Enabling Geometry Based 3D Tele-Immersion with Real-Time Mesh Compression and Linear Rateless Coding

Rufael Mekuria, Michele Sanna, Ebroul Izquierdo, Dick C.A. Bulterman, Pablo Cesar

***Abstract*—3D Tele-immersion enables participants in remote locations to share, in real-time, an activity. It offers users interactive and immersive experiences, but it challenges current networking solutions. Work in the past has mainly focused on the efficient delivery of image-based 3D videos and on the realistic rendering and reconstruction of geometry-based 3D objects. The contribution of this paper is a real-time streaming component for 3D Tele-Immersion with reconstructed geometry. This component includes both a novel fast compression method and a rate-less packet protection scheme specifically designed towards the requirements imposed by real-time transmission of live-reconstructed mesh geometry. Tests on a large dataset show an encoding and decoding speed-up of upto 10 times at comparable compression and quality rates, when compared to the high-end MPEG-4 SC3DMC mesh encoders. The implemented rate-less code ensures complete packet loss protection of the triangle mesh object and a delivery delay within interactive delay bounds. Contrary to most linear fountain codes, the designed codec enables real-time progressive decoding allowing partial decoding each time a packet is received. This approach is compared to a transmission over TCP and heavily outperforms it in packet loss rates and latencies typical in managed WAN and MAN networks. The component has been integrated into a larger environment that includes state of the art 3D reconstruction and rendering modules. This resulted in a prototype that can capture, compress transmit and render triangle mesh geometry in real-time in realistic internet conditions as shown in experiments. Low interactive end-to-end delay and frame rates over 3 times higher compared to alternative methods are achieved.**

***Index Terms*—Multimedia Communication, Multimedia Systems, Block Codes, 3D Mesh Streaming, 3D Tele-Immersion, Source Coding, Geometry Processing**

I. Introduction

ADVANCES in 3D reconstruction and rendering and the success of inexpensive consumer grade depth cameras enable, in real-time, the creation of highly realistic representations of participants as triangle mesh models (see Fig. 1). Efficient real-time transmission of these representations opens up new possibilities for 3D Tele-Immersion mixing real and virtual environments. 3D Tele-Immersion has been studied in the past for a variety of application areas such as creative dancing, cyber-archeology, medicine, and gaming [1] [2] [3] [4]. For example, a user can stream a 3D geometric representation to a remote site interacting with several other users in a 3D space for working on a common task. However, existing video codecs and transmission schemes do not support live reconstructed geometry well and neither do geometry streaming mechanisms intended for downloading and interacting with remotely stored geometry-based objects. This paper takes a step towards the direction of real-time streaming of live reconstructed geometry, by reporting on our efforts in developing a complete end-to-end prototype system that is capable, in real-time, of efficiently transmitting this media type between remote locations. Results show that our solution outperforms existing mechanisms, when transmitting high quality reconstructed 3D geometry representations. Looking ahead in the future, when challenges regarding calibration and synchronization of the different depth cameras are solved, 3D Tele-immersion can become integrated into social network experiences similar to current video conferencing or gaming features. In order to enable real-time transmission of such geometric representations over the Internet an efficient compression and streaming mechanism that can operate in a realistic environment is needed. For multiview video, many standardized compression and transmission methods exist, but for live reconstructed geometry support is currently limited. This paper introduces the streaming requirements and describes a working prototype for real-time streaming of live reconstructed geometry. The experiments are run both by using the integrated prototype with real-time rendering and reconstruction and by using offline collected data captured with five consumer grade depth-cameras (see Fig 1). This paper is an extension of our previous conference publication [5], in the current paper we extend this configuration with a novel connectivity driven mesh codec. Also, we have made the rateless coding progressive and performed a more extensive performance evaluation of the complete integrated media pipeline in representative network conditions.

Manuscript received December 1, 2012; revised December 27, 2012.
Corresponding author: R. Mekuria (email: rufael.mekuria@cwi.nl).

A. 3D Representation

In this paper we investigate geometry-based 3D mesh representation reconstructed in real time from multiple depth cameras as in [7]. In Eq. 1 - 3 we define a sequence of reconstructed triangle meshes introduced by a live reconstruction system. The individual vertices v, contain 3D coordinates (x, y, z), the normals (n_x, n_y, n_z). and the colors (r, g, b). The faces f are triangulations of the vertices that are indexed (v_1, v_2, v_3). Each mesh in the sequence from $i = 1, ..., K$ contains a set of vertices V and faces F that index vertices and represent a participant by a 3D surface.

$$M^i = (V^i, F^i), i = 1....K, \quad (1)$$

Fig. 1. A screenshot of a live reconstructed mesh with the system described in [7], datasets of this system are available online and currently used in for evaluation in Motion Picture Experts Group (MPEG)

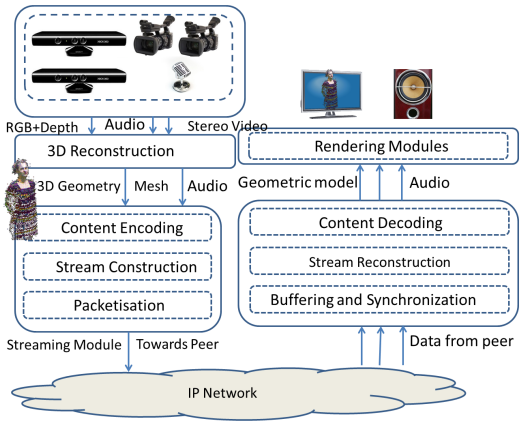


Fig. 2. Pipeline for 3D Tele-immersion with live-reconstructed geometry, instead of live captured video, a 3D Geometric mesh is reconstructed for visual communication

$$V^i = (v_1^i, v_2^i, v_3^i ....v_{Ni}^i), \tag{2}$$

$$F^i = (f_1^i, f_2^i ... f_{Mi}^i) \tag{3}$$

B. Media Pipeline

Fig. 2 shows the 3D tele-immersive media pipeline based on live reconstructed geometry. First, participants are captured in real-time and the 3D reconstruction module reconstructs the 3D geometry. The result is a 3D Mesh sequence Eq. 1 that is transmitted in real-time to a remote host over an IP network. The receiver renders the received mesh in the scene, compositly with other participants.

C. Research Questions and Objectives

Real-Time streaming of live-captured (multi-view) video is common in video conferencing systems, but streaming of live reconstructed mesh geometry objects has rarely been considered. There are various reasons why we consider efficient real-time transmission of live-captured triangle geometry essential. First, modern graphics cards can take advantage of advanced rendering methods, such as multiple views for stereo and multi-stereoscopic or free-viewpoint rendering. Second, it allows integration with virtual worlds, where triangle mesh representations are common. Finally, novel application areas can emerge such as natural interactions between people in real and virtual worlds. Geometry streaming solutions can also be beneficial for applications like camera or terrain surveillance, where geometric data is live-captured and needs to be available in real-time to by observers or Geographic information systems. In this paper we are concerned with the 3D Tele-Immersion application between participants. In particular, this paper aims to answer the following research question: *What is an efficient way to transmit live-captured triangle mesh geometry in real-time over the internet, as needed for 3D tele-immersion?* Our main contribution is the design and development of a streaming module that takes into account the specific requirements for streaming captured meshes. The requirements are the following:

Support a full 3D triangle mesh representation: the engine should support streaming of the common 3D triangle mesh representation. That is, a list of points with properties (coordinates, normals colors) and a list of faces indexing these points, resulting into a surface in the 3D space. Low end-to-end Frame Delay is generally considered an important factor in 3D tele-immersion, as the pipeline consists of bandwidth and computation savvy operations. For this paper we aim to keep the end-to-end frame latency below 300 ms, based on video conferencing requirements.

Flexible I/O representation: the data should efficiently flow from the capturing and reconstruction blocks, via the streaming engine, to the renderers without blocking. To allow for minimum pipeline delay and possible synchronization between different streams a flexible I/O scheme is needed.

Adaptability: the mechanism should be able to adapt to changing network conditions such as bandwidth, possibly reducing the quality of the captured stream. Some rate/complexity control is desirable.

