Region-of-Interest Prediction for Interactively Streaming Regions of High Resolution Video

Aditya Mavlankar, David Varodayan and Bernd Girod
Information Systems Laboratory, Stanford University
Stanford CA 94305 U.S.A.

{maditya, varodayan, bgirod}@stanford.edu

***Abstract*— This paper investigates region-of-interest (ROI) prediction strategies for a client-server system that interactively streams regions of high resolution video. ROI prediction enables pro-active pre-fetching of select slices of encoded video from the server to allow low latency of interaction despite the delay of packets on the network. The client has a buffer of low resolution overview video frames available. We propose and study ROI prediction schemes that can take advantage of the motion information contained in these buffered frames. The system operates in two modes. In the manual mode, the user interacts actively to view select regions in each frame of video. The ROI prediction in this mode aims to reduce the distortion experienced by the viewer in his desired ROI. In the tracking mode, the user simply indicates an object to track and the system supplies an ROI trajectory without further interaction. For this mode, the prediction aims to create a smooth and stable trajectory that satisfies the user's expectation of tracking. While the motion information enables the tracking mode, it also improves the ROI prediction in the manual mode.**

I. Introduction

High resolution digital imaging sensors are becoming more widespread. In addition, high spatial resolution videos can also be stitched from views from multiple cameras, as implemented by Hewlett-Packard in their video conferencing product Halo [1]. However, challenges in delivering this high resolution content to the client are posed by the limited resolution of display panels and/or limited bit-rate for communications. Suppose that a client limited by one of these factors requests 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 or bit-rate. However, with this approach, the client might not be able to watch a local region-of-interest (ROI) in the highest captured resolution. We propose a video delivery system which enables virtual pan/tilt/zoom functionality during the streaming session such that the server can adapt and stream only those regions of the video content that are desired at that time at the client's end.

The proposed interactive delivery of video from pre-stored bit-streams necessitates video coding schemes that allow for sufficient random access to arbitrary spatial resolutions (zoom factors) as well as arbitrary spatial regions within every spatial resolution. We propose such a video coding scheme in [2]. Additionally, we have also developed a user interface with real-time interaction for ROI selection while watching the video sequence. This has been developed using OpenGL [3]. 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.

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. 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.

In Section II, we describe the streaming system and the two modes of interaction. Section II also mentions some prior related work and describes the novelty of our proposed approach. Section III presents the algorithms for ROI prediction and Section IV shows the experimental results.

II. Interactive Streaming System

The overall system is shown in Fig. 2. Notice that in our system, the client indicates its ROI and the desired spatial resolution (zoom factor) real-time to the server. The server then reacts by sending relevant video data which are 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. However, streaming over a best-effort packet-switched network implies delay, delay jitter as well as loss of packets. We design the system to work for a known value of the worst-case delay. Since the overview video always displays the entire scene, we allow some start-up delay and always send some number of frames of the overview video ahead of time. We can choose the size of the buffer such that it enables us to always deliver a minimum number of frames of the overview video in advance despite the worst-case delay of the network.

The client's ROI is not known to the server beforehand, since it is being decided by the user real-time at the

1-4244-0981-0/07/$25.00 ©2007 IEEE.

Fig. 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.

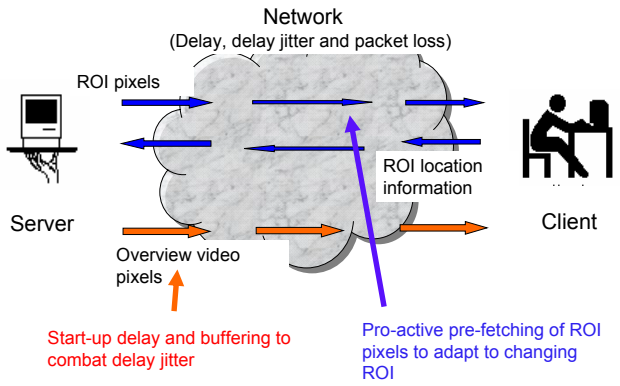


Fig. 2. Interactive streaming over a realistic network is challenged by delay, delay jitter and loss of packets. In the proposed system, some number of overview video frames are always delivered beforehand to the client. The ROI is predicted at the client's end at least one roundtrip time in advance and the appropriate ROI pixels are pro-actively pre-fetched.

client's side. Meeting the display deadline for the ROI display area is particularly challenging for the following reason; at the client's side, as soon as the ROI location information is obtained from the mouse, the requested ROI is rendered immediately. Notice that the client could delay rendering the requested ROI but this detracts from the user-experience, so we avoid it. In order to render the ROI spontaneously despite the delay of packets, we propose to predict the ROI of the user beforehand and use this to pro-actively pre-fetch those regions from the server. A timeline is shown in Fig. 3. The lookahead is d, i.e., we pre-fetch the ROI d frame-intervals in advance. Specifically, we have the following two modes of interaction:

- Manual mode: Predict the user's ROI d frame-intervals ahead of time. Note that the user continues to indicate his choice of the ROI in this mode.
- Tracking mode: The user right-clicks on an object in the ROI. The aim is to track this object automatically in order to render it within the ROI until the user switches this mode off. Note that in the tracking mode, the user is not required to actively navigate with the mouse.

A. *Manual mode*

Predicting the future ROI could be done by extrapolating the mouse moves observed until the current time instant. Ramanathan used a simple autoregressive moving average (ARMA) model for predicting the future viewpoint of the user in his work on interactive streaming of lightfields [4]. Kurutepe *et al*. used a more advanced linear predictor, namely the Kalman filter, to predict the future viewpoint for interactive 3DTV [5] [6]. However, all these approaches are agnostic of the video content itself. In this paper, we investigate the improvement of the ROI prediction by processing the frames of the overview video which are already present in the client's buffer. Notice that the motion estimated through the processing of the buffered overview frames can also be combined with the observations of mouse moves to predict the ROI even better.

