ISRR 2017

The International Journal of
Robotics Research
1–15
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364919844824
journals.sagepub.com/home/ijr
SAGE

Learning to calibrate: Reinforcement learning for guided calibration of visual–inertial rigs

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 in the context of simultaneous localization and mapping. 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.

Keywords
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 parameters 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 (Kelly and Sukhatme, 2011). While there has been work on determining the observability of different motions (Kelly and Sukhatme, 2011), 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.

In the context of life-long autonomous operation, self-calibrating systems are a necessity because they allow for compensation of errors induced over time, such as after a collision or due to sensor changes. Of particular challenge are small changes in calibration parameters, which are both challenging to observe owing to their effects only on certain degrees of freedom, but also potentially disastrous owing to their contribution to accumulated drift over time. Robust self-calibration also allows the same algorithm to be used 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

Autonomous Robotics and Perception Group, University of Colorado, Boulder, CO, USA

Corresponding author:
Christoffer Heckman, Autonomous Robotics and Perception Group, University of Colorado, 1111 Engineering Drive, UCB 430, ECOT 717, Boulder, CO 80309, USA.
Email: christoffer.heckman@colorado.edu

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 that render parameters observable.
2. **Learning**: determine which motions render the system observable through experiences.

The first option tackles the problem by optimizing over some measure of observability (Hausman et al., 2016; Preiss et al., 2017), 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. Indeed this approach is still regressing parameters for which some model exists in the sensor rig itself, but this does not require a process model for any motions that the rig might have available to it in order to regress these calibration parameters. Furthermore, even in the event that there is a motion model available, such models can be limited in their expressibility when considering notions of "preference" or other factors that can be difficult to encode.

The use case under consideration in this work is a non-expert operator needing to calibrate a rig (physical assembly with sensors mounted) with a camera and an inertial measurement unit (IMU). In currently available calibration libraries (Heckman et al., 2014), the user is tasked with waving the rig around until the calibration parameters are obtained with satisfactory uncertainty. Our approach allows 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 owing 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 (Nobre et al., 2016), 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, for example, 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 owing to noisy measurements, leading to a physically uninterpretable solution. An example of this is a rank-deficient matrix, which is made numerically full-rank owing to noise. Although 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, because 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 the authors' knowledge no published self-calibration algorithm is able to cope with issue (3) without requiring a process model, some work in this direction has been done by Richardson et al. (2013) 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 Fisher information matrix (FIM) and nonlinear observability (Jauffret, 2007). 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. This is not done without cost, however; the approach discussed in this work is rig-specific, so the addition of new sensors or of new motions that can be imparted on the rig requires new data be collected. Even so, the data collected for these methods to succeed may be reused over and over and shared among similar rigs to allow them to be calibrated based on historical data.

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? Nobre and Heckman (2017) considered and evaluated an initial investigation into this problem by means of comparison with an inexperienced operator attempting to calibrate a visual–inertial sensor rig. In this work, a detailed formulation of the methodology has been provided, including discussion of parameter observability and its implications on parameter estimation. As opposed to Nobre and Heckman (2017), we have explicitly embedded the problem into the formalism of visual–inertial simultaneous localization and mapping (SLAM). This work also includes an extended experimental validation of the method, both in physical experiments as well as in simulation from various initial conditions of parameters and over different parameter sets than were originally considered. This new formulation estimates the intrinsic and extrinsic parameters of a visual–inertial rig, whereas previous work only estimated the extrinsic parameters. Finally, our discussion and conclusions have been expanded significantly, including directions for future work and conclusions regarding transfer learning.

2. Related Work

The use of a known calibration pattern such as a checkerboard coupled with nonlinear 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 (Sturm and Maybank, 1999) and extrinsic calibration between heterogeneous sensors (Zhang and Pless, 2004). 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 that 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 (Martinelli et al., 2006) or smoothing (Kümmerle et al., 2011) techniques. Filtering techniques based on the Kalman filter are appealing owing to their inherently online nature. However, in the case of nonlinear systems, smoothing techniques based on iterative optimization can be superior in terms of accuracy (Strasdat et al., 2010).

Our approach does not rely on formal observability analyses to identify degenerate paths of the calibration run as in Brookshire and Teller (2013), because these approaches still expect non-degenerate excitations.

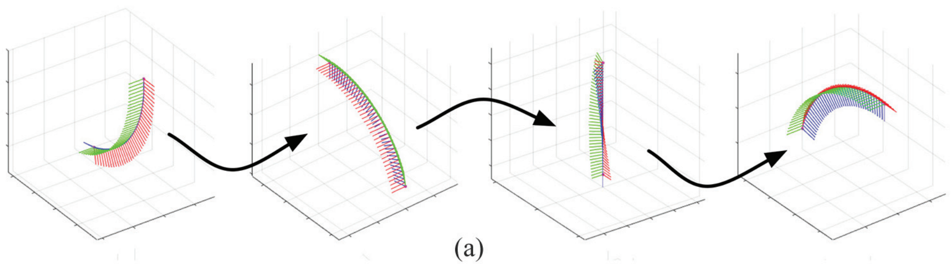
A final class of methods relies on an energy function to be minimized. For instance, Levinson and Thrun (2014) have defined an energy function based on surfaces and Sheehan et al. (2012) on an information-theoretic quantity measuring point cloud quality. Hausman et al. (2016) proposed an observability-aware trajectory optimization for quadcopters that was extended by Preiss et al. (2017): both of these approaches require an analytical motion model and do not explicitly handle non-holonomic platforms.

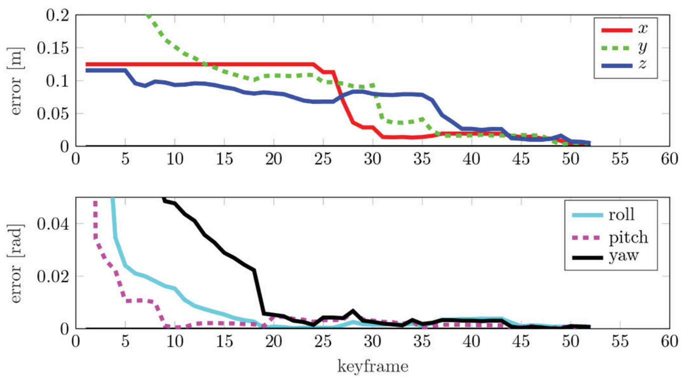
MDPs and partially-observable Markov decision processes (POMDPs) have been used extensively in the context of robotics (Grady et al., 2015). Winterer et al. (2017) used POMDPs for motion planning and Chatterjee et al. (2015) used POMDPs for qualitative analysis; however, to the best of the authors' knowledge, the sensor calibration problem has not been cast as a MDP or POMDP. The work that is most similar to ours is in the *active learning* field by Jaulmes et al. (2005). POMDPs more accurately describe the problem, given the inherent noise in sensor measurements; however, we explore the feasibility of leveraging observability properties of cost surfaces specific to the SLAM sensor calibration problem to make the single belief state assumption and benefit from the simplicity and efficiency of a MDP.

