A Deep Learning-Based Hand-eye Calibration Approach using a Single Reference Point on a Robot Manipulator

Ozan Bahadir[1], Jan Paul Siebert[1] and Gerardo Aragon-Camarasa[1]

***Abstract*— We present a hand-eye calibration approach based on a deep learning-based regression architecture to find the transformation between the robot end-effector and an external camera. For this, we hypothesise that it is possible to track a single reference point in the robot's end-effector to estimate the hand-eye geometric transformation using a deep neural network and a 3D vision system. To explore this hypothesis, we design three experiments to study the different components of our proposed network architecture while solving isolated cases of the hand-eye calibration problem. Our experimental results using a simulated environment show that our proposed approach has less than 1 mm error for translation and less than 2.31 degrees error for orientation. We also carried out experiments for our third approach in two real robotic testbeds (a Universal Robot 3 and the Rethink Baxter robot). Our approach achieves 2 mm and 5.9 degrees, 4.53 mm and 9.2 degrees of errors for the Universal Robot UR3 and the Rethink Baxter robot.**

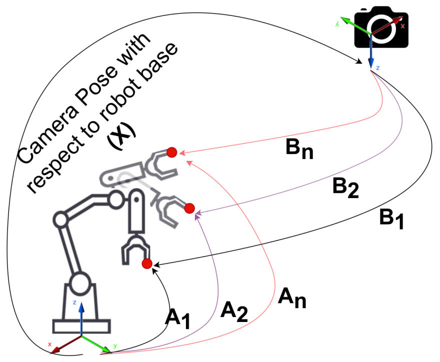

Fig. 1. The geometric representation of the proposed hand-eye calibration approach. The homogeneous transformation between the robot base and the camera is represented by $\mathbf{X}$. $\mathbf{A_{1....n}}$ represents the pose of the reference point with respect to the robot base, while $\mathbf{B_{1....n}}$ are the camera pose with respect to the reference point.

I. INTRODUCTION

With the advances in sensor and artificial intelligence technologies, robots have found new applications in aerospace, medicine and agriculture, to name a few. Robots sense their environment via different sensors, but vision is the most prominent sensor because of their low price and the ability to provide an understanding of the robot's environment. Vision sensors need to be integrated within the robot's kinematic frame, and this process is called hand-eye calibration. Hand-eye calibration involves finding the geometric transformation between the sensor and the robot to transfer camera measurements into the robot workspace. In the literature, the hand-eye calibration problem is solved by collecting several cameras poses and the corresponding robot's end-effector poses which are then passed to a hand-eye calibration algorithm that estimates the transformation between the camera and the robot [1].

Early approaches that have successfully solved the hand-eye calibration problem relied on offline data collection to acquire the required poses of the camera and robot's end-effector to perform hand-eye calibration. The main limitation of offline hand-eye calibration approaches is that they are not flexible against camera pose changes with respect to the robot base after obtaining the geometric transformation model between the camera and the robot's world coordinate frame. To overcome this problem, online hand-eye calibration approaches [2] [3] have been developed over the last decade, the most recent leveraging deep learning. Although current hand-eye calibration methods have significant success, a robust and flexible method that recalibrates the system without data recollection and is easily deployable to other robotic environments is needed.

In this paper, we propose that *it is possible to carry out hand-eye calibration by tracking the known transformation of a single reference point on the robot's end-effector with respect to both the robot base and the camera via a 3D vision system*. For this, we devise three experiments that explore the hand-eye calibration problem (Fig. 1). We implement our experiments in a simulated environment; while we test our final neural network architecture in two real-world robotic testbeds (a Universal Robot and a Rethink Baxter, as shown in Fig. 2). In this paper, the camera is not attached to the robot but is located at a given pose in the environment, as shown in Fig. 1. Our contributions are threefold:

1) We demonstrate that it is possible to estimate the camera pose by tracking a single reference point defined by the robot's kinematic chain and automatically detecting this reference point using a 3D vision system.
2) We show that hand-eye calibration with single RGB and depth images has more competitive accuracy for position and orientation errors than classical hand-eye calibration methods and the state-of-the-art.
3) We show that it is possible to carry out hand-eye calibration without explicit camera extrinsic calibration (i.e. camera pose).

