# Reinforcement Learning for Assisted Visual-Inertial Robotic Calibration

Fernando Nobre and Christoffer Heckman

**Abstract** We present a new approach to assisted intrinsic and extrinsic calibration with an observability-aware visual-inertial calibration system that guides the user through the calibration procedure by suggesting easy-to-perform motions that render the calibration parameters observable. This is done by identifying which subset of the parameter space is rendered observable with a rank-revealing decomposition of the Fisher information matrix, modeling calibration as a Markov decision process and using reinforcement learning to establish which discrete sequence of motions optimizes for the regression of the desired parameters. The goal is to address the assumption common to most calibration solutions: that sufficiently informative motions are provided by the operator. We do not make use of a process model and instead leverage an experience based approach that is broadly applicable to any platform. This is a step in the direction of long term autonomy and "power-on-and-go" robotic systems, making repeatable and reliable calibration accessible to the non-expert operator.

**Key words:** calibration, reinforcement learning, observability, extrinsic, intrinsic

## 1 Introduction

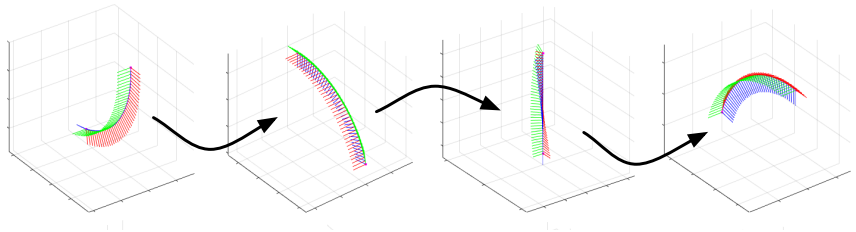
Common to all robotic applications are some set of parameters—camera intrinsics, sensor extrinsics, biases, scale factors, model parameters, etc.—that are essential for higher level robotic tasks such as state estimation, planning and control. These pa-

Fernando Nobre
University of Colorado, Boulder, e-mail: fernando.nobre@colorado.edu

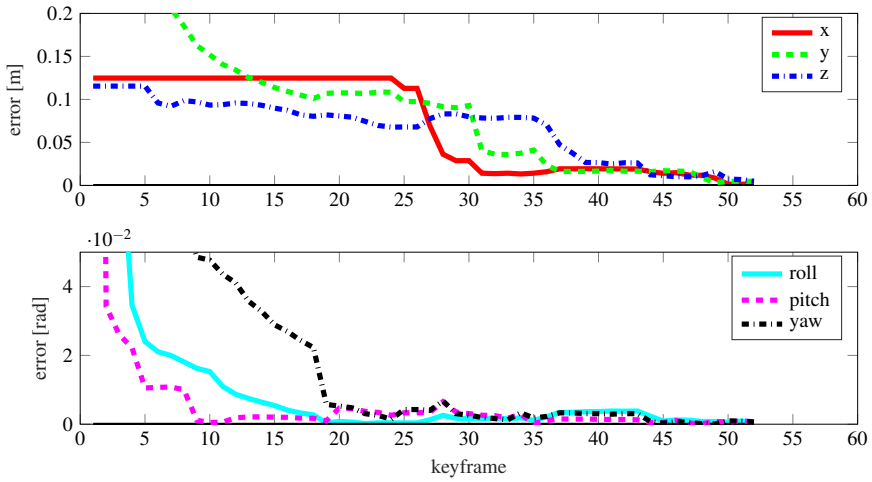
Christoffer Heckman
University of Colorado, Boulder e-mail: christoffer.heckman@colorado.edu.

This work was supported by DARPA award no. N65236-16–1–1000.

rameters are called *calibration parameters* and are usually obtained offline with specific and sometimes sophisticated calibration routines, or online in a self-calibrating framework that relies on sufficiently exciting motions and naturally occurring data. "Sufficiently exciting motions" is a recurring phrase in almost all expositions of self-calibration in the literature [13]. While there has been work on determining the observability of different motions [13], the full observability of the calibration parameters may not be guaranteed for an arbitrary measurement sequence. This renders the calibration procedure non-trivial for an inexperienced operator, especially in non-holonomic platforms such as ground vehicles; it is simply not obvious how to physically move the platform so as to collect measurements that allow the desired parameters to be inferred.



(a) Suggested motions




(b) Extrinsics calibration with suggested motions

Fig. 1: (a) Motions suggested to an inexperienced operator for camera-to-IMU extrinsic calibration, drawn from the learned policy. (b) Convergence of translation and rotation over the motions suggested in (a). Note how the first suggested movements provide little to no information on translation, but allow rotation to converge. This follows our practical knowledge that estimating rotation first improves convergence on translation.

In the context of life-long autonomous operation, self-calibrating systems are a necessity since they allow for compensation of errors induced over time, such as after a collision or due to sensor changes. Robust self-calibration also allows for using the same algorithm on multiple platforms and foregoing the offline calibration routine entirely. The downside of self-calibration is that it considerably increases the dimensionality of the state space while providing no additional measurements. Thus the task of collecting measurements that render the full state-space observable is non-trivial. In practice this means that self-calibrating systems require specific motions that are executed by an expert operator who excites the platform until the desired states have converged to acceptable values, or when no operator is available, hoping the platform will undergo sufficient excitation so as to render the states observable. This problem can be addressed in two ways:

1 **Optimization**: optimize over trajectories in order to generate motions which render observable parameters.
2 **Learning**: determine which motions render the system observable through experiences.

