3D Object Modeling with a Kinect Camera

Mayoore Jaiswal[*], Jun Xie[#] and Ming-Ting Sun[†]
[*#†]Department of Electrical Engineering, University of Washington, Seattle, USA
[*]E-mail: mayoore@uw.edu [#]E-mail: junx@uw.edu
[†]E-mail: mts@uw.edu

***Abstract*— RGB-D (Kinect-style) cameras are novel low-cost sensing systems that capture RGB images along with per-pixel depth information. In this paper we investigate the use of such cameras for acquiring multiple images of an object from multiple viewpoints and building complete 3D models of objects. Such models have applications in a wide range of industries. We implemented a complete 3D object model construction process with object segmentation, registration, global alignment, model denoising, and texturing, and studied the effects of these functions on the constructed 3D object models. We also developed a process for objective performance evaluation of the constructed 3D object models. We collected laser scan data as the ground truth using a Roland Picza LPX-600 Laser Scanner to compare to the 3D models created by our process.**

I. INTRODUCTION

3D object modelling is the process of developing a mathematical representation of the three dimensional surface of an object. A 3D model can represent a 3D object using a collection of points in the 3D space known as a 3D point cloud. These points can also be connected by various geometric entities such as triangles, lines, and curved surfaces. There is a plethora of 3D modelling applications in the media, movie, video game, and industries.

Laser scanners and stereo cameras can be used for 3D model constructions. Laser scanners are precise but are slow to compute a full scan, and are expensive. Stereo cameras can be cheap but require complex processing to compute depth estimations, and have poor depth estimations in homogeneous areas. The Kinect camera is capable of delivering real-time, good accuracy, and dense 3D scans at an economical cost. However, one of the limitations of Kinect is that the depth map is noisy and may contain regions without data (holes) due to the surface property of the object and the occlusions. This makes the construction of high quality 3D object models using Kinect a challenging task.

A notable system KinectFusion [1] and its variants [2] [3] create 3D reconstructions of indoor scenes using only the depth data captured by Kinect in real time. This system relies on slow motion of the Kinect camera and high cluttered environment to maintain its tracking. KinectFusion uses spatial and temporal averaging to reduce the noise. In the registration of the 3D partial point clouds from different views, KinectFusion uses the ICP (Iterative Closest Point) algorithm which requires the 3D point clouds to be close to each other. Thus, the scene has to be scanned slowly in order to make sure the neighboring views have significant overlaps so that the denoising and the registration can work well. Another potential problem is that in the registration KinectFusion relies on salient structural features in the registration, thus, it may have difficulty in handling objects lacking salient structural features, such as round objects. CopyMe3D [4] proposes an approach to create 3D model of a person sitting on a swivel chair who is rotated while being scanned by a Kinect. ReconstructMe and KScan are two commercial products which use a Kinect to create 3D models of objects.

In this paper, we investigate a scenario where only a limited views from Kinect are available for constructing the 3D object models. Since only a relatively small number of frames are available, the overlapped areas between frames from different views are rather limited, and thus, the denoising and registration methods used in KinectFusion may not be as effective. Also, the algorithm needs to be able to take care of objects lacking salient structural features and provide good texture for the constructed 3D object models. We propose a framework to overcome these problems. Specifically, in the registration process, we add an initial registration step using Random Sample Consensus (RANSAC) before the ICP fine registration. We also modify the ICP algorithm to include texture features so that it can handle objects without salient structural features. We detect the loop closure and perform global alignment to overcome the error-propagation problem. We also perform model denoising and texturing to remove the noise and give texture to the 3D object models.

The rest of this paper is organized as follows. In Section 2, a brief overview of Kinect is given. In Section 3, we describe our detailed 3D object modelling framework. In Section 4, we discuss the denoising and the texturing for the 3D object models. In Section 5, we discuss the evaluation of the 3D object models. The paper is concluded in Section 6.

II. OVERVIEW OF KINECT

We will focus our discussions based on Kinect v1 which is used in our current implementation. RGB image has a 640 x 480 pixel resolution. Each pixel has 8 bits. The depth image is acquired using an Infrared laser projector and a Monochrome CMOS sensor using structured light imaging. The data from the IR sensor are represented in 11 bits, which includes 1 bit allocated to mark the validity of the depth data. With 10 bits to represent the value, the IR sensor has 1024 levels of sensitivity to store the disparity measurements. After converting to the real world distance in millimeters – the depth image has 13 bits of data for each pixel with the low-order 3 bits for player index. The depth image resolution is 640 x 480 pixels. The spatial (x/y) resolution is 3mm and the depth resolution is 3 mm at 2m distance from the camera plane [5].

