Stereo Frame Decomposition for Error-Constrained Remote Visualization

Steven Martin
The Ohio State University

Han-Wei Shen
The Ohio State University

ABSTRACT

As growth in dataset sizes continues to exceed growth in available bandwidth, new solutions are needed to facilitate efficient visual analysis workflows. Remote visualization can enable the colocation of visual analysis compute resources with simulation compute resources, reducing the impact of bandwidth constraints. While there are many off-the-shelf solutions available for general remoting needs, there is substantial room for improvement in the interactivity they offer, and none focus on supporting stereo remote visualization with programmable error bounds. We propose a novel system enabling efficient compression of stereo video streams using standard codecs that can be integrated with existing remoting solutions, while at the same time offering error constraints that provide users with fidelity guarantees. By taking advantage of interocular coherence, the flexibility permitted by error constraints, and knowledge of scene depth and camera information, our system offers improved remote visualization frame rates.

Keywords: remote visualization,stereo video,error constrained encoding

1. INTRODUCTION

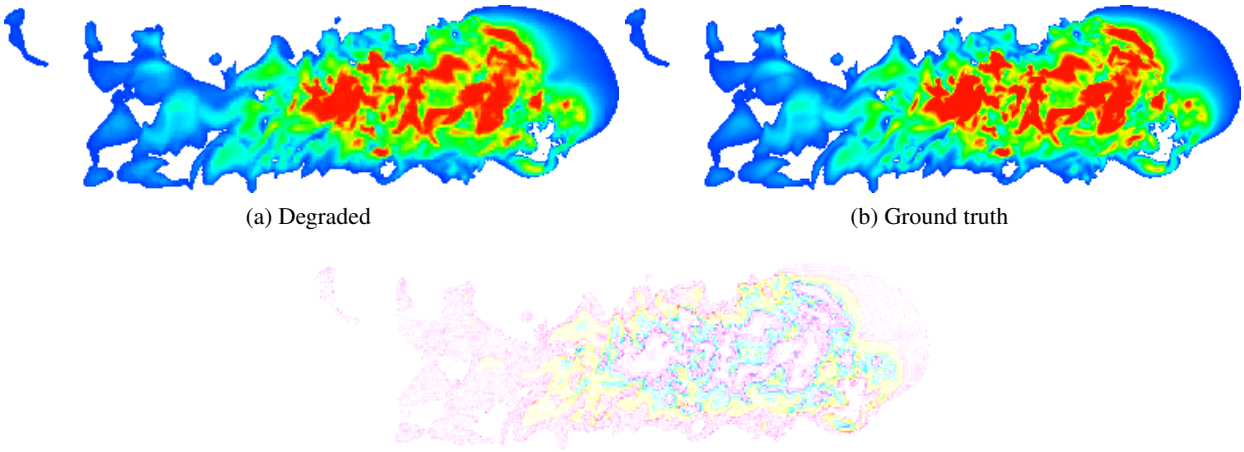
Continued growth of dataset sizes relative to bandwidth availability continues to provide challenges for visualization systems. Simultaneously, trends in computing continue to move applications into the cloud, with workstations being replaced by thin clients with limited bandwidth for cloud access. Additionally, stereo video solutions have become lower-cost and more common, with devices such as NVIDIA 3D Vision increasing the potential for their use by a wider range of visualization users.

Consider the case of an engineer seeking to interactively visualize the results of a simulation using a thin client, with very limited memory and compute resources, with simulation compute resources and their associated storage solutions being remotely located. Even for modestly sized datasets, the volume of data to analyze will be considerably larger than the number of pixels in one frame. Additionally, the size of the data will likely be larger than the memory available on the client. In this case, video-based remote visualization techniques are likely to be more effective than techniques that seek to move the simulation result data directly to the client.

Interactive visualization requires reasonably high framerates of at least 10 FPS.[1] This means that even for resolutions as low as 720p, with 24 bits per pixel per eye, greater than 52MiB/second is required for uncompressed transmission. Compression is clearly needed, but currently available lossy video codecs do not support visualization-specific error constraints that consider aspects such as what transfer functions are being used. Lossless compression could be an option, except that it wastes space by transmitting the information needed for lossless reconstruction rather than the minimal information needed to satisfy less restrictive error constraints. This can be observed in figures 1 and 3.

We propose a solution for video-based remote visualization that enables lossy coding of stereo video streams subject to user-provided error constraints. The stereo color and depth frame streams are decomposed into one depth, one color, and two residual streams. A novel video+depth coding algorithm is used to take advantage of coherence between the eyes and a novel residual coding technique is used to enable the use of off-the-shelf lossy codecs for color and depth stream transmission while adhering to error constraints. A novel framework is proposed that enables integration with existing remoting solutions and the utilization of multiple CPUs and GPUs. Visualization techniques that can benefit from the enhanced depth perception permitted by stereo, such as maximum intensity projection and shaded isosurfaces, are then used in experiments to demonstrate the efficacy of the technique.

This paper is organized as follows. The technique itself is described in detail in §3, and some suggested error constraints that may be used with it are described in §4. Finally, the results are discussed in §5.



(a) Degraded (b) Ground truth

(c) Color difference

Figure 1: The difference between the ground truth and the error-constrained degraded image is wasted information that would need to be transmitted, if lossless encoding were used.

2. RELATED WORK

Fundamentally, our technique seeks to adapt remoting techniques used for tasks other than visualization, developed in much larger markets that have funded considerable innovation, to work well for visualization. Of close relation to our technique are stereo reprojection and view synthesis, residual coding, stereo video coding, and other remote visualization schemes.

We consider two different categories of remote visualization tools: those specifically designed for visualization, and those designed for more general use. While the visualization-specific tools may offer better performance for a limited set of applications, the general purpose tools may be more accessible on a wider variety of platforms and be lower cost to implement. Our technique seeks to bridge the gap between more general remoting and stereo visualization.

TightVNC is a commonly used general remoting solution that offers a lossy JPEG codec, not supported by standard VNC, as an encoding option in addition to the standard lossless and simple lossy VNC codecs. A more advanced solution, demonstrated at SIGGRAPH 2011, is the NVIDIA Monterey Reference Platform, which leverages GPUs to enable low-latency streaming of H.264[2] video while supporting Android and Windows clients.

Within the context of visualization, MPEG has been used[3] to stream images to workstations. It was found to improve the temporal resolution of the models visible for the bitrates tested, so it is reasonable to think that H.264 may also perform well in that regard. ParaView offers a range of options for remote visualization, though none are similar to our technique.[4] More similar to our technique is the work using Chromium by Lamberti et al.[5] except that they do not offer error constraints on the results, stereo support, or the ability to apply remapping (§3.4) to improve compression performance.

Stereo and multiview video coding has been explored in contexts outside of visualization. The current H.264/MPEG-4 AVC Standard offers standard extensions to support stereo and multiview video coding.[6] These extensions define ways of packing multiple views of a scene into a single encoded image stream. However, they do not directly consider depth information, though it is mentioned as a possibility for future research.

Smolic et al.[7] provide a good overview of techniques for stereo video coding that are relevant to the context in which our technique operates. Three general categories of approaches are discussed: conventional stereo, video + depth stereo, and layered depth video.