[1] School of Computing Science, University of Glasgow, G12 8QQ, Scotland, United Kingdom; {Ozan.Bahadir}, {Paul.Siebert}, {Gerardo.AragonCamarasa}@glasgow.ac.uk

Ozan Bahadir thanks the funding received by the Turkish Ministry of National Education for this research

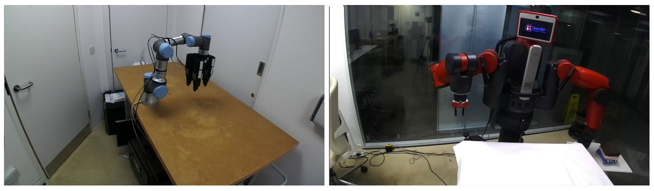
(a) The Universal Robot 3 (b) The Rethink Baxter Robot

Fig. 2. (a) and (b) depict the two real-world robotic testbeds used in this paper, as observed from the camera view.

II. RELATED WORK

By having knowledge of the end-effector poses and the corresponding camera poses, either a mathematical model or deep learning model is needed to find the geometric transformation between the robot's end-effector and the camera. $AX = XB$ is the simplest mathematical formulation for hand-eye calibration, where X is the hand-eye geometric transformation, while A and B represent the relationship between two or more successive end-effector poses and the camera pose, respectively. This formulation can be decomposed into orientation and position and solved separately or simultaneously. Tsai [4] proposed a closed-form solution to separately estimate the orientation and position. However, Tsai's approach has introduced hard constraints, such as the upper bound for the distance between the camera and the robot centre. It also suffers from singularities for 0 and π radians. Hence, Chou and Kamal [5] used quaternions to address this singularity problem. On the other hand, [6] and [7] proposed simultaneous approaches to solve $AX = XB$.

$AX = XB$ approaches are highly affected by the quality of the camera and robot calibrations. To solve this problem, $AX = YB$ has emerged as a new mathematical representation for hand-eye calibration. In this formulation, A and B represent the pose of the robot's end-effector and the pose of the camera, respectively; while Y is the geometric transformation between the robot base and the world coordinate base, and X is the hand-eye transformation. Zhuang *et. al* [8] and Hirsh *et. al*[9] proposed to separately get orientation and position components of the transformation between the robot and the camera for $AX = YB$ formulation. Li *et. al* [10] propose a Kronecker product-based approach to simultaneously calculate both the translation and orientation components of X and Y. They also reveal that the Kronecker product has better performance than quaternion and dual-quaternion approaches. Zhao [11] also proposed a linear programming solution to simultaneously obtain components of unknown transformations of X and Y.

Another model for hand-eye calibration is to utilise the 3D to 2D points projection error. In this formulation, differences between the estimated pixel locations of n keypoints and their actual pixel location in m different frames are used as an objective function for hand-eye calibration. Andreff *et al* [12] proposed an approach that consisted of estimating the camera pose via Structure-from-Motion (SfM). Haller *et al* [13] extended [12] and estimated the position and orientation components of the unknown hand-eye transformation between the camera and the robot. Zhi and Schwertfeger [14] propose an iterative projection-based approach for hand-eye calibration.

In contrast to classic feature detection methods, deep learning-based keypoint detection approaches have gained attention over the last decade. In [3] and [15], deep learning architectures were proposed to detect 2D key points on images. Then they employed a Perspective-N-Point (PnP) algorithm [16] to get the camera pose with respect to the robot base. To the best of our knowledge, only one approach [17] directly employs deep learning to solve hand-eye calibration. In contrast to our approach, they considered the eye-in-hand configuration: in other words, they attached the camera to the robot's end-effector. Although they employed three architectures for hand-eye calibration, they presented that a direct regression model has better performance than others. However, they only used images to regress the camera pose with respect to the robot's end-effector, while ignoring the corresponding end-effector poses. The camera pose could be acquired from images as in our approach, but the corresponding end-effector poses should also be considered as noted in Section II.

To the best of our knowledge, only one approach [17] directly employs deep learning to solve hand-eye calibration. In contrast to our approach, they considered the eye-in-hand configuration: in other words, they attached the camera to the robot's end-effector. Although they employed three architectures for hand-eye calibration, they presented that a direct regression model has better performance than others. However, they only used images to regress the camera pose with respect to the robot's end-effector, while ignoring the corresponding end-effector poses. The camera pose could be acquired from images as in our approach, but the corresponding end-effector poses should also be considered as noted in Section II.

