Determination of 3D Object Pose in Point Cloud with CAD Model

Duc Dung Nguyen, Jae Pil Ko, and Jae Wook Jeon
School of Electrical and Computer Engineering
Sungkyunkwan Univ., Korea 440–746
Email: nddunga3@skku.edu, kosama@skku.edu, jwjeon@yurim.skku.ac.kr

***Abstract*—This paper introduces improvements to estimate 3D object pose from point clouds. We use point-pair feature for matching instead of traditional approaches using local feature descriptors. In order to obtain high accuracy estimation, a discriminative descriptor is introduced for point-pair features. The object model is a set of point pair descriptors computed from CAD model. The voting process is performed on a local area of each key-point to boost the performance. Due to the simplicity of descriptor, a matching threshold is defined to enable the robustness of the algorithm. A clustering algorithm is defined for grouping similar poses together. Best pose candidates will be selected for refining and final verification will be performed.**

The robustness and accuracy of our approach are demonstrated through experiments. Our approach can be compared to state-of-the-art algorithms in terms of recognition rates. These high accurate poses especially useful for robot in manipulating objects in the factory. Since our approach does not use color feature, it is independent to light conditions. The system give accurate pose estimation even when there is no light in the area.

I. Introduction

For years, the robot industry has been going through enormous development. Robot nowadays not only receives commands from the computer/human but also has the ability to make decision itself based on various input sources. With the appearance of depth sensors, robot can now recognize shape of objects. Object recognition, therefore, becomes an important part of robotic perception. On the other hand, the quality of depth sensors lead to different approaches in object recognition. In this work, the Kinect camera is used to capture depth image for recognition task. Kinect camera is affordable, and is available for a wide range of applications in both industry and academy.

One popular approach in object recognition is keypoint matching. It has been well known that stable keypoint descriptors successfully made their way in object recognition field [1], [2]. These keypoint descriptors are invariant to to illumination changes and geometric transformation, and keypoint correspondences can be determined reliably. In order to obtain 6-DOF pose, the keypoint coordinates in 3D must be estimated. These coordinates can be determined using structure from motion [3], or back-projecting 2D keypoints to 3D CAD model [4]. Since these descriptors work for texture objects, they fail for textureless objects, which are the common cases. The edge feature turns out to be a solution for this case since it can represent object boundaries. The common approach is edge matching between the edge image with a set of edge image templates, known as priories. The recent work involving this approach is LINEMOD [5], [6]. LINEMOD exploits both depth and color images to capture the appearance and 3D shape of the object in a set of templates covering different views of the object. That being said, LINEMOD still suffers from the presence of false positives and noise.

With the rise of 3D data usage for perception tasks, 3D keypoint descriptors become useful tools in object recognition and pose estimation. A set of popular descriptors are available in PCL (Point Cloud Library) [7]. These descriptors encoding local information of the keypoint can be categorized as geometric based descriptors. Descriptors such as PFH (Point Feature Histogram) [8], FPFH (Fast Point Feature Histogram) [9], and VFH (Viewpoint Feature Histogram) [10] are in this group. These descriptors encode relative orientation of normals and sistances between point pairs in a fixed coordinate frame, which is constructed using the surface normal of the keypoint. Changes in the surface normal of keypoint can lead to significant changes in the descriptors. Thus, these descriptors are considered to be sensitive to noise. On the other hand, SHOT (Signature of Histogram of Orientation) [11] represents topological traits by encoding a signature of histogram. For each spherical grid sector, one dimensional histogram is constructed by accumulating angles between point normals and keypoint normal. Histograms is orderly juxtaposing according to the local reference frame. SHOT, therefore, is invariant to rotation and translation, and robust to noise and clutter. That being said, even SHOT descriptor suffers from the effect of noise if noise occurs near the keypoint. These descriptors then will be matched with the model, usually following by correspondence grouping to estimate 3D pose. This approach works for some experimental cases but it failed in our test, as we will show later in the manuscript. Along with these, other local invariant features have been proposed such as surface curvature [12], spin image [13], and surface normal distribution [14].

The segmented point cloud can also be used for pose estimation. In another approach, Rusu *et al.* proposed Viewpoint Feature Histogram (VFH) descriptor [10] for this purpose. It encodes angular distributions of surface normals on a segmented point cloud. This makes VFH suffer from occlusion, and therefore, a full pose estimation cannot be obtained. A global descriptor called CVFH (Clustered Viewpoint Feature Histogram) [15] was proposed for this case. Although recognizing object poses efficiently, these approaches rely on the point cloud segmentation result. That being said, the noise and quality of point cloud still have great impacts on these approaches.

