Resource-Aware Large-Scale Cooperative 3D Mapping using Multiple Mobile Devices

Chao X. Guo, Kourosh Sartipi, Ryan C. DuToit, Georgios A. Georgiou, Ruipeng Li, John O'Leary, Esha D. Nerurkar, Joel A. Hesch, and Stergios I. Roumeliotis

***Abstract*—In this paper, we address the problem of cooperative mapping (CM) using datasets collected by multiple users at different times, when the transformation between the users' starting poses is unknown. Specifically, we formulate CM as a constrained optimization problem, where each user's independently estimated trajectory and map are merged together by imposing geometric constraints between commonly-observed point and line features. Additionally, we provide an algorithm for efficiently solving the CM problem, by taking advantage of its structure. The proposed solution is proven to be batch-least-squares (BLS) optimal over all users' datasets, while it is less memory demanding and lends itself to parallel implementations. In particular, our solution is shown to be faster than the standard BLS solution, when the overlap between the users' data is small. Furthermore, our algorithm is resource-aware as it is able to consistently trade accuracy for lower processing cost, by retaining only an informative subset of the common-feature constraints. Experimental results based on visual and inertial measurements collected from multiple users within large buildings are used to assess the performance of the proposed CM algorithm.**

***Index Terms*—cooperative mapping, 3D mapping, visual and inertial sensor fusion, constrained optimization problem, resource-aware system**

I. Introduction and Related Work

Creating an accurate 3D map within a GPS-denied area is required for many applications, such as human (or robot) indoor navigation and localization, augmented reality, and search and rescue. A camera and inertial measurement unit (IMU) sensor pair, typically found on most modern mobile devices (e.g., smart phones, tablets, wearable computers), is ideal for such task due to their complementary sensing capabilities. To date, most research has focused on how to efficiently create a map from *a single* dataset (e.g., [1], [2], [3], [4]) using batch least-squares (BLS) methods. In many practical applications, however, the device used for recording data may not have sufficient resources (e.g., storage space or battery) to collect data covering a large area. Additionally, it may not be convenient for a single user to navigate the whole building at once. Furthermore, any time a portion of the map changes, or is deemed of insufficient quality or accuracy, the mapping process must be repeated.

The aforementioned limitations motivate investigating cooperative mapping (CM) methods that can process visual and inertial data collected by multiple users at different times. In particular, we are interested in the most general case where the transformation between the users' starting positions and orientations (poses) is not known.

Early work on CM [5], as well as more recent distributed approaches [6], determine the relative pose between users by aligning their individual maps without estimating the resulting combined map. In these methods, a global optimization over all users' measurements is missing, resulting in a non-optimal solution. Other approaches employ relative pose measurements among the users (or robots) to determine the transformation between their trajectories [7] and maps [8], [9]. Such methods require the users to visit the same location at the same time for capturing inter-user measurements, which greatly reduces their applicability to general use cases. To overcome this limitation, recent works employ observations of common features for merging the users' maps. Depending on the allocation of the processing requirements between the users and a central computer, these approaches can be classified into decentralized and centralized.

In decentralized methods, each user creates a merged map independently using both its own measurements and those received from other users. Specifically, in DDF-SAM [10], each robot summarizes its mapping data by first marginalizing its trajectory and non-common features, and then sending to its neighbors the computed inferred measurements relating *only* the commonly-observed features. Since marginalization is computationally expensive, alternative exact and approximate summarization approaches are provided in DDF-SAM 2.0 [11]. Note that [10] and [11] are approximate methods, as the received inferred measurements cannot be relinearized when a better estimate is obtained through BLS iterations. Furthermore, since mapping is a computationally-demanding process and each user creates a merged map on its own, all users need to carry powerful processors to perform many, often redundant, computations.

In contrast, centralized methods maintain a low-processing cost in the user end, by allocating expensive computations to a central computer (e.g., cloud server). In particular, Riazuelo *et al.* present an extension to PTAM [12] in which each user optimizes over *only* its own trajectory assuming the map is known, while the cloud server optimizes over the merged map assuming all users' trajectories to be known [13]. In [14]

C. X. Guo and E. D. Nerurkar are with Google Daydream, Mountain View, CA 94043, USA (email: chaoguo|eshanerurkar@google.com)

K. Sartipi, R. C. DuToit, and S. I. Roumeliotis are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, USA (e-mail: sarti009|dutoi006|roume003@umn.edu)

