International Journal of Innovative
Computing, Information and Control
Volume **14**, Number **2**, April **2018**

ICIC International ©2018 ISSN 1349-4198
pp. 537–547

3D POSE ESTIMATION USING GENETIC-BASED ITERATIVE CLOSEST POINT ALGORITHM

Huei-Yung Lin[1,2], Chin-Chen Chang[3,*] and Shih-Cheng Liang[1,2]

[1]Department of Electrical Engineering
[2]Advanced Institute of Manufacturing with High-Tech Innovation
National Chung Cheng University
No. 168, Sec. 1, University Rd., Minhsiung, Chiayi 62102, Taiwan

[3]Department of Computer Science and Information Engineering
National United University
No. 2, Lien Da, Nan Shih Li, Miaoli 36063, Taiwan
*Corresponding author: ccchang@nuu.edu.tw

Received August 2017; revised December 2017

Abstract. *Three-dimensional (3D) pose estimation is an essential problem for computer vision. In this paper, we present an improved model-based approach for 3D pose estimation. Target models are obtained from computer-aided design models and reference models are captured by a depth camera. We propose a technique to address estimation errors stemming from different sources. Moreover, pose estimation results derived through the conventional iterative closest point (ICP) approach are usually affected by the initial pose of an object. We propose an improved ICP algorithm with a genetic technique to overcome this problem. Finally, we develop a simulation system for the proposed approach and simulate a manipulator to accomplish the pick-and-place task.*
Keywords: 3D pose estimation, Point cloud, Feature point, Object clustering

1. **Introduction.** Some computer vision techniques have been utilized for industrial automation applications. One crucial technique is to estimate the three-dimensional (3D) pose of an object in a scene. A variety of 3D pose estimation approaches have been proposed [3,7,9,20]. However, estimating the 3D pose of an object is still a challenging task. A major difficulty for 3D pose estimation is that the captured data may have noise from environmental light, shadow, or sensors. 3D registration [1,2,4,8,14,19] plays a crucial role in 3D pose estimation. There are roughly two classes of 3D registration approaches. One is based on the iterative closest point (ICP) algorithm [4], and the other is based on the random sample consensus (RANSAC) framework [8].

In this paper, we estimated the 3D pose of an object in a scene based on depths acquired by a red-green-blue-depth (RGB-D) camera [17,19]. First, we built the ground truth of the object in a synthetic scene. We designed a simulation of the depth camera capturing 3D images in a virtual environment. We then used our proposed pose estimation algorithm to compute the translation and orientation of each simulated object and verify correctness [7,20]. Finally, according to the estimated 3D pose of each object, a simulated manipulator was instructed to move to a suitable position, pick up the object, and place it on a target location to complete an industrial automation task.

The proposed approach consisted of two main parts: a 3D object pose estimation and evaluation system, and a virtual environment simulation system. For the 3D pose estimation and evaluation system, a computer-aided design (CAD) model of the object is created and placed at an initial position in the environment. Our 3D pose estimation

algorithm aligns the CAD model and the 3D scene captured by the depth camera [8]. Because the input source from the depth camera usually contains noise, the matching with the ideal 3D model can be inaccurate. We introduce an evaluation mechanism to verify the pose estimation results. For the virtual environment simulation system, we first consider the image formation of the depth camera. The depth camera calculates a perspective projection to generate 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 a real manipulator. The inverse kinematics of the robot motion is calculated and used to demonstrate the pick-and-place task.

The remainder of this paper is structured as follows. Section 2 reviews related works. Section 3 describes our approach to 3D pose estimation. Section 4 describes the experimental results. Finally, Section 5 presents the conclusions and future directions.

2. **Related Works.** Consider a scene model p and a CAD model q; the goal of 3D pose estimation is to find transformations that p can register in q. A common way to estimate a 3D pose is to minimize the error defined by

$$\sum \|Rp + T - q\|^2, \tag{1}$$

where R is a rotation matrix and T is a translation matrix. Generally, the scene model and the CAD model have different numbers of points and the point data from different sources tend to cause estimation errors.

