Learning Optimal Trajectories for Visual-inertial System Calibration via Deep Reinforcement Learning

Le Chen
D-ITET
ETH Zurich
lechen@ethz.ch

Yunke Ao
D-MAVT
ETH Zurich
yunkao@ethz.ch

Abstract: We present a new approach to learn optimal trajectories for visual-inertial system calibration using deep reinforcement learning. Precise calibration is essential to visual-inertial systems applications. Typically, it requires performing sophisticated motions in front of a calibration target manually. Our key contribution is to model the calibration process as a partially observable Markov decision process and use deep reinforcement learning to establish a sequence of trajectories performed by a robot arm to collect measurements considering camera coverage, observability and path length. Our experiments show that the trajectories generated by our learned policy allows us to collect data efficiently that yield desired sensor parameters.

Keywords: Visual-Inertial system, Calibration, Deep Reinforcement Learning

1 Introduction

In recent years, the visual-inertial system, which consists of several cameras and IMUs (Inertial Measurement Unit), has become increasingly popular [1]. Before using it, a set of sensor parameters need to be calibrated, including camera intrinsics, camera-IMU extrinsics, and the time offset between sensors. The calibration could be done offline or online following sophisticated routines with sufficiently exciting motions, making it non-trivial for an inexperienced operator to collect enough measurements that allow the desired sensor parameters to be inferred [2][3]. So instead of performing this task by hand or with a manually programmed operator, we want to use deep reinforcement learning (RL) to learn the best motion primitives and perform them on a robot arm.

Designing best motion primitives for calibration is still challenging and not fully studied. Previous works include applying trajectory optimization and reinforcement learning to address this problem. For the class of optimization methods that maximize observability Gramian of the trajectories on calibration parameters [4][5], it is still difficult to include other practical objectives such as target points coverage for camera calibration and trajectory length for energy saving in the optimization frameworks, because these objectives are hard to model. As for the learning-based methods [3], the trajectories of motions are only pre-defined empirically and Q-learning was applied to learn to choose the best sequences of those pre-defined trajectories to render sufficient observability. In this way, there are two aspects that could be further studied based on previous works:

- **Optimize more objectives:** it is not obvious whether the trajectory that renders the sufficient observability of states is the one that gives the best calibration. Except for observability, other empirical requirements can also be included in the objectives to optimize.
- **Learn the trajectory:** as a current learning-based method only learns how to choose the predefined trajectories, the parameterized trajectories themselves can also be included in the policy and learned.

The above gaps are both addressed in our work. We aim to design an algorithm to learn the best sequence of trajectories to collect measurements that achieve multiple objectives including both observability and practical requirements. The sequential decision of trajectories will be based on historical information including parameter estimation results, actions, and information gain of parameters. The approach will be applied to calibrate camera intrinsics and camera-IMU extrinsics for different kinds of visual-inertial platforms. In order to achieve the learned sequence of trajectories automatically, the trajectories are executed by a robot arm during training in the simulation and should be feasible to be executed by a real robot arm as well.

To solve our problem, we model calibration as a partially observable Markov decision process (POMDP) and use RL to establish the sequence of motion trajectories which optimizes sensor calibration. Compared with other solutions, the action of our model has fewer constrains, making it possible to learn a more general policy for calibration. In addition, we not only consider different information-theoretic metrics of the trajectories but also take camera coverage and path length into account in the reward of our RL model. For the RL algorithms, Deep Deterministic Policy Gradient (DDPG) [6] and Twin Delayed DDPG (TD3) [7] are applied. Different from Q-learning that can only learn policys in discrete action space, DDPG and TD3 are capable to learn a complex policy for continuous action spaces such as the trajectory parameter space.

The main contributions of this work are the following:

- To the best of our knowledge, our approach is the first of its kind that models the calibration as a POMDP and solve it using deep reinforcement learning.
- Our method addresses different practical requirements other than observability, for example, path length and camera coverage.
- Our method learns complete policies for generating motion trajectories based on previous parameter estimation, information gain, and actions.