(a)

(b)

Fig. 1. The target object (a) CAD model, (b) real objects.

On the other hand, point-pair features (PPF) has been proved to be an efficient feature for 3D-recognition since it was introduced by Drost *et al.* [16]. Due to its simplicity, PPFs can be computed fast and be matched easily with the model. The object model is defined as a set of PPFs computed from model point cloud. For fast searching, those PPFs is stored in a hash table as in [17]. These PPFs can be seen as a successor of surflet pairs [18]. Points will be sampled from the scene and PPFs will be matched with the model. Every matched pair will vote for a pose hypothesis. Finally, the high votes represent possible object pose hypotheses. Recently, this approach was improved by incorporating visibility context [19], or considering object boundary information. It has been noted that these PPFs is well-suited for recognizing objects that have rich variations in the surface normals. It is not very efficient in representing planar surface or self-symmetric objects. Even though this problem can be solved through voting process, it greatly degrades the performance of the algorithm. Another improvement, which include color information in PPF descriptor, was introduced in [20] to over come this problem. With color information in the descriptor, the hash function can reduce number of PPFs that fall into the same slot. It employs the idea that the color information can prune potential false matches and thus improve the matching process. There are also various modified Hough transformation for estimating 3D object pose [21] using SRT distance [22].

Color information is useful in removing false matches and eliminating unnecessary checks in the hash table. It is, however, sensitive to light conditions as well as object material. In practice, the light reflection on object surface can significantly change the color of the object. The surface textures also contribute to variance in the color. In some cases, the material such as metal can be rusted and change the color which lead to missed, or wrong matches. In this work, we perform manipulation task on objects in the factory, which are mostly metal objects. The object is defined by a CAD model as in Fig. 1a. Fig. 1b shows color variation on the object surface. This cause difficulties for detection method based on color information such as [20]. In this work, we propose a discriminant point-pair feature, which is inspired by the work in [16], [20]. To improve performance, we define the local feature as a set of PPFs inside the sphere centered at the keypoint. In addition, we propose an alternative pose clustering algorithm to reduced number of poses to be verified. The following experiment section will demonstrate the performance of the proposed algorithm and the accuracy of the estimated poses.

II. 3D Pose estimation

Most objects in the factory have associated mesh models, which make it more convenient for the manipulation task. However, back projection points onto the mesh is inefficient, especially when the mesh is huge and complex. In addition, transforming the mesh to the scene cloud for verifying will also result in high computation cost. To resolve this problem, we perform surface sampling to convert the mesh to point cloud. This object point cloud is downsampled using voxel filter before calculating features.

A. Pair feature descriptor

The pair feature is constructed from two distinct points along with their normals. Let $\mathbf{p}_i$ be the location of point *i*-th in the point cloud, and let $\mathbf{n}_i$ be the associated normal of point *i*-th. The point-pair feature (PPF) of two points *i*-th and *j*-th is defined in [16], [20] as follows:

$$\mathbf{F}_{PPF}(\mathbf{p}_i, \mathbf{n}_i, \mathbf{p}_j, \mathbf{n}_j) = \begin{bmatrix} \|\mathbf{v}_{ij}\|_2 \\ \angle(\mathbf{n}_i, \mathbf{v}_{ij}) \\ \angle(\mathbf{n}_j, \mathbf{v}_{ij}) \\ \angle(\mathbf{n}_i, \mathbf{n}_j) \end{bmatrix} \tag{1}$$

where $\mathbf{v}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ is the vector from $\mathbf{p}_i$ to $\mathbf{p}_j$ and the term $\angle(\mathbf{a}, \mathbf{b}) \in [0; \pi]$ denotes the angle between two vectors $\mathbf{a}$ and $\mathbf{b}$. The first element $\|\mathbf{v}_{ij}\|_2$ is the Euclidean distance between two points on the surface. The second and third elements encode angles between normal vectors and $\mathbf{v}_{ij}$. The last element represent the angle between two normal vectors $\mathbf{n}_i, \mathbf{n}_j$. This descriptor is ambiguous since there are at least two pair features having the same descriptor.

