2013 Data Compression Conference

Cross Segment Decoding for Improved Quality of Experience for Video Applications

Jiangtao Wen, Shunyao Li, Yao Lu, Meiyuan Fang, Xuan Dong, Huiwen Chang, Pin Tao

Abstract

In this paper, we present an improved algorithm for decoding live streamed or pre-encoded video bitstreams with time-varying qualities. The algorithm extracts information available to the decoder from a high visual quality segment of the clip that has already been received and decoded, but was encoded independently from the current segment. The proposed decoder is capable of significantly improve the Quality of Experience of the user without incurring significant overhead to the storage and computational complexities of both the encoder and the decoder. We present simulation results using the HEVC reference encoder and standard test clips, and discuss areas of improvements to the algorithm and potential ways of incorporating the technique to a video streaming system or standards.

I. Introduction

Video encoding and communications systems have traditionally been designed under the assumption that the encoding system has a much higher processing power and much larger storage than the decoder system. With the ever widening popularity of mobile multimedia applications, especially with user generated content, this assumption is no longer valid. Many widely watched clips on YouTube were captured on mobile phones, but are played back not only on mobile phones or tablet computers but also on smart TVs, smart set-top-boxes, as well as laptop and desktop computers, all of which may possess much more computational and storage resources than the cell phone on which the video clip was originally captured and encoded.

Due to the high complexity associated with implementing a state-of-the-art video encoding system using the H.264/AVC [1] and especially the up-coming HEVC standards [2], [3], encoders found in mobile devices, although compliant to the standard syntaxes, were often designed with sub-optimal implementations of only small subsets of the numerous encoding tools supported by the standard. Typical video encoding tools that are not fully implemented include sub-pel motion estimation (ME), many Inter partition sizes and/or Intra prediction directions, and etc. The number of reference frames used in the ME is also often limited.

On top of the video encoding tools that are out-right unsupported, the algorithms for selecting the tools that *are* supported, conducting rate-distortion optimized ME, quantization, macroblock (MB) mode decision, frame prediction type decision and group of picture (GOP) reference structures are also drastically simplified in many encoding systems. Various implementation constrains (e.g. cache sizes) as well as application requirements (e.g. video conferencing or live streaming) make it very difficult to perform bit rate allocation optimized globally for the entire clip.

Furthermore, when the bitstreams generated by such encoders are streamed over the network, in response to network bandwidth variations, the streaming servers will often adjust the rate at which the clip is encoded downward so as to prevent the playback on the receiving device from stalling.

Jiangtao Wen, Shunyao Li, Meiyuan Fang, Xuan Dong, Huiwen Chang and Pin Tao are with Tsinghua University, Beijing, China (email: jtwen@tsinghua.edu.cn)

Yao Lu was with Tsinghua University, Beijing, China, and is now with the University of California San Diego, La Jolla, CA 92093 (email: luyao@ucsd.edu)

The work was supported by the National Science Fund for Distinguished Young Scholars of China (Grant No. 61125102), the State Key Program of National Natural Science of China (Grant No. 61133008) and the National Significant Science and Technology Projects of China (Grant No. 2012ZX01039-001-003-2).

1068-0314/13 $26.00 © 2013 IEEE
DOI 10.1109/DCC.2013.31

All of the above factors contribute to loss in coding efficiency as well as visual quality variations in the bitstream that the playback device receives. Although it is possible to deploy cloud based systems that transcode such low quality (in terms of encoding performance) bitstreams to higher quality bitstreams, cloud based systems are not viable generically due to scalability issues and privacy concerns.

It is desirable, therefore, to come up with a system that can improve the quality of experience (QoE) of the end user given all the limitations, constraints and variations present in the content encoding, transcoding, streaming and playback processes.

In this paper, we present an improved algorithm for decoding live streamed or pre-encoded video bitstreams with time-varying qualities. The algorithm utilizes information received by the decoder in a segment of the clip that 1) has already been received and decoded, but 2) was encoded independently from the current segment, and 3) has a higher visual quality than the current segment. By extracting information contained in such a segment that is available to the decoder but was not taken advantaged by the encoder, the proposed decoder is capable of significantly improve the QoE of the user without incurring significant overhead to the storage and computational complexities of both the encoder and the decoder.

