A case study for tele-immersion communication applications: from 3D capturing to rendering

Dimitrios Alexiadis[1], Alexandros Doumanoglou[2], Dimitrios Zarpalas[3], Petros Daras[4]
Information Technologies Institute, Centre for Research and Technology - Hellas
6th Km Charilaou-Thermi, Thessaloniki, Greece
[1]dalexiad@iti.gr, [2]aldoum@iti.gr, [3]zarpalas@iti.gr, [4]daras@iti.gr

***Abstract*—The primary objective of this paper is to present and analyze key aspects related to next-generation tele-immersion applications, studying the end-to-end chain from 3D capturing of remote users to rendering. The key modules for 3D reconstruction of moving humans and their mesh compression, are presented and discussed. The chain performance is evaluated in terms of frame-rates, delay, and visual quality.**

***Index Terms*—Tele-immersion, 3D reconstruction, mesh compression, real-time, Free-Viewpoint rendering**

I. Introduction

This work focuses on a future Internet application, called Tele-Immersion (TI) [1] and 3D Tele-Presence, motivated by the impressive advances in 3D capturing, visualization and computer graphics. TI technology aims to enable geographically-distributed users to communicate and interact inside a shared virtual world, as if they were physically there. The substantial factors for maximizing the users' immersion and natural interactiveness is the 3D content quality, as well as the data exchange rates between remote sites. Therefore, fast methodologies for 3D data generation and compression that allow real-time interaction, are of paramount importance. The choice of 3D data representation determines the overall structure and performance of a TI system. The available 3D representations lie between two extremes [2]: Image-based modeling does not use any 3D geometry at all and virtual views are synthesized from natural camera views by interpolation. At the other extreme, geometry-based modeling, a scene is represented in the basis of 3D meshes. In between, other methods such as Video plus Depth (VD) combine image-based modeling with 3D information (i.e. depth maps).

In a state-of-the-art TI system [1] multiple depth-maps are generated in real-time from stereo pairs. Taking advantage of the employed triangulation scheme, only the depth values for the triangles' nodes are transmitted. This is an efficient scheme, but restricted to the specific triangulation rule. At the rendering site, intermediate views for given view-points are synthesized by combining the depth data. A similar TI system is described in [3], where the depth-maps are captured by multiple Kinects. However, the data compression/transmission part is not studied. In another Kinect-based TI approach [4], a 3D mesh is reconstructed at the capture site [5] and compressed in real-time before transmission. However, coding of the triangles (connectivity) information is restricted to a specific triangulation rule. Additionally, in all above approaches, the color information is coded as an RGB value per vertex. This scheme requires the compression and transmission of very dense geometry information to avoid blurring and aliasing in the rendered images. Finally, in [6] an image-based TI scheme is used, where an intermediate view is interpolated at the capture site and compressed using standard video coding.

In this paper, we study a 3D geometry representation-based TI pipeline, arguing that geometry-based representations would be preferable as opposed to image-based ones, for the following reasons: a) Usually, multiple users are "immersed" in a common space, augmented with multiple virtual objects. Such a TI application should normally implement structuring and navigation functionalities (like collision detection). Such kind of functionalities can be straightforwardly achieved by using full 3D geometry-based representations; b) Geometry-based representations can be Free-View-Point and 3D rendered using standard computer graphics, along with any 3D virtual object. Therefore, such a selection introduces uniformity in a TI framework; c) In a multi-party system, a pure image-based representation scheme would require the interpolation and compression of a large number of views at the capture-site, equal to the number of users. Additionally, in a central server-based system (opposed to a peer-to-peer framework), each capture-site should send all views that were interpolated for all users. For these reasons, the overall system's performance might be degraded for a large number of users; d) Finally, a 3D geometry-based TI system could benefit from the capabilities offered by future rendering systems, where the 3D content can be rendered in a full 3D, holography-like mode.

II. Tele-immersion platform

A. System overview

An overview of the proposed and studied TI communication chain is given in Fig. 1. Multiple calibrated RGB-Depth sensors capture the user and the depth data are used to reconstruct the 3D shape of the user, in the form of a triangle mesh. Given that the mesh resolution is adaptable, affecting the reconstruction details and visual quality, the whole chain becomes scalable in real-time.

