REAL-TIME MULTI-VIEW VOLUMETRIC RECONSTRUCTION OF DYNAMIC SCENES USING KINECT V2

Andrej Satnik, Ebroul Izquierdo

Multimedia & Vision Group
School of Electronic Engineering and Computer Science
Queen Mary University of London
andrej.satnik@qmul.ac.uk, ebroul.izquierdo@qmul.ac.uk

ABSTRACT

A key challenge when displaying and processing sensed real-time 3D data is efficiency of generating and post-processing algorithms in order to acquire high quality 3D content. In contrast, our approach focuses on volumetric generation and processing volumetric data using an efficient low-cost hardware setting. Acquisition of volumetric data is performed by connecting several Kinect v2 scanners to a single PC that are subsequently calibrated using planar pattern. This process is by no means trivial and requires well designed algorithms for fast processing and quick rendering of volumetric data. This can be achieved by fusing efficient filtering methods such as Weighted median filter (WM), Radius outlier removal (ROR) and Laplace-based smoothing algorithm. In this context, we demonstrate the robustness and efficiency of our technique by sensing several scenes.

Index Terms — volumetric, render, Kinect v2, voxel, multi-view, real-time

1. INTRODUCTION

Dynamic reconstruction for digital representation and description of 3D objects has been at the centre of research for many years. Nowadays, 3D sensorial devices are becoming pervasive and several approaches have emerged for visualization and reconstruction.

There is a key difference between how the data is acquired and the method used to reconstruct it. Accelerated direct volume rendering (DVR) on consumer platforms is nowadays the standard approach for rendering 3D grids of voxels storing a scalar value representing density. The volume ray caster then generates a ray through the 3D grid for every pixel in the image plane, samples the voxel data along the ray, and composites the colour information given by the transfer function [1]. In our approach traditional colour estimation of voxels is replaced by colour mapping method to compute final pixel colour. Volumetric representation of data has been also studied mainly with relation to recognition of volumetric data [2][3] and rendering [4][5][6][7].

Additionally, low-cost commercially available hardware and software for 3D real-time reconstruction the Microsoft Kinect v2 and corresponding SKD is a popular choice. However, it is not designed for multi-view multi-Kinect volume reconstruction using a single conventional PC. The methodology of connection and hardware settings of multiple Kinects v2 are described in our previous work [8].

In this paper, real-time reconstruction is presented using by fusing filtering techniques such as Radius Outlier Removal filter (ROR), Weighted Median filter (WM) and Laplacian-based smoothing algorithm (LS). The main novelty of the comprehensive approach presented in this paper does not reside in any of the individual components but rather on an integrated low-cost hardware with Kinect v2 devices and software suite able to deliver volumetric representation of moving objects. The performance of the introduced system has been assessed in a real-time application and it is presented in section 5.

2. RELATED WORK

Early solutions addressing multi-view reconstruction by Kinect sensors suffered from severe drawbacks derived from the limitations of the available sensing hardware.

As stated in our previous research [8], the number of Kinects v2 sensors that can be connected to a single PC is limited by the number of internal USB buses in the PC system. A single bus can handle multiple Kinect streams, as long as the high-speed serial computer expansion bus card (PCI-Express) is able to cope with the correspondingly data transfer stream and USB Filesystem (USBFS) memory buffer has predefined bandwidth for controller's firmware.

Several papers have been focused on multi-view real-time reconstruction. A multi-Kinect system to seamlessly render a scene from a wide range of angles was presented in [7]. The main drawback of solutions relates to using expensive hardware configuration and Kinect V1 sensors with lower transfer bandwidth and non-constant accuracy [9]. 3D reconstruction also features prominently in previous works. For instance, in [10] and [11] system using multiple Kinect v2 device for reconstruction of dynamic scenes. Unfortunately, these solutions are using separate PCs to distribute computational load among multiple devices. Another challenge with multi-view sensing lays in filtration and correction of sensed data. For computationally demanding algorithms [12], the frame rate becomes too low to achieve real-time processing as described in [2].

The calibration problem has been also studied thoroughly and a large number of related publications are available. In our previous work [8], calibration technique was presented receiving decent results with non-iterative operations.

