Robot-Based Auto-labeling System for 6D Pose Estimation

Hsien-I. Lin and Jun-Shiang Chang

Abstract 6D object pose estimation is an ongoing research area in the field of computer vision. Many existing methods rely on supervised deep learning models which require multiple accurate 6D pose annotations to predict object poses. However, labeling the 6D pose is complex and time-consuming in traditional methods. In this study, we propose a robotic-arm-based 6D object pose auto-labeling approach which has limited human interaction involved. Translations and rotations of the object in the camera coordinate system can be calculated using a sequence of known robot poses and the transformation between the camera and the robot. We also implemented our custom dataset generated by the auto-labeling system in the existing 6D object pose estimation approach. Evaluation results show that the model can recognize our own test dataset and attempted 90% accuracy using ADD metric with 0.05 threshold.

Keywords 6D object pose estimation · Pose annotation · Robotic-arm-based · Annotations · Auto-labeling

1 Introduction

Estimating object pose is an important technique that provides 3D information for the system to make further decisions. It can be implemented in various applications in industries such as robot manipulation tasks, autonomous driving, and augmented reality. In recent years, many deep learning approaches have provided end-to-end object pose estimation solutions that can handle objects with less texture, symmetric,

H.-I. Lin (✉)
Institute of Electrical and Control Engineering, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
e-mail: sofin@nycu.edu.tw

J.-S. Chang
Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei, Taiwan

X.-S. Yang et al. (eds.), *Proceedings of Eighth International Congress on Information and Communication Technology*, Lecture Notes in Networks and Systems 695,
https://doi.org/10.1007/978-981-99-3043-2_10

and irregular shapes. Many researchers established various 6D object pose training datasets such as Linemod, T-less, and YCB-V [1], which were all collected in the Bop challenge benchmark [2]. These datasets allow users to evaluate their pose estimation model and compare it with others methods with the same standard.

Despite having those open source datasets, users in industries need to create their custom datasets to implement object pose estimation models on the new object. However, in traditional methods [3, 4], complex mechanism designs and additional markers are required to generate different camera viewpoints and mark the object's pose in camera frames. To address these problems, [5, 6] proposed simulation methods to create a synthetic dataset for training. In spite of the help of 3D simulators, high-quality mesh models are demanded to create realistic simulation photos. Moreover, the colors of the object and shadows would differ from the actual scenes captured using the camera due to the different light positions.

To generate object pose annotations more conveniently, in a relevant study that a depth camera was mounted on an industrial robot end-effector to autonomously capture RGB-D images from numerous perspectives. The robot was operated to push objects in the real world to change the object's orientation and collect training data continuously. While in this method, the pre-trained model was demanded to detect the object in the beginning. In recent, self-supervised 6D pose estimation methods [7, 8] were proposed to overcome the difficulty of acquiring real pose annotations. With the collections of object 6D pose datasets, several object pose estimations can be implemented. In [9, 10], the estimation is done only from a single RGB image. 6D pose problems can be addressed by adding depth information in the source, as presented in [11], which may increase the computation cost and equipment level.

The objective of this research is to develop a novel robot-arm-based 6D object pose auto-labeling approach. The system has limited human interaction involved and can be easily operated by users without prior knowledge. Object poses in the camera coordinate system are automatically calculated according to a sequence of known robot poses and the predefined transformation from the camera to the robot. In addition, the system imports object CAD models to create segmentation masks simultaneously. We also implemented the dataset generated by the auto-labeling system in the existing 6D object pose estimation deep learning model. The proposed system can be applied in factories equipped with robotic arms. Automatically collecting training data for AI models drastically reduces the data labeling time which was conducted by human in tradition.

