Abstract—We describe a method for dynamically variable region-of-interest editing of MPEG-2 video streams in the compressed domain to reduce bandwidth. This method is used in our second-generation mutually-immersive mobile telepresence system, where only the region containing a user's head in an acquired video stream needs to be displayed at the remote location. This editing is performed in a way that leaves the resulting stream consistent with the MPEG-2 standard. We currently reduce the transmission of data by skipping macroblocks or deleting DCT coefficients outside the region of interest in P and B pictures, and deleting DCT coefficients in I frames. This results in a savings of approximately 50% of the bandwidth of the original stream for our application. Since we are transmitting four video streams to the surrogate over our wireless network, the total bandwidth is reduced by about 5Mbps. As an added benefit, the CPU requirements for decoding the edited stream in software are also reduced by approximately 16%.

I. INTRODUCTION

Mutually-Immersive Mobile Telepresence uses a teleoperated robotic surrogate to visit remote locations as a substitute for physical travel [1]. Its goal is to recreate to the greatest extent practical, both for the user and the people at the remote location, the sensory experience relevant for business interactions of the user actually being in the remote location. The system includes up to 360 degree surround CD-quality audio[2] and 360 degree surround high-resolution video at both the remote and user's locations, preserves gaze and eye contact, and presents local and remote participants to each other at life size. We have recently completed our second-generation system surrogate (see Figure 1).

In order to provide the user of the system freedom of movement in the display cube at their location, we capture close to a 90 degree field of view of the user's head from each corner of the display cube (see Figure 2). Then in combination with positional data obtained from head tracking, we crop the video for display of the user's head on the head of the surrogate.

In our second-generation system we send four bidirectional 704x480 MPEG-2 coded video streams at rates of up to 30fps between the surrogate and the user station. These streams stress even modern 802.11a wireless networks. (802.11a networks currently support peak data rates between 54Mbps to 6Mbps depending on the range.) Since we are only interested in the user's head at the surrogate end, we can improve network bandwidth usage by transmitting only the relevant portions of the video (i.e., data containing a classic portrait cropping of the user's head). The coordinates of the Region of Interest (ROI) vary depending on the position of the person in the display cube, and are constantly changing as the person moves. The head tracking subsystem updates the coordinates of the ROI at video frame rates.

In order to reduce the computational load compressing the four video streams, hardware MPEG-2 encoder PCI cards are used in PCs to capture and compress the video. (We decode the streams in software, since the computational requirements of decoding are less than encoding, and each PC only has a limited number of PCI slots.) The video cameras are connected directly to the hardware encoder PCI cards, and the compressed video is transferred over the network by the PCs. We have not found any hardware encoder card that supports ROI cropping in real time in the pixel domain. Moreover, in an MPEG-2 coded video stream, the height and width information is coded in the sequence header which is part of the highest syntactic structure. This would necessitate starting a fresh sequence whenever the ROI coordinates change. Thus, if it was possible to crop the video in the pixel domain with different
dimensions at different time instances according to the ROI coordinates, the resulting MPEG-2 stream would consist of many concatenated sequences. Generation of such a video stream and its decoding is not easy to implement, since the behavior of the decoder when encountering such a stream is not well defined in the MPEG-2 standard.

Our solution to the problem of dynamically varying ROI editing of video is to compress the entire video from the camera at full frame size using the encoding hardware, and then edit the stream in the compressed domain to remove or reduce video data outside the ROI. This reduces the data transmitted over the wireless network in our telepresence system. This method does not alter the stream properties, so any decoder conforming to the standard can decode the ROI edited video stream. We implemented and tested our method in our real-time telepresence video streaming environment. Experimental results have shown that the amount of data transmitted is reduced significantly. Furthermore, the CPU usage of the software decoder at the far end is also reduced. This is due to the fact that the ROI edited stream presents less data to the decoder to decode.