Besl and McKay [4] presented an ICP algorithm for registering point clouds. In each iteration, each point of the target point cloud is used to find the nearest point of the reference point cloud. Using these nearest points, they calculated the corresponding rotation and translation matrices. Let the number of scene models be n and the number of CAD models be m. The centroids of the models are defined as

$$\bar{p} = \frac{1}{n}\sum p \text{ and } \bar{q} = \frac{1}{m}\sum q. \tag{2}$$

The points derived from the centroids are given by

$$p_i' = p_i - \bar{p} \text{ and } q_i' = q_i - \bar{q}. \tag{3}$$

The ICP algorithm is to find the rotation matrix R and translation matrix T to minimize the error defined by

$$E = \sum_{i=1}^{n} \|Rp_i + T - q_i\|^2. \tag{4}$$

Chang et al. [5] utilized appearance-based specular features to estimate the 3D poses of specular objects. Instead of estimating 3D geometries or depths, they used observed specular reflections and specular flows as cues. Their approach can handle scenes with multiple specular objects, partial occlusions, inter-reflections, cluttered backgrounds, and changes in ambient illumination. Aldoma et al. [3] proposed an approach to extract 3D features for object recognition and object pose estimation. For CAD models of objects, they introduced a clustered viewpoint feature histogram and a camera roll histogram to compute the rotation around the roll axis of the camera and estimate the poses.

Aiger et al. [2] introduced the four-point congruent sets method and alignment scheme for 3D point sets. Their method extracts all coplanar four-point sets from a 3D point set that are approximately congruent to a given set of four coplanar points. They also proposed an extension to handle similarity and affine transforms. Rusu et al. [14] proposed point feature histograms and their variants as multi-dimensional features for point

cloud data. The features are pose invariant and reasonable candidates for point correspondence search in 3D registration. Moreover, they proposed an approach for the online computation of features for real-time applications.

Ahmed et al. [1] presented a 3D registration approach for two point clouds with noise. They first extracted the intersection of parametric surfaces from the two point clouds. Next, they established correspondences of the intersection points for 3D registration. Their approach was able to obtain desirable registration results. Wang et al. [19] presented a three-degree-of-freedom (3-DOF) registration method to align two overlapping point clouds. Their approach followed the RANSAC framework [8] and was based on finding a pair of congruent triangles in the source and target clouds. This approach maintained a certain level of accuracy and reduced the number of RANSAC iterations.

Most approaches may avoid the estimation error because, in general, reference models and target models are captured from the same depth camera. These models have similar data structures. Moreover, the captured data may have noise from the environment light, shadow or sensors. Most approaches can solve the noise problem and obtain desired results.

3. **Proposed Approach.** In the proposed approach, we first create the mesh-based CAD model of an object, and convert that model to point cloud data. We capture the scene model of an object with a depth camera. In addition to RGB-D images taken from the real world, we also generate a computer graphics model for the object [15]. Then, we down-sample the data points of the captured scene model to reduce the computation cost of 3D pose estimation. Next, we remove the background and noise, cluster objects in the scene, and identify the vertices of each particular object. After that, we register the 3D pose of each object using the ICP algorithm [11]. Because the estimated result is generally inaccurate, in particular, we introduce an improved technique for the ICP algorithm with a genetic algorithm to overcome this problem. Finally, we verify the 3D pose estimation results to evaluate if the object is suitable for manipulator picking and placing. Figure 1 shows a flowchart of the proposed approach.

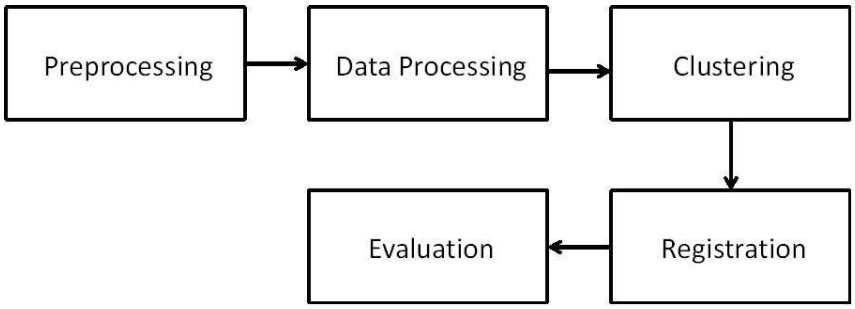


FIGURE 1. Flowchart of the proposed approach

3.1. **Preprocessing.** Because the dimensions of the created CAD model may not be equivalent to the dimensions of the real-world object, the system must scale the CAD model and make it comparable to the captured 3D object data for pose estimation [6,14]. Let the scales of the object and the CAD model be S and M, respectively. Let the CAD model be represented by $P(V)$, where $V = \{v_1, v_2, \ldots, v_n\}$, and let n be the number of the points. The new CAD model with the same scale as the object is given by

$$P_{new}(V) = \mu \cdot P(V), \tag{5}$$

where $\mu = \frac{S}{M}$.