Even though this descriptor effectively encodes geometric information of the surface, its ambiguity leads to wrong matches. To overcome this problem, we propose a discriminant descriptor that uniquely represent the pair feature. The proposed descriptor is a four components vector $\mathbf{F} \in \mathrm{R}^4$ as follows:

$$\mathbf{F}(\mathbf{p}_i, \mathbf{n}_i, \mathbf{p}_j, \mathbf{n}_j) = \begin{bmatrix} \|\mathbf{v}_{ij}\|_2 \\ \angle(\mathbf{n}_i, \mathbf{v}_{ij}) \\ \angle(\mathbf{n}_j, \mathbf{v}_{ij}) \\ \theta \end{bmatrix} \tag{2}$$

Algorithm 1: Building Object model

Input: Mesh of object model
Output: Hash table of point-pair descriptors: $\mathcal{M}$

Generate point cloud from the mesh
Down-sampling model point cloud
$\mathcal{M} \coloneqq \varnothing$
for $i \coloneqq 1 \rightarrow N$ **do**
  for $j \coloneqq 1 \rightarrow N$ **do**
    if $i \neq j$ **then**
      Compute descriptor $\mathbf{F}(\mathbf{p}_i, \mathbf{n}_i, \mathbf{p}_j, \mathbf{n}_j)$
      h = hash($\mathbf{K}(\mathbf{F})$)
      $\alpha_m = \text{RotAngle}(\mathbf{p}_i, \mathbf{n}_i, \mathbf{p}_j)$
      insert($\mathcal{M}$, $[\mathbf{F}, \alpha_m, i]^\top$, h)

The first three elements are similar to original PPF. Note that the angles depend on direction of vector $\mathbf{p}_{ij}$:

$$\begin{aligned} \angle(\mathbf{n}_i, \mathbf{v}_{ij}) &= \text{acos}(\mathbf{n_i} \cdot \mathbf{v}_{ij}) \\ \angle(\mathbf{n}_j, \mathbf{v}_{ij}) &= \text{acos}(\mathbf{n_j} \cdot \mathbf{v}_{ij}) \end{aligned} \tag{3}$$

Next, we compute the normal vectors of planes $(\mathbf{v}_{ij}, \mathbf{n}_i)$ and $(\mathbf{v}_{ij}, \mathbf{n}_j)$ as follows

$$\begin{aligned} \mathbf{u}_i &= (\mathbf{v}_{ij} \times \mathbf{n}_i)/\|\mathbf{v}_{ij} \times \mathbf{n}_i\|_2 \\ \mathbf{u}_j &= (\mathbf{v}_{ij} \times \mathbf{n}_j)/\|\mathbf{v}_{ij} \times \mathbf{n}_j\|_2 \end{aligned} \tag{4}$$

The angle θ between $\mathbf{u}_i$ and $\mathbf{u}_j$ is calculated based on the direction of vector $\mathbf{v}_{ij}$ as follows:

$$\theta = \text{atan2}(\mathbf{v}_{ij} \cdot (\mathbf{u}_i \times \mathbf{u}_j), \mathbf{u}_i \cdot \mathbf{u}_j) \tag{5}$$

where the angle θ is defined in range $[0; 2\pi)$. With this definition, the descriptor $\mathbf{F}$ defines a unique pair feature, with respected to the reference point $\mathbf{p}_i$ and direction $(\mathbf{v}_{ij}, \mathbf{u}_i)$.

B. Object model

For training stage, the object model must be learned. In this case, the object model is defined as a set of PPF descriptors calculated from every point pair of the object point cloud. In [20], the authors retrieve object model by capturing several object point clouds from different angles. While this approach can easily deal with real object without CAD models, it suffers from noise. In our case, Kinect camera is used for capturing the scene cloud. The quality of point cloud captured from Kinect camera is good for normal objects but metal surfaces. Corrupted clouds reduces the accuracy of the training process. Errors in the training can lead to larger errors in matching step. Therefore, using CAD model in the training stage is always a better choice.

