Real-Time 3D Video Compression for Tele-Immersive Environments

Zhenyu Yang, Yi Cui, Zahid Anwar, Robert Bocchino, Nadir Kiyanclar
Klara Nahrstedt, Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
{zyang2, yicui, anwar, bocchino, kiyancla}@uiuc.edu
{klara, rhc}@cs.uiuc.edu

William Yurcik
National Center for Supercomputing Applications (NCSA)
byurcik@ncsa.uiuc.edu

FINAL SUBMISSION*

ABSTRACT

Tele-immersive system can improve the productivity and aid communication by allowing distributed parties to exchange information via a shared immersive experience. The *TEEVE* research project at the University of Illinois (UIUC) and the University of California at Berkeley (UC Berkeley) seeks to promote the application of tele-immersive environments by a holistic integration of existing components that capture, transmit, and render three-dimensional (3D) scenes in real time to convey the sense of an immersive space. However, the transmission of 3D videos poses a great challenge. First, it is bandwidth-intensive as multiple large volume 3D video streams have to be transmitted. Second, the video stream contains not only color but also depth information, which requires different treatment. While color information may be compressed in a lossy manner, the depth information should be compressed losslessly. Therefore, a 3D real-time compression algorithm must be deployed to accommodate both the bandwidth requirement and the variety of data,

In this paper, we present and evaluate two compression schemes for compressing 3D video streams containing color and depth information. For the first scheme, we use color reduction followed by background removal to compress the color information, which is then compressed along with the depth information using zlib. For the second scheme, we use motion JPEG to compress the color information and run length (RLE) coding followed by Huffman coding to compress the depth information. The experimental results of 3D videos captured from real tele-immersive environment show that: (1) the compressed data preserves enough information to communicate the 3D images effectively (minimum PSNR > 40) and (2) even without inter-frame motion estimation, very high compression ratios (average > 15) are achievable at speeds sufficient to allow real-time communication (average ≈ 13 ms for each video frame).

Keywords: Tele-Immersion, Real-Time Compression

1. INTRODUCTION

Tele-immersive environments are now emerging[1–3] as the next generation of communication medium to allow distributed users more effective interaction and collaboration in joint activities. To achieve seamless immersive experience, the tele-immersive system must acquire, reconstruct, stream and render realistic video and sound in 3D space and in real time. Accordingly, several basic components of these tele-immersive environments need to be included such as *3D camera array*, *sound system*, *networking* and *displaying system*. Recent research efforts have been made to develop and experiment individual components or tele-immersive environments with partially integrated components.[1–5]

The *TEEVE* (Tele-immersive Environments for EVErybody) project currently carried out by University of Illinois at Urbana-Champaign and University of California at Berkeley serves as a platform to promote a holistic integration,

*This paper has a preliminary submission on July 5th, 2005 which MUST be ignored.

deployment and experimentation with the existing tele-immersive components to explore the possibilities and challenges of deploying this cutting edge technology in a more cost effective manner and across broader audience. The TEEVE system features a *distributed application model* as depicted in Figure 1 where users are placed in distributed spaces and engaged in coordinated physical activities. Each user has a setup of cameras and audio devices around him/her to capture the full body movement. The user input is captured by the capturing tier of the TEEVE system, and then transmitted over the transmission tier to the receiving user where the multi-view/display rendering/displaying tier will allow the viewer to see the physical activity from arbitrary angles and perform it himself/herself accordingly.

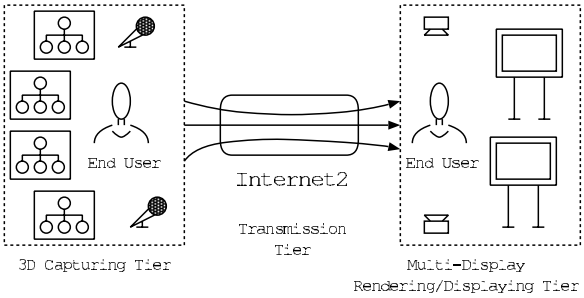


Figure 1. TEEVE Application Model

This paper is focused on the compression techniques of 3D image data whose transmission is very bandwidth intensive. In TEEVE, developing quality, real-time and immersive communication demands a basic bandwidth at the Gbps level. There are several 3D compression schemes proposed in the literature which are highly influenced by the underlying data models, such as the 3D wavelet tranform[6,7] used on *volumetric data* and Edgebreaker[8] used on *triangular meshes*. On the other hand, the tele-immersive system uses the data model of *depth images*[9,10] to capture and reconstruct the 3D scene. At each time instant, a 3D camera of one viewpoint produces a depth image which contains the depth and color information for each pixel and each camera corresponds to one 3D video stream. Multiple depth images generated from various viewpoints under a global coordinate system are then aggregated to render the 3D scene. Due to the difference of data models, previous compression schemes cannot be directly and efficiently applied in this data model. Another issue is that most of those schemes assume the availability of the 3D model before compression, which no longer holds in the tele-immersive environment where the 3D data are distributed.

Sang-Uok Kum et al. have pioneered in the research of 3D depth image compression and proposed a real-time inter-stream compression scheme[11,12] where multiple depth streams are analyzed to remove the redundant points. In contrast to their approach, our work concentrates on investigating intra-stream compression schemes, which are important in at least two scenarios as compared with the inter-stream compression. First, the intra-stream compression can be used in the initial data collection stage (followed by inter-stream compression) to save local network traffic. More important, after each stream is compressed using inter-stream compression, the remaining non-redundant pixels can be further compressed with the intra-stream compression before the remote transmission starts. As shown in the paper, our intra-stream compression scheme is complementary to the earlier work of inter-stream compression and can be incorporated for overall performance enhancement.