The rest of the paper is organized as the following: Section II introduces some related ideas for improving user QoE and decoder quality. A detailed description of the proposed algorithm is given in Section III with experimental results using the HEVC standard and standard test clips presented in Section IV. Finally in Section V, we discuss various areas for improving the algorithm and potential ways of incorporating support for the proposed decoding into various video coding and streaming standards.

II. Related Work

Scalable, error resilient and high quality streaming of video content over networks has been studied extensively for well over a decade. In additional to taking the tradeoff between scalability, error resilience and coding efficiency (as measured in rate-distortion performances) into the consideration when designing the video encoding algorithms (e.g. for ME, rate control, and mode decision), many additional tools were introduced to various video coding standards so as to facilitate video streaming with low start-up latency, easier and drift free bitstream switching. These include scalable video coding support in the MPEG family of video coding standards, S-frames [4], as well as SI and SP frames in the AVC/H.264 standard [1].

When encoding video content in a rate-disortion-scalability-error-resiliency optimal manner, to prevent error propagation and to facilitate drift-free bitstream switching, encoders usually do not rely on video encoding tools that aggressively eliminate temporal redundancies, e.g. long term motion prediction with a large number of reference frames. Quite the contrary, the encoders will usually introduce Intra coded frames that will serve as re-synchronization points or drift-free switching pointers to between bitstreams of different bitrates. Due to the low coding efficiency of Intra frames, the more efficient SP and SI frames are proposed, which preserves the drift-free characteristic, but with coding efficiencies significantly improved over Intra frames, albeit still significantly lower than P or B frames. Using Intra, SI and SP frames as switching points, an encoder may encode a given video clip to multiple possible bitstreams with multiple configurations of the reference structure, and/or bitrates. When streaming, depending on network packet losses and network bandwidth variations, the server/streamer may compose a bitstream to be streamed to the client on the fly, based on information about frame losses and bandwidth.

In contrast to encoding a given video clip at multiple bitrates and references structures, scalable video coding (SVC) [5] encodes the video clip into one scalable bitstream that can be parsed by the server to produce bitstreams of the target bandwidth. Although more efficient than storing multiple code representations of the same content, SVC still significantly under-performs conventional, non-scalable coding systems with similar computational and storage resources and video encoding tools. Special, SVC-compliant decoders are also usually required at the receiving end.

On the other hand, many algorithms and systems have been introduced to improve the QoE on the receiving end when packet losses and bandwidth variations occur. The algorithms include various error concealment and error resilient decoding techniques, many of which were reviewed in [6] and [7]. In [8], a technique (termed "Adaptive Media Playout") was introduced to dynamically adjust the playback speed of video and audio based on the playback buffer of the receiver, thereby preventing stalls. In [9], when decoding the prediction residual information in a received frame using an algorithm called "delayed decoding", an optimized estimate of the transform coefficients to be decoded is obtained by the decoder using information contained in frames that are received both before and after the current frame, as opposed to decoding as soon as the current frame is received.

In this paper, we introduce an improved decoding algorithm that also enhances decoded quality without proactively modifying the encoding process (e.g. the ME, mode decision and frame type decision processes), as in the case of scalable coding and encoding with S, SI and SP frames. Similar to delayed decoding, the proposed algorithm also achieves improvement to the decoded quality by using information the decoder already processes but is not traditionally utilized by conventional decoders. However, unlike the case for delayed-decoding, the extra delay and storage and computational requirements introduced is much lower.

III. Algorithm Description

The proposed algorithm is more easily understood in the context of bitrate adaptive streaming of video over the Internet, where to facilitate fine granularity bitrate adaptation in reaction to changes in network conditions, a video clip is divided into relatively short segments, each of which is encoded independently of each other, as illustrated in Figure 1. In Figure 1, a video clip is divided into 3 segments, each encoded at 3 different bitrates, $bitrate1 < bitrate2 < bitrate3$.

