3D reconstruction system using Kinect camera

Thuong Le-Tien
Ho Chi Minh City University of Technology
Vietnam National University HCM
Ho Chi Minh City, Vietnam
thuongle@hcmut.edu.vn

Thang-Anh-Quan Nguyen
Ho Chi Minh City University of Technology
Vietnam National University HCM
Ho Chi Minh City, Vietnam
quan.nguyen0125@hcmut.edu.vn

***Abstract*—The main object of this paper is to develop a 3D reconstruction system. This problem involves estimating camera pose change by projecting 3D key points to point cloud and finding a rigid transformation from one point cloud into another so that they align. Iterative Closest Point (ICP) and its variance provide simple and easily implemented methods for this task, but are not efficient and easily get stuck in local minima, another coarse alignment method is Sample Consensus Initial Alignment (SAC-IA) is used to improve the overall fitness. As the constructed system could obtains acceptable results with a low-price device, it can be applied to object scanning and indoor mapping application.**

***Keywords*—Kinect camera, 3D reconstruction, indoor mapping, ICP**

I. INTRODUCTION

In recent years, with the development of 3D sensors, point clouds have been widely used in many fields such as autonomous vehicles, augmented and virtual reality, commercial, agriculture as well as geological measurement. Optical 3D scanners are becoming popular due to their flexibility in digitizing surface data and providing high resolution and precision compared to other devices. With their excellent performance, the development and integration of scanners for 3D reconstruction had become an increasingly common part of the world we live in.

Kinect Fusion [1] was developed by Microsoft Research in 2011, allowing users to reconstruct a 3D scene in real-time by moving the Microsoft Kinect sensor. This has been an influence on a lot of Structures from Motion (SfM) [2], visual odometry, and Simultaneous Localization And Mapping (SLAM) systems. Most RGBD reconstruction systems use visual features such as SIFT, SURF, or ORB for matching [3, 4]. However, the 3D geometric properties can be lost if the views are not sufficiently similar. In complex perception problems such as finding correspondences in multiple overlapping datasets, another alternative representation especially point cloud is extremely useful as we could find the characterization of geometry around a point.

In this work, we use a 3D point cloud processing approach to develop object scanning with the help of RGBD cameras such as Kinect. We will conduct registration methods from several obtained viewpoints of models and discuss implementation for indoor mapping application.

II. THE KINECT CAMERA

Kinect is an RGBD sensor, which is officially released by Microsoft in June 2010. It can capture the depth image and RGB image at the same time, the frame rate is high, and the price is relatively low. RGBD cameras such as Kinect are based on structured lights. Fig. 1 shows the overall structure of the camera. Kinect contains four main parts: A depth-sensing system, an RGB camera, a tilt motor, and a microphone array.

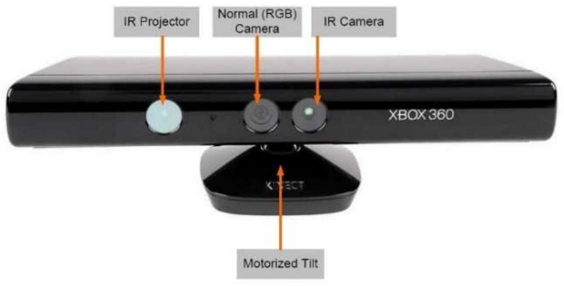


Fig. 1. Microsoft Kinect camera

A. Calibration

In a depth image, each pixel is related to the distance between the image plane and the object's surface. We can project those data into a point cloud giving the pinhole camera model [5].

$$z = \frac{bf}{\frac{1}{8}(1090-d)} \tag{1}$$

$$x = -\frac{z}{f_x}(u - c_x f_x) \tag{2}$$

$$y = -\frac{z}{f_y}(v - c_y f_y) \tag{3}$$

where $(u, v)^T$ is a pixel coordinate in the depth image with d as its disparity value. $(x, y, z)^T$ is the corresponding 3D point coordinates. f_x and f_y, the focal lengths of the camera, together with (c_x, c_y), the camera center coordinate, are the intrinsic parameters that can be computed by a calibration algorithm. b is the horizontal baseline.