Our contribution in this paper is to present the design, implementation, and evaluation of two intra-stream compression schemes to be used in the tele-immersive environments for compressing 3D depth images containing color and depth information in real time. To the best of our knowledge, we are the first one to consider the design of a comprehensive intra-stream compression scheme for tele-immersion systems. For the first scheme, we apply color reduction to reduce the number of bits representing the color information. After that, the image is further processed to remove the background pixels, and compressed using zlib together with the depth information. In the second scheme, we use motion JPEG to compress the color information and run length encoding followed by Huffman coding to compress the depth information. Our experimental results show that the compressed data preserves enough information to communicate the 3D images effectively. We also show that even without exploiting the temporal redundancy, very high compression ratios are achievable at speeds sufficient to allow real-time communication.

The rest of the paper is organized as follows. First, Section 2 introduces the data model, the challenges, and the design outline. Section 3 describes the real-time 3D compression schemes. Section 4 presents the experimental evaluation and comparison. Section 5 reviews our approach in the light of related work. Finally, Section 6 concludes this paper.

2. OVERVIEW

In this section, we introduce the TEEVE framework, the 3D data model, the challenges and design methodolgy of 3D compression, and its evaluation metrics.

2.1. TEEVE Framework

Figure 2 illustrates a layering view of TEEVE framework with a focus on the video part. The 3D multi-camera environment in the application layer at the sender end is responsible for 3D scene acquisition and reconstruction. The environment is composed of multiple 3D cameras which are placed around a subject and synchronized using hotwires. Each 3D camera is a cluster of four two-dimensional (2D) cameras with three calibrated black and white (b/w) cameras and one color camera. The images from b/w cameras are processed by the PC connected with them to capture the depth information using a trinocular stereo algorithm.[13,14] The 3D video streams generated contain both depth and color information, which are further processed by the service gateways of the service middleware layer for streaming to the receiver end. The middleware layer is responsible for tasks including traffic shaping, multi-stream coordination and synchronization, and skew control such that remote clients can generate, in real time, the realistic 3D model of the subject. The presence of depth information allows the renderer to provide 3D video effects, such as displaying the subject in an appropriate place on video monitors arranged in real space, or allowing a user to shift or rotate the image to see different points of view.

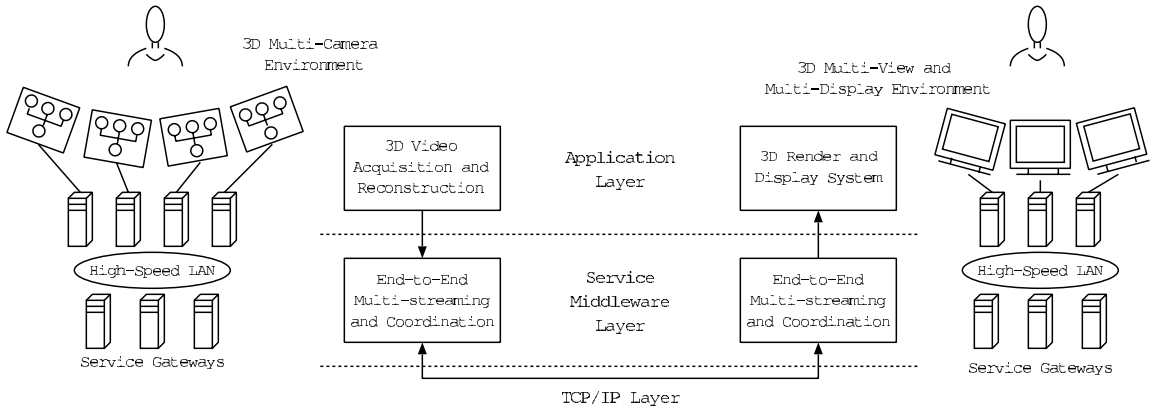


Figure 2. TEEVE Integrated Framework

2.2. 3D Data Model

The TEEVE capturing tier consists of a set of equal N 3D cameras organized around the subject and synchronized. At a discrete time instant T_j, each camera i ($i \in \{1, 2, ..., N\}$) takes one frame of *depth image* from its viewpoint, denoted as $f_{i,j}$. The depth image contains the depth and color information for each pixel. This means that at time T_j the capturing tier must have N 3D reconstructed frames that constitute a *macro-frame* F_j ($F_j = \{f_{1,j}, f_{2,j}, ..., f_{N,j}\}$) to show the same scene from different viewing angles by the remote renderer.

For the service middleware layer, one major challenge is to accommodate the huge data rate from the application layer. This is especially true at higher frame resolutions and frame rates. To exemplify, we aim to experiment with the video streams of 640×480 pixels, which is the maximum resolution supported by the underlying hardware. The frame format of depth image generated by each 3D camera is given in Figure 3. Each pixel consists of 5 bytes with 2 bytes for depth and 3 bytes for color information yielding an uncompressed frame size of $640 \times 480 \times (2 + 3) = 1,536,000$ bytes or approximately 12.3 Mbits. Furthermore, we expect to utilize 10 such 3D video cameras at the UIUC side, where each 3D camera produces a 3D video frame every 100 ms (10 frames per second). The whole setup indicates a basic

bandwidth requirement of 1.23 Gbps. Therefore, certain 3D real-time compression scheme must be deployed along with other techinques to alleviate the bandwidth requirement.

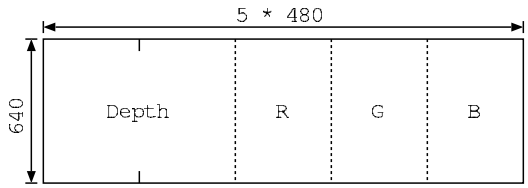


Figure 3. Format of 3D Depth Image

Another challenge for 3D video compression lies in the diversity of data. The existing image/video compression algorithms such as JPEG, MPEG and H.263 cannot be applied directly to 3D video compression because the video frame not only includes color information, but also depth information which would be lost if using those standard algorithms. Thus, we need to design a comprehensive compression scheme that can handle both color and depth information.

2.3. Design Methodology

Because of their different properties, the color and depth data should be compressed differently. The human visual system is relatively insensitive to variations in color so that existing algorithms such as JPEG and MPEG can take advantage of spatial and temporal redundancy in video streams to perform lossy compression that still performs well on color video images. However, any loss of depth information may distort the rendered volumetric image. Therefore, the depth information must be compressed using a lossless algorithm. In addition, the compression scheme must meet the following goals.

