REDUCING LATENCY IN VIDEO ENCODING AND DECODING

BACKGROUND

[001] Engineers use compression (also called source coding or source encoding) to reduce the bit rate of digital video. Compression decreases the cost of storing and transmitting video information by converting the information into a lower bit rate form. Decompression (also called decoding) reconstructs a version of the original information from the compressed form. A "codec" is an encoder/decoder system.

[002] Over the last two decades, various video codec standards have been adopted, including the H.261, H.262 (MPEG-2 or ISO/IEC 13818-2), H.263 and H.264 (AVC or ISO/IEC 14496-10) standards and the MPEG-1 (ISO/IEC 11172-2), MPEG-4 Visual (ISO/IEC 14496-2) and SMPTE 421M standards. More recently, the HEVC standard is under development. A video codec standard typically defines options for the syntax of an encoded video bitstream, detailing parameters in the bitstream when particular features are used in encoding and decoding. In many cases, a video codec standard also provides details about the decoding operations a decoder should perform to achieve correct results in decoding.

[003] A basic goal of compression is to provide good rate-distortion performance. So, for a particular bit rate, an encoder attempts to provide the highest quality of video. Or, for a particular level of quality/fidelity to the original video, an encoder attempts to provide the lowest bit rate encoded video. In practice, depending on the use scenario, considerations such as encoding time, encoding complexity, encoding resources, decoding time, decoding complexity, decoding resources, overall delay, and/or smoothness in playback also affect decisions made during encoding and decoding.

[004] For example, consider use scenarios such as video playback from storage, video playback from encoded data streamed over a network connection, and video transcoding (from one bit rate to another bit rate, or one standard to another standard). At the encoder side, such applications may permit off-line encoding that is not at all time-sensitive. Therefore, an encoder can increase encoding time and increase resources used during encoding to find the most efficient way to compress video, and thereby improve rate-distortion performance. If a small amount of delay is also acceptable at the decoder side, the encoder can further improve rate-distortion performance, e.g., by exploiting inter-picture dependencies from pictures farther ahead in a sequence.

[005] On the other hand, consider use scenarios such as remote desktop conferencing, surveillance video, video telephony and other real-time communication scenarios. Such applications are time-sensitive. Low latency between recording of input pictures and playback of output pictures is a key factor in performance. When encoding/decoding tools adapted for non-real-time communication are applied in real-time communication scenarios, overall latency is often unacceptably high. The delays that these tools introduce during encoding and decoding may improve performance for regular video playback, but they disrupt real-time communication.

SUMMARY

[006] In summary, the detailed description presents techniques and tools for reducing latency in video encoding and decoding. The techniques and tools can reduce latency so as to improve responsiveness in real-time communication. For example, the techniques and tools reduce overall latency by constraining latency due to reordering of video frames, and by indicating the constraint on frame reordering latency with one or more syntax elements that accompany coded data for the video frames.

[007] According to one aspect of the techniques and tools described herein, a tool such as a video encoder, real-time communication tool with a video encoder, or other tool, sets one or more syntax elements that indicate a constraint on latency (e.g., a constraint on frame reordering latency consistent with inter-frame dependencies between multiple frames of a video sequence). The tool outputs the syntax element(s), thereby facilitating simpler and quicker determination of when reconstructed frames are ready for output in terms of output order of the frames.

[008] According to another aspect of the techniques and tools described herein, a tool such as a video decoder, real-time communication tool with a video decoder, or other tool, receives and parses one or more syntax elements that indicate a constraint on latency (e.g., a constraint on frame reordering latency). The tool also receives encoded data for multiple frames of a video sequence. At least some of the encoded data is decoded to reconstruct one of the frames. The tool can determine the constraint on latency based on the syntax element(s), then use the constraint on latency to determine when a reconstructed frame is ready for output (in terms of output order). The tool outputs the reconstructed frame.

[009] The foregoing and other objects, features, and advantages of the invention will become more apparent from the following detailed description, which proceeds with reference to the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[010] Figure 1 is a diagram of an example computing system in which some described embodiments can be implemented.

[011] Figures 2a and 2b are diagrams of example network environments in which some described embodiments can be implemented.

[012] Figure 3 is a diagram of an example encoder system in conjunction with which some described embodiments can be implemented.

[013] Figure 4 is a diagram of an example decoder system in conjunction with which some described embodiments can be implemented.

[014] Figures 5a-5e are diagrams showing coded order and output order for frames in several example series.

[015] Figure 6 is a flowchart showing an example technique for setting and outputting one or more syntax elements that indicate a constraint on latency.

[016] Figure 7 is a flowchart showing an example technique for reduced-latency decoding.

DETAILED DESCRIPTION

[017] The detailed description presents techniques and tools for reducing latency in video encoding and decoding. The techniques and tools can help reduce latency so as to improve responsiveness in real-time communication.

[018] In video coding/decoding scenarios, some delay is inevitable between the time an input video frame is received and the time the frame is played back. The frame is encoded by an encoder, delivered to a decoder and decoded by the decoder, and some amount of latency is caused by practical limitations on encoding resources, decoding resources and/or network bandwidth. Other latency is avoidable, however. For example, latency might be introduced by an encoder and decoder to improve rate-distortion performance (e.g., to exploit inter-frame dependencies from pictures farther ahead in a sequence). Such latency can be reduced, although there may be a penalty in terms of rate-distortion performance, processor utilization or playback smoothness.

[019] With techniques and tools described herein, latency is decreased by constraining latency (hence, limiting the temporal extent of inter-frame dependencies) and indicating

the constraint on latency to a decoder. For example, the constraint on latency is a constraint on frame reordering latency. Alternatively, the constraint on latency is a constraint in terms of seconds, milliseconds, or another time measure. The decoder can then determine the constraint on latency and use the constraint when determining which frames are ready for output. In this way, delay can be reduced for remote desktop conferencing, video telephony, video surveillance, web camera video and other real-time communication applications.

[020] Some of the innovations described herein are illustrated with reference to syntax elements and operations specific to the H.264 and/or HEVC standard. Such innovations can also be implemented for other standards or formats.

