2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Robust Vision-Aided Navigation Using Sliding-Window Factor Graphs

Han-Pang Chiu, Stephen Williams, Frank Dellaert, Supun Samarasekera, Rakesh Kumar

Abstract— **This paper proposes a navigation algorithm that provides a low-latency solution while estimating the full nonlinear navigation state. Our approach uses Sliding-Window Factor Graphs, which extend existing incremental smoothing methods to operate on the subset of measurements and states that exist inside a sliding time window. We split the estimation into a fast short-term smoother, a slower but fully global smoother, and a shared map of 3D landmarks. A novel three-stage visual feature model is presented that takes advantage of both smoothers to optimize the 3D landmark map, while minimizing the computation required for processing tracked features in the short-term smoother. This three-stage model is formulated based on the maturity of the estimation of the 3D location of the underlying landmark in the map. Long-range associations are used as global measurements from matured landmarks in the short-term smoother and loop closure constraints in the long-term smoother. Experimental results demonstrate our approach provides highly-accurate solutions on large-scale real data sets using multiple sensors in GPS-denied settings.**

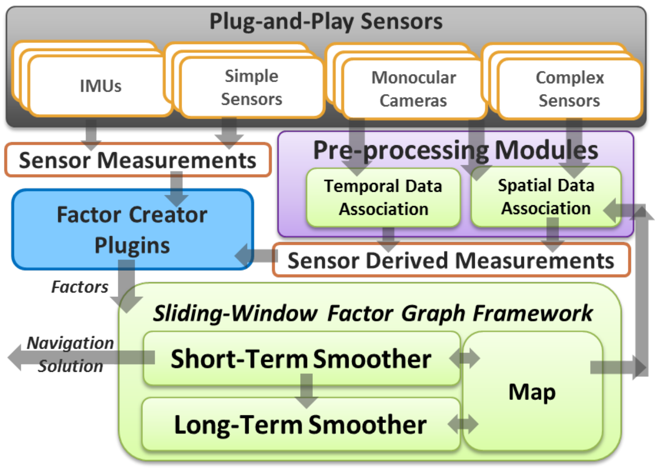

Fig. 1: The architecture for our navigation approach.

I. Introduction

Existing navigation algorithms for robot systems generally achieve only one of two competing goals: they either provide a navigation solution with minimal latency, or they estimate the optimal solution given all of the measurements. Traditional filtering methods achieve fast estimation time by limiting updates to operate on just the most recent states. Previous states are marginalized out when new states are added, a procedure that results in information loss. Further, these methods select linearization points for the new states at the current time and maintain these linearizations throughout the estimation. Unless the system is truly linear, linearization errors will accumulate over time, causing the solution to drift.

In contrast, full simultaneous localization and mapping (full-SLAM) algorithms find the optimal estimate of the robot state at all-time steps based on all available measurements. To obtain the optimal solution, a non-linear least-squares problem is created to incorporate all received measurements. Solutions range from single batch optimizations to recent incremental update approaches. However, none of these methods operate in constant time in the presence of loop closure constraints.

Our approach solves the navigation problem of providing low-latency navigation updates while calculating the full optimal trajectory by splitting the estimation into two components with a shared map of 3D landmarks. The first component is a fast smoother that achieves short-term optimization with fixed computation cost by estimating the states over a constant-length sliding window. The second component is a slower smoother that calculates the optimal solution of the full non-linear problem, including loop closure constraints. Our approach (Figure 1) therefore forms a robust framework to provide a fast highly-accurate navigation solution.

The short-term smoother is implemented using a novel methodology called *Sliding-Window Factor Graphs* (SWFG). Factor graphs [13] are an increasingly popular graph-based smoothing method for Bayesian inference. They provide a flexible foundation for plug-and-play sensing. Each measurement function is encoded into a factor, and the framework performs inference on these factors. SWFG maintain only the portion of the total factor graph that exists inside of a sliding time window, similar to standard fixed-lag smoothing approaches. However, unlike traditional linear smoothers, SWFG support relinearization and efficient updates by extending recent incremental smoothing techniques [1].

The long-term smoother uses standard full-SLAM processing techniques, incorporating loop closure constraints to optimize the robot states and 3D mapped landmarks involved inside the loop. A spatial data association module performs long-range feature matching between the map and new feature measurements observed on re-visited scenes to form these loop closure constraints.

For navigation using image-based sensors, we propose a new three-stage visual feature model that takes advantage of the sliding window in the short-term smoother. Observed scene points are tracked over time and their image locations are exploited in the short-term smoother over three different

H. Chiu, S. Samarasekera, and R. Kumar are with Vision and Robotics Laboratory, SRI International, Princeton, NJ 08540, USA {han-pang.chiu,supun.samarasekera,rakesh.kumar}@sri.com
S. Williams and F. Dellaert are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA {swilliams8,dellaert}@cc.gatech.edu

stages, based on the maturity of the estimation of the underlying 3D location of the scene point (landmark) in the map. The first stage avoids unstable initialization of the 3D location of the landmark points while still incorporating the landmark image observation into the optimization. The second stage estimates both the 3D location of the landmark and the navigation state. Once the uncertainty of the 3D landmark state becomes small, the third stage treats the computed 3D location of the landmark as a fixed quantity when estimating the navigation state, saving computation time. The short-term smoother uses this three-stage model to process tracked features from a temporal association module at consecutive video frames. It also takes long-range feature associations as immediate global measurements from the optimized map to correct drifts in the navigation state estimate. This method ensures both real-time processing and global accuracy.

