[PLR] Learn to calibrate

Le Chen
lechen@student.ethz.ch

Yunke Ao
yunkao@student.ethz.ch

Abstract

We present a new approach to learn sufficiently informative motions necessary to calibrate a visual-inertial system using deep reinforcement learning. Precise calibration is essential to the applications of visual-inertial systems. 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 observability, camera coverage, and path length. Our experiments show that it allows us to collect data efficiently that yield desired sensor parameters.

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 [5][8]. So instead of performing this task by hand or with a manually programmed operator, we want to use 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 addressing this problem include trajectory optimization and reinforcement learning. For the class of optimization methods [9][4], it is still difficult to include other practical objectives such as target points coverage of camera and trajectory length in the optimization frameworks. As for the learning-based methods [8], the algorithms only learn the choices of motion trajectories, and the trajectories are pre-defined empirically.

The above gaps are all addressed in our work. We model calibration as a partially observable Markov decision process (POMDP) and design an algorithm to learn the best sequence of trajectories to collect measurements that achieve multiple objectives including both observability and practical requirements using deep RL. The sequential decision of trajectories will be based on historical information including parameter estimation results, actions, and observability. The approach could be applied to calibrate camera intrinsics and camera-IMU extrinsics for different kinds of visual-inertial platforms. Compared with other solutions, the action of our model has fewer constrains, making it possible to learn a more general policy for calibration. Additionally, since we not only consider different information-theoretic metrics of observability but also take camera coverage into account, the reward of our RL model is more reasonable.

The main contributions of this work are the following:

- Our approach is the first of its kind that models the calibration as a POMDP and learns to generate the best motion trajectories using RL.
- Our method addresses different practical requirements other than observability, for example, path length and camera coverage.
- Our method learns complete policies for motion trajectories based on information about previous parameter estimation, observability, and actions.

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 and discussion on the experimental evaluation. Finally, in Section 5, we summarize our contribution and draw the conclusion.

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 [11] and extrinsic calibration [12]. For the visual-inertial system, most methods rely on external markers. An approach presented in [7] applies EKF to jointly estimate the relative pose between sensors. A parametric method proposed in [2] 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 [4] 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. [9] 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 observability 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 pre-designed library. Nobre et al. [8] 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 [2], 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 [10], the FIM 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 FIM of calibration parameters Σ_θ can be further extracted and normalized to $\overline{\Sigma}_\theta$. The observability can be evaluated with different metrics based on the FIM:

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

All those metrics can 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, 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 because 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 observability 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 observability status given all the current measurements.

Finally, the reward of each step is composed of four parts: empirical reward e_t, calibration parameter observability 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 observability part includes different evaluation metrics like determinant, trace and eigenvalue of FIM. 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 Deep Deterministic Policy Gradient (DDPG) [6] and Soft Actor-Critic (SAC) [3] to learn the best policy of the POMDP.

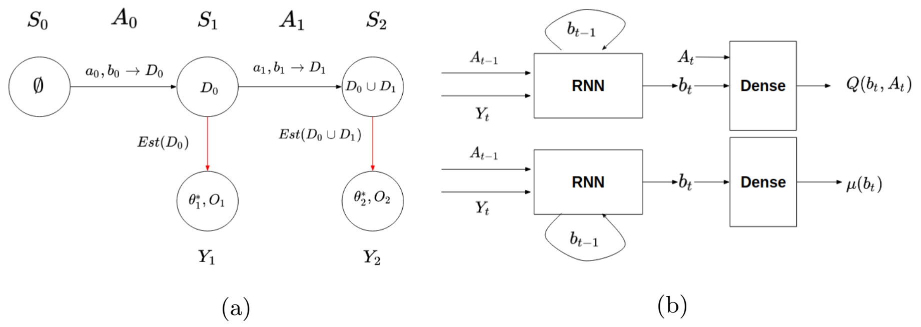

Figure 1: POMDP for calibration (a) and actor-critic RL (b)

For now, instead of calibrating the whole visual-inertial system, we simplify our problem and only calibrate the camera intrinsic parameters. The experiment results show that our method did work for camera intrinsic calibration. In the next step, we will take IMU into account and calibrate the whole visual-inertial system. In addition, we will apply some latest state-of-the-art RL algorithms such as Soft Actor Critic (SAC) or Proximal Policy Optimization (PPO), and try to make our method more robust. We follow up on the timeline we set before and we think it would be realistic to complete all the tasks we planned.

