Source coding for transmission of reconstructed dynamic geometry: a rate-distortion-complexity analysis of different approaches

Rufael N. Mekuria*[a], Pablo Cesar[a], Dick C.A. Bulterman[bc]
[a]Centrum Wiskunde Informatica, 123 Science Park, 1098XG Amsterdam, NL;
[b]Dept. of Computer Science, VU University, De Boelelaan 1081, 1081 HV, Amsterdam, NL
[c]FX Palo Alto Laboratory Inc., 3174 Porter Dr., 94304 Palo Alto, CA, United States

ABSTRACT

Live 3D reconstruction of a human as a 3D mesh with commodity electronics is becoming a reality. Immersive applications (i.e. cloud gaming, tele-presence) benefit from effective transmission of such content over a bandwidth limited link. In this paper we outline different approaches for compressing live reconstructed mesh geometry based on distributing mesh reconstruction functions between sender and receiver. We evaluate rate-performance-complexity of different configurations. First, we investigate 3D mesh compression methods (i.e. dynamic/static) from MPEG-4. Second, we evaluate the option of using octree based point cloud compression and receiver side surface reconstruction.

Keywords: Surface Compression, 3D Reconstruction, 3D Mesh Compression, MPEG-4

1. INTRODUCTION

The rise of 3D Media and its large scale consumer adoption has led to large challenges in transmission of this content over networks. Key problems with the transmission of 3D Media include increased bandwidth usage [1], a myriad of different formats and representations [2] and a lack in understanding of the overall end-to-end user experience / achieved quality [1]. Based on previous work as presented in [3], we distinguish two broad categories of 3D Media representations. Firstly, image based video representations that use additional images to support additional views/3D functionality. For example, stereoscopic video uses two views, one targeting the left and one targeting the right eye. Compression of these representations is currently supported in state of the art video codecs such as AVC/H.264 in specifically defined extensions such as for multi-view [4]. Secondly, we consider geometry based representations, where full 3d scene coordinates are available. A full 3D triangular mesh model is an example of such a format. Because the triangles are defined in a 3d space the 3D triangular mesh model can be rendered from any arbitrary angle. These formats traditionally occur in video games, virtual environments, movie production, and in industrial applications such as computer aided design (CAD). These objects are usually authored by specialists using specific 3D toolsets and creative skills. However, with the rise of depth/image acquisition devices and 3d reconstruction techniques based on computer vision, it is also becoming possible to acquire geometry based representations directly representing natural objects. In this case, compression and transmission of these 3d representations becomes highly critical to enable applications in the area of 3d based tele-presence, conferencing; gaming or 3d tele-immersion. In Table 1 we outline some of the different representations in the two distinct categories, the image based representation such as multi-view plus depth or stereo and geometry based representations such as meshes (3D vertices combined with connectivity), point clouds (set of 3d vertices without connectivity) or 3d animations (fixed connectivity with time-varying 3d vertices).

Table 1.Image Based and Geometry Based 3D Formats

IMAGE BASED 3D FORMATS

Image Based	Geometry Based
Stereo Video	3D Mesh
Color plus Depth	3D Point Cloud
Multi-view video	3D Time Varying Mesh
Multi-view plus depth video	3D Animation

In Figure 1 we show a screenshot of our prototype based on 3d geometry based conferencing on the receiver side. The natural person next to the chair was reconstructed as a 3d mesh, and subsequently compressed and transmitted. The 3d data was received and rendered jointly into the synthetic room with the synthetic avatar. In this case the transmission and compression methods used where the ones described in our previous work [5] with an end-end delay (< 300 ms) and frame rates over 8 fps. We showed in this work that the design and implementation of a prototype for reconstructed geometry based 3d conferencing results in different challenges and requirements, compared to streaming pre-authored, synthetic 3D content.

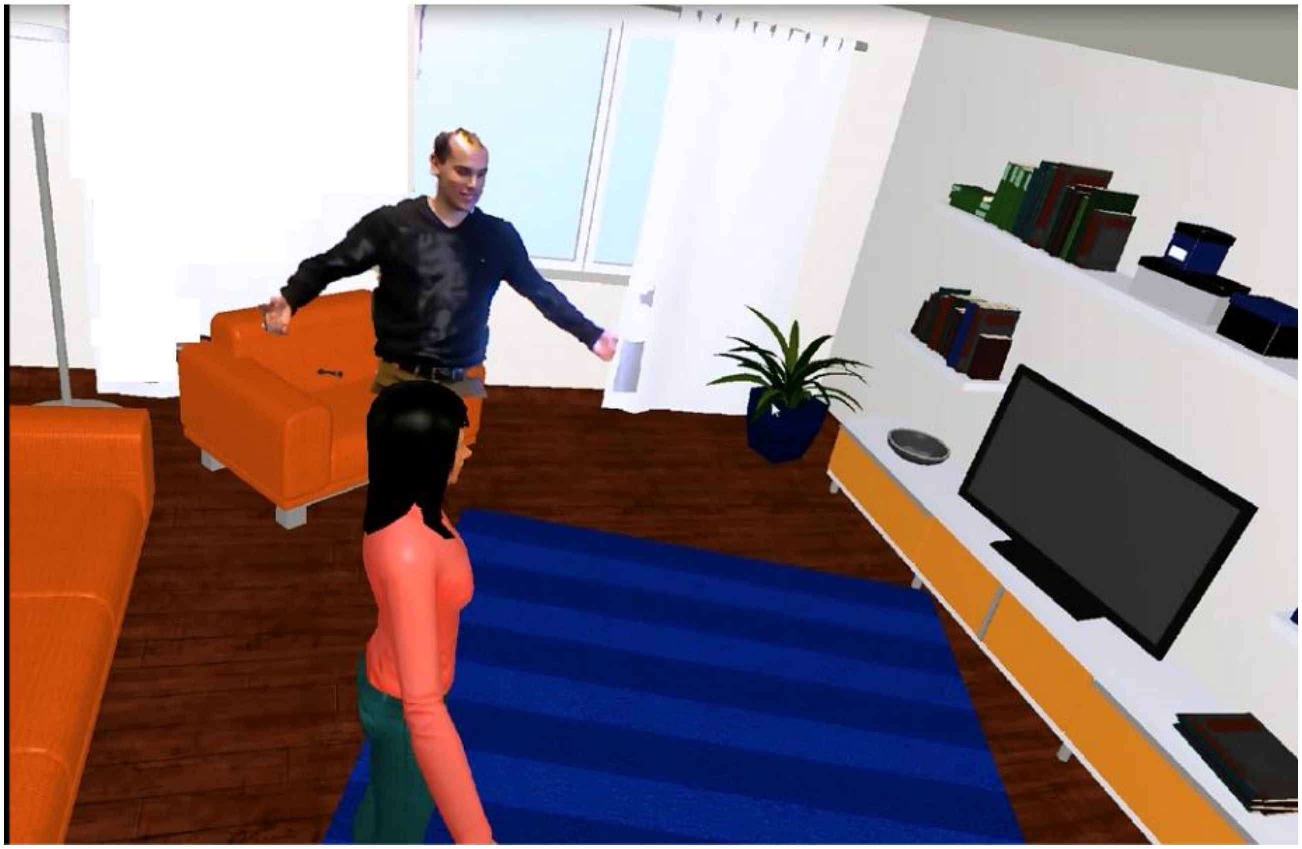

Figure 1 Example of a received 3d geometry based representation rendered jointly in a 3D world

The acquisition of natural objects in a geometry based format via 3d reconstruction makes us reconsider the compression and transmission problem for reconstructed geometry/surfaces. We re-evaluate some of the state of the art tools, keeping the following three research questions in mind:

1. What is the achieved compression ratio and quality (i.e. rate distortion) with different tools?
2. What is the result on the computational complexity on both the sender and receiver sides?
3. What are restrictions and advantages on the 3D reconstruction methods?

