REAL-TIME, REALISTIC FULL-BODY 3D RECONSTRUCTION AND TEXTURE MAPPING FROM MULTIPLE KINECTS

Dimitrios S. Alexiadis, Dimitrios Zarpalas and Petros Daras

Information Technologies Institute,
Centre for Research and Technology - Hellas,
6th km Charilaou - Thermi, GR-57001, Thessaloniki, Greece

ABSTRACT

Multi-party 3D Tele-Immersive (TI) environments, supporting realistic interaction among distant users, is the future of tele-conferencing. Real-time, full-body 3D reconstruction, an important task for TI applications, is addressed in this paper. A volumetric method for the reconstruction of watertight models of moving humans is presented, along with details for appropriate texture-mapping to enhance the visual quality. The reconstruction uses the input from multiple consumer depth cameras and specifically Kinect sensors. The presented results verify the effectiveness of the proposed methodologies, with respect to the visual quality and frame rates.

Index Terms— 3D reconstruction, texture mapping, real-time, Microsoft Kinect, volumetric fusion

1. INTRODUCTION

Future communication systems will enable multiple geographically distributed users to share common experiences, as if were in the same location. Such realistic inter-personal communications can be supported by the realization of multi-party 3D Tele-Immersive (TI) environments [1] (Fig. 1). In this paper, we deal with the challenging task of full 3D reconstruction of moving humans in real-time, an important task for TI applications. Reconstruction is achieved by the fusion of multiple RGB-Depth data, captured by multiple Microsoft Kinect sensors.

Many accurate methods for full-geometry 3D reconstruction from multiple passive RGB cameras can be found in the literature (e.g. [2]). Unfortunately, these are not applicable in real-time applications, since they require a processing time of several minutes per frame. Regarding methods that use active direct-ranging sensors (e.g. [3]), most of them present high accuracy but have not been used in real-time applications. Instead, they are applied off-line to combine range data captured by a single sensor. On the other hand, most of the relevant real-time TI-oriented approaches (e.g. [1, 4]), fuse partial 3D data only at the rendering stage, in order to synthesize intermediate 2D views for user-given viewpoints. In contrast, the proposed method produces a full-geometry 3D textured mesh that can be rendered using standard computer graphics. Several previous works on Kinect-based 3D reconstruction can be found [5, 4, 6]. In KinectFusion [5], the problem of 3D indoor static scenes reconstruction is addressed through the fast fusion of multiple depth scans from different positions, captured by a single hand-held Kinect sensor. The ReconstructMe system (`http://reconstructme.net/`) share similar ideas to off-line generate models from a single handheld Kinect. In [4], an efficient TelePresence-oriented system with 6 Kinects is presented. However, it combines the separate 3D meshes from each Kinect only at the rendering stage to produce intermediate stereo views, rather than producing a single 3D mesh model. In [6], we have presented a multi-Kinect system that can produce a single 3D mesh of the upper human body in real-time. In contrast to [6], the presented work fuses implicitly the information from all Kinects to produce water-tight models. In addition, the current paper presents ideas to improve the visual quality of the textured 3D reconstructions.

2. CAPTURING SETUP AND CALIBRATION

The capturing system is comprised of $K = 5$ Kinect sensors, connected on a single host PC, which features an Intel i7 processor (3.2GHz) and 8GB RAM, as well as a CUDA-enabled graphics card NVidia GTX 560. One sensor is placed horizontally at a height of 1.30m, to capture the front upper body. The remaining four sensors are placed vertically, at a height of approximately 1.80m, with a small angle (approximately 25^o) towards the floor to capture the whole human body. They are positioned on a circle of diameter 3.60m and all pointing to the center of the working volume. Given this spatial arrangement and the cameras' FoV, an "active" circle of diameter 2.40m is introduced, where the whole human body is captured.

In order to fully calibrate a single Kinect, we used the method of [7], which simultaneously estimates the depth and the RGB camera intrinsic parameters (with radial distortions), as well as the relative pose between them. Additionally, it

This work was supported by the EU funded IP project REVERIE under contract 287723.

