applied optics

Fast and precise 6D pose estimation of textureless objects using the point cloud and gray image

Wang Pan,[1,2] Feng Zhu,[1,3,4,*] Yingming Hao,[1,3,4] and Limin Zhang[1,2]

[1]Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Key Laboratory of Opto-Electronic Information Processing, CAS, Shenyang 110016, China
[4]Key Laboratory of Image Understanding and Computer Vision, Liaoning Province, Shenyang 110016, China
*Corresponding author: fzhu@sia.cn

Received 20 April 2018; revised 2 August 2018; accepted 7 August 2018; posted 9 August 2018 (Doc. ID 328945); published 25 September 2018

Pose estimation for textureless objects is a challenging task in robotics, due to the scanty information of surfaces. In this paper, we design a vision system for fast and precise position and orientation measurement of textureless objects with a depth camera and a CCD camera. The corresponding process includes two parts: object segmentation in the point cloud and pose measurement in the gray image. Considering the relation between the object and its fixed panel, we first extract the panel in the point cloud by combining a random sample consensus algorithm with local surface normal. We then coarsely segment the possible area of the object based on an oriented bounding box. Finally, we transform the point cloud coordinates into the image coordinate system, and measure the precise pose of the object with a view-based matching method. Two types of cameras are brought together to make their respective advantages play well. The downscale method and coarse-to-fine strategy are utilized sufficiently to increase efficiency. Experiments show that our vision system achieves high pose measurement precision and enough efficiency. The average error is less than 2 mm for *x* and *y*, less than 4 mm for *z*, and 1° in orientation, meeting the requirements for our robotic grasping task.

OCIS codes: (150.4232) Multisensor methods; (150.6910) Three-dimensional sensing; (330.5000) Vision - patterns and recognition.

https://doi.org/10.1364/AO.57.008154

1. INTRODUCTION

A visual perception system that can recognize and locate objects is required for robotic grasping and manipulation [1–3]. In our application, the 6D pose (location and orientation) of an object, such as the metal handrail shown in Fig. 1(b), must be determined with high precision. In general, such objects have no texture and the surface is smooth and shiny, which causes their detection and pose estimation to be difficult. Therefore, edge-based template matching methods [4,5] are commonly used solutions at present. However, this kind of method becomes less effective and less robust in large scenes, especially when similar interference characteristics exist in cluttered backgrounds.

This paper presents a visual perception system that precisely measures the position and orientation of a textureless object fixed on a panel. The system combines a time-of-flight (ToF) active ranging camera and a high-resolution CCD camera to take advantage of the complementary nature of these two imaging technologies and allow the system to provide fast and accurate 6D pose. The depth camera has the following advantages over the CCD camera: range data can be obtained directly with a high frame rate and it is less affected by changes in lighting. The drawbacks are as follows: low resolution and accuracy, large noise, and visual features are not as intuitive as 2D images. The point cloud obtained with the depth camera and the gray image captured with the CCD camera are combined to accurately measure the pose of the object. A 3D CAD model of the object needs to be known in advance to create templates. The full process consists of point cloud segmentation and an efficient view-based matching in a gray image, as shown in Fig. 1. In the point cloud segmentation process, the region of the object is extracted from the cluttered scenes. Considering the relationship between the object and the supporting surface in 3D space, we propose a 3D region-of-interest segmentation algorithm based on an oriented bounding box to extract the possible area of the object in the point cloud first, avoiding the cluttered background of the scenes and reducing the image search space. During the gray image matching, search efficiency and correct recognition rate are improved because cluttered backgrounds are circumvented. To further improve

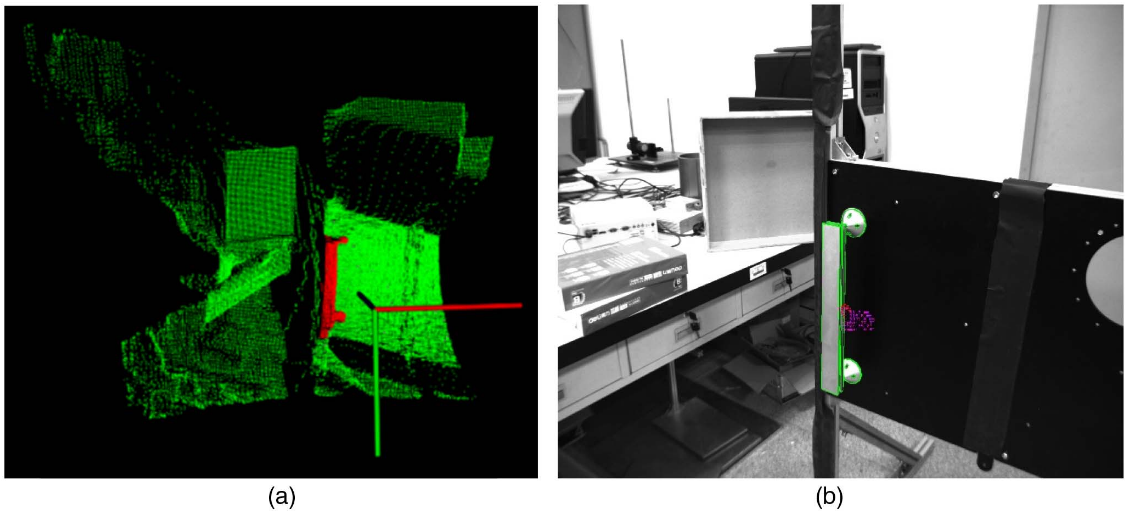

Fig. 1. Our vision system is developed for measuring the precise 6D pose of a textureless 3D object in large scenes. (a) The result of object segmentation in the point cloud; the red, green, and blue lines represent the x, y, and z axes of the depth camera coordinate system, respectively. (b) The accurate result of pose measurement in the gray image.

the efficiency, down-sampling techniques and a coarse-to-fine strategy are used fully both in the 3D point cloud and 2D image space.

Two major problems need to be settled in our case when using the state-of-art shape-based matching method [6]. One is that it is quite common to find a wrong image position in a large cluttered scene, particularly when error matching the degenerated view of the object with a similar feature in the background. Additionally, as the measurement range increases, more templates need to be created, which is more prone to such mismatching problems. Our proposed segmentation method based on an oriented bounding box avoids cluttered background interference. The other problem is the efficiency of the algorithm. The 6 degrees of freedom of an object in 3D space cause a huge number of 2D views that need to compare with the image, and the best matching is always found through exhaustive search. These make recognition and positioning inefficient. A high-resolution camera can ensure high-precision pose measurement, but it is too time-consuming due to the large search space. Our proposed pre-segmentation strategy solves this problem by reducing the image space and range space.

To summarize, the main contributions of this paper propose to improve these problems are as follows.

- We design a multi-sensor visual system combining the advantages of two kinds of cameras. To get both high accuracy and efficiency, the depth camera is utilized for rough positioning and the CCD camera is for precise measurement. Because they are based on multiple sensors, our measurement results are more stable and reliable.
- We propose a strategy of initial segmentation and then measurement, which solves the problem of misidentification of similar structures in background regions and solves the problem of huge search matching space in existing algorithms.
- We propose a 3D region-of-interest segmentation algorithm based on the oriented bounding box, considering the relationship between objects and their fixed plane.

The remainder of this paper is as follows. Related work is discussed in Section 2, and Section 3 describes the hardware composition of the system. The object recognition and pose estimation combining point cloud segmentation and template matching in gray images are illustrated in Section 4. Section 5 shows the experimental results of our approach. Finally, the conclusion is given in Section 6.