If the wrong regions are pre-fetched then the user's desired ROI is still rendered by interpolating the colocated ROI from the overview video. The quality of the rendered ROI is lower in this case. Assuming this concealment scheme, the impact of the ROI prediction can be judged by the mean distortion in the rendered ROI. The lower bound is the mean distortion in the rendered ROI resulting from a perfect prediction of the ROI, i.e., distortion due to only the quantization at the encoder. The upper bound is the mean distortion when the ROI is always rendered via concealment.

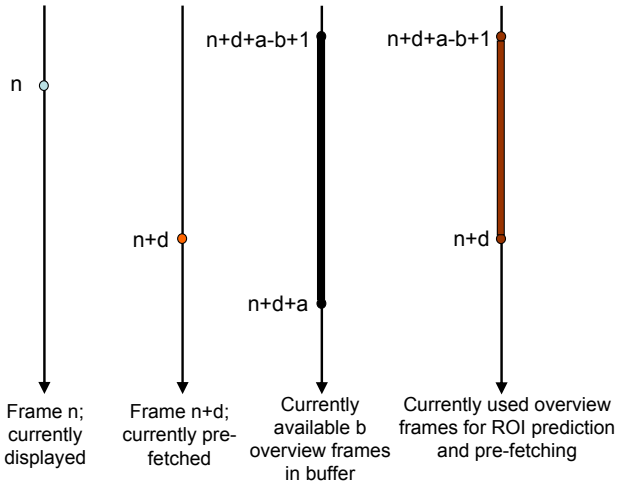


Fig. 3. Timeline: The ROI is predicted d frame-intervals in advance. There are b buffered overview video frames up to frame $(n+d+a)$ available. A subset of these b frames are processed in order to better predict the user's ROI. If the manual mode is switched on then the mouse translations are known till time instant n. In the tracking mode, the user does not indicate the ROI explicitly.

B. Tracking mode

Note that in this mode, the system is allowed to shape the ROI trajectory at the client's side; the fact that the pre-fetched ROI is simply rendered on the client's screen obviates error concealment when all the pre-fetched slices arrive by the time of rendering. Hence, the distortion in the rendered ROI is only due to the quantization at the encoder. However, it is important to create a smooth and stable trajectory that satisfies the user's expectation of tracking.

The application of tracking for reducing the rendering latency is one of the original contributions of this paper. The tracking of objects and/or features among frames of a video sequence itself has been demonstrated previously, for example in [7]–[13]. A variety of techniques for motion estimation, including optical flow estimation and block matching, have been employed in prior work on tracking.

III. ROI Prediction Algorithms

The objective of all our algorithms is to predict the user's ROI d frames ahead of the currently displayed frame n. Notice that this requires a 3D prediction in two spatial dimensions and one zoom dimension. As discussed in the previous section, there are two modes of operation. In the manual mode the algorithms may process the user's ROI trajectory history up to the currently displayed frame n. In the tracking mode, this source of information does not exist. A novel aspect of our work is that in both modes, the buffered overview frames (including n through $n+d$) are available at the client, as shown in Fig. 3. This means our algorithms may exploit the motion information in those frames to assist the ROI prediction.

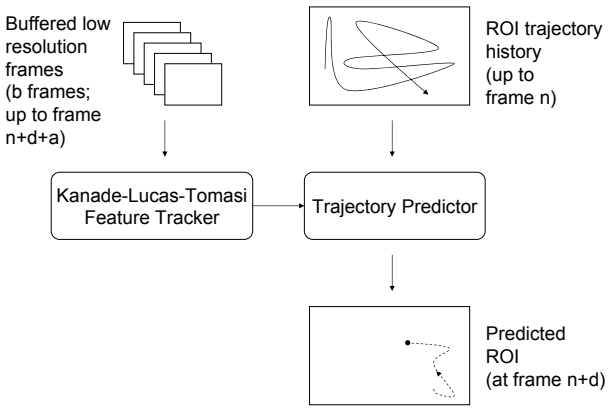


Fig. 4. Flowchart for the Kanade-Lucas-Tomasi feature tracker based trajectory prediction algorithm for the manual mode.

A. Manual Mode Algorithms

1) Autoregressive Moving Average (ARMA) Model Predictor: This algorithm is in fact agnostic of the video content. We apply the straightforward ARMA trajectory prediction algorithm of [4] to extrapolate the spatial co-ordinates of the ROI. Suppose, in the frame of reference of the overview frame, the spatial co-ordinates of the ROI trajectory are given by $p_t = (x_t, y_t)$ for $t = 0, 1 \ldots, n$. Then, we recursively estimate the velocity v_n according to

$$v_t = \alpha v_{t-1} + (1-\alpha)(p_t - p_{t-1}). \quad (1)$$

The parameter α trades off responsiveness to the user's ROI trajectory and smoothness of the predicted trajectory. The predicted spatial co-ordinates $\hat{p}_{n+d} = (\hat{x}_{n+d}, \hat{y}_{n+d})$ of the ROI at frame $n+d$ are given by

$$\hat{p}_{n+d} = p_n + dv_n, \quad (2)$$

suitably cropped if they happen to veer off the extent of the overview frame. The zoom co-ordinate of the ROI is not predicted in this way because our rendering system allows a small number of discrete zoom factors. Instead, we choose to predict the zoom z_{n+d} at frame $n+d$ as the observed zoom z_n at frame n.

