Unified Data Collection for Visual-Inertial Calibration via Deep Reinforcement Learning

Yunke Ao*, Le Chen*, Florian Tschopp, Michel Breyer, Andrei Cramariuc, Roland Siegwart

Abstract—Visual-inertial sensors have a wide range of applications in robotics. However, good performance often requires different sophisticated motion routines to accurately calibrate camera intrinsics and inter-sensor extrinsics. This work presents a novel formulation to learn a motion policy to be executed on a robot arm for automatic data collection for calibrating intrinsics and extrinsics jointly. Our approach models the calibration process compactly using model-free deep reinforcement learning to derive a policy that guides the motions of a robotic arm holding the sensor to efficiently collect measurements that can be used for both camera intrinsic calibration and camera-IMU extrinsic calibration. Given the current pose and collected measurements, the learned policy generates the subsequent transformation that optimizes sensor calibration accuracy. The evaluations in simulation and on a real robotic system show that our learned policy generates favorable motion trajectories and collects enough measurements efficiently that yield the desired intrinsics and extrinsics with short path lengths. In simulation we are able to perform calibrations $10 \times$ faster than hand-crafted policies, which transfers to a real-world speed up of $3 \times$ over a human expert.

I. INTRODUCTION

In recent years, visual-inertial (VI) sensors, which include one or more cameras and an inertial measurement unit (IMU), have become a key component of many autonomous mobile robot systems since they can provide precise motion estimates and are well suited for use in various high-level robotic tasks [1], [2], [3], [4], [5]. Precise calibration, which for VI sensors refers to the parameters for the camera intrinsics, the camera-IMU extrinsics, and the time offset between the different sensors, are of great importance to the accuracy and performance of VI systems [6], [7], [8]. However, it is usually non-trivial to perform the calibration by hand or with a manually pre-programmed operator, since it often requires complex motion routines ensuring observability in controlled environments [8], [9], [10], [11]. In order to guarantee high calibration accuracy, typically large amounts of data are recorded for optimization. But the calibration time for the trajectory quickly grows as more measurements are collected, which makes calibration a quite time-consuming process [12]. In addition, given the current pose and calibration data collection state, the movement that renders the best calibration results effectively and efficiently for the next step is not obvious.

In early studies, several works applied reinforcement learning (RL) to get the best sequence of trajectories for VI calibration. Nobre et al. [9] proposed using Q-learning to select a sequence of motions from a predefined library aiming to render sufficient observability of the calibration problem. However, this approach only suggests which predefined motions to choose from the empirically designed library without exploring new possible motion primitives. To look into new possible trajectories, Chen et al. [13] modeled the calibration process as a Markov decision process (MDP) and proposed a model-based deep RL calibration framework with an adapted version of particle swarm optimization (PSO) [14] for sampling to generate different trajectories for intrinsic and extrinsic calibration. However, the MDP in this method suffers from high dimensionality and the learned open-loop trajectories do not adapt to the feedback of the current state.

To deal with the shortcomings of current systems (see Section II), we model calibration as an MDP in a compact way and use model-free deep RL [15] to learn a policy that guides the movements of the VI sensors. A robot arm then executes the motions. According to the current pose and collected measurements, the policy generates the next transformation that optimizes calibration accuracy and efficiency. Additionally, we simulate different VI sensor configurations including symmetric and asymmetric distortions to model the variability that also exists in the real world, enabling better transferability to real scenarios. Compared with other solutions, the action space of our model has fewer constraints, making it possible to learn a more general policy for calibration. Moreover, our method obtains only one policy that can be used both for intrinsic and extrinsic calibration. The main contributions of this work are as follows:

- To the best of our knowledge, we are the first to use model-free deep RL to learn a policy that generates trajectories for efficiently collecting calibration measurements.
- We propose a novel formulation to jointly calibrate both camera intrinsic and VI extrinsic parameters with only one trajectory.
- The simulated evaluation shows that the trajectories generated by the learned policy deliver more accurate and efficient calibrations compared with other trajectories. The real experiments confirm the transferability of our approach to real scenarios.