- As tele-immersive system is intended for interactive communication, the compression/decompression must be performed fast enough with desirable compression ratio to accommodate the real-time requirement and the underlying limitation of bandwidth.
- The compression algorithm should balance between information loss and time/space cost to allow effective and real-time transmission of 3D images with an acceptable visual quality.
- The compression algorithm needs to take into account the specifics of the 3D video taken from tele-immersive environments and the compatibility issue with inter-stream compression algorithm.

This paper presents two compression schemes for 3D depth image based on the above design guidelines. Both schemes use different compressions to handle color and depth information, which will be given with more detail in later sections.

3. COMPRESSION SCHEMES

In this section, we present the design of two compression schemes, namely *Scheme A* and *Scheme B*, for 3D video compression. As mentioned earlier, both schemes contain two components: (a) a lossy compression for the color data and (b) a lossless compression for the depth data. The major difference lies in the approaches of color compression and we will justify our choice as well.

3.1. Scheme A

The first one is a hybrid compression scheme that applies *color reduction* to compress the color information, and then uses *background pixel removal* and *zlib* to further compress the whole data including the depth information. The color reduction is one of the popular image compression techniques[15, 16] whose major application is to economize the use of bits for colors (e.g. reducing from 24 bits to 8 bits per pixel). For most of the practical cases, the color reduction can maintain pretty high visual quality. In addition, there are several reasons for choosing the color reduction as the compression method for 3D tele-immersive video.

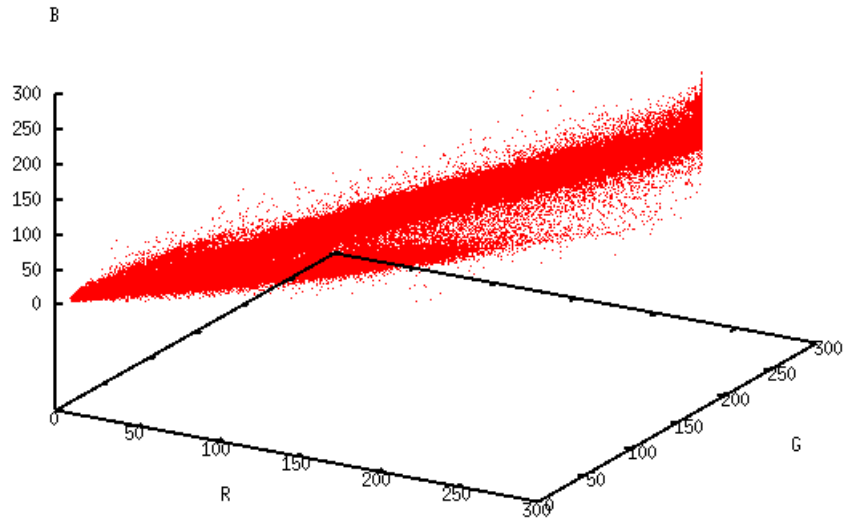


Figure 4. Color Distribution of Tele-immersive Video

- The color reduction is suitable for the context of a simple color composition which is typical in a tele-conferencing setup. For example, Figure 4 shows the distribution of colors in RGB space from 612 depth images captured in our tele-immersive environment featuring a person moving against a blank background (Figure 10).
- We show in the design and experiment that if the color composition does not change so frequently the color reduction will become a light-weighted function whose running overhead can be kept very small. This property also fits well to the tele-immersive environment.
- The color reduction is a pixel-level operation, which can be performed on a *partial image* (i.e. image with some of its pixels removed). Therefore, it can be easily adopted as a second-step compression after the inter-stream 3D compression[11] to further compress the partial image where the redundant points are removed.

3.1.1. Color Reduction

Here, we briefly describe one color reduction algorithm tailored for the tele-immersive application. The algorithm is based on ImageMagick[15] which contains three phases: *color classification*, *color reduction*, and *color assignment*.

Color classification builds a color description tree for the image in RGB color space. As shown in Figure 5, the root of the tree represents the entire space from $(0,0,0)$ to $(C_{max}, C_{max}, C_{max})$ where C_{max} equals 255. Each lower level is generated by subdividing the cube of one node into eight smaller cubes of equal size. For each pixel in the input image, classification scans downward from the root of the color description tree. At each level of the tree, it identifies the single node which represents a cube containing the color of the pixel and updates the statistics of such node.

Color reduction collapses the color description tree until the number it represents is at most the number of colors desired in the output image. The goal is to minimize the numerical discrepancies between the original colors and quantized colors. For this, color reduction repeatedly prunes the tree. On any given iteration over the tree, it selects those nodes whose quantization error is minimal for pruning and merges their color statistics upward.

Finally, color assignment generates the output image from the pruned tree. It defines the *color map* of the output image and sets the color of each pixel via indexing into the color map. For the example of 24-bit to 8-bit color reduction, the output image contains the color map of 256 bytes and an index array of pixels which is one third the size of the original pixel array, achieving a total compression ratio close to 3.

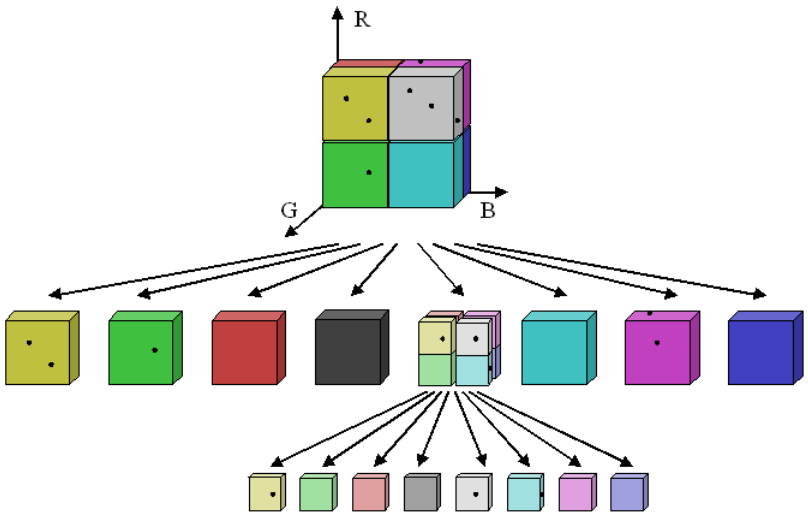


Figure 5. Color Description Tree

3.1.2. Background Pixel Removal

Following the color reduction, we apply another compression technique of *background pixel removal*. Background pixels often take considerable portion, sometimes even the majority of the entire frame. For tele-immersive application, it is the foreground pixels that carry more useful image information. The background pixels are usually blank after the initial background segmentation and can be entirely eliminated (e.g. Figure 10). Thus, background pixel removal saves the storage space of an image for both of its color and depth information. For simplicity, the background pixel removal is only applied on color information.

