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

ABSTRACT

Tele-immersive systems can improve 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 at Urbana-Champaign and the University of California at Berkeley seeks to foster the development and use 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 a sense of immersive space. However, the transmission of 3D video poses significant challenges. First, it is bandwidth-intensive, as it requires the transmission of multiple large-volume 3D video streams. Second, existing schemes for 2D color video compression such as MPEG, JPEG, and H.263 cannot be applied directly because the 3D video data contains depth as well as color information. Our goal is to explore from a different angle of the 3D compression space with factors including complexity, compression ratio, quality, and real-time performance. To investigate these trade-offs, we present and evaluate two simple 3D compression schemes. For the first scheme, we use color reduction to compress the color information, which we then compress along with the depth information using zlib. For 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. We apply both schemes to 3D videos captured from a real tele-immersive environment. Our experimental results show that: (1) the compressed data preserves enough information to communicate the 3D images effectively (min. PSNR > 40) and (2) even without inter-frame motion estimation, very high compression ratios (avg. > 15) are achievable at speeds sufficient to allow real-time communication (avg. ≈ 13 ms per 3D video frame).

Keywords: Tele-Immersion, Real-Time 3D 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 a 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, a tele-immersive environment requires several basic components, including a 3D camera array and sound system, a communication network, and a display system. Recently, several researchers have developed and experimented with individual components or tele-immersive environments with partially integrated components.[1–5]

The **T**ele-immersive **E**nvironments for **EVE**rybody (TEEVE) project[6] currently underway at the University of Illinois at Urbana-Champaign and the University of California at Berkeley aims to integrate, deploy, and experiment with existing tele-immersive components, in order to explore the possibilities and challenges of deploying this cutting edge technology in a more cost effective manner and across a broader audience. The TEEVE system features a distributed multi-tier application model (Figure 1) in which each user is surrounded by cameras and audio devices to capture his or her body movements. The capturing tier of the TEEVE system captures user input and transmits it across the transmission tier to

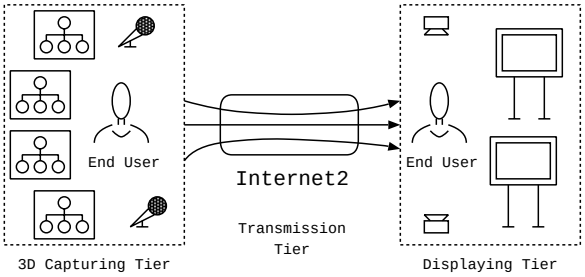


Figure 1: TEEVE Application Model

the receiving user. The displaying tier allows the receiving user to observe the transmitting user's physical activity from arbitrary angles via multi-view/display devices.

This paper focuses on compression techniques for 3D image data, the transmission of which is very bandwidth intensive. In TEEVE, developing quality, real-time, and immersive communication demands a basic bandwidth at the Gbps level. TEEVE uses the data model of *depth images*[7, 8] 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.

There are several 3D compression schemes proposed which are highly influenced by the underlying data models, such as 3D wavelet transform[9, 10] for volumetric data and Edgebreaker[11] for triangular meshes. These schemes may not be directly and efficiently applied to the depth images. Further, those schemes assume the availability of the 3D model before compression, which no longer holds in the tele-immersive environment. Sang-Uok Kum et al. have done pioneering research on depth image compression and proposed inter-stream compression scheme[12, 13] where multiple depth streams are analyzed to remove redundant points. In contrast, our work concentrates on *intra-stream* compression schemes, which are important in multiple scenarios. First, intra-stream compression can be used in the initial data collection stage to save network traffic. Second, for scalability it does not incur the traffic complexity involved with stream propagation as each stream is compressed separately. Finally, our intra-stream compression scheme is complementary to the inter-stream compression and can be integrated with it for overall performance enhancement.

Our contribution in this paper is to present the design and evaluation of two simple intra-stream compression schemes for use in tele-immersive environments. For the first scheme (namely, the *CR-zlib scheme*), we apply color reduction to reduce the number of bits representing the color information. The image is further compressed using zlib on both color (reduced) and depth information. In the second scheme (namely, the *JPEG-RH 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 inter-frame 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. 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; and the challenges, design methodology, and evaluation metrics of 3D compression.