The CAD model will be sampled to generate object point cloud along with normal vectors. The object cloud is then down sampled using voxel filter to reduce data size and make the matching is faster. The PPF descriptor will be calculated for every point-pair on the object cloud. These descriptors are hashed and are stored in a hash table for fast retrieving. That being said, many of point pairs has the same descriptor and we must same all of them in the table. This causes inevitable collisions, which reduce performance of searching. The vector to be hashed is given as follows:

$$\mathbf{K}(\mathbf{F}) = \begin{bmatrix} \lfloor \|\mathbf{v}_{ij}\|_2 / \sigma_d \rfloor \\ \lfloor \angle(\mathbf{n}_i, \mathbf{v}_{ij}) / \sigma_\theta \rfloor \\ \lfloor \angle(\mathbf{n}_j, \mathbf{v}_{ij}) / \sigma_\theta \rfloor \\ \lfloor \theta / \sigma_\theta \rfloor \end{bmatrix} \tag{6}$$

where σ_d and σ_θ are quantization levels for distance and angle, respectively. We use JDB hash algorithm to hash the key in (6). Algorithm 1 describes the process of building object descriptor. This hash table stores not only descriptors $\mathbf{F}$ but also index of the reference point i and the angle α_m between the plane and xy-plane. This allow computing the final pose of the object using intermediate transformation.

Algorithm 2: Pose estimation

Input: Scene point cloud $\mathfrak{S}$, object model $\mathcal{M}$
Output: List of estimated poses

Down-sampling scene point cloud
$\mathcal{P} \coloneqq \varnothing$
for $i \in \mathcal{S}$ **do**
  Reset vote table $\mathbf{C}_\alpha$
  for $j \coloneqq 1 \rightarrow N$ **do**
    if $i \neq j$ *or* $\|\mathbf{v}_{ij}\|_2 > \tau$ **then**
      Compute $\mathbf{F}(\mathbf{p}_i, \mathbf{n}_i, \mathbf{p}_j, \mathbf{n}_j)$ on $\mathfrak{S}$
      $\alpha_s = \text{RotAngle}(\mathbf{p}_i, \mathbf{n}_i, \mathbf{p}_j)$
      $\mathbf{L} = \text{search}(\mathcal{M}, \mathbf{K}(\mathbf{F}), \delta)$
      for $(\mathbf{K}_m, \alpha_m, i_m) \in \mathbf{L}$ **do**
        $\alpha = \alpha_s - \alpha_m$
        $\mathbf{C}_\alpha(i_m, \lfloor \frac{\alpha}{\sigma_\theta} \rfloor) = \mathbf{C}_\alpha(i_m, \lfloor \frac{\alpha}{\sigma_\theta} \rfloor) + 1$
  Compute intermediate transform $\mathbf{T}_{sg}$
  Pick high votes in $\mathbf{C}_\alpha$: $\mathcal{V}$
  for $(\alpha_m, i_m) \in \mathcal{V}$ **do**
    Compute intermediate transform $\mathbf{T}_{mg}$
    Compute object pose $\mathbf{P}$
    $\mathcal{P} \coloneqq \mathcal{P} \cup \mathbf{P}$
Clustering poses in $\mathcal{P}$
Verifying poses

C. Pose estimation

Given a scene point pair and a matched pair on the model, a transformation can be estimated. This transformation, unfortunately, is 12-dimensions vector. With a huge number of matched pairs, the pose clustering and verification process become impractical. The voting scheme is an efficient approach to solve this problem [16], [20]. First, both point-pairs are aligned in a reference frame, usually camera origin. The position alignment constrains 3-DOF and the normal alignment constrains 2-DOF rotation. After the alignment, the 6-DOF transformation can be reduced to one rotation operation. The alignment puts the reference point $\mathbf{p}_i$ at the origin and the normal vector $\mathbf{n}_i$ will be aligned along fixed axis (in this case is x-axis). Let $\mathbf{n}_x$ be the unit vector of x-axis, and $\mathbf{R}_{\mathbf{n},\alpha}$ be the matrix that rotates around $\mathbf{n}$-axis a angle α. Then, the intermediate transformation is calculated as follows:

$$\mathbf{T}_{mg} = \begin{bmatrix} \mathbf{R}_{\mathbf{v},\beta} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{p}_i \\ \mathbf{0} & 1 \end{bmatrix} \tag{7}$$

where $\mathbf{I}$ is 3×3 identity matrix, and $(\mathbf{v}, \beta)$ is calculated as:

$$\mathbf{v} = (\mathbf{n}_1 \times \mathbf{n}_x)/\|\mathbf{n}_1 \times \mathbf{n}_x\|_2 \quad (8)$$

$$\beta = \text{acos}(\mathbf{n}_1 \cdot \mathbf{n}_x) \quad (9)$$

Let $\mathbf{v}_{ij}^{'}$ be the normalized vector of $\mathbf{v}_{ij}$, and $\mathbf{v}_{ij}^{''}$ is the transformed vector of $\mathbf{v}_{ij}^{'}$:

$$\mathbf{v}_{ij}^{''} = \mathbf{R}_{\mathbf{v},\beta}\mathbf{v}_{ij}^{'} \quad (10)$$