978-616-361-823-8

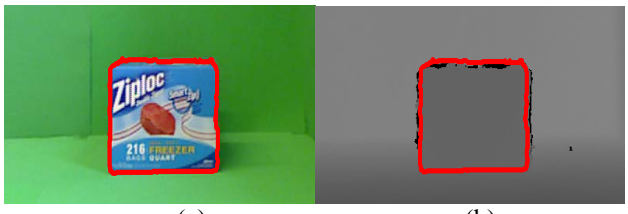
An example of RGB and depth images captured by a Kinect camera is shown in Figure 1. We can see that the depth image has holes, where data are not available, and is relatively noisy. The errors of the depth measurements of a Kinect camera increase quadratically with increasing distance from the sensor [5]. In general, for 3D scanning applications the data should be acquired within 1–3 m distance to the sensor. At larger distances, the quality of the data is degraded by the noise and low resolution of the depth value measurements.

Kinect v2 is available for preorder in the summer of 2014 and its SDK will be officially released by the end of 2014. The new Kinect uses time of flight technology to obtain depth images. It has 60% wider field of vision. The RGB frame has 1920 × 1080 pixel resolution and depth frame has 512 × 424 pixel resolution. This makes the quality of the frames much superior to that of Kinect v1. It will be our future work to investigate the results using Kinect v2.

III. A 3D Object Modeling Framework

Figure 2 shows a flowchart for our 3D object modeling process. In the framework, first, the synchronized RGB and depth images of an object are captured with a Kinect. The foreground object is then segmented out from the background and represented in a 3D point cloud using the corresponding RGB and depth data. The partial 3D point clouds from different views are then registered together to form a complete 3D point cloud for the object.

In order to handle the situation in which the neighboring views may have relatively small overlaps, we add RANSAC in the registration process to perform an initial registration before using ICP for the fine registration. After a complete cycle, due to the error propagation, the last view of the object may not align well with the 3D point cloud of the first view. We perform a global alignment to adjust the model to minimize the misalignment due to the error propagation. After that, the combined 3D point cloud model is further de-noised to result in the 3D object model. The model can be transformed to other 3D representations such as polygon mesh, volumetric representations **[6]**, or Surfels **[7]** for different applications. In the following sub-sections, we provide details for each sub-process.



(a) (b)

Figure 1: (a) and (b) show the RGB and depth images, respectively, captured by a Kinect camera, with their object boundary, obtained from the highlighted segmentation mask.

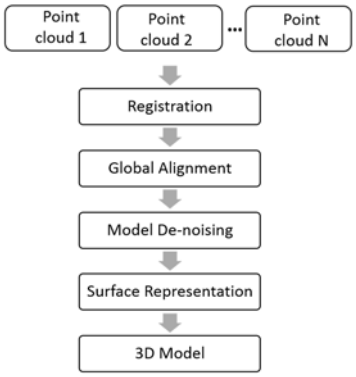


Figure 2: A flowchart of the Kinect based 3D object modeling process.

A. *Forming partial 3D point clouds from individual views*

General object segmentation is an ill-posed problem. Since the 3D object modeling can be conducted in a controlled environment, we place the object on a green surface and in front of a green screen and use a simple color-based segmentation which has a low complexity to segment out the object from the RGB image. This foreground object mask is then applied to the depth image. Figure 1 illustrates an example of the segmentation result. With the cropped RGB and depth images, the 3D point cloud of one view can be generated.

B. *Registration*

In KinectFusion and related methods, initial alignment is not performed [1] [2] [3]. These methods use a video stream of RGB and depth data at 30 fps. Hence two adjacent frames have no significant changes in viewpoints. This eliminates the need for an initial alignment before the data is fed to the ICP algorithm to align the two point clouds. However, in our case we have frames with limited overlapping regions. ICP could perform poorly, if these frames are directly fed to it. For this reason, we employ a coarse to fine registration procedure as described below.

With the RGB and depth information, we first use the RANSAC algorithm to perform a coarse alignment. In the coarse alignment procedure, point clouds from two neighboring views are roughly registered based on the SIFT features in the RGB images.

The second phase of registration is the ICP based fine registration. The standard ICP algorithm aligns two point clouds by iteratively associating points through a nearest-neighbor search and estimating the transformation parameters using a mean square cost function [8].

