BLENDING REAL WITH VIRTUAL IN 3DLIFE

K.C. Apostolakis[*], D.S. Alexiadis[*], P. Daras[*], D. Monaghan[•], N.E. O'Connor[•], B. Prestele[‡], P. Eisert[‡], G. Richard[□], Q. Zhang[†], E. Izquierdo[†], M.B. Moussa[↑], N. Magnenat -Thalmann[↑]

[*] Centre for Research and Technology - Hellas, Information Technologies Institute, Thessaloniki, Greece
[•] CLARITY: Centre for Sensor Web Technologies, Dublin City University, Ireland
[‡] Fraunhofer HHI / Humboldt University, Berlin, Germany
[□] Institut Telecom / Telecom ParisTech, Paris, France
[†] Multimedia and Vision Group (MMV), Queen Mary University, London, UK
[↑] MIRALab, University of Geneva, Geneva, Switzerland

ABSTRACT

Part of 3DLife's major goal to bring the 3D media Internet to life, concerns the development and wide-spread distribution of online tele-immersive (TI) virtual environments. As the techniques powering challenging tasks for user reconstruction and activity tracking within a virtual environment are maturing, along with consumer-grade availability of specialized hardware, this paper focuses on the simple practices used to make real-time tele-immersion within a networked virtual world a reality.

1. INTRODUCTION

Networked virtual environments oriented towards TI [1] [2] [3] are becoming a frontier in social computing. Scientific interest has been focused towards allowing users to fully immerse themselves within the interactive experience, as the boundaries between real and virtual are blurred. With the introduction of consumer-grade devices such as Microsoft's Kinect sensor, striving to introduce real-time interaction within immersive virtual environments to everyday average users over the Internet, poses the next logical challenge in the TI research frontier.

In this paper, research work towards achieving immersion of users within a networked virtual world is demonstrated. Within the scope of a collaborative dance masterclass use case scenario, we showcase how users can immerse themselves within the virtual environment as they are being reconstructed in real time, or puppeteering a pre-created virtual character (avatar). In both cases, skeleton tracking algorithms collect data in real time, and a dance comparison algorithm assesses comparative results between the user's current performance and a pre-recorded choreography uploaded by another user, highlighting environmental reaction to user intervention. The experience

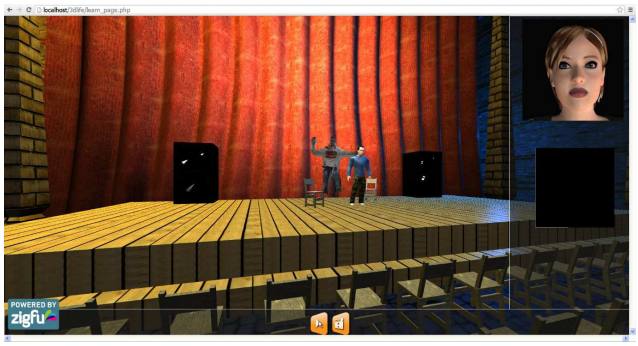

Fig. 1. Screenshot of the integrated 3DLife application: Reconstructed user interacts with a virtual character within an immersive environment running in Google Chrome.

can further be enriched with additional effects, such as auto stereoscopic rendering for glasses-free 3D display and spatial 3D surround sound. All these are achieved by connecting to a standard client-server network using just a Kinect sensor and a web browser. A screenshot of the dance studio use case scenario application developed for 3DLife can be seen in Fig. 1.

The remainder of this paper is organized as follows: Section 2 describes the collaborative dance masterclass use case motivating the integration, while Section 3 summarizes the minimum setup prerequisites. Real-time online 3D reconstruction of users is covered in Section 4, a method for avatar embodiment using skeleton tracking is presented in Section 5 and integration of on-line dance evaluation algorithm is explained in section 6. This paper concludes with a brief overview on related topics of interest in Section 7, before final conclusions are drawn in Section 8.