B. Supporting libraries

1) OpenNI: OpenNI is a multi-languages and multi-platformed framework, it defines API for natural interacting applications. The main purpose of OpenNI is to build standard APIs to interact with the camera, microphones, and their middleware.

2) Open3D: Open3D [6] is an open-source library that supports the rapid development of software to deal with 3D data. The Open3D frontend exposes a set of carefully selected data structures and algorithms in both C++ and Python. The backend is highly optimized and set up for parallelization.

III. METHODOLOGY

Fig. 2 demonstrates the main phases of a point cloud-based 3D reconstruction system. First, point clouds are acquired through RGBD images. After being processed to improve quality, and followed by estimating 3D feature descriptors, the registration step aligns those point clouds into one complete object. The final model can then be integrated in 3D graphics applications such as Unity. The following section will give a detailed description of each module's operational principle.

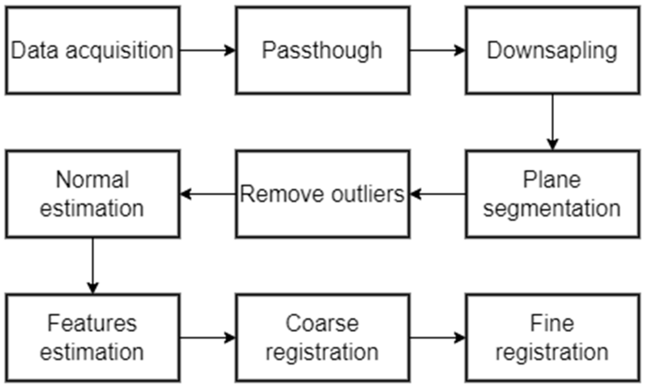


Fig. 2. Main steps in point cloud scanning application.

A. *Point cloud pre-processing*

Acquiring point clouds from Kinect faces many challenges as the data are huge and sensor noises around the same position are variant at different acquisition times. As a result, a sequence of filters a first applied to each of the clouds to remove redundant data and reduce computational costs. These steps are:

- Passthrough filter: point clouds may contain unnecessary information such as floor plane, background, etc., other than the object. Passthrough is simply cutting off values that are either inside or outside a given range along specified dimensions.
- Down sampling: the idea of this filter is to create volumetric pixel (voxel) grids over the input cloud data. Then, all points present in each voxel will be approximated (down sampled) by the voxel's centroid (instead of the center), this approach is slower but it gives a better representation of the underlying surface. The larger the leaf size, the fewer points remain after this step.
- Plane segmentation [7]: data in the point cloud can be approximated by a set of basic geometric models, for example, a plane can present the wall, floor,... or a cylinder can present a mug. In this paper, we're using RANSAC to estimate a plane model, mostly the floor to separate from the object since it might negatively affect the registration step.
- Statistical outliers removal [8]: this method is based on a statistical analysis of one point's neighbor. With each point, we first compute the mean distance $\bar{d}$ to its k closest neighbors. By assuming a distribution from those distances is Gaussian, after estimating the mean and standard derivative (μ, σ), we simply keep the points whose distance to k closest neighbors is similar to the rest. The remaining point cloud is represented as:

$$\mathcal{P}^* = \{p_q^* \in \mathcal{P} | (\mu - \alpha \cdot \sigma) \leq \bar{d} \leq (\mu + \alpha \cdot \sigma)\} \quad (4)$$

where α is the filter's interval defined hyperparameter.

B. *Normals estimation*

To estimate the surface normal, we instead estimate the normal of the tangent plane to the surface [9] (a least-square plane fitting problem in the neighbor set $\mathcal{P}^k$). The plane is represented by a point x and a normal vector $\vec{n}$. Take:

$$x = \bar{d} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \quad (5)$$

as a centroid of $\mathcal{P}^k$ and solve for $\vec{n}$ by using principal components analysis (PCA) on the covariance matrix of $\vec{n}$, expressed as:

$$\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} \xi_i \cdot (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \quad (6)$$

where $\mathcal{C} \cdot \vec{v_j} = \lambda_j \cdot \vec{v_j}, j \in \{0, 1, 2\}$. If $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$, the eigenvector $\vec{v_0}$ corresponding to the smallest eigenvalue λ_0 is the approximation of $\vec{n} = \{n_x; n_y; n_z\}$.

