Intra-Stream Encoding for Multiple Depth Streams

Sang-Uok Kum
kumsu@cs.unc.edu

Ketan Mayer-Patel
kmp@cs.unc.edu

University of North Carolina at Chapel Hill
CB #3175, Sitterson Hall
Chapel Hill, NC 27599 USA

ABSTRACT

Using multiple cameras, a dynamic real world scene can be captured, represented, and streamed for realistic interaction in a 3D immersive system. One method to represent the dynamic environment is to use multiple depth streams – streams with color and depth information. However the captured data is too large to be streamed uncompressed. Fortunately, these data exhibit spatial and temporal coherence that can be used for compression.

In this paper, we examine the initial step in compressing multiple depth streams. To use spatial coherence between streams to compress multiple depth streams, base reference streams – intra-streams – have to be encoded without the use of other depth streams. We investigate encoding of this base depth stream using only temporal coherence. The proposed algorithm extends video stream encoding techniques to encode color and depth. We show that for encoding a depth stream which requires high quality, such as base reference streams, using separate motion vectors to encode color and depth performs better than using a single motion vector.

1. INTRODUCTION

Recently, there has been growing interest in systems for capturing, representing, and transmitting dynamic real world scenes. The acquired real world environments are used in systems for realistic immersive real-time interaction such as 3D immersive systems [7], tele-immersion video conferencing systems [12, 22, 2], 3D TV [6, 19], and medical consultation and surgical training [1].

These systems use multiple cameras for dynamic scene acquisition. One approach for scene acquisition is to use multiple video streams captured from varying viewpoints in order to form a dynamic light field and then to use Image Based Rendering (IBR) techniques for generating virtual views of the scene [19]. Another approach is to use the imagery from the multiple cameras to derive a 3D representation of the environment that is then transmitted with color [12, 6, 22, 2, 7]. The dynamic light field approach has the advantage of being able to handle complex scenes where 3D reconstruction would be difficult. Also dynamic light field systems can transmit the captured video streams with little further processing, while deriving 3D information requires extensive processing before transmission. However more cameras are needed for dynamic light fields since light fields need to be acquired more densely to be as effective [4].

A common 3D representation for dynamic environments is to use multiple *depth streams* [6, 12, 22, 26]. Multiple depth streams are needed to properly represent a dynamic scene. In general, increasing the number of depth streams generated will increase the quality of the reconstruction from virtual viewpoints. A depth stream is like a traditional video stream that has been augmented with per-pixel depth information. Typically this depth information is calculated using stereo-based vision algorithms. The highly parallelizable nature of these algorithms makes real-time depth generation possible [13]. Furthermore, there is active research in developing cameras that can acquire per-pixel depth and color information directly [25, 3].

In this paper, we are concerned with one specific aspect of multiple depth stream systems. Specifically, we are interested in investigating methods for extending traditional video encoding techniques in order to represent and transmit depth streams.

Uncompressed multiple depth streams, even sparsely sampled, require significant transmission bandwidth. The data set used for this paper has 8 depth streams with 1024x768 resolution and 32 bits per pixel – 1 byte each for RGB and depth – at 15 frames per second (fps). Uncompressed, this would require 2.8 Gb/sec of bandwidth. Therefore different methods have been proposed for compressing and transmitting multiple depth streams.

One proposed approach is to merge images from multiple depth streams to create a layered depth image (LDI) [20], and use a stream of LDIs [6]. The stream is compressed using traditional video compression techniques, such as MPEG-2 [8] and AVC [10], which suffices for its target application of 3D TV. But LDIs impose limits on sampling multiple depth images due to its fixed resolution [5]. It also centralizes processing which makes it difficult to scale such a system and increase the number of depth streams used.

Another approach is to exploit the fact multiple depth streams exhibit strong spatial coherence since they are capturing the same environment from different angles. The depth value associated with a pixel in one depth stream (we will refer to it as the *non-reference stream or inter-stream*) can be used to reproject that pixel into the frame of another depth stream (we will refer to this second stream as the *reference stream or intra-stream*). If the color of the corresponding pixel in the reference stream is similar to the color of the reprojected pixel of the target inter-stream, we no longer have to encode the color information for that pixel in the target stream, but instead simply encode the depth and indicate that the color can be derived from the reference stream.