Given the reconstructed mesh, the input depth maps and the calibration data, the system finds the RGB camera(s) onto which each mesh vertex projects. For vertices visible in more than one cameras, the two "best" cameras with respective IDs (e.g. ID_1, ID_2) are found, along with the corresponding two

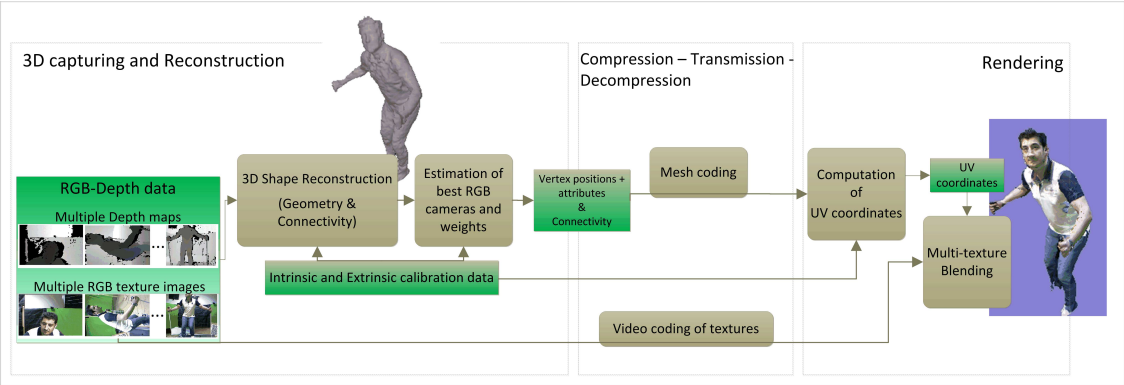


Fig. 1. An overview of the studied TI chain

weights w_1 and w_2 that represent the projection "quality" and sum up to unity. The original captured depth-maps are exploited to robustly infer about vertex "visibility". Each mesh vertex is represented as:

$$\mathbf{v}: \ [\, x, y, z, \ \mathrm{ID}_1, \mathrm{ID}_2, w_1 \,],$$

namely by its position and the triplet $[\mathrm{ID}_1, \mathrm{ID}_2, w_1]$, which is the described texture-related vertex attributes. In case a vertex projects into only one RGB camera, then $\mathrm{ID}_1 = \mathrm{ID}_2$ and $w_1 = 1$. Notice that ID_1 and ID_2 are integers that can be represented using only a few bits, since the number of cameras is small, while the value of w_1 can be quantized using also a few bits without affecting the visual quality. Additionally, notice that neighbor vertices normally share similar values for these attributes, resulting into high compression ratios with coders that exploit the spatial redundancy of vertex attributes.

The mesh vertices and connectivity are compressed using a static mesh coder. Additionally, the original texture image sequences are lossy compressed using a H.264 video codec. The compressed data are transmitted and decoded. The receiver, having available also the calibration data (which have to be transmitted only once), calculates for each vertex the UV coordinates on the corresponding RGB images. Then, the 3D mesh is rendered with multiple textures, via multi-pass texturing and making use of the texture weights w_1 and w_2.

The UV texture coordinates could have been computed at the reconstruction site and compressed/transmitted as additional vertex attributes. However, this would result into a large number of additional required bits. On the contrary, given that each vertex has to be reprojected onto one or two RGB images, this reprojection procedure is computationally "cheap" and can be performed at the receiver/rendering site, without significant computational burden. One may also opine that ID_1, ID_2, and w_1, w_2 could similarly be calculated at the receiver/rendering site. Notice however, that apart from requiring extra computational time, the original captured depth-maps are exploited to infer for vertex "visibility". This would require the availability of the original depth-maps at the receiver site.

B. 3D capturing and reconstruction of moving humans

1) 3D capturing: The utilized capturing system is comprised of $K = 5$ Kinect sensors that provide 360^o coverage of the whole human body. In order to internally calibrate a single Kinect, the method of [7] was used, while for the external calibration of the multiple Kinect sensors, our automatic method that makes use of large planar chessboard surface is utilized. The method simultaneously estimates the extrinsic parameters for all cameras, in an all-to-all registration manner. Further details are beyond the scope of this paper.

