ENHANCING REAL-TIME FULL 3D RECONSTRUCTION OF HUMANS WITH PRE-SCANNED MESHES

Nicholas Vretos, Member, IEEE, Dimitrios Alexiadis, Dimitrios Zarpalas and Petros Daras, Senior Member, IEEE

Centre for Research and Technology Hellas, Information Technologies Institute
6th Km Charilaou-Thermi Road, Thessaloniki, Greece,
e-mail:{vretos; dalexiad; zarpalas; daras}@iti.gr

ABSTRACT

In this paper, we propose a novel, full-body, real-time 3D reconstruction framework that makes use of pre-scanned body parts (more precisely pre-scanned 3D heads) so as to provide a more detailed 3D reconstruction mainly in the semantically important head area. Our framework deals with 3 major challenges: a) multiple depth sensors collaboration, b) pre-scanned head positioning and c) reconstruction and texturing. In all the above challenges, we propose novel solutions so as to cope with time and space complexity, synchronization and 3D mesh quality. Experimental evaluation provides evidence that superior 3D mesh quality can be achieved compared to simple (not enhanced) use of depth cameras' data.

Index Terms— 3D reconstruction, 3D head tracking, RGB-D sensors

1. INTRODUCTION

Real-time, full 3D human reconstruction is an important task with multiple applications in the industry. From tele-immersion to gaming and from novel virtual collaboration platforms to novel social networks' platforms, real-time, full 3D reconstruction is in the core of such promising technologies. Although a great demand, from the industry, is emerging, technological advancements in the area are still in a prototype phase and usually demands for either large and expensive capturing apparatuses and setups, to provide better accuracy in depth measurement, or considerable computational power that prohibit such technologies to home users.

After the commercialization of Kinect and other depth acquisition devices (Time of Flight cameras), the research community has shown a great interest towards achieving full, real-time 3D reconstruction. Several such solutions have been proposed so far. Most of them try to tackle the way the 3D reconstruction is undertaken with a wide series of publication to that direction [1]-[4]. Other research works focus on the real-time aspect of the whole framework and usually transfer and modify methods of 3D reconstruction to GPU implementations, using the very powerful CUDA framework [5], aiming to achieve better results with respect to time complexity [6], [7]. The full 3D reconstruction aspects is also significantly researched [8]-[11]. Finally, the dynamic aspect of full 3D reconstruction is much less studied due to the high complexity of the real-time depth capturing devices collaboration schemes that need to take under consideration several aspect, such as, synchronization, network streaming of "heavy" data (i.e., coding), texturing from multiple devices, spatial calibration of different devices (online or offline) and others. Due to these inherent issues, full 3D reconstruction of dynamic scenes is lagging behind, but nevertheless some very interesting works have also been reported [12]-[14].

In the present work, we propose a novel framework for real-time, full 3D reconstruction of moving humans from multiple depth capturing devices (Kinect V2), exploiting pre-scanned heads to semantically enhance important facial details. To achieve our goal and provide a holistic approach, several novelties in all the above domains enter the play. The most important ones are the following:

- Real-time face (2D) capturing from multiple depth sensors and head (3D) positioning,
- frame synchronization between different sensors,
- fusion of pre-scanned 3D data with live raw 3D data,
- weighted texturing from multiple RGB cameras.

To the best of the authors knowledge, no similar work has yet been reported making use of predefined scans that integrate with real-time, raw data coming from multiple Kinects to provide a more detailed reconstructed model.

2. PROPOSED FRAMEWORK

2.1. Overview

The proposed framework provides a holistic solution for capturing and 3D reconstruction, while at the same time it enhances the raw data with a pre-scanned 3D head mesh. Firstly, the head is scanned offline with the KinectFusion algorithm

This work was supported by the EU funded project PATHway under contract 643491.

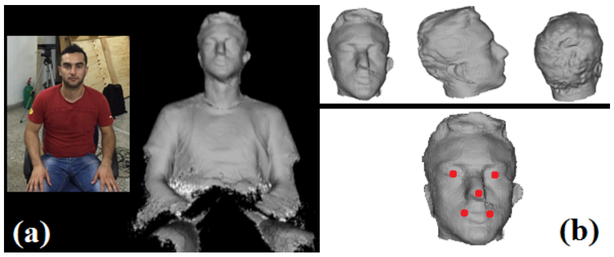

Fig. 1: KinectFusion 3D scanning and head post processing.

