Fast and smooth 3D reconstruction using multiple RGB-Depth sensors

Dimitrios Alexiadis[1], Dimitrios Zarpalas[2], Petros Daras[3]
Information Technologies Institute, Centre for Research and Technology - Hellas
6th Km Charilaou-Thermi, Thessaloniki, Greece

[1]dalexiad@iti.gr, [2]zarpalas@iti.gr, [4]daras@iti.gr

***Abstract*—In this paper, the problem of real-time, full 3D reconstruction of foreground moving objects, an important task for Tele-Immersion applications, is addressed. More specifically, the proposed reconstruction method receives input from multiple consumer RGB-Depth cameras. A fast and efficient method to calibrate the sensors in initially described. More importantly, an efficient method to smoothly fuse the captured raw point sets is then presented, followed by a volumetric method to produce watertight and manifold meshes. Given the implementation details, the proposed method can operate at high frame rates. The experimental results, with respect to reconstruction quality and rates, verify the effectiveness of the proposed methodology.**

***Index Terms*—Tele-immersion, 3D reconstruction, real-time, Microsoft Kinect**

I. Introduction

Realistic inter-personal communications can be supported by the realization of multi-party 3D Tele-Immersion (TI) [1] environments. The problem of real-time robust 3D reconstruction of humans, an important and challenging task for TI applications, is addressed in this paper. Although accurate 3D reconstruction methods from passive RGB cameras can be found in the literature (e.g. [2], [3]), they are not applicable in TI applications, since they require a processing time of several minutes per frame. Other, mainly visual hull-based methods [4], are quite fast, yet they lack the ability to reconstruct concavities. Regarding methods that use active direct-ranging sensors (e.g. [5]), they have not been used in real-time applications; instead, they are applied off-line to combine range data captured by a single sensor.

Most of the relevant real-time TI-oriented approaches [1], [6], fuse partial 3D data only at the rendering stage, in order to synthesize intermediate 2D views. In [1], a high quality TI system is described, including an accurate method for the generation of multiple depth-maps from stereo clusters, which are then combined at the rendering stage to synthesize views for given viewpoints. A similar TI system is described in [6], where the depth-maps are captured by multiple MS Kinect sensors. On the other hand, a Kinect-based 3D reconstruction system is presented in [7] that produces a single full 3D mesh, independently to the rendering stage. However, the explicit mesh "zippering" method of [7] may reject a significant portion of the captured information during the described "hard" procedure of decimating overlapping mesh regions. Moreover, it does not produce watertight and manifold meshes. On the contrary, in this paper, the captured data are fused in a weighted manner during an initial smoothing step, based on the confidence of the corresponding depth measurements. Data are then implicitly fused to generate manifold watertight meshes, resulting in superior visual quality, at higher frame rates, as experimentally demonstrated.

II. Capturing setup and calibration

The RGB-Depth data used in this paper were captured by 5 Kinect sensors, placed in a circular spatial arrangement that provide full-body 360^o coverage of a human. In the following, a method for the calibration of such a multiple-Kinects system is proposed. We also present experimental results using the multiple-Kinects dataset of the Huawei/3DLife ACM Multimedia Grand Challenge 2013 [8], captured by a similar circular configuration. The reader is referred to [8] for details.

In order to accurately estimate the internal parameters of each single Kinect, the method of [9] is used. For the external calibration of the system, a simple, yet efficient, method is proposed that makes use of a large planar chessboard surface. The idea is based on the detection of a large number of 3D point correspondences in sets of sensors. Then, the extrinsic parameters are estimated for all cameras in an all-to-all manner. The method is summarized as follows.

- A large number of frames is initially captured, with the calibration pattern being visible to at least two sensors.
- For each RGB camera, the chessboard corners are detected (where visible). The Open CV library is used.
- For each sensor that captures the chessboard pattern, the dominant plane is detected: a) The depth data are thresholded and a raw 3D point cloud is generated, which is then transformed to the coordinate system of the RGB camera; b) The dominant sample-consensus plane model is estimated using RANSAC, with an inliers distance threshold equal to 2cm. The PCL library [10] is used.
- The 2D chessboard corners are then backprojected to the estimated 3D plane and a set of 3D features is extracted.
- Since the chessboard corners are indexed, 3D points correspondences in pairs of sensors are established.