2) Real-time 3D reconstruction: A volumetric 3D reconstruction method is used, where the captured 3D surface is implicitly defined by a volumetric function $V(\mathbf{X})$, which is zero near the surface of the captured solid. The employed approach [8] is similar to the volumetric Signed Distance Function (SDF)-based one [9]. The reader is referred to [8], [9] for details. Here, an overview is provided for completeness.

The foreground object of interest is initially segmented out of the depth maps. The foreground object's 3D bounding box is discretized and each voxel $\mathbf{X}$ is projected on every depth-map $k = 1, \ldots, K$ to calculate the SD functions $V_k(\mathbf{X}) = \mathrm{sgn}(Z_k^o - Z) \cdot d(\mathbf{X}, \mathbf{X}_k^o)$, where $\mathbf{X}^o$ is the "observed" 3D point, according to the value on the depth map. A combined volumetric SD function is calculated from the weighted combination of the separate functions: $V(\mathbf{X}) = \sum_k w_k(\mathbf{X}) V_k(\mathbf{X}) / \sum_k w_k(\mathbf{X})$, where $w_k(\mathbf{X})$ are appropriate weights, discussed in the next paragraph. The extraction of the isosurface $V(\mathbf{X}) = 0$ using the Marching Cubes [10] produces the final complete and manifold 3D mesh. The described volumetric approach is suitable for a CUDA-based parallel computing implementation, since most of its stages involve pixel/voxel-wise calculations.

Appropriate weights: In [8], the selection of the weight $w_k(\mathbf{X})$ is based on the "quality" of the depth measurements. The "quality" of a depth measurement depends on the "viewing" angle, i.e. the angle between the unit vector $\hat{\mathbf{X}}_k = -\mathbf{X}_k / ||\mathbf{X}_k||$ and the observed surface's normal $\mathbf{N}_k$. Therefore, a confidence value for voxel $\mathbf{X}$ is calculated from $C_k^1(\mathbf{X}) = \max\{\hat{\mathbf{X}}_k \cdot \mathbf{N}_k, 0\}$. Additionally, according to observations in practice, the depth measurements near the foreground object's boundaries are noisy. Therefore, an associated confidence map $C_k^2(\mathbf{u})$ is calculated based on the distance of a depth pixel $\mathbf{u}$ to the depth-image background region. The weights are finally calculated from the product of the two confidence metrics.

3) Calculation of texture-related vertex attributes: For each vertex of the mesh, the two "best" RGB cameras have to be found, along with the corresponding two weights that

represent the projection "quality". Firstly, notice that the weights $w_k(\mathbf{X}), k = 1, \ldots, K$ contain the visibility information. Secondly, the projection "quality" strongly depends on the "viewing" angle. Moreover, the transition from one texture image to the other should be smooth, i.e. the weights w_1, w_2 should vary smoothly at mesh regions that are visible in more than one camera. Finally, inaccurate Depth-to-RGB camera calibration may lead to artifacts near object boundaries, thus such regions on the images should be given a small weight. All these observations are incorporated in the weights $w_k(\mathbf{X})$. Therefore, the weights $w_k(\mathbf{X})$ are used in order to speed-up the process. The two highest weights w_{k_1} and w_{k_2} are found and normalized such that $w_{k_1} + w_{k_2} = 1$. The indices k_1 and k_2 are set as the texture IDs for the corresponding vertex.

C. Textured mesh data compression

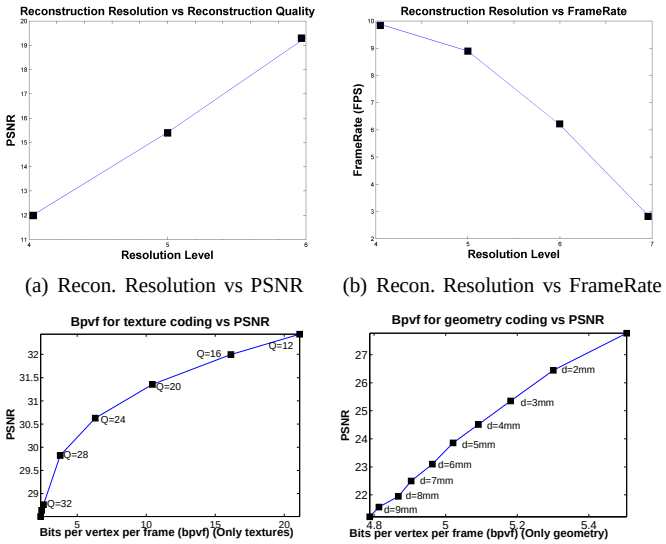
In order to transmit the reconstructed geometry over the network, mesh data compression is employed. In the hereby study, the reconstructed geometry, along with the vertex attributes, is compressed using the open source static mesh compression scheme OpenCTM (http://openctm.sourceforge.net/). For each vertex of the mesh, the two texture IDs (k_1 and k_2) along with w_{k_1} are encoded as vertex attributes, while w_{k_2} is implicitly defined as $w_{k_2} = 1 - w_{k_1}$. The texture IDs are quantized with a precision step equal to 1 (since they are integer numbers), while the weights w_{k_1} are quantized with a large precision step, equal to 0.2, without affecting the observed visual quality. The rest of the selected parameters for the CTM encoder are discussed and analyzed in the experimental section.