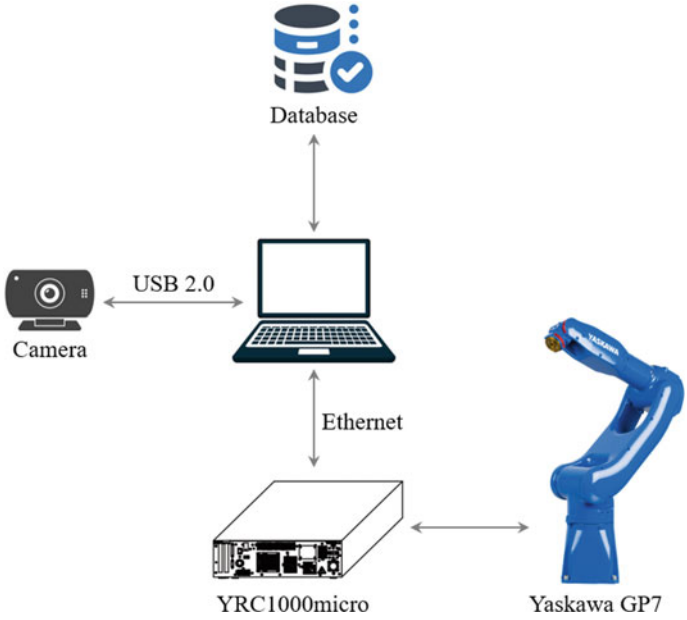


Fig. 1 System architecture

2 Methodology

2.1 System Overview

The proposed auto-labeling system employs a 6-DoF industrial robotic arm and a camera in this study. The robotic arm was required to move the object mounting on the end-effector, while the camera helped to collect the image of the object. Figure 1 shows the architecture of the auto-labeling system. The robot of the auto-labeling system is assisted by the Yaskawa GP7, which is used to rotate the object and provide known robot poses for inferring ground truth object poses. We used a laptop to communicate with the camera and control the robot through Ethernet. The database saved all the training images and corresponding annotations.

2.2 Hand-Eye Calibration

We conducted the eye-to-hand calibration to know the transformation from the robot to the camera. We prepared the checkerboard and attached it to the robot end-effector. We collected calibration data by moving the robot in several different poses and

recorded images with corresponding robot coordinates at the same time. The calibration process comprises the computation of extrinsic and intrinsic parameters. The extrinsic parameters represent a transformation from the checkerboard to the camera. The intrinsic parameters represent a projective transformation from the 3D camera coordinate into the 2D image coordinates.

It is explained that the transformation matrix from camera to robot base (${}^{\mathrm{B}}_{\mathrm{C}}T$) can be solved through the mathematical function, as in Eqs. (1) and (2). The calibration toolbox loads images and extracts grid corners of each image, and the transformation from the checkerboard to the camera (${}^{\mathrm{C}}_{\mathrm{Cb}}T$) is calculated with the known grid size. We can acquire the robot end-effector pose relative to the robot base from the controller and convert it to the transformation matrix $\left({}^{\mathrm{E}}_{\mathrm{B}}T\right)$ by following z–y–x Euler angle rotation. The calibration procedure is repeated until the minimum error value is obtained.

$$ {}^{\mathrm{E}}_{\mathrm{B}_1}T\,{}^{\mathrm{B}_1}_{\mathrm{C}_1}T\,{}^{\mathrm{C}_1}_{\mathrm{Cb}}T = {}^{\mathrm{E}}_{\mathrm{B}_2}T\,{}^{\mathrm{B}_2}_{\mathrm{C}_2}T\,{}^{\mathrm{C}_2}_{\mathrm{Cb}}T \tag{1} $$

$$ {}^{\mathrm{E}}_{\mathrm{B}_2}T^{-1}\,{}^{\mathrm{E}}_{\mathrm{B}_1}T\,{}^{\mathrm{B}_1}_{\mathrm{C}_1}T = {}^{\mathrm{B}_2}_{\mathrm{C}_2}T\,{}^{\mathrm{C}_2}_{\mathrm{Cb}}T\,{}^{\mathrm{C}_1}_{\mathrm{Cb}}T^{-1} \tag{2} $$

2.3 Object 6D Pose Computation

We annotate the object pose in the first image with the 6D object pose annotation toolbox, which is an online open source [12]. In this step, we manually align the 3D mesh model to the object in the 2D picture by rotating and moving the object mesh model along the x–y–z-axis. The object 3D mesh model is projected onto the image with intrinsic parameters to visualize the alignment result, as shown in Fig. 2. After the annotation process, the transformation matrix from the object to the camera (${}^{\mathrm{C}}_{\mathrm{O}}T$) will be recorded, including the translation and the rotation part.