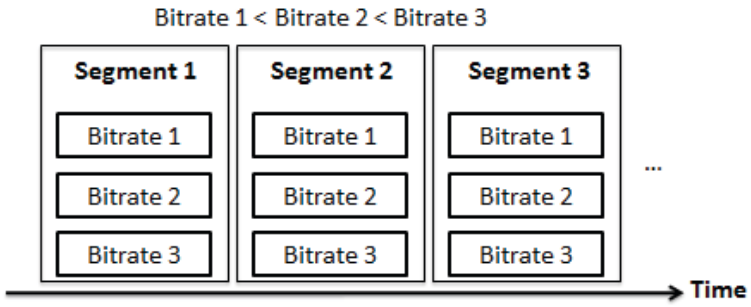


Fig. 1: Segment Based Bitstream Switching for Adaptive Video Streaming

When the clip encoded using the system in Figure 1 is streamed over a network where the bandwidth varies, the server may "stitch" together bitstreams for neighboring segments that have been encoded at different bitrates, as shown in Figure 2, resulting in variations of video quality over time. In many applications, such variations in visual quality is noticeable, annoying and significantly impairs user QoE.

Similar variations in visual quality may also occur when an encoder with a rate allocation algorithm that is not able to allocate the target bitrate in a globally optimized manner over the entire clip. This may be the case due to lack of multiple pass encoding (e.g. for encoding live events) or sufficient look ahead (due to memory or delay requirements), and/when the complexity of the input video varies significantly over time. Figure 3 shows an example, where a uniform 1Mbps was allocated to encode the "Chroma Key" test clip. The bitrate distribution over time when the sequence is encoded with 1-pass H.264 encoding

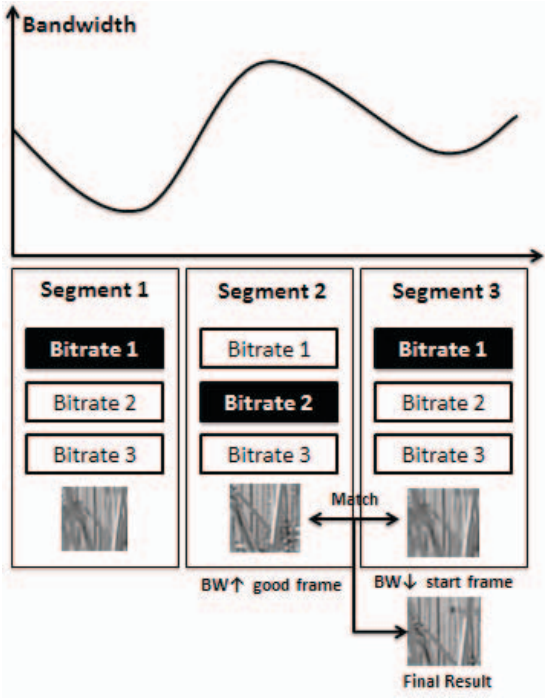


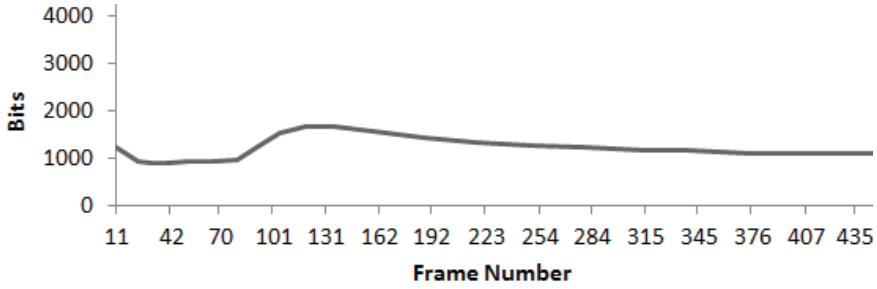
Fig. 2: Stitching Bitstreams for Segments for Streaming over Bandwidth Varying Channel

using the x264 [10] encoder with the default settings and 1-pass encoding is shown in Figure 3(a). Figure 3(b) and (c) show two frames of roughly the same size after encoding, but the visual qualities of the two are noticeably different.

