Constrained Optimal Selection for Multi-Sensor Robot Navigation Using Plug-and-Play Factor Graphs

Han-Pang Chiu, Xun S. Zhou, Luca Carlone, Frank Dellaert, Supun Samarasekera, Rakesh Kumar

***Abstract*— This paper proposes a real-time navigation approach that is able to integrate many sensor types while fulfilling performance needs and system constraints. Our approach uses a plug-and-play factor graph framework, which extends factor graph formulation to encode sensor measurements with different frequencies, latencies, and noise distributions. It provides a flexible foundation for plug-and-play sensing, and can incorporate new evolving sensors. A novel constrained optimal selection mechanism is presented to identify the optimal subset of active sensors to use, during initialization and when any sensor condition changes. This mechanism constructs candidate subsets of sensors based on heuristic rules and a ternary tree expansion algorithm. It quickly decides the optimal subset among candidates by maximizing observability coverage on state variables, while satisfying resource constraints and accuracy demands. Experimental results demonstrate that our approach selects subsets of sensors to provide satisfactory navigation solutions under various conditions, on large-scale real data sets using many sensors.**

I. Introduction

The goal of multi-sensor fusion for robot navigation is to combine measurements from a set of sensors to improve the quality of the solution. Different sensor types provide diverse physical characteristics. Even data from the same type of sensors could be quite varied in their error characteristics depending on their configuration. Combining complementary or redundant information from multiple sensors provides more robust estimation than using a single sensor.

However, adding more sensors requires more computation and power resources. Often due to practical considerations, robot systems are equipped with only a few sensors, which provide complementary information to contribute to the navigation solution. These systems are not designed to dynamically incorporate different types of sensors to improve the overall robustness and accuracy. They cannot tolerate situations when a sensor becomes unavailable due to signal loss or sensor fault.

In addition, incorporating measurements from multiple sensors operating at different frequencies on a single robot could be cumbersome for inference. Traditional inference methods require updating the whole augmented state vector from sensor measurements at multiple rates. It causes computational burden since the vector dimension may be large, especially for SLAM problems. It may also have unnecessary calculations since some state variables may remain unchanged. Handing delayed measurements is also non-trivial in these systems.

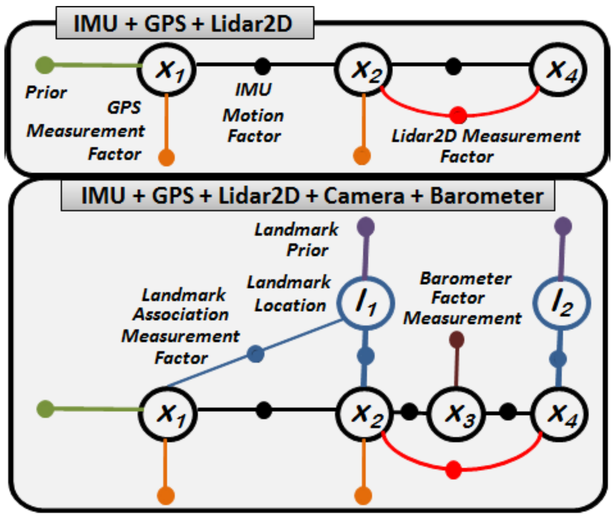


Fig. 1: The plug-and-play factor graph formulation for multi-sensor navigation problem, where state variable nodes are shown as large circles (navigation state: *x*, landmark location: *l*), and factor nodes (sensor measurements) with small solid circles. It progresses when two more sensor types are added at different rates, which have delayed measurements. Note this formulation covers full-SLAM problems, which keep past states for optimization.

Our approach solves the multi-sensor navigation problem while fulfilling performance requirements and system constraints, by using a plug-and-play factor graph framework (Figure 1) coupled with a novel constrained optimal sensor selection mechanism. A factor graph [7] is a probabilistic graphical model of state variables and factor nodes. It encodes the posterior probability of the states over time, given all available sensor measurements as factors. While calculating the full navigation solution over the entire factor graph seems to be expensive, the recent incremental smoothing technique [1] can provide the solution in real time by dynamically optimizing only a small part of the factor graph.

Using the factor graph representation, we develop a highly adaptable framework to incorporate numerous types of sensors for plug-and-play sensing. When sensors get plugged or come online, their sensor measurements are directly mapped to addition of graph nodes and factors that connect these nodes. This framework is capable of handling sensor measurements arriving at different rates and latencies, providing real-time navigation solution at all times. We have also extend factors to encode sensor measurements with linear, nonlinear and multi-modal noise characteristics. Our framework

H. Chiu, X. S. Zhou, S. Samarasekera, and R. Kumar are with Center for Vision Technologies, SRI International, Princeton, NJ 08540, USA {han-pang.chiu,xun.zhou,supun.samarasekera,rakesh.kumar}@sri.com

L. Carlone and F. Dellaert are with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA {lcarlone6,dellaert}@cc.gatech.edu

therefore is able to easily incorporate new evolving sensors due to rapidly advancing sensor technology.

To achieve the desired navigation accuracy with fewest sensors necessary on a single robot, we propose a constrained optimal selection mechanism to guarantee performance while minimizing computation and power usage in real-time system by incorporating only the optimal subset of available sensors. This mechanism is applied to choose the subset of sensors during two situations. One is the initial period in a given environment. The other is when sensor configuration or system constraint changes. For example, it needs to update the subset of sensors when a sensor goes on-line, off-line, or the quality of its measurements dramatically changes. It also adjusts the subset when the system has power shortage.