With the hand-eye transformation matrix computed in the preparation step and the transformation matrix from object to camera frame in the first image annotated using the toolbox, the transformation from object to robot end-effector can be computed using Eq. (3) and used to calculate object poses. The object is fixed on the robot end-effector, which will not be moved during the data collection. The transformation between the object and end-effector, denoted as ${}^{\mathrm{E}}_{\mathrm{O}}T$, remains the same and only needs to be calculated once. With the known camera-robot transformation, combined with a sequence of recorded robot poses, the object 6D pose in each image can be solved by following Eq. (4). One concern is that the accuracy of the annotations relies on the quality of the first manual annotation to some degree.

$$ {}^{\mathrm{E}}_{\mathrm{O}}T = \left({}^{\mathrm{C}}_{\mathrm{B}}T\,{}^{\mathrm{B}}_{\mathrm{E}}T\right)^{-1}\,{}^{\mathrm{C}}_{\mathrm{O}}T \tag{3} $$

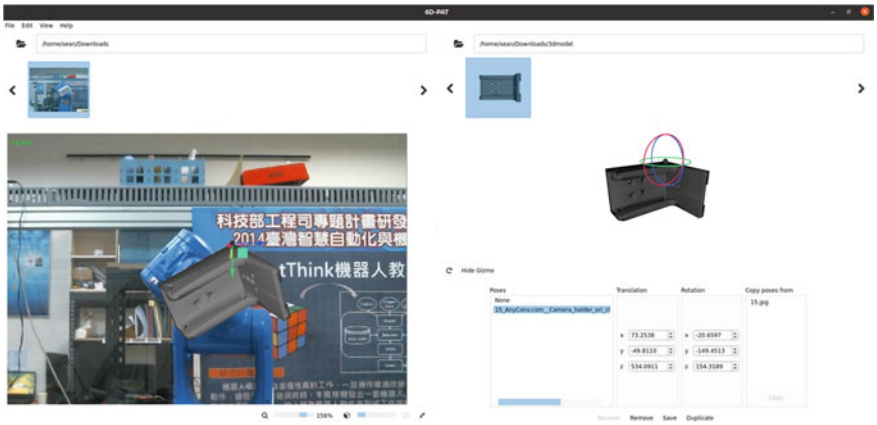


Fig. 2 6D pose annotation tool

$$ {}^{C}_{O}T = {}^{C}_{B}T\,{}^{B}_{E}T\,{}^{E}_{O}T \tag{4} $$

2.4 Projection of 3D Mesh Model on 2D Image

The purpose of this process is to create binary masks of the object. The object mesh model is required in this approach which contains 3D points of the object surface. After the object pose in the camera coordinate system is calculated, the mesh model will be rotated and translated within the 3D space according to the pose calculated in the previous step. The projection of each 3D point of the object in a 2D image plane can be computed using the triangulation method (Fig. 3).

In this process, camera intrinsic parameters are involved, which contain the camera focal length, and the optical center. In Eq. (5) below, X_c, Y_c, and Z_c are 3D points of the object represented in the camera coordinate system. f_x, f_y are focal lengths in pixels representing the distance from the camera frame's origin to the image sensor plane. c_x, c_y are optical centers in the image sensor plane represented in pixels. The computation in Eq. (6) yields the 2D pixel coordinate (u, v) projected from 3D points. In this procedure, segmentation masks (Fig. 4) and the object 2D bounding box location are recorded, which can be used to train an object detector or instance segmentation model.