The rest of the paper is structured as follows, in section 3 we present an overview of some of the state of the art methods for 3D reconstruction and the resulting datasets that we will use for evaluation. In section 4 we present the state of art compression tools for geometry based representations such as 3D Mesh, animation and point cloud. In this section we also provide comparisons of these tools based on state of the art reconstructed 3d mesh data. In section 5 we conclude and give an outlook on future work. In the next section we present some of the relevant related work in this area.

2. RELATED WORK

Like compression of color video, compression of static 3d geometric objects and animation is an important area of research. It has been important enough that specific ISO standards have been developed in the MPEG working group such as TFAN [6] and SVA [7] for static mesh objects and AFX FAMC [8] for animations representing meshes with a fixed connectivity and time varying geometry. These technologies are currently part of the MPEG-4 standard, and have been introduced fairly recently, superseding previous mesh codecs in MPEG-4, that did not address specific requirements such as fast decoding and an efficient handling of non-manifolds mesh models. The TFAN and SVA codecs are both connectivity driven mesh codecs, i.e. they code connectivity first and use the connectivity to efficiently compress the geometric data via inter-vertex prediction between connected vertices. These codecs also include Context Adaptive Binary Arithmetic Coding (CABAC) of the resulting residuals similar as in the MPEG-4 AVC/H.264 standard resulting in good compression gain. These technologies provide efficient compression for offline authored 3D geometric content for download and interactive viewing. They can also be used for compression of geometry that is reconstructed on the fly representing natural objects. We have integrated these codecs in our 3D Tele-immersive framework with on the fly 3D reconstruction, transmission and rendering as shown in Figure 1. However we achieved low frame-rates with the current implementations of TFAN and SVA and we developed an alternative codec with a faster encoding time instead. This specific connectivity driven codec achieved a compression ratio close to TFAN (15% less), at much faster encoding times resulting in 3 times higher frame rates in the overall system [5].

Compression of reconstructed geometry that represents a human in conversational, interactive or gaming scenarios is fairly recent. Due to its specific nature, various alternative methods compared to traditional mesh compression have been proposed to better address this case.

Nguyen et al. [9] propose a codec where instead of as a triangular mesh, the human body is reconstructed as a quad-mesh. They utilize a specific wavelet transform (Bi-orthogonal Graph Wavelet Filter Banks) on a bipartite graph (i.e. the fixed quad mesh) to both achieve a multi-resolution representation and a specific pattern in the subdivision that is exploited via context adaptive entropy coding. They compare the method to multi-view coding and receiver side reconstruction, which is heavily outperformed.

Chen et al. [10] propose a specific method, optimized for the 3dti type of scenario with low delay, that introduces "boosted frames", that are fully predicted inter-frames that increase the frame rate. A key idea here is that based on user experience and activity a specific scheme of compression is deployed with boosted frames. While this method is only compared to standard entropy coding via the zlib library and introduces some artifacts in the boosted frames, the idea is quite novel and might be adopted in a better way in later work.

Domanouglou et al, in [11] also tried to exploit the activity aware aspects and tried to exploit motion periodicity in common sports activities (like jogging or skiing) to achieve a compression advantage.

While the results in this work where not yet very good compared to the state of art, further tuning may prove that motion periodicity can indeed be exploited for time varying geometry compression.

Another work in [12] proposes a method for compression of time varying point clouds, which instead of the full mesh, code only the vertex positions and colors . This allows changing number of vertices per frame, which is resulting from some of the 3D reconstruction systems. This method is based on a hierarchical octree representation upto depth d, beyond that depth the vertices and colors remaining in the leaf voxels of the octree are optionally coded with a range coder or decimated. This method also introduces a way to exploit temporal redundancy by applying an XoR operation on the binary serialized octree representation between subsequent frames, that are highly similar for static scene objects.

In this paper we review two important state of the art 3D reconstruction methods and the related compression tools. We evaluate the tools on datasets created by these two reconstruction systems that are quite different in nature and pose very different challenges. We mainly compare tools by changing the distribution of reconstruction operations between sender and receiver resulting in different compressed representations. We compare different compression technologies on each of these representations to provide a broad comparative study.

3. 3D RECONSTRUCTION

In this section we present some of the methods available for 3d reconstruction where a 3d mesh is obtained directly from multiple color + depth images, acquired by sensors like Microsoft's Kinect. We categorized two directions in 3d reconstruction, image based methods which we will handle in the next section and shape-space based which we will handle in section 3.2.

3.1 Image Based Methods

We start with an example of an image based method where the mesh is directly obtained and calculated from multiple depth images (without prior shape information, or shape consistency between frames). In this case existing methods for static 3d reconstruction from multiple range scans are implemented efficiently and optimized for use with consumer grade depth camera and moving participants. This gives many extra problems regarding calibration, interference and real-time implementation. We selected the work in [13] as an example; this is a system that we have experience working with that gives good results in practice with Microsoft Kinect. The method includes the following steps shown in Figure 2. First, the Kinects are to be calibrated, both the intrinsic parameters (that is the calibration between the infra-red and the color camera) and external parameters (i.e. between different Kinects done via a checkerboard pattern and pairwise matching) need to be found via calibration. The input Kinect streams are filtered by a 2D smoothing filter, removing noise due to various reasons including infra-red interference between the Kinects. Via the intrinsic and extrinsic parameters the depth images can be merged into a point cloud with unified coordinates. On this point clouds, separate meshes are calculated via a triangulation algorithm (Delaunay triangulation). This is one of the critical computation steps when the reconstruction needs to be done in real-time. In the next stage, the work in [13] deploys ICP-based refinement of the different calculated mesh surfaces to make the shapes aligned with each other. After that, redundant (i.e. duplicate overlapping) triangles are removed and the different meshes are zippered together (zippering method as known in the literature from [19] is a way to stitch meshes obtained from multiple range images together). Alternatively methods in the same context include Poisson surface reconstruction and smooth signed distance surface reconstruction [22]. The advantage of such image based approach is that based on the input data only, a surface is reconstructed. This means that any object can be reconstructed on the fly as a surface and no prior shape information is needed.

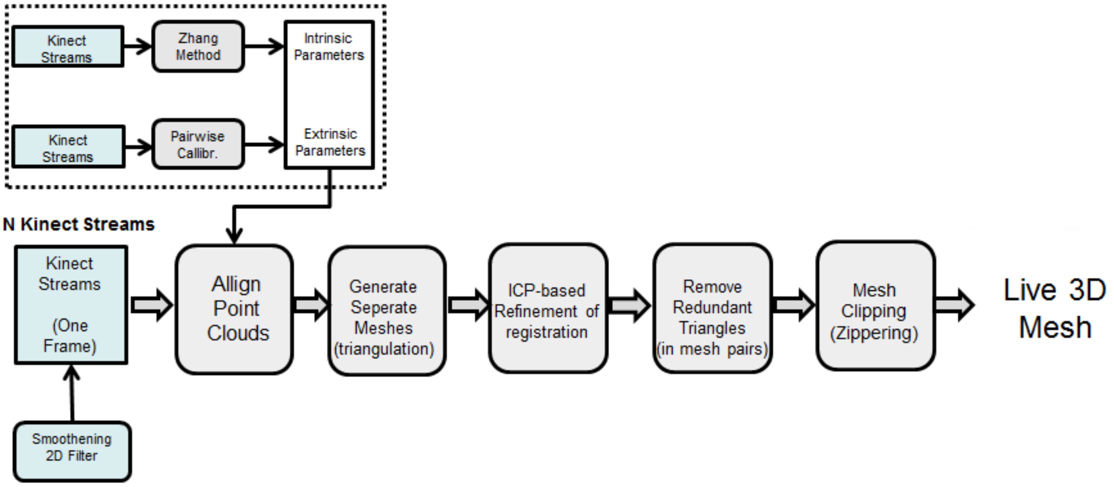
Kinect Streams
Zhang Method
Intrinsic Parameters
Kinect Streams
Pairwise Callibr.
Extrinsic Parameters
N Kinect Streams
Kinect Streams (One Frame)
Allign Point Clouds
Generate Seperate Meshes (triangulation)
ICP-based Refinement of registration
Remove Redundant Triangles (in mesh pairs)
Mesh Clipping (Zippering)
Live 3D Mesh
Smoothening 2D Filter