III. METHODOLOGY

Fig 1 shows the hand-eye calibration formulation in this paper. A, B, and X are the pose of the reference point with respect to the robot base, the pose of the camera with respect to the reference point, and the geometric transformation between the camera and the robot base, respectively. The poses of the end-effector or reference points and the corresponding poses of the camera in consecutive frames should be known for hand-eye calibration. Note that the reference point we used in this paper is defined as the last reference frame in the robot's end-effector that is connected to the robot's kinematic chain. Hence, we proposed three main research questions that test the usability of deep learning-based approaches as a solution to hand-eye calibration.

Q1 In contrast to closed-form hand-eye approaches, is it possible to find the geometric transformation between the camera and the robot where camera and robot calibrations are known in advance by using a neural network?

Q2 Is it possible to find the camera's pose with respect to the reference point by observing the motions of this reference point via a 3D vision system and using deep learning as a calibration model?

Q3 Is it possible to carry out hand-eye calibration where camera calibration is not known by observing the motions of a reference point via a 3D vision system and employing a deep learning-based regression architecture?

We conduct experiments in a simulation environment for all these research questions. For **Q3**, we also conduct experiments in the real world because it is not possible to get ground truth for **Q1** and **Q2** in the real world. In **Q1**, we aim to test the feasibility of a deep learning-based approach as a hand-eye calibration model. In this experiment, it is assumed that the robot calibration and the camera calibration are known. We thus design a deep learning-based regression architecture that takes as input the known transformation of the reference point (**A**) and the camera pose (**B**), and it outputs the unknown hand-eye transformation (**X**).

In **Q2**, we investigate the applicability of a deep learning-based regression model for camera calibration. In this experiment, the extrinsic camera parameters are unknown, as opposed to the **Q1** configuration. We thus estimate these extrinsic parameters with respect to the reference point by tracking the reference point using a 3D vision system. We also extend the regression model in **Q1** to input the pose (**A**) of the reference point with respect to the robot's world frame and the corresponding RGB and depth images. This architecture outputs the pose (**B**) of the camera with respect to the robot's world reference frame.

Finally, in **Q3**, we aim to directly carry out hand-eye calibration by using a deep learning-based regression architecture. In this configuration, the robot calibration and the camera's intrinsic parameters are assumed to be known, while the camera pose and the geometric transformation between the camera and the robot base are unknown. Poses of the reference point (**A**) in n frames and the corresponding RGB and depth images are used as inputs to a DL-based regression architecture that estimates the unknown geometric transformation (**X**). Fig. 2 shows the the experimental setup and network architecture for **Q3**.

IV. SIMULATION EXPERIMENTS

We conducted experiments for all research questions in Section III in a simulation environment. Data generation, designed deep learning-based regression architectures and experimental results are discussed in the following subsections.

A. Data Generation

A virtual Universal Robot 10 (UR10) and an RGBD camera are deployed in the PyBullet simulation environment. Figure 3(a) shows our data generation approach. We place this camera in over 106 locations in the PyBullet environment, and the robot end-effector is moved to 50 different configurations for each camera pose. For **Q1**, the reference point poses with respect to the robot base (**A**) and the camera base (**B**), and the homogeneous transformation between the robot base and the camera (**X**) are recorded for each end-effector configuration. For **Q2**, we capture RGB and depth images, and the reference point poses with respect to the robot base (**A**), and the homogeneous transformation (**B**) between the camera and the reference point (camera's extrinsic parameters with respect to the reference point) for each end-effector configuration. Finally, we capture RGB and depth images for each end-effector configuration as well as the geometric transformations of **A** and **X** for **Q3**. We randomly select 21 camera poses for the test set in 106 camera configurations, which means we have $21x50$ unseen data points.

In our data, **X**, **A**, and **B** consist of a seven-element array where the first three elements represent the position, and the last four elements are the orientation parameters (unit quaternion) of the transformation. Each RGB image are $16-bit$ with a resolution of $1024x1024$ pixels, while each depth image is $16-bit$ with a resolution of $1024x1024$ pixels.