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 following 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 SLAM model (Durrant-Whyte and Bailey, 2006) when discussing measurements, queues, and motion samples, and therefore is limited to SLAM and may only conceivably be extended to other applications with other definitions of states with different sorts of measurements. 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 IMU sampling linear acceleration and angular velocity. A front-end


(a)



(b)

Fig. 1. (a) Schematic of suggested motions in sequence; for instance, a demonstration of the motions swooping upward and downward, twisting or turning the camera downward. Motions suggested to an inexperienced operator for camera-to-IMU extrinsic calibration, drawn from the learned policy. (b) Extrinsics calibration with suggested motions as keyframes are added to the visual–inertial calibration optimization set. 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.

inspired by Leutenegger et al. (2015) is tasked with establishing correspondences between sensor's measurements and landmarks, which is explained in Section 4.

4. Methodology

Let $\mathcal{X}=\{\mathbf{x}_{0:K}\}$ be a set of latent random variables (LRVs) representing robot states up to timestep K, $\mathcal{L}=\{\mathbf{l}_{1:N}\}$ a set of LRVs 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 Θ a LRV representing the calibration parameters of the robot's sensor. The goal of the calibration procedure is to compute the posterior marginal distribution of Θ 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 (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,

$$\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}=\underset{\Theta\mathcal{X}\mathcal{L}}{\operatorname{argmax}}\,p(\Theta,\mathcal{X},\mathcal{L}|\mathcal{Z})=\underset{\Theta\mathcal{X}\mathcal{L}}{\operatorname{argmin}}-\log p(\Theta,\mathcal{X},\mathcal{L}|\mathcal{Z}) \tag{3}$$

we further define our model by specifying 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 solves a linearized version of the problem. We employ the *Gauss–Newton* algorithm for this purpose.

Directly from (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 Durrant-Whyte and Bailey (2006), 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

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

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 $\mathbf{J}^{\mathrm{T}}\mathbf{G}^{-1}\mathbf{J}$ is the 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, because, as described in the following, the numerical rank of the FIM provides information about the numerical observability of the parameters for a given batch of data.

A keyframe-based (Klein and Murray, 2007) pose-and-landmark nonlinear maximum likelihood estimation is performed for real-time updates on landmarks. To compute the posterior distribution Θ we use the cost terms and Jacobians for both camera intrinsics and camera-to-IMU extrinsics (Nobre et al., 2016). In this formulation, the visual and inertial residuals are partitioned and summed to form the full residual. The calibration parameters, including time-varying IMU biases, are also estimated alongside the pose and landmark parameters on the informative segments of the trajectory. The complete state vector is

$$\boldsymbol{X} = \left[\left\{ \boldsymbol{x}_{wp_n} \quad \boldsymbol{v}_{w_n} \quad \boldsymbol{b}_{\boldsymbol{g}_n} \quad \boldsymbol{b}_{\boldsymbol{a}_n} \right\} \quad \{\rho_k\} \quad \{\Theta\} \right]^{\mathrm{T}} \tag{5}$$

where $\boldsymbol{x}_{wp_n} \in \mathrm{SE}(3)$ is the transformation from the coordinates of the nth keyframe to world coordinates, $\boldsymbol{v}_{w_n} \in \mathbb{R}^3$ is the velocity vector of the nth keyframe in world coordinates, and $\boldsymbol{b}_{\boldsymbol{g}_n} \in \mathbb{R}^3$ and $\boldsymbol{b}_{\boldsymbol{a}_n} \in \mathbb{R}^3$ are the gyroscope and accelerometer bias parameters for the nth keyframe, respectively. Here $\{\rho_k\}$ is the one-dimensional inverse-depth (Civera et al., 2008) parameter for the kth landmark and $\{\Theta\}$ are the calibration parameters. Orientation of gravity may be calculated by regressing a resultant between the axes of the accelerometer that approximate the magnitude of gravity, after which the gravity vector is deemed fixed (Mur-Artal and Tardós, 2017). Note that $\mathbf{x}_{wp_n}$ has six degrees of freedom: three for translation and three for rotation. To avoid singularities arising from a minimal representation (e.g. using Euler angles), the rotation component of the transformation is represented as a quaternion, with the optimization lifted to the tangent space (at the identity) of the SO(3) manifold.

Measurements are formed by tracking image keypoints across frames. A landmark parameterized by inverse depth is projected onto an image forming a projected pixel coordinate $\boldsymbol{p}_{proj}$ that is formulated via a transfer function $\boldsymbol{T}$ as follows:

$$\begin{aligned} \boldsymbol{p}_{proj} &= T\left(\boldsymbol{p}_r, \boldsymbol{T}_{wp_m}, \boldsymbol{T}_{wp_r}\boldsymbol{T}_{pc}, \rho\right) \\ &= \mathcal{P}(\boldsymbol{T}_{pc}^{-1}\boldsymbol{T}_{wp_m}^{-1}\boldsymbol{T}_{wp_r}\boldsymbol{T}_{pc}\mathcal{P}^{-1}(\boldsymbol{p}_r, \rho)) \end{aligned} \tag{6}$$