Unlike most navigation systems which are specifically designed to handle only a few sensors, our mechanism fully exploits advantages due to sensor fusion by dynamically evaluating many possible sensor combinations. It quickly constructs candidate subsets of sensors based on heuristic rules and a ternary tree expansion algorithm, by representing each possible subset as a tree node. Then it selects the optimal subset among candidates by maximizing observability coverage on state variables, while satisfying resource constraints and accuracy requirements. We also develop a maturity-based method to select visual measurements. It further reduces cost without sacrificing navigation accuracy.

The remainder of this paper begins with a discussion of related work in Section II. Section III introduces our plug-and-play factor graph framework, and illustrates how it incorporates many sensor types and handles delayed measurements. Section IV describes our constrained optimal selection mechanism in detail. Section V demonstrates our approach on real large-scale scenarios. It automatically selects subsets of sensors to provide navigation solutions while satisfying different accuracy requirements and resource constraints. Conclusions are made in Section VI.

II. Related Work

Many navigation systems are capable of integrating multiple sensors. Examples include GPS-aided [5] and vision-aided [8], [9], [14] inertial navigation systems. There are also navigation systems using factor graphs [8], [13] to solve the full simultaneous localization and mapping (SLAM) problem. These systems typically rely on a few types of sensors with complementary modality to generate solutions.

There are many existing works on greedy sensor selection [16] in resource constrained distributed sensor networks, for target tracking applications. A general overview can be found in [2]. These works schedule only a subset of the sensors from sensor network for transmission in each time slot, in order to gain the most information in the most efficient way.

There are also works on maintaining satisfactory accuracy in resource-constrained navigation system by reducing the number of processed measurements, mostly for camera sensors. These works either only keep measurements from a subset of distributed camera poses [17], or lower the number of detected or processed features on each frame [4], [12].

S. Sukumar et al. [10] developed a sensor selection algorithm to ignore non-functional sensors for navigation. This algorithm is designed for a single robot equipped with at most four types of sensors. Their system groups sensors with consistent measurement information, and improves the robustness of the navigation solution by detecting and removing outliers from faulty sensors. However, their method does not exploit the complementary nature of different sensors, and hence can not perform better than the best sensor in the system [11].

Our work is designed to continuously identify an optimal subset of available sensors among numerous sensor types on a single platform for navigation application. It maintains robustness and accuracy of the solution while minimizing power usage and computational resources, which is a new challenging research problem. In order to utilize many sensor types, we extend the factor graph formulation to incorporate sensors with different frequencies, latencies, and noise distributions. This extended framework provides an adaptable foundation for plug-and-play sensing, and is designed to incorporate new evolving sensors. Our selection mechanism in resource-constrained system exploits the complementary properties of different sensor types, and focuses on optimal selection both at the sensor level and measurement level.

III. Plug-and-Play Factor Graphs

In this section we introduce our plug-and-play factor graph framework, which is capable of incorporating numerous types of sensors with different rates, latencies, and error characteristics. This is achieved through abstraction of sensor measurements into factor representations, depending on how a measurement affects the appropriate navigation state variables. This framework is also scalable to new evolving sensors with updated parameters and noise models.

A. Factor Graphs

A factor graph [7] represents the navigation estimation problem at all times as a bipartite graph model $G = (\mathcal{F}, \Theta, \mathcal{E})$ 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

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

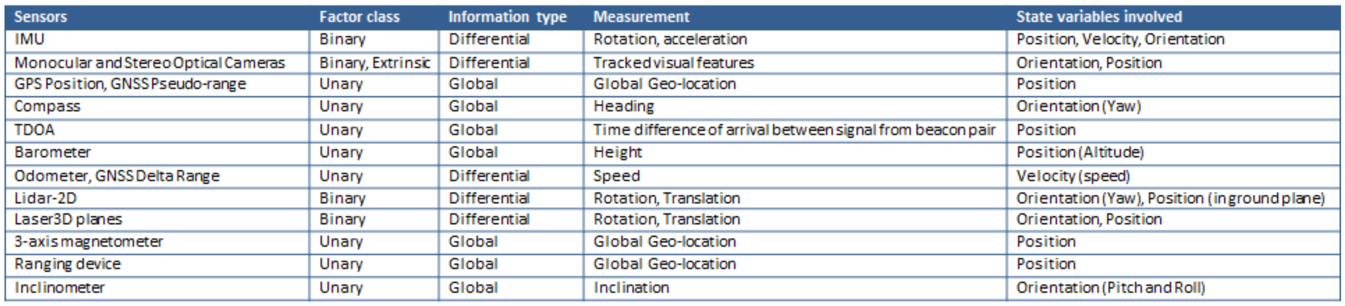
where Θ_i is the set of all state variables θ_j involved in factor f_i, and independent relationships are encoded by edges e_{ij}.