$$ x' = \frac{X_c}{Z_c}, \; y' = \frac{Y_c}{Z_c} \tag{5} $$

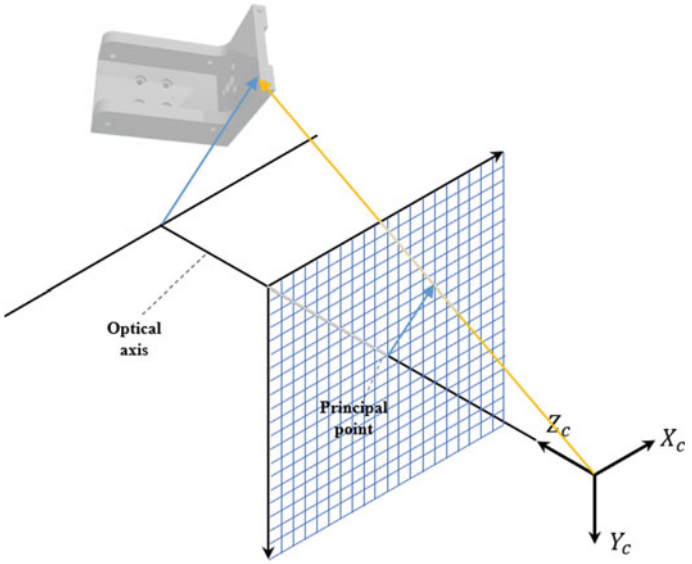


Fig. 3 3D mesh projection illustration

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{6}$$

3 Experiment

3.1 Robot Movement

A sequence of robot poses was created before collecting the object pose dataset. We represented the target robot pose in motor pulse value instead of Cartesian representation. Pulse value directly describes the joint movement, which is more intuitive. We created different robot poses by adjusting the pulse value of the 4th, 5th, and 6th axes computed within certain ranges and followed specific intervals, as shown in Table 1. We used three layers of loop function with 4th-axis on the top, 5th-axis in the middle, and 6th-axis in the inner to create a list of pulse sets. Figure 5 illustrates the moving direction of each joint, highlighted in red.

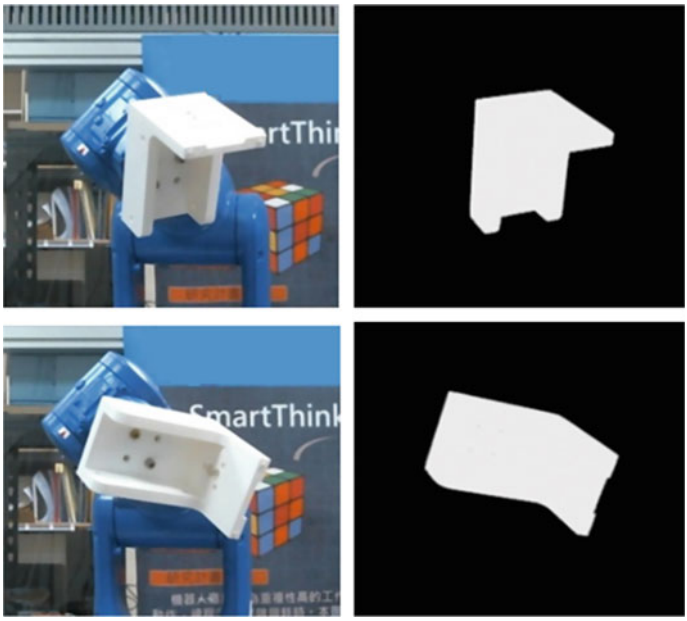


Fig. 4 Segmentation mask created by 3D mesh projection

Table 1 Rotation range and interval of axes

Axis	Min value	Max value	Interval
4th axis (R)	−40,000	40,000	4,000
5th axis (B)	−40,000	60,000	4,000
6th axis (T)	0	100,000	10,000

3.2 Training Data Collecting

Figure 6 shows the environment setup of the system. The camera was fixed in front of the robotic arm to capture the images of the object. The target object was attached to the robot end-effector. The robotic arm was operated to move the object to generate diverse poses of the object in 3D space as comprehensively as possible. The test object in this study is a jig used in the factory. The 3D model of the object is pre-designed, as shown in Fig. 7.

Figure 8 shows the data collection process. The PC sends a move instruction command to move the robot and capture an image with a corresponding robot pose when the robot arrives at the target position. The acquired robot pose is the input in (4) to compute the object's pose. With the object pose, the segmentation mask is created by 3D mesh model projection. We save the transformation matrix from object to camera, object binary masks, and images into the training dataset.