2. RELATED WORK

The current approaches for recognizing 3D objects consist mainly of four types. Traditional feature-based approaches use features, such as gray value edges, line segment intersections, or more complex features that consist of single features [7–10]. These methods consider that these features correspond to texture edges, geometric edges, or object corners. The extracted features are matched to the corresponding object features and the pose is obtained from the corresponding characteristics. The search space that results from the establishment of correspondence between the object features and the image features can be extremely large. In addition, the number of extracted features and the search space will increase in a cluttered scene. Using more complex features will reduce the search space, but then the algorithm becomes less robust, especially when parts of the target are occluded or object edges are not clearly visible. These make this kind of approach too slow for most practical applications.

Descriptor-based approaches extract key points from artificial views of the object and then create discriminative descriptors derived from the surroundings of the key points [11–13]. A classifier is trained with these descriptors. In the searching step, the correspondence between the search image and the model is determined by classifying the descriptors derived from the search image. The most benefit of these types of methods is that the run time is independent of the size of the search space. Outstanding performance has been shown in several scenarios, but they have serious problems when searching for textureless objects, such as shiny metal or plastic surfaces without any print on them, such as the metal handrail shown in Fig. 1(b).

Another kind of method is view-based approaches [14]. The search image is compared with prepared 2D views of the object to estimate the object pose. The views are clustered, or aspect graphs [15] are used to deal with the full geometric search space. Because the pose of the object is arbitrary and unknown, the number of views and aspects is generally very high. To decrease the complexity of aspect graphs, some methods decompose them into several primitives. Some methods group similar 2D views of the object into a few aspects to accelerate the search, and the similarity is measured by using a shape similarity metric. Then, every single prototype is utilized to represent each aspect. In the recognition phase, a search image is compared with prototypes instead of all possible 2D views. A hierarchical shape-based approach that combines the idea of scale-space aspect graphs of [16] with the concept of similarity-based aspect graphs of [17] is proposed in [6]. This method increases efficiency to a certain extent, but it will cost much time and have a tendency to increase the error probability in highly cluttered scenes. A modified template-matching method called LINE-MOD was presented by Hinterstoisser and co-workers [18,19] to estimate the location of textureless objects. Both the image gradients cue and the 3D surface normal orientation cue in the depth image are used. However, the sparse distribution of the template limits the accuracy of the initial estimated location. Thus, the method cannot output accurate 6D pose of the object, and similar structures could easily produce false positives, particularly when the templates seen under some viewpoints are not discriminative.

Convolutional neural networks (CNN)-based methods are also used for textureless object detection [20–22]. CNN is used to obtain descriptors of object views that efficiently capture both the object identity and the 3D pose. The trained descriptor is shown to generalize to unknown objects. The experimental results show that CNN outperforms state-of-the-art methods to recognize 3D objects from novel viewpoints. CNN does not specialize in the geometry or appearance of specific objects, and it can be used with objects of vastly different shapes and appearances, and in different backgrounds. However, it is a data-driven method, needs a huge number of images to learn, and does not output an accurate 6D pose of the object.

In this paper, conventional visual feature matching is useless due to the absence of rich texture features. Shape-based matching methods in a gray image can easily be affected by illumination variation, and the template is often matched with false edges in cluttered backgrounds. Our approach uses an *a priori* acquired 3D CAD model as input, and recognizes and precisely determines the 6D pose of the instance of the object using the point cloud and gray image. First, the point cloud obtained by a depth camera is utilized to segment the panel where the object is fixed. Because the ToF measurement precision and the depth camera resolution are both limited, the precise 6D pose cannot be obtained by using only a depth camera. However, the region of the object can be coarsely determined. This greatly reduces the search space in gray images and obviously decreases the error recognition rate in cluttered scenes.

3. HARDWARE ARCHITECTURE

In this section, we present the hardware architecture, including the hardware composition, coordinate system definition, and multiple sensor calibrations. As shown in Fig. 2, the proposed visual perception system, which is designed for a humanoid robot, has a SwissRanger 4000 depth camera and two compact color cameras mounted on the robot head. In nominal operation, the SR4000 depth camera can achieve an absolute accuracy of 15 mm for 11×11 central pixels of the camera at 99% target reflectivity and over a calibrated range from 0.8 to 8.0 m. The pixel array size of the depth camera is $176(h) \times 144(v)$ and the field of view is $43.6°(h) \times 34.6°(v)$. Color cameras (2448×2050 pixels, 8 mm lens) that we used here are Prosilica GC2450C, produced by Allied Vision Technologies. Only one is employed in this paper and the other one is auxiliary. The depth camera serves as a synchronization trigger to ensure that all cameras capture images at the same time. The objects to be recognized are mounted on a panel with screws. The average error is ± 4 mm in depth and ± 2 mm in the other two directions, and the absolute angle error is $\pm 2°$, which are high accuracy requirements. In addition to the accuracy requirement, the robot is 0.8–1.3 m far away from the panel, which is a large distance range, making simple methods based on single camera impossible.

A. Coordinate System Definition

As shown in Fig. 3, the depth camera coordinate system O_d is at the intersection of the optical axis with the front face of the camera, while the color camera coordinate system O_c is located at the center of the left camera lens. Camera coordinate systems used here are "right-handed," with the z coordinate increasing along the optical axis away from the camera, the y coordinate increasing vertically downward, and the x coordinate increasing horizontally to the right, all from the point of view of the cameras (or someone standing behind them). The world coordinate system O_w is set on a small metal plate, which is fixed to the cameras. There are three machining holes on the small board used for placing the reflective ball of the laser tracker to obtain the ground truth pose of the object relative to the world coordinate system.

B. Multiple Sensor Calibrations

To estimate the 6D pose of the object accurately, we need to calibrate the vision system first. As shown in Fig. 3, calibration

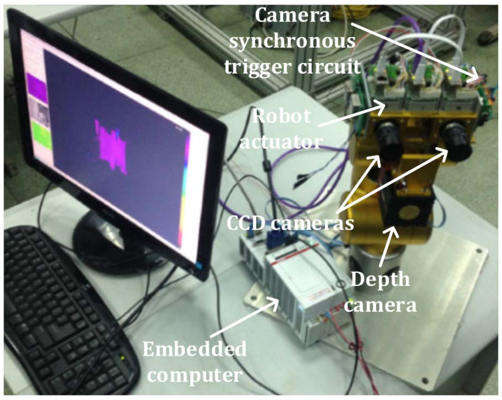

Fig. 2. Hardware presentation of the visual perception system.

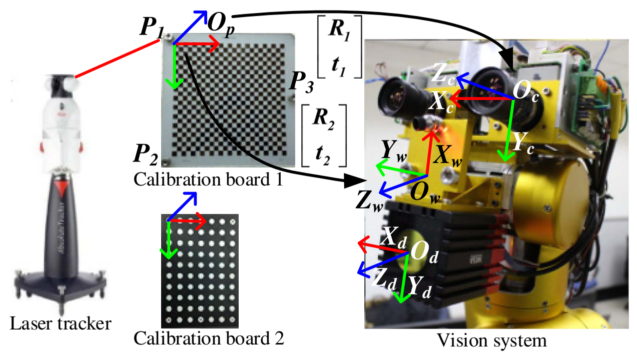

Fig. 3. Coordinate system definition and calibration of the visual system.

board 1 is used for the CCD camera and board 2 is for the depth camera. Zhang's method [23] is utilized here to complete CCD camera calibration of intrinsic parameters. The depth camera had been calibrated by the manufacturer before we bought it, so we just need to calibrate extrinsic parameters of the different camera coordinate systems to the same world coordinate system. The intrinsic parameters of the cameras are given in Table 1. According to the calibrated parameters, the corners on the calibration board are projected to the image space. The pixel coordinates of the projected points are calculated and compared with the corner pixel coordinates extracted directly from the images. The mean reprojection error is 0.1754 pixel for the left camera and 0.1898 pixel for the right camera, which indicates that the calibration error is small.