II. RELATED WORKS

Calibrating a VI system can be divided into two parts: camera calibration and camera-IMU calibration. The goal of camera calibration is to determine the camera intrinsics and the distortion model coefficients [16]. Camera-IMU
calibration implies obtaining the camera-IMU extrinsics and the time offset between different sensors [6], [7]. The classical method for camera calibration is to construct constraints using a known calibration pattern and formulate an optimization problem to determine the parameters [17], [18]. For camera-IMU extrinsic calibration, the most popular and reliable methods also require a calibration board. Furgale et al. [10], [11] proposed to parameterize the pose and IMU bias trajectories using B-splines and applied continuous-time batch estimation to jointly calibrate the time offsets and spatial transformations between multiple sensors. Those methods are able to retrieve the calibration parameters of the sensors with good accuracy. However, the optimal trajectories for measurement collection in terms of efficiency, observability, and accuracy, remain unknown. Calibration still requires expert knowledge to collect enough effective data and is often a time-consuming process for inexperienced people.

In recent years, automatic calibration of VI sensors has become an active research topic in robotics communities. The approaches for automatic calibration can be divided into two categories: trajectory optimization based and RL based. Most of the trajectory optimization based methods focus on optimizing the observability Gramian of the trajectories on the calibration parameters [19], [20]. However, these methods ignore some practical requirements of calibration, such as minimizing path length, because they are difficult to model.

RL based methods can better include those general requirements. Nobre et al. [9] defined a library containing different motions and applied Q-learning to select a sequence of motions from the library with the goal to obtain enough observability for calibration. However, the performance, as well as efficiency of calibration, is constrained by the predefined library because many possible motion sequences are not included or explored.

Chen et al. [13] addressed this problem by learning continuous parameters of calibration trajectories using a model-based RL algorithm with adapted PSO. This work trained two networks, one for intrinsic and one for extrinsic calibration. They define the action at each time as a looped trajectory which is described by 36 parameters. The state is defined as the concatenation of all actions and calibration results of previous time steps. Its number of dimensions thus grows with increasing time steps, resulting in a high dimensional MDP that is difficult to solve. In addition, the proposed method learns open-loop trajectories which do not adapt to the feedback of the current state.

In contrast, our proposed approach models the calibration process in a compact way. We define an action as the transformation to the next pose, which only requires 6 dimensions. And the state, consisting of the information status and the current pose, now has a fixed number of dimensions. Thanks to the compactness and efficiency of the proposed MDP, we are able to apply a model-free algorithm [15] to learn policies. Moreover, our current approach includes symmetric as well as asymmetric distortion in the experiment and trains only one network which can be used for both intrinsic and extrinsic calibration.

III. METHOD

Our goal is to generate a sequence of direct transformations that form a trajectory for the end-effector with the VI-sensor fixed on it, based on which accurate intrinsic and extrinsic parameters can be calibrated using the acquired sensor data. We solve this problem by learning a policy for generating the step-by-step transformation based on feedback information at each step. The whole proposed learning framework is shown in Figure 1 including the simulator, calibrator, MDP embedding module, and the RL agent.

A. Visual-Inertial Calibration

In our method, both camera intrinsic and camera-IMU extrinsic calibration are performed using the Kalibr [10], [11] framework. In Kalibr, calibration is formalized as a large-scale optimization problem, using the Levenberg-Marquardt algorithm, to minimize the error between the predicted and obtained measurements. Other outputs from the framework also provide useful information for estimating the calibration quality, e.g. reprojection error distribution, motion bias curve,
Fig. 2: Illustration of image coverage. The left image shows the division of the 4 regions of the image view and their corresponding vertices. The right part shows an example of propagation of image coverage with increasing steps. The orange polygon defined by the four yellow vertices always covers the positions swept by the target board. The red arrows show the approximated movement of the target board in the image view during each step.

e.g. In each episode of interaction for the agent, we only call Kalibr once at the last step if the data acquisition status fulfills our predefined requirements, in order to reduce the computational cost for training.

B. MDP Model for Calibration

a) Action Definition: As the whole trajectory consists of a sequence of direct transformations, we define the action of each step as a translation and rotation of the pose of the end-effector $A = [p, \theta, \phi, \alpha, \beta, \gamma]$. Here $p$ is the length of translation, $\theta$, and $\phi$ are azimuth and polar angles indicating the direction of translation. $\alpha$, $\beta$, and $\gamma$ are Euler angles that determine the rotation. We also set constraints on $p$, $\alpha$, $\beta$, and $\gamma$ to guarantee feasible motions.

b) State Definition: Regarding the problem formulation of calibration, the final calibration results depend on all the data acquired from the whole trajectory. In [13], all the trajectories and calibration outputs of the previous steps are used as a description of the status of the data acquired so far. To simplify the modeling and reduce the difficulty of learning, we only use a fixed number of variables that define the range of the previously acquired data as our state, rather than using the whole history.

