Adaptive Encoding of Zoomable Video Streams based on User Access Pattern

Ngo Quang Minh Khiem, Guntur Ravindra, Wei Tsang Ooi
Department of Computer Science
National University of Singapore
Singapore 117417
E-mail:{nqmkhiem,ravindra,ooiwt}@comp.nus.edu.sg

ABSTRACT

Zoomable video allows users to selectively zoom and pan into regions of interest within the video for viewing at higher resolutions. Such interaction requires dynamic cropping of RoIs on the source video. In this paper, we consider how the bandwidth needed to transmit the RoIs can be reduced by carefully encoding the source video. The key idea is to exploit user access patterns to the RoIs, and encode different regions of the video with different encoding parameters based on the popularity of the region. We show that our encoding method can reduce the expected bandwidth by up to 27%.

Categories and Subject Descriptors: H.5.1 [Multimedia Information Systems]: Video H.4.3 [Communications Applications]:Video Streaming

General Terms: Algorithms, Design, Experimentation, Performance

Keywords: Zoomable Video, Region-of-Interest Streaming, Monolithic Streaming, Tile Streaming, Optimal Tiling, Bandwidth Efficient, Encoding

1. INTRODUCTION

Two opposing trends in digital video capture and display have emerged. On one hand, consumers are increasingly capable of capturing high resolution digital video using off-the-shelf cameras. On the other hand, it is increasingly common for consumers to play back digital video on portable devices with limited screen size and resolution. Such impedance mismatch between captured and displayed video resolutions leads to introduction of zoom and pan as two new interaction primitives during video playback.

We are interested in enabling zoom and pan interaction in a video streaming setting, a paradigm we call *zoomable video streaming*. A video server stores multiple high resolution videos. Remote clients can request for these videos to be transmitted for playback at a smaller resolution. A lower resolution version of the video is streamed to the client for playback. The client may zoom into a region of interest (RoI) in a video, i.e., to view a cropped region in the video at a high resolution. The RoI coordinates are sent to the server. The server then sends the RoI at a higher resolution to the client for playback. The client may also pan around the video, i.e., to view a different region but at the same resolution. The server in this case crops a different RoI and streams the RoI to the client.

There are several ways the server can support dynamic cropping of RoIs at different resolutions. The server can pre-compute different versions of the same video at different resolutions and switch between these different versions when different zoom levels are requested. This approach is known as *bitstream switching*. Alternatively, scalable video coding can be used to encode the video at different resolutions.

The key challenge for zoomable video streaming is dynamic cropping of RoI, which can be implemented in a few ways. A naive way is to pre-encode each possible RoI as an independent video stream and serve the requested RoI to the client when requested. Such approach results in huge storage requirements. One could reduce the pre-encoded RoIs to only the popular ones, but this reduces the flexibilities of user interactions. The other naive method is to encode the RoI on the fly. The video server crops the RoI from the video with appropriate resolution, encodes the RoI into a video stream, and transmits. Such implementation, while is flexible enough to support any RoI, is not scalable to large number of clients.

We have previously [16] proposed two scalable schemes in which the server only encodes the video once, but can flexibly support multiple clients with any RoI. Our first scheme is called *tiled streaming*. Tiled streaming divides each video frame into a grid of tiles, and encodes each tile as an independent video stream, called tiled stream. When an RoI is requested, the server sends the tile streams that overlaps with the RoIs. Figure 1 illustrates how tiled streaming works.

The second scheme we proposed is called *monolithic streaming*. Here, each video is encoded as a single monolithic video. When the server needs to send an RoI, the server sends all macroblocks that overlap with the RoI, as well any other macroblocks that are needed in order to decode the macroblocks in the RoI. The server needs to parse through the encoded video and build a data structure that identifies the dependencies (including motion vector dependencies and VLC dependencies) among the macroblocks. This data structure is looked up during streaming to decide whether a

MMSys'11, February 23–25, 2011, San Jose, California, USA.

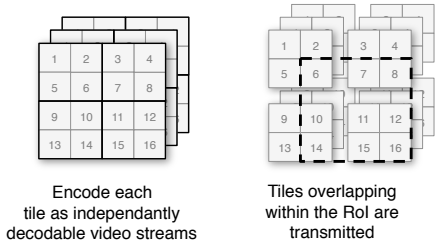


Figure 1: Tiled Streaming. Macroblocks at the same position in different frames are grouped into tiles. Each tile is independently encoded and thus can be independently decoded. Tiles overlapping with the requested RoI are sent.

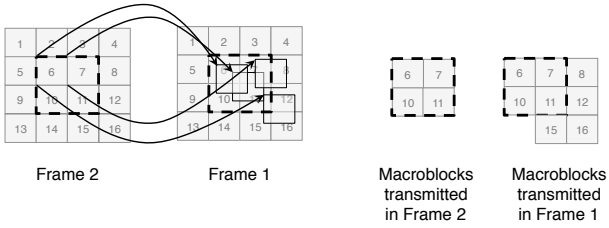


Figure 2: Monolithic Streaming. Macroblocks belonging to the RoI (box with dash line) are sent, along with any other macroblocks outside of the RoI that are dependent (curved arrows) on by macroblocks in the RoI in subsequent frames. We show only motion vector dependencies for simplicity.

macroblock needs to be sent given the current RoI. Figure 2 illustrates the monolithic streaming scheme.

Unlike the naive schemes, however, both monolithic streaming and tiled streaming pay the price of sending additional bits outside the RoI that are not displayed by client. In the case of tiled streaming, region of a tile that falls outside of RoI will not be displayed, but is sent anyway to the clients. To reduce the wasted bits, one can reduce the dimension of the tiles. But since each tile is encoded independently, small tiles lead to lower compression ratio, increasing the number of bits needed for the RoI. In the case of monolithic streaming, macroblocks outside of the RoI that are depended upon (either directly or indirectly) by the macroblocks inside the RoI are sent. To reduce these macroblocks, we can reduce the amount of dependencies in the video stream by tuning the encoding parameters (such as motion vector search range and number of B-frames). Reducing dependencies, however, lead to lower compression ratio, and increases the number of bits needed for the macroblocks in the RoI. Our previous paper has carefully study the trade-offs in different encoding parameters and tile sizes for these two schemes [16].

This paper presents our investigation into how we further reduce the wasted bits when transmitting an RoI. We exploit the fact that not every possible RoI is equally interesting to the users – some RoIs are requested more frequently than others. For instance, in a lecture video, the RoIs centering on either the lecturer or the projector display tends to be more popular. Our user study on viewing behavior [3] has revealed that the user interest tends to be consistent, i.e., the access frequency to the RoIs is highly skewed and popular regions are spatially clustered.

Given historical traces of how users access the RoIs, we can adapt the encoding parameters for both tiled streaming and monolithic streaming such that, more popular RoIs (i.e., RoIs with higher probability of being requested) require fewer bits to transmit. As a result we can expect that the *expected* bandwidth needed to transmit the RoIs will be reduced. Specifically, we can adapt the motion vector search range for monolithic streaming, to reduce the expected amount of motion vector dependencies from within an RoI to outside of RoI and to increase the expected amount of motion vector dependencies within an RoI. For tiled streaming, we adapt the tile size, such that the tiles that overlap with the RoIs have a smaller region falling outside of these RoIs. Although the two approaches seem to be unrelated, both of them rely on a common technique, namely, to restrict motion search range by different amount for every macroblock in the video.

We evaluate our proposed approaches using user access patterns that we have collected on four standard test video sequences exhibiting different scene complexity and motion. We find that by adapting the motion vector search range of monolithic streams, we can reduce the expected bandwidth by up to 21%; by adapting the tile size in tiled streams, we can reduce the expected bandwidth by up to 27%. Our results are encouraging and shows the efficacy of our approach.