In order to perform background pixel removal, a *bitmap* is generated for each frame. Attached at the frame head, the bitmap shows for each pixel whether it belongs to the background (value is 0) or foreground (value is 1). Since each entry only consumes 1 bit, the additional overhead to add such a bitmap is acceptable and well justified, given the huge space saving by the background pixel removal. In a typical image captured from the tele-immersive environment, the background pixels account for around 70% of total pixels, yielding a compression ratio over 2.3 for color compression.

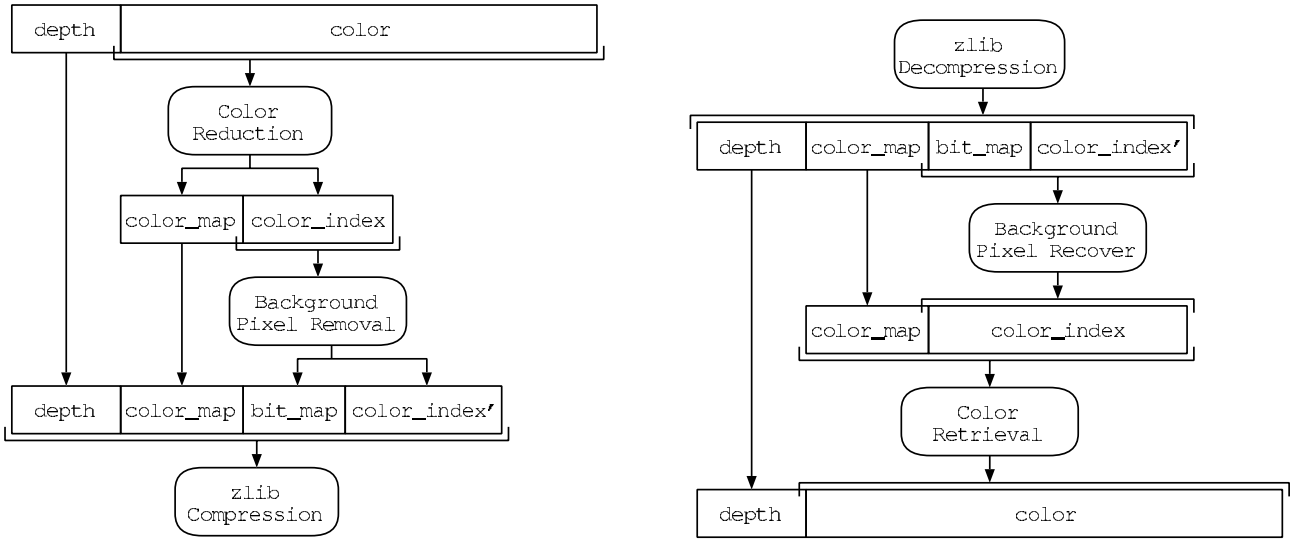
3.1.3. zlib Compression

After the previous two steps, the color information is shrunk to almost one seventh of its original size. As the last step, the *zlib*[17] compression, a powerful, generic and openly available compression tool, is applied to the depth information along with the reduced color information to further improve the compression ratio.

The decompression follows similar reverse steps except that color reduction is much simpler. Once the bitmap, the color map, and the color index array are restored, the original color information can be easily recovered. As a summary, the overall compression and decompression procedures are illustrated in Figure 6.

3.1.4. Practical Issues

Color reduction algorithm is regarded as an expensive operation, which may impact the performance of real-time image transmission. However, considering the fact that the basic color layout of all image frames contained in a tele-immersion session does not change dramatically (the colors of people's hair, face and clothes do not change), the full-fledged color reduction algorithm does not have to be performed for each frame. Once the color description tree has been set up and properly pruned (via classification and reduction phases), the output color-reduced image can be generated during the assignment phase. While the first frame of the session has to go through all three phases to have its color description tree ready, the follow-up frames can simply reuse the same tree to directly generate the output image which only involves the assignment phase. Since most computing-intensive operations of the algorithm happen during the classification and reduction phases, much computing overhead is saved.



(a) Compression Diagram

(b) Decompression Diagram

Figure 6. Compression and Decompression of Scheme A

Furthermore, a runtime monitoring feedback loop can be introduced at the sender side to periodically calculate the color reduction error. If the errors become consistently larger than a threshold, it implies that the original pruned tree becomes less accurate at representing the color layout of the current scene. In this case, we rerun the entire algorithm on the current image frame to generate a new tree, and then continue the same procedure as described before. The larger computing cost can be amortized over n frames. For the tele-immersive environment where the color layout is assumed to be very stable, the average value of n could be very big.

As shown in the experimental results, this compression scheme is very well suited to our problem domain, giving good compression ratios as well as good real-time performance and visual quality. Although color reduction is usually considered a time-consuming operation, we show that under the context of tele-immersive environments most of the computing overhead can be eliminated to achieve very high performance for 3D video compression.

3.2. Secheme B

In the second scheme, the color and depth information are processed in separate steps. This scheme uses motion JPEG for color compression and the run length coding plus Huffman coding for depth compression. This treatment also gives good performance with regards to compression ratio, time and visual quality.