where ρ is the inverse depth of the landmark, $\boldsymbol{T}_{wp_r}$ is the transformation from the coordinates of the reference keyframe (in which the landmark was first seen and initialized) to world coordinates, $\boldsymbol{T}_{wp_m}$ is the transformation from the measurement keyframe to world coordinates, $\boldsymbol{p}_r$ is the 2D image location where the original feature was initialized in the reference keyframe, $\boldsymbol{p}_m$ is the measured 2D image location in the measurement keyframe, $\boldsymbol{T}_{pc}$ is the transformation from the camera to the keyframe coordinates, $\mathcal{P}^{-1}$ is the 2D to 3D back-projection function, and $\mathcal{P}$ is the 3D to 2D camera projection function that returns the predicted 2D image coordinates. The camera-to-keyframe transformation $\boldsymbol{T}_{pc}$ is non-identity as the keyframe is collocated on the inertial frame (the frame in which inertial measurements are made), to simplify the inertial integration. Here $\boldsymbol{T}_{pc}$ is the calibration parameter we have interest in estimating. The usual approach is to assume Gaussian noise and minimize a nonlinear least-squares problem with the following residual function:

$$r_{\mathcal{V}_{m,k}} = \| \boldsymbol{e}_{\mathcal{V}_{m,k}} \|^2_{\Sigma_{\boldsymbol{p}_{m,k}}} = \| \boldsymbol{p}_{m,k} - \boldsymbol{p}_{proj} \|^2_{\Sigma_{\boldsymbol{p}_{m,k}}} \tag{7}$$

where $\boldsymbol{p}_{m,k}$ is the measured 2D image location of the kth landmark in the mth keyframe with covariance $\boldsymbol{\Sigma}_{\boldsymbol{p}_{m,k}}$.

4.1. Nonlinear least-squares

Based on the assumptions laid out in Section 4, the joint posterior model of (3) over poses $\mathcal{X}$, landmarks $\mathcal{L}$, and calibration parameters Θ may be resolved through nonlinear optimization. Equation (4) expresses the update to the parameter set provided by the calibration procedure including those updates to the calibration parameters themselves, as regressed using the Gauss–Newton method for minimizing the sum of squares. In practice, this procedure is often relegated to optimization software frameworks such as Ceres solver (Agarwal et al., 2018), which are capable of seamlessly applying robust norms and analytical automatic differentiation provided the cost functions targeted for minimization are compatible with templates known as jet types.

4.2. Observability

There exists a solution to (4) if and only if the FIM is invertible, i.e. it is of full rank. In the context of this work, the FIM is approximated by the inverse Hessian under the Gauss–Newton approximation of the Hessian with a Gaussian measurement assumption. An intuitive way of interpreting the FIM is looking at the Cramér–Rao lower bound, which allows us to determine that the precision to which we can estimate some parameter Θ is determined by the Fisher information of the likelihood function. As such it can be seen as a *precision* matrix for the MAP estimator. The link between the rank of the FIM and observability of the parameters being estimated is well established in Jauffret (2007). 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 (1977), 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 calibration dataset collected.

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 (Gibbs, 2011; Hansen, 1998; Low, 2007). The *numerical rankr* of a matrix is defined as the index of the smallest singular value σ_r that is larger than a pre-defined tolerance ϵ,

$$r = \underset{i}{\operatorname{argmax}} \sigma_i \geqslant \epsilon \tag{8}$$

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 ϵ 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. The scaling matrix S can be computed as

$$S = \operatorname{diag}\left\{ \frac{1}{||J(:,1)||}, \ldots, \frac{1}{||J(:,n)||} \right\} \tag{9}$$

where $||J(:,n)||$ denotes the column norm of the Jacobian matrix, for column n.

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

$$(\mathbf{LJ})^{\mathrm{T}}(\mathbf{LJ})\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -(\mathbf{LJ})^{\mathrm{T}}\mathbf{Lr}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}) \tag{10}$$

which are the normal equations for the linear system $(\mathbf{LH})\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -\mathbf{Lr}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}})$. 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 (Golub and Van Loan, 2012) and QR decomposition (Golub and Van Loan, 2012) 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:

$$\mathbf{LJ} = \mathbf{USV}^{\mathrm{T}} \tag{11}$$

where $\mathbf{U}$ is $m \times n$ and orthogonal, $\mathbf{S} = diag(\sigma_1, \ldots, \sigma_n)$ the singular values, and $\mathbf{V}$ an $n \times n$ matrix, also orthogonal. From (11) and the orthogonality of $\mathbf{U}$ and $\mathbf{V}$ we can solve (4) as

$$\delta\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}} = -\mathbf{VS^{-1}ULr}(\hat{\mu}_{\Theta\mathcal{X}\mathcal{L}}) \tag{12}$$

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 (Hansen, 1998), only using the first r rows of $\mathbf{S}$, as defined in (8). Specifically, according to Hansen (1998), we can efficiently obtain the update as

$$\delta\hat{\Theta} = \sum_{i=1}^{r_\epsilon} \frac{\mathbf{u}_i^{\mathrm{T}}\mathbf{r}_\Theta}{\mathrm{s}_i}\mathbf{v}_i, \tag{13}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the column vectors of $\mathbf{U}$ and $\mathbf{V}$. 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 ϵ 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.

4.3. 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 now describe using that result in order to learn which sequence of motions leads 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* composed 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 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 MDP, 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 a POMDP in which the (partially observable) states are the calibration parameters, actions to be taken are any specified actions of motion that can be imparted on the rig (see, e.g., Figure 1 for a few examples), and transitions between states are the imparted observability of a calibration parameter as a result of an action.

To give an example, Figure 2 gives as the parameters to be estimated the camera focal length and central point, although we also consider other parameters in this work to give further results on evaluated performance. The only *observable* quantities in this formulation are the SLAM

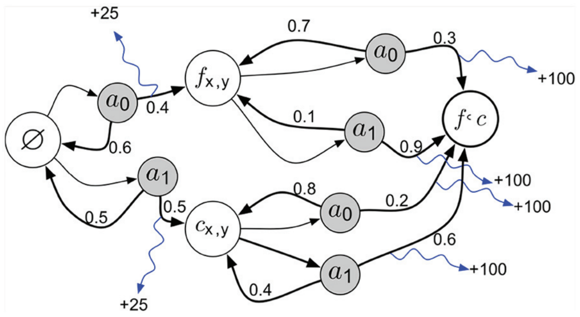


Fig. 2. Example of discretized state space and actions for camera intrinsic calibration (pinhole camera model). Dark circles correspond to possible stochastic actions $\{a_1, a_2\}$. Blue arrows indicate a reward for that transition. We have grouped $\{f_x, f_y\}$ and $\{c_x, c_y\}$, focal length and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