G. A. Georgiou is with GM Cruise, San Francisco, CA 94103, USA (e-mail: georg529@umn.edu)

R. Li is with Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, USA (email: li50@llnl.gov)

J. O'Leary is with Apple, Sunnyvale, CA 94085, USA (email: olear121@umn.edu)

J. A. Hesch is with Facebook Oculus, Menlo Park, CA 94025, USA (email: joel@umn.edu)

and [15], each user runs onboard visual odometry and sends the estimated trajectory to a server. The server then computes the BLS solution over only the users' trajectories [14] or a selective set of users' poses and their observed landmarks [15]. The centralized algorithms of [13], [14], and [15] combine datasets collected by multiple users into a single one, and then solve the resulting BLS problem. Such strategy has two limitations: (i) It employs the standard BLS formulation for solving the CM problem (i.e., they express all variables of interest w.r.t. a single common frame of reference), and does not explicitly consider one of the problem's key characteristics: When collaboratively mapping a building, users typically collect data from different areas with a small overlap so as to maximize coverage. These datasets, in general, comprise a limited number of common features and thus constraints between their trajectories. In contrast, the user poses within each dataset are "strongly" constrained by not only loop-closure measurements but also IMU data and feature tracks across consecutive images. In our CM formulation, we take advantage of this structure by expressing the variables of each map in its own frame while ensuring coherence between the maps by imposing common-feature constraints; this leads to significant speedups or memory savings. (ii) [13], [14], [15] employ approximations (e.g., removing all features from the estimated state as in [14]), whose impact on processing is difficult to assess. Specifically, the BLS solving time depends not only on the problem size and number of non-zero elements, but also on the structure of the Hessian matrix. Therefore, since reducing the number of variables, and thus processed measurements, will significantly affect the Hessian's structure, one cannot tractably predict the processing savings. Instead in our work, we first present the exact CM solution and then offer an adjustable approximation which selects and applies a subset of the common-feature constraints, while optimizing over all variables of interest. As shown in Sect. V-C, our formulation allows quantifying the expected gain in speed as a function of the number of common-feature constraints.

Furthermore, most works to date on localization and mapping have focused on fusing inertial measurements with only *point-feature* observations extracted and matched across sequences of images. In such systems, the rotation around the gravity direction (yaw) has been shown to be unobservable [16]. To obtain additional attitude information, *line* features, especially those aligning with the cardinal directions of a "Manhattan world"[1], have been used together with point features. Specifically, both point and line features were employed for improving the localization and mapping accuracy of vision-only [17], and visual-inertial [18], [19] filtering algorithms. In the context of BLS, [20] simultaneously estimates the camera motion and the surrounding structure using point and line features. This method, however, requires observing six lines (three parallel and three non-parallel) in triplets of consecutive images. Additionally, [20] decouples the camera motion into two rotations and two translations, and solves for each of them successively, thus resulting in a sub-optimal estimate. When lines are employed for localization and mapping, line-loop-closure measurements are rarely used in practice, with the exception of (to the best of our knowledge) [21] and [22]. Specifically, Lee *et al.* [21] use line features for loop-closure detection, but not as measurements for estimation. On the other hand, Zhang *et al.* [22] propose a method for utilizing loop-closure line measurements, but under the limiting assumption that the observed lines either reside within, or are perpendicular to, the plane corresponding to the floor.

Our proposed centralized CM algorithm provides a solution that is mathematically equivalent to the BLS solution, takes advantage of the problem structure to gain in efficiency, and utilizes both consecutive and loop-closure observations of point and line features to achieve high accuracy. Specifically, we formulate CM as a constrained optimization problem, in which the cost function is expressed as the summation of the cost functions from all users, while the constraints express the geometric relationship between the features commonly observed by two or more users. In particular, the main contributions of this paper are:

- The proposed CM formulation is modular (i.e., maps or submaps can be added or removed), lends itself to parallel implementation, and the solution is memory efficient. In addition, the proposed algorithm is able to leverage each individual user's intermediate mapping results to reduce the processing cost.
- Our CM formulation allows for consistently[2] trading estimation accuracy for computational-cost savings by appropriately reducing the number of commonly-observed-feature constraints.
- We utilize points, "free lines" (lines not aligned with the cardinal directions of the building), and "Manhattan lines" to improve the estimation accuracy. Additionally, to the best of our knowledge, we are the first to detect and apply loop-closure line measurements in a visual-inertial mapping system.
- We quantitatively validate our algorithm in large-scale 3D experiments using datasets collected from multiple mobile devices. Additional results from various buildings and conference sites are available through our online interactive visualization [24].

A previous, shorter version of this paper was presented at [25]. As compared to [25], this paper presents the commonly-observed feature sparsification technique that enables the proposed CM algorithm to trade estimation accuracy for computational savings. Additionally, we provide a thorough experimental evaluation of the CM algorithm's processing cost and memory requirements, as well as a comparison to the standard BLS solution.

The rest of the paper is structured as follows: In Sect. II, we define the CM problem and present an overview of our proposed algorithm. In Sect. III, we provide the parameterization and measurement models for point, free-line, and Manhattan-line features. In Sect. IV, we present the geometric constraints

[1]The Manhattan world assumption states that the world is built on a Cartesian grid and hence all the surfaces are aligned with three dominant perpendicular directions.

[2]As defined in [23], a state estimator is consistent if the estimation errors are zero-mean, and the estimated covariance is not smaller than the true covariance.

that arise when two features, defined in two separate user maps, correspond to the same physical point or line. In Sect. V, we briefly review the BLS solution for a single user, explain our method for finding the initial relative-pose estimate between users, and present our efficient CM algorithm in detail. In Sect. VI, the proposed method is thoroughly evaluated both in terms of accuracy and processing cost. Concluding remarks and future directions of research are provided in Sect. VII.