The angle α_m in algorithm 1 is computed as follows:

$$\alpha_m = \text{atan2}(-w_1, w_2) \quad (11)$$

where $\mathbf{w} = [w_0, w_1, w_2]^\top$ is the normal vector of the point-pair plane:

$$\mathbf{w} = (\mathbf{n}_x \times \mathbf{v}_{ij}^{''})/\|\mathbf{n}_x \times \mathbf{v}_{ij}^{''}\|_2 \quad (12)$$

Similarly, we can compute the intermediate transformation $\mathbf{T}_{sg}$ for the scene point-pair and the rotation angle α_s using (7) and (11). Then, the estimated pose of the object can be represented as a 4×4 matrix:

$$\mathbf{P} = \mathbf{T}_{sg}^{-1}\mathbf{R}_{\mathbf{n}_x,\alpha}\mathbf{T}_{mg} \quad (13)$$

where $\alpha = \alpha_s - \alpha_m$.

Having a point-pair in the scene, each matched pair in the model defines an object pose. The intermediate transformation of the object is known and the angle difference α is all we need to recover the object pose. This leads to the voting scheme in algorithm 2. The radius τ defines the local feature range of the key point. This improve performance of the matching process as well as reduce false matched in cluttered scene. Due to sensor noise, the point cloud is not perfect and so the normals vectors. Therefore, when searching for the matches in hash table, we select all the features with descriptor similarity smaller than a predefined threshold ζ. In addition, the angle difference need to be quantized with parameter σ_θ. Because the reference point on the model is required to recover the pose, we cast the vote on a pair (α, i_m). Therefore, the vote table $\mathbf{C}_\alpha$ is defined as a 2D matrix with N_m rows (number of model points) and $2\pi/\sigma_\theta$ columns.

After voting process, the pairs (α, i_m) with high votes will be selected as pose candidates. The pose calculation follows equation (13). Since there are many similar pose candidates, we need to perform clustering step and select only distinctive candidates for verification. Finally, we perform verification process on pose candidates to pick up the right pose of the objects in the scene.

D. Pose clustering

There are many similar poses in the pose set $\mathcal{P}$ as a result of matching descriptor $\mathbf{F}$ of keypoint $\mathbf{p}$ and remaining points around $\mathbf{p}$. It indicates that another keypoint near $\mathbf{p}$ can also result in similar poses in $\mathcal{P}$. Therefore, clustering poses in $\mathcal{P}$ is required to produce stable estimation [16], [19], [20]. Some advanced clustering algorithm such as mean sift [22], [23] usually consume lot of computation power. With high dimensional data of 3D pose (12-elements vectors), the clustering algorithm should be fast and simple. Thus, we proposed an efficient agglomerative clustering for this problem. Algorithm 3 describes steps of the algorithm. The pose is

Algorithm 3: Pose clustering

Input: Pose pool $\mathcal{P}$, weight factor $\mathbf{w}$
Output: Pose clusters $\mathcal{C}_p$

```
Sort the poses in P in decreasing order of votes
C_p := ∅
Build the search tree T for P
while P ≠ ∅ do
    Pick pose P in P
    Q = FindNearPose(T, Pw)
    if Q is invalid then
        Create cluster C = {P}
        C_p := C_p ∪ C
        remove P from P
    else
        if Q in cluster C then
            if P can be added to C then
                C := C ∪ {P}
                remove P from P
        else
            S = FindNearPose(T, Qw)
            if S = P then
                Create cluster C = {P, Q}
                C_p := C_p ∪ C
                remove P, Q from P
    if Cannot expand any cluster then
        Remove clusters from search tree T
```

defined by 12-components vector but the translation components should be treated differently to the rotation components in searching step. Therefore, we employ the weight factor $\mathbf{w}$ to control the metric between similar poses. In our case, the weight for translation components is chosen to be 100.