2.1 TEEVE Framework

Figure 2 illustrates a layering view of the 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 placed around a subject and synchronized using hotwires. Each 3D camera is a cluster of four calibrated two-dimensional cameras, with three black and white (b/w) cameras and one color

camera. The images taken at the same time instant from the b/w cameras are processed by the PC connected to them to compute the depth information using a trinocular stereo algorithm,[14, 15] while the color camera extracts appearance from the corresponding viewpoint. Thus, the generated 3D video images (i.e., depth images) contain both depth and color information. This information is further processed by the service gateways of the service middleware layer to perform tasks including traffic shaping, multi-stream coordination and synchronization, and skew control so 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 physical space, or allowing a user to shift or rotate the image to see different points of view.

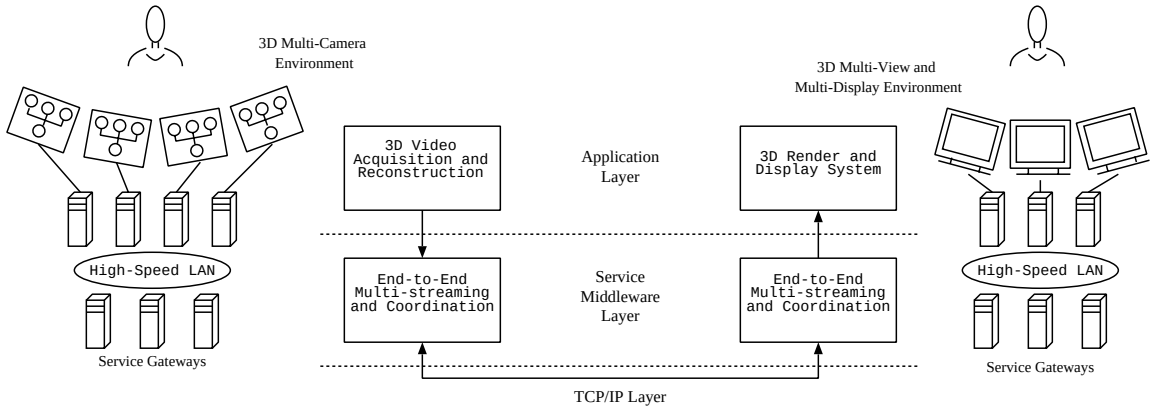


Figure 2: TEEVE Integrated Framework

2.2 3D Data Model

The capturing tier consists of a set of N equal 3D cameras organized around the subject and synchronized. At time T_j, each camera i ($i \in \{1,2,...,N\}$) captures one frame of depth image from its viewpoint, denoted $f_{i,j}$. The depth image contains the depth and color information for each pixel. This means that at time T_j the remote renderer uses N 3D reconstructed frames constituting 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.

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 rates. For example, we aim to experiment with video streams of 640×480 pixels, which is the maximum resolution supported by the underlying hardware. Each pixel consists of 5 bytes, with 2 bytes for depth and 3 bytes for color information, yielding an uncompressed frame size of $640 \times 480 \times 5 = 1,536,000$ 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, which is too large to be practical. Thus, we must use real-time 3D compression to alleviate this bandwidth requirement. Another challenge for 3D video compression lies in the diversity of data. Existing image/video compression algorithms such as JPEG, MPEG, and H.263 cannot be applied directly to 3D video compression because the video frame includes not only color information, but also depth information. Thus, we need to design a compression scheme that can handle both color and depth information.

2.3 Design Methodology

Because of their different properties, the color and depth data must be considered differently. The human visual system is relatively insensitive to variations in color. 2D compression algorithms such as MPEG and JPEG take advantage of this insensitivity by using lossy compression methods that still perform well on color video images. However, any loss of depth information may distort the rendered volumetric image. Therefore, we decided to use a lossless algorithm to compress the depth information. As future work, we plan to investigate the effect of lossy compression on depth rendering. In addition, the compression scheme must meet the following goals:

- The compression and decompression must be performed fast enough for real-time interactive communication.
- There must be sufficient compression to accommodate the underlying bandwidth limitation.

