Enabling Geometry-Based 3-D Tele-Immersion With Fast Mesh Compression and Linear Rateless Coding

Rufael Mekuria, Michele Sanna, *Student Member, IEEE*, Ebroul Izquierdo, Dick C. A. Bulterman, and Pablo Cesar

***Abstract*—3-D tele-immersion (3DTI) enables participants in remote locations to share, in real time, an activity. It offers users interactive and immersive experiences, but it challenges current media-streaming solutions. Work in the past has mainly focused on the efficient delivery of image-based 3-D videos and on realistic rendering and reconstruction of geometry-based 3-D objects. The contribution of this paper is a real-time streaming component for 3DTI with dynamic reconstructed geometry. This component includes both a novel fast compression method and a rateless 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 speed-up up to ten times at comparable compression ratio and quality, when compared with the high-end MPEG-4 SC3DMC mesh encoders. The implemented rateless code ensures complete packet loss protection of the triangle mesh object and a delivery delay within interactive 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 with transmission over TCP in packet loss rates and latencies, typical in managed WAN and MAN networks, and heavily outperforms it in terms of end-to-end delay. The streaming component has been integrated into a larger 3DTI environment that includes state of the art 3-D 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. Compared with alternative methods, lower interactive end-to-end delay and frame rates over three times higher are achieved.**

***Index Terms*—Block codes, geometry processing, multimedia communication, multimedia systems, source coding, 3-D mesh streaming, 3-D tele-immersion.**

I. Introduction

ADVANCES in 3-D reconstruction and the success of inexpensive consumer grade depth cameras enable real-time acquisition of realistic 3-D triangle mesh representations of participants (see Fig. 1). Efficient real-time transmission of these representations opens up possibilities for 3D tele-immersion (3DTI) mixing real and virtual reality. In 3DTI, a user transmits complete and realistic 3-D representations to a remote site with other users. The wide potential of 3DTI has been studied in areas as diverse as creative dancing, cyber-archeology, medicine, and gaming [1]–[4]. However, 3DTI is still challenging and requires large amounts of network bandwidth and computational resources. 3DTI with live reconstructed mesh geometry is particularly challenging, as existing video codecs and streaming solutions do not support this 3-D representation well. This paper takes a step in the direction of efficient 3DTI with live reconstructed geometry. We introduce the specific streaming requirements and present our prototype for real-time compression and transmission of live reconstructed geometry. We evaluate the performance in a fully integrated prototype with real-time rendering with both online reconstruction with one depth camera and with offline reconstructed data from five consumer grade depth-camera streams (see Fig. 1). This paper is an extension of our previous solution in [5]. We extend this configuration with a novel connectivity driven mesh codec. Also, we have made the rateless packet coding progressive (i.e. allowing online per packet decoding) and we performed a more extensive performance evaluation of the complete integrated media pipeline in different, more representative network conditions. When transmitting photo-realistic reconstructed dynamic 3-D mesh geometry, results show that our solution outperforms existing mechanisms in terms of frame-rate and end-to-end delay. Looking ahead in the future, when challenges regarding calibration and synchronization of the different depth cameras are solved, 3DTI based on reconstructed geometry

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

Manuscript received November 04, 2013; revised April 08, 2014; accepted June 04, 2014. Date of publication June 18, 2014; date of current version October 13, 2014. This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under the REVERIE project, Grant ICT- 2011-7-287723. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Klara Nahrstedt.

R. Mekuria and P. Cesar are with the Distributed and Interactive Systems Department, Centrum Wiskunde Informatica, Amsterdam 1098 XG, The Netherlands (e-mail: rufael.mekuria@cwi.nl; P.S.Cesar@cwi.nl).

M. Sanna and E. Izquierdo are with the Department of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, U.K. (e-mail: michele.sanna@eecs.qmul.ac.uk; izquierdo@eecs.qmul.ac.uk).

D. C. A. Bulterman is with the Department of Computer Science, Vrije Universiteit, Amsterdam 1081 HV, The Netherlands (e-mail: Dick.Bulterman@few.vu.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2014.2331919

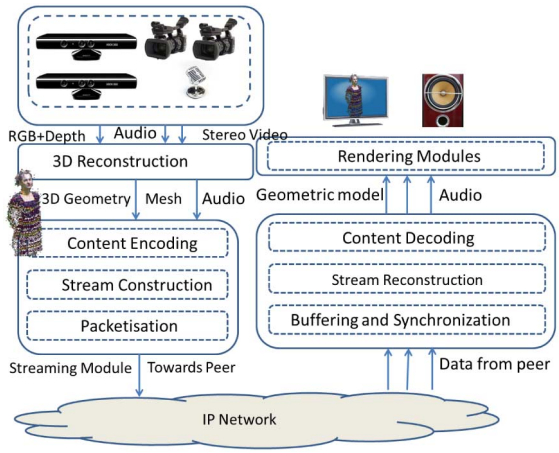


Fig. 2. Pipeline for 3DTI with live-reconstructed geometry, instead of live captured video; a 3-D mesh is reconstructed for visual communication.

can become integrated into online gaming and shared social network experiences. For large-scale consumer level adoption in existing networks, real-time compression and transmission for live reconstructed dynamic geometry will be key enabling technologies.