parameter updates, not the calibration parameters' observabilities. Given the inherent measurement noise, the states may be modeled as *belief states*, which implicitly encode the covariance on the state. The agent only has access to the output of the estimator, so information in this problem is strictly limited. At the outset, the state space of this problem would include the state vector in (5), actions would include some "control inputs" on the rig $\boldsymbol{u}_i$ which together with a process model of the rig's rigid-body motion $\boldsymbol{x}_{wp_{i+1}} = p(\boldsymbol{x}_{wp_i}, \boldsymbol{u}_i)$ would provide the next pose of the rig, and belief states would be only those which can be estimated: $\hat{\boldsymbol{X}}$. Addressing the full POMDP formulation of this problem would require simultaneously estimating the correct actions $\boldsymbol{u}_i$ to apply to the rig such that the calibration parameters Θ are rendered observable. However, this is challenging for many reasons, the primary one being that the so-called control inputs $\boldsymbol{u}_i$ are generally exerted by a human operator's moving a rig manually. It would be highly appealing to instead write the actions in terms of sequences of poses of the rig, assuming we can apply those faithfully. In our treatment of this problem, we make a set of simplifying assumptions that allow the problem to be represented as a MDP rather than a full POMDP. First, we assume full state observability in the context of intrinsic and extrinsics sensor calibration in SLAM. By using a conservative value of ϵ as described in Section 4.2, we define the states to represent *sufficient* observability of each individual direction in the parameter space. Here *sufficient* is implicitly encoded by ϵ. In this MDP formulation, we define

$$
\begin{aligned}
s &= \hat{\Theta} \\
a &= \{\boldsymbol{x}_{wp_i}, \ldots, \boldsymbol{x}_{wp_{i+k}}\} \\
&\quad \text{(rig motions from } i \text{ to } i+k) \\
r &= +R \text{ if } \operatorname{cov}(\hat{\Theta}_j) < \epsilon
\end{aligned}
$$

to be the approximate MDP for this problem, where $\hat{\Theta}$ is the *estimated* calibration parameter vector and $\{\boldsymbol{x}_{wp_i}, \ldots, \boldsymbol{x}_{wp_{i+k}}\}$ is the actual sequence of transformations imbued on the rig. Note that, in practice, we assume these sequences are drawn from a prototypical sequence of actions rather than an actual sequence of actions, as will be described in Section 4.3. The $+R$ reward is a tunable parameter based on the particular parameters to be estimated. The MDP also has a transition model function $P(s'|a, s)$ that gives the probability that a new state s' will be obtained from state s based on action a; this is, in general, unknown for a particular calibration problem (Russell and Norvig, 2016).

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 also 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 ε as described in Section 4.2, which is the threshold that determines the rank deficiency and therefore the observable directions of the parameter space. We have empirically set $\epsilon = 0.015$, however it does need to be adjusted if we were to use a system with a different noise profile. This further allows us to have a coarser state representation where varying degrees of uncertainty are not captured explicitly.

Q-learning is well suited for the class of problem. Over the past several years, *Q*-learning and its extensions which rely on deep neural networks has been applied to a wide variety of problems (Arulkumaran et al., 2017) from control theory and perception in robotics (Kober et al., 2013) to more general artificial-intelligence-type problems such as playing Atari games (Mnih et al., 2015). Although *Q*-learning is extraordinarily well-researched and applied broadly in some areas, it is not widely applied to robotic calibration problems. Owing to the possible unfamiliarity of the reader with this technique, we describe the general framework for this method here. Briefly, within reinforcement learning, *Q*-learning is a model-free technique that 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 owing to the stochastic nature of the environment and this to down-weigh the uncertain future rewards. This method essentially consists