[021] More generally, various alternatives to the examples described herein are possible. Certain techniques described with reference to flowchart diagrams can be altered by changing the ordering of stages shown in the flowcharts, by splitting, repeating or omitting certain stages, etc. The various aspects of latency reduction for video encoding and decoding can be used in combination or separately. Different embodiments use one or more of the described techniques and tools. Some of the techniques and tools described herein address one or more of the problems noted in the background. Typically, a given technique/tool does not solve all such problems.

I. Example Computing Systems.

[022] Figure 1 illustrates a generalized example of a suitable computing system (100) in which several of the described techniques and tools may be implemented. The computing system (100) 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 systems.

[023] With reference to Figure 1, the computing system (100) includes one or more processing units (110, 115) and memory (120, 125). In Figure 1, this most basic configuration (130) is included within a dashed line. The processing units (110, 115) execute computer-executable instructions. A processing unit can be a general-purpose central processing unit (CPU), processor in an application-specific integrated circuit (ASIC) or any other type of processor. In a multi-processing system, multiple processing units execute computer-executable instructions to increase processing power. For example, Figure 1 shows a central processing unit (110) as well as a graphics processing unit or co-processing unit (115). The tangible memory (120, 125) 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, accessible by the processing unit(s). The memory (120, 125) stores software (180) implementing one or more innovations for reducing latency in video encoding and decoding, in the form of computer-executable instructions suitable for execution by the processing unit(s).

[024] A computing system may have additional features. For example, the computing system (100) includes storage (140), one or more input devices (150), one or more output devices (160), and one or more communication connections (170). An interconnection mechanism (not shown) such as a bus, controller, or network interconnects the components of the computing system (100). Typically, operating system software (not shown) provides an operating environment for other software executing in the computing system (100), and coordinates activities of the components of the computing system (100).

[025] The tangible storage (140) 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 in a non-transitory way and which can be accessed within the computing system (100). The storage (140) stores instructions for the software (180) implementing one or more innovations for latency reduction in video encoding and decoding.

[026] The input device(s) (150) 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 system (100). For video encoding, the input device(s) (150) may be a camera, video card, TV tuner card, or similar device that accepts video input in analog or digital form, or a CD-ROM or CD-RW that reads video samples into the computing system (100). The output device(s) (160) may be a display, printer, speaker, CD-writer, or another device that provides output from the computing system (100).

[027] The communication connection(s) (170) 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 can use an electrical, optical, RF, or other carrier.

[028] The techniques and tools can be described in the general context of computer-readable media. Computer-readable media are any available tangible media that can be accessed within a computing environment. By way of example, and not limitation, with the computing system (100), computer-readable media include memory (120, 125), storage (140), and combinations of any of the above.

[029] 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 system 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 system.

[030] The terms "system" and "device" are used interchangeably herein. Unless the context clearly indicates otherwise, neither term implies any limitation on a type of computing system or computing device. In general, a computing system or computing device can be local or distributed, and can include any combination of special-purpose hardware and/or general-purpose hardware with software implementing the functionality described herein.

[031] For the sake of presentation, the detailed description uses terms like "determine" and "use" to describe computer operations in a computing system. These terms are high-level abstractions for operations performed by a computer, and should not be confused with acts performed by a human being. The actual computer operations corresponding to these terms vary depending on implementation.

II. Example Network Environments.

[032] Figures 2a and 2b show example network environments (201, 202) that include video encoders (220) and video decoders (270). The encoders (220) and decoders (270) are connected over a network (250) using an appropriate communication protocol. The network (250) can include the Internet or another computer network.

[033] In the network environment (201) shown in Figure 2a, each real-time communication ("RTC") tool (210) includes both an encoder (220) and a decoder (270) for bidirectional communication. A given encoder (220) can produce output compliant with the SMPTE 421M standard, ISO-IEC 14496-10 standard (also known as H.264 or

AVC), HEVC standard, another standard, or a proprietary format, with a corresponding decoder (270) accepting encoded data from the encoder (220). The bidirectional communication can be part of a video conference, video telephone call, or other two-party communication scenario. Although the network environment (201) in Figure 2a includes two real-time communication tools (210), the network environment (201) can instead include three or more real-time communication tools (210) that participate in multi-party communication.

[034] A real-time communication tool (210) manages encoding by an encoder (220). Figure 3 shows an example encoder system (300) that can be included in the real-time communication tool (210). Alternatively, the real-time communication tool (210) uses another encoder system. A real-time communication tool (210) also manages decoding by a decoder (270). Figure 4 shows an example decoder system (400), which can be included in the real-time communication tool (210). Alternatively, the real-time communication tool (210) uses another decoder system.

[035] In the network environment (202) shown in Figure 2b, an encoding tool (212) includes an encoder (220) that encodes video for delivery to multiple playback tools (214), which include decoders (270). The unidirectional communication can be provided for a video surveillance system, web camera monitoring system, remote desktop conferencing presentation or other scenario in which video is encoded and sent from one location to one or more other locations. Although the network environment (202) in Figure 2b includes two playback tools (214), the network environment (202) can include more or fewer playback tools (214). In general, a playback tool (214) communicates with the encoding tool (212) to determine a stream of video for the playback tool (214) to receive. The playback tool (214) receives the stream, buffers the received encoded data for an appropriate period, and begins decoding and playback.

[036] Figure 3 shows an example encoder system (300) that can be included in the encoding tool (212). Alternatively, the encoding tool (212) uses another encoder system. The encoding tool (212) can also include server-side controller logic for managing connections with one or more playback tools (214). Figure 4 shows an example decoder system (400), which can be included in the playback tool (214). Alternatively, the playback tool (214) uses another decoder system. A playback tool (214) can also include client-side controller logic for managing connections with the encoding tool (212).

[037] In some cases, the use of a syntax element to indicate latency (e.g., frame reordering latency) is specific to a particular standard or format. For example, encoded data can contain one or more syntax elements that indicate a constraint on latency as part of the syntax of an elementary coded video bitstream defined according to the standard or format, or as defined media metadata relating to the encoded data. In such cases, the real-time communication tool (210), encoding tool (212) and/or playback tool (214) with reduced latency may be codec dependent, in that decisions they make can depend on bitstream syntax for a particular standard or format.