2. COLLABORATIVE DANCE MASTERCLASS

In order to demonstrate how real-life interaction within a virtual world can be achieved, a simple use case was devised revolving around a fictional social network service

This work was supported by the European Commission under contract FP7-247688 3DLife (http://www.3dlife-noe.eu/3DLife)

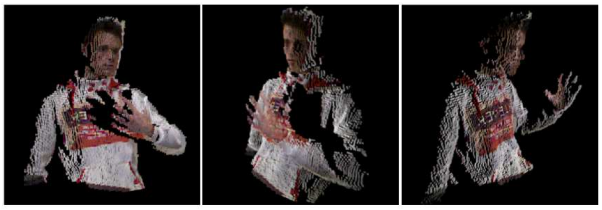

Fig. 2. Examples of reconstructed mesh point clouds rendered in 60 FPS running on Google Chrome web browser, a single Kinect, the ZigFu browser plugin and Three.js 3D Library.

designed to help users improve their dancing skills with the help of certified professionals. Users are able to create accounts and identify themselves as either dance tutors or dance students, allowing them to enter 3D virtual dance scenes and try out simple dance routines, showcasing and comparing their dance moves to other users sharing the network. This use case succeeds in covering all research aspects concerning the blending of real with virtual interaction boundaries.

3. MINIMUM USER REQUIREMENTS

Along the proposed literature 3D tele-immersive systems [4] [5], in this paper we demonstrate a cost-efficient, simple approach in order to reach a wide audience of average users and make 3D TI systems applicable in real-life situations. The importance of such an off-the-shelf approach generates a need to introduce consumer-grade devices to the pipeline, lifting the burden of time-consuming external software dependency installations. For the integrated demo, a Microsoft Kinect sensor is required. Furthermore, in order to allow communication of JavaScript client-based code with the Kinect skeleton tracking libraries, the ZigFu browser plug-in (available at http://zigfu.com/) is required. Installation of this plugin is a familiar process that takes no more than a couple of seconds, similar the installation of other web browser plugins, such as Adobe's Flash player.

4. REAL-TIME ON-LINE RECONSTRUCTION

In order to achieve real-time reconstruction of users over the web, the Kinect sensor's depth and color streams are used to generate a colored 3D point cloud as described in [6], which can then be placed inside the virtual scene. The Kinect depth stream is represented as a 3-channel color video, obtained through the ZigFu browser plugin, where the Least Significant Byte (LSB) is encoded in the red channel $D_R(u,v)$ and the Most Significant Byte (MSB) in the green channel $D_G(u,v)$. Namely, the depth value at pixel (u,v) is given in mms by $D(u,v) = 256 \cdot \left[D_R(u,v) + 256 \cdot D_G(u,v) \right]$. The depth map is masked by the user-tracking label map in order to subtract background. For each nonzero pixel (u,v) in the masked depth image $D(u,v)$, a 3D point (vertex) $\mathbf{V}(u,v) = [X(u,v), Y(u,v), Z(u,v)]^{\mathrm{T}}$ is created. The Z coordinate (depth) of each vertex is obtained from the depth map, at the corresponding pixel $D(u,v)$. Afterwards, the 3D point coordinates X and Y are calculated using the standard pinhole camera model:

$$X = \left(u - u_0\right) \cdot Z \,/\, f_x$$
$$Y = \left(v - v_0\right) \cdot Z \,/\, f_y \,,$$

where (u_0, v_0) is the depth camera's principal point and f_x, f_y the depth camera's focal length components. With each vertex 3D position $\mathbf{V}$ known, re-projection to the Kinect RGB camera can provide information on the vertex color. The vertex $\mathbf{V}$ is initially transformed from the depth camera to the RGB camera 3D coordinate system, according to equation $\mathbf{V}' = \mathrm{R} \cdot \mathbf{V} + \mathbf{T}$, where the matrix R and vector $\mathbf{T}$ represent the relative rotation and translation between the two sensors. Then, re-projection of $\mathbf{V}'$ to the Kinect RGB camera using the (pinhole) RGB camera's model, gives the corresponding pixel on the RGB map, and the 3D vertex color is retrieved.

