2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Real-Time Robot End-Effector Pose Estimation with Deep Network

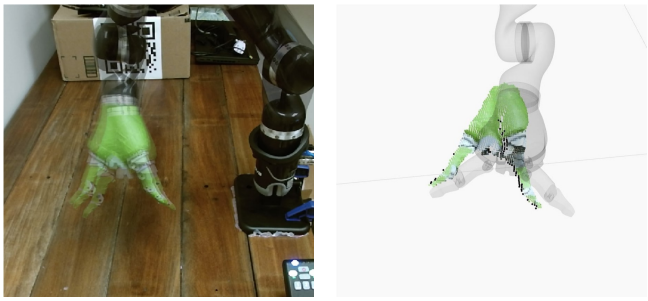
Hu Cheng, Yingying Wang and Max Q.-H. Meng*, *Fellow, IEEE*

***Abstract*— In this paper, we propose a novel algorithm that estimates the pose of the robot end effector using depth vision. The input to our system is the segmented robot hand point cloud from a depth sensor. Then a neural network takes a point cloud as input and outputs the position and orientation of the robot end effector in the camera frame. The estimated pose can serve as the input of the controller of the robot to reach a specific pose in the camera frame. The training process of the neural network takes the simulated rendered point cloud generated from different poses of the robot hand mesh. At test time, one estimation of a single robot hand pose is reduced to 10ms on gpu and 14ms on cpu, which makes it suitable for close loop robot control system that requires to estimate hand pose in an online fashion. We design a robot hand pose estimation experiment to validate the effectiveness of our algorithm working in the real situation. The platform we used includes a Kinova Jaco 2 robot arm and a Kinect v2 depth sensor. We describe all the processes that use vision to improve the accuracy of pose estimation of the robot end-effector. We demonstrate the possibility of using point cloud to directly estimate the robot's end-effector pose and incorporate the estimated pose into the controller design of the robot arm.**

I. INTRODUCTION

The development of the perception and control algorithm for robot arm and gripper has made complex tasks possible, like grasping novel household objects and folding clothes. To achieve better performance in the tasks, we need the grasp pose detection or trajectory generation of the robot hand itself to be executed accurately with high efficiency. However, the high accuracy and heavy load is not the primary design demand for the service robot arm. The service robots working in the human living environment should keep the safety of humans as the priority. Thus, they are usually designed with relatively small stiffness, therefore low absolute accuracy. To achieve a higher success rate in tasks for service-robot, we need visual servoing on the service robot to provide accurate robot pose.

Robot manipulation suffers from all kinds of noise. Inaccurate data from the low-cost sensor, poorly-calibrated zero position of the joint encoder, and the gap between worn physical machinery and the robot model provided by the manufacturer lower the performance of robot manipulator, even lead to failure, as shown in Fig. 1. It is not an issue for the industrial robot with a precise sensor and a highly rigid arm that works in a controlled environment. However, in a domestic setting, the positioning error of the robot arm degrade with safety concerns. The rigidity, weight, and cost for domestic robots are also lower than the industrial one, leading to much more noise to deal with.



(a) (b)

Fig. 1: (a) shows the repeatability of the robot arm, which is caused by mechanical imperfecions. (b) shows the misalignment between the pose of the end effector and the point cloud.

To tackle this problem, we tried offline calibration to estimate the erroneous parameters of the robot system. However, this method is limited to static parameters that do not vary during robot execution. This process needs to be repeated after a while as hardware aging and often requires high expertise. Thus it is difficult to perform for a customer without any domain knowledge.

Instead of offline calibration, the vision-only close-loop method bypass most sources of noise [1]. By directly using a visual signal as feedback, this method avoids coordinate transformation between camera and robot base. Besides, the vision system itself should be responsible for proprioceptive sensing, e.g., accurately capture the pose of robot end effector. Thus it is unaffected to hand-eye calibration, structure bending, and zero drift of joint encoder. However, accurate pose estimate of robot link is a non-trivial task with a cheap visual sensor [2].

