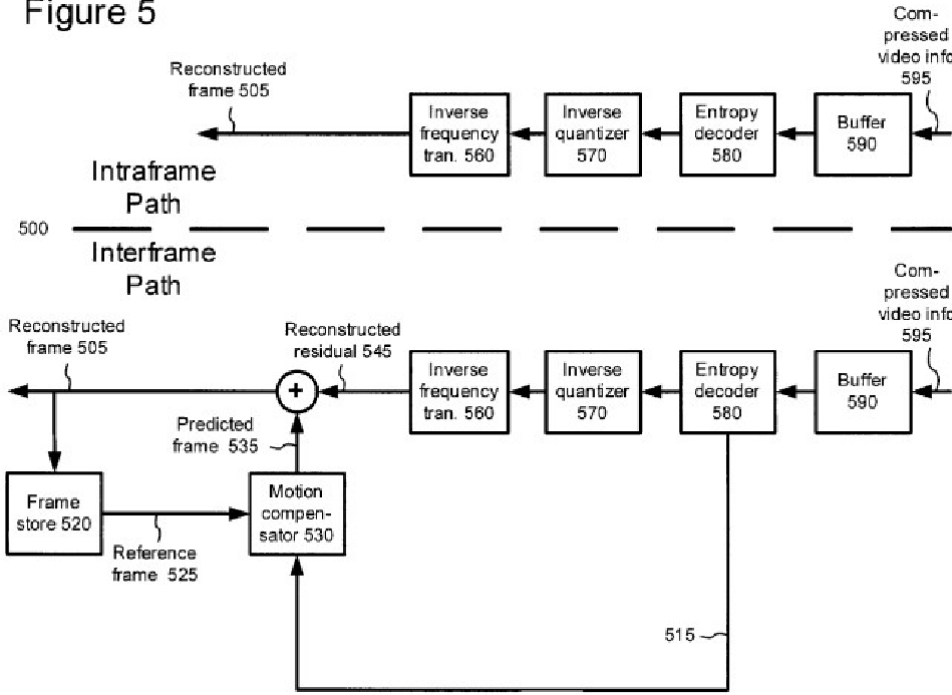
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.	<i>See claim 2.</i>
15. The computer program product according to claim 14, wherein said first prediction 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.	Srinivasan, as evidenced by the example citations below, discloses that the first prediction is obtained by interpolation using values of the first reference block by right shifting a sum of a P-tap filter using values of the first reference block. <i>See claim 3.</i>
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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the second motion vector points to an integer sample, the second prediction is obtained by shifting values of the second reference block to the left. <i>See claim 4.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
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.	Srinivasan, as evidenced by the example citations below, discloses that the program code instructions configured to decrease the precision of the 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 the decreasing. <i>See claims 5 and 11.</i>
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.	Srinivasan, as evidenced by the example citations below, discloses that 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 the first prediction and values of the second prediction. <i>See claim 6.</i>

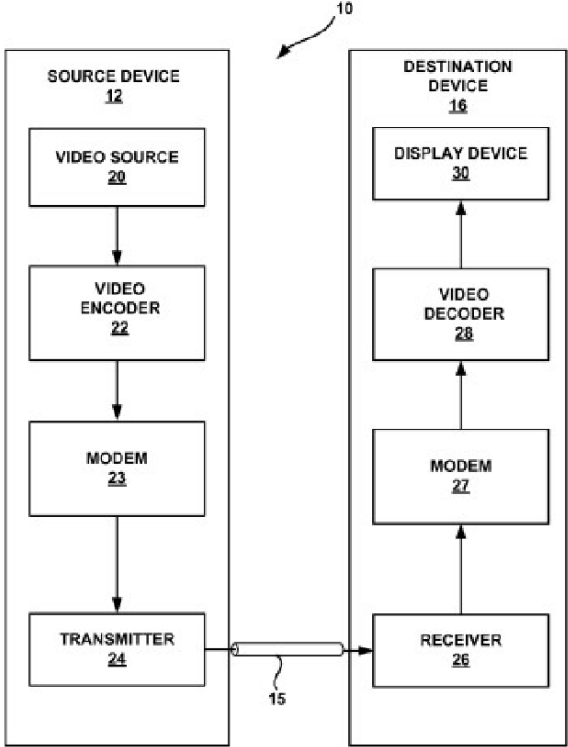
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
19[preamble] A method for decoding a block of pixels, the method comprising:	Srinivasan, as evidenced by the example citations below, discloses a method for decoding a block of pixels. <i>See, e.g.</i>, Srinivasan at 1:20-26. TECHNICAL FIELD Techniques and tools for sub-pixel shifting in motion estimation/compensation in video coding/decoding applications are described. For example, a video encoder and decoder defer bit shifting when performing multi-stage interpolation for pixel values at sub-pixel positions in a reference frame. <i>See, e.g.</i>, Srinivasan at 6:3-4. FIG. 5 is a block diagram of a generalized video decoder system used in several described embodiments. <i>See, e.g.</i>, Srinivasan at 10:47-11:57. B. Video Decoder FIG. 5 is a block diagram of a general video decoder system (500). The decoder system (500) receives information (595) for a compressed sequence of video frames and produces output including a reconstructed frame (505). Particular embodiments of video decoders typically use a variation or supplemented version of the generalized decoder (500). The decoder system (500) decompresses predicted frames and key frames. For the sake of presentation, FIG. 5 shows a path for key frames through the decoder system (500) and a path for forward-predicted frames. Many of the components of the decoder system (500) are used for