The laser tracker (Leica AT901) shown in Fig. 3 is utilized to calibrate the extrinsic parameters. The extrinsic parameters represent a rigid transformation from the world coordinate system to the camera's coordinate system. We conduct extrinsic parameter calibration of the CCD camera by using calibration board 1. The laser tracker can precisely locate the 3D position of the reflector. First, we obtained the 3D position of the three reflectors that were placed on the metal plate of the camera set. The world coordinate system O_w was established through these three points, as shown in Fig. 3. We then obtained the 3D position of the three reflectors that were placed on calibration board 1. P_1, P_2, and P_3 are the locations of the reflectors. The coordinates of these three points in the world coordinate system O_w were measured by the laser tracker. Thus, we can compute the transformation between the calibration board 1 coordinate system O_p and O_w. Meanwhile, the transformation between calibration board 1 and the CCD camera system O_C can be computed through correspondence between 2D image points and the 3D points on the calibration board. Therefore, we can obtain the transformation from the world coordinate system to the CCD camera system, that is, the extrinsic parameter of the CCD camera. Similarly, we conduct extrinsic parameter calibration of the depth camera by using calibration board 2. The extrinsic parameters of the three cameras are

$$T_L = \begin{bmatrix} -0.00969 & 0.999935 & -0.00598 & 44.38937 \\ -0.99995 & -0.00971 & -0.00193 & 56.51871 \\ -0.00199 & 0.005963 & 0.99998 & 17.90749 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_R = \begin{bmatrix} -0.00699 & 0.999968 & -0.00383 & -45.2178 \\ -0.99998 & -0.00699 & 0.000936 & 56.70116 \\ 0.000909 & 0.003833 & 0.999992 & 16.61782 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_D = \begin{bmatrix} 0.01009 & 1.011823 & -0.00329 & 0.702914 \\ -1.00921 & 0.008537 & -0.00676 & -47.8274 \\ -0.00659 & 0.003417 & 1.02123 & -31.8345 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We got the conversion matrices of different cameras to the same world coordinate system after calibration. Thus, the point correspondence of two kinds of cameras was obtained as follows:

$$P_c = R_c \cdot P_w + t_c, \quad (1)$$

$$P_d = R_d \cdot P_w + t_d, \quad (2)$$

$$P_c = R_c \cdot (R_d^{-1} \cdot P_d - R_d^{-1} \cdot t_d) + t_c. \quad (3)$$

P_w, P_c, and P_d represent coordinates of the point in different coordinate systems. $[R_c t_c]$ and $[R_d t_d]$ represent the extrinsic parameters of the CCD camera and depth camera, respectively.

Table 1. Intrinsic Parameters of Cameras

Intrinsic Parameters	Left Camera	Right Camera
f_x	2331.10171	2314.63265
f_y	2330.82152	2314.23571
u_0	1203.2753	1195.91035
v_0	1049.47257	991.01816
k_1	−0.0840758	−0.0874617
k_2	0.0807441	0.0719448
p_1	0.0002128	0.0003057
p_2	−0.0000821	0.0003541

4. RECOGNITION AND LOCATION STRATEGY

The object pose is used to determine the coordinate transformation between object coordinate system O_o and world coordinate system O_w. The 6D pose can be expressed as $(x, y, z, \alpha, \beta, \gamma)$, which represent the translation vector (x, y, z) and the rotation angle (α, β, γ) around the three axes, as shown in Fig. 4.

In general, high-precision measurement for attitude and position of textureless objects in only a gray-scale image or a point cloud is inefficient, because an exhaustive search throughout the entire scene has a lot of useless computational costs, particularly in a complex background full of edges. In addition, when edge features similar to the object appear in the

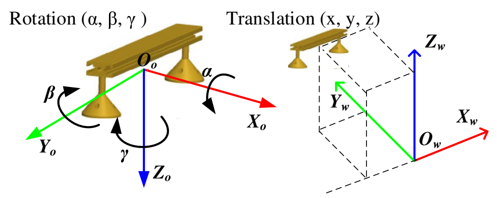

Fig. 4. Definition of the 6D pose.

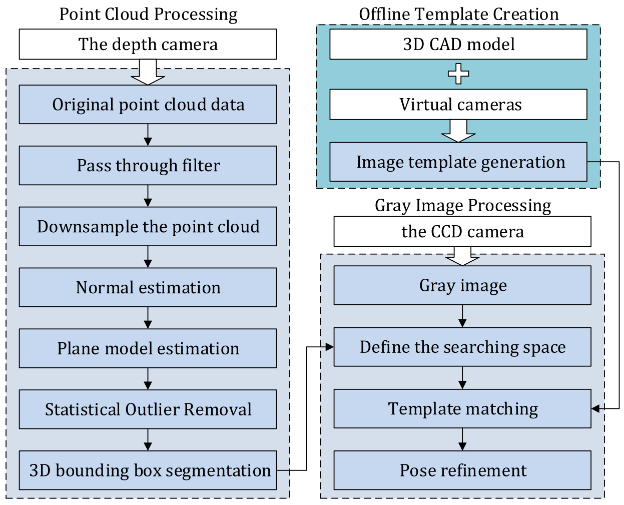

Fig. 5. Flow chart of the proposed method.

background, they would tend to be misidentified as the object, and the matching algorithm becomes less robust. To deal with the problem introduced by the large scene and wide depth range, and meet the precision requirement, our handling strategy is divided into two stages as depicted in Fig. 5:

(1) panel segmentation in the point cloud to remove the background from the large scene and a rough location of the object based on the premise that the object is fixed on a panel and

(2) precise pose measurement in the gray image with the view-based template-matching method and a least-squares adjustment.

The point cloud acquired by the depth camera is processed first. Because objects are fixed on a panel in general, we extract the panel in the point cloud prior to matching the object. Then, an oriented bounding box is created to obtain the possible area of the object. Such a region of interest in the point cloud can be transformed into an appropriate search space in the gray image for template matching. The distance information of the object is coarsely determined. Our application scenario can be described as follows. Given the 3D CAD model of an object, we generate hierarchical models based on the geometry information in the offline stage. In the online phase, our visual system identifies the object and measures its position in cluttered scenes.

A. Object Segmentation in the Point Cloud

We use the depth camera to acquire point cloud data, estimate the panel, and get the approximate region of the object. We then transform the coordinates of these points into the gray image coordinate system to reduce the image search space of the view-based matching. According to the z coordinate values of the points, we get the search range in the depth direction. Thus, cluttered backgrounds are circumvented and the search space for template matching in the gray image is extremely reduced. The segmentation information that an image can provide is only a gray scale or RGB vector, whereas a 3D point cloud can provide more information. Therefore, point clouds are more advantageous than images in segmentation, and segmentation is a key operation in point cloud processing. The infrastructure of point cloud segmentation has been established in the Point Cloud Library [24].

