EFFICIENT COMMUNICATION OF VIDEO USING METADATA

Aous Thabit Naman, Duncan Edwards, and David Taubman

School of Electrical Engineering and Telecommunications,
University of New South Wales, Sydney, Australia.

ABSTRACT

In traditional video coding schemes, motion information is tightly coupled to the prediction strategy. In this preliminary work, we depart from this model by utilizing metadata to convey motion information to the client; in particular, metadata conveys crude boundaries of objects together with motion information for these objects. Here, we are interested in applications where metadata itself carries semantics that the client is interested in, such as tracking information in surveillance applications. To keep things simple, we focus on the case where we have a single object to track. Therefore, we model each frame as a background region with a foreground region/object, enclosing each region by a quadrilateral that identifies it. The foreground quadrilateral does not follow the exact boundaries of the foreground object; it leaves the task of identifying these boundaries to the client. The advantages of metadata is that it provides a global representation of motion, which allows predicting a given object from potentially all the frames that contain that object. The approach is applicable in fully open loop systems such as in the case of the JPEG2000-Based Scalable Interactive Video (JSIV) paradigm. In this work, we present the concepts behind the proposed approach and detail the modifications introduced to the JSIV server and client policies, presenting some promising preliminary results.

Index Terms— Teleconferencing, video signal processing, image coding, image communication, motion compensation

1. INTRODUCTION

In conventional video coding schemes, such as the MPEG1 through MPEG4 and H.261 through H.264 standards, motion information for a given predicted frame is tightly-coupled to that frame and its reference frames (i.e., tightly-coupled to the prediction strategy for that frame). This arrangement is not well suited for interactive browsing or navigation of media, because, in such applications, the client may not have all the reference frames assumed by a given motion model, or higher quality is possible if the client uses frames other than those required by that motion model; it is worthwhile for the client to investigate other prediction strategies, but this is not possible with existing motion descriptions.

To provide better flexibility and interactivity for video browsing, Naman and Taubman proposed the JPEG2000-Based Scalable Interactive Video (JSIV) paradigm [1–3]; JSIV relies on JPEG2000 to independently store individual frames and provide for quality and spatial resolution scalability, and on a dynamic prediction policy and conditional replenishment to exploit temporal redundancy.

JSIV overcomes the difficulty associated with the existing tightly-coupled strategies by proposing loosely-coupled client-server policies in which the server selects, based on an assumed client policy, the optimal number of quality layers for each JPEG2000 precinct delivered, and the client makes the most of what is available in its cache. In such an arrangement the server can adapt its assumed prediction policy based on the data available to the client.

Motion information is metadata since it is derived from the video sequence itself. Since JSIV only sends data from independently encoded frames (does not send residues), it is useful to investigate methods that can exploit some of the redundancy between the encoded video frames and the data derived from them. In particular, we seek to develop dynamic prediction strategies to combine metadata with available frame contents to form predictions for other frames, so we can use JSIV to deliver only those portions of predicted frames that are not well predicted. This involves modifying both the JSIV client and JSIV server. Here, we are more interested in applications where metadata carries important semantics by itself, such as tracking information in surveillance applications; in these cases, it is very likely that the user/client is interested in receiving the tracking data as much as he/she is interested in receiving the video sequence.

The advantage of using metadata over conventional motion modeling schemes is that metadata are global descriptions of motion; i.e., it is possible to predict a given frame from potentially any frame that contains the same tracked object. This opens up a lot of possibilities which are not fully exploited here. Another advantage of metadata is that the amount of data associated with them is relatively smaller than that associated with conventional motion models as explained in Section 3.

The questions we try to answer in this preliminary work are how effectively the client policy recovers the boundaries of the tracked object, using metadata and partial frame data, and how well our scheme fairs in comparison to standard JSIV [1, 2].

Not much research has been done in this area, but the closest related research to this work is the one by Kim *et al.* [4]. In their work they investigate the effectiveness of implicit segmentation of macroblocks in the context of H.264; given two candidate macroblock predictors, segmentation is applied to the difference of these two predictors. Then, each segment is predicted from a different weighted sum of these two predictors.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to JSIV and Section 3 gives a brief overview of metadata. Section 4 details the policies employed by the server and the client in order to recover the boundaries. Section 5 gives experimental results and comparisons against alternate approaches. Finally, Section 6 gives our conclusions and future work.