Let the number of chessboard corners, i.e. feature points per view per frame, be I (I=9×5 in our pattern). Let also K and N denote the number of sensors and number of frames, respectively. Then, the i-th 3D feature point for frame n and view k is denoted as $\mathbf{v}^i_{k,n}$. The corresponding 3D point, registered in the global coordinate system using the matrix T_k (to be estimated), is denoted as $\mathbf{p}^i_{k,n} = \mathsf{T}_k \cdot \mathbf{v}^i_{k,n}$. Let finally, $\mathbf{P}_{k,n}$ denote the $I \times 3$ matrix containing the points $\mathbf{p}^i_{k,n}$.

The exploited method for the estimation of the unknown external calibration matrices, in an all-to-all manner, is based on the idea of Global Procrustes Analysis (GPA) [11]. In the initial formulation of GPA theory, it is assumed that multiple point-sets contain the same points, which are expressed in K different 3-D coordinate systems. In our problem however, not all I points are present (visible) in each view. Additionally, we have points-sets and correspondences for multiple frames. Therefore, in order to take into account the above issues, the following modified GPA objective function is realized: $e(\mathsf{T}_1, \mathsf{T}_2, \dots, \mathsf{T}_K) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathtt{tr}\left((\mathbf{P}_{k,n} - \mathbf{C}_n)^{\intercal} \mathsf{B}_{k,n} (\mathbf{P}_{k,n} - \mathbf{C}_n) \right)$, where $\mathsf{B}_{k,n}$ is a binary diagonal matrix of size $I \times I$, indicating the "visibility" of the feature points in the k-th view and n-th frame. Additionally, $\mathbf{C}_n$ is a $I \times 3$ matrix containing the geometrical centroid of the registered points for frame n and is calculated from: $\mathbf{C}_n = \left(\sum_k \mathsf{B}_{k,n}\right)^{-1} \left(\sum_k \mathsf{B}_{k,n} \mathbf{P}_{k,n}\right)$.

With the above definitions, the unknown matrices T_k are found in a few iterations of the following steps: 1) The centroid matrices $\mathbf{C}_n$ are calculated; 2) The registration matrices T_k are updated by minimizing $e(\mathsf{T}_1, \mathsf{T}_2, \dots, \mathsf{T}_K)$, 3) The point-sets are registered using the estimated transformations.

III. 3D RECONSTRUCTION

A. Raw point-cloud reconstruction

In a preprocessing step, a binary silhouette map $S_k(\mathbf{u}) \in \{0, 1\}$ of the foreground object (captured human) is generated for each depth map $D_k(\mathbf{u})$, $\mathbf{u} = (u_x, u_y)^{\intercal}$, $k = 1, \dots, K$. The simplest approach to segment the foreground object is thresholding of the depth-maps. Then, for each "foreground" pixel $\mathbf{u} : S_k(\mathbf{u}) = 1$ on the k-th depth-map, a "raw" 3D point is reconstructed: $\mathbf{X}(\mathbf{u}; k) = \Pi_k^{-1}\{\mathbf{u}, D_k(\mathbf{u})\}$, where Π_k^{-1} defines the back-projection operation according to the estimated depth-camera intrinsic and extrinsic parameters. We use the notation $\mathbf{X}(\mathbf{u})$ to highlight that each reconstructed 3D point is associated with a pixel on the depth map. Additionally, for each "foreground" pixel the raw 3D normals $\mathbf{N}(\mathbf{u}; k)$ are estimated by using simple terrain Step Discontinuity Constraint Triangulation (SDCT) [7] on the depth-image plane. Additionally, a weak 2D moving average filter of diameter 3pixels, is applied to smooth the noisy normal estimates.

B. Confidence-based 3D smoothing

The reconstructed 3D points and their associated normals are noisy. Therefore, the overlapping surfaces from adjacent sensors contain a large portion of overlapping "Z-fighting" regions, which introduce geometrical and visual artifacts [7]. To handle this, [7] applied an iterative procedure that decimates overlapping surface regions from different sensors, until they just meet and then "zipper" the surfaces at the boundaries. Based on our observation that this "hard" approach of surface decimation may reject a significant portion of information and does not make use of the corresponding measurements' "quality", we propose a weighted smoothing approach for dealing with the overlapping adjacent surface regions.