Ultimately, though, there will exist some stream(s) that serve as the *base reference stream(s)* that must be coded independently without the use of information from any other streams. This paper is about encoding these base reference depth streams. More specifically, this paper attempts to extend existing single stream video encoding techniques that make use of motion compensation. The basic idea is to simply add depth as a fourth plane of information to the existing three plane representation for color. Our experiments show how well such a scheme works and investigates several ways of extending motion compensation to the new depth plane. Surprisingly, we find that it is sometimes more efficient (in a compression rate sense) to calculate and encode separate motion vectors for the color planes and the depth plane than to use a single motion vector for all four planes.

Our contributions presented in this paper include:

- An algorithm for encoding depth streams using motion compensation.
- An argument for using separate motion vectors for depth and color when encoding a base reference depth stream.
- A metric called the *accumulative residual histogram*, for measuring the effectiveness of predicted reference images from motion compensation.

2. RELATED WORK

Video stream encoding, such as MPEG-2 [8] and MPEG-4 [9], exploit temporal coherence between frames for effective compression. The temporal coherence is utilized by estimating the current color from the reference frames using motion vectors. Each frame is divided into blocks, and for each block a motion vector is assigned. The motion vector points to a reference block in the reference frame. The difference from this reference block is then quantized. The quantization scale is used to control image quality and compression rate – higher quantization leads to higher compression but lower image quality. The motion vector and the quantized values are then encoded using entropy encoding.

A light field represents a static 3D scene by modeling its light rays with a set of 2D images [16]. These images have very high spatial coherence since a point in the scene is observed in multiple images. Magnor and Girod [17] used this spatial coherence to compress a static light field. A dynamic light field was used for a 3D TV system [19]. The dynamic light field encoding for the system did not use spatial coherence between streams and was encoded by processing each image stream separately and using standard video encoding for each stream. The MPEG Ad-Hoc Group on 3D Audio and Video (3DAV) [21] are currently investigating standardization for encoding of dynamic light fields and are in the process of evaluating various proposals.

Depth streams (i.e., streams of images with color and depth information per pixel) have also been used for a 3D TV system [6]. The color and depth are encoded as separate streams in order to be backward compatible with conventional TV transmission. Therefore various different video codecs were investigated for encoding the depth information as stream of grayscale images. Also a representation format for depth streams is proposed in MPEG-4 AFX [11], but it does not define the encoding of the depth streams.

Multiple depth streams used for tele-immersion [22] were compressed by removing points redundant in the reference depth streams from the non-reference depth streams [15]. While this approach is scalable to the number of depth streams, the performance on real world data sets is not excellent due to imperfect depth values [14]. Also it only takes advantage of spatial coherence between different streams but not the temporal coherence between frames of the same stream. Zitncik et al. [26] compressed multiple depth streams used for 3D video by using temporal coherence to compress reference depth streams and spatial coherence to compress non-reference depth streams.

Other 3D representations used for 3D immersive systems include the image-based visual hull (IBVH) [18] that is used in Coliseum [2] and video fragments [23] used in the blue-c system [7]. IBVH projects the object silhouettes from different cameras and computes the intersection to generate a 3D model. Video fragments are a point-based representation, which exploits spatio-temporal coherence by identifying differential fragments in 2D image space and updating the 3D point representation of the scene.

3. INTRA-STREAM ENCODING

In this section, we describe our scheme for extending traditional video encoding techniques to encode base reference depth streams. We then focus our investigation on the question of the best way to employ motion-compensation given the extra per-pixel depth information that we need to encode.

Multiple depth streams exhibit spatial coherence between streams as well as temporal coherence between frames within a stream. For effective encoding of multiple depth streams, these characteristics should be exploited. In order to utilize spatial coherence between streams, a base reference stream (intra-stream), which other streams (inter-stream) refer to, must be encoded such that it can be decoded independently of any other streams. However, the intra-stream can use temporal coherence between its frames for encoding.