In this paper, we propose a close-loop control model based on a novel hand pose estimation algorithm that can position the end-effector accurately in real-time. Instead of transforming the target pose back to the robot root frame, both the control signal and target are given only in the camera frame. As a result, it bypasses the large absolute measuring error of the depth sensor, because both the target and feedback are calculated only in the camera frame.

The main contributions of this paper are as follows: First, we directly estimate the robot hand pose with respect to the

Hu Cheng and Yingying Wang are with the Robotics, Perception and Artificial Intelligence Lab in the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong {hcheng, yingyingw}@link.cuhk.edu.hk

Max Q.-H. Meng is with the Department of Electronic and Electrical Engineering of the SouthernUniversity of Science and Technology in Shenzhen, China, on leave from the Department ofElectronic Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute of the Chinese University of Hong Kong in Shenzhen, China max.meng@ieee.org

* corresponding author

depth camera frame. With the 3D point cloud perception neural network, our algorithm gives each estimation at 100HZ on gpu, which can be used for the real-time close loop robot hand control policy. Second, following the estimation pose of the robot end effector, we design the control policy that targets at reducing the distance of the target status and the current status. Both statuses are calculated or estimated on the camera frame. Third, we validate our algorithm in the grasping task, and the robot arm can make precise execution and grasp small objects.

II. Related work

A. Robot Arm Tracking

There are many methods proposed to solve the robot arm pose estimation problem. [3], [4] tracked the key point of the robot arm, using SIFT or learning-based features, and optimized the pose with the motion model of the robot arm. The pose is initialized by joint angel distribution or by solving the PnP problems. Apart from those algorithms that rely on tracking the arm, [5]–[7] proposed frame-by-frame pose estimators using depths image as input. Those estimators use current depth images as the input to the probabilistic filter or random forest based posed estimator to output the pose of the articulated robot arm directly. They can achieve a fast robot hand pose estimation without knowing the previous pose. Consider the stability of the joint angle readings, [7], [8] used the probabilistic filter to form a multi-sensor fusion pipeline. [9]–[11] estimated the relative relationship between the real point cloud and the point cloud simulated by the camera intrinsic parameters and the joint angles. They calculated the distance between those two point clouds and then calculated its gradient with respect to the robot joints. This approach directly gave them the motion of the joint angles that reduced the distance between two point clouds, or we can regard the approach as the offset of the joint that used as compensation to the robot measurement system error.

B. Visual Servoing

According to [12], visual servoing refers to using computer vision as another system to control the motion of the robot. There are mainly two kinds of robot settings for visual servoing methods, one is the eye in hand [1], [13], and the other is eye to hand [14], [15]. As for the input to the visual servoing controller, some researchers first estimate the current pose of the robot [16], and reduce the distance between the target pose and current pose, others extract the image features from the image and design the controller that try to align the image features of current pose and the estimated image features of target pose.

C. Point Cloud Perception

Point cloud based perception is of increasing interest to the computer vision and machine learning community. PointNet [17] is the pioneer to directly process point clouds with a deep neural network, followed by many variants [18]–[21]. It extracts features for each point with a shared multi-layer perceptron (MLP) and outputs with an aggregation function invariant to permutation. [18]–[20] improve PointNet by introducing different strategies to sample and group points, and different modules to extract local region features.

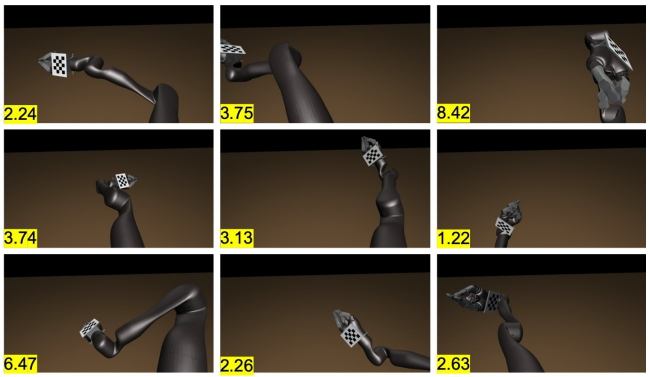


Fig. 2: Selected positions to estimate the error of checkerboard-based pose estimation. We random place the end effector attached with checkerboard in virtual environment. Then we can calculate the error between estimated pose and ground-truth provided by simulator. The number in the represents the position-only error in millimeter.