A. 3-D Representation

The 3-D representation is a 3-D mesh, typically reconstructed in real time from multiple depth cameras as for example presented in [7]. In Eq. (1)–(3) we define the sequence of meshes introduced by the reconstruction system. The individual vertices v in $\mathbf{V}^i$, contain 3-D 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 $i = 1, \ldots, K$ contains a set of vertices $\mathbf{V}^i$ and faces $\mathbf{F}^i$ that index vertices and represent a participant by a 3-D surface. Note that for the realistic meshes resulting from the reconstruction system [7], not only the geometry information, but also the connectivity and number of vertices typically changes each frame i.e. $\mathbf{F}^i \neq \mathbf{F}^j$ and $|\mathbf{V}^i| \neq |\mathbf{V}^j|$ for $j \neq i$, this is often referred to as inconsistent time-varying mesh geometry.

$$\mathbf{M}^i = (\mathbf{V}^i, \mathbf{F}^i), \qquad i = 1 \ldots K, \tag{1}$$

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

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

B. Media Pipeline

Fig. 2 shows the 3-D tele-immersive media pipeline based on live reconstructed geometry. In this configuration, first multiple color plus depth streams are acquired from multiple depth cameras. Based on these color plus depth streams, a 3-D reconstruction module computes the surface geometry as a 3-D mesh. The 3-D reconstruction module achieves this via an optimized implementation of a known 3-D surface reconstruction method based on multiple depth images like presented in [7]. The result from the reconstruction module is a sequence of 3-D meshes as in Eq. (1). This sequence is subsequently compressed, packetized and transmitted to a remote host over an IP network. The receiver renders the mesh in the scene, possibly in combination with other reconstructed participants and synthetic content. Contrary to transmission and compression of depth images, in this configuration the complete surface geometry is compressed and transmitted directly.

C. Research Questions and Objectives

Real-Time streaming of live-captured video is common in video conferencing systems, but streaming of live reconstructed mesh geometry sequences has rarely been considered. There are various reasons why we consider efficient real-time transmission of such representations essential. First, modern graphics cards can take advantage of advanced rendering methods, i.e. stereoscopic, autostereoscopic and free-viewpoint rendering. Second, it enables integration with virtual worlds, where triangle mesh representations are common. Geometry streaming solutions can also be beneficial for applications like camera or terrain surveillance, where live captured geometric data needs to be available in real-time for observers or geographic information systems. In this paper we are concerned with the 3DTI application between participants. 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 3DTI?* 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 3-D triangle mesh representation: the engine should support streaming of the 3-D mesh representation. That is, a set of vertices with properties (coordinates, normals colors) and a set of faces indexing these vertices, resulting into a surface in the 3-D space. Low end-to-end frame delay is considered important in 3-D 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. It should be possible to reconstruct the triangle mesh at the receiver in case of packet loss. Generally, the dependency between connectivity and geometry data in compressed mesh geometry makes it 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 in practical mesh compression such as the MPEG-4 SC3DMC standard. 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 object that is captured should be streamed.

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

This also means that no pre-stored avatars or models can be transmitted as placeholders.

Real-time live-captured triangle meshes should be supported. Contrary to live captured video, where frames generally consist of a fixed number of pixels across frames, we are dealing with *inconsistent time-varying geometry*. Compared to time-consistent geometry or video, it is much harder to exploit temporal correlation, as no direct relation between pixels/vertices via the rectangular sample grid or fixed connectivity exists.

D. Contribution and Outline

This paper presents a component that can stream live reconstructed triangle mesh geometry in real-time. The component includes both real-time compression and transmission. We achieve a compression rate near the state of art TFAN MPEG encoder [13] at comparable distortion, but with a near 10 times encoding speed up. This meets our requirements on latency and bandwidth. Second, the transmission scheme based on rateless coding meets the requirements of robustness, latency and adaptability to changing network conditions. This paper is structured as follows. Section II overviews the related work regarding existing 3DTI systems, mesh compression, and geometry streaming solutions. Section III presents the compression method. In Section IV we evaluate this method, introducing the datasets, quality criteria and the compression results. Section V presents the transmission method and its evaluation. Section VI then presents our 3DTI system, highlighting integration with rendering and reconstruction and the performance of the overall system. We discuss the conclusions and implications of our work in Section VII. In Table I we provide some of the common abbreviations used throughout the paper.

II. RELATED WORK

A. 3DTI

3D Tele-immersion has received considerable attention; we highlight some of the current advances. Vasudevan *et al.* proposed a 3DTI system for capturing and rendering based on multiple meshed depth images. An advantage of this technique is that by interpolating vertices and by changing the size of the mesh faces 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 3-D space. An overview of technical challenges and related user experiences based on over a decade of experimental 3DTI 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 from a single stream.

B. Geometry Compression

A considerable body of work on geometry compression exists and a survey is available [11]. 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 3-D meshes can be manifold or not, open, closed regular or irregular, oriented or not. Also, manageable computational complexity is needed for real-time operation. Therefore, the MPEG standardization has adopted the MPEG SC3DMC standard that can compress irregular, non-manifold meshes with very fast decoding times. Evaluations show that a compression gain of 5% compared to state of art is achieved on manifold datasets and even more on large industrial datasets that includes disconnected (isolated) components [12]. 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 [12]. In the case of 3DTI 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 3DTI.

C. Geometry Streaming

Various solutions have been proposed for efficient progressive download and streaming of compressed mesh geometry from a server. Methods such as 3TP [14] and comparable solutions such as [15] generally utilize an extensive offline optimization scheme based on minimizing a geometric distortion metric to the original geometry to decide on an optimal transmission scheme. This is not possible in 3DTI where the original geometry is reconstructed and therefore not available offline *a priori*. In a recent work [16] connection oriented transmission is seen as the most desirable for transporting geometric data, optimizing the scheduling of vertex split packets anticipating retransmissions. In this paper we propose compressed mesh transmission with a linear rateless code optimized for real-time 3D Tele-immersion and compare it with connection oriented transmission via TCP.

III. FAST COMPRESSION METHOD FOR LIVE-RECONSTRUCTED GEOMETRY

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

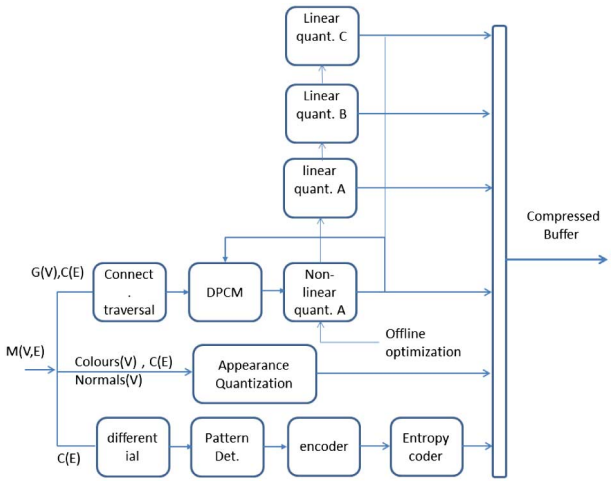


Fig. 3. Outline of geometry compression scheme.