3.2.1. Color Compression

JPEG[18]- a standardised image compression mechanism stands for Joint Photographic Expert Group, a standard committee that had its origin with International Standard Organisation (ISO). It provides a compression method that is capable of compressing colour or gray scale continuous tone images of real world subjects such as photographs, still videos or any complex graphics that resemble natural subjects. But JPEG doesn't compress so well on lettering, sample cartoons or line drawings. Although earlier implementation of JPEG was only used for still images, recent developments have seen that it is slowly branching into compression for motion-picture. Unlike other compression techniques, JPEG does not operate on a single algorithm. Instead it has been built up by various compression techniques which serve as its tools. JPEG allows various configuration of these tools depending on the needs of the user. Motion JPEG uses JPEG still image compression on each frame separately, with no inter-frame motion estimation as in MPEG. This scheme sacrifices compression efficiency, but it eliminates the difficult problem of error propagation that occurs when frame packets are dropped in the context of motion-estimated compression.[19]

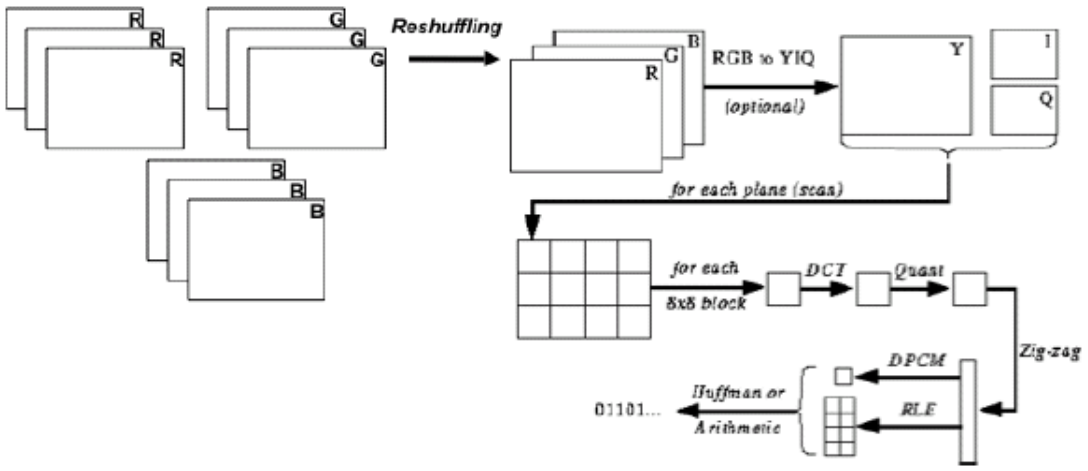


Figure 7. Detail of the color compression algorithm

3.2.2. Depth Compression

The depth information of an entire frame was concatenated and compressed using run-length encoding followed by Huffman coding. We use run-length encoding (RLE) because of the large amount of spatial redundancy in the transmitted image of an object against a blank background. Huffman coding following RLE is a natural choice because the result of RLE is a set of symbols representing run lengths that occur with varying frequencies.

3.2.3. Implementation

For the color compression, we use the Independent JPEG Group's library software 6b. This algorithm assumes a pattern of RGB values arranged in scan lines, similar to the arrangement of pixels on a TV screen. Therefore, we first had to rearrange the color information generated by the cameras (Figure 3) to be in this form. After compressing and decompressing the color information, we put it back in the original form (all the red information, followed by all the green and blue information) before sending it to the renderer. Figure 7 illustrates the color compression steps we perform.

For the depth compression, we use the Basic Compression Library by Marcus Geelnard. This is a freely available library of basic compression algorithms including RLE and Huffman coding. It is not the fastest possible implementation of these algorithms. However, it uses a compact Huffman tree, it is freely available, and it is simple and written in portable ANSI C. As such, it is suitable for our purposes.

3.2.4. Network Frame Encoding

To accommodate the compression, we changed the frame encoding format to consist of the following:

1. The length of the RLE compressed depth information
2. The length of the Huffman compressed depth information
3. The length of the JPEG compressed color information
4. The compressed depth data
5. The compressed color data

Figure 8 provides a schematic representation of our compression algorithm, and figure 9 provides a schematic representation of our decompression algorithm.

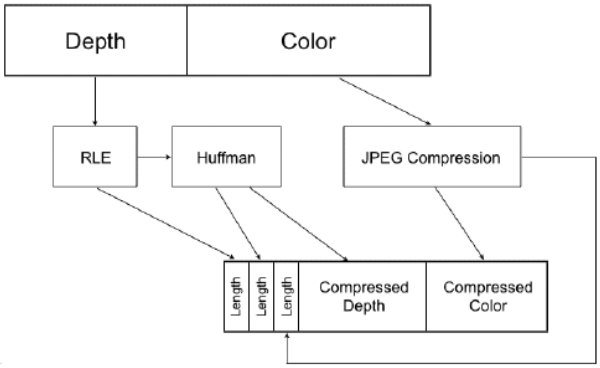


Figure 8. Schematic diagram of the compression algorithm

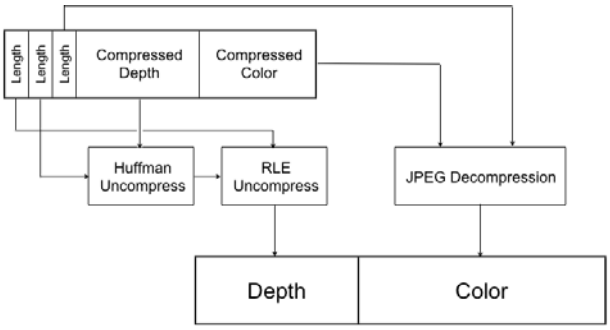


Figure 9. Schematic diagram of the decompression algorithm

3.3. Evaluation Metrics

According to the design goals, we evaluate and compare the performance of the two compression schemes in terms of the *time cost*, *compression ratio*, and *visual fidelity*.