We organize the rest of the paper into six sections. Section 2 discusses the existing literature on encoding videos with RoIs. Section 3 describes a web-based zoomable video system and how we obtained access patterns from users. Section 4 presents the exploitation of user access pattern to adapt the tile size for tiled streaming. Section 5 presents how we exploit the user access pattern to reduce the expected bandwidth when monolithic streaming is used to transmit RoIs. We evaluate both schemes and present our results in Section 6. Finally, we conclude in Section 7.

2. RELATED WORK

There has been significant study in the field of what constitutes a RoI, its characteristics, and how users interact in the context of video and image viewing [11, 18, 20, 15, 4]. We focus this related work section on issues related to encoding of videos with considerations to RoIs.

In the context of zoomable video streaming, bitstream switching has been proposed as a possible solution to multi-resolution representation [6, 9, 8]. Video is encoded at multiple resolutions, with the lowest resolution being streamed by default. Users can zoom into the video by selecting a region from the low resolution video. The corresponding region is cropped from a higher resolution video and transmitted. In essence, the video server switches between (and crops from) different resolution videos when users zoom in and out.

Different approaches have been proposed to encode videos with bit-stream switching. In the context of viewing a selected RoI from a high-resolution panoramic video stream [6], a grid-based approach is proposed. The panoramic view is broken into rectangular tiles of size 512×512 pixels, where each tile is an independently decodable entity. All tiles falling within and intersecting with the RoI boundary are streamed. Tiling creates independence among regions as unnecessary tiles can be dropped and new tiles included into the scene as and when the RoI changes.

A practical approach to adaptive bit rate streaming of panoramic video with tiling is proposed recently in [8]. The proposed approach exploits the fact that MPEG4 multi-view coding standard provides inherent synchronization among tiles when tiles are encoded as separate streams. Video frames are broken into tiles, and each tile is encoded as a separate stream at three different qualities. The lowest resolution stream along with meta-data representing the tile identifiers is streamed. The viewer can select a region, and tiles corresponding to that region are selected and streamed from the highest resolution panoramic video. Each selected tile from the panoramic stream is selected at a quality matching available bandwidth. This approach supports only two resolutions, but handles cropping with bit rate scaling well.

In the context of H.264 video, special frames [9], called SI and SP frames, can be used for bit-stream switching. These frames allow switching between video streams encoded with different parameters at non-GoP boundaries. This feature is possible only when the two video streams are of a single video sequence. The SP frames are transmitted whenever there is a need to switch the bit-stream. In the context of zoomable video, SP frames can be transmitted when the zoom level of the RoI changes. The display area of the RoI itself is handled by RoI cropping.

Another approach to efficiently encode videos in the context of zoomable video using scalable video, is to use background extraction [14]. In this approach, considering the case where the camera is static, the authors construct an intracoded *background frame* using temporal median operator. There are two reference choices for inter-coded blocks: either refer to the background frame or upsampled base-layer. Motion-compensated prediction is limited to long-term memory motion-compensated prediction only, while motion-compensation among consecutive frames is avoided.

As RoI-based streaming involves cropping rectangular regions, Feng et al. suggested an encoding approach that breaks the frame into tiles by constraining motion estimation [5] to within the tile area. All tiles falling within the RoI or intersecting with the RoI boundary are selected for streaming. This approach allows composition of rectangular RoIs of any dimension. An important issue to be addressed in this approach is the size of the tile and the impact of the size on bandwidth efficiency. In the context of H.264 encoding, Mavlankar et al. showed that using a slice (tile) of dimension 4×4 macroblocks [13] is optimal for RoI cropping.

The related work discussed above changes motion compensation in video encoding to allow for random access to RoIs. This approach often sacrifices compression efficiency, increasing file size and transmission bandwidth. To further reduce the bandwidth, methods that vary the quantization scale based on the RoIs have been proposed. The idea is simple: non-RoI areas are accessed less often and thus can be encoded with bigger quantization scale and lower quality.

One such proposal exploits flexible macroblock ordering (FMO) [19] in H.264 video streams. RoIs are encoded as single FMO and all macroblocks falling within this FMO are quantized using a fixed quantization scale. Every time the RoI changes, FMO is redefined for the new RoI. Macroblocks falling outside the FMO use a quantization scale that results in significantly lower bit allocation for this region. The quantization scale for the FMO region is determined by empirical evaluation. Although the bandwidth of the video reduces and the PSNR for the RoI increases, this approach results in an abrupt quality transition between RoI and non-RoI regions.

To handle the issue of abrupt quality transition, Huang and Lin propose a transition region [7], encoded with a quantization scale different from the RoI region and the non-RoI region. A more formal way to choose quantization scale while handling the case where a RoI can have varying degree of viewership is described by Lai et al. [10]. A rate-distortion optimization formulation is used to determine the quantization scale. All macroblocks in a RoI have a viewership weight assigned. One can view this weight as an access probability. The distortion metric for each macroblock is a product of the weight, variance in the macroblock region with respect to the reference, and exponentially weighted bit rate for that macroblock.

All quantization-scale based approaches attempt to maintain a fixed bit rate while improving upon the PSNR. Nevertheless, Reingold and Loschky [17] have shown that continued perceptual variations caused by changing quality in RoI-based encoding results in perceptual distraction. Loschky and Wolverton also [12] found that using a higher quality encoding at the point of gaze/attention when compared to the peripheral areas, results in reduced saliency when users try to identify other RoIs. Besides distraction, another problem with quantization scale-based rate-distortion optimization is that of reduced quality when users select a non-RoI region. These approaches assume that once a RoI is defined users are likely to access that RoI alone. This is not true in interactive RoI access systems and in systems where there can be multiple RoIs all of which cannot be guessed apriori.

Our approach in this paper differs from the two major approaches above, and can be viewed as a loose hybrid of the two approaches. On one hand, we play with the motion compensation component of video encoding by defining a per-macroblock motion vector search range for monolithic streaming, and by constraining motion vectors to within a tile for tiled streaming. On the other hand, we distinguish between popular RoI and non-popular RoIs, and encode them differently. Our approach, however, can encode each region with the same quality, avoiding distraction caused by variations in perceptual quality.

3. ROI ACCESS PATTERNS

In this section, we describe a web-based system that was used to log users' interaction with a zoomable video. The interaction log provides us with a map of RoI accesses along with the access frequency, from which we model the probability of access of each macroblock.

The web-based system consists of a HTML5 interface that provides users with abilities to zoom and pan in a video during playback. An interaction area displays the video corresponding to the user's current region-of-interest (RoI) in a window of size 320×192 pixels. The entire video frame is visible in a smaller 160×90 pixel thumbnail display, showing the context of the RoI. One can use the arrow buttons on the keyboard to pan, or zoom/unzoom using the (+,-) buttons. The mouse can also be used to zoom/unzoom using the scroll wheel, or pan by clicking and dragging.

The highest resolution video has a resolution of 1920×1080, from which six levels of zooming (0-5) corresponding to six bitstreams with different resolutions are derived. At Level 0, users see the whole video at 320×192 resolution. At higher zoom level, the RoI is cropped and scaled down to 320×192

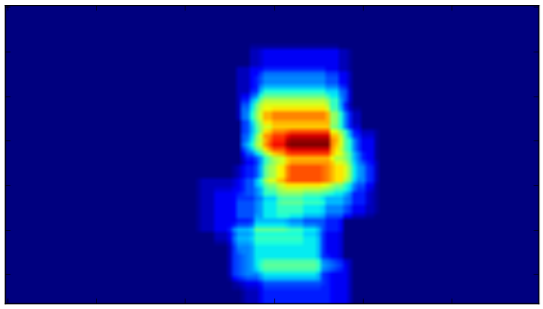


Figure 3: RoI Access Pattern

for display. All interactions are done locally at the client after the original video is downloaded. Thus, the response time for zooming and panning is negligible as the data to display is already available at the client.