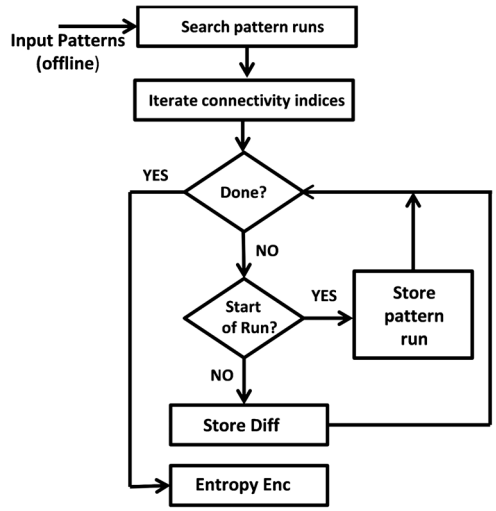


Fig. 4. Outline of connectivity compression scheme.

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 3DTI. Fig. 3 outlines the compression system. We introduce a fast connectivity traversal method combined with connectivity-driven DPCM coding and non-linear layered quantization. We explain the rationale and operations for compressing the connectivity $\mathbf{C}(\mathbf{E})$ in the next sub-section. The differential encoding of the geometry $\mathbf{G}(\mathbf{V})$ followed by non-linear quantization is presented in III-B. In Section III-C we discuss how the appearance quantization is handled.

A. Pattern-Based Connectivity Coding

The idea behind the connectivity coding approach presented in this paper is that 3-D 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 allowing efficient entropy coding. The resulting data vector is entropy encoded via the zlib library [19].

TABLE II
DATASTRUCTURE PATTERN RUN

Mode	Diff1	Diff2	Start	Count

TABLE III
EXAMPLE OF A PATTERN IN THE CONNECTIVITY LIST

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

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 3-D geometry this has several disadvantages. Firstly, the dynamic range of the vertices that are quantized in a 3-D space is large compared to the details in the 3-D surface. This results in quantization artifacts that quickly become visible to the viewer. The surface is vulnerable to quantization distortion that increases with lower ($QP < 8$) quantization parameters. When a low quantization parameter ($QP < 8$) is used for geometry compression, decoded vertices may share positions resulting in degenerate triangles (i.e. faces with a zero surface) and a blocky appearance. Therefore, instead we propose quantization after the DPCM transform 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, which is important, as large differential quantization errors can

also quickly become 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 in the densely sampled 3-D reconstruction, connected vertices are co-located. The primary quantization vector codes over 95% 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 the 4 bit non-linear quantization vector was computed offline. Since this vector can also be dependent on the reconstruction method, we envision that such information can be exchanged out-of band (in our 3DTI 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 16 bit 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. Also, we allow re-scaling of this quantization vectors to meshes with different coordinate ranges in their 3 axes by computing a bounding box over the differentials. The lower the threshold *thresh* of the 4-bit 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 change the order of the vertices in the list. Each of the faces is traversed, and the first connected vertex that was previously stored 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% 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 bit vector, values between $[-0.1, 0.1]$ are quantized with 8 bits and appended to the eight bit 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 $[-0.250.25]$ with 4 bits and the rest with eight bits. We code color differences between -25 and 25 with 4 bits and the rest with 8 bits which suffices covering the entire range $[-255, 255]$. Again, the QP information can be communicated out of band prior to the media streaming session to configure the encoder and decoder properly if needed.

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 and provided by the authors of [7]. 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 in different activities. We have also integrated the reconstruction system with our 3-D 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 $d(M, M') \neq d(M', M)$ we use the symmetrical Haussdorf distance which is defined in Eq. (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

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

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

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

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

TABLE IV
QUALITY METRICS NOTATIONS

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
$d_{uniform}(M.M')$	quantization error(rms) uniform source and quantizer

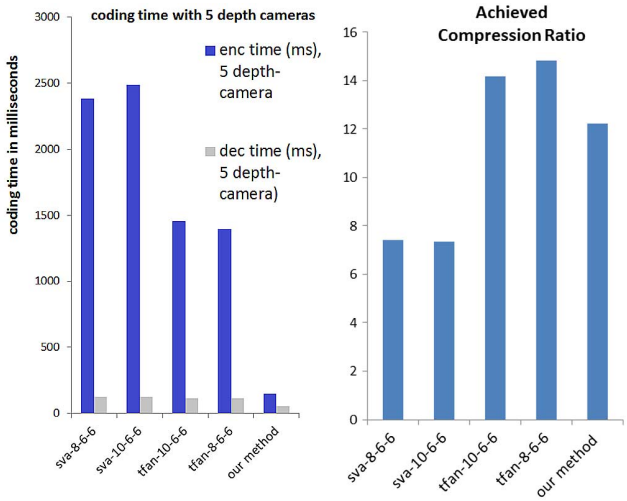


Fig. 5. Results: encoding and decoding with five cameras.

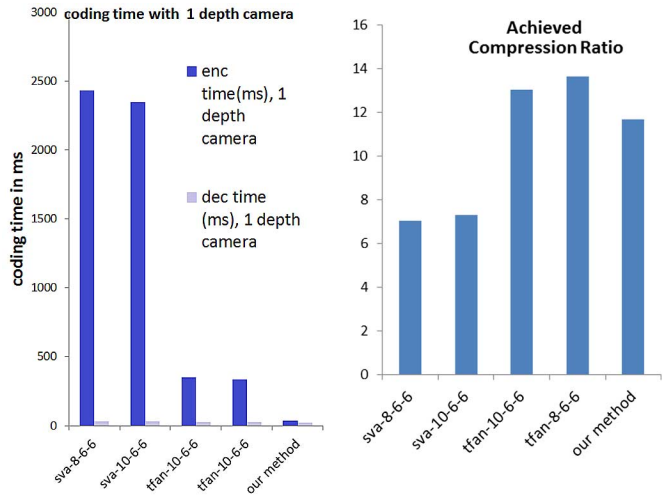


Fig. 6. Results: encoding and decoding with one camera.

