Learning Markerless Robot-Depth Camera Calibration and End-Effector Pose Estimation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Traditional approaches to extrinsic calibration use fiducial markers. Learning based approaches rely heavily on simulation data. In this work, we present a learning based markerless extrinsic calibration system that uses a depth camera. We learn models for end-effector (EE) segmentation, single-frame rotation prediction and keypoint detection, from automatically generated real-world data. We use a transformation trick to get EE pose estimates from rotation predictions and a matching algorithm to get EE pose estimates from keypoint predictions. We further utilize the iterative closest point (ICP) algorithm, multiple-frames and outlier detection to increase calibration robustness. Our results on the test set with previously unseen camera locations give $1.18cm$ and $2.78°$ average calibration errors and $0.48cm$ and $3.58°$ average pose estimation errors.

Keywords: Camera Calibration, Pose Estimation, Perception

1 Introduction

Camera to robot calibration is an important step in many robotic applications. Keeping the correct calibration is a challenge for robots in dynamic and uncontrolled environments, even if the robot and the camera are meant to be static. Multiple factors alter this calibration such as re-positioning the robot and/or the camera for better workspace coverage or for different applications (e.g. research projects), inadvertently moving them for cleaning, people bumping into camera rigs and robots, wear-and-tear and backlash on low-cost fixtures etc. As such calibration is a needed but time-consuming process. At the very least, calibration needs to be double-checked before the real work can begin. This is an all-too-real issue for robotic researchers working with robot arms, which is getting more wide-spread as cage-free robot arms and mobile manipulators become more common in the industry.

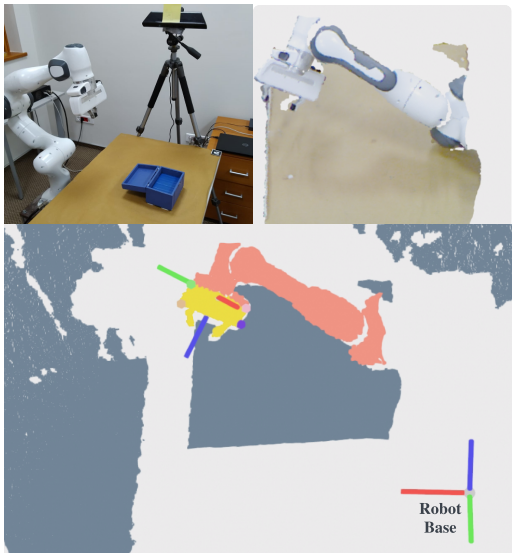

Figure 1: Top-left: experimental setup, top-right: sample frame from Kinect V1, bottom: segmentation, keypoints and pose predictions

Traditional extrinsic calibration approaches rely on fiducial markers. Checkerboard patterns and augmented reality tags are commonly used in research settings. Adding markers is error-prone due to intrinsic calibration errors, sensor noise, robot-to-marker fixture quality (which introduces errors in transformation between the markers and the robot) etc. Almost all the approaches benefit

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

from post-processing steps to improve the calibration quality such as the iterative closest point (ICP) algorithm, if a reference model is available. High precision systems employ precision machined fixtures and active markers (e.g. leds, infrared reflective materials along with infrared light-sources) which are costly. Thus, using markers is not an ideal solution for most applications.

In this work, we develop a system that can handle the extrinsic calibration between a depth camera and a robot arm without additional hardware, markers or simulation. We utilize recent developments in deep learning models for point clouds in combination with the ICP algorithm. Our system collects its own data, negating the need for manual labelling. Two side benefits of our system is that (1) it provides high quality semantic segmentation information about the background, end-effector and the rest of the robot, and (2) estimates the end-effector (EE) pose with high accuracy.

The main contributions of our system are; (1) high performance (average test errors of $1.18cm$ and $2.78°$) markerless extrinsic calibration, (2) high performance (average test errors of $0.48cm$ and $3.58°$) EE pose estimation wrt camera, and (3) automatic real-life data collection methodology.