We recorded four videos at resolution of 1920×1080, consisting of magic shows and gymnastic competitions. User's interactions with these videos were logged as *sessions*. A session consists of a sequence of interaction activity on a video by a user, starting from the time the user loads the video to the time the user leaves the page. There were a total of 53 users whose interactions were logged over a period of two weeks, with a total of 11183 interaction events (zoom, pan, play, and stop), averaging about 13 users per video. Each video was seen to have between 800-1300 RoI changes, out of which about 66-375 RoI transitions are at zoom level 5. Once users zoom into the video (an RoI transition) they tend to view the video at that zoom level for consecutive frames.

The frequency of access of a RoI is computed as the number of times the RoI has been viewed by all users. The RoI is allowed to change only at a GoP boundary, and depending on motion and user interest, RoIs can be spread about in a frame. Figure 3 shows one such access pattern. The figure shows that many RoIs are clustered around specific regions. The bright red regions have a higher probability of access when compared to the paler regions. The probability of access of a RoI is computed as the normalized frequency of occurrence of the RoI in a GoP. Hence, the probability of access of a region/macroblock is the sum of probabilities of all the RoIs that the region/macroblock is a part of.

4. ADAPTIVE TILING

4.1 Tiled Streaming

Tiled streaming is one of the RoI streaming methods we introduced in [16]. It is inspired by how Web-based map services work, where huge maps are divided into smaller grids of images. Only the images that overlap with the RoI are sent to users for display. In the context of RoI streaming, video frames are divided into a grid of tiles. Tiles of the same (x, y) co-ordinates from all frames in a GoP are grouped together and encoded to form an independent tiled stream. Motion vectors and other dependencies are constrained to within the tile area. For each RoI requested, the server transmits a minimal set of tiled streams that cover the RoI. As such, tiled streaming introduces simplicity at the server side. Besides, publish-subscribe paradigm can be applied to this method. The server can multicast each tiled stream to a channel. A client that is interested in some RoI may subscribe to channels which have tiles necessary for the decoding that RoI.

One major drawback of tiled streaming is that data outside the RoI may be transmitted to users. Since the RoI is not always aligned with tile boundaries, some tiles may partially overlap with the RoI. These tiles are transmitted in whole and hence, some bandwidth is wasted in transmitting regions outside the RoI. Our previous study [16] has shown that tile size is one of the most influential factors to the performance of tiled streaming in terms of compression and bandwidth efficiency. Small tile size helps reduce bandwidth wastage but reduces compression efficiency due to the fact that motion estimation is constrained within a tile. On the contrary, large tile size helps achieve better compression efficiency but leads to more redundant data being transmitted for partially overlapped tiles.

We previously limited our study to *regular tiling*, i.e., covering the frames with tiles of the same size (except at the boundary of the frames). Removing this limitation, by allowing tiles of different sizes, can further reduce the transmission bandwidth of RoIs. To get an intuition of why this is the case, consider the following simple scenario, where every user is interested in viewing the same RoI. Assume that we start with a regular tiling. All tiles that completely fall inside the RoI can be merged into a single larger tile. Since larger tiles have better compression efficiency, the overall bandwidth needed to transmit the RoI reduces. Further, tiles at the boundary of the RoI that partially overlap with the RoI can be split into smaller tiles. This reduces the area of the region from outside of RoIs that has to be sent. Note that in this case, splitting a larger tile into smaller ones is not always beneficial, since smaller tiles may consume more bandwidth due to reduced compression efficiency.

We now extend the above extreme case to a more general case, where different users may be interested in viewing different RoIs. Now, each macroblock has some probability of belonging to an RoI. Suppose that the access probability to each macroblock is known, we would like to find the best way to tile the video such that the expected bandwidth to transmit the set of RoIs is minimum. We refer to this problem as Adaptive Tiling Problem. The formulation of this problem is discussed below.

4.2 The Adaptive Tiling Problem

In this section, we will formally describe the adaptive tiling problem studied in our paper as well as some definitions and notations related to the problem.

We abstract the problem of tiling a video into tiling of rectangles. A rectangle is always identified by its top-left coordinates (x, y) and its width and height (w, h), where $x, y, w, h \in \mathbb{Z}$. We write a rectangle r as (x, y, w, h). A rectangle r_i is *contained* in another rectangle r_j if and only if the four corners of r_i are either on or within the boundary of r_j. We write $r_i \sqsubseteq r_j$ if r_i is contained in r_j. We say $r_i \cap r_j \neq \phi$ if the rectangles r_i and r_j overlap.

We denote the rectangle that we are interested in *tiling* as R. A tile map T of R consists of a set of non-overlapping rectangles, called *tiles* and denoted $t_1, t_2, ...t_n$, with each of t_i contained in R and collectively covering exactly R. Each tile t is assigned a cost $c(t)$, which is given by a cost function c.

The *heat map* for a rectangle R is a function p that maps any rectangle r contained in R to a non-negative real number $p(r)$. Given an rectangle r, we denote $\Omega(r, T)$ as a set of tiles in T that overlap with r.

Adaptive Tiling Problem: Given a rectangle R, a heat map function p, a cost function c, find a tile map $T(R)$ such that

$$\sum_{r \sqsubseteq R} p(r) \sum_{t \in \Omega(r,T)} c(t) \tag{1}$$

is minimized.

It can be shown that Expression 1 is equivalent to

$$\sum_{t \in T(R)} c(t) \sum_{r \sqsubseteq R, r \cap t \neq \emptyset} p(r) \tag{2}$$

Proof: Let the expression in 1 be E. Then,

$$E = \sum_{r \sqsubseteq R} \sum_{t \in T} c(t) f_t(r) \tag{3}$$

$$where\ f_t(r) = \begin{cases} p(r) & \text{if } t \in \Omega(r,T) \\ 0 & \text{if } t \notin \Omega(r,T) \end{cases}$$

$$\Rightarrow E = \sum_{t \in T} c(t) \sum_{r \sqsubseteq R} f_t(r)$$

$$\Rightarrow E = \sum_{t \in T} c(t) \sum_{r \sqsubseteq R, r \cap t \neq \emptyset} p(r)\ \text{(Shown)}$$

The relationship between the above formulation and adaptive tiling for zoomable video streaming is as follows: The rectangle R is the video frame. The heat map function p maps an RoI to its access probability. The cost function is the compressed size (in bytes) of the tile. Expression 1 and 2 therefore compute the expected amount of data transmitted for an RoI. Equivalently, our goal is to minimize the *expected data rate* or *expected bandwidth*. Note that in adaptive tiling, we are given a set of RoIs and its access probability, while Expression 1 sums over all possible rectangles contained in the video frames. These two are equivalent if we set $p(r)$ to 0 for any r that is not an RoI (i.e., r has a different dimension than an RoI).

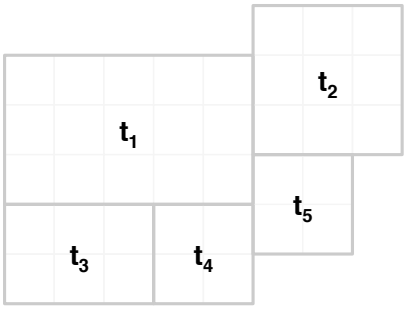
4.3 A Greedy Heuristic

Finding the optimal tile map that minimizes the expected bandwidth is likely to be NP-complete. We therefore use a greedy heuristic to find a tile map that reduces the expected bandwidth. We present our heuristic in this section.

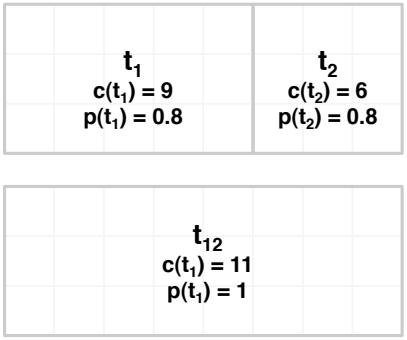
Let's first define a few more notations and terms. Given two tiles $t_i = (x_i, y_i, w_i, h_i)$ and $t_j = (x_j, y_j, w_j, h_j)$, t_j is said to be *right neighbor* of t_i if $x_i + w_i = x_j$ and $y_j \leq y_i \leq y_j + h_j - 1$. t_i is *right mergeable* with its right neighbor t_j if $y_i = y_j$ and $h_i = h_j$. We similarly define *bottom neighbor* and *bottom mergeable*.