For image data, we use the coverage of the center position, change in size, and skew of the target board in the image view to indicate the calibration progress. For the coverage of the target board center position, instead of only recording the maximum and minimum value of X and Y coordinates of the center position, we use a polygon with four vertices to describe the 2D range in a more specific way. As is shown in Figure 2, we divide the image equally into four parts: left-up, right-up, right-down, and left-down, denoted by $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{R}_3$, and $\mathcal{R}_4$ respectively. In each region $\mathcal{R}_i$, we place one vertex $V_i$ that has to stay inside the respective region. Given all the positions that the target board has covered $[u_j, v_j]_{j=1}^N$, the update rules for the vertices at each step are:

$$V_1 = \min_{[u_j, v_j] \in \mathcal{R}_1} u_j, \min_{[u_j, v_j] \in \mathcal{R}_1} v_j$$
$$V_2 = \max_{[u_j, v_j] \in \mathcal{R}_2} u_j, \max_{[u_j, v_j] \in \mathcal{R}_2} v_j$$
$$V_3 = \min_{[u_j, v_j] \in \mathcal{R}_3} u_j, \max_{[u_j, v_j] \in \mathcal{R}_3} v_j$$
$$V_4 = \max_{[u_j, v_j] \in \mathcal{R}_4} u_j, \min_{[u_j, v_j] \in \mathcal{R}_4} v_j$$

The range of the target board sizes is encoded by the minimum and maximum proportion of the area of the target board in the image view, denoted by $A_{min}$ and $A_{max}$. For skew coverage, we use the range of angles between two fixed edges of the target board in the image view $\eta_{min}$ and $\eta_{max}$.

To encode the range of motion, we use the maximum differences between the current and the next poses in the previous steps $\Delta p_{max}, \Delta \Theta_{max}$, where $p$ and $\Theta$ denote the position and orientation of the sensor, respectively. This corresponds to the span of acquired IMU sensor data because larger differences will result in larger velocities and accelerations for direct transformations with the default controller. For orientation, we approximate the difference as the difference of Euler angles. Note that the coverages are all normalized between 0 and 1 to benefit training and terminal condition design. In addition to the range of acquired data, we also include the current pose of the end-effector $p, \Theta$ in the state. In this way, the full state of data acquisition is defined as:

$$S_t = [V_1^t, V_2^t, V_3^t, V_4^t, A_{min}^t, A_{max}^t, \eta_{min}^t, \eta_{max}^t, \Delta p_{max}^t, \Delta \Theta_{max}^t, p^t, \Theta^t] \in \mathbb{R}^{24}. \quad (2)$$

c) Terminal Condition: In contrast to the approach proposed by Chen et al. [13], we additionally define a terminal condition to assist the agent in learning a desired policy that could generate trajectories with fewer steps. This could also alleviate unnecessary further data collection and reduce the time cost for calibration.

The MDP process is terminated if all the coverage parameters have reached predefined thresholds, and an additional terminal reward is given. This design is intended to encourage the agent to maximize the coverage of all properties, rather than only extremely cover one or two of them. Although the values for the thresholds are set to be not too difficult to achieve to stabilize the learning process, in practice the agent will also learn to achieve as much coverage as possible by maximizing the total rewards.

d) Reward Design: The rewards include step-wise rewards $R_t$ and a terminal reward $R_T$. For the step-wise rewards $R_t$, we assign a positive reward for increased coverage and a negative reward for higher path length. Here we include path length because the shortness of the calibration sequence heavily reduces the run time of the calibration software that
is a non-linear optimization process. For the step-wise reward we have a weighted sum

\[ R_t = \| S_t[0:18] - S_{t-1}[0:18] \|_1 - c_1 \| \rho^t - \rho^{t-1} \|_2 - c_2 \| \Theta^t - \Theta^{t-1} \|_2, \]  

(3)

where \( c_1 \) and \( c_2 \) are tunable positive hyperparameters. The terminal reward \( R_T \) is designed to favor behavior that reaches the terminal condition. In our case, \( R_T \) includes a positive constant reward and dynamic rewards for accurate and less uncertain calibration results. Given the ground truth and estimated calibration parameters \( \Phi \) and \( \hat{\Phi} \) respectively, the terminal reward is defined as

\[ R_T = c_3 + \frac{c_4 \| \Phi \|_2}{\| \Phi - \hat{\Phi} \|_2}, \]  

(4)

where \( c_3 \) and \( c_4 \) are tunable positive hyperparameters. High terminal rewards are given for low calibration errors to encourage the agent to learn accurate calibrations.