- The compression algorithm should balance information loss against time and space cost to allow effective real-time transmission of 3D images with acceptable visual quality.
- The intra-stream compression algorithm must take into account particular 3D representations used in tele-immersive environments and should be compatible with inter-stream compression algorithms.

In the following sections, we present two compression schemes for 3D depth images motivated by the design considerations given above.

3 Compression Schemes

In this section, we present the design of two compression schemes, namely the *CR-zlib scheme* and the *JPEG-RH scheme*, for 3D video compression. As mentioned earlier, both schemes contain two components: (a) lossy compression for the color data and (b) lossless compression for the depth data. The major difference lies in the approach to color compression.

3.1 CR-zlib Scheme

The CR-zlib scheme applies *color reduction* to compress the color information and then uses *zlib* to further compress all the data, including the depth information. Color reduction is a popular image compression technique[16,17] that reduces the number of bits used to represent colors (e.g. reducing from 24 bits to 8 bits per pixel). Color reduction can maintain high visual quality in most cases. In addition, there are several reasons for using color reduction in the context of 3D tele-immersive video:

- Color reduction is suitable in the case of simple color composition, which is typical in a tele-conferencing setup. For example, Figure 3 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 8).
- As our experiments show, if the color composition is relatively constant, then the runtime overhead of color compression can be kept small. This property also fits well with tele-immersive environments.
- Color reduction is a pixel-level operation that 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[12] to compress the partial image further once the redundant points have been removed.

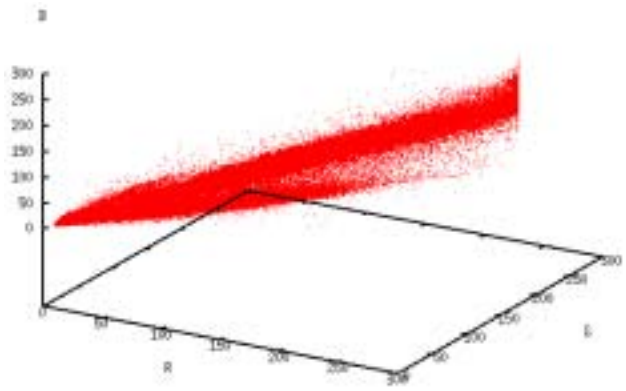


Figure 3: Color Distribution of Tele-immersive Video

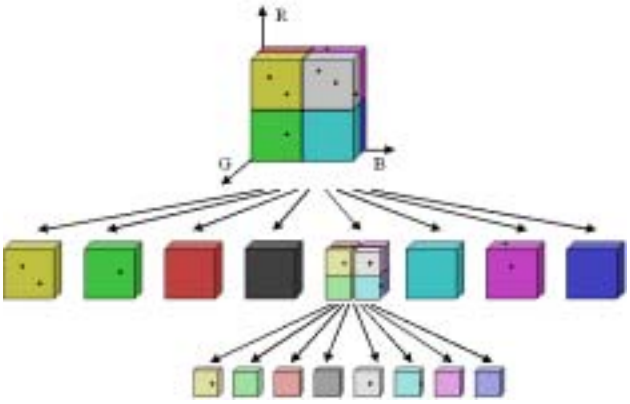


Figure 4: Color Description Tree

3.1.1 Color Reduction

We briefly describe one color reduction algorithm tailored for tele-immersive applications. The algorithm is based on ImageMagick,[16] which contains three phases: *classification*, *reduction*, and *assignment*.

Color classification builds a color description tree for the image in RGB color space. In Figure 4, the root of the tree represents the entire space from $[0,0,0]$ to $[C_{max}, C_{max}, C_{max}]$ (usually $C_{max} = 255$). 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 that represents a cube containing the color of the pixel and updates the statistics including the quantization error in that node.