The first option tackles the problem by optimizing over some measure of observability [9], such as the condition number of the linearized system, in order to produce trajectories that maximize the observability of the state space. While appealing in its elegance, this requires knowledge of the motion model of the platform, and can be computationally demanding. The second option, which is the approach taken in this paper, forgoes the process model and uses a model-free reinforcement learning technique to learn which sequence of motions render the desired states observable. The appeal of this approach is that it can be applied to any platform, holonomic or non-holonomic, without the simplifying assumption of a process and environment model. A use case is a non-expert operator needing to calibrate a rig (physical assembly with sensors mounted) with a camera and an IMU (Inertial Measurement Unit). In currently-available calibration libraries [2] the user is tasked with waving the rig around until the calibration parameters are obtained with satisfactory uncertainty. Our approach allows for the system to learn which sequences of motions contain useful segments that render the desired state-space observable. Once the informative sequence of motions is learned, these can be suggested to any user, removing the random "hope-that-this-motion-is-sufficiently-exciting" approach in favor of a series of suggested motions that will obtain the desired calibration parameters deterministically.

Unsurprisingly, self-calibration has received considerable attention in the robotics community, and has proven to be a hard problem due to the slow time-varying nature (drift over time) and inference over naturally occurring data. The first of these problems has been addressed in part by existing self-calibrating algorithms [21], the second gives rise to a series of problems that are less commonly considered:

1 **Observability during normal operation**: normal operation may not render calibration parameters observable if e.g. two cameras do not share an overlapping field of view. In this case, planar motion renders the problem of finding the camera-to-camera extrinsic parameter degenerate.

2 **Parameters that appear observable, but in fact are not were noise not present**: related to the general observability of calibration parameters, it is possible that a solution is numerically obtained even in degenerate cases due to noisy measurements, leading to a physically uninterpretable solution. An example of this is a rank-deficient matrix which is made numerically full-rank due to noise. While a solution is possible, it has no physical meaning. This is rarely addressed in calibration systems.
3 **Which motions will render unobservable dimensions in parameter space observable**: calibration experiments are usually hand-engineered to guarantee that all parameters become observable, especially for platforms for which a process model is not available or desirable, this makes calibration a very tedious and error-prone activity for the average operator, since it is not obvious how the sensor has to be excited.

Existing algorithms handle (1) by hoping sufficiently exciting motions are provided. Some work has been done to address (2) but it is largely unexplored in most calibration systems. To the best of our knowledge no published self-calibration algorithm is able to cope with issue (3) without requiring a process model, some work on this direction has been done by [22] but it is limited to camera intrinsics and our approach does not require the user to follow the given instructions.

In this paper we propose an algorithm to deal with all of the presented difficulties in calibration. Observability is handled by exploiting the link between the Fischer Information matrix and nonlinear observability [11]. Treating numerically unobservable parameters is performed by detecting directions in the parameter space that are unobservable through singular value thresholding of the scaled information matrix and avoiding updating the current state estimate in those directions. Deciding which motions will generate measurements that allow for inference over the desired parameters is done by incorporating reinforcement learning for empirically learning which sequence of motions maximizes the inference of the desired parameters.

The calibration procedure is modeled as a Markov Decision Process (MDP) and Q-learning is employed to learn the optimal action-selection policy. The action-space is discretized into motions that can be easily performed by a human operator, and the state-space is defined as the combination of possible parameter states. For example, the desired final state is when every direction of the parameter space can be inferred, intermediate states are composed of the subsets of the parameter space that may be independently observed—when calibrating camera-to-IMU extrinsics, a set of measurements may render the rotation observable, but not the translation, then another set of measurements may provide information on the translation (see Figure 1). Thus the only requirement of our system is that the sensors be equipped on a platform that is *capable* of performing motions that render the parameters observable, and that the human-operator is able to loosely follow instructions on moving the platform. The main question we aim to answer is: can a MDP with delayed reward regress a convergent policy for the calibration problem? The remainder of this paper will cover related work in Section 2, the mathematical background in Section 3, the theoretical foundation of our method is explained in Section 4, Section 5 val-

idates our approach with simulated and real-world experiments. Finally Section 6 provides a discussion on our findings.

## 2 Related Work

The use of a known calibration pattern such as a checkerboard coupled with non-linear regression has become the most popular method for camera calibration in computer vision during the last decade; it has been deployed both for intrinsic camera calibration [25] and extrinsic calibration between heterogeneous sensors [26]. While being relatively efficient, this procedure still requires expert knowledge to reach a discerning level of accuracy. It can also be quite inconvenient on a mobile platform requiring frequent recalibration (e.g experimental platforms which undergo constant sensor changes). In an effort to automate the process in the context of mobile robotics, several authors have included the calibration problem in a state-space estimation framework, either with filtering [19] or smoothing [14] techniques. Filtering techniques based on the Kalman filter are appealing due to their inherently online nature. However, in case of nonlinear systems, smoothing techniques based on iterative optimization can be superior in terms of accuracy [24].

Our approach does not rely on formal observability analyses to identify degenerate paths of the calibration run as in [3], since these approaches still expect non-degenerate excitations.