1. Preprocessing of the Point Cloud

Data preprocessing, such as filtering, is required before point cloud segmentation. The data output from the SR4000 camera includes the XYZ Cartesian coordinates, distance data, amplitude data, and confidence map. The XYZ Cartesian coordinates are preprocessed with a 3×3 median filter to remove noise, and then they are used to form an original point cloud, whose size is 176×144 here. The calibrated range captured by the SR4000 is 0.8–8.0 m; however, what we are interested in is the object closed to the robot. Too many data points will increase subsequent computational overhead and make the follow-up procedure inefficient. Thus, we perform a simple pass-through filter along the z axis first to remove points that are too far away from the camera. Here the specified range is 0.3–1.5 m, and the value depends on the length of the robotic arms and the actual applicative requirements. It traverses the point cloud once, serving two purposes. First, spurious invalid points are removed. Second, any points that lie outside the interval are cut off. We then get a smaller-sized point cloud for following operations. Figure 6 shows a sample result of the filtering. The original point cloud has 25,344 data points, and it has 14,637 data points after filtering. About 42% of the points are removed. Of course, this is related to the scene captured by the camera.

2. Plane Estimation

After getting the filtered points, we want to get a plane model from the data. A plane can be defined using a parametric model with four components (A, B, C, D), using the equation $Ax + By + Cz + D = 0$. The objective of this step is to find the parameters that best fit into the point cloud to extract the dominant plane of the scene. Since various points are not part of the panel, classical fitting techniques such as least squares will not perform well. To fit parametric models to data with numerous outliers, the random sample consensus (RANSAC) family of algorithms has proven to be very successful [25–27].

The RANSAC algorithm is an iterative method that can robustly estimate the parameters of a mathematical model from a

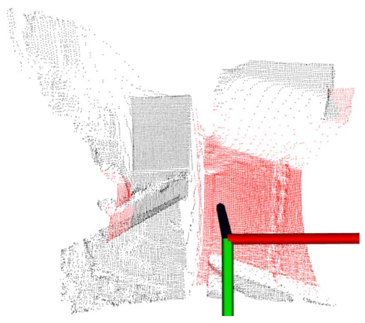

Fig. 6. Sample result of pass-through filtering: original point cloud (black) and the filtered point cloud (red). The red, green, and blue lines represent the x, y, and z axes of the depth camera coordinate system, respectively.

set of observational data that contain outliers. The advantage of RANSAC is its ability to do a robust estimation of the model parameters, i.e., it can estimate the parameters with a high degree of accuracy even when a significant number of outliers exists in the dataset. Here, we utilize the RANSAC to estimate the plane.

However, there are two common problems with the RANSAC method. One is that it is too time-consuming and has no upper bound on the time computing these parameters. When the number of iterations computed is limited, the solution obtained may not be optimal, and it may not even be one that fits the data in a good way. Therefore, a trade-off is made: by computing a greater number of iterations, the probability of a reasonable model being produced is increased. The other problem is that RANSAC requires the setting of problem-specific thresholds, i.e., it fits points to the plane according to the distance error. Noting that three points are sufficient to estimate a plane equation, the idea of RANSAC is to iteratively draw three random points and check how many of the remaining points lie on the estimated plane (inliers), using some distance threshold. The plane that got the highest number of inliers is finally kept, and its equation is refined using least-squares fitting with the inliers. If there is a dominant plane, it will be found in a few iterations, since there is a high probability that three random points belong to it.

Point cloud downsampling. Because this iterative process is too time-consuming, before that we use a voxel grid method to further simplify the point cloud while maintaining the shape characteristics and details. All the points that lie within each voxel are represented by their centroid. The choice of grid size has a significant influence on the RANSAC algorithm's efficiency. As shown in Fig. 7, the number of points and the time cost of RANSAC are negatively correlated with the size of the voxel grid. The larger the grid, the fewer points are sampled and the less time the RANSAC algorithm takes. Nonetheless, if the size is too large, too few points will lead to inaccurate or incorrect plane estimation. Without the downsampling process, there are more than 14,000 points and it takes more than 4 s to estimate a plane model. This downsampling operation will remove many redundant data points and return an

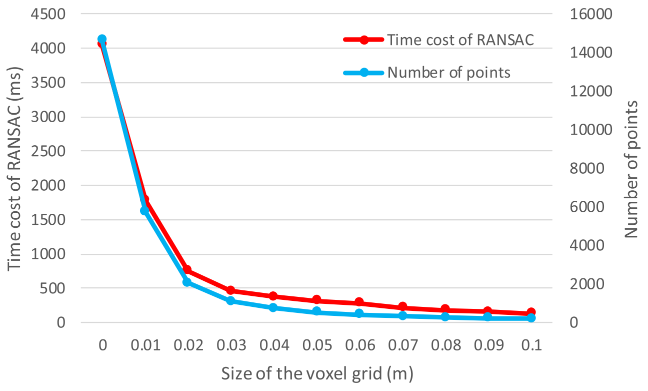

Fig. 7. Influence of grid size on the number of sampling points and efficiency of RANSAC.

equivalent point cloud with fewer points, which is of great importance for speeding up the RANSAC algorithm. The size of the grid is set to 9 cm here. The RANSAC method takes about 153 ms to estimate the plane model in 223 data points. As can be seen, downsampling saves 96% of the time consumption of the RANSAC method.

The distance-based fittings may be not particularly accurate. Therefore, we use a more robust plane fitting model that adds an additional constraint: the surface normal of the inliers has to be parallel to the surface normal of the output plane, within a maximum specified angular deviation. First, we compute the surface normal at each point in the cloud. In the next step, we use a RANSAC algorithm to locate the dominant plane and compute the plane parameters. When selecting the optimal plane, the constraint should be taken into account.

Surface normal estimation. Surface normals are important features of geometric surfaces. A normal to a surface is a vector perpendicular to a surface at a given point. The problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane tangent to the surface, which in turn becomes a least-square plane fitting estimation problem. The solution for estimating the surface normal is therefore reduced to an analysis of the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbors of the query point. More specifically, for each point p_i, we assemble the covariance matrix C as follows:

$$\boldsymbol{C} = \frac{1}{\boldsymbol{k}}\sum_{i=1}^{\boldsymbol{k}} (\boldsymbol{p_i} - \bar{\boldsymbol{p}}) \cdot (\boldsymbol{p_i} - \bar{\boldsymbol{p}})^T, \tag{4}$$

$$\boldsymbol{C} \cdot \vec{\boldsymbol{v}}_j = \lambda_j \cdot \vec{\boldsymbol{v}}_j, \qquad j \in \{0, 1, 2\}, \tag{5}$$

where k is the number of point neighbors considered in the neighborhood of p_i, $\bar{\boldsymbol{p}}$ represents the 3D centroid of the nearest neighbors, $\boldsymbol{\lambda_j}$ is the jth eigenvalue of the covariance matrix, and $\vec{\boldsymbol{v}}_j$ the jth eigenvector. In general, because there is no mathematical way to solve for the sign of the normal, its orientation computed via principal component analysis (PCA) is ambiguous and not consistently oriented over an entire point cloud dataset. The solution to this problem is trivial if the viewpoint $\boldsymbol{v_p}$ is in fact known. To orient all normals $\vec{\boldsymbol{n}}_i$ consistently toward the viewpoint, they need to satisfy the equation

$$\vec{\boldsymbol{n}}_i \cdot (\boldsymbol{v_p} - \boldsymbol{p_i}) > 0. \tag{6}$$

Surface normals are estimated at a point from the surrounding point neighborhood. There are two options, k (neighborhood) or r (search radius), to determine the set of nearest neighbors of a point. Here we use a k neighborhood search of 50. We calculate surface normals before applying the RANSAC to improve the robustness of plane estimation and increase the accuracy. The result of surface normal estimation is shown in Fig. 8. The normal vector at each point in the downsampled point cloud is computed.