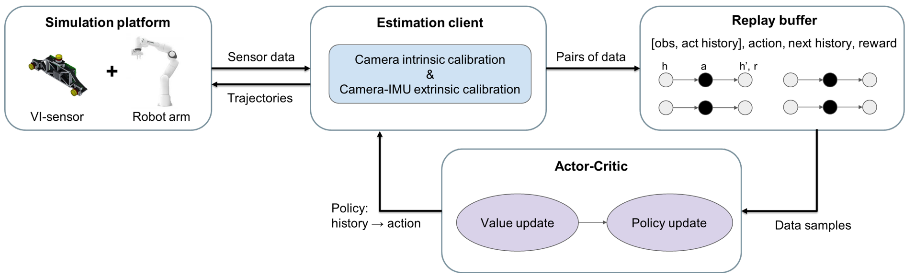


Figure 1: Overview of our calibration framework

The overall structure of our paper is as follows: In Section 2, we review several related works and their limitations. In Section 3, we detail our method from problem formulation to algorithm design. In Section 4, we provide implementation details of our framework. Experimental results are shown in section 5. Finally, in Section 6, we summarize our contribution and discuss the drawbacks and possible future research directions.

2 Related Work

The most popular method for camera calibration during the last decades is to use a known calibration pattern and apply nonlinear regression to obtain the parameters. This method has been used both for camera intrinsic [8] and extrinsic calibration [9]. For the visual-inertial system, most methods rely on external markers. An approach presented in [10] applies EKF to jointly estimate the relative pose between sensors. A parametric method proposed in [11] represents the pose and bias trajectories using B-splines and introduces a batch estimator in continuous-time. While those methods are relatively efficient, they still require expert knowledge to reach discern accuracy. For inexperienced operators, it is not easy to perform a good motion to infer the sensor parameters.

The problem of designing the best motion primitive is first addressed by methods based on trajectory optimization to find the trajectory that renders the highest observability of calibration parameters. A method proposed in [5] optimizes the expended empirical local observability gramian of unknown parameters based on the measurement model to solve for the best state and input trajectories. Preiss et al. [4] further extended the method to be obstacle-free and more balanced among multiple objectives. Both of the approaches only optimize observability evaluated by observability gramian of the trajectories, where empirical and general requirements that are difficult to model cannot be included.

Our framework combines different evaluation metrics of information gain for variables in the reward design and learn to obtain most rewards using model-free RL.

Another class of methods applies RL to get the best sequence of trajectories selected from a predesigned library. Nobre et al. [3] modeled the calibration process as an MDP, where the states are estimated parameters and actions are choices of trajectory from the library at each step. The MDP planning problem is solved by Q-learning algorithm. However, the method only yields suggestions on predefined motions to choose from rather than the trajectories themselves. Our method includes the trajectory parameters in the policy, which are learned using more powerful deep RL algorithms for continuous action space.

3 Method / Your Approach

3.1 Visual-inertial Calibration

Our estimator follows the Kalibr framework [11], where Levenberg-Marquardt algorithm is applied to minimize the loss between the obtained measurements and predicted measurements to maximize the likelihood of unknown parameters $P(X, \theta | Z, L)$, where X is the estimated pose trajectory, Z is the measurements of visual-inertial sensors and L is the known position of landmarks on the calibration target. As is covered in detail in [12], the covariance matrix of the known parameters can be obtained by $\Sigma_{X\theta} = (J^T G J)^{-1}$, where J is the Jacobian of all error terms and G is the stack error covariance. Then covariance of calibration parameters Σ_θ can be further extracted and normalized to $\overline{\Sigma}_\theta$. The information gain can be evaluated with different metrics based on the covariance matrix:

- **A-Optimality:** $H_{Aopt} = trace(\overline{\Sigma}_\theta)$
- **D-Optimality:** $H_{Dopt} = \det(\overline{\Sigma}_\theta)$
- **E-Optimality:** $H_{Eopt} = \max(eig(\overline{\Sigma}_\theta))$

Minimizing those metrics leads to maximization of the richness of information. All those metrics would be included in the reward design for motion learning, as is explained in the following section.