Robustness to packet loss as it occurs in congested networks is desired. Some quality degradation may happen, but it should be possible to reconstruct the triangle mesh at the receiver in case of packet loss. Generally, dependencies between connectivity and geometry data and the nature of mesh codecs transmission of mesh geometry is sensitive to data losses.

Bandwidth: the triangle mesh stream should not consume too much bandwidth; some form of compression is desired that matches the state of the art. No a priori information of the geometric properties of the objects can be assumed (non-manifold/ open/closed oriented or not). Similar to video conferencing, any mesh object that is captured should be streamed. This also means that no pre-stored avatars or models can be transmitted as placeholders. realistically, we aim for state of art rates as currently provided in the MPEG-4 standards for mesh coding.

Real-time live-captured triangle meshes should be supported. Contrary to live captured video, where frames generally consist of a fixed number of points (320x240, 640x480), captured triangle mesh frames can have different numbers of vertices and no such vertex correspondence between frames can be made unless supported by the reconstruction system. This is often referred as time-varying geometry.

TABLE I
TERMS AND ABBREVIATIONS USED THROUGHOUT THE PAPER

3DTI	3D Tele-Immersion
MPEG-4	multimedia coding standard, includes 3D mesh codecs
MPEG-4 tfan	high-end mesh codec in MPEG-4
MPEG-4 sva	low-end mesh codec in MPEG-4
CZLoD	3DTI representation color+ depth in [9]
DPCM	Differential pulse code modulation

D. Contribution and Outline

This paper presents a component that can efficiently stream in real-time triangle mesh geometry, which is live captured and reconstructed. The component has two main subcomponents: encoding and transmission. First, we introduce an encoding mechanism that reduces the size of the reconstructed mesh to rate-distortion values comparable to those obtained with a state of the art TFAN MPEG encoder, but approximately 10 times faster. This meets our requirements on latency and bandwidth. Second, we provide a transmission scheme that uses rate-less coding, meeting the requirements of robustness, latency and adaptability to changing network conditions shown experimentally via network emulations. This paper is structured as follows. Section II overviews the related work regarding existing 3D Tele-Immersion systems, mesh geometry compression algorithms, and mesh geometry transmission solutions. Section III presents a compression method we have designed for the live meshes In Section IV we evaluate this method, introducing the datasets, quality criteria and the compression results. Section V describes our proposed transmission scheme based on a rate-less code, reporting computational and network transport delay. Section VI then presents our 3D Tele-Immersion system, highlighting integration and the performance of the overall system. We discuss the conclusions and implications of our work in Section VII.

II. RELATED WORK

A. 3D Tele-Immersion

3D Tele-immersion has received considerable attention; we highlight some of the current advances. Vasudevan et al. proposed a 3D tele-immersion system for capturing and rendering that can reconstructed multiple meshed depth images. An advantage of this technique is that by interpolating points and by changing the size of the triangles it is possible to adapt the level of detail [8] . Wu et al. developed a streaming engine that exploits this representation, Color plus depth (Z) and Level of Detail (CZLoD) [9]. The authors found the just noticeable degradation and just acceptable degradation levels in a user study, and subsequently applied dynamic adaptation of the CZLoD, depending on network and user conditions. With this engine, CZLoD can be adapted to match user perception in real-time for the given available network bandwidth and user conditions. Contrary to our approach, the CZLoD representation is a triangulation on a depth image, while the polygonal mesh we consider is a triangulation in a full 3D space. An overview of technical challenges and related user experiences based on over a decade of experimental 3D Tele-Immersion research can be found [6]. Fast frame compression has been a bottleneck in 3D Tele-immersive systems design and several methods have been proposed to address this issue. [10] propose a real-time compression method for a multi view plus depth based 3D tele-immersive system based on clustering streams together and predicting all streams in a cluster to a single stream . Yang et al. propose a method for streaming multiple depth streams with segmented background based on zlib library enabling real-time performance with reasonable compression gains [11]. These methods do not support the 3D mesh representation that has only has become available recently for use in 3D Tele-immersive systems [7].

B. Geometry Compression

A considerable body of work on geometry compression and a survey is available [12]. In theory, high compression rates up to 4 bits per vertex can be achieved when specific topologic and geometric properties are met. However, in practice 3D meshes can be manifold or not, open, closed regular or irregular, oriented or not and manageable computational complexity is needed. Therefore, the MPEG standardization has adopted the novel Triangle Fan coder that can compress irregular, non-manifold meshes with state of art compression rates and very fast decoding times. Evaluations show that a compression gain of 5 perc. compared to state of art is achieved on manifold datasets and even more on large industrial datasets that includes disconnected (isolated) components [13]. It was also shown that in the case multiple isolated components exist state of art geometry compression methods based on spectral, wavelet and progressive mesh introduce artifacts [13]. In the case of 3D Tele-immersion with geometry reconstructed on the fly, encoding complexity/time becomes critical when compared to previously considered applications. The constraints on encoding time and geometric properties make most methods in the literature not directly applicable to 3D Tele-Immersion.

C. Geometry Streaming

Various solutions have been proposed for efficient progressive download and streaming of progressively compressed mesh geometry from a server. Methods such as 3TP [14] and extensions such as [15] generally utilize an extensive offline optimization scheme based on minimizing a geometric distortion metric to the orignal geometry to decide on an optimal transmission scheme. This is not possible in 3D tele-immersion where the original geometry is reconstructed and therefore not available offline. In a recent work [16] a TCP like connected transmission is seen as the most desirable for transporting geometric data. Instead, in this paper we propose compressed mesh transmission with a linear rate-less code optimized for real-time operation for 3D Tele-immersion and compare it with connection oriented transmission via TCP.

III. FAST COMPRESSION METHOD FOR LIVE-RECONSTRUCTED GEOMETRY

We consider the reconstructed 3D mesh sequences, as a temporally incoherent mesh sequence as defined in Eq. (1). We

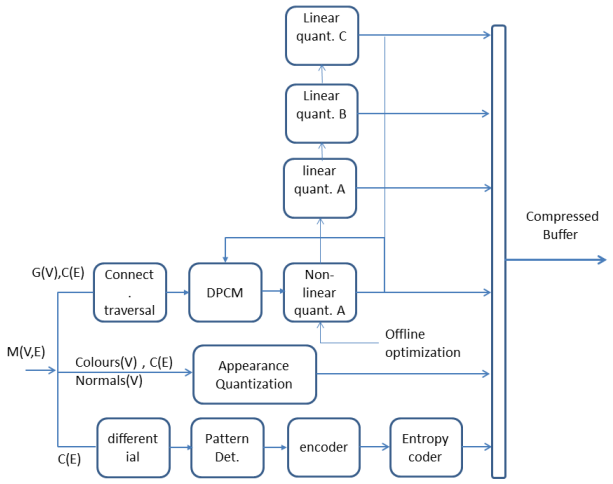


Fig. 3. Outline of Proposed Geometry Compression Scheme

TABLE II
DATASTRUCTURE PATTERN RUN

mode	Diff1	Diff2	Start	Count

TABLE III
AN EXAMPLE OF A PATTERN IN THE CONNECTIVITY LIST

1632	1520	1633
1520	1521	1633
1633	1521	1634
1521	1522	1634

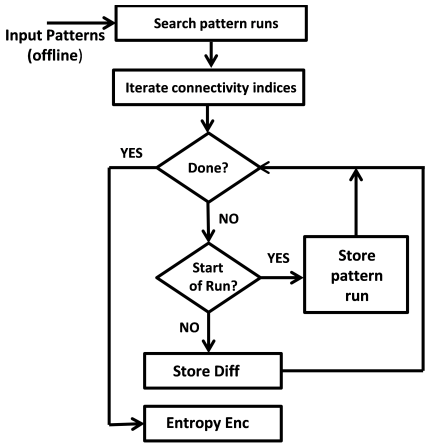


Fig. 4. Connectivity Compression scheme outline

aim to compress this data with a rate-distortion comparable to available methods, but with a lower computational complexity resulting in real-time encoding making it suitable for 3D Tele-Immersion. Fig. 3 outlines the compression system. We introduce a fast connectivity traversal method combined with connectivity-driven (offline) optimized DPCM coding and layered quantization. We explain the rationale and operations for compressing the connectivity C(E) in the next sub-section. The differential encoding of the geometry $G(V)$ followed by non-linear quantization is presented in III-A. In section III-B. we discuss how the appearance ($normals(V)$ and $colors(V)$) are handled (appearance quantization).