C. *Features estimation*

Point Features Histogram (PFH) encodes the geometry features of a point's neighbor set (this set's radius is larger than in the normal estimation step) using multi-dimensional histograms. Rusu defined a Darboux frame [10] (uvw local coordinate system which is presented in Fig. 3) to calculate the relative difference between two points and their normals. This reduces the number of features from 12 to 4.

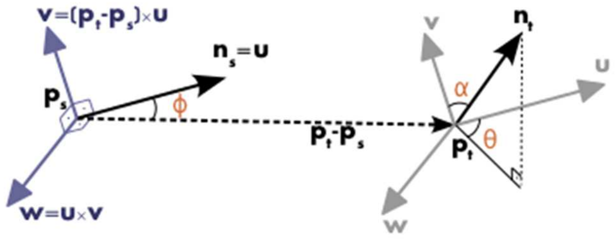


Fig. 3. Representation of the Darboux frame.

$$u = n_s \quad (7)$$

$$v = u \times \frac{(p_t - p_s)}{||p_t - p_s||_2} \quad (8)$$

$$w = u \times v \quad (9)$$

A quadruplet of features $\langle \alpha, \varphi, \theta, d \rangle$ can be chosen as follows:

$$\alpha = v \times n_t \quad (10)$$

$$\phi = u \cdot \frac{(p_t - p_s)}{d} \quad (11)$$

$$\theta = \arctan(w \cdot n_t, u \cdot n_t) \quad (12)$$

$$d = \left||p_t - p_s|\right|_2 \tag{13}$$

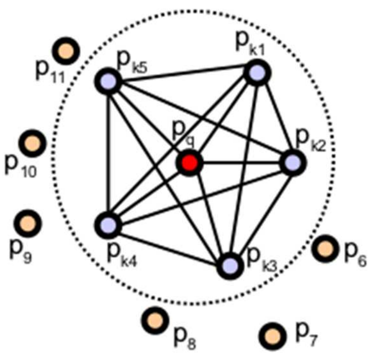


Fig. 4. Point Feature Histogram diagram. The query point and its neighbors.

Each quadruplet value is separated into some subdivisions, and their number of occurrences is recorded to form a histogram description.

Fast Point Feature Histogram (FPFH) [10] reduces computational complexity from $O(nk^2)$ to $O(nk)$ while still having the discriminative power of the PFH. By canceling the d value in the quadruplet, this becomes Simplified Point Feature Histogram (SPFH). Finally, SPFH values of the query point and its neighbors are used for the final FPFH. We also denote the weight factor ω_i to encode the distance information to the query point p_q for the neighbors' SPFH:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_i}SPFH(p_i) \tag{14}$$

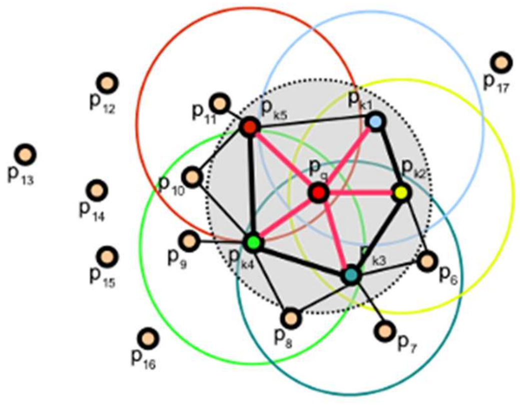


Fig. 5. Fast Point Feature Histogram diagram. The query point is connected only to its neighbors. Each neighbor is connected to its own neighbors and the SPFH result is weighted together with SPFH of the query point.