E. Pose verification

The poses estimated using algorithm 2 are not all corrected. The noise in point cloud and error in normal estimation can result in to wrong poses. Therefore, high votes do not guarantee correct poses. First, the pose must be refined using iterative-closest point (ICP) [24]. Then, the model point cloud will be transformed to the scene to be verified. For each point of the model, we search for the closest point in the scene. If the distance between those two is larger than sampling size, we can skip the check. Otherwise, we check if the normal of model point is fitted on the scene. Let $\mathbf{n}_{m,i}$ and $\mathbf{n}_{s,j}$ be be the normal of model point and scene point, respectively. The model point is considered to be fitted on the scene if

$$\mathbf{n}_{m,i} \cdot \mathbf{n}_{s,j} \geq \nu$$

where ν is the threshold for angle difference. Due to quality of the scene normals, we select the angle threshold to be $\pi/9$ (ie. $\nu = \cos(\pi/9)$).

The number of right and wrong normals give clues to decide which pose is correct. Normally, a pose is correct if the number of right normals is dominant. Let N_r, N_w be the number of right and wrong normals respectively. The pose is considered to be correct if $\frac{N_r}{N_r+N_w} > \kappa$ and $N_r > \eta$.

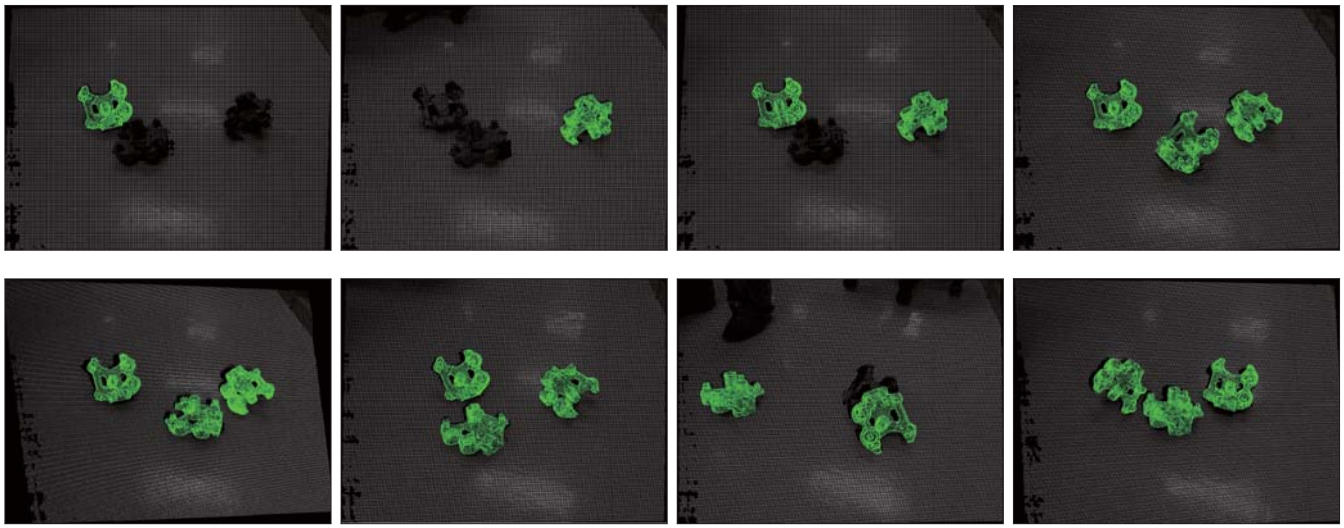

Fig. 2. Detection results from various viewpoints and poses.

It is interesting that the constant κ depend on each model. If two opposite surfaces of the object has a big gap (two times larger than the sampling size), $kappa$ must be large (e.g. 0.8). Unfortunately, this doesn't hold for objects with complex shapes and the gap of two opposite surfaces are small. As in Fig. 1, the object has many slim parts, which cause problem in verification process. Therefore, the constant κ is selected to be 0.5.

III. Experiment

The experiment is performed with depth images captured from Kinect camera. As mentioned before, the color variance in object color (see Fig. 1) makes CPPF [20] fail in detecting the object. The color of the CAD model may not fit with colors captured by camera. In addition, the point cloud quality is poor with this metal object. There are holes along the edge and inside object area. This leads to wrong estimation of surface normals, which in turn cause wrong estimation of the object pose. In this section, we compare our results to Drost *et al.* [16]. Our method provide more robust results in the synthetic tests thank to the unambiguous descriptor (2). In the experiment with the Kinect camera, our method outperform the result of Drost *et al.* [16] and provide the most accurate estimation. Since our proposed descriptor is unique for the point-pair feature, it eliminates all wrong matches. As a result, the vote result is more accurate.

