OPTIMAL SLICE SIZE FOR STREAMING REGIONS OF HIGH RESOLUTION VIDEO WITH VIRTUAL PAN/TILT/ZOOM FUNCTIONALITY

Aditya Mavlankar, Pierpaolo Baccichet, David Varodayan, and Bernd Girod

Max Planck Center for Visual Computing and Communication
Stanford University, Stanford, CA 94305, USA
phone: + (1) 650 723 3476, fax: + (1) 650 724 3648, email: {maditya,bacci,varodayan,bgirod}@stanford.edu

ABSTRACT

High resolution digital imaging sensors are becoming more widespread. The challenges in delivering this high resolution content to the client are posed by limited resolution of display panels and/or limited bit-rate for communications. We propose a video coding scheme which enables virtual pan/tilt/zoom functionality during the streaming session. This way the server can adapt and stream only those regions of the video content that are desired at that time at the client's end. Apart from generating a multi-resolution representation, our coding scheme uses P slices from H.264/AVC for random access to arbitrary regions within every spatial resolution. We study the trade-off in the choice of slice size. A larger slice size enables higher coding efficiency for representing the entire scene but increases the pixel overhead. The pixel overhead is due to superfluous pixels that are transmitted but not displayed at the client's end. The optimal slice size achieves the best trade-off and minimizes the expected number of bits transmitted to the client per frame. Our analysis helps to predict the optimal slice size, which depends on the signal as well as the display resolution. Experimental results confirm the optimality of the predicted slice size for various test cases.

1. INTRODUCTION

High-resolution digital video will be broadly available in the near future. This will be driven by the increasing spatial resolution offered by digital imaging sensors and the increasing capacities of storage devices. Also, algorithms for stitching a comprehensive high-resolution view from multiple cameras have been well studied. For example, this has been recently deployed by Hewlett-Packard in their video conferencing product called Halo [1].

Despite the availability of high-resolution video, there are challenges posed by the limited spatial resolution of display panels or limited data-rate availability for delivering the content to the client. Assume that a client limited by one of these factors is requesting the server to stream a high spatial resolution video. One approach would be to stream a spatially downsampled version of the entire video scene to suit the client's display window resolution. However, with this approach, the user at the client terminal might not be able to watch a local region-of-interest (ROI) in the highest captured resolution. To overcome this problem, we propose the following interactive features. The client indicates its ROI and the desired spatial resolution (zoom factor) real-time to the server. The server then reacts to this by sending relevant video data which will be decoded and displayed at the client's side. The server should be able to react to the client's changing ROI with as little latency as possible.

Some practical scenarios where this kind of interactivity is well-suited are: interactive playback of a high-resolution video from a locally stored file, interactive TV for watching content captured with very high detail, providing virtual pan/tilt/zoom within a wide-angle and high-resolution scene from a surveillance camera, and streaming instructional videos captured with high spatial resolution.

Interactive delivery of video from pre-stored bit-streams necessitates video coding schemes that allow for sufficient random access to arbitrary ROIs, while keeping the transmission data-rate and the file-size on the storage device as low as possible. Section 2 reviews related work and also discusses the challenges in providing random access to arbitrary regions as well as spatial resolutions. Section 3 discusses the proposed user interface and the video coding scheme. Section 4 shows how to select the optimal slice size knowing the signal, the display window dimensions and the desired set of zoom factors. The optimal slice size minimizes the transmission data-rate by striking the best compromise between storage space at the server and superfluous pixel transmission. Section 5 shows experimental results which confirm our prediction of optimal slice size for various test cases.

2. RELATED WORK

Taubman *et al.* proposed a solution for interactive browsing of images using JPEG2000 [2]. This leverages the multi-resolution representation of an image using wavelets. JPEG2000 encodes blocks of wavelet transform coefficients independently. This means that every coded block has influence on the reconstruction of a limited number of pixels of the image. Moreover, the coding of each block results in an independent, embedded sub-bitstream. This makes it possible to stream any given block with a desired degree of fidelity. Taubman *et al.* also developed a protocol for communication between client and server that supports interactive browsing of JPEG2000 coded images [3]. The server keeps track of the ROI trajectory of the client as well as the parts of the bitstream that have already been streamed to the client. Given a rate of transmission for the current time interval, the server then solves an optimization problem to determine which parts of the bit-stream need to be currently sent in order to maximize the quality of the current ROI. This is very similar to packet scheduling algorithms proposed in [4] for streaming of video. It should be noted, however, that an accurate model for the distortion reduction due to successful delivery of any particular packet is necessary.

