15-01

15th IAPR International Conference on Machine Vision Applications (MVA)
Nagoya University, Nagoya, Japan, May 8-12, 2017.

Model-Based 3D Pose Estimation for Pick-and-Place Application

Shih-Cheng Liang, Huei-Yung Lin
Department of Electrical Engineering
Advanced Institute of Manufacturing with High-Tech Innovation
National Chung Cheng University
168 University Road, Min-Hsiung, Chiayi 621, Taiwan

Chin-Chen Chang
Department of Computer Science and Information Engineering
National United University, Miaoli 360, Taiwan

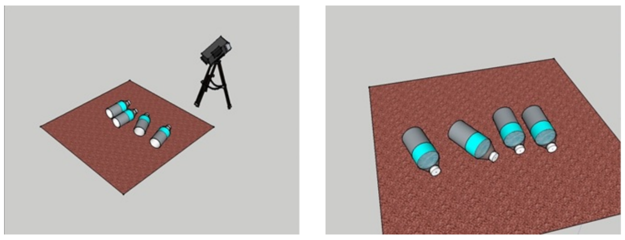
Abstract

Due to the recent development of industrial automation, some applications have been improved with computer vision techniques. One important task is to recognize and estimate the 3D pose of the object in the scene. In this work, we use a depth camera to capture the 3D information of a scene, and proposed a 3D pose estimation algorithm. A main difficulty of the 3D object recognition and pose estimation is the captured data may have noise from the environment light, shadow or sensors. In general, the reference model and target model are captured from the same depth camera, so they will have similar data structures. However, in our work, we consider the target model generated from Computer-Aided-Design, and the reference model is captured from the depth camera. The data from different sources will cause the estimation error. In this work, we have addressed this problem. Finally, we develop the simulation system for our proposed method, and also simulate a manipulator to accomplish the pick-and-place task.

1 Introduction

The objective of this work is to derive the 3D pose of an object in the scene based on the depth information acquired by an RGB-D camera [11, 9]. First, we develop a simulation system to build the ground truth of the object in the synthetic scene, and design a method to simulate the depth camera to capture the 3D images in the virtual environment. Our proposed pose estimation algorithm is then used to compute the translation and orientation of the object and verify the correctness [12, 3]. Finally, according the estimated 3D pose of the object, a manipulator is instructed to move to a suitable position, pick up the object, and place it on the target location to complete an industrial automation task.

In this paper, we address the technical issues of the pick-and-place application. The proposed approach mainly contains two parts: "3D object pose estimation and evaluation" and "simulation system for the virtual environment." For the simulation system, we first consider the image formation of the depth camera in the virtual environment. It takes the perspective projection into consideration to generate the 3D point clouds in the camera coordinate system. A graphical model of a five-axis robotic arm is then constructed in the virtual environment to emulate the movement of the real manipulator. The inverse kinematics of the robot motion is calculated and used to demonstrate the pick-and-place task.


(a) Environment. (b) Camera view.
Figure 1. The schematic diagram of the camera and environment setting for depth image acquisition and 3D object pose estimation.

For 3D pose estimation and evaluation, the CAD (computer-aided design) model of the object is created and placed at an initial position in the environment. The alignment between the CAD model and the 3D scene captured by the depth camera is achieved using our 3D pose estimation algorithm, which is capable of computing the translation and rotation with respect to the initial CAD model pose [1]. Since the input source from the depth camera usually contains noise, the matching with the ideal 3D model can be inaccurate. We have proposed an evaluation mechanism to verify the pose estimation results.

To estimate the 3D pose of the object, an RGB-D camera is adopted. A schematic diagram is shown in Fig. 1. The virtual environment for 3D scene acquisition and the camera view are illustrated in Figs. 1(a) and 1(b), respectively. In the experiments, we deal with a more complicated situation where some objects are occluded by others.