2 Related Work

Calculating the transformation between a robot's base and a camera, usually called extrinsic calibration, is a decades old challenge [1]. Conventional techniques use fiducial markers to estimate camera to robot transformations. Most popular approaches stick a marker on robot's EE [2, 3, 4, 5] or other relevant parts [6, 7, 8] and collect data from multiple robot joint angle configurations to be used in offline calibration calculation. Major disadvantages of using such markers include wear-and-tear, distance dependent camera noise [9], fixture backlash and build quality etc. There are also single frame methods that use fiducial markers to estimate the EE pose [10, 11] in real-time. These methods start to fail when the reference objects are not as visible enough as anticipated. In this work, we aim to develop a markerless approach to handle extrinsic calibration.

The advent of deep learning brought developments in areas that are closely related to EE pose estimation such as 6-DoF object pose estimation [12, 13] and keypoint detection [14, 15]. Several methods have produced successful results on robot pose estimation from a single frame. Lambrecht [16] and Lee et al. [17] learn to estimate keypoints of entire robot arm from RGB image trained with newly created real world data and synthetic data, respectively, then to predict the robot pose combining keypoint data with forward kinematics information. Similarly, Labbé et al. [18] learns to predict robot pose and joint angles from RGB images via iterative CAD to image matching using a synthetic dataset [17]. Our system diverges from these methods in the following areas: (1) they require majority of the robot arm to be visible while our only requirement is to see the EE in the frame (2) our method uses depth data whereas others rely on Perspective-n-Point algorithm to compute affine transformations since they utilize 2D images, (3) we can readily apply the ICP algorithm to refine our pose estimates due to the depth data usage, and (4) we do not require any synthetic/simulation data and only use real-world data. It can be argued that we are solving a simpler problem (only EE and depth data) in exchange for higher calibration quality.

Hand-in-eye calibration, where camera is mounted to the EE, is another example of camera to robot calibration problem to which conventional solutions that we explained earlier are also applicable. Partially similar to our system, a recent method work on hand-in-eye calibration via deep learning [19]. This method uses synthetic data and 2D images, and a different robot-camera setup.

3 Method

Our system consists of two main stages; (1) single frame EE 6-DoF pose estimation, and (2) multi-frame extrinsic calibration. The output of the former along with forward kinematics is enough to get a calibration estimate but multiple frames increases robustness. Fig. 2 shows our single frame EE pose estimation and Fig. 4 shows our calibration workflows.

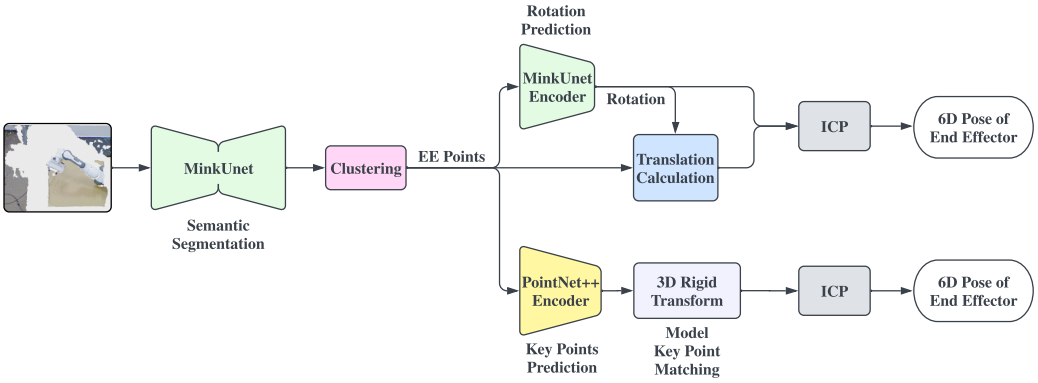

Figure 2: Single frame prediction architecture.

For EE pose estimation, we first segment the EE from a point cloud using a semantic segmentation approach. Then we utilize two approaches to estimate the EE pose. The first one does EE rotation prediction followed by a point cloud transformation step to get the EE translation. The second one extracts keypoints from the EE and matches these with reference points calculate the EE pose. Both approaches are followed by an ICP step initialized with the EE pose estimates to match the EE points and the EE CAD model. These steps are summarized in Fig. 2.

For calibration, the system collects multiple frames and the corresponding EE pose estimates as described above. To increase robustness, sanity check and outlier detection steps are employed. The remaining poses are then averaged in the rigid-body transformation space (the SE(3) manifold), to get the final extrinsic calibration result. These steps are summarized in Fig. 4.

In this section, we present the details of our work starting from the data collection step.