We can merge a tile (x, y, w, h) with its right mergeable neighbor (x', y, w', h), creating a new tile $(x, y, w + w', h)$. Similarly, we can merge a tile (x, y, w, h) with its bottom mergeable neighbor (x, y', w, h'), forming a new tile $(x, y, w, h + h')$.

Figure 4(a) shows some examples of neighbors and mergeable neighbors. t_1 has bottom neighbors t_3 and t_4 and right neighbors t_5. t_2 is not a right neighbor of t_1. Only t_3 and t_4 are right mergeable. After merging, however, t_1 is bottom mergeable with the merged tile.



(a) Example of neighbors and mergeable neighbors.



(b) Example of a good merge.

Figure 4: Example of Neighbors and Merge Conditions

We abuse the notation of heat map function p for the *access probability p(t)* of a tile t. *Access probability* of a tile t is the sum of access probabilities of RoIs overlap with t. Accordingly, Expression 2 can be written as:

$$\sum_{t \in T} c(t)p(t) \tag{4}$$

We are using Expression 4 in our algorithm to compute the expected bandwidth. We say that two tiles t_i and t_j form a *good merge* if the new tile t_k formed by merging t_i and t_j helps reduce expected bandwidth, i.e.,

$$p(t_i)c(t_i) + p(t_j)c(t_j) \geq p(t_k)c(t_k) \tag{5}$$

Note that the compressed size of the new tile stream $c(t_k)$ is likely to reduce ($c(t_k) \leq c(t_i)+c(t_j)$), while its access probability $p(t_k)$ may increase ($p(t_k) \geq p(t_i) + p(t_j)$). As such, even if the expected data rate remains the same, merging is performed if it helps to reduce size of the newly formed tiled stream. Figure 4(b) shows an example of a good merge.

Given the above introduction, we are now ready to present our heuristic. We start with a tile map consisting of regular 1×1 tiles. The idea underlying our heuristic is to repeatedly merge a tile with its neighbor (right or bottom neighbors) as long as it is a good merge (and update the tile map with the merged tile). This step is repeated for each tile in the tile map until no good merge is possible. In our algorithm, we encode that tile on the fly to obtain the compressed size of

a tile. This helps our algorithm make better decision when growing tile.

The pseudocode to merge a tile with its neighbors is shown in Algorithm 1. The methods RIGHT and BOTTOM returns the right- and bottom-mergeable neighbors of a tile respectively, if such neighbor exists. The methods ISMERGEABLE checks if the conditions for mergeable are satisfied. Finally, the method TRYMERGE checks if the given tiles form a good merge. If so, it merges the tiles, updates the tile map, and returns the reduction in the expected bandwidth due to the merge. Otherwise, it returns 0. Algorithm 1 does not show error handling cases for simplicity.

The method, called GROW, basically considers first the basic cases of merging with the right neighbor and bottom neighbor. Then it checks if there is a diagonal neighbor (bottom neighbor of right neighbor) that is mergeable with both the tile's right neighbor and bottom neighbor. If so, this means that the tile can potentially merge with its right, bottom, and diagonal neighbor to form a larger tile. Here, the method tries all possible merging configurations. Finally, it picks the configuration that maximizes the reduction in expected bandwidth, and updates the tile map accordingly.

The method GROW is applied for every tile in the tile map to further reduce expected bandwidth. Tiles are sequentially traversed in the increasing order of y-coordinate followed by x-coordinate of their top-left corners. If a tile is merged with its neighbors at the end of GROW, this method is again applied for the newly-formed tile to greedily grow it further if possible. Otherwise, the next tile will be traversed. After the last tile has been reached, more passes are made through the tile map to find other possible merges until no further reduction in expected bandwidth is achieved. This step is necessary since after every loop, the tile configuration changes, resulting new mergeable neighbors.

Algorithm 1 GROW(tile t, tile map T)

```
{create some temp tile maps}
T_h ← T
T_v ← T
T_hv ← T
{try to merge with right neighbor}
t_r ← RIGHT(t)
c_h ← TRYMERGE(t, t_r, T_h)
{try to merge with bottom neighbor}
t_b ← BOTTOM(t)
c_v ← TRYMERGE(t, t_b, T_v)
if ISMERGEABLE(t, t_r) and ISMERGEABLE(t, t_b) then
    t_d ← BOTTOM(RIGHT(t))
    if ISMERGEABLE(t_r, t_d) and ISMERGEABLE(t_b, t_d) then
        {a "diagonal" neighbor forms a rectangle}
        c_h ← TRYMERGE(t_b, t_d, T_h) + c_h
        c_v ← TRYMERGE(t_r, t_d, T_v) + c_v
        c_hv ← TRYMERGE(t, t_r, t_b, t_d, T_hv)
    end if
end if
{find the merge with largest reduction in cost among the possibilities and use that as the new tile map}
i ← arg max_{i∈{h,v,hv}} c_i
T ← T_i
```

4.4 Resulting Tile Maps

Figure 5 shows some example inputs and outputs from our heuristic. The top row shows three heat maps from three different GoPs. The color spectrum indicates the popularity of regions (i.e., probability that it will fall within an RoI), with red being popular and blue being not popular. The bottom rows show the output tile maps from our heuristic.

There are several interesting observations we would like to point out here. First, un-popular regions tend to be covered by huge tiles. Such huge tiles contribute to better compression efficiency of the video. Note that this does not help in reducing the expected bandwidth. On the contrary, in the rare event that a user selects an RoI that intersects with such huge tiles, the tiles need to be transmitted, increasing the transmission bandwidth.

Second, the center area of popular regions (such as the yellow region in the left most heat map and green region in the middle heat map) tend to be encoded with tiles slightly smaller than a RoI. These regions typically show interesting contents that users tend to zoom into and are contained in the many requested RoIs. Encoding these regions with such tiles leads to better compression compared to using regular, small, tiles. Since these regions are requested often by the user, good compression in the regions leads to lower expected transmission bandwidth. Further, since such regions are contained within many RoIs, there is less chance that they partially overlap with the RoIs, hence less chance of wastage.

Finally, the boundaries of the RoIs tend to be coded with long thin tiles. Such tiles are used to encode regions surrounding popular regions with interesting content. Since different users select their RoIs at slightly different positions, such thin tiles neatly cover the boundary of the RoIs, reducing the wasted out-of-RoI areas in tiles that partially overlap with the RoI.

One final point to note regarding adaptive tiling. By integrating the cost of a tile into our model, our tiling can adapt to characteristics of the video content. For instance, if a video has much horizontal motion, then our heuristic tends to create more horizontal tiles, since such tiles can be better compressed compared to vertical ones.

5. MONOLITHIC STREAMING WITH ROI-AWARE CODING

Monolithic streaming (MS) refers to streaming videos encoded with a conventional standards compliant video encoder. Raw video frames are encoded as a sequence of GoPs and macroblocks are organized as slices while being encoded as intracoded or predictively coded. Predictively coded macroblocks have motion vectors that are forward/backward referencing or coded in direct mode.