D. Registration

After acquiring point clouds from different views, the next step is aligning them into a complete model. The main purpose is to find rigid transformations between these individual scans so that their intersecting area could overlap by matching correspondences in source cloud to target cloud. The transformation matrix $\boldsymbol{T} \in \mathbb{R}_{4\times4}$ consists of a rotation matrix $\boldsymbol{R} \in \mathcal{SO}(3)$ and a translation vector $\boldsymbol{t} \in \mathbb{R}_{3\times1}$ in 3D space [11, 12]:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \tag{15}$$

ICP is a popular method for point cloud registration [13, 14] based on minimizing Euclidean distances in the overlapping areas. The advantage of ICP is that it's simple to find the closest neighbors with the help of kd-tree. However, additional drawbacks of this method and its variances are that they can get stuck in local optima if data contain many outliers as they rely on a rough alignment as initialization. Moreover, they require a high number of iteration steps until convergence.

In order to solve the point correspondences matching problem, Rusu proposed the registration module [8] by constructing geometry constraints between points using FPFH as they are rotation-invariance, known as SAC-IA coarse registration. Using RANSAC , s sample candidates are picked from the source point cloud. Their corresponding points in the target point cloud will be considered if their histograms are similar to the sample points' histograms. The matches are used to compute the rigid transformation. The above process is repeated N times to pick the best $\boldsymbol{T}$. Huber Loss is chosen as the error metric for the quality of the transformation. However, SAC-IA usually produces less tight results, so a refine method (such as ICP) uses this initial alignment to further reach the global optima.

The ICP algorithm can be summarized in two steps:

- Find pairs of closest points between two clouds.
- Compute a transformation which minimizes the distance between those points.

$$\boldsymbol{T} = argmin\left\{\sum_i \left||p_i - T\cdot q_i|\right|_2\right\} \tag{16}$$

where p_i and q_i are the two closest points in the datasets.

These above steps are repeated until the result convergences or reaches certain stopping conditions.

The Point-to-plane [15] variance instead tries to minimize error along the surface normal, which doesn't restrict tangential movement. Therefore, it accelerates the convergence speed, by simply changing the classical ICP's objective function into:

$$\boldsymbol{T} = argmin\left\{\sum_i \left||n_i\cdot(p_i - T\cdot q_i)|\right|_2\right\} \tag{17}$$

where n_i is the normal to the surface at point q_i.

IV. SYSTEM'S DETAILS

Usually, there is no specific number of scans required for object scanning applications. It depends on the complexity of the object and the experience of the implementer. For large object or application such as mapping, pairwise registration which register each pair of clouds sequentially is implemented for faster speed. However, false pairwise alignments can outnumber correctly aligned pairs. Error in global convergence may cause the whole system to collapse. A pose graph is implemented here to optimize the past drifted nodes to their global optima by creating a loop constrain (edge) between two nodes whether they overlap.

Let $\boldsymbol{x} = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_i \cdots \boldsymbol{x}_n]^T$ be the vectors of camera's 6D pose from t_1 to t_n. The result of $\boldsymbol{x}_i$ in the global space can be achieved through the rigid transformation matrix $\boldsymbol{T}_i$ in the registration step. The purpose is to estimate $\boldsymbol{x}$:

$$\boldsymbol{x} = argmin \sum_i \left(\boldsymbol{e}_{i,i+1}^T \Omega_{i,i+1} \boldsymbol{e}_{i,i+1}\right) + \sum_{ij} \left(\boldsymbol{e}_{ij}^T \Omega_{ij} \boldsymbol{e}_{ij}\right) \quad (18)$$

with $\boldsymbol{e}_{ij}(\boldsymbol{x}) = \boldsymbol{z}_{ij} - \hat{\boldsymbol{z}}_{ij}$ being the error function between the relative pose and the estimate roto-translation, and the information matrix Ω_{ij} presents the covariance of the alignment. Each constrain is modeled with an information matrix and an error function. Nonlinear least-squares (NLS) is widely used to solve this problem [16]. Iterative techniques (Gauss-Newton or Levenberg-Marquardt) approximate the error function with its Taylor expansion around the initial guess.