II. REGION OF INTEREST EDITING OF VIDEO

In this section we present our method for ROI editing of MPEG-2 video streams. We have evaluated our method for nonscalable profiles at 4:2:0 format. For more information on MPEG-2 and details of bitstream structure refer to [3], [4], or [5]. For an overview of prior compressed domain video processing, please refer to [6].

In order to explain our ROI editing method, the picture of a video frame along with a typical ROI is shown in Figure 2. A possible position of the ROI in terms of slices and macroblocks is shown in Figure 3. The ROI consists of a set of whole macroblocks and partial macroblocks. The video data to be removed is found in regions above, below, right and left of the ROI. Since a macroblock is a syntactic structure in the coded stream, all data associated with the macroblocks that are either partially or fully included in the ROI is required to be preserved. Conceptually, the data corresponding to the macroblocks completely outside the ROI can be removed, since they contain video data not required at the surrogate display. However, the blocks which are not needed may still have implications on the coded bitstream structure.

The general philosophy of our approach is to retain the original structure and properties of the coded stream, and remove as much data as possible from regions outside the ROI. The width, height and other information coded in the sequence header is not altered.

We use two methods to remove data from the unused regions: Skipping Macroblocks (SkipMB) and Deleting DCT Coefficients (DeleteDCT). These operations are performed in the compressed domain on MPEG-2 streams. SkipMB can be applied on P and B frames, but not on I frames. DeleteDCT can be applied on I, P, and B frames. When a decoder decodes such a modified stream, it will be able to decode it without any trouble because the modifications ensure the standard conformance of the stream. However, in the video obtained by decoding the ROI edited stream, regions outside the ROI may contain invalid (corrupt) data. This corruption of data is not an issue, since the video outside the ROI is not used for display after decoding. Details of each type of editing are discussed next.

A. Editing P and B Pictures by Skipping Macroblocks Above and Below the ROI

The macroblocks in the regions completely above and below the ROI are contained in one or more complete slices from the left edge to the right edge of the video frame. These slices can be easily located in the MPEG-2 stream by locating the slice start code. The slice start code indicates the vertical position and thus whether it is inside or outside the ROI. Theoretically slices completely outside the ROI can be completely removed. However, complete removal of a slice alters the structure of the video stream and hence cannot be used in practice. According to the rules of the MPEG-2 standard, the first and last macroblocks in a slice should be preserved (i.e., they cannot be skipped macroblocks).
In the slices completely above and below the ROI, the macroblocks except the first and last are skipped. For every macroblock skipped, the macroblock address increment of the next coded macroblock is increased by one. When all the macroblocks in a slice are skipped except the first and last, the macroblock address increment of the last macroblock is modified accordingly. The beginning of macroblock data is identified by the fixed length macroblock escape (if it exists) and the VLC corresponding to the macroblock address increment, since the macroblock data doesn't begin with a start code or with a header. In order to implement the SkipMB method the stream must be partially decoded as specified in the standard. Partial decoding is required to identify the beginning of each macroblock. VLC decoding of DCT coefficients is also required to move through a block's data to identify its end and to locate the beginning of the next macroblock. However, steps like reverse zig-zag scanning, de-quantization, and IDCT are not performed.

This approach helps to remove the majority of the data associated with the slices above and below the ROI. As a result the actual macroblock structure of these slices are altered. For example, an intra-coded macroblock within the slice is also skipped. The motion vectors associated with the macroblocks which are skipped are lost. When a decoder encounters parts of a video stream modified through SkipMB, it reconstructs the video in the skipped macroblocks from previously decoded frames through prediction. As a result regions corresponding to these macroblocks in the reconstructed video are corrupt, but are eventually discarded at the display as it is outside the ROI and will not be visible.