Confidence values: The objective here is to extract a confidence value for each reconstructed 3D point $\mathbf{X}(\mathbf{u}; k)$ and its associated normal. In practice, it has been observed that the Kinect depth measurements near the foreground object's boundaries are noisy. In order to exploit this observation, an associated confidence map $C_k^1(\mathbf{u}) \in [0, 1]$ is calculated from the binary silhouette maps $S_k(\mathbf{u})$. A fast approach to calculate such a confidence value for a pixel $\mathbf{u}$ is to count the number of neighbors that belong to the foreground. This is implemented efficiently by a moving average filter on $S_k(\mathbf{u})$, of radius 30pixels in our experiments. Moreover, the "quality" of a depth measurement depends on the "viewing" angle, i.e. the angle between the Kinect's line of sight, defined by the unit vector $\hat{\mathbf{X}} = -\mathbf{X}/||\mathbf{X}||$, and the surface normal at point $\mathbf{X}$. Based on this, a confidence value for pixel (vertex) $\mathbf{u}$ is computed from $C_k^2(\mathbf{u}) = \max\{< \hat{\mathbf{X}}(\mathbf{u}; k), \mathbf{N}(\mathbf{u}; k) >, 0\}$, where $< \cdot, \cdot >$ denotes the inner vector product. The total confidence map is calculated from the product $C_k(\mathbf{u}) = C_k^1(\mathbf{u}) \cdot C_k^2(\mathbf{u})$.

Smoothing: For each "raw" point $\mathbf{X}$, all 3D neighbors inside a sphere of fixed radius, set equal to 40mm in our experiments, are found. To avoid time-consuming searching for neighbors in 3D space, we exploit the fact that each 3D point is associated with a pixel on the depth map, thus 3D neighborhoods define 2D neighborhoods on it. We use the approach that is summarized in Fig. 1.

Let the set of 3D neighbors of $\mathbf{X}$ be denoted as $\mathcal{N}(\mathbf{X})$. Then, we calculate a new vertex position as the weighted average: $\mathbf{X}' = \sum_{\mathbf{X}_n \in \mathcal{N}(\mathbf{X})} C(\mathbf{X}_n) \cdot \mathbf{X}_n / \sum_{\mathbf{X}_n \in \mathcal{N}(\mathbf{X})} C(\mathbf{X}_n)$, where $C(\mathbf{X})$ defines the confidence value for point $\mathbf{X}$. Exactly the same approach is used to calculate a "smooth" 3D normal $\mathbf{N}'(\mathbf{X})$, as the weighted average of the corresponding point normals. Instead of replacing $\mathbf{X}$ with $\mathbf{X}'$, it is preferable to move the 3D point along only the surface normal vector, to avoid bad spacing of the points. We therefore calculate the projection of the displacement $d\mathbf{X} = \mathbf{X}' - \mathbf{X}$ onto the normal vector $\mathbf{N}'(\mathbf{X})$, i.e. $d\mathbf{X}' =< d\mathbf{X}, \mathbf{N}'(\mathbf{X}) > d\mathbf{X}$. The new set of 3D points and normals $\mathbf{X}'' = \mathbf{X} + d\mathbf{X}'$ and $\mathbf{N}'$, respectively, constitute the output of the proposed confidence-based smoothing approach.

C. Final scalable reconstruction and triangulation

The bounding box of the foreground object is initially estimated. To remove outliers, the 5% min and max percentiles of the raw 3D points positions are rejected. The box is discretized into $2^r \times 2^{r+1} \times 2^r$ voxels, where r defines the targeted volume resolution. The objective is to calculate a characteristic volumetric function $A(\mathbf{q})$ ($\mathbf{q}$ stands for voxel), where the 3D surface is implicitly defined as the isosurface at an appropriate level. For this purpose, we follow a Fourier Transform (FT)-based approach [12], which is summarized below from an implementation point-of-view, highlighting a few proposed modifications.