- *Time Cost*. The time cost represents the compression and decompression time for one image frame. For compression, we are also interested in measuring the time cost of each individual components: (1) for *Scheme A*, the time cost of each phase of color reduction and zlib compression, (2) for *Scheme B*, the cost of color and depth compression.
- *Compression Ratio* (CR). The compression ratio is defined as: $CR = \frac{size\ of\ original\ data}{size\ of\ compressed\ data}$.
- *Visual Fidelity*. The video fidelity is measured in terms of the *peak signal-to-noise ratio* (PSNR), which is commonly used as a measure of reconstruction quality in image compression.

4. EXPERIMENT

In this section we compare the performance of the two compression schemes with the implementation and metrics described in the previous section.

4.1. Environment

Both two compression schemes are written in C++ and tested on Dell Precision 450 (Dual Xeon processor with 1 GBytes memory) computers running Fedora Core 2 system. We integrate our compression algorithm with the UIUC/UC Berkeley tele-immersion system so that we could (1) transmit the compressed frames and ensure that our compression and decompression are correct and (2) evaluate the performance of the system with our compression installed as compared with the base system. For testing and evaluation, we use one of the actual 3D video streams pre-recorded in the tele-immersive environment showing a person and his physical movement (Figure 10). Currently, we are using an image resolution of 320×240. The video stream contains 612 depth frames with a total size of 235 MBytes ($612 \times 320 \times 240 \times 5$).

In establishing our testing and evaluation environment, we want to reuse the existing TEEVE system as much as possible. Therefore, we deploy the compression and decompression code into the service gateways (shown in Figure 2) which makes the compression and decompression transparent to the capturing and displaying tiers and allows us to deploy the code with the minimum system modification. The experiment contains several runs, each run involving with compressing and decompressing one depth frame. The service gateway at the sender end compresses one frame and transmits it to the receiver where it is decompressed and compared with the original frame to measure the visual fidelity.[†]

[†]In the real usage, the receiver will not have the original frame. The setting here is for conveniency.

4.2. Compression Time

Table 1 shows the compression time of both schemes. The time unit is one millisecond (ms). For *Scheme A*, the most expensive time cost is color classification and reduction with an average around 35 ms. However, as mentioned earlier it is reasonable to assume a stable color layout in tele-conferecing environment such that the classification and reduction phases only need to be performed for the first frame of the session or every n frames based on a monitoring mechanism. The rest frames only need to go through the last three steps. For a stable color layout, n could take a pretty large value (e.g. $n = 100$ as we assume an average recalculation period of 10 seconds and a frame rate of 10 frames per second). Therefore, the average compression time can be taken as around 10 ms.

For *Scheme B*, the average compression time is around 13 ms. The color (JPEG) compression performed better than the depth compression (even though the color contains more data). This is probably because (1) the depth compression uses two different types of compression (run-length encoding and Huffman coding), (2) the JPEG implementation we use may be more efficient than the implementations of RLE and Huffman coding, and (3) the Huffman coding implementation recomputes the Huffman table each time, whereas JPEG uses a fixed Huffman table.

Table 1. Compression Time of Two Schemes

(a) Scheme A (unit: ms)

	min	avg	max
classification	0.416	7.11	14.4
reduction	0.001	27.6	80.9
assignment	0.878	4.19	9.29
background removal	0.641	0.708	1.54
zlib compression	2.66	4.93	11.1

(b) Scheme B (unit: ms)

	min	avg	max
color compression	3.59	5.99	10.2
depth compression	3.68	7.02	10.5

The timing results of both compression schemes are quite good, with even the worst time of around 20 ms and well below 100 msec, which would be basic requirement for processing ten 3D frames per second. These results show that our compression schemes are feasible for processing 3D videos demanded by the TEEVE system.

4.3. Decompression Time

Table 2 shows the result of decompression time for both schemes. The decompression time of *Scheme A* is better than that of *Scheme A* (with average 1.7 ms versus 9.2 ms). Further analysis shows that, for *Scheme B*, the color decoding (JPEG) is on average faster than encoding, but depth decoding is significantly slower than depth encoding. This agrees with the manual for the Basic Compression Library we use for Huffman coding, which says that the implementation of decoding is slow and encoding is faster.

Table 2. Decompression Time of Two Schemes

(a) Scheme A (unit: ms)

	min	avg	max
decompression	1.36	1.72	3.99

(b) Scheme B (unit: ms)

	min	avg	max
decompression	2.87	9.23	16.6

4.4. Compression Ratio

Table 3 gives the compression ratio achieved for the 612 depth frames. The performance of compression is very impressive (26.7 for *Scheme A* and 15.4 for *Scheme B*), which implies that on average the overall frame size (both color and depth) can be shrunk from 384 KBytes ($5 \times 320 \times 240$) to below 15 KBytes. This excellent compression reflects the large amount of spatial redundancy in the 3D depth image, which consists of a person moving against an empty background (Figure 10). For color compression, color reduction (after background pixel removal) and JPEG have comparable average compression performance. On the other hand, the depth compression performed by zlib is better than RLE plus Huffman coding.

Table 3. Compression Ratio of Two Schemes

(a) Scheme A

	min	avg	max
color ratio	10.1	14.2	22.2
overall ratio	14.9	26.7	358

(b) Scheme B

	min	avg	max
color ratio	9.57	15.8	126
depth ratio	7.69	14.9	272
overall ratio	8.69	15.4	200

4.5. Visual Fidelity

The visual fidelity of the color information after decompression is measured using PSNR (in dB). The results are given in Table 4, which indicates that both schemes have pretty high compression quality with *Scheme A* achieving slightly higher. We also visually judge the image quality by comparing two similar frames before and after data compression as in Figure 10. No quality degradation is observed. We note that in many tele-immersion scenarios such as the one shown, the color layout of the image is rather simple, which yields unnoticeable degradation.