[038] In other cases, the use of a syntax element to indicate a constraint on latency (e.g., frame reordering latency) is outside a particular standard or format. For example, syntax element(s) that indicate a constraint on latency can be signaled as part of the syntax of a media transmission stream, media storage file or, more generally, a media system multiplexing protocol or transport protocol. Or, syntax element(s) that indicate latency can be negotiated between real-time communication tools (210), encoding tools (212) and/or playback tools (214) according to a media property negotiation protocol. In such cases, the real-time communication tool (210), encoding tool (212) and playback tool (214) with reduced latency may be codec independent, in that they can work with any available video encoder and decoder, assuming a level of control over inter-frame dependencies set during encoding.

III. Example Encoder Systems.

[039] Figure 3 is a block diagram of an example encoder system (300) in conjunction with which some described embodiments may be implemented. The encoder system (300) can be a general-purpose encoding tool capable of operating in any of multiple encoding modes such as a low-latency encoding mode for real-time communication, transcoding mode, and regular encoding mode for media playback from a file or stream, or it can be a special-purpose encoding tool adapted for one such encoding mode. The encoder system (300) can be implemented as an operating system module, as part of an application library or as a standalone application. Overall, the encoder system (300) receives a sequence of source video frames (311) from a video source (310) and produces encoded data as output to a channel (390). The encoded data output to the channel can include one or more syntax elements that indicate a constraint on latency (e.g., frame reordering latency) to facilitate reduced-latency decoding.

[040] The video source (310) can be a camera, tuner card, storage media, or other digital video source. The video source (310) produces a sequence of video frames at a frame rate of, for example, 30 frames per second. As used herein, the term "frame" generally refers to source, coded or reconstructed image data. For progressive video, a frame is a progressive video frame. For interlaced video, in example embodiments, an interlaced video frame is de-interlaced prior to encoding. Alternatively, two complementary interlaced video fields are encoded as an interlaced video frame or separate fields. Aside from indicating a progressive video frame, the term "frame" can indicate a single non-paired video field, a complementary pair of video fields, a video object plane that represents a video object at a given time, or a region of interest in a larger image. The video object plane or region can be part of a larger image that includes multiple objects or regions of a scene.

[041] An arriving source frame (311) is stored in a source frame temporary memory storage area (320) that includes multiple frame buffer storage areas (321, 322, … , 32*n*). A frame buffer (321, 322, etc.) holds one source frame in the source frame storage area (320). After one or more of the source frames (311) have been stored in frame buffers (321, 322, etc.), a frame selector (330) periodically selects an individual source frame from the source frame storage area (320). The order in which frames are selected by the frame selector (330) for input to the encoder (340) may differ from the order in which the frames are produced by the video source (310), e.g., a frame may be ahead in order, to facilitate temporally backward prediction. Before the encoder (340), the encoder system (300) can include a pre-processor (not shown) that performs pre-processing (e.g., filtering) of the selected frame (331) before encoding.

[042] The encoder (340) encodes the selected frame (331) to produce a coded frame (341) and also produces memory management control signals (342). If the current frame is not the first frame that has been encoded, when performing its encoding process, the encoder (340) may use one or more previously encoded/decoded frames (369) that have been stored in a decoded frame temporary memory storage area (360). Such stored decoded frames (369) are used as reference frames for inter-frame prediction of the content of the current source frame (331). Generally, the encoder (340) includes multiple encoding modules that perform encoding tasks such as motion estimation and compensation, frequency transforms, quantization and entropy coding. The exact operations performed by the encoder (340) can vary depending on compression format.

The format of the output encoded data can be a Windows Media Video format, VC-1 format, MPEG-x format (e.g., MPEG-1, MPEG-2, or MPEG-4), H.26x format (e.g., H.261, H.262, H.263, H.264), HEVC format or other format.

[043] The coded frames (341) and memory management control signals (342) are processed by a decoding process emulator (350). The decoding process emulator (350) implements some of the functionality of a decoder, for example, decoding tasks to reconstruct reference frames that are used by the encoder (340) in motion estimation and compensation. The decoding process emulator (350) uses the memory management control signals (342) to determine whether a given coded frame (341) needs to be reconstructed and stored for use as a reference frame in inter-frame prediction of subsequent frames to be encoded. If the control signals (342) indicate that a coded frame (341) needs to be stored, the decoding process emulator (350) models the decoding process that would be conducted by a decoder that receives the coded frame (341) and produces a corresponding decoded frame (351). In doing so, when the encoder (340) has used decoded frame(s) (369) that have been stored in the decoded frame storage area (360), the decoding process emulator (350) also uses the decoded frame(s) (369) from the storage area (360) as part of the decoding process.

[044] The decoded frame temporary memory storage area (360) includes multiple frame buffer storage areas (361, 362, …, 36*n*). The decoding process emulator (350) uses the memory management control signals (342) to manage the contents of the storage area (360) in order to identify any frame buffers (361, 362, etc.) with frames that are no longer needed by the encoder (340) for use as reference frames. After modeling the decoding process, the decoding process emulator (350) stores a newly decoded frame (351) in a frame buffer (361, 362, etc.) that has been identified in this manner.

[045] The coded frames (341) and memory management control signals (342) are also buffered in a temporary coded data area (370). The coded data that is aggregated in the coded data area (370) can contain, as part of the syntax of an elementary coded video bitstream, one or more syntax elements that indicate a constraint on latency. Or, the coded data that is aggregated in the coded data area (370) can include syntax element(s) that indicate a constraint on latency as part of media metadata relating to the coded video data (e.g., as one or more parameters in one or more supplemental enhancement information ("SEI") messages or video usability information ("VUI") messages).

[046] The aggregated data (371) from the temporary coded data area (370) are processed by a channel encoder (380). The channel encoder (380) can packetize the aggregated data for transmission as a media stream, in which case the channel encoder (380) can add, as part of the syntax of the media transmission stream, syntax element(s) that indicate a constraint on latency. Or, the channel encoder (380) can organize the aggregated data for storage as a file, in which case the channel encoder (380) can add, as part of the syntax of the media storage file, syntax element(s) that indicate a constraint on latency. Or, more generally, the channel encoder (380) can implement one or more media system multiplexing protocols or transport protocols, in which case the channel encoder (380) can add, as part of the syntax of the protocol(s), syntax element(s) that indicate a constraint on latency. The channel encoder (380) provides output to a channel (390), which represents storage, a communications connection, or another channel for the output.