2 Our Approach

Our proposed technique for 3D pose estimation consists of five steps. In the first step, the CAD model generated with mesh data is converted to the point cloud data. The scene model of the object is then captured by a depth camera. In addition to the RGB-D images taken from the real world, we also generate a computer graphical model for the target object [7]. Furthermore, the data points of the captured scene model are down-sampled to reduce the computational cost of the pose estimation algorithm. Third, the clustering step removes the background and noise, and clusters the object in the scene. It is used to identify the vertices of the same object. The 3D pose of each object is then calculated by the ICP algorithm [4]. Since the estimated result is generally not accurate, the ICP algorithm is improved by a novel idea proposed in this work. Finally, we verify the 3D pose estimation results to evaluate if the object is suitable for manipulator picking.

The preprocessing stage consists of two parts: scaling the CAD model and generating the point cloud from the mesh model. Since the dimension of the generated CAD model may not be equivalent to the dimension of the real world object, it is necessary to scale the CAD model and make it comparative to the captured 3D object data for pose estimation [6, 2]. Let the scale of the object and the CAD model are S and M, respectively. If the CAD model is represented by $P(V)$, where $V = v_1, v_2, \cdots, v_n$, and n is the number of the points. Then the new CAD model with the same scale as the object is given by

$$P_N(V) = \mu P(V), \qquad \mu = \frac{S}{M}$$

As for generating the point cloud from the mesh model, the objective is to fill the points in the triangular mesh of the CAD model. Let

$$F(k_m) = (v_{m1}, v_{m2}, v_{m3}), \qquad m = 1, 2, \cdots, N$$

represent the triangles of the mesh, where N is the number of triangles, and v_{m1}, v_{m2}, v_{m3} are the vertices of the triangle. We first define the vectors

$$\vec{v}_1 = v_{m2} - v_{m1} \quad \text{and} \quad \vec{v}_2 = v_{m3} - v_{m1}$$

the new points inside the triangle can be generated by

$$v_n = v_{m1} + \alpha\vec{v}_1 + \beta\vec{v}_2$$

where α and β are arbitrary constants with the conditions

$$0 \leq \alpha, 0 \leq \beta, \alpha + \beta \leq 1$$

Setting the density S_d (points/unit) by adjusting the parameters α and β, we can fill the triangular mesh uniformly with additional points.

To capture the point cloud of the scene, a depth camera is placed above the objects and facing downwards. Several cuboids of the same size are used to simulate the real pick-and-place task. Fig. 2 shows the image and 3D point cloud captured by a structured light depth camera. The data set contains 233,491 points. Since a large number of points will cause the pose estimation algorithm running slowly, it is down-sampled using voxel grid filtering for further processing. In the next step, the 3D point cloud is processed with the clustering algorithm stated in **Algorithm 1** to separate the objects.

The original ICP algorithm is commonly affected by the initial position so that the 3D pose estimation will possibly not converge to the correct results. If some extra amount of movement is applied to the wrong pose, there is a better chance not to fall into the incorrect local minimum in the next iteration. Thus, we propose a technique to use the genetic algorithm to provide the additional poses to adjust the model position and find the correct 3D pose [5].

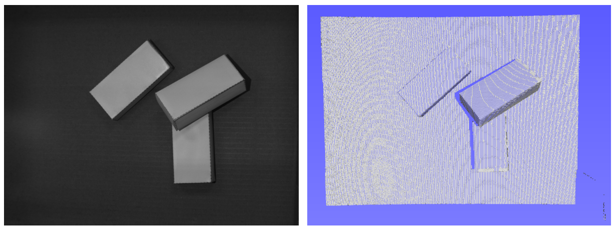

(a) The original image. (b) Point cloud data points.

Figure 2. Real scene with several objects.

