Constant-time Monocular Self-Calibration

Nima Keivan and Gabe Sibley

Abstract— **This paper describes an extensible framework for real-time self-calibration of cameras in the simultaneous mapping and localization (SLAM) setting. The system is demonstrated to calibrate both pinhole and fish-eye camera models from unknown initial parameters while seamlessly solving the online SLAM problem in real-time. Self-calibration is performed by tracking image features, and requires no predetermined calibration target. By automatically identifying and using only those portions of the sequence that contain useful information for the purpose of calibration the system achieves accurate results incrementally and in constant-time vs. the number of images. Furthermore, no special initialization movements are necessary. Parameters estimated by the framework are shown to closely match the batch solution as well as offline calibration values, but are computed live in constant-time. By not rolling information into an assumed prior distribution, the system avoids inconsistencies caused by early linearization – a problem that limits filtering techniques. The system is evaluated with experimental data and shown to be accurate vs. both the offline and batch calibration estimates.**

I. Introduction

For both mobile and service robots, images obtained from cameras are one of the primary sensor modalities used in localization, mapping, object detection and planning. However an accurate calibration of the camera is required in order to utilize the images for metric estimation. While offline calibration provides an initial estimate to the calibration parameters, it is not always feasible and convenient to calibrate cameras using pre-fabricated calibration targets. Calibration parameters could also change during storage or between uses, requiring recalibration. A true "power on and go" solution would both enable a simple camera calibration solution for mobile robots, as well as removing a major barrier of entry for students and hobbyists wishing to incorporate metric vision in their robotic projects. Camera self-calibration in the SLAM framework is the process of estimating the intrinsic parameters of a camera considering solely a sequence of images, without any assumptions as to the content of the scene, whilst simultaneously estimating the map and camera location. The calibration parameters typically consist of the focal length, principal point and several distortion parameters. An online implementation of self-calibration would recursively estimate these parameters as new images are obtained, attempting to provide the best estimates considering all available information. Traditionally, calibration is performed offline using images of a known calibration target, or as part of a large offline batch optimization. Once calibrated, the parameters are fixed for the lifetime of the robot. This approach presents many drawbacks, such as the vulnerability to any changes in the camera parameters, the inconvenience of producing and imaging a known calibration target, along with the bespoke software written explicitly to detect it. Apart from continually improving the calibration parameter estimates as new images become available, online self-calibration also paves the way for robust change detection and estimation, dealing with situations where the physical calibration parameters intentionally or unintentionally change. For mobile robotics, easy self-calibration and hence "power on and go" functionality could for instance allow naive users, such as young students, to experiment with visual simultaneous localization and mapping algorithms without first becoming experts in camera calibration.

Due to these advantages, online self-calibration for visual and visual-inertial systems has been of interest to research recently [10], [2], [7]. However, the estimation of calibration parameters in constant-time considering all present and previous data presents challenges which have so far prevented online self-calibration from becoming commonplace. Filtering methods have been the main approach in cases where the estimation of parameters requires processing all present and past information. The estimation is made constant-time by rolling information from past image measurements into a parametric distribution, or prior. Ideally, the prior would incorporate the influence of the now-absent parameters and measurements on the smaller, active set of parameters, enabling a constant-time estimation of these active parameters that considers the entire trajectory up to the present. Unfortunately, filtering methods present some drawbacks that are especially critical for the case of self-calibration. Due to the nonlinearity of camera models, linearized parametric distributions often cannot represent past measurements without introducing inconsistencies in the estimation. In the case of SLAM, it is well known that filtering methods can be inconsistent. Recent work has focused on how to tackle these inconsistencies [8], [5], as their presence can cause overconfidence in the estimated parameters, an especially undesirable situation in the case of self-calibration. Even when complex parametric models are used, early linearization used to obtain the parametric distributions can lead to inaccuracies in the prior which lead to inconsistency. Furthermore, updates to the calibration parameters induce large and often highly non-linear changes to the parameter estimates, further exasperating the effects of early linearization.