II. Algorithm Overview

Consider multiple datasets consisting of visual and inertial measurements collected by several users with a camera and an IMU. We examine the most general case, where the relative transformations of the users are unknown, and no relative-pose measurements between them are provided. Furthermore, we assume that there exist enough (two or more) common point features between pairs of users to determine the transformation between all maps. Such a multi-user CM scenario is illustrated in Fig. 1.

The objective of this work is to find a BLS solution over all users' trajectories and maps. Our algorithm comprises three main steps:

1) Obtain a BLS solution for each individual user's trajectory and map *independently*, using measurements from only their dataset.
2) Generate an initial estimate of the users' relative poses, using their observations of common point features.
3) Find the optimal BLS solution of all users' trajectories and maps utilizing all available sensor data, and all, or a subset of, the constraints that arise from commonly-observed point and line features.

Our formulation of the CM problem has three desirable characteristics: (i) Each user's dataset becomes a modular component of the final solution. Thus, if a user's dataset contains unreliable measurements, or we need to extend the map to a previously-unknown area, we can remove or add users to the CM problem without recomputing all BLS solutions or all initial relative-pose estimates; (ii) The algorithm can reuse the result of each individual BLS when generating the CM solution, leading to processing savings; (iii) The CM algorithm provides a convenient mechanism for trading estimation accuracy for computational-cost savings, by reducing the number of constraints imposed by commonly-observed features.

III. System State and Measurement Models

In this section, we first describe the system states, and then present the measurement models for processing IMU data and visual observations to point, free-line, and Manhattan-line features.[3]

For the remainder of the paper, we denote the position and orientation of frame $\{F_1\}$ in frame $\{F_2\}$ as ${}^{F_2}\mathbf{p}_{F_1}$ and ${}^{F_2}_{F_1}\mathbf{C}$, respectively, where $\mathbf{C}$ is a 3×3 rotation matrix. We also define $[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3] = \mathbf{I}_3$, where $\mathbf{I}_3$ is the 3×3 identity matrix. For the reader's convenience, we provide a summary of this nomenclature in the end of the paper.

[3]Note that CM is agnostic to the method employed for finding these feature measurements. Our visual-processing pipeline is detailed in Sect. VI-A.

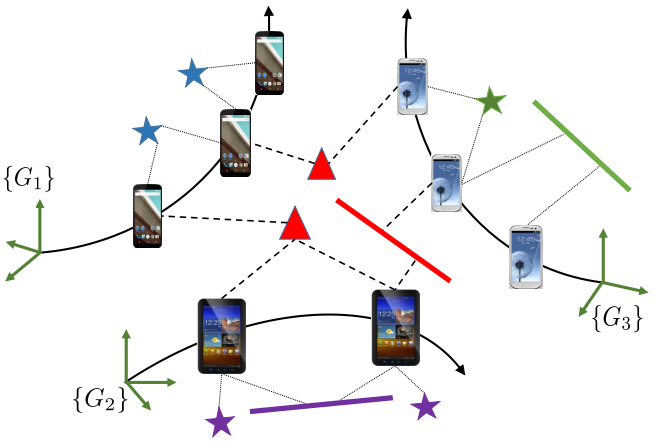


Fig. 1: Non-common (stars) and common (triangles) point features, as well as line features observed by one or more mobile devices.

A. System State

Our CM system maintains the following state vector:

$$\mathbf{x} = \begin{bmatrix} {}^1\mathbf{x}_u^T & \cdots & {}^N\mathbf{x}_u^T & \mathbf{x}_\tau^T \end{bmatrix}^T$$

where ${}^j\mathbf{x}_u$, $j = 1, \ldots, N$, denotes the state vector corresponding to user j, and $\mathbf{x}_\tau$ is the transformation between different users' global frames of reference. Furthermore, ${}^j\mathbf{x}_u$ is defined as:

$${}^j\mathbf{x}_u = \begin{bmatrix} {}^j\check{\mathbf{p}}^T & {}^j\mathbf{f}^T & {}^j\mathbf{q}_B^T \end{bmatrix}^T \tag{1}$$

where ${}^j\check{\mathbf{p}} \triangleq \begin{bmatrix} {}^j\check{\mathbf{p}}_1^T & \cdots & {}^j\check{\mathbf{p}}_K^T \end{bmatrix}^T$, ${}^j\check{\mathbf{p}}_k$, $k = 1, \ldots, K$, represents the user's pose along with their velocity and IMU biases at time step k [see (3)], ${}^j\mathbf{f}$ is the vector of all point, ${}^j\mathbf{x}_{P_i}$, free-line, ${}^j\mathbf{x}_{L_i}$, and Manhattan-line, ${}^j\mathbf{x}_{V_i}$, features defined as

$${}^j\mathbf{f} = \begin{bmatrix} {}^j\mathbf{x}_{P_1}^T & \cdots & {}^j\mathbf{x}_{P_{N_p}}^T & {}^j\mathbf{x}_{L_1}^T & \cdots & {}^j\mathbf{x}_{L_{N_l}}^T & {}^j\mathbf{x}_{V_1}^T & \cdots & {}^j\mathbf{x}_{V_{N_v}}^T \end{bmatrix}^T$$

and ${}^j\mathbf{q}_B$ is the quaternion representation of the user's orientation in the Manhattan world. Defining the z axis of the Manhattan world frame to align with the gravity direction, ${}^j\mathbf{q}_B$ denotes a time-invariant yaw rotation of angle ${}^j\alpha_B$, i.e.,

$${}^j\mathbf{q}_B = \begin{bmatrix} 0 & 0 & sin(\frac{{}^j\alpha_B}{2}) & cos(\frac{{}^j\alpha_B}{2}) \end{bmatrix}^T \tag{2}$$