Algorithm 1 Clustering

```
Require: $p$ (Point Cloud of Scene Model)
Ensure: clustering object
1: $p$ build the kd-tree
2: select a never visited point, then push back to stack buffer
3: let $a$ is a point that pop up from stack buffer, check the point whether to visited, then $a$ be a search point to find neighbor point in kd-tree $p$.
4: if the distance from neighbor point to $a$ is less than t, we can push back that point to stack buffer and that point is a same object with $a$.
5: repeat step 3, if stack buffer is empty, then repeat step 2.
6: if all points are visited, the algorithm is end.
```

Consider six unknown parameters, including the translation movement (M_x,M_y,M_z) and the rotation angles (α,β,γ) along three axes. The fitness function of the genetic algorithm is to minimize the equation

$$E = \sum \|Rp + T - q\|^2$$

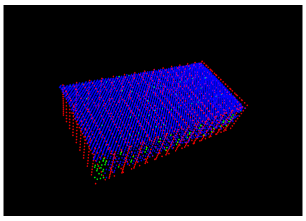
where $R = R_z(\gamma)R_y(\beta)R_x(\alpha)$, $T = [M_x \ M_y \ M_z]^\top$, p is the scene model, and q is the CAD model. Since the computational cost of the genetic algorithm is relatively high, it is carried out only the following conditions hold:

$$|E(k+1) - E(k)| \leq 10^{-6}, \quad E(k+1) \geq T$$

That is, the error difference between two consecutive iterations is less than 10^{-6} and the error of the current iteration is greater than a threshold T. The former case means the ICP solution falls into the local minimum, and the latter condition indicates the current 3D pose is not correct.

Each of iteration of the genetic algorithm uses the current pose to calculate the fitness function. After the calculation is completed the genetic algorithm obtains a solution, and the current pose information is updated. The iteration of the ICP algorithm then continues. It should be noted that the genetic algorithm does not always give the best solution and find the correct pose. The idea is to significantly adjust the overall pose by the characteristics of the genetic algorithm, and then followed by the ICP algorithm to achieve the correct pose.

In the last step, we verify the accuracy of the 3D pose estimation. Since the scene model and the CAD model are obtained from the depth camera and the computer graphical model, respectively, the density distributions


(a) Large overlap ratio.

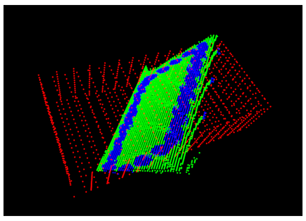

(b) Small overlap ratio.

Figure 3. Verification and evaluation.

of the input data from different sources are not consistent. As a result, the 3D points of the scene and the generated point cloud of the CAD model do not overlap pointwise directly, even with a correct 3D pose. Thus, we propose two methods to evaluate the accuracy and correctness of the 3D pose estimation results. We first consider the ratio of overlap by calculating the percentage of overlapping regions with respect to the CAD model. The ratio of inverse overlap is then computed in terms of the percentage of overlapping regions with respect to the scene model. The details of these two evaluation techniques are described in **Algorithms 2** and **3**, respectively.

Algorithm 2 The ratio of overlap

```
Require: CAD Model ($q$) has $C$ points, Scene Model ($p$) has $S$ points
Ensure: The ratio of the overlay (%)
1: $q$ build kd-tree
2: $p$ be a search point to find neighbor point in kd-tree $q$. if the distance from neighbor point to $p$ is less than t, we can mark the neighbor point.
3: Cumulative the mark point as $P$
4: The ratio=$\frac{P}{C} \times 100\%$
```

Algorithm 3 The inverse of the ratio of overlap

```
Require: CAD Model ($q$) has $C$ points, Scene Model ($p$) has $S$ points
Ensure: The inverse of the ratio of the overlay (%)
1: $p$ build KD-tree
2: $q$ be a search point to find neighbor point in kd-tree $p$. if the distance from neighbor point to $q$ is less than t, we can mark the neighbor point.
3: Cumulative the mark point as $P$
4: The ratio=$\frac{P}{S} \times 100\%$
```