TABLE V
ENCODER PARAMETER SETTINGS

tfan-8-6-6	tfan 8 bit coordinate 6 bit normal, 6 bit color
tfan-10-6-6	tfan 10 bit coordinate 6 bit normal, 6 bit color
sva-8-6-6	sva 8 bit 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

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 3-D which is given by

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

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.

C. Experimental Results

Our scheme was implemented in C++ and the test machine was a desktop machine with Intel i7 CPU, 8 GB of RAM and a Geforce GTX 760 video card. The MPEG-4 Codecs were compiled from source code as available at [22] with the MS Visual Studio 2010 compiler. To avoid overhead of 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 loaded directly from the input mesh. This way any overhead delays in accessing the MPEG SC3DMC 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 302 K vertices. Fig. 5 shows that 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 KBytes (ratio 12.3:1) compared to 1266 KiloBytes with TFAN MPEG (14:1) with 10 bit QP and 6 bit colors and normals. While the measured compression ratio is about 15% lower compared to TFAN, but we achieve a speedup of upto 10 times in encoding time. 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 72 K vertices) are consistent. Here we encode meshes in 35 ms on average and the average frame size is reduced to 370 Kb (a 12:1 ratio). In Table V we show the encoder settings used for the quality evaluation. 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 normals 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 bit 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 vertices we compared only with SVA and to $d_{uniform}$ in Fig. 9. 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 3-D reconstruction method and not the compression method. In the future, with better acquisition devices and 3-D reconstruction software,

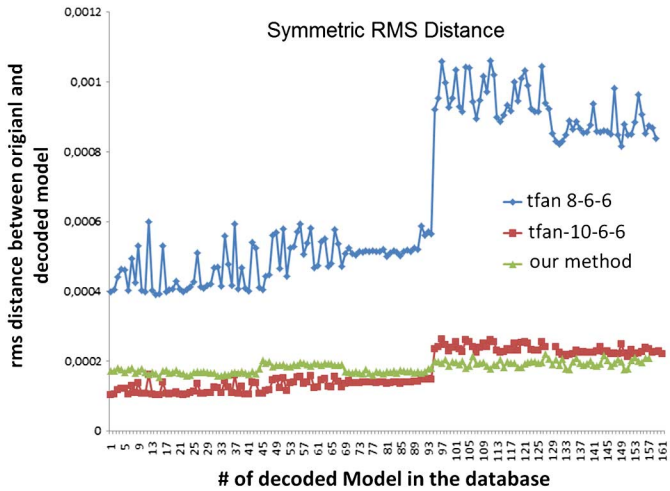


Fig. 7. distortion between original and decoded models: $d_{rmssym}(M, M')$.

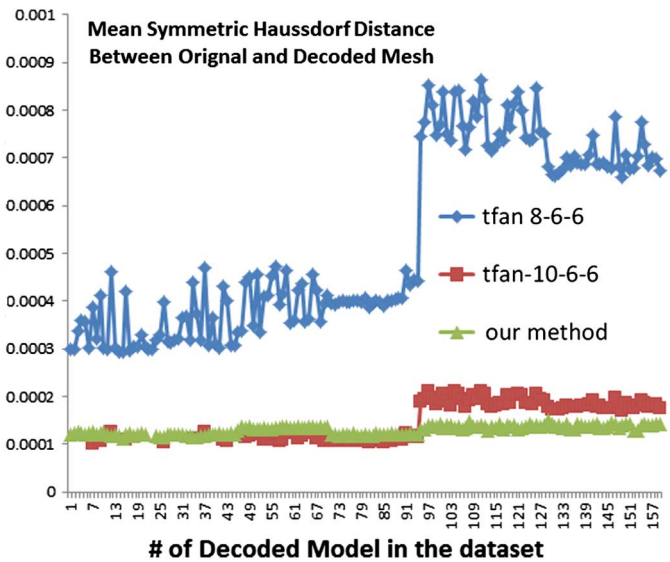


Fig. 8. distortion between original and decoded models: $d_{sym}(M, M')$.

these artifacts will reduce. We show the resulting bandwidth requirements in Table VI.

V. Real-Time Transmission

In this section we introduce the concept of progressive rateless coding. We compare it to resilient transmission via TCP, based on a number of lab experiments. We show its favorable properties for geometry transmission and 3DTI. Symbol-based inter-packet coding, together with progressive decoding based on Gauss-Jordan elimination are used to produce a rateless 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%. In this case, a single frame of around 300Kbyte would result in 300 UDP packets; the chance a frame would arrive $E[framearrives] = E[nolostpacket]$ would be less than 5% in case of a 1% loss rate.

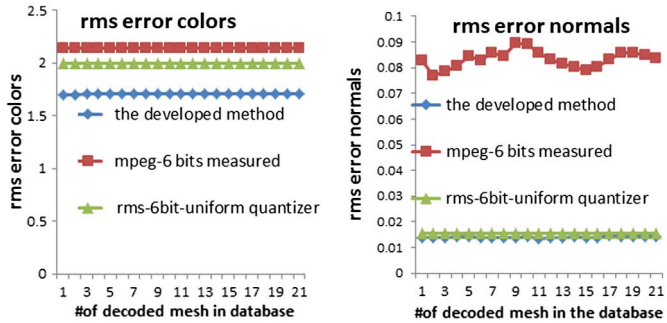


Fig. 9. Quality comparison between colors and normals original and decoded models $d_{app}(M, M')$.