To convert the point cloud from the CAD mesh model, we must fill the points in the triangles of the CAD mesh model. Let $F(k_m) = (v_{m1}, v_{m2}, v_{m3})$ represent the triangles of the mesh, where $m = 1, 2, \ldots, N$; N is the number of triangles and v_{m1}, v_{m2}, and v_{m3} are the triangle vertices. We first define the vectors

$$\vec{v}_1 = v_{m2} - v_{m1} \text{ and } \vec{v}_2 = v_{m3} - v_{m1}. \tag{6}$$

Then a new point inside the triangle can be generated by

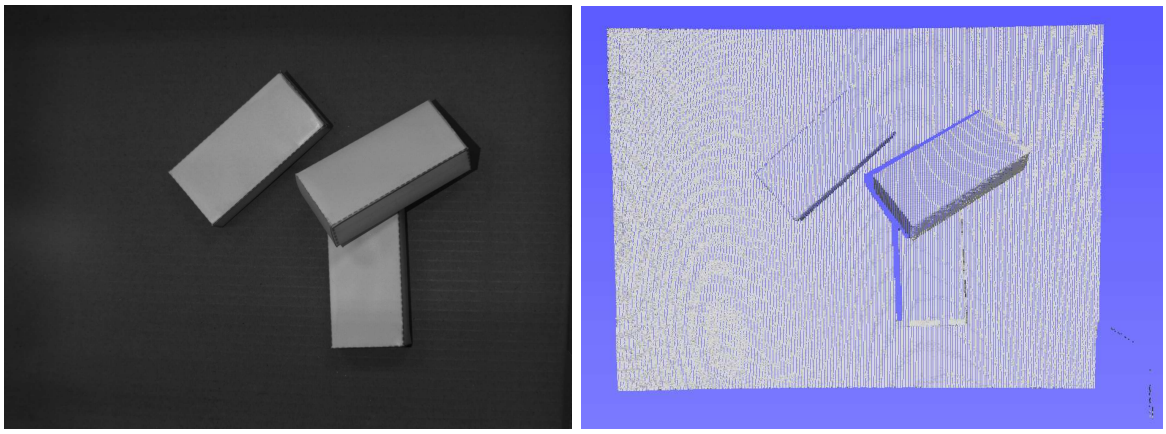
$$v_{new} = v_{m1} + \alpha\vec{v}_1 + \beta\vec{v}_2, \tag{7}$$

where α and β are arbitrary constants with the conditions

$$0 \leq \alpha,\ 0 \leq \beta,\ \alpha + \beta \leq 1. \tag{8}$$

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

3.2. **Data processing.** To capture the point cloud of the scene model, a depth camera is placed above the objects, facing downwards. Several cuboids of the same size are used to simulate a real pick-and-place task. Figure 2 shows the image captured by a structured light depth camera and the corresponding 3D point cloud. The data set contains 233,491 points. Because a large number of points will cause the pose estimation algorithm to run slowly, the points are down-sampled using voxel grid filtering [21] for further processing.



(a) (b)

FIGURE 2. Real scene with several objects: (a) the original image; (b) point cloud data points

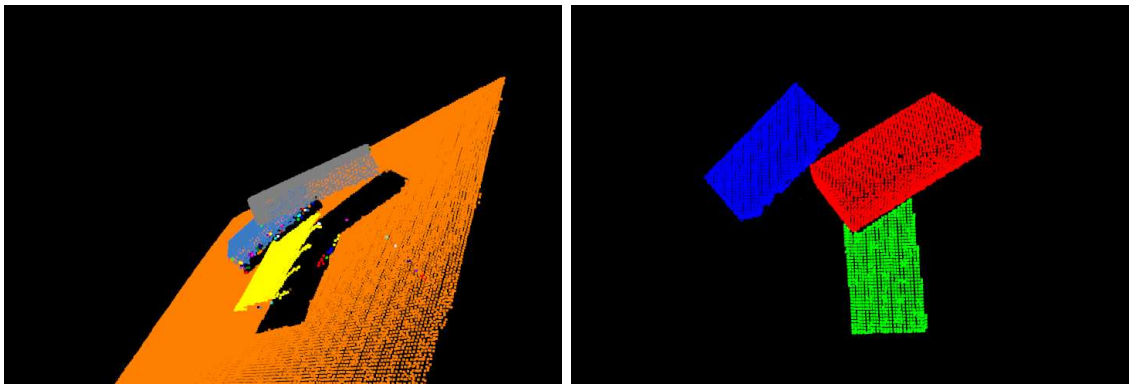
3.3. **Clustering.** The 3D point cloud of a scene model is processed with Algorithm 1 to cluster the objects. For an input point cloud of a scene model, we first build a kd-tree for the point cloud. Then we select an unvisited point and push this point into a stack buffer. After that, we pop a point a from the stack buffer and use point a to search neighbor points in the kd-tree. If the distance from a neighbor point to point a is less than a threshold, we push the neighbor point into the stack buffer and the neighbor point belongs to the same object as point a. If the stack buffer is empty, then we find all points of an object and repeat Step 2; else repeat Step 3. If all points have been visited, then we find all objects and the algorithm terminates.