The texture images of the mesh are separately compressed using H.264 video coding. A wrapper for the openH264 (http://www.openh264.org/) library was developed and used for that purpose. The Intra-frame period was set equal to 10, as an often selection in video applications. Image denoising was enabled, which practically showed to increase significantly the compression ratio. In the experimental section, we study the performance of the system with respect to the quantization parameters of base quality layer.

III. Experiments

In this section we study the performance of the end-to-end TI chain in terms of visual quality, as well as timing measurements that affect the total tele-immersion experience. In order to provide actual and accurate comparisons with respect to the different reconstruction and mesh compression parameters, the experiments were conducted by using a prerecorded sequence of 310 multi-Kinect frames and simulating the transmission part. Data were transmitted between two PCs in our intranet and a network delay was simulated by considering different line bandwidths. The PCs used in the experiments feature an Intel i7 processor (3.2GHz), 8GB RAM, and a CUDA-enabled graphics card NVidia GTX 650.

In Fig. 2(a) we evaluate the visual quality of the reconstruction method for different voxel grid sizes (i.e resolution levels). For the resolution level r, the corresponding voxel grid size is $2^r \times 2^r \times 2^r$. The visual quality is evaluated in terms of PSNR, which is calculated by rendering the textured meshes



(a) Recon. Resolution vs PSNR (b) Recon. Resolution vs FrameRate

(c) Bpvf (only textures) vs PSNR (d) Bpvf (only geometry) vs PSNR

Fig. 2. Evaluation Diagrams

onto 36 virtual views, around the reconstructed object. The PSNR for different resolution levels is calculated with respect to the resolution level 7, which is considered as the reference reconstruction resolution level. In Fig. 2(b), the reconstruction frame-rate in frames per second (FPS) is depicted with respect to different reconstruction resolution levels. For the rest of the experiments, the reconstruction resolution 6 was chosen as a reasonable compromise between visual quality and frame-rates.

The study continues by investigating the impact of the H.264 compression to the visual quality. In Fig. 2(c), the PSNR of the sequence is plotted with respect to the quantization parameters of base quality layer, denoted here as Q, and the corresponding average Bits Per Vertex per Frame (bpvf) for compressing the texture images. The PSNR is calculated with respect to the rendered mesh using the original uncompressed texture images. The quantization parameters is selected equal to $Q = 28$, which corresponds to approx. 4bpvf and a PSNR close to 30dB, as a reasonable compromise between visual quality and compressed textures size.

Having chosen a reconstruction level $r = 6$ and set the H.264 quantization parameter to $Q = 28$, we further investigate the impact of the absolute vertex precision, used in the quantization process of OpenCTM compression (denoted here as d), to the final visual quality. The results are depicted in Fig. 2(d). A vertex precision of $d = 8mm$, which corresponds to approx. 4.8bpvf and a PSNR close to 22dB, was chosen to have reasonable quality outcome after subjective evaluation. For the reader's convenience, a frame of the evaluation sequence is depicted in Fig. 3 under different resolution levels and compression parameters for subjective evaluation.

Finally, for the chosen visual quality settings, we evaluate the performance of the whole chain in terms of frame-rate at the receiver side, as well as the delay between the capturing time and rendering time at the receiver. The factor that further

(a) $r = 7$, uncompressed.

(b) $r = 6$, uncompressed.