Unlike image browsing, providing interactive features to video delivery presents its own challenges. To achieve good compression efficiency, video compression schemes must exploit the significant correlation between successive frames.

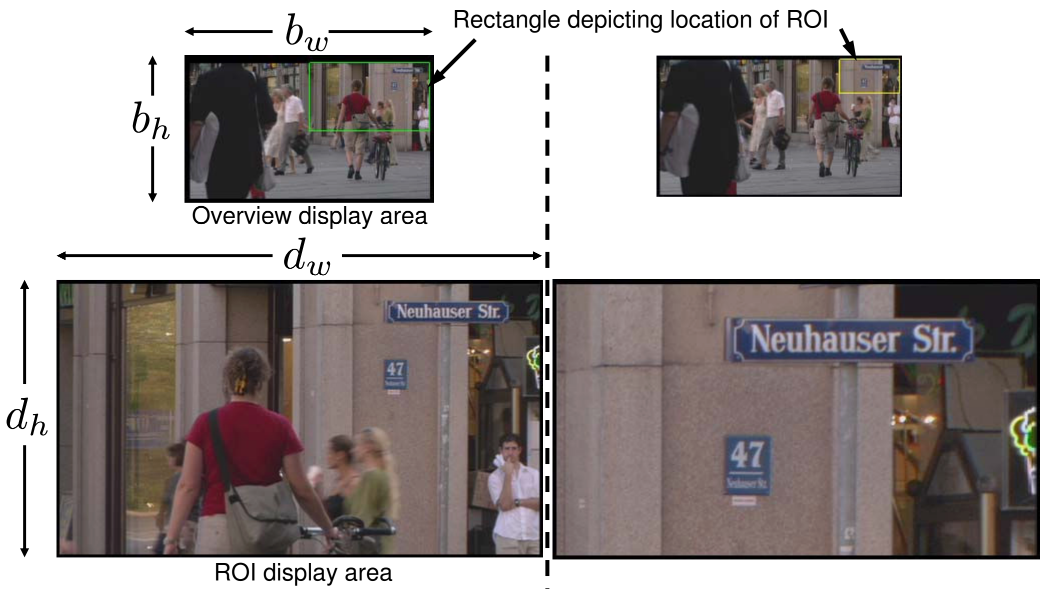

Figure 1: User interface: The display screen consists of the overview display area and the ROI display area. The effect of changing the zoom factor can be seen by comparing left and right hand sides of the figure.

This is typically accomplished through motion-compensated interframe prediction. However, this makes it difficult to provide random access, since the decoding of a particular block of pixels usually requires the decoding of several other blocks. The prior work discussed below deals with similar issues.

Coding, transmission and rendering of high-resolution panoramic videos using MPEG-4 is proposed in [5]. A limited part of the entire scene is transmitted to the client depending on his viewpoint. In this work, only intraframe coding is used to allow random access. The scene is coded into independent slices. The authors also mention the possibility of employing interframe coding to gain more compression efficiency. However, they comment that this involves transmitting slices from the past if the current slice requires those for its decoding. A longer intraframe period entails significant complexity for slices from the latter frames in the group of pictures, as this dependency chain grows.

Interactive streaming of light-fields has been studied by Ramanathan *et al.* in [6]. The above mentioned growing dependency chain is avoided by using multiple representations coding based on two new picture types defined in the H.264/AVC standard, viz. SP and SI picture types proposed by Karczewicz *et al.* in [7]. Ramanathan *et al.* also extend rate-distortion optimized packet scheduling, based on the framework in [4], to multiple representations coding for lightfields. However, in their setup, only entire pictures from the lightfield data-set are streamed and there is no provision of spatial region random access within a picture.