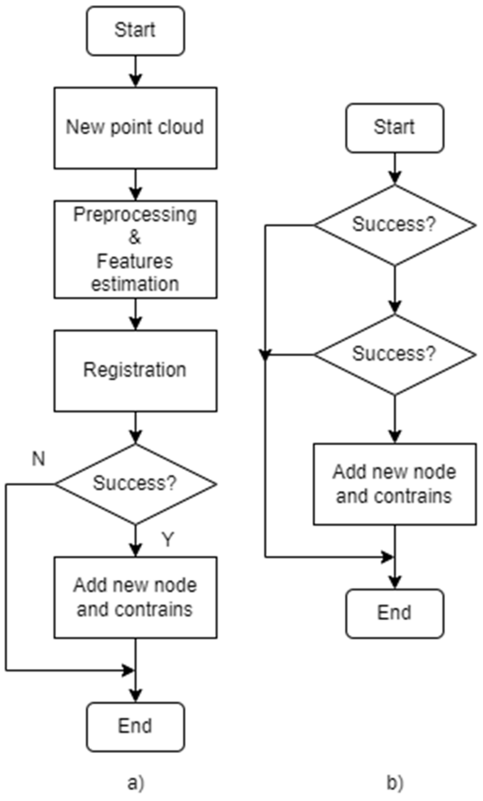

Start
New point cloud
Preprocessing & Features estimation
Registration
N
Success?
Y
Add new node and contrains
End
a)
Start
Success?
Success?
Add new node and contrains
End
b)

Fig. 6. Block diagram of the overall 3D reconstruction system. a) Local registration of pairs of point clouds. b) Pose graph optimization to fix global poses.

Fig. 6 describes the flowchart of the entire system. *Local registration* (left) processes the input data, estimates pose and updates graph structure. *Graph Optimization* (right) optimizes partial or entire graph structure by pulling past nodes closer to their global optima. Loop detecting depends on distance metrics to maintain global map consistency and reduce drift. The mechanism is simple, we create a loop constraint between two nodes whenever the two scans overlapped by a certain threshold.

V. RESULTS

A. Implementation

The experimental hardware environment of this paper is processor Intel(R) Core(TM) i5-9400F CPU @2.9GHz, memory 16.00GB. The computer is running on Ubuntu 18.04.

First, we make a comparison between the traditional point-to-point ICP with sample consensus initial alignment (SAC-IA) implementations on Stanford University Graphics Lab's “Bunny”, “Dragon”, “Armadillo” and “Happy Buddha” models.

Evaluation metrics for the above algorithms are fitness which measures the overlapping area, the higher the better, and RMSE which measures the RMSE of all inlier correspondences, the lower the better.

$$fitness = \frac{k}{N} \quad (19)$$

where k is the number of inlier correspondences and N is the number of points in target.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left|\left|p_i - q_j\right|\right|^2}, (1 \le j \le N) \quad (20)$$

where p_i and q_j are the two closes point in the datasets.

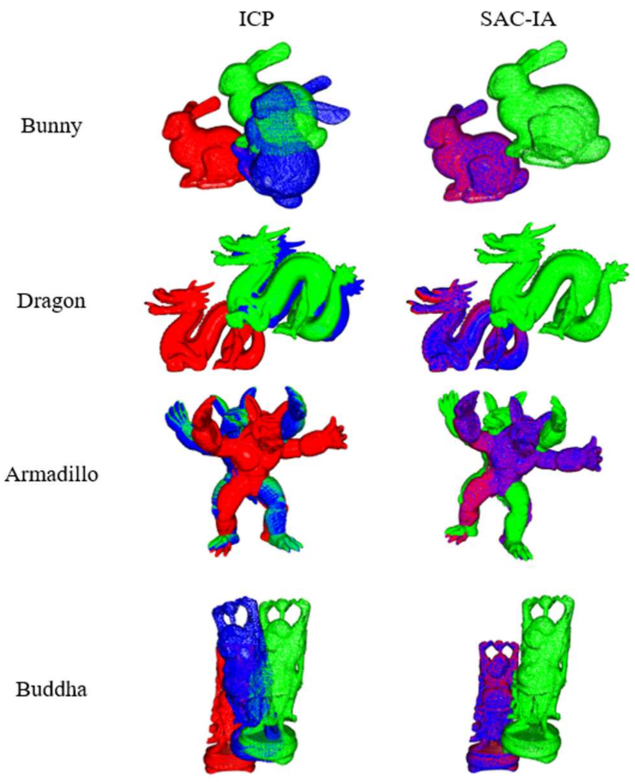


Fig. 7. Testing registration algorithm on various models.