Traditionally, video coding standards are decoder specifications which mandate *exactly* how standard compliant decoders process received standard compliant bitstreams. Any further improvements of the video quality after decoding, e.g. error concealment [7], deblocking and other enhancements are strictly "out-of-the-loop" operations that are essentially post-processings for improving the visual quality of the video frames before they are displayed to the end user, while bearing no impact on the core decoding processes (i.e. entropy decoding, motion compensation, inverse-transform and dequantization) process. Only since the introduction of the H.264/AVC and HEVC standards [1], [2] has deblocking been incorporated into the motion compensation loop of the decoder as a normative and integral part of the decoder. Consequently, the same deblocking module is also incorporated into the encoder.

Our proposed algorithm is also a technique that resides inside the decoding loop. In contrast to H.264/AVC and HEVC deblocking filter however, in the preliminary simulations we have run for and reported in this paper, the technique is not required to be, and therefore not yet, incorporated into the encoder.

In the preliminary implementation, when the visual quality of an input bitstream to a video decoder incorporating the proposed algorithm varies over time, at the transition from a good quality segment to a segment whose video quality is poorer (Figure 2), the decoder will usually have access to information across the segment boundaries, i.e. information from both the good and poor visual quality segments. On the other hand, the same information may or may not have been utilized by the encoder when producing the bitstream.



(a) Bitrate Distribution

(b) Frame No.1

(c) Frame No.250

Fig. 3: Two Frames of Almost Identical Size after Compression but Different Quality. Variations in Video Complexity and Uniformly Distributed Bitrate Result in Significant Variation of Quality over Time.