A depth stream is very similar to a video stream in that both streams have three channels of color to encode. They both also have temporal coherence between frames. Therefore video stream encoding techniques should be well suited for adoption in intra-stream encoding. The three color channels of intra-stream are encoded similar to standard video encoding. However depth is encoded as a grayscale image of the inverse depth, since the inverse of depth is linear in perspective space.

In our scheme, each frame is encoded either as an I-Frame, or a P-Frame just like a standard video. However, unlike some video encodings, no B-Frames are used because B-Frames refer to frames in the future, which increase coding latency. Since our target applications are generally real-time interactive tele-immersion systems, we want to avoid that extra latency. All frames are encoded on a block basis and each block is encoded as an I-Block or a P-Block. An I-Block can be decoded independently of other frames so all I-Frame blocks are encoded as I-Blocks. A P-Block is associated with a *motion vector*, with which reference values for the block can be predicted. The *residual* is the difference between the values of the block to be encoded and the predicted values. This residual is then encoded along with the motion vector. A P-Frame uses both I-Blocks and P-Blocks for encoding. An I-Block encoding is used if it is more efficient than a P-Block encoding (i.e., no good reference block can be found, or the block does not have any high frequency information).

For each block of any type, a Discrte Cosine Transform (DCT) is applied to either the direct pixel data in the case of I-Blocks or the residual data in the case of P-Blocks. The resulting coefficients are then quantized. Thus, the quantizer and the motion vector are the two parameters that can be specified for color and depth which most effect the encoding of intra-streams.

3.1 Quantizer

Quantization during encoding affects image quality and compression ratio. Increasing the quantization step size reduces the range of coefficients that need to be represented and eliminates

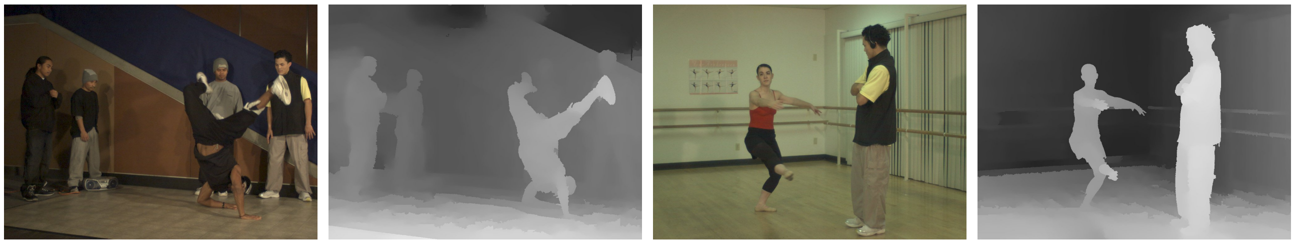


Figure 1: A Frame from Breakdancers and Ballet data

small coefficients by reducing them to zero. In effect, the larger the quantizer, the higher the compression ratio. However, larger quantizers leads to large distortions in the encoded images. The images become blurrier since high frequency content is filtered out. For video streams the three color components are encoded with the same quantizer. However for intra-streams, depth and color should not share the same quantizer since they do not have the same properties. As shown in Fig. 1, depth values are usually smooth over a local region and only show high frequency content at object borders. So a higher quantizer can be used for depth than color to achieve the same PSNR.

Fig. 2 shows the effect of quantization on image quality averaged over 100 frames. Color and depth is given for two separate data sets – Ballet and Breakdancers. The depth quantizer can be set higher than the color quantizer to achieve the same PSNR.

3.2 Motion Vectors

P-Blocks must use extra bits to encode motion vectors compared to I-Blocks. However the residuals for the P-Blocks will be very small if the predicted values match very well – for blocks that match very closely, there maybe little or no residual left to encode. This results in P-Blocks being encoded more efficiently than I-Blocks. Therefore finding a good motion vector is critical for effective encoding of P-Blocks. Video codecs generally use the same motion vector, based on the luminance (Y) plane, for encoding all three color planes with good results. However for intra-streams, each P-Block has two motion vectors that can be associated with it. One is the *Y motion vector* (MV_Y), which is based on the Y plane. The other is the *depth motion vector* (MV_D) based on the depth plane. So for encoding P-Blocks of intra-streams, the following are possible:

- Only use MV_D for encoding depth and color.
- Only use MV_Y for encoding depth and color.
- Use MV_D to encode depth, and MV_Y to encode color.