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

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

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 rateless and fountain codes is that any amount of packets can be generated at the sender (i.e. rateless). The first practical random linear codes were first proposed in [24]. An advantage of the rateless 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 rateless 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.

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 K_g blocks and the size of the blocks is chosen to avoid possible Ethernet fragmentation, i.e., 1024 bytes. Linear

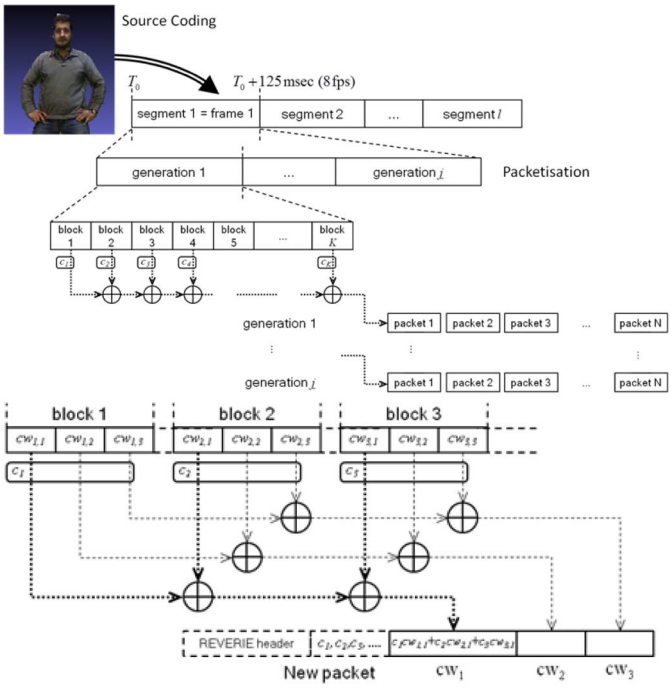


Fig. 11. Top: arrangement of blocks for rateless coding. Bottom: schematic encoding of an outgoing packet.

coding is then performed on a finite fields i.e. 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 W code words:

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

The superscript (k, g) indicates the k-th block of generation g. A field size $\mathrm{q} = 256(\mathrm{m} = 8$ 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 $c_1, c_2, \ldots, c_{K_g}$. A coded block is generated from generation g as:

$$\mathbf{b'}^{(n,g)} = [b'^{(n,g)}_1, \ldots, b'^{(n,g)}_{(W)}], \quad n = 1, \ldots, N_g, g = 1, \ldots, G. \quad (12)$$

Each code word is calculated as:

$$b'^{(n,g)}_j = \sum_{k=1}^{K_g} \left(c_k b_j^{(k,g)} \right), \qquad j = 1, 2, \ldots, W. \quad (13)$$

Therefore, $N_g > K_g$ outgoing blocks are generated. As only K_g linearly independent blocks are needed to decode the original data, around $N_g - K_g$ 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 $K_g x K_g$ 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 an 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 reduces the 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 rateless coding has been restricted to one linear system per generation (as opposed to the 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_g	Number of coded blocks per generation
$GF(q = 2^m)$	Galois Field of size q, and word length m bits
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, $i = 1, \ldots, 2^m - 1$ 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% information rate (or 10 times redundancy for loss protection), and decoded from the first 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 bit 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. (13). Additionally, region multiplication is used for all row-operations needed for Gaussian elimination, reducing the matrix to row echelon form.

D. Experimental Results of Rateless Coding

In order to assess the delay performance of the rateless 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, and decoding the packets. A network emulator [28] was used to emulate latency and packet loss inside the LAN between the two machines. We compare our system performance to performance of a TCP

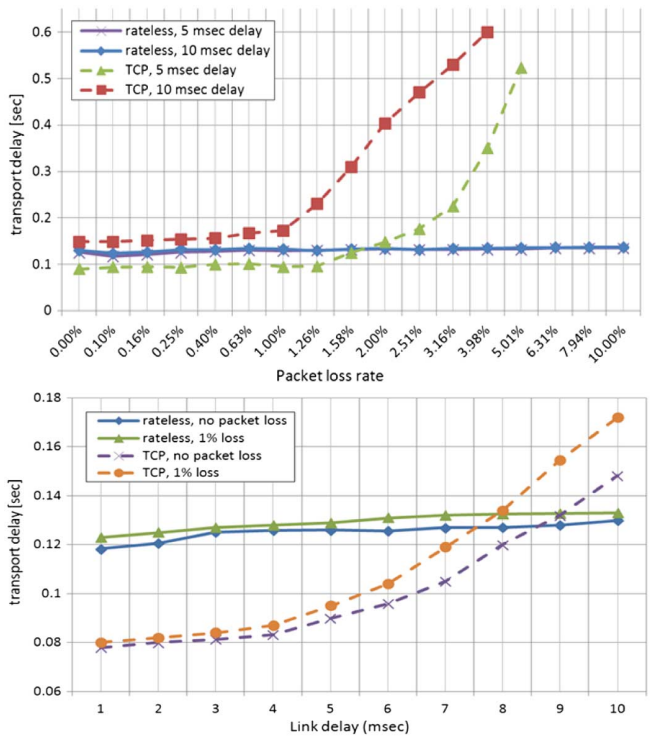


Fig. 12. End-to-end delay of rateless code implementation based on packet loss (top) and latency (bottom) in the network.