Conventional stereo methods transmit color information separately for each eye. Özbek et al.[8] apply this using lossy codecs, adaptively choosing different bitrates for the two views. Rate balancing is also important in the context of our technique, and is addressed in §3.5. We compare our technique to a conventional (discrete) stereo technique, using off-the-shelf codecs, in section 5.4.

Layered depth video methods bear some similarity to the conventional stereo methods, except that they separate foreground and background objects, encoding them separately, possibly using video + depth encoding. Other techniques proposed by Moellenhoff et al.,[9] Jiang et al.,[10] Yan et al.,[11] and Yang et al.[12] are similar to the joint coding technique we use for comparison. They encode one or more primary views (in the case of multiview) then encode other views differentially with respect to the primary views.

Video + depth stereo techniques, such as those proposed by Smolic et al.,[7] Smolic et al.,[13] and Merkle et al.[14] transmit the color and depth information for one view, then use that information to reconstruct the image for both views. In contexts such as live action broadcast video, the depth buffer is not known. Thus, it must be estimated. This has been a long-standing image processing problem, and has been addressed by many works. Some techniques of particular relevance to video + depth encoding are proposed by Roy et al.,[15] Saxena et al.,[16] Yang et al.,[17] and Saxena et al.[18] However, it is still a fundamental source of error in depth frames.

However, for many visualization applications, depth information is available from the rendering process. For example, for isosurface rendering techniques, the depth information for the surface is known. Given this, we take a video + depth approach in our system, synthesizing the right frame from the color and depth buffers of the left frame.

For a video + depth solution to work, we must be able to synthesize one view from the other view. The depth information combined with the camera transformation associates each pixel in the source view with a world space position. These pixels can then be projected to the destination view using the camera transformation of the destination view.

Conceptually, this is simple, but there are some challenges. Firstly, what may be a dense sampling in one view may be a sparse sampling in another, leaving gaps. Secondly, with the typical asymmetric frustum parallel axis projection used in stereo rendering, occlusion will create gaps because all points visible from the right eye are not necessarily visible from the left eye. Pixels that have multiple world space positions contributing to their colors due to transparency introduce additional complexity which is handled via residual coding (§3.3) in our system.

Sample reprojection for view synthesis been looked at in previous works in multiview video compression. Martinian et al.[19] apply a technique considering camera information to reproject point samples. While the technique does have some similarity to ours in that they transform the point samples with matrix multiplications and use H.264, we use a different technique for filling the gaps (§3.1.1), apply a different camera transformation because we know the exact camera in our case (§3.1), propose a GPU implementation, and correct the results using a residual (§3.2) to enable its use for visualization.

In our system, residuals are required both because we use lossy codecs for the color streams and because view reprojection cannot, in general, reconstruct views completely. Residual coding has been applied previously in many widely used predictor-corrector based techniques. Examples of mainstream codecs include PNG (Portable Network Graphics), FFV1,[20] and LJPEG.[21] In techniques like these, a predictor operates to reconstruct samples from previously reconstructed samples, then a corrector stores the resulting residual from a comparison versus the ground truth. Conceptually, this is similar to our technique except that our residuals are lossy (controlled by the error constraint) and the predictor is a lossy video compression technique like H.264.

Another way of looking at residual coding in the context of image compression is to look at lossy wavelet-based methods such as lossy JPEG 2000 as predictor-corrector based methods. Effectively, the low pass filter of each filter stage acts as a predictor and the high pass filter acts as a corrector.[22] The choice of what wavelet coefficients to zero for the purposes of compression is analogous to the choice of residual entries to zero in the context of our residual decimation technique (§3.2.)

We are not aware of any techniques like ours that apply lossy residual coding, with visualization-centric error constraints (§4), as a corrector on top of mainstream video codecs.

3. TECHNIQUE

The goal of the technique is to reduce the bandwidth needed for remoting while adhering to user-provided error constraints. The approach we take accomplishes this, in addition to enabling support for the use of existing off-the-shelf lossy codecs that can take advantage of temporal coherence.

The stereo input frame, containing left color (LC), left depth (LD), and right color streams (RC), is decomposed into an encoded left depth primary stream (LDE), an encoded left color primary stream (LCE), an encoded left residual stream (LRE), and an encoded right residual stream (RRE) as in figure 2a. Different components are encoded using different

techniques to improve performance. By reconstructing (§3.1) the output right color stream using the LCE, LDE, and camera transformation (CX), coherence between the left and right images can be utilized.

The LCE and LDE can be encoded with off-the-shelf codecs, such as H.264, that are hardware accelerated on mobile devices. This allows for the technique to take advantage of motion compensation and other features offered by contemporary video codecs to further improve performance. Additionally, this enables piggybacking of our technique onto existing remoting solutions such as the NVIDIA Monterey Reference Platform.

Because lossy video codecs, in general, will be incapable of meeting the user-provided error constraints proposed in §4, the primary streams are augmented with residual streams as in figure 2a. Each of these residual streams contains an approximately minimal amount of information to correct the lossily encoded frames (generated using the techniques described in §3.2 and §3.3) to adhere to user-defined error constraints. With the algorithm described in section 3.4, colors are remapped using known properties of the transfer functions (XF) being used. This further improves compression by reducing the amount of information that needs to be stored in the residual.

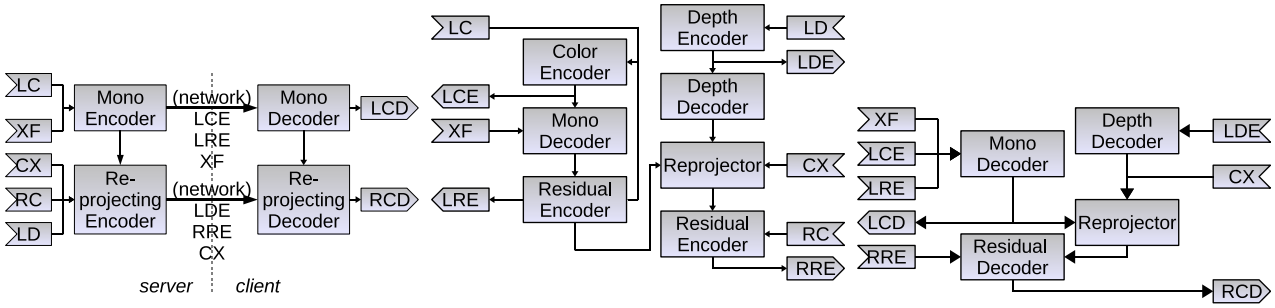
Details about the technique are in the following sections.

3.1 Reprojection

Substantial redundancy exists between sample values of each eye.[7] For example, consider looking at a surface from both eyes, as in figure 6b. If the frames for the two eyes are transmitted separately, then the color information for a point that is visible in both eyes is transmitted twice. The goal of reprojection is to minimize this kind of redundancy by only transmitting the color information for one eye while using camera and depth information to reconstruct the image for the other eye.

The reprojection algorithm uses the color image and depth image for one eye, combined with the camera transformations for both eyes, to synthesize the view for the other eye. In the data flows shown in figures 2b and 2c, this is used to reconstruct the right eye using information from the left.