Table 4. PSNR of Two Schemes

(a) Scheme A (unit: dB)

	min	avg	max
color	45.9	51.5	74.3

(b) Scheme B (unit: dB)

	min	avg	max
color	41.5	45.7	68.3

4.6. Lessons Learned

For color compression, the performance data indicates that color reduction could be a better choice than JPEG in terms of compression time, compression ratio, and visual fidelity. However, the time cost of color reduction depends largely on the real situation of tele-immersion application. If the color change of the scene occurs very frequently and dramatically, then JPEG may give a much better performance. Hence as an extension, we propose a periodic monitoring and switching mechanism which operates at two levels. In the lower level, it decides whether a full-fledged execution of color reduction is necessary by measuring quality after decompression. In the higher level, if the most recent window of history shows a higher overhead of color reduction it may, depending on other source of information such as the bandwidth estimation, decide to switch to JPEG compression. The switching between the two schemes is very flexible as the compression method can be indicated in the frame header by using an extra one bit. For depth compression, the performance of zlib is better than run length coding plus Huffman coding which is majorly an implementation issue of different compression libraries.

4.7. Streaming Test

Our ultimate goal is to incorporate the compression schemes to the TEEVE system in a robust manner. By far, we have done some empirical studies of how compression schemes can improve the streaming quality. In our first preliminary experiment, we transmit five pre-recorded 3D depth streams over TCP connection using five Linux machines in the Computer Science Department of UIUC to an NCSA machine for the renderer. When we increase the frame rate to ten frames per second we notice that the player quality degrades markedly for the uncompressed streams. The synchronization of macro-frames often fails, causing the images from the different streams not to be composed properly. Instead of a single image of a person in one place, there are multiple incomplete images in several places. This is partly due to the high network load, which causes different latency and jitter experienced by different video streams. When we add our compression and decompression to the transmission, this problem is alleviated dramatically. In the second experiment, we run streaming tests from senders at UIUC and a renderer in UC Berkeley at five frames per second, where we observe similar enhancement of streaming quality. More extensive experiment will be done in our future work to provide more concrete evaluation.

(a) Visual Quality before Compression

(b) Visual Quality after Compression

Figure 10. Visual Quality Comparison

5. RELATED WORK

The data model of 3D image has a direct impact on the design of compression algorithm. Therefore, we divide related work into three categories based on the underlying data models and present them in the order of relevance to our work including: (a) compression based on *depth images*, (b) compression based on *volumetric data*, and (c) compression based on *triangular meshes*.

The data model of the first category is used in tele-immersive environments as described in Section 2 and 3. Under this model, a 3D image (*macro-frame*) is represented with multiple 2D depth images captured by individual 3D cameras from different viewpoints at the same time instant. Different from an ordinary 2D image, the depth image has extra depth

information for every pixel. Depending on the different exploitation of data redundancy, there are two major compression methods: *inter-stream* and *intra-stream* compression. In inter-stream compression,[11,12] the 2D image closest to the viewpoint of the user is selected as the reference image which is not compressed. Other images are compared with the reference image to remove redundant pixels within certain threshold of depth distance. The method decreases the total number of pixels that need to be rendered as the non-reference images are reduced to differential images. The problems of inter-stream compression include the considerable communication overhead involved with the broadcast of reference image and the diminishing of redundacy between images of larger viewpoint difference. To alleviate that, a two-level referencing and grouping scheme is applied. For the intra-stream compression as presented in this paper, each 2D depth image is independently analyzed to exploit the redundancy within itself. It is pointed out that the intra-stream compression is complementary to the inter-stream compression and can be naturally combined to improve the overall performance. For example, the intra-stream compression based on color reduction can be used to compress both reference image and differential image before they are transmitted over narrow bandwidth paths.

The 3D image of volumetric data model in the second category refers to a regular 3D grid whose *voxels* contain RGB or grayscale information. The 3D data are derived from a discrete collection of samples generated by scientific simulations or by volumetric imaging scanners such as computerised tomography (CT). The image size is usually very large which attracts research attention in compression. For example, one 3D image from the male dataset of the National Library of Medicine (NLM) contains 1,878 cross-sectional (2D) images (*slices*) taken at 1mm intervals. The slice has a resolution of 512×512 and each voxel needs 16 bits to store grayscale information. The *3D wavelet transform*[6,7,20] is the most important compression method for this data model. To perform 3D wavelet transform, a 3D image is first divided into *unit blocks* of size $16 \times 16 \times 16$ to take advantage of the spatial coherence in the cube. The wavelet coefficients are computed for each unit block. After that, non-zero coefficients are further processed using quantization and entropy encoding. Although 3D wavelet transform achieves very high compression ratio, its application in 3D video of tele-immersive environments is not straightforward and the overhead may offset its advantages. For example, it is not benefitial to apply 3D wavelet tranform on sparse dataset such as the contour of the subject. To make it more efficient, certain tranformation function is needed to first pack the data in a more concrete form while maintaining a high spatial coherence within unit blocks. This function may involve with significant time cost.

The last category of triangular meshes represents the most widely used 3D geometric model in computer graphics for purposes such as manufacturing, scientific computation, and gaming industry due to the fact that polygonal surfaces can be efficiently triangulated. The triangular representation of 3D geometry has two major parts: vertex coordinates and triangles (or the *connectivity* of vertices), along with their associated properties including color. The data size is also huge for the representation which may contain millions of triangles. On average, the number of triangles is twice the number of vertices and they have different kinds of data redundancy. Hence, the popular mesh compression algorithms treat vertices and triangles differently. The vertex data can be compressed using the *vertex spanning tree* as in the Topological Surgery approach[21] adopted by the MPEG-4 standard. The spanning tree is constructed to exploit the spatial coherence that the proximity of the nodes in the tree often implies geometric proximity of the corresponding vertices. The ancesors of the tree can be used as predictors and only the differences between prediction and actual value are encoded. The differences are further processed using quantization and entropy coding. The property information can be compressed in a similar way. For triangle data, the basic idea of compression is *triangle strip* which defines a particular order of traverse such that each new triangle can be represented by adding one vertex. Among those proposed, the *Edgebreaker*[8,22] is the most advanced scheme which encodes the traversal using an alphabet of five symbols with an overall performance of less than 2 bits per triangle. Recent efforts[22–24] aim to reduce the number of bits per triangle and extend the basic scheme to more arbitrary topology. The 3D compression based on triangular meshes also achieves very high compression ratio. However, the challenge of applying this technique lies in the real-time and automatic acquisition of 3D model which is becoming a very active research area.