Fig. 3 shows an example of our evaluation and verification technique. The correct pose result is shown in Fig. 3(a) with the ratio of overlap about 52%. In other words, there is a 52% of point region overlap between the CAD model and the scene model. The inverse of the ratio of overlap is about 99%, which means that the scene model points has 99% overlap with CAD model. Fig. 3(b) shows an example of incorrect 3D pose estimation. The red, green, and blue colors represent the CAD model, scene model, and the corresponding points from the CAD model to the scene model, respectively. The CAD model points provide only about 29% overlap with the scene model (i.e., the ratio of overlap), and the inverse of the ratio of overlap is about 34%. Based on our evaluation method, the ratio of overlap is lower if an object is occluded by another. In the pick-and-place step, the high ratio indicates that the 3D pose estimation is more accurate, and is generally more suitable for the application.

Table 1. Pose estimation overlap (in percentage).

	Box	1×2 Lego	2×4 Lego	Joystick
ICP	61.76	8.82	8.82	29.41
ICP-G	98.24	84.12	90.89	84.99

Table 2. Pose estimation computation (in sec).

	Box	1×2 Lego	2×4 Lego	Joystick
ICP	0.22	2.26	2.26	2.16
ICP-G	0.61	7.30	3.17	20.18

3 Experiments

This section presents the experimental results, including the accuracy of the estimated 3D pose and the simulation system with a virtual camera and multiple objects pick-and-place of in the virtual scene. The experimental environment is shown in Fig. 4. In the experiments, the objects are segmented perfectly with our clustering algorithm so the single objects are used to evaluate the pose estimation method. Four test objects, Box, 1×2 Lego, 2×4 Lego and Joystick, as shown in Fig. 5, are used for our experiments. Figs. 6 and 7 show the CAD models and the scene models captured by a depth camera, respectively.

As described previously, the result obtained from the ICP algorithm are affected by the initial position and orientation, thus we use 34 initial poses of the CAD model to evaluate our pose estimation technique. The poses are set up by rotating the CAD model with 30-330 degrees along the x, y, z axes respectively. The comparison between two methods, ICP algorithm and ICP with genetic algorithm (ICP-G), is shown in Tables 1 and 2 with the overlap ratio and computation time, respectively. Two types of registration models are the scene models (as shown in Fig. 7) and the CAD models (as shown in Fig. 6). The iteration number of the ICP algorithm is set as 400 if no solution converges earlier. For the ICP with genetic algorithm, if the ICP converges then a new pose will be assigned for further computation.

In the experiments, the Box object is a simple model, and the accuracies of ICP and ICP-G are 61.76%and 98.24%, respectively. For a more complex object 1×2 Lego, the accuracy of ICP-G is much better (84.12% vs. 8.82%) at the cost of much longer computation time (7.30 sec vs. 2.26 sec). Similar results can be

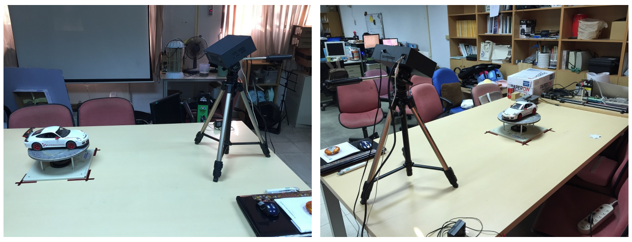


Figure 4. The experimental environment.

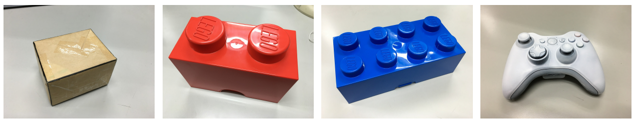

(a) Box (b) 1×2 Lego (c) 2×4 Lego (d) Joystick

Figure 5. Test objects used in the experiments.