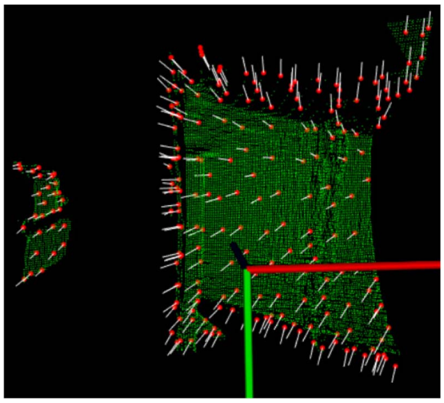

Fig. 8. Result of surface normal estimation. The white lines represent the surface normal vectors, the red points are the points after voxel grid downsampling, which are magnified into view for visibility reasons, and the green points show the point cloud after the pass-through filtering. The red, green, and blue lines represent the x, y, and z axes of the depth camera coordinate system, respectively.

Algorithm 1: Random Sample Consensus (RANSAC)

```
Input: data – observed data points.
model – a model that can be fitted to the data points.
i – maximum number of iterations allowed in the algorithm.
n – minimum number of data required to fit the model.
d – threshold value to determine when a data point fits a model.
m – number of close data points required to assert that a model fits
well to data.
Output: best_fit – model parameters which fit the data best (or null if
no good model is found).
While iterations < i
    possible_inliers = n randomly selected items from data
    possible_model = model parameters fitted to possible_inliers
    also_inliers = empty set
    For every point in data not in possible_inliers
        If point fits possible_model with an error < d then
        add point to also_inliers
        End if
    End for
    If the number of points in also_inliers is > m then
    (This implies that a good model is found; now test how
good it is.)
    better_model = model parameters fitted to all points in
possible_inliers and also_inliers
    this_error = a measure of how well model fits these points
        If this_error < best_error then
        (The found plane is better than the previously found plane.)
        best_fit = better_model
        best_error = this_error
        End if
    End if
    increment iterations
End while
Return best_fit
```

Plane segmentation using RANSAC and surface normal. The iterative procedure of the RANSAC algorithm is presented in Algorithm 1. Iterate through the points and calculate the distances from them to the plane. When calculating the distance from the points to the plane, the estimated surface normals are taken into account. Suppose the coordinate of a point P is (x, y, z); the Euclidean distance to the plane is computed by

$$D_{euclid} = |Ax + By + Cz + D|/|\vec{N}|, \tag{7}$$

where the plane equation is defined as $Ax + By + Cz + D = 0$, and the normal vector $\vec{N} = (A, B, C)$. Then, the angular distance between the point normal and the plane normal is calculated by

$$D_{normal} = \arccos\left(\frac{\vec{n}_i \cdot \vec{N}}{|\vec{n}_i||\vec{N}|}\right). \tag{8}$$

The error, which is a measure of how well the points fit the model, is computed by

$$error = w \cdot D_{normal} + (1 - w) \cdot D_{euclid}, \tag{9}$$

where w represents the weight coefficient of the surface normal, and the value range is 0 to 1. We use an error threshold of 6 cm for any point to fit the plane model, which means that when the error is not larger than 6 cm, the point is considered to be an inlier. Note that this threshold is closely related to the surface reflectivity of the plane. Because the range measurement of the depth camera is based on the ToF, the noise of distance measurements depends mainly on signal amplitude and background illumination. Signal amplitude, in turn, depends on object distance and object reflectivity. The reflectivity of the measured object has a large influence on the repeatability (noise) of the measurement. In our experiments, a black surface plane absorbs a large proportion of the incoming light. As a result, the noise of the range measurement is relatively large, and a high threshold value is used in our application.

The output for our scene is displayed in Fig. 9. The input data contain inliers and outliers. The inliers are the points that fit the given model, while the outliers cannot fit the model. The red points in the figure represent the inliers. This method can estimate the parameters of the plane robustly even if a number of outliers exist. By using this detector, we can get the equation of the dominant plane in the point cloud, along with the list of point indices belonging to it. The blue arrow shows the normal

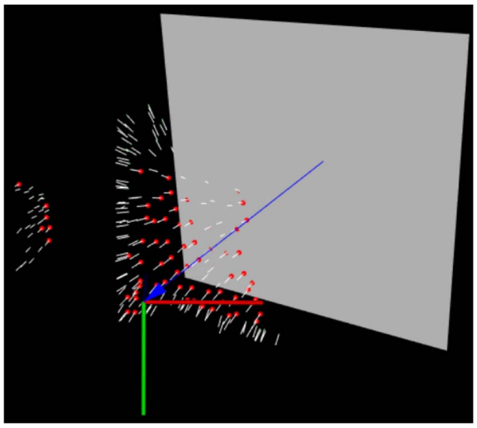

Fig. 9. Plane estimation result includes the plane equation and the inliers. The red, green, and blue lines represent the x, y, and z axes of the depth camera coordinate system, respectively.

vector of the plane. Note that the plane actually has no boundaries.

3. Panel Region Segmentation

After the previous step, we obtain a plane equation and all the inlier points. Nevertheless, some inlier points belonging to the estimated plane are outside the panel, and some are noisy points. These points are not the authentic panel points that we want to find, and they will influence the determination of the target area. Thus, to further obtain the actual panel region, the statistical outlier removal technique is employed here to eliminate these unwanted points based on point neighborhood statistics.

The statistical outlier removal filter removes points that do not have enough neighbors around them. It iterates through the plane inlier point cloud twice. First, for each point, the filter will compute the mean distance from it to its nearest k neighbors. Here, the value of k is set according to the point resolution computed previously. By means of assuming that the result is a Gaussian distribution with a mean and a standard deviation, during the next iteration all points whose mean distances are outside of a threshold defined by the global distances' mean and standard deviation will be regarded as noise and outliers. Here, we set the number of neighbors to be analyzed for each point to 50 and set the standard deviation multiple to 1, which means that if the distance of a point is more than 1 standard deviation from the average distance, that point is marked as an outlier, and will be removed.

4. Object Extraction

After obtaining the panel inliers, we want to retain only points fixed on the panel to identify the location of the object. We need to limit the searching area to the actual boundaries of the panel. These can be efficiently approximated by projecting the inliers onto the plane using the estimated equation and computing its minimum enclosing rectangle. This rectangle delimits the area where the object can be fixed. To filter out points that do not lie over this area, the next step is to create a 3D bounding box over the rectangle and eliminate points that are not inside the box using a geometric check. To discard the points outside the chosen 3D box, a crop box filter is implemented, which is a pass-through filter on the three spatial dimensions. The crop box filter is to obtain all the data inside of a given box. It is defined by two points, $p_{\min} = (x_{\min}, y_{\min}, z_{\min})$ and $p_{\max} = (x_{\max}, y_{\max}, z_{\max})$, which are the two diagonal points of the cube. The points in the box satisfy the following equations:

$$x_{\min} \le x \le x_{\max}, \tag{10}$$

$$y_{\min} \le y \le y_{\max}, \tag{11}$$

$$z_{\min} \le z \le z_{\max}. \tag{12}$$

The box filter here is used to determine the region of the panel. Once the minimum enclosing rectangle has been determined, the centroid (x_c, y_c, z_c) is computed first. Then, we transform the point cloud by the translation vector $(-x_c, -y_c, -z_c)$. According to the previously obtained plane normal vector (A, B, C) and vectors (0, 0, 1), the rotation matrix can be calculated by Rodrigues' rotation formula

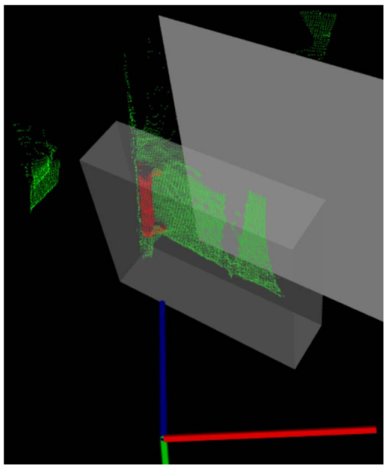

Fig. 10. Result of the crop box filter in 3D space. The red points are the cropped result. The red, green, and blue lines represent the x, y, and z axes of the depth camera coordinate system, respectively.