3. USER INTERFACE AND VIDEO CODING SCHEME

3.1 User Interface

We have developed a user interface with real-time interaction for ROI selection while watching the video sequence. This has been developed using OpenGL [8]. As shown in Fig. 1, the display screen at the client's side consists of two areas:

- The first area displays a downsampled version of the entire scene. We call this the overview display area. It is b_w pixels wide and b_h pixels tall.
- The second area displays the client's ROI. We call this the ROI display area. It is d_w pixels wide and d_h pixels tall.

Let the original video be o_w pixels wide and o_h pixels tall. For simplicity, we consider the dyadic case where every zoom-out operation corresponds to downsampling by 2 both horizontally and vertically, i.e., corresponding to N zoom factors, the entire original scene is available in dimensions $(o_{w,i} = 2^{-(N-i)}o_w$ by $o_{h,i} = 2^{-(N-i)}o_h)$ for $i = 1 \ldots N$. The zoom factor can be controlled with the scroll of the mouse. For any zoom factor, the ROI can be moved around by keeping the left mouse-button pressed and moving the mouse. Depending on the display area dimensions and the dimensions of the original video, the entire scene might fit in the ROI display area for the lowest zoom factor and in this case no translation of the ROI is possible with the left mouse-button. As shown in Fig. 1, the location of the ROI is depicted in the overview display area by overlaying a corresponding rectangle on the video. The color and size of the rectangle vary according to the zoom factor.

3.2 Video Coding Scheme

Ideally, the video coding scheme for this kind of an application should not only provide spatial resolution random access to each of the resolutions $(o_{w,i}$ by $o_{h,i})$ but also random access to arbitrary regions within each resolution. It should be possible to cater to zoom factor change as well as ROI translation for any frame of the video.

As shown in Fig. 2, in the proposed scheme, we first encode the overview video $(b_w$ by $b_h)$ using hierarchical B pictures, since this gives good compression efficiency and also no spatial random access is required within the overview display area. The reconstructed overview video frames are upsampled by a factor of $2^{(i-1)}g$ horizontally and vertically and

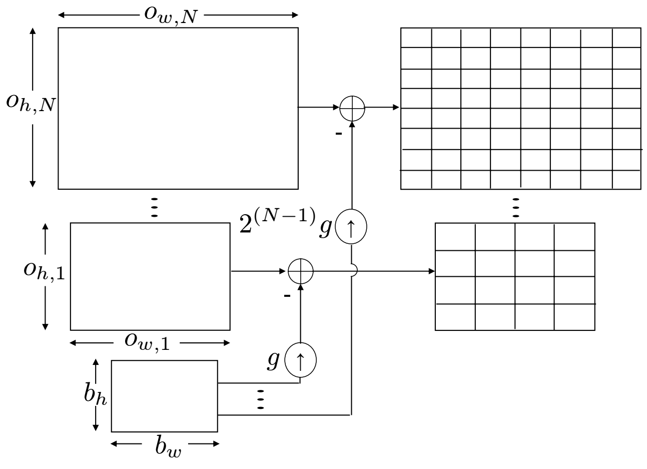

Figure 2: Coding Scheme: The base layer is coded using hierarchical B pictures. The video signal corresponding to every zoom factor is coded using P slices. For doing so, the signal reconstructed from the base layer is upsampled by an appropriate factor and used as prediction. (Note: Resolutions corresponding to successive zoom factors increase dyadically. The figure is not drawn to scale and hence does not reflect this.)

used as prediction signal for encoding video of dimensions ($o_{w,i}$ by $o_{h,i}$), where $i = 1 \dots N$ and $g = \frac{o_{h,1}}{b_h} = \frac{o_{w,1}}{b_w}$. Furthermore, every frame of dimensions ($o_{w,i}$ by $o_{h,i}$) is coded into independent P slices. This is depicted in Fig. 2, by overlaying a grid on the residual frames. This allows spatial random access to local regions within any spatial resolution. For every frame interval, the request of the client can be catered by the corresponding frame from the overview video and few P slices from exactly one resolution layer.

Notice that when $b_h = o_{h,1} = d_h$ and $b_w = o_{w,1} = d_w$, then multiple slices are not required for zoom factor = 1, since the two areas of the display would then show the same pixels.