With large-scale 3D datasets, *e.g*. ShapeNet [22] and ScanNet [23], PointNet based approaches achieve great success on shape classification, semantic segmentation, and instance segmentation. Besides, there are a few works focusing on pose estimation [24], [25] and point cloud registration [26], [27]. [28] is the most relevant to our method. It takes a segmented point cloud of one object as input and predicts the 6-DoF object pose (axis-angle) with PointNet. It shows more robustness against occlusion compared to methods with RGB information only. Inspired by previous works, we use PointNet to estimate the pose of the end effector given its point cloud segment. Since it is difficult to acquire ground truth poses for supervision, we adopt a sim-to-real strategy and focus on narrowing the domain gap, which differs our method from others.

III. Robot Platform Settings

In this section, we give the detail description of a typical robot grasp process, errors from the hardware setting, and key component of our robot platform.

A typical robot task execution close loop consists of two parts, perception and control. For the perception part, the robot uses all its sensors data to estimate its status and the surrounding environment. Then, the robot calculates the distance of the target status and the current status, then the corresponding control policy for the actuators. Finally, the actuators move the robot to its target status. Take the grasping task as an example. The Kinect depth sensor takes one depth image of the objects and find the 6d robot hand pose [x, y, z, pitch, yaw, roll] where the robot hand can successfully grasp the object when close its gripper. Then, the 6d hand pose is transformed from the camera frame to the robot base frame using the transformation matrix, H_B^C. Finally, the robot arm

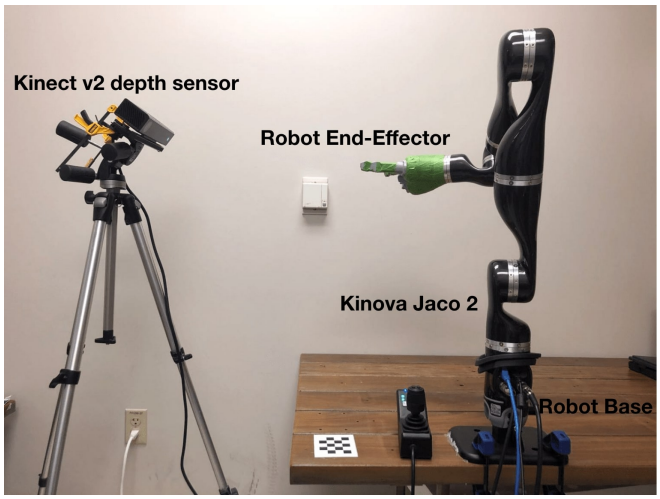


Fig. 3: The robot platform of our system. We use Kinova Jaco 2 robot arm and Kinect v2 depth sensors. The camera position is set to eye-to-hand setting.

controls the end-effector to move to the specific pose with regard to the base.

In the grasping task, the robot should not only optimize the grasp poses but also execute the command precisely to make the grasp success. However, two errors make the task failed even when the grasp pose detection is correct. First, the robot itself is limited by repeatability and accuracy. Second, for the hand-eye calibration of the H_B^C, there is a reprojection error. Moreover, after working for hours, the temperature of the robot arm joint actuator rises result in the decreased accuracy of the joint angle position. All those errors together make grasping tasks fail in the end. Among all these errors, the hand-eye calibration error is the only error that can be reduced by offline hand-eye-calibration.