These issues motivate the work presented in this paper, which involves estimating the calibration parameters online, by considering only the best segments of the trajectory,

Nima Keivan (nimski@gwu.edu), and G. Sibley (gsibley@gwu.edu) are with the Autonomous Robotics and Perception Group, Department of Computer Science, The George Washington University, Washington, DC 20052, USA

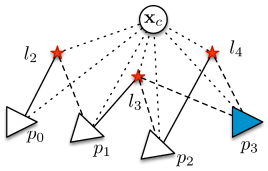

Fig. 1. Graphical model representing the self-calibrating SLAM problem. Edges between the calibration parameters $\mathbf{x}_c$ and all other parameters make the problem difficult to factor.

without a prior distribution. The motivation behind the work is the realization that not all segments in a SLAM sequence are equal as far as their information content regarding the calibration parameters. This can be easily conceptualized due to the existence of degenerate configurations which provide no information [14]. If the best segments in the trajectory are considered, a constant time estimate for the calibration parameters can still be obtained. Moreover, due to the lack of a prior, inconsistencies are caused only by modeling and measurement uncertainty errors, rather than a product of the framework, easing their mitigation. Due to the self-selection of these segments, special attention is paid to ensuring unbiased estimates, as not all measurements are considered. While the proposed system is only concerned with the calibration of camera intrinsics, it can easily be extended to camera extrinsics and multi-sensor self-calibration.

II. Methodology

The self-calibrating SLAM problem can be represented as a graphical model as per Fig. 1. It is observed that the calibration parameters $\mathbf{x}_c$ are linked to all pose and landmark parameters through measurements, making a direct solution infeasible in an online setting. However, the amount of information obtained about the calibration parameters varies from each measurement. Depending on the motion of the camera, and the structure of the scene, measurements may add little to the observability of the calibration parameters. In certain degenerate cases of camera motion, measurements may indeed provide no information.

To obtain a constant time estimate for the calibration parameters, the best segments of the trajectory are stored in a priority queue as follows: When a new image is received, the set of measurements obtained from it and the additional measurements obtained from a fixed-length of previous images are used to estimate the camera pose and landmark parameters in the window, as well as the camera calibration parameters. This window is then scored based on the uncertainty of the calibration parameters, with a low uncertainty signifying a high score. This score is then compared against each window already in the priority queue. If the score is better than the worst score in the priority queue, the worst window in the priority queue is replaced with the new candidate window. Once such an update takes place, all the windows in the priority queue are used to jointly re-estimate the camera calibration parameters. Since the maximum number of windows in the priority queue is fixed and pre-determined, the calibration parameters are estimated in constant-time in the event of a priority update.

A. Optimization Formulation

The calibration parameters are estimated alongside the camera pose and landmark parameters in a non-linear maximum likelihood estimation framework. This framework is also used to extract the uncertainty of the calibration parameters. Measurements are formed by tracking salient points of the image across multiple frames. Each measurement is modeled as a projection of a landmark parameterized in inverse depth [11] into an image, to form a minimal landmark representation. Note: parameterizing the landmark in this way is consistent with the maximum likelihood estimate, as measurements are formed by tracking an initial reference patch, which remains along the initial reference ray. The projected pixel coordinate $\mathbf{p}_2$ of a landmark in the current frame is formulated via the transfer function $\mathcal{W}$ as

$$\begin{aligned}\mathbf{p}_2 &= \mathcal{W}\left(\mathbf{p}_1, \mathbf{T}_{21}, \rho\right) \\ &= \mathcal{P}\left(\pi_{3d}\left(\mathbf{T}_{wc_2}^{-1}\mathbf{T}_{wc_1}\left[\mathcal{P}^{-1}\left(\mathbf{p}_1, \mathbf{x}_c\right); \rho\right]\right), \mathbf{x}_c\right) \end{aligned} \quad (1)$$