While the ICP algorithm has been widely used for dense point cloud matching, it is limited in its ability to produce accurate results when objects lack structural features or undergo significant changes in camera view. To address this problem, we used a fine registration algorithm we proposed [9]. In this algorithm a SIFT based term is added into the cost function to overcome the case when objects lack structural features. To utilize the texture information of the 3D points without intensive computation of the SIFT descriptor for every 3D point, a constraint involving the spatial distances of the

SIFT feature corresponding pairs is added. In addition to this constraint, a dynamic weight is used to properly balance the significance of structural and photometric terms. It is referred to [9] for the details.

C. *Global Alignment*

In the registration process, registration errors could accumulate as more point clouds are added to the model. Figure 3(a) illustrates the effects of this error accumulation. Slight inaccuracy at each registration step leads to significant misalignment between the last and first frame. This error could be eliminated by a global alignment which is to detect the last frame of a loop closure, register this frame with its previous frame and the first frame, and distribute the error evenly amongst other frames [10].

Loop Closure Detection

Previous techniques have used the Euclidean distance between each frame and all previous frames along with a threshold of minimal number of intermediate scans [10] [11] to detect loop closures. Our loop closure detection algorithm is based on the assumption that the camera captures the views of the object in either clockwise or counter-clockwise direction. When we capture data we make sure that the last frame has a large overlap with the first frame so that the loop is properly closed and we could detect the loop closure relatively easily.

Global Alignment Procedure

When a loop closure is detected, we formulate a loop graph with each vertex representing one frame of the 3D point cloud. The edge between the successive frames is the pose transformation between successive frames. We first use ICP to align the first and last frame with loop closure and get the transformation ΔY. Then, $T_1T_2 \ldots T_N = \Delta Y$, where T_i is the pose transformation found in the registration process. If the registration does not have any errors, ΔY should be an identity matrix. The non-identity ΔY is due to the errors of registration. Denote the inverse of ΔY as ΔX, then $\Delta X\, T_1T_2 \ldots T_N = I$, where I is the identity matrix. ΔX can be distributed among all poses in the loop to minimize the loop closure mismatch. In order to achieve a consistent map, we need to calculate the weights for the vertices that specify the fraction of ΔX by which the transformation needs to be changed. The weights are computed as follows:

$$w_i = \frac{d(v_i, v_{i-1})}{d(v_s, v_e)} \tag{1}$$

where v_s is the first vertex in the loop and v_e is the last one. $d(v_l, v_k)$ is the sum of the edge weights on the path from v_l to v_k. The weighting $w_i \in [0, 1]$ is such that:

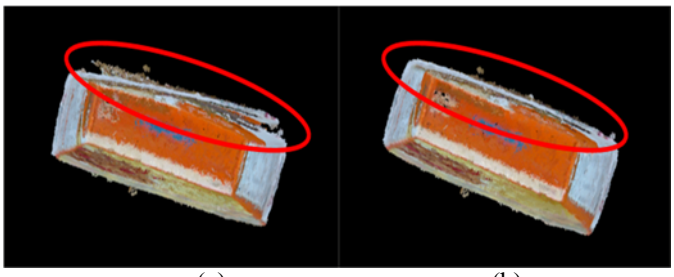
$$\Delta X = \sum_i (w_i * \Delta X) \tag{2}$$

The final modified transformation for each frame i after the optimization is:

$$T_i' = w_i * \Delta X * T_i \tag{3}$$

where T_i' is the modified transformation [11]. The whole process is repeated several times until the convergence is achieved.

Figure 3 shows the 3D modeling result before and after the global alignment. The global alignment has eliminated the discrepancy between the first and the last frames used to create the 3D model.



(a) (b)

Figure 3: Modeling result after the global alignment. (a) Before the global alignment (after registration) (b) After the global alignment procedure.

IV. Model Denoising And Texturing

A. *3D Object Model De-Noising*

After the registration and the global alignment, the 3D model is well registered. However it is still very noisy due to the inaccurate depth acquisition from the depth camera. Therefore 3D point cloud de-noising is needed to refine the 3D object model.

In this work, we use the Moving Least Square (MLS) 3D model denoising method [12]. MLS uses the assumption that on a 3D surface, the 3D point set defines a manifold. The purpose of MLS is to find the manifold that the input 3D point set defines. Based on the manifold assumption, we can approximate the MLS surface by a function.

Let $s_i \in R^3, i \in \{1, \ldots M\}$ be points in the noisy 3D point cloud. The goal is to project the points onto a two-dimensional surface that approximates s_i. For each small local neighborhood in the point cloud, we fit the points by a local tangent plane H. Denote the height of s_i over H as r_i, and h the origin of the reference domain define by H, we can estimate a polynomial approximation g so that the weighted squares error:

$$\sum_{i=1}^{M} (g(x_i, y_i) - r_i)^2 \theta(\|s_i - h\|) \tag{4}$$

is minimized, where θ is a smooth monotonically decreasing function, which is positive on the whole space (e.g., Gaussian approximation in our implementation). (x_i, y_i) is the coordinate of the s_i's projection on H. After minimizing the above function, coefficients of the polynomial function g can be computed. The projection approximation process is run for every point in the 3D point set and it has been proved that it can nicely preserve the manifold property [12].