B. Network Architectures

We design a deep learning-based regression architecture for **Q1** which outputs the geometric transformation between the robot base and the camera. We train our architecture separately to estimate this transformation's position and orientation components since the solution spaces of these components are in different units. The designed architecture for **Q1** consists of three fully connected layers with a ReLU activation function. We use euclidian and quaternion spaces for position and orientation, respectively. This network starts with 512 neurons and systematically halves each layer except the last layer, which is three and ten for position and orientation, respectively. For position estimation, the last three neurons directly represent the camera's position with respect to the robot base in 3D cartesian space. However, for orientation estimation, the last ten neurons are converted 4x4 symmetric matrix $\mathbf{S}(\theta)$ and passed the Quadratically Constrained Quadratic Program-based model presented [18] to produce the unit quaternion representing the camera's orientation in 3D cartesian space. The designed architecture was trained end-to-end for 50 epochs using Adam optimiser[19], with a learning rate and batch size of $1e-3$ and 100, respectively, for the position. We used 100 epochs for orientation without changing other parameters because orientation learning is slower than the position.

A deep learning-based regression architecture was also developed for **Q2** and **Q3**. As opposed to the **Q1** architecture, our new network takes as inputs the RGB and depth images (*256x256* dimension) as well as the poses of the reference point with respect to the robot base. For this architecture, we were inspired by the U-Net architecture [20], and we tailored its encoder part to extract features from RGB and depth images separately (as shown in 3(b)). The encoders consist of two double Conv2D-3x3 layers (stride set to 1 and padding 0), followed by a Maxpool-2x2 layer with ReLU activation functions. Then, we concatenate encoded features for RGB and depth images with the known transformation **A**. Finally, we employ a fully connected network with three

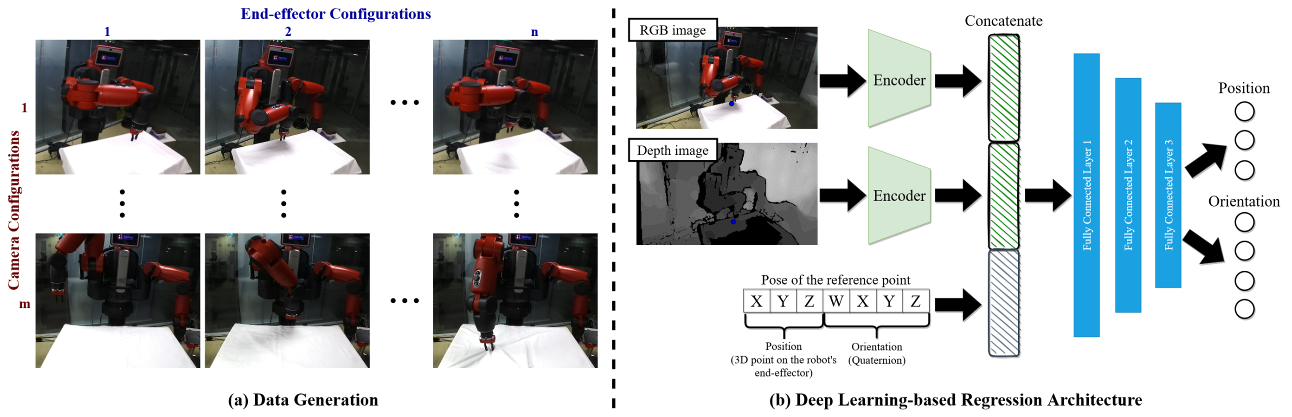

Fig. 3. This figure shows our experimental design for **Q2** and **Q3**. (a) our data collection process is shown where the camera is placed at m different locations, and n different end-effector configurations are run for each camera pose. (b) we show our deep learning-based architecture. We separately extract features from the RGB and depth images by using U-Net type encoders. Then, we concatenated these features with the known transformation of the reference point with respect to the robot base (**A**). We trained our network separately for position and orientation estimation because their solutions are in different metric spaces. This network outputs either the translation or orientation component of the unknown geometric transformations (**B** for **Q2** and **X** for **Q3**).

TABLE I

EXPERIMENTAL RESULTS IN SIMULATION

	Position Error (mm)		Orientation Error (degree)	
	Mean	**1 Std**	**Mean**	**1 Std**
Q1	0.415	0.016	2.31	0.85
Q2	5.44	0.343	10.52	2.05
Q3	0.296	0.086	5.84	1.23

hidden layers to estimate either the translation or orientation component of the unknown transformation **B** and **X** for **Q2** and **Q3**, respectively. We trained our network separately for position and orientation because of differences in their solution spaces. For **Q2** and **Q3**, we trained our network for 20 and 100 epochs by using Adam optimiser[19] for position and orientation, respectively. We also used 4 mini-batches and $1e-3$ learning rate for both of them.