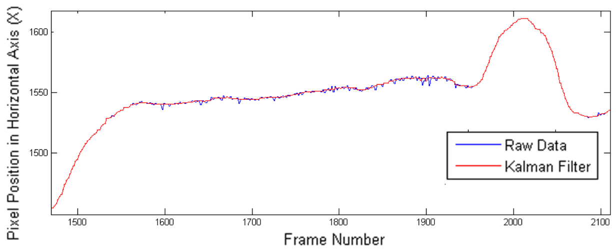

Fig. 2: Kalman filtering to facial 3D points trajectory

[4] and therefore the scanned head is pre-processed to cut information other than the head (shoulders, neck, etc.) as well as other inconsistences that may appear in the reconstructed object. In Fig. 1, the preprocessing steps are shown from raw KinectFusion reconstruction to the head scan. Subsequently, we chose 5 landmarks (nose, left eye, right eye, left mouth, right mouth) on the pre-scanned head.

At the hardware level, the proposed framework comprises of 4 Kinect V2 cameras and 5 PCs (i.e., 4 client PCs for each Kinect V2 and 1 Master-Server PC collecting data). Each client PC streams data (color, depth and facial points) to the server PC in real time rates (15fps) through a network interface. The reason for using several computers is the fact that the new Kinect V2 does not support installation of multiple Kinects onto the same computer, due to bandwidth limitations. Each Kinect client (i.e. Kinect V2 and PC) is responsible to provide data to the server, which will perform on-the-fly a 3D reconstruction, fusing also the pre-scanned 3D head data in the reconstruction process. At the client level, other than simple depth, color and facial points acquisition, smoothing of the head position and orientation extraction is performed.

The overall setup provides a functional (acquisition) space of diameters larger than 4 meters, which is adequate for most known tele-immersion applications.

2.2. Face Tracking and Head Positioning

2.2.1. Face Tracking

With the term "face tracking" we refer to the 2D tracking of the face from each Kinect V2 device. Kinect's SDK provide a framework for tracking a face in 2D for each Kinect and more particulary in the Infrared data (IR). From an algorithmic point of view, the Kinect SDK uses the method presented in [15]. By doing so, we obtain 5 points in the 2D space of the IR image for each Kinect that has a valid face. Subsequently, we transform these points from the 2D space to the world space (3D space) of each Kinect. Although the Kinect sensor seems to behave quite good in localizing the face, results for the facial points are quite noisy. To resolve the latter and provide robustness to our framework, we apply 5 distinct Kalman filters [16] to each of the 3D facial points extracted from the Kinect comprising the positions and velocities of each 3D point (giving a 6-dimensional Kalman Filter). An example of such data smoothness can be seen in Fig. 2 for the X axis of a facial point (left eye). All the above are implemented at each client separately and each client will then transmit the filtered positions, in order to save calculation burden from the server. It has to be noted that we apply Kalman filtering in the 3D space and not in the 2D space of the extracted features to also compensate for inaccuracies in the reprojection function. The physics implemented within the Kalman filter were experimentally deduced through a trail-error procedure.

2.2.2. Head Positioning

In order to choose among all Kinects the one that will provide the head positioning information, we apply the following steps:

- find all Kinects that detected a face,
- get the face orientation in local Kinect world space,
- compare the orientations and choose the one with the minimum absolute value angle around the Y axis.

Once the winning Kinect has been chosen, we gather a position vector $\mathbf{v} = (X, Y, Z) \in \mathbb{R}^3$ and quaternion $\mathtt{q} = (x, y, z, w) \in \mathbb{R}^4$ extracted from this Kinect sensor for the head position and orientation, respectively. Therefore, based on these two vectors, we construct a transformation matrix $\mathtt{RT}$ as in [17]. The application of this transformation to the pre-scanned 3D head gives the initial position estimation of the head. Subsequently, the manually annotated facial points on the pre-scanned head are registered with the position acquired. The final $\mathtt{RT}$ transformation is obtained by minimizing the squared distance between the facial landmark 3D points and the facial 3D points returned from the Kinect sensors.