RoI is streamed by transmitting all macroblocks within the RoI along with macroblocks referenced as a result of motion vectors. Referenced macroblocks may fall within the RoI or may lie outside the RoI. Referenced macroblocks falling within the RoI would anyway be transmitted when the reference frame is streamed. Macroblocks referenced from outside the RoI need to be transmitted before the referencing frame is transmitted. A single motion vector may refer to at most four other macroblocks in the reference frame. In turn, these four macroblocks may refer to four others. This form of motion vector dependency can result in a large

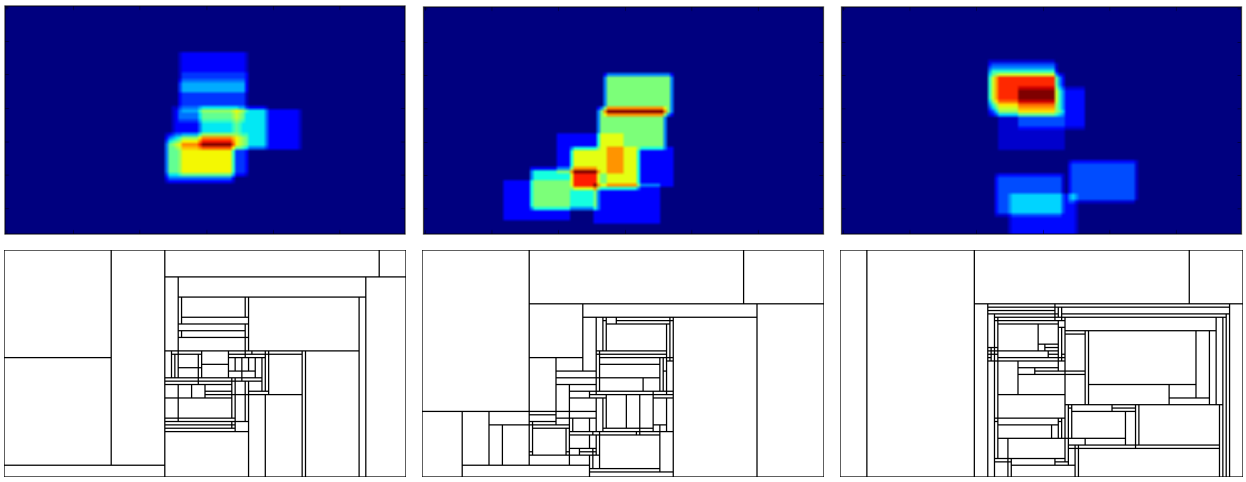


Figure 5: Top row: Heat maps from three different GoPs. Bottom row: Tile maps produced by our algorithms using the corresponding heat maps in the same column.

.

region outside the RoI to be transmitted. Figure 6(a) shows a rectangular RoI with colored area around it. The colored area is the motion vector dependency that needs to be transmitted in order to decode the RoI. The colored area outside the RoI represents the extent of spatial spread of motion vector dependency. The color itself represents the extent of spread in the temporal direction. Bright red regions represent macroblocks which are referenced many more times compared to the blue regions. One of the test videos that we used in our evaluation showed that macroblocks in the last few frames of the GoP need macroblocks from the first few frames. One of the factors contributing to larger effective RoI is the nature of content in the video, as a result of which, macroblocks may be encoded with long motion vectors. One option is to force the encoder to limit the motion search range. But, it has been shown [16] that using short motion vectors is less bandwidth efficient, and also results in larger video files.

5.1 Limiting Motion-Based Dependency

In this section we describe an approach to statistically limit the spread of motion vector dependency in MS. We refer to this approach as Probabilistic Boxing (PB), and the encoder which uses this approach will be referred to as MS-PB. Probabilistic boxing involves constraining or "boxing" motion vector search to within pre-determined rectangular region. Unlike tiling, probabilistic boxing limits the motion vectors to regions based on access probability and regions can be non-rectangular.

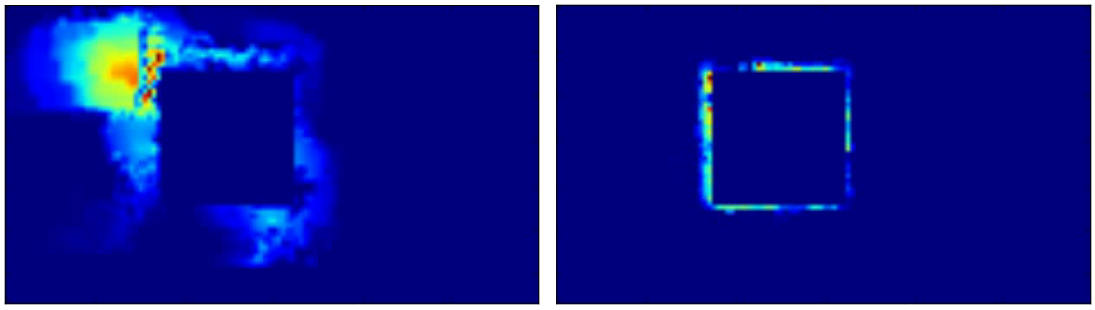
MS-PB uses a two pass encoding approach. In the first pass, the raw video is encoded in a standard format and made available to users. Once user interactions have been logged, the encoder analyzes the motion vector dependency of macroblocks in the encoded video with reference to the RoIs viewed by users. The result of the analysis step is a series of instructions specifying the motion search range for every macroblock in every frame of the video. The specified motion search range would allow the encoder to run a second pass encoding, such that, the motion vector dependency is confined to the expected regions-of-interest. As a result of confining motion vectors, the amount of data sent from outside the regions-of-interest is minimized.

Confining motion search has a negative impact on the size of encoded video. The first pass encoding would have determined the optimal reference macroblocks for predictively coded macroblocks. Limiting motion search may result in the reference macroblocks that are less than optimal. Hence the prediction error is higher there by increasing the size of predictively coded macroblocks. To account for this problem, MS-PB uses the probability of access of RoIs as well. If the expected size of macroblocks after the second pass encoding is lower than the expected size in the first pass, then, motion search for macroblocks may be constrained. As the expected size cannot be determined without actually encoding the video, MS-PB uses heuristics.

Figure 7 depicts how the RoIs are expanded to account for the variations in access pattern. The top left corner of each RoI from the user interaction log is quantized by a constant factor. The quantized co-ordinates are clustered in a fixed sized grid, the elements of which are squares with dimension corresponding to the variance of the distribution. Probabilistic boxing is performed by taking the smallest rectangular boundary of the grid elements enclosing all the RoIs for each cluster. In our implementation, we choose grid elements to be 2×2 macroblocks in dimension. In paragraphs to follow, we describe how the motion search range is determined when the regions-of-interest and access probabilities are known.

5.2 Problem Formulation

Figure 8 illustrates a case where two RoIs (R_1, R_2) have different probability of access $(p_1,\ p_2)$ with motion vector based dependency of R_1 spreading into R_2. Macroblocks in R_1 contributing to the spread are labeled A, and macroblocks in the spread are labeled B. In this particular example, boxing of motion vectors to within the region common to R_1 and R_2, would result in all macroblocks in the overlap area to have dependencies within the overlap region.



(a) Effective RoI with 0 B-Frames (b) Effective RoI with 2 B-Frames

Figure 6: Motion Vector Spread and Effective RoI

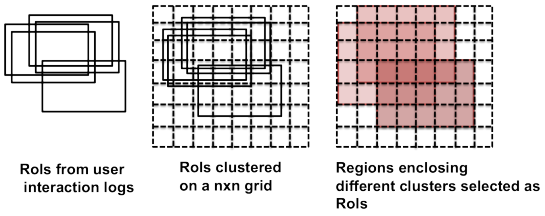


Figure 7: RoIs Fitting the User's Interactions

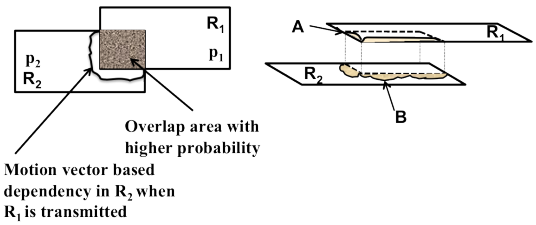


Figure 8: Example of Motion Vector Spread in Overlapping RoIs

In the default encoding case, predicted macroblocks are chosen based on error minimization. As a result, choice of any other predicted macroblock would invariably result in increasing the size of encoded macroblocks. As a result, boxing can result in a higher expected bandwidth. On the other hand, PB accounts for the probability of transmission of the dependency, estimate for the size of the dependency, and estimate for the increase in size due to boxing.