Thus, Many online tracking methods have been proposed to make a better estimation of the robot hand pose or use the servoing strategy to control the robot arm move to the target pose. Those online methods expressed the current status and target status of the robot in the same measurement system, for example, the camera. Furthermore, the control policy is also established on the distance measured between the current status and the target status, which can eliminate all the other measurement system errors, like joint actuators noise or hand-eye calibration errors. For example, most pose based visual servoing (PBVS) algorithm use the checkerboard to estimate the pose of the target object and the robot end effecor. However, the camera noise make the estimation inaccuracy in specific poses. The details are shown in Fig. 2.

The industrial robot arm, like UR5, attached to the fixed platform stably could provide high absolute accuracy, but they cannot serve in household as service robot. Industrial robot arm usually equipped with a considerable control box that needs the 110V-240V alternating power which is not possible in household. The service robot arm working in a human house must be fixed at a mobile platform to increase the working space, which means the robot is powered by the battery. Thus, in our experiment platform, we choose to use the low-power designed robot arm, Kinova Jaco 2.

Common depth sensor like Kinect2 can not capture the environment well when the object is close to the camera, normally within 0.3-0.5 meters. Thus, eye-to-hand setting can get a more complete view than fix the camera with respect to the robot gripper. Our platform contains two measurement systems, including joint encoders readings direct from the robot actuator and visual information from Kinect2 camera. The robot frame is established on the robot base, and we can calculate the robot hand pose with respect to the robot base using the joint encoders readings and the robot link parameters (D-H table).

The noise of joint encoder reading will vary within the workspace due to gravity-driven deformation. Reported by manufacturer, the overall accuracy of the Kinova Jaco 2 arm is 1.5cm in average. For Kinect2, the average deviations within 1m is around 2.5mm, and the standard deviation is smaller than 0.5mm [29]. The camera measurement system is more accurate than the robot arm for pose estimation purpose. Furthermore, we only care about the relative positional error instead of absolute positioning error of depth sensor for close-loop control. Fig. 3 shows the platform that we use.

IV. Method

A. Pose Estimation

In this section, we describe the structure of our pose estimation network. The input is a point cloud $x \in \mathcal{R}^{n\times 3}$. The network outputs the rotation $R \in \mathcal{R}^{3\times 3}$ and the translation $t \in \mathcal{R}^3$. Following [25], we use 6D vectors to represent rotations rather than quaternion or Euler angles. Concretely, the 6D continuous representation of one rotation R is $r \in \mathcal{R}^6$, which is the vectorization of the first two columns of R. Namely, $r = [R_{11}, R_{12}, R_{13}, R_{21}, R_{22}, R_{23}]^T$. We can apply the Gram-Schmidt process to convert a 6D representation r to the corresponding rotation matrix. We refer readers to [25] for more details. Thus, the final output of the pose estimation network is a 9D vector, which consists of a 6D vector for rotation and a 3D vector for translation.

The network architecture is illustrated in Fig. 4. Our pose estimation network is based on PointNet [17]. Specifically, we employ two stacked PointNet layers, similar to PCN [21]. For the first layer, the input point cloud x is processed by a 2-layer MLP to get point-wise features $f^1 \in \mathcal{R}^{n\times d_1}$, where d_1 is the feature dimension. We apply a point-wise max-pooling on f^1 to get a global feature $g^1 \in \mathcal{R}^{d_1}$. The structure of the second layer is the same as that of the first layer, while the input is the concatenation of the point-wise features f^1 and the global feature g^1. The output of the second layer is another global feature of g^2, followed by a 3-layer MLP to predict the 9D pose vector. Batch normalization and ReLU are added for each MLP.