All of the above calculations are performed on the GPU using WebGL shaders. Compensating on the reconstruction quality by reducing depth and color map resolution to 160x120 pixels, reconstruction over the web is achievable in real-time with rendering performance reaching 60 FPS on a mid-range PC. Examples of reconstructed meshes are shown in Fig. 2.

5. AVATAR CONTROL VIA SKELETON TRACKING

A second milestone for user immersion within the virtual scene alongside capturing and reconstruction, avatar embodiment allows users to puppeteer a virtual character created by an external authoring application. The process requires that the character mesh is parented to an articulated structure of control elements called *bones*. Bones can be viewed as oriented 3D line segments that connect transformable *joints* (such as a knee or shoulder). Avatar embodiment within the collaborative dance masterclass use case is achieved by allowing users to create avatars, rigged with a pre-defined 17-joints hierarchy (similar to the OpenNI joint tracking structure), as in [7].

For each frame, the OpenNI skeleton-tracking module generates joint position and rotation data with respect to the Kinect camera's world coordinate system. However, since the avatar skeletal rig generated using Blender 3D Modeling application follows a hierarchical structure that requires transformations to be applied in each joint's local axis space in relation to its parent, the data obtained by the skeleton tracking module is translated to avatar skeletal animation data using the following algorithm:

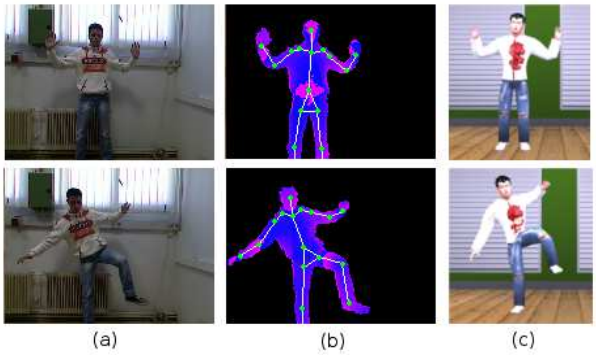

Fig. 3. Example of the avatar embodiment algorithm.

1. For each avatar rig joint $\mathbf{J}_i, i=1,\ldots,17$, the bone lengths $L_i^c,\ c \in \mathrm{CH}\{i\}$ to its first-level children joints $\mathrm{CH}\{i\}$ are calculated in the avatar rig hierarchy, along with the corresponding unit vector directions $\mathbf{d}_i^c = \left(\mathbf{J}_c - \mathbf{J}_i\right) / \left\|\mathbf{J}_c - \mathbf{J}_i\right\|$. These calculations are performed once off-line, before tracking is initiated. For notational simplicity, we drop the superscript c (referring to the "children"), wherever it is implied.
2. For each tracking frame, the user's joint positions $\mathbf{P}_i$ with respect to the camera's world coordinate system are given by the skeleton tracking algorithm.
3. The direction of each user-tracked bone is determined by the unit-normalized vector $\mathbf{s}_i = \left(\mathbf{P}_c - \mathbf{P}_i\right) / \left\|\mathbf{P}_c - \mathbf{P}_i\right\|$, similar to the calculations made in the first offline step. The result is then multiplied by its corresponding length value L_i, to ensure that all tracked joints are within pre-defined avatar bone length distances of each other.
4. The axis of rotation $\mathbf{m}_i$ is determined by calculating the cross product $\mathbf{d}_i \times \mathbf{s}_i$, while the angle of rotation is calculated by $\theta_i = \cos^{-1}(\mathbf{d}_i \cdot \mathbf{s}_i)$, i.e. from the inner product of $\mathbf{d}_i$ and $\mathbf{s}_i$.
5. Avatar new joint position is set by traversing the joint hierarchy from the parent joint and adding the offset $L_i \cdot \mathbf{s}_i$. The local rotation quaternion $\mathrm{q}_i^{\mathrm{lo}}$ is calculated from the axis $\mathbf{m}_i$ and the eigen-angle θ_i. The final rotation quaternion q_i is calculated by multiplying the inverse rotation quaternion of the parent q_{pa} with $\mathrm{q}_i^{\mathrm{lo}}$, i.e $\mathrm{q}_i = \left(\mathrm{q}_{\mathrm{pa}}\right)^{-1} \cdot \mathrm{q}_i^{\mathrm{lo}}$.