2) Kanade-Lucas-Tomasi (KLT) Feature Tracker Based Predictor: This algorithm does exploit the motion information in the buffered overview video frames. As shown in the processing flowchart in Fig. 4, we first apply the Kanade-Lucas-Tomasi (KLT) feature tracker [14] to perform optical flow estimation on the buffered overview video frames. This yields the trajectories of a large (but limited) number of feature points from frame n to frame $n+d$. The trajectory predictor then incorporates these feature trajectories into the ROI prediction for frame $n+d$. We now describe the KLT feature tracker briefly and then the trajectory predictor.

We modify the implementation of the KLT feature tracker that is available in [15] such that it begins by analyzing frame n and selecting a specified number of the most suitable-to-track feature windows. The question

of which feature windows are most suitable for tracking has been extensively studied, for example in [10] and [12]. The selection of these feature windows in our implementation is as described in [14] and it avoids flat areas and single edges and prefers corner-like features. Then it solves the Lucas-Kanade equation for each selected window in each subsequent frame up to frame $n+d$. Figs. 5 and 6 show the features tracked from frames 250 to 259 of the *Tractor* and *Sunflower* sequences, respectively. Note that most (but not all) feature trajectories are propagated to the end of the buffer.

The trajectory predictor uses these feature trajectories to make the ROI prediction at frame $n+d$. Among the features that survive from frames n to $n+d$, the trajectory predictor finds the one nearest the center of the ROI in frame n. It then follows two basic prediction strategies. The centering strategy predicts spatial co-ordinates $\hat{p}_{n+d}^{centering}$ of the ROI to center that feature in frame $n+d$. The stabilizing strategy predicts spatial co-ordinates $\hat{p}_{n+d}^{stabilizing}$ that keep that feature in the same location with respect to the display. The eventual ROI prediction is blended from these two predictions according to a parameter β:

$$\hat{p}_{n+d}^{blended} = \beta \hat{p}_{n+d}^{centering} + (1-\beta)\hat{p}_{n+d}^{stabilizing}. \quad (3)$$

As in the ARMA model predictor, the zoom z_{n+d} is predicted as the zoom z_n and the predicted spatial co-ordinates of the ROI are cropped if they veer off the extent of the overview frame.

3) H.264/AVC Motion Vectors Based Predictor: This prediction algorithm exploits the motion vectors contained in the encoded frames of the overview video already received at the client. From the center pixel of the ROI in frame n, we use the motion vectors to find a plausible propagation of this pixel in every subsequent frame up to frame $n+d$. The location of the propagated pixel in frame $n+d$ is the center of the predicted ROI. The zoom factor z_{n+d} is predicted as the zoom factor z_n.

We tried three algorithms for implementing the pixel propagation from one frame to the next frame. Imagine there is a pixel p_n in frame n, which we want to propagate to frame $n+1$.

- Algorithm 1: Choose any pixel in frame $n+1$, which is temporally connected to p_n via its motion vector. Note that this simple approach is not robust enough and the pixel might drift out of the object that needs to be tracked.
- Algorithm 2: In addition to finding a temporally connected pixel for p_n also find one temporally connected pixel for each of the pixels in frame n, which are in the spatial 4-connected neighborhood of p_n. This is shown in Fig. 7. Out of these five pixels in frame $n+1$, choose that pixel which minimizes the sum of the squared distances to the remaining four pixels in frame $n+1$.
- Algorithm 3: Find the five pixels in frame $n+1$ similar to algorithm 2. Out of these five pixels in frame $n+1$, choose that pixel which has the minimum squared difference in intensity value compared to p_n.

Note that in the algorithms above, for any pixel, if there exists no connection via motion vectors then the colocated pixel is declared as the connected pixel. Also note that if there are multiple connections then one connected pixel is chosen randomly.

4) Median Predictor: The different predictors described above are quite diverse and every predictor characteristically performs very well under specific conditions. The conditions are dynamic while watching a video sequence in an interactive session. Hence, we combine several of the above predictors by choosing the median of their predictions, separately in each dimension. This choice guarantees that for any frame interval, if one of the predictors performs particularly badly compared to the rest, then the median operation does not choose that predictor.

B. Tracking Mode Algorithms

Algorithms for the tracking mode differ from those for the manual mode in two significant ways. Firstly, these algorithms cannot expect ongoing ROI information as input from the user. Instead a single click on a past frame indicates which object to track in the scene. (Notice though that the user may still control the zoom factor for better viewing of the object.) Consequently, the KLT feature tracker and H.264/AVC motion vectors based predictors are modified, and the ARMA model based predictor is ruled out because it is altogether agnostic of the video content. The second difference is that the predicted ROI trajectory in tracking mode is actually presented to the user. This imposes a smoothness requirement on the ROI trajectory for pleasant visual experience.

1) Kanade-Lucas-Tomasi (KLT) Feature Tracker Based Predictor: Just as for the manual mode, we employ the KLT feature tracker to extract motion information from the buffered overview frames. The trajectory predictor again produces a blended ROI prediction from centering and stabilizing predictors. In the absence of ongoing ROI information from the user, both of these predictors begin by identifying the feature nearest the user's initial click. Then they follow that feature trajectory into future frames, centering it or keeping it in the same location, respectively. Whenever the feature being followed disappears during propagation, these predictors start following the surviving feature nearest to the one that disappeared at the time of disappearance.

Note that the centering strategy introduces jerkiness into the predicted ROI trajectory each time the feature being followed disappears. On the other hand, the stabilizing predictor is designed to create very smooth trajectories but runs the risk of drifting away from the object selected by the user. The blended predictor trades off responsiveness to motion cues in the video and its trajectory smoothness via parameter β.