A generative model

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

predicts a sensor measurement z_i using a function $h_i(\Theta_i)$ with measurement noise v_i. The difference between 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

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



Sensors	Factor class	Information type	Measurement	State variables involved
IMU	Binary	Differential	Rotation, acceleration	Position, Velocity, Orientation
Monocular and Stereo Optical Cameras	Binary, Extrinsic	Differential	Tracked visual features	Orientation, Position
GPS Position, GNSS Pseudo-range	Unary	Global	Global Geo-location	Position
Compass	Unary	Global	Heading	Orientation (Yaw)
TDOA	Unary	Global	Time difference of arrival between signal from beacon pair	Position
Barometer	Unary	Global	Height	Position (Altitude)
Odometer, GNSS Delta Range	Unary	Differential	Speed	Velocity (speed)
Lidar-2D	Binary	Differential	Rotation, Translation	Orientation (Yaw), Position (in ground plane)
Laser3D planes	Binary	Differential	Rotation, Translation	Orientation, Position
3-axis magnetometer	Unary	Global	Global Geo-location	Position
Ranging device	Unary	Global	Global Geo-location	Position
Inclinometer	Unary	Global	Inclination	Orientation (Pitch and Roll)

Fig. 2: Factor properties for 15 sensor types used in our plug-and-play framework on a single ground vehicle.

where $||\cdot||^2_\Sigma$ is the Mahalanobis distance.

The factor graph representation on the full non-linear optimization problem has led to the recent development of an incremental solution, iSAM2 [1]. Using a Bayes tree data structure, iSAM2 keeps all past information and only updates variables influenced by each new measurement. It obtains same result as a batch solution to the full non-linear optimization, and can achieve real-time performance even with the presence of loop closures [8], [13].

B. *Factors for Plug-and-Play Sensor Measurements*

Upon the factor graph representation, we develop a highly adaptable plug-and-play framework to convert sensor measurements with numerous forms into abstract factors. Adding new sensors is directly mapped to addition of graph nodes and measurement factors that connect these nodes.

In our system, we define the navigation state for a given time as $x = \{p, v, b\}$. Each state x covers three kinds of nodes: pose node *p* includes 3d translation *t* and 3d rotation *r*, velocity node *v* represents 3d velocity, and *b* denotes sensor-specific bias block which are varied for different sensors. Pose node and velocity node are included in every navigation state, while sensor-specific bias block are propagated through only the navigation states which have measurements from the correspondent sensor. To simplify the notation, we assume all sensors have the same center, which is the origin of the body coordinate system.

Figure 2 shows the factor properties for sensors used in our framework. We group these factors into three different classes: unary, binary, and extrinsic. The factor class is defined by sensor measurement connectivity: the navigation states each measurement involves. The detail factor formulation is based on measurement information, noise characteristics, and the portion of navigation state variables that a measurement affects.

Note measurements from simple sensor types can be directly fed into factor graphs. However, complex sensor types such as cameras require pre-processing mechanisms to derive meaningful measurements by finding associations between sensed data. We discuss factors for different sensor types as follows.

1) IMU Motion Factor: A single factor typically encodes only one sensor measurement. However, IMU sensors produce measurements at a much higher rate than other sensor types. To fully utilize high-frequency IMU data while saving the time to create factors, we design a single factor to summarize multiple consecutive IMU measurements. A navigation state is only created at the time when a non-IMU measurement comes or no non-IMU measurement arrives after a certain interval (such as one second), and the IMU factor is built to connect two sequential navigation states by integrating IMU measurements between them (see Figure 1).

We formulate this factor using an error-state IMU propagation mechanism [9], and implement it [8] as a binary factor between two consecutive states x_{i-1} and x_i. It generates 6 degrees of freedom relative pose and corresponding velocity change as the motion model. It also tracks the IMU-specific bias as part of the state variables for estimating motion.

We use this factor instead of traditional process models in our system. The linearization point to integrate non-IMU measurements at x_i is computed from this factor, which is based on the linearization point for x_{i-1} and IMU readings between x_{i-1} and x_i. If there is no IMU available, we use constant velocity assumption as the process model.

In contrast to tradition filtering techniques, the IMU motion factor is part of the full non-linear optimization process in factor graph. The value of IMU integration changes during re-linearization for iterative optimization.

2) Unary Factor: The unary factor only involves a state at a single time. The model for measurement z with added noise v arriving at time i is as follows.

$$z = h(x_i) + v \tag{4}$$

The measurement can affect different variables in navigation state $x_i = \{p_i, v_i, b_i\}$, depending on its sensor type. For example, GPS position measurement updates 3d translation t_i in pose node p_i. Compass measurement involves the heading (yaw) of 3d rotation r_i in pose node p_i.

3) Binary Factor: A binary factor involves two navigation states at different times. The model for measurement z with added noise v between time $i-1$ and time i is as follows.

$$z = h(x_{i-1}, x_i) + v \tag{5}$$

One example is the derived measurements from 2D Lidar sensors. Each measurement describes the change in location and orientation on the ground plane between two time instances. It is realized as a relative pose between p_{i-1} and p_i, without contributions to altitude, roll and pitch. The IMU motion factor can be also viewed as binary factor, although it incorporates multiple measurements inside one single factor.

4) Extrinsic Factor: Each extrinsic factor involves a navigation state and an unknown extrinsic entity. The landmark association measurement derived from cameras is the most popular example (see Figure 1). The model for measurement z with added noise v involves both navigation state at time i and the state of unknown landmark position l.

$$z = h(x_i, l) + v \tag{6}$$

Using these measurements to estimate both navigation states and landmark positions simultaneously is very popular in SLAM problem formulation. It is also known as bundle adjustment [3] in computer vision.