Figure 2 Pipeline of real-time reconstruction via image based method based on [13]

We refer to the images of the reconstructed 3d data presented in [13]. From our experience rendering this 3d data interactively with an advanced rendering framework, we found the quality to be quite good despite some obvious imperfections. Datasets of the meshes resulting from this system are publicly available: http://vcl.iti.gr/reconstructions/..

3.2 Shape-Space based Methods

Another way to reconstruct 3d mesh data, or actually 3d mesh animations, is to fit a previously defined shape (possibly scanned with a much higher quality scanning device such as a laser scanner) to incoming color plus depth streams. A recent example of such a method was presented in [14]. In Figure 3 we show a block diagram to describe the basic functionalities of this system. Again, both internal and external calibration of multiple color + depth cameras (in this case Microsoft Kinect) is needed. In this system, first a skeleton is calculated based on aligned segmented depth images. This skeleton is then used to get a rough, coarse first surface by deforming the original offline computed 3d mesh model and the previous frame. In the next step based on the segmented images from the Kinect streams, this surface is refined, lastly the skeleton is also refined based on this refined surface, resulting in both a high quality mesh surface and skeleton. An additional advantage of this method is the topology/connectivity remains fixed, as each reconstruction step only updates the vertex positions. This eases later compression tasks, as only the changing geometry over time needs to be encoded. On the other hand, it restricts the input to some extend due to the topological restriction.

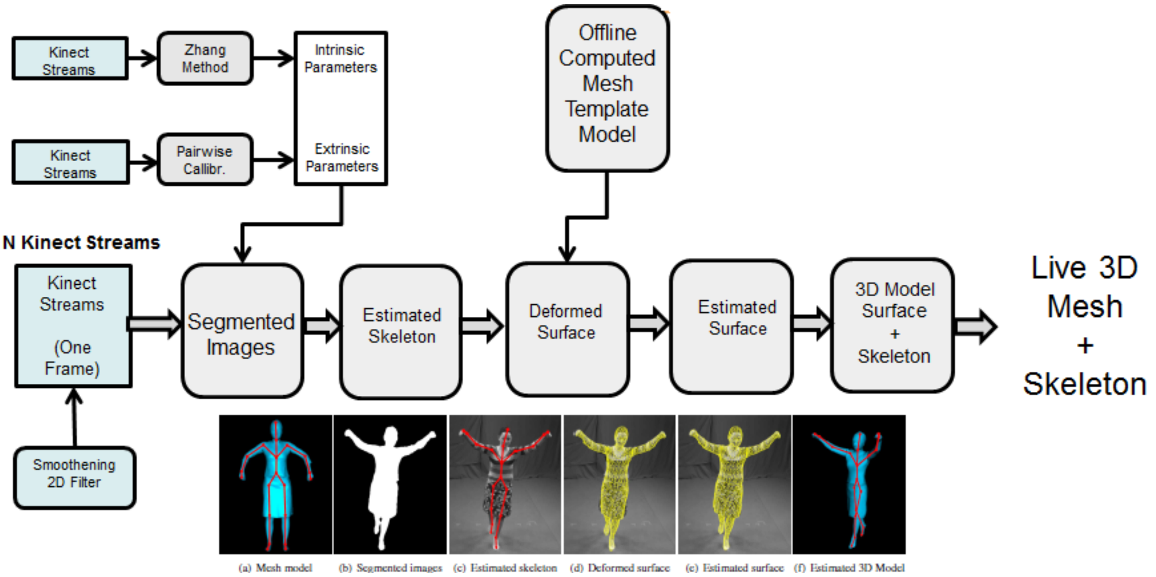

Figure 3 Example pipeline of shape based 3d reconstruction in [14]

4. 3D COMPRESSSION

4.1 Compression Position

In the two presented reconstruction methods in section 3, multiple color plus depth streams are aligned, (resulting in a point cloud or aligned segmented images), and the surface is reconstructed based on this as a mesh. As the steps from the reconstruction process can be distributed between receiver and sender side, this allows the option of different compressed representations for transmission. One could send the multiple color plus depth streams (i.e. simulcasting), one could send the point cloud and reconstruct at the receiver side or one could send the compressed mesh. An option with very light receivers could even be to send a snapshot of the rendered mesh as a video or image (i.e. pre-rendering). It would be logical to choose the configuration that gives the best compression gains and lowest byte size. Nevertheless, the choice also depends on the computational load introduced on both the client and receiver sides and the constraints of the reconstruction system. We outline the various alternatives in Table 2. In this paper we will compare the options of simulcast (section 4.5.), point cloud compression (4.4.) animation compression (4.3) and static mesh compression (4.2.) on both types of reconstruction methods.

Table 2. Distribution of 3D Reconstruction between sender and receiver and resulting format

Distribution of 3D Reconstruction between sender and receiver and resulting format

Sender Side Operations	Receiver Side Operations	Compresssed Representation
RGB-D capture only	Sync., point cloud, mesh reconstruction	Multiple Video + Depth
RGB-D capture only + synchronization	point cloud, mesh reconstruction	Multiple Video + Depth
Point Cloud Reconstruction	Mesh Reconstruction	Point Cloud
Mesh Reconstruction	Rendering + Composition	3D Mesh/Animated Mesh
Mesh Rendering+ Composition	2D Rendering	Video/Image

4.2 Static Mesh Compression

When each mesh is reconstructed independently as in section 3.1., and the mesh needs to be transmitted to multiple sites, static mesh compression is a reasonable option as it is harder to exploit temporal correlation when the mesh connectivity is not fixed. In this case, the full 3D mesh reconstruction is done at the sender side, and the compressed representation for transmission is a mesh. In this case, the static mesh compression defined in MPEG Scalable Complexity 3D Mesh

Coding (SC3DMC) is of particular interest as it includes algorithm with manageable/customizable complexity that can decode in real-time. SC3DMC includes both the triangle fan (TFAN) codec and the Shared Vertex Analysis Codec (SVA).

The TFAN codec provides state of art compression rates for mono-resolution connectivity. As the authors of this codec indicated, their method can be seen as an extension of the connectivity coder by Touma and Gotsman [14], which is often considered to give the best compression rates in the literature. The important extensions of TFAN are that instead of only coding based on valence, different configurations are coded, making it applicable to non-oriented non-manifold surfaces that can result from 3D reconstruction and mesh simplification operations.