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 (calibration parameters);
    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 (12);
        compute observable directions according to 8;
        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
        TD-learning update to $Q$-table:
            $Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma max_{a'} Q[s',a'] - Q[s,a])$
    end
end
```

of establishing a discrete set of actions (see Section 4.3.1) and states, a reward table for state transitions, and a Q-table that 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 can learn the optimal actions without knowing the model $P(s'|a,s)$ because we can assert that the following constraint equation must hold at equilibrium when the Q-values are correct:

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a') \qquad (14)$$

which is the reward for the state $R(s)$ plus the maximum possible future reward for the next state s' over possible actions a'. In this setting, temporal difference (TD) learning may be applied until convergence of the Q-function as described in Algorithm 1. The idea behind Q-learning is that we can stochastically converge on the Q-table; 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 π, given the current state: $\pi(s) = \text{argmax}_a Q(s,a)$ and suggest that action to the user.

We apply this to both regressing camera intrinsics and extrinsics between the camera and the IMU. Note that the important data association, outlier rejection, residual, and Jacobian calculation steps are delegated to a sparse visual–inertial keyframe-based SLAM system and we do 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). The design of reward functions is an area of intense research (Hussein et al., 2017) and is largely dependent on the application; in this work, reward values were chosen through manual tuning, a method that was available owing to the fact that the number of states is manageable through discrete Q-learning. Figure 1 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 (François-Lavet et al., 2015) 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 $max\_iter = 1,000$.

4.3.1 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 informative 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 80 cm. 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 explained in Section 5.2 or of 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.

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 versus 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

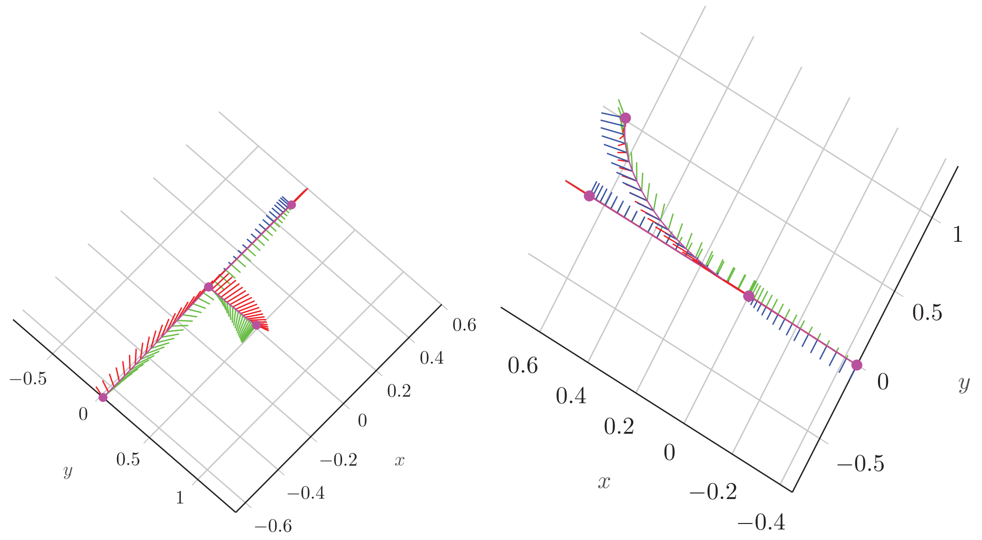


Fig. 3. Top image shows learned policy for regressing camera intrinsics for a radial–tangential distortion model, with a narrow field of view lens ($f = 460$ pixels) and central point in the middle of the image. Changing to a fisheye lens and a much wider field of view ($f = 220$ pixels) changes the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

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* (Leutenegger et al., 2011) feature descriptors and Ceres solver (Agarwal et al., 2018) as the nonlinear solver. In order to evaluate the proposed method, experiments were run on a sensor platform known as a “rig.” The rig was equipped with a monocular camera and a commercial grade microelectromechanical system (MEMS)-based IMU. The camera is equipped with a wide field-of-view lens at $2,040 \times 1,080$ resolution downsampled to 640×480 coupled with a MEMS IMU, sampled at 200 Hz. 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 were generated by creating a virtual world in OpenGL and simulating movement corresponding to the accelerometer measurements. Visual data were captured at 15 fps with IMU updates at 100 Hz. For each projected feature point from the simulated images (640×480 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 $< 10^{-3}$. The dataset generation was conducted using the simulation environment from Nobre et al. (2017) in which a dense 3D world of visual data is projected into a simulated camera and simulated accelerometer and gyroscope data are backed out to support the path traversed by the camera.

This simulated environment allowed for the rapid examination of different calibration objectives while keeping in comparison the ground-truth values for the calibration parameters. An experiment was run to determine whether 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 (field-of-view distortion) camera, which resulted in three considerably distinct set motions (see Figure 3). 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.

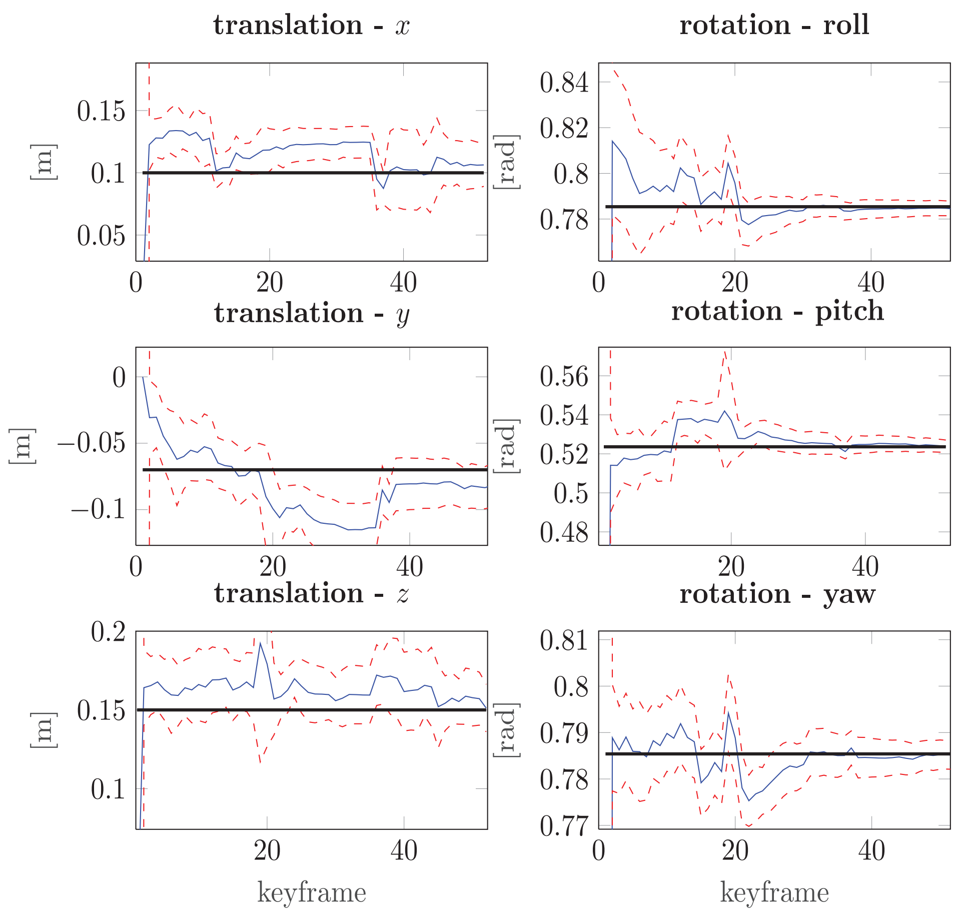