3.2 Learning Motions

The whole calibration process is modeled as a POMDP, as is shown in figure 2, which includes state S_t, action A_t, observation Y_t, transition model $P(S_{t+1}|S_t, A_t)$, observation model $P(Y_t|S_t)$, and reward R_t at each time step t. We define the action at each time A_t as a looped parameterized trajectory with the same start and terminal pose. The pose trajectory $\{[x_\tau, y_\tau, z_\tau, r_\tau, p_\tau, y_\tau]^T\}_{\tau=1:K}$ are parameterized by $\{\sum_{k=1,2,4} \boldsymbol{a}_k(1 - \cos k\tau) + \boldsymbol{b}_k \sin k\tau\}_{\tau=1:K}$, where $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ are 6×1 vectors. So for one trajectory, there are 36 parameters needed in total, which is the action space dimension of POMDP. The reason to choose $\sin k\theta$ and $\cos k\theta$ as basis functions is that they are symmetric and periodic, which are satisfied by many good empirical trajectories and automatically impose more constraints so that reduces the number of parameters needed.

The state is defined as all the measurements acquired so far. Let D_t be the measurements acquired with the action A_t, then $S_t = \bigcup_{i=0}^{t-1} D_i$. In this way, the state transition satisfies the Markov property, so the transition model is:

$$S_{t+1} = \bigcup_{i=0}^{t} D_i = (\bigcup_{i=0}^{t-1} D_i) \cup D_t = S_t \cup h(A_t) \tag{1}$$

where h represents the map from the trajectory parameters to the measurements acquired with the trajectory. The observation of POMDP is a vector that stacks all the calibration parameters and their information gain status, which is determined purely by the current state and the observation model:

$$Y_t = [\theta_t^*, O_t]^T = Est(\bigcup_{i=0}^{t} D_i) = Est(S_t) \tag{2}$$

where Est encodes the estimation model that returns the optimal calibration parameters and their respective information gain status given all the current measurements. The information gain status

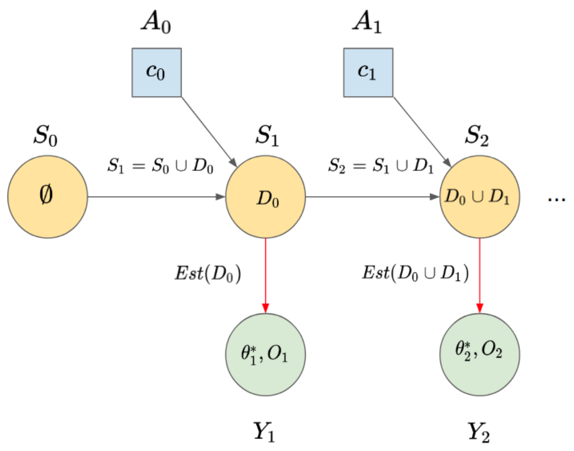


Figure 2: POMDP for calibrarion

is encoded by O_t, an vector stacking all eigenvalues of the covariance matrix of parameters. This is only included in observations for learning camera-IMU extrinsic calibration.

Finally, the reward of each step is composed of four parts: empirical reward e_t, information gain for parameter calibration o_t, trajectory length l_t and difference from ground truth parameters d_t. Empirical reward encodes intuitive requirements such as target points coverage. The information gain part includes different evaluation metrics like determinant, trace and eigenvalues of covariance matrix. The trajectory length is computed by summing up absolute position change and Euler angle change at each time step. The difference from ground truth is for encouraging accurate calibration, which can be obtained in simulation. The reward is a weighted sum of increments of each term at each time step:

$$R_t = a_1 \delta e_t + a_2 \delta o_t - a_3 \delta d_t - a_4 \delta l_t \tag{3}$$

where $a_1, \cdots, a_4$ are the weights that are larger than 0 and can be tuned for each part.

For POMDP, decisions are made based on the belief state $b(S_t) = b(Y_{1:t}, A_{1:t-1})$. We use Recurrent Neural Network (RNN) to encode the dependence on historical observations and actions respectively for actor and critic of the RL architecture. For each time, the inputs are a sequence of historical observations and actions, and the outputs are the policy and Q-function respectively for the actor and critic networks, where the critic network will take the current action as input as well. Then the actor and critic will be trained using DDPG or TD3 to learn the best policy of the POMDP. We choose off-policy deterministic algorithms in order to improve sample efficiency. The whole network structure is shown in figure 3.