Figure 3(a) shows the clustering result. It contains a substantial quality of noise and the background object. The noise has fewer points and the background object has more points.

Algorithm 1 Clustering

Input: p (point cloud of a scene model)
Output: clustering objects

1. Build a kd-tree for p.
2. Select an unvisited point and then push it back to a stack buffer.
3. Let a be a point that pops up from the stack buffer, and then let a be a search point to find neighbor points in the kd-tree.
4. If the distance from a neighbor point to a is less than a threshold t, we push back that point to the stack buffer and that point belongs to the same object as a.
5. If the stack buffer is empty, then repeat Step 2; else repeat Step 3.
6. The algorithm terminates if all points have been visited.



(a) (b)

FIGURE 3. Clustering objects: (a) clustering result; (b) filtered result

Thus, we set the double thresholds T_{lower} and T_{upper} to filter the noise and background object.

$$T_{lower} \leq X_{number} \leq T_{upper}, \tag{9}$$

where X_{number} is the number of points of the background object. The values of T_{lower} and T_{upper} are determined by heuristics. We first set different values of the lower and upper thresholds for the experiments. Then we choose the appropriate values for the thresholds. Figure 3(b) shows the filtered result.

3.4. **3D pose estimation.** The original ICP algorithm is commonly affected by the initial position of the object. Therefore, the 3D pose estimation will possibly not converge to the correct results. If extra movement is applied to an incorrect pose, the system has a higher than usual probability of avoiding an incorrect local minimum in the next iteration. Thus, we propose a technique for using a genetic algorithm [12] to provide additional poses to adjust the model position and find the correct 3D pose. The idea is to substantially adjust the overall pose with characteristics from the genetic algorithm, and then to apply the ICP algorithm to achieving the correct pose.

Consider six unknown parameters, including the translation vector (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

$$E = \sum \|Rp + T - q\|^2, \tag{10}$$

where $R = R_z(\gamma)R_y(\beta)R_x(\alpha)$, $T = [M_x\ M_y\ M_z]$, p is the scene model, and q is the CAD model. Because the computational cost of the genetic algorithm is relatively high, it is

performed only if the following conditions hold:

$$|E(k+1) - E(k)| \leq 10^{-6} \text{ and } E(k+1) \geq T. \tag{11}$$

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 first condition means the ICP solution has fallen into a local minimum, and the second condition indicates the current 3D pose is not correct. Each iteration of the genetic algorithm uses the current pose to calculate the fitness function. After the calculation has been completed, the genetic algorithm obtains a solution, and the current pose information is updated. The iteration of the ICP algorithm then continues. Note that the genetic algorithm does not always give the optimal solution or find the correct pose.

3.5. **Verification.** We verify the accuracy of the 3D pose estimation. Because the scene and CAD models are obtained from the depth camera and the computer graphics model, respectively, the density distributions of the input data from different sources are inconsistent. As a result, the 3D points of the scene model and the generated point cloud of the CAD model do not overlap point-wise, even with a correct 3D pose. Thus, we propose two methods for evaluating 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 Algorithm 2 and Algorithm 3, respectively.

Algorithm 2 Ratio of the overlay

Input: A CAD model (q) with C points and a scene model (p) with S points

Output: The ratio of the overlay (%)

1. Build a kd-tree for q.
2. Let a be a search point to find neighbor points in the kd-tree. If the distance from a neighbor point to a is less than a threshold t, we mark the neighbor point.
3. Accumulate the number of the marked points as N_{marked}.
4. The ratio $= \frac{N_{marked}}{C} \times 100\%$.