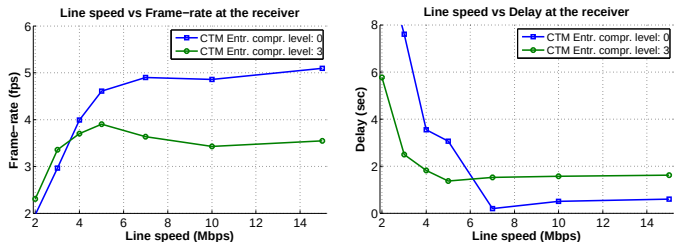
(c) $r = 6$, $d = 8$mm.

(d) $r = 6$, $d = 20$mm.

Fig. 3. A frame of the evaluation sequence rendered under different parameters. r refers to resolution level and d to mesh vertex precision.

influences these measurements is the entropy compression level of the OpenCTM mesh codec. The higher the compression level the lower the compressed output in terms of bpvf. However, the processing time is proportional to the entropy level coding. The results of this experiment are shown in Fig 4 for different network lines: 2-5, 7, 10, and 15MBps.

Some important conclusions that can be drawn are: a) For slow network lines, <5Mbps, the frame-rate and delay are respectively proportional and inversely proportional to the line speed. However, for faster lines, both the delay and the frame-rate do not depend on the line. In the former case (<5Mbps), the bottleneck of the whole chain is the transmission, while in the latter case the bottleneck is the compression process; b) Based on the previous conclusion, for fast lines it is preferable to have inefficient (in terms of bpvf) but faster compression. The opposite holds for slow lines. This can be verified in the diagrams of Fig 4, where it is obvious that an entropy compression level equal to 0 is preferable for lines faster than 5Mbps. The reason is that better entropy coding requires additional processing time, without actually improving user experience, in terms of delay and frame-rate; c) Finally, the diagrams of Fig. 4 contend that for a network line <5Mbps, the total experience is expected to be poor for the quality standards we have set. The delay between the endpoints of the



(a) Line speed vs fps-rate at the receiver (b) Line speed vs delay at the receiver

Fig. 4. Time measurements for the whole capture-to-render chain.

chain is 2sec at minimum. Contrariwise, with faster network channels (>7Mbps), the experience is significantly improved. The delay is approx. 0.5 sec (blue curve) and the rates at the receiver are close to the reconstruction rates in the capture-site.

IV. Conclusions and Future Work

In this paper, an end-to-end TI chain was proposed and its key elements were analyzed. Its performance with respect to the various parameters was in detail evaluated. An advantage of the system is its scalability, since both the 3D reconstruction and compression modules are scalable and adaptable to the characteristics of the users' workstations and the network. Studying variants and potential improvements of the separate chain modules (e.g. time-varying vs static mesh compression) and more importantly variants of the chain as a whole (e.g. considering multi-view VD coding and 3D reconstruction at the receiver site) is also in our future plans.

Acknowledgment

This work was supported by the EU-funded project 3D-Live, GA 318483.

References

[1] R. Vasudevan, G. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Bajcsy, and K. Nahrstedt, "High quality visualization for geographically distributed 3D teleimmersive applications," *IEEE Trans. on Multimedia*, vol. 13(3), June 2011.

[2] A. Smolic, "3D video and free viewpoint video-From capture to display," *Pattern Recognition*, vol. 44, no. 9, Sep. 2011.

[3] A. Maimone and H. Fuchs, "Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras," in *10th IEEE ISMAR 2011*.

[4] R. Mekuria, D. Alexiadis, P. Daras, and P. Cesar, "Real-time encoding of live reconstructed mesh sequences for 3D tele-immersion," in *Intern. Conf. on Multimedia and Expo*, 2013.

[5] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Trans. on Multimedia*, vol. 15, no. 2, pp. 339–358, 2013.

[6] B. Dai and X. Yang, "A low-latency 3D teleconferencing system with image based approach," in *12th Int. Conf. on Virtual-Reality Continuum and Its Appl. in Industry*, 2013.

[7] D. Herrera, J. Kannala, and J. Heikkila, "Joint depth and color camera calibration with distortion correction," *IEEE Trans. Pattern Anal Mach. Intell.*, vol. 34, 2012.

[8] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, realistic full-body 3D reconstruction and texture mapping from multiple kinects," in *11th IEEE IVMSP*, Seoul, Korea, 2013.

[9] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. SIGGRAPH*, 1996.

[10] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Comp. Graphics*, vol. 21, 1987.