Let $p(A)$ and $p(B)$ be the probability of transmitting A and B respectively. Let $p(A|B)$ be the probability of transmitting A, when B is not the region from which motion vectors of A are estimated. Hence $p(A|B)$ is nothing but the probability that A and B are transmitted together as a result of being co-located in the same region. Hence the probability of transmitting A independent of B is $p(A) - p(A|B)$. If the condition $p(B) \geq p(A) - p(A|B)$ is satisfied, then it implies that the region which A depends on, is more likely to be transmitted than A itself. As motion vector prediction reduces the size of encoded macroblocks, prediction from a higher probability region should be allowed. Hence probabilistic boxing constrains motion search of A only if the condition

$$p(A) - p(A|B) > p(B) \tag{6}$$

is satisfied. As mentioned earlier, constraining motion search results in a larger encoded macroblock size. The increase in size should not counter the benefit of reduction in motion vector related spread. Hence (6) is modified to

$$[p(A) - p(A|B)]\,\overline{S(A)} > p(B)S(B) \tag{7}$$

where $\overline{S(A)}$ is the size of A after boxing motion vectors of A, and $S(B)$ is the size of B.

5.3 Intuition

During encoding, allowing longer motion vectors results in better compression. As a result the data rate while transmitting a RoI is also lower. On the other hand, the amount of motion vector dependency increases resulting in transmitting macroblocks from outside the RoI. Figure 6(a) shows a case where there is a large region outside the RoI that is need to decode the RoI. If the size of dependency is larger than the benefits due to better compression with longer motion vectors, the purpose of using longer motion vectors is defeated. If we reduce motion search range (use shorter motion vectors) macroblocks are predicted from less optimal references. Hence the size of macroblocks increases. Now let us refer back to Figure 8. If we constrain the motion vectors in the overlap area, R_1 will be transmitted with a larger size for the overlap area but without motion vector dependency. On the other hand, R_2 will be transmitted with a larger size corresponding to the overlap area, but does not have any benefits of reduced motion vector dependency. If R_2 is accessed significantly more number of times than R_1, the expected bandwidth would have increased instead of decreasing.

5.4 Determining Motion Search Range

The encoder performs motion search for every macroblock. Hence we need to specify the motion search boundary for every macroblock that is likely to be predicted from outside the RoI. Algorithm 2 shows the steps involved in determining the bounding box for every macroblock $x \in A$.

In step-2, the boundary of RoI overlap ($R(x)$) in which x is present is determined. Then, in step-3, all the macroblocks that are needed to decode x and falling outside $R(x)$ are determined. Boxing x alone does not guarantee that the

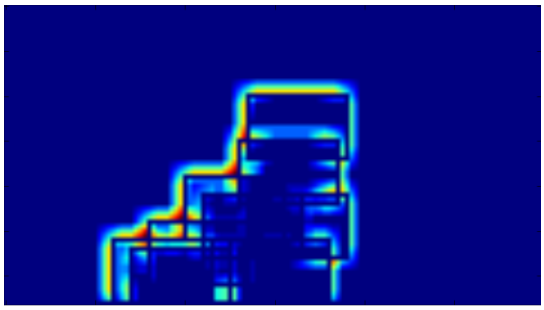


Figure 9: Motion Vector Spread Constrained to Within RoI Boundaries after Encoding with MS-PB

existing dependency $C(x)$ will vanish. Other macroblocks in the same region as x may refer to the same dependency. To eliminate the macroblock dependency of x, all macroblocks in $R(x)$ sharing $C(x)$ have to be boxed as well. In step-8, Z is the set of macroblocks in $R(x)$ sharing $C(x)$ and the size of the dependency that will be eliminated if x is boxed is $|C(Z)|$. Hence $S(B)$ in Inequality (7) is set to $|C(Z)|$ in step-11.

In steps 10-11, the size of a motion constrained macroblock x is set to the number of macroblocks that would have been constrained as a result of sharing the same motion vector based dependency as x. The size of dependency removed is set to the total number of macroblocks in the dependency. All macroblocks needed to decode x need not be present in the same RoI overlap. Hence each of these macroblocks can have a different probability. In steps 12-15, the lowest probability macroblock $y \in C(Z)$ is chosen, and its probability is selected as the probability of the dependent region. The sum of probabilities of all RoIs common to y and x is assigned to $p(x|y)$.

In our video encoder, we set the motion search range to 72 for all four directions by default. Once motion search range is constrained by boxing, macroblocks may have smaller search range that is different in four directions.

Figure 9 helps visualize the impact of constraining motion vectors to within RoI boundaries. The figure was generated by applying MS-PB to one of the test videos, and by plotting the motion search range for each macroblock. RoIs used during encoding are shown as rectangular boxes. To visualize the modified motion search range, we count, for each macroblock m, the number of other macroblocks whose motion vector cannot fall into m due to modified motion search range. We then plot this value in Figure 9, with higher values shown in reds and lower values in blues.

From Figure 9, we observe that motion vectors tend to be constrained at the boundary of RoIs, where the probability of access is low, as can be seen from the red and yellow bands around the RoIs. We also observe that, in regions where there are many overlapping RoIs, the motion vectors are not constrained, since the probability of access to such regions and regions around it tends to be high.

6. EVALUATION

The bandwidth efficiency of adaptive tiling (AT) and probabilistic boxing (MS-PB) was verified using four 1080p non-interlaced HD video clips [1, 2]. The clips used in our experiments are *Rush-Hour, Tractor, Bball,* and *Rainbow.*

Algorithm 2 Steps in probabilistic boxing

1: **for** $x \in A$ **do**
2: $R(x) \leftarrow$ region bounding x
3: $C(x) \leftarrow$ MBs required to decode x and lying outside $R(x)$
4: $p(x) \leftarrow$ sum of probabilities of RoIs in which x is present
5: **if** $|C(x)| == 0$ **then**
6: box the search range for x to $R(x)$
7: **else**
8: find $Z \in R(x)$ and sharing MBs from $C(x)$
9: $C(Z) \leftarrow$ MBs required to decode each $z \in Z$
10: $\overline{S(x)} \leftarrow |Z|$
11: $S(B) \leftarrow |C(Z)|$
12: $p(B) \leftarrow$ smallest probability in C(Z)
13: $p(x|B) \leftarrow$ smallest probability in C(Z) and in a region shared with x
14: **if** $\overline{S(x)}(p(x) - p(x|B)) > S(B)p(B)$ **then**
15: box the search range for x to $R(x)$
16: **end if**
17: **end if**
18: **end for**

Figure 10: Thumbnail Images of Frames from Test Videos

Rush-Hour is a 500-frame clip of a street scene during rush hour captured with a fixed camera. *Tractor* is a 688-frame clip showing a tractor ploughing a field. The camera tracks the tractor's motion, producing both foreground and background motion. Further, there is a gradual zoom-out at the end of the clip. *Bball* is a 200-frame clip of a basket ball match with camera tracking the play area along with zoom. *Rainbow* is a 350-frame clip of colorful objects on a rotating table. The camera is in a fixed position and gradually zooms into the objects at the end of the clip. Figure 10 shows a thumbnail of the four test sequences used.

FFMPEG encoder version 0.5 was used to encode these clips into MPEG-4 videos. The motion search range was set to 72 with motion vector scaling. The encoder was modified to scale motion search range r for P-frames and B-frames to $r \times (1 + d)$, where d is the distance to the reference frame in terms of number of frames. The quantization scale was set to 2 and full motion search was employed. The encoder was configured to arrange macroblocks into slices of length 64 bytes, and there were 25 frames per GoP.

Evaluation of AT and MS-PB was conducted by a training-testing framework. The two methods use one set of RoIs (training set) to guide the encoding parameters, and use another set of RoIs (test set) to verify the bandwidth efficiency

of the encoding. The test set is derived from the same distribution as the training set.