Fig. 7 illustrates the results with evaluation metrics mentioned in Table I. In the table, the RMSE drops 2 to 4 times when coarse registration is used.

TABLE I. REGISTRATION RESULTS ON FOUR MODULES

Module	**ICP**		**SAC-IA**	
	RMSE	***Fitness (%)***	***RMSE***	***Fitness (%)***
Bunny	0.0032	8.90	0.0016	97.74
Dragon	0.0027	7.23	0.0018	96.75
Armadillo	0.0055	5.30	0.0017	94.42
Buddha	0.0022	21.07	0.0010	97.83

It can be seen that, if the difference between the two poses is huge and the overlapping areas are small, only ICP couldn't produce good results. The key to this problem is that SAC-IA has to be implemented first as a way to improve alignment fitness score. On the other hand, two-stage registration could perform even better when noises appear in data.

B. Simulation

The whole system's performance is tested on augmented ICL-NUIM RGBD benchmark Dataset [17], a synthetic dataset consisting of RGB and depth images for two different scenes: a living room and an office recorded at 30Hz. Ground truth trajectories are also provided for evaluation.

Several registration methods mentioned above could be applied for mapping for reconstructing 3D scenes or deformable motion tracking. To keep the scene's details as many as possible, passthrough and plane segmentation steps can be temporarily removed. Registering each point cloud against the one preceding it is used in odometry. Then we assemble a point cloud map by aligning the registered point clouds using their optimized absolute poses. In addition, given the first pose $\boldsymbol{P_0}$, odometry transformation $\boldsymbol{T_i}$ could be utilized to generate pose information of the camera $\boldsymbol{P_i} = \boldsymbol{T_i P_{i-1}}$ at each time interval, therefore calculate the trajectory. ATE and RPE metrics [18] are selected for quantitative evaluation of localized error between the estimated pose and ground truth.

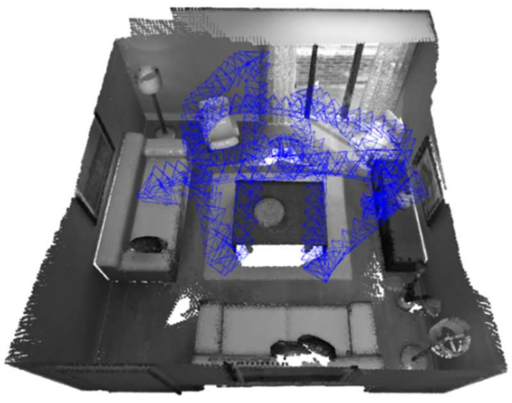


Fig. 8. Reconstructing result of Living 2 with camera trajectory.

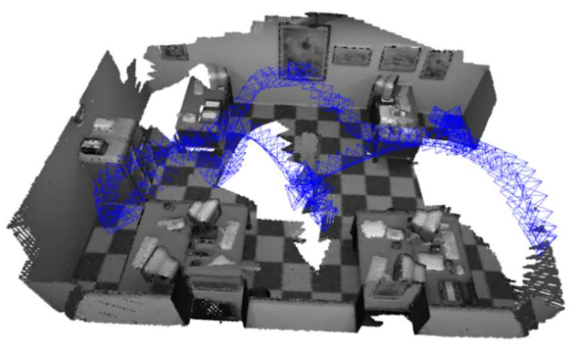
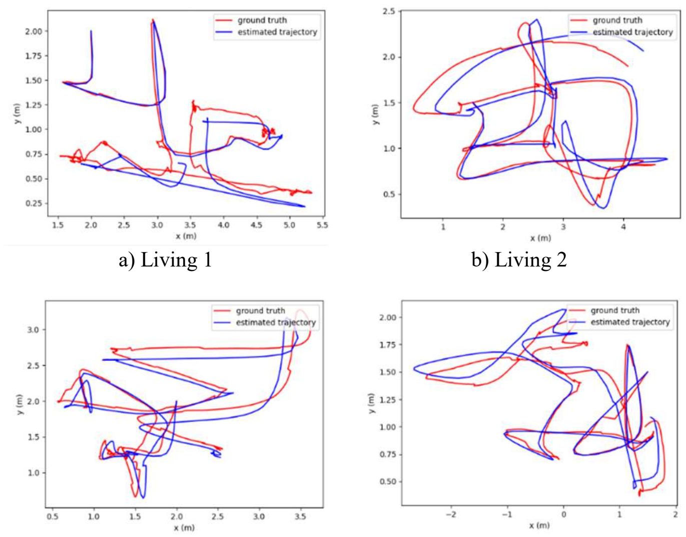