Fig. 1. An advanced user is reconstructed from multiple Kinects (middle), a simple user from one Kinect (right) and autonomous avatars are used to populate the world (left).

estimates a depth distortion model for the depth sensor.

With respect to the external calibration of multiple Kinects, we developed a method that uses a custom-made calibration object with three intersecting large planar surfaces and exploits the input from the depth sensors: i) The calibration object is moved to various positions inside the working volume and captured simultaneously by all sensors; ii) For each captured frame, the three planar surfaces are detected in the depth images; iii) We initially perform a fast initial pairwise coarse calibration based on the normal vectors of the detected planes; iv) Then, refinement of the sensors' pose estimate is realized, by minimizing the mean squared distance of the reconstructed points on the three planes with the corresponding planes in a reference camera; v) Finally, a global, all-to-all and for all frames ICP optimization procedure is followed. If one has a good initial estimate for the camera poses (e.g. when the camera positions have been slightly changed), one can go directly to the last phase and without using the specific calibration object. Instead, he/she can capture any object of interest (e.g. a human) in a large number of static poses.

3. RECONSTRUCTION AND TEXTURE MAPPING

3.1. Preprocessing

- A weak 2D bilateral filter is applied to the depth maps to reduce Kinect measurements noise.
- A binary silhouette map $S_k(\mathbf{u})$ is generated from each depth image $k = 1 \dots K$, by segmenting out the foreground object of interest (captured human).We are interested in removing existing objects in the scene (e.g. floor, chairs) and reconstruct only the captured human. Therefore, before the human enters the scene, a robust "background" depth image is constructed by appropriately accumulating multiple depth frames. Then, the foreground object is segmented out by checking the absolute difference of the current frame from the accumulated background, considering a fixed threshold equal to 20mm.
- An associated confidence map is calculated: $C_k(\mathbf{u}) = \max\{d(\mathbf{u})/d_0, 1\}$, where $d(\mathbf{u})$ is the Euclidean 2D distance of pixel $\mathbf{u}$ to the background pixels and $d_0 = 20$ pixels is a predefined parameter. This kind of map is associated to the confidence of a depth measurement, accounting for the fact that Kinect measurements near object boundaries are noisy.
- A normal map, let $\mathbf{N}_k(\mathbf{u})$, is calculated by performing a simple terrain Step Discontinuity Constraint Triangulation (SDCT) [6] on the depth-image plane and assigning to each vertex (pixel) the vector normal to the plane introduced by the vertex and its connected neighbor vertices.
- The 3D bounding box of the foreground object is estimated and discretized into $N_X \times N_Y \times N_Z = 2^7 \times 2^8 \times 2^6$ voxels.

Table 1. Processing time/rate & number of GL Primitives

METHOD / ENHANCEMENT	PROCESSING TIME/RATE	NUMBER OF VERT./FACES
Separate meshes	78ms / 13fps	228,581 / 445,447
Volumetric simple	110ms / 9.1fps	43,431 / 84,802
+Mesh smoothing	131ms / 7.6fps	"
+Weighted text. map.	134ms / 7.5fps	"
+Color matching	170ms / 5.9fps	"

3.2. Surface reconstruction

We aim at computing an appropriate volumetric function $V(\mathbf{X})$, which is negative (empty) outside the captured solid, positive (occupied) inside the solid and (almost) zero near its surface. All of the K depth camera observations contribute in a weighted manner to $V(\mathbf{X})$. Therefore, a cumulative weight function $W(\mathbf{X})$ is also used. The employed approach is similar to the volumetric Signed Distance Function (SDF)-based one [3, 5], but takes into account the fact that opposite sides of the solid objects are captured by opposite Kinects and produces almost watertight models. A parameter μ is used in the algorithm and selected equal to 30mm. Very large values of μ results to interfering of surfaces on opposite sides of the object, while small values of μ will not robustly fuse the noisy depth data. In order to simplify the description of the algorithm, we use the "voxel-state" flag values EV and UOV to represent "Empty Voxels" and "Unknown-Occupancy Voxels", respectively. Additionally, the flag OSV is used to characterize a voxel that projects Outside all the Silhouette images $S_k(\mathbf{u}), k = 1 \dots K$.