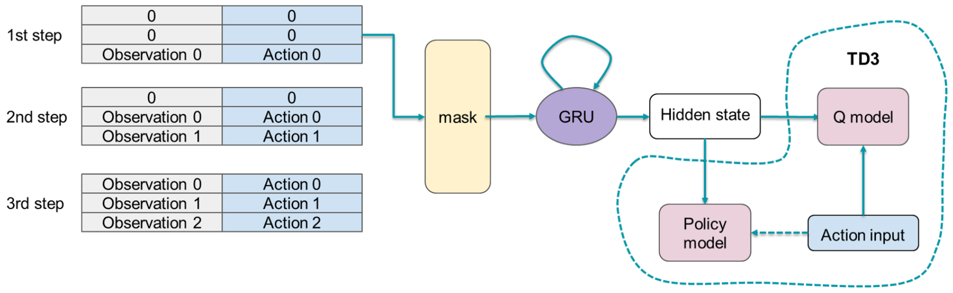


Figure 3: Actor Critic for POMDP

4 Implementation

4.1 Simulation

We evaluate our proposed framework by experiments in the simulation platform. The platform is built based on Gazebo, which includes a visual-inertial system, a checkerboard, and a FRANKA EMIKA Panda robot arm. The visual-inertial system, equipped with a monocular camera and an IMU, is mounted on the end-effector of the robot arm. The setting of camera and IMU sensor in the simulation is shown in table 1 and table 2. As mentioned in section 3.2, we use sums of sinusoidal and cosinusoidal functions of time for position and orientation with 36 parameters to describe the trajectory. The end-effector, mounted with sensors, moves following the given trajectory in front of the checkerboard, as is shown in figure 4.

update rate	acceleration drift	acceleration noise	augular velocity drift	angular velocity noise
200 Hz	0.006	0.000038785	0.000038785	0.0003394

Table 1: Simulation settings for IMU

update rate	width	height	horizontal field of view	noise	distortion
7 Hz	640	480	1.0	0.0	None

Table 2: Simulation settings for camera

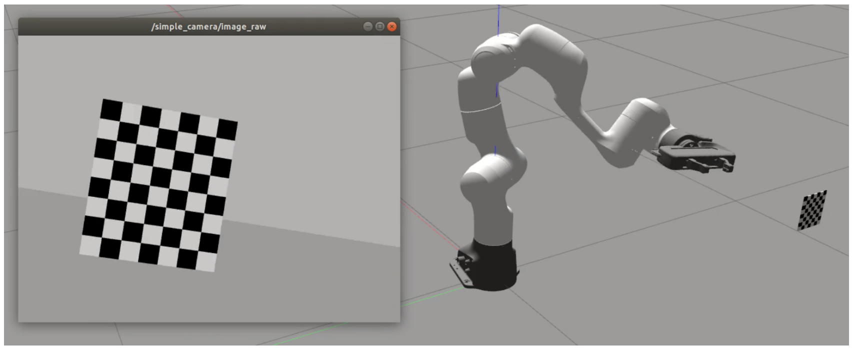


Figure 4: Simulation platform for visual-inertial system calibration

4.2 Camera Intrinsic Calibration

For the learning framework of camera intrinsic calibration, not all samples are included in the database for calibration in practice. Instead, given a new sample, the algorithm compares the variety of this sample with all the samples in the database and judge whether the motion speed is sufficiently low between this frame and the previous frame. If this sample is different enough from others and the movement speed is not too low, it would be added to the database. The algorithm also judges if the samples in the database are good enough to do the calibration by computing how much progress has been made toward adequate variation. When good enough, the intrinsic parameters would be calibrated.

In each trial of calibration, we limit the maximum step size to be 6 (run at most 6 looped trajectories), and the trial terminates when the parameters are within around 0.5% relative calibration error from the ground truth. We limit the range of each element of action within ± 0.03 and set the exploration Gaussian noise to be 0.005. The empirical reward for camera calibration is coverage of targets in the image view, which consists of 'X', 'Y', 'size', and 'skew' coverage progress. The information gain reward is the number of images selected and stored in the database. The agent is trained using off-line data sampled from the replay buffer in each 3 on-line interaction steps.