The TFAN works by first decomposing the mesh into triangle fans, which are an ordered set of *k* triangles and k+2 vertices. Each of the ordered triangles inside a triangle fan are neighbors, and the triangles have the same orientation and share a common vertex (the central vertex). The decomposition of the mesh in triangle fans results in overlapping triangles between fans. In Fig 4 we show an example triangle fan as it is detected in [6], in this case the blue triangles in the TFAN are “non-visited”, while the purple ones are “visited” as they occurred in a previously coded fan The key to TFAN is the detection definition of 9 different configurations and their relative frequency of occurrence. Based on this, coding bits can be efficiently allocated. These configurations are defined by the degree of the triangle fan (number of triangles), the set of visited and non-visited vertices and a relative index vector (each triangle fan uses a relative vector for indexing resulting in a lower number of bits for indexing). These operations result in a very compact representation of the connectivity. As a side effect this representation introduces changes to the order of the triangles (which does not matter for rendering or further processing the mesh, but might matter in some very specific situations). Subsequently the connectivity information is used to compress the vertex coordinates and attributes via inter-vertex prediction based on DPCM or parallelogram prediction (the latter uses three vertices to predict a fourth vertex). The resulting prediction residuals are compressed via context adaptive arithmetic coding (CABAC) or other entropy coding schemes.

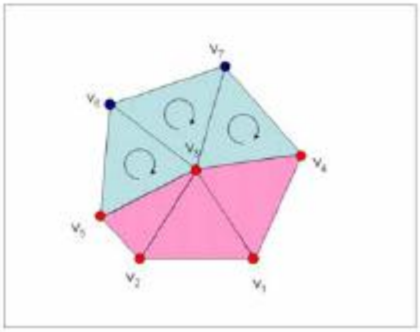

Figure 4 An example of a common tfan configuration in a (manifold) mesh [6]

Shared vertex Analysis, described in [7] is a slightly simpler and more direct approach to code the connectivity. It works by analyzing the face list of vertex indices, and tries to detect shared vertices between consecutive faces. It also allows reordering of the vertices inside the triangle, which may change the orientation of the triangle but is harmless in general. In this codec, again the connectivity is used to apply inter-vertex prediction via like in TFAN [6] with residuals quantization via CABAC or other entropy coding schemes.

An alternative approach was our codec that we use for 3D communication in [5], its schematics illustrated in Figure 5, in this codec we also compress the connectivity first. We do this by computing differences by subsequent face indices in the face list directly and looking for repetitive patterns. We code the resulting data vector via the zlib library entropy encoder. Subsequently, instead of applying inter-vertex prediction on the quantized geometry and using arithmetic coding, we directly quantize the differential floating point values between connected vertices with an optimized local non-linear layered quantization vector. This local quantization vector uses four bits and spans a very small range that is common between connected vertices in a dense mesh. Larger values are encoded additionally defined linear quantizers B, C and D that use 8, 16 or 32 bits respectively depending on the differential value. The color and normal attributes are compressed in a comparable way, but with slightly modified quantization vectors. This allows us to code over 95% of the differentials with 4 bits and it results in lower encoding complexity, which yields higher frame rates in end-to-end communication as described in [5]. Key advantage of this approach is that entropy coding is skipped, resulting in less computational overhead and faster encoding times.

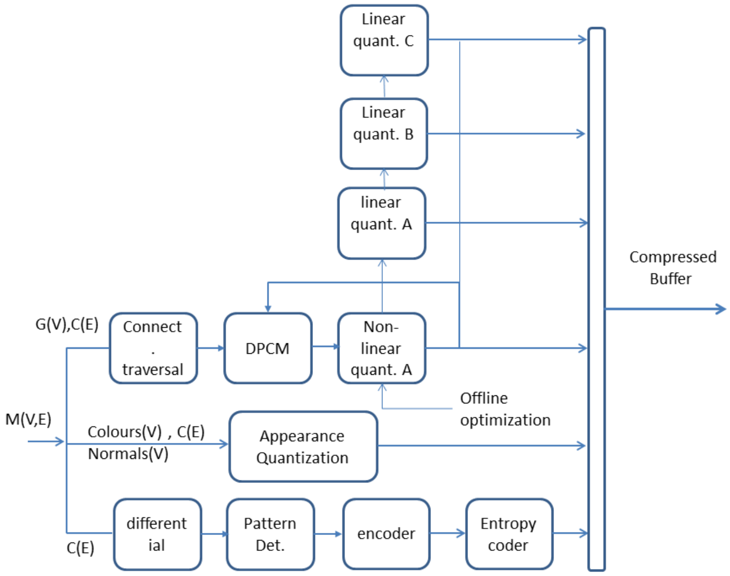

Figure 5 Architecture of CWI Codec for fast static mesh frame compression

In Figure 6 we show the performance results of encoding reconstructed mesh frames with the system described in [13]. The TFAN method gives the best compression results while the cwi codec was the fastest. The resulting meshes have been measured of approximately equal quality with symmetric L2 (rms) and L infinite (Haussdorf distance) metrics. The Quality resulting from the CWI codec was comparable to tfan-10-6-6 and SVA 10-6-6. Figure 7 gives a qualitative comparison of the original and decoded meshes by showing them rendered from the front side.

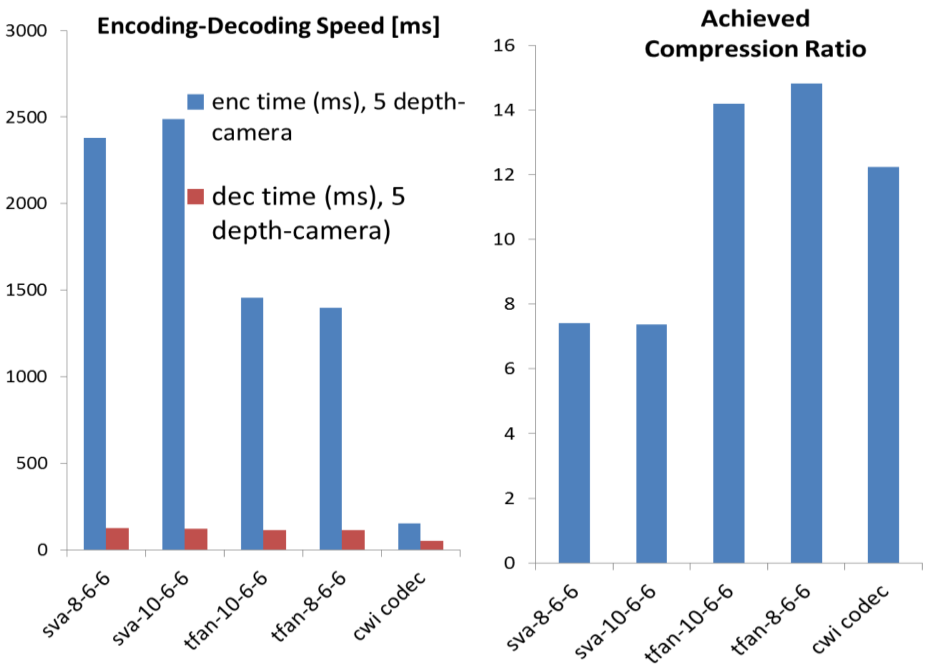

Figure 6. Static mesh compression on datasets of reconstructed surfaces based on [3] and [16]

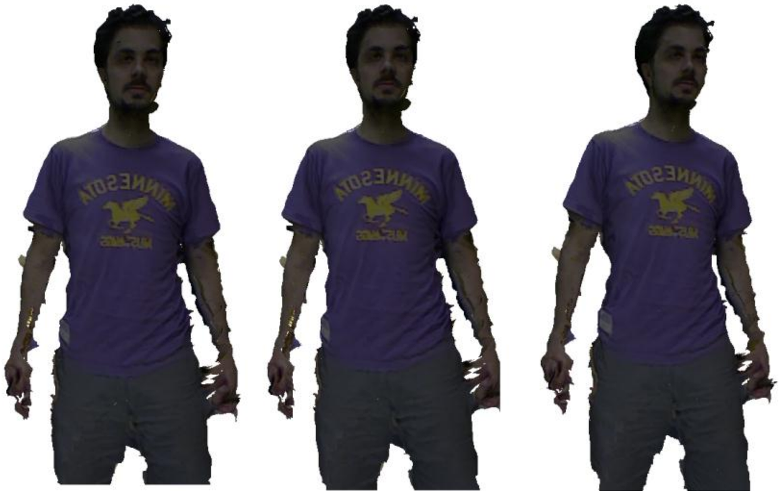

Figure 7 Compression results (original left, MPEG-4 TFAN in the middle, CWI-Codec, right)

4.3 Animation Compression

FAMC [8] is a standardized technology for compression of mesh based animations, that are mesh sequences based on key-frames where only the vertex positions change. The standard is the latest addition to MPEG-4 part 16 [7], that was developed to compress synthetic animation. It is also currently the state of art codec for compression of mesh animations in terms of performance [8].