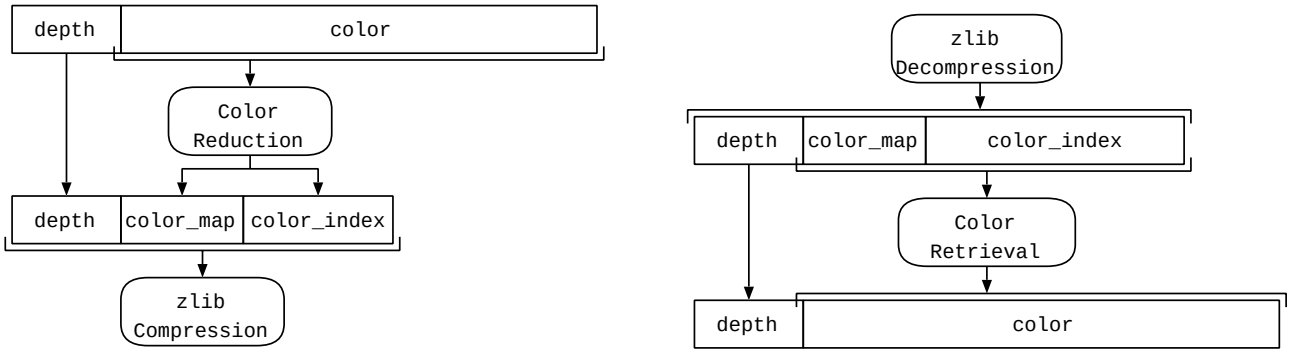
Color reduction collapses the color description tree until the number it represents is at most the number of colors desired in the output image. The goal is to minimize the numerical discrepancies between the original colors and quantized colors. For this, color reduction repeatedly prunes the tree. 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 768 bytes and an index array of pixels that is one third the size of the original pixel array, achieving a total compression ratio close to 3.

3.1.2 zlib Compression

After color reduction, *zlib*[18] 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 steps in reverse except that color reduction is much simpler. Once 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 5.



(a) Compression Diagram (b) Decompression Diagram

Figure 5: Compression and Decompression of Hybrid Scheme

3.1.3 Practical Issues

Color reduction is regarded as an expensive operation, which may affect the performance of real-time image transmission. However, because the basic color layout of the image frames in a tele-immersion session does not change dramatically (the colors of people's hair, face and clothes do not change), we need not perform the full-fledged color reduction algorithm for each frame. Once the color description tree has been set up and properly pruned (via the 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, the follow-up frames can simply reuse the tree generated in the first frame to generate the output image directly; this involves only the assignment phase. Since most compute-intensive operations of the algorithm happen during the classification and reduction phases, much computing overhead is saved.

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, the original pruned tree has become 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 above. The larger computing cost can be amortized over n frames. For tele-immersive environments, in which the color layout is assumed to be very stable, the average value of n could be large.

As shown in the experimental results below, 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 JPEG-RH Scheme

The JPEG-RH scheme uses motion JPEG for color compression and run-length coding (RLE) plus Huffman coding for depth compression. This approach also gives good performance with regard to compression ratio, time, and visual quality.

3.2.1 Color Compression

JPEG[19] provides a compression mechanism that is capable of compressing color or gray scale continuous tone images of real world subjects such as photographs, still videos, or any complex graphics that resemble natural subjects. JPEG was originally used only for still images, but more recently it has been used for video as well. 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.[20]

3.2.2 Depth Compression

In the separate scheme, we compress the depth information of an entire frame 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. For the depth compression, we use the Basic Compression Library[21] 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. 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 6 provides a schematic representation of our compression algorithm, and Figure 7 provides a schematic representation of our 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 the individual components: (1) for the CR-zlib Scheme, the time cost of each phase of color reduction and zlib compression, (2) for the JPEG-RH Scheme, 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.