Additionally, we assume the IMU and the camera are co-located to simplify the ensuing derivations. In our experiments, we estimate the IMU-camera extrinsic calibration parameters (6 dof transformation) of each user concurrently with their trajectories and maps.

Note that the IMU and feature measurement models employed for each user are similar. For this reason, we drop the user index when describing them in the following section.

B. User Pose State and IMU Measurement Model

An IMU (i.e., gyroscope and accelerometer) provides the user's rotational velocity, $\boldsymbol{\omega}_m$, and linear acceleration, $\mathbf{a}_m$, contaminated by white Gaussian noise and time-varying biases. To model the IMU propagation process, we define the user's (augmented) pose at time step k as:

$$\check{\mathbf{p}}_k = \begin{bmatrix} {}^{C_k}\mathbf{q}_G^T & \mathbf{b}_{g_k}^T & {}^G\mathbf{v}_{C_k}^T & \mathbf{b}_{a_k}^T & {}^G\mathbf{p}_{C_k}^T \end{bmatrix}^T \tag{3}$$

where ${}^{C_k}\mathbf{q}_G$ is the orientation of the global frame, $\{G\}$, in the IMU's frame of reference, $\{C_k\}$, ${}^G\mathbf{p}_{C_k}$ and ${}^G\mathbf{v}_{C_k}$ are the position

and velocity of $\{C_k\}$ in $\{G\}$, and $\mathbf{b}_{g_k}$ and $\mathbf{b}_{a_k}$ are gyroscope and accelerometer biases, respectively.

The continuous-time IMU propagation model describing the time evolution of the user's pose state is:

$$\begin{aligned}
{}^{C}\dot{\mathbf{q}}_G(t) &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m(t)-\mathbf{b}_g(t)-\mathbf{n}_g(t)){}^{C}\mathbf{q}_G(t)\\
{}^{G}\dot{\mathbf{v}}_C(t) &= \mathbf{C}({}^{C}\mathbf{q}_G(t))^T(\mathbf{a}_m(t)-\mathbf{b}_a(t)-\mathbf{n}_a(t))+{}^{G}\mathbf{g}\\
{}^{G}\dot{\mathbf{p}}_C(t) &= {}^{G}\mathbf{v}_C(t)\\
\dot{\mathbf{b}}_a(t) &= \mathbf{n}_{wa}\\
\dot{\mathbf{b}}_g(t) &= \mathbf{n}_{wg}
\end{aligned} \tag{4}$$

where $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is defined as $\begin{bmatrix}-\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega}\\ -\boldsymbol{\omega}^T & 0\end{bmatrix}$ and $\lfloor\boldsymbol{\omega}\rfloor$ represents the skew-symmetric matrix of $\boldsymbol{\omega}\in\mathbb{R}^3$, $\mathbf{C}({}^{C}\mathbf{q}_G(t))$ denotes the rotation matrix corresponding to ${}^{C}\mathbf{q}_G(t)$, $\mathbf{n}_g$ and $\mathbf{n}_a$ are white Gaussian measurement noise of $\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$, respectively, while ${}^{G}\mathbf{g}$ denotes the gravitational acceleration in $\{G\}$. Finally, $\mathbf{n}_{wa}$ and $\mathbf{n}_{wg}$ are zero-mean white Gaussian noise processes driving the gyroscope and accelerometer biases $\mathbf{b}_g$ and $\mathbf{b}_a$.

The corresponding discrete-time system is determined by integrating (4) between time steps k and $k+1$, employing the method described in [16], [26], and [27]. This process can be described by the following equation:

$$\check{\mathbf{p}}_{k+1} = \mathbf{g}(\check{\mathbf{p}}_k,\mathbf{u}_k)+\boldsymbol{\zeta}_k \tag{5}$$

where $\mathbf{g}(\check{\mathbf{p}}_k,\mathbf{u}_k)$ is a nonlinear function corresponding to the IMU measurement model, $\mathbf{u}_k \triangleq \begin{bmatrix}\boldsymbol{\omega}_{m_k}^T & \mathbf{a}_{m_k}^T\end{bmatrix}^T$, and $\boldsymbol{\zeta}_k$ is the IMU measurement noise, which is assumed to be zero mean, Gaussian with covariance $\mathbf{Q}_k$, computed through IMU characterization [28].