3. EXTRINSIC CALIBRATION

In the extrinsic calibration stage, we separately calculate extrinsic parameters for each device using by double-sided planar pattern to determine the location of the camera in the scene. A camera can be described by pinhole model. A projection from the world coordinates $P = [X, Y, Z]^T$ to the image coordinates $p = [u, v]^T$

978-1-5386-6125-3/18/$31.00 ©2018 Crown

can be represented as follows:

$$p \sim K(RP + \vec{t}) \tag{1}$$

where K is the camera intrinsic matrix, R a 3×3 orthonormal matrix representing the cameras orientation, and t a representing its position. Extrinsic parameters correspond to rotation and translation vectors which translates coordinates of a 3D point to a coordinate system. With the extrinsic matrix we get the orientation between the two Time-of-Flight (ToF) cameras. Therefore, for given colour image and undistorted depth map produced by Kinect v2 representative corner points were detected for one pair of devices. In the next step, we find a covariance matrix H between two sets of 3D points $Q_A = (x, y, z)$ and $Q_B = (x, y, z)$ by expressing:

$$H = \sum_{i=1}^{N} (Q_A^i - C_A)(Q_B^i - C_B), Q_A \neq Q_B \tag{2}$$

where N is number of corner points and CA, CB are centroids of QA and QB. Using Singular Value Decomposition (SVD) [13] rotation matrix R can be expressed:

$$H = U\Sigma V^T \tag{3}$$

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & |VU^T| \end{bmatrix} U^T \tag{4}$$

where U is an orthogonal matrix whose columns are the eigenvectors of HH^T, V is an orthogonal matrix whose columns are the eigenvectors of $H^T H$, and Σ is a diagonal matrix represented by singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. The determinant corrects rotation matrix R to ensure a right-handed coordinate system. The translation vector $\vec{t}$ is calculated as the difference of centroids C_A and C_B.

In our final configuration, we captured a few image frames of the calibration double-sided pattern for each Kinect pair to calculate rotation matrix R and translation vector $\vec{t}$ where each pair representing 3D correspondences (Fig.3). With a known extrinsic matrix from every camera we can assign the orientation among each other's. Nevertheless, we use multiple sets of images and interpolate data with previous calibrations to increase precision of relative pose between devices.

4. POST-PROCESSING APPROACHES FOR VOLUME GENERATION

The quality of the depth image is a key factor for volume generation. The infrared projector of Time-of-Flight cameras (ToF) emits the pseudo random pattern light by a diffractive mask [9], which cause a distortion in the depth channel generating distorted depth data. This in turn leads to inaccurate volume and presentation of "jumping" voxels. Our goal is to improve the captured raw points by using a several existing filtering approaches.

Weighted median filter (WM) is well known filtering approach with aim to fill missing depth information. For each zero-pixel weighted median was calculated within local neighbourhood. Additionally, Radius outlier removal filter (ROR) removes "flying pixels" outside of clusters with defined radius. Due to perspective projection of ToF sensor, the distance between points is increasing. Therefore, the radius r_n of ROR is defined accordingly:

$$r_d = z_d \frac{0.5 - c_x}{f_x} - z_d \frac{2r_n + 0.5 - c_x}{f_x} \tag{5}$$

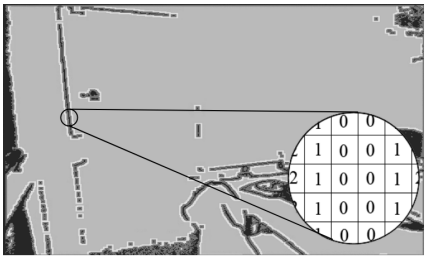

Figure 1: Thinning algorithm with two iterations.

where r_n represents size of window in pixels and c_x and f_x are intrinsic parameters of camera. Furthermore, the Laplacian-based smoothing (LS) algorithm was used for each non-zero pixel of depth map. Similarly as WM, we weighted neighbour zero-value pixels $w_{p=0} = 0$. Therefore, new pixel value p in local neighbourhood X can be calculated as:

$$p = \frac{1}{N_{p \neq 0}} \sum_{i=0}^{X_p} p_i w_i \tag{6}$$

where $N_{p \neq 0}$ represents number of non-zero pixels in neighbourhood X_p.