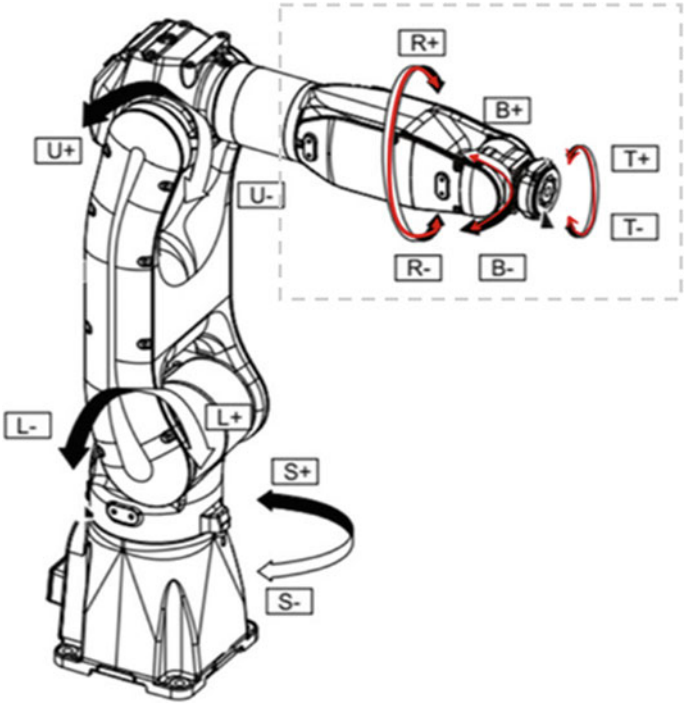


Fig. 5 Illustration of robot movement

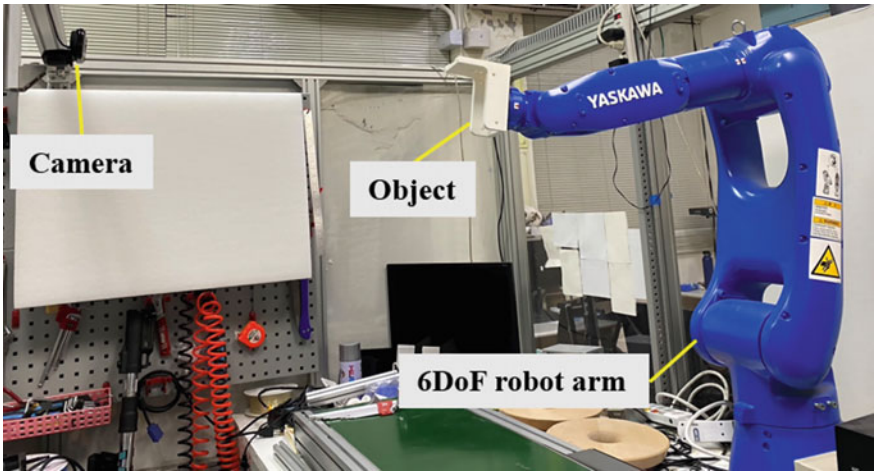
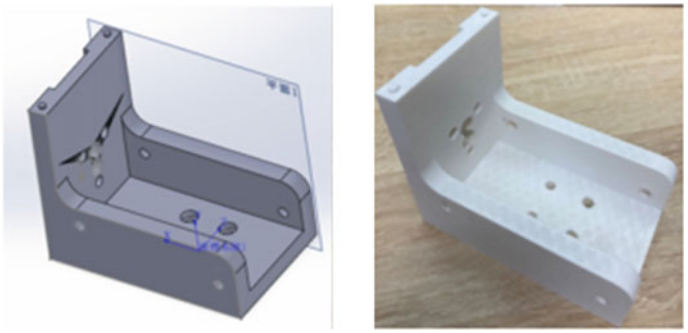


Fig. 6 Environment setup



(a) (b)