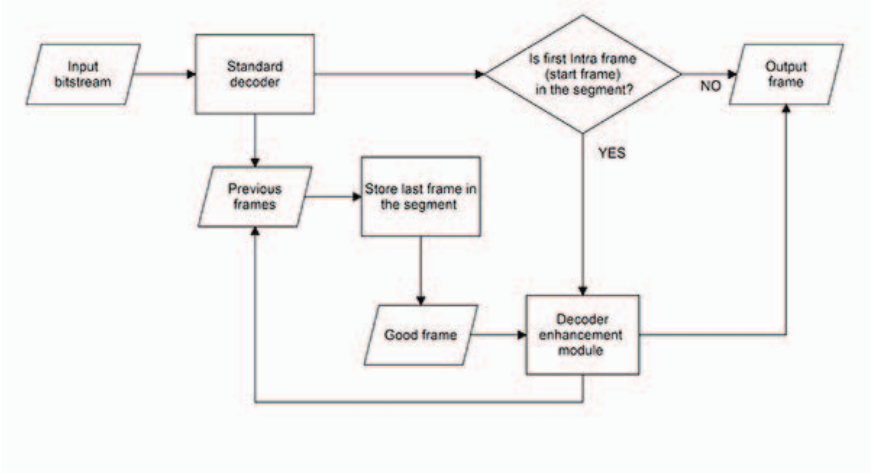


Fig. 4: Flow Diagram of Cross Segment Decoding

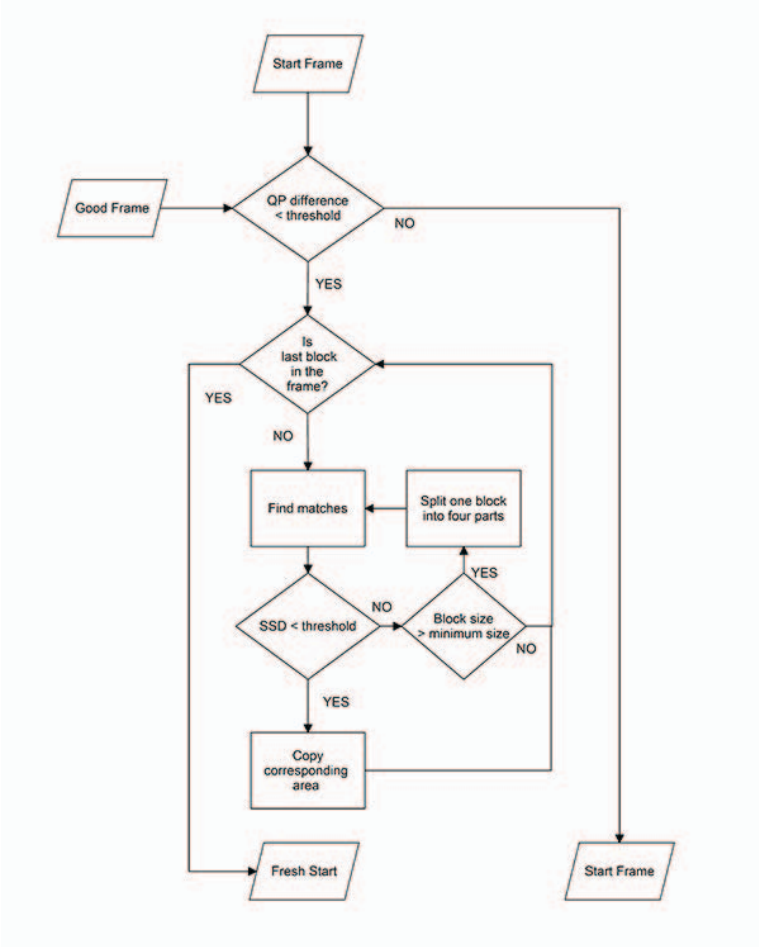


Fig. 5: Flow Diagram of the Proposed Patch Enhancement Algorithm

In this paper, we term the last frame (in display order) in the good quality segment preceding a poor quality segment that was not used as a reference for encoding the poor quality segment a "good frame", and the first Intra frame of the poor quality segment the "start frame".

In our video enhancement algorithm, for video clips that are mostly static and contain little motion, we simply replace patches in the start frame with patches at the co-located positions in the good frame (referred to as "Simple Stitching"). For those clips with more motion content, we use the following "Patch Matching" algorithm: for each patch s in the start frame S, we first find a patch in the good frame R that is most similar to s by using the following iterative propagation update and random search algorithm in [11]. We further optimized the algorithm by incorporating prior knowledge of the temporal movements between two frames, and by leaving some gap between patches, which made the patch sparser, thereby reducing the blurring caused by weighted average calculations.

Algorithm 1 Enhancement (Start frame S, Good frame R)

Step 1. Search approximate similar patch
for each patch s ∈ S **do**
  Initialize SimilarPatch(s) in R by random.
end for
for each patch s ∈ S **do**
  Refine the current guess SimilarPatch(s) by propagating to the neighboring patches and randomly search in a neighborhood range
end for
Step 2. Filter and Tile
for each patch s ∈ S **do**
  if distance(SimilarPatch(s), s)<similarity_threshold **then**
    for every p∈ s **do**
      Add the corresponding pixels in SimilarPatch(s) into the Candidate(p)
    end for
  end if
end for
Set p's color value to the weighted average of pixels in Candidate(p)

In our implementation, we calculated the sum of squared differences (SSD) of patch pixel values in both the $L*a*b$ color space and in the YUV space. A match was found when the SSD between two patches of the same size was below a threshold, otherwise, each of the two patches were recursively divided into 4 patches of the same size, for which the SSD was calculated again, This process was repeated until a match was found, or the sizes of the patches reached the minimum size (8x8 in the experiment).

Once tiling is finished, the resulted frame (termed the fresh start) will be used for all subsequent decoding of the current segment. No further modification to the information and the decoding process is made for the remainder of the segment the same decoded information (headers, MVs, and texture information) and decoding process would be used.