2. BRIEF INTRODUCTION TO JSIV

In JSIV, at the client, the samples of a given precinct in a given frame are either predicted from nearby frames or are decoded from received precinct data; received data always come from independently encoded frames (no residues are used in JSIV). Thus, the server in JSIV selects the optimal number of quality layers, q_n^π, for each precinct, $\mathcal{P}_n^\pi$, of each frame, f_n, within the group of frames, $\mathcal{G}_s$, be-

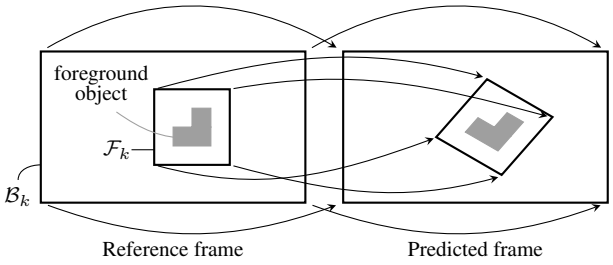

Fig. 1. The relation between quadrilaterals in reference and predicted frames.

ing optimized, and transmit the data associated with these layers to the client. The server optimization problem is cast as a dependent Lagrange-style rate-distortion optimization given by:

$$J_\lambda = \sum_{f_n \in \mathcal{G}_s} \sum_{\mathcal{P}_n^\pi \in f_n} (D_n^\pi + \lambda \cdot |q_n^\pi|) \quad (1)$$

where D_n^π is the distortion associated with $\mathcal{P}_n^\pi$, $|q_n^\pi|$ is the number of bytes in q_n^π layers, and λ is the Lagrangian parameter that is adjust until the target bit rate is achieved. This is a dependent optimization problem because some of these precincts are predicted from other precincts which themselves are being optimized. The reader is referred to [1, 2] for a detailed discussion of how to solve this optimization problem. Aided by a client policy, the client selects between either using prediction or decoding the received quality layers (possibly 0) for each precinct of each frame.

3. OVERVIEW OF METADATA

In this work, the metadata associated with a given object is a quadrilateral enclosing that object. For convenience of implementation, these quadrilateral are expressed as quadrilateral regions of interest in a way that is compatible with a proposed amendment to the JPEG2000 Part-2 standard [5]. In total, each quadrilateral requires around 60 bytes; this number of bytes can be significantly reduced by some compression algorithm since this representation was not designed to be minimal in size.

When the number of objects being tracked is not very high, we expect a compressed representation of metadata to compare favorably against conventional motion models because conventional models tend to spend a considerable amount of data in describing motion around the usually irregular boundaries of objects; something that made variable block sizes, introduced by Chan *et al.* [6], very popular in recent MPEGx and H.26x standards.

Besides identifying different objects, the quadrilaterals in this work carries motion information for these objects; the position and size of a given quadrilateral in a reference frame is chosen such that its mapping to the corresponding quadrilateral in the target frame using an affine model produces a good prediction for the object inside the corresponding quadrilateral in the predicted frame, as depicted in Figure 1.

4. PREDICTION WITH METADATA

To focus on the basic concepts in this preliminary work, we choose a video sequence that has only a single object to track; therefore, each frame can be decomposed into a single background region and a single foreground region. However, the approach described here can potentially be extended to any number of potentially overlapping regions.

We associate a *foreground mask*, $\nu_k[\mathbf{n}]$, with each point $\mathbf{n}$ of each frame f_k such that $\nu_k[\mathbf{n}] = 1$ when $f_k[\mathbf{n}]$ belongs to the foreground *object* and $\nu_k[\mathbf{n}] = 0$ when $f_k[\mathbf{n}]$ belongs to the background. However, since it is not generally possible to have absolute knowledge of the foreground/background status of each sample, we allow $\nu_k[\mathbf{n}]$ to take on any value in the range of 0 to 1, depending on our confidence in the designation at location $\mathbf{n}$.

We enclose the foreground object by a metadata quadrilateral, denoting the region enclosed by this quadrilateral by $\mathcal{F}_k$. Similarly, we enclose the background region, which usually covers the whole frame, by another metadata quadrilateral, denoting this region by $\mathcal{B}_k$; both of these regions are shown in Figure 1. We express the coordinates of the foreground and background quadrilaterals on a high-resolution JPEG2000-compliant code-stream canvas (higher than the resolution of the actual image); this provides for sub-integer accuracy in specifying these coordinates.