[28]. We rotate the point cloud by the rotation matrix. The $x_{\min}$, $x_{\max}$ and $y_{\min}$, $y_{\max}$ can be set by the length and width of the minimum enclosing rectangle, and the $z_{\min}$ and $z_{\max}$ are determined by the distance of the object in front of the panel. For example, here we set $z_{\min} = -0.2$ m and $z_{\max} = -0.02$ m, which means that the object must lie between 2 and 20 cm in front of the panel. After the filtering, the point cloud is transformed by the inverse transformation matrix. Finally, only the points lying between 2 and 20 cm in front of the panel are segmented from the scene through this cropping phase.

Figure 10 illustrates the effect of the crop box filter. To avoid the noise points in front of the panel, we translate the plane 0.02 m in the normal vector direction. It is worth noting that the cropped point cloud in this step is the pass-through filtered cloud, not the downsampled one.

Obviously, here we define a cuboid in 3D space to obtain the possible area of objects. Only the points lying inside it are segmented from the scene through this cropping phase. Through these techniques, we can acquire the pure point cloud of the target object. We treat these points as possible object candidates and transform the coordinates of the points into gray image coordinate space through the extrinsic parameters of the cameras. In the gray image, the u, v coordinates can be used to restrict the search space, so the search time is drastically reduced. Furthermore, the z coordinate can be utilized as another range restriction for the following precise template matching in gray image.

B. Pose Measurement in the Gray Image

The corresponding region of the gray image is obtained by homogeneous coordinate transformation, because the CCD camera and the depth camera have been calibrated in the same world coordinate system. By having such a region of interest, the image search space is extremely reduced for view-based template matching. The CAD view-based method previously presented by Ulrich *et al.* [6] is used here to measure the pose of the object accurately. Often the CAD models of rigid objects are accessible because they are really helpful in the industrial field. They are utilized here as input data to create shape templates. Compared with machine learning approaches, the CAD-View methods need not collect a large number of images of objects for training. The method includes offline stage and online phase. First, the CAD model is used to generate a hierarchical shape model offline, and then the hierarchical shape model is used to find the objects online from an input image.

1. Generate the Hierarchical Model Offline

The explanation of generating the hierarchical model offline is as follows. Suppose there is a virtual ball in the 3D space and the CAD model is put in the center of the ball, making the origin of the object coordinate system coincide with the center of the ball, then a virtual camera is placed on the surface of the ball to capture a great number of images from different places. The radius of the ball is artificially set according to the possible distance between camera and object. Different views are generated by predefining viewpoints in a spherical coordinate system whose origin is the same as the origin of the object coordinate system. The sampling points of the virtual camera form a sphere. The virtual camera position range is specified by $[\phi_{\min}, \phi_{\max}]$, $[\theta_{\min}, \theta_{\max}]$, and $[r_{\min}, r_{\max}]$.ϕ, θ, and r represent longitude, latitude, and spherical radius, respectively, as shown in Fig. 11.

The 2D projection similarity between two adjacent views determines the sample density. Aspect graphs are generated by clustering the views according to the similarity measurement computed by

$$c = \left| \frac{1}{n} \sum_{i=1}^{n} \frac{m_i \cdot s_i}{\|m_i\| \cdot \|s_i\|} \right|. \tag{13}$$

The gradients $m_i (i = 1, \ldots, n)$ of the 2D model points are compared with the gradients s_i by using the dot product. The views whose similarities are higher than a defined value T_{merge} are combined to form an aspect graph. Thus, the aspect graph can be a set of views or only one view. The similarities of adjacent aspect graphs are below T_{merge}. After completing the merge, we sample the aspect graph on the next higher image pyramid level and repeat the merging process. Eventually, hierarchical view templates that span different image levels are generated.

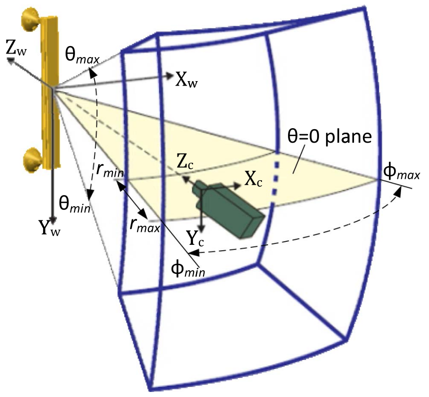

Fig. 11. Spherical coordinate system for model generation offline.

2. Searching Objects Online

After the establishment of the view templates, finding the most similar template is the purpose of the search phase. We detect object edge points in the template image by applying the threshold T_{amp} to the edge amplitude. The gradients of n edge points are calculated. Each 2D template image consists of n resulting edge points and corresponding gradients. The gradients are calculated for all pixels in the search image, and no thresholding is utilized. The gradients m_i in the view template are compared with the underlying gradients s_i in the search image by using the dot product [Eq. (10)]. The similarity measurement is applied within a template-matching framework to find a transformed (translated, rotated, and scaled) template image in the search image.

It can be seen that the view templates form a large searching space, and it is almost impossible to finish the work by brute force searching. The online search object in the input image is divided into two steps. First, find the most similar template in the search image using the hierarchical view model. A top-down efficient searching algorithm [29] is utilized, which corresponds to the hierarchical templates. Second, the pose is refined after a successful search. The precision of the obtained pose is limited to the sample density of the views and the sampling of the 2D poses during the 2D matching. The pose refinement is necessary because it is not enough for practical applications. The pose refinement starts with the previously obtained pose. Then the pose is used to project 3D CAD model edges into the image dynamically. Finally, the refined 6D pose is obtained through nonlinear optimization using the Levenberg–Marquardt algorithm [30]. We directly optimize the six pose parameters to minimize the squared distances between the projected CAD model edges and the image edge points. Because the optimization algorithm is a loop iterative process, it is another time-consuming process.

5. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results are presented and analyzed in this section. All tests were performed on a laptop with Intel Core i7-7700HQ CPU 2.80 GHz and 8.00 GB memory. As shown in Fig. 12, three CAD models are used as the input objects to generate shape templates in the offline phase, including the space handle, drawer handle, and watering can. The experimental environment and the objects fixed on a panel are shown in

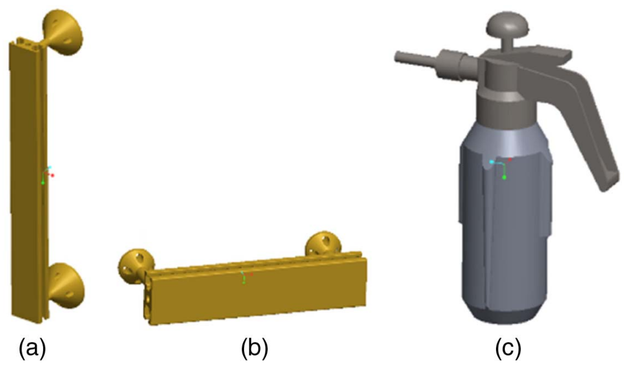

Fig. 12. CAD models of (a) space handle, (b) drawer handle, and (c) watering can.