3.1 Data Collection

Our EE segmentation and pose estimation rely on learning and as such require robot specific data. We collect ground truth semantic segmentation labels, EE poses with respect to the depth camera and keypoint information. We collect this data from a real world setup to eliminate the need to build a simulation environment.

The first step is to perform an initial calibration. This can be done by existing approaches. We use a combination of markers, manual tuning on RViz [20] and ICP [21] towards this end. This is the only manual step that is needed for our system and is only required for data collection. Once our system is up and running, no such steps are necessary. This calibration information gives us the robot poses with respect to the camera frame.

The second step is to collect semantic segmentation data. We label points as background, EE and the rest of the robot automatically. Background points are obtained by utilizing frames without the robot. Then, the robot is moved to multiple visible positions to collect more frames. Semantic segmentation data for the robot arm is generated by subtracting the background points from every frame [22]. The EE points are further obtained by transforming the EE bounding box using the calibration information and taking the points inside this box. The CAD model of the robot along with joint positions and the calibration information can be used to get the arm and EE points, and consequently background points, as well. This would be needed when the robot arm is visible in the frame no matter what, to get the background points.

The third step is to generate keypoint information. We extract a total of six keypoints; four are located at each corner of the EE and two are located at the tip of the gripper fingers as shown in Fig. 3. We define six reference points within the EE bounding box which we already transform

based on the pose information in the previous step. The EE points closest to these reference points are selected as keypoints, as long as their distances are below a threshold. If no EE points for a given reference points is within the threshold, we deduce that the corresponding keypoint doesn't exist.

For this work, we collected 10000 frames from 3 different camera poses with 2 different camera heights and 2 different levels of lighting. To increase the generalization, we applied 3D data augmentation methods such as elastic distortion, noise injection and point dropouts during model training. A potential benefit of separating the background data is the ability provide different backgrounds to increase generalization performance, especially if the environment is prone to dramatic changes. However, we did not need this step for the presented work.

3.2 End-Effector Segmentation

The EE pose estimation starts by extracting the semantic segmentation information from an input point cloud. The outputs of this step are the EE, rest of the arm and the background labels for the points as shown in Fig. 1. For this step, we utilize a custom, sparse version of the Unet architecture named the Minkowski Unet network [23, 24, 25]. We use the full version of MinkUNet18D [23] trained with the raw point clouds as the input and the collected semantic segmentation data as the target. We then apply linkage clustering [26] to points classified as EE in order to reject superfluous predictions.

3.3 Pose Estimation with Rotation Prediction and Transform Calculation: RPT

In this approach, we first predict the EE rotation from EE points by utilizing the encoder part of another MinkUNet18D [23] backbone with the same configuration as the previous one. We train this model with the segmented EE points as input and the EE with respect to camera rotation as the target, using the loss of the PoseCNN model [12].

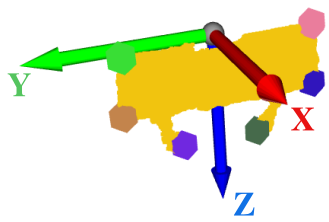

Figure 3: Visualization of the end-effector segmentation (yellow), end-effector frame, and keypoints (colored hexagons).

To calculate the EE translation, we use the inverse of the obtained rotation to rotate the points to their canonical locations wrt. the camera. Then we use the relationship between the center of the EE frame and the points to calculate the EE translation in the canonical pose. We finally rotate this back to get the EE translation wrt. the camera.

The Eq. 1 is used on each of the EE points to get the canonical locations, where R^{EE} is the rotation prediction of the model, P^{EE} is the position of the EE point and the $\bar{P}^{EE}$ is its rotated position.

$$\bar{P}^{EE} = (R^{EE})^{-1} \times P^{EE} \tag{1}$$

When the points are rotated in the camera frame, the relationship between them and the EE frame center is as follows; the x-axis is 0.015m inside the surface of the EE, the y-axis is in the middle and the z-axis is the closest to the camera. The Fig. 3 shows these axes superimposed on the segmented EE points. We use the following equations to capture this relationship where $\bar{t}^{EE}$ and $\bar{P}^{EE}$ denote the axes positions of the EE frame and EE points in the rotated frame.