5) Tracked Visual Features: In our system, visual feature derived from camera data is a special case, which is modeled by two factor classes: binary and extrinsic. In this paper, we process only continuously tracked features from cameras. However, loop-closures can still be incorporated using a parallel architecture, such as [8], [13]. We use the 3-stage method [8] to represent each visual feature tracked across multiple navigation states. This method is based on the maturity of the estimation of the underlying 3D location of the landmark.

The first stage avoids unstable initialization of the 3D location of the landmark points while still incorporating the landmark image observation into binary factor formulation for optimization. The second stage utilizes the extrinsic factor to estimate both navigation states and the 3D location of the associated landmark. Once the uncertainty of the 3D landmark state becomes small, the third stage switches back to binary factor formulation but treats the computed 3D location of the landmark as a fixed quantity when estimating the navigation state, saving computation time.

6) Non-Gaussian Noise Models: To adapt to more sensor types, we extend our framework to support non-Gaussian noise distributions in factors. These include robust error models based on loss functions, and multi-modal noise models using the EM algorithm.

This extension is scalable to new evolving sensors. One example is 3d-axis magnetometer, which relies on a pre-built database and reports 20 possible geo-locations (at most one is true) as signal peaks based on current robot position. Each peak is a Gaussian prior on robot position, and the collection of 20 peaks is treated as mixture of Gaussians for the error model in the 3d-axis magnetometer factor.

The 3d-axis magnetometer factor may be re-evaluated multiple times during optimization, which effectively changes the weighting of possible positions at each iteration. This EM procedure eventually makes the global location estimation converge to the closest peak that is consistent with estimation based on all other sensors.

C. Handling of Delayed Measurements

Our framework is able to naturally handle out-of-order measurements due to high latencies. Figure 1 shows when a delayed measurement from the new added barometer arrives, the system first locates the IMU motion factor between x_2 and x_4 which covers the actual measurement time. It then creates a new navigation state x_3 for this delayed measurement, and properly divides the located IMU motion factor into two new IMU motion factors which connect these three states (x_2,x_3,x_4). Finally it adds a new unary factor for this delayed barometer measurement to navigation state x_3.

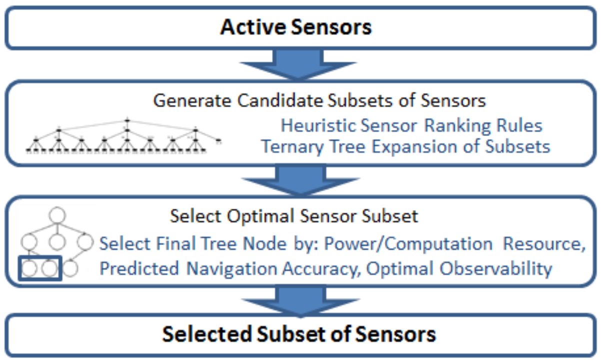

Active Sensors
Generate Candidate Subsets of Sensors
Heuristic Sensor Ranking Rules
Ternary Tree Expansion of Subsets
Select Optimal Sensor Subset
Select Final Tree Node by: Power/Computation Resource, Predicted Navigation Accuracy, Optimal Observability
Selected Subset of Sensors

Fig. 3: The procedure of our constrained optimal selection mechanism.

D. Periodic Updates

Our framework generates the navigation estimation at a particular rate, which is set according to application requirement. It collects factors during the specified time interval, and added them into the factor graph for inference periodically. This way avoids performing updates every time when a new state gets created, which increases computational overhead if there are high-frequency non-IMU sensors.

IV. Constrained Optimal Selection

In this section we propose a new constrained optimal selection mechanism to identify the optimal subset of active sensors, as shown in Figure 3. We show the details on how we select the subset among candidates as ternary tree nodes by maximizing observability coverage on state variables, while following system constraints and accuracy requirements. Note this mechanism needs to be applied not only in the initial period, but also when sensor configuration (such as a sensor goes online, offline, or the quality of its measurements dramatically decreases due to environment change) or system constraint changes. We also use a maturity-based method to select visual measurements for reducing computation without sacrificing performance.

A. Generate Candidate Subsets of Sensors

Our goal is to find a subset of sensors, which is able to provide satisfactory navigation solution under constrained resource, for a given environment. Finding the optimal solution to sensor selection is generally computationally intensive, since this problem is essentially combinatorial in nature. To identify the optimal subset of sensors as soon as possible, our mechanism leverages heuristic sensor ranking rules and the hierarchical structure in a ternary tree.

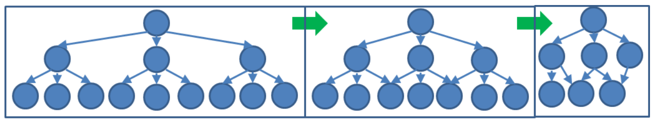


Fig. 4: Three steps (from left to right) to expand level-3 of the ternary tree: generating, merging, and pruning. In this case, the upper limit of nodes for each level is three.

1) Heuristic Sensor Ranking Rules: We first sort all available sensors based on coverage and amount of information on state variables they measured, leading to heuristic rules for sensor ranking. For example, IMU is always first selected because it provides our motion model. Powerful absolute sensor types such as GPS are selected with high priority if their signals are available. Camera ranks higher than compass, because it covers 5 degrees of freedom of relative pose while compass only measures the heading direction.