C. Loss Function and Metric

We used the Mean Square Error (MSE) to train our models for the position. After training, we used the Root Mean Square Error (RMSE) metric to obtain the accuracy of the solution in the original units (meters). After calculating the RMSE for positions, we converted them to millimetres to interpret them. For orientation, we employed the chordal loss function presented in [18]. To interpret the orientation parameters, we converted them to degrees.

D. Experimental Results

We trained our network three times for each research question with parameters mentioned in Section IV-B. Table I shows the mean and standard deviation of all research questions' position and orientation results. We directly compared **Q1** and **Q3** because they are hand-eye calibration solutions. However, direct comparison of **Q2** with **Q1** and **Q3** is not possible because **Q2** is about the camera's extrinsic parameters only.

Our approach has 0.42 and 0.29 mm positional errors for **Q1** and **Q3**. This means that by using RGB and depth images as inputs instead of processing only geometric transformations of the reference point has resulted in better results. However, using images caused a higher standard deviation (0.016 mm for **Q1** and 0.086 mm for **Q3**). As for **Q2**, the position error is 5.44 mm (standard deviation, 0.343 mm) and orientation error is 10.52 degrees (standard deviation 2.05 degrees). Although these results are competitive, using multiple reference points instead of one such as in our approach can increase the success rate of explicit camera pose estimation. However, the estimating the camera extrinsic parameters is outside of the scope of this paper as we only consider the hand-eye calibration problem. Future work would comprise exploring the effects of multiple reference points to gather further evidence to support **Q2**.

V. REAL-WORLD EXPERIMENTS

A. Data Generation

a) Universal Robot (UR3) robotic testbed: We equipped the UR3 robot with a Shadow Modular Grasper [21] with three fingers and three joints for each finger (total, 9 degrees of freedom) and selected the last link of one of these fingers as a reference point. The UR3 has a repeatability of 0.1 mm. Then, we placed a Stereolab's ZED camera [22] in 24 different locations around the robot's workspace. We used the Robot Operating System (ROS)[23] to control the robot and connect the camera. We employed the Tsai hand-eye calibration method [4] for each camera pose to get a baseline for the geometric transformation between the robot base and the camera for each of the 24 different camera locations (19 for training and 5 for testing). We carried out Tsai's method[4] five times for each camera configuration, and we used the best calibration result as the input to our regression neural network. The rationale for repeating the calibration five times is that Tsai's calibration depends on the quality of pose sequences. Table II shows the average of the best of these five hand-eye calibration attempts for the UR3 via [4]. Then, we allowed the robot to plan for 100 different

TABLE II
AVERAGE MEAN AND STD OF m CAMERA CONFIGURATION FOR TSAI HAND-EYE CALIBRATION METHOD (GROUND-TRUTH) ON TWO ROBOTIC TESTBEDS

Baxter		UR3	
Mean	**1 Std**	**Mean**	**1 Std**
2.411380316	0.9667983579	1.039966529	0.4237020321

robot end-effector poses for each camera location, and we captured RGB and depth images using the camera while we recorded the poses of the reference point with respect to the robot base using the ROS the Transform Library (TF)[24] library. Overall, we captured 2400 RGB and depth images of *1920×1080* image resolution.

b) The Rethink Baxter robotic testbed: The Baxter robot with electric parallel grippers was used as a second robotic testbed. Baxter has a repeatability in the left and right arm of 2.9 mm and 3.3 mm, respectively [25]. As it can be observed, the UR3 has better repeatability than Baxter. This repeatability difference allows us to quantitatively measure whether our approach can accommodate tear and wear as Baxter compliant actuation system inherently introduces errors in the robot's end-effector positioning.

The Seterolab's ZED camera was placed in 19 configurations (14 for training and 5 for testing), and Tsai hand-eye calibration was employed to get a baseline for each camera pose. As with the UR3, we carried out Tsai's hand-eye calibration five times for each camera configuration and used the best calibration as the input to the neural network. Table II shows the average of the best of these five hand-eye calibration attempts for the Baxter via [26]. The right finger of the right gripper in Baxter was selected as the reference point for **Q3**. Similar to the UR3, we employed the ROS and TF library to control the robot and get the reference point transformations. For each camera configuration, we allowed the end-effector on the right arm of the Baxter robot to move to 85 different poses for each camera location. We also capture the RGB and depth images and record the geometric transformation between the robot base and the reference point. Overall, we collected 1,615 RGB and depth images with an image resolution of *1920×1080* pixels.