$$\bar{t}_x^{EE} = \max(\bar{P}_x^{EE}) - 0.015 \tag{2}$$

$$\bar{t}_y^{EE} = \frac{\max(\bar{P}_y^{EE}) - \min(\bar{P}_y^{EE})}{2} \tag{3}$$

$$\bar{t}_z^{EE} = \min(\bar{P}_z^{EE}) \tag{4}$$

We finally rotate $\bar{t}^{EE}$ back to get the EE translation, t^{EE}, wrt. the camera using Eq. 5.

$$t^{EE} = R^{EE} \times \bar{t}^{EE} \tag{5}$$

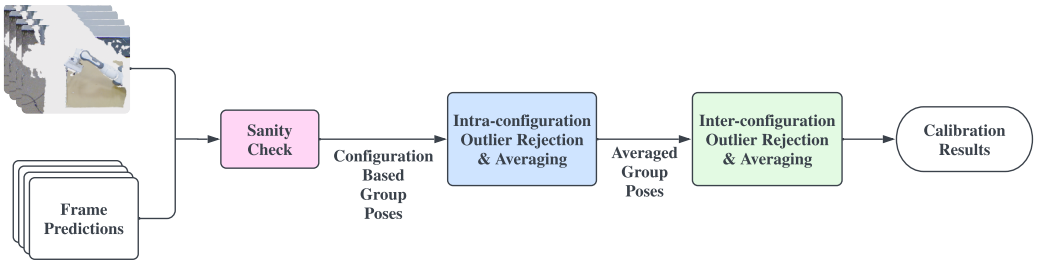

Figure 4: Calibration architecture.

Calculating the EE translation with such an approach is efficient and accurate given that the rotation predictions are accurate and that the EE is visible. However, full EE visibility is not a given for all the frames and as such, this method cannot be trusted for single frame pose estimation by itself.

3.4 End-Effector Pose Estimation via Keypoint Matching: KPM

In this approach, we first predict keypoints among the EE points. We use a dense point cloud encoder, PointNet++ [27] trained with segmented EE points as input and the collected ground truth keypoints as target. We then use a version of the l-squares fitting [28] algorithm to find the rigid transformation between predicted keypoints and the reference points, which gives us the EE pose. This is possible when there are at least four high quality keypoint predictions.

3.5 Pose Estimation Refinement via the Iterative Closest Point Algorithm

We described two different techniques to predict the EE pose from segmented EE points; RPT in Sect. 3.3 and KPM in Sect. 3.4. Even though these approaches produce relatively good results, there is room for improvement. We use an off-the-shelf ICP algorithm [21, 29] to improve the pose estimation accuracy. An ICP algorithm requires three inputs; the initial pose estimate (needs to be fairly accurate), a source point cloud, and a target point cloud. In our case, the initial pose estimate comes from either the RPT or the KPM methods, and the target point cloud comes from the EE segmentation. To generate the source points, we convert the CAD model of the robot's EE, provided by the manufacturer [30], to a point cloud, and transform the points based on the current EE configuration. The ICP algorithm outputs the transformation needed to match the source and target points which we use refine our initial pose estimates.

3.6 Camera to Robot Base Calibration

A single EE pose with respect to the camera frame can be used to calculate the transformation between the camera and the robot using Eq. 6, where T_X^Y represents the homogeneous transformation between frames X and Y, and the letters B, C, and EE correspond to the robot base, camera and the EE respectively. The EE in the base frame, T_B^{EE}, is obtained by forward kinematics and the EE in the camera frame, T_C^{EE}, is estimated by either the RPT or the KPM methods.

$$T_C^B = T_C^{EE}(T_B^{EE})^{-1} \tag{6}$$

However, a single frame is not robust enough to get a good calibration. We move the robot around and capture multiple frames at each robot pose to get more accurate estimates We first run the semantic segmentation model and perform a sanity check on the output. If the number of EE points or the size of the EE bounding box are below their respective thresholds, we remove the frame. This is done to eliminate frames where the EE is not visible enough for accurate pose estimation.