Fig. 4. Convergence of extrinsic parameters by an inexperienced operator following suggested motions. The solid black line represents the ground truth as obtained by a batch optimization with a target-based routine, the blue line represents the estimate as the rig is moved, and the dotted red line the 3σ 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.

5.3. Real-world data

Experiments with the rig described in Section 5 require an interface for suggesting motions to the user. Owing to the simplicity of the motions we resort to a textual representation, 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 with a publicly available standard calibration pipeline. To that end, we first train the algorithm's reinforcement learning framework offline, then calibrate the same camera with a fisheye lens by following the motions suggested by our system. We then also calibrate the rig by waving it in front of a calibration target (see Figure 7) and using Heckman et al. (2014), a publicly available calibration library, to optimize over the camera intrinsics.

Heckman et al. (2014) also uses the keyframe-based approach described in Section 4; different from Nobre et al. (2016), however, a calibration target is leveraged. The target consists of a sequence of 4×4 patterns of small and large shaded circles which are printed on a rigid board against a white background and printed at a fixed scale; a resized version is shown in Figure 7. The 4×4 patterns are unique in the grid, thereby allowing for partial occlusion across the target while still providing a unique pose estimate. The calibration procedure then evolves first by estimating intrinsics of the rig, then estimating extrinsics, over the course of a calibration. To determine the target's location with respect to the camera, an ellipse finder is applied on a thresholded camera image with at least partial observability of the target and centers of the ellipses and their radii are estimated. These quantities determine which 4×4 pattern is viewed as well as its relative location to the camera. Random sample consensus (RANSAC) is then used to solve the perspective n-point (PNP) problem and determine the estimated location of the target conditioned on the current guess at calibration parameters. This process first estimates the intrinsic parameters, then the extrinsics.

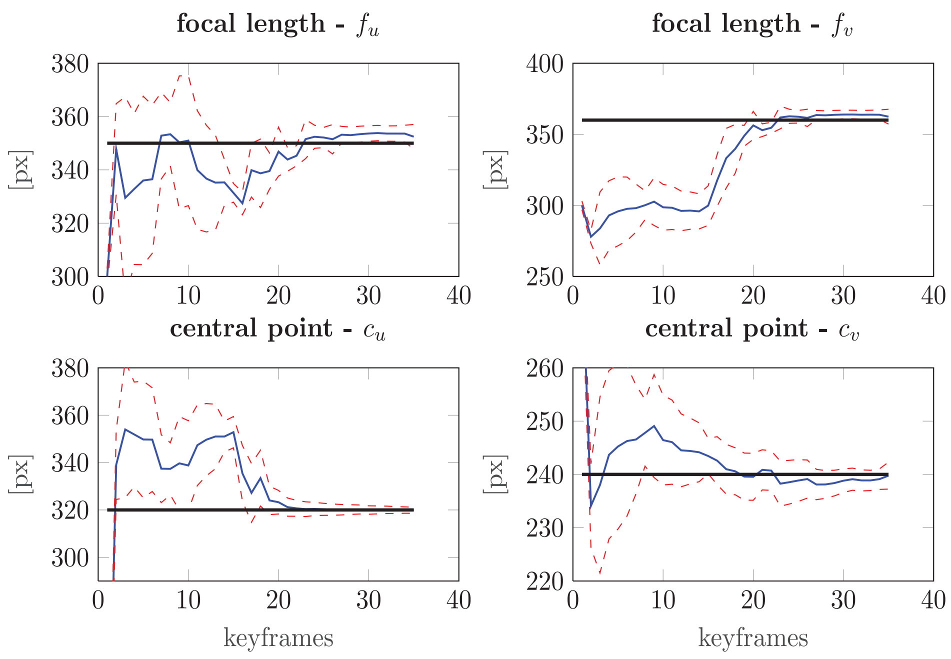


Fig. 5. Convergence of camera intrinsic calibration parameters with real data by following motion suggestions. The dotted red line is the 3σ bound, the solid black line is the ground-truth value.

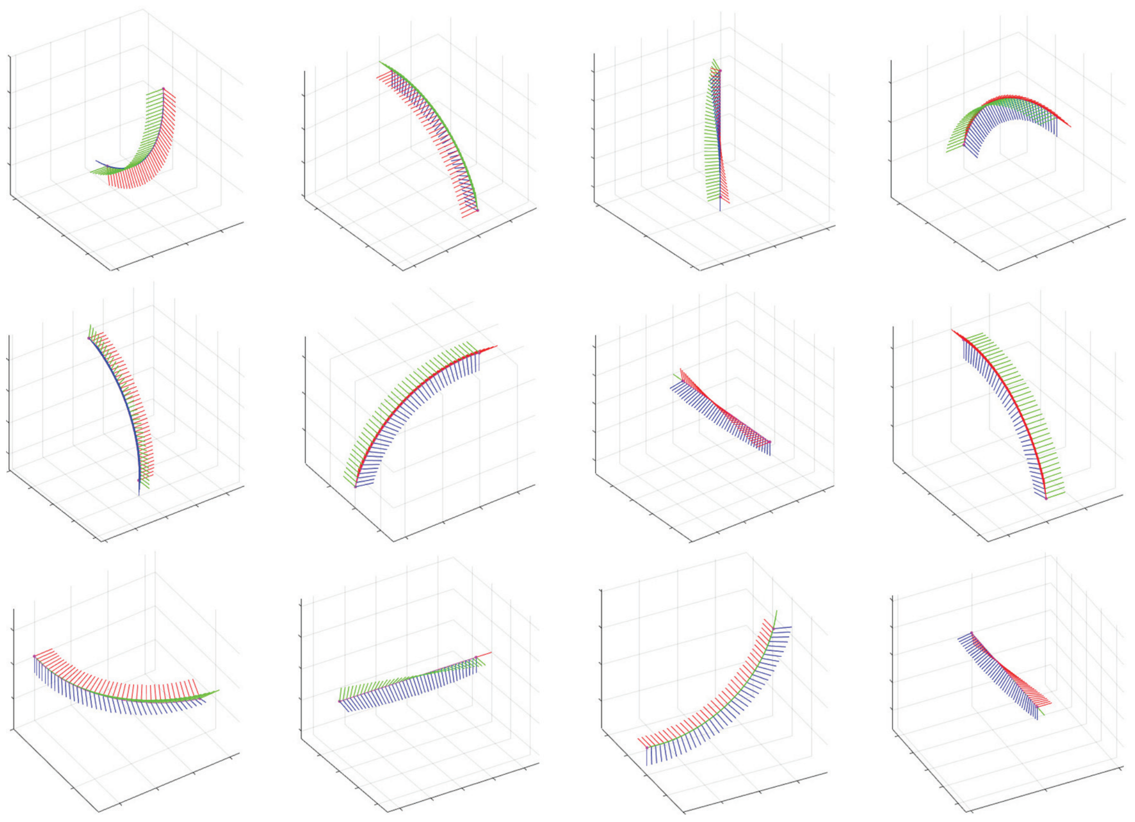


Fig. 6. Library of motions from which the *Q*-learning framework chose suggestions. These motions were chosen empirically as they represent the common motions an operator might imbue on a rig as it is being calibrated. The motions include combinations of vertical and horizontal swoops as well as rolls (as indicated by the rotating motion around the forward axis).

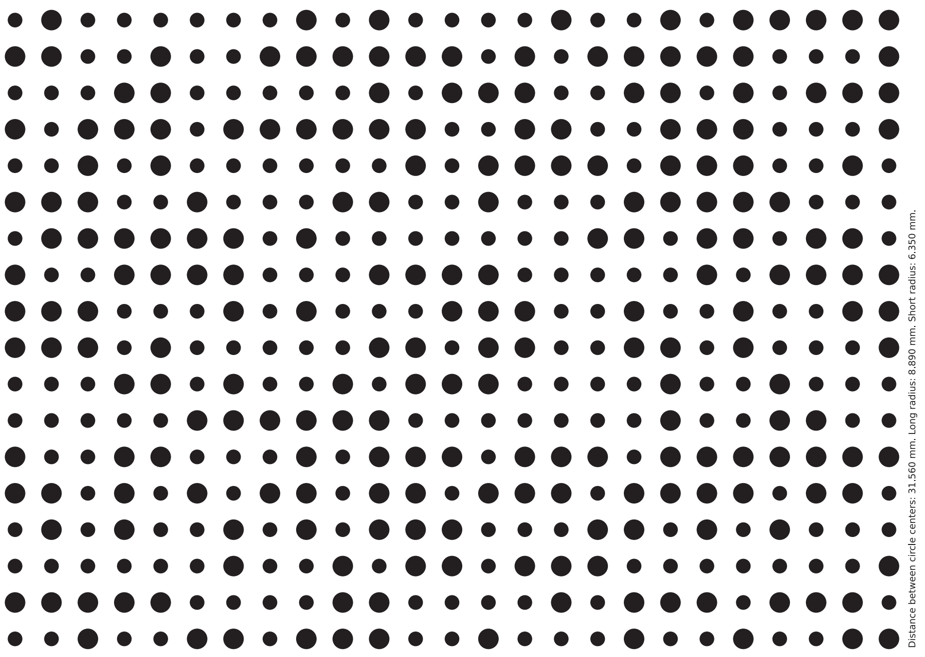

Distance between circle centers: 31.560 mm. Long radius: 8.890 mm. Short radius: 6.350 mm.

Fig. 7. Scaled calibration target used by Heckman et al. (2014), which provides robustness to partial occlusion as well as resolution of the estimated location of the target's origin with respect to the center of the camera, required for tracking in visual–inertial calibration.

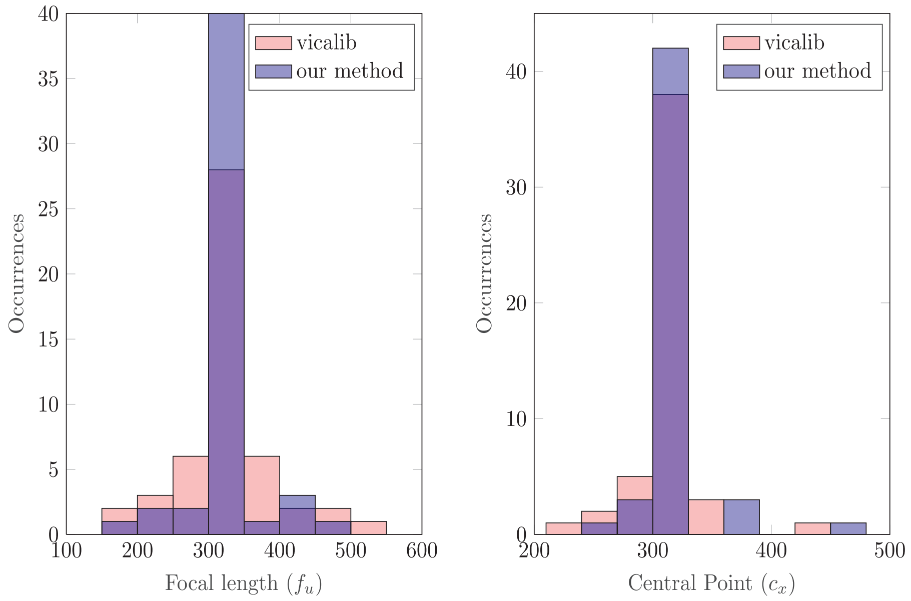


Fig. 8. Distribution of focal lengths and central point for all trials. The mean parameter values between our method and vicalib are similar (focal length: 332.1px for vicalib and 334.8px for our method), the standard deviations are much higher (32.1 vs. 21.2) indicating more consistent and repeatable results.

This software system was chosen over others (e.g. Furgale et al., 2013) as a matter of support on available hardware and previous experience with the codebase, with the caveat that this software was only used to compare the learned calibration motions framework with a dedicated calibration benchmark.

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 8 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 3σ 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. 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.

Calibration is a particularly time-intensive, tedious process by which many rigs used in common robotics applications must be initialized. Unfortunately this process also is generally opaque; the exact kinds of motions an operator must imbue on a rig are difficult to know *a priori* and the sequence of these actions have a non-obvious effect on the calibration results. If the goal of providing ubiquitous robotics is to be realized, they must be able to self-calibrate in even the most challenging environments and on any non-holonomic platform. The use of untrained humans to calibrate the rig as demonstrated in this work is interesting as generally a trained perception engineer must be requested to freshly calibrate a rig and ensure confidence in the results. This work shows that even a casual user is capable of calibrating a multi-sensor rig with even better results than experienced users leveraging an offline calibration routine.