The transferred pixel coordinate is obtained by first back-projecting a reference pixel coordinate obtained via harris corner detection, $\mathbf{p}_1$, given the back-projection function $\mathcal{P}^{-1}$, to obtain the reference ray. This ray is then homogenized given the inverse depth parameter ρ. The resulting 4d homogeneous vector is transformed into the current camera reference frame by the homogeneous transformation $\mathbf{T}_{wc_2}^{-1}\mathbf{T}_{wc_1}$ where $\mathbf{T}_{wc_n} \in \mathrm{SE3}$ is the 4×4 transformation matrix from coordinates of frame n to world coordinates. The result is de-homogenized by π_{3d} and projected into the current image by the projection function $\mathcal{P}$. More specifically, $\mathcal{P}^{-1}\left(\mathbf{p}_1, \mathbf{x}_c\right) \in \mathbb{R}^3$ is the 2d to 3d camera back-projection function which outputs a unit ray given the reference 2d pixel coordinates $\mathbf{p}_1$ and the camera calibration parameters $\mathbf{x}_c$, $\mathcal{P}\left(\mathbf{u}, \mathbf{x}_c\right) \in \mathbb{R}^2$ is the 3d to 2d projection function which outputs a 2d pixel coordinate given a de-homogenized 3d point and $\pi_{3d}\left(\right)$ is the 3d de-homogenization function defined as:

$$\pi_{3d}\left(\begin{bmatrix} x & y & z & w \end{bmatrix}^T\right) = \begin{bmatrix} x/w & y/w & z/w \end{bmatrix}^T$$

As an example, in the case of the pinhole camera model, the calibration parameters $\mathbf{x}_c = \begin{bmatrix} f_x & f_y & c_x & c_y \end{bmatrix}$ are comprised of the focal length and principal point parameters in x and y, and the projection function $\mathcal{P}$ is defined as $\mathcal{P}\left(\mathbf{u}, \mathbf{x}_c\right) = \pi_{2d}\left(\mathbf{K}\left(\mathbf{x}_c\right)\mathbf{u}\right)$ where $\mathbf{K}\left(\mathbf{x}_c\right)$ forms the camera projection matrix given the intrinsic parameters, $\mathbf{u} \in \mathbb{R}^3$ is a 3d point location, and π_{2d} is the 2d de-homogenization function defined as $\pi_{2d}\left(\begin{bmatrix} x & y & z \end{bmatrix}^T\right) = \begin{bmatrix} x/z & y/z \end{bmatrix}^T$. Similarly, the back projection function $\mathcal{P}^{-1}$ for a pinhole camera is defined as $\mathbf{u}_1 = \mathcal{P}^{-1}\left(\mathbf{p}_1, \mathbf{x}_c\right) = \mathbf{K}^{-1}\left(\mathbf{x}_c\right)\begin{bmatrix} \mathbf{p}_1 \\ 1 \end{bmatrix}$.

Given the aforementioned residual model, the state space vector of the optimization is defined as

$$\mathbf{x} = \begin{bmatrix} \{\mathbf{x}_{wi} : i = 1, \ldots, n\} & \{\rho_j : j = 1, \ldots, m\} & \mathbf{x}_c \end{bmatrix}^T$$

where $\mathbf{x}_{wi} \in \mathrm{se3}$ is the 6d tangent space representation of the update to transformation $\mathbf{T}_{wi}$, ρ_j is the inverse depth

parameter for landmark j, and $\mathbf{x}_c$ is the vector of calibration parameters. Given the projection in (1), the residual for the measurement of the jth landmark, with reference (first measurement) frame k, in the ith frame is formulated as

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \mathcal{W}\left(\mathbf{p}_j, \mathbf{T}_{wc_i}^{-1}\mathbf{T}_{wc_k}, \rho_j\right) \tag{2}$$

where $z_{ij} \in \mathbb{R}^2$ is the pixel coordinate measurement of the jth landmark in the ith frame, $\mathbf{T}_{wc_k}$ is the transformation from the reference frame coordinates for landmark j to world coordinates, $\mathbf{T}_{wc_k}$ is the transformation from the coordinates of the measurement frame i to world coordinates, $\mathbf{p}_j$ is the reference pixel coordinate for landmark j, and ρ_j is the inverse depth parameter for landmark j. The total error minimized in the optimization is formulated as

$$e = \sum_{i=1}^{n}\sum_{j=1}^{m} \|\mathbf{r}_{ij}\|^2_{\Sigma_{ij}} \tag{3}$$

where the notation $\|\mathrm{x}\|^2_{\Sigma_{ij}}$ signifies the mahalanobis distance given the measurement uncertainty $\Sigma_{ij} \in \mathbb{R}^{2\times 2}$.