Figure 4 shows the result of 3D model before and after MLS. From the result, we can see that the surface of the 3D

box model is much smoothed and the outlier noises have been significantly removed.

(a) (b)

Figure 4. (a) Before 3D de-noising (after registration). (b) After MLS de-noising.

B. *Meshing and Texturing*

The initial 3D model is represented in point clouds. Performing meshing and texture mapping makes the model resemble the real-life object with texture. For this purpose we triangulate the point clouds to build a mesh surface and then map the RGB color from the RGB image to the mesh.

For our 3D modeling application, we use the Delaunay triangulation method [13] to convert the 3D point clouds into meshes. A Delaunay triangulation for a set P of points in a plane is defined as a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$. We can interpolate space inside the triangles to get a locally smoothed surface.

After meshing, we assign the color to each vertex and simply interpolate the color in each triangle faces. This texture mapping requires that the resolution of triangulation should be high enough so that interpolation will not introduce many artifacts. The built 3D point cloud based model can be converted to other 3D representations.

V. Evaluation Of 3D Kinect Models

A. *Error Evaluation*

To investigate the error in a 3D model obtained using the above outlined procedure, a comparison was made with a point cloud obtained by a laser scanner. The laser scanner point cloud was obtained from the same object by a calibrated Roland Picza LPX-600 laser scanner. This laser scanner produces high-resolution point cloud of objects using a non-contact red laser and high quality optics to capture parts. The nominal range accuracy of the laser scanner is +/-0.05 mm [14]. It is therefore assumed that the laser scanner point cloud is sufficiently accurate to serve as a reference for the accuracy evaluation of the Kinect point cloud. In the absence of any systematic errors the mean square of discrepancies between the two point clouds is expected to be close to zero.

We propose the following methodology to calculate the error between the 3D model constructed and the 3D point cloud obtained from the laser scanner which is used as the ground truth. Assume the 3D object model contains R points, and the ground truth 3D point cloud contains R' points (usually R'>>R). We first randomly sample R points from the ground truth 3D point cloud. Random sampling is done to ensure that the R points are located in random locations in the ground truth point cloud and not concentrated in a particular location or part of the ground truth point cloud. Before the evaluation of the model errors, both point clouds need to be registered accurately, because any registration error may be misinterpreted as error in the Kinect model. We use the coarse to fine registration methodology described before to align the two point clouds. First, RANSAC is applied to the point clouds of the two 3D models to coarsely align the two point clouds. Then we use our improved ICP registration as described before to achieve a fine registration. After this registration, the 3D object model error is then quantified using the Root Mean Square Error (RMSE) of the closest distances in the fine registration. The RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{R}\sum_r d(r)^2} \qquad (5)$$

where $d(r)$ is the closest distance between the point r in the ground truth point cloud points and its closest point in the Kinect model point cloud. We repeat the RMSE calculation N times and calculate the average RMSE. Average of N RMSE readings reduce possible errors due to the random sampling. In our experiments we used $N = 5$.

In order to verify this error calculation methodology, the laser point cloud is randomly sampled twice to get two point clouds of p points each. The average RMSE between these point clouds is calculated and tabulated in Table 1. Since these two point clouds are subsets of the same initial point cloud the error between these point clouds should be close to zero. This error, as expected, is close to zero, thus validating the proposed error calculation method.

TABLE 1
THE AVERAGE ROOT MEAN SQUARE ERROR BETWEEN TWO RANDOMLY SUBSAMPLED POINT CLOUDS FROM THE SAME LASER POINT CLOUD.

Object	**Average RMSE (mm)**
Biscuit box	0.347
Mug	0.661
House model	0.545
Nut container	0.398

B. *Results*

The algorithm was implemented in C++ using the Point Cloud Library (PCL) and tested on a computer with Intel Xeon 64 bit processor with 16GB of RAM. The GPU was a Quadro K600 with 1 GB memory and 192 cores. Our algorithm took around 45 – 60 seconds to create 3D models.

TABLE 2
TABULATES THE AVERAGE RMSE OF THE OBJECT MODELS OBTAINED FROM KINFU AND OUR 3D MODELLING ALGORITHM. THE AVERAGE RMSE IS TABULATED FOR DIFFERENT NUMBER OF INPUT FRAMES.