A last class of methods relies on an energy function to be minimized. For instance, Levinson and Thrun [17] have defined an energy function based on surfaces and Sheehan et al. [23] on an information theoretic quantity measuring point cloud quality.

Despite considerable work in the field of calibration, very little is known regarding how to efficiently and actively deal with degenerate cases frequently occurring during a calibration routine. The current state of the literature frequently assumes that optimization routines are executed on well-behaved data. As demonstrated in the next sections, this can be a critically flawed assumption in real-world scenarios.

## 3 Problem Formulation

In the following exposition, we borrow the formalism of the probabilistic discrete-time Simultaneous Localization and Mapping (SLAM) model [4]. For the sake of clarity, we consider here a robot with a single camera observing a known number of landmarks at each timestep and a single inertial measurement unit sampling linear acceleration and angular velocity. A front-end inspired by [16] is tasked with establishing correspondences between sensor's measurements and landmarks [1].

[1] The details of this implementation are omitted due to space limitations.

## 4 Methodology

Let $\mathcal{X} = \{\mathbf{x}_{0:K}\}$ be a set of latent random variables (LRV) representing robot states up to timestep $K$, $\mathcal{L} = \{\mathbf{l}_{1:N}\}$ a set of LRV representing $N$ landmark positions, $\mathcal{Z} = \{\mathbf{z}_{1_{1:N}:K_{1:N}}\}$ a set of LRV representing $K \times N$ landmark measurements, and $\Theta$ an LRV representing the calibration parameters of the robot's sensor. The goal of the calibration procedure is to compute the posterior marginal distribution of $\Theta$ given all the measurements up to timestep $K$,

$$p(\Theta|\mathcal{Z}) = \int_{\mathcal{X},\mathcal{L}} p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}). \tag{1}$$

The full joint posterior on the right-hand side may be factorized into:

$$p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}) \propto p(\Theta, \mathbf{x}_0, \mathcal{L}) \prod_{k=1}^{K} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \prod_{k=1}^{K} \prod_{i=1}^{N} p(\mathbf{z}_{k_i}|\mathbf{x}_k, \mathbf{l}_i, \Theta). \tag{2}$$

By making the common assumption that Eq. (2) is normally distributed with mean $\mu_{\Theta\mathcal{X}\mathcal{L}}$ and covariance $\Sigma_{\Theta\mathcal{X}\mathcal{L}}$ we can derive a Maximum a Posteriori (MAP) solution for the mean and covariance,

$$\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = \underset{\Theta\mathcal{X}\mathcal{L}}{\arg\max}\, p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}) = \underset{\Theta\mathcal{X}\mathcal{L}}{\arg\min} -\log p(\Theta, \mathcal{X}, \mathcal{L}|\mathcal{Z}); \tag{3}$$

we further define our model by defining an observation model $\mathbf{z}_{k_i} = \mathbf{g}(\mathbf{x}_k, \mathbf{l}_i, \Theta, \mathbf{n}_k)$ where $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{N}_k)$ is a normally distributed observation noise variable, with known covariance $\mathbf{N}_k$. Given that the observation model is usually nonlinear, such as a camera projection with lens distortion, we resort to a nonlinear least squares method that iteratively solve a linearized version of the problem. We employ the *Gauss-Newton* algorithm for this purpose.

Directly from Eq. (3) and the assumption of normally distributed measurements, we can turn the MAP problem into the minimization of a sum of squared error terms. This is covered in detail in [4], so we will briefly cover only the aspects that are relevant to our approach. The Gauss-Newton method only requires the Jacobian matrix of error terms, $\mathbf{J}$. In block matrix form the update is

$$(\mathbf{J}^T \mathbf{G}^{-1} \mathbf{J}) \delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -\mathbf{J}^T \mathbf{G}^{-1} \mathbf{r}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}), \tag{4}$$

where $\mathbf{G}$ is the error covariance matrix built from diagonal blocks of $\mathbf{N}_k$ and $\mathbf{r}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}})$ is the error evaluated at the current state estimate. At convergence the quantity $\mathbf{J}^T \mathbf{G}^{-1} \mathbf{J}$ is the *Fischer Information Matrix* (FIM) and also the inverse of the estimate covariance matrix, $\hat{\Sigma}_{\Theta\mathcal{X}\mathcal{L}}$. This is a key aspect of this work, since as will be described below, the numerical rank of the FIM provides information about the numerical observability of the parameters for a given batch of data. Specifically for computing the posterior distribution $\Theta$ we use the cost terms and Jacobians for both camera intrinsics and camera-to-IMU extrinsics as defined in [21, 12].

### *4.1 Observability*

There exists a solution to Eq. (4) iff the FIM is invertible, i.e. it is of full rank. The link between the rank of the FIM and observability of the parameters being estimated is well established in [11]. A singular FIM corresponds to some unobservable directions in the parameter space given the current set of observations. Classical observability analysis, for example the method of Hermann and Krener [10], proves structural observability—that there exists some dataset for which the parameters are observable—but it does not guarantee that the parameters are observable for any dataset.