Using only MV_D gives the optimal encoding for the depth values, at the expense of color value encoding. Using only MV_Y optimizes the encoding of the color values while the depth value encoding is effected. When using a single motion vector for encoding, whether the MV_D or the MV_Y, the total efficiency of intra-stream encoding depends on how well the specified motion vector can predict the other component.

Using both MV_D and MV_Y for encoding will result in the best encoding for both color and depth values. However this incurs the overhead of searching for and encoding an additional motion vector. Searching for a good motion vector is not a trivial task – a large part of video encoding is spent on searching for motion vectors. Therefore the overhead of searching for an extra motion vector cannot be ignored.

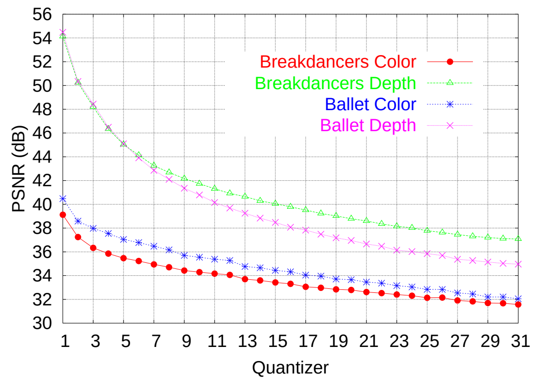


Figure 2: Quantizer effect on image quality

4. RESULTS

Two data sets from [26], Ballet and Breakdancers, were used to test intra-stream encoding (Fig. 1). Both data sets have 8 depth streams that are 100 frames long at 1024x768 resolution and were captured at 15 fps. Each frame has 24 bit color (RGB) and 8 bit depth information for each pixel. The depth was computed using the technique described in [26].

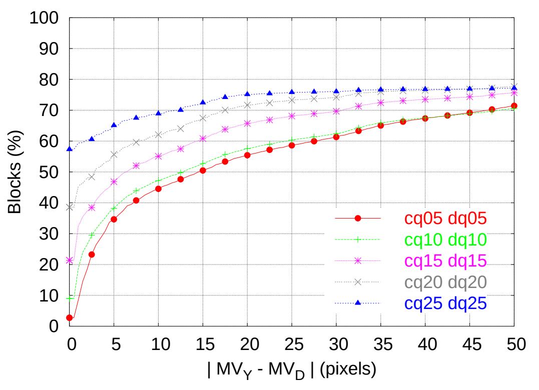
A modified version of a well known video codec, XviD [24], was used to encode the intra-streams. XviD is an open source ISO MPEG-4 [9] compliant video codec. XviD was selected mostly because of source code availability, which can be modified without any restrictions. The modified XviD codec is able to encode frames with four components and the option to specify the motion vector used for encoding P-Blocks.

All frames were encoded as P-Frames with the previous frame encoded as an I-Frame. This was done to eliminate error propagation when a P-Frame is encoded using a P-Frame as a reference. Also the quantizer for all blocks in a frame was kept constant for same components. This insures that the encoding is affected similarly across blocks within the same frame, and a fair comparison can be made between luminance (Y) and depth.

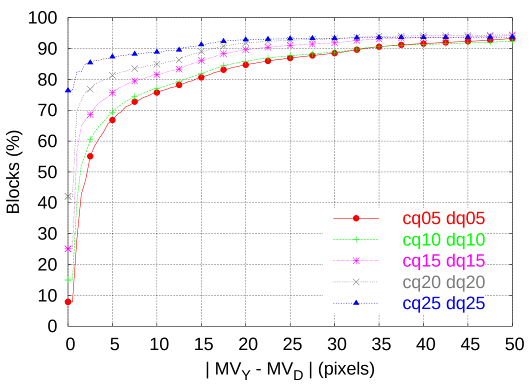
4.1 Motion Vector

When encoding P-Blocks, two motion vectors can be considered. One is the *Y motion vector* (MV_Y) which is based on the luminance plane. The other is the *depth motion vector* (MV_D) based on the depth plane. If only one motion vector is used for encoding, the encoding efficiency will depend on how close the selected motion vector matches the non-selected one.