Fig. 9. Reconstructing result of Office 2 with camera trajectory.

TABLE II. MEAN AND STD OF ATE AND RE ON FOUR DATASETS

Dataset	Trajectory Length (m)	ATE (m)		RE (deg)	
		Mean	*Std*	*Mean*	*Std*
Living 1	26.1382	0.1113	0.0472	2.1227	1.0753
Living 2	30.5995	0.2250	0.1467	5.1459	2.1301
Office 1	25.4632	0.2671	0.0667	2.8061	1.2211
Office 2	30.0325	0.1562	0.1002	2.9323	1.2167

Looking at Fig. 8 and Fig. 9, it is apparent that the system constrains must be chosen carefully so that they don't cause map data to collapse or distort.



a) Living 1 b) Living 2

c) Office 1 d) Office 2

Fig. 10. Estimated trajectory (blue line) and ground truth (red line) of four datasets.

As can be seen from Table II, ATE and RE fluctuate around 0.1-0.25m and 2-5 degree respectively, thereby confirming the system's reliability despite the fact that there exists some randomness in the system's operation.

C. Experiments

In the experiment, we're using Kinect to capture several point clouds of the object in different views as shown in Fig. 11. The runtime of each step is recorded in Table III.

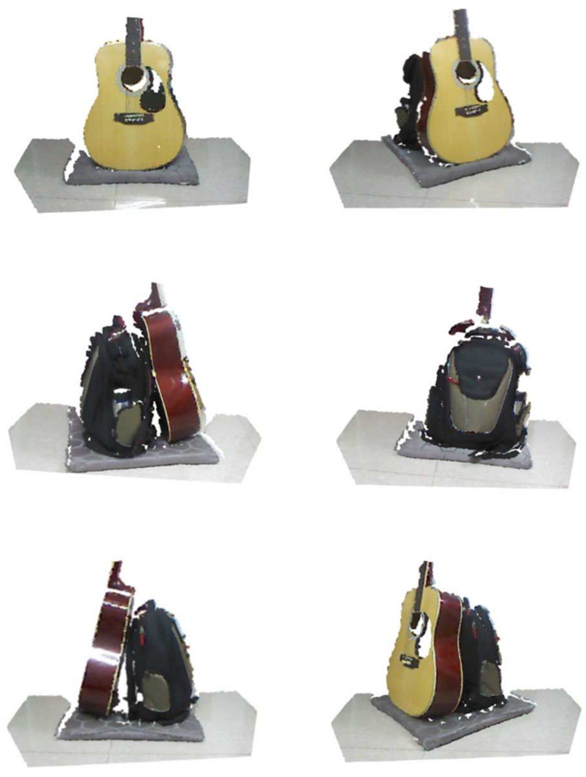


Fig. 11. Cloud acquired from different views

In object scanning application, simplistic pairwise approach above produces a small amount of drift over each capture. As the camera loops back around the object, the last capture pose may not overlap completely with the first. Optimization is also applied in this case. We're using the global registration method, which minimizes accumulation errors by estimating all transformations of all possible pairs of scans. This method obviously requires high computational costs, but it achieves better results.

TABLE III. RUNTIME (IN MILISECOND) ON ONE PAIR OF CLOUD OF EACH MODULES

Module	Runtime (ms)
Passthrough	6.40
Down sampling	5.26
Plane segmentation	134.85
Outlier removal	3.09
Normals estimation	2.60
FPFH estimation	33.50
Registration	55.42
Total	**241.12**

Fig. 12. illustrates the complete model, each registration step has reached RMSE = 0.005, fitness = 60% as an average. Since each pair of clouds is not fully overlapped, this result is highly acceptable.