Using singular-value decomposition (SVD) on the FIM we can identify a numerically rank-deficient matrix by analyzing its singular values and consequently the numerical observability of the system [8] [6] [18]. The *numerical rank* $r$ of a matrix is defined as the index of the smallest singular value $\sigma_r$ which is larger than a pre-defined tolerance $\varepsilon$,

$$r = \arg\max_i \sigma_i \geq \varepsilon. \tag{5}$$

It is important to note that the numerical rank corresponds to the algebraic rank of the unperturbed matrix within a neighborhood defined by the parameter $\varepsilon$ proportional to the magnitude of the perturbation matrix, which in this case can be interpreted as the noise affecting the measurements. When the noise affecting the matrix entries has the same scale (by using column or row scaling) then the numerical rank can be determined by the singular values.

Specifically we decompose the error covariance matrix $\mathbf{G}$ from Eq. (4) into its square root form by using Cholesky decomposition, $\mathbf{G}^{-1} = \mathbf{L}^T\mathbf{L}$, we can re-write Eq (4) in standard form:

$$(\mathbf{LJ})^T(\mathbf{LJ})\delta\hat{\mu}_{\Theta\mathcal{XL}} = -(\mathbf{LJ})^T\mathbf{Lr}(\hat{\mu}_{\Theta\mathcal{XL}}), \tag{6}$$

which are the normal equations for the linear system $(\mathbf{LH})\delta\hat{\mu}_{\Theta\mathcal{XL}} = -\mathbf{Lr}(\hat{\mu}_{\Theta\mathcal{XL}})$. Thus we can directly use a *rank-revealing* decomposition to estimate the numerical rank of the FIM, and consequently the numerical observability of the system. Both SVD [7] and QR decomposition [7] are rank-revealing; here we use SVD to demonstrate the method, though we use the more computationally efficient QR decomposition in practice. Let $(\mathbf{LJ})$ be a $m \times n$ matrix with the following SVD decomposition:

$$\mathbf{LJ} = \mathbf{USV^T}, \tag{7}$$

where $\mathbf{U}$ is $m \times n$ and orthogonal, $\mathbf{S} = diag(\sigma_1, ..., \sigma_n)$ the singular values and $\mathbf{V}$ an $n \times n$ matrix, also orthogonal. From Eq. (7) and the orthogonality of $\mathbf{U}$ and $\mathbf{V}$ we can solve (4) as

$$\delta\hat{\mu}_{\Theta\mathcal{XL}} = -\mathbf{VS^{-1}ULr}(\hat{\mu}_{\Theta\mathcal{XL}}) \tag{8}$$

In order to only update the observable directions of the parameter space, we apply the truncated SVD (or truncated QR decomposition) which establishes the rank of the system by analyzing its singular values [8], only using the first $r$ rows of $\mathbf{S}$, as

defined in Eq. (5). This allows us to only update the observable directions of the parameter space and maintain the other directions at their initial value. Establishing the value of $\varepsilon$ to use is specific to the amount of noise expected in the measurements and is treated in Section 5. Using the method described in this section we are able to identify which subset of the calibration parameters are observable, and when the full parameters space is observable, which is a key element of the algorithm described in the following section.

### *4.2 Learning Motions*

The methodology described up to this point allows us to optimize for the calibration parameters while identifying unobservable directions in the parameter space. We will now describe using that result in order to learn which sequence of motions lead to the regression of the full parameter space.

We consider a robot moving through space (or manipulated by an operator) as an *agent* situated in some feature-rich *environment* comprised of both the position of the robot in the environment and the latent calibration variables. The agent can perform certain actions in the environment (e.g. move to a new location, make new measurements). These actions may result in a *reward* (e.g. information on a latent variable). Actions can thus transform the environment (e.g. new configuration of latent variables) and lead to a new state, which the agent can perform another action on, and so forth. The rules for how to chose which action to take given the current state is called a *policy*, which must consider the stochasticity of the environment (e.g. the choice of action, such as moving the robot forward, may or may not obtain

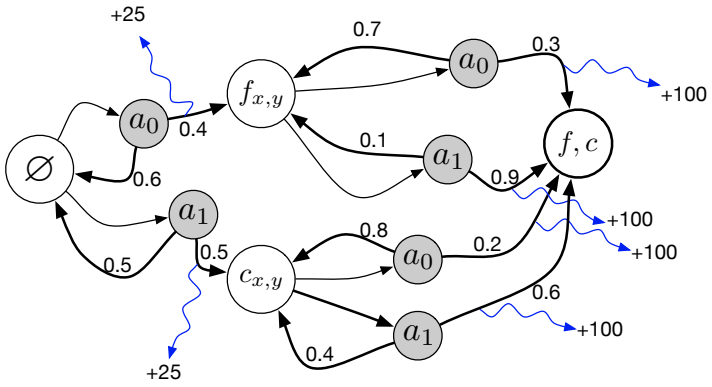



Fig. 2: Example of discretized state space and actions for camera intrinsic calibration (pinhole camera model). Dark circles correspond to possible stochastic actions $\{a_1, a_2\}$. Blue arrows indicate a reward for that transition. We have grouped $\{f_x, f_y\}$ and $\{c_x, c_y\}$ for simplicity. All possible transitions are not shown for readability.

information about a latent variable depending on the structure of the world). The set of states and actions along with the rules for transitioning make up a Markov decision process, and one *episode* of this process corresponds to a sequence of state, action, rewards. The episode ends when the *terminal* state is reached.