To facilitate the training process, we preprocess the input point cloud. Given an input point cloud, we centralize it according to its axis-aligned bounding box and normalize it by setting the longest size of the bounding box to be 1m. The

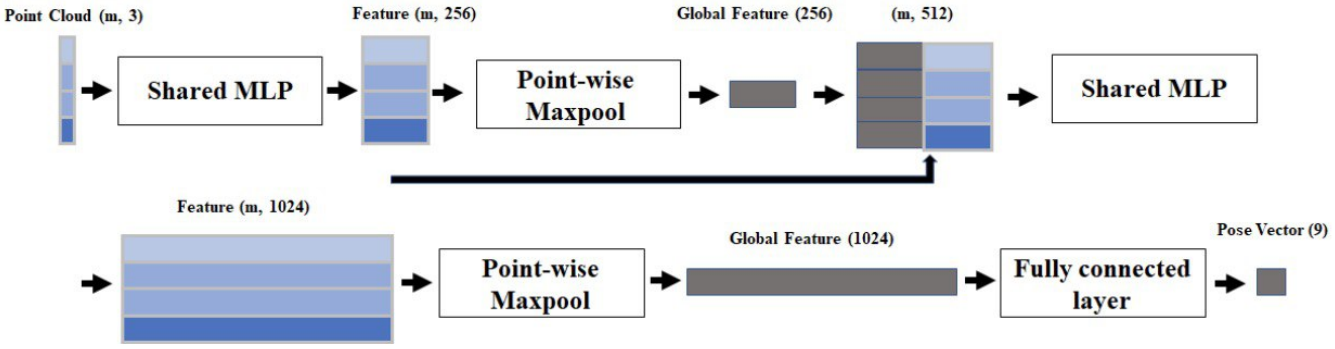


Fig. 4: The architecture of pose estimation network. The input of the network is the robot end-effector point cloud and the network output the pose directly.

preprocessing does not require any ground truth information and can be done efficiently.

B. Close-loop Controller

We designed a close loop controller that uses the vision-only measurement system. First, we estimate the point cloud using the method described in Sec. IV-A. Then, we calculate the error between the target and the current pose. A Cartesian velocity controller is designed based on this error.

Here are the details of the controller:
Step 1: Estimate the robot hand pose using the cropped point cloud.
Step 2: Calculate the Cartesian velocity of the robot hand using the PD controller:

$$v(q) = \mathbf{J}|_q^+ (H_{base}^{target} - H_{base}^{hand}) \tag{1}$$

Where $\mathbf{J}|_q^+ = (J^T J)^{-1} J^T$ is the geometrical Jacobian pseudo-inverse for the robot hand (end-effector) and $v(q)$ is the joint velocity.
Step 3: Calculate the corresponding velocity $v(q)$ of each joint based on the Cartesian Velocity using differential inverse kinematics.

It is necessary to use imprecise hand-eye transformation in order to command the robot. However, this error caused by hand-eye calibration error can be eliminated with iterations. The magnitude of joint velocity will diminish when the hand gripper is closed to the target pose. Thus, this error will not influence the final result of close-loop control.

V. Experiment

A. Dataset Construction

Given the CAD model of the end-effector, we generate a synthetic dataset to train our pose estimation network. The model of the robot hand is placed with a random pose, and a depth image is rendered to acquire the point cloud. To sample different poses, we represent rotations by pitch, yaw, and roll. The ranges of pitch, yaw and roll are $[0°, 360°]$, $[-45°, 45°]$, $[-150°, 60°]$ respectively. For translation, we sample the distance ranging from 0.6m to 0.8m and rotate it by an angle ranging from $-20°$ to $20°$. There are a total of 200K point clouds for training.

We also built a real-world gripper pose dataset of 540 gripper pose, capturing both the point cloud from Kinect2 sensor and robot configuration. We select 10 different positions, where each position includes 54 different orientations that cover most of the manipulation poses.

B. Implementation Details

The pose estimation network is trained by the Adam optimizer for 100K steps. The learning rate is 0.001, and the batch size is 32. To narrow the gap between the synthetic and the real, we employ data augmentation. During training, 2048 points are randomly sampled as input. Besides, up to 50% points may be dropped out to simulate the occlusion and noise in the real world. In addition, points are jittered up to 5mm along the radial axis to mimic the sensor noise.

C. Baseline and Evaluation Metric

We compare our approach against the offline-calibration baseline. For the offline-calibration baseline, the pose of the end-effector is calculated using forward kinematics based on a manufacturer-provided robot model. The baseline method is subjected to arm bending and wearing.