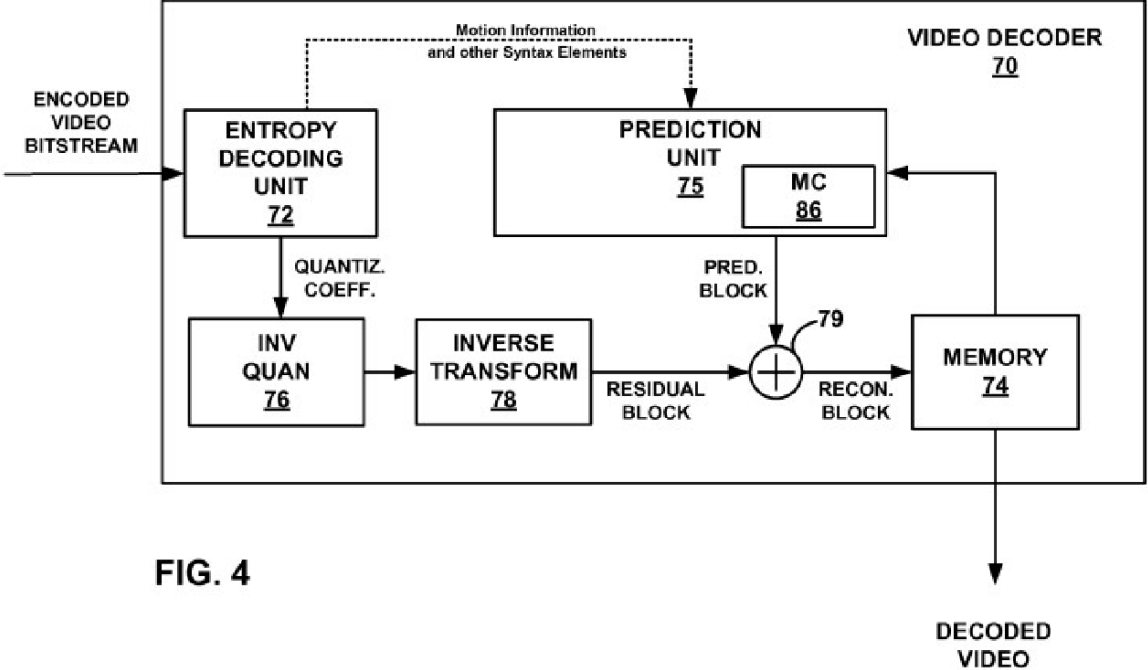
ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	decompressing both key frames and predicted frames. The exact operations performed by those components can vary depending on the type of information being compressed. A buffer (590) receives the information (595) for the compressed video sequence and makes the received information available to the entropy decoder (580). The buffer (590) typically receives the information at a rate that is fairly constant over time, and includes a jitter buffer to smooth short-term variations in bandwidth or transmission. The buffer (590) can include a playback buffer and other buffers as well. Alternatively, the buffer (590) receives information at a varying rate. Before or after the buffer (590), the compressed video information can be channel decoded and processed for error detection and correction. The entropy decoder (580) entropy decodes entropy-coded quantized data as well as entropy-coded side information (e.g., motion information (515), quantization step size), typically applying the inverse of the entropy encoding performed in the encoder. Entropy decoding techniques include arithmetic decoding, differential decoding, Huffman decoding, run length decoding, LZ decoding, dictionary decoding, and combinations of the above. The entropy decoder (580) frequently uses different decoding techniques for different kinds of information (e.g., DC coefficients, AC coefficients, different kinds of side information), and can choose from among multiple code tables within a particular decoding technique. If the frame (505) to be reconstructed is a forward-predicted frame, a motion compensator (530) applies motion information (515) to a reference frame (525) to form a prediction (535) of the frame (505) being reconstructed. For example, the motion compensator (530) uses a macroblock motion vector to find a macroblock in the reference frame (525). A frame buffer (520) stores previous reconstructed frames for use as reference frames. Alternatively, a motion compensator applies another type of motion compensation. The prediction by the motion compensator is rarely perfect, so the decoder (500) also reconstructs prediction residuals. When the decoder needs a reconstructed frame for subsequent motion compensation, the frame store (520) buffers the reconstructed frame for use in predicting the next frame. In some embodiments, the