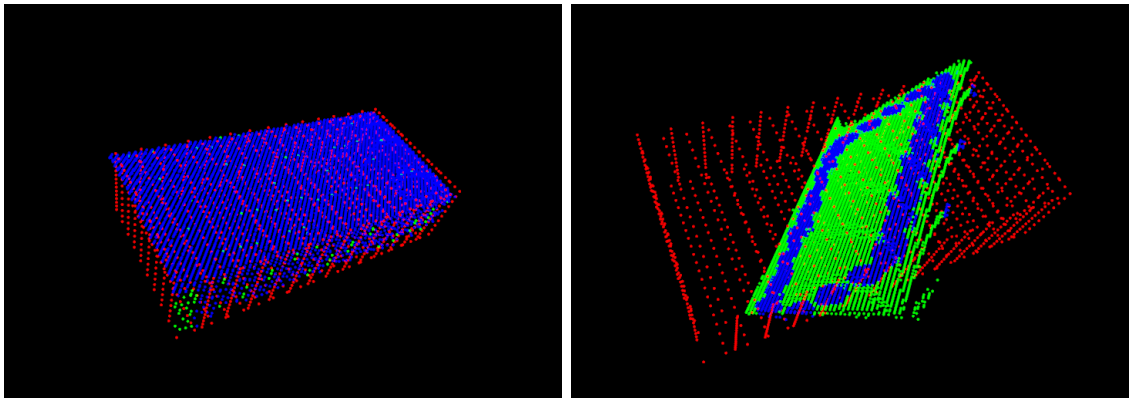
Algorithm 3 Inverse of the ratio of the overlay

Input: A CAD model (q) with C points and a scene model (p) with S points

Output: The inverse of the ratio of the overlay (%)

1. Build a kd-tree p.
2. Let a be a search point to find neighbor points in the kd-tree. If the distance from a neighbor point to a is less than a threshold t, we mark the neighbor point.
3. Accumulate the number of the marked points as N_{marked}.
4. The ratio $= \frac{N_{marked}}{S} \times 100\%$

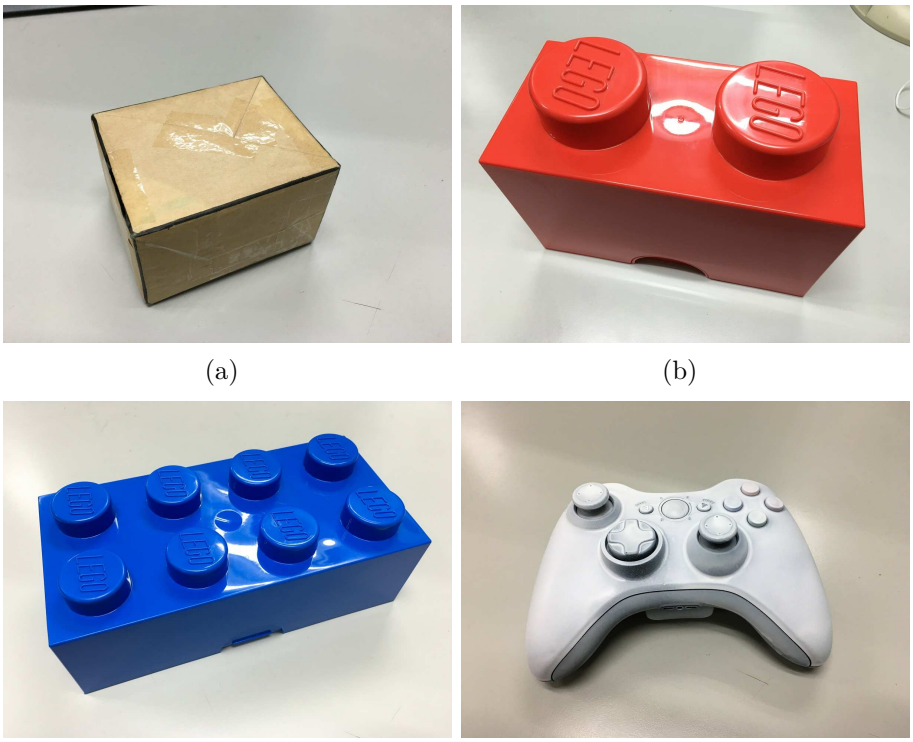
Figure 4 shows an example of our evaluation and verification technique. The correct pose result is shown in Figure 4(a) with a ratio of overlap of approximately 52%. In other words, 52% of the points in the region overlap between the CAD model and the scene model. The inverse of the ratio of overlap is approximately 99%, which means that 99% of the points in the scene model overlap with the CAD model. Figure 4(b) shows an example



(a) (b)

FIGURE 4. Verification and evaluation: (a) large overlap ratio; (b) small overlap ratio

of incorrect 3D pose estimation. The CAD model points provide only approximately 29% overlap with the scene model (i.e., the ratio of overlap), and the inverse of the ratio of overlap is approximately 34%. Based on our evaluation method, the ratio of overlap is lower if an object is occluded by another. For a pick-and-place application, a high ratio indicates that the 3D pose estimation is more accurate and is generally more suitable for the application.