Under these conditions, $\nu_k[\mathbf{n}] = 0$ for all $\mathbf{n} \notin \mathcal{F}_k$ while for $\mathbf{n} \in \mathcal{F}_k$, $\nu_k[\mathbf{n}]$ range from 0 to 1; that is, not all the locations in $\mathcal{F}_k$ belong to the foreground object. The determination of the foreground mask, $\nu_k[\mathbf{n}]$, is an important aspect of both the client reconstruction policy and the server optimization policy; we discuss the exact method employed in finding $\nu_k[\mathbf{n}]$ in Subsection 4.2.

4.1. Client Policy

In this work, the server assumes that the client is capable of following a hierarchical B-frame prediction arrangement similar to that of H.264 [7]. The client, however, is free to follow whichever prediction arrangement it wishes, but if the server optimizes and sends data assuming hierarchical arrangement, an intelligent client should infer that better quality is achieved if it also follows a similar arrangement. It is important to note that the exact prediction arrangement is not and need not be communicated between the server and the client.

In the hierarchical prediction arrangement, frames are arranged in a dyadic structure with temporal decimation levels $T_0, T_l, \ldots T_L$, where T_L is the highest temporal level (slowest frame rate). Assuming that frames are sequentially numbered at any given temporal level, a frame belongs to the T_l^{th} temporal level, denoted by f_k^l, if its position is such that $f_k^l = f_{2\cdot k}^{l-1}$. In the hierarchical arrangement, f_k^L are not predicted from any other frame while each frame, f_{2k+1}^l, at other temporal levels can be predicted partially[1] or completely from $f_{2k}^l = f_k^{l+1}$ and $f_{2k+2}^l = f_{k+1}^{l+1}$.

In this work, the client receives the foreground and background quadrilaterals for each frame it needs to reconstruct. Moreover, the client also receives frame data, a number of quality layers (possibly 0) for each precinct of each frame. The client reconstructs the frames at the T_L^{th} temporal level directly from received data while employing prediction to help in reconstructing frames in other temporal level, as explained in the following subsections.

The prediction strategy employed in this work is different from that employed in conventional JSIV[2] [1–3]. In this work, the client maps each region in the reference frame to the target frame using an affine transform; the transform parameters are obtained from the quadrilaterals associated with the region being mapped. For example, the affine transform operator, $\mathcal{W}_{k\to j}^{\mathcal{B}}$, that warps the background region, $\mathcal{B}_k$, into the coordinated of $\mathcal{B}_j$ is obtained from the coordinates of the quadrilaterals associated with these regions, $\mathcal{W}_{k\to j}^{\mathcal{B}} = \mathcal{W}(\mathcal{B}_k, \mathcal{B}_j)$; in the current implementation, $\mathcal{W}(\mathcal{B}_k, \mathcal{B}_j)$ is obtained

[1] The JSIV server supplies precincts from poorly predicted regions of the target frame, employing Lagrange-style rate-distortion optimization.

[2] In our earlier work, a prediction for the target frame is obtained by motion compensating the left and right reference frames using a translational motion model and then averaging the resulting predictors.

by dividing $\mathcal{B}_k$ and $\mathcal{B}_j$ into triangles and using each pair of corresponding triangles to define an affine warping operator. Although not considered here, it is possible to use other more complicated motion models.

Prediction is performed on reconstructed frames; we choose not to introduce a new notation here, but rather think of f_k^l as reconstructed frames. To predict a frame f_{2k+1}^l from $f_{2k}^l = f_k^{l+1}$ and $f_{2k+2}^l = f_{k+1}^{l+1}$, the client first warps each of the reference source frames and their foreground masks to the coordinate system of f_{2k+1}^l, using each of the foreground and background warping affine operators. The resultant mapped images and masks are denoted by: $f_{2k\to 2k+1}^{l,\lambda}$, $f_{2k+2\to 2k+1}^{l,\lambda}$, $\nu_{2k\to 2k+1}^{l,\lambda}$ and $\nu_{2k+2\to 2k+1}^{l,\lambda}$, where $\lambda = \mathcal{B}$ if the background warping operator is used and $\lambda = \mathcal{F}$ if the foreground warping operator is used.