The remainder of this paper begins with a discussion of related work in Section II. Section III introduces our core SWFG framework, including an efficient long-term smoother and a short-term smoother for low-latency updates based on a new incremental inference method. Section IV describes the three-stage landmark system for image-based sensors in detail, while Section V focuses on the integration of other sensors in our navigation system. We demonstrate our approach provides highly-accurate navigation solutions on several real large-scale GPS-denied scenarios in Section VI followed by our conclusions in Section VII.

II. Related Work

Filtering-based approaches [2], using the Kalman filter and its variants as the estimation workhorse, are capable of integrating multiple sensors to generate the navigation solution in real-time. They avoid the computation in estimating the states for all time by marginalizing out previous states and selecting linearization points for the new states. Since the linearization points cannot be updated at later times, the result cannot be optimized when more measurements become available for improvement. Fix-lag smoothing algorithms alleviate this problem by maintaining a sliding time window of poses for estimation [9], [18] or further relinearization [10], [11]. However, they cannot integrate loop closure constraints which involve states that are outside the window.

Many navigation methods, such as the full simultaneous localization and mapping (full-SLAM) algorithms[1], keep past states and look for the optimal estimate of the robot state at each time step given all available measurements. This nonlinear least squares problem, which is also known as bundle adjustment [4] in computer vision, can be solved efficiently by sparse matrix methods [3]. The increasingly popular graph-based smoothing methods, such as Factor Graphs [13], view solving such a problem as finding the maximum a posteriori (MAP) estimate to a Bayesian network. This view supports recent advances in incremental smoothing techniques, such as iSAM2 [1], which only recalculate the subsection of the factor graph that is affected by new measurements. However, despite these reductions in computation, smoothing solutions are not constant time when closing large loops since all states inside the loop chain need to be updated. They are therefore not directly suitable for navigation purposes when loop closure constraints are available.

A parallel architecture is required to provide both low-latency updates and a fully optimized map. Klein and Murray [5] proposed parallel tracking and mapping (PTAM) of a single camera, where localization and map updates are performed in parallel. While the idea of performing the slower map updates in parallel is directly applicable to navigation, PTAM re-localizes the camera with respect to the current map instead of integrating navigation measurements into a fused estimate. The same parallelization is also used in more recent works [7]. However, these works do not support other sensors and are only demonstrated in small offices.

Mourikis and Roumeliotis [6] combined camera and inertial estimation in a dual-layer estimator: The first Kalman filter layer provides fast navigation solution, and the second bundle adjustment layer feeds back updates to the first layer. However, to limit the accumulation of linearization errors in the filter, they reduce computation in the second layer by discarding old states. It therefore cannot process loop closures if the involved states are removed.

Kaess et al. [14] also proposed a concurrent filtering and smoothing approach that parallelizes low-latency filter updates and high-latency smoother updates. This work focuses on the information sharing between the filter and the smoother, ensuring the combined system remains consistent and does not double-count any measurements. Though this approach does update the current state based on loop closure constraints, limiting the drift in new state estimates, methods for handling visual landmark observations are still unclear.

Our approach utilizes a parallel framework but replaces the filter component in [6], [14] by developing a short-term sliding-window smoother to achieve better linearization performance of highly nonlinear measurements, such as monocular landmark observations. We also develop a new three-stage model to process visual features based on the maturity of the estimation of the underlying 3D landmark. Our framework, based on sliding-window factor graphs, supports both constant-time navigation updates and full, nonlinear smoothing of the entire trajectory using a new incremental inference method. It also provides an adaptable and flexible foundation for any combination of sensors.

III. Sliding-Window Factor Graphs

In this section we introduce a new methodology called Sliding-Window Factor Graphs (SWFG), which maintains only the portion of measurement factors that were received within a sliding time window. SWFG extends the factor graph based incremental smoothing and mapping algorithm (iSAM2) [1], originally designed to solve the full smoothing problem. Based on this methodology, we split the navigation estimation problem into a short-term smoother, a long-term smoother, and a shared map of 3D landmarks.

[1]We focus our discussion on full-SLAM methods, which keep past robot states for optimization. There are other SLAM approaches, such as EKF-SLAM, which eliminate the past trajectory using filtering-based algorithms.

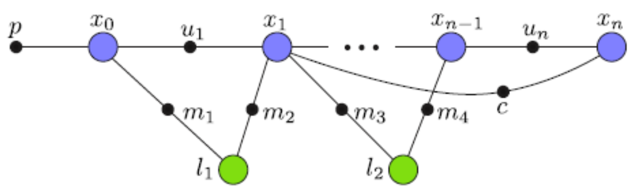

Fig. 2: Factor graph formulation of the SLAM problem, where state variable nodes are shown as large circles, and factor nodes (measurements) with small solid circles. This example illustrate a loop closure constraint *c*, landmark measurements *m*, odometry measurements *u* and a prior *p*.