We propose an algorithm similar in goal and overall approach to Martinian et al.,[19] in that we first synthesize a new view from existing views, then fill any gaps left by the view synthesis. However, the specifics of the approach we take are substantially different. In contrast to their approach, we are reconstructing a view from only one source view, with knowledge of the depth and camera projection information. Additionally, to avoid an unnecessary pass of color remapping (§3.4), we do not synthesize any new colors in the gap filling algorithm. Finally, we are more focused on efficiency as decoding needs to be lightweight enough for thin clients and needs to be easy to implement with a data-parallel programming paradigm such as that offered by CUDA.



(a) Framework (b) Reprojecting encoder (c) Reprojecting decoder

Figure 2: The framework decomposes the left and right frames into one depth stream, one color stream, and two residual streams in the encoder (§3), which are then reconstructed into left and right frames in the decoder. Because the depth stream generally takes much less space than the color stream, and the error introduced by reprojection is small, this yields better performance than encoding the streams separately. Additionally, transmission of partial residuals subject to user-defined error constraints enables fidelity guarantees for visualization applications.

3.1.1 View synthesis and Filtering

Each pixel in the source eye image has a camera space position defined by its depth buffer value and image position. The inverse camera transformation for the left eye is applied to pixels' camera space positions to produce world space positions. These world space positions are then projected to the destination eye using the camera transformation of the destination eye. The color of the source pixel is written to the single pixel at the projected destination position. Depth testing is applied so that the write closest to the camera will be used if the same pixel is written multiple times,. Pixels with the background color are not projected, as both framebuffers have been previously cleared with the background color. This algorithm is easily implemented with CUDA on platforms supporting the CUDA 1.1 atomicMin operation.

While this algorithm was found to produce images of reasonable quality, some gaps are present in the results due to occlusion and varying sampling rates of the world space from the slightly different perspective views. The residual codec could be applied to correct any artifacts, but this would increase the bitrate required. Instead, a gap filling filter is applied.

Each pass of the gap filling filter iterates over all of the pixels in the destination. Each pixel in the destination view that is not set to the background color, has an uninitialized depth value, and has at least one initialized neighbor will have its color and depth set from the neighbor with the minimum depth value. Passes are applied until the gaps have been sufficiently reduced. Like the reprojection algorithm, this gap filling algorithm is also easily implemented with CUDA.

The resulting destination eye image from this process will be an approximation of the view from the destination eye. Any artifacts remaining that violate the error constraint will be corrected by residual coding.

3.1.2 Depth encoding

Depth buffers are typically dominated by low spatial frequency regions with a few high spatial frequency regions at boundaries, similarly to many natural images, as in figure 6a. Because of this, lossy video codecs such as H.264 can be used to encode depth.[7] We observed that the bitrate required, as shown in graph 4b, to encode typical depth buffers while still providing good quality reprojections was substantially lower than the color bitrate required to provide color reconstructions, as in graph 4a. Additionally, storing the depth buffer at 8 bits per pixel was still found to be very effective for our test cases, given that small variations in depth value have very small effects on the spatial position for reprojection.

3.2 Residual Decimation

The lossy codec (such as H.264) used for the primary streams (LCE and LDE) produces frames different from the ground truth. The difference between the ground truth and the decoded lossy codec is the residual.

Because the goal is to perform lossy coding subject to an error constraint rather than to perform lossless coding, the entire residual is typically not needed. The residual can be simplified, producing a *decimated residual*, to reduce the amount of wasted information, as seen in figure 1.

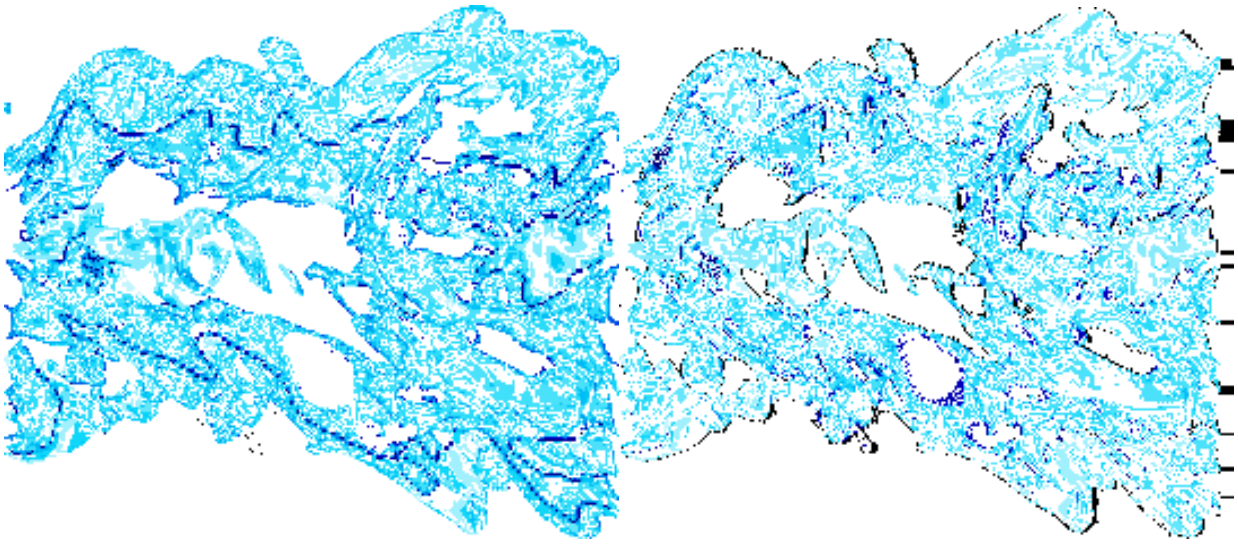
Residuals are decimated by replacing every nonzero value with a zero, when the replacement will not result in a user-specified error constraint (discussed in §4) being violated. This is similar in motivation to how JPEG applies quantization to produce repeated zero coefficients.[23] Figure 3 shows decimated residuals in comparison to undecimated residuals. Because the decimated residuals are less complex, they are easier to compress.

The decoder does not need to be aware of the decimation strategy, as it simply adds the decimated residual to the degraded image to correct the image. This means that other strategies could easily be applied for residual decimation. For example, a decimation strategy could be tuned to improve performance for the specific codecs being used, taking advantage of hardware support.