C. Point-feature Measurement Model

By defining the position of a point feature with respect to the camera pose, $\{C_\eta\}$, that first observes it as ${}^{C_\eta}\mathbf{x}_{P_i}$, the camera pose $\{C_k\}$, measures the bearing angle to the feature as:[4]

$${}^{C_k}\mathbf{z} = \boldsymbol{\pi}\left({}^{C_k}\mathbf{p}_{C_\eta}+{}^{C_k}_{C_\eta}\mathbf{C}{}^{C_\eta}\mathbf{x}_{P_i}\right)+\mathbf{n}_k \tag{6}$$

where $\boldsymbol{\pi}(\begin{bmatrix}x & y & z\end{bmatrix}^T) \triangleq \frac{1}{z}\begin{bmatrix}x & y\end{bmatrix}^T$ represents the camera perspective-projection model, $\mathbf{n}_k$ is the measurement noise, and ${}^{C_k}_{C_\eta}\mathbf{C}$ and ${}^{C_k}\mathbf{p}_{C_\eta}$ can be expressed as:

$${}^{C_k}_{C_\eta}\mathbf{C} = {}^{C_k}_{G}\mathbf{C}\,{}^{C_\eta}_{G}\mathbf{C}^T \tag{7}$$

$${}^{C_k}\mathbf{p}_{C_\eta} = {}^{C_k}_{G}\mathbf{C}({}^{G}\mathbf{p}_{C_\eta}-{}^{G}\mathbf{p}_{C_k}) \tag{8}$$

D. Free-line Feature State and Measurement Model

In this work, we use the same 4 dof free-line parameterization as in [30]. Consider the line $\boldsymbol{\ell}_i$ in Fig. 2 which is first observed by camera pose $\{C_\eta\}$. We define a coordinate frame $\{L_i\}$ for this line whose origin, $\mathbf{p}_{L_i}$, is the point on the line at minimum distance, d_{L_i}, from $\{C_\eta\}$, x-axis is aligned with the line's direction $\boldsymbol{\ell}_i$, and z-axis points away from the origin of $\{C_\eta\}$. Then, the line is represented with respect to $\{C_\eta\}$ by the parameter vector ${}^{C_\eta}\mathbf{x}_{L_i} = \begin{bmatrix}{}^{C_\eta}\mathbf{q}_{L_i}^T & d_{L_i}\end{bmatrix}^T$. Defining ${}^{C_\eta}_{L_i}\mathbf{C} \triangleq \mathbf{C}({}^{C_\eta}\mathbf{q}_{L_i})$, the origin of the line frame in the camera pose $\{C_\eta\}$ can be written as ${}^{C_\eta}\mathbf{p}_{L_i} = d_{L_i}\cdot{}^{C_\eta}_{L_i}\mathbf{C}\mathbf{e}_3$, while the line direction is $\boldsymbol{\ell}_i = {}^{C_\eta}_{L_i}\mathbf{C}\mathbf{e}_1$.

[4]In our implementation, we employ the inverse-depth parameterization with respect to the first observing camera so as to improve numerical stability and also to more accurately represent the depth uncertainty [29]. For simplicity, in the ensuing derivations, we use Cartesian coordinates.

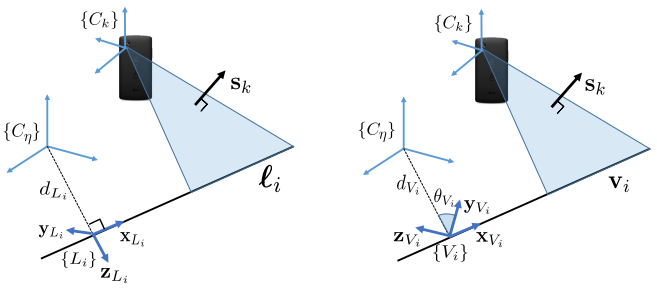


Fig. 2: The parameterization and measurement of free lines (left) and Manhattan lines (right).

In the absence of noise, any line measurement $\mathbf{s}_k$ in frame $\{C_k\}$, which is defined as a 2-dof unit vector perpendicular to the plane passing through the line $\boldsymbol{\ell}_i$ and the origin of $\{C_k\}$, imposes two constraints on the line and the observing camera: $\mathbf{s}_k$ is perpendicular to both the line direction and the displacement between the origins of $\{C_k\}$ and $\{L_i\}$, i.e.,

$$\mathbf{s}_k^T{}^{C_k}_{C_\eta}\mathbf{C}\,{}^{C_\eta}_{L_i}\mathbf{C}\mathbf{e}_1 = 0 \tag{9}$$

$$\mathbf{s}_k^T\left({}^{C_k}_{C_\eta}\mathbf{C}{}^{C_\eta}\mathbf{p}_{L_i}+{}^{C_k}\mathbf{p}_{C_\eta}\right) = 0 \tag{10}$$

where ${}^{C_k}_{C_\eta}\mathbf{C}$ and ${}^{C_k}\mathbf{p}_{C_\eta}$ are expressed in terms of the global poses of cameras $\{C_\eta\}$ and $\{C_k\}$, as shown in (7)-(8). In the presence of noise, the measured normal vector is $\mathbf{s}_k' = \mathbf{C}\left(\mathbf{s}_k^{\perp\!\!\!\perp},n_2\right)\mathbf{C}\left(\mathbf{s}_k^{\perp},n_1\right)\mathbf{s}_k$, where the rotational matrices $\mathbf{C}\left(\mathbf{s}_k^{\perp},n_1\right)$ and $\mathbf{C}\left(\mathbf{s}_k^{\perp\!\!\!\perp},n_2\right)$ express the effect of the noise perturbing the true unit vector $\mathbf{s}_k$ about two perpendicular axes, $\mathbf{s}_k^{\perp}$ and $\mathbf{s}_k^{\perp\!\!\!\perp}$, by angles of magnitude n_1 and n_2, respectively.