In the next step, we used ray-casting algorithm to produce volume content casting segments of rays passing through depth maps. This method is based on simple projection of undistorted depth map to generate 3D volume P_{ijk} to as:

$$P_{ijk} = (RD_o + \vec{t}) \tag{7}$$

If i, j represents undistorted coordinates of pixel p on depth map and R and $\vec{t}$ are extrinsic parameters of calibrated system described in Section 3 then $D_o = (i, j, D_p)$. In Fig. 3 is shown approach of projection depth map into 3D volume.

During the phase of volume generation, we used the thinning algorithm [14] to label depth values with weights around edges. This approach has advantages for overlapping parts to set colour values from more accurate regions of depth map. At first, binary mask D_L numbers was calculated.

$$D_L^p = \begin{cases} 1, & \text{if } D_p > 0. \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The new weight value is assigned local neighbourhood X_p for each pixel p as

$$W_p = \min[D_L^p \in X_p] + 1 \tag{9}$$

Projected depth pixels with higher weight W_p has higher priority to be assigned for particular voxel. Thinning algorithm is shown in Fig.1.

5. VOLUME RENDERING

Volume Rendering visualizes 3D grids of voxels where each voxel stores a scalar value representing density. The volume raycasting algorithm casting segments of rays through volume to acquire colour information for certain pixel. Images are created by sampling the volume along all viewing rays and accumulating the resulting optical properties [1]. According to computed colour and opacity, model is described in equation:

$$C = \sum_{i=1}^{N} C_i \prod_{j=1}^{i-1} (1 - A_j) \tag{10}$$

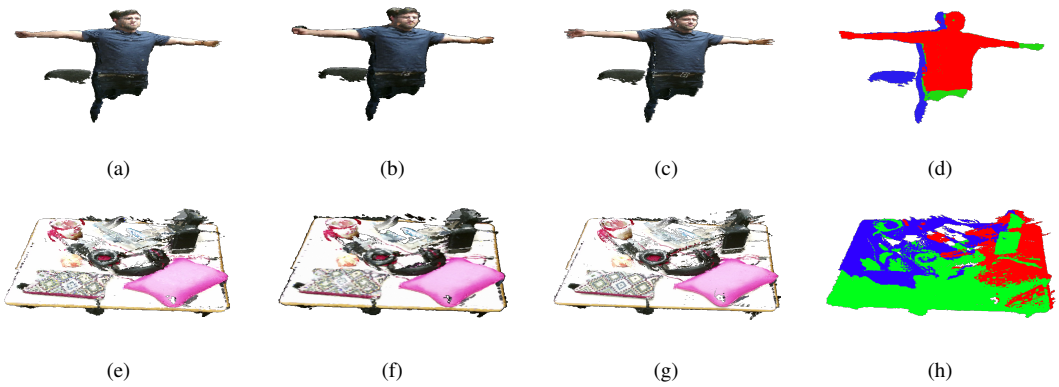

Figure 2: Two examples of volume rendering with three methods (volume size is 300^3) : (a,e) Direct colour estimation (b,f) Trilinear interpolation (c,g) Texture mapping (d,h) Separated volume by colour for each Kinect v2 sensor.

where C_i and A_i are the colour and opacity assigned by the colour map to the data value at sample i. Conventional methods calculate colour for each voxel using transfer function [1]. The main idea of voxel colouring solutions is assigning colours to voxels in a 3D volume to maximize integrity with a set of input images. As was described in Sec.4, thinning algorithm has been used to prioritize non-edge pixels. In this approach we compared three basic methods of colour estimation.

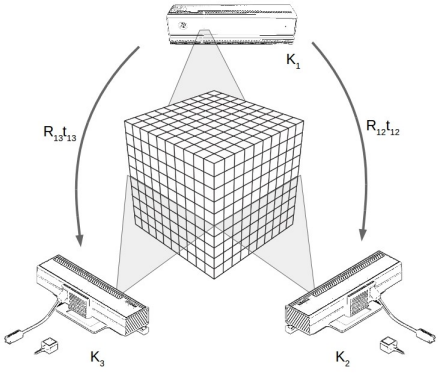

Figure 3: Principle of volume acquisition from multiple calibrated devices.