A. Factor Graphs

A factor graph [13] represents the navigation estimation problem as a graphical model. It is a bipartite graph $G = (\mathcal{F}, \Theta, \mathcal{E})$ with two node types: *factor nodes* $f_i \in \mathcal{F}$ and *state variable nodes* $\theta_j \in \Theta$. An edge $e_{ij} \in \mathcal{E}$ exists if and only if factor f_i involves state variables θ_j. The factor graph G defines the factorization of a function $f(\Theta)$ as

$$f(\Theta) = \prod_i f_i(\Theta_i) \tag{1}$$

where Θ_i is the set of all state variables θ_j involved in factor f_i, and independent relationships are encoded by edges e_{ij}.

A generative model

$$z_i = h_i(\Theta_i) + v_i \tag{2}$$

predicts a sensor measurement z_i is using a function $h_i(\Theta_i)$ with added measurement noise v_i. The difference between the measurement function $h_i(\Theta_i)$ and the actual measurement $\tilde{z}_i$ is encoded into a factor. Assuming the underlying noise process is Gaussian with covariance Σ, the resulting factor is

$$f_i(\Theta_i) = ||h_i(\Theta_i) - \tilde{z}_i||^2_\Sigma \tag{3}$$

where $||\cdot||^2_\Sigma$ is the Mahalanobis distance. Factors can also model more general functions, such as robust estimators.

Figure 2 shows a factor graph for a SLAM problem. A loop closure constraint *c*, landmark measurements *m*, and odometry measurements *u* are examples of factors.

We apply Gaussian variable elimination to the factor graph as a basis for efficient inference [1], [3]. To eliminate a variable θ_j from the factor graph, we first form the joint density $f_{joint}(\theta_j, S_j)$ from all factors adjacent to θ_j, where the variable set S_j consists of all involved variables except for θ_j. Applying the chain rule, we obtain a conditional density $p(\theta_j \mid S_j)$ and a new factor $f_{new}(S_j)$ that represents the remaining information on variables S_j provided by the adjacent factors. After all variables are eliminated, a new factored joint density is formed:

$$p(\Theta) = \prod_j p(\theta_j \mid S_j). \tag{4}$$

While the described variable elimination process is valid for general Bayesian systems, it can only be computed efficiently for the linear Gaussian case. To find the optimal solution for graphs of nonlinear factors, iterative solving methods are employed that successively linearize the system around different operating points until convergence.

B. Incremental Smoothing

The factor graph approach has led to the recent development of an incremental solution to the full SLAM problem, iSAM2 [1]. The iSAM2 algorithm converts the factored joint density $p(\Theta)$ into a Bayes tree data structure by using cliques, subsets of fully-connected variables, as the nodes in the tree [17]. Each node encodes a density on the clique variables, conditioned on variables of its ancestors. In this way, the Bayes tree also encodes a factorization of the full joint density $p(\Theta)$.

Due to this conditional structure, the Bayes tree can be incrementally updated when a new factor arrives. Any clique in the Bayes tree that is connected to the new factor is removed from the tree, as are all cliques along the path to the root of the tree. Because the orphaned branches are merely conditioned on the top of the tree, their conditional densities are unaffected by the addition of the new factor. The top of the tree, along with the new factor, are re-eliminated to form a new tree top, and the orphaned branches are re-attached to form the complete tree. The unaffected branches often comprise most of the Bayes tree, allowing for efficient updates.

To allow relinearization and incremental nonlinear optimization, iSAM2 re-eliminates the affected part of the Bayes tree from the original nonlinear factors instead of the factored conditionals. This requires special treatment for any factor that spans the affected tree top and the unaffected orphan branches. For this set of spanning factors, only the information on the tree top variables is included. This is exactly the information contained in the factors $f_{new}(S_j)$ that were calculated during the original elimination. By caching these factors during the elimination procedure, they need not be recalculated during the incremental updates.

C. Sliding-Window Extension

SWFG extend the iSAM2 algorithm to support inference in a sliding time window over the graph. While iSAM2 is capable of constant-time updates when the new factors only affect the upper cliques of the Bayes tree, the inclusion of landmark states breaks this assumption, as the same landmark may be seen over many frames. To maintain constant-time updates, we extend iSAM2 to efficiently remove states that are outside of the constant-length window.

Removing a variable θ_j from the Bayes tree is equivalent to marginalizing out the variable from the full joint probability density, as:

$$p(\theta_1 ... \theta_{i-1}, \theta_{i+1} ... \theta_n) = \int_{\theta_i} p(\Theta) \tag{5}$$

In general, this is a computationally expensive operation. However, due to the factored nature of the Bayes tree, as illustrated in (6), the variable θ_n of a leaf clique may be marginalized easily.

$$p(\theta_1 ... \theta_{n-1}) = \int_{\theta_n} p(\theta_1) p(\theta_2|\theta_1) ... \quad (6)$$
$$... p(\theta_{n-1}|\theta_1 ... \theta_{n-2}) p(\theta_n|\theta_1 ... \theta_{n-1})$$
$$= p(\theta_1) p(\theta_2|\theta_1) ... p(\theta_{n-1}|\theta_1 ... \theta_{n-2})$$

Since θ_n does not occur anywhere except the last conditional probability, integrating out θ_n only requires dropping the conditional from the end of the joint distribution product. By using a temporal variable ordering during the elimination phase, we can ensure the oldest variables in the factor graph will occur in leaf cliques.