2) H.264/AVC Motion Vectors Based Predictor: The user's click at the beginning of the tracking mode indicates the object to be tracked. As described earlier, the

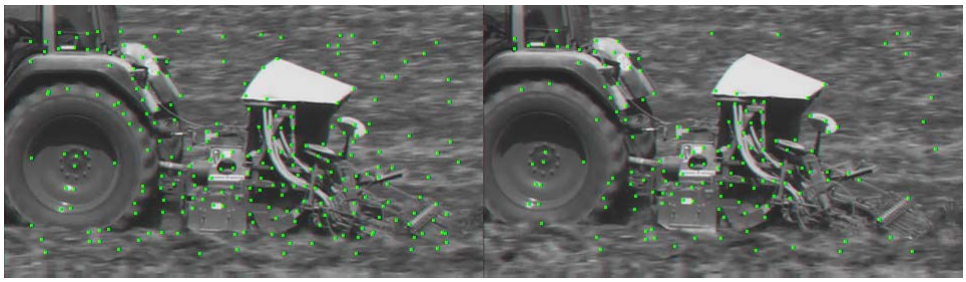


Fig. 5. Features propagated from frame 250 (left) of the *Tractor* sequence to frame 259 (right)

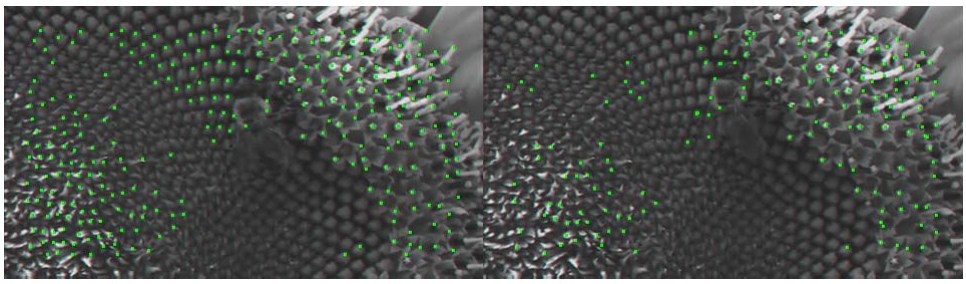


Fig. 6. Features propagated from frame 250 (left) of the *Sunflower* sequence to frame 259 (right)

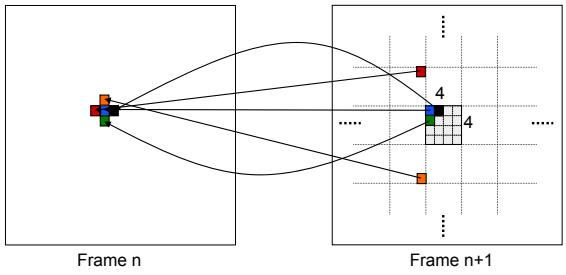


Fig. 7. There are five temporally connected pixels in frame $n+1$. These are connected via their respective motion vectors to p_n and the four spatial 4-connected neighbors of p_n.

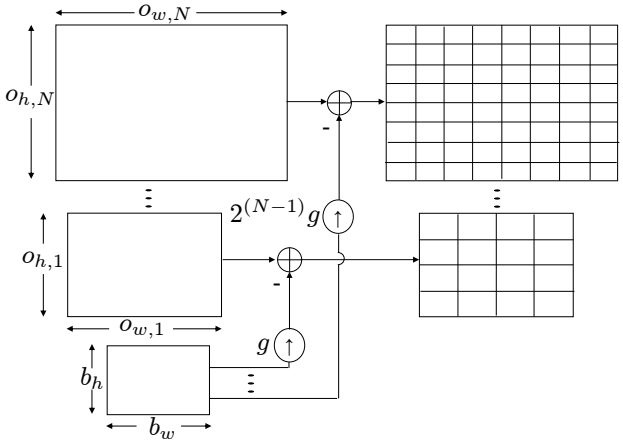


Fig. 8. Coding Scheme: The base layer is coded using H.264/AVC without B frames. The video signal corresponding to every zoom factor is coded using P slices. For doing so, the signal reconstructed from the base layer is upsampled by an appropriate factor and used as prediction. (Note: Resolutions corresponding to successive zoom factors increase dyadically. The figure is not drawn to scale and hence does not reflect this.)

motion vectors sent by the server for the overview video frames are used for propagating the point indicated by the user into the future frames. This predictor always sets the propagated point as the center of the rendered ROI. Notice that this is different from the KLT feature tracker based predictor in which the tracked feature might disappear.

IV. Results

Our experiments use three high resolution video sequences, *Sunflower*, *Tractor* and *Cardgame*. The *Sunflower* sequence of original resolution 1920x1088 shows a bee pollinating a sunflower. The bee moves over the surface of the sunflower. There is little camera movement with respect to the sunflower. In the *Tractor* sequence of original resolution 1920x1088, a tractor is shown tilling a field. The tractor moves obliquely away from the camera, and the camera pans to keep the tractor in view. The *Cardgame* sequence is a 3584x512 pixel 360° panoramic video stitched from several camera views. The camera setup is stationary and only the card game players move.

For *Sunflower* and *Tractor*, we provision for 3 dyadic levels of zoom with the ROI display being 480x272 pixels. The overview video is also 480x272 pixels. For *Cardgame*, there are 2 levels of zoom with the ROI display being 480x256 pixels. The overview video is 896x128 pixels and provides an overview of the entire panorama.

We employ the video coding scheme that we proposed in [2]. It is explained here in brief. Let the original video be o_w pixels wide and o_h pixels tall. Since every zoom-out operation corresponds to downsampling by 2 both horizontally and vertically, the input to the coding scheme is the entire scene in multiple resolutions; available in dimensions ($o_{w,i} = 2^{-(N-i)}o_w$ by $o_{h,i} = 2^{-(N-i)}o_h$) for $i = 1 \ldots N$, where N is the number of zoom factors. As

shown in Fig. 8, we first encode the overview video (b_w by b_h) using H.264/AVC without B frames. Notice that 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 used as prediction signal for encoding video of dimensions ($o_{w,i}$ by $o_{h,i}$), where $i = 1 \ldots N$ 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. 8, 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.