1. Initialization: Set $V(\mathbf{X}) \leftarrow$ UOV and $W(\mathbf{X}) \leftarrow 0, \forall \mathbf{X}$.

2. Volume calculation: For each depth camera k sequentially, and for each voxel $\mathbf{X}$ in parallel CUDA threads:

- If $V(\mathbf{X}) =$ EV, leave it unchanged. Otherwise,
- Transform $\mathbf{X}$ to the camera's coordinate system, to obtain $\mathbf{X}_k = [X_k, Y_k, Z_k]^{\mathrm{T}}$ and project to the image plane to get the observed depth $D_k(\mathbf{u}_k)$, $\mathbf{u}_k = \Pi(\mathbf{X}_k)$, where $\Pi(\mathbf{X}_k)$ denotes the projection of $\mathbf{X}_k$ to the 2D depth image plane.

(a) If $\mathbf{X}_k$ projects outside the object's silhouette and $V(\mathbf{X}) =$ UOV, set $V(\mathbf{X}) \leftarrow$ OSV. Otherwise,

(b) Based on the observed depth, get the actual 3D point $\mathbf{X}_k^a$. Calculate the SDF $d_1 = \mathtt{sgn}(Z_k^a - Z) \cdot d(\mathbf{X}_k, \mathbf{X}_k^a)$, where $d(\mathbf{X}_k, \mathbf{X}_k^a)$ is the distance between $\mathbf{X}_k$ and $\mathbf{X}_k^a$.

Fig. 2. i) Volumetric reconstruction after appropriate background subtraction (no enhancement); ii) Smoothed model without texture; iii) After weighted texture mapping; iv) After color matching; v) 5 separate meshes, reconstructed with terrain triangulation on the image plane [6].

(c) If $d_1 < -\mu$, set $V(\mathbf{X}) \leftarrow \mathsf{EV}$. Otherwise,

(d) If $d_1 > \mu$, do nothing. Otherwise,

(e) Set:

$$V(\mathbf{X}) \leftarrow \frac{W(\mathbf{X}) \cdot V(\mathbf{X}) + w_k(\mathbf{X}) \cdot d}{W(\mathbf{X}) + w_k(\mathbf{X})}$$
$$W(\mathbf{X}) \leftarrow W(\mathbf{X}) + w_k(\mathbf{X}), \quad (1)$$

where $w_k(\mathbf{X})$ is appropriate weight, discussed later.

3. Finalization: Replace the flag values with scalars:

$$V(\mathbf{X}) \leftarrow \quad \mu \quad \forall \mathbf{X} : V(\mathbf{X}) = \mathsf{UOV}$$
$$V(\mathbf{X}) \leftarrow -\mu \quad \forall \mathbf{X} : V(\mathbf{X}) = \mathsf{EV} \text{ or } \mathsf{OSV}$$

The surface is reconstructed by the extraction of the zero-level isosurface of $V(\mathbf{X})$, using the marching cubes algorithm [8]. The volumetric reconstruction part of this subsection, was implemented using the CUDA processing architecture.

Appropriate weights: The selection of the weight $w_k(\mathbf{X})$ have to be based on the "quality of the corresponding measurement, i.e. the depth measurement on the pixel $\mathbf{u}_k = \Pi(\mathbf{X}_k)$. Depth measurements near the object boundaries are noisy and this information is contained in the confidence maps $C_k(\mathbf{u})$ (subsection 3.1). Additionally, the "quality of a depth measurement strongly depends on the "viewing" angle with which a Kinect captures the actual surface. Therefore, $w_k(\mathbf{X})$ should additionally depend on the angle between the surface's normal vector at point $\mathbf{X}$ and the unit vector $\mathbf{X}_k^1 = -\mathbf{X}_k/||\mathbf{X}_k||$ that connects $\mathbf{X}_k$ with the depth camera. According to the above, the weights are calculated from: $w_k(\mathbf{X}) = \max\{\mathbf{X}_k^1 \bullet \mathbf{N}_k(\mathbf{u}), 0\} \cdot C_k(\mathbf{u})$ where $\mathbf{u} = \Pi_k(\mathbf{X}_k)$ and "$\bullet$" denotes the inner product, while the normals $\mathbf{N}_k(\mathbf{u})$ have been calculated during preprocessing (subsection 3.1).