IV. Example Decoder Systems.

[047] Figure 4 is a block diagram of an example decoder system (400) in conjunction with which some described embodiments may be implemented. The decoder system (400) can be a general-purpose decoding tool capable of operating in any of multiple decoding modes such as a low-latency decoding mode for real-time communication and regular decoding mode for media playback from a file or stream, or it can be a special-purpose decoding tool adapted for one such decoding mode. The decoder system (400) can be implemented as an operating system module, as part of an application library or as a standalone application. Overall, the decoder system (400) receives coded data from a channel (410) and produces reconstructed frames as output for an output destination (490). The coded data can include one or more syntax elements that indicate a constraint on latency (e.g., frame reordering latency) to facilitate reduced-latency decoding.

[048] The decoder system (400) includes a channel (410), which can represent storage, a communications connection, or another channel for coded data as input. The channel (410) produces coded data that has been channel coded. A channel decoder (420) can process the coded data. For example, the channel decoder (420) de-packetizes data that has been aggregated for transmission as a media stream, in which case the channel decoder (420) can parse, as part of the syntax of the media transmission stream, syntax element(s) that indicate a constraint on latency. Or, the channel decoder (420) separates coded video data that has been aggregated for storage as a file, in which case the channel

decoder (420) can parse, as part of the syntax of the media storage file, syntax element(s) that indicate a constraint on latency. Or, more generally, the channel decoder (420) can implement one or more media system demultiplexing protocols or transport protocols, in which case the channel decoder (420) can parse, as part of the syntax of the protocol(s), syntax element(s) that indicate a constraint on latency.

[049] The coded data (421) that is output from the channel decoder (420) is stored in a temporary coded data area (430) until a sufficient quantity of such data has been received. The coded data (421) includes coded frames (431) and memory management control signals (432). The coded data (421) in the coded data area (430) can contain, as part of the syntax of an elementary coded video bitstream, one or more syntax elements that indicate a constraint on latency. Or, the coded data (421) in the coded data area (430) can include syntax element(s) that indicate a constraint on latency as part of media metadata relating to the encoded video data (e.g., as one or more parameters in one or more SEI messages or VUI messages). In general, the coded data area (430) temporarily stores coded data (421) until such coded data (421) is used by the decoder (450). At that point, coded data for a coded frame (431) and memory management control signals (432) are transferred from the coded data area (430) to the decoder (450). As decoding continues, new coded data is added to the coded data area (430) and the oldest coded data remaining in the coded data area (430) is transferred to the decoder (450).

[050] The decoder (450) periodically decodes a coded frame (431) to produce a corresponding decoded frame (451). As appropriate, when performing its decoding process, the decoder (450) may use one or more previously decoded frames (469) as reference frames for inter-frame prediction. The decoder (450) reads such previously decoded frames (469) from a decoded frame temporary memory storage area (460). Generally, the decoder (450) includes multiple decoding modules that perform decoding tasks such as entropy decoding, inverse quantization, inverse frequency transforms and motion compensation. The exact operations performed by the decoder (450) can vary depending on compression format.

[051] The decoded frame temporary memory storage area (460) includes multiple frame buffer storage areas (461, 462, ..., 46*n*). The decoded frame storage area (460) is an example of a decoded picture buffer. The decoder (450) uses the memory management control signals (432) to identify a frame buffer (461, 462, etc.) in which it can store a

decoded frame (451). The decoder (450) stores the decoded frame (451) in that frame buffer.

[052] An output sequencer (480) uses the memory management control signals (432) to identify when the next frame to be produced in output order is available in the decoded frame storage area (460). To reduce latency of the encoding-decoding system, the output sequencer (480) uses syntax elements that indicate constraints on latency to expedite identification of frames to be produced in output order. When the next frame (481) to be produced in output order is available in the decoded frame storage area (460), it is read by the output sequencer (480) and output to the output destination (490) (e.g., display). In general, the order in which frames are output from the decoded frame storage area (460) by the output sequencer (480) may differ from the order in which the frames are decoded by the decoder (450).

V. Syntax Elements that Facilitate Reduced-latency Encoding and Decoding.

[053] In most video codec systems, the coded order (also called the decoding order or bitstream order) is the order in which video frames are represented in coded data in a bitstream and, hence, processed during decoding. The coded order may differ from the order in which the frames are captured by a camera before encoding and differ from the order in which decoded frames are displayed, stored, or otherwise output after decoding (output order or display order). Reordering of frames relative to the output order has benefits (primarily in terms of compression capability), but it increases the end-to-end latency of encoding and decoding processes.

[054] Techniques and tools described herein reduce latency due to reordering of video frames and, by providing information about constraints on the reordering latency to decoder systems, also facilitate latency reduction by the decoder systems. Such latency reduction is useful for many purposes. For example, it can be used to reduce the time lag that occurs in interactive video communication using a video conferencing system, so that the conversation flow and interactivity of communication between remote participants will be more rapid and natural.

A. Approaches to Output Timing and Output Ordering.

[055] According to the H.264 standard, a decoder can use two approaches to determine when a decoded frame is ready to be output. A decoder can use timing information in the form of decoding timestamps and output timestamps (e.g., as signaled in picture timing SEI messages). Or, the decoder can use buffering capacity limits

signaled with various syntax elements to determine when a decoded frame is ready to be output.

[056] Timing information can be associated with each decoded frame. The decoder can use timing information to determine when a decoded frame can be output. In practice, however, such timing information may be unavailable to a decoder. Moreover, even when timing information is available, some decoders do not actually use this information (e.g., because a decoder has been designed to work regardless of whether timing information is available).