4. OPTIMAL SLICE SIZE SELECTION

Notice that for the given coding scheme, the slice size for every resolution can be independently optimized given the residual signal for that zoom factor. Thus, the strategy proposed here can be independently used for all zoom factors $i = 1 \dots N$. Given any zoom factor, we assume that the slices form a regular rectangular grid, so that every slice is s_w pixels wide and s_h pixels tall. The slices on the boundaries can have smaller dimensions due to the picture dimensions not being integer multiples of the slice dimensions.

The number of bits transmitted to the client depends on the slice size as well as the user's ROI trajectory over the streaming session. Furthermore, the quality of the decoded video depends on the Quantization Parameter (QP) used for encoding the slices. Nevertheless, it should be noted that for the same QP, almost the same quality is obtained for different slice sizes, even though the number of bits are different. Hence, given the QP, our goal is to choose the slice size in order to minimize the expected number of bits per frame transmitted to the client.

Decreasing the slice size has two contradictory effects on the expected number of bits transmitted to the client. On one hand, the smaller the slice size the worse is the coding efficiency. This is because of increased number of slice headers, lack of context continuation across slices for context adaptive coding and inability to exploit any inter-pixel correlation across slices. On the other hand, a smaller slice size entails lower pixel overhead for any ROI trajectory. The pixel overhead consists of pixels which have to be streamed because of the coarse slice division, but which are not finally displayed at the client. For example, the shaded pixels in Fig. 3 show the pixel overhead for the shown slice grid and location of the ROI.

In the following analysis, we assume that the ROI location can be changed with a granularity of one pixel both horizontally and vertically. Also every location is equally likely to be selected. Depending on the application scenario, the slices might be put in different transport layer packets. The packetization overhead of layers below the application layer, for example RTP/UDP/IP, has not been taken into account but can be easily incorporated into the proposed optimization framework.

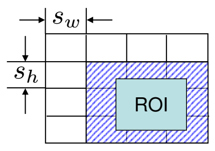

Figure 3: Depending on the ROI display dimensions and the slice size and the location of the ROI with respect to the slice grid, there is an overhead of pixels that are transmitted but not displayed on the client's screen. The shaded portion depicts the pixel overhead in this example.

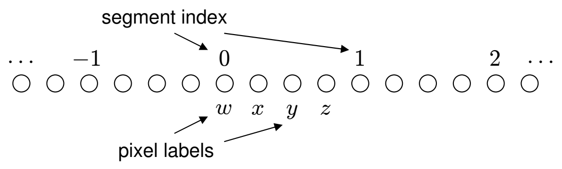

Figure 4: The infinitely long line of pixels is divided into segments. In this example, the length of each segment is $s = 4$. The length of the display segment is $d = 3$.

4.1 Pixel Overhead

To simplify the analysis, we first consider the 1-D case and then extend it to 2-D.

4.1.1 *Analysis of Overhead in 1-D*

Imagine an infinitely long line of pixels. This line is divided into segments of length s. For example, in Fig. 4, $s = 4$. Also given is the length of the display segment d. Assume $d = 3$ in this example. In order to calculate the pixel overhead, we are interested in the probability distribution of the number of segments that need to be transmitted. This can be obtained by testing for locations within one segment, since the pattern repeats every segment. For locations w and x, we would need to transmit a single segment, whereas for locations y and z, we would need to transmit 2 segments. Let N be the random variable representing the number of segments to be transmitted. Given s and d, we can uniquely choose $m, d^* \in \mathbb{N}$ such that $m \geq 0$ and $1 \leq d^* \leq s$ and also the following relationship holds

$$d = ms + d^*.$$

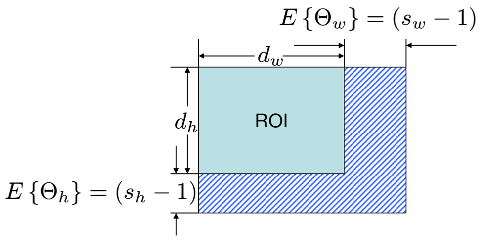

Figure 5: The overhead is determined by the expected number of superfluous rows and the expected number of superfluous columns that need to be transmitted due to slice dimensions (s_w by s_h).