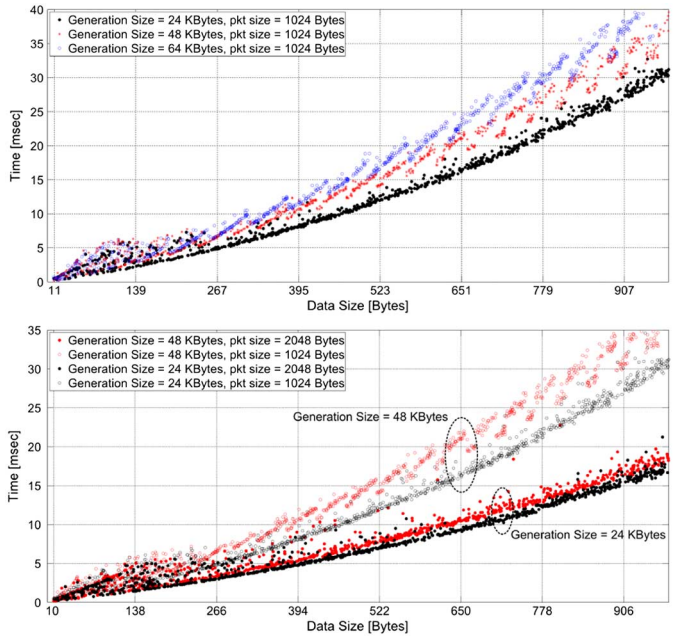


Fig. 13. Frame decoding time with fixed packets sizes. Top: fixed packet size. Bottom: different packet sizes.

Fig. 14. Live reconstructed mesh rendered remotely.

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 1.5% 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 ms, 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 rateless 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 9 ms, and the rateless code is expected to yield a better delay performance. While this type of coding introduces overhead (we set 33% in our case), this can be adapted to match network conditions making appropriate use of available bandwidth.

VI. 3D Tele-Immersion Pipeline Integration and Application Performance

The developed compression and transmission components have been integrated with a 3-D reconstruction system and a modular rendering engine to test the complete end-to-end system performance. We developed two modules: the *acquisition module* for the aquisition, compression and transmission and the *reception module* for packet reception, decoding and real-time rendering (see Fig. 14). The rendering engine provides the illumination and shading techniques and manages the composition into the scene. The end-to-end streaming performance is tested in a lab environment that consists of two computers. The first running the acquisition module, with an Intel i7 2.8 Ghz CPU and 8 GB of RAM plus Microsoft Kinect depth camera. The second, with an Intel i7 3.4 Ghz CPU with 16 GB of RAM, is running the reception module. At the side of the reception module, a network emulator [28] is setup that introduces random network impairments such as delay, jitter and packet loss. To measure overall performance, we deployed a monitoring system integrated in the software that measures synchronized timestamps in each step of the transmission pipeline. We monitor each event (i.e. the time a frame was captured, when a frame was compressed, scheduled, rendered etc.). This way we assess the real-time streaming performance of the entire mesh based 3DTI System pipeline

TABLE VIII
ITU-T G 1050E INDUSTRY ACCEPTED IMPAIRMENT LEVELS IN DIFFERENT TYPES OF NETWORKS [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

from acquisition to rendering. We matched the network impairment introduced by the emulator to the levels presented in ITU G.1050E [29] which are values widely considered in 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) (this is the class of network that provides QoS based on separate router queues only like DiffServ). For our evaluation, our attention focusses on latency and jitter values as in Table VIII that in case of TCP and large frame sizes cause large delays. We measured the end-to-end frame delay from the capture instant to the instant when the mesh is remotely rendered. The frame capture/reconstruction time varies from around 30 milliseconds to a maximum 50 milliseconds per frame depending on the number of vertices captured. The reconstruction and compression introduces bitrates as in Table VI. As the rateless 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 the transmission component cannot keep up with the incoming data rate. This policy is implemented in both the acquisition and the reception modules to minimize the user level end-to-end delay. In Table IX we present the time intervals that we assess. The overall network delay caused by the transmission is the *network time* (the time the acquisition module uses to send out packets (sender thread) *plus data reconstruction time*, which is the additional time the reception module uses to incrementally reconstruct the original data from the packets via progressive decoding. The other time intervals are all computational latencies and depend on the system hardware configuration. We have synchronized virtual clocks between the machines operating the application to monitor the time intervals to do the measurements. Fig. 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

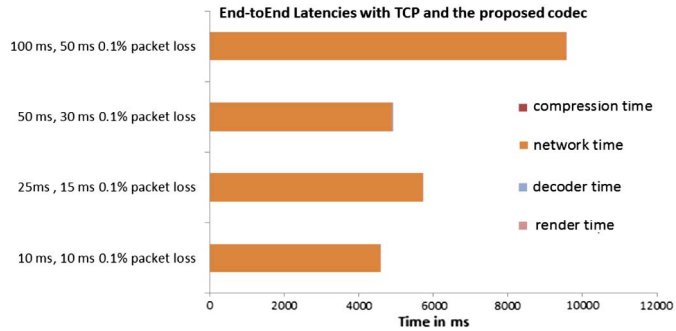


Fig. 15. End-to-end delay with TFAN and TCP.

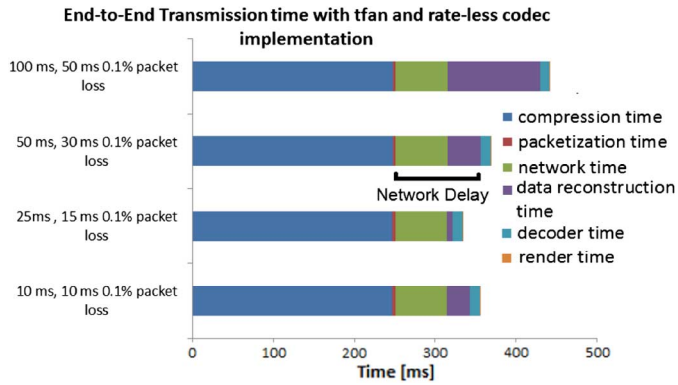


Fig. 16. Average end-to-end delay with Rateless coding and MPEG TFAN-10, encoder is bottleneck.