4 Experiments / Evaluations

We will evaluate our proposed framework in both simulation and real-world experiments. Currently, we only trained and evaluated our method using simulated data. The real-world experiments will be conducted after the midterm.

For simulation, a 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. As mentioned in 3.2, we use sums of sinusoidal and cosinusoidal functions of time for position and orientation with 36 parameters to describe an action trajectory. The end-effector, mounted with sensors, moves following the given trajectory in front of the checkerboard. For now, our experiment focuses on camera intrinsic calibration. The sampling frequency for image data is 7 Hz. In practice, not all samples are included in the database for calibration. 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.

For the learning part, the network follows the actor-critic architecture mentioned in Section 3. More specifically, the action sequence and observation sequence first go through one dense layer respectively, and then be combined together 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 layer have 256 units. The whole network is trained with a batch size of 32.

The RL agent is trained using the above platform and network architecture, for each episode, we limit the maximum step size to be 6 (run at most 6 looped trajectories), and the episode terminates when the parameters are within around 0.5% relative calibration error from the ground truth. When using DDPG, we limit the range of each element of action within ± 0.03 and set the exploration Gaussian noise to be 0.005. For the target networks, the moving average momentum is 0.01. In each 5 steps, we will evaluate the policy once by running an episode without noise in the action and recording the total reward.

With the above settings, our algorithm finally learned a policy for camera calibration. As is shown in Figure 2, 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 field of view (FOV) with same image size (640×320) is shown in table 1, where our policy is trained using the camera with horizontal FOV 1. For each camera, we did calibration experiments using the same policy for 5 times. The results show that the proposed framework

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 1: Relative errors of calibration for different intrinsics with trained policy

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.

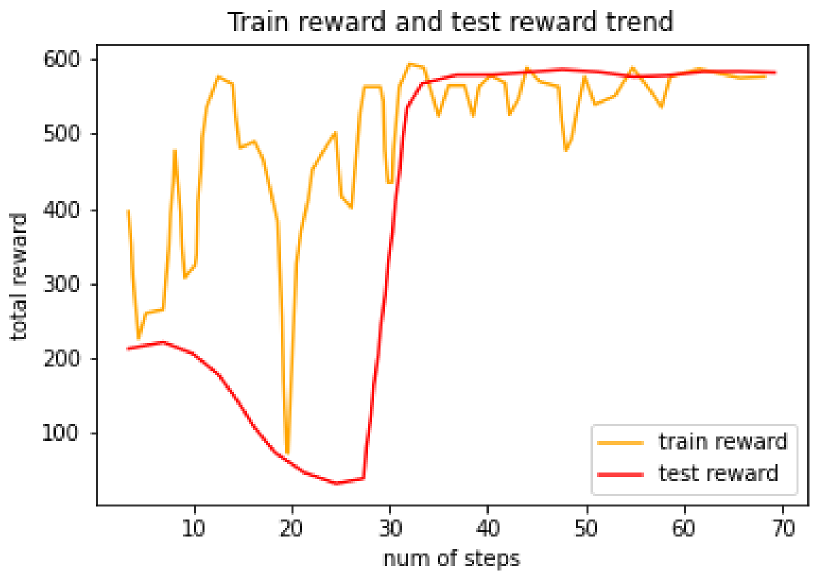

Figure 2: Trend of total reward during training

5 Conclusion

The main outcome of our project is to obtain a reliable trained model that enables the robot arm to perform efficient motion trajectories and collect good samples to calibrate the visual-inertial system. 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 would explore the high potential of our method in the marker-less self-calibration task.

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] 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.

[3] T. Haarnoja, A. Zhou, 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.

[4] 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.

[5] 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.

[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] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.