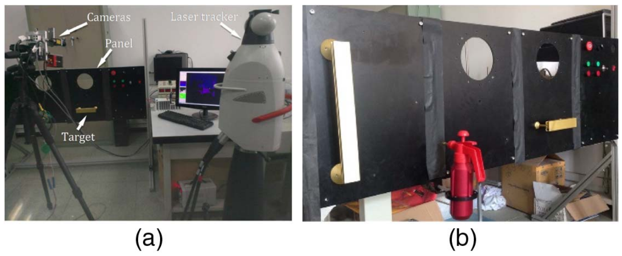

Fig. 13. (a) Experimental environment and (b) targets and the fixed panel.

Fig. 13. We took images and the corresponding point clouds at 1200 different camera positions for each object. The view-based matching method easily finds wrong results in large cluttered backgrounds, and it takes too much time to search in a large pose range. We segment the target area in the point cloud in advance to skip the template matching in cluttered backgrounds. Thus, the matching is more robust. Meanwhile, the image searching space and the depth range are both reduced due to the point cloud segmentation. The final pose measurement results are shown in Fig. 14. The green contour is the measured object projected to the image at the estimated pose, and the magenta edges are the result of edge detection by a Sobel operator.

A. Recognition Rate

According to the requirements of the project, we considered the detected 6D pose as correct if the errors of the result were within 2 mm along the x–y axes, 4 mm along the z axis, and 2° around the x–y–z axes. Our approach is compared with the view-based method proposed in Ref. [6]. For each object, the hierarchical model within the pose range $\phi = [-45, +45]^\circ$, $\theta = [-45, +45]^\circ$, and $r = [0.8, 1.3]$ m is generated, as shown in Fig. 11. Table 2 shows the final results that we obtained. When there is no similar edge or noise in the scenes, we got comparable robustness for both approaches. However, when looking from some special viewpoints and the densely clustered scene appears, it makes sense to segment in advance, and the robustness of the matching is greatly improved. The ratio of right detected images to all captured images increases by more than 10%. Particularly for the watering can, the correct recognition rate increases by 18%. This is because it is always wrongly detected on the edges of the white sponge. Our method solves this problem by identifying the area of interest, eliminating cluttered backgrounds, and reduces the error rate obviously. Experimental results also show that the shape feature is not unique and distinctive enough.

B. Accuracy

To evaluate the accuracy, we selected nine different positions, and at each position we acquired 10 images. The average of the pose detected in 10 images was taken as the measured value, and the pose measured by the Leica Absolute Tracker AT901 was considered the ground truth. The result is shown in Table 3. The accuracy of the watering can is obviously worse than that of the other two objects. This is because the can has no sharp object edges compared to the handles. Round object edges in general do not show a corresponding edge in the image. Often one round object edge is imaged into one or two image edges that are offset from the true position. This offset depends on the relative position of the light source with respect to the camera. This is a very major source of error for all three objects. In addition, the upper and lower parts of the watering can are in a threaded connection, which might make the real object a bit different from the CAD model. This also influences the measurement accuracy. The error of the rotation angle is related with the shape of the object. If the object rotates around an axis, making a significant change in imaging, the error of the rotation angle around this axis is relatively small. On the other hand, If the object rotates a certain angle around an axis, making no obvious change in imaging, which means that it is almost impossible to distinguish the difference of these poses from the image, the angle error around this axis may be relatively large. Furthermore, the translation error also fits the same inference. For example, according to the experimental result, we can see that the error in the z direction is larger than the other two directions in general. The cause of the large error in the depth direction is analyzed as follows. The translation of an object in the depth direction does not cause significant scale changes in image space, so the distance of the object cannot be evaluated more accurately from the image.

C. Time Efficiency

For the view-based approach, the runtime primarily depends on the size of the pose range and the size of the searching space in the image. To prove the computational benefits obtained by incorporating the result of point cloud segmentation into

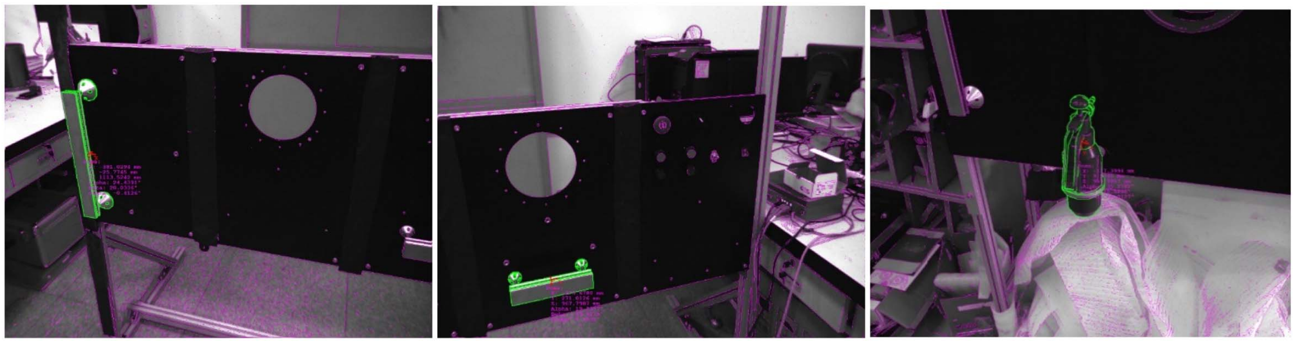

Fig. 14. Pose measurement results.

Table 2. Comparison of the Recognition Rate

Object	View-Based Approach	Our Approach
Space handle	85.47%	100%
Drawer handle	89.60%	100%
Watering can	79.28%	97.4%

Table 3. Results of the Accuracy Evaluation

Object		E_x (mm)	E_y (mm)	E_z (mm)	E_α (°)	E_β (°)	E_γ (°)
Space handle	Mean error	0.64	0.62	1.53	0.51	0.74	0.09
	Max error	1.67	1.88	3.23	1.26	1.42	0.24
Drawer handle	Mean error	0.68	0.59	1.77	0.34	0.08	0.07
	Max error	1.58	1.10	3.97	0.73	0.24	0.18
Watering can	Mean error	1.91	1.05	3.78	0.81	0.91	0.29
	Max error	3.79	1.96	4.94	1.70	2.40	0.59

the image search space, we performed a runtime test by processing all of the captured data. We kept the parameters of the view-based matching approach unchanged and recorded the runtime necessary for the two approaches. Since the robot head moved arbitrarily, different levels of clutter background were captured in the images. The more edges in the images, the more time would be spent to search the object. Figure 15 shows the comparison of calculation time between the view-based method and our approach. The average processing time of the view-based approach is 2120.43 ms and that of our approach is 942.80 ms. The time efficiency is improved by 55.54%. This indicates that the reduction of image searching space improves the search efficiency significantly. Additionally, the image resolution used in this paper is 2448 × 2050, which is quite large for applications of machine vision, and the accuracy parameter used in the view-based approach is set to the highest one. Finally, the average time spent on each step of our method is shown in Fig. 16. Precisely measuring the pose of objects in a gray image is the most time-consuming procedure due to the large size of the pose range. The 6 degrees of freedom of an

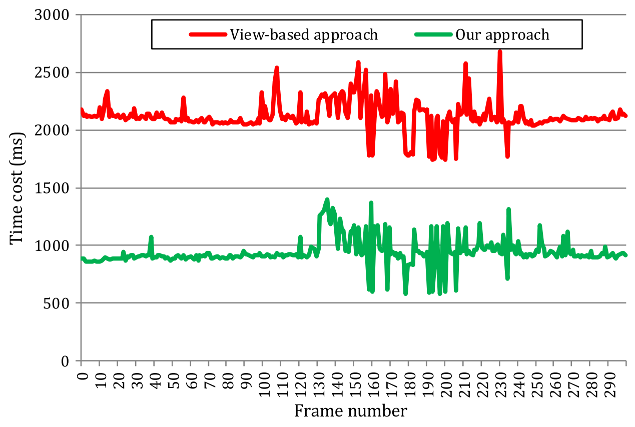

Fig. 15. Comparison of time consumption.