The basic idea behind the prediction algorithm is that foreground locations should be estimated from the average of $f_{2k\to 2k+1}^{l,\mathcal{F}}$, $f_{2k+2\to 2k+1}^{l,\mathcal{F}}$, while background locations should be estimated either as the average of $f_{2k\to 2k+1}^{l,\mathcal{B}}$, $f_{2k+2\to 2k+1}^{l,\mathcal{B}}$, or as one of $f_{2k\to 2k+1}^{l,\mathcal{B}}$ or $f_{2k+2\to 2k+1}^{l,\mathcal{B}}$, depending on the visibility of the background location within each of the prediction source frames, as determined by the warped mask foreground values. Thus, the estimate for f_{2k+1}^l, denoted here by $f_{\to 2k+1}^l[\mathbf{n}]$, is

$$\begin{aligned} f_{\to 2k+1}^l[\mathbf{n}] &= \frac{1}{2}\nu_{2k\to 2k+1}^{l,\mathcal{F}} \cdot f_{2k\to 2k+1}^{l,\mathcal{F}} \\ &+ \frac{1}{2}\nu_{2k+2\to 2k+1}^{l,\mathcal{F}} \cdot f_{2k+2\to 2k+1}^{l,\mathcal{F}} \\ &+ \left(1 - \frac{1}{2}\nu_{2k\to 2k+1}^{l,\mathcal{F}} - \frac{1}{2}\nu_{2k+2\to 2k+1}^{l,\mathcal{F}}\right) \cdot b_{\to 2k+1}^l \end{aligned} \tag{2}$$

where the background estimate is

$$\begin{aligned} b_{\to 2k+1}^l &= \frac{\left(1 + \delta - \nu_{2k\to 2k+1}^{l,\mathcal{B}}\right)}{2 + 2\delta - \nu_{2k\to 2k+1}^{l,\mathcal{B}} - \nu_{2k+2\to 2k+1}^{l,\mathcal{B}}} f_{2k\to 2k+1}^{l,\mathcal{B}} \\ &+ \frac{\left(1 + \delta - \nu_{2k+2\to 2k+1}^{l,\mathcal{B}}\right)}{2 + 2\delta - \nu_{2k\to 2k+1}^{l,\mathcal{B}} - \nu_{2k+2\to 2k+1}^{l,\mathcal{B}}} f_{2k+2\to 2k+1}^{l,\mathcal{B}} \end{aligned} \tag{3}$$

In the above, all computations are performed point-wise at each location $\mathbf{n}$, and δ is a small positive value (the actual value used is $\delta \approx 0.004$) whose purpose is only to ensure that the background estimate is well-defined everywhere. In a practical implementation, we also need to be careful to clip the warped mask values to the range 0 to 1 if the possibility exists for numerical expansion of the dynamic range during mask warping.

4.2. Determination of Foreground Mask

To determine the foreground mask, the client first warps frames f_{k-1}^l and f_{k+1}^l onto the coordinate system of f_k^l using each of the foreground and background warping operators, producing $f_{k-1\to k}^{l,\lambda}$ and $f_{k+1\to k}^{l,\lambda}$, where $\lambda = \mathcal{B}$ if the background warping operator is used and $\lambda = \mathcal{F}$ when foreground warping operator is used, exactly as before.

The basic idea behind the mask estimation algorithm is that a foreground location should be one in which the average of $f_{k-1\to k}^{l,\mathcal{F}}$ and $f_{k+1\to k}^{l,\mathcal{F}}$ is a better estimate for f_k^l than either $f_{k-1\to k}^{l,\mathcal{B}}$ or $f_{k-1\to k}^{l,\mathcal{B}}$. To this end, we define background and foreground squared prediction errors by

$$\begin{aligned} D_{\mathcal{B}}[\mathbf{n}] &= \min\left\{\left(f_k^l[\mathbf{n}] - f_{k-1\to k}^{l,\mathcal{B}}[\mathbf{n}]\right)^2, \left(f_k^l[\mathbf{n}] - f_{k+1\to k}^{l,\mathcal{B}}[\mathbf{n}]\right)^2\right\} \\ D_{\mathcal{F}}[\mathbf{n}] &= \left(f_k^l[\mathbf{n}] - \frac{1}{2}f_{k-1\to k}^{l,\mathcal{F}}[\mathbf{n}] - \frac{1}{2}f_{k+1\to k}^{l,\mathcal{F}}[\mathbf{n}]\right)^2 \end{aligned} \tag{4}$$