While marginalization is easy within the factored Bayes tree structure, allowing relinearization and nonlinear optimization of the remaining variables requires modifying the nonlinear system instead. All of the original nonlinear factors from which the Bayes tree was constructed that involve the marginalized variable must be removed. To prevent a loss of information from the removal of these factors, the information contained in these factors on variables that are still in the sliding window must be inserted. As in the case of incremental updates in iSAM2, the required information that must be inserted is exactly the $f_{new}(S_j)$ factors calculated during the previous variable elimination step. By caching these factors during elimination, they need not be recalculated during marginalization.

Within the Bayes tree structure, modifying the factors related to a variable generally requires the recalculation of the entire branch, as described in Section III-B. However, in the case of marginalization, the structure of the remaining tree and the optimized values of the remaining variables are unchanged. Thus, no further action is required.

Finally, the SWFG is a hybrid system consisting of nonlinear factors that are wholly within the sliding window and the linear factors $f_{new}(S_j)$ for factors that were adjacent to both variables inside the sliding window and variables that have been marginalized. To maintain probabilistic consistency, the linearization point of any variable adjacent to a linear factor is kept constant, as suggested in [10], [19].

D. Short-Term Smoother

The short-term smoother used the newly devised SWFG to support a constant-length sliding window for better optimization of highly nonlinear factors. Traditional filtering methods only keep the current state, and linearize measurements only once at the time of arrival. However, some states require several measurements before a good estimate can be obtained. This is particularly true of 3D landmark positions estimated from tracked visual features. Using the original linearization point for these states may lead to poor estimation and convergence performance. The short-term smoother relinearizes factors inside the window at a particular frequency, and achieves a more optimal solution than filtering methods by checking consistency across a larger collection of sensor measurements. Currently the length of the window is set to meet the computation requirements of the host system.

E. Long-Term Smoother

The long-term smoother is equivalent to a full smoothing system because the length of the window is set to infinity. This effectively reverts SWFG back to the standard iSAM2 algorithm. The main goal of the long-term smoother is to integrate expensive loop closure constraints, producing a full, high-quality smoothed trajectory and an optimized map. In this paper, loop closure constraints are identified using long-range matches between stored landmarks and features observed on revisited scenes.

F. Interaction

The interaction between two smoothers is mainly through a shared map of 3D landmarks (see Figure 1). The 3D landmarks in the map are optimized by both smoothers. The short-term smoother models 3D landmarks of tracked features, and improves the estimation with multiple observations. The long-term smoother takes summarized information from the short-term smoother and forms relative pose constraints to connect states. It also utilizes loop closures, which are formed from long-range feature matches between re-visited scenes and the map, to further optimize the 3D mapped landmarks involved in the loops. These long-range matches are also used to derive global known landmark measurements as unary factors in the short-term smoother to immediately correct drifts. More details are in Section IV.

IV. Three-Stage Landmark Representation

In this section, we propose a new three-stage representation to model visual features based on the maturity of the underlying 3D landmark in the map. We show the details on how we formulate feature measurements into factors based on this three-stage representation. To simplify the notation, we assume all sensors have the same center, which is the origin of the body coordinate system. As shown in Figure 2, we define a state variable node x_i at time *i* in our SWFG as follows:

$$x_i = (R_i, t_i, v_i, b_i) \quad (7)$$

where *R* represents rotation from world coordinate system to local body's coordinate system, *t* is the position of the origin of the local coordinate system in world coordinate system, *v* is the 3-dimension velocity in the world coordinate system, and *b* represents the sensor-specific bias variables.

A. Continuously Tracked Features

This new three-stage method utilizes the relationship between the modeled 3D landmark and the projected 2D points across frames. It is first initialized in the short-term smoother, when a feature is tracked by a temporal data association module [15]. The temporal data association module efficiently associates sequential data elements from complex sensors such as monocular cameras. It tracks measurements and features over time in a sequential process. The three-stage method for each tracked visual feature is shown in Figure 3:

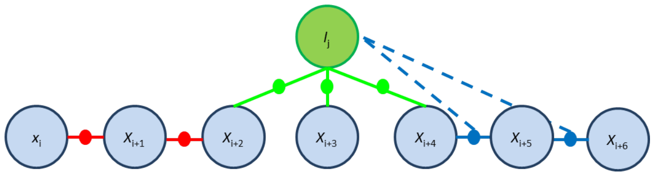

Fig. 3: One example of our three-stage modeling on factors for a tracked feature observed from 7 sequential states. The involved state nodes are shown with big circles, and factor nodes are represented with small solid circles (first stage: red, second stage: green, third stage: blue).

- First stage: When one feature track is first observed, we compute the 3D landmark coordinate by triangulation and then initialize the feature as a binary factor across two consecutive states x_k and x_{k+1}. The 3D information is marginalized out in the factor formulation.
- Second stage: Once the feature has been seen from enough valid frames, we formulate the tracked feature as re-projection factors connecting a 3D landmark state l_j and pose variables (R, t) in states x where the feature is observed. We keep updating the involved states when the feature is tracked across more frames. For the re-projection factor formulation, we refer to [1], [3].
- Third stage: Once the uncertainty of the 3D landmark state converges, we marginalize out the landmark state and switch back to the binary factor formulation in the first stage. However, the 3D position of the feature for the binary factor formulation comes from the marginalized landmark state, estimated in the second stage.