2) Ternary Tree Expansion of Subsets: We use a ternary tree expansion algorithm to construct candidate subsets of sensors in a greedy manner. In the ternary tree, each node corresponds to a possible subset of sensors. Nodes at the same level of the tree have same number of sensors in the subset. The root (level-1) of the tree includes only 1 sensor with highest ranking (typically IMU).

Figure 4 shows three steps to expand child nodes at next level: generating, merging, and pruning. In the first step, each node at current level generates at most three child nodes by adding one more sensor into its subset of sensors. For example, each level-2 node includes the inherited level-1 sensor and one new sensor. Sensors with higher ranking will be selected with higher priority.

The second step merges nodes with the same subset of sensors at the new level to avoid repeated sensor combinations. For example, two different parent nodes with subset (A,B) and (B,C) respectively may both generate a node with same subset (A,B,C). These two nodes need to be merged into one node. The third step prunes the nodes at the new level according to an upper-bound threshold, which is set based on system resource. Only nodes with higher overall sensor ranking of its subset are kept, and will be used to generate child nodes for the next level.

After iterating the tree expansion algorithm for each level, it generates only one node at the last level, which represents all available sensors.

B. Select Optimal Sensor Subset

Upon the completely built ternary tree, our mechanism locates only a portion of tree nodes based on available resource and desired accuracy in real-time system. The optimal subset of sensors is then identified based on observability index evaluation from these selected tree nodes as candidates.

1) Available Resource: The required computation and power resource is calculated for each node, and only nodes within power and resource budgets are considered. Currently we assume each sensor occupies one resource unit (same fixed amount of computational cost and power usage), and only consider nodes with subset size less than the maximum number of units the system can afford. For example, if the system only allows three sensors to be used for navigation, our mechanism only locates tree nodes at level-3.

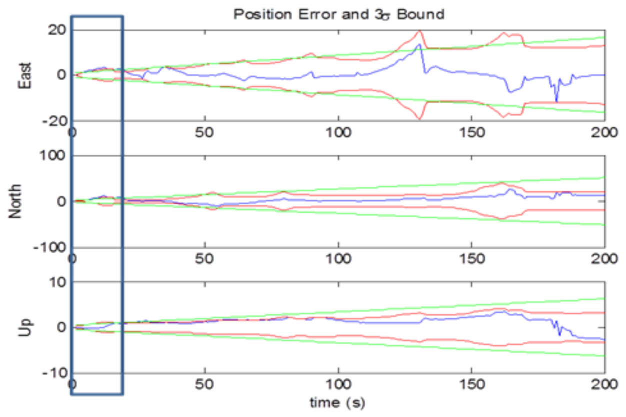


Fig. 5: Position error (blue) and 3 sigma bound (red) along east, north, and up direction respectively for a 200-second scenario. The predicted drift rate (green) is generated using linear regression on samples of 3 sigma bound during the first 20 second (inside the blue box). Note the actual error is within the estimated 3 sigma bounds.

2) Navigation Accuracy: For each selected node, we build and update a factor graph over a very short period of time (such as 20 seconds) using the sensor subset chosen by the node. The size of factor graph is small for such a short period, and the number of selected nodes at each level is limited by a threshold. This way allows the system to operate selected nodes at different parallel threads.

We then use linear regression to fit a predictive error model to 3 sigma bound samples from the posterior covariance in each factor graph. 99.7% of error estimates should be within 3 sigma bound based on the empirical rule. The fitted model is an indication of upper limit of 3D position error in the future, as shown in Figure 5. Note, sensor subsets which include any absolute sensors such as GPS often have lower predicted drift rates due to smaller state uncertainty. Hence, they have a bigger chance of being selected within the computation and power budgets in real-time system.

This fitted model can be interpreted as the predicted drift rate, which represents the upper bound of the predicted 3D position error after a period of time. It is used to select nodes which would maintain the desired navigation accuracy in the future. For example, if our system aims to maintain 3D location error less than 10 meters after 10 minutes, we only consider nodes with predicted drift rate less than 0.0167 meter per second for final evaluation. Note if there are no nodes satisfying the desired accuracy, the system would choose the most optimal subset of sensors which can be operated within the power and computation resource budgets.

3) Observability Index Computation: We find the most optimal subset of sensors by comparing observability index on factor graphs from selected nodes. Different observability indexes [6] have been used to choose a set of pose measurements that can optimally calibrate robots. We apply the same idea to this new problem: select a subset of sensors that optimizes the navigation estimation.