2.3. Multiple Devices Collaboration

2.3.1. Network collaboration and synching

In order to be able to work with multiple Kinects V2 we need to create a network infrastructure and resolve issues that span from communication and calibration to synchronization. The client program is responsible to gather data and stream them

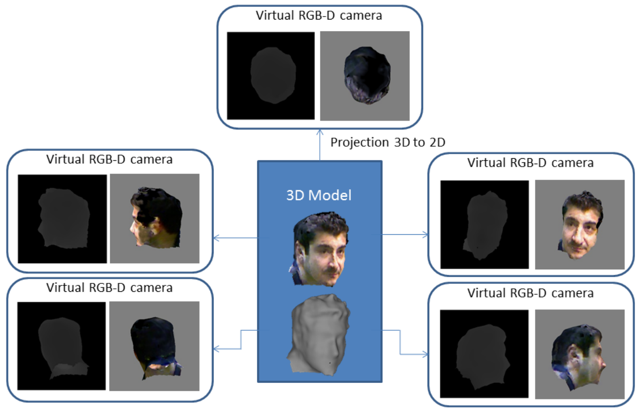

Fig. 3: Projection of 3D head model onto multiple "virtual" RGB-D cameras.

(in real-time) to the server. The vast amount of data are the HD color image that needs to be send through the network. We use a standard JPEG compression scheme [18] before sending the image through the network and thus achieve real-time frame rates at the network level transmissions.

On the other hand, a crucial aspect of the algorithm is to gather synchronized frames from all the capturing apparatuses. To achieve this goal:

- each client grabs frames asynchronously,
- the server broadcasts a "ready-to-receive" message,
- each client sends the last grabbed frame,
- the server collects and process the data.

The above protocol is based on a *wait for all* principle. Under this principle the server has, each time it asks for, a global synchronised frame (consisting of the frames from all Kinects) grabbed from all clients at practically the same time, since the broadcasted "ready-to-receive" message arrives to all clients at practically the same time.

2.4. 3D reconstruction - Fusion of pre-scanned head data

This section addresses the problem of seamlessly fusing the raw RGB-D data from multiple Kinects with the pre-scanned head model information, to produce a single textured, manifold and watertight mesh, on a per-frame basis. The basic 3D reconstruction framework is initially shortly described, before discussing the integration of the pre-scanned head model information in that framework.

Basic full-body reconstruction framework

2.4.1. Raw point-normal generation

For each "foreground" pixel $\mathbf{u} \in \mathcal{F}_k$ on the k-th depth-map, a raw 3D point $\mathbf{X}_k(\mathbf{u})$ is generated. The corresponding raw 3D normal $\mathbf{N}_k(\mathbf{u})$ is calculated based on the connectivity information, obtained using terrain Step-Discontinuity Constraint Triangulation (SDCT) [13].

2.4.2. 3D volume reconstruction

To produce a manifold and watertight surface from the raw 3D point-normals, a volumetric FT-based approach [19] is employed, enriched with a smoothing scheme. The aim is to calculate a scalar volume function $A(\mathbf{q})$, which implicitly contains the surface information as the isosurface at an appropriate level L. For details, the reader is referred to [19]. In the standard version of the method, each raw point-normal sample is "clapped" to its nearest voxel. In the proposed method's variation, each normal $\mathbf{N}_k(\mathbf{u})$ is smoothly "splatted", according to:

$$\mathbf{V}(\mathbf{q}) = \frac{1}{d(\mathbf{q})} \sum_k \sum_{\mathbf{u}\in\mathcal{F}_k} g(\mathbf{X}_k(\mathbf{u}), \mathbf{q}; \sigma_1) \cdot \mathbf{N}_k(\mathbf{u}), \tag{1}$$

where $g(\mathbf{X}_k(\mathbf{u}), \mathbf{q}; \sigma_1)$ are "splatting" weights based on the distance d of point $\mathbf{X}$ from voxel $\mathbf{q}$ using the Gaussian $g(d;\sigma_1) = \sigma_1^{-1}\exp(-d^2/\sigma_1^2)$. The normalization factor $d(\mathbf{q}) = \sum_k \sum_{\mathbf{u}\in\mathcal{F}_k} g(\mathbf{X}_k(\mathbf{u}), \mathbf{q}; \sigma_2)$ is an estimate of the points density at the voxel $\mathbf{q}$, i.e. kernel density estimation [20] is employed, using a Gaussian kernel. To avoid singularities, σ_2 should always be larger than σ_1. It was experimentally selected to be equal to $\sigma_2^2 = \frac{3}{2}\sigma_1^2$. With respect to σ_1, the larger its value, the smoother the output gradient-field and the final reconstruction will be, and it was set equal to voxel radius.

The method was implemented with CUDA, to achieve high-reconstruction rates.