After the sanity check, we predict the EE poses (Sect. 3.3 and 3.4) and the camera to robot base predictions (Eq. 6). We then group the predictions based on the robot configuration (recall that we capture multiple frames per robot pose). Per group, we run a modified Z-score outlier detection algorithm using the absolute deviations about the median [31] for each translation axis of the camera

Class	Precision	Recall	Accuracy
End-Effector	0.88	1.00	-
Arm	1.00	0.96	-
Background	1.00	1.00	-
Overall	0.96	0.99	0.99

Table 1: Semantic segmentation results

to robot base predictions. We cannot apply this outlier detection algorithm directly to individual elements of the rotation matrices or quaternions. In order to detect outliers amongst rotation predictions, we calculate rotational distances of every prediction to a reference unit quaternion. We run the outlier detection algorithm on these distances. We remove pose predictions that are marked as outliers in any of outlier detection runs (the three axes and the rotation). Then, remaining camera to base transformations are averaged by computing the arithmetic average for translation predictions and utilizing quaternion averaging [32] for rotation predictions. This gives us the individual calibration estimations for each robot configuration.

Finally, we employ the same outlier detection and averaging steps to the calibration estimates of each robot configuration to get the final estimate.

4 Experimental Results

We evaluate the semantic segmentation, single pose estimation and camera calibration performances of our system. We test our approach with ground truth segmentation labels as well to gauge the effects of the accuracy of the segmentation model on the pose and calibration estimation. We also provide results without the ICP step to highlight its benefits.

All of our experiments are conducted in real-life setup. Our hardware setup only includes a Kinect V1 and a Franka Emika Panda robot arm as shown in Fig. 1. For evaluation purposes, we collect a new test set from two previously unseen point of views (POV), using the steps described in Sect. 3.1. We put the robot arm in six different configurations for each POV and collect 10 frames in various lighting conditions. All the results reported in this section uses this test dataset.

We use the translation error (ϵ_t), rotational error (ϵ_R) and the average distance (ADD) metrics to measure the pose estimation and calibration performances. The ϵ_t is the Euclidean distance between the ground truth translation and predicted translation. The ϵ_R is the minimum rotation in degrees between the ground truth rotation and the predicted rotation. ADD is the average of point-to-point Euclidean distances between the EE points transformed with the ground truth pose and transformed with the predicted pose [33]. ADD is the acknowledged way of computing object pose prediction errors which considers both translation and rotation. To measure the accuracy of our semantic segmentation model, we use the standard accuracy, precision and recall metrics.

4.1 Semantic Segmentation

Semantic segmentation performance is important since it is the first step and its EE segmentation output is used as input to multiple the succeeding steps. We use the per-class precision and recall statistics along with the overall accuracy to assess the performance of segmentation model which is described in Sect. 3.2.

Table 1 presents the semantic segmentation results. Overall, the segmentation model performs well. The recall ratio for EE segmentation is $\sim 100\%$, i.e., the model is able to catch all the EE points. On the other hand, the table shows that some arm points (0.96 recall) are confused as EE points (0.88 precision) which is rather expected as EE is attached to the arm. Recall that we use linkage clustering to get rid of spurious segmentation results to improve down-stream performance.

Method	ϵ_t [cm]	ϵ_R [degree]	ADD [cm]
RPT	2.03 ± 0.46	7.49 ± 3.74	2.12 ± 0.52
RPT + ICP	0.55 ± 0.23	$\mathbf{3.28 \pm 1.04}$	0.62 ± 0.22
KPM	1.41 ± 0.14	5.55 ± 1.55	1.54 ± 0.16
KPM + ICP	$\mathbf{0.48 \pm 0.18}$	3.58 ± 0.78	$\mathbf{0.55 \pm 0.18}$
Robot Calibration	$\mathbf{1.18 \pm 0.43}$	$\mathbf{2.78 \pm 0.44}$	-

Table 2: End-effector pose estimation and calibration results with semantic segmentation predictions. Top 4 results are for EE poses wrt the camera while the last result is for the camera to robot base calibration.

Method	ϵ_t [cm]	ϵ_R [degree]	ADD [cm]
RPT	1.70 ± 0.34	5.78 ± 2.90	1.74 ± 0.34
RPT + ICP	0.50 ± 0.22	3.28 ± 1.19	0.60 ± 0.18
KPM	1.19 ± 0.22	5.51 ± 1.6	1.33 ± 0.21
KPM + ICP	0.50 ± 0.22	3.53 ± 1.23	0.60 ± 0.19
Robot Calibration	1.13 ± 0.17	2.77 ± 0.26	-

Table 3: End-effector pose estimation and calibration results with ground truth semantic segmentation. Top 4 results are for EE poses wrt the camera while the last result is for the camera to robot base calibration.