IV. Experimental Results

In our experiments, we used the HEVC HM 6.0 encoder and the low delay configuration to encode the test bitstreams. For each test clip, we ran the HEVC encoder for the first 32 frames of the clip (first segment), followed by HEVC encoding (with the same HEVC low delay configuration) of the remaining frames with frame No. 33 encoded as an Intra frame (second segment).

Num of Enh Frames	BballDrill	Vdo1	Vdo3	Vdo4	Traffic	SEdit	K&S	4People	Johnny	CNSpd	BQSq	GL
30	0.46	1.65	1.08	0.56	0.72	3.10	1.54	1.95	1.09	0.41	0.25	0.01
60	0.37	1.45	1.02	0.49	0.57	2.64	1.37	1.78	0.91	0.39	0.18	-0.02
100	0.31	1.22	0.98	0.50	0.47	2.38	1.25	1.62	0.82	0.46	0.13	-0.04
300	0.17	0.89	0.77	0.51	-	-	1.00	1.20	0.65	0.57	0.02	-0.05
All	0.13	0.67	0.60	0.36	0.44	1.71	0.92	1.06	0.58	0.66	-0.02	-0.05

TABLE I: Average PSNR Improvements (dB).

(a) Yacht (b) Same Frame using Proposed Decoder

Fig. 6: Subjective Quality Comparisons

The average PSNR improvements of enhancing the first 30, 60, 100, and 300 frames after the good frame, as well as enhancing the entire remainder of the sequence can be found in Table I. In the table, Vdo1-Vdo4, CNSPd, BballDrill, BQSq, K&S, SEdit are the HEVC Vidyo, ChinaSpeed, Baseketball Drill, BQSquare, KristenAndSara and SlideEdit test sequences. The GL sequence is the GreenLeaves test sequence of ETSI. The QP of the good frame was 37, and the QP of start frame was 42. Both "Simple Stitching" and "Patch Matching" with SSD calculated in the YUV and $L * a * b$ spaces were experimented, with the one producing the better average PSNR gain for the clip reported in the table. As we can see, the PSNR improvements were significant for all sequences except for the last two with high motion content. However, even though there was a slight PSNR loss after using the proposed decoder for these two sequences, the subjective quality still improved, as can be seen in Figure 7 for "GreenLeaves".

It should be noted that when performing the patch matching and enhancement, certain parameters such as the patch size and threshold for determining match needed to be fine-tuned based on the QP and the content. In "Simple Stitching", the patch size was set between 64x64 and 8x8. The SSD threshold for match in YUV color space was selected from values between 10 to about 600 in increments of 5, while the threshold in $L * a * b$ color space varied from 6 to 21 in increments of 1. The maximum difference between each pixel before and after enhancement ranged from 1 to 14. For "Patch Matching", the size of patches was 8x8, while the neighborhood search range was 12x12, with a gap of 2 between. The SSD threshold here was between 1000 and 100000 in this case, and a value of 50000 was able to produce reasonably good results for most clips.

We are working on algorithms that could automatically select the patch matching algorithm, and determining the corresponding parameters. Given the improvements in PSNR and the relatively very small amount of information needed, even without automatic algorithm for determining such information in the decoder and therefore they have to be sent from the encoder (e.g. in SEI fields) using straightforward natural binary representations of the information will lead to negligible overhead in coding efficiency.

In terms of complexity, the preliminary implementation of the proposed decoder used in the experiments roughly doubles the decoding time of the frame to be enhanced (i.e. the start frame). As only one frame needs to be enhanced, the overall increase to decoding complexity is negligible.