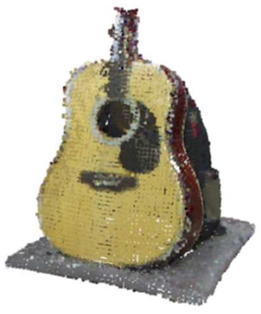


Fig. 12. Final registered object.

VI. CONCLUSION

In this paper, we propose a 3D reconstruction system using Kinect camera, a low-cost, and easily operated depth-sensing device compared to other professional scanning systems. Through experimental results, we have made improvements in the reconstruction performance. The system can be used in an interactive display system for commercial or robotics localization.

ACKNOWLEDGMENT

We would like to thank Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for the support of time and facilities for this study.

REFERENCES

[1] Richard A. Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality. 2011, pp. 127–136. doi: 10.1109/ISMAR.2011.6092378.

[2] Onur Ozyesil et al. A Survey of Structure from Motion. 2017.

[3] Mathieu Labbé and Fran¸cois Michaud. “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation”. In: Journal of Field Robotics 36.2 (2019), pp. 416–446. doi: 10.1002/rob.21831.

[4] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. “ORB- SLAM: A Versatile and Accurate Monocular SLAM System”. In: IEEE Transactions on Robotics 31.5 (2015), pp. 1147–1163. doi: 10.1109/tro.2015.2463671.

[5] Xing Zhou. “A Study of Microsoft Kinect Calibration” (2012). url: https://cs.gmu.edu/~xzhou10/doc/kinect-study.pdf

[6] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: CoRR (2018). doi: 10.48550/ARXIV.1801.09847.

[7] Michael Ying Yang and Wolfgang Fo¨rstner. “Plane Detection in Point Cloud Data”. In: Proceedings of the 2nd int conf on machine control guidance 1 (Feb. 2010).

[8] Radu Bogdan Rusu. “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments”. In: KI - Kunstliche Intelligenz 24 (2010), pp. 345–348.

[9] J. Berkmann and T. Caelli. “Computation of surface geometry and segmentation using covariance techniques”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 16.11 (1994), pp. 1114–1116. doi: 10.1109/34.334391.

[10] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. “Fast Point Feature Histograms (FPFH) for 3D registration”. In: 2009 IEEE International Conference on Robotics and Automation. 2009, pp. 3212–3217. doi: 10.1109/ROBOT.2009.5152473.

[11] Ben Bellekens et al. “A Benchmark Survey of Rigid 3D Point Cloud Registration Algorithms”. In: International Journal On Advances in Intelligent Systems 1 (June 2015).

[12] Xiaoshui Huang et al. “A comprehensive survey on point cloud registration”. In: CoRR abs/2103.02690 (2021).

[13] Zhengyou Zhang. “Iterative point matching for registration of free-form curves and surfaces”. In: International Journal of Com- puter Vision 13 (1994), pp. 1573–1405. doi: 10.1007/BF01427149.

[14] P.J. Besl and Neil D. McKay. “A method for registration of 3D shapes”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 14.2 (1992), pp. 239–256. doi: 10.1109/34.121791.

[15] Kok-Lim Low. “Linear Least-Squares Optimization for Point-to- Plane ICP Surface Registration”. In: (2004).

[16] Rainer Ku¨mmerle et al. “G2o: A general framework for graph optimization”. In: 2011 IEEE International Conference on Robotics and Automation. 2011, pp. 3607–3613. doi: 10.1109/ICRA .2011.5979949.

[17] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. “Robust Reconstruction of Indoor Scenes”. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015.

[18] Ankur Handa et al. “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM”. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1524–1531. doi: 10.1109/ICRA.2014.6907054.

[19] Thuong Le-Tien et al. “3D Reconstruction using Kinect Sensor and Parallel Processing on 3D Graphics Processing Unit”. In: REV Journal on Electronics and Communications 3 (Oct. 2013). doi: 10.21553/rev-jec.55.R.