The object bounding box size is $125 \times 120 \times 90mm^3$. Therefore, the search distance τ of local feature in algorithm 2 is selected to be 0.15 $(150mm)$. The quantization levels σ_d, σ_θ are selected to be 0.004 (equal with the scene sampling size, $4mm$) and $\pi/18$ respectively. Two descriptors $\mathbf{K}$ are considered to be similar if the distance between them is below 0.087 (with respect to 5° angle error). The vote threshold for selecting high votes is 80. In addition, poses are qualified if the number of right normals (N_r) is above 100 and the portion of N_r satisfies the condition in the verification algorithm.

Fig. 2 shows our detection results. The first three tests on the top row show how badly the noise can affect the detection algorithm. The holes inside objects and broken part

Fig. 3. The detection result when the light is turned off.

borders have great impacts on the results. In fact, by lowing the thresholds, the objects can be detected. However, the number of poses candidates will also rise significantly, and therefore, greatly reduce the performance of the algorithm. It should be noted that the performance is very important in this application. Therefore, we select the parameter sets that guarantees both performance and detection accuracy. After verifying, all estimated poses attain required accuracy for manipulation tasks in the factory. In addition, our proposed method works in different light conditions and also endures the noise caused by depth sensing cameras as we can observe in Fig. 3. The results show that our method is very suitable for robotic applications such as manipulating objects in the factory.

IV. Conclusion

This manuscript introduced a voting-based algorithm for 3D object detection using a discriminative pair descriptor. We also introduce some improvements in matching and voting process. An efficient clustering algorithm is provided to en-

hance performance of the processing pipeline. The verification step reject wrong poses to ensure that only accurate poses are produced in the final result. As shown in the experiment, our results is very accurate and useful for manipulating tasks. That being said, the performance of the algorithm can also be improved by further optimization steps to enable real-time performance for robot applications.

ACKNOWLEDGMENT

This work was supported by Samsung Electronics under Project Number S-2014-0253-000.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[3] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *in IEEE ICRA. Kobe: IEEE*, 2009, pp. 48–55.

[4] C. Choi and H. Christensen, "Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 498–519, 2012.

[5] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes." in *ICCV*, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds. IEEE, 2011, pp. 858–865.

[6] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. R. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes." in *ACCV (1)*, ser. Lecture Notes in Computer Science, K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, Eds., vol. 7724. Springer, 2012, pp. 548–562.

[7] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)." in *In Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2011.

[8] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3384–3391.

[9] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration." in *In Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 3212–3217.

[10] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 10/2010 2010.

[11] F. Tombari, S. Salti, and L. di Stefano, "Unique signatures of histograms for local surface description." in *ECCV (3)*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6313. Springer, 2010, pp. 356–369.

[12] C. Dorai and A. K. Jain, "Cosmos - a representation scheme for 3d free-form objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1115–1130, 1997.

[13] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[14] F. Stein and G. Medioni, "Structural indexing: Efficient 3-d object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 125–145, 1992.

[15] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. R. Bradski, "Cad-model recognition and 6dof pose estimation using 3d cues." in *ICCV Workshops*. IEEE, 2011, pp. 585–592.

[16] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, 2010, pp. 998–1005.

[17] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1584–1601, Oct. 2006.

[18] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification," in *3DIM'03*, 2003, pp. 474–482.

[19] E. Kim and G. G. Medioni, "3d object recognition in range images using visibility context." in *IROS*. IEEE, 2011, pp. 3800–3807.

[20] C. Choi and H. I. Christensen, "3d pose estimation of daily objects using an rgb-d camera." in *IROS*. IEEE, 2012, pp. 3342–3349.

[21] O. Woodford, M.-T. Pham, A. Maki, F. Perbet, and B. Stenger, "Demisting the hough transform for 3d shape recognition and registration," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 32.1–32.11.

[22] M.-T. Pham, O. J. Woodford, F. Perbet, A. Maki, B. Stenger, and R. Cipolla, "A new distance for scale-invariant 3d shape recognition and registration." in *ICCV*, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds. IEEE, 2011, pp. 145–152.

[23] O. Tuzel, R. Subbarao, and P. Meer, "Simultaneous multiple 3d motion estimation via mode finding on lie groups," in *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, vol. 1. IEEE, 2005, pp. 18–25.

[24] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.121791