The point normals are initially splatted to obtain the gradient vector field $\mathbf{V}(\mathbf{q})$. Instead of simply clapping a sample to the containing voxel, the normals are smoothly splatted according

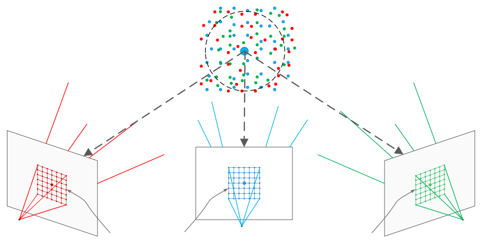

Fig. 1. Fast search for neighbor 3D vertices

TABLE I
AVERAGE RECONSTRUCTION TIME AND RATES

PREPROCESS	RAW RECONSTR	SMOOTHING	TOTAL
17 msec	18 msec	41 msec	76 msec

	RESOLUT. $r=6$	RESOLUT. $r=7$
FFT RECONSTR.	22 msec	79 msec
TEXTURE MAPPING	1 msec	2 msec
PCL RECONSTR.	76 msec	
TOTAL TIME	**99** msec	**157** msec
RECONSTR. RATE	**10.1** fps	**6.37** fps
RECONSTR. RATE OF [7]	**3.52** fps	

to: $\mathbf{V}(\mathbf{q}) = \sum_{\mathbf{X}} w(\mathbf{X};\mathbf{q}) \cdot \mathbf{N}(\mathbf{X}) / \sum_{\mathbf{X}} w(\mathbf{X};\mathbf{q})$, where $w(\mathbf{X};\mathbf{q})$ are appropriate weights based on the Euclidean distance of $\mathbf{X}$ from the center of voxel $\mathbf{q}$. In order to speed up execution, the weights $w(\mathbf{X};\mathbf{q})$ outside the 27-neighbors region around the central voxel are considered equal to zero.

The vector field is transformed to the 3D FT domain $\hat{\mathbf{V}}(\boldsymbol{\omega})$ and multiplied (convolution in the FT domain to speedup calculations) with the integration filter $\hat{\mathbf{F}}(\boldsymbol{\omega}) = j\boldsymbol{\omega}/||\boldsymbol{\omega}||$, where $j = \sqrt{-1}$ and $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^{\intercal}$. This is performed separately for each X,Y and Z component. The function $A(\mathbf{q})$ is calculated by applying inverse 3D FT on the integrated (filtered) vector field and addition of its X, Y, Z components.

The final 3D mesh surface (vertex position, normals and connectivity) is obtained by the extraction of the isosurface $A(\mathbf{q}) = L$ using the marching cubes algorithm [13], where L is an appropriate level calculated as the average value of $A(\mathbf{q})$ at the input sample locations $\mathbf{X}$.

D. *Multi-texture mapping*

In case of rendering dense point-clouds (subsection III-A), colors from multiple RGB cameras are fused to produce a single color per vertex. In case of rendering low-poly meshes (produced by the method of subsection III-C), the vertices are projected onto the RGB images to find UV coordinates and each triangle is assigned multiple textures, which are rendered with OpenGL multi-texture blending. As appropriate weights for the fusion of colors/textures, we use the already calculated weights $C_k(\mathbf{X})$ to speed up execution. This approach is sensible since: a) the confidence values $C_k(\mathbf{X})$ contain visibility information; b) they are small near the object boundaries, where inaccurate Depth-to-RGB camera calibration often leads to color-mapping artifacts; c) the confidence values incorporate the practical observation that the captured color quality and detail depend on the “viewing” angle.

IV. EXPERIMENTAL RESULTS

The experiments ran on a PC with an i7 processor (3.2GHz), 8GB RAM and a CUDA-enabled NVidia GTX 560.

Implementation details: The reconstruction approach was implemented in CUDA [14] to exploit the parallel computing capabilities of GPU, since most of its stages involve pixel-wise

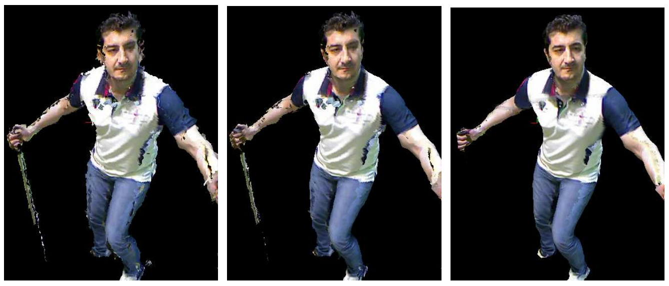

Fig. 3. Skiing: Raw, smoothed and final reconstruction.