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	encoder applies a deblocking filter to the reconstructed frame to adaptively smooth discontinuities in the blocks of the frame. An inverse quantizer (570) inverse quantizes entropy-decoded data. In general, the inverse quantizer applies uniform, scalar inverse quantization to the entropy-decoded data with a step-size that varies on a frame-by-frame basis or other basis. Alternatively, the inverse quantizer applies another type of inverse quantization to the data, for example, a non-uniform, vector, or non-adaptive quantization, or directly inverse quantizes spatial domain data in a decoder system that does not use inverse frequency transformations. An inverse frequency transformer (560) converts the quantized, frequency domain data into spatial domain video information. For block-based video frames, the inverse frequency transformer (560) applies an inverse DCT [“IDCT”] or variant of IDCT to blocks of the DCT coefficients, producing motion prediction residual data (545). Alternatively, the frequency transformer (560) applies another conventional inverse frequency transform such as a Fourier transform or uses wavelet or subband synthesis. In some embodiments, the inverse frequency transformer (560) applies an inverse frequency transform to blocks of spatial prediction residuals for key frames. The inverse frequency transformer (560) can apply an 8x8, 8x4, 4x8, or other size inverse frequency transforms. <i>See, e.g., Srinivasan at FIG. 5.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	Figure 5  The diagram illustrates a video decoding system with two main paths: an Intraframe Path and an Interframe Path. The Intraframe Path (top) consists of a Buffer (590) receiving compressed video info (595), followed by an Entropy decoder (580), an Inverse quantizer (570), and an Inverse frequency tran. (560), which outputs a Reconstructed frame (505). The Interframe Path (bottom) also starts with a Buffer (590) receiving compressed video info (595), followed by an Entropy decoder (580), an Inverse quantizer (570), and an Inverse frequency tran. (560), which outputs a Reconstructed residual (545). This residual is added to a Predicted frame (535) at a summing junction (+). The Predicted frame (535) is generated by a Motion compensator (530), which takes a Reference frame (525) from a Frame store (520) and a signal (515) from the Entropy decoder (580) as inputs. The final output of the Interframe Path is a Reconstructed frame (505). <i>See, e.g., Srinivasan at 20:33-38.</i> A component such as the encoder or decoder described with reference to FIGS. 4 and 5, respectively, may perform the multi-stage interpolation (1200, 1300, 1400, 1500). Alternatively, another encoder or decoder, or another type of component, may perform the multi-stage interpolation (1200,1300, 1400, 1500).