5.1. Direct colour estimation (DCE)

Voxels can contain multiple scalar values describing colour, transparency or additional material properties. Considering each voxel can be described with scalar vector $v_c = (R, G, B, A)$, casted ray passing through voxel on position directly obtain voxel colour C_i. Result of direct colour estimation rendering is shown in Fig.2a.

5.2. Trilinear colour interpolation (TI)

Based on the volume data type, missing information can be approximated via interpolating methods. Trilinear interpolation is a common operation in 3D data processing applications. The colour of pixel is approximated by building a weighted sum over all neighbours of the voxel position. For given ray passing through volume, values between eight neighbour voxels are interpolated as is described in [15].

$$C_i = \sum_{i=0}^{X} v_i w_i \tag{11}$$

where v_i represents colour of voxel in neighbourhood of eight voxels X and w_i are coefficients representing normalized areas[15] for each neighbour voxel (Fig.2b).

5.3. Texture mapping (TM)

The depth image represents organized 3D point cloud where adjacent points are connected, and each ray can acquire colour information direct from colour image capture by camera.

Fused 3D volume from depth maps of each Kinect v2 sensor is labelled with index of corresponding Kinect. This approach helps to reproject colour from Kinect camera to volume using reversed transformation (Section 3) as:

$$C_i = C_o[K(R^T v_i - \vec{t})] \tag{12}$$

where C_o represents colour image captured from Kinect v2 and v_i is intersection position between ray and voxel in world space coordinates.

In Fig.2c is shown result of texture mapping. Since Direct Colour Estimation method (DCE) requires store each colour information to a voxel, Texture mapping (TM) is using captured colour channel from device. Therefore, the size of volume in GPU memory can be reduced to single value voxel and colour map can be stored alongside to volume with transformation parameters.

6. EXPERIMENTAL EVALUATION

In this section, we present experimental results of the proposed volume rendering method in real time using by three Kinect v2 RGB-Depth sensors connected to a single PC. For our experiments we used CUDA parallel computing with NVIDIA GTX780

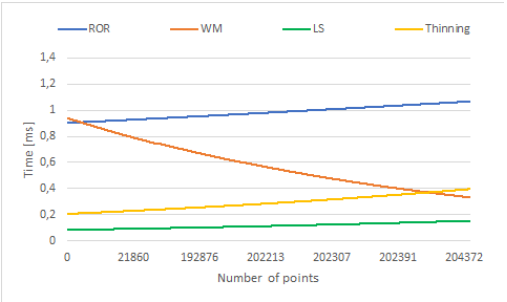

Figure 4: Processing time of each filtering method in ms.

graphic card using 1580 cores. The project was written in C++ with operating system Linux Ubuntu 16.04LTS. Each Kinect v2 sensor is connected to single machine by using USB 3.0 PCI-Express enhancement card with maximum transfer rate 5Gbps and libFreenect2 driver [16]. We calibrated our system with double sided planar pattern with size 6x9 corners and 5cm corner distance to find extrinsic parameters for each Kinect v2 IR camera. According to our previous research [8] precision of calibration was giving decent result with error 0.9mm. We decided to evaluate calibration in distance of 4.5 meters from depth sensor. Depth data beyond this distance yields inconsistent and lead to false calculations of extrinsic parameters.

Additionally, we measured processing time of each filtering method for one Kinect v2 (Fig.4). The crucial component of the system configuration are processing methods for real-time visualisation and content generation. For this experiment, we set sliding window of WM and ROR to 2.

Table 1: Processing time of rendering methods in ms.

Rendering method	**Size of volume**					
	128^3	256^3	512^3	128^3	256^3	512^3
DCE	16.2	24.21	52.3	22.12	42.51	62.3
TI	24	75.21	120.5	40.36	112.21	315.1
TM	19.2	28.4	75.84	25.2	31.94	82.14
	Scenario 1			Scenario 2		

In second part of experimental evaluation, we measured time for each rendering method to display volume content for two different scenarios (Fig.2), person with filtered background and office objects placed on table. The Tab.1 shown average processing times (averaged 3000 frames) for each rendering approach for three different sizes of volumes. As described in the Section 5, image resolution is important part in displaying 3D content. Therefore, resolution of output image was set to 1280x800px.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented the multi-view volumetric reconstruction approach using Kinect V2 sensors. We gave insight into volumetric reconstruction and rendering methods used in real-time application. As was described in section 6, we compared several existing filtering approaches to improve depth map accuracy using GPU processing by measuring processing speed with relation to number of points.