Post smoothing: A fast mesh smoothing operation is applied to the reconstructed mesh, which calculates the average position of each vertex with its connected neighbor vertices.

3.3. Texture mapping

There are vertices in the reconstructed model that are visible in more than one RGB cameras. Therefore, colors from more than one Kinect have to be combined in a weighted manner to produce a single color per vertex. Notice that the weights $w_k(\mathbf{X})$ themselves contain visibility information ($w_k(\mathbf{X})$ are normally zero at those $\mathbf{X}$ that are invisible in the k-th camera). Additionally, they are small near the object boundaries, where inaccurate Depth-to-RGB camera registration (calibration) might lead to color-mapping artifacts (e.g. color of the background assigned on the reconstructed foreground object). Therefore, we use the already calculated weights $w_k(\mathbf{X})$ themselves in order to speed-up the color-mapping process. An additional important fact that have to be taken into account is that volumetric-based 3D reconstruction methods produce generally a relatively low number of number triangles and vertices (depending on the voxels number), lower than the number of pixels in the original 2D domain. Therefore, a color-per-vertex approach will lead to color aliasing, producing low visual quality. To deal with this, we employ full texture mapping by projecting vertices onto the RGB images to find UV coordinates and assigning multiple textures in each triangle with OpenGL multi-texture blending.

3.3.1. Color matching in multiple RGB cameras

The Kinect RGB sensor allows only automatic white-balance and exposure control. This may yield color values that vary between adjacent Kinect RGB views. We use an on-the-fly technique that builds color matching/correction functions during reconstruction (in a parallel working thread). Such a function aims at minimizing the color difference between pairs of pixels in two cameras that capture (approximately) the same 3D points on diffuse surfaces. The approach is similar to the one used in [4], with some significant differences however.

Obtaining pixel correspondences: Given the reconstructed 3D model, we project the 3D vertices onto the RGB image planes to get color correspondences in adjacent RGB cameras. In order to build robust color matching functions, we accumulate color correspondences in multiple frames. Otherwise, there may be too few correspondences, especially if the captured surfaces have a limited range of colors.

Fig. 3. Example reconstructions, in various poses.

Fig. 4. Example reconstructions, in various poses.

Estimating color matching functions: We found in practice that an RGB-separately approach [4] may lead to color artifacts when the range of colors in the foreground object is limited (e.g. a specific color channel is missing). Therefore, we preferred to work in the HSV color-space, which is also closer to the human visual perception. Moreover, it is expected that the Hue (H) component is not affected by the exposure control. We also experimentally found that the Saturation (S) component is only slightly affected and mainly the V (Value) component is practically modified. Therefore, we build a single color mapping function for the V component. A linear model is used, found by robust regression on the data.

3.4. Reconstruction results

The improvement of visual quality by the proposed steps is demonstrated in Fig. 2. A comparison with terrain triangulation on the depth-image [6] is also given. Results with respect to the processing time and the average number of generated GL primitives are presented in Table 1. These were obtained for a whole sequence of 500frames, with a representative frame as in Fig. 2. With much fewer GL primitives the visual quality is superior and the real-time constraints are still satisfied. Finally, textured results are also given in Figs 3, 4.

4. CONCLUSIONS

We presented a complete system for realistic 3D reconstruction of moving humans, which can used in real-time applications, such as 3D TI. The system is composed of multiple Kinect sensors. A volumetric reconstruction method was described, which when implemented in CUDA can run in real-time. The reconstruction method, together with the presented texture mapping methodology, can produce quite accurate and realistic results, even under the real-timeness constraints.

5. REFERENCES

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

[2] Y. Furukawa and J. Ponce, "Carved visual hulls for image-based modeling," *IJCV*, vol. 81, pp. 53–67, 2009.

[3] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH '96*, 1996.

[4] 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*.

[5] Richard Newcombe, et. al., "KinectFusion: Real-time dense surface mapping and tracking," in *10th IEEE ISMAR*, 2011.

[6] D. 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, pp. 339–358, Feb. 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] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21(4), pp. 163–169, July 1987.