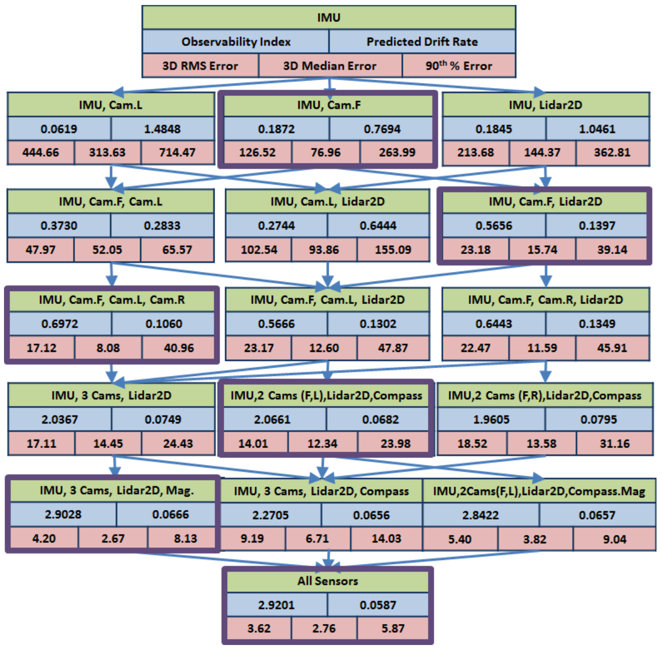


Fig. 6: The expanded ternary tree of 7 available sensors on our ground vehicle for Scenario 1. The upper limit of the number of nodes for each level is 3. For each node, the first row represents the candidate subset of sensors. The second row shows (from left to right) observability index and predicted drift rate (meter/second) computed from initial 20 seconds for our selection. The node with maximum observability index is highlighted for each level. To verify our selection, the third row shows (from left to right) final 3D position RMS error (meter), median error (meter), and 90th % error (meter) over entire 420-second scenario (2.58 kilometers).

We use the minimum singular value of the information matrix as observability index. This observability index represents the worst observability of the state variable error as the criterion, and is best to minimize uncertainty of the estimation. Since the information matrix I is the basic structure [1] to store the entire factor graph for inference, we directly retrieve I from the factor graph for each located node. And we perform singular value decomposition on I.

$$I = U \Sigma V^* \tag{7}$$

Note this observability index σ_{min} is the minimum diagonal value of Σ decomposed from I. We then select the optimal subset of sensors as the node n, which has the factor graph with maximum observability index.

$$node\, n \;= \underset{node\, n}{arg\, max}\, (\sigma_{min}) \tag{8}$$

Our mechanism hence selects the optimal subset among candidates by maximizing observability coverage on state variables, while satisfying resource constraints and accuracy requirements. The final decision is based on the information matrix which is the inverse of posterior state covariance of the factor graph. It therefore utilizes state observability, sensor complementarity, static sensor properties, and dynamic performance properties from fusion of sensor measurements. The system then only uses the selected optimal subset of sensors with the correspondent factor graph to compute navigation estimation.

C. *Maturity-Based Measurement Selection*

Since we treat each tracked visual feature as a single measurement, camera sensor is the only sensor type that generates multiple measurements at the same time. To further reduce computation while still maintaining accuracy, we develop a maturity-based method to select feature measurements within the 3-stage representation (Section III-B.5).

This method is triggered by two conditions. The first condition occurs when there are enough state-2 and stage-3 features at the same navigation state. We then only use later stage features for estimation, since optimized 3D features are more valuable.

The second condition happens when there are too many stage-1 features, typically during slow motion. These features have poor 3d information due to short moving distance. However, the track length (the number of tracked frames) can be long for these features. Therefore, when the number of stage-1 features exceeds a threshold, we analyze track length distribution among these features. Then we select longer tracks based on a length threshold dynamically decided by the number of features we want. This way keeps features which are relatively more valuable and saves computation.

V. Experimental Results

This section demonstrates that our approach provides real-time navigation solutions with desired accuracy under various resource constraints, by selecting subsets of sensors on two large-scale real scenarios. These two scenarios provide different aspects to show the strength of our approach. Both data sets are collected using a car that drives inside a city. The car stops many times during navigation due to traffic signs. The initial global position and orientation of the vehicle is assumed known. Ground Truth is obtained by using the RTK differential GPS technique [15].

A. *Scenario 1: Initial Constrained Optimal Selection*

The first data set has 7 sensors (from 5 sensor types) available for the entire scenario, including one 100-hz IMU, three 4-hz monocular cameras, one 1-hz 2d lidar scanner, one 1-hz compass, and one 10-hz 3d-axis magnetometer. Three cameras are placed at front side, left side, and right side of the car respectively. The total travel distance is 2.58 kilometers, and the travel time is 420 seconds.

1) Sensor Selection: Figure 6 shows the expanded ternary tree of 7 available sensors. Since the active sensor configuration never changes, we only apply the selection mechanism during the initial period. We set the upper limit of the number of nodes as 3 for each tree level, after 3-step expansion. The underlying heuristic sensor ranking for these 7 sensors (from high to low) is Camera Front (Cam.F), 2d lidar scanner (Lidar2D), Camera Left (Cam.L), Camera Right (Cam.R), compass, and 3d-axis magnetometer (Mag.).