Four user ROI trajectories were captured for these sequences, one for *Sunflower*, two for *Tractor* and one for *Cardgame*. All trajectories begin at the lowest zoom factor. The *Sunflower* trajectory follows the bee, at zoom factors 1, 2 and 3 in succession. Trajectory 1 for *Tractor* follows the tractor mostly at zoom factor 2, and trajectory 2 follows the machinery attached to the rear of the tractor mostly at zoom factor 3. The *Cardgame* trajectory follows the head of one of the players mostly at zoom factor 2.

The right-click of the mouse was used to select and deselect the tracking mode. The user made a single object-selection click in the middle of each trajectory on the respective object, since the tracked object was always present thereafter until the end of the video. In each of the four ROI trajectories above, in spite of selecting the tracking mode, the user continued to move the mouse as if it were still the manual mode. The mouse coordinates were recorded for the entire sequence. This serves two purposes:

- evaluation of the manual mode predictors over a greater number of frames.
- comparison of the tracking capability of the tracking mode predictors with a human operator's manual tracking.

A. Manual Mode Results

The manual mode predictors are evaluated based on the distortion per rendered pixel they induce in the user's display for a given ROI trajectory through a sequence. If the ROI prediction is perfect then the distortion in the rendered ROI is only due to the quantization at the encoder. A worse prediction implies that more pixels of the ROI are rendered by upsampling the colocated pixels of the overview video. In the worst case, when no correct slices are pre-fetched, the entire ROI is rendered via upsampling from the base layer. This corresponds to the upper bound on the distortion. It should be noted that due to the encoding being in slices, there is some margin for error for any ROI predictor. This is because some excess number of pixels are sent to the client depending on the coarseness of the slice grid and the location of the predicted ROI. The slice size for our experiments is 64x64 pixels, except for zoom factor of 1 for *Cardgame*, where it is 64x256 pixels. This is because there is no vertical translation of the ROI possible for zoom factor of 1 for *Cardgame* and hence no slices are required in the vertical direction. (For zoom factor of 1 for *Sunflower* and *Tractor*, no spatial random access is needed.)

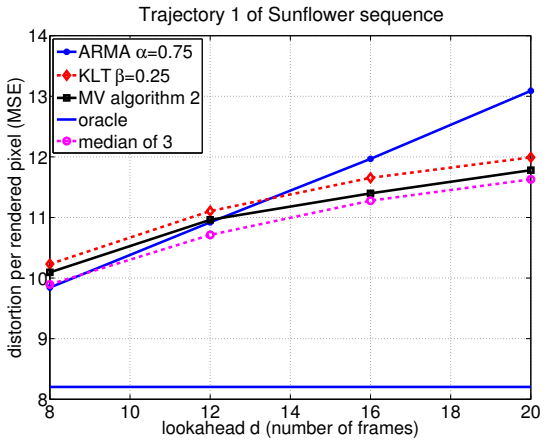


Fig. 9. Distortion per rendered pixel versus lookahead d for manual mode predictors for *Sunflower* trajectory 1

We simulated three different ARMA model based predictors with $\alpha = 0.25$, 0.5 and 0.75, respectively. Among these, the $\alpha = 0.75$ ARMA model based predictor yielded the lowest distortion per rendered pixel. The KLT feature tracker based predictor was set to track 300 features per frame and was tested with $\beta = 0$, 0.25, 0.5 and 1. Note that $\beta = 1$ corresponds to the centering strategy and $\beta = 0$ to the stabilizing strategy. The blended predictor with $\beta = 0.25$ was the best manual mode predictor in this class. Among the three predictors based on H.264/AVC motion vectors, MV algorithm 2 (which selects the pixel that minimizes the sum of squared distances from the other candidate pixels) performed best in the manual mode.

Figs. 9, 10 and 11 show the distortion per rendered pixel due to the best predictor in each category as the lookahead d varies, for the *Sunflower* and *Tractor* ROI trajectories. Furthermore, the performance of the median predictor that combines these three basic predictors is presented. Also shown is the lower bound on distortion resulting from a prediction oracle. These plots demonstrate that the relative performance of the basic predictors varies significantly depending on video content, user's ROI trajectory and number of lookahead frames. The median predictor, on the other hand, is often better than its three constituent predictors and is always much better than the worst.

For the *Sunflower* sequence, the distortion when no correct slices are pre-fetched is about 30 for zoom factor of 2 and 35 for zoom factor of 3. For the *Tractor* sequence, these numbers are about 46 and 60, respectively. There is no random access for zoom factor of 1. The distortion induced in the rendered ROI by the proposed predictors is closer to that of perfect ROI prediction and hence we omit the distortion upper bounds in the plots.

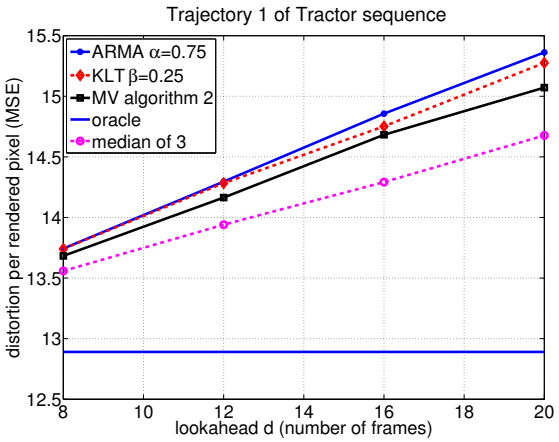


Fig. 10. Distortion per rendered pixel versus lookahead d for manual mode predictors for *Tractor* trajectory 1