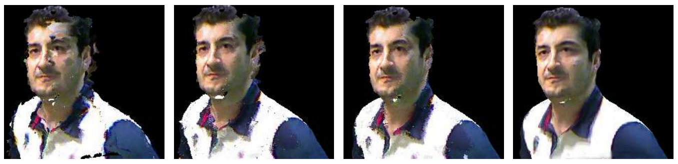

Fig. 4. Skiing - From left to right: (a) Raw SDCT reconstruction, (b) Output of [7], (c) Output of [7] after applying the proposed weighted smoothing method, (d) proposed watertight reconstruction.

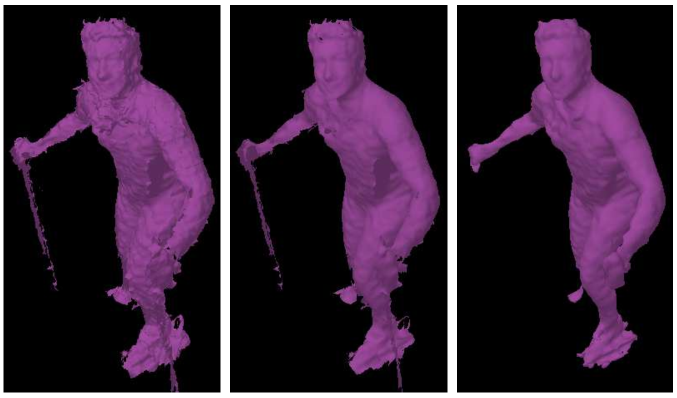

Fig. 2. Skiing: Raw reconstruction, smoothed/“stitched” reconstruction and final reconstruction.

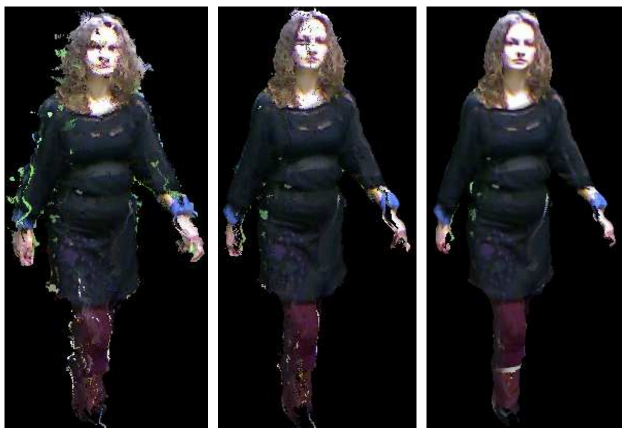

Fig. 5. Xenia: Raw point-set, the output of smoothing operation and the final reconstruction.

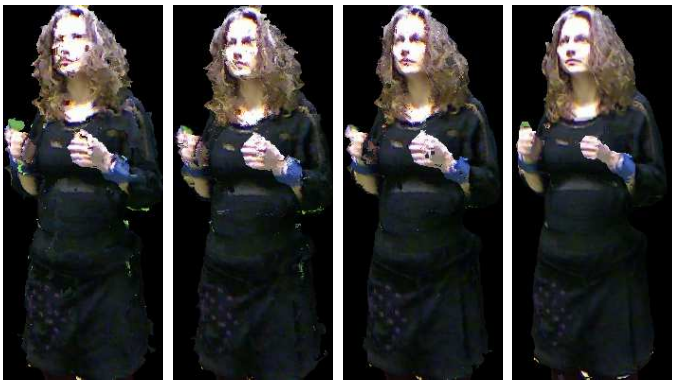

Fig. 6. Xenia - From left to right: (a) Raw SDCT reconstruction, (b) Output of [7], (c) Output of [7] after applying the proposed weighted smoothing method, (d) proposed watertight reconstruction.

or voxel-wise calculations. Additionally, the CUFFT CUDA library was used for fast FT calculations.