For sensor ranking, the value of information from cameras depends on the quality of tracked features. Due to its placement, front camera typically perceives more longer tracked features than side cameras. Compared to side cameras, 2d

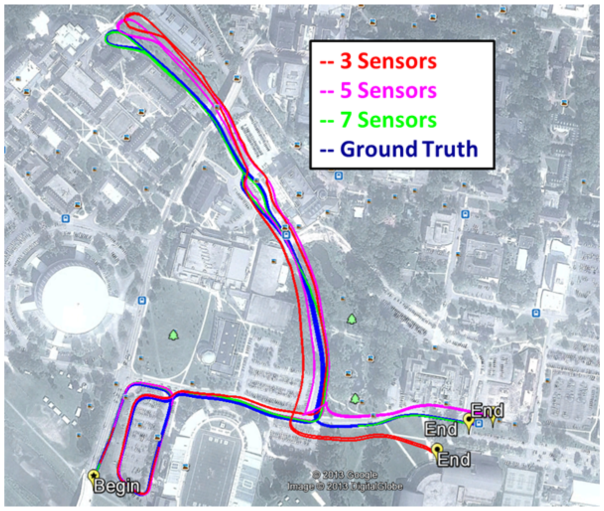


Fig. 7: Ground truth (blue), and navigation trajectories estimated using 3 sensors (red, 3D RMS error is 23.18 meters), 5 sensors (pink, 3D RMS error is 14.01 meters), and 7 sensors (green, 3D RMS error is 3.62 meters) selected from the ternary tree in Figure 6.

lidar usually provides more stable information while only updating relative 2d location and orientation in the ground plane. The global 3d location measurements from magnetometer are opportunistic depending on the vehicle's position, so it has lowest ranking.

Our selection mechanism locates tree nodes based on the number of allowed sensors (tree levels) and the predicted drift rate, computed from the initial 20 seconds. It selects the optimal subset of sensors from located nodes with maximum observability index, which is also measured from the same initial period. For example, if the system allows at most 5 sensors with predicted drift rate less than 0.075 (meter/sec), there are two candidates (See Figure 6). One is IMU, 3 cameras, and Lidar2D. The other is IMU, 2 Cameras (F,L), Lidar2D, and compass. The system will select the second candidate due to its bigger observability index. This selected set eventually generates navigation solution with 3D RMS error in 14.01 meters, best among all level-5 tree nodes. Note the overall 3D RMS error of the trajectory is less than the predicted error (our upper-limit predicted drift rate multiplied by navigation time) computed at the start.

We highlight the node with maximum observability index for each level in Figure 6. The highlighted node generates lowest final 3D position error among choices with same number of allowed sensors. Combining information from more sensors improves performance. The 3D RMS error is reduced from 126.52 meters (2 sensors) to 3.82 meters (7 sensors). Figure 7 shows the estimated trajectory is closer to Ground Truth by adding more sensors. The ternary tree also exhibits the effectiveness of different sensor combinations. For example, the mixture of front camera and 2d lidar is more valuable than any two cameras.

2) *Visual Measurement Selection:* Note we set the update rate (inference frequency) as 1-Hz to generate navigation solutions (see Section III-D) for all experiments. As shown in Figure 8, the number of tracked features from a camera

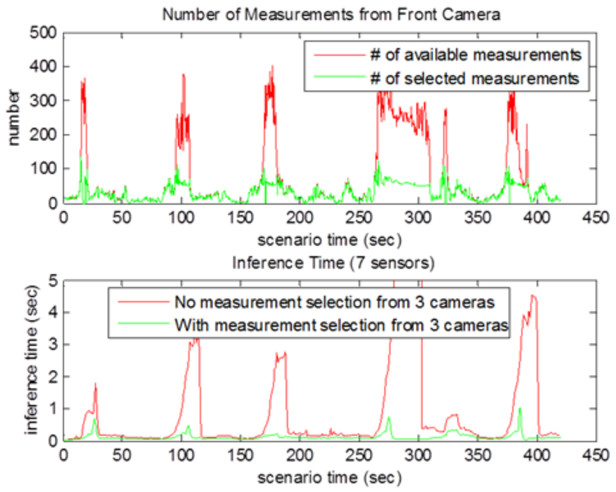


Fig. 8: (Top) The number of available measurements and selected measurements from the front camera. (Bottom) The inference time without (3D RMS error: 3.59 meters) and with measurement selection (3D RMS error: 3.62 meters) from 3 cameras, using all 7 sensors in our system.

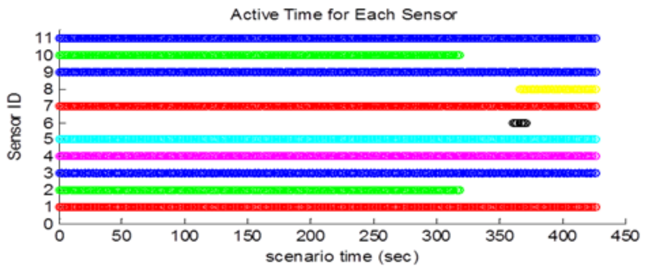


Fig. 9: The active period for each sensor in Scenario 2 (426 seconds, 1.61 kilometers). Sensor ID (y-axis) - 1: IMU, 2: pesudo-range. 3: camera A, 4: magnetometer, 5: compass, 6: ranging device, 7: barometer, 8: odometer, 9: camera B, 10: delta-range, 11: camera C. There are at most 9 sensors available at a given time (x-axis).

dramatically increases (more than 200 features) when the car stops. Since we treat each tracked feature as an individual measurement, the inference time grows as the number of measurement increasing.

Using our maturity-based measurement selection method described in Section IV-C, the number of selected features from each camera can be roughly controlled under a threshold, which is set as 100 features based on our system resource. Our incremental optimization hence takes less than 1 second on each update for this 7-sensor scenario to achieve real-time navigation without sacrificing accuracy (3D RMS error increases only 3 centimeters with visual measurement selection). These timing results were conducted using a single core of an Intel i7 CPU running at 2.70 GHz.