3.3 Decimated Residual Codec

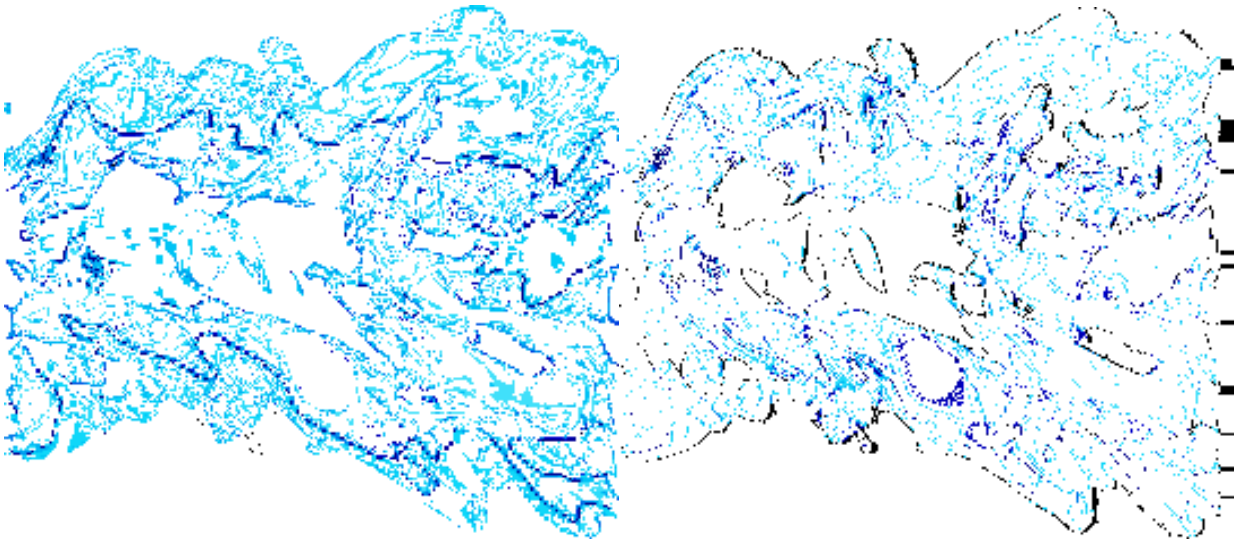
Once a decimated residual has been computed as in §3.2, it must be losslessly compressed using a technique that is well-suited to the characteristics of decimated residuals.

Typical decimated residuals look like figures 3c and 3d. They have sparse arrangements of single pixel differences, with narrow contiguous regions of pixels near boundaries. The eye whose image is reconstructed using the reprojection algorithm (§3.1) generally has more boundary artifacts. These are due to differences in visibility between the two eyes, as determined by eye separation.



(a) Original left

(b) Original right



(c) Decimated left

(d) Decimated right

Figure 3: The per-pixel color magnitudes of the residuals are shown for both eyes, before and after decimation subject to an error constraint, with darker colors meaning greater magnitude.

In general, the codecs used for encoding the decimated residuals must be good at encoding sparse single pixels and curves of pixels, rather than continuous regions with gradients. This means that DCT-based codecs such as JPEG, which were designed for photographic image encoding,[23] are not well suited to the task.

While many different codecs could be used for encoding the residual, we found the combination of Zero-Run-Length Encoding (ZRE) with Lempel-Ziv-Oberhumer (LZO[24]) to offer good performance in terms of encoding speed, compression ratio, and simplicity of implementation. For encoding, ZRE is first applied then LZO is applied. For decoding, the reverse is done.

Other configurations such as ZRE alone, LZO alone, applying a Hilbert Curve reordering of samples, and applying multi-frame temporal-differential coding were tested, but the ZRE+LZO scheme was found to be the most effective. Temporal-differential coding can work well on images that are mostly static, but these are not encountered in our test suite. Automatically switching between temporal-differential and non-differential based on scene dynamicity would be trivial.

ZRE collapses sequences of zeroes into a single value. This is similar in approach to run length encoding (RLE), except that ZRE does not need to store the symbol type for a repeating sequence, by assuming that it is zero. ZRE is well-suited to encoding residuals because our decimation technique produces long runs of zeros with very little continuous repetition of nonzero values.

LZO is a lossless sliding dictionary-based compression algorithm designed to offer fast encoding and and very fast decoding while still offering competitive compression rates.[24] This makes it ideal for our circumstances, where we need to encode and decode data in real-time on thin clients with limited compute resources. It has also been used previously in the context of remote visualization,[25] though they probably could have improved performance by applying some data preparation (similar to ZRE) before the LZO coding. Because ZRE tends to produce strings of words that have some repetition, LZO is well-suited for encoding.

Residual encoding and decoding can both be implemented in parallel, for use on mobile devices with multiple relatively-slow cores, with an acceptable compression performance penalty. This can be done by breaking the image into multiple blocks of pixels and applying the residual codec on a block-wise basis, with one thread per block.

3.4 Remapping

When an image is encoded with a lossy primary codec such as H.264, the set of output colors is generally not a subset of the set of input colors. By definition, any color in the output that is not in the input is a color in error. Knowing the set of colors in the input, we can remap the set of output colors back to that set of input colors. This offers two key benefits.

First, with remapping being applied to the decoded image before the residual is computed, much of the information that would otherwise be necessary to transmit as part of the residual to map samples back to the ground truth does not need to be sent. This is because a remapped sample is more likely to be correct in terms of the error constraints than an unremapped one, thus improving compressibility of the residual.

Secondly, some error constraints, such as ITFC (§4.3) and TFD (§4.2), require that there is a corresponding transfer function space position for every image space sample. Because the samples produced by a lossy primary codec are not necessarily in the transfer function, there must be a mapping between possible result colors from the lossy primary codec and the transfer function space. The remapping operation maps a set of samples whose values belong to a set B to values within a set A. We experimented with a couple different operators:

Simple inverse transfer function Given a sample color value, the color value with the smallest color difference (as defined in CIE 1976[26]) to it within the transfer function is used.

Frame history For each frame, the mapping of observed samples to the ground truth samples is recorded, permitting estimation of the conditional probability that a sample will have some remapped value $y \in A$ given an observed value of $x \in B$. For remapping a frame, the probabilities of the previous frame are used.

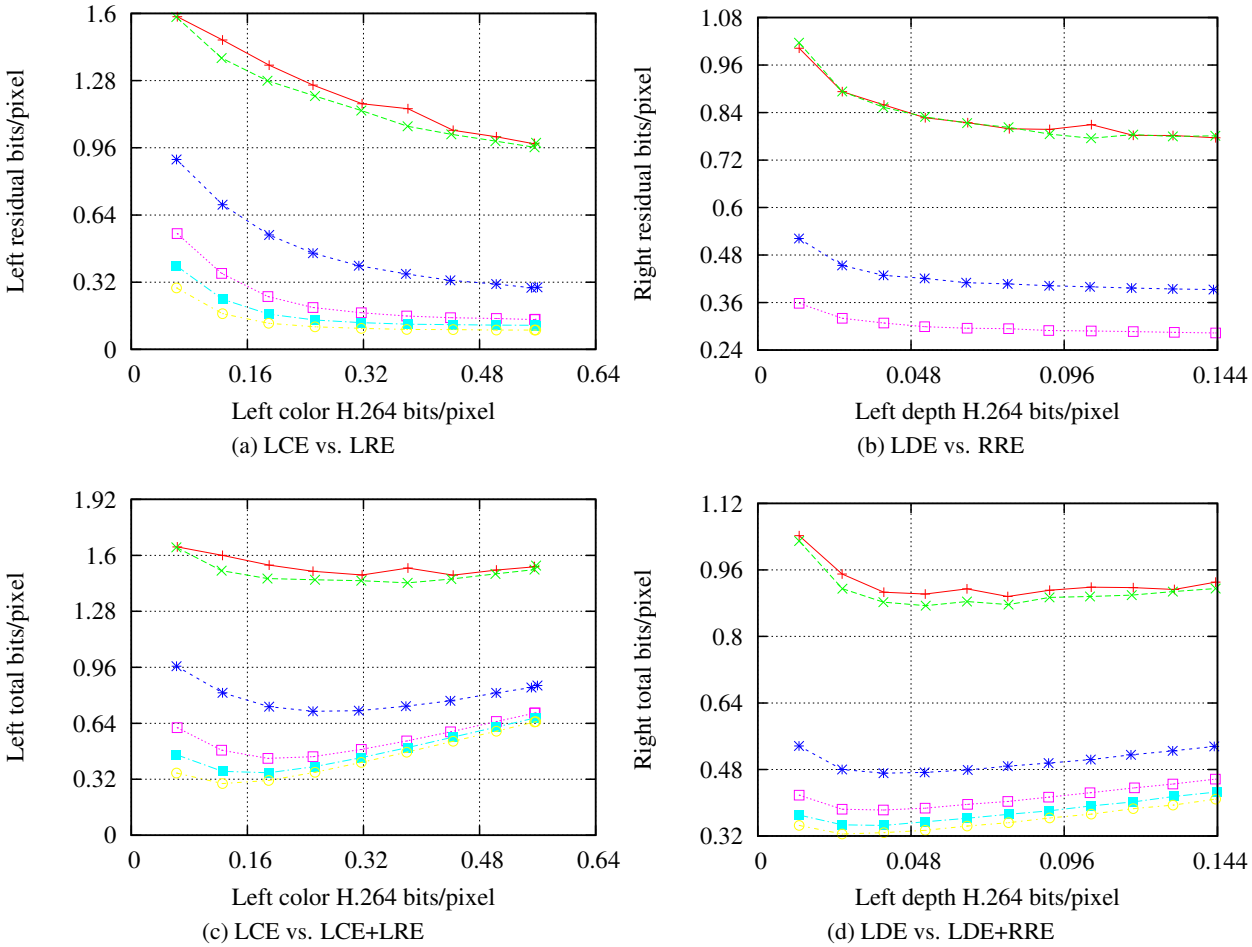
In practice, the simple inverse transfer function method was found to outperform the frame history method and was substantially more simplistic to implement. The frame history method was found to yield conditional probabilities that tend to produce incorrect results, because they depend on lossily encoded samples.