The *delta motion vector* (ΔMV), defined as $|MV_Y - MV_D|$, estimates how similar the two motion vectors match between corresponding blocks. If both motion vectors are a perfect match, the

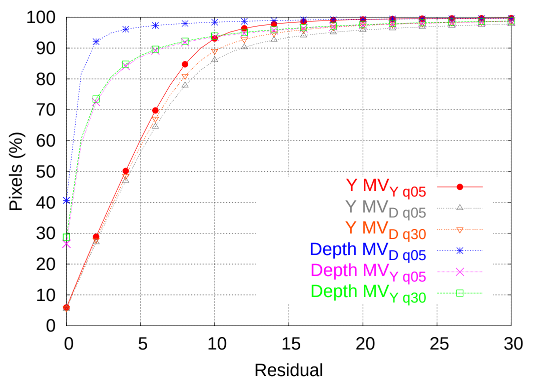


(a) Breakdancers ΔMV

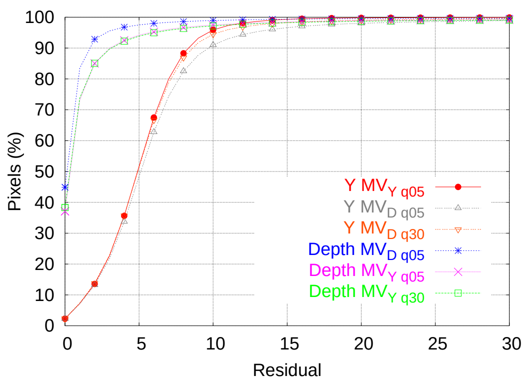


(b) Ballet ΔMV

Figure 3: Motion Vector difference between Y motion vector (MV_Y) and depth motion vector (MV_D) for data sets Ballet and Breakdancers. 'cq' is the color quantizer, and 'dq' the depth quantizer.



(a) Breakdancers



(b) Ballet

Figure 4: Accumulative residual histogram of Breakdancers and Ballet for depth and luminance (Y) at quantizer 5. Luminance and depth predicted reference images were predicted from Y motion vector (MV_Y) and depth motion vector (MV_D) at quantizers 5 and 30.

size of the delta motion vector will be zero. As the vectors deviate, ΔMV will increase.

Fig. 3 shows the accumulative distribution histogram of the ΔMV for different quantizer values averaged over 100 frames. Since there are no motion vectors defined for I-Blocks, any I-Blocks are ignored. This appears in the figure by the accumulative values not reaching 100% but plateauing at a lower point. The figures show that Breakdancers data has more I-Blocks than the Ballet data. This is because the Breakdancers have faster motion between frames, which result in less coherence between frames. It also shows that quantizer has little effect on the number of I-Blocks per frame.

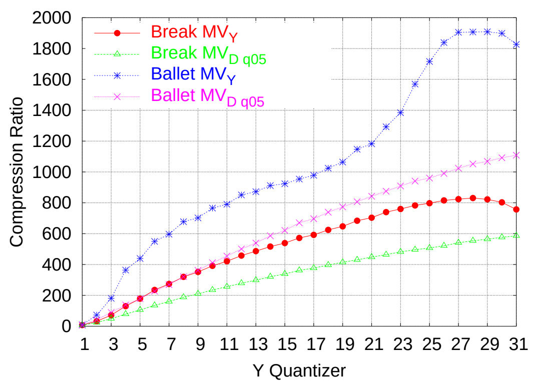
As the quantizer is increased the high frequency content is reduced, resulting in a blurrier image. And as the image becomes blurrier, the feature differences in depth and color decrease. This results in more similar motion vectors.