The pose estimation and extrinsic calibration performances are calculated with both the model outputs (Table 2) and ground truth segmentation labels (Table 3). The robot calibration rows show that there is around $0.05cm$ translation and $0.05°$ rotation difference between the average errors, with less variance on the ground truth end. These are negligible when the camera noise is taken into account [9]. **Combining this with the segmentation performance, we conclude that the segmentation model performs well enough to generate reliable inputs for the subsequent modules.**

4.2 Single Frame Pose Prediction and Multi-Frame Calibration

The first four rows of the Table 2 show the pose estimation performances of the RPT and KPM approaches with and without the ICP post-processing step with semantic segmentation predictions. The Table 3 shows the same with ground truth segmentation labels. Their last row shows the multi-frame calibration results.

ICP step provides considerable improvement for pose estimation: The addition of the ICP post-processing step improves both algorithms in all metrics in both segmentation input cases. In addition, the absolute performance is very impressive with about $0.5cm$ translation and $3°$ rotation errors.

KPM has a slight edge over RPT: Both methods perform more or less the same with the ICP step, KPM being slighly better in translation estimation and RPT in rotation. KPM outperforms RPT when we remove the ICP step.

Pose Estimation Comparison to Existing Work: It is difficult to compare the methods since the setups are significantly different. However, we are going to attempt comparison wherever there are similarities while conceding that our analysis is not apples to apples.

RPT and KPM approaches even without the ICP step outperform state-of-the-art object pose detection methods [12, 13] having $2.12cm$ and $1.54cm$ average ADDs, respectively. In other words, these modules predicts EE pose with 100% accuracy in the case that ADD threshold is chosen above approximately $2.12 + 0.52 = 2.64cm$ for RPT and $1.54 + 0.16 = 1.7cm$ for KPM as shown in Table 2. The performance gets much better when ICP step is included. Moreover, we outperform a similar single frame pose estimation method [17] whose maximum ADD rate is below 60% for

Kinect V1. However, the generic object pose detection methods deal with multiple objects [34] and Lee et al. [17]'s result is for the entire robot arm.

Valassakis et al. [19] report ϵ_t $(1.34cm)$ and ϵ_R $(4.4°)$ errors on their simulation data and ϵ_t $(1.04cm)$ on their real world data, for their eye-in-hand calibration method. Our ϵ_t and ϵ_R errors are lower; however there is a significant setup discrepancy.

High Performance Multi-Frame Calibration: The average translation and rotational error between the predicted calibration and the ground truth calibration are $\epsilon_t = 1.18cm$ and $\epsilon_R = 2.78°$, respectively. This absolute performance would allow for most manipulation applications outside of assembly or high precision tasks. These results are affected by the sensor noise, and potentially the intrinsic calibration and the initial extrinsic calibration errors. We argue that a more recent camera or an active marker system is needed to go beyond these values.

The average calibration translation error is higher than the best average translation error of single frame pose estimation. At a first glance this may seem contradictory. However, the errors are for different frames; the calibration error is for the camera-to-robot base transformation. Going from EE pose wrt camera to the base requires transformations (Eq. 6) which, due to rotations, increases the translation error.

5 Limitations and Future Work

Our system assumes that there are no objects in the camera view during calibration since we do not collect data for them. Our system uses a relatively high-capacity model, MinkUnet18D [23], for the semantic segmentation task involving only three classes. As such, it is prone to overfitting. An out of sample input (i.e. a point cloud with objects) would be difficult to handle. Using lower capacity models such as MinkUNet101 and MinkUNet14A deteriorated our semantic segmentation performance. Superimposing object point clouds as a data augmentation approach is possible in the future to circumvent this limitation. However, we argue that this is overall a mild assumption to handle extrinsic calibration prior to any application.

Currently, we provide the desired keypoints by hand. Even though it is enough to do this once for each EE, it is not the most ideal approach since the user may not be familiar enough with robotics to do so or the selected keypoints may not be easy to distinguish. An automated keypoint selection algorithm can be added in the future to remove this burden from the user.

We collect data with the robot arm in different joint configurations for the calibration operation. These joint configurations should be diverse for a better calibration estimate. The test set in this work manually selected the joint configurations. A fully automated system should chose these by itself while keeping the EE in the camera frame.