The FAMC Coder codes animations in 4 steps (for more details see [8]). First skinning based motion compensation is applied, which is a form of motion compensation that segments the mesh in clusters that can be tracked for their motion (which is accurately described by an affine transform). The motion of these clusters is subsequently used to predict vertex positions. Subsequently the resulting residual values (difference between predicted and original values) are transform coded via either a discrete cosine transform (DCT) or a bi-orthogonal wavelet transform. On the subsequent transform coefficients, layered prediction is performed to exploit additional dependencies between coefficients (that might result from the wavelet/DCT transform). The resulting values are coded via Context Adaptive Binary Arithmetic Coding.

We have experimented with this codec in combination with a set of mesh animations resulting from the reconstruction process described in section 3.2 from [14]. In Figure 8 we show meshes from this animation dataset. The 3d reconstruction process introduces time consistency between topology/connectivity over frames, this means mesh animation compression instead of static compression.

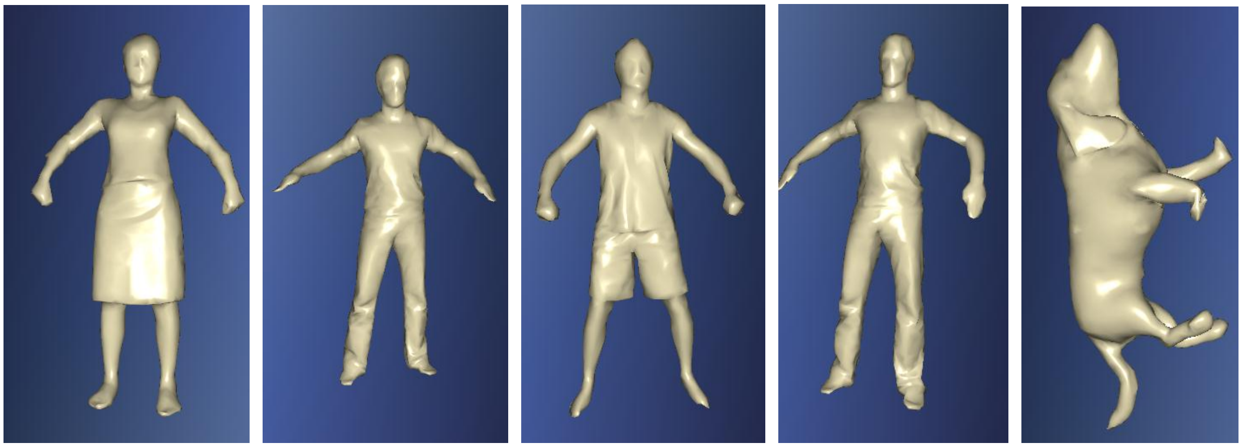

Figure 8 Sample reconstructed mesh animation data with the method of [14] from left to right skirt mesh, hand stand mesh, dance mesh, wheel mesh and dog mesh

In Figure 9 we show the uncompressed size of the vertices only (without the connectivity), per frame. The uncompressed size does not vary per frame as the number of vertices and size remains fixed over time. The meshes contain 2501 vertices (except dance mesh which contains 1501 vertices), which result in 30 Kb per frame uncompressed.

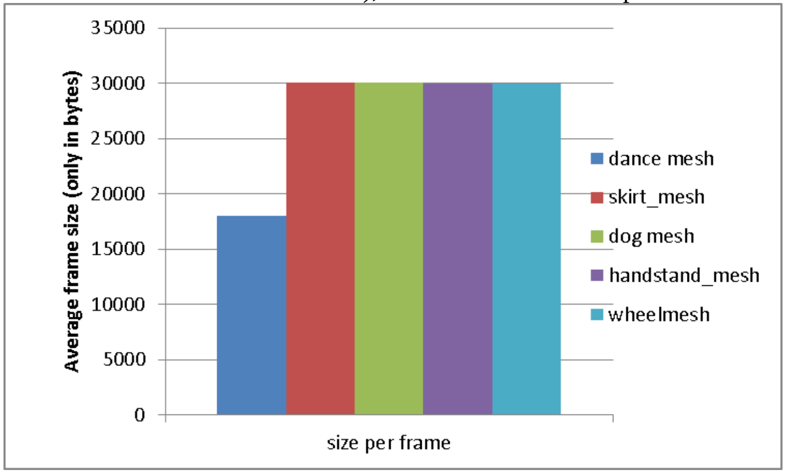

Figure 9 Original size of time consistent reconstructed mesh frames (vertex positions only) based on data from [14]

In Figure 10 we show the results of compression of the animation with Famc. We have chosen 8 bits for quantization of the coordinates and 12 bits for compression of the affine transform used by the skinning-based motion compensation. We chose single layer prediction between the transform coefficients and set the number of B frames between I frames to 3. We have done two experiments, one with larger (N=45) segments and 30fps reconstruction and one with smaller (N=15) segments and 10 fps reconstruction. Figure 10 shows that the compression gains at 30 fps reconstruction and larger segments (of 45 frames) are better, and a decrease up to 13 times compared to the original vertex frame size has been achieved. For the live and online scenario with 3d reconstruction, it is expected that a lower frame rate and smaller segment sizes will be realized. This would result in frame rates around 10fps and segments of maximum of 15 frames. In this second case we achieved a compression ratio upto 10 compared to the original vertex frame size as shown in Figure 10.

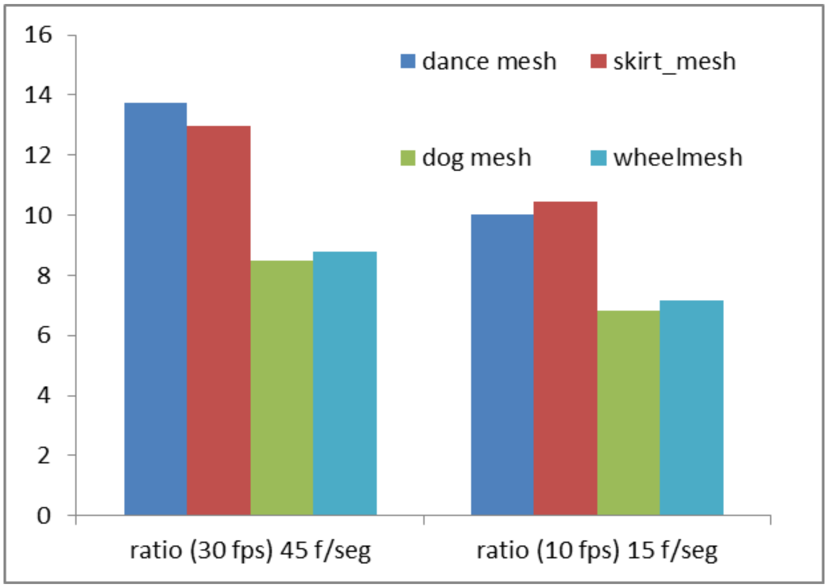

Figure 10 Compression ratio size with MPEG-4 FAMC and DCT transform coding based on data from [14]

This analysis does not include the compression of the first key frame of the animation that also include the connectivity, this can be done via any static mesh codec or general purpose compression tool in the beginning of each animation segment. In Figure 11 we show some of the decoded frames in the animation which shows that compression artifacts are hardly visible. We have also not included the texture/color data that was not included with the data in [14].