The inverse transfer function can be easily computed in parallel on a GPU using a map-reduce algorithm to find the nearest color in the transfer function for each color that may occur in a decoded image. This results in a 3D lookup table mapping decoded image colors to transfer function colors. In cases where some colors are not invertible due to ambiguities, errors in color remapping will be corrected by the residual if they violate the user-defined error constraint.

3.5 Rate Balancing

As can be seen in figure 2a, the technique decomposes the input into two primary streams and two residual streams. The bitrate used for the codec (such as H.264) for each primary stream is configurable and may vary over time.

The bitrates of the residuals are strongly related to the artifacts introduced by the primary codec streams, which implies that they are strongly related to the bitrates of the primary codec streams. As shown in table 1 and figure 4a, an increase in the bitrate of the left color primary stream (LCE) yields a decrease in the bitrate of the left residual stream (LRE). This is reasonable because the only two things that can vary in bitrate that contribute to the left color frame are the LCE and the



(a) LCE vs. LRE (b) LDE vs. RRE

(c) LCE vs. LCE+LRE (d) LDE vs. LDE+RRE

Figure 4: Increasing the LCE (left color encoded) bitrate decreases the LRE (left residual encoded) bitrate. Increasing the LDE (left depth encoded) bitrate decreases the RRE (right residual encoded) bitrate. The curves, from top to bottom, have ITFC error constraints of 0,6,12,18,24,and 32. More-restrictive constraints tend to require higher LCE and LDE bitrates for optimal performance.

LRE. The left depth primary stream (LDE) bitrate does not affect the LRE bitrate because the LDE is not an input to any functional blocks producing the left color frame (LCD.)

Similarly, as can be seen in figure 4b, an increase in the bitrate of the LDE yields a decrease in the bitrate of the right residual stream (RRE.) However, the bitrate of the LCE is largely decoupled from the RRE because the reprojection process is done using the post-residual-application left color frame (LCD), not the raw decoded frame from the LCE. A circumstance under which the coupling may become substantial is if very loose error constraints are used, but in practice we did not find any reasonable error constraints that produce substantial coupling.

This decoupling allows for the optimal bitrate to be chosen independently for the LDE and the LCE, which substantially simplifies the optimization problem. Graphs 4c and 4d exhibit overall bitrate as a function of LDE and LCE bitrate. As would be expected, the optimal bitrate increases as the error constraint is tightened. The curves vary in data, transfer function, error constraint, and primary codec-dependent ways, so a closed-form formula to find the optimal bitrate is not practical.

Applying a simple direct optimization approach was found to work well over a range of bitrates. First, a function in the form $y = \frac{a}{x} + b$ is fit to the LRE (y) and RRE (y) bitrates as a function of LCE (x) and LDE (x) bitrates, respectively. Independent optimization problems, in the form $\operatorname{argmin} \frac{a}{x} + x + b$ can be defined for the LCE bitrate and LDE bitrate. This can be directly solved, by solving for a positive zero of $1 - \frac{a}{x^2}$. One challenge with this is finding the values of a and b.

One approach is that, periodically, the system can sweep the bitrate x to find a set of (x,y) tuples to fit for the values a and b. Different curves can be applied for the fit, as needed. In general, the overall bitrate is not strongly sensitive to small changes in the LCE or LDE bitrate.

An alternative to direct optimization is to apply PID (proportional-integral-derivative) control for the bitrate for the left eye (LCE + LRE), and for the right eye (LDE + RRE.) Because the system is not ill-conditioned, and has fairly limited frame latency, standard techniques like Ziegler-Nichols can be used to find the PID coefficients for the use cases of interest. If codecs are used that have substantial frame latency, such as H.264 with B-frames, the proportional gain possible will be limited due to the potential of oscillations.

Finally, manual control of the bitrates can be a reasonable alternative, if the implementation cost of the other techniques is prohibitive, because the curves in graphs 4c and 4d have easy to observe global minima.

	Left Color	Left Residual	Left Depth	Right Residual
Left Color	1.00	-0.96	0.00	-0.02
Left Residual	-0.96	1.00	0.00	0.02
Left Depth	0.00	0.00	1.00	-0.83
Right Residual	-0.02	0.02	-0.83	1.00

Table 1: Cross correlations were computed between the bitrates for many observed trials.

4. ERROR CONSTRAINTS

Error constraints control how different a frame can be from the ground truth. Within the context of our system, an error constraint is defined by implementing a boolean function $T(c,c')$ where c is the ground truth color and c' is a potential replacement color for that ground truth color. If and only if this function returns true may c be allowed to be replaced by c'.

We suggest three different error constraints that may be used in different circumstances depending on user intent and transfer function properties: color difference (CD), transfer function distance (TFD), and integrated transfer function contrast (ITFC). Other error constraints can be applied, if needed for a particular application.

The TFD and ITFC error constraints all share one requirement: there must be a mapping from remapped(§3.4) image space color to data-domain value. Examples of applications where this may be appropriate are transfer function-shaded isosurface rendering, maximum intensity projection volume rendering, and volume rendering where there is no semi-transparency. A related limitation is that the error constraints are most useful when there is a one-to-one mapping from image domain positions to data domain positions.