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	This element would have further been known in the art and otherwise disclosed in the other references charted in Respondents’ Invalidity Contentions, at least for the reasons set forth in these Contentions. For example, Karczewicz discusses a method for decoding a block of pixels. <i>See, e.g.,</i> Karczewicz at 4:37-39. FIG. 1 is a block diagram illustrating one exemplary video encoding and decoding system that may implement techniques of this disclosure. <i>See, e.g.,</i> Karczewicz at 4:45-46. IG. 4 is an example of a video decoder that may perform offset techniques consistent with this disclosure. <i>See, e.g.,</i> Karczewicz at FIG. 1.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	 FIG. 1 <i>See, e.g., Karczewicz at FIG. 4.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	 FIG. 4 See, e.g., Karczewicz at 4:57-5:3. This disclosure describes video encoding and decoding techniques applicable to bi-directional prediction. In bi-directional prediction, a video block is predictively encoded and decoded based on two different lists of predictive reference data. In one aspect of this disclosure, rounding adjustments to bi-directional predictive data may be purposely eliminated to provide predictive data that lacks any rounding bias. In this case, rounded and unrounded predictive data may both be considered in a rate-distortion analysis to identify the best data for prediction of a given video block. One or more syntax elements may be encoded to indicate the selection, and a decoder may interpret the one or more syntax elements in order to determine whether rounding should be used in the decoding process.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	<i>See, e.g., Karczewicz at 8:37-42.</i> A video sequence typically includes a series of video frames. Video encoder 22 and video decoder 28 may operate on video blocks within individual video frames in order to encode and decode the video data. The video blocks may have fixed or varying sizes, and may differ in size according to a specified coding standard. Each video frame may include a series of slices or other independently decodable units. Each slice may include a series of macroblocks, which may be arranged into sub-blocks. As an example, the ITU-T H.264 standard supports intra prediction in various block sizes, such as 16 by 16, 8 by 8, or 4 by 4 for luma components, and 8×8 for chroma components, as well as inter prediction in various block sizes, such as 16 by 16, 16 by 8, 8 by 16, 8 by 8, 8 by 4, 4 by 8 and 4 by 4 for luma components and corresponding scaled sizes for chroma components. Video blocks may comprise blocks of pixel data, or blocks of transformation coefficients, e.g., following a transformation process such as discrete cosine transform or a conceptually similar transformation process. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>
19[a] 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	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 1[a].</i> <i>See, e.g., Srinivasan at 1:59-65.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
have values with a first precision;	Microsoft Corporation’s Windows Media Video, Version 8 (“WMV8”) includes a video encoder and a video decoder. The WMV8 encoder uses intraframe and interframe compression, and the WMV8 decoder uses intraframe and interframe decompression. Interframe compression in the WMV8 encoder uses block-based motion compensated prediction coding followed by transform coding of the residual error. <i>See, e.g., Srinivasan at 21:22-52.</i> FIG. 14 shows a diagram of multi-stage interpolation (1400) with deferred bit shifting. In the first stage (1410), the component applies (1412) a first filter F1 to one or more x-bit range input values (1411). In conjunction with or after application of the first filter F1, a shortened shift (1414) is performed. The shortened shift (1414) is less than that required to guarantee an x-bit range output value (in view of the expansion factor of the coefficients of the first filter F1), and thus is less than that typically associated with the first filter F1. Accordingly, the shortened shift (1414) produces one or more intermediate values having a dynamic range (of y bits) greater than x bits. For example, the input values have dynamic range of 8 bits, and the intermediate values have a dynamic range of greater than 8 bits. In each of zero or more intermediate stages (1422), which are not shown in detail, the component applies a filter to the one or more intermediate values (1420). One or more intermediate values (1429) having a dynamic range of z bits (greater than x bits) are output from the zero or more intermediate stages (1422) and, in a last stage (1430), the component applies (1432) a last filter FL to the values (1429). The final output from the last filter FL is shifted (1434) by an amount larger than that typically associated with the last filter FL, thereby contracting the dynamic range of output value (1436) to a specified bit depth. For example, the dynamic range (in bits) of the output value (1436) is equal to x or x+1. In one implementation, the shifting of the first stage and any intermediate stage is deferred as much as possible until the final stage. The amount by which the shifting is deferred may depend on the available bit depth for intermediate calculations and the expansion factors of the respective filters.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	See, e.g., Srinivasan at FIG. 14. Figure 14 This element would have further been known in the art and otherwise disclosed in the other references charted in Respondents’ Invalidity Contentions, at least for the reasons set forth in these Contentions. For example, JVT-R201 (Joint Draft 5: Scalable Video Coding (in integrated form with ITU-T Rec. H.264 ISO/IEC 14996-10, published on January 2006) and Noda disclose that the pixels of the current block, the first reference block, and the second reference block have values with a first precision. See e.g., JVT-R201 at 66 (“bit_depth_luma_minus8 specifies the bit depth of the samples