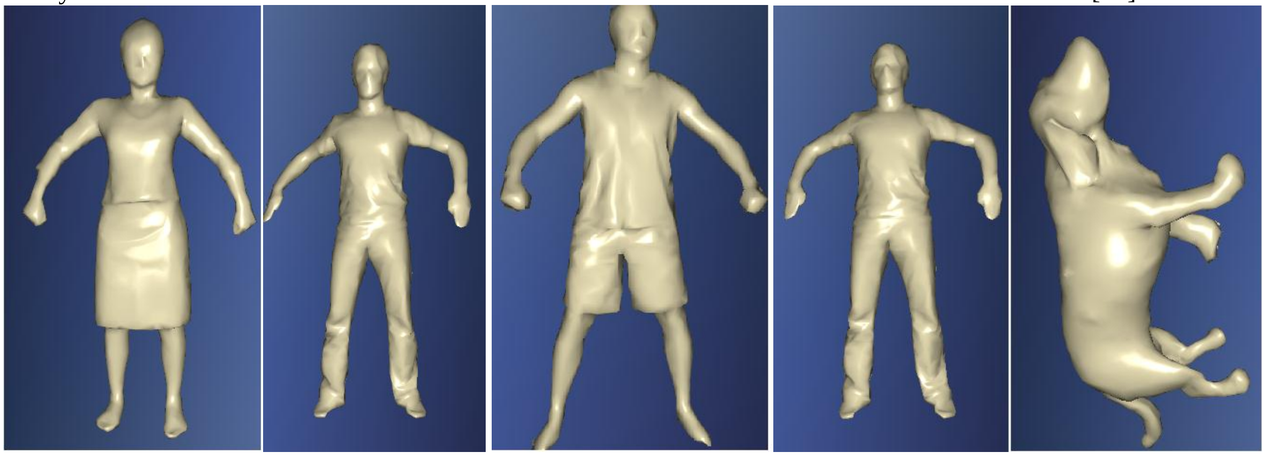

Figure 11 Decoded Models from the dataset in [14] compressed with FAMC

4.4 Time Varying Point Cloud Compression

Instead of sending the full mesh data, one could send the point cloud data and reconstruct the surface at the receiver and render it compositely in the scene as in Figure 1. In this case point cloud compression and transmission become relevant. An efficient compression scheme for time varying point clouds was recently proposed in [12] and is publicly available in the open source library PCL [18]. Point clouds resulting from laser scanners and 3D reconstruction methods with Kinects like [16] result in a large number of vertices. For example in the datasets from [16], 200 K – 300K vertices per frame are common (resulting in frames in the range of 15 MB). Specifically for such dense cases, octree composition is useful. The octree composition of the mesh is obtained by first defining the bounding box in 3D space containing all vertices, and then sub-dividing the bounding box into 8 sub-cubes repetitively and coding either an empty or non-empty cube. By continuously subdividing in top-down fashion, either the entire cloud is coded or a predefined target cube size is reached and the iteration is stopped. Figure 12 illustrates this composition method and shows how each octree subdivision can be coded in a single byte where 1 represents a non-empty sub-cube and 0 an empty sub-cube. In case a sub-cube is empty no further subdivisions are necessary. The resulting octree representation is progressive, i.e. a limited number of subdivisions (bytes) give a coarse representation of the cloud. Sending only a limited number of subdivisions (bytes) of the octree is therefore a lossy form of compression. The method in [12] uses entropy coding to code the resulting serialized point cloud (i.e. the bytes representing the subdivision). In Figure 13 we show the resulting rate-distortion vs byte size on datasets described in section 3.1. We compress only vertex coordinates and attributes with the codec from [12]. We vary the rate via different final voxel (cube) sizes. The larger the final voxel size, the coarser the representation and the lower the bit-rate and the higher the distortion. To do the comparison we re-mesh the point cloud at the receiving end with methods available in the PCL and measure the distortion based on symmetric rms and haussdorf distance (similar as in section 4.2). In this case, contrary to the previous sections, we get a large but controlled distortion. The results show that significantly smaller byte sizes are achieved with this form of compression compared to coding with a standard static mesh codec like TFAN at still acceptable quality (but with less detail). Drawbacks are the higher distortion and the computational complexity of re-meshing the surface at the receiver, especially with finer final voxel sizes. Overall, for low bit rate compression that reduces the quality of a dense surface, octree compression can be quite useful.

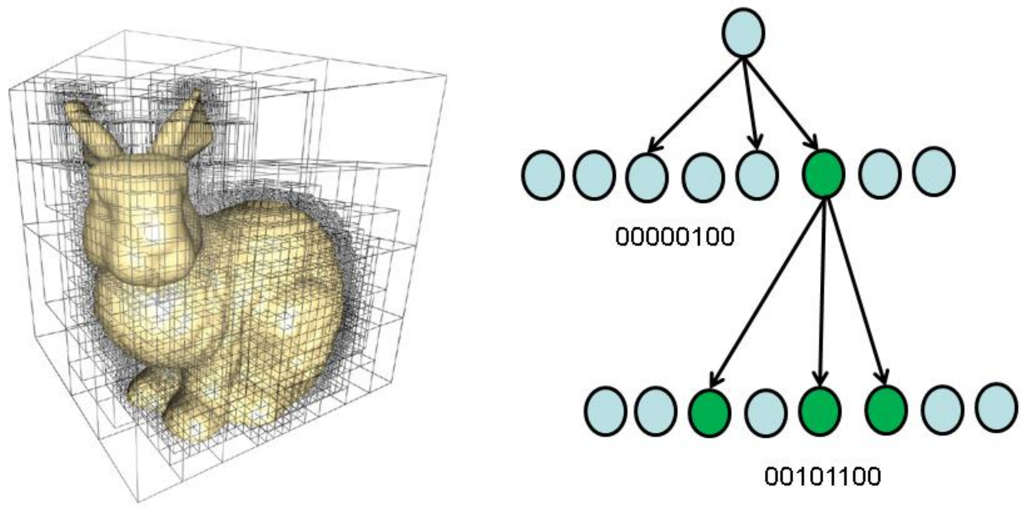

Figure 12 Octree representation: the mesh is repeatedly subdivided in 8 sub-cubes and each subdivision is a byte