E. Manhattan-line Feature State and Measurement Model

Manhattan lines are aligned with one of the building's cardinal directions, and thus have only 2 dof. Assuming a line aligns with the x-axis of the Manhattan world $\{B\}$ (i.e., $\mathbf{v}_i = \mathbf{e}_1$), as in Fig. 2, the Manhattan line with respect to its first observing camera pose $\{C_\eta\}$ is represented by the parameter vector ${}^{C_\eta}\mathbf{x}_{V_i} = \begin{bmatrix}\theta_{V_i} & d_{V_i}\end{bmatrix}^T$, where θ_{V_i} is the angle between ${}^{C_\eta}\mathbf{p}_{V_i}$ and the y-axis of the Manhattan world (i.e., $\mathbf{e}_2$), and d_{V_i} is the distance between the origins of $\{C_\eta\}$ and the Manhattan-line's frame $\{V_i\}$. Using this parameterization, ${}^{C_\eta}\mathbf{p}_{V_i}$ is expressed as:

$${}^{C_\eta}\mathbf{p}_{V_i} = d_{V_i}{}^{C_\eta}_{G}\mathbf{C}{}^{G}_{B}\mathbf{C}(\cos\theta_{V_i}\mathbf{e}_2+\sin\theta_{V_i}\mathbf{e}_3) \tag{11}$$

where ${}^{G}_{B}\mathbf{C} = \mathbf{C}({}^{G}\mathbf{q}_B)$ [see (2)] is the rotation matrix expressing the orientation of the Manhattan world frame $\{B\}$ with respect to the global frame $\{G\}$. Similar to (9)-(10), the geometric constraints corresponding to Manhattan lines are:

$$\mathbf{s}_k^T{}^{C_k}_{C_\eta}\mathbf{C}\,{}^{C_\eta}_{G}\mathbf{C}\,{}^{G}_{B}\mathbf{C}\mathbf{e}_1 = 0 \tag{12}$$

$$\mathbf{s}_k^T\left({}^{C_k}_{C_\eta}\mathbf{C}{}^{C_\eta}\mathbf{p}_{V_i}+{}^{C_k}\mathbf{p}_{C_\eta}\right) = 0 \tag{13}$$

where ${}^{C_k}_{C_\eta}\mathbf{C}$ and ${}^{C_k}\mathbf{p}_{C_\eta}$ are defined in (7)-(8).

IV. Common-Feature Constraints

In our problem formulation, if a feature is observed by multiple users, we first define it as a different feature in each user's map but then ensure geometric consistency by imposing

constraints on the common features to be the same physical point or line. In what follows, we present the geometric constraints for common point, free-line, and Manhattan-line features.

A. Point-feature Constraint

Consider a point feature $\mathbf{x}_{P_i}$ observed by two users whose global frames of reference are $\{G_1\}$ and $\{G_2\}$, respectively, and expressed as ${}^{C_{\eta_1}}\mathbf{x}_{P_i}$ and ${}^{C_{\eta_2}}\mathbf{x}_{P_i}$ with respect to the first observing camera poses in the two users' maps. The geometric constraint between them is:

$$ {}^{C_{\eta_1}}\mathbf{x}_{P_i} - {}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C}\,{}^{C_{\eta_2}}\mathbf{x}_{P_i} - {}^{C_{\eta_1}}\mathbf{p}_{C_{\eta_2}} = \mathbf{0} \tag{14} $$

where ${}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C}$ and ${}^{C_{\eta_1}}\mathbf{p}_{C_{\eta_2}}$ are then expressed as:

$$ {}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C} = {}^{C_{\eta_1}}_{G_1}\mathbf{C}\,{}^{G_1}_{G_2}\mathbf{C}\,{}^{C_{\eta_2}}_{G_2}\mathbf{C}^T \tag{15} $$

$$ {}^{C_{\eta_1}}\mathbf{p}_{C_{\eta_2}} = {}^{C_{\eta_1}}_{G_1}\mathbf{C}\left({}^{G_1}\mathbf{p}_{G_2} - {}^{G_1}\mathbf{p}_{C_{\eta_1}} + {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{p}_{C_{\eta_2}}\right) \tag{16} $$

Note that (15)-(16) involve the two camera poses $({}^{G_1}\mathbf{p}_{C_{\eta_1}}, {}^{C_{\eta_1}}_{G_1}\mathbf{C})$ and $({}^{G_2}\mathbf{p}_{C_{\eta_2}}, {}^{C_{\eta_2}}_{G_2}\mathbf{C})$, as well as the 4 dof transformation $({}^{G_1}\mathbf{p}_{G_2}, {}^{G_1}_{G_2}\mathbf{C})$ between the two maps.