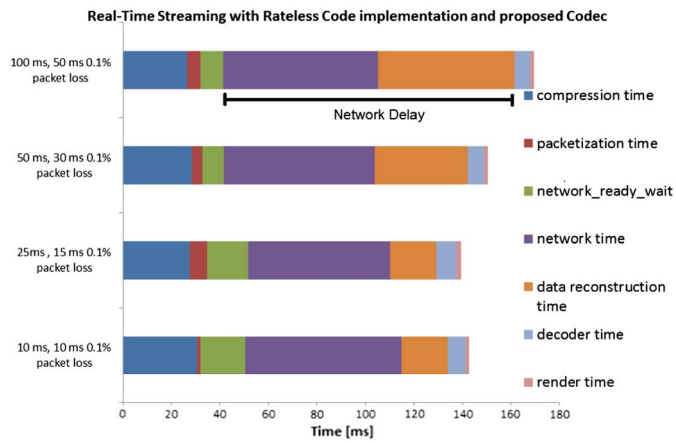


Fig. 17. Average end-to-end delays: emulated network delays $N(\mu, \sigma)$.

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 rateless channel coder with progressive decoding. The information rate is set to 75% (75% information with 25% redundancy, i.e. 33% overhead) and the source encoding is based on MPEG TFAN in Fig. 16. In this case over 50% 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 rateless coding transmission. In this case the compression is no longer the bottleneck. Instead

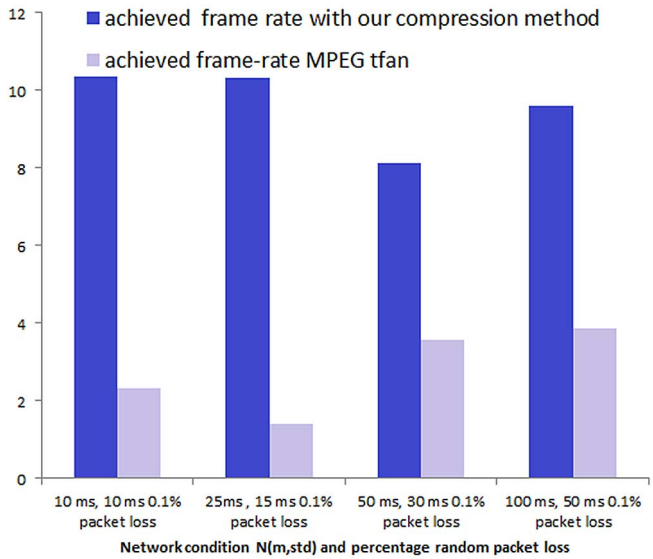


Fig. 18. Resulting Frame-rate at the receiver when the sender is capturing at a target rate of 15 in various network conditions, our method gives up to 4x higher frame rates.

the outgoing UDP socket system calls (i.e. network time) sometimes cannot keep up with the incoming data rate from the compression module introducing an extra network ready wait time which is 13,3 ms on average. In this case the end-to-end delay stays below 300 ms for the large majority of the frames. The frame rates achieved at the remote site are much higher in this configuration. Compared to TFAN as shown in Fig. 18, we achieve frame rates of approximately 10 fps on average in the 10 ms 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 3 fps, therefore, we achieved a 3 times increase in frame rate.

VII. Discussion and Conclusion

This paper presented a prototype implementation that enables 3-D mesh based conferencing between remote participants, focusing on the compression and the transmission components for reconstructed mesh geometry. The main motivation of this work is that real-time streaming of live reconstructed geometry will enable novel applications that can integrate real and virtual worlds and enhance social interactions. 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 always been considered appropriately. The proposed codec has low complexity, and compared to the current TFAN implementation, it achieves comparable (only slightly lower) quality/rate. On the other hand, much better results compared to MPEG SVA that originally targeted this application have been achieved. 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 3DTI. Our proposed method achieves frame rates 3 times higher in the configuration in the 3DTI framework with rendering and recontruction. In future work we will explore scalable mesh coding and resolution reduction to enable better network, rendering and application context awareness.

Streaming: multimedia systems that efficiently transmit geometry in real-time, generally dealt with stored objects instead of streaming captured reconstructions. With our rateless code we achieve good loss-resilience and latency performance and adaptability to changing network conditions. As our rateless 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 rateless code and reducing the mesh resolution to meet frame-rate and delay requirements.

3DTI: 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 enables to mix full local 3-D reconstructions in real-time with remote synthetic and computer controlled content. 3-D rendering techniques have been integrated. This is contrary to most previous 3DTI systems, that only deploy point clouds or systems based on multiple depth images. In the future we are planning more objective and subjective evaluation in sample use cases in entertainment and education. We are also planning integration of spatial audio, enhancing the immersive experience further.

Acknowledgment

The authors would like to thank D. Alexiadis and P. Daras from CERTH, Greece, for providing datasets and the reconstruction software, and T. Boubekeur from Telecom ParisTech and S. Poulakos and M. Lang from Disney research for providing the rendering framework. The authors would like to thank MPEG and Institut Telecom for making their codecs available publicly.

References

[1] R. Bajcy, L. Wymore, G. Kurillo, K. Mezur, R. Sheppard, Z. Yang, W. Wu, and K. Nahrstedt, "Symbiosis of tele-immersive environments with creative choreography," in *Proc. ACM Workshop Support. Creative Arts Beyond Dissemination*, 2007.

[2] G. Forte and M. Kurillo, "Experimenting with tele-immersive archeology," in *Proc. 16th Int. Conf. Virtual Syst. Multimedia*, 2010, pp. 155–162.

[3] R. Bajcy, O. Kreylos, M. Rodriguez, and G. Kurillo, "Tele-immersive environment for remote medical collaboration," in *Proc. Medicine Meets Virtual Reality Conf.*, 2009, pp. 148–150.

[4] W. Wu, A. Arefin, Z. Huang, P. Agarwal, S. Shi, R. Rivas, and K. Nahrstedt, ""I'm the Jedi!" - A case study of user experience in 3d tele-immersive gaming," in *Proc. IEEE Int. Symp. Multimedia*, Dec. 13–15, 2010, vol. , no. , pp. 220–227.

[5] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. S. Cesar, "3D tele-immersion system based on live captured mesh geometry," in *Proc. 4th ACM Multimedia Syst. Conf.*, Oslo, Norway, 2013, pp. 24–35.

[6] R. Bajcsy and 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 Trans. Multimedia*, vol. 15, no. 2, pp. 339–358, Feb. 2013.