4.2 Predicted Reference Image

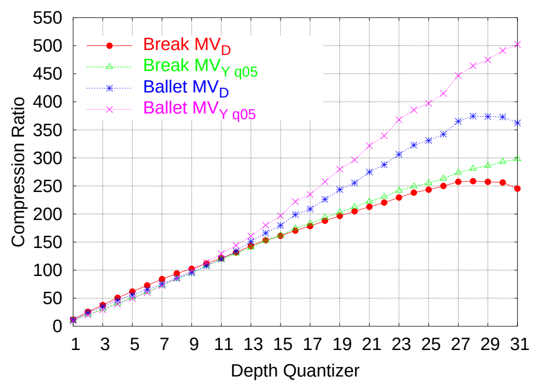
In order to determine the relative merits of our three possible methods for motion compensation, we introduce a metric for measuring the effectiveness of motion compensation called the *accumulative residual histogram*. This histogram represents the amount of information that remains to be encoded after motion compensation.

To calculate this histogram for a particular P-Frame, we first generate a *predicted reference image*. For P-Blocks, the predicted reference image is simply the predicted pixel value derived from motion compensation. For I-Blocks where no motion vector is present, the predicted reference image for the block is set to the I-Block's DC value. We motivate this choice by noting that a block of a P-Frame is encoded as an I-Block for one of following two reasons. One is that the block does not have high frequency content and encodes much better as an I-Block, in which case using the DC component as the reference is a good estimation of the block. The other is where no good motion vector can be found. In this case the DC component, which represent the average intensity of the block, is a good representation of the block for the purposes of calculating the residual information that remains to be coded as non-DC coefficients.

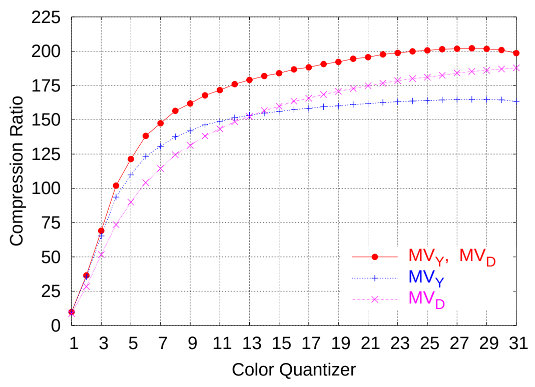
Once we have a predicted reference image, we can derive the *accumulative residual histogram* as the number of the pixels in the image that have a residual equal or less than a particular value. Fig. 4 shows the average accumulative residual histogram of Breakdancers and Ballet for depth and luminance (Y) at quantizer 5. Predicted reference images for Depth and luminance using both the Y motion vector (MV_Y) and the depth motion vector (MV_D) at quantizer 5 and 30 is shown. As expected, for luminance prediction us-



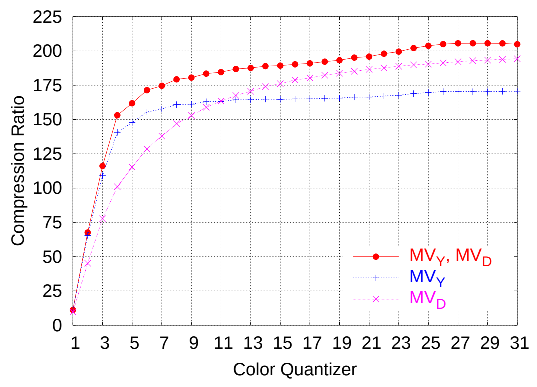
(a) Luminance



(b) Depth



(c) Breakdancer



(d) Ballet

Figure 5: (a) Compression ratio of color using Y motion vector (MV_Y) and depth motion vector at quantizer 5 (MV_{Dq05}) for Breakdancers and Ballet. (b) Compression ratio of depth using Y motion vector at quantizer 5 (MV_{Yq05}) and MV_D for Breakdancers and Ballet. (c) Compression ratio of color and depth at quantizer 5 using MV_Y, MV_D, and MV_Y & MV_D for Breakdancers. (d) Compression ratio of color and depth at quantizer 5 using MV_Y, MV_D, and MV_Y & MV_D for Ballet.

ing MV_Y yields the best result, and for depth prediction using MV_D is the best. Also depth has many more pixels with very small residual to encode than color since depth does not have as much high frequency content to encode as luminance. It is also noticeable that the Ballet data have better prediction overall than the Breakdancers data, which reflects the fact that Breakdancers have faster motion between frames.