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	of the luma array and the value of the luma quantisation parameter range offset QpBdOffset_Y”), 67 (“bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantisation parameter range offset QpBdOffset_C”), 161 (“8.3.2.1 Reference picture selection process”); Noda at 24:51-56 (“There will now be described an operation of the image encoding apparatus with reference to flowchart of FIG.20B. A video signal wherein the bit depth of each pixel is, for example, N bit depth is input to this image encoding apparatus as an input image signal 100 in units of a frame, for example (S121).”), 25: 15-18 (“The local decoded picture signal 112 of the N bit depth stored in the frame memory 114 is input to the pixel bit depth increase unit 1001, to increase a value of each pixel by Mbits (S126)”). A person of ordinary skill in the art would have found it obvious to incorporate these teachings into Srinivasan and would have had a reasonable expectation of success in doing so, at least because it was well known in the art to have the same precision for pixels of the current block and reference block. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (the same precision for pixels of the current block and reference block) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance). In addition, the following disclosures of Karczewicz would satisfy or render obvious 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. <i>See, e.g., Karczewicz at 17:30-51.</i> FIG. 4 is a block diagram illustrating an exemplary video decoder 70, which may perform the reciprocal decoding techniques to the encoding techniques described above. Video decoder 70 may include an entropy decoding unit 72, a prediction unit 75, an inverse quantization unit 76, an inverse transform unit 78, a memory 74 and adder 79. Prediction unit 75 may include a motion compensation

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	(MC) unit 88, as well as spatial prediction components, which are not shown for simplicity and ease of illustration. Video decoder 70 may receive encoded video data, and one or more syntax elements that indicate whether a rounding adjustment was used to encode the encoded video data. MC unit 86 of prediction unit 75 may generate weighted prediction data that depends on two or more lists of data, as described herein. In accordance with this disclosure, the weighted prediction data does not include the rounding adjustment if the one or more syntax elements indicate that the rounding adjustment was not used to encode the encoded video data. Video decoder 70 can decode the video data using the weighted prediction data, e.g., by invoking adder 79 to add the weighted prediction data (e.g., a prediction block) to residual data (e.g., a residual block). In general, entropy decoding unit 72 receives an encoded bitstream and entropy decodes the bitstream to generate quantized coefficients, motion information and other syntax. The motion information (e.g., motion vectors) and other syntax are forwarded to prediction unit 75 for use in generating the predictive data. Prediction unit 75 performs bi-directional prediction consistent with this disclosure, avoiding rounding adjustments in some cases, and possibly implementing default, implicit or explicit weighted prediction according to the received syntax elements. The syntax elements may identify the type of weighted prediction that to be used, coefficients and offset if explicit weighted prediction to be used, and whether rounding adjustments should be used in the decoding. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
19[b] using said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;	Srinivasan, as evidenced by the example citations below, discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision. <i>See limitation 1[b].</i>
19[c] using said second reference block to obtain a second prediction, said second prediction having the second precision;	Srinivasan, as evidenced by the example citations below, discloses using the second reference block to obtain a second prediction, the second prediction having the second precision. <i>See limitation 1[c].</i>
19[d] obtaining a combined prediction based at least partly upon said first prediction and said second prediction;	Srinivasan, as evidenced by the example citations below, discloses obtaining a combined prediction based at least partly upon the first prediction and the second prediction. <i>See limitation 1[d].</i>
19[e] decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and	Srinivasan, as evidenced by the example citations below, discloses decreasing a precision of the combined prediction by shifting bits of the combined prediction to the right. <i>See limitation 1[e].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