The calibration problem can be interpreted as an MDP in which each state represents the regression of each direction of the calibration parameter space (see Figure 2), and the actions are a discretized set of movements that can be performed by the operator. Drawing a parallel to a game, we wish to discover the optimal policy, and therefore sequence of actions, for reaching our final state taking into account not only the immediate reward of an action (i.e. learning about a specific parameter) but the future rewards. To illustrate how this applies to calibration, consider a simplified example: If the robot knows nothing about its camera-to-IMU calibration, but it knows that given a certain movement it will learn the camera-IMU translation along the $x$ direction, and given another movement, it will learn the rotation about the $x$-axis. Note that we assume here that no feasible movement will learn both simultaneously. From an immediate reward standpoint it may seem arbitrary, but by first regressing rotation we are able to reach the full final calibration in fewer movements. We wish to learn the optimal policy for calibration. The state discretization is based on the choice of $\varepsilon$ as described in Section 4.1 which is the threshold that determines the rank deficiency and therefore the observable directions of the parameter space. We have empirically set $\varepsilon = 0.015$, however it does need to be adjusted if we were to use a system with a different noise profile.

Q-Learning is well suited for the class of problem. Briefly, within reinforcement learning, Q-Learning is a model-free technique which can be used to find an optimal action-selection policy for a finite MDP. The basic principle is to maximize the discounted future reward. The discounted aspect is due to the stochastic nature of the environment and this to down-weigh the uncertain future rewards. This method essentially consists of establishing a discrete set of actions (Section 4.2.1) and states, a reward table for state transitions, and a Q-table which contains one row for each state and a column for each possible action, which encodes the "quality" of a certain action in a given state. Given our finite state space and discretized action space we are able to iteratively approximate the Q function using the *Bellman equation*:

$$Q(s,a) = r + \gamma max_{a'} Q(s',a'), \tag{9}$$

Which is the reward for the state transition plus the maximum possible future reward for the next state. The idea behind Q-learning is that we can iteratively converge on the Q-table using the Bellman equation, see Algorithm 1. In practice we also use a learning rate parameter $\alpha = 0.1$ which essentially limits the *step size* for each iteration. Once the Q-table has converged we can simply follow the learned policy $\pi$, given the current state: $\pi(s) = \arg\max_a Q(s,a)$ and suggest that action to the user.

**Algorithm 1:** Observability-aware calibration Q-Learning

**Data:** sensor measurements, $\gamma$, $\alpha$, Reward matrix
**Result:** converged Q-table
Initialize Q-table $[states, actions] \leftarrow 0$;
**while** *Q-table not converged* **do**
  select random initial state;
  **while** *goal state not reached* **do**
    select possible action, $a$, given current state;
    display action $a$ to be carried out to user;
    compute $\delta\hat{\mu}_{\Theta\mathcal{XL}} \leftarrow -\mathbf{VS^{-1}ULr}(\hat{\mu}_{\Theta\mathcal{XL}})$ according to (8);
    compute observable directions according to 5;
    **if** *at least one calibration direction is observable* **then**
      go to corresponding state $s'$ and observe reward $r$;
    **else**
      stay in current state and set $r \leftarrow 0$;
    **end**
    update Q-table: $Q[s,a] \leftarrow Q[s,a] + \alpha\,(r + \gamma max_{a'} Q[s',a'] - Q[s,a])$;
  **end**
**end**

We apply this to both regressing camera intrinsics and extrinsics. Note that the important data association, outlier rejection, residual and Jacobian calculation steps are delegated to a sparse visual-inertial keyframe-based SLAM system which we will not go into detail here. In both intrinsic and extrinsic cases the reward table is straightforward: a reward of 25 is given for non-final state transitions and a reward of 100 is given for any transition to the final state (full calibration). Figure 2 shows a simple example where only two actions are possible, and the blue arrows indicate the reward value for that state transition. These values were obtained empirically. We found that using a constant $\alpha = 0.1$ works well, although experimenting with variable step sizes could lead to quicker convergence. We initialize the Q-table to zero and use $\gamma = 0.9$. We experimented with a variable discount factor [5] but found no practical benefit. The outer loop is executed until the convergence criteria is met. We use a relative change metric to quit the learning process, with a maximum number of iterations $max\_iter = 1000$.

#### 4.2.1 Action Discretization

Given that this work is focused on providing intuitive and simple calibration instructions to a inexperienced operator, we discretize the initially continuous action space (possible movements performed by the rig) into a set of movements that can be easily conveyed and performed by the operator. This consists of translation along one degree of freedom combined with rotation around a single axis. Empirically we found this to be the optimal trade-off for basis motions that are both sufficiently in-

formative and simple enough for the operator to execute. Degenerate motions such as pure rotation or translation were not included. This results in a tractable set of only 18 actions. The motions are also limited to the range of motion of the average human arm, between 60 and 80cm. Given the goal of providing easy-to-understand motion suggestions we chose simpler basis motions which may lead to more actions being performed (i.e. a more complex motion could contain more information than the simple motions suggested here, but would be harder to execute). Figure 3 shows a few learned motions from the discrete set of actions. The motion discretization described requires that the platform execute those motions when training. The training procedure consists of either simulations, as shown in section 5.2 or the the rig being moved around the environment (either by an operator or autonomously) in the latter case we suggest the discretized motions to be executed by the operator.