[8] R. Vasudevan, K. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Baycsy, and K. Nahrstedt, "High quality visualization for geographically disctributed 3-D tele-immersive applications," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 573–584, Mar. 2011.

[9] W. Wu, A. Arefin, G. Kurillo, P. Argawal, K. Nahrstedt, and R. Bajcy, "Color-plus-depth level of detail in 3D tele-immersive video: A psychophysical approach," in *Proc. 19th Int. Conf. Multimedia*, 2011, pp. 13–22.

[10] K. M. Patel, H. Fuchs, and S. U. Kum, "Real-time compression for dynamic 3D environments," in *Proc. ACM Multimedia*, Berkeley, CA, USA, 2003, pp. 185–194.

[11] J. Peng, C. S. Kim, and J. Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Commun. Image Process.*, vol. 16, no. 6, pp. 688–733, 2005.

[12] K. Mamou, "Compression de maillages 3D statistiques et dynamiques," Ph.D. dissertation, UFR de Mathematique et Informatique, Universite Rene Descartes–Paris V, Paris, France, 2008.

[13] K. Mamou, T. Zaharia, and F. Preteux, "TFAN a low complexity 3D mesh compression algorithm," in *Proc. Comput. Animations Virtual Worlds*, 2009, pp. 343–354.

[14] G. Al-Regib and Y. Altunbasak, "3TP: An application layer protocol for streaming 3D Graphics," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1149–1156, Dec. 2005.

[15] M. Li, B. Prabakaran, and H. Li, "Middleware for streaming 3D progressive meshes over lossy networks," *ACM Trans. Multimedia Comput. Commun. Applic.*, vol. 2, no. 4, pp. 282–317, 2006.

[16] W. Cheng, W. Tsang Ooi, S. Mondet, R. Grigoras, and G. Morin, "Modeling progressive mesh streaming: Does data dependency matter?," *ACM Trans. Multimedia Comput. Commun. Applic.*, vol. 7, no. 2, Mar. 2011.

[17] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proc. 21st Annu. Conf. Comput. Graphics Interactive Tech.*, New York, NY, USA, 1994, pp. 311–318.

[18] E. S. Jang, S. Lee, B. Koo, D. Kim, and K. Son, "Fast 3D mesh compression using shared vertex analysis," *ETRI J.*, vol. 32, no. 1, pp. 163–165, 2010.

[19] M. Gailly and J. L. Adler, "zlib, A Massively Spiffy Yet Delicately Unobtrusive Compression Library," Oct. 2013 [Online]. Available: http://www.zlib.net/

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

[21] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "MESH, measuring error between surfaces using the Haussdorff distance," in *Proc. IEEE Int. Conf. Media Expo*, 2002, pp. 705–708.

[22] Institut Telecom, 2013 [Online]. Available: mymultimediaworld.com

[23] B. Jovanova, M. Preda, and F. Preteux, "MPEG-4 Part 25: A graphics compression framework for XML-based scene graph formats," *Signal Process.: Image Commun.*, vol. 24, no. 1/2, pp. 101–114, Jan. 2009.

[24] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. Allerton Conf. Commun. Comput.*, Monticello, IL, USA, 2003, pp. 40–49.

[25] C. R. Horn and R. A. Johnson, "Analysis," in *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ., 1991.

[26] K. Greenan, E. L. Miller, and J. S. Plank, "Screaming fast Galois field arithmetic using Intel SIMD extensions," in *Proc. 11th Conf. File Storage Syst.*, San Jose, CA, USA, Jan. 2013.

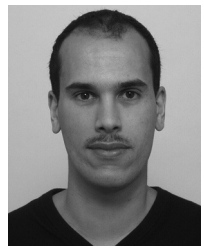
[27] E. L. Miller, W. B. Houston, and J. S. Plank, GF-Complete: A comprehensive open source library for Galois field arithmetic. ver. 0.1, Univ. Tennessee.

[28] NEWT Network Emulator for Windows. Microsoft Asia, 2013.

[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 *Proc. IEEE Symp. Comput. Sci.*, 2002, pp. 271–280.

[31] A. Shokralli, M. Watson, T. Stockhammer, and M. Luby, "RFC 5053 raptor forward error correction scheme for object delivery," 2007.



Rufael Mekuria received the B.Sc. and M.Sc. degrees in electrical engineering from Delft University of Technology, Delft, The Netherlands, in 2007 and 2011, respectively. He is currently working toward the Ph.D. degree at the Centrum Wiskunde & Informatica: CWI, Amsterdam, The Netherlands.

He has been active in international standardization activities in ISO IEC Motion Picture Experts Group (MPEG) since April 2013.

Michele Sanna (S'12) received the B.Sc. degree in electronic engineering and M.Sc. degree in telecommunications engineering, both with honors (*summa cum laude*), from the University of Cagliari, Cagliari, Italy, in 2006 and 2008, respectively. He is currently working toward the Ph.D. degree in electronic engineering at the Queen Mary, University of London, London, U.K.

His interest include media transmission and coding, including network coding and peer-to-peer networks.

Ebroul Izquierdo received the Ph.D. degree from Humboldt University, Berlin, Germany. His dissertation focused on the numerical approximation of algebraic-differential equations.

He 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, London, U.K. He has been a Senior Researcher with the Heinrich-Hertz Institute, Berlin, Germany, and the Department of Electronic Systems Engineering of the University of Essex. He has authored or coauthored over 400 technical papers including chapters in books and patents.

Dick C. A. Bulterman received the Ph.D. degree in computer science from Brown University, 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, he has been Chief Operating Officer of FX Palo Alto Laboratory, Inc. (FXPAL), Palo Alto, CA, USA. Before that, he led the DIS lab at CWI, Amsterdam, The Netherlands.

Pablo Cesar received the Ph.D. degree from the Helsinki University of Technology, Helsinki, Finland, in 2006.

He leads the Distributed and Interactive Systems group at Centrum Wiskunde & Informatica: CWI. He has authored or coauthored 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 a coeditor 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.