19[f] reconstructing the block of pixels based on the combined prediction.	Srinivasan, as evidenced by the example citations below, discloses reconstructing the block of pixels based on the combined prediction. <i>See, e.g.</i>, Srinivasan at 2:43-48. The WMV8 decoder also uses the motion vectors for macroblocks of the predicted frame to determine the predictors for the macroblocks from the reference frame. To reconstruct the motion-predicted macroblocks of the predicted frame, the WMV8 decoder decompresses the residuals and adds them to the predictors for the macroblocks. <i>See, e.g.</i>, Srinivasan at 10:47-53. B. Video Decoder FIG. 5 is a block diagram of a general video decoder system (500). The decoder system (500) receives information (595) for a compressed sequence of video frames and produces output including a reconstructed frame (505). Particular embodiments of video decoders typically use a variation or supplemented version of the generalized decoder (500). In addition, the following disclosures of Karczewicz would satisfy or render obvious reconstructing the block of pixels based on the combined prediction. <i>See, e.g.</i>, Karczewicz at 17:39-18:5. Video decoder 70 may receive encoded video data, and one or more syntax elements that indicate whether a rounding adjustment was used to encode the encoded video data. MC unit 86 of prediction unit 75 may generate weighted prediction data that depends on two or more lists of data, as described herein. In accordance with this disclosure, the weighted prediction data does not include the rounding adjustment if the one or more syntax elements indicate that the rounding adjustment was not used to encode the encoded video data. Video decoder 70 can decode the video data using the weighted

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	prediction data, e.g., by invoking adder 79 to add the weighted prediction data (e.g., a prediction block) to residual data (e.g., a residual block). In general, entropy decoding unit 72 receives an encoded bitstream and entropy decodes the bitstream to generate quantized coefficients, motion information and other syntax. The motion information (e.g., motion vectors) and other syntax are forwarded to prediction unit 75 for use in generating the predictive data. Prediction unit 75 performs bi-directional prediction consistent with this disclosure, avoiding rounding adjustments in some cases, and possibly implementing default, implicit or explicit weighted prediction according to the received syntax elements. The syntax elements may identify the type of weighted prediction that to be used, coefficients and offset if explicit weighted prediction to be used, and whether rounding adjustments should be used in the decoding. The quantized coefficients are sent from entropy decoding unit 72 to inverse quantization unit 76, which performs inverse quantization. Inverse transform unit 78 then inverse transforms the de-quantized coefficients back to the pixel domain to generate a residual block. Adder 79 combines the prediction data (e.g., a prediction block) generated by prediction unit 75 with the residual block from inverse transform unit 78 to create a reconstructed video block, which may be stored in memory 74 and/or output from video decoder 70 as decoded video output. <i>See, e.g., Karczewicz at 18:52-67.</i> Prediction unit 75 invokes motion compensation unit 86 for block based predictive decoding. In doing so, motion compensation unit 86 generates weighted prediction data based on the syntax (113). Accordingly, if the one or more syntax elements indicate that a rounding adjustment was used, then motion compensation unit 86 generates weighted prediction data that includes the rounding adjustment. However, if the one or more syntax elements indicate that a rounding adjustment was not used, then motion compensation unit 86 generates weighted prediction data that lacks the rounding adjustment. Video decoder 70 can then decode the video data using the weighted prediction data (114). In particular, video decoder 70 may invoke adder 79 to combine weighted prediction data

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
	(e.g., a prediction block) with residual video data (e.g., a residual block) in order to generate a reconstruction of the video data (e.g., a reconstructed video block). Srinivasan and Karczewicz are analogous references as they are in the same field of video encoding/decoding techniques using bi-directional prediction including H.263 and H.264 video encoding/decoding established by the ITU (International Telecommunication Union). Like Srinivasan, FIG. 2 of Karczewicz discloses a block diagram showing a configuration example of an image encoding apparatus generating a prediction having a higher precision to decrease a prediction error signal and upgrade encoding efficiency. A person of ordinary skill in the art would have been motivated to incorporate these teachings of Karczewicz into Srinivasan to improve coding efficiency as specified by Srinivasan. A person of ordinary skill in the art would have had a reasonable expectation of success in doing so, at least because it was well-known in the art for a video encoding system to create a prediction having a higher precision in order to reduce a rounding error that occurs in bi-directional inter prediction. A person of ordinary skill in the art would have seen a benefit in doing so, so that coding efficiency is improved. Further, a person of ordinary skill in the art would have viewed this combination as