The problem is then solved by iteratively updating the state vector $\mathbf{x}$ in a dog-leg trust region minimization framework[12]. The dog-leg update consists of a mixture of the Gauss-Newton and steepest descent solutions denoted as δ_{GN} and δ_{SD} respectively which are formulated as

$$\delta_{GN} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{g} \qquad \delta_{SD} = \frac{\mathbf{g}^\mathbf{T}\mathbf{g}}{\mathbf{g}^T\mathbf{J}^T\mathbf{J}\mathbf{g}}\mathbf{g} = \frac{\|\mathbf{g}\|}{\|\mathbf{J}\mathbf{g}\|}\mathbf{g}$$

where $\mathbf{g} = \mathbf{J}^T\mathbf{r}$, $\mathbf{J} = \mathbf{W}^{\frac{1}{2}}\frac{\partial \mathbf{r}}{\partial \mathbf{x}}$ is the jacobian matrix of the residual vector with respect to the state vector, $\mathbf{r} = [r_1 \dots r_n]^T$ is the vector of residuals formed as per (2), and $\mathbf{W}$ is the residual weight matrix. The weight matrix $\mathbf{W}$ is a combination of residual weights given by the inverse of the measurement covariance Σ_{ij} from (3), as well as a re-weighted huber norm for outlier rejection. Note that the jacobian has been represented in standard form, which includes the square root of the weight matrix $\mathbf{W}$. This is necessary for the correct calculation of the steepest descent update δ_{SD}. The Gauss-Newton delta is obtained by solving for δ_{SD} using the cholesky factorization of the reduced-camera matrix, obtained by the Schur complement trick[13][1]. The optimization is iterated until either the error or parameter change is smaller than a specific threshold.

Since the MLE framework presented above does not make use of a prior, the system will exhibit a number of unconstrained degrees of freedom, manifesting as null spaces in the system hessian. The monocular SLAM problem in particular exhibits 7 null spaces. The first 6 null spaces are due to the unobservability of the global translation and rotation, and are handled by removing the parameters for the first pose $(\mathbf{x}_{w0})$ from the optimization. The seventh nullspace is due to scale unobservability and is handled by removing a single inverse depth parameter ρ_j from the optimization. The landmark j for which the inverse depth parameter is removed is chosen to have the largest number of measurements, and therefore be well estimated, to ensure minimal impact on the optimization.

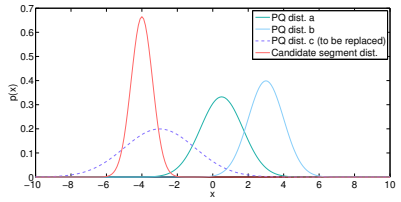

Fig. 2. One dimensional example of the priority queue update operation. Segment c in the priority queue (denoted by the dashed line) has the highest entropy and will be replaced with the candidate segment which has a lower entropy.

B. Priority Queue

The proposed method requires each candidate segment of the trajectory to be scored depending on the observability of the camera parameters during the segment. This enables a comparison between it and the segments already in the priority queue. The proposed score is based on the entropy of the distribution of the calibration parameters for a given segment, with the ultimate goal of obtaining the segments which have the least uncertainty about their individual estimates for the calibration parameters. Given the assumption of gaussian distributions, a posterior can be computed over the calibration parameters by inverting the Fisher information matrix $\mathcal{I}$ and extracting the submatrix associated with the parameters $\mathbf{x}_c$:

$$\Sigma_{\mathbf{x}_c} = \mathcal{I}\left[\mathbf{i}_{\mathbf{x}_c}, \mathbf{i}_{\mathbf{x}_c}\right] = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\left[\mathbf{i}_{\mathbf{x}_c}, \mathbf{i}_{\mathbf{x}_c}\right] \tag{4}$$

where the notation $[\mathbf{x}, \mathbf{y}]$ denotes a submatrix extracted from rows specified in the vector $\mathbf{x}$ and columns specified in the vector $\mathbf{y}$, $\mathbf{J}$ is the jacobian matrix as per section II-A, and $\mathbf{i}_{\mathbf{x}_c} \in \mathbb{R}^m$ is a vector containing the indices of the m camera calibration parameters in the state vector $\mathbf{x}$. Since the calibration parameters are not of the same scale, an entropy calculated based on $\Sigma_{\mathbf{x}_\mathbf{c}}$ will be skewed towards variables with larger variances. In order to correct for this skew, we normalize the posterior covariance matrix