B. Scenario 2: Plug-and-Play Navigation

The total travel distance for the second scenario is 1.61 kilometers, and the travel time is 426 seconds. This data set has 11 sensors (from 9 sensor types), including one 100-hz IMU, three 4-hz monocular cameras, one 1-hz compass, one 10-hz 3d-axis magnetometer, one 1-hz ranging device, one 25-hz barometer, one 1-hz odometer, one 1-hz perudo-range, and one 1-hz delta-range. Both pseudo-range and delta-range

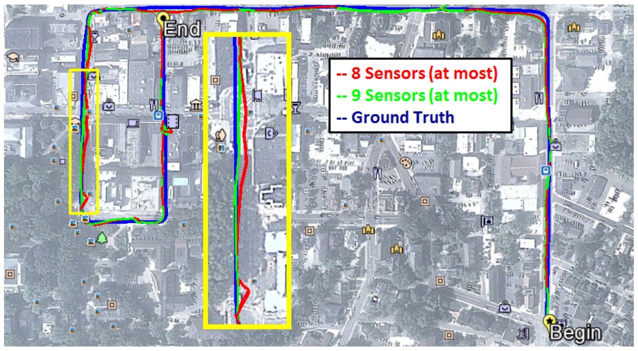


Fig. 10: Ground truth (blue), and navigation trajectories estimated using at most 8 sensors (red: 3D RMS error is 6.73 meters), and at most 9 sensors (green: 3D RMS error is 5.42 meters), selected from our mechanism for Scenario 2. The yellow enlarged portion shows that incorporating one more sensor (compass) improves the overall solution.

measurements are generated by signals from only 1 satellite.

However, not all sensors are available all the time. A few sensors get plugged-in or become unavailable during navigation. There are at most 9 sensors available at a given time. Figure 9 shows the active period for each sensor. When a sensor gets online or offline, our mechanism re-evaluates the addition or removal of a sensor in the selected subset.

We use this data set to show our framework allows plug-and-play sensing during navigation. We apply our constrained optimal selection mechanism to decide the subset of active sensors, during initial period and when sensor configuration changes. Figure 10 shows the accuracy can be improved if the system affords to fuse measurements from more sensors. The 3D RMS error is reduced from 6.73 meters (at most 8 sensors) to 5.42 meters (at most 9 sensors). Based on our mechanism, the last sensor that gets used is compass sensor. In addition, even using all possible sensors in this scenario, our incremental optimization still achieves real-time performance at 1-hz update rate.

VI. Conclusions

In this paper, we present a robotic solution to address a new challenging problem for real-time navigation: identify and incorporate an optimal subset of sensors among many online sensors to achieve desired accuracy under system constraints. We described a framework which expands factor graph formulation to encode sensor measurements with different rates, latencies, and noise distributions. It allows plug-and-play sensing, and incorporates new evolving sensors.

We introduced a novel constrained optimal selection mechanism to gain the biggest improvements from sensor combinations in the most efficient way. This mechanism uses a ternary tree expansion algorithm to construct candidate subsets of sensors as tree nodes based on heuristic sensor ranking rules. It identifies the optimal subset among candidates by maximizing observability coverage, while satisfying both available resource (the upper limit of allowed sensors) and navigation accuracy (predicted drift rate), at a very short period of time. Experiments demonstrate our approach provides best solutions among choices of sensor combinations on large-scale scenarios, based on various system needs.

Future work is to ensure our approach operates under any kind of system conditions. We plan to estimate detailed power and computation necessary for different sensors, instead of assuming all sensors take same amount of resource. This way can further optimize navigation performance under limited system resource.

Acknowledgments

This material is based upon work supported by the DARPA All Source Positioning and Navigation (ASPN) Program under USAF/ AFMC AFRL Contract FA8650-13-C-7322. 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] 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.
[4] H. Strasdat, C. Stachniss, and W. Burgard, "Which landmark is useful? Learning selection policies for navigation in unknown environments," in Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), 2009.
[5] J. Farrell, "Aided navigation: GPS with high rate sensors," McGraw-Hill, 2008.
[6] Y. Sun and J. Hollerbach, "Observability index selection for robot calibration," in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2008.
[7] F. Kschischang, B. Fey, and H. Loeliger, "Factor graphs and the sum-product algorithm," IEEE Trans. Inform. Theory, vol. 47, no. 2, Feb 2001.
[8] H. Chiu, S. Williams, F. Dellaert, S. Samarasekera, and R. Kumar, "Robust vision-aided navigation using sliding-window factor graphs," in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2013.
[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] S. Sukumar, H. Bozdogan, D. Page, A. Koschan, and M. Abidi, "Sensor selection using information complexity for multi-sensor mobile robot localization," in Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA), 2007.
[11] N. Rao, "On fusers that perform better than best sensor," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 8, Aug. 2001.
[12] M. Li and A. Mourikis, "Vision-aided inertial navigation for resource-constrained systems," in Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), 2012.
[13] 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.
[14] 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.
[15] J. Sinko, "RTK performance in highway and racetrack experiments," Navigation, Vol. 50, No. 4, pp 265–275, 2003.
[16] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in Proc. IEEE Intl. Conf. on Decision and Control (CDC), 2010.
[17] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," IEEE Transactions on Robotics, vol. 24, no. 5, Oct. 2008.