By inspection, the p.m.f. of random variable N is given by

$$\Pr\{N = m+1\} = \frac{s-(d^*-1)}{s}, \quad \Pr\{N = m+2\} = \frac{d^*-1}{s}$$

and zero everywhere else.

Theorem: Given that $d, s \in \mathbb{N}$, the expected pixel overhead

- *increases monotonically with s and*
- *is independent of d.*

Proof: From the p.m.f. of N,

$$\begin{aligned} E\{N\} &= (m+1)\frac{s-(d^*-1)}{s} + (m+2)\frac{d^*-1}{s} \\ &= (m+1) + \frac{d^*-1}{s} \end{aligned}$$

Let P be the random variable which denotes the number of pixels that need to be transmitted and Θ be the random variable which denotes the pixel overhead in 1-D.

$$\begin{aligned} E\{P\} &= s \times E\{N\} \\ &= (m+1)s + d^* - 1 \\ &= d + s - 1 \\ E\{\Theta\} &= E\{P\} - d \\ &= s - 1 \end{aligned} \quad (1)$$

Surprisingly, the expected overhead in 1-D is $s-1$. It increases monotonically with s and is independent of the display segment length d.

4.1.2 Analysis of Overhead in 2-D

We define two new random variables, viz., Θ_w, the number of superfluous columns and Θ_h, the number of superfluous rows that need to be transmitted. Θ_w and Θ_h are independent random variables. From the analysis in 1-D we know that

$$E\{\Theta_w\} = s_w - 1, \quad E\{\Theta_h\} = s_h - 1. \quad (2)$$

Figure 5 depicts the situation by juxtaposing the expected column and row overheads next to the ROI display area (d_w by d_h). The expected value of the pixel overhead is then given by

$$E\{\Theta\} = (s_w-1)(s_h-1) + d_h(s_w-1) + d_w(s_h-1) \quad (3)$$

and it depends on the display area dimensions. Let random variable P denote the total number of pixels that need to be transmitted per frame for the ROI part. The expected value of P is then given by

$$E\{P\} = (d_w + s_w - 1)(d_h + s_h - 1) \quad (4)$$

4.2 Coding Efficiency

For any given resolution layer, if the slice size is decreased then more bits are needed to represent the entire scene for the same QP. We can vary the slice size (s_w, s_h) and see the effect on η, the bit per pixel for coding the entire scene. In the following, we write η as a function of (s_w, s_h).

Finally, the optimal slice size can be obtained by minimizing the expected number of bits transmitted per frame.

$$\begin{aligned} (s_w, s_h) &= \arg\min_{(s_w,s_h)} \eta(s_w,s_h) \times E\{P\} \quad (5) \\ &= \arg\min_{(s_w,s_h)} \eta(s_w,s_h) \times (d_w + s_w - 1)(d_h + s_h - 1) \end{aligned}$$

4.3 Simplification of Search

We can model the variation of η as a function of (s_w, s_h) by fitting a parametric model to some sample points. For example,

$$\eta(s_w, s_h) = \eta_0 - \gamma s_w - \phi s_h - \lambda s_w s_h$$

is one such model with parameters η_0, γ, ϕ and λ. This is, however, not required if we can narrow down our search to a few candidate pairs (s_w, s_h). In this case η can be obtained for those pairs from some sample encodings.

In practice, the slice dimensions have to be multiples of the macroblock width. Also slice dimensions in a certain range can be ruled out because they are very likely to be suboptimal, e.g., s_h greater than or comparable to d_h is likely to incur a huge pixel overhead. Consider a case where for some resolution layer, $o_{h,i} = d_h$, i.e., the ROI can have only horizontal translation and no vertical translation. Clearly, in this case the best choice for s_h is $s_h = o_{h,i} = d_h$. Constraints like these help us to practically narrow down our search. Knowing $\eta(s_w, s_h)$, the optimal slice size can be obtained using equation 5 without actually observing the bits transmitted per frame over a set of sample ROI trajectories.