C. Soft-Actor Critic Reinforcement Learning

In our framework, we apply SAC [15] to train the agent to learn the optimal policy. During the online training process, our agent jointly learns a policy network, a Q-network, and a value network together. The parameters of networks are updated using the maximum entropy framework, which introduces entropy into the original actor-critic update to improve exploration of the environment. Furthermore, this framework also enables off-policy learning that improves the sample efficiency. Due to the high computational cost for calibration, we chose SAC as it requires few experiences to reach high performance.

IV. EXPERIMENTS

A. Evaluation with real platform

In this section, we conduct experiments to train and evaluate our learned policy. The whole pipeline includes policy training, simulation evaluation, and validation on a real platform. The results show that our learned policy performs best among all the candidate trajectories and policies for both intrinsic and extrinsic calibration. We also verify that our policy can be applied on a real robot arm to calibrate a real VI-sensor.

B. Policy Training

We use Gazebo [21] for dynamics and sensor simulation. Our simulation environment includes a checkerboard and a VI sensor consisting of a pinhole camera and an IMU mounted on the end-effector of a FRANKA EMika Panda robot arm.1 The detailed sensor settings are shown in Table I and II. We include noise and drift for the IMU and distortion for the camera to achieve a more realistic simulation. During training, the parameters for camera intrinsics and camera-IMU extrinsics are re-sampled from Gaussian distributions after each episode, to ensure that the model learns to generalize well also to other similar sensors. Our target board is a

1https://erdalpekel.de/?p=55

| Parameters | Values |
|------------|--------|
| frequency  | 200 Hz |
| accelerometer drift | 0.006 m/s² |
| accelerometer noise | 0.004 m/s² |
| gyroscope drift | 0.000038785 rad/s |
| gyroscope noise | 0.0003394 rad/s |

TABLE I: Simulation settings for the IMU.

| Parameters | Values |
|------------|--------|
| frequency  | 10 Hz  |
| width      | 640 px |
| height     | 480 px |
| nominal horizontal FOV | 1.0 rad |
| model      | pinhole-radtan |

TABLE II: Simulation settings for the camera.

| Intrinsics                  | FOV [rad] | Mean | Std. |
|-----------------------------|-----------|------|------|
| Distortion                  |           |      |      |
| X [m]                       | 0.06      | 0.01 |
| Y [m]                       | 0.00      | 0.01 |
| Z [m]                       | -0.10     | 0.01 |
| Roll [rad]                  | 0.00      | 0.10 |
| Pitch [rad]                 | 0.00      | 0.10 |
| Yaw [rad]                   | 1.57      | 0.10 |

| Extrinsics                  |           |      |      |
|----------------------------|-----------|------|------|

7 × 6 checkerboard with 7 cm × 7 cm squares. The distance from the target to the robot arm’s initial pose is 1.5 m. In each step, the direct transformation is executed by the robot arm using the MoveIt motion planning package [22]. If the terminal condition is satisfied at the last step, both intrinsic and extrinsic calibration will be performed using Kalibr, where the intrinsic result is used as input for the extrinsic calibration.

With these environment settings and predefined MDP, we perform SAC to learn the optimal policy for calibration. For the value network, Q-network, and policy networks of the agent, we use fully connected networks with 2 hidden layers and a size of 256. We restrict the maximal number of steps for each episode to be 20 and train the agent for nearly 15000 steps. At that stage, the performance of the agent converges, and it can achieve the terminal condition for most of the episodes.