B. Free-line Feature Constraint

Consider a free-line feature $\boldsymbol{\ell}$ observed by two users, and expressed with respect to the first observing camera poses $\{C_{\eta_1}\}$ and $\{C_{\eta_2}\}$ in their maps, respectively. As evident from Fig. 3, the common free line is represented in the two maps using frames of different *origins*. For deriving the geometric constraint between two free lines, we employ the following relation between frames $\{C_{\eta_1}\}$, $\{L_{i_1}\}$, $\{C_{\eta_2}\}$, and $\{L_{i_2}\}$:

$$ {}^{L_{i_2}}\mathbf{p}_{L_{i_1}} = {}^{C_{\eta_2}}_{L_{i_2}}\mathbf{C}^T\left[{}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C}^T\left({}^{C_{\eta_1}}\mathbf{p}_{L_{i_1}} - {}^{C_{\eta_1}}\mathbf{p}_{C_{\eta_2}}\right) - {}^{C_{\eta_2}}\mathbf{p}_{L_{i_2}}\right] \tag{17} $$

where ${}^{L_{i_2}}\mathbf{p}_{L_{i_1}} = -d_c\mathbf{e}_1$. To remove d_c from (17), we define $\mathbf{E}_{23} \triangleq \begin{bmatrix}\mathbf{e}_2 & \mathbf{e}_3\end{bmatrix}^T$ and multiply with it both sides of (17) to obtain the 2 dof constraint:

$$ \mathbf{E}_{23}\,{}^{C_{\eta_2}}_{L_{i_2}}\mathbf{C}^T\left({}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C}^T\left({}^{C_{\eta_1}}\mathbf{p}_{L_{i_1}} - {}^{C_{\eta_1}}\mathbf{p}_{C_{\eta_2}}\right) - {}^{C_{\eta_2}}\mathbf{p}_{L_{i_2}}\right) = \mathbf{0} \tag{18} $$

Then, since the x-axes of frames $\{L_{i_1}\}$ and $\{L_{i_2}\}$ are both defined according to the *same* line direction, we have the additional 2 dof constraint:

$$ \mathbf{E}_{23}\left({}^{C_{\eta_1}}_{L_{i_1}}\mathbf{C}\mathbf{e}_1 - {}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C}\,{}^{C_{\eta_2}}_{L_{i_2}}\mathbf{C}\mathbf{e}_1\right) = \mathbf{0} \tag{19} $$

Lastly, by employing again (15) and (16) we can express the constraints in (18) and (19) as a function of the two camera poses and the transformation between the two maps.

C. Manhattan-line Feature Constraint

The common Manhattan-line features also satisfy (17), with the additional information that a Manhattan line is aligned with one of the building's cardinal directions. For example, if the line's direction is $\mathbf{e}_1$, we have:

$$ {}^{C_{\eta_2}}_{V_{i_2}}\mathbf{C}\,{}^{V_{i_2}}\mathbf{p}_{V_{i_1}} = -d_c\,{}^{C_{\eta_2}}_{B}\mathbf{C}\mathbf{e}_1 \tag{20} $$

Similar to (18), the 2-dof translational common-Manhattan-line constraint can be written as:

$$ \mathbf{E}_{23}\,{}^{C_{\eta_2}}_{B}\mathbf{C}^T\left({}^{C_{\eta_1}}_{C_{\eta_2}}\mathbf{C}^T\left({}^{C_{\eta_1}}\mathbf{p}_{V_{i_1}} - {}^{C_{\eta_1}}\mathbf{p}_{C_{\eta_2}}\right) - {}^{C_{\eta_2}}\mathbf{p}_{V_{i_2}}\right) = \mathbf{0} \tag{21} $$

Note also that since the Manhattan lines align with the building's cardinal directions, the orientation constraint corresponding to (19) is automatically satisfied and need not be considered.

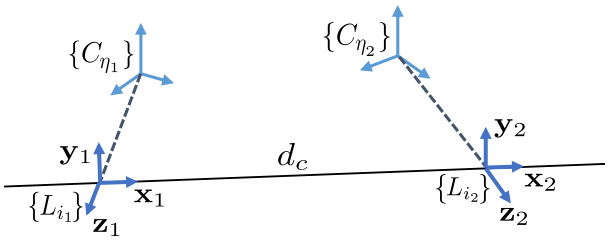


Fig. 3: Depiction of line constraints.

V. Algorithm Description

In what follows, we first briefly review the BLS method for determining the trajectory and map of each user based on only its own measurements, and then (Sect. V-B) describe our approach to find an initial estimate for the relative poses between users. Subsequently, we introduce our CM algorithm in Sect. V-C and present a method for "sparsifying" (i.e., reducing the number and thus spatial density of) the commonly-observed-feature constraints in Sect. V-D.