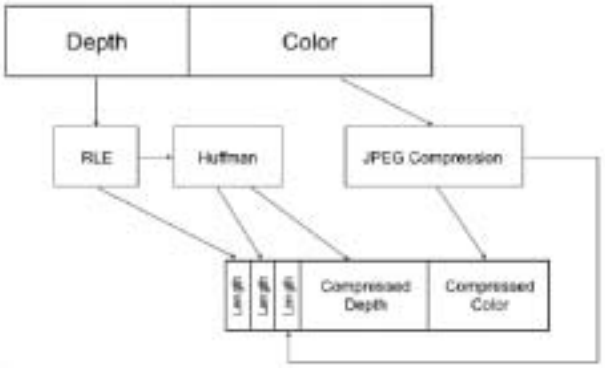


Figure 6: Schematic diagram of the compression algorithm

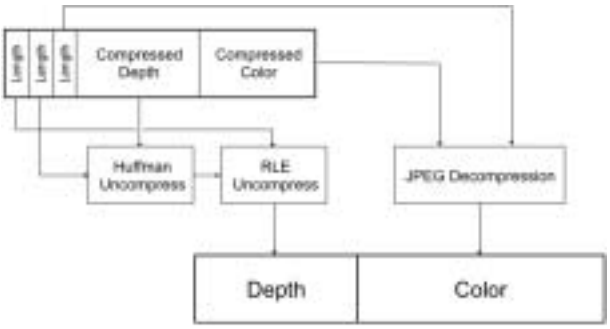


Figure 7: Schematic diagram of the decompression algorithm

4 Experimental Evaluation

In this section we compare the performance of the two compression schemes using the metrics described in section 3.

4.1 Environment

We implemented both compression schemes in C/C++ on the Linux operating system and tested our code on Dell Precision 450 (Dual Xeon processor with 1 GBytes memory) computers running Fedora Core 2. We decided to 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 were correct and (2) evaluate the performance of the system with our compression installed as compared with the base system. For testing and evaluation, we used a 3D video scene pre-recorded in the tele-immersive environment that shows a person and his physical movement (Figure 8). Currently, our system uses an image resolution of 320×240. Each uncompressed video stream in our test tele-immersive recording contains 612 depth frames with a total per-stream size of 235 MBytes ($612 \times 320 \times 240 \times 5$).

In establishing our testing and evaluation environment, we wanted to reuse the existing TEEVE system as much as possible. Therefore, we integrated the compression and decompression code into the service gateways (shown in Figure 2). This integration made the compression and decompression transparent to the capturing and displaying tiers and allowed us to deploy the code with minimal system modification. The experimentation consisted of several runs, each involving the compression and decompression of individual video frames as follows: The service gateway at the sender end compressed a frame and transmitted it to the receiver, where it was decompressed and compared with a local copy of the original frame to measure degradation of visual fidelity.

4.2 Compression Time

Table 1 shows the compression time of both schemes. The time unit is one millisecond (ms). For the CR-zlib scheme, the most expensive operation in terms of time taken 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 a tele-conferencing environment such that the classification and reduction phases need to be performed only for the first frame of the session or every n frames based on a monitoring mechanism. In between, frames may omit the classification and reduction steps. For a stable color layout, n could be large (e.g. $n = 100$). Therefore, the average compression time can be taken as around 10 ms, as we assume an average recalculation period of 10 seconds and a frame rate of 10 frames per second.

For the JPEG-RH scheme, 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 for each frame, whereas JPEG uses a fixed Huffman table.

The timing results of both compression schemes are quite good, with even the worst time being around 20 ms and well below 100 msec, which is the basic requirement for processing 10 3D frames per second. These results show that our compression schemes are feasible for the 3D video processing demanded by the TEEVE system.

Table 1: Compression Time of Two Schemes

(a) CR-zlib Scheme (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
zlib compression	2.66	4.93	11.1

(b) JPEG-RH Scheme (unit: ms)

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

4.3 Decompression Time

Table 2 shows the decompression times for both schemes. The decompression time of the CR-zlib scheme is better than that of the JPEG-RH scheme (with average 1.7 ms versus 9.2 ms). For the JPEG-RH scheme, 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 states that decoding is slower than encoding.