Number of frames	Average RMSE (mm)							
	Biscuit Box		Nut Container		House Model		Mug	
	KinFu	3D model	KinFu	3D model	KinFu	3D model	KinFu	3D model
65	3.284	2.597	3.939	1.724	2.974	3.088	3.025	3.183
55	4.362	2.732	5.237	1.801	3.482	3.228	3.892	3.259
45	Failed	2.863	Failed	1.932	3.927	3.362	4.823	3.468
35	Failed	2.959	Failed	2.066	Failed	3.485	Failed	3.621
25	Failed	3.099	Failed	2.189	Failed	3.609	Failed	3.798
15	Failed	3.298	Failed	2.311	Failed	3.784	Failed	3.918

This process runs only on a CPU. It could be considerably speed up using a GPU. We compared our method with KinectFusion. We used the open source implementation, KinFu, which is available in the PCL library. KinFu was capable of creating 3D models in under 10 seconds. However KinFu takes advantage of the parallel processing capabilities of a GPU.

We used data from the RGB-D dataset [15] for our initial experiments. We then collected RGB and depth data by scanning 360 degree view of the objects in Figure 5. We then created subsets of the data consisting of 15, 25, 35, 45, 55 and 65 frames. Next we created 3D models by running KinFu and our algorithm with the datasets. These models are compared with the ground truth, and their average RMSEs are calculated and tabulated in Table 2. KinFu failed to create 3D models with small numbers of views, whereas our algorithm created 3D models with acceptable average RMSE.

Figure 5: The objects that we used to create 3D models

VI. CONCLUSION

In this report, we present our work on 3D object modeling using a Kinect camera. We investigate the use of such cameras for acquiring RGB and depth images of small objects from limited number of viewpoints and building a complete 3D model of the object. We address some problems such as fine registration, global alignment, texture mapping, denoising, and objective error measurement of the 3D object models. For future works, we will investigate various denoise and other techniques to further improve the 3D models.

REFERENCES

[1] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim and A. Fitzgibbon, "KinectFustion: real-time 3D reconstructin and interaction using a moving depth camera," *IEEE International Symposium on Mixed and Augmented Reality,* pp. 127-136, 201.

[2] H. Roth and M. Vona, "Moving Volumne KinectFusion," in *Proceedings of the British Machine Vision Conference*, 2012.

[3] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, 2012.

[4] J. Sturm, E. Bylow, F. Kahl and D. Cremers, "CopyMe3D: Scanning and Printing Persons in 3D," in *German Conference on Pattern Recognition (GCPR)*, 2013.

[5] K. Khoshelham, "Accuracy Analysis of Kinect Depth Data," *Internationsl Archives of the Photogrammetry, Remote Sensing and Spatial Infomation Sciences,* Vols. XXXVIII-5, 2011.

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

[7] H. Pfister, M. Zwicker, J. van Baar and M. Gross, "Surfels: Surface Elements as Rendering Primitives," in *Proc. SIGGRAPH 2000*, 2000.

[8] P. Besl and N. McKay, "A method for registration of 3D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 14, pp. 239-256, 1992.

[9] J. Xie, Y. F. Hsu, R. Feris and M. T. Sun, "Fine Registration of 3D Point Clouds with ICP Using an RGB-D Camera," in *IEEE International Symposium on Circuits adn Systems(ISCAS)*, 2013.

[10] F. Lu and E. Milios, "Global Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots,* vol. 4, pp. 333-349, 1997.

[11] J. Sprickerhof, A. Nuchter, K. Lingemann and J. Hertzberg, "A Heuristic Loop Closing Technique for Large-Scale 6D SLAM," *Journal for Control, Measurement, Electronics, Computing and Communications, Special Issue with selected papers from the European Conference on Mobile Robots 2009.,* 2011.

[12] H. Avron, A. Sharf, C. Greif and D. Cohen, "L1-sparse Reconstruction of Sharpe Point Set Surfaces," *ACM Transactions on Graphics,* vol. 29, 2010.

[13] M. Isenburg, Y. Liu, J. Shewchuk and J. Snoeyink, "Streaming Computation of Delaunay Triangulations," in *SIGGRAPH*, 2006.

[14] "http://www.rolanddg.co.uk," [Online]. Available: http://www.rolanddg.co.uk/files/>PX_DS_brochure.pdf. [Accessed 25/3/2014].

[15] K. Lai, L. Bo, X. Ren and D. Fox, "A Large-Scale Hierarchical Multi-View RGB-D Object Dataset," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.