$$\Sigma'_{ij} = \frac{1}{\sqrt{\sigma_i}\sqrt{\sigma_j}}\Sigma_{ij}$$

where σ_i and σ_j are the expected true variances of the ith and jth camera calibration parameters respectively. These quantities should encode the underlying differences between the variances of the parameters given their units and range, and are obtained from the posterior $\Sigma_{\mathbf{x}_c}$ obtained in a batch solution over a large sample trajectory. For the experiments discussed in this paper, a sample trajectory of 2000 frames was used. The quantity $\Sigma'_{\mathbf{x}_\mathbf{c}}$ is similar to the posterior correlation matrix, but with variances from a different, sample trajectory used as the normalizing factors. Since the jacobian matrix $\mathbf{J}$ in (4) contains terms for all poses, landmarks, and the calibration parameters, the inverse can be very costly. Therefore, in order to obtain the marginal distribution, we use the method outlined in[4]:

$$\mathcal{I}\Sigma = \mathbf{I} \;\rightarrow\; \mathcal{I}\Sigma_i = \mathbf{e}_i \tag{5}$$

where the equality in (5) is due to the duality between the information matrix $\mathcal{I}$ and the covariance Σ, and consequently

the ith column of the covariance Σ_i can form an equality with the ith unit vector. Equation (5) can be solved using the cholesky factorization of $\mathcal{I}$, requiring a forward/backward substitution per column of the covariance. As the calibration parameters $\mathbf{x}_c$ are at the end of the state vector, the number of substitutions per covariance column can be limited to the dimensionality of $\mathbf{x}_c$[4]. Given the normalized covariance of the calibration parameters, the entropy of the distribution is then given by $h = \frac{1}{2} ln \left| 2\pi e \Sigma'_{\mathbf{x}_c} \right|$ where the bars denote the matrix determinant. The priority queue update condition is therefore

$$h_c < \alpha \max \{h_i : i = 1, \ldots, n\}$$

where h_c is the entropy of the candidate segment, $\{h_i : i = 1, \ldots, n\}$ is the set of the entropies of the segments already in the priority queue, and $\alpha = 0.95$ is a heuristic to ensure at least a 5% reduction of entropy for an update operation. If this condition is met, the candidate segment is replaced with the segment with the largest entropy. If a candidate segment overlaps with a segment already in the priority queue, the update condition is only checked against the overlapping segment. If the condition is met, the overlapping segments are swapped. This is to ensure that no overlapping segments are present in the priority queue, as the optimization would then double-count the overlapping measurements. Once a swap takes place, the optimization is run again to minimize the error in (3) jointly over all segments of the priority queue. The resulting calibration parameters are then used to continue the SLAM estimation. The distribution of the calibration parameters of the priority queue can then be extracted from the converged hessian of the problem. It must be noted that the nullspace regularization discussed in section II-A must be applied to each segment in the joint priority queue optimization, as the segments do not overlap and therefore each exhibit the aforementioned 7 nullspaces.

It is important that the priority queue update condition not be biased in any way, as rather than solely optimizing over the calibration parameters, the measurements over which the optimization takes place are actively being selected. Any selection criteria which attempts to reduce the entropy of the priority queue distribution, or increase the information gained as a result of the update, risks biasing the solution towards the current estimates for $\mathbf{x}_c$, given the assumptions of gaussian distribution. As an example, the true distribution for a particular parameter may be multi-modal, with measurements first observing one mode, and then the other. In a batch solution with a gaussian distribution assumption, the observation of the second mode would in fact increase the entropy of the distribution over the parameter, as the gaussian assumption is incorrect, and cannot explain the underlying distribution. It is important that this behavior also be replicated in the priority queue solution. The proposed scoring solution is therefore solely based on the entropy of the posterior distribution over the segment itself, not how it relates to the posterior distribution of the priority queue.

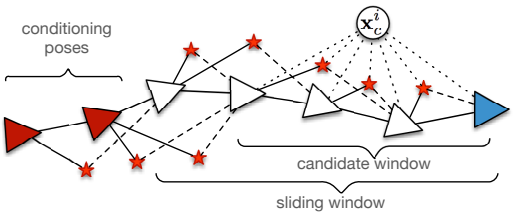

Fig. 3. The sliding and candidate windows, on a sample segment of the trajectory. The sliding window is conditioned on poses outside the window based on co-visible landmarks, whereas the candidate window is considered separately. $\mathbf{x}_c^i$ is the calibration parameter vector estimated by considering only the measurements in the ith candidate window. The sliding, and candidate windows can be processed in parallel.