(a) (b)

(c) (d)

FIGURE 5. Test objects used in the experiments: (a) box; (b) 1×2 LEGO brick, (c) 2×4 LEGO brick, and (d) controller

4. **Experimental Results.** This section presents our experimental results, including the accuracy of the estimated 3D pose and the simulation system with a virtual camera and pick-and-place operations on multiple objects in the virtual scene. In the experiments, the objects were segmented perfectly with our clustering algorithm. In each run, a single object was used to evaluate the pose estimation method. Four test objects, a box, a 1 × 2 LEGO brick, a 2 × 4 LEGO brick and a game console controller, as shown in Figure 5, were used for our experiments. Figures 6 and 7 show the CAD models and the scene models captured by a depth camera, respectively.

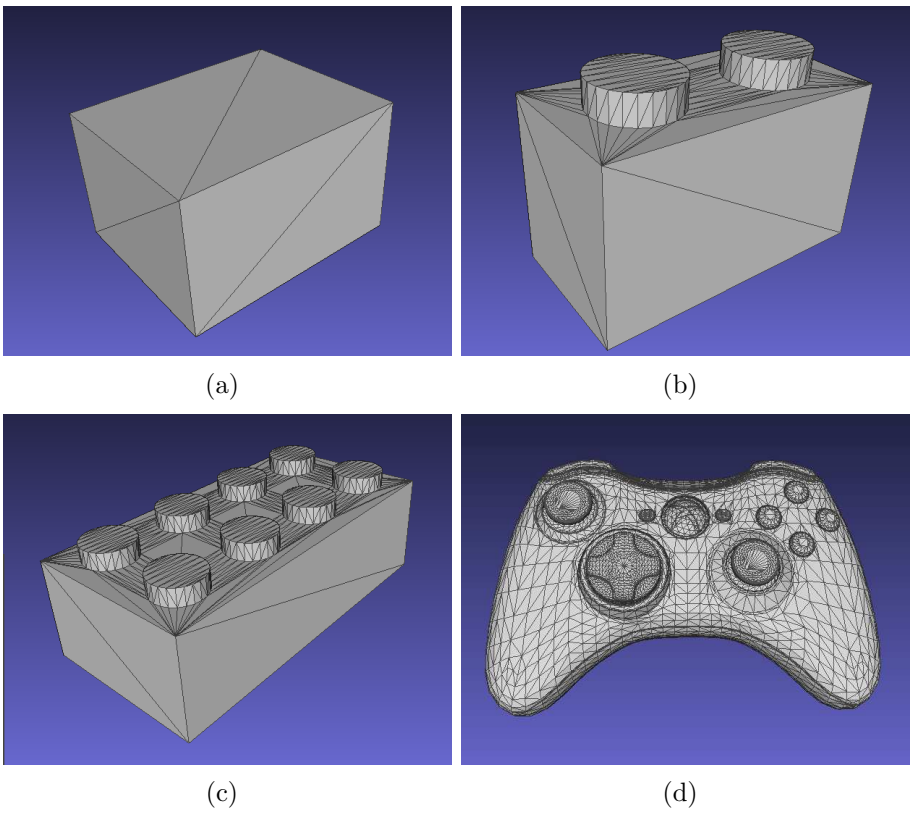
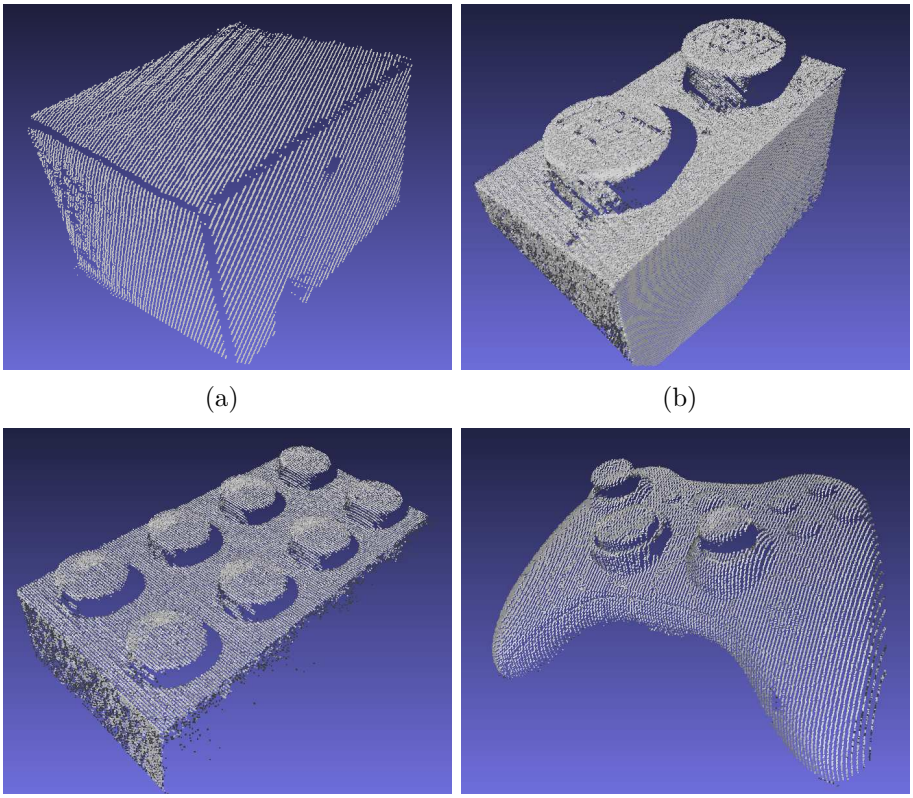