In our current implementation, we determine a preliminary estimate for $\nu_k^l[\mathbf{n}]$ using

$$\nu_k^l[\mathbf{n}] = \begin{cases} 1 & D_{\mathcal{F}}[\mathbf{n}] < D_{\mathcal{B}}[\mathbf{n}] \\ 0 & D_{\mathcal{B}}[\mathbf{n}] \le D_{\mathcal{F}}[\mathbf{n}] \end{cases} \tag{5}$$

This preliminary estimate is then subjected to a 5×5 uniform low-pass filter (moving average) to reduce sensitivity to noise. Of course, this estimate need only be formed within the foreground region $\mathcal{F}_k^l$, since $\nu_k^l[\mathbf{n}]$ is known to be 0 outside that region. Although this strategy produces usable results, future research will focus on measures of similarity that are better matched to the multi-resolution nature of the source data, producing better localization of foreground/background regions where more high frequency sub-band data is available.

The mask estimation process described here produces different results in the server from those in the client, as the server has access to the original frames while the client does not. In the client, masks are estimated hierarchically, starting from the highest level in the temporal hierarchy and working toward the lowest. At the highest level, T_L, each mask ν_k^L is estimated using the available reconstructed frames. The server optimization policy can generally be expected to assign the highest quality to frames at this level of the hierarchy. For each successively lower level T_l in the temporal hierarchy, we first form a predictor, $f_{\to 2k+1}^l$, based on the frame samples and masks from level T_{l+1}. The JSIV client policy then forms reconstructed frames, f_{2k+1}^l, by decomposing $f_{\to 2k+1}^l$ into its wavelet sub-bands, substituting explicitly received code-blocks for frame f_{2k+1}^l to the extent that these are expected to lower distortion, and then synthesizing the reconstructed frame from its sub-band samples. Once this is done, mask estimation can proceed for level l. In the event that no code-block data is available for frame f_{2k+1}^l, $f_{2k+1}^l = f_{\to 2k+1}^l$ depends only on frames f_k^{l+1} and f_{k+1}^{l+1} and their respective masks; in this case, the mask estimation algorithm should yield a similar mask for ν_{2k+1}^l to those found in the surrounding frames ν_k^{l+1} and ν_{k+1}^{l+1}, subject to the appropriate geometric warping. This is because foreground regions in the prediction source frames f_k^{l+1} and f_{k+1}^{l+1} can be expected to generate compatible content in the predicted frame $f_{\to 2k+1}^l$.

4.3. Server Policy

Ideally, a JSIV server estimates the distortion associated with a given frame (predicted or otherwise) without actually reconstructing it; however, in this preliminary work, the server reconstruct the frames to calculate the distortion associated with them. The server implements the same policies implemented by the client; unlike the client, the server has access to the original frames, which are used in calculating (4) to obtain better estimates for $\nu_k^l[\mathbf{n}]$ than those available to the client. Other details of the server are the same as in [1,2].

5. EXPERIMENTAL RESULTS

Two test sequences[3] are used in this work, "Ship" and "Book". These test sequences were shot in-house at 30 frames/s, and they

[3] "Ship" and "Book" test sequences are available at `http://www.eet.unsw.edu.au/~taubman/sequences.htm`.

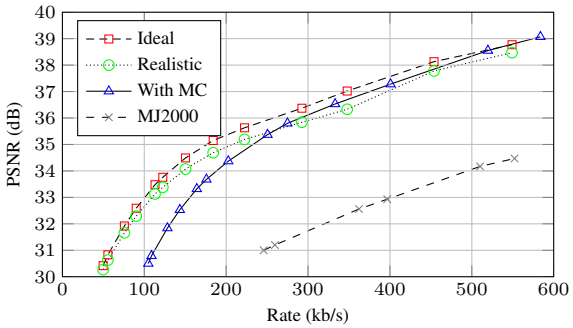

Fig. 2. A comparison of the performance of various schemes for the "Book" sequence.