Using MV_Y to predict depth performs badly in a situation where depth values within the block differ while color values do not. An example would be a person with his arm in front of him in a long-sleeved shirt. The arm and the chest would have similar color but the depth would be different. So MV_Y may select a reference block where the arm and the chest is both present in order to encode a block where only the chest is present if the color information is still a good match. However this results in bad encoding for depth since it would have to encode the high frequency information resulting from the depth discontinuity between arm and chest.

MV_D may not be a good predictor for luminance on surfaces that have high frequency texture, such as a poster. If the poster is flat and assumed to have consistent depth, a rotation of the poster would not result in depth changes. Therefore MV_D is likely to predict no change in depth, but the texture of the poster would have rotated and would result in a bad reference block for luminance since this would not be considered. It is more common for MV_D to mispredict luminance than MV_Y to mispredict depth. This is because in a real-world environment there are generally more smooth (or depth-coherent) surfaces with high-frequency color texture than depth discontinuities between surfaces with the same color texture.

4.3 Compression Ratio

Using two motion vectors incurs the extra overhead of specifying an extra motion vector for each block. Therefore it will only outperform the single motion vector algorithms if the extra motion vector predictions are much better such that the residuals encode more efficiently (i.e. require fewer bits).

Fig. 5a shows the compression ratio for color at various quantizers using Y motion vector (MV_Y) and depth motion vector at quantizer 5 (MV_{Dq05}). As expected using MV_Y is always better for encoding color.

Fig. 5b shows the compression ratio for depth at various quantizers using depth motion vector (MV_D) and Y motion vector at quantizer 5 (MV_{Yq05}). It is interesting that using MV_D does worse than MV_{Yq05} at high quantizers, but from Fig. 4, it is clear that using MV_D does a better job of predicting reference images. This suggests that the bits needed to encode MV_D are significant compared to the coefficients generated as the depth quantizer increases, and at high depth quantizer values MV_Y does a good job of predicting depth.

Comparison of using a single motion vector (MV_Y, MV_D) and two motion vectors (MV_Y and MV_D) for various color quantizers at depth quantizer 5 is shown in Fig. 5c and d. Using two motion vectors achieves better performance when high quality encoding of depth is required. Also if only one motion vector can be used, due to limits on resources, it is better to use MV_Y. However if the required color quality is low using MV_D results in a better encoding of the stream due to the following reasons. One is that the high frequency information has been removed enough from quantization that MV_D is a decent motion vector to use for color prediction. The other is that more bits of the total are used to encode depth, so it is more efficient to improve depth encoding by using MV_D.

When encoding a depth stream as a reference stream for encoding multiple depth streams (i.e. intra-stream), depth accuracy affects the encoding of non-reference streams (inter-stream). Therefore using both motion vectors for color and depth is a better option. However there is a significant computational difference between searching for one motion vector and searching for two motion vectors.

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented an encoding algorithm for depth streams where spatial coherence cannot be used. The algorithm extends the motion compensation used for video encoding to also encode depth. We also explore different methods of using motion compensation for encoding color and depth.

The experiments show that for depth streams that require depth to be encoded at high quality, such as a reference stream, it is more efficient to use separate motion vectors for encoding. The overhead of an extra motion vector per block is less than the increased residual by using one motion vector. However the cost of calculating an extra motion vector is not cheap. So when only one motion vector can be used, due to limits on resources, the motion vector for color is usually the better choice than the motion vector for depth. The exception would be when encoding high quality depth and low quality color such that the bits for encoding depth is much higher.

We would like to further examine the following to improve encoding:

- It would be interesting to see if depth can be subsampled like chrominance. Since depth is locally smooth, subsampling by a factor of two might yield higher compression rates without much loss in quality.
- When encoding with motion vectors for color and depth, not all blocks improve on encoding. It might be more efficient to use a flag to indicate which motion vectors are present on a block basis.
- When encoding base reference streams for multiple depth streams, the effect of depth quality on spatial coherence estimation should be examined to determine the required depth quality.