C. Online Self-Calibration

The online implementation of the proposed method consists of two windowed optimizations. The first is a conditioning sliding window[9] estimator, which does not optimize the calibration parameters. It is tasked with estimating poses and landmarks within an active window, while being conditioned on past poses. The secondary window estimates poses, landmarks as well as the calibration parameters, and is used to compute the marginals $\Sigma_{\mathbf{x}_c}$. This optimization does not condition on information outside the window. The marginals are then used in the update condition of the priority queue. These two estimators are separated since the observability of the calibration parameters is not guaranteed over the candidate window. Poorly observed calibration parameters could then affect the quality of the pose and landmark estimates, affecting the reliability of navigation and localization.

If an update to the priority queue is carried out, an optimization over the entire queue is performed resulting in new estimates for the calibration parameters. These new estimates are then fed back to the sliding window estimator. Special attention has been paid to the startup sequence where zero prior information is assumed over the calibration parameters. To handle this case, a batch optimization is run which includes the calibration parameters, until the entropy of the posterior over the calibration parameters falls below a preset heuristic. This batch optimization is run in conjunction with the priority queue update procedure. Once the batch entropy is lower than a specific heuristic, further estimation is handled by the windowed optimization shown in Fig. 3, and the priority queue is used to further update thet estimates of the calibration parameters.

A keyframing system [9] is implemented in order to increase performance, and the information content of each pose regarding the calibration parameters. As each new image is received, a number of heuristics are used to decide whether or not a keyframe should be created. These heuristics are formed on the distance and angle between the current frame and the previous keyframe, as well as on the percentage of landmarks successfully tracked. The result is that no new keyframes are placed if images from a semi-stationary camera are received. A secondary result is that keyframes are only placed when there is sufficient excitation of the camera to trigger one of the heuristics. Keyframing does not

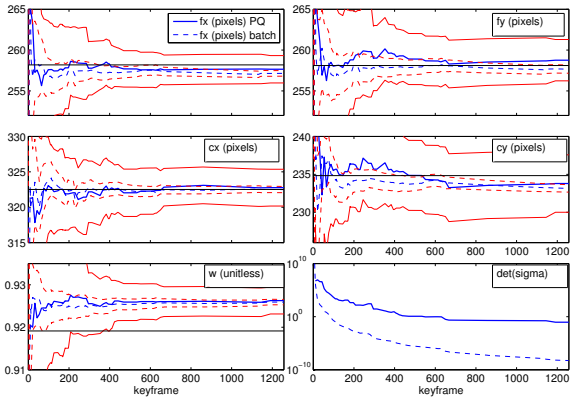

Fig. 4. Results of the GW loop dataset spanning 5300 frames and 1200 keyframes, tracking 128 features on average. The priority queue consisted of 10 segments each with 10 keyframes. Solid and dashed red lines represent 3 sigma bounds for the priority queue and batch solutions respectively. f_x, f_y, c_x, and c_y are the x and y focal length and principal point values respectively, and w is the FOV model distortion parameter. $\det(\Sigma)$ plots the determinant of the calibration parameter covariance Σ'_{ij}. Solid black lines represent offline calibration values. The priority queue and batch statistics are only calculated when the update condition is met and a swap takes place in the priority queue.

completely guarantee observability of the camera parameters due to the existence of degenerate motions which still trigger new keyframes. However, a number of cases of degenerate motion will be successfully avoided, such as the stationary camera case.