The binary factor formulation used in the first and third stage is inspired from [9]. However, we use different ways to estimate the 3D feature position and support better optimization of these highly nonlinear factors. Consider a single feature s tracked from state x_{i-1} to state x_i, this factor only involves variables $\Pi = (R, t) \in x$. The nonlinear measurement model for observations of s on $\Pi_{i-1}$and Π_i is

$$z_k = h(P_s, \Pi_k) + n_k = h(P_s^k) + n_k, k = i-1, i \quad (8)$$

where $P_s = [Z_1\ Z_2\ Z_3]^T$ is the unknown 3D position of this feature in world coordinate system, $P_s^k = R_k(P_s - t_k)$ is the 3D feature position transformed from world coordinate system to body coordinate system on state x_k, $h(P_s^k) = \begin{bmatrix} Z_1/Z_3 & Z_2/Z_3 \end{bmatrix}^T$is the projection on the normalized image plane, and n_k is the 2-dimension image noise vector with covariance matrix $C_k = \sigma_{im}^2 I_2$.

In the first stage, P_s is estimated by triangulation from the 2D feature positions on three feature-observed states: the first state Π_o, the previous state $\Pi_{i-1}$and the current state Π_i. In the third stage, the value of P_s is retrieved from the converged 3D landmark state l_s for feature s. This way ensures the formulation directly uses the optimized 3D landmark value to form the binary constraint. Once this 3D landmark estimation is obtained, we compute the measurement residual and linearize the estimates of Π_k and P_s as:

$$r_k = z_k - \hat{z_k} = z_k - h(\hat{P_s}, \hat{\Pi_k}) \simeq H_{\Pi_k}\delta\Pi_k + H_{s_k}\delta P_s + n_k \quad (9)$$

where H_{Π_k} and H_{s_k} are the Jacobians of the measurement z_k with respect to Π_k and P_s respectively. We then stack z_{i-1} and z_i together as:

$$r \simeq H_\Pi \delta\Pi + H_s \delta P_s + n \quad (10)$$

where $r = [r_{i-1}; r_i]$, $H_\Pi = \left[H_{\Pi_{i-1}}; H_{\Pi_i}\right]$, $H_s = \left[H_{s_{i-1}}; H_{s_i}\right]$, and $n = [n_{i-1}; n_i]$ with covariance matrix $C = \sigma_{im}^2 I_4$. To marginalize out the 3D landmark state P_s in the formulation, we project r on the left nullspace of H_s to get r_o using a unitary matrix U whose columns form the basis of the left nullspace of H_s:

$$r_o = U^T(z - \hat{z}) \simeq U^T H_\Pi \delta\Pi + U^T n = H_o \delta\Pi + n_o \quad (11)$$

where r_o is a 1-dimension vector after projection. Then we split H_o into $H_{o_{i-1}}$ and H_{o_i} for state $\Pi_{i-1}$and Π_i respectively. This results in the following linearized constraint between two states for our factor formulation, and has been shown [9] to yield better results than epipolar constraints.

$$r_o = H_{o_{i-1}}\delta\Pi_{i-1} + H_{o_i}\delta\Pi_i + n_o \quad (12)$$

This three-stage method finds a balance between efficiency and optimization on modeling tracked features. It avoids immediate initialization of the landmark state, which may cause instability, but still utilizes the feature track for navigate state estimate before the landmark state is constructed. The method optimizes both the 3D location of landmark state and the navigation state in the second stage.

The spirit of the transition from the first stage to the second stage is similar to [8]. However, the inverse depth parametrization on landmark states used in [8] is not suitable for our incremental smoothing algorithm, because extra parameters on the camera position where the feature was first observed need to be optimized during each update.

Our model enters into the third stage if the uncertainty of the 3D landmark state becomes small. This way saves computation by treating the landmark distributions as fixed. It also ensures only long-tracked features, which are more reliable, can move to the second and third stages.

Note each of the tracked features processed at a particular state will be first observed at different times, so each feature is handled independently. Features used for inference at any particular time are therefore in different stages of processing. Also a feature may not be tracked for sufficient length of time and in that case may not reach the third stage maturity level and get established as a fixed landmark.

B. Long-Range Landmark Matching

In addition to tracked features, we have a spatial data association module [12] that establishes spatial associations across sensor measurements taken at different times. Newly discovered associations, which are between stored landmarks in the map and current observed scene features, determine two kinds of factors used in the short-term smoother and the long-term smoother respectively.

If the uncertainty of a matched 3D landmark is low, we formulate one unary factor in the short-term smoother for each correspondence between a 2D feature on the query camera frame and this matched 3D landmark feature in the map. Since the position of the matched 3D landmark point is already optimized in the map based on the three-stage model, we treat it as a fixed 3D point in the global coordinate system. We then transform this fixed 3D point to the body coordinate system, based on rotation R_i and translation t_i in query state x_i, to generate the measurement model. A detailed derivation for this factor formulation is available in [15].

This factor formulation is applied to all the point correspondences on matured landmarks returned from the spatial data association module. These unary factors perform immediate global corrections in the short-term smoother.

We also form loop closure constraints (Figure 2) [1], [3] from the spatial data association module, and integrate them into the long-term smoother for full optimization. This further improves the maturity of the loop-involved 3D landmarks in the map by optimization, and generates a full smooth trajectory on all past states.