5. EXPERIMENTAL RESULTS

We use two 1920x1080 MPEG test sequences, *Pedestrian Area* and *Tractor*, and convert the resolution to 1920x1088 pixels by padding extra rows. The third is a panoramic video sequence[1] called *Making Sense* of resolution 3584x512. For *Making Sense* the ROI is allowed to wrap around while translating horizontally, since the panorama covers a full 360 degree view. For the first two sequences, we have 3 zoom factors, viz., $(o_{w,1} = 480 \times o_{h,1} = 272)$, $(o_{w,2} = 960 \times o_{h,2} = 544)$ and $(o_{w,3} = 1920 \times o_{h,3} = 1088)$. The display dimensions are $(b_w = 480 \times b_h = 272)$ and $(d_w = 480 \times d_h = 272)$. Notice that we do not need multiple slices for zoom factor of 1. For the panoramic video, we have 2 zoom factors, viz., $(o_{w,1} = 1792 \times o_{h,1} = 256)$ and $(o_{w,2} = 3584 \times o_{h,2} = 512)$. The display dimensions are $(b_w = 896 \times b_h = 128)$ and $(d_w = 480 \times d_h = 256)$. The overview area shows the entire panorama. Notice that for zoom factor of 1, $s_h = 256$ is the best choice because the ROI cannot translate vertically for this zoom factor.

The overview video, also called as base layer, is encoded using hierarchical B pictures of H.264/AVC. The PSNR @ bit-rate for *Pedestrian Area*, *Tractor* and *Making Sense* are 32.84 dB @ 188 kbps, 30.61 dB @ 265 kbps and 33.24 dB @ 112 kbps respectively. For encoding the residuals at all

[1]We thank Joe Rosen, Dirk Farin and the Stanford Center for Innovations and Learning (SCIL) for providing the panoramic video sequence.

Table 1: Kilobits per frame, J, transmitted for ROI display for the sequences *Pedestrian Area* (left) and *Making Sense* (right). $J_1(s_w,s_h)$ is the prediction using equation 5. $J_2(s_w,s_h)$ is the average of the observations over 5 sample ROI trajectories. Additionally shown is $f(s_w,s_h)$, the percentage increase in number of bits stored at the server for the residual of the respective zoom factor with respect to coding the entire residual in one slice. (Note that $\eta(s_w,s_h)$ is related to $f(s_w,s_h)$ as follows: $\eta(s_w,s_h) = \eta(o_{w,i},o_{h,i})\left[1+\frac{f(s_w,s_h)}{100}\right]$, where $f(o_{w,i},o_{h,i}) = 0$.)

Resolution ($o_{w,i} \times o_{h,i}$)	Slice size $s_w \times s_h$	$J_1(s_w,s_h)$ kbit/frame	$J_2(s_w,s_h)$ kbit/frame	$f(s_w,s_h)$ %	Resolution ($o_{w,i} \times o_{h,i}$)	Slice size $s_w \times s_h$	$J_1(s_w,s_h)$ kbit/frame	$J_2(s_w,s_h)$ kbit/frame	$f(s_w,s_h)$ %
960x544 (Zoom factor 2)	160x160	76.6	70.2	4	1792x256 (Zoom factor 1)	256x256	70.6	74.9	1
	128x128	69.0	62.7	7		128x256	58.8	62.8	2
	64x64	57.3	53.0	18		64x256	53.6	57.6	4
	32x32	63.1	59.6	53		32x256	52.0	56.0	7
1920x1088 (Zoom factor 3)	160x160	50.4	45.2	8	3584x512 (Zoom factor 2)	256x256	91.5	95.7	3
	128x128	45.9	40.6	12		128x128	59.1	67.7	9
	64x64	41.2	37.6	34		64x64	49.8	61.5	25
	32x32	52.0	49.2	99		32x32	57.2	70.8	70

zoom factors we choose QP=28. This gives high quality of reconstruction for all zoom factors; roughly 40 dB for both *Pedestrian Area* and *Tractor* and roughly 39 dB for *Making Sense*.