2.4.3. Weighted texture blending

To produce the texture of the reconstructed mesh, a weighted texture blending approach is followed, which is detailed in [21]. In a nutshell, the color of a reconstructed vertex is obtained as the weighted sum of the colors on the "visible" RGB images, using weights $W_k(\mathbf{u}) \in [0, 1]$ that depend on: a) the "viewing" angle of the captured surface, i.e. on the angle between the line-of-sight and the vertex normal, and b) the 2D distance of the projection pixel $\mathbf{u}$ from the foreground human's 2D silhouette.

Fusion of pre-scanned head information

2.4.4. Fusion of head geometry information

The raw point-normal cloud, as e.g. depicted at the left of Fig. 4(a), generated according to subsection 2.4.1, is "augmented" (i.e. by concatenation) by the aligned heads point-normal cloud, as e.g. at the left of Fig. 4(b). The "augmented" point-normal cloud is used as input to the volumetric reconstruction process of paragraph 2.4.2. Since the head model is much denser than the raw point-cloud, its contribution in the volumetric reconstruction process will be locally larger.

2.4.5. Exploiting the head texture

Due to self-occlusions, or when Kinect sensors do not provide enough coverage of the human (as e.g. shown in Fig. 6),

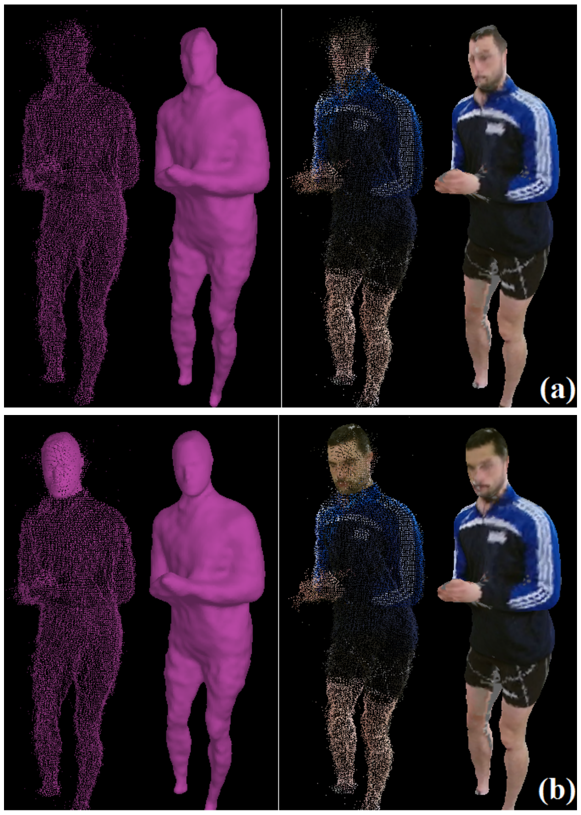

Fig. 4: “Gaelic football” sequence - (a) Without and (b) with fusion of of pre-scanned head in the reconstruction. Left: Raw point cloud and reconstruction (geometry only). Right: Reconstruction with texture.

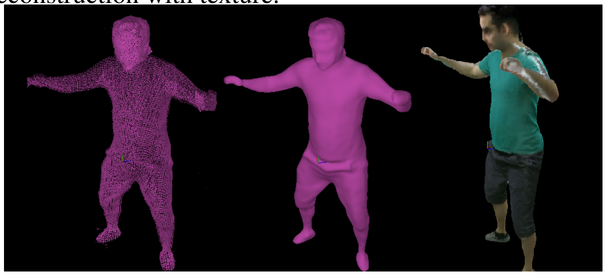

Fig. 5: Argyris sequence with fusion of pre-scanned head. Left: Raw point cloud, “augmented” with the head and reconstruction. Right: Reconstruction with texture.

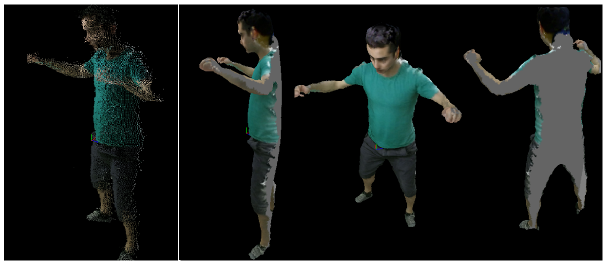

Fig. 6: Argyris sequence - Results using only two frontal captured views. Left: Raw point cloud without head. Right: Reconstruction with the head.

many reconstructed vertices may be invisible to all cameras. However, the texture information existing in the pre-scanned head model can be used to “paint” occluded or non-captured regions of the head. To seamlessly fuse this information, the following approach is used.