A. Pattern Based Connectivity Coding

The idea behind the connectivity coding approach presented in this paper is that 3D reconstruction can introduce specific regularities in the connectivity information that can be exploited for compression purposes. For example, in the zippering method [17] multiple range images are tessellated first into range surfaces that are subsequently zippered (stitched) together. If these range surfaces are tessellated in a consistent order, this can introduce a more regular and predictable connectivity structure. Such patterns were also found in the connectivity of the reconstruction data [7]. As such, patterns occur many times in a row, we actively search for them and use them for efficient encoding. An example of how following differences occur is shown in table III , where each next index is an increment of 1. While such patterns might differ per reconstruction method and need to be found before they can be exploited, they are a key to enable low-complexity encoding of large meshes. Fig 4 illustrates the connectivity compression scheme. First, the entire connectivity information is searched for repeated regularities which are counted and stored in the data structure pattern run shown in Table II. The *mode* field represents the type of pattern, the *diff* fields the two differences that occur and the *count* field signals the number of repetitions. The *start* field is used to reference the position of the first index in the connectivity list. This value is used to detect the start of a run in the next encoding step and in the decoder. Next, all indices are iterated again. If an index is the start of a run the pattern run (indicated by the *start* field), is stored and the connectivity index iterator is increased with the *count* field. If an index is not stored in a run, the difference with the index in the previous face is stored instead. Storing differences instead of absolute values yields mostly small numbers skewed around 0 and 1 and therefore allows efficient entropy coding. The resulting data vector is entropy encoded via the zlib library [19].

B. Geometry Coding With Delayed Differential Quantization

Uniform quantization is used for geometry compression in MPEG-4 SC3DMC and other methods, which have been designed with high quality graphical models with large quantization parameter (QP 10-20). For the low-bit rate requirements for streaming live reconstructed 3D geometry this has several disadvantages. Firstly, the dynamic range of the vertices that are quantized in a 3D space is large compared to the details in the 3D surface. This results quantization artifacts quickly become visible to the viewer. I.e. the surface is vulnerable to quantization distortion that increases with low-quantization parameters. When low quantization parameter are used for geometry compression, vertcices may share positions resulting in degenerate triangles (faces with a zero surface) and a blocky appearance. Therefore, instead we propose quantization after the transform (DPCM) with a variable number of bits. In our case we propose a

layered structure: most vertices are quantized with 4 bits, but values outside the 4-bits range are quantized with 8 bits, 16-bits and 32-bits respectively when needed, i.e., if the differences become large. This way we avoid large errors in the geometry, as even for a small amount of vertices large quantization errors can become clearly visible when the mesh is rendered. We code differences between vertices that are connected to each other. In this case the differences are even more fine-grained as by the nature of densely sampled 3D reconstruction, connected vertices are closely co-located. The primary quantization vector codes over 95 percent of the values and is defined as for the given datasets: $[-.011, -.0063, -.0042, -.00315, -.00236, -.0012, -.0004, 0, .0004, .0012, 0.00236, .00315, .0042, .0063, .011]$. In this configuration for the 4 bits non-linear quantization vector was computed offline. Since it can be dependent on the reconstruction method, we envision that such information can be exchanged out-of band (in our 3D Tele-Immersion system we use a custom session management system to exchange such information). Subsequently, we utilize a layered structure with 8-bits for the range from 0.11 to 0.1, and 16bit between 0 and 2 to cover the entire dynamic range. This information again depends on the reconstruction system and can also be sent out of band when a user joins a 3D-Tele-Immersive session with a specific setup. The lower the threshold *thresh* of the 4-bits range, the better quality and lower the compression gain will be. To perform connectivity driven differential encoding, we defined and implemented a traversal method that does not changes the order of the vertices in the list. Each of the faces is traversed, and the first connected vertex that was previously stored (set) is used for differential prediction. If none of the connected vertices have been previously traversed and set, we set the first vertex index in the face by adding this value to the reserve values list. The other connected vertices in the face are then differentially encoded. At the decoder, we traverse the connectivity in a similar manner, unset vertices are loaded from the reserve values list and the other vertices are recovered via inverse differential prediction. This method works well as the reconstructed meshes are relatively coherent (connected vertices are also closely located in the indexed face list). In practice less than 0.1 percent of the vertices are stored in the reserve list in some of the reconstruction systems we have tested. Values in the range $[-0.011, 0.011]$ are stored and appended in the 4-bits vector, values between $[-0.1, 0.1]$ are quantized with 8 bits and appended to the eight bits queue vector. Larger differential values are stored in the 16-bits or 32 bits queue but occur very rarely. They need to be stored accurately to guarantee decoding without large distortions.

C. Appearance Quantization

In the case of surface mesh geometry, the perceived appearance depends not only on the color but also on the geometry coordinates and surface normal data. We deploy the same system of late differential quantization for the appearance data (normal and colors), however the layers are assigned differently. We quantize normal components in the range from -0.25 0.25 with 4 bits and the rest with eight bits (8 bits plus the sign bit retrieved from the 4 bits quantization vector). We code color differences between -25 and 25 with 4 bits and the rest with 8 bits (precisely 8bits plus the sign bit retrieved from the 4 bits quantization value covering the entire range $[-255, 255]$. Again, such information can be communicated out of band prior to the media streaming session to configure the encoder and decoder properly.

IV. EVALUATION OF FAST COMPRESSION METHOD

A. Datasets

For evaluation of the compression method we use datasets of meshes reconstructed with five depth cameras that are currently used in the 3DG group of Motion Picture Experts Group (MPEG). They have been created by the Center for Research and Technology Hellas (CERTH) based on the method reported in [7] and contain participants performing different activities. The datasets are publicly available at the website currently hosted at [20]. These datasets represent the case where a user is in a room captured by multiple depth cameras at a distance of 300 cm. We have also integrated the reconstruction system with our 3D mesh streaming engine and have captured several test sets with one depth-camera representing a user that is sitting in front of his computer (at 1.30 meters). We use both datasets to evaluate the developed method.

B. Quality Evaluation Metrics

We utilize two metrics to assess the geometric quality of the mesh: the symmetric Haussdorf distance and similarly the symmetrical root mean squared distance, computed between the original and decoded surface. We give the definition of this metric in this section. The distance between a point p and surface M is defined as:

$$d(p, M) = min_{p' \epsilon M} |p - p'|_2 \tag{4}$$

From this definition the Haussdorf distance between surface M and M' is defined as:

$$d(M, M') = max_{p \epsilon M} d(p, M') \tag{5}$$

As this metric is generally not symmetric $d(M, M')$ $d(M', M)$ we use the symmetrical Haussdorf distance which is defined in equation 6 as:

$$d_{sym}(M, M') = max[d(M, M'), d(M', M)] \tag{6}$$

Similarly the root mean square error, defined as

$$d_{rmse}(M, M') = \sqrt{1/|M| \int \int_{p \epsilon M} (d(p, M'))^2 dM} \tag{7}$$

where $|M|$ denotes the area of surface mesh M, and the symmetrical root mean square error is then defined as

$$d_{rmssym}(M, M') = max[d(M', M), d(M, M')] \tag{8}$$

TABLE IV
QUALITY METRICS NOTATION

M	original surface mesh
M'	distorted/decoded surface mesh
$d_{rmssym(M,M')}$	symmetric geometric rms distance between M and M'
$d_{sym(M,M')}$	symmetric hausdorf distance between M and M'
$d_{app(M,M')}$	rms error in appearance on a per vertex basis
$d_{uniform}$	quantization error(rms) uniform source and quantizer