To evaluate the performance of pose estimation, we report the average distance between the point cloud and the surface of the end-effector mesh. Given an estimated pose, we first transform the observed point cloud in the camera frame to the end-effector frame. Then we compute the distance between each point to the closest surface of the end-effector. The average surface distance measures how close the transformed point cloud is to the end-effector model, which indicates how good the pose estimation is. Note that the noise of the depth sensor is around 3mm, which can be considered as the lower bound of the reported metric.

VI. Result

A. Pose Estimation

Table I shows the average surface distance for 540 collected real point clouds. Our pose estimation network outperforms the offline-calibration baseline by a large margin.

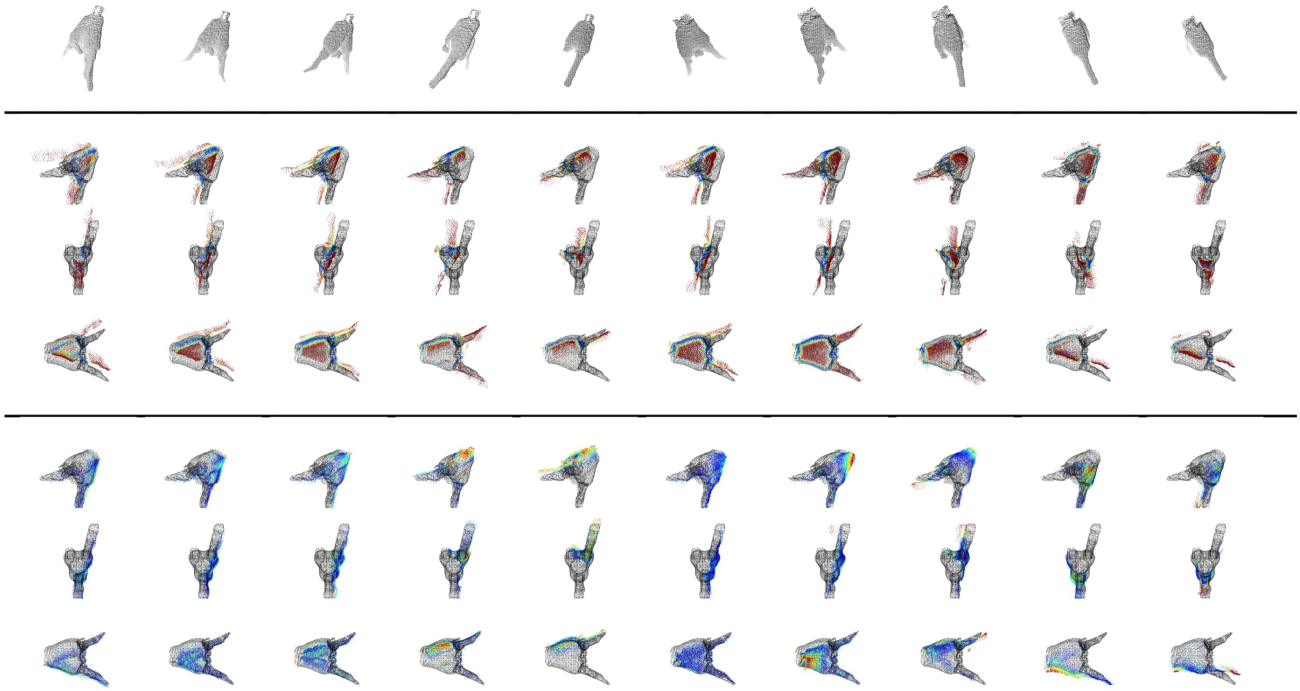


Fig. 5: Qualitative comparison between our pose estimation and the baseline. Each column illustrates one instance. The first row shows the input point cloud. We show both the transformed point cloud and the end-effector mesh from three different fixed viewpoints in the other rows. The 2nd to 4th rows show the results of the baseline, and the 5th to 7th rows show the results of our approach. The color indicates the surface distance of each point, where a warmer color implies a larger distance.