6. CONCLUSION

In this paper, we design, implement, and compare two intra-stream real-time 3D video compression schemes suitable for 3D video transmission in tele-immersive environments. To accommodate the different requirements of color and depth compression, our first scheme uses color reduction and background pixel removal to compress the color data followed by zlib compression for the complete data. Our second scheme uses motion JPEG to compress the color frames and encode the depth frames using lossless entropy coding. We demonstrate that both algorithms perform very well, achieving an average

compression ratio over 26 (the first scheme) and 15 (the second scheme) for the 3D video captured from the tele-immersive environment, which we believe is representative of the data that would be transmitted in a real 3D tele-conferencing system. Moreover, the compression is fast for both schemes with an average around 10 ms. For the decompression, the first scheme uses an average time less than 2 ms while the second scheme uses around 9 ms. Both schemes maintain similar and very high visual quality. In our initial streaming experiment, we observe a remarkable improvement in performance due to reduced network load when we introduce our compression and decompression schemes into the video transmission.

We further discuss the trade-off between the two compression schemes. The scheme based on color reduction has better performance but is restricted to scenes of simple and stable color composition, while the scheme based on JPEG could be applied to more general situations with some loss of performance. Hence, we propose a monitoring and switching mechanism to dynamically tune the compression performance according to the real situation.

As future work, we will explore ways to achieve additional compression including:

- *Temporal redundancy* - Using video compression based on motion estimation, which would be feasible with TCP (but more difficult with UDP, because of error propagation);
- *View-based stream selection* - Transmitting streams only to viewers that actually need them (for example, a stream representing the reverse side of a three-dimensional object need not be sent until the object is rotated to bring that side in view);
- *Inter-stream redundancy* - Exploiting redundancy in the data received from different cameras, which aims to extend the previous inter-stream compression algorithm.[11]

Finally, we will continue on a robust integration of the compression schemes with the TEEVE system for an overall performance enhancement.

ACKNOWLEDGMENTS

This work was supported by the NSF grant under NSF ANI 03-23434. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

REFERENCES

1. P. Kauff and O. Schreer, "An immersive 3d video-conferencing system using shared virtual team user environments," in *CVE '02: Proceedings of the 4th international conference on Collaborative virtual environments*, pp. 105–112, ACM Press, (New York, NY, USA), 2002.
2. D. E. Ott and K. Mayer-Patel, "Coordinated multi-streaming for 3d tele-immersion," in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 596–603, ACM Press, (New York, NY, USA), 2004.
3. H. H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, J. MacCormick, K. Yuasa, W. B. Culbertson, and T. Malzbender, "Computation and performance issues in coliseum: an immersive videoconferencing system," in *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pp. 470–479, ACM Press, (New York, NY, USA), 2003.
4. J. Leigh, A. Johnson, M. Brown, D. Sandin, and T. DeFanti, "Visualization in teleimmersive environments," *Computer* **32**(12), pp. 66–73, 1999.
5. P. Narayanan, P. Rander, and T. Kanade, "Constructing virtual worlds using dense stereo," in *Proceedings of International Conference on Computer Vision*, pp. 3–10, 1998.
6. J. Wang and H. K. Huang, "Medical image compression by using three-dimensional wavelet transformation," *IEEE Tran. on Medical Imaging* **15**, pp. 547–554, August 1996.
7. C. Bajaj, I. Ihm, and S. Park, "3d rgb image compression for interactive applications," *ACM Trans. Graph.* **20**(1), pp. 10–38, 2001.
8. J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Transactions on Visualization and Computer Graphics* **5**(1), pp. 47–61, 1999.

9. L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system," in *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 39–46, ACM Press, (New York, NY, USA), 1995.
10. J. Shade, S. Gortler, L. wei He, and R. Szeliski, "Layered depth images," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 231–242, ACM Press, (New York, NY, USA), 1998.
11. S.-U. Kum, K. Mayer-Patel, and H. Fuchs, "Real-time compression for dynamic 3d environments," in *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pp. 185–194, ACM Press, (New York, NY, USA), 2003.
12. S.-U. Kum and K. Mayer-Patel, "Real-time multidepth stream compression," *ACM Trans. Multimedia Comput. Commun. Appl.* **1**(2), pp. 128–150, 2005.
13. J. Mulligan and K. Daniilidis, "Real time trinocular stereo for tele-immersion," in *International Conference on Image Processing*, pp. III: 959–962, 2001.
14. J. Mulligan, V. Isler, and K. Daniilidis, "Trinocular stereo: A real-time algorithm and its evaluation," *International Journal of Computer Vision* **47**, pp. 51–61, April 2002.
15. "Imagemagick, http://www.imagemagick.org/script/quantize.php."
16. "Color reduction, http://www.catenary.com/appnotes/colred.html."
17. "Zlib 1.2.2, http://www.zlib.net."
18. "Jpeg, http://www.jpeg.org."
19. I. Rhee, "Retransmission-based error control for interactive video applications over the internet," in *Proceedings of International Conference on Multimedia Computing and Systems (ICMCS'98)*, pp. 118–127, June 1998.
20. I. Ihm and S. Park, "Wavelet-based 3d compression scheme for very large volume data," *Graphics Interface* , pp. 107–116, June 1998.
21. G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM transactions on Graphics* **17**(2), pp. 26–34, 1998.
22. J. Rossignac, A. Safanova, and A. Szymczak, "3d compression made simple: Edgebreaker with zip&wrap on a corner table," in *Shape Modeling International Conference*, (Genova, Italy), May 2001.
23. T. Lewiner, H. Lopes, J. Rossignac, and A. Vieira, "Efficient edgebreaker for surfaces of arbitrary topology," in *Proceedings of 17th Brazilian Symposium on Computer Graphics and Image Processing*, pp. 218–225, Oct. 2004.
24. H. Lopes, J. Rossignac, A. Safanova, A. Szymczak, and G. Tavares, "Edgebreaker: A simple compression algorithms for surfaces with handles," *Computers and Graphics International Journal* **27**(4), pp. 553–567, 2003.