B. Network Architecture

We develop a deep learning-based regression architecture for **Q3**. This architecture is similar to the designed architecture (Fig.3(b)) for **Q3** in the simulation environment (Section IV-B). However, we increase the double Conv2D-3x3 layers (stride set to 1 and padding, 0) from two to five because of the real-world complexity. The shallow network had remarkable success in the simulation environment, and during the development of this architecture, we found that we needed a deeper network for these experiments. We train our network for 20 epochs using the Adam optimiser with four mini-batch and a learning rate of $1e-4$ for position and orientation parameters estimation. We train our network three times with the same seed to account for the variation coming from the optimiser. We record all training repetitions in our results.

TABLE III
EXPERIMENTAL RESULTS FOR BAXTER AND UR3 ROBOTS

	Position Error (mm)		Orientation Error (degree)	
	Mean	**1 Std**	**Mean**	**1 Std**
Baxter	4.543	0.081	9.2	1.35
UR3	2.07	0.331	5.8	1.02

C. Real-world Results

Table III summarises the position and orientation results for **Q3** for the Baxter and UR3 robotic testbeds. As stated in Section V-B, we employed Tsai's hand-eye calibration to obtain a baseline and as the input to our neural network. For the UR3 robotic environment, our approach has 2.07 mm and 5.8 degrees of error in the test set (i.e. unseen camera and robot's end-effector poses) for translation and orientation, respectively. For Baxter, our approach obtained 4.54 mm and 9.2 degrees of error in the test set for translation and orientation, respectively. As expected, the difference between the UR3 and Baxter calibration results is due to Baxter being inherently inaccurate; however, our approach is able to only double the error and still remain below similar approaches reported in the literature (see Table IV).

We employ the indirect error function presented and used in [4], [17] to make a fair comparison with the literature because the results are highly dependent on the experimental setup, the robot and the camera. However, this error function shows the repeatability of a hand-eye calibration approach which is independent to the robotic environment. That is, the indirect error function is given in Eq. V-C.

$$\epsilon_{std} = \frac{1}{k}\sum_{k=1}^{K}\frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{n}\sum_{i=1}^{n}\|\overline{x}_{kji} - \mu_{kj}\|_2^2\right)^{\frac{1}{2}} \qquad (4)$$

where k, m, and n are the neural network training repetitions (see Section V-B), the total number of unseen camera configurations and the total number of end-effector configurations for each camera configuration.$\overline{x}_{kji}$ represents the predicted position value for kth repetition, jth camera configuration, and ith end-effector configuration. Moreover, μ_{kj} is the positional mean of the kth repetition and jth camera configuration.

Table IV shows the comparison between our baseline, the state-of-the-art and our approach. The first two rows summarise the best Tsai's hand-eye calibrations in our experiments for the Baxter and UR3 robots. We run five times Tsai's hand-eye calibration method for each camera configuration. In comparison with the state-of-the-art deep learning approaches, our approach has less variance and is close to the best Tsai's calibration we obtained in our experiments. That is, Lee et al. [3] and [17] report an indirect error of 27.4 mm and 10.4 mm with respect to their sample mean, respectively. This represents a difference of 23.45 mm and 6.45 mm with respect to our approach in the UR3. We note that [3] and [17] do not use a reference point that is aligned to the robot's kinematic chain but, instead,

TABLE IV
COMPARISON WITH THE STATE-OF-THE-ART

Method	**Position**($\epsilon_{\mathtt{std(mm)}}$)
Tsai-Baxter	(22.41±29.96)
Tsai-UR3	(15.5±19.24)
Keypoint+PnP lee2020camera	(27.4±4.7)
Direct Regression valassakis2022learning	(10.4±4.0)
Our approach-Baxter	(9.66±1.58)
Our approach-UR3	(3.95±1.25)

they relied on the captured images to infer the hand-eye calibration parameters. In our approach, we explicitly defined a reference point from which a neural network can create an embedding that takes into account the robot's end-effector and the kinematic chain (i.e. robot calibration in Sec. II).