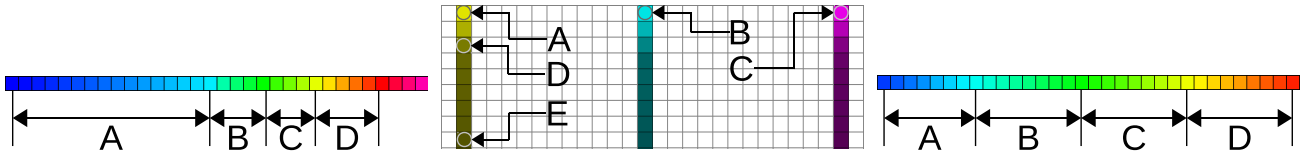
4.1 Color Difference

Simply considering the color difference, in terms of CIE 1976,[26] can be a viable option in cases in which the important measure of difference within a transfer function is a difference in color. It is defined as:

$$T_{\mathrm{CD}}(c,c') = \begin{cases} \text{true} & |c-c'| < D_{\max} \\ \text{false} & \text{otherwise} \end{cases} \tag{1}$$

where $|c-c'|$ is the CIE 1976 color difference between c and c' and $D_{\max}$ is the error constraint value.

This constraint may be a good choice when the color space distances within the transfer function are substantially different from the corresponding transfer function space distances (§4.3). This commonly occurs when users want to emphasize some ranges of values more than other ranges of values. Figure 5a exemplifies this kind of transfer function.



(a) Nonlinear context-free (b) Nonlinear context-sensitive (c) Linear context-free

Figure 5: Different types of transfer functions are appropriate for different types of error constraints

4.2 Transfer Function Distance

Considering the distance within the transfer function space itself is another alternative to color distance. It requires that the input colors c and c' each correspond to a single position within the transfer function, which can be accomplished with remapping (§3.4). It is defined as:

$$T_{\text{TFD}}(c,c') = \begin{cases} \text{true} & |M(c) - M(c')| < D_{\max} \\ \text{false} & \text{otherwise} \end{cases} \tag{2}$$

where $|M(c) - M(c')|$ is the Euclidean distance between $M(c)$ and $M(c')$. $M(c)$ and $M(c')$ are transfer function positions of the colors c (ground truth) and c' (degraded), respectively, within the transfer function. $M(c)$ and $M(c')$ are not defined for colors that are not within the transfer function, but this does not matter because remapping is used to map colors to the set of colors in the transfer function. $D_{\max}$ is the error constraint value.

This constraint may be a good choice when color space distances within the transfer function are similar to a linear scaling of the transfer function space distances in the transfer function. Figure 5c exemplifies this kind of transfer function. This stands in contrast to figure 5a, which is likely to be inappropriate for TFD because color distances in figure 5a are not linearly proportional to transfer function space distances.

4.3 Integrated Transfer Function Contrast

In some transfer functions, the distance between two values depends not only on the values themselves, but also on the path between the instances of the values within the transfer function. We call these transfer functions context-sensitive.

For example, consider the transfer function in figure 5b, where A, B, and C refer to specific samples within it. If the color space distance metric (CD – §4.1) were used, then $|A-B|$ would be similar to distance $|A-C|$. If this was the intent of the user then this may be acceptable. However, if the intent of the user was for A to be more dissimilar from C than B, then CD is not an appropriate metric. Additionally, if the intent was for A to be more dissimilar from D than D is from E, then the transfer function distance (TFD – §4.2) is not appropriate. Integrated transfer function contrast (§4.3) can resolve both of these problems inherent in context-sensitive transfer functions, resulting in the distance $|D-A|$ being greater than $|E-D|$ and $|A-C|$ being greater than $|B-C|$.

The ITFC error constraint is defined as follows:

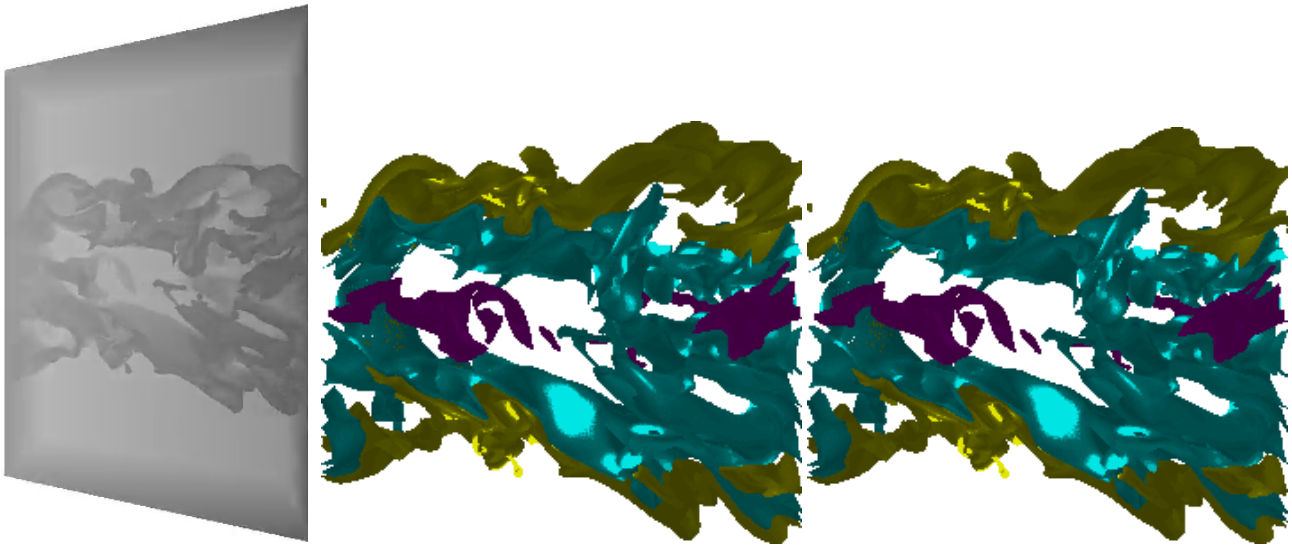
$$v = M(c') - M(c) \tag{3}$$

$$T_{\text{ITFC}}(c,c') = \begin{cases} \text{true} & \int_0^1 \nabla C(M(c)+uv) \cdot \frac{v}{|v|} du < D_{\max} \\ \text{false} & \text{otherwise} \end{cases} \tag{4}$$

where $\nabla C(M(c)+uv) \cdot \frac{v}{|v|}$ is the directional derivative of color in terms of CIE 1976[26] color differences, which is effectively contrast per unit of distance in transfer function space. $M(c)$ is the mapping of colors to transfer function positions.

5. RESULTS

Experiments were conducted to examine how the technique performs with different error constraints, primary codecs, eye separations, transfer functions, and datasets.



(a) Left Depth (b) Left and right color

Figure 6: Stereo rendering of the combustion dataset (§5.1) using isosurfacing with a 2D transfer function (figure 5b)

5.1 Data sets

Two datasets were used for experimentation: the combustion dataset used by Akiba et al.[27] and the plume dataset used by Akiba et al.[28] The combustion dataset was rendered using shaded isosurfacing. The plume dataset was rendered using maximum value projection.