Fig. 5 demonstrates the qualitative results. It is observed that our pose estimation can induce point clouds much closer to the end-effector model.

TABLE I: The average surface distance(mm) for 540 collected real point clouds.

Ours	Baseline
8.8	13.7

Besides, we compare the chamfer distance between the point cloud transformed by the ground-truth poses calculated from the 3D marker and the estimated poses. Table II compares the chamfer distance against the ground truth.

TABLE II: The chamfer distance(cm) between the ground truth and the estimated pose.

Ours	Baseline
1.64	2.33

B. Experiment

We validate the effectiveness of our robot end effector pose estimation algorithm by grasping the small and complex object. The grasp pose is generated using the grasp detection algorithm proposed by [30], then the robot hand move to the target pose with open loop control and visual servoing, respectively. We find that for the same detected grasp pose, an inaccuracy execution of the end effector could result in failure trial. The details are shown in Fig. 6.

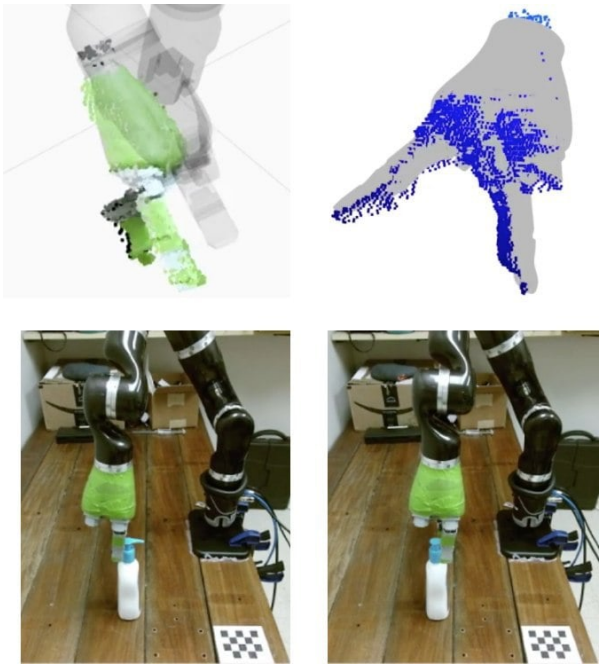


Fig. 6: From left to right, the meshes in upper row images show the pose of the robot end effector calculated by the robot forward kinematics and estimated by our pose estimator algorithm, respectively. From left to right, the lower row images show the result of robot grasp trial with or without pose estimator, respectively.

VII. Conclusion and Future Work

In this paper, we propose the whole scheme for using the Kinect depth sensor as the additional measurement system to increase the accuracy of the robot hand pose estimation. Our method uses the cropped point cloud as the system input and directly output the estimated pose with respect to the camera using a deep pose estimator. The processing time for each point cloud sample satisfy the real-time requirement and estimate the pose frame-by-frame. Thus, the accuracy of the robot hand pose estimation has been increased compared with the robot measurement system.The grasping experiment shows an increment of the accuracy of robot arm pose estimation make grasp small object become possible, which usually failed due to the robot arm can not move to the desired pose.

As for future work, we think the robot arm articulated structure provides extra prior information about the pose of the end effector. Instead of only estimating the pose of the hand of the robot, combine the estimated poses of all the links should make a better estimation of the end effector than only estimating the end effector itself.

ACKNOWLEDGMENT

This work is partially supported by Hong Kong ITC ITSP Tier 2 grant # ITS/105/18FP: An intelligent Robotics System for Autonomous Airport Passenger Trolley Deployment awarded to Max Q.-H. Meng..

References

[1] D. Morrison, J. Leitner, and P. Corke, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," *Robotics: Science and Systems XIV*, 2018.

[2] C. Garcia Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic Articulated Real-Time Tracking for Robot Manipulation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 577–584, 2017.