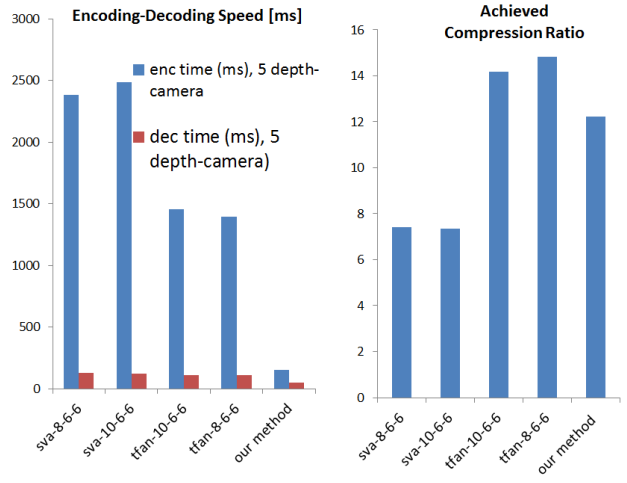


Fig. 5. Encoding and decoding results with 5 cameras

In the rest of the paper, when we compare surfaces, rms refers to 8. To facilitate the computation of these metrics we use the tool developed in [21]. Alternatively, we compare the quality of the colors and normal (appearance) and the normal with root mean square error on a per vertex basis.

$$d_{app} = \sqrt{1/(|V|) \sum_{p \epsilon M, p' \epsilon M'} |(p - p')|_2^2} \quad (9)$$

Where p and p' denote the 3 coordinate color/normal in original mesh M and decoded mesh M' respectively. As the tfan codec does not preserve the order of the vertices we compare to the SVA codec at the same quantization parameter. Also, we compare against the quantization error introduced by quantization of a uniform source with a uniform quantizer in 3D which is given by

$$d_{uniform} = \sqrt{3}\delta/\sqrt{12}) \quad (10)$$

Where δ is the quantization step size. Lastly we compared the encoding and decoding times in the integrated codec in the 3D Tele-immersive system in milliseconds and provide screenshots of the original and decoded meshes. In table IV we give an overview of the notations of the quality metrics.

V. EXPERIMENTAL RESULTS

Our scheme was implemented in C++ and the test machine was a desktop machine with Intel i7 CPU and 8,00 GB or RAM and a Geforce GTX 760 video card. The MPEG-4 Codecs were compiled from source code with the same compiler as available on [22]. To avoid overhead of

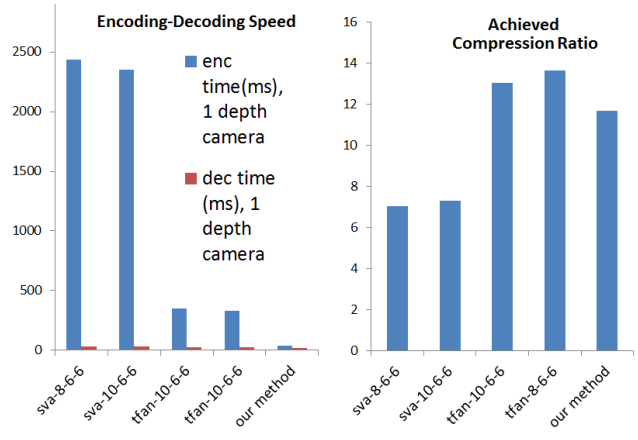


Fig. 6. Encoding and decoding results with 1 camera

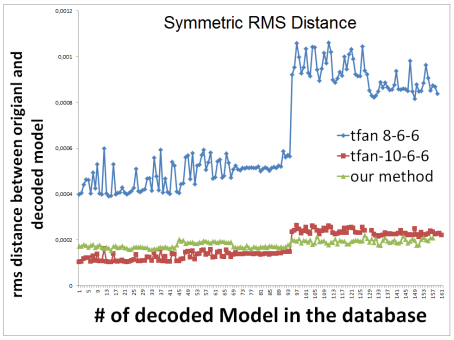


Fig. 7. distortion between original and decoded models : sym. rms

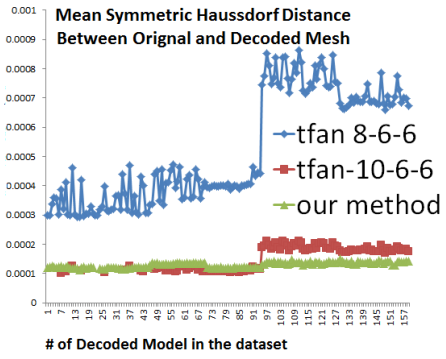


Fig. 8. distortion between original and decoded models : sym. haussdorf

TABLE V
ENCODER PARAMETER SETTINGS

tfan-8-6-6	tfan 8 bits coordinate 6 bit normal, 6 bit color
tfan-10-6-6	tfan 10 bits coordinate 6 bit normal, 6 bit color
sva-8-6-6	sva 8 bits coordinate 6 bit normal, 6 bit color
sva-10-6-6	sva 10 bits coordinate 6 bit normal, 6 bit color
our method	uses proposed method with delayed 4 bit quantization

MPEG-4 part 25 that deals with loading the supported text formats [23](XMT/Collada), we interfaced directly the class SC3DMCEncoder and SC3DMCDecoder with an MPEG indexed face set structure that is directly loaded from the input

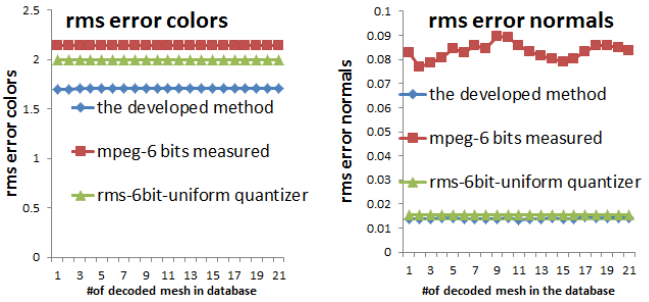


Fig. 9. Quality Comparison colors and normals between original and decoded models (rms), the proposed codec gives lower rms error compared to the mpeg configurations tested

Fig. 10. Quality Comparison, original, mpeg and our method decoded meshes

mesh. This way any overhead delays in accessing the mpeg codec are avoided. All files are first completely loaded into memory before the compression routine is started. The running times are recorded via CPU wall clock times provided by boost C++ library with a resolution of 366 ns. We ran all methods on N=158 Meshes with 5 depth cameras with average of 302K Vertices. As Fig. 5 shows, with our method a significant speedup is achieved compared to other methods (151 milliseconds with 5 Kinect data in average in comparison to 1398 milliseconds on average with TFAN or 2400 ms with SVA). In terms of compression size, in Fig. 5 we achieved on average mesh size of 1,460 KiloBytes (ratio 12.3:1) compared to 1266 KiloBytes with TFAN MPEG (14:1) with 10 bits QP and 6 bits colors and normals. While the measured compression ratio is about 15 percent lower compared to TFAN, we achieve a speedup of over 10 times. A comparable result is achieved with 1 camera as shown in Fig. 6. The results when the meshes are reconstructed with 1 depth camera (average of 72K vertices) are consistent. Here we encode meshes in 35 ms on average and the average frame size is reduced to 370Kb (a 12:1 ratio). The evaluation of the symmetric quality metrics Fig. 7 and Fig. 8 show that comparable quality of geometry is achieved in both metrics. The quality of the normal and colors is a bit better with our approach compared to MPEG-4 as lower symmetric root mean square error is achieved. The results show that the 8 bits MPEG achieves a lower quality. The 10-6-6 bit mpeg encoded meshes and those with our method are closely clustered around 0,0002 (rms) indicating that their quality is approximately equal. We compared the quality of the colors and normals on a per vertex basis. As TFAN re-orders

TABLE VI
AVERAGE BANDWIDTH REQUIREMENT RESULTING FROM COMPRESSION

.	5 fps	8fps	10fps	12 fps
1 Camera	14.8 Mbit	23 MBit	29.6 MBit	35.52 MBit
5 Camera	58 Mbit	93 MBit	116 Mbit	134 Mbit

vertices we compared only with SVA, Fig. 9 and to $\sqrt{3}\delta/\sqrt{12}$ which is the root mean square quantization error introduced when quantizing a uniformly distributed source uniformly in 3D with step size δ. This value roughly corresponds to the quantization error achieved with the MPEG SC3DMC codec TFAN. The results show that our method achieves slightly lower distortion of the normal data and the colors compared to this value and the measured value. In Fig. 10 we show snapshots of the decoded meshes, most artifacts are related to the 3D reconstruction method and not the compression method. In the future, with better acquisition these artifacts will reduce. We show the resulting bandwidth requirements in table VI.