V. Navigation System

In this section, we show how we formulate other sensor measurements into factors in our system. In addition to monocular cameras, we use an IMU, an odometer, and a barometer for experiments in this paper. An odometer measurement is formulated as a binary factor to describe the traveled distance between consecutive states. A barometer measurement reports a height estimate at a particular instance of time, so we directly formulate it as a unary factor on the affected variables in a single state.

To fully utilize high-frequency IMU data, we formulate the IMU mechanism based on the indirect form [9] on the state variables from the inertial system error propagation equations. This avoids the dynamic modeling of the complex kinematics associated with chaotic or rapid movements, and replaces the system dynamics with a motion model derived from IMU propagation.

We derive this error-state IMU mechanism within our factor graph framework, and implemented it as a binary IMU motion factor between two consecutive states x_{i-1} and x_i. This IMU motion factor integrates IMU readings between two time steps, and then generates the 6 degrees of freedom relative pose and corresponding velocity change as the motion model instead of traditional process models. It also tracks the IMU-specific bias as part of the state variables for estimating motion. The linearization point for x_i is computed from this motion model, which is based on the linearization point for x_{i-1} and IMU readings between x_{i-1} and x_i.

VI. Experimental Results

This section demonstrates that our approach provides fast, highly-accurate navigation solutions with multiple sensors on two large-scale real scenarios both assuming GPS is not available for the navigation state estimate. These two scenarios provide different aspects to show the strength of our approach. The initial global position and orientation of the vehicle is assumed known. Ground Truth is obtained by using the RTK differential GPS technique [16].

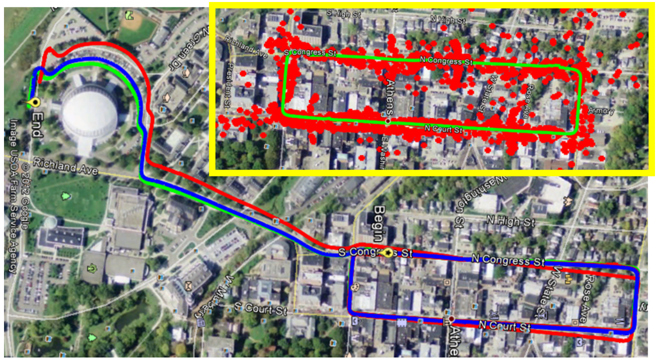

Fig. 4: Large City Navigation Scenario (3.3 kilometers): Ground truth (blue), solution using only 1-step short-term smoother (red, 3D RMS error: 19.19 meters), solution using both smoothers (green, 3D RMS error: 5.85 meters), and the 3D landmarks constructed during the first loop (red points inside the highlighted yellow box).

A. Large City Navigation

This data set is collected using a car that drives inside a city. The car repeats a large loop and then goes to south. It stops many times during navigation due to traffic signs. The total travel distance is 3.3 kilometers, and the travel time is more than 14 minutes. The sensor set includes one 100-hz IMU, one 1-hz odometer, one 25-hz barometer, and one 4-hz monocular camera.

1) The Construction of the Landmark Database and Map: The combination of two smoothers forms a robust framework to provide accurate navigation solutions. As shown in Figure 4, the spatial association module performs long-range feature matching between the map (3D landmarks with projected 2D features on stored images constructed during the first loop) and new feature measurements from cameras on revisited scenes. The short-term smoother then immediately takes these long-range feature associations as global measurements. The long-term smoother uses these associations to form loop closure constraints to further optimize the 3D landmarks involved in the loop. To investigate only the improvement brought by these long-range associations, we set the sliding window in our short-term smoother to only cover one time step. The short-term smoother therefore degrades to a traditional filtering method, and cannot relinearize the factors. However, long-range associations dramatically decrease the 3D RMS error from 19.19 meters to 5.85 meters. The drift during the second loop in the filter-only solution is corrected by long-range associations across two loops.

2) Analysis on 3-Stage Feature Model: Next, we set a constant 12-second window in our short-term smoother, and adopt the 3-stage model to generate factors for tracked features. Figure 5 shows the distribution of all features among the three stages. Note that each feature is handled by

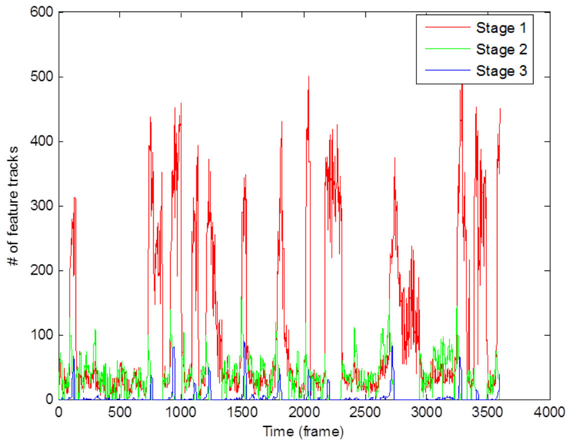

Fig. 5: The maturity distribution (Y-axis: number of features) among all features during navigation (X-axis: navigation time in video frames). Note when the car stops, most detected features are in the first stage (red). When the car moves, there are more long-tracked features (stage 2: green, stage 3: blue) than short features. These matured features improve the result using the optimized 3D landmark estimation.

the 3-stage method independently. Tracked features observed at the same time are therefore distributed among the three stages. For example, when the car stops, a larger number of features are matched across frames. However, as new observations from a stationary camera do not improve the triangulation, most of these features stay in the first stage. When the car moves, there are more long-tracked features in the latter two stages. These matured features are more reliable, and utilizes the optimized 3D landmark estimation to improve the accuracy. The incorporation of this three-stage model reduces the 3D RMS error to 3.76 meters, our best result on this scenario.