Table 2: Decompression Time of Two Schemes

(a) CR-zlib Scheme (unit: ms)

	min	avg	max
decompression	1.36	1.72	3.99

(b) JPEG-RH Scheme (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 (with a compression ratio of 25.9 for the CR-zlib scheme and 15.4 for the JPEG-RH scheme), 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 8). For color compression, color reduction 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) CR-zlib Scheme

	min	avg	max
color ratio	2.97	2.97	2.97
overall ratio	14.6	25.9	338

(b) JPEG-RH Scheme

	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 high compression quality, with the CR-zlib scheme achieving slightly better fidelity to the original image.

We also visually judge the image quality by comparing two similar frames before and after data compression as in Figure 8. No subjective 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) CR-zlib Scheme (unit: dB)

	min	avg	max
color	45.9	51.5	74.3

(b) JPEG-RH Scheme (unit: dB)

	min	avg	max
color	41.5	45.7	68.3

(a) Visual Quality before Compression

(b) Visual Quality after Compression

Figure 8: Visual Quality Comparison

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 particular tele-immersive video being processed. 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, a monitoring thread 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 for color reduction, the monitor may (depending on other factors such as 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 one extra bit. For depth compression, the performance of zlib is better than run length coding plus Huffman coding, though this is primarily an implementation issue of the different compression libraries.

4.7 Streaming Test

Our ultimate goal is to incorporate the compression schemes in the TEEVE system in a robust manner. So far, we have done some empirical studies of how compression schemes can improve streaming quality, even without explicit stream quality preservation mechanisms. In our first preliminary experiment, we transmitted five pre-recorded 3D depth streams over TCP connections from five Linux machines in the Computer Science Department of UIUC to an NCSA Linux system, which does the rendering. When we increased the frame rate to ten frames per second we noticed that the player quality degraded markedly for the uncompressed streams. The synchronization of macro-frames often failed, causing the images from the different streams not to be composed properly. Instead of a single image of a person in one place, multiple incomplete images of the subject were displayed in several places onscreen. The increased variation in frame latency and resultant inter-stream jitter was due primarily to the high network load incurred in the uncompressed scheme. When we

added our compression and decompression to the transmission, this problem was alleviated dramatically. In the second experiment, we ran streaming tests from senders at UIUC and a renderer in UC Berkeley at five frames per second, where we observed similar enhancement of streaming quality upon the incorporation of compression and decompression. Extensive experimentation will be done in a future work to provide more concrete evaluation of the real-world performance of the system.

5 Related Work

The data model of a 3D image has a direct impact on the design of compression algorithms. Therefore, we divide the 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. In contrast to an ordinary 2D image, the depth image has extra depth information for every pixel. There are two major compression methods: *inter-stream* and *intra-stream* compression. This paper presents a method of intra-stream compression, in which each 2D depth image is independently analyzed to exploit the redundancy within itself. In inter-stream compression,[12, 13] 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 a 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 in broadcasting the reference image and the diminishing redundancy between images of larger viewpoint difference. To alleviate these problems, a two-level referencing and grouping scheme is applied. It should be noted that the intra-stream compression is complementary to the inter-stream compression and that the two can be naturally combined to improve overall performance.

The 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 computerized tomography (CT) scanners. The typically large size of the resulting voxel datasets has been a driving factor in research targeting compression of volumetric images. 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*[9, 10, 22] is the most important compression method for this data model; it is an extension of 2D wavelet compression techniques for 2D images. To perform a 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. Non-zero coefficients are then further processed using quantization and entropy encoding. Although the 3D wavelet transform achieves a very high compression ratio, its application to the 3D video of tele-immersive environments is not straightforward and the overhead of the algorithm may offset its advantages. For example, it is not beneficial to apply a 3D wavelet transform on a sparse dataset such as the contour of the subject. To make a wavelet transform more efficient in a tele-immersive context, a transformation function is needed to first pack the data into a more dense volumetric form while maintaining a high spatial coherence within unit blocks. This function may involve a 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 due to the fact that polygonal surfaces can be efficiently triangulated. The triangular representation of 3D geometry has two major components: vertex coordinates and triangles, and their associated properties including color. As complex scene representations may contain millions of triangles, the amount of data required to store such scenes may be quite large. On average, the number of triangles is twice the number of vertices, and each type of data may exhibit 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[23] adopted by the MPEG-4 standard. The spanning tree is constructed to exploit spatial coherence, or the observation that proximity of the nodes in the tree often implies geometric proximity of the corresponding vertices. The ancestors of the tree can be used as predictors and only the differences between the predicted and actual values 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 unit of compression is the *triangle*