VI. V. REAL-TIME TRANSMISSION

In this section we introduce the concept of rate-less coding and we compare it to resilient transmission via TCP, based on a number of lab experiments, and show its favorable properties for geometry transmission and 3D tele-immersion. Symbol-based inter-packet coding, together with progressive decoding based on Gauss-Jordan elimination are used to produce a rate-less coded stream of arbitrary size, with controlled decoding complexity. We aim at the more reliable and bandwidth enabled inter-networks where packet loss ranges between 0 and 2 percent. In this case, a single frame of around 300Kb would result in 300 UDP packets the chance a frame would arrive $E[framearrives] = E[nolostpacket]$ would be less than 5 percent in case of a 1 percent loss rate. Note that the computationally heavy 3DTI receiver can also miss packets at the interface due to threads missing hte packer, increasing the loss rate. To combat this we introduce the rateless codec to maintain proper delay characteristics and gauranteed frame transmission.

A. Rateless Coding

Random Linear Coding aims to achieve packet loss protection with near optimal rate and quick adaptation to the network conditions. The idea of rate-less and fountain codes is that any amount of packets can be generated at the sender (i.e. rate-less). The first practical random linear codes were first proposed in [24]. An advantage of the rate-less property is that in case of increased packet loss in the network, the data generated can be increased for extra protection. This constitutes one of the main advantages compared to traditional fixed rate FEC codes such as Reed Solomon codes. The receiver then only has to receive a minimum set of packets to make sure the reconstruction of the frame is possible. A symbol based version of rate-less codes, more similar to our proposed technique, has been also adopted in the field of network coding to allow receivers to decode from incoming packets from several independent paths.

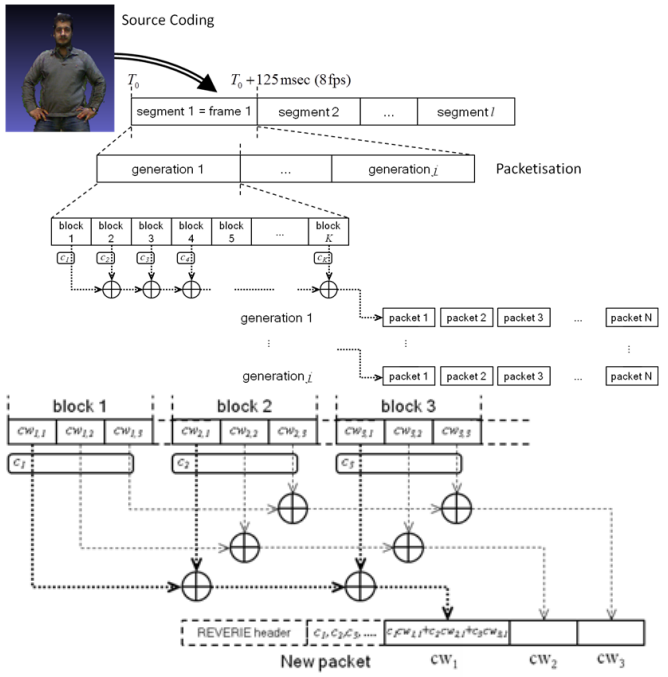


Fig. 11. Arrangement of blocks for rate-less coding(up) Schematic encoding of an outgoing packet(below)

B. Implementation

The data stream is divided in segments that are encoded together, e.g., frames containing the compressed triangular mesh at a certain instant. We further divide these segments in generations and data blocks as shown in Fig. 11. Each generation consists of 10 blocks and the size of the blocks is chosen to avoid possible Ethernet fragmentation, i.e., 1024 bytes. Linear coding is then performed on finite fields Galois Field (GF) of size $q = 2^m$, where m is the number of data bits covered by a codeword. Each block is considered as a sequence of N_W code words:

$$\mathbf{b}^{(k,g)} = [b_1^{(k,g)}, ..., b_{N_W}^{(k,g)}], k = 1, ..., K_g, g = 1, ..., G \quad (11)$$

The superscript (k, g) indicates the $k-th$ block of generation g. A field size q=256 (m=8 bits) has been chosen to allow fast algebraic operations. Outgoing packets are generated by linearly combining only the K_g source blocks of the specific generation, with coefficients $c_1, c_2, , c_{K_g}$. A coded block is generated from generation g as:

$$\mathbf{b'}^{(n,g)} = [b_1'^{(n,g)}, ..., b'^{n,g}_{(N_W)}], n = 1, ..., N, g = 1, ..., G \quad (12)$$

And each code word is calculated as in ((eq. 13)):

$$b_j'^{n,g} = \sum_{k=1}^{K_g} (c_k b_j^{(k,g)}), j = 1, 2, ..., N_W \quad (13)$$

Therefore, $N > K_g$ out-going blocks are generated. As only K_g linearly independent blocks are needed to decode the original data around $N - K_g$ redundant blocks are generated. The coefficients of the linear combination are embedded in the packet header. They are needed by the receiver to build the linear system. As soon K_g linearly independent packets are received for a generation, the $K_g K_g$ linear system is recovered and solved. The solution is then applied on the received blocks to recover the original data. In order to reduce the decoding computational load, we construct n intermediate composite matrix of data and coefficients of the incoming packets. Gaussian elimination is performed each time a new packet is received. This spreads the computational cost of solving the linear system over time and drastically reducing the eventual decoding time when K_g linearly independent blocks are received. The complexity can still be reduced more by reducing the dimension of the coding space. Therefore, in contrast with traditional fountain codes [31] [30], our rate-less coding has been restricted to one linear system per generation (as opposed to the N_W kernels per generation employed by normal fountain codes). In table VII we summarize the notations of the rateless code.

TABLE VII
NOTATION LINEAR RATELESS CODING

G	number of generations to partition source data
K_g	number of source blocks per generation
N	number of linear coded blocks per generation
$GF(q = 2^m)$	Galois finite field with q words represented by m bits
N_W	number of codewords per block
$\mathbf{b}^{(k,g)}$	source block k in generation g
$\mathbf{b'}^{(n,g)}$	coded block n in generation g

C. GF arithmetic

We implemented the non-sparse deterministic codes based on Vandermonde matrices and the progressive decoding via Gauss-Jordan elimination. The Vandermonde matrices are generated as a geometric progression of the rows as follows [25]:

$$V = [v_i^{j-1}], \quad (14)$$

where v_i, $i = 1, ..., 2^m - 1$ are the elements of the Galois field excluding the null element. We augment the matrix by concatenating an identity matrix of size K_g. Our code space of size K_g can be thus spanned by all $K_g + 2^{m-1}$ columns of the augmented matrix. By using a field of size 256 (8 bits per symbol), a generation composed of 25 blocks (25 KBytes) can be encoded into 280 independent packets, which is equivalent to a 9 perc. information rate (or 10 times redundancy for loss protection), and decoded from any 25 packets that arrive to destination first. We made use of a library implementing fast Galois Filed arithmetic [27] [26] using the latest Intel Streaming Single-Instruction Multiple-Data (SIMD) extensions. Such set of instruction allows 128 bits numbers to be handled in the CPU, making operations such the region multiplication of GF symbols, extremely fast with respect to previously available implementations. We make heavy use of the region multiplication, one example being the encoding operation in Eq. 12. Additionally, region multiplication is used for all row-operations needed for Gaussian-elimination, reducing the matrix to row-echelon form.