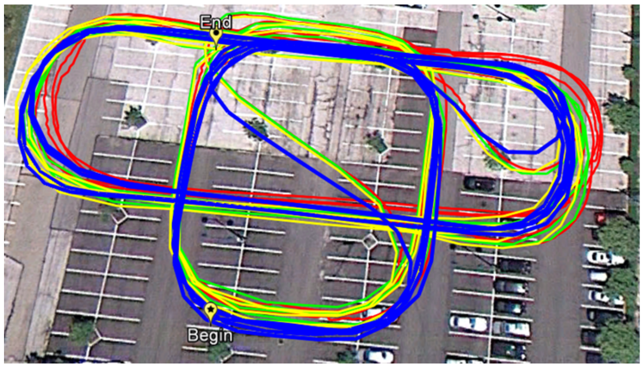

Fig. 6: Trajectories on parking lot scenario (1.74 kilometers): Ground Truth (blue), our short-term smoother solution (yellow, 3D RMS error: 2.11 meters), the first baseline method (red, 3D RMS error: 2.88 meters), and the second baseline method (green, 3D RMS error: 2.46 meters). Note the repeated loops stick tighter (such as the right portion of trajectories) in our solution.

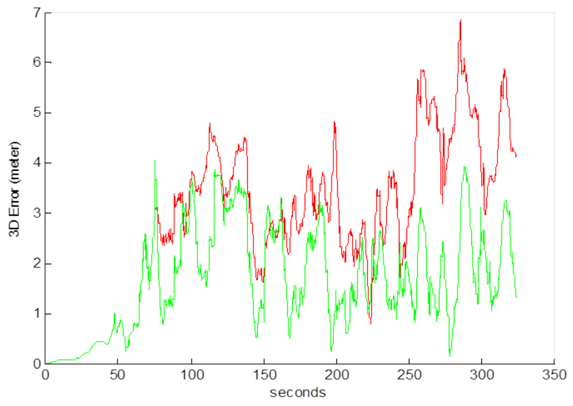

Fig. 7: The 3D error (Y-axis in meters) during navigation (X-axis in seconds) in parking lot scenario: solution using only the short-term smoother (red), and our full solution using both smoothers (green). Note the 3D error doesn't increase over time using our full solution.

B. Parking Lot Driving

This data set is collected using a car that drives and turns very fast inside a parking lot. The scenario includes many repeated loops. The car travels 1.74 kilometers in 325 seconds. The sensor set includes one 100-hz IMU, one 1-hz odometer, and three 4-hz monocular cameras.

1) The Improvement from 3-Stage Feature Model: To focus on demonstrating the influence by our 3-stage method, we first set our system to run using only the short-term smoother with a constant 8-second sliding window. The long-term smoother and the spatial data association module are not used. We compare our short-term smoother to two baseline methods. The first method is the iterative extended Kalman filter (EKF) solution [9], which also fuses IMU motion model and binary constraints from tracked features in a tightly-coupled manner. The second method uses the same setting as our short-term smoother except the tracked features are always formulated as binary constraints. There are no 3D landmark states used in this baseline method.

Figure 6 shows results of our short-term smoother using the 3-stage tracked feature model and the two baseline methods. Compared to the previous scenario, the track length of features in this scenario is much shorter because the high-speed car turning causes many feature breaks. However, by re-linearizing factors inside the sliding window, our short-term smoother still performs better using the 3-stage method. The repeated loops are grouped tighter in our solution.

2) The Combination of Two Smoothers: The combination of two smoothers is extremely powerful in situations with many repeated loops. In this scenario, it further reduces the 3D RMS error to 1.59 meters (the short-term smoother only solution produces 3D RMS error of 2.11 meters), our best result in this scenario. Figure 7 shows the 3D error is not increasing during navigation using this combination. It is because the long-range feature associations across repeated loops provides global correction and avoids drifts.

3) Computation Cost: Our framework uses an incremental smoothing algorithm extended from iSAM2 [1] to perform inference on Sliding-Window Factor Graphs with frequent relinearization. As shown in Figure 8, our incremental optimization takes less than 200 milliseconds on each update outputted from the short-term smoother on this scenario. Compared to traditional batch optimization method, it is

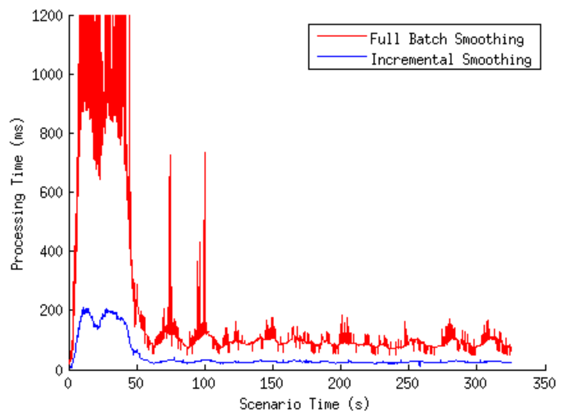

Fig. 8: The processing time along the parking lot scenario: traditional batch optimization (red), and our incremental smoothing method (blue).

much faster and more suited for real-time operation. In the first 50 seconds of the scenario, the vehicle is still and there are 100~200 tracked features in each video frame. Once the car starts to move, the number of tracked features drops (<50) and both methods take less time to process measurements inside the sliding buffer. All timing results were conducted using a single core of an Intel i7 CPU running at 2.40 GHz.