5.2 Lossy Codecs

Multiple lossy primary codecs were tried for encoding the color and depth primary streams (LCE, LDE). Two different versions of H.264[2] were tried, one being the main profile, and one being the constrained baseline profile. The constrained baseline profile (BCP) is more commonly available on mobile devices, which would be a common client target platform for our system. Additionally, the BCP does not use B-frames, which introduce more frame latency.[29] MJPEG and MPEG-4 were both tried to check if more simplistic primary codecs could still yield good performance.

The H.264 main profile outperformed all of the other codecs in all of the tests. This is reasonable, considering that it offers more features for motion estimation and bidirectionally-aware temporal coding than the other codecs. However, the substantially simpler H.264 BCP offered performance very similar (within 5%, in terms of bitrate) to the H.264 main profile, so it may be a better choice for many applications, especially when latency matters. Thus, for our technique, we apply the H.264 BCP as the codec for the color and depth primary streams (LCE, LDE) for all subsequent comparisons.

5.3 Lossless Codecs

Lossless codecs are required for comparison for two primary reasons:

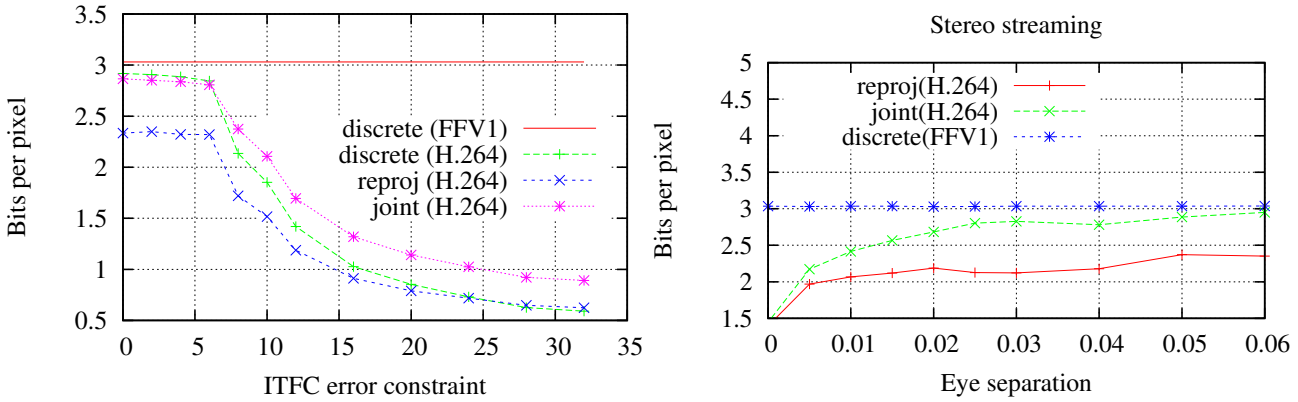
1. Lossy codecs will tend to cause a color shift in the results. While this color shift may be acceptable for some error metrics, such as CD (§4.1,) it will not enable computation of the TFD (§4.2) or ITFC (§4.3) error constraints because there is not a clear mapping from colorspace to transfer function space, without a remapping operator such as that used by our technique (§3.4.)

2. It is not, in general, possible to directly compare the output of the lossy codecs mentioned in §5.2 versus our system using the same lossy codecs because these lossy codecs do not offer compatible error constraints and may not be able to adhere to the error constraints even at the highest quality available. This can be observed in figure 4a for the H.264 codec, where the residual bitrate required to attain lossless or even near-lossless performance does not converge to zero, even as the maximum possible bitrate produced by the H.264 codec is approached.

We experimented with two lossless codecs: LJPEG[21] and FFV1.[20] Both of these are prediction-correction based methods that apply a predictive pass, then encode the residual between the prediction and the actual image.

LJPEG was chosen because it is a well known codec, and FFV1 was chosen because it is a widely available[30] codec that often outperforms LJPEG. In fact, we found that FFV1 outperformed LJPEG by such a substantial margin in our test cases that we only present results of FFV1 versus our technique.

5.4 Compression Performance



(a) Effects of the error constraint on bitrate

(b) Effects of eye separation on bitrate

Figure 7: The benefit of using our reprojection technique or a joint coding technique over discrete coding techniques increases as the eye separation is reduced, as explained in §5.4.2. Additionally, the benefit of using the reprojection technique increase as the error constraints are loosened.

Experiments were performed to examine the sensitivity of the compression performance to error constraints, eye separation, and alternate datasets. Four different techniques were compared: our reprojection technique, discrete coding of frames with a lossless codec, discrete coding of frames with a lossy codec combined with residual coding, and joint coding of frames.

Discrete coding is simply the case where each eye is treated as an entirely separate video stream, with no coupling between the two eyes. Effectively this means transmitting the LCE, LRE, RCE, and RRE, but not the LDE. It is implemented as two monocular codecs running in parallel. Other techniques (such as those proposed by Özbek et al.[8] and Smolic et al.[7]) do use this, though they may use different bitrates for each eye. The joint technique is similar to the discrete technique, except that the LCE is also used as the RCE. Basically, this means that only one primary stream, LCE, is sent over the network along side the LRE and RRE, with the difference between the two eyes being encoded entirely into the RRE. Conceptually this is similar to many other techniques that have been used for stereo video transmissions, especially when depth information is unknown or difficult to compute. Some examples of these techniques are those proposed by Moellenhoff et al.,[9] Jiang et al.,[10] Yan et al.,[11] and Yang et al.[12]

In all of our test cases, including lossless ones, our technique substantially outperformed discrete lossless and discrete lossy video compression. It also outperformed simple joint coding except for very loose error constraints.

5.4.1 Sensitivity to Error Constraint Values

As the error constraint is loosened (increased), the compressibility increases. This is because the number of entries that can be decimated (as in §3.2) in the residual increases, yielding greater compressibility.

Figure 7a exhibits this. In this case the combustion dataset was rendered with an eye separation of 0.03, producing images like figure 6b, for different ITFC error constraint values.

It is reasonable that the technique would outperform the discrete lossless technique because, even for an error constraint of 0, there is still wasted information sent due to the fact that the colors in the image are constrained to those in the transfer function by the (figure 5b) remapping operation. Additionally, with the reprojection technique, we avoid duplication of interocularly-coherent color information that occurs with all discrete techniques.

Similarly, it can be seen that the reprojection technique outperforms the joint coding technique until the error constraint becomes very loose. This is reasonable because the joint technique has to encode the disparity due to the difference in the camera projection between the left and right eye, while the reprojection technique only needs to encode corrections for regions where the depth is either inaccurate, or where occlusion resulted in a lack of samples. However, with very loose error constraints, there is sufficient freedom within the constraint to ignore much of the data in the residual due to the camera projection difference, thus there is little benefit to sending the depth information over the communication channel for reprojection in this case.

5.4.2 Sensitivity to Eye Separation

Intuitively, one can expect that the compression performance of a technique that reprojects the pixels from one eye's camera space to the other eye's camera space should decrease as the separation between the eyes increases because the disparity between the two eyes will increase.[10] This effect is indeed seen in graph 7b. For the camera configuration of our test cases for our test scenes, an eye separation of 0.05 is about the maximum that can be used without causing eye strain or completely preventing stereopsis. Even for those extreme eye separations the reprojection technique outperforms the joint and discrete codecs, and for the less extreme eye separations its performance is further improved.