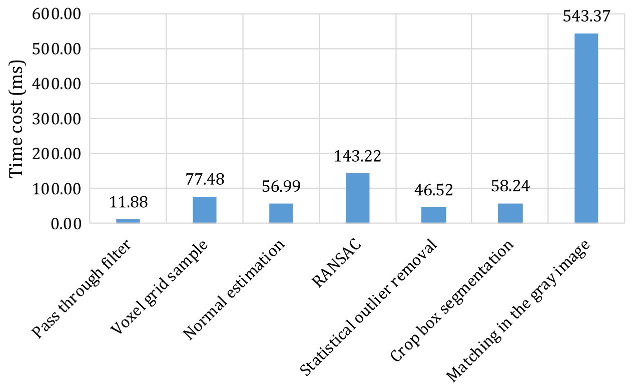

Fig. 16. Time cost analysis of each procedure.

object in 3D space lead to a huge number of 2D views that must be compared to the image. In the point cloud processing, RANSAC takes the most time because it is an iterative method. Thus, it can be seen that our method could greatly reduce computational time and enhance the recognition rate, while maintaining pose measurement precision.

6. CONCLUSION

This paper presents a comprehensive visual system that integrates a ToF depth camera with an industrial CCD camera to achieve highly precise measurements for position and orientation of textureless objects. First, the cluttered background is removed by point cloud segmentation, and the object is coarsely positioned. Then, the fast shape-based matching is performed to precisely estimate the pose of the object. The searching space and calculation amount are reduced significantly due to downsampling techniques and a coarse-to-fine strategy. The precision of pose measurement is also guaranteed. The results show that our method is effective, and the correct rate and running efficiency are improved dramatically. Meanwhile, the robustness of the visual system is enhanced, compared to the single-camera-based approach. We test the system in our task, and the mean absolute error is less than 2 mm for x, y, and less than 4 mm for z, and the angle error is less than 1°. Our future effort will be devoted to utilizing deep learning methods for robotic applications.

Funding. National Natural Science Foundation of China (NSFC) (U1713216).

REFERENCES

1. I. Lysenkov, V. Eruhimov, and G. Bradski, "Recognition and pose estimation of rigid transparent objects with a kinect sensor," in *IEEE International Conference on Robotics and Automation* (2013), pp. 273–280.
2. C. S. Chen, P. C. Chen, and C. M. Hsu, "Three-dimensional object recognition and registration for robotic grasping systems using a modified viewpoint feature histogram," Sensors **16**, 1969 (2016).
3. M. Zhu, K. G. Derpanis, Y. F. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3D object detection and pose estimation for grasping," in *IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2014), pp. 3936–3943.

4. Y. Konishi, Y. Kotake, Y. Ijiri, and M. Kawade, “Fast and precise template matching based on oriented gradients,” in *European Conference on Computer Vision* (Springer, 2012), pp. 607–610.
5. Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto, “Fast 6D pose estimation from a monocular image using hierarchical pose trees,” in *European Conference on Computer Vision* (Springer, 2016), pp. 398–413.
6. M. Ulrich, C. Wiedemann, and C. Steger, “Combining scale-space and similarity-based aspect graphs for fast 3D object recognition,” IEEE Trans. Pattern Anal. Mach. Intell. **34**, 1902–1914 (2012).
7. Z. Luo, K. Zhang, Z. Wang, J. Zheng, and Y. Chen, “3D pose estimation of large and complicated workpieces based on binocular stereo vision,” Appl. Opt. **56**, 6822–6836 (2017).
8. L. Zhang, F. Zhu, Y. Hao, and W. Pan, “Optimization-based noncooperative spacecraft pose estimation using stereo cameras during proximity operations,” Appl. Opt. **56**, 4522–4531 (2017).
9. M. S. Costa and L. G. Shapiro, “3D object recognition and pose with relational indexing,” Comput. Vis. Image Underst. **79**, 364–407 (2000).
10. P. David and D. DeMenthon, “Object recognition in high clutter images using line features,” in *10th International Conference on Computer Vision* (IEEE, 2005), pp. 1581–1588.
11. H. Bay, T. Tuytelaars, and L. V. Gool, “SURF: speeded up robust features,” in *European Conference on Computer Vision* (Springer, 2006), pp. 404–417.
12. S. Hinterstoisser, S. Benhimane, and N. Navab, “N3M: natural 3D markers for real-time object detection and pose estimation,” in *IEEE 11th International Conference on Computer Vision (ICCV)* (2007).
13. V. Lepetit and P. Fua, “Keypoint recognition using randomized trees,” IEEE Trans. Pattern Anal. Mach. Intell. **28**, 1465–1479 (2006).
14. H. Borotschnig, L. Paletta, M. Prantl, and A. Prinz, “Appearance based active object recognition,” Image Vision Comput. **18**, 715–727 (2000).
15. R. D. Schiffenbauer, “A survey of aspect graphs,” Tech. Rep. TR-CIS-2001-01 (Polytechnic University, 2001).
16. D. W. Eggert, K. W. Bowyer, C. R. Dyer, H. I. Christensen, and D. B. Goldgof, “The scale space aspect graph,” IEEE Trans. Pattern Anal. Mach. Intell. **15**, 1114–1130 (1993).
17. C. M. Cyr and B. B. Kimia, “A similarity-based aspect-graph approach to 3D object recognition,” Int. J. Comput. Vis. **57**, 5–22 (2004).
18. 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 *IEEE International Conference on Computer Vision (ICCV)* (IEEE, 2011), pp. 858–865.
19. S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, “Gradient response maps for real-time detection of texture-less objects,” IEEE Trans. Pattern Anal. Mach. Intell. **34**, 876–888 (2012).
20. P. Wohlhart and V. Lepetit, “Learning descriptors for object recognition and 3D pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2015), pp. 3109–3118.
21. A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, “Learning analysis-by-synthesis for 6D pose estimation in RGB-D images,” in *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, 2015), pp. 954–962.
22. S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, “On pre-trained image features and synthetic images for deep learning,” arXiv:1710.10710 (2017).
23. Z. Zhang, “A flexible new technique for camera calibration,” IEEE Trans. Pattern Anal. Mach. Intell. **22**, 1330–1334 (2000).
24. R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and automation (ICRA)* (IEEE, 2011).
25. S. Choi, T. Kim, and W. Yu, “Performance evaluation of RANSAC family,” in *Proceedings of the British Machine Vision Conference (BMVC)* (2009).
26. R. A. Zeineldin and N. A. El-Fishawy, “Fast and accurate ground plane detection for the visually impaired from 3D organized point clouds,” in *Proceedings of the IEEE SAI Computing Conference* (IEEE, 2016), pp. 373–379.
27. D. Lv, J.-F. Sun, Q. Li, and Q. Wang, “3D pose estimation of ground rigid target based on ladar range image,” Appl. Opt. **52**, 8073–8081 (2013).
28. R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation* (CRC Press, 1994).
29. M. Ulrich, C. Wiedemann, and C. Steger, “CAD-based recognition of 3D objects in monocular images,” in *IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2009), pp. 1191–1198.
30. M. Kaj, B. Hans, and T. Ole, *Methods for Non-Linear Least Squares Problems* (Technical University of Denmark, 1999).