Fig. 7 **a** 3D mesh model of the object; **b** The actual 3D print object

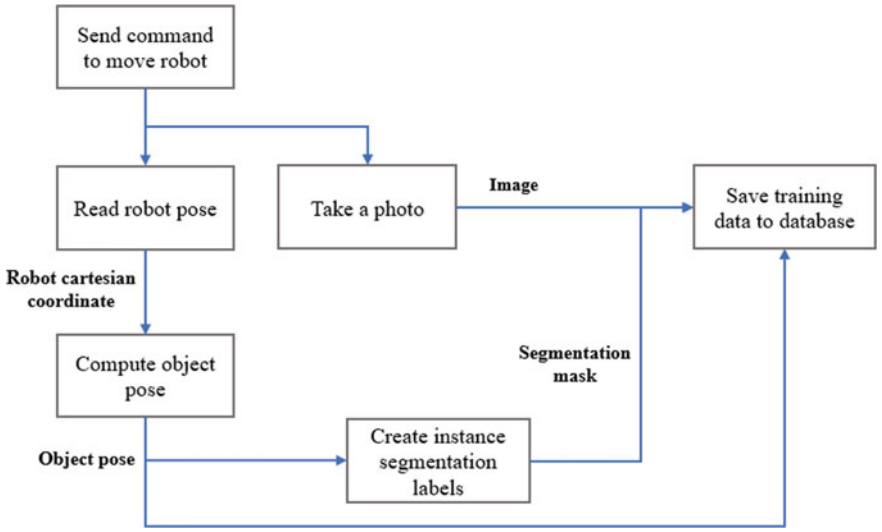


Fig. 8 Data collection process

Figure 9 shows the visualization of the front 800 object poses in the training dataset we collected. We transferred the object pose to camera views relative to the coordinate system of the object. The coordinate frame on the top of the object represents the camera coordinate system which contains the x-axis in green, y-axis in red, and the z-axis in blue. Since the robot movement was regular, relative camera views were uniformly distributed over the top of the object. We collected 3000 training data within 1 h with no humans involved.

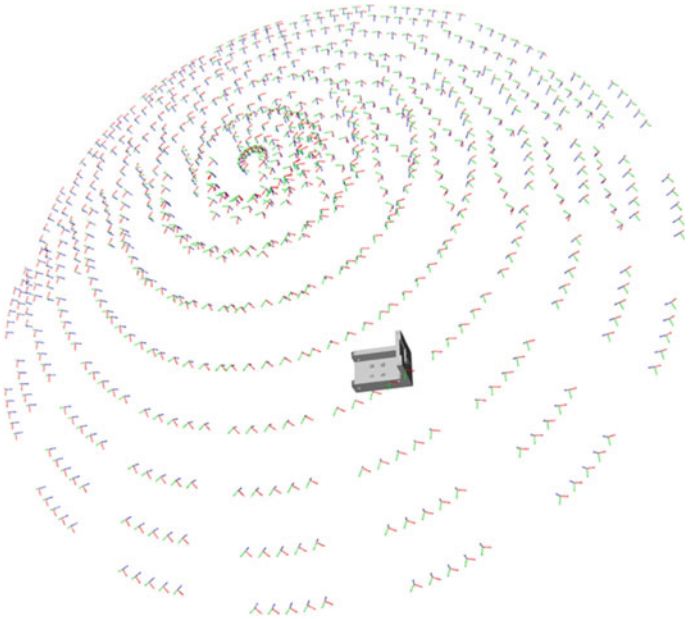


Fig. 9 Visualization of relative camera view

3.3 Object Pose Estimation Model

We implemented our custom dataset in a state-of-the-art two-stage object pose estimation method, Pixel-wise Voting Network [10], which predicts 2D object key points in images and computes the object pose through 2D-3D correspondences with a PnP algorithm. We split our dataset into two parts, 2,400 images for training and 600 images for testing, which were randomly selected from the dataset. We trained the key points detection network through 240 epochs.

We apply ADD metric in the evaluation which calculates the average distance between two converted 3D model points using the predicted rotation $\tilde{R}$ and translation $\tilde{T}$ and ground truth rotation R and translation T:

$$\text{ADD} = \frac{1}{m} \sum_{x \in \mathcal{M}} \left\| (Rx + T) - \left(\tilde{R}x + \tilde{T} \right) \right\|, \tag{7}$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points. It is claimed that the predicted pose is correct when the distance is less than 10% of the model's diameter. For symmetric objects, ADD-S metric [40] was applied, where the average distance is calculated based on the closest point distance in a 3D point set:

Table 2 ADD result in different threshold values

ADD threshold	ADD metric
0.1	0.9983
0.05	0.9
0.04	0.7983
0.03	0.6533

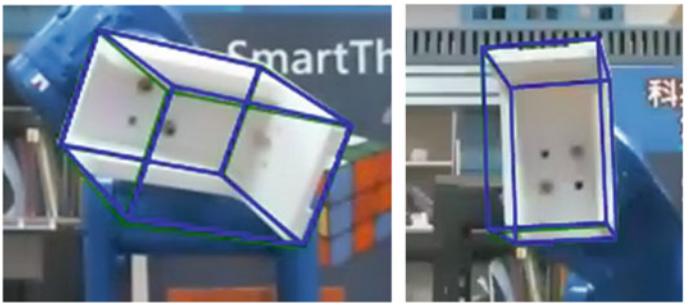


Fig. 10 A 6D pose estimation result

$$\text{ADD - S} = \frac{1}{m} \sum_{x_1 \in \mathcal{M}} \min_{x_2 \in \mathcal{M}} \left\| (Rx_1 + T) - \left(\tilde{R}x_2 + \tilde{T} \right) \right\|. \tag{8}$$

The result in Table 2 shows that even we decrease the threshold value to stricter conditions, over 90% of predictions meet the grasping condition. In this case, the threshold to the real distance is around 8 mm. Figure 10 shows the visualization results of two test images, where the green line represents the ground truth and the blue one is the prediction.

4 Conclusion and Future Works

The experiment in this study shows that the proposed object pose annotation system can automatically generate sufficient amounts of training data, including segmentation labels and object poses with respect to the camera frame. Although the annotation process still requires one manual labeling data, the rest of the collected data are automatically annotated using the known robot pose. It drastically reduces the data labeling time, which was performed by humans in tradition. The custom dataset generated by the auto-labeling system has been tested in existing pose estimation deep learning models. In the future, we will verify the pose estimation results in the real grasping scenario to demonstrate the feasibility of our training procedure. We also will discuss more methods to increase the generalization ability of the 6D pose estimation model in the aspect of data collection. In addition, data augmentation

can be implemented in training datasets, and we can compare the performance of different models.

References

1. Xiang Y, Schmidt T, Narayanan N, Fox D (2018) Posecnn: a convolutional neural network for 6D object pose estimation in cluttered scenes. In: Robotics: science and systems (RSS)
2. Hodan T et al (2018) BOP: benchmark for 6D object pose estimation. In: European conference on computer vision (ECCV), pp 19–34
3. Singh A, Sha J, Narayan KS, Achim T, Abbeel P (2014) BigBIRD: a large-scale 3D database of object instances. In: International conference on robotics and automation (ICRA), pp 509–516
4. Calli B, Singh A, Walsman A, Srinivasa S, Abbeel P, Dollar AM (2015) The YCB object and model set: towards common benchmarks for manipulation research. In: International conference on advanced robotics (ICAR), pp 510–517
5. Schwarz M, Behnke S (2020) Stillleben: realistic scene synthesis for deep learning in robotics. In: IEEE international conference on robotics and automation (ICRA), pp 10502–10508
6. Periyasamy AS, Schwarz M, Behnke S (2021) SynPick: a dataset for dynamic bin picking scene understanding. In 17th international conference on automation science and engineering (CASE), pp 488–493
7. Wang G, Manhardt F, Shao J, Ji X, Navab N, Tombari F (2020) Self6d: self-supervised monocular 6D object pose estimation. In: European conference on computer vision, pp 108–125. Springer, Cham
8. Wang G, Manhardt F, Liu X, Ji X, Tombari F (2021) Occlusion-aware self-supervised monocular 6D object pose estimation. IEEE Trans Pattern Anal Mach Intell
9. Kehl W, Manhardt F, Tombari F, Ilic S, Navab N (2017) SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: International conference on computer vision (ICCV), pp 1530–1538
10. Peng S, Zhou X, Liu Y, Lin H, Huang Q, Bao H (2022) PVNet: pixel-wise voting network for 6DoF object pose estimation. IEEE Trans Pattern Anal Mach Intell 44(6):3212–3223
11. Wang C et al (2019) DenseFusion: 6D object pose estimation by iterative dense fusion. In: IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 3338–3347
12. Blume F, 6D pose annotation tool (6D-PAT), https://github.com/florianblume/6d-pat. Last accessed 11 Oct 2022