Experiments were also run for different datasets, using different rendering techniques, different transfer functions, and different error constraints. In all cases we found the technique to continue to exhibit roughly the same compression performance behavior with respect to changes in error constraints.

This is expected because most datasets and transfer functions will permit our system to improve compression performance over lossless techniques by utilizing coherence between the eyes and the color information limits imposed by the transfer functions using reprojection (§3.1) and remapping (§3.4.)

5.4.3 Monocular Viewing

Experiments were also performed to verify that the concept of transmitting residuals in addition to a primary codec stream was also useful for monocular streams. The results were approximately the same as the discrete techniques in figure 7a, though with one half the bitrate. This is because the frames for both eyes are very similar, and we only need to transmit the frame for one of the two eyes in the monocular case. Even though there is no potential for taking advantage of coherence between eyes in the monocular case, there is still the potential for utilizing the flexibility permitted by error constraints, motion estimation, and color information limits imposed by the transfer functions.

6. CONCLUSION

We have proposed a video-based remote visualization solution enabling transmission of stereo video streams using efficient lossy codecs while adhering to user-defined error constraints. A novel video+depth coding algorithm is used to take advantage of coherence between eyes and a novel residual coding technique is used to enable the use of arbitrary lossy codecs for transmitting primary streams. The novel framework proposed enables integration with existing remoting solutions such as VNC and NVIDIA Monterey, as well as the utilization of multiple CPUs and GPUs. The system enables transmission of remote stereo visualization video streams at lower bitrates than would be possible with traditional lossless techniques, while providing support for visualization-specific error constraints.

REFERENCES

[1] Zuiderveld, K., van Ooijen, P., Chin-A-Woeng, J., Buijs, P., Olree, M., and Posti, F., "Clinical evaluation of interactive volume visualization," in [*Visualization '96. Proceedings.*], 367 –370 (November 1996).

[2] Sullivan, G. and Wiegand, T., "Video compression - from concepts to the H.264/AVC standard," *Proceedings of the IEEE* **93**, 18 –31 (January 2005).

[3] Ellsworth, D., Green, B., Henze, C., Moran, P., and Sandstrom, T., "Concurrent visualization in a production supercomputing environment," *IEEE Transactions on Visualization and Computer Graphics* **12**, 997 –1004 (September-October 2006).

[4] Cedilnik, A., Geveci, B., Ahrens, K., and Favre, J., "Remote large data visualization in the ParaView framework," *Eurographics Parallel Graphics and Visualization* , 163–170 (2006).

[5] Lamberti, F. and Sanna, A., "A streaming-based solution for remote visualization of 3D graphics on mobile devices," *Visualization and Computer Graphics, IEEE Transactions on* **13**, 247 –260 (March-April 2007).

[6] Vetro, A., Wiegand, T., and Sullivan, G., "Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard," *Proceedings of the IEEE* **99**, 626 –642 (April 2011).

[7] Smolic, A., Mueller, K., Merkle, P., Kauff, P., and Wiegand, T., "An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution," in [*Proceedings of the 27th conference on Picture Coding Symposium*], *PCS'09*, 389–392, IEEE Press, Piscataway, NJ, USA (2009).

[8] Özbek, N., Tekalp, A. M., and Tunali, E. T., "Rate allocation between views in scalable stereo video coding using an objective stereo video quality measure," in [*ICASSP (1)*], 1045–1048 (2007).

[9] Moellenhoff, M. S. and Maier, M. W., "Transform coding of stereo image residuals," *IEEE Transactions on Image Processing* **7**(6), 804–812 (1998).

[10] Jiang, J. and Edirisinghe, E. A., "A hybrid scheme for low bit-rate coding of stereo images," *IEEE Transactions on Image Processing* **11**(2), 123–134 (2002).

[11] Yan, L., Zhaoyang, Z., and Ping, A., "Stereo video coding based on frame estimation and interpolation," *IEEE Transactions on Broadcasting* **49**, 14 – 21 (March 2003).

[12] Yang, Y., Stankovic, V., Zhao, W., and Xiong, Z., "Multiterminal video coding," in [*ICIP (3)*], 25–28 (2007).

[13] Smolic, A., Müller, K., Stefanoski, N., Ostermann, J., Gotchev, A., Akar, G. B., Triantafyllidis, G. A., and Koz, A., "Coding algorithms for 3DTV - a survey," *IEEE Trans. Circuits Syst. Video Techn.* **17**(11), 1606–1621 (2007).

[14] Merkle, P., Wang, Y., Muller, K., Smolic, A., and Wiegand, T., "Video plus depth compression for mobile 3D services," *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009* , 1–4 (2009).

[15] Roy, S., "Stereo without epipolar lines: A maximum-flow formulation," *International Journal of Computer Vision* **34**(2-3), 147–161 (1999).

[16] Saxena, A., Chung, S. H., and Ng, A. Y., "3D depth reconstruction from a single still image," *International Journal of Computer Vision* **76**, 53–69 (January 2008).

[17] Yang, R., Pollefeys, M., Yang, H., and Welch, G., "A unified approach to real-time, multi-resolution, multi-baseline 2D view synthesis and 3D depth estimation using commodity graphics hardware," *International Journal of Image and Graphics (IJIG* **4**, 2004 (2004).

[18] Saxena, A., Schulte, J., and Ng, A. Y., "Depth estimation using monocular and stereo cues," in [*In IJCAI*], (2007).

[19] Martinian, E., Behrens, A., Xin, J., and Vetro, A., "View synthesis for multiview video compression," in [*IN PICTURE CODING SYMPOSIUM*], (2006).

[20] Niedermayer, M., "Description of the FFV1 video codec," *http://mplayerhq.hu/ michael/ffv1.html* (Mar. 2004).

[21] Schaefer, G., Starosolski, R., and Zhu, S. Y., "An evaluation of lossless compression algorithms for medical infrared images," in [*Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*], 1673 –1676 (January 2005).

[22] Secker, A. and Taubman, D., "Highly scalable video compression with scalable motion coding," in [*ICIP (3)*], 273–276 (2003).

[23] Wallace, G. K., "The JPEG still picture compression standard," *Commun. ACM* **34**, 30–44 (April 1991).

[24] Oberhumer, M., "LZO real-time data compression library," *http://www.oberhumer.com/opensource/lzo/* (Aug. 2011).

[25] Engel, K., Sommer, O., and Ertl, T., "A framework for interactive hardware accelerated remote 3D visualization," in [*Proceedings of Joint Eurographics - IEEE VGTC Symposium on Visualization*], 167–177 (2000).

[26] Jain, A. K., [*Fundamentals of digital image processing*], Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1989).

[27] Akiba, H., Fout, N., and Ma, K.-L., "Simultaneous classification of time-varying volume data based on the time histogram," in [*EuroVis*], 171–178 (2006).

[28] Akiba, H., Ma, K.-L., and Clyne, J., "End-to-end data reduction and hardware accelerated rendering techniques for visualizing time-varying non-uniform grid volume data," *International Workshop on Volume Graphics* , 31–225 (2005).

[29] Flierl, M. and Girod, B., "Generalized B pictures and the draft H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology* **13**, 587 – 597 (July 2003).

[30] The MPlayer Project, "MPlayer - The Movie Player," *http://www.mplayerhq.hu/* (2011).