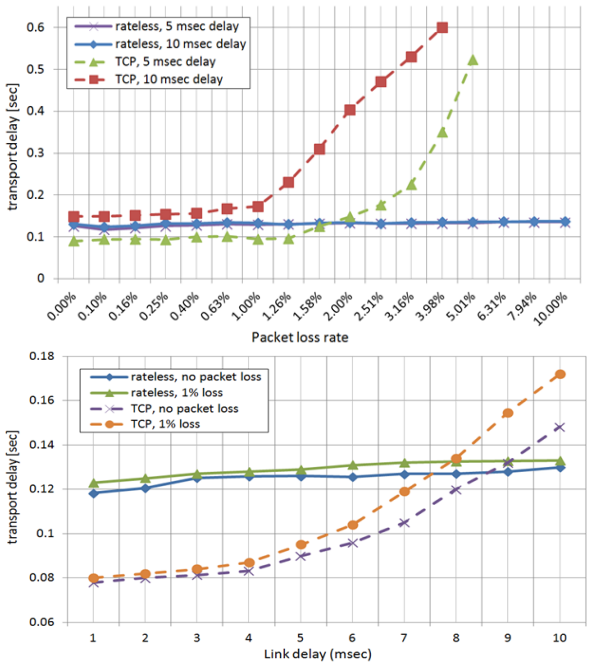


Fig. 12. End-to-End delay of rateless code implementation based on packet loss in the network

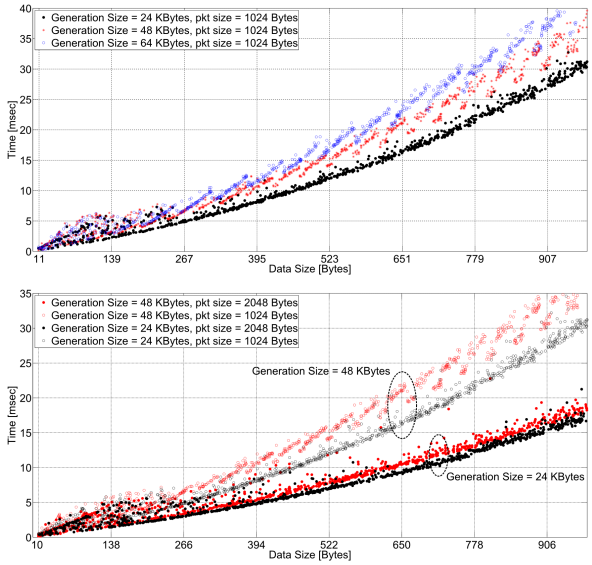


Fig. 13. Frame decoding times with fixed packes size and variable generation size and with different packet sizes

D. Experimental Results Rateless Codec

In order to assess the delay performance of the rate-less coding system we tested a point-to-point transmission between two PCs, one compressing and streaming out the data, and the second one receiving from the network, decoding the packets. A network emulator [28] was used to emulate latency and packet inside the LAN between the two machines. We compare our system performance to performance of a TCP connection tuned for low delay. Fig. 12 shows the performance comparison between the two in case of packet loss (below) and link delay (above). Contrary to TCP, the end to end delay remains small for packet loss over 1.5p and link latency over 9 ms in our approach. Fig. 13 shows the decoding times of data frames with sizes ranging from a few Kbytes to around 1 MByte. The decoding times consistently stay below 30 msec, enabling decoding above 30 frames per second. Fig. 13 (up) shows configurations with different generation size, which implicate bigger decoding kernels and increased complexity but better packet loss protection. Fig. 13 (bottom) shows reduced complexity at equal generation size, when using larger packets (but possibly larger information loss). Our rate-less transmission scheme is always able to sustain the transmission when the source throughput and the channel rate are higher than the data rate and the packet loss. Delays and packet losses affect only linearly the transport latency. This is due to the fact that, regardless the number of packets that are lost, as long as at least K_g packets are received the data is reconstructed. Realistic network delays will be over 9ms, and the rateless code is expected to yield a better delay performance. While this type of coding introduces overhead (we set 33 percent in our case), the overhead can be decreased to match network conditions making approriate use of the available bandwidth.

VII. 3D TELE-IMMERSION PIPELINE INTEGRATION AND APPLICATION PERFORMANCE

The developed compression and channel coding components have been integrated with a 3D reconstruction system and a modular rendering engine to test the end-to-end performance. We developed modules for both the transmission and rendering/composition of the 3D Meshes in the scene (see 14). The Rendering engine implements illumination and shading and manages the different objects in the scene. The complete end-to-end streaming system is tested in a lab environment that consists of two computers. The first, with an intel i7 2.8 Ghz CPU and 8.00 GB of RAM plus Microsoft Kinect depth camera and a second with an intel i7 3.4 Ghz CPU with 16.00 GB of RAM acting as receiver. At the receiver side a network emulator [28] is set up that introduces randomly (normally) distributed latencies and packet loss on the incoming packets (in the operating system kernel). We deployed a monitoring system integrated in the software that measures the synchronized timestamps in each step of the transmission pipeline. We monitor each event (frame captured time, frame compressed time etc.). This way we assess the real-time streaming performance of the pipeline. We matched the simulation parameters to the impairment levels presented in ITU G.1050E [29] which are values widely considered by the industry. VIII shows the parameters for well-managed IP connections (class A based as based on leased line) in the first and second column. In the third and fourth column we show established characteristics for partially managed networks (Class B) (that provides some QoS for example based on seperate qeueus like DiffServ but where data flows over shared paths). For our evaluation, our attention focusses on latency and jitter values as VIII that in case of TCP and large frame sizes cause large delays. We measured the end-to-end frame delay from the capture timestamp to the timestamp when the mesh is remotely rendered. The frame capture/reconstruction time varies from around 30 milliseconds to a maximum 50 milliseconds per frame (this depends

Fig. 14. Live reconstructed mesh rendered remotely

TABLE VIII
ITU-T G 1050E INDUSTRY ACCEPTED IMPAIRMENT LEVELS FOR WELL MANAGED (CLASS A) IP NETWORK AND PARTIALLY MANAGED (CLASS B) [29]

delay	Regional (A)	IC (A)	Regional (B)	IC (B)
latency(ms)	20-100	90-300 ms	20-100	90-400
jitter(ms)	0-50	0-50	0-150	0-500
packet losses (p)	0-0.05p	0-0.05p	0 to 2p	0 to 2p

TABLE IX
EXPLANATION OF TIME INTERVAL MEASUREMENTS OF THE STREAMING PERFORMANCE

compression time	time to compress the frame
packetization time	time to generate coded packets
network ready wait	time until network interface is available
network time	time writing UDP datagrams (i.e. sendto calls)
data reconstruction time	time to receive and progressively decode data
mesh decoding time	time for source decoding the mesh
mesh render time	time spend before rendering the mesh

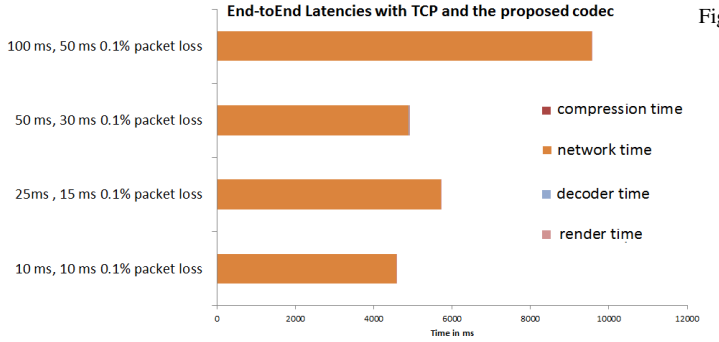


Fig. 15. end-to-end delay with tfan and tcp

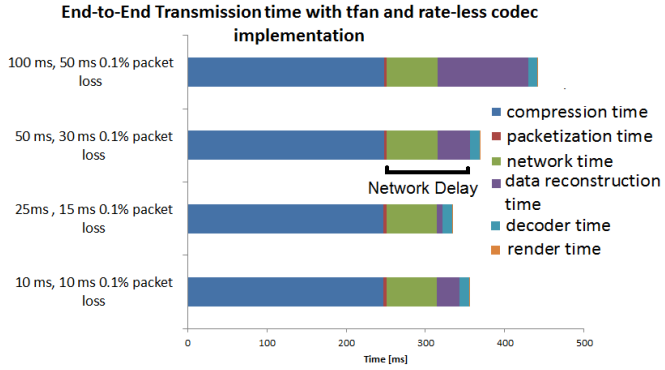


Fig. 16. Average end-to-end delay with Rate-less coding and MPEG TFAN-10, encoder is bottleneck