C. Evaluation in Simulation

In the simulation evaluation experiments, we test 6 different types of policies or trajectories for 3 calibration tasks: pure intrinsic calibration, pure extrinsic calibration with known intrinsic ground truth, and calibration of both intrinsic and extrinsic parameters. For comparison, we obtained baselines that include handcrafted trajectories for intrinsic and extrinsic calibration from expert knowledge (a short and a long version) and trajectories learned with model-based RL [13]. In addition, our learned policy is also compared with randomly parametrized trajectories introduced in [13] and a random policy of direct transformations that parametrizes movement similarly to our proposed approach.
We mainly compare the extrinsic calibration accuracy, which had to perform cyclical motions and could therefore with other trajectories. This could be because the random policies are trained without asymmetric distortions, the hand-crafted trajectory also achieve nearly the highest accuracy. What we show is that different calibration policies can reach an optimal result given enough time, but our proposed policy is able to learn better motion primitives that human experts. We are able to do in less than 5 minutes what would take a human more than 50 minutes.

D. Evaluation on Real Platform

With satisfactory results in the simulation, we continue to further evaluate our learned policy on the real platform. We use a tightly time-synchronized VI sensor system including an ADIS IMU and a global shutter fisheye camera [6]. We calibrate the intrinsic of the monochrome camera and extrinsic between the fisheye camera and the IMU sensor. To evaluate our approach, we fix it on the end-effector of a real FRANKA EMIKA robot arm and run the learned policy online to generate a calibration trajectory, as is shown in Figure 3.

We add an additional automatic adjustment stage before each episode when the robot tries to align the center of the

![Image 3: Simulation (left) and real experiment setup (right) for VI calibration. The example images from the camera sensor are shown respectively.](image-url)
image view and the center of the target board. Given a real VI sensor, we directly run the learned policy to generate different motions according to the current state sequentially and record the generated trajectory. The recorded data is used both for camera intrinsic calibration and VI extrinsic calibration, in the same way as the simulation experiment. The results are compared with the calibration results done by hand by a human expert.

| Intrinsic calibration | Mean error [%] | Error std [%] | Path length [m] | Calibration time [s] |
|-----------------------|----------------|---------------|-----------------|---------------------|
| random trajectory     | 14.81          | 7.27          | 2.27            | 67                  |
| hand-crafted trajectory (long) | 3.27          | 1.90          | 4.05            | 148                 |
| hand-crafted trajectory (short) | 8.79          | 2.39          | 2.77            | 63                  |
| model-based learned trajectory | 7.65          | 7.59          | 2.04            | 133                 |
| random moving policy  | 6.60           | 6.33          | 1.16            | 44                  |
| model-free learned policy (ours) | 3.90          | 2.83          | 0.86            | 39                  |

**TABLE VIII:** Comparison of intrinsic calibration results between hand-crafted trajectory and learned trajectory.

| Extrinsic calibration | Mean error [%] | Error std [%] | Path length [m] | Calibration time [s] |
|-----------------------|----------------|---------------|-----------------|---------------------|
| random trajectory     | 9.48           | 6.87          | 2.27            | 1908                |
| hand-crafted trajectory (long) | 1.80          | 1.06          | 4.05            | 3019                |
| hand-crafted trajectory (short) | 2.66          | 2.14          | 0.56            | 611                 |
| model-based learned trajectory | 1.76          | 1.19          | 1.20            | 756                 |
| random moving policy  | 1.73           | 1.34          | 1.16            | 569                 |
| model-free learned policy (ours) | 1.16          | 0.63          | 0.86            | 213                 |

**TABLE IX:** Comparison of calibration outputs between hand-crafted trajectory and learned trajectory.

A comparison of intrinsic and extrinsic calibration results is shown in Table VII and Table VIII respectively. For both intrinsic and extrinsic calibration, our calibrated results are similar to the handcrafted results obtained by an expert. The comparison of calibration metrics is shown in Table IX further proving that our proposed method and the human expert produce similar quality results. Also in this experiment our policy is able to perform the calibration faster than the expert, requiring only a third of the time. These experiments prove the transferability of our method for unified intrinsic and extrinsic calibration in the real case.