VI. CONCLUSIONS

We have described a deep learning-based hand-eye calibration approach that enabled us to find the unknown geometric transformation between the robot base and an external camera without data recollection after training the network. In our experiments, the motion of a single point (reference point) selected on the robot's end-effector were tracked via a 3D vision system. We also proposed three research questions (Section III). For **Q1**, we showed that a deep learning-based regression model can be used as a hand-eye calibration model instead of mathematical models (e.g. [4]). In this approach, we assumed that the robot and camera calibration are known, and we directly estimated the unknown transformation via a deep learning-based regression model. For **Q2**, camera calibration is unknown; hence, we tailored this regression approach by adding two encoders for RGB and depth images to find the camera's extrinsic parameters. Finally, we employed this network architecture to estimate the unknown transformation between the camera and the robot base in **Q3**. We conducted experiments in a simulation environment for all these research questions, and our experimental results reveal that a deep learning model can be used as a hand-eye calibration. To further validate our approach, we also performed experiments for **Q3** on two real robotic environments (a Universal Robot 3 and the Rethink Baxter robot), where we obtained an overall position error of 2.07 and 4.54 mm, and orientation error of 5.8 and 9.2 degrees, respectively. These results show that our approach is robust to camera pose changes while only having a repeatability error of 3.95% and 9.6% for unseen camera poses (e.g. camera poses that were not used for training our network).

REFERENCES

[1] J. Jiang, X. Luo, Q. Luo, L. Qiao, and M. Li, "An overview of hand-eye calibration," *The International Journal of Advanced Manufacturing Technology*, pp. 1–21, 2021.

[2] 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)*. IEEE, 2016, pp. 2332–2339.

[3] 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)*. IEEE, 2020, pp. 9426–9432.

[4] R. Y. Tsai, R. K. Lenz *et al.*, "A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[5] J. C. Chou and M. Kamel, "Finding the position and orientation of a sensor on a robot manipulator using quaternions," *The international journal of robotics research*, vol. 10, no. 3, pp. 240–254, 1991.

[6] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[7] R. Horaud and F. Dornaika, "Hand-eye calibration," *The international journal of robotics research*, vol. 14, no. 3, pp. 195–210, 1995.

[8] H. Zhuang, Z. S. Roth, and R. Sudhakar, "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form ax= yb," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 549–554, 1994.

[9] R. L. Hirsh, G. N. DeSouza, and A. C. Kak, "An iterative approach to the hand-eye and base-world calibration problem," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 3. IEEE, 2001, pp. 2171–2176.

[10] A. Li, L. Wang, and D. Wu, "Simultaneous robot-world and hand-eye calibration using dual-quaternions and kronecker product," *International Journal of Physical Sciences*, vol. 5, no. 10, pp. 1530–1536, 2010.

[11] Z. Zhao, "Simultaneous robot-world and hand-eye calibration by the alternative linear programming," *Pattern Recognition Letters*, vol. 127, pp. 174–180, 2019.

[12] N. Andreff, R. Horaud, and B. Espiau, "Robot hand-eye calibration using structure-from-motion," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 228–248, 2001.

[13] J. Heller, M. Havlena, A. Sugimoto, and T. Pajdla, "Structure-from-motion based hand-eye calibration using l minimization," in *CVPR 2011*. IEEE, 2011, pp. 3497–3503.

[14] X. Zhi and S. Schwertfeger, "Simultaneous hand-eye calibration and reconstruction," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1470–1477.

[15] 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)*. IEEE, 2019, pp. 136–141.

[16] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[17] E. Valassakis, K. Dreczkowski, and E. Johns, "Learning eye-in-hand camera calibration from a single image," in *Conference on Robot Learning*. PMLR, 2022, pp. 1336–1346.

[18] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly, "A smooth representation of belief over so (3) for deep rotation learning with uncertainty," *arXiv preprint arXiv:2006.01031*, 2020.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] 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*. Springer, 2015, pp. 234–241.

[21] S. R. Company, "Modular grasper," https://modular-grasper.readthedocs.io/en/latest/user_guide/1_introduction/, 2022, last accessed 10 February 2022.

[22] S. Inc., "Zed," https://www.stereolabs.com/zed, 2022, last accessed 10 February 2022.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[24] T. Foote, "tf: The transform library," in *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013, pp. 1–6.

[25] K. S. Chen, "Application of the iso 9283 standard to test repeatability of the baxter robot," 2015.

[26] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.