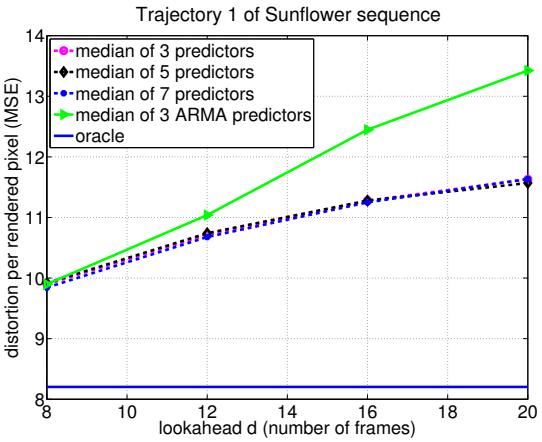


Fig. 12. Distortion per rendered pixel versus lookahead d for manual mode median predictors for *Sunflower* trajectory 1

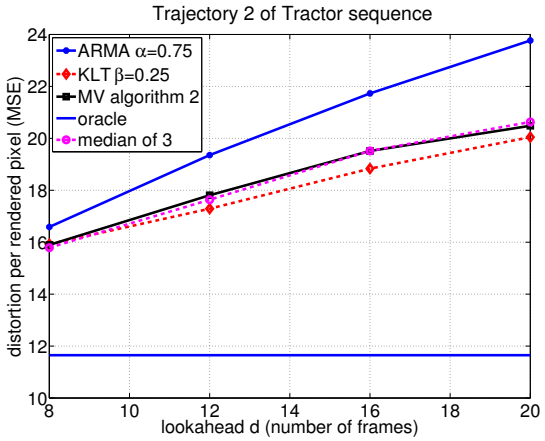


Fig. 11. Distortion per rendered pixel versus lookahead d for manual mode predictors for *Tractor* trajectory 2

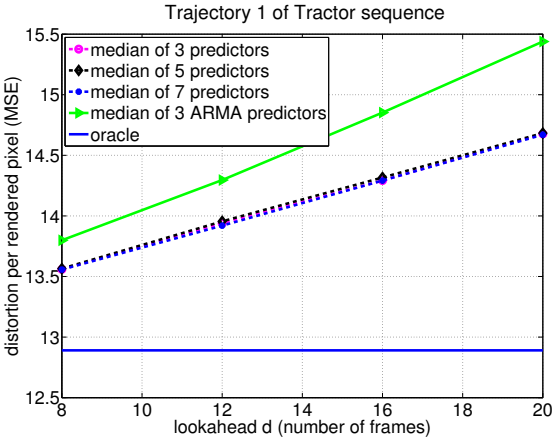


Fig. 13. Distortion per rendered pixel versus lookahead d for manual mode median predictors for *Tractor* trajectory 1

In Figs. 12, 13 and 14, the performance of several median predictors is compared, again for the *Sunflower* and *Tractor* ROI trajectories. The median of 3 predictor is the same as that in Figs. 9, 10 and 11; that is, the median of the ARMA model predictor with $\alpha = 0.75$, the KLT feature tracker predictor with $\beta = 0.25$ and the MV algorithm 2 predictor. The median of 5 predictor additionally incorporates the KLT feature tracker predictor with $\beta = 0.5$ and the MV algorithm 3 predictor. The median of 7 predictor additionally incorporates the ARMA model predictors with $\alpha = 0.25$ and 0.5. A content-agnostic median predictor is also considered; it combines only the three ARMA model predictors of $\alpha = 0.25$, 0.5 and 0.75. As before, the lower bound on distortion resulting from a prediction oracle is shown. These plots show that the content-agnostic median predictor performs consistently worse than the median predictors that do make use of the video content. Moreover, this effect is magnified as the lookahead increases. Another observation is that increasing the number of predictions fed into the median predictor seems to improve performance, but only marginally. This is perhaps because the additional basic predictions are already correlated to existing predictions. We speculate that significant gains are only possible if we devise novel basic predictors and incorporate them into the median predictor.

B. Tracking Mode Results

In the tracking mode, the predicted ROI trajectory is pre-fetched and actually displayed at the client. So the evaluation of tracking mode prediction algorithms is purely visual since the user experiences no distortion due to concealment. The predicted ROI trajectory should be both accurate in tracking and smooth. Since the user is not required to actively navigate with the mouse, the ARMA model based predictors cannot be used. Instead we apply various KLT feature tracker based predictors set to track 300 features per frame, and the H.264/AVC motion vectors based predictors.

We test the KLT feature tracking predictors with $\beta = 0$, 0.25, 0.5 and 1 on trajectory 2 of the *Tractor* sequence. The centering strategy ($\beta = 1$) accurately tracks the tractor machinery through the sequence, because it centers the feature in frame $n + d$ that was nearest the center in frame n. But it produces a visually-unappealing jerky trajectory whenever the central feature disappears. On the

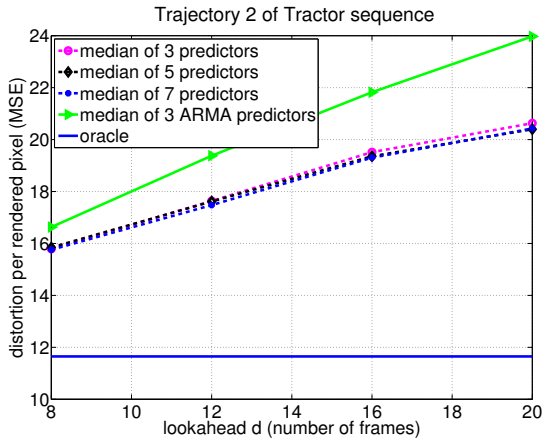


Fig. 14. Distortion per rendered pixel versus lookahead d for manual mode median predictors for *Tractor* trajectory 2

other hand, the stabilizing ($\beta = 0$) produces a smooth trajectory because it keeps the nearest-to-center feature in frame n in the same location in frame $n + d$. The drawback is that the trajectory drifts away from the tractor machinery. Subjective experimentation suggests that the best compromise is achieved by the blended predictor with parameter $\beta = 0.25$. This blended predictor also works well for the trajectory recorded for the *Cardgame* sequence, but fails to track the bee in the *Sunflower* sequence.