6 Conclusion

We presented a learning based extrinsic calibration system that does not require fiducial markers, additional hardware or simulation. Our system collects and labels its own data. The only manual intervention steps are the initial calibration for data collection and keypoint selection. Our system performs extrinsic calibration from multiple-frames and employs steps to increase its robustness. In addition to extrinsic calibration, our system outputs high quality arm and EE segmentation information and camera-to-EE pose estimation.

We tested our approach in different camera configurations then the ones used for training. Our results showed that the ICP algorithm significantly improves estimation performance and the absolute calibration results are close to the limit of what is possible with the used depth camera. The main limitations of our approach are the lack of object consideration, which is mild if the only aim is extrinsic calibration, and manual keypoint selection.

References

[1] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand-eye calibration. In *Proceedings of the 4th international symposium on Robotics Research*, pages 287–297, 1988.

[2] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[3] M. Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.

[4] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

[5] M. Hwang, B. Thananjeyan, S. Paradis, D. Seita, J. Ichnowski, D. Fer, T. Low, and K. Goldberg. Efficiently calibrating cable-driven surgical robots with rgbd fiducial sensing and recurrent neural networks. *IEEE Robotics and Automation Letters*, 5(4):5937–5944, 2020.

[6] Q. Zhan and X. Wang. Hand–eye calibration and positioning for a robot drilling system. *The International Journal of Advanced Manufacturing Technology*, 61(5):691–701, 2012.

[7] X. Liu, H. Madhusudanan, W. Chen, D. Li, J. Ge, C. Ru, and Y. Sun. Fast eye-in-hand 3-d scanner-robot calibration for low stitching errors. *IEEE Transactions on Industrial Electronics*, 68(9):8422–8432, 2020.

[8] O. Kroeger, J. Huegle, and C. A. Niebuhr. An automatic calibration approach for a multi-camera-robot system. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1515–1518. IEEE, 2019.

[9] O. Wasenmüller and D. Stricker. Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision*, pages 34–45. Springer, 2016.

[10] H. Peng, X. Yang, Y.-H. Su, and B. Hannaford. Real-time data driven precision estimator for raven-ii surgical robot end effector position. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 350–356. IEEE, 2020.

[11] K. Pauwels and D. Kragic. Integrated on-line robot-camera calibration and object pose estimation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2332–2339. IEEE, 2016.

[12] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 06 2018. doi:10.15607/RSS.2018.XIV.019.

[13] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[14] Y. You, Y. Lou, C. Li, Z. Cheng, L. Li, L. Ma, C. Lu, and W. Wang. Keypointnet: A large-scale 3d keypoint dataset aggregated from numerous human annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13647–13656, 2020.

[15] T. Jakab, R. Tucker, A. Makadia, J. Wu, N. Snavely, and A. Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12783–12792, 2021.

[16] J. Lambrecht. Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection. In *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)*, pages 136–141. IEEE, 2019.

[17] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield. Camera-to-robot pose estimation from a single image. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9426–9432. IEEE, 2020.

[18] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic. Single-view robot pose and joint angle estimation via render & compare. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1654–1663, 2021.

[19] E. Valassakis, K. Dreczkowski, and E. Johns. Learning eye-in-hand camera calibration from a single image. *CoRR*, abs/2111.01245, 2021. URL https://arxiv.org/abs/2111.01245.

[20] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345, 2015.

[21] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[22] H. Sankoh, A. Ishikawa, S. Naito, and S. Sakazawa. Robust background subtraction method based on 3d model projections with likelihood. In *2010 IEEE International Workshop on Multimedia Signal Processing*, pages 171–176. IEEE, 2010.

[23] C. Choy, J. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[24] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 806–814, 2015.

[25] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[26] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[27] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[28] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987. doi: 10.1109/TPAMI.1987.4767965.

[29] Q.-Y. Zhou, J. Park, and V. Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

[30] Franka Emika. Franka Emika ROS. https://github.com/frankaemika/libfranka, 2022. [Online; accessed 9-June-2022].

[31] B. Iglewicz and D. Hoaglin. *Volume 16: how to detect and handle outliers, The ASQC basic references in quality control: statistical techniques, Edward F. Mykytka*. PhD thesis, Ph. D., Editor, 1993.

[32] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007.

[33] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[34] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.