For every zoom factor, we encode the residual using up to 8 different slice sizes and calculate $\eta(s_w,s_h)$ for every slice size. The optimal slice size is then predicted by evaluating equation 5. For *Pedestrian Area*, the optimal slice size, (s_w,s_h), is (64x64) for both zoom factor of 2 and zoom factor of 3. For *Tractor* zoom factor of 2, the cost function is very close for slice sizes (64x64) and (32x32). For *Tractor* zoom factor of 3, the optimal slice size is (64x64). For *Making Sense*, the optimal slice sizes are (32x256) and (64x64) for zoom factor of 1 and zoom factor of 2 respectively.

To confirm the predictions from the model, we use the user interface that we have developed and record 5 ROI trajectories within every resolution layer and add the bits used for encoding the relevant slices that need to be transmitted according to the trajectories. We find that the predictions using equation 5 are correct. This is shown in Table 1 for the two sequences *Pedestrian Area* and *Making Sense*.

As an aside, we encoded the sequence *Pedestrian Area* directly in resolution 1920x1088 using the same hierarchical B pictures coding structure that we used for the base layer in the above experiments. To achieve similar quality for the interactive ROI display as with the random access enabled bitstreams above we need a transmission bit-rate which is roughly 2.5 times.

6. CONCLUSIONS

We present a video coding scheme for streaming regions of high resolution video with virtual pan/tilt/zoom functionality. Our scheme generates a coded representation which allows random access to a set of spatial resolutions and also arbitrary regions within every resolution. Note that this coded representation is pre-stored at the server and obviates the necessity for real-time compression.

The slice size directly influences the expected number of bits transmitted per frame. The slice size has to be optimized in accordance with the signal and the ROI display area dimensions. We analyze theoretically the trade-offs involved in reducing the slice size. Our analysis dramatically simplifies the procedure to choose the optimal slice size. The experimental results with some representative ROI trajectories confirm the correctness of the slice size thus obtained. In this paper, the optimization of the slice size is carried out given the reconstructed base layer signal. However, a joint optimization of coding parameters and QPs for the base layer *and* the residuals of the different zoom factors would reduce the overall transmitted bit-rate further.

We plan to design the proposed application such that it runs over a realistic network. It should be noted that in this scenario, the ROI requests on the back channel could also be lost. A bigger slice size will add robustness and help to render the desired ROI at the client in spite of this loss on the back channel. Also, if the packetization overhead of lower layers is considered when each slice needs to be put in a different transport layer packet, then a bigger slice size is more likely to be optimal. A sample scenario is application layer multicasting to a plurality of peers/clients where each client can subscribe/unsubscribe to requisite slices according to its ROI. This is part of future research.

REFERENCES

[1] Halo: Video Conferencing Product by Hewlett-Packard. Web: http://www.hp.com/halo/index.html.

[2] D. Taubman and R. Rosenbaum, “Rate-Distortion Optimized Interactive Browsing of JPEG2000 Images,” in *Proc. IEEE International Conference on Image Processing (ICIP)*, Barcelona, Spain, vol. 3, pp. 765-768, Sept. 2003.

[3] D. Taubman and R. Prandolini, “Architecture, Philosophy and Performance of JPIP: Internet Protocol Standard for JPEG2000,” in *Proc. International Symposium on Visual Communications and Image Processing (VCIP)*, Lugano, Switzerland, SPIE vol. 5150, pp. 649-663, Jul. 2003.

[4] P. Chou and Z. Miao, “Rate-Distortion Optimized Streaming of Packetized Media,” *IEEE Trans. Multimedia*, vol. 8, No. 2, Apr. 2006.

[5] S. Heymann, A. Smolic, K. Mueller, Y. Guo, J. Rurainsky, P. Eisert, and T. Wiegand, “Representation, Coding and Interactive Rendering of High-Resolution Panoramic Images and Video using MPEG-4,” in *Proc. Panoramic Photogrammetry Workshop (PPW)*, Berlin, Germany, Feb. 2005.

[6] P. Ramanathan and B. Girod, “Rate-Distortion Optimized Streaming of Compressed Light Fields with Multiple Representations,” in *Proc. Packet Video (PV)*, Irvine, CA, USA, Dec. 2004.

[7] M. Karczewicz and R. Kurceren, “The SP- and SI-Frames Design for H.264/AVC,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, No. 7, Jul. 2003.

[8] OpenGL Application Programming Interface. Web: http://www.opengl.org.