Tracking with the H.264/AVC motion vectors of the buffered overview video proves to be much more successful. MV algorithm 3 (which selects the candidate pixel that minimizes the pixel intensity difference) is particularly robust. In addition to the four recorded trajectories, we tried tracking several points in each video sequence starting from the first frame. For example, we successfully tested the MV algorithm 3 for tracking various parts of the tractor. Figs. 16 and 15 show the tracking of a single point over hundreds of frames of video. In fact, the tracking is so accurate that the displayed ROI trajectory is much smoother than a manual trajectory generated by mouse movements. Factors like camera motion, object motion, sensitivity of the mouse, etc., pose challenges for a human to track an object manually by moving the mouse and keeping the object centered in the ROI. Automatic tracking can overcome these challenges.

V. Conclusions

We propose and evaluate several ROI predictors for interactively streaming regions of high resolution video. ROI prediction enables pre-fetching which is crucial for low latency of interaction despite the delay of packets on the network.

The literature on motion estimation and tracking consists of a rich variety of algorithms that lend themselves to the proposed video-content-aware ROI prediction. A few illustrative algorithms are evaluated in this paper and their performance is compared to conventional video-content-agnostic ROI prediction.

In the manual mode of operation, we demonstrate that exploiting the motion information in the buffered overview frames to assist pre-fetching can reduce the distortion experienced by the user. It is clear, however, that the best prediction strategy depends on both the trajectory and the video content. The median predictor provides an appealing framework for profiting from the diversity of various predictors that are geared towards exploiting different portions of the available information.

The tracking mode of operation greatly enhances system usability by relieving the human operator of some navigation burden. We show that tracking objects is feasible, not only using sophisticated optical flow estimation on the buffer of overview frames, but simply by exploiting H.264/AVC motion vectors already available in the buffered bitstream. Extending the proposed tracking algorithms to work with B frames and multiple hypotheses of motion-compensated prediction in general is part of future research.

In this paper, we assume that the ROI prediction is performed at the client's side. If this task is moved to the server, then slightly stale mouse co-ordinates would be used to initialize the ROI prediction since these then need to be transmitted from every client to the server. Also the ROI prediction load on the server increases with the number of clients. However, in this case, the server is not restricted to use low resolution frames for the motion estimation. The server can also have several feature trajectories computed beforehand to lighten real-time operation requirements.

Acknowledgments

We thank Dr. John Apostolopoulos of Hewlett-Packard Laboratories for useful discussions. We thank the Stanford Center for Innovations and Learning (SCIL) for providing the panoramic video sequence.

References

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

[2] 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," *Proc. of 15th European Signal Processing Conference (EUSIPCO), Poznan, Poland*, September 2007, in press.

[3] "OpenGL Application Programming Interface," Website: http://www.opengl.org.

[4] P. Ramanathan, "Compression and interactive streaming of light-fields," March 2005, Doctoral Dissertation, Department of Electrical Engg., Stanford University, Stanford CA, USA.

[5] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, "A receiver-driven multicasting framework for 3DTV transmission," *Proc. of 13th European Signal Processing Conference (EUSIPCO), Antalya, Turkey*, September 2005.

[6] ——, "Interactive transport of multi-view videos for 3DTV applications," *Journal of Zhejiang University , SCIENCE A 7(5)*, 2006.

[7] K. Hariharakrishnan and D. Schonfeld, "Fast object tracking using adaptive block matching," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 853–859, October 2005.

[8] J. Lee, K. Rhee, and S.Kim, "Moving target tracking algorithm based on the confidence measure of motion vectors," *Proc. of IEEE Intl. Conference on Image Processing (ICIP), Thessaloniki, Greece*, vol. 1, pp. 369–372, October 2001.