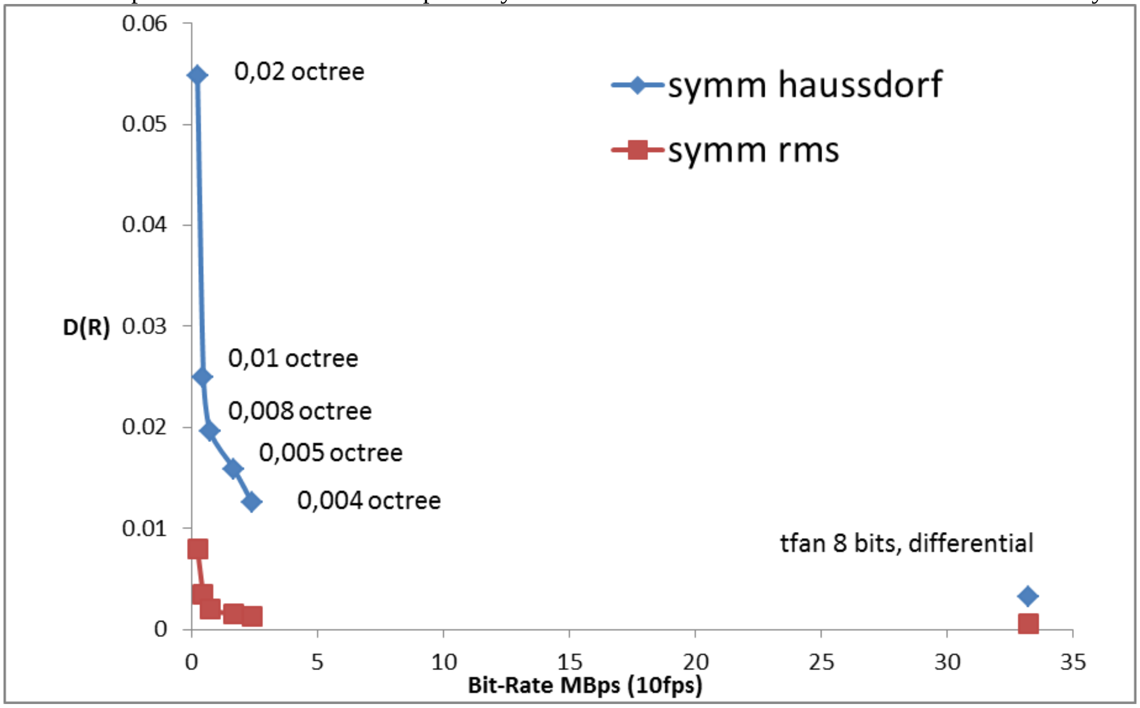

Figure 13 Rate distortion achieved with octree compression versus mesh compression

Additionally, the point cloud codec from [12] that we tested has an option for exploiting temporal correlation across frames via XoR operations between the subsequent serialized octree byte streams. In serialized octrees of subsequent frames, bytes in the beginning of the stream are expected to be similar due to slow temporal variation of the coarse surface. So each similar subdivision between frames would result in a zero byte that can be efficiently entropy. We tested this method on our dataset in [16] but did not get an improvement in coding efficiency. As Figure 14 shows, for the coarser octree representation (larger final voxel size), only very slight improvement was observed. On the other hand, for the more precise octree representation (smaller final voxel size of 0.001) no improvement and even a larger byte size was achieved. These results are quite data specific, but in our case this mechanism of XoRing between frames did not work well.

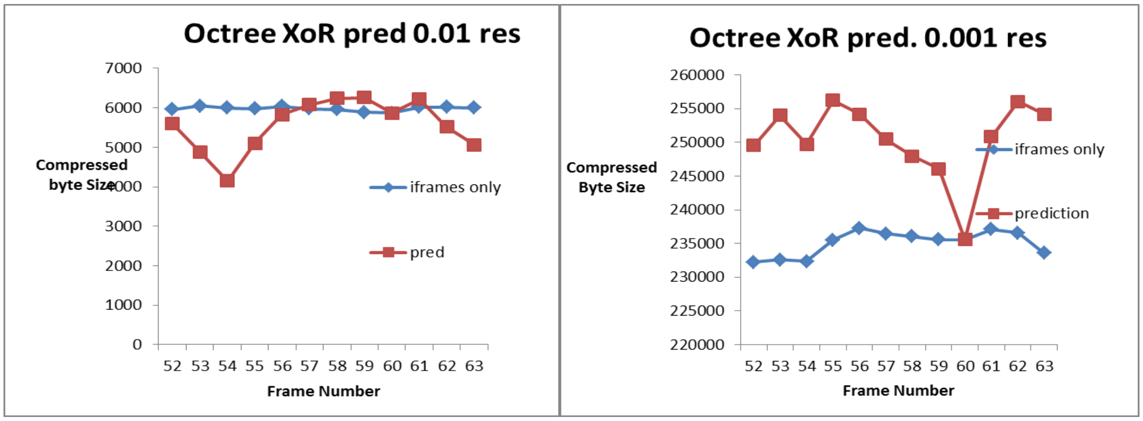

Figure 14. Xor prediction between frames for coarse octree (left) and precise octree (right)

4.5 Simulcast Based Multi-view compression

Lastly, we do a comparison with direct compression of the different views and depth images and reconstruction at the receiver side. We do the test for the reconstructed surfaces from [3] and [16]. We obtained the original Kinect streams resulting from 5 Kinects from [20]. We converted the original image plus depth formats and compressed them with the x264 encoder for MPEG-4 part 10/AVC video compression. While the multi-view codecs might give a better performance, we chose the x264 encoder due to its stability and the possibility to send views independently (simulcasting) and its real-time performance. For comparison purpose, we compress the entire multiple view plus depth data stream of N =8961 frames, measure the bytes size and divide it by the number of frames to get the average byte size per possible frame reconstructed.. One critical aspect is the compression of the depth information with a video codec while maintaining the geometric quality. It is hard to predict/know what the effect of depth image compression with a video codec is on the quality of the reconstructed surface. A recent work that tries to address this problem by modifying the 3D reconstruction itself was presented in [21]. Instead we leave this question open, we will compress the depth video with Quantization Parameter (QP) in x264 ranging from 8 (very good) to 48 very bad creating a global picture. We code the colors with QP 32 in x264 which is a medium quality. For both the depth and color we tuned the zero latency profile in the encoder. For each depth/color stream we execute the following command in x.264, resulting in raw compressed h264 streams.

```
X264 –qp  %QP% –tune zerolatency –o img_qp%QP%  img_in_file_0
```

We compare the results of the compression ratio achieved with the static mesh codecs TFAN and the CWI-Codec and indirectly with FAMC codec (which actually uses a different dataset, but it is still interesting to qualitatively compare). The results are shown comparing the average size per frame in Figure 15. The representations have very different qualities and represent different vertex count making them hard to compare. The static mesh codecs give the highest byte size as they preserve all the relevant vertices and do not exploit temporal prediction. The coding with H.264 comes in second in the tested range of parameters. This is a very advanced codec that can exploit motion very well, but apparently sending all the views to extract the surface introduces an overhead. The point cloud compression comes in with the third largest file size, this is mainly achieved by a quality reduction and a reduction in the vertex count. The smallest size was achieved with famc which was tested on a different dataset and without colors and is therefore marked with an asterisk. This result is partially due to the high quality reconstruction with a low number of vertices and the efficiency of the famc codec / time varying mesh animation.