6.1 Generating Training and Test RoI Sets

To evaluate the merits of the encoding methods proposed in this paper, we pick user interactions from one specific video viewed via the web-based interface. This video is of a magic trick where the performance is always within the camera view and centered around the middle of the frame.

Note that in order to evaluate the efficiency of our proposed adaptive encoding method, we choose to use access patterns from one of our own video but test the encoding efficiency of a range of standard test video sequences. The rationale is that the standard test video sequences contain a variety of motions and complexity, and is good for evaluating video encoders. These test video sequences, however, are boring to watch over a zoomable video interface and are not suitable to gather realistic user interaction patterns.

We now explain how we generate the training sets and test sets from user interaction patterns. We generate one training set and one test set for each 25-frames GoP. Since RoI changes occur at the GoP boundary, the access patterns (and thus, the encoding parameters) can be different across different GoPs. Further, the access patterns can be different across different zoom levels. We focus our evaluations on Zoom Level 5, since this is the most challenging in terms of video resolutions (1920×1080) and most interesting in terms of RoI viewing patterns.

For each GoP at Zoom Level 5, we compute the frequency of access f_i of a RoI i as the number of frames for which i is viewed multiplied by the number of users who viewed i.

In our traces, there are 12 - 30 unique RoIs per GoP, which comprise of the training set. The video frame is divided into a grid of macroblocks organized in the x and y directions. The RoI is macroblock-aligned and are of size 20×12 macroblocks (same dimension as the viewing interface, 320×192 pixels). We use only 11 GoPs worth of training sets to match the length of the test video sequences (our user traces is much longer).

These training set is used as input to our encoding algorithm for both monolithic streaming and tiled streaming. To evaluate the expected bandwidth, however, we cannot use the same set since it is unrealistic that future users would access exactly the same set of RoIs. We thus need to generate another sets of RoI access patterns to mimic new users. The user access patterns, however, are expected to be the same as we have shown in [3].

To generate the test set, we approximate the distributions of the RoI, and generate a new set of RoIs following the same distribution. To do so, we create bins of RoIs locations by quantizing the top left corner of each RoI. All RoIs with top-left corners falling into the same bin are grouped as a cluster. We use a bin size of 2×2.

The frequency of occurrence of a cluster is the sum of frequencies of occurrence of all RoIs in that cluster. From this frequency of occurrence, we can derive the probability that a requested RoIs fall into a cluster. We can now generate the test set by randomly choosing an RoI according to the cluster probability, once a cluster is chosen, the exact position of RoI is uniformly and randomly chosen from the four possible positions within the cluster.

With the generated RoIs in the test sets, we can now evaluate the performance of our encoding algorithms.

6.2 Per-RoI Encoding

We compare the performance of our algorithms to a reference encoding approach we denote as *PerRoI*. This method crops the RoI from the raw video and re-encode it as a standard MPEG-4 video before transmission. In essence, PerRoI encodes videos exactly to the RoI dimensions and position within a frame. It is not scalable when there are large number of users. Nevertheless, PerRoI serves as a benchmark when we want to evaluate our encoding algorithms, since PerRoI transmits zero bytes from outside a specified RoI.

6.3 Comparison of Encoding Approaches

We compare the performance of the proposed encoding approaches in terms of their ability to compress a video file as well as the bandwidth efficiency while streaming RoIs. We compute the expected bandwidth as the total number of bits transmitted for a given RoI, multiplied by the probability of occurrence of the RoI. Every GoP has a different set of RoIs. If the number of test set RoIs are less than the total number of GoPs in the test video, we re-use the same set of RoIs in a round-robin fashion. The expected bandwidth of each RoI in a GoP is added to give the expected bandwidth for that GoP. Bandwidth across GoPs is added and averaged to give the expected bandwidth for the entire video clip. We use PerRoI along with MS-PB and AT for comparison purposes. We also use two tiled steams encoded with 16x16 tiles (TS16x16) and 4x4 tiles (TS4x4) for benchmark purposes. We compared the PSNR values for MS-PB, AT, MS and found that the difference was insignificant.

6.3.1 Compression Efficiency

Figures 12 and 11 show a comparison of encoded video files using AT and MS-PB. We observe that PerRoI is the most efficient in compression. In fact, it is unfair to compare the file size of PerRoI with other methods, as PerRoI compresses cropped frames while other methods compress the full frame. The size of the RoI in our experiments is 20x12 macroblocks. This size is significantly less than the frame dimensions of 120x67 macroblocks. Nevertheless, we do show PerRoI's file size for completeness. Monolithic streaming is the most efficient in compressing the video file, since the video is encoded using standard encoder without constraints of motion compensation. MS-PB and AT result in increased size as a result of constraining motion vector search. This trend is seen in all the test videos.

A point to highlight is the comparison between AT and TS16x16. Among the several regular tiling configurations we evaluated, TS16x16 achieves the best compression. AT is found to be comparable to TS16x16 (no more than 2.5% larger).

Another observation is that the compression efficiency is better when B-frames are not used in spite of using an increased motion search range for P-frames in the presence of B-frames. All the test videos have reasonably high motion. When we encode B-frames we do not have the option of choosing intracoded macroblocks if the variance between the macroblock and its reference is high. In P-frames, macroblocks can be intra-coded when the variance between the macroblock and the best reference macroblock is high.

6.3.2 Expected Data Rate

We compare the expected data rate for the case where there are two B-frames between P-frames and the case where

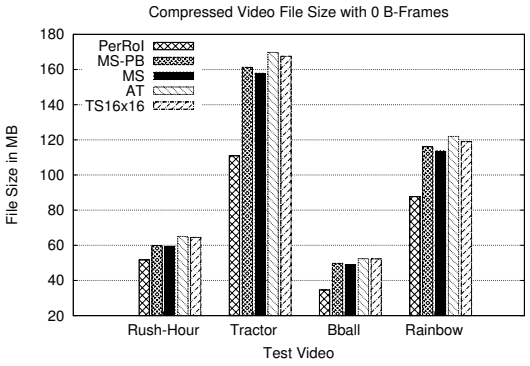


Figure 11: File Size Without B-Frames

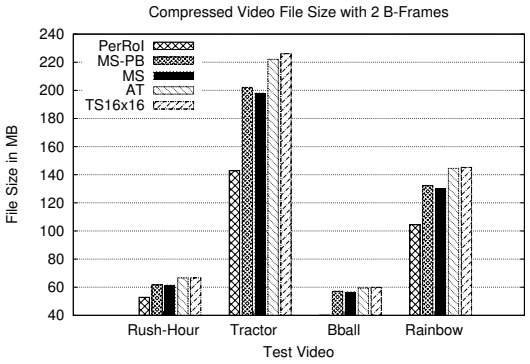


Figure 12: File Size With 2B-Frames

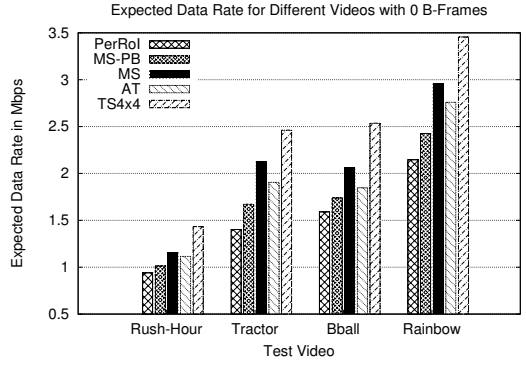


Figure 13: Data Rate Without B-Frames

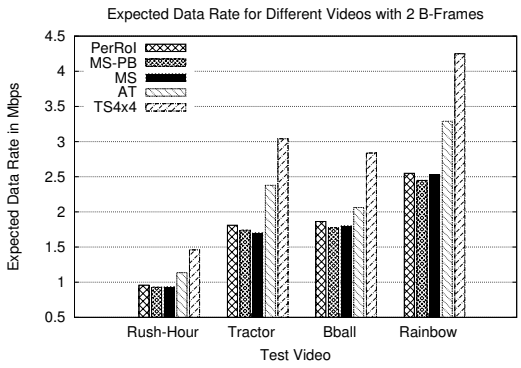


Figure 14: Data Rate With 2B-Frames

the video is encoded without B-frames in Figures 6.3.1 and 6.3.1. We use TS4x4 as a benchmark because smaller tiles result in less wasted bits in the case of tiled streaming.