Additionally, we have described algorithms to generate volumetric content for real-time rendering. Results have proven that Direct colour estimation (DCE) is common and the fastest method to render colourized volume data. However, Texture mapping (TM) method achieved the best result by enhancing 3D volume contend with captured image with Kinect v2 sensor.

In future, we also plan to perform more complex algorithms to enhance 3D content for reconstruction and recognition.

8. REFERENCES

[1] R. Fernando, *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, Pearson Higher Education, 2004.

[2] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2015, pp. 922–928.

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1912–1920.

[4] T. Fogal and J. H. Krüger, "Tuvok, an architecture for large scale volume rendering.," in *VMV*, 2010, pp. 139–146.

[5] N. F. Polys, S. Ullrich, D. Evestedt, A. D. Wood, and M. Aratow, "A fresh look at immersive volume rendering: Challenges and capabilities," in *IEEE VR Workshop on Immersive Volume Rendering, Orlando*, 2013.

[6] L. Zheng, Y. Wu, and K. L. Ma, "Perceptually-based depth-ordering enhancement for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 3, pp. 446–459, March 2013.

[7] B. Kainz, S. Hauswiesner, G. Reitmayr, M. Steinberger, R. Grasset, L. Gruber, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg, "Omnikinect: Real-time dense volumetric data acquisition and applications," in *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology*, New York, NY, USA, 2012, VRST '12, pp. 25–32, ACM.

[8] A. Satnik, E. Izquierdo, and R. Orjesek, "Multiview 3d sensing and analysis for high quality point cloud reconstruction," *Proc.SPIE*, vol. 10696, pp. 10696 – 10696, 2018.

[9] O. Wasenmüller and D. Stricker, "Comparison of kinect v1 and v2 depth images in terms of accuracy and precision," in *Computer Vision – ACCV 2016 Workshops*, C.-S. Chen, J. Lu, and K.-K. Ma, Eds., Cham, 2017, pp. 34–45, Springer International Publishing.

[10] P. Palasek, H. Yang, Z. Xu, N. Hajimirza, E. Izquierdo, and I. Patras, "A flexible calibration method of multiple kinects for 3d human reconstruction," in *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, June 2015, pp. 1–4.

[11] M. Kowalski, J. Naruniec, and M. Daniluk, "Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors," in *2015 International Conference on 3D Vision*, Oct 2015, pp. 318–325.

[12] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the Art in Surface Reconstruction from Point Clouds," in *Eurographics 2014 - State of the Art Reports*, Strasbourg, France, Apr. 2014, vol. 1 of *EUROGRAPHICS star report*, pp. 161–185.

[13] Y. B. Jia, "Singular value decomposition," *Department of Computer Science, Iowa State University, Ames, Iowa*, 2013.

[14] D. Li, X. Wu, and X. He, "Depth-based thinning: A new non-iterative skeletonization algorithm for 2d digital images," in *2014 9th IEEE Conference on Industrial Electronics and Applications*, June 2014, pp. 1193–1197.

[15] R. Wagner, "Multi-linear interpolation," *Beach Cities Robotics*, 2008.

[16] L. Xiang, F. Echtler, C. Kerl, T. Wiedemeyer, Lars, hanyazou, R. Gordon, F. Facioni, laborer2008, R. Wareham, M. Goldhoorn, alberth, gaborpapp, S. Fuchs, jmtatsch, J. Blake, Federico, H. Jungkurth, Y. Mingze, vinouz, D. Coleman, B. Burns, R. Rawat, S. Mokhov, P. Reynolds, P. Viau, M. Fraissinet-Tachet, Ludique, J. Billingham, and Alistair, "libfreenect2: Release 0.2," 2016.