Actual animation is computed by traversing the avatar joint hierarchy from its root and setting up the final skeleton pose. A set of joint influences and corresponding weights are applied onto each vertex of the mesh via skinning equations. The resulting mesh resembles the virtual character mimicking the user's pose, as can be seen in Fig. 3. The effect can also be applied with pre-recorded skeleton tracking data as input, allowing virtual characters to playback past recorded animations.

6. ON-LINE VIRTUAL DANCE EVALUATION

In order to support collaborative dance features of the integrated 3DLife demo, and explore virtual environment feedback on immersed user activities, a variant of the automated dance evaluation algorithm described in [8] was integrated onto the demo application. In a nutshell, the employed algorithm makes use of quaternions to represent the user's joint position data relative to the skeletal hierarchy root joint (Torso). An instantaneous evaluation score is calculated for each frame, by computing the "all-joints" quaternionic correlation coefficient, within the temporal window of the last few frames.

A web-oriented implementation of the algorithm was used to evaluate user real-time dance performance in comparison to a pre-recorded choreography uploaded by another user of the network. More specifically, user data is placed within a FIFO queue, with a frame capacity of $\Delta T = 150$ frames (this corresponds to 5sec, considering the Kinect's frame rate 30fps). The same time window is applied for consideration of pre-recorded teacher skeleton data in the same time reference. Then:

1. The joint positions, relative to the Torso for both the teacher and student are calculated, by subtracting from each joint the Torso position. These positions are then represented as "pure" quaternions and stored in quaternionic matrices $\mathrm{P}(i;t)$ and $\mathrm{Q}(i;t)$ of sizes $17 \times \Delta T$.
2. Additionally, the temporal mean quaternions $\overline{\mathrm{P}}(i)$ and $\overline{\mathrm{Q}}(i)$ are calculated and subtracted, to produce the zero-mean (along time) quaternionic matrices $\widetilde{\mathrm{P}}(i;t) = \mathrm{P}(i;t) - \overline{\mathrm{P}}(i)$ and $\widetilde{\mathrm{Q}}(i;t)$, respectively.
3. The "all-joints" quaternionic correlation coefficient is now calculated from:

$$S = \frac{|\mathrm{c}_{\mathrm{P,Q}}|}{\sqrt{\mathrm{c}_{\mathrm{P,P}} \cdot \mathrm{c}_{\mathrm{Q,Q}}}},$$

where

$$\mathrm{c}_{\mathrm{P,Q}} = \sum_{i,t} \widetilde{\mathrm{P}}(i;t) \cdot \widetilde{\mathrm{Q}}(i;t)$$
$$\mathrm{c}_{\mathrm{P,P}} = \sum_{i,t} \widetilde{\mathrm{P}}(i;t) \cdot \widetilde{\mathrm{P}}(i;t).$$
$$\mathrm{c}_{\mathrm{Q,Q}} = \sum_{i,t} \widetilde{\mathrm{Q}}(i;t) \cdot \widetilde{\mathrm{Q}}(i;t)$$

The resulting score value lies in the interval $[0,1]$. This instantaneous scoring information can be utilized by

environmental assets, such as autonomous virtual tutors and graphic displays, to notify users about their performance.

7. IMPLEMENTATION SPECIFICS

The following paragraphs provide a brief summary of the techniques used to address some issues presented during the development phase, concerning relative areas of interest.