[057] Buffering capacity limits are indicated with several syntax elements according to the H.264 standard (and draft versions of the HEVC standard), including the syntax element max_dec_frame_buffering, the syntax element num_reorder_frames, relative ordering information (termed "picture order count" information) and other memory management control information signaled in the bitstream. The syntax element max_dec_frame_buffering (or the derived variable specified as MaxDpbFrames) specifies the required size of a decoded picture buffer ("DPB") in units of frame buffers. As such, the syntax element max_dec_frame_buffering expresses a top-level memory capacity used for a coded video sequence, so as to enable a decoder to output pictures in the correct order. The syntax element num_reorder_frames (or max_num_reorder_frames) indicates the maximum number of frames (or complementary field pairs, or non-paired fields) that can precede any frame (or complementary field pair, or non-paired field) in coded order and follow it in output order. In other words, num_reorder_frames specifies a constraint on the memory capacity necessary for picture reordering. The syntax element max_num_ref_frames specifies the maximum number of short-term and long-term reference frames (or complementary reference field pairs, or non-paired reference fields) that may be used by the decoding process for inter prediction of any picture in the sequence. The syntax element max_num_ref_frames also determines the size of the sliding window for decoded reference picture marking. Like num_reorder_frames, max_num_ref_frames specifies a constraint on required memory capacity.

[058] A decoder uses the max_dec_frame_buffering (or MaxDpbFrames) and num_reorder_frames syntax elements to determine when a buffering capacity limit has been exceeded. This happens, for example, when a new decoded frame needs to be stored in the DPB, but there is no available space remaining in the DPB. In this situation, the decoder uses picture order count information to identify, among the pictures that have

been decoded, which is the earliest in output order. The picture that is earliest in output order is then output. Such processing is sometimes called "bumping" because a picture is "bumped out" of the DPB by the arrival of a new picture that needs to be stored.

[059] Information indicated with the max_dec_frame_buffering (or MaxDpbFrames) and num_reorder_frames syntax elements suffices for determining memory capacity needed in a decoder. When used to control the "bumping" process for picture output, however, use of such information can introduce latency unnecessarily. As defined in the H.264 standard, the max_dec_frame_buffering and num_reorder_frames syntax elements do not establish a limit on the amount of reordering that can be applied to any particular picture and, hence, do not establish a limit on end-to-end latency. Regardless of the values of these syntax elements, a particular picture can be kept in the DPB for an arbitrarily long time before it is output, which corresponds to substantial latency added by pre-buffering of the source pictures by an encoder.

B. Syntax Elements that Indicate Constraints on Frame Reordering Latency.

[060] Techniques and tools described herein reduce latency in a video communication system. An encoding tool, real-time communication tool, or other tool sets a limit on the extent of reordering that can be applied to any frame in a coded video sequence. For example, the limit is expressed as a number of frames that can precede any given frame in a coded video sequence in output order and follow it in coded order. The limit constrains the reordering latency allowed for any particular frame in the sequence. Stated differently, the limit constrains the temporal extent of reordering (in terms of frames) between coded order and output order that can be applied to any particular frame. Limiting the extent of reordering helps reduce end-to-end delay. Also, establishing such a limit can be useful in real-time system negotiation protocols or application specifications for use scenarios in which reducing latency is important.

[061] One or more syntax elements indicate the constraint on frame reordering latency. Signaling a constraint on frame reordering latency facilitates system-level negotiation for interactive real-time communication or other use scenarios. It provides a way to directly express constraints on frame reordering latency and characterize properties of a media stream or session.

[062] A video decoder can use an indicated constraint on frame reordering latency to enable reduced-latency output of decoded video frames. In particular,

compared to frame "bumping" processes, signaling a constraint on frame reordering latency enables a decoder to more simply and quickly identify frames in a DPB that are ready to be output. For example, a decoder can determine the latency status of a frame in the DPB by computing the difference between coded order for the frame and output order. By comparing the latency status of the frame to the constraint on frame reordering latency, a decoder can determine when the constraint on frame reordering latency has been reached. The decoder can immediately output any frame that has reached this limit. This can help the decoder more rapidly identify frames that are ready for output, compared to "bumping" processes that use a variety of syntax elements and tracking structures. In this way, the decoder can rapidly (and earlier) determine when a decoded frame can be output. The more rapidly (and earlier) the decoder can identify when frames can be output, the more rapidly (and earlier) the decoder can output video to a display or subsequent processing stages.

[063] Thus, using the constraint on frame reordering latency, a decoder can begin output of frames from the decoded frame storage area before the decoded frame storage area is full, but still provide conformant decoding (i.e., decoding all frames such that the frames are bit-exact matches of frames decoded using another, conventional scheme). This significantly reduces delay when the delay (in frames) indicated by the latency syntax element is much lower than the size (in frames) of the decoded frame storage area.

[064] Figures 5a-5e illustrate series (501-505) of frames having different inter-frame dependencies. The series are characterized by different values for (1) the constraint on the memory capacity necessary for picture reordering (that is, the number of frame buffers used to store reference frames for purposes of reordering, e.g., as indicated with a syntax element num_reorder_frames, and (2) a constraint on frame reordering latency, e.g., as specified by a variable MaxLatencyFrames. In Figures 5a-5e, for a given frame F_j^k the subscript j indicates position of the frame in output order and the superscript k indicates position of the frame in coded order. The frames are shown in output order – the output order subscript value increases from left to right. Arrows illustrate inter-frame dependencies for motion compensation, according to which preceding frames in coded order are used for prediction of subsequent frames in coded order. For simplicity, Figures 5a-5e show inter-frame dependencies at the frame level (and not at the level of macroblocks, blocks, etc. at which reference frames can change), and Figures 5a-5e show at most two frames as reference frames for a given frame. In practice, in some

implementations, different macroblocks, blocks, etc. in a given frame can use different reference frames, and more than two reference frames can be used for the given frame.

[065] In Figure 5a, the series (501) includes nine frames. The last frame F_8^1 in output order uses the first frame F_0^0 as a reference frame. The other frames in the series (501) use both the last frame F_8^1 and first frame F_0^0 as reference frames. This means the frame F_0^0 is decoded first, followed by the frame F_8^1, followed by the frame F_1^2, etc. In the series (501) shown in Figure 5a, the value of num_reorder_frames is 1. At any point in the processing of the decoder system, among the frames shown in Figure 5a, there is only one frame (F_8^1) stored in the decoded frame storage area *for reordering purposes*. (The first frame F_0^0 is also used as a reference frame and is stored, but is not stored for reordering purposes. Because the output order for the first frame F_0^0 is less than the output order of the intermediate frames, the first frame F_0^0 is not counted for purposes of num_reorder_frames.) Despite the low value for num_reorder_frames, the series (501) has a relatively high latency – the value of MaxLatencyFrames is 7. After encoding the first frame F_0^0, the encoder waits until it has buffered eight more source frames before encoding the next frame F_1^2 in output order, because the next frame F_1^2 depends on the last frame F_8^1 in the series (501). The value of MaxLatencyFrames is effectively the maximum allowed difference between the subscript value and the superscript value for any particular coded frame.