The 3D scanned head model is off-line projected onto five “virtual” RGB-D cameras, as shown in Fig. 3, i.e. to OpenGL cameras, with the OpenGL depth and color buffers serving as the corresponding depth and RGB images. The distance of these cameras from the heads centroid is set equal to 400mm, while their poses $\mathtt{RT}_k^0$, $k = 1, \ldots, 5$ are selected such that they cover completely the head. These cameras are supposed to have captured the head when it is positioned-oriented by $\mathtt{RT}_{\text{head}}^0 = I_4$ where I_4 is the unitary 4x4 matrix. Let now that for a frame n during the on-line reconstruction process, the head is positioned-oriented by $\mathtt{RT}_{\text{head}}^n$. The virtual cameras pose matrices are accordingly updated, i.e. $\mathtt{RT}_k^n = \mathtt{RT}_{\text{head}}^n \cdot \mathtt{RT}_k^0$. Given the OpenGL projection matrices and the pose matrices $\mathtt{RT}_k^n$, the “virtual” cameras’ RGB-D data are directly used in the employed texture mapping method (paragraph 2.4.3), in exactly the same manner as the Kinect cameras. The only difference is that the weight maps $W_k(\mathbf{u})$ (calculated also off-line) are multiplied by a small factor, equal to 0.1. This way, mainly the live captured texture is used at non-occluded regions. While, the weighted blending approach results into texturing without significant texture discontinuities.

3. EXPERIMENTAL RESULTS

Results with respect to the enhancement of the reconstruction by exploiting the pre-scanned head information are given in figures 4, 5 and 6. As can be verified, the pre-scanned head information can improve the reconstructed geometry: It helps to get more accurate reconstruction at the semantically important face region (the nose, eyes, etc. are better resolved) and at the non-captured regions, such as the top of the head. Additionally, it results into more complete texturing at the head region, as shown mainly in Fig. 6.

4. CONCLUSIONS

In this paper, a novel methodology for enhanced real-time 3D reconstruction of moving humans was presented. Providing 3D reconstructions of better quality is based on the idea of fusing the real-time captured 3D point-cloud data with a static pre-scanned mesh, adapted to the real-time positioning of the scene under consideration. Results presented in the experimental section provide evidence that the use of pre-scanned meshes can improve the final quality of the rendered 3D reconstructed model.

5. REFERENCES

[1] G. K. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, "A real time system for robust 3d voxel reconstruction of human motions," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 714–720.

[2] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

[3] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*. IEEE, 2012, pp. 524–530.

[4] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

[5] C. Nvidia, "Programming guide," 2008.

[6] I. García, S. Lefebvre, S. Hornus, and A. Lasram, "Coherent parallel hashing," in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6. ACM, 2011, p. 161.

[7] T. Reiner, S. Lefebvre, L. Diener, I. García, B. Jobard, and C. Dachsbacher, "A runtime cache for interactive procedural modeling," *Computers & Graphics*, vol. 36, no. 5, pp. 366–375, 2012.

[8] R. A. Newcombe, D. Fox, and S. M. Seitz, "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 343–352.

[9] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan, "Scanning 3d full human bodies using kinects," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, no. 4, pp. 643–650, 2012.

[10] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *Consumer Depth Cameras for Computer Vision*. Springer, 2013, pp. 71–98.

[11] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Estimating human 3d pose from time-of-flight images based on geodesic distances and optical flow," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*. IEEE, 2011, pp. 700–706.

[12] A. Maimone and H. Fuchs, "Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, 2011, pp. 137–146.

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

[14] M. Zollhöfer, M. Martinek, G. Greiner, M. Stamminger, and J. Süßmuth, "Automatic reconstruction of personalized avatars from 3d face scans," *Computer Animation and Virtual Worlds*, vol. 22, no. 2-3, pp. 195–202, 2011.

[15] N. Smolyanskiy, C. Huitema, L. Liang, and S. E. Anderson, "Real-time 3d face tracking based on active appearance model constrained by depth data," *Image and Vision Computing*, vol. 32, no. 11, pp. 860 – 869, 2014.

[16] A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[17] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH computer graphics*, vol. 19, no. 3. ACM, 1985, pp. 245–254.

[18] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[19] M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *Proc. 3rd Eurographics symposium on Geometry processing*, 2005.

[20] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory Probab. Appl*, vol. 14, no. 1, pp. 153–158, 1969.

[21] D. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, realistic full-body 3D reconstruction and texture mapping from multiple kinects," in *Proc. IEEE IVMSP*, June 2013.