When macroblocks are removed this way, it may cause problems for the macroblocks whose motion vectors point to these skipped macroblocks. We want to ensure that all data inside the ROI can be decoded correctly. However, it is possible that the motion vector of a macroblock inside the ROI points to a region outside the ROI. To overcome this problem, portions of data in the immediate neighborhood of the ROI are left untouched. This guard ring of pixels surrounding the ROI also needs to be decoded correctly. In our telepresence system, there is only one object of interest (the user's head) and its motion is not normally fast. Experiments found that guard ring of 2 slices above and below works well under normal conditions.

B. Editing P and B Pictures by Skipping Macroblocks to the Right of the ROI

In this case, all macroblocks that fall either fully or partially inside the ROI are retained and remaining macroblocks on the right side are skipped except the last one. Macroblocks on the right side are skipped using the SkipMB method as described in the previous section. To create the guard ring of pixels (to support motion vectors pointing outside the ROI), a few macroblocks after the ROI boundary on the right side are also retained. The macroblock address increment of the last macroblock is updated to reflect the number of skipped macroblocks.

C. Editing P and B Pictures by Deleting DCT Coefficients to the Left of the ROI

Portions of the video in the coded stream corresponding to the left side of the ROI pose unique problems, since there is predictive coupling for the motion vectors from macroblocks on the left side to the right side in a slice. Thus if a macroblock is skipped, the motion vectors of the subsequent macroblocks that are inside the ROI are not decoded correctly. Therefore the contents of the macroblock on the left side of a ROI should be retained while the actual pixel data coded through DCT coefficients can be modified. To reduce data in unwanted regions on the left side of a ROI, we delete DCT coefficients in all blocks with coded data. A guard ring of pixels is also maintained in this case by keeping a few unedited macroblocks on the left side of the ROI. All other blocks in a macroblock that have coded data are modified.

Each macroblock consists of six blocks. In each block the DCT coefficients after quantization are VLC coded in the run, level, and sign format. The End of Block (EOB) code defined in the standard is used to indicate the end of DCT coefficients. For these blocks, the first VLC coded data (DC coefficient) is retained, the rest of the VLC coded data is deleted, and finally the EOB code is retained. According to the standard, the first data in a block can not be EOB. So we keep a VLC code and an EOB in such blocks. When a decoder decodes the stream modified this way, after the EOB it fills any remaining DCT coefficients with zeroes. So the decoded video in these regions is corrupt, but it does not matter since this part of the video is discarded before display.

It is also possible to apply this method on the first and last macroblocks that remain after SkipMB is applied in the slices above and below the ROI. However, we have not implemented this yet since the savings are relatively minor.

D. Editing I Pictures by Deleting DCT Coefficients for Regions Outside the ROI

I frames have only intra coded macroblocks, and it is not possible to skip macroblocks on I pictures. However, deleting DCT coefficients can be applied in the regions outside the ROI on I frames, similar to our editing of P or B frames. We applied DeleteDCT on all blocks in the macroblocks in the I frame outside the ROI with an initial guard ring of 2 macroblocks. Only the DC coefficient is retained and all the AC coefficients are deleted until the EOB.

E. Telescopic Reduction of Guard Rings

The guard ring provides a reserve so that future motion out of the guard ring into the ROI can be handled correctly. Note that there is no use in having a guard ring on the last P frame in an IPPP GOP. Furthermore, assuming motion is relatively constant during the time of the GOP, on each successive frame in the GOP the guard ring could be reduced in size. Ideally the initial guard ring for the I-frame should be positioned and sized based on the amount of motion in the ROI. However this is more complicated, and jitter and other artifacts in the ROI coordinate stream can cause difficulties for this approach.
Instead we use a simple symmetric fixed initial guard ring of width 2 with telescopic reduction. Specifically, for a GOP length of 7 we use guard rings of 2, 2, 2, 2, 1, 1, and 0 macroblocks in the first through seventh frames of the GOP. We have found that this works well in practice.

III. IMPLEMENTATION AND RESULTS

The ROI editing methods described above were implemented in C++, leveraging an existing MPEG-2 decoder. All the structures used in the decoder were not changed. Decoding of the stream was restricted to partial decoding as required by the method.