[066] In Figure 5b, the series (502) includes nine frames, as in the series (501) of Figure 5a, but the inter-frame dependencies are different. Temporal reordering of frames occurs over short extents. As a result, the series (502) has a much lower latency – the value of MaxLatencyFrames is 1. The value of num_reorder_frames is still 1.

[067] In Figure 5c, the series (503) includes ten frames. The longest inter-frame dependency is shorter (in temporal extent) than the longest inter-frame dependency in Figure 5a, but longer than the longest inter-frame dependency in Figure 5b. The series (503) has the same low value of 1 for num_reorder_frames, and it has a relatively low value of 2 for MaxLatencyFrames. The series (503) therefore allows a lower end-to-end latency than the series (501) of Figure 5a, although not as low as the allowable latency of the series (502) of Figure 5b.

[068] In Figure 5d, the series (504) includes frames organized in a temporal hierarchy with three temporal layers according to inter-frame dependencies. The lowest temporal resolution layer includes the first frame F_0^0 and last frame F_8^1. The next

temporal resolution layer adds the frame F_4^2, which depends on the first frame F_0^0 and last frame F_8^1. The highest temporal resolution layer adds the remaining frames. The series (504) shown in Figure 5d has a relatively low value of 2 for num_reorder_frames but a relatively high value of 7 for MaxLatencyFrames, at least for the highest temporal resolution layer, due to the difference between coded order and output order for the last frame F_8^1. If only the intermediate temporal resolution layer or lowest temporal resolution layer is decoded, the constraint on frame reordering delay can be reduced to 1 (for the intermediate layer) or 0 (for the lowest layer). To facilitate reduced-latency decoding at various temporal resolutions, syntax elements can indicate constraints on frame reordering latency for different layers in a temporal hierarchy.

[069] In Figure 5e, the series (505) includes frames organized in a temporal hierarchy with three temporal layers according to different inter-frame dependencies. The lowest temporal resolution layer includes the first frame F_0^0, middle frame F_4^1 and last frame F_8^5. The next temporal resolution layer adds the frames F_2^2 (which depends on the first frame F_0^0 and middle frame F_4^1) and F_6^6 (which depends on the middle frame $F_4^1$and last frame F_8^5). The highest temporal resolution layer adds the remaining frames. Compared to the series (504) of Figure 5d, the series (505) of Figure 5e still has a relatively low value of 2 for num_reorder_frames but has a lower value of 3 for MaxLatencyFrames, at least for the highest temporal resolution layer, due to the difference between coded order and output order for the middle frame F_4^1 and last frame F_8^5. If only the intermediate temporal resolution layer or lowest temporal resolution layer is decoded, the constraint on frame reordering delay can be reduced to 1 (for the intermediate layer) or 0 (for the lowest layer).

[070] In the examples shown in Figures 5a-5e, if the value of MaxLatencyFrames is known, a decoder can identify certain frames as being ready for immediate output upon receipt of the preceding frame in output order. For a given frame, the frame's output order value minus the frame's coded order value may be equal to the value of MaxLatencyFrames. In this case, the given frame is ready for output as soon as the frame preceding it in output order is received. (In contrast, such frames could not be identified as ready for output using num_reorder_frames alone until additional frames were received or the end of the sequence was reached.) In particular, a decoder can use the value of MaxLatencyFrames to enable earlier output of the following frames:

- In the series (501) of Figure 5a, the frame F_8^1.

- In the series (502) of Figure 5b, the frames F_2^1, F_4^3, F_6^5, and F_8^7.
- In the series (503) of Figure 5c, the frames F_3^1, F_6^4, and F_9^7.
- In the series (504) of Figure 5d, the frame F_8^1.
- In the series (505) of Figure 5e, the frames F_4^1, and F_8^5.

In addition, declaration or negotiation of the value of MaxLatencyFrames at the system level can provide a summary expression of the latency characteristics of the bitstream or session in a way that is not enabled by measurement of reordering storage capacity and indication of such capacity using num_reorder_frames.

C. Example Implementations.

[071] Syntax elements that indicate a constraint on frame reordering latency can be signaled in various ways, depending on implementation. The syntax elements can be signaled as part of a sequence parameter set ("SPS"), picture parameter set ("PPS"), or other element of the bitstream, signaled as part of SEI messages, VUI messages or other metadata, or signaled in some other way. In any of the implementations, a syntax element indicating a constraint value can be encoded using unsigned exponential-Golomb coding, some other form of entropy coding, or fixed length coding, then signaled. A decoder performs corresponding decoding after receiving the syntax element.

[072] In a first implementation, a flag max_latency_limitation_flag is signaled. If the flag has a first binary value (e.g., 0), no constraint on frame reordering latency is imposed. In this case, the value of a max_latency_frames syntax element is not signaled or is ignored. Otherwise (the flag having a second binary value such as 1), the value of a max_latency_frames syntax element is signaled to indicate the constraint on frame reordering latency. For example, in this case, the value signaled for the max_latency_frames syntax element can be any non-negative integer value.

[073] In a second implementation, a syntax element max_latency_frames_plus1 is signaled to indicate the constraint on frame reordering latency. If max_latency_frames_plus1 has a first value (e.g., 0), no constraint on frame reordering latency is imposed. For other values (e.g., non-zero values), the value of the constraint on frame reordering latency is set to max_latency_frames_plus1 – 1. For example, the value of max_latency_frames_plus1 is in the range of 0 to $2^{32} - 2$, inclusive.

[074] Similarly, in a third implementation, a syntax element max_latency_frames is signaled to indicate the constraint on frame reordering latency. If max_latency_frames has a first value (e.g., a maximum value), no constraint on frame reordering latency is

imposed. For other values (e.g., values less than the maximum value), the value of the constraint on frame reordering latency is set to max_latency_frames.