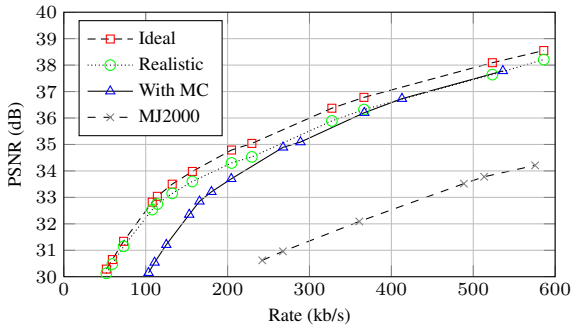

Fig. 3. A comparison of the performance of various schemes for the "Ship" sequence.

have a static background with only one swinging foreground object. Both sequences have 33 frames[4] with a resolution of 1024×768 and a bit depth of 8 bits per sample. Both sequences are converted to JPEG2000 using Kakadu[5]. Five levels of irreversible DWT are employed for both sequences with a code-block size of 32×32 and 20 quality layers. Motion is estimated at $\frac{1}{4}$-pixel accuracy, using 7-tap interpolation kernels formed by windowing cubic splines. Motion is estimated using the color version of these sequences, but only the Y-component is used for all the tests reported here.

Figure 2 shows the PSNR obtained from the average MSE for the reconstructed "Book" sequence while Figure 3 shows it for the "Ship" sequence. In these figures, "Ideal" refers to the client using a foreground mask obtained from full-quality frames, similar to that used in the server; this is unrealistic as the client does not have access to these frames, but it is useful in determining the upper bound on performance. "Realistic" refers to the realistic client policy in which that client uses a foreground mask obtained from reconstructed frames. In both of these cases the server uses the foreground mask obtained from full-quality frames. "With MC" refers to the case in which we employ conventional motion compensation using an advanced block-based embedded scalable motion model with geometry information [8]. "MJ2000" refers to optimizing each frame independently of the others (no prediction is employed).

Compared to "MJ2000", it can be seen that both "Ideal" and "Realistic" perform considerably better. Compared to "With MC", "Realistic" performs better at low data rates and comparably or slightly worst at higher rates while "Ideal' performs better at all the considered data rates. It is important to mention that the metadata motion used here is still quite inaccurate and does not follow the foreground as well as could be hoped, yet results are still favorable compared to full motion compensation.

6. CONCLUSION AND FUTURE WORK

We have presented a novel approach for communicating motion information. Each moving object in each frame is enclosed by a quadrilateral that identifies the object's crude boundaries and implicitly signals its motion information. These quadrilaterals do not follow the exact boundaries of the objects but rather leave it to the client to identify them, using received frame data and their associated quadrilateral metadata. The advantage of this representation is that it is global, allowing an object to be predicted from potentially many reference frames. Preliminary implementation of the proposed approach in the context of JSIV shows very promising results.

This work can be extended by considering the problem of more than one foreground object. Foreground object identification at the client can be improved by using more advance segmentation methods or by utilizing more than two frames. Server policy can be improved by employing mechanisms that can take into consideration the actual reconstruction policy employed at the client. The server can also be improved by employing approximate distortion estimation rather than actually reconstructing the target frames as is done in this work.

[4]The length of 33 is selected because it is suitable for a 3-level hierarchical B-frame prediction arrangement.

[5]`http://www.kakadusoftware.com/`, Kakadu software, version 5.2.4.

7. REFERENCES

[1] A.T. Naman and D. Taubman, "JPEG2000-based scalable interactive video (JSIV)," *Image Processing, IEEE Transactions on*, vol. 20, no. 5, pp. 1435 –1449, May 2011.

[2] A.T. Naman and D. Taubman, "JPEG2000-based scalable interactive video (JSIV) with motion compensation," *Image Processing, IEEE Transactions on*, available in early access.

[3] A.T. Naman, *JPEG2000-Based Scalable Interactive Video (JSIV)*, Ph.D. thesis, School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia, 2010.

[4] Jae Hoon Kim, A. Ortega, Peng Yin, P. Pandit, and C. Gomila, "Motion compensation based on implicit block segmentation," *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 2452 –2455, October 2008.

[5] ISO/IEC 15444-2:2004/DAM 3, "Extended ROI boxes, XML boxing, compressed channel definition boxes, representation of floating point data and box-based file format for JPEG-XR," Under development.

[6] M.H. Chan, Y.B. Yu, and A.G. Constantinides, "Variable size block matching motion compensation with applications to video coding," *Communications, Speech and Vision, IEE Proceedings I*, vol. 137, no. 4, pp. 205 –212, August 1990.

[7] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[8] R. Mathew and D.S. Taubman, "Scalable modeling of motion and boundary geometry with quad-tree node merging," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 2, pp. 178 –192, February 2011.