III. Results

The proposed system was validated with real data to evaluate its performance and convergence characteristics. In all cases, 2d feature tracks were obtained from the images and used as measurements in the aforementioned optimizations. The experimental datasets were captured with a wide-angle lens and calibrated using the FOV model [3]. The calibration parameters were initialized as follows: the x and y focal lengths were set equally to $90°$, the x and y principal point parameters were set to half the image width and height respectively, and the distortion parameter w of the FOV model was set to represent an ideal fisheye lens $(w = 1.0)$. As discussed in section II-C, an initial batch optimization comprising all poses, landmarks and calibration parameters is run until its entropy h is below a certain threshold, at which point the calibration parameter estimation is handed over to the priority queue. This batch estimation stage lasts for approximately 10 keyframes. A large convergence basin was observed for both the focal length and the distortion parameter w. Focal lengths corresponding to FOVs between $30°$ and $110°$ combined with values of w between 0.5 and 1.0 were observed to converge, while outside values generally diverged due to a combination of tracking and batch estimation failure.

Fig. 4 shows the results of the system running on the GW loop dataset. The dataset was captured around the GWU campus with a wide angle lens camera. It can be seen after the first 10 keyframes in which the batch solution

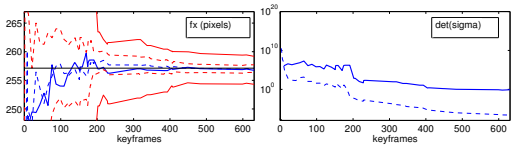

Fig. 5. Focal length and covariance determinant for an indoor corridor sequence. The capture, sequence and priority queue were performed identical to the results in Fig. 4.

is active, the priority queue successfully tracks the mean calibration parameter values obtained by the batch solution, albeit with a higher uncertainty. This is expected for the priority queue, as it is computed using a much smaller subset of the available measurements. The key aspect of the system is that is able to include sections which significantly increase the observability of the parameters, as can be seen in the determinant plot in Fig. 4, which plots the determinant of the normalized calibration parameter covariance matrix Σ'_{ij}. At keyframes 300 and 600, the batch solution determinant dips, signaling a reduction in the total uncertainty of the calibration parameters. The priority queue solution is able to mirror these reductions in uncertainty, by swapping in the relevant segments. This is seen more prominently in the indoor dataset results shown in Fig. 5, where at keyframe 200 a large reduction in uncertainty is visible in both the batch and priority queue solutions.

The effects of the size of the priority queue are demonstrated in Fig. 7, where it is evident that priority queues which consist of more segments tend to better match the batch solution. However, even when using just 5 segments of 10 keyframes each, accurate estimations are observed compared to both the batch and offline estimations. As expected, using a larger number of segments also results in a lower uncertainty for the priority queue estimate. Higher volatility is also observed when fewer segments are present in the priority queue as swapping any individual segment can have a higher impact on the overall priority queue solution.

Discrepancies are observed between some estimates from the priority queue and the offline calibration values (such as w in Fig. 4). This discrepancy can be caused by a number of factors, such as the quality of the offline calibration, feature tracking, the given uncertainty of the visual measurements, and lack of observability over the parameters. However, it is observed that in all cases both the batch and priority queue solution deviate together, and the priority queue closely matches the batch solution. A particular failure case for the system is if each new image adds an equal amount of information to the parameters. In this case, no segment will ever be swapped in the priority queue, as all segments have equal score. However it has been observed that with real and synthetic data, this is an unrealistic case. The determinant based scoring system can also exhibit a failure case, if the uncertainty along a single parameter is exceptionally small compared to the rest due to observability issues. This could artificially inflate the score of a segment, even though it adds little information to most other parameters. It has also been observed that with real and synthetic data that this is not the

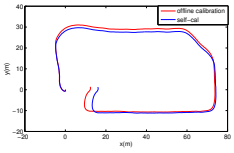

Fig. 6. Comparison between the estimated poses on a 200m loop trajectory for pre-calibrated and self-calibrating estimators. The mean error between the two trajectories is 1.76% of the traveled distance. The self-calibrating estimator's initial calibration parameters were set as per section III.

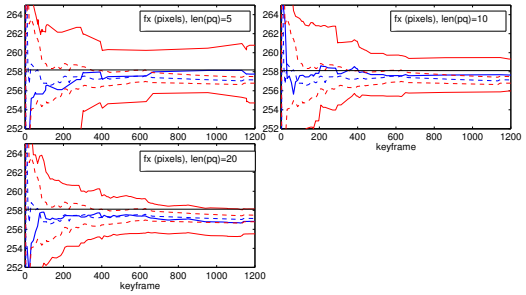

Fig. 7. x focal length over multiple runs of the GW dataset with differing priority queue sizes. Larger priority queues tend to more accurately match the evolution of the batch solution (shown as the dashed line).

general case, however depending on the particular camera motion it could transpire, and may warrant a more discerning scoring system.