An argument could be made that a much larger state space would be possible when using a deep network (Mnih et al., 2015) instead of the Q-table, but this is not without its potential drawbacks. For instance, it is likely that such a network would overfit to joint action-appearance space without very careful treatment of training and a balancing of the training corpus. It is particularly notable that despite this straightforward Q-table's being trained on data in simulation in a virtual world that is made up of rich features as in Nobre et al. (2017), the suggested motions generalized well to the real-world environment. This suggests that the motions learned are largely feature-agnostic as they have performed well in a variety of visual scenes. Another very significant concern about this method is scalability for high-dimensional spaces. There exists a tremendous body of literature covering these kinds of applications which may be readily brought to bear against such challenges. No notions of optimality are considered, which is a drawback against analytical approaches, but scalability concerns exist in either case when attempting to generate sample paths that obey motion constraints while also assisting in rig calibration.

Finally, a considerable challenge in the proposed framework is designing the human interface so that the user will follow the instructed motions. This might be even more challenging if the motions are particularly exotic such as those which might result from a deep learning solution. Even so, this is an area in which warrants further research; in the projected fully autonomous mode where motion suggestions are the input to a controller, the human interface can be removed from the pipeline and exotic motions might be desired if they immediately recalibrate the rig. In summary, we argue that for the calibration problem as stated here, there is little practical advantage of using a deep network to represent the learned policy or value function.

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. An exciting next opportunity arising out of this framework might be the application of inverse reinforcement learning, or the learning of the reward function directly, from the actions of an experienced user who is calibrating a particular rig. This would refine our work through the encoding of useful and salient rewards rather than hand-tuned rewards as has been pursued here. Another opportunity lies in the ability to learn which motion segments are helpful for regressing a certain calibration parameter, and replaying that sequence of actions to an autonomous platform when calibration has degraded. Such an effort might go someway to realizing the goal of building self-calibrating platforms through both active and passive methods depending on the availability of those methods afforded by the environment and planning objectives.

Funding

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Award No. N65236-16-1-1000. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

ORCID iD

Christoffer Heckman https://orcid.org/0000-0002-9651-6866

References

Agarwal S and Mierle K and Others (2018) Ceres solver. http://ceres-solver.org.

Arulkumaran K, Deisenroth MP, Brundage M and Bharath AA (2017) Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34:26–38.

Brookshire J and Teller S (2013) Extrinsic calibration from per-sensor egomotion. *Robotics: Science and Systems VIII*, pp. 504–512.

Chatterjee K, Chmelk M, Gupta R and Kanodia A (2015) Qualitative analysis of pomdps with temporal logic specifications for robotics applications. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 325–330.

Civera J, Davison AJ and Montiel JMM (2008) Inverse depth parametrization for monocular SLAM. *Transactions on Robotics* 24(5): 932–945.

Durrant-Whyte H and Bailey T (2006) Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine* 13(2): 99–110.

François-Lavet V, Fonteneau R and Ernst D (2015) How to discount deep reinforcement learning: Towards new dynamic strategies. In: *NIPS 2015 Workshop on Deep Reinforcement Learning*.

Furgale P, Rehder J and Siegwart R (2013) Unified temporal and spatial calibration for multi-sensor systems. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 1280–1286.