[075] In a fourth implementation, the constraint on frame reordering latency is indicated relative to a maximum size of frame memory. For example, the latency constraint is signaled as an increase relative to the num_reorder_frames syntax element. Ordinarily, the constraint on frame reordering latency (in terms of frames) is greater than or equal to num_reorder_frames. To save bits in signaling of the latency constraint, the difference between the latency constraint and num_reorder_frames is encoded (e.g., using unsigned exponential-Golomb coding, some other form of entropy coding) then signaled. A syntax element max_latency_increase_plus1 is signaled to indicate the constraint on frame reordering latency. If max_latency_increase_plus1 has a first value (e.g., 0), no constraint on frame reordering latency is imposed. For other values (e.g., non-zero values), the value of the constraint on frame reordering latency is set to num_reorder_frames + max_latency_increase_plus1 – 1. For example, the value of max_latency_increase_plus1 is in the range of 0 to $2^{32} - 2$, inclusive.

[076] Alternatively, one or more syntax elements that indicate a constraint on frame reordering latency are signaled in some other way.

D. Other Ways of Indicating Constraints on Latency.

[077] In many of the preceding examples, the constraint on latency is a constraint on frame reordering latency that is expressed in terms of a count of frames. More generally, the constraint on latency is a constraint on delay that can be expressed in terms of a frame count or expressed in terms of seconds, milliseconds or another time measure. For example, the constraint on latency can be expressed as an absolute time measure such as 1 second or 0.5 seconds. An encoder can convert such a time measure to a count of frames (considering the frame rate of video), then encode the video such that inter-frame dependencies between multiple frames of a video sequence are consistent with the frame count. Or, regardless of frame reordering and inter-frame dependencies, the encoder can use the time measure to limit the extent to which delay is used to smooth out short-term fluctuations in bit rate of encoded video, encoding complexity, network bandwidth, etc. A decoder can use the time measure to determine when a frame can be output from a decoded picture buffer.

[078] The constraint on latency can be negotiated between a transmitter side and receiver side so as to trade-off responsiveness (lack of delay) with ability to smooth out

short-term fluctuations in bit rate of encoded video, ability to smooth out short-term fluctuations in encoding complexity, ability to smooth out short-term fluctuations in network bandwidth and/or another factor that benefits from increased delay. In such negotiations, it may be helpful to establish and characterize the constraint on latency in a way that is independent of frame rate. The constraint can then be applied during encoding and decoding, considering the frame rate of the video. Or, the constraint can be applied during encoding and decoding regardless of the frame rate of the video.

E. Generalized Techniques for Setting and Outputting Syntax Elements.

[079] Figure 6 shows an example technique (600) for setting and outputting syntax elements that facilitate reduced-latency decoding. For example, a real-time communication tool or encoding tool described with reference to Figure 2a and 2b performs the technique (600). Alternatively, another tool performs the technique (600).

[080] To start, the tool sets (610) one or more syntax elements that indicate a constraint on latency (e.g., frame reordering latency, latency in terms of a time measure) consistent with inter-frame dependencies between multiple frames of a video sequence. When the tool includes a video encoder, the same tool can also receive the frames, encode the frames to produce encoded data (using the inter-frame dependencies that are consistent with the constraint on frame reordering latency), and output the encoded data for storage or transmission.

[081] Typically, a constraint on frame reordering latency is a reordering latency allowed for any frame in the video sequence. The constraint can be expressed in various ways, however, and have various other meanings. For example, the constraint can be expressed in terms of a maximum count of frames that can precede a given frame in output order but follow the given frame in coded order. Or, the constraint can be expressed as a maximum difference between coded order and output order for any frame in the video sequence. Or, focusing on an individual frame, the constraint can be expressed as the reordering latency associated with a given, specific frame in the video sequence. Or, focusing on a group of frames, the constraint can be expressed as the reordering latency associated with the group of frames in the video sequence. Or, the constraint can be expressed in some other way.

[082] Next, the tool outputs (620) the syntax element(s). This facilitates determination of when reconstructed frames are ready for output in terms of output order of the multiple frames. The syntax element(s) can be output as part of a sequence

parameter set or picture parameter set in an elementary coded video bitstream, as part of syntax for a media storage file or media transmission stream that also includes encoded data for the frames, as part of a media properties negotiation protocol (e.g., during exchange of stream or session parameter values in system-level negotiation), as part of media system information multiplexed with encoded data for the frames, or as part of media metadata relating to encoded data for the frames (e.g., in SEI messages or VUI messages). Different syntax elements can be output to indicate memory capacity requirements. For example, a buffer size syntax element (such as max_dec_frame_buffering) can indicate maximum size of a DPB, and a frame memory syntax element (such as num_reorder_frames) can indicate maximum size of frame memory for reordering.

[083] The value of the constraint on latency can be represented in various ways, as described in section V.C. For example, the tool outputs a flag that indicates presence or absence of the syntax element(s). If the flag indicates the syntax element(s) are absent, then the constraint on latency is undefined or has a default value. Otherwise, the syntax element(s) follow and indicate the constraint on latency. Or, one value of the syntax element(s) indicates the constraint on latency is undefined or has a default value, and other possible values of the syntax element(s) indicate an integer count for the constraint on latency. Or, for cases in which the constraint on latency is a constraint on frame reordering latency, a given value of the syntax element(s) indicates an integer count for the constraint on frame reordering latency relative to a maximum size of frame memory for reordering, which is indicated with a different syntax element such as num_reorder_frames. Alternatively, the constraint on latency is represented in some other way.

[084] In some implementations, the frames of the video sequence are organized according to a temporal hierarchy. In this case, different syntax elements can indicate different constraints on frame reordering latencies for different temporal layers of the temporal hierarchy.

F. Generalized Techniques for Receiving and Using Syntax Elements.

[085] Figure 7 shows an example technique (700) for receiving and using syntax elements that facilitate reduced-latency decoding. For example, a real-time communication tool or playback tool described with reference to Figure 2a and 2b performs the technique (700). Alternatively, another tool performs the technique (700).