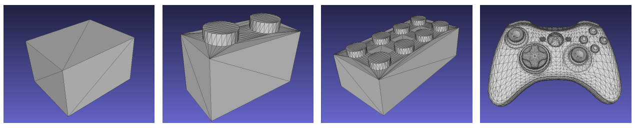

(a) Box (b) 1×2 Lego (c) 2×4 Lego (d) Joystick

Figure 6. CAD models of the objects in Fig. 5.

seen for other complex objects, 2×4 Lego and Joystick, where ICP-G provides better accuracy but requires more computation.

For the simulation system, we consider a virtual camera with the perspective projection incorporated with a manipulator for pick-and-place task [8]. The mesh-based CAD model of an object is placed in front of the virtual camera and each pixel of the image (with the resolution of 2080×1552) is back-projected into the 3D space to find the object points. Consider all of the back-projected rays, the intersections of the vectors and surface triangles of the CAD model are calculated [10].

To simulate a manipulator to complete the pick-and-place task, we consider a virtual scene consisting of a 5-DOF robot arm, a depth camera and a conveyor belt. After performing the point cloud data acquisition and 3D pose estimation, the manipulator picks an object and places it on the conveyor belt according to the derived location and orientation of the object. We design a five-axis manipulator with controllable rotation angles $(R_1, R_2, R_3, R_4, R_5)$ and a pair of claw to catch objects in the virtual environment. The robot arm is operated according to the functions with rotation angle parameters.

On the robot arm, two points on the claw are defined as touch points to manipulate the objects. Let the midpoint (x, y, z) be the initial position, and the target position from the estimated pose be (m_x, m_y, m_z). In the experimental setup, the rotation angles R_1, R_2, R_3 and R_4 are solved by the genetic algorithm with the constraint on the range of rotation angle.

4 Conclusions

In this paper we present a vision-based 3D pose estimation technique for random bin picking applications. The pose estimation results from the conventional ICP-based approach are usually affected by the initial pose of the object. We propose a technique with the genetic algorithm to overcome this problem. In our method, the CAD model of the object is created and converted to point cloud data. It is then used to register with the 3D scene obtained from a depth camera to derive the 3D pose of the object. The experiments are carried out with several real objects, and the performance evaluation have demonstrated the feasibility of our approach.

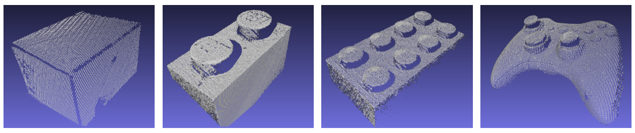

(a) Box (b) 1×2 Lego (c) 2×4 Lego (d) Joystick

Figure 7. Point clouds from a depth camera.

Acknowledgment

The support of this work in part by the Ministry of Science and Technology of Taiwan under Grant MOST 104-2221-E-194-058-MY2 is gratefully acknowledged.

References

[1] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592, Nov 2011.

[2] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(23):114 – 141, 2003. Nonrigid Image Registration.

[3] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 998–1005, June 2010.

[4] N. Mellado, D. Aiger, and N. J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. *Comput. Graph. Forum*, 33(5):205–215, Aug. 2014.

[5] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[6] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, May 2009.

[7] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

[8] R. B. Rusu, A. Holzbach, M. Beetz, and G. Bradski. Detecting and segmenting objects for mobile manipulation. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 47–54, Sept 2009.

[9] J. Serafin and G. Grisetti. Nicp: Dense normal based point cloud registration. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 742–749, Sept 2015.

[10] O. Tropp, A. Tal, and I. Shimshoni. A fast triangle to triangle intersection test for collision detection. *Computer Animation and Virtual Worlds*, 17(5):527–535, 2006.

[11] X. Wang, H. Zhang, and G. Peng. 3-dof point cloud registration using congruent triangles. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1943–1948, Sept 2015.

[12] C. Zach, A. Penate-Sanchez, and M. T. Pham. A dynamic programming approach for fast and robust object pose recognition from range images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 196–203, June 2015.