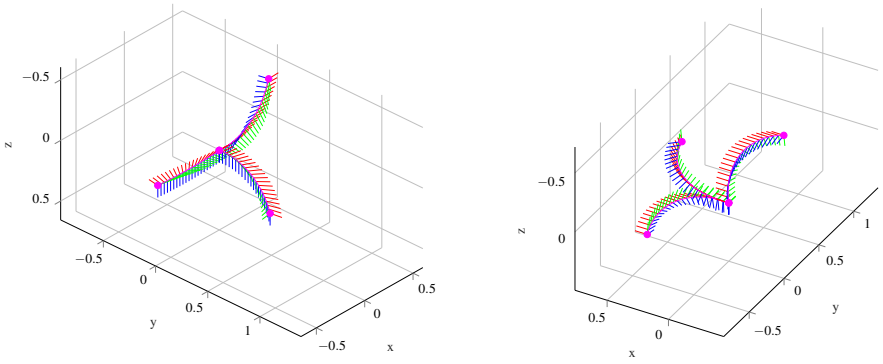



Fig. 3: Left image shows learned policy for regressing camera intrinsics for a radial-tangential distortion model, with a narrow field of view lens ($f = 460$px) and central point in the middle of the image. Changing to a fisheye lens and a much wider field of view ($f = 220$px) changes the learned optimal action sequence (right figure). Each graph depicts the three suggested motions for that camera.

## 5 Experiments

We evaluate the proposed framework on several fronts. First we describe results from our simulated experiments, using synthetic visual and inertial data to train our system and evaluate the performance of calibration with the synthetically trained motions vs. trained with real data. We then demonstrate that the system can be successfully applied to both camera intrinsics and camera-to-IMU extrinsics, and that our method performs equally or better than publicly available calibration tools. Finally we demonstrate that this system can be used for life-long learning by adapting to changes in hardware configuration that impact the calibration routine.

### 5.1 Experimental Setup

The proposed method was implemented in C++ and integrated into our existing sparse keyframe-based visual-inertial SLAM pipeline which uses *BRISK* [15] feature descriptors and ceres-solver [1] as the non-linear solver. In order to evaluate the proposed method, experiments were run on a sensor platform known as "rig." The rig was equipped with a monocular camera and a commercial grade MEMS-based IMU. The camera is equipped with a wide field-of-view lens at $2040 \times 1080$ resolution downsampled to $640 \times 480$ coupled with a MEMS IMU, sampled at 200Hz. The camera captures images at 30 frames per second.

In order to generate simulated measurements visual and inertial data corresponding to each of the 18 basis motions was generated. A differentiable quaternion spline through the SO(3) element of each pose was used to obtain smooth gyroscope measurements, and a cubic spline was run through the euclidean positions for accelerometer measurements. Simulated visual data was generated by creating a virtual world in OpenGL and simulating movement corresponding to the accelerometer measurements. Visual data was captured at 15fps with IMU updates at 100Hz. For each projected feature point from the simulated images ($640 \times 480$ resolution), independent zero-mean Gaussian noise with $\sigma = 0.5$ was added to the $(u,v)$ pixel coordinates. Zero-mean Gaussian noise with $\sigma = 10^{-3}$ was also added to the IMU accelerometer and gyroscope measurements and biases.

### 5.2 Simulations

We generate noisy simulated measurements corresponding to the 18 basis functions and run through the training procedure in Algorithm 1 until the relative change in values for the Q-table is $< 1e^{-3}$.

An experiment was run to determine if different calibration parameters would suggest different motions. To that end we generated synthetic data to regress the Q-table for narrow (radial-tangential distortion) and wide field of view fisheye (fov distortion) camera, which resulted in three considerably distinct set motions (see Figure3). We then used the suggested motions to execute the calibration procedure with simulated measurements for both cameras. Using the suggested motions corresponding to the camera which was used for training yielded on average a 22% better mean and 8% lower variance, suggesting a correlation between the learned motions and the camera distortion model and field of view.

### 5.3 Real-world Data

Experiments with the rig described in Section 5 require an interface for suggesting motions to the user. Due to the simplicity of the motions we resort to a textual rep-