4.3 Camera-IMU Extrinic Calibration

For the learning framework of camera-IMU extrinsic calibration, we make use of the Kalibr toolbox to estimate the extrinsic. Each episode contains 3 steps (3 trajectories), after each step, new sensor data from the current trajectory will be merged with all previous data and fed together to Kalibr for calibration. In the first two steps of the episode (trial), we only call Kalibr with one optimization iteration to get the covariance matrix and extract optimality metrics as feedback information while reducing time cost. For the last step, Kalibr is called with at most 3 optimization iterations to get the final calibration result. The range of each element of action is limited to ± 0.02. The whole process is shown in figure 5. The empirical reward for extrinsic calibration is the entropy of IMU measurement data of 6 dimensions. The information gain reward is negative A-Optimality computed based on the covariance matrix. If the calibration result has an Euclidean distance less than 0.006 from the ground truth, a large terminal reward will also be added. The agent is trained using off-line data sampled from the replay buffer in each single interaction step.

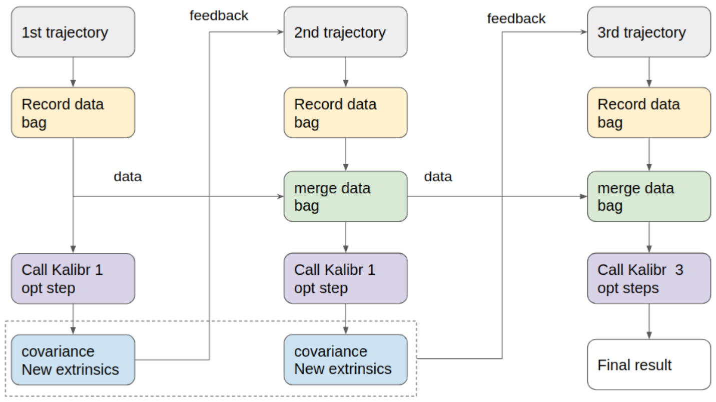


Figure 5: Each episode for learning camera-IMU extrinsic calibration

4.4 Network

The network follows the actor-critic architecture mentioned in Section 3. More specifically, the action sequence and observation sequence are combined to be fed to the GRU network, which outputs a final hidden state of 256 units. Then for the actor-network, the hidden state is fed to another dense layer and output the policy. For the critic network, the current action is input after going through a dense layer and be combined with the GRU hidden state. Finally, the Q-value is output after another dense layer. All the dense hidden layers have 256 units. For the target networks, the moving average momentum is 0.01. The whole network is trained with a batch size of 32.

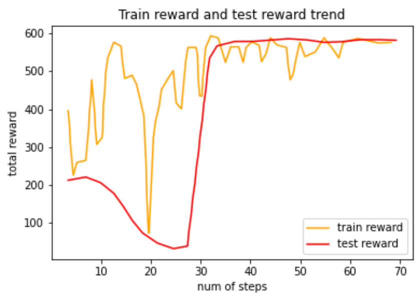
5 Experiment Result

5.1 Camera Intrinsic Calibration

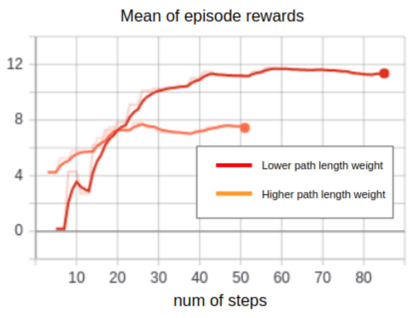
With the above settings, our algorithm could learn a policy for camera calibration. As is shown in Figure 6a, both the training reward and testing reward converged within 70 trials. Although our maximum step size is 6, the learned policy only takes one trajectory and then reaches the terminal state, and the numbers of samples obtained in the database are all between 10 to 20. We further evaluate our learned policy by calibrating cameras with intrinsics different from the camera used for training. The average relative calibration errors w.r.t the ground truth for different horizontal fields of view (FOV) with the same image size (640×320) is shown in the table 3, where our policy is trained using the camera with horizontal FOV 1.0. For each camera, we perform calibration experiments using the same policy for 5 times. The results show that the proposed framework could learn how to perform good motion trajectories and collect enough measurements efficiently that yield the desired camera intrinsic parameters. And the policy trained using one camera generalizes well for other cameras with a similar FOV.