the application of a known technique (obtaining a prediction with a higher precision) to a known bi-directional prediction method ready for improvement to yield predictable results, and would have understood that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces (e.g., optimizing performance and reducing errors). Therefore, a person of ordinary skill in the art would have been motivated to combine aspects of Srinivasan with Karczewicz in order to make modifications, e.g., to the manner in which the video processing would operate. <i>Further, to the extent Complainant contends this element is not expressly or inherently disclosed by Srinivasan, it would have been obvious in view of the disclosures in Srinivasan, in combination with the knowledge of a person of ordinary skill in the art, or in light of one or more of any of the references identified in Exhibit E-9.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
20. The method according to claim 19, 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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the first motion vector points to a subpixel, the first prediction is obtained by interpolation using pixel values of the first reference block. <i>See claim 2.</i>
21. The method according to claim 20, 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.	Srinivasan, as evidenced by the example citations below, discloses that the first prediction is obtained by interpolation using values of the first reference block by right shifting a sum of a P-tap filter using values of the first reference block. <i>See claim 3.</i>
22. The method according to claim 20, 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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the second motion vector points to an integer sample, the second prediction is obtained by shifting values of the second reference block to the left. <i>See claim 4.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
23. The method according to claim 19, 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.	Srinivasan, as evidenced by the example citations below, discloses that the decreasing the precision of the combined prediction by shifting bits of the combined prediction to the right, further comprises inserting a rounding offset to the combined prediction before the decreasing. <i>See claim 5.</i>
24. The method according to claim 19, 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.	Srinivasan, as evidenced by the example citations below, discloses that 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 the first prediction and values of the second prediction. <i>See claim 6.</i>
25[preamble] An apparatus for decoding a block of pixels, the apparatus comprising:	Srinivasan, as evidenced by the example citations below, discloses an apparatus for decoding a block of pixels. <i>See limitation 19[preamble].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
25[a] 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:	Srinivasan, as evidenced by the example citations below, discloses that at least one processor and at least one memory including computer program code, the at least one memory and computer program code. <i>See limitation 7[a].</i>
25[b] 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;	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 19[a].</i>
25[c] use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;	Srinivasan, as evidenced by the example citations below, discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision. <i>See limitation 19[b].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
25[d] use said second reference block to obtain a second prediction, said second prediction having the second precision;	Srinivasan, as evidenced by the example citations below, discloses using the second reference block to obtain a second prediction, the second prediction having the second precision. <i>See limitation 19[c].</i>
25[e] obtain a combined prediction based at least partly upon said first prediction and said second prediction;	Srinivasan, as evidenced by the example citations below, discloses obtaining a combined prediction based at least partly upon the first prediction and the second prediction. <i>See limitation 19[d].</i>
25[f] decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and	Srinivasan, as evidenced by the example citations below, discloses decreasing a precision of the combined prediction by shifting bits of the combined prediction to the right. <i>See limitation 19[e].</i>
25[g] reconstruct the block of pixels based on the combined prediction.	Srinivasan, as evidenced by the example citations below, discloses reconstructing the block of pixels based on the combined prediction. <i>See limitation 19[f].</i>
26. The apparatus according to claim 25, wherein in an instance in which said first motion vector points to a	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the first motion vector points to a subpixel, the first prediction is obtained by interpolation using pixel values of the first reference block.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.	<i>See claim 20.</i>
27. The apparatus according to claim 26, 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.	Srinivasan, as evidenced by the example citations below, discloses that the first prediction is obtained by interpolation using values of the first reference block by right shifting a sum of a P-tap filter using values of the first reference block. <i>See claim 21.</i>