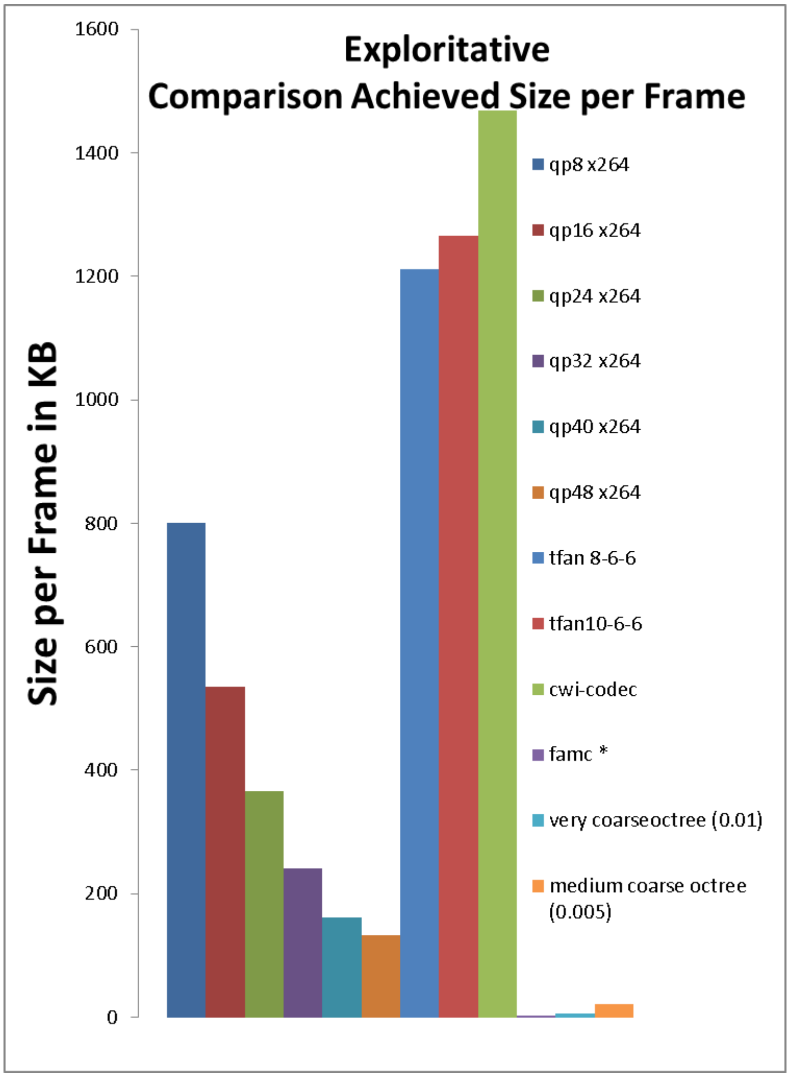

Figure 15. Comparative result of exploration coding of meshes, point clouds and multi-view

5. DISCUSSION

This paper presents an exploration study of the compression of reconstructed surfaces for tele-presence and 3D communications. Its main aim was to illustrate some of the tools and systems challenges associated with this. We have compared different methods based on distributing the functionalities of 3D reconstruction between client and receiver. We showed that compared to static mesh coding, multi-view coding of the original depth data may give better compression/bandwidth performance. However, this configuration loads the receiver side with more computation, especially when the number of senders increases. Therefore, such a solution seems unpractical in realistic systems. The best performance was achieved with FAMC (but without colors), which restricts the reconstruction system to produce time consistent geometry. We have not yet developed a full system with consistent 3d reconstruction and famc, and we expect that it is still quite challenging to develop such a system that works well in practice and operates in real-time.

We have explored point cloud compression, which turned out as an efficient way to reduce the size of dense surfaces at the cost of increased, but controlled distortions. While the point cloud codec operates in real-time, we sometimes had difficulties re-meshing the surface in terms of computational cost. For this reason, we are now looking into adding connectivity information to the compressed octree, based on the simplified original dense 3D mesh. We did integrate the point cloud compression in our realistic 3d immersive framework and we did achieve a good performance in terms of achieved frame-rates and transmission time, making it an already practically feasible solution.

REFERENCES

[1] Gürler, C.G.; Görkemli, B.; Saygili, G.; Tekalp, AM., "Flexible Transport of 3-D Video Over Networks," Proc. of the IEEE , vol.99, no.4, pp.694,707, April (2011)

[2] Diab K., Elgamal T., Calagari K., and Hefeeda M.. 2014. "Storage optimization for 3D streaming systems". *In* Proc. of the 5th ACM Multimedia Systems Conference *(MMSys '14). ACM, New York, NY, USA, 59-69 (2014)*

[3] Smolic A.: "3D video and free viewpoint video - From capture to display". In Pattern Recognition 44 (9): 1958-1968 (2011)

[4] ISO/IEC 14496-10:2008 (Amendment 1)., "Multiview Video Coding (MVC) Document number N9580 "January (2008)

[5] Mekuria R**.,** Sanna M., Izquierdo E., Bulterman D.C.A., Cesar P. "Enabling 3D Tele-Immersion with Live Reconstructed Mesh Geometry with Fast Mesh Compression and Linear Rateless Coding" *IEEE Transactions on Multimedia(2014)*

[6] Mamou K., Preteux T. Zaharia, and F., "TFAN a low complexity 3D Mesh Compression algorithm," In Proc. of Computer Animations and Virtual Worlds (CASA'09) pp. 343-354, (2009).

[7] Jang E.S.., Lee S., Koo B., Kim D., Son K. , "Fast 3D Mesh Compression using Shared Vertex Analysis," ETRI Journal, vol. 32, no. 1, (2010).

[8] Mamou K., Zaharia T., Preteux F. , "FAMC: The MPEG-4 standard for Animated Mesh Compression," in Proc. Of IEEE ICIP, San Diego, (2008), pp. 2676 - 2679.

[9] Nguyen H.. Q., Chou P., Chen Y. "Compression of Human Body Sequences Using graph wavelet filterbanks" In Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing Florence May (2014)

[10] Chen S., Xia P., and Nahrstedt K. (2013). "Activity-aware adaptive compression: a morphing-based frame synthesis application in 3DTI". In Proc. of ACM Multimedia *(*MM '13*)*. (2013).

[11] Domonouglou A., Alexiadis D., Asteriadis S., Zarpalas D., Daras P. , "On time varying mesh compression exploiting activity related characteristics" in Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing , (IEEE ICASSP 2014), Florence, Italy, May 4-9, (2014)

[12] Kammerl J., Blodow N., Rusu, R.B., Gedikli, S., Beetz, M., Steinbach E., "Real-time compression of point cloud streams," in Proc of IEEE ICRA, 2012, Saint Paul, MN, (2012), pp. 778 - 785.

[13] Alexiadis D., Zarpalas D., and Daras P.., "Real-Time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras,". In IEEE Trans. on Multimedia, vol. 15, pp. 339-358, (2013).

[14] Gall J., Stoll C., de Aguiar E., Theobalt C., Rosenhahn B., and Seidel H.P..
"Motion Capture Using Joint Skeleton Tracking and Surface Estimation" Proc. Of IEEE CVPR (2009)

[15] Touma C., Gotsman C., "Triangle Mesh Compression". In Proceedings of Graphics Interfaces, Canada (1998); pp. 24-34

[16] Alexiadis. D. (2013, Oct.) Datasets of Multiple Kinects-based 3D reconstructed meshes. [Online].
"http://vcl.iti.gr/reconstructions/"

[17] ISO/IEC 14496 16: MPEG 4 Part 16, Animation Framework eXtension (AFX).

[18] Rusu R.B., "3D is here: Point Cloud Library (PCL)," in IEEE International Conference on Robotics and Automation, Shanghai, (2011), pp. 1-4.

[19] Levoy M., Turk G. , "Zippered polygon meshes from range images,". In Proc. of SIGGRAPH '94, (1994), pp. 311-318

[20] Huawei/3DLife ACM Multimedia Grand Challenge for 2013: 3D human reconstruction and action recognition from multiple active and passive sensors http://mmv.eecs.qmul.ac.uk/mmgc2013/ (online, last accessed 7/23/2014)

[21] Sun W., Cheung G., Chou P.A. , Florêncio D., Zhang C., Oscar C. Au:" Rate-distortion optimized 3D reconstruction from noise-corrupted multiview depth videos".Proc of ICME (2013): 1-6

[22] Calakli F. and Taubin G., "SSD: Smooth Signed Distance Surface Reconstruction", Computer Graphics Forum, Vol. 30, No. 7, (2011)