Horizontal FOV	0.85	0.9	0.95	1.0	1.05	1.1	1.15
fx error (%)	0.0882	0.0443	0.0839	0.0964	0.0754	0.0203	0.1018
fy error (%)	0.0030	0.0016	0.0022	0.00273	0.0016	0.0013	0.0023
cx error (%)	0.0581	0.0219	0.0748	0.0778	0.0654	0.0124	0.0958
cy error (%)	0.00434	0.0011	0.0011	0.0036	0.0027	0.0022	0.0035

Table 3: Relative errors of calibration for different intrinsics with trained policy



(a) Camera intrinsic learning curve

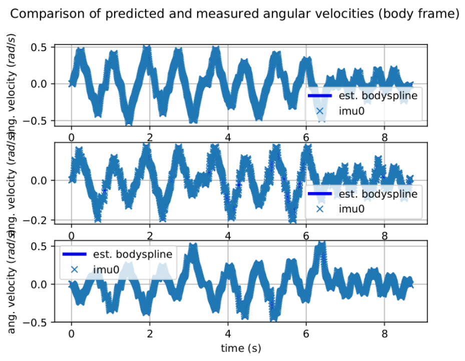


(b) Camera-IMU extrinsic learning curve

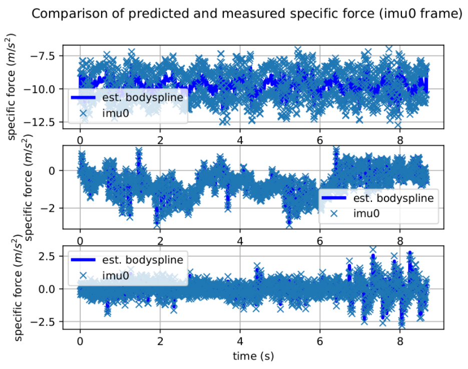
Figure 6: Learning curves for calibration

5.2 Camera-IMU Extrinsic Calibration

We also train an agent to learn a policy for camera-IMU extrinsic calibration. During the training process, it could be observed from the learning curves shown in figure 6 that the agent suddenly reaches a local optimum in several initial training steps and remains stable for tens of timesteps regardless of the weight of path length in the total rewards. So we stop the training progress and test the learned trajectory. It turns out that even with trajectories of small sizes, the calibrator can still obtain an accurate result. This is possibly caused by perfect modeling of the sensors in the simulation, where no further information is needed from more coverage of target in the image view given true camera intrinsic. Even though the scale of IMU data is small, the motion is sufficiently exciting in this small range. The total reward is the weighted sum of information gain, entropy, calibration error, and path length. As information gain and entropy are not directly influenced by scale, such small trajectories obtain high accuracy and small path length, which are potential optimal solutions in our problem settings. The calibration results including reprojection error and IMU data output by Kalibr given sensor data obtained using our trajectory are shown in figure 7 and 8a.



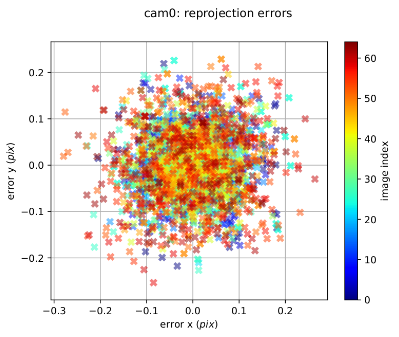
(a) Estimated and measured angular velocities



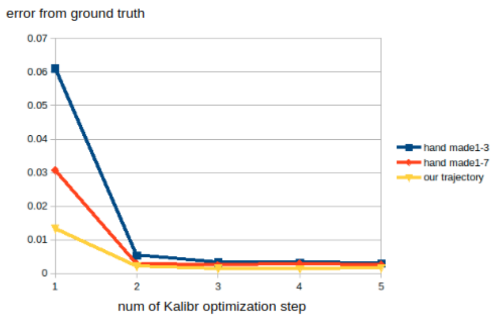
(b) Estimated and measured accelerations

Figure 7: Predicted and measured IMU data with learned trajectory

Then we compare path length, information gain and calibration error between the trajectory we get and hand-designed sequence of trajectories as is shown in table 4. We also compared the trend of calibration error w.r.t the maximum number of Kalibr optimization iterations, which is shown in figure 8b. The hand-designed sequence including 7 trajectories: translation along X, Y and Z axis,