[3] C. Rauch, V. Ivan, T. Hospedales, J. Shotton, and M. Fallon, "Learning-driven Coarse-to-Fine Articulated Robot Tracking," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2019, pp. 6604–6610.

[4] J. Lambrecht, "Robust Few-Shot Pose Estimation of Articulated Robots using Monocular Cameras and Deep-Learning-based Keypoint Detection," *2019 7th International Conference on Robot Intelligence Technology and Applications, RiTA 2019*, pp. 136–141, 2019.

[5] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3143–3150, 2014.

[6] F. Widmaier, D. Kappler, S. Schaal, and J. Bohg, "Robot arm pose estimation by pixel-wise regression of joint angles," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 616–623, 2016.

[7] C. Garcia Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic Articulated Real-Time Tracking for Robot Manipulation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 577–584, 2017.

[8] P. Hebert, N. Hudson, J. Ma, T. Howard, T. Fuchs, M. Bajracharya, and J. Burdick, "Combined shape, appearance and silhouette for simultaneous manipulator and object tracking," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2405–2412, 2012.

[9] M. Klingensmith, T. Galluzzo, C. Dellin, M. Kazemi, J. A. Bagnell, and N. Pollard, "Closed-loop Servoing using Real-time Markerless Arm Tracking," *IEEE International Conference on Robotics and Automation Workshops*, 2013.

[10] T. Schmidt, R. Newcombe, and D. Fox, "DART: Dense Articulated Real-Time Tracking," no. 1, 2015.

[11] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in hand model acquisition," *Proc. of the Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation at the Int. Conf. on Robotics & Automation (ICRA), Anchorage, Alaska*, vol. 30, no. 9, pp. 1–34, 2010.

[12] S. Hutchinson, "Part I : Basic Approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. December, pp. 82–90, 2006.

[13] J. A. Piepmeier and H. Lipkin, "Uncalibrated eye-in-hand visual servoing," *The International Journal of Robotics Research*, vol. 22, no. 10-11, pp. 805–819, 2003.

[14] K. Pauwels, V. Ivan, E. Ros, and S. Vijayakumar, "Real-time object pose recognition and tracking with an imprecisely calibrated moving RGB-D camera," in *IEEE International Conference on Intelligent Robots and Systems*. Institute of Electrical and Electronics Engineers Inc., 10 2014, pp. 2733–2740.

[15] W. Burger, E. Dean-Leon, and G. Cheng, "Robust second order sliding mode control for 6d position based visual servoing with a redundant mobile manipulator," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1127–1132.

[16] W. J. Wilson, C. C. Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, 1996.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 44, no. 9, 6 2017, pp. 1517–1526.

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph Cnn for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, 2019.

[20] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds," 2019.

[21] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, pp. 728–737, 2018.

[22] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," 2015.

[23] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2432–2443, 2017.

[24] W. Yuan, D. Held, C. Mertz, and M. Hebert, "Iterative Transformer Network for 3D Point Cloud," 2018.

[25] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 5738–5746, 2019.

[26] Y. Wang and J. M. Solomon, "Deep Closest Point: Learning Representations for Point Cloud Registration," 2019.

[27] J. Grob, A. Osep, and B. Leibe, "AlignNet-3D: Fast Point Cloud Registration of Partially Observed Objects," *Proceedings - 2019 International Conference on 3D Vision, 3DV 2019*, pp. 623–632, 2019.

[28] G. Gao, M. Lauri, J. Zhang, and S. Frintrop, "Occlusion resistant object rotation regression from point cloud segments," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11129 LNCS, pp. 716–729, 2019.

[29] E. Lachat, H. Macher, T. Landes, and P. Grussenmeyer, "Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling," *Remote Sensing*, vol. 7, no. 10, pp. 13 070–13 097, 2015.

[30] H. Cheng and M. Q.-H. Meng, "A grasp pose detection scheme with an end-to-end cnn regression approach," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 544–549.