“Skiing” sequence: The input dataset contains a human on a ski simulator. Reconstruction results for a specific frame are given in Fig. 2. In this figure, the reconstructed raw points are illustrated (rendered using triangles generated by SDCT [7]), along with the output of the confidence-based smoothing operation and the volumetric reconstruction. The noise in the raw point-set, as well as the effect of “Z-fighting” surfaces is obvious. On the other hand, the smoothed point-sets define a smooth surface. However, the triangular surface obtained by terrain SDCT, although visually more pleasant, is non-manifold and contains cracks and holes. Therefore, a final watertight and manifold surface is reconstructed using the method of subsection III-C. Textured reconstruction results are also given in Fig. 3. Notice that very thin objects, such as the skiing poles, may not be reconstructed due to the noisy nature of the captured input and the finite voxel resolution, which is an inherent characteristic of any volumetric method. However, it is clear that the final reconstruction results contain much less artifacts and are visually superior.

Using the same sequence, Fig. 4 provides some comparative results with the “zippering” method of [7]. One can verify that a) the proposed methods produces reconstructions with fewer artifacts and moreover, b) the method of [7] can benefit from the proposed weighted smoothing method.

Table I summarizes the execution time results obtained for 200 frames of the sequence. One can observe that high reconstruction rates, close to 10 fps, can be achieved. Additionally, the frame-rates achieved with an optimized GPU implementation of [7] are given. Notice that the achieved frame-rates are slightly smaller than those reported in [7], because more Kinects are used in this paper (5 vs 4) and the overlap between adjacent views is larger.

ACM multimedia GC 2013 sequences: Experimental results using the dataset of [8] are also given. Figure 5 presents results for a specific frame of the “Xenia” sequence. The raw textured point-set is presented on the left, followed by the output of the smoothing operation and the final reconstructed mesh. Fig. 6 provides comparative results with the “zippering” method of [7]. Finally, Fig. 7 presents the reconstruction results for two frames of the “Stavroula” sequence.

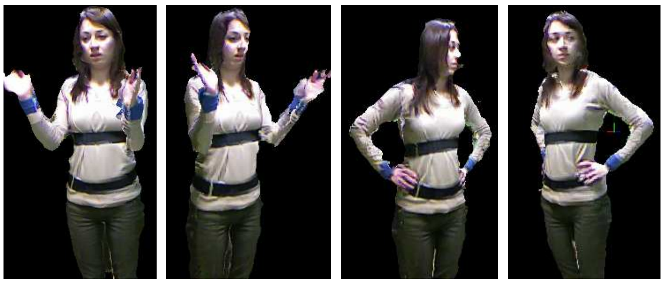

Fig. 7. Stavroula: Results for two frames and two view-points

V. Conclusions

This paper presented a complete full-body 3D reconstruction system using multiple consumer RGB-Depth sensors, appropriate for real-time applications, such as 3D TI. A fast and efficient method to smooth in a weighted manner the separate input raw point sets was presented, followed by a volumetric method to produce watertight and manifold meshes. The whole reconstruction process can operate in real-time when implemented in CUDA. As verified, the method produces quite accurate and realistic results, even under the real-time constraints, compared to other works.

Acknowledgment

This work was supported by the EU-funded project 3D-Live under contract 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 Transactions on Multimedia*, vol. 13(3), June 2011.

[2] G. Vogiatzis, C. Hernandez, P. Torr, and R. Cipolla, “Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency,” *IEEE Trans. Pattern Anal Mach. Intell.*, vol. 29(12), pp. 2241 –2246, 2007.

[3] Y. Furukawa and J. Ponce, “Carved visual hulls for image-based modeling,” *Int. Journal of Computer Vision*, vol. 81, p. 5367, 2009.

[4] J.-S. Franco and E. Boyer, “Efficient polyhedral modeling from silhouettes,” *IEEE Trans. Patt. Anal Mach. Intell.*, vol. 31, pp. 414–427, 2009.

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

[6] 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*.

[7] 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, pp. 339–358, 2013.

[8] Online: http://mmv.eecs.qmul.ac.uk/mmgc2013/.

[9] 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.

[10] “The Point Cloud Library,” Online: http://pointclouds.org/.

[11] R. Toldo, A. Beinat, and F. Crosilla, “Global registration of multiple point clouds embedding the generalized procrustes analysis into an ICP framework,” in *3DPVT’10*, 2010.

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

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

[14] “CUDA,” Online: www.nvidia.com/object/cuda_home_new.html.