It must be noted that although the calibration parameters are re-estimated as the priority queue is updated, the past portions of the trajectory are not. This introduces error in the global camera pose estimate, while local estimates remain optimal. Fig. 6 shows the estimated camera poses for pre-calibrated and self-calibrating runs for a sample 200m trajectory with a fixed window size of 10 keyframes. It can be seen that as expected, due to scale ambiguity, the loop estimate does not close correctly. However it is also observed that without any specific measure to address global optimality, the two solutions remain close, with a mean error of 1.76% of the distance traveled between them. This signifies that the calibration is estimated accurately and quickly enough to ensure estimation close to the pre-calibrated trajectory. If explicit global optimality is desired, a scheme such as Asynchronous Adaptive Conditioning [6] can be used to match the optimal global solution adaptively in constant-time. A single-threaded, synchronous implementation of the proposed method runs at an average of 27fps over the course of the 200m trajectory in Fig. 6 on a 2.6Ghz Intel i7. An asynchronous implementation would significantly improve this as the sliding window estimation would not be halted by calibration related estimation operations.

IV. Conclusions and Future Work

A framework has been presented which enables live and constant-time self-calibration in a SLAM setting. The performance of the system has been experimentally validated, where it has been shown to perform successful SLAM estimation with no initial information about the parameters of the given camera model. The system runs in real-time and is able to closely match the batch calibration solution for the trajectory. Furthermore, as no parametric prior is assumed, no inconsistencies are introduced as a result of early marginalization. Particular attention has been paid to select an update condition which does not artificially bias the priority queue solution towards current estimates. The system automatically identifies measurement sequences that are useful for calibration and saves these sequences in a priority queue. Results show that the mean of the priority queue tracks that of the batch solution, although the uncertainty of the priority queue estimates will be understandably higher than that of the batch solution. For future work, the proposed framework presents an ideal platform for change-detection wherein a system reacts to intentional or unintentional changes to the physical calibration parameters. The inclusion of stereo and visual-inertial extrinsics calibration is straightforward given the framework, and would greatly increase the utility of the system. A number of potential failure cases outlined in section III also warrant further development of the priority queue update condition.

References

[1] Sameer Agarwal, Noah Snavely, Steven M. Seitz, and Richard Szeliski. Bundle adjustment in the large. In *Proceedings of the 11th European Conference on Computer Vision: Part II*, ECCV'10, pages 29–42, Berlin, Heidelberg, 2010. Springer-Verlag.

[2] Javier Civera, Diana R. Bueno, Andrew J. Davison, and J. M. M. Montiel. Camera self-calibration for sequential bayesian structure from motion. In *International Conference on Robotics and Automation*, pages 403–408, 2009.

[3] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. In *In SPIE, volume 2567*, 2001.

[4] Ryan Eustice, Hanumant Singh, John Leonard, Matthew Walter, and Robert Ballard. Visually navigating the rms titanic with slam information filters. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[5] Joel A. Hesch, Dimitrios G. Kottas, Sean L. Bowman, and Stergios I. Roumeliotis. Towards consistent vision-aided inertial navigation. In *Workshop on Algorithmic Foundations of Robotics*, volume 86 of *Springer Tracts in Advanced Robotics*, pages 559–574. Springer, 2012.

[6] Nima Keivan and Gabe Sibley. Asynchronous adaptive conditioning for visual-inertial slam. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2014.

[7] Jonathan Kelly and Gaurav S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research*, pages 56–79, 2011.

[8] Mingyang Li and Anastasios I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *Int. J. Rob. Res.*, 32(6):690–711, May 2013.

[9] Christopher Mei, Gabe Sibley, Mark Cummins, Paul M. Newman, and Ian D. Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94(2):198–214, 2011.

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

[11] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[12] M. Powell. *Numerical Methods for Nonlinear Algebraic Equations*. Gordon and Breach Science, 1970.

[13] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *J. Field Robotics*, 27(5):587–608, 2010.

[14] Peter Sturm. On Focal Length Calibration from Two Views. In *IEEE International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 145–150. IEEE Computer Society, 2001.