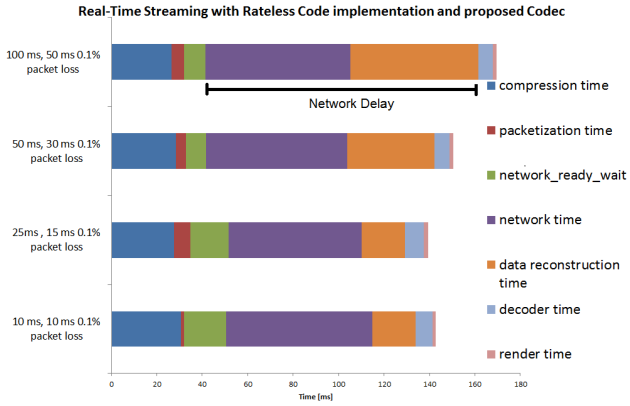


Fig. 17. Average End to End delays: emulated network delays $N(\mu, \sigma)$.

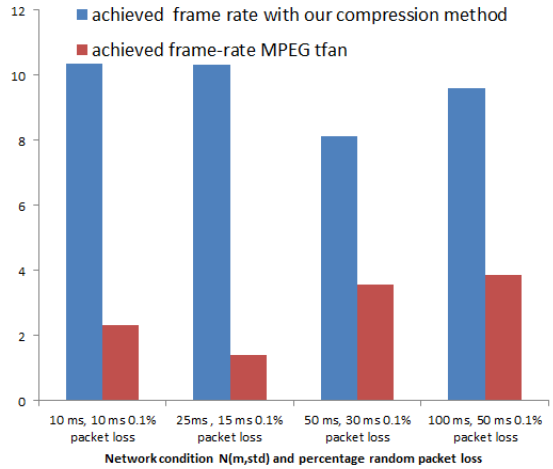


Fig. 18. Resulting Frame-rate at the receiver when the sender is capturing at a target rate of 15 in various network conditions, our method gives upto 5x higher frame rates

on the number of vertices captured). The reconstruction and compression introduces bitrates as defined in VI. As the rate-less coding and UDP transmission (outgoing data rate-control) each run in their own thread, it may not keep up with the rate introduced by the capture/compression threads. In such cases a frame will be skipped in the inter-thread exchange of frames. Only the newest frame is exchanged in each stage. This avoids delay to buildup in the pipeline that can happen for example when a TCP connection blocks or a Sendto operation blocks. This policy is implemented in both the sender and the receiver modules to minimize the user level end-end delay and avoid a buildup of delay in the pipeline.

In Table IX we present the time intervals that we assess. The overall time packets of a frame spend in the network is network time (the time the sender uses to send out packets (sender thread)) *plus* data reconstructed, which is the time the receiver incrementally reconstructs the data each time a packet is received (progressive decoding). The other time intervals are all computational latencies and depend on the

system configuration. We have synchronized virtual clocks between the machines operating the application to monitor the time intervals. 15 shows how TCP handles the large mesh frames (without rateless encoding) in the represented network conditions, almost all time (4-10s) is spent waiting on receiving the frame data and as a result, very low throughput and frame-rate is achieved. This makes TCP unsuitable for real-time transmission of large media frames that require low delay in our tested configuration based on realistic conditions. Fig. 16 shows that the end-to-end delay with the rate-less channel coder with progressive decoding. The information rate is set to 75 percent (75 percent information with 25 percent redundancy, i.e. 33 percent overhead) and the source encoding is based on MPEG TFAN in 16. In this case over 50 percent of the total end-to end latency is caused by the TFAN encoder which has become the performance bottleneck in the system. Fig. 17 shows the delay performance of the optimized transmission system integrated with both our proposed compression method and the progressive rate-less codec. In this case no longer the compression is the bottleneck. Instead the outgoing UDP socket system calls (network time) are the bottleneck and frames spend network ready wait milliseconds before they start to actually be send to the receiver (13,3 ms on average in the 4 tested network conditions). In this case the end-to-end delay still stays below 300 ms for the large majority of the frames frames. The frame rates achieved at the remote site are much better in this configuration. Compared to TFAN as shown in Fig. 17, we achieve frame rates of approximatly 10 fps on average in the 10ms and 25 ms scenarios and between 8 and 9 fps in the 50 and 100 ms scenarios. On the other hand, the current MPEG implementation causes low frame-rates at the receiver site of around 3fps.

VIII. DISCUSSION AND CONCLUSION

This paper presented a prototype implementation that enables 3D Based conferencing between remote participants, focusing on the compression and the transmission components. The main motivation for this work is that we believe that real-time streaming of live reconstructed geometry will enable novel applications that can integrate real and virtual worlds. The prototype addressed some of the significant challenges that triangle mesh representation poses to the media streaming pipeline in terms of latency, data-volume and robustness to losses. The contributions of this research can be summarized as follows:

Encoding/Decoding: Triangle mesh codecs have not been designed with the interactive scenario in mind. Therefore encoding complexity is often too large and low-bitrate/time varying applications and arbitrary geometries have not been considered appropriately. The proposed codec is low complexity, and compared to the current TFAN implementation, it achieves comparable (only slightly lower) quality/rate but much better compared to MPEG SVA that originally targeted this application. Also, the relatively simple operations of this method allow fast implementations. In this context, the MPEG-3DG (3D Graphics group) is currently exploring possibilities to extend its mesh coding standards to support live reconstructed geometry for 3D Tele-Immersion. Our proposed method achieves a frame rates 3 times better in the developed framework. In future work we will explore scalable mesh coding and resolution reduction to enable better network, rendering and application context awareness.

Streaming: systems that efficiently transmit geometry in real-time, generally dealt with stored objects instead of captured reconstructions. With our rate-less code we achieve good loss-resilience and latency performance and adaptability to changing network conditions. As our rate-less code is quite in line with state of art network coding solutions, an exploration of 3D Tele-Immersive streams via network coding is envisioned with the developed framework. In future work, we will investigate adaptive streaming of live reconstructed geometry in networks with congestion and bandwidth fluctuation by controlling the rate-less code and reducing the mesh resolution to meet frame-rate and delay requirements.

3D Tele-Immersion: this paper presents a prototype of a triangle mesh based 3D tele-immersion system. This prototype is integrated with state of the art capturing and rendering components. It is enables a mix full local 3D reconstructions in real-time with remote synthetic and computer controlled content. High quality rendering techniques have been integrated. In the future we are planning more objective and subjective evaluation in sample use cases in entertainment and education.

ACKNOWLEDGEMENT

We would like to thank Dimitrios Alexiadis and Petros Daras at the CERTH for providing datasets and the reconstruction software. We thank Tamy Boubekeur, Steven Poulakos and Manuel Lang for providing the Rendering framework. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. ICT-2011-7-287723 (REVERIE project). We thank the people at MPEG and Institut Telecom for making their codecs available.

REFERENCES

[1] R. Bajcy , L. Wymore,G. Kurillo , K. Mezur ,R. Sheppard , Z. Yang , and W. Wu., K. Nahrstedt, *"Symbiosis of tele-immersive environments with creative choreography,"* in ACM Workshop on Supporting Creative Arts Beyond Dissemination (in conjunction with CCC'07), 2007.

[2] Forte G. Kurillo., M., *"Experimenting with Tele-immersive Archeology,"*, in 16th international conference on Virtual Systems and Multimedia (VSMM 2010) pp. 155-162, 2010.

[3] R. Bajcy ,O. Kreylos ,M. Rodriguez., G. Kurillo, *"Tele-immersive environment for remote medical collaboration,"* in Medicine meets virtual reality (MMVR17) pp. 148-150, 2009.