FIGURE 6. CAD models of the objects: (a) box, (b) 1 × 2 LEGO brick, (c) 2 × 4 LEGO brick, and (d) controller

We used 34 initial poses of the CAD model to evaluate our pose estimation technique. The poses were set up by rotating the CAD model through angles of 30° to 330° along the x, y, and z axes. To test affection of noise models for pose estimation results, we used two noise models, namely, uniform noise and Gaussian noise, to add noise to point clouds of CAD models. And we used the original ICP algorithm for pose estimation. The results are shown in Table 1. The accuracies of 3D pose estimation were lower because the ICP algorithm was affected by the initial position of the object. From the results, the accuracies were not greatly affected by noise models.

The comparisons between two methods, the ICP algorithm and ICP with genetic algorithm (ICP-G), are shown in Tables 2 and 3. The two types of registration models were the CAD models and scene models. The maximal iteration number of the ICP algorithm was set to 400, but this number of iterations was only used if no solution had converged earlier. For the ICP with genetic algorithm, if the ICP converged, then a new pose was assigned for further computation.



(a) (b)

(c) (d)

FIGURE 7. Scene model from a depth camera: (a) box, (b) 1 × 2 LEGO brick, (c) 2 × 4 LEGO brick, and (d) controller

TABLE 1. Noise models for pose estimation results (in percentages)

	Box	1×2 LEGO	2×4 LEGO	Controller
Original	58.82	5.88	8.82	23.53
Uniform	58.82	5.88	8.82	26.47
Gaussian	58.82	2.94	5.88	26.47

TABLE 2. Pose estimation overlap (in percentages)

	Box	1×2 LEGO	2×4 LEGO	Controller
ICP	61.76	8.82	8.82	29.41
ICP-G	98.24	84.12	90.89	84.99

TABLE 3. Pose estimation computation (in seconds)

	Box	1×2 LEGO	2×4 LEGO	Controller
ICP	0.22	2.26	2.26	2.16
ICP-G	0.61	7.30	3.17	20.18

In the experiments, the box object was a simple model, and the accuracies of ICP and ICP-G were 61.76% and 98.24%, respectively. For the 1 × 2 LEGO brick, which was more complex than the box, the accuracy of ICP-G was much higher (84.12% vs. 8.82%) at the cost of much longer computation time (7.30 vs. 2.26). Similar results can be seen

for the complex objects, namely the 2×4 LEGO brick and the controller, where ICP-G provided higher accuracy but required more computation.

In the simulation system, we simulated a virtual camera with perspective projection incorporated with a manipulator for pick-and-place task [16]. The mesh-based CAD model of an object was placed in front of the virtual camera and each pixel of the image (at a resolution of 2080×1552) was back-projected into the 3D space to find the object points. Considering all of the back-projected rays, we calculated the intersections of the vectors and surface triangles of the CAD model [18].

To simulate a manipulator that completed the pick-and-place task, we considered 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 picked an object and placed it on the conveyor belt according to the derived location and orientation of the object. We designed a five-axis manipulator with controllable rotation angles $(R_1, R_2, R_3, R_4, R_5)$ and a pair of claws to catch objects in the virtual environment. The robot arm was operated according to the previously discussed functions with rotation angle parameters. On the robot arm, two points on the claw were defined as touch points for object manipulation. The midpoint (x, y, z) denoted the initial position, and the target position from the estimated pose was denoted as (m_x, m_y, m_z). In the experimental setup, the rotation angles R_1, R_2, R_3, and R_4 were solved by the genetic algorithm with a constraint on the ranges of rotation angles.