(a) Reprojection errors with learned trajectory



(b) Error from ground truth w.r.t number of Kalibr iterations

Figure 8: Metrics for calibration accuracy

rotation around Z-axis, the combination of translation around the X and Y axis and finally the 8-shape trajectories. We also test calibration on a hand-designed sequence with only 3 trajectories. The scale of these trajectories is set to fully cover the position of the target in image view as is suggested empirically. As is shown in table 4, the trajectories generated by learned policy obtains higher calibration accuracy with a shorter path length than the hand-designed sequence. For information gain, learned trajectories obtain similar A-Optimality value as hand-designed sequence of 3 trajectories. With 7 hand designed trajectories, information gain is higher because the estimation of parameters are more certain as more measurements are acquired.

Trajectories	A-optimality	Path length	Calibration error after 5 Kalibr iterations
hand designed 3 trajectories	0.445	3.07	0.00299
hand designed 7 trajectories	0.218	4.65	0.00254
Our trajectories	0.459	2.302	0.00156
random trajectories	-	-	0.00418

Table 4: Comparison of learned trajectory with hand designed and random trajectories

6 Conclusion

This work introduced a novel framework that uses deep RL to generate optimal trajectories for collecting enough measurements efficiently to calibrate the desired visual-inertial sensor parameters. The experiment shows that the trajectories generated by our learned policy lead to better calibrations.

There remain some drawbacks in this work. Firstly, the training time for each episode is long because we incorporate Kalibr in our framework. Considering the computing resource we have, we were not able to generate a very large number of episodes and train the agent based on that. With a small number of experiences, the training process is more fragile and easily result in worse performance as training steps increase. Secondly, for the camera intrinsic calibration part, generating ground truth intrinsic parameters from a distribution is better than simply setting to fixed values. Thirdly, in real life, the lens is not perfectly modeled and in addition, is definitely non-symmetric and has way more weird optic effects at the edges. However, we have not incorporated distortion in our framework yet.

To further extend this work, we will try to reduce time cost for calibration and seek off-line learning algorithms that are more robust with a small size of experiences. Then we will try to train an agent to learn to calibrate the intrinsic and extrinsic at the same time using one sequence of trajectories. What's more, some non-symmetric imperfections to the lens or other distortion models will be added to our framework to improve robustness. And most importantly, we will test our method in real life. At a high level, our method is to learn how to collect data efficiently for certain tasks, which could not only be calibration but also be other robotic tasks that require good data to perform optimization. In the future, we will explore the high potential of our method in the marker-less self-calibration task.

Acknowledgments

The authors gratefully thank Dr. Cesar Dario Cadena Lerma, Dr. Jen Jen Chung, Mr. Florian Tschopp, Mr. Andrei Cramariuc, Mr. Michel Breyer and other people in ETH Zurich Autonomous System Lab for valuable discussion and suggestions.

References

[1] P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing. *The International Journal of Robotics Research*, 26(6):519–535, 2007.

[2] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[3] F. Nobre and C. Heckman. Learning to calibrate: Reinforcement learning for guided calibration of visual–inertial rigs. *The International Journal of Robotics Research*, 38(12-13):1388–1402, 2019.

[4] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss. Trajectory optimization for self-calibration and navigation. In *Robotics: Science and Systems*, 2017.

[5] K. Hausman, J. Preiss, G. S. Sukhatme, and S. Weiss. Observability-aware trajectory optimization for self-calibration with application to uavs. *IEEE Robotics and Automation Letters*, 2(3): 1770–1777, 2017.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[7] S. Fujimoto, H. Van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[8] 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)*, volume 1, pages 432–437. IEEE, 1999.

[9] 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)*, volume 3, pages 2301–2306. IEEE, 2004.

[10] F. M. Mirzaei and S. I. Roumeliotis. A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. *IEEE transactions on robotics*, 24 (5):1143–1156, 2008.

[11] 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*, pages 1280–1286. IEEE, 2013.

[12] T. Schneider, M. Li, C. Cadena, J. Nieto, and R. Siegwart. Observability-aware self-calibration of visual and inertial sensors for ego-motion estimation. *IEEE Sensors Journal*, 19(10):3846–3860, 2019.