VII. Conclusions

In this paper, we present a robotic navigation solution capable of real-time updates while calculating a full smoothed trajectory by using a methodology called Sliding-Window Factor Graphs (SWFG). SWFG efficiently maintains factors in a sliding time window, and relinearizes stored factors frequently for optimization. We split the estimation problem into a fast short-term smoother, a slower but fully global smoother, and a shared map of 3D landmarks. The short-term smoother improves linearization performance, which is particularly important for highly nonlinear measurements. It achieves a more optimal solution than traditional filtering methods, while still maintaining constant-time updates as loop closures are handled by the long-term smoother.

We also introduced a three-stage landmark representation to take advantage of both smoothers that optimize a landmark map, while minimizing the computation required on the short-term smoother side. Each tracked feature in the short-term smoother is processed depending on the maturity of the underlying landmark. Long-range associations are used as two kinds of factors: global point factors on matured landmarks in the short-term smoother, and loop closure constraints in the long-term smoother. Experiments demonstrate our approach provides fast highly-accurate solutions on large-scale scenarios.

Future work is to ensure our approach operates in real-time while still estimates the full nonlinear navigation state under any conditions, such as limited computation resource. One way is to control the number of factors while maintaining satisfactory accuracy. For example, we plan to select visual features based on spatial distribution, uncertainty, and maturity. Since features with optimized 3D estimations are more reliable than unstable short feature tracks, we could only use matured features to contribute the solution if there are enough third-stage features.

Acknowledgments

This material is based upon work supported by the DARPA All Source Positioning and Navigation (ASPN) Program under USAF/ AFMC AFRL Contract FA8650-11-C-7137. 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 the US government/ Department of Defense.

References

[1] M.Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, “iSAM2: Incremental smoothing and mapping using the Bayes tree,” Intl. J. of Robotics Research, vol. 31, pp.217-236, Feb 2012.
[2] D. Smith and S. Singh, “Approaches to multisensor data fusion in target tracking: A survey,” IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 12, p. 1696, Dec 2006.
[3] F. Dellaert and M. Kaess, “Square root SAM: Simultaneous location and mapping via square root information smoothing,” Intl. J. of Robotics Research, 2006.
[4] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment - a modern synthesis,” Lecture Notes in Computer Science, vol. 1883, pp. 298-375, Jan 2000.
[5] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in Proc. International Symposium on Mixed and Augmented Reality (ISMAR), 2007.
[6] A. Mourikis and S. Roumeliotis, “A dual-layer estimator architecture for long-term localization,” In Proc. Of the Workshop on Visual Localization for Mobile Platforms at CVPR, 2008.
[7] R. Newcombe, S. Lovegrove, and A. Davison, “DTAM: Dense tracking and mapping in real-time,” in Intl. Conf. on Computer Vision (ICCV), 2011.
[8] J. Civera, A. J. Davison, and J. M. M. Montiel, “Inverse depth to depth conversion for monocular SLAM”, in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2007.
[9] A. Mourikis and S. Roumeliotis, “A multi-state constraint Kalman filter for vision-aided inertial navigation,” in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2007.
[10] T. Dong-Si and A. Mourikis, “Motion tracking with fixed-lag smoothing: Algorithms and consistency analysis,” in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2011.
[11] G. Sibley, L. Matthies, and G. Sukhatime, “Sliding window filter with applications to planetary landing,” Journal of Field Robotics, vol. 27, no. 5, pp. 587-608, 2010.
[12] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. Sawhney, “Tenfold improvements in visual odometry using landmark matching,” in Intl. Conf. on Computer Vision (ICCV), 2007.
[13] F. Kschischang, B. Fey, and H. Loeliger, “Factor graphs and the sum-product algorithm,” IEEE Trans. Inform. Theory, vol. 47, no. 2, Feb 2001.
[14] M. Kaess, S. Williams, V. Indelman, R. Roberts, J. Leonard, and F. Dellaert, “Concurrent filtering and smoothing,” in Intl. Conf. on Information Fusion (FUSION), 2012.
[15] T. Oskiper, H. Chiu, Z. Zhu, S. Samarasekera, and R. Kumar, “Stable vision-aided navigation for large-area augmented reality,” in IEEE Intl. Conf. on Virtual Reality (VR), 2011.
[16] J. Sinko, “RTK performance in highway and racetrack experiments,” Navigation, Vol. 50, No. 4, pp 265–275, 2003.
[17] M. Kaess, V. Ila, R. Roberts, and F. Dellaert, “The bayes tree: An algorithmic foundation for probabilistic robot mapping,” in Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR), 2010.
[18] M. Li and A. Mourikis, “Improving the accuracy of EKF-based visual-inertial odometry,” in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2012.
[19] T. Dong-Si and A. Mourikis, “Consistency Analysis for Sliding-Window Visual Odometry,” in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2012.