5. **Conclusions.** We have presented a model-based 3D pose estimation approach that is applicable to pick-and-place tasks. The pose estimation results of the conventional ICP-based approaches are usually affected by the initial poses of the objects. We have proposed an ICP technique augmented with a genetic algorithm to overcome this problem. The performance of our technique in a simulated environment has demonstrated the feasibility of our approach.

If the object clustering step cannot calculate a precise clustering, an error will occur in the next step. Moreover, if the objects are positioned close together, the probability that the proposed clustering cannot represent objects perfectly is high. Therefore, future research should improve this clustering technique.

Acknowledgment. This paper is a revised and expanded version of a paper entitled, "Model-Based 3D Pose Estimation for Pick-and-Place Application", presented at the Fifteenth IAPR International Conference on Machine Vision Applications, Nagoya, Japan, 2017.

REFERENCES

[1] M. T. Ahmed, M. Mohamad, J. A. Marshall and M. Greenspan, Registration of noisy point clouds using virtual interest points, *Proc. of the 12th Conference on Computer Robot Vision*, 2015.

[2] D. Aiger, N. J. Mitra and D. Cohen-Or, 4-points congruent sets for robust pairwise surface registration, *ACM Trans. Graphics*, vol.27, no.3, 2008.

[3] 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, *Proc. of IEEE International Conference on Computer Vision Workshops*, pp.585-592, 2011.

[4] P. J. Besl and N. D. McKay, A method for registration of 3-D shapes, *IEEE Trans. Pattern Analysis Machine Intelligence*, vol.14, no.2, pp.239-256, 1992.

[5] J. Y. Chang, R. Raskar and A. Agrawal, 3D pose estimation and segmentation using specular cues, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[6] H. Chui and A. Rangarajan, A new point matching algorithm for non-rigid registration, *Computer Vision and Image Understanding*, vol.89, nos.2-3, pp.114-141, 2003.

[7] B. Drost, M. Ulrich, N. Navab and S. Ilic, Model globally, match locally: Efficient and robust 3D object recognition, *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.998-1005, 2010.

[8] M. A. Fischler and C. B. Robert, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM*, vol.24, no.6, pp.381-395, 1981.

[9] S. C. Liang, H. Y. Lin and C. C. Chang, Model-based 3D pose estimation for pick-and-place application, *Proc. of the 15th IAPR International Conference on Machine Vision Applications*, Nagoya, Japan, 2017.

[10] C. C. Lin, Y. C. Tai, J. J. Lee and Y. S. Chen, A novel point cloud registration using 2D image features, *EURASIP Journal on Advances in Signal Processing*, vol.5, 2017.

[11] N. Mellado, D. Aiger and N. J. Mitra, Super 4PCS fast global point cloud registration via smart indexing, *Computer Graphics Forum*, vol.33, no.5, pp.205-215, 2014.

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

[13] V. Renò, M. Nitti, T. D'Orazio and E. Stella, A modified iterative closest point algorithm for 3D point cloud registration, *Computer-Aided Civil and Infrastructure Engineering*, vol.31, no.7, 2016.

[14] R. B. Rusu, N. Blodow and M. Beetz, Fast point feature histograms (FPFH) for 3D registration, *Proc. of IEEE International Conference on Robotics and Automation*, pp.3212-3217, 2009.

[15] R. B. Rusu and S. Cousins, 3D is here: Point cloud library (PCL), *Proc. of IEEE International Conference on Robotics and Automation*, pp.1-4, 2011.

[16] R. B. Rusu, A. Holzbach, M. Beetz and G. Bradski, Detecting and segmenting objects for mobile manipulation, *Proc. of IEEE the 12th International Conference on Computer Vision Workshops*, pp.47-54, 2009.

[17] J. Seran and G. Grisetti, NICP: Dense normal based point cloud registration, *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.742-749, 2015.

[18] O. Tropp, A. Tal and I. Shimshoni, A fast triangle to triangle intersection test for collision detection, *Computer Animation and Virtual Worlds*, vol.17, no.5, pp.527-535, 2006.

[19] X. Wang, H. Zhang and G. Peng, 3-DOF point cloud registration using congruent triangles, *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.1943-1948, 2015.

[20] C. Zach, A. Penate-Sanchez and M. T. Pham, A dynamic programming approach for fast and robust object pose recognition from range images, *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp.196-203, 2015.

[21] *Point Cloud Library*, http://pointclouds.org/.