V. Conclusions and Future Work

In this paper, we presented preliminary results using a preliminary implementation of a cross-segment decoder for improving the quality of HEVC decoding. The same algorithm could also be used for AVC decoding. Experimental results using the HEVC test clips showed great potential of the algorithm for applications in a number of scenarios, including adaptive bandwidth video streaming, low complexity encoding coupled with high complexity decoding for mobile video capturing applications and etc.. Areas for further improvement of the algorithm include improvements of the patch matching algorithm, determining the algorithm parameters automatically, incorporating the knowledge of decoder side support of cross segment decoding for improving coding efficiency in low complexity, low delay applications, and etc. Furthermore, the core idea of the system can also be used to intentionally simplify the encoding process on a mobile terminal so that the power consumption for encoding and/or bandwidth for uploading the encoded content could be reduced, with the uploaded content enhanced by a cloud based system or the playback terminal to achieve a similar QoE as if the content had been encoded with a more sophisticated encoder or at a higher bandwidth.

References

[1] T.Wiegand, G.J.Sullivan, G.Bjontegaard, and A.Luthra, "Overview of the H.264/AVC video coding standard," in *IEEE Trans Circuits and Systems for Video Technology*, 2003, pp. 560–576.

[2] G.Sullivan, J.Ohm, W.Han, and T.Wiegand, "Overview of the high efficiency videocoding (HEVC) standard," in *IEEE Trans Circuit and System for Video Technology*, 2012.

[3] J.Ohm, G.Sullivan, H.Schwarz, T.Tan, and T.Wiegand, "Comparison of the coding efficiency of video coding standards including high efficiency video coding (HEVC)," in *IEEE Trans Circuit and System for Video Technology*, 2012.

[4] N. Farber and B. Girod, "Robust H.263 compatible video transmission for mobile access to video servers," in *in Proceedings of the Picture Coding Symposium*, 1996, pp. 575–578.

[5] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," in *IEEE Trans Circuits and Systems for Video Technology*, 2007, pp. 1103–1120.

[6] Y.Wang, S. Wenger, J. Wen, and A. Katsaggelos, "Error resilient video coding techniques," in *IEEE Signal Processing Magazine*, 2000, pp. 61–82.

[7] Y.Wang and Q. Zhu, "Error control and concealment for video communication: A review," in *Proc. of the IEEE*, 1998, pp. 974–997.

[8] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low-delay video streaming over error-prone channels," in *IEEE Trans. Circuits and Systems for Video Technology*, 2010, pp. 841–851.

[9] J. Han, V. Melkote, and K. Rose, "Estimation-theoretic delayed decoding of predictively encoded video sequences," in *Proc. of the IEEE Data Compression Conference (DCC)*, 2010, pp. 119–128.

[10] L. M. E. Al, "X264: A high performance H.264/AVC encoder," 2006.

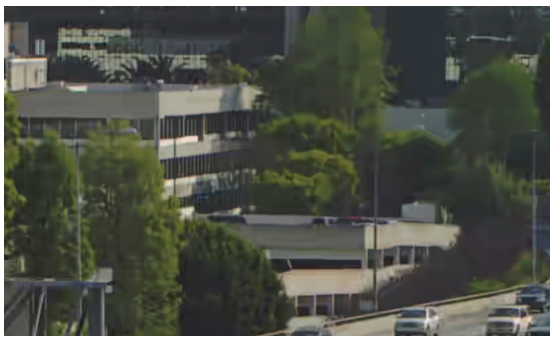
[11] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2009.

(a) Yacht

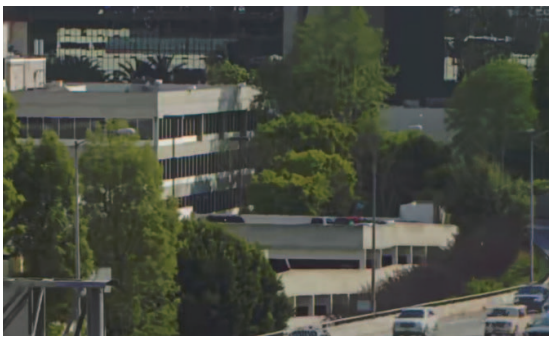
(b) Same Area using Proposed Decoder

(c) Vidyo1

(d) Same Area using Proposed Decoder



(e) Traffic Details



(f) Same Area using Proposed Decoder

(g) GreenLeaves

(h) Same Area using Proposed Decoder

Fig. 7: Detailed Comparisons