7.1. Streaming Kinect data over the web

As the depth channel frames are represented as 3-channel color video, and current web technologies only support video data being subjected to considerable compression (MP4, WebM and OGG), recording and streaming the highly sensitive depth data stream will inevitably lead to data corruption. To address this issue each depth frame is encoded using a Base64 encoding scheme. Consecutive frames are thus saved as an array of ASCII strings. A Base64 decoder is similarly required on the client-side to decode the string array and obtain the original depth image. Similar techniques are used to preserve the quality of the color and label map stream data. This method of storage is practically lossless, and guarantees the preservation of the depth data integrity, enabling uploading and downloading of pre-recorded depth data for reconstruction. However the resulting file data volume being stored onto a remote server is analogous to the recording time and frame resolution. As a counter-measure, all recordings within the integrated use case scenario are limited to 1 min. 30 sec. per dance routine.

7.2. Auto-stereoscopic rendering

To further enhance modern 3D immersive experience, auto-stereoscopic rendering post-processing effects can be applied to the WebGL renderers, to support glasses-free 3D wherever a specialized display is available. To this end, a Three.js (http://mrdoob.github.com/three.js/) effect module for rendering WebGL 3D content to the Tridelity MP 4210 auto-stereoscopic display was implemented and integrated onto the demo. As the module applies the effect after the scene has been rendered, real-time performance of the rendering pipeline is not burdened.

7.3. Web 3D spatial sound

Complete immersion within the virtual environment requires that the world is enriched with realistic surround audio. Using 3D spatial sound adds an additional layer of realism to the virtual scene, as users are able to identify where each audio signal originated as well as their proximity to its source location by strategically increasing or decreasing sound volume as users approach or withdraw from the source. Modern web technologies support the simultaneous playback of multiple audio signals, and as such, the development of real-time web-based methods that split a single audio signal into spatial components is considered a highly added value.

8. CONCLUSIONS

In this paper, research work on immersing users within networked virtual environments and blending the boundaries between real and virtual was demonstrated. An integrated collaborative dance masterclass use case around which a prototype networked virtual world was built, serves as a prime example of how different methodologies for online real-time user reconstruction, avatar embodiment via skeleton tracking and real-time environmental feedback in the form of a dance evaluation algorithm, can be applied to enhance user 3D web immersive experience. Additionaly, other topics of interest have been addressed and could serve as possible areas for future research. We have shown with this integrated demo that such technologies have reached a maturity level that can capitalize on current consumer-grade hardware, and have hopefully paved the first step towards wide-spread application of web-based 3D TI environments.

9. REFERENCES

[1] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo and R. Bajcy, "Enabling multi-party 3D tele-immersive environments with ViewCast," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 6(2), 2010, pp. 1-30.

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

[3] D.S. Alexiadis, G. Kordelas, K.C. Apostolakis, J.D. Agapito, J.M. Vegas, E. Izquierdo and P. Daras, "Reconstruction for 3D immersive virtual environments," *2012 13th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS),*.IEEE, 2012, pp. 1–4.

[4] Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S.H. Jung and R. Bajscy, "Teeve: The next generation architecture for tele-immersive environments," in *Multimedia, Seventh IEEE Symposium on*. IEEE, 2005, pp. 8–pp.

[5] W. Wu, R. Rivas, A. Arefin, S. Shi, R.M. Sheppard, B.D. Bui and K. Nahrstedt, "MobileTI: a portable tele-immersive system," in *Proceedings of the 17th ACM International conference on Multimedia*. ACM, 2009, pp. 877–880.

[6] J. Kramer, M. Parker, D. Herrera, N. Burrus and F. Echtler, "Hacking the Kinect," Apress, 2012.

[7] A. Sanna, F. Lamberti, G. Paravati and F.D. Rocha, "A kinect-based interface to animate virtual characters," in *Journal on Multimodal User Interfaces*. 2012, pp. 1-11.

[8] D.S. Alexiadis, P. Kelly, P. Daras, N.E. O'Connor, T. Boubekeur and M.B. Moussa, "Evaluating a dancer's performance using kinect-based skeleton tracking," in *Proceedings of the 19th ACM International conference on Multimedia*. ACM, 2011, pp. 659-662.