Comparing AT and TS4x4, we see that AT's expected bandwidth are about 20% - 27% smaller than TS4x4 with or without B-frames for all four test sequence. Comparing MS-PB with MS, we found that MS-PB only perform significantly better than MS in the case without B-frames (12% - 21%). With B-frames, MS-PB is comparable with MS.

The presence of B-frames also affects the relative performance of PerRoI and MS-PB. Without B-frames, PerRoI requires 7% - 16% less bandwidth than MS-PB. With B-frames, however, PerRoI requires 3% - 5% more bandwidth than MS-PB. The latter is surprising to us, since intuitively we expect PerRoI to have smaller data rate, as it only transmit data within the RoIs.

We explain the surprising effects of B-frames in the following section.

6.4 Discussion

We found that, with the presence of B-frames, the motion vectors tends to spread less (see Figure 6(b)) in the temporal dimension. Without B-frames, the motion vectors spread more widely (see Figure 6(a)) and plays to the advantage of MS-PB's capabilities. With B-frames, the motion vector dependency is tightly packed around the RoI, MS-PB can do as best as MS.

The wide spread of motion vectors without B-frames also explains why PerROI is significantly better than MS-PB without B-frames. With wide spread of motion vectors, MS-PB has to transmit more data outside of the RoI due to dependencies (albeit less than MS). With B-frames, however, MS-PB needs to transmit much less due to the much smaller spreads. MS-PB now has an advantage over PerRoI – the macroblocks near the boundary of RoIs has more options during motion compensation, since the motion search range can extend outside of the RoIs. On the other hand, the PerROI motion compensation is limited within the RoIs. The fall in compression efficiency in PerRoI causes the overall bandwidth to transmit RoI to increase.

The same rational causes difference in the performance of AT compared to MS and MS-PB as well. AT is better than MS when there are no B-frames. Adaptive tiling allows for the creation of areas within which the motion search is

constrained. In some sense it is very similar to MS-PB. As the motion vector spread is high when B-frames are absent, AT is able to limit the spread at the expense of increased macroblock size and file size. Hence AT is more bandwidth efficient. On the other hand, AT is less bandwidth efficient when compared to MS and MS-PB when there are B-frames. As the motion vector spread is not high in this case, limiting motion search by tiling does not play a significant role.

7. CONCLUSION AND FUTURE WORK

This paper proposes an adaptive encoding approach in the context of a zoomable video streaming systems. Based on historical user access patterns, we compute the probability of each RoIs being requested, and encode the source video considering the RoI access probability. We adapt the motion vector search range in the case of monolithic streaming, and tile size in the case of tiled streaming. We show that by adapting the encoding parameters to RoIs request probability, we are able to reduce the bandwidth by up to 21% and 27% for monolithic streaming and tiled streaming respectively.

There are several practical issues that needs to be further looked into. The first is the computation time required. Both adaptive encoding methods are expensive and requires multiple passes of encoding. The fact that this is done offline in-frequently alleviate the problem but can still be an issue for large-scale video-on-demand systems with many video content. The second issue is frequency of adaptation. Our study over 53 users show that user interests tends to be stable over a short period of time, but it is possible for user interests to evolve over time. For instance, in a lecture video, users interest may be different when they view the video the first time (to copy notes from blackboard) and when they view it subsequently (just for revisions) The third issue is that different user profiles can have diverse user interest. For instance, in a sport video, supporters for different teams tend to zoom in to different players. In this case, the users need to be categorized, with the system providing multiple different versions of videos to cater for different access patterns. Prompt adaptation to changes of user interest over time and clustering of user behaviors into different profiles remain as open questions.

Acknowledgment

This research is supported by National University of Singapore Academic Research Fund R-252-000-368-112.

8. REFERENCES

[1] http://media.xiph.org/video/derf/.

[2] http://www.cdvl.org/.

[3] A. Carlier, G. Ravindra, and W. T. Ooi. Towards characterizing users' interaction with zoomable video. In *Proc. of International Workshop on Social, Adaptive, and Personalized Mutlimedia Interaction and Access (SAPMIA 2010)*, Florence, Italy, October 2010.

[4] L.-Q. Chen, X. Xie, X. Fan, W.-Y. Ma, H.-J. Zhang, and H.-Q. Zhou. A visual attention model for adapting images on small displays. *Multimedia Systems*, 9(4):353–364, 2003.

[5] W. Feng, T. Dang, J. Kassebaum, and T. Bauman. Supporting region-of-interest cropping through constrained compression. In *Proc. of ACM MM'08*, pages 745–748, Vancouver, British Columbia, Canada, 2008.

[6] 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 Work-shop PPW'05*, Feb 2005.

[7] C.-M. Huang and C.-W. Lin. Multiple-priority region-of-interest H.264 video compression using constraint variable bitrate control for video surveillance. *Optical Engineering*, 48(4):047004, 2009.

[8] M. Inoue, H. Kimata, K. Fukazawa, and N. Matsuura. Interactive panoramic video streaming system over restricted bandwidth network. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1191–1194, New York, NY, USA, 2010. ACM.

[9] M. Karczewicz and R. Kurceren. The SP- and SI-frames design for H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):637–644, 2003.

[10] W. Lai, X.-D. Gu, R.-H. Wang, W.-Y. Ma, and H.-J. Zhang. A content-based bit allocation model for video streaming. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 2, pages 1315 –1318, jun 2004.

[11] H. Liu, X. Xie, W.-Y. Ma, and H.-J. Zhang. Automatic browsing of large pictures on mobile devices. In *Proc. of ACM MM'03*, pages 148–155, Berkeley, CA, USA, 2003.

[12] L. C. Loschky and G. S. Wolverton. How late can you update gaze-contingent multiresolutional displays without detection? *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(4):1–10, 2007.

[13] A. Mavlankar, P. Baccichet, D. Varodayan, and B. Girod. Optimal slice size for streaming regions of high resolution video with virtual pan/tilt/zoom functionality. In *Proc. of EUSIPCO'07*, 2007.

[14] A. Mavlankar and B. Girod. Background extraction and long-term memory motion-compensated prediction for spatial-random-access-enabled video coding. In *PCS'09: Proceedings of the 27th conference on Picture Coding Symposium*, pages 61–64, Piscataway, NJ, USA, 2009. IEEE Press.

[15] A. Mavlankar, D. Varodayan, and B. Girod. Region-of-Interest prediction for interactively streaming regions of high resolution video. In *Proc. of International Packet Video Workshop, PV2007*, Lausanne, Switzerland, Nov. 2007.

[16] N. Quang Minh Khiem, G. Ravindra, A. Carlier, and W. T. Ooi. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proc. of MMSYS '10*, Phoenix, Arizona, USA, 2010.

[17] E. M. Reingold and L. C. Loschky. Reduced saliency of peripheral targets in gaze-contingent multi-resolutional displays: blended versus sharp boundary windows. In *ETRA '02: Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 89–93, New York, NY, USA, 2002. ACM.

[18] A. Santella, M. Agrawala, D. DeCarlo, D. Salesin, and M. Cohen. Gaze-based interaction for semi-automatic photo cropping. In *Proc. of ACM CHI '06*, Montréal, Québec, Canada, 2006.

[19] P. Sivanantharasa, W. Fernando, and H. K. Arachchi. Region of interest video coding with flexible macroblock ordering. In *Industrial and Information Systems, First International Conference on*, pages 596 –599, aug. 2006.

[20] X. Xie, H. Liu, S. Goumaz, and W.-Y. Ma. Learning user interest for image browsing on small-form-factor devices. In *Proc. of ACM CHI '05*, Portland, Oregon, USA, 2005.