28. The apparatus according to claim 26, 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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the second motion vector points to an integer sample, the second prediction is obtained by shifting values of the second reference block to the left. <i>See claim 22.</i>
29. The apparatus according to claim 25, wherein the at least one memory and computer code are configured	Srinivasan, as evidenced by the example citations below, discloses that the at least one memory and computer code are configured to cause the apparatus to decrease the precision of the combined

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
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.	prediction by shifting bits of the combined prediction to the right, by inserting a rounding offset to the combined prediction before the decreasing. <i>See claim 23.</i>
30. The apparatus according to claim 25, 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.	Srinivasan, as evidenced by the example citations below, discloses that 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 the first prediction and values of the second prediction. <i>See claim 24.</i>
31[preamble] A computer program product for decoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program	Srinivasan, as evidenced by the example citations below, discloses a computer program product for decoding 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. <i>See limitations 19[preamble], 25[preamble], and 25[a].</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
code portions stored therein, the computer executable program code portions comprising program code instructions configured to:	
31[a] 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;	Srinivasan, as evidenced by the example citations below, discloses 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. <i>See limitation 19[a].</i>
31[b] use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;	Srinivasan, as evidenced by the example citations below, discloses using the first reference block to obtain a first prediction, the first prediction having a second precision, which is higher than the first precision. <i>See limitation 19[b].</i>
31[c] use said second reference block to obtain a second prediction, said	Srinivasan, as evidenced by the example citations below, discloses using the second reference block to obtain a second prediction, the second prediction having the second precision.

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
second prediction having the second precision;	<i>See limitation 19[c].</i>
31[d] obtain a combined prediction based at least partly upon said first prediction and said second prediction;	Srinivasan, as evidenced by the example citations below, discloses obtaining a combined prediction based at least partly upon the first prediction and the second prediction. <i>See limitation 19[d].</i>
31[e] decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and	Srinivasan, as evidenced by the example citations below, discloses decreasing a precision of the combined prediction by shifting bits of the combined prediction to the right. <i>See limitation 19[e].</i>
31[f] reconstruct the block of pixels based on the combined prediction.	Srinivasan, as evidenced by the example citations below, discloses reconstructing the block of pixels based on the combined prediction. <i>See limitation 19[f].</i>
32. The computer program product according to claim 31, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the first motion vector points to a subpixel, the first prediction is obtained by interpolation using pixel values of the first reference block. <i>See claim 20.</i>

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
values of said first reference block.	
33. The computer program product according to claim 32, 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.	Srinivasan, as evidenced by the example citations below, discloses that the first prediction is obtained by interpolation using values of the first reference block by right shifting a sum of a P-tap filter using values of the first reference block. <i>See claim 21.</i>
34. The computer program product according to claim 32, 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.	Srinivasan, as evidenced by the example citations below, discloses that in an instance in which the second motion vector points to an integer sample, the second prediction is obtained by shifting values of the second reference block to the left. <i>See claim 22.</i>
35. The computer program product according to claim 31, wherein the program code instructions configured to	Srinivasan, as evidenced by the example citations below, discloses that the program code instructions configured to decrease the precision of the combined prediction by shifting bits of the combined

ASSERTED CLAIMS	PRIOR ART REFERENCE: “Srinivasan” + “Karczewicz” + “Noda”
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.	prediction to the right, further comprise program code instructions configured to insert a rounding offset to the combined prediction before the decreasing. <i>See claims 23 and 29.</i>
36. The computer program product according to claim 31, 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.	Srinivasan, as evidenced by the example citations below, discloses that 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 the first prediction and values of the second prediction. <i>See claim 24.</i>