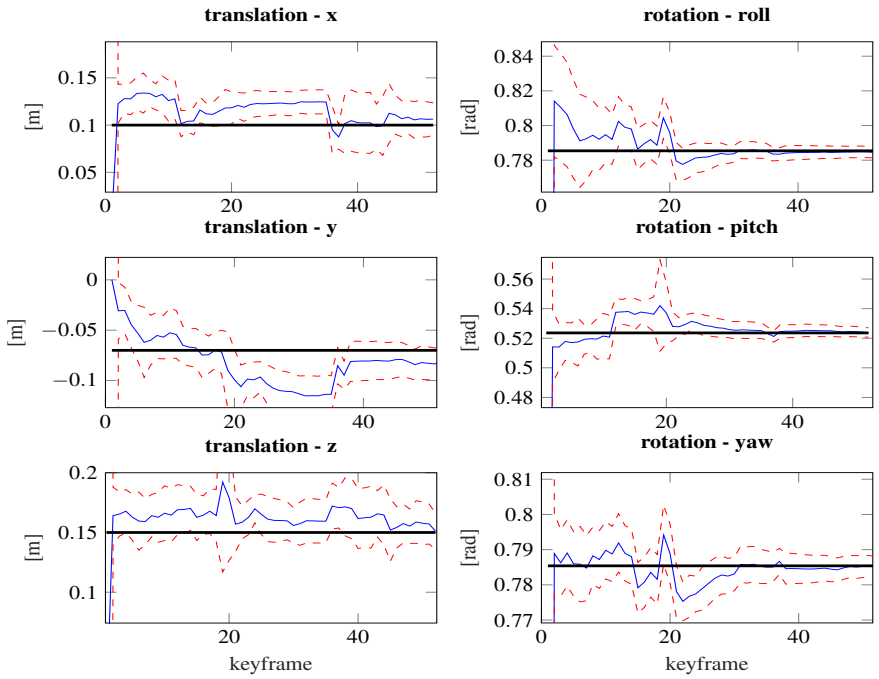



Fig. 4: Convergence of extrinsic parameters by an inexperienced operator following suggested motions. The solid black line represents the ground truth as obtained by a batch optimization with a target-based routine, the blue line represents the estimate as the rig is moved and the dotted red line the $3\sigma$ bounds. The jumps in values correspond to the moment a new segment is processed. A total of four motions were suggested for this calibration run.

resentation, indicating what direction the rig should be moved in and rotation about which axis. A graphical interface with visual feedback is in development. The main objectives of this experiment are to verify that an inexperienced user can use the suggested motions to easily and reliably calibrate a robotic system and obtain lower variance compared to a publicly available standard calibration pipeline. To that end we manually trained algorithm, then calibrated the same camera with a fisheye lens by following the motions suggested by our system and also by waving a calibration target in front of the camera and using vicalib [2], a publicly available calibration library, to optimize over the camera intrinsics. This experiment was repeated 50 times for each calibration method in order to obtain statistically significant results. The users selected to perform the calibration ranged from lab members to random volunteers that did not have experience calibrating cameras. The results can be seen in Figure 6 which clearly show that both methods converge to the same mean, but the distribution is tighter around our method. Furthermore both extrinsic (Figure 4) and intrinsic (Figure 5) calibration using motion suggestions converges to within $3\sigma$

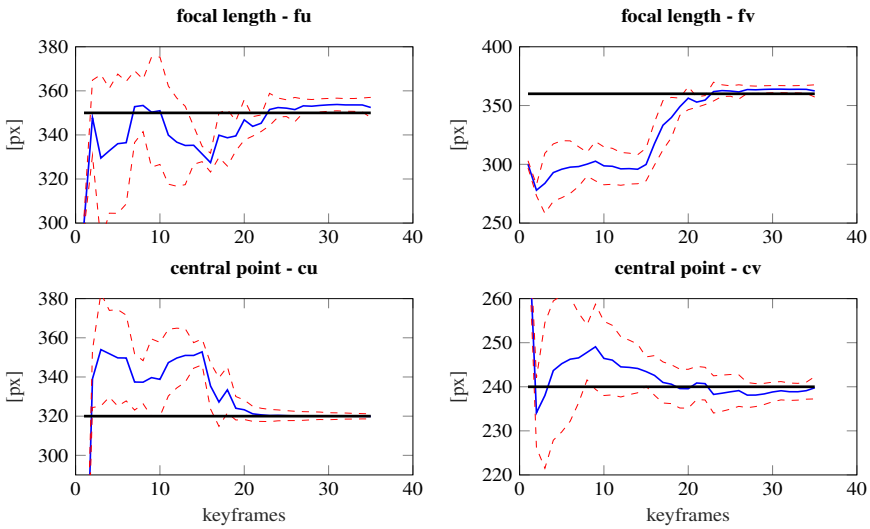



Fig. 5: Convergence of camera intrinsic calibration parameters with real-data by following motion suggestions. Dotted red line is the $3\sigma$ bound, solid black line is the ground truth value.

of the ground truth in as few as 40 keyframes. Finally Figure 1 shows an example of a sequence of motions suggested for extrinsics calibration and the corresponding convergence of translation and rotation. It is important to note that a feature-rich environment assumption is made: there must be enough salient features for visual tracking. This is the most common scenario so we are not sacrificing much by forgoing an analysis on the effects of the structure of the environment on calibration.

## 6 Discussion

In this paper, we have presented a novel calibration tool that uses reinforcement learning to provide live feedback on the state of calibration and produces accurate calibration parameters even when used by inexperienced operators. We have leveraged truncated SVD/QR decomposition to deal with unobservable directions and reinforcement learning for motion suggestions to perform model-free calibration for both camera intrinsics and camera-to-IMU extrinsics. We have evaluated the proposed system in a variety of scenarios and shown that it can be used as a replacement for currently available calibration toolkits. Our principal question was to assess the suitability of reinforcement learning when applied to the calibration problem. We have shown that through the discretization of both the action and state we are able to leverage Q-learning to improve the calibration experience. An argument