[086] To start, the tool receives and parses (710) one or more syntax elements that indicate a constraint on latency (e.g., frame reordering latency, latency in terms of a time measure). For example, the parsing includes reading the one or more syntax elements that indicate the constraint on latency from the bitstream. The tool also receives (720) encoded data for multiple frames of a video sequence. The tool can parse the syntax element(s) and, based upon syntax element(s), determine the constraint on latency. Typically, a constraint on frame reordering latency is a reordering latency allowed for any frame in the video sequence. The constraint can be expressed in various ways, however, and have various other meanings, as described in the previous section. The syntax element(s) can be signaled as part of a sequence parameter set or picture parameter set in an elementary coded video bitstream, as part of syntax for a media storage file or media transmission stream, as part of a media properties negotiation protocol, as part of media system information multiplexed with the encoded data, or as part of media metadata relating to the encoded data. The tool can receive and parse different syntax elements which indicate memory capacity requirements, for example, a buffer size syntax element such as max_dec_frame_buffering and a frame memory syntax element such as num_reorder_frames.

[087] The value of the constraint on latency can be represented in various ways, as described in section V.C. For example, the tool receives a flag that indicates presence or absence of the syntax element(s). If the flag indicates the syntax element(s) are absent, then the constraint on latency is undefined or has a default value. Otherwise, the syntax element(s) follow and indicate the constraint on latency. Or, one value of the syntax element(s) indicates the constraint on latency is undefined or has a default value, and other possible values of the syntax element(s) indicate an integer count for the constraint on latency. Or, for cases in which the constraint on latency is a constraint on frame reordering latency a given value of the syntax element(s) indicates an integer count for the constraint on frame reordering latency relative to a maximum size of frame memory for reordering, which is indicated with a different syntax element such as num_reorder_frames. Alternatively, the constraint on latency is signaled in some other way.

[088] Returning to Figure 7, the tool decodes (730) at least some of the encoded data to reconstruct one of the frames. The tool outputs (740) the reconstructed frame. In doing so, the tool can use the constraint on latency to determine when the reconstructed

frame is ready for output, e.g., in terms of output order of the frames of the video sequence.

[089] In some implementations, the frames of the video sequence are organized according to a temporal hierarchy. In this case, different syntax elements can indicate different constraints on frame reordering latencies for different temporal layers of the temporal hierarchy. The tool can select one of the different constraints on frame reordering latencies depending on temporal resolution of the output.

[090] In view of the many possible embodiments to which the principles of the disclosed invention may be applied, it should be recognized that the illustrated embodiments are only preferred examples of the invention and should not be taken as limiting the scope of the invention. Rather, the scope of the invention is defined by the following claims. I therefore claim as my invention all that comes within the scope and spirit of these claims.

I claim:

1. In a computing system that implements a video decoder, a method comprising:

receiving and parsing one or more syntax elements that indicate a constraint on latency;

receiving encoded data for plural frames of a video sequence;

with the video decoder, decoding at least some of the encoded data to reconstruct one of the plural frames; and

outputting the reconstructed frame.

2. The method of claim 1 wherein the constraint on latency is a constraint on frame reordering latency, the method further comprising:

parsing the one or more syntax elements;

determining the constraint on frame reordering latency based upon the one or more syntax elements; and

using the constraint on frame reordering latency to determine when the reconstructed frame is ready for output in terms of output order of the plural frames of the video sequence.

3. The method of claim 2 wherein the plural frames of the video sequence are organized according to a temporal hierarchy, wherein different syntax elements indicate different constraints on frame reordering latencies for different temporal layers of the temporal hierarchy, the method further comprising selecting one of the different constraints on frame reordering latencies depending on temporal resolution of the output.

4. The method of claim 1 wherein the constraint on latency is a constraint on frame reordering latency, and wherein the constraint on frame reordering latency is a reordering latency allowed for any frame in the video sequence.

5. The method of claim 1 wherein the one or more syntax elements and the encoded data are signaled as part of syntax for a coded video bitstream, the method further comprising:

receiving and parsing a buffer size syntax element that indicates maximum size of a decoded picture buffer and a frame memory syntax element that indicates maximum size of frame memory for reordering, wherein the buffer size syntax element and the frame memory syntax element are different than the one or more syntax elements that indicate the constraint on latency.

6. The method of claim 1 wherein the one or more syntax elements are signaled as part of a sequence parameter set, a picture parameter set, syntax for a media storage file that also includes the encoded data, syntax for a media transmission stream that also includes the encoded data, a media properties negotiation protocol, media system information multiplexed with the encoded data or media metadata relating to the encoded data.

7. The method of claim 1 wherein:

one possible value of the one or more syntax elements indicates the constraint on latency is undefined or has a default value, and wherein other possible values of the one or more syntax elements indicate an integer count for the constraint on latency; or

a value of the one or more syntax elements indicates an integer count for the constraint on latency relative to a maximum size of frame memory for reordering, the maximum size frame memory for reordering being indicated with a different syntax element.

8. In a computing system, a method comprising:

setting one or more syntax elements that indicate a constraint on latency consistent with inter-frame dependencies between plural frames of a video sequence; and

outputting the one or more syntax elements, thereby facilitating determination of when reconstructed frames are ready for output in terms of output order of the plural frames.

9. The method of claim 8 wherein the constraint on latency is a constraint on frame reordering latency, and wherein the computing system implements a video encoder, the method further comprising:

receiving the plural frames of the video sequence;

with the video encoder, encoding the plural frames to produce encoded data, wherein the encoding uses the inter-frame dependencies that are consistent with the constraint on frame reordering latency; and

outputting the encoded data for storage or transmission.

10. A computing system comprising a processor, memory and storage, the computing system being adapted to perform a method comprising:

receiving and parsing one or more syntax elements that indicate a constraint on frame reordering latency;

determining the constraint on frame reordering latency based upon the one or more syntax elements, wherein the constraint on frame reordering latency is expressed in terms of a maximum count of frames that can precede a given frame in output order but follow the given frame in coded order;

receiving encoded data for plural frames of a video sequence;

with the video decoder, decoding at least some of the encoded data to reconstruct one of the plural frames; and

outputting the reconstructed frame, including using the constraint on frame reordering latency to determine when the reconstructed frame is ready for output in terms of output order of the plural frames of the video sequence.