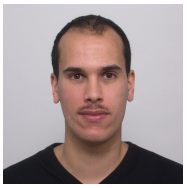
[4] W. Wu, Arefin, A., Z. Huang, P. Agarwal, S. Shi, Rivas, R. K. Nahrstedt, *"I'm the Jedi!" - A Case Study of User Experience in 3D Tele-immersive Gaming,"* Multimedia (ISM), 2010 IEEE International Symposium on , vol., no., pp.220,227, 13-15 Dec. 2010 doi: 10.1109/ISM.2010.39

[5] R.Mekuria, M.Sanna, S.Asioli, E.Izquierdo, D.C.A. Bulterman, P.S.Cesar *"3D tele-immersion system based on live captured mesh geometry,"* in Proceedings of the 4th ACM Multimedia Systems Conference (MMSys '13). ACM, Oslo, Norway, 2013, pp. 24-35.

[6] R. Bajcsy. G. Kurillo, *"3D teleimmersion for collaboration and interaction,"* Springer Virtual Reality, vol. 17, pp. 29-43, Jan. 2013.

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

[8] R. Vasudevan, K. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos., R. Baycsy, K. Nahrstedt *"High Quality Visualization for Geographically Disctributed 3-D Tele-Immersive Applications,"* IEEE Transactions on Multimedia, vol. 13, no. 3, pp. 573-584, 2011.
[9] W. Wu, A. Arefin, G. Kurillo , P. Argawal , K. Nahrstedt , R. Bajcy , *"Color-plus-Depth Level of Detail in 3D Tele-Immersive Video: a psychophysical approach,"* in 19th International conference on Multimedia (MM'11) pp. 13-22, 2011.
[10] K.M. Patel, H.Fuchs. S.U. Kum, *"Real-Time Compression for dynamic 3D Environments,"* in ACM Multimedia'03, Berkeley, CA, USA, 2003.
[11] Z. Yang, Yi Cui, Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R. CampBell, W. Yurcik , *"Real-Time 3D Video Compression for 3D Tele-Immersive Environments,"* in MMCN'06, 2006.
[12] J. Peng C.S. Kim , J. Kuo , *"Technologies for 3D Mesh Compression: A survey,"* Elsevier Journal of Visual Communication and Image processing, pp. 688-733, 2005.
[13] K. Mammou, *"Compression de maillages 3D statistiques et dynamiques,"* l'Universite Rene Descartes - Paris V, Paris, PhD Thesis 2008.
[14] G. Al-Regib Y. Altunbasak, *"3TP: An application layer protocol for streaming 3D Graphics,"* IEEE Transactions on Multimedia, vol. 7, no. 6, pp. 1149-1156, 2005.
[15] M. Li, and B. Prabakaran H. Li, *"Middleware for streaming 3D progressive meshes over lossy networks,"* ACM Transactions on Multimedia Computing and Communications Applications (TOMMCAPP), vol. 2, no. 4, pp. 282-317, 2006.
[16] W. Cheng, W. Tsang Ooi, S. Mondet, R. Grigoras, and G. Morin. 2011. *Modeling progressive mesh streaming: Does data dependency matter?*. ACM Trans. Multimedia Comput. Commun. Appl. 7, 2,
[17] G.Turk and M. Levoy. 1994. *"Zippered polygon meshes from range images"*. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH '94). ACM, New York, NY, USA, 311-318.
[18] E.S. Jang. S. Lee,B. Koo, D. Kim,K. Son , *"Fast 3D Mesh Compression using Shared Vertex Analysis,"* ETRI Journal vol. 32, no. 1, 2010.
[19] M. Gailly J.L. Adler. (2013, Oct.) *zlib, A Massively Spiffy Yet Delicately Unobtrusive Compression Library*. [Online]. http://www.zlib.net/
[20] D. Alexiadis. (2013, Oct.) *Datasets of Multiple Kinects-based 3D reconstructed meshes.* [Online]. http://vcl.iti.gr/reconstructions/
[21] N. Aspert, D. Santa-Cruz, T. Ebrahimi, *"MESH, Measuring Error between Surfaces using the Haussdorff distance,"* in IEEE International conference on Media and Expo, 2002
[22] Institut Telecom. (2013) mymultimediaworld.com. [Online]. mymultimediaworld.com
[23] B. Jovanova, M. Preda, F.Preteux, *"MPEG-4 Part 25: A graphics compression framework for XML-based scene graph formats,"* Signal Processing: Image Communication, vol. 24, no. 1/2, p. 101/114, January 2009.
[24] P.A. Chou, Wu Y., and K. Jain, *"Practical Network Coding,"* in Allerton Conference in Communications and Computing, Monticello, IL, 2003.
[25] C.R. Horn , R. A Johnson, analysis, *Topics in matrix Analysis.* Cambridge: Cambridge University Press., 1991.
[26] K. Greenan., E. L. Miller., J. S. Plank., *"Screaming Fast Galois Field Arithmetic Using Intel SIMD Extensions"* in Proceedings of the 11th Conference on File and Storage Systems "(FAST 2013), San Jose, CA, January 2013.
[27] E. L. Miller, W. B. Houston, J. S. Plank. (2013., January) *GF-Complete: A Comprehensive Open Source Library for Galois Field Arithmetic.* Version 0.1, University of Tennessee.
[28] Microsoft Asia, *NEWT Network Emulator for Windows*, 2013, Microsoft Software.
[29] ITU-T, *"Network model for evaluating multimedia ITU-T G.1050,"* ITU-T, Geneva, G-Series System Recommendation G.1050, 2011.
[30] M. Luby, *"LT Codes"* in IEEE Symposium of Computer Science pp. 271-280, 2002.
[31] A. Shokralli, M. Watson, T. Stockhammer M. Luby, *"RFC 5053 Raptor Forward Error Correction scheme for object Delivery,"* 2007.



Rufael Mekuria received the B.Sc in Electrical Engineering and the M.Sc. in Electrical Engineering from Delft University of Technology, Delft, The Netherlands, in 2007 and 2011 respectively. In September 2011 he joined the Centrum Wiskunde & Informatica: CWI where he is a PhD Student. He is active in international standardization activities in ISO IEC Motion Picture Experts Group (MPEG) since april 2013.

Michele Sanna received the B.Sc. in Electronic Engineering and the M.Sc. in Telecommunications Engineering, both with honors (summa cum laude), from the University of Cagliari, Italy, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree in Electronic Engineering at the Queen Mary, University of London, His interest include media transmission and coding, including network coding and peer-to-peer networks. He is student member of IEEE since 2012.

Ebroul Izquierdo PhD, MSc, CEng, FIET, SMIEEE, MBMVA, is Chair of Multimedia and Computer Vision and head of the Multimedia and Vision Group in the school of Electronic Engineering and Computer Science at Queen Mary, University of London. For his thesis on the numerical approximation of algebraic-differential equations, he received the Dr. Rerum Naturalium (PhD) from the Humboldt University, Berlin, Germany. He has been a senior researcher at the Heinrich-Hertz Institute, Berlin, Germany, and the Department of Electronic Systems Engineering of the University of Essex. Prof. Izquierdo has published over 400 technical papers including chapters in books and patents.

Dick Bulterman Dr. Bulterman received his Ph.D. in computer science from Brown University in Providence RI (USA) in 1981. He has been co-chair of the W3C working group on synchronized multimedia since 2007; this group released the SMIL 3.0 Recommendation in late 2008. Since October 2013 Dick Bulterman is Chief Operating Officer (COO) of FX Palo Alto Laboratory, Inc. (FXPAL). Before he led the dis lab at CWI.

Pablo Cesar leads the Distributed and Interactive Systems group at Centrum Wiskunde & Informatica:CWI. He received his PhD from the Helsinki University of Technology in 2006. He has (co)authored over 70 articles (conference papers and journal articles) about multimedia systems and infrastructures, social media sharing, interactive media, multimedia content modeling, and user interaction. He is involved in standardization activities (e.g., W3C, MPEG, ITU) and has been active in a number of European projects. He is co editor of the book Social Interactive Television: Immersive Shared Experiences and Perspectives and has given tutorials about multimedia systems in prestigious conferences such as ACM Multimedia and the WWW conference.