In the display cube station of the telepresence system there are cameras mounted in the seams between adjacent screens. When the user is closer to one camera, his or her head occupies a large portion in the video captured by that camera. In this case, the corresponding ROI is also large and the amount of data that can be deleted is less. When the user is further away from a camera, his or her head occupies relatively less area in the video, resulting in a smaller ROI. Here relatively large amounts of data can be deleted from regions outside the ROI. Because the cameras are mounted at right angles to each other in the corners of the display cube, when the user is near one camera he or she is far away from the opposite camera. So there will always be a significant savings in data when all four video channels are considered.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Test Conditions</th>
<th>Without ROI Editing</th>
<th>With ROI Editing</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video data rate (kbps)</td>
<td>user in center</td>
<td>2282</td>
<td>1142</td>
<td>50%</td>
</tr>
<tr>
<td>Decoder CPU</td>
<td>user near corner</td>
<td>2065</td>
<td>912</td>
<td>56%</td>
</tr>
<tr>
<td>Editor CPU</td>
<td>user in center</td>
<td>25</td>
<td>21</td>
<td>16%</td>
</tr>
</tbody>
</table>

Table I shows the results obtained by various types of ROI editing in our telepresence environment. These experiments were carried out using PCs based on Intel P4 3.06 GHz CPUs and DDR2 memory systems. A baseline bandwidth of 2.5Mbs per stream with a GOP length of 7 is used. The GOP contains only P frames after the initial I frame. The first row shows the average data rate in kbps of all four streams transmitted to the surrogate with and without ROI editing, when the user is in a central position. The bandwidth savings in this case is 50%. The second row shows the average data rate among all four streams when the user is close to one corner and camera in the display cube. In this case the user also is much farther from one camera and somewhat farther from the two other cameras. Thus one ROI gets larger and three ROIs get smaller. The average bandwidth savings in this case is 56%.

Once the stream is edited, the amount of data to be decoded is also reduced. Because of this there is also a reduction in the CPU requirements of the decoder. The fourth row in Table I shows the CPU usage of the software decoder was reduced from 25% to 21% of the CPU when ROI editing is used.

The last row of Table I shows the CPU usage of the ROI editor when running on a PC at the user's site that is similar to the PC in the surrogate. The CPU usage of the editor is relatively low compared to the usage required for decoding. This is because most of the time the editor is copying data, and it requires relatively little computation. Note that in our telepresence system it is advantageous to move computation from the surrogate to the user station wherever possible, since machines at the user station run off wall power while the PCs in the mobile surrogate have both size and power constraints. Even though the total CPU usage including both the user and surrogate PCs has increased, the fact that ROI editing reduces the surrogate CPU requirements is still a plus.

IV. CONCLUSIONS

We have described a method of editing MPEG-2 video streams in the compressed domain according to a dynamically varying region of interest. This method is used in our second-generation mobile telepresence system, where only the region containing a user's head in an acquired video stream needs to be displayed at the remote location. This editing is performed in a way that leaves the resulting stream consistent with the MPEG-2 standard. We currently reduce the transmission of data by skipping macroblocks or deleting DCT coefficients outside the region of interest in P and B pictures and deleting DCT coefficients in I frames. We surround the ROI in the I frame with a guard ring that gets progressively smaller in successive P frames as the GOP progresses. This results in a savings of 50% of the bandwidth of the original stream for our application. As an extra benefit, the CPU requirements for decoding the edited stream in software are also reduced by approximately 16%.

ACKNOWLEDGEMENTS

The authors wish to acknowledge Sam Liu of HP Labs and the MPEG Simulation Group for permitting to use their MPEG-2 video decoder software. We would also like to thank April Slayden of HP Labs, who implemented the head tracking system and integrated it with the ROI editor.

REFERENCES