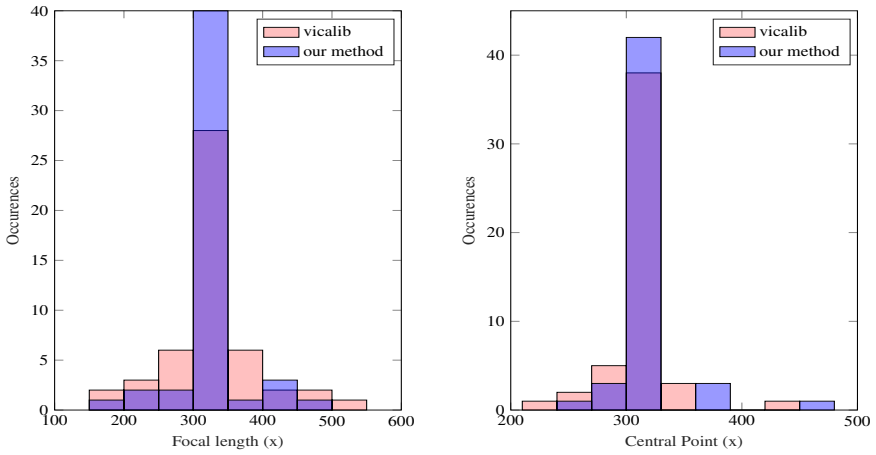



Fig. 6: Distribution of focal lengths and central point for all trials. The mean parameter values between our method and vicalib are similar (focal length: 332.1px for vicalib and 334.8px for our method), the standard deviations are much higher (32.1 vs. 21.2) indicating more consistent and repeatable results.

could be made that a much larger state space would be possible when using a deep network [20] instead of the Q-table, but we argue that for the calibration problem as stated here, there is little practical advantage. A considerable challenge in the proposed framework is designing the human interface so that the user will follow the instructed motions. This is an area in which warrants further development, however in the projected fully autonomous mode where motion suggestions are the input to a controller, the human interface can be removed from the pipeline. An interesting result that lends itself to long term autonomy is the ability to capture different sensor configurations. As shown in Figure 3 having radically different calibration parameters results in a different set of optimal motions, reinforcing the point that the "SLAM wobble" or any pre-determined calibration maneuver is not guaranteed to provide acceptable parameter inference. If a robot is expected to calibrate itself autonomously it cannot have a pre-determined routine or hope that random navigation will render its calibration parameters observable. A system that is robust to sensor configuration changes by re-learning how to calibrate itself is a step in the direction of both "power-on-and-go" robotics and long term autonomy.

## References

1. Agarwal, S., Mierle, K., Others: Ceres solver. `http://ceres-solver.org`
2. Autonomous Robotics and Perception Group (ARPG): VICalib visual-inertial calibration suite. `https://github.com/arpg/vicalib` (2016)

3. Brookshire, J., Teller, S.: Extrinsic calibration from per-sensor egomotion. Robotics: Science and Systems VIII, Jul pp. 504–512 (2013)
4. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part i. IEEE robotics & automation magazine **13**(2), 99–110 (2006)
5. François-Lavet, V., Fonteneau, R., Ernst, D.: How to discount deep reinforcement learning: Towards new dynamic strategies. arXiv preprint arXiv:1512.02011 (2015)
6. Gibbs, B.P.: Advanced Kalman filtering, least-squares and modeling: a practical handbook. John Wiley & Sons (2011)
7. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU Press (2012)
8. Hansen, P.C.: Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion. SIAM (1998)
9. Hausman, K., Preiss, J., Sukhatme, G.S., Weiss, S.: Observability-Aware Trajectory Optimization for Self-Calibration with Application to UAVs. arXiv.org (2016)
10. Hermann, R., Krener, A.: Nonlinear controllability and observability. IEEE Transactions on automatic control **22**(5), 728–740 (1977)
11. Jauffret, C.: Observability and fisher information matrix in nonlinear regression. IEEE Transactions on Aerospace and Electronic Systems **43**(2) (2007)
12. Keivan, N., Sibley, G.: Constant-time monocular self-calibration. Robotics and Biomimetics (ROBIO) pp. 1590–1595 (2014)
13. Kelly, J., Sukhatme, G.S.: Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. The International Journal of Robotics Research **30**(1), 56–79 (2011)
14. Kümmerle, R., Grisetti, G., Burgard, W.: Simultaneous calibration, localization, and mapping. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 3716–3721. IEEE (2011)
15. Leutenegger, S., Chli, M., Siegwart, R.Y.: Brisk: Binary robust invariant scalable keypoints. In: Computer Vision (ICCV), 2011 IEEE International Conference on, pp. 2548–2555. IEEE (2011)
16. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research **34**(3), 314–334 (2015)
17. Levinson, J., Thrun, S.: Unsupervised calibration for multi-beam lasers. In: Experimental Robotics, pp. 179–193. Springer (2014)
18. Low, K.H.: Industrial robotics: programming, simulation and applications. I-Tech (2007)
19. Martinelli, A., Scaramuzza, D., Siegwart, R.: Automatic self-calibration of a vision system during robot motion. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pp. 43–48. IEEE (2006)
20. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
21. Nobre, F., Heckman, C., Sibley, G.: Multi-sensor slam with online self-calibration and change detection. In: International Symposium on Experimental Robotics (ISER). (2016)
22. Richardson, A., Strom, J., Olson, E.: Aprilcal: Assisted and repeatable camera calibration. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 1814–1821. IEEE (2013)
23. Sheehan, M., Harrison, A., Newman, P.: Self-calibration for a 3d laser. The International Journal of Robotics Research **31**(5), 675–687 (2012)
24. Strasdat, H., Montiel, J., Davison, A.J.: Real-time monocular slam: Why filter? In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 2657–2664. IEEE (2010)
25. Sturm, P.F., Maybank, S.J.: On plane-based camera calibration: A general algorithm, singularities, applications. In: Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., vol. 1, pp. 432–437. IEEE (1999)
26. Zhang, Q., Pless, R.: Extrinsic calibration of a camera and laser range finder (improves camera calibration). In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3, pp. 2301–2306. IEEE (2004)