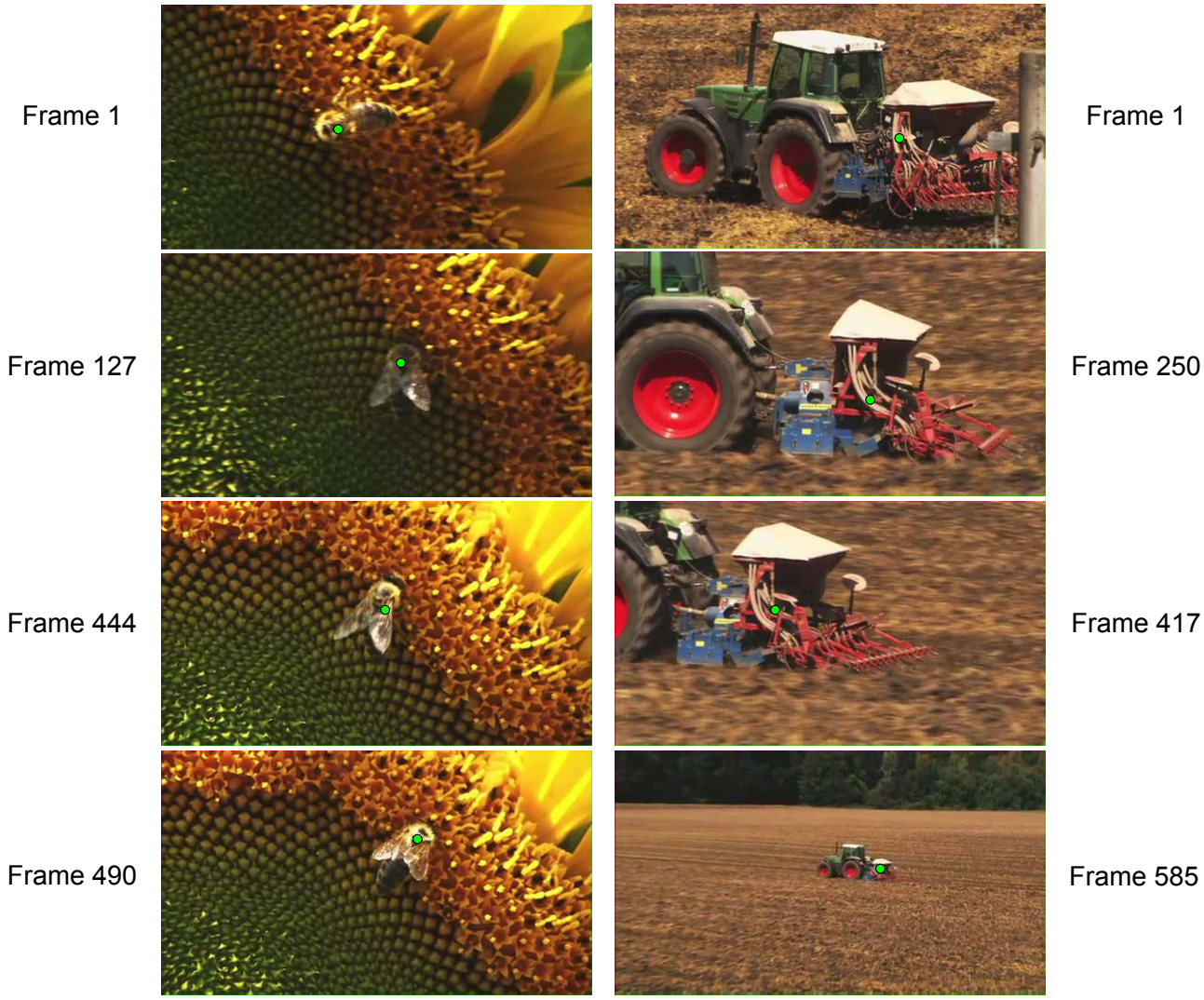


Fig. 15. Tracking using H.264/AVC motion vectors based prediction algorithm 3 for *Sunflower* (500 frames) and *Tractor* (600 frames). The point to be tracked can be seen in the first frame of the respective sequence. The tracked point is highlighted for better visibility.

[9] S. Park and J. Lee, "Object tracking in MPEG compressed video using mean shift algorithm," *Proc. of IEEE Intl. Conference on Information and Communications Security (ICICS), Singapore*, vol. 2, pp. 748–752, December 2003.

[10] J. Shi and C. Tomasi, "Good features to track," *Proc. of IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA*, pp. 593–600, June 1994.

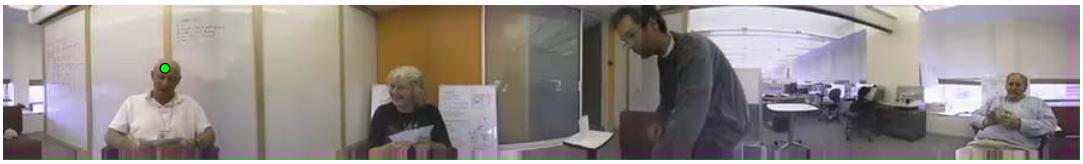
[11] A. Gyaourova, C. Kamath, and S. Cheung, "Block matching for object tracking," October 2003, LLNL Technical Report, UCRL-TR-200271.

[12] C. Kamath, A. Gezahegne, S. Newsam, and G. M. Roberts, "Salient points for tracking moving objects in video," *Proc. of SPIE Conference on Image and Video Communications and Processing, San Jose, CA, USA*, vol. 5685, pp. 442–453, January 2005.

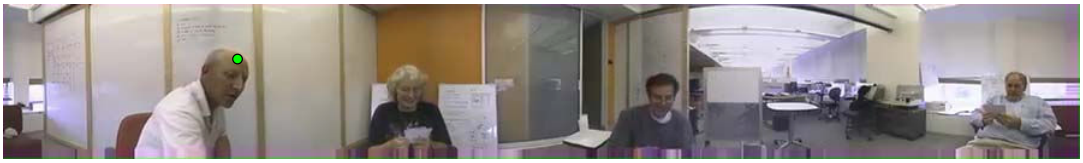
[13] K. Takaya and R. Malhotra, "Tracking moving objects in a video sequence by the neural network trained for motion vectors," *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 153–156, August 2005.

[14] C. Tomasi and T. Kanade, "Detection and tracking of point features," April 1991, Carnegie Mellon University Technical Report CMU-CS-91-132.

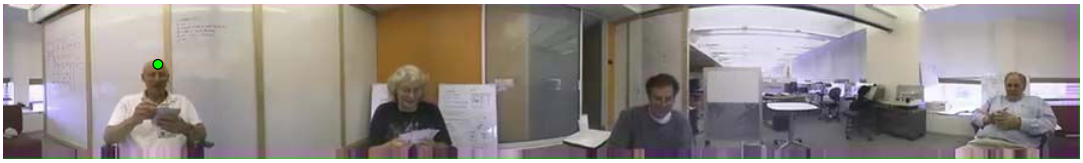
[15] "KLT: Kanade-Lucas-Tomasi Feature Tracker," Website: http://www.ces.clemson.edu/∼stb/klt/index.html.



Frame 1



Frame 232



Frame 271

Fig. 16. Tracking using H.264/AVC motion vectors based prediction algorithm 3 for *Cardgame* (300 frames). The point to be tracked, shown in the first frame, is on the forehead of one of the players. The tracked point is highlighted for better visibility.