6. REFERENCES

[1] 3D Telepresence for Medical Consultation. `http://www.cs.unc.edu/Research/nlm`.

[2] H. H. Baker et al. The Coliseum immersive teleconferencing system. In *ITP 2002*, Juan Les Pins, France, Dec. 2002.

[3] CanestaVision. `http://www.canesta.com`.

[4] J.-X. Chai et al. Plenoptic sampling. In *ACM SIGGRAPH*, pages 307–318, New Orleans, LA, July 2000.

[5] C.-F. Chang, G. Bishop, and A. Lastra. LDI tree: A hierarchical representation for image-based rendering. In *26th ACM SIGGRAPH*, pages 291–298, Los Angeles, CA, USA, Aug. 1999.

[6] C. Fehn et al. An evolutionary and optimised approach on 3D-TV. In *International Broadcast Conference*, pages 357–365, Amsterdam, The Netherlands, Sept. 2002.

[7] M. Gross et al. blue-c: A spatially immersive display and 3D video portal for telepresence. *ACM SIGGRAPH 2003*, 22(3):819–827, July 2003.

[8] ISO/IEC JTC1/SC29/WG11. *Information Technology – Generic Coding of Moving Pictures and Assoicated Audio Information: Video*. ISO/IEC 13818-2, 1992.

[9] ISO/IEC JTC1/SC29/WG11. *Information Technology – Coding of Audio-Visual Objects – Part2: Visual*. ISO/IEC 14496-2, 2004.

[10] ISO/IEC JTC1/SC29/WG11. *Information Technology – Coding of Audio-Visual Objects – Part10: Advanced Video Coding*. ISO/IEC 14496-10, 2005.

[11] ISO/IEC JTC1/SC29/WG11. *Information Technology – Coding of Audio-Visual Objects – Part16: Animation Framework eXtension*. ISO/IEC 14496-16, 2005.

[12] P. Kauff and O. Schreer. An immersive 3D video-conferencing system using shared virtual team user environments. In *4th ACM CVE*, pages 105–112, Bonn, Germany, Sept. 2002.

[13] N. Kelshikar et al. Real-time terascale implementation of tele-immersion. In *the Terascale Performance Analysis Workshop*, Melbourne, Australia, June 2003.

[14] S.-U. Kum and K. Mayer-Patel. Real-time multidepth stream compression. *ACM TOMCCAP*, 1(2):128–150, May 2005.

[15] S.-U. Kum, K. Mayer-Patel, and H. Fuchs. Real-time compression for dynamic 3D environments. In *the 11th ACM Multimedia*, pages 185–194, Berkeley, CA, USA, Nov. 2003.

[16] M. Levoy and P. Hanrahan. Light field rendering. In *23rd ACM SIGGRAPH*, pages 31–42, New Orleans, LA, USA, Aug. 1996.

[17] M. Magnor and B. Girod. Data compression in image-based rendering. *IEEE Trans. on Circuits and Systems for Video Technology*, 10(3):338–343, Apr. 2000.

[18] W. Matusik et al. Image-based visual hulls. In *ACM SIGGRAPH*, pages 369–374, New Orleans, LA, July 2000.

[19] W. Matusik and H. Pfister. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *ACM SIGGRAPH 2004*, 23(3):814–824, 2004.

[20] J. Shade et al. Layered depth images. In *ACM SIGGRAPH*, pages 231–242, Orlando, FL, July 1998.

[21] A. Smolic and P. Kauff. Interactive 3-D video representation and coding technologies. *Proc. of the IEEE*, 93(1):98–110, Jan. 2005.

[22] H. Towles et al. 3D tele-immersion over internet2. In *ITP 2002*, Juan Les Pins, France, Dec. 2002.

[23] S. Würmlin, E. Lamboray, and M. Gross. 3D video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics*, 28(1):3–14, Feb. 2004.

[24] XviD. `http://www.xvid.org`.

[25] ZCam Depth Camera. `http://www.3dvsystems.com`.

[26] C. L. Zitnick et al. High-quality video view interpolation using a layered representation. *ACM SIGGRAPH 2004*, 23(3):600–608, 2004.