strip which defines a particular order of traversal such that each new triangle can be represented by adding one vertex. Among triangle-based compression techniques proposed, *Edgebreaker*[11, 24] is the most advanced scheme; it encodes the traversal using an alphabet of five symbols with an overall performance of less than 2 bits per triangle. Recent efforts[24–26] aim to reduce the number of bits per triangle and extend the basic scheme to more arbitrary topologies. 3D compression based on triangular meshes achieves a very high compression ratio. However, the challenge of applying this technique lies in the real-time and automatic acquisition of 3D models, which is currently an active research area.

6 Conclusion

In this paper, we have presented the design, implementation, and comparison of 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 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 encodes the depth frames using lossless entropy coding. We demonstrated that both algorithms perform very well, achieving an average compression ratio over 26 to 1 (the first scheme) and 15 to 1 (the second scheme) for the 3D video captured from TEEVE 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 a typical frame compression time around 10 ms. For the decompression, the first scheme spends an average time of less than 2 ms per frame in decompression while the second scheme takes around 9 ms per frame. Both schemes maintain similar and very high visual quality. In our initial streaming experiment, we have observed a remarkable improvement in performance due to reduced network load when introducing our compression and decompression schemes into the video transmission. We further discussed the trade-off between the two compression schemes. The scheme based on color reduction was demonstrated to have better performance but is only suitable for 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 proposed a monitoring and switching mechanism to dynamically tune the compression performance according to the 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 into view);
- *Inter-stream redundancy* - Exploiting redundancy in the data received from different cameras, which aims to extend the previous inter-stream compression algorithm.

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

ACKNOWLEDGMENTS

We would like to acknowledge the support of this research by the National Science Foundation (NSF SCI 05-49242, NSF CNS 05-20182). The presented views are those of authors and do not represent the position of NSF. We would also like to thank Sang-hack Jung and Prof. Ruzena Bajscy of UC Berkeley for providing the tele-immersive videos.

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. Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S. hack Jung, and R. Bajscy, "Teeve: The next generation architecture for tele-immersive environments," in *IEEE International Symposium on Multimedia (ISM2005)*, (Irvine, CA, USA), 2005.
7. 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.
8. 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.
9. 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.
10. C. Bajaj, I. Ihm, and S. Park, "3d rgb image compression for interactive applications," *ACM Trans. Graph.* **20**(1), pp. 10–38, 2001.
11. J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Transactions on Visualization and Computer Graphics* **5**(1), pp. 47–61, 1999.
12. 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.
13. S.-U. Kum and K. Mayer-Patel, "Real-time multidepth stream compression," *ACM Trans. Multimedia Comput. Commun. Appl.* **1**(2), pp. 128–150, 2005.
14. J. Mulligan and K. Daniilidis, "Real time trinocular stereo for tele-immersion," in *International Conference on Image Processing*, pp. III: 959–962, 2001.
15. 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.
16. "Imagemagick, http://www.imagemagick.org/script/quantize.php."
17. "Color reduction, http://www.catenary.com/appnotes/colred.html."
18. "Zlib 1.2.2, http://www.zlib.net."
19. "Jpeg, http://www.jpeg.org."
20. 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.
21. "Basic compression library, http://bcl.sourceforge.net."
22. I. Ihm and S. Park, "Wavelet-based 3d compression scheme for very large volume data," *Graphics Interface* , pp. 107–116, June 1998.
23. G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM transactions on Graphics* **17**(2), pp. 26–34, 1998.
24. 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.
25. 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.
26. 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.