Gibbs BP (2011) *Advanced Kalman Filtering, Least-squares and Modeling: A Practical Handbook*. New York: John Wiley & Sons.

Golub GH and Van Loan CF (2012) *Matrix Computations*, Vol. 3. Baltimore, MD: JHU Press.

Grady DK, Moll M and Kavraki LE (2015) Extending the applicability of POMDP solutions to robotic tasks. *IEEE Transactions on Robotics* 31(4): 948–961.

Hansen PC (1998) *Rank-deficient and Discrete Ill-posed Problems: Numerical Aspects of Linear Inversion*. Philadelphia, PA: SIAM.

Hausman K, Preiss J, Sukhatme GS and Weiss S (2016) Observability- aware trajectory optimization for self-calibration with application to UAVs. *IEEE Robotics and Automation Letters* 2(3):1770–1777.

Heckman C, Keivan N, Falquez J and Sibley G (2014) VICalib visual–inertial calibration suite. https://github.com/arpg/vicalib.

Hermann R and Krener A (1977) Nonlinear controllability and observability. *IEEE Transactions on Automatic Control* 22(5): 728–740.

Hussein A, Gaber MM, Elyan E and Jayne C (2017) Imitation learning: A survey of learning methods. *ACM Computing Surveys* 50(2): 21.

Jauffret C (2007) Observability and Fisher information matrix in nonlinear regression. *IEEE Transactions on Aerospace and Electronic Systems* 43(2): 756–759.

Jaulmes R, Pineau J and Precup D (2005) Active learning in partially observable markov decision processes. In: *European Conference on Machine Learning*. New York: Springer, pp. 601–608.

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

Klein G and Murray D (2007) Parallel tracking and mapping for small AR workspaces. In: *International Symposium on Mixed and Augmented Reality*. IEEE, pp. 225–234.

Kober J, Bagnell JA and Peters J (2013) Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32(11): 1238–1274.

Kümmerle R, Grisetti G and Burgard W (2011) Simultaneous calibration, localization, and mapping. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3716–3721.

Leutenegger S, Chli M and Siegwart RY (2011) Brisk: Binary robust invariant scalable keypoints. In: *2011 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 2548–2555.

Leutenegger S, Lynen S, Bosse M, Siegwart R and Furgale P (2015) Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research* 34(3): 314–334.

Levinson J and Thrun S (2014) Unsupervised calibration for multi-beam lasers. In: *Experimental Robotics*. New York: Springer, pp. 179–193.

Low KH (2007) *Industrial Robotics: Programming, Simulation and Applications*. I-Tech.

Martinelli A, Scaramuzza D and Siegwart R (2006) Automatic self-calibration of a vision system during robot motion. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006 (ICRA 2006)*. IEEE, pp. 43–48.

Mnih V, Kavukcuoglu K, Silver D, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Mur-Artal R and Tardós JD (2017) Visual–inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters* 2(2): 796–803.

Nobre F and Heckman C (2017) Reinforcement learning for assisted visual–inertial robotic calibration. In: *International Symposium on Robotics Research*. International Foundation of Robotics Research.

Nobre F, Heckman C and Sibley G (2016) Multi-sensor SLAM with online self-calibration and change detection. In: *International Symposium on Experimental Robotics (ISER)*.

Nobre F, Kasper M and Heckman C (2017) Drift-correcting self-calibration for visual–inertial SLAM. In: *2017 IEEE International Conference on Robotics and Automation Robotics and Automation (ICRA)*. IEEE.

Preiss JA, Hausman K, Sukhatme GS and Weiss S (2017) Trajectory optimization for self-calibration and navigation. *Robotics: Science and Systems XIII*.

Richardson A, Strom J and Olson E (2013) AprilCal: Assisted and repeatable camera calibration. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1814–1821.

Russell SJ and Norvig P (2016) *Artificial Intelligence: A Modern Approach*. Malaysia: Pearson Education Limited.

Sheehan M, Harrison A and Newman P (2012) Self-calibration for a 3D laser. *The International Journal of Robotics Research* 31(5): 675–687.

Strasdat H, Montiel J and Davison AJ (2010) Real-time monocular SLAM: Why filter? In: *2010 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2657–2664.

Sturm PF and Maybank SJ (1999) On plane-based camera calibration: A general algorithm, singularities, applications. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999*, Vol. 1. IEEE, pp. 432–437.

Winterer L, Junges S, Wimmer R, et al. (2017) Motion planning under partial observability using game-based abstraction. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 2201–2208.

Zhang Q and Pless R (2004) Extrinsic calibration of a camera and laser range finder (improves camera calibration). In: *Proceedings 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, Vol. 3. IEEE, pp. 2301–2306.