V. CONCLUSION

In this work, we proposed the use of model-free deep RL to learn a policy that guides the movement of a robotic arm holding a VI sensor to efficiently collect measurements that can be used for both camera intrinsic calibration and VI extrinsic calibration. For joint intrinsic and extrinsic calibration our proposed method outperforms all other benchmarks in terms of both lowest mean calibration error and shortest path length. Evaluations on the simulation platform and real robot system show that our learned policy is able to generate better motion primitives than human experts, leading to a much faster calibration process. In simulation our proposed method is more than 10× faster than a hand-crafted trajectory, which in our real-world experiment transfers to a speed-up of 3× over humans. Our method is ideally suited for a factory scenario where high throughput calibration is needed, or as a learning tool to teach optimal calibration motions to human experts.

In future work, the use of an Aprilgrid [23] could enhance observability as it allows for different viewing angles. In addition, partially visible Aprilgrid targets can be detected without problems, which would greatly simplify the data collection process.
REFERENCES

[1] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, “Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback,” The International Journal of Robotics Research, vol. 36, no. 10, pp. 1053–1072, 2017.

[2] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018.

[3] F. Tschopp, T. Schneider, A. W. Palmer, N. Nourani-Vatani, C. Cadena, R. Siegwart, and J. Nieto, “Experimental comparison of visual-aided odometry methods for rail vehicles,” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1815–1822, 2019.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual–inertial odometry using nonlinear optimization,” The International Journal of Robotics Research, vol. 34, no. 3, pp. 314–334, 2015.

[5] P. Corke, J. Lobo, and J. Dias, “An introduction to inertial and visual sensing,” The International Journal of Robotics Research, vol. 26, no. 6, pp. 519–535, 2007.

[6] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, “A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam,” in 2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014, pp. 431–437.

[7] F. Tschopp, M. Riner, M. Fehr, L. Bernreiter, F. Furrer, T. Novkovic, A. Pfrunder, C. Cadena, R. Siegwart, and J. Nieto, “Versavis—an open versatile multi-camera visual-inertial sensor suite,” Sensors, vol. 20, no. 5, p. 1439, 2020.

[8] J. Kelly and G. S. Sukhatme, “Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration,” The International Journal of Robotics Research, vol. 30, no. 1, pp. 56–79, 2011.

[9] F. Nobre and C. Heckman, “Learning to calibrate: Reinforcement learning for guided calibration of visual–inertial rigs,” The International Journal of Robotics Research, vol. 38, no. 12-13, pp. 1388–1402, 2019.

[10] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013, pp. 1280–1286.

[11] P. Furgale, T. D. Barfoot, and G. Sibley, “Continuous-time batch estimation using temporal basis functions,” in 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 2088–2095.

[12] D. Scaramuzza and Z. Zhang, “Visual-inertial odometry of aerial robots,” arXiv preprint arXiv:1906.03289, 2019.

[13] L. Chen, Y. Ao, F. Tschopp, A. Cramariuc, M. Breyer, J. J. Chung, R. Siegwart, and C. Cadena, “Learning trajectories for visual-inertial system calibration via model-based heuristic deep reinforcement learning,” in Proceedings of the 4th Conference on Robot Learning (CoRL), November 2020.

[14] P. Furgale, S. Leutenegger, and R. Siegwart, “Optimal trajectory design for self-calibration,” in 2016 IEEE International Conference on Robotics and Automation. IEEE, 2016, pp. 2169–2175.

[15] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95-International Conference on Neural Networks (Cat. No PR00149), vol. 1. IEEE, 1995, pp. 1942–1948.

[16] T. Haarnoja, A. Zou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” arXiv preprint arXiv:1801.01290, 2018.

[17] A. M. Andrew, “Multiple view geometry in computer vision,” Kybernetes, 2001.

[18] P. F. Sturm and S. J. Maybank, “On plane-based camera calibration: A general algorithm, singularities, applications,” in Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), vol. 1. IEEE, 1999, pp. 432–437.

[19] Q. Zhang and R. Pless, “Extrinsic calibration of a camera and laser range finder (improves camera calibration),” in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2301–2306.

[20] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95-International Conference on Neural Networks (Cat. No PR00149), vol. 1. IEEE, 1995, pp. 1942–1948.

[21] T. Haarnoja, A. Zou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” arXiv preprint arXiv:1801.01290, 2018.

[22] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ros topics],” IEEE Robotics & Automation Magazine, vol. 19, no. 1, pp. 18–19, 2012.

[23] E. Olson, “Apriltag: A robust and flexible visual fiducial system,” in 2011 IEEE International Conference on Robotics and Automation. IEEE, 2011, pp. 3400–3407.