A. Single-user Batch Least-squares

For a user j, computing the BLS estimate requires minimizing the following non-linear cost function:

$$ \mathscr{C}_j = \left|\left|{}^j\breve{\mathbf{p}} - \mathbf{g}({}^j\breve{\mathbf{p}}, {}^j\mathbf{u})\right|\right|^2_{{}^j\mathbf{Q}} + \left|\left|{}^j\mathbf{z} - \mathbf{h}({}^j\mathbf{x}_u)\right|\right|^2_{{}^j\mathbf{R}} \tag{22} $$

where the first term corresponds to the cost function arising from IMU measurements (5), while the second term is due to visual observations of point (6), free-line (9)-(10), and Manhattan-line (12)-(13) features. Also, in (22) ${}^j\breve{\mathbf{p}}$ and ${}^j\mathbf{x}_u$ denote the users' poses and full state [see (1)], respectively, ${}^j\mathbf{u}$ includes all IMU measurements, ${}^j\mathbf{z}$ comprises all visual observations, ${}^j\mathbf{Q}$ and ${}^j\mathbf{R}$ are the covariance matrices of the corresponding measurement noises.

The cost function (22) can be minimized by employing Gauss-Newton iterative minimization. In particular, by denoting the error states of ${}^j\mathbf{x}_u$ as $\delta^j\mathbf{x}_u$, in each Gauss-Newton iteration, we have the linearized cost function:

$$ \mathscr{C}'_j = \left|\left|\mathbf{J}_j\delta^j\mathbf{x}_u - \mathbf{b}_j\right|\right|^2 \tag{23} $$

where $\mathbf{J}_j$ and $\mathbf{b}_j$ are the Jacobian and residual, respectively. Since matrix $\mathbf{J}_j$ is typically of full column rank in a visual-inertial mapping problem, a unique $\delta^j\mathbf{x}_u$ can be solved efficiently by employing the Cholmod algorithm [31]. Specifically, defining $\mathbf{G}_j$ as the Cholesky factor of the Hessian, i.e., $\mathbf{G}_j\mathbf{G}_j^T = \mathbf{J}_j^T\mathbf{J}_j$, we minimize (23) with respect to $\delta^j\mathbf{x}_u$ as follows:

$$ \begin{aligned} &\mathbf{J}_j^T\mathbf{J}_j\delta^j\mathbf{x}_u = \mathbf{J}_j^T\mathbf{b}_j \Leftrightarrow \mathbf{G}_j\mathbf{G}_j^T\delta^j\mathbf{x}_u = \mathbf{J}_j^T\mathbf{b}_j \\ \Leftrightarrow\quad &\mathbf{G}_j\delta^j\mathbf{y}_u = \mathbf{J}_j^T\mathbf{b}_j, \quad \text{with } \delta^j\mathbf{y}_u \triangleq \mathbf{G}_j^T\delta^j\mathbf{x}_u \end{aligned} \tag{24} $$

which involves consecutively solving two triangular systems. Once $\delta^j\mathbf{x}_u$ is computed, it is used to update the estimates for ${}^j\mathbf{x}_u$, and initiate a new Gauss-Newton iteration until convergence $\left(||\delta^j\mathbf{x}_u|| < \text{size}(\delta^j\mathbf{x}_u)\times 10^{-5}\right)$.

Note that the Gauss-Newton minimization described above can be performed by each user independently (in parallel or at different times) to compute an estimate of each user's state ${}^j\mathbf{x}_u$. These estimates and the Choleskey factor, $\mathbf{G}_j$, will be provided to the CM algorithm (see Sect. V-C) for merging all

maps. Before computing the merged map, however, an initial estimate of the transformation between the users' reference frames is needed. This initialization process using visual observations of common, amongst the different users' maps, point features is described in the next section.

B. Initial Estimate of the Users' Relative Poses

In what follows, we first describe our algorithm for computing the transformation between two users, which can be used in a minimal solver in conjunction with RANSAC [32] for outlier rejection and/or for finding an approximate least-squares solution. Then, we explain our approach for computing the transformation between all users.

1) Transformation between pairs of users: As shown in [16], when using visual and inertial measurements, the roll and pitch angles of each user's orientation are observable in the inertial frame of reference. Therefore, the transformation between any two users' global reference frames has four dof: One corresponding to their relative yaw angle and three corresponding to their relative position. By defining the two users' frames of reference as $\{G_1\}$ and $\{G_2\}$, we seek to estimate the position and orientation of $\{G_2\}$ with respect to $\{G_1\}$, denoted as ${}^{G_1}\mathbf{p}_{G_2}$ and ${}^{G_1}_{G_2}\mathbf{C}$, respectively. Note that ${}^{G_1}_{G_2}\mathbf{C}$ corresponds to a rotation about the global z-axis, which is aligned with gravity, and thus:

$$ {}^{G_1}_{G_2}\mathbf{C} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{25} $$

When two users observe the same point feature $\mathbf{x}_{P_i}, i = 1,\ldots,M$, the geometric constraint between them is:

$$ {}^{G_1}\mathbf{x}_{P_i} = {}^{G_1}\mathbf{p}_{G_2} + {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{x}_{P_i} \tag{26} $$

where ${}^{G_1}\mathbf{x}_{P_i}$, ${}^{G_2}\mathbf{x}_{P_i}$ are the point feature $\mathbf{x}_{P_i}$'s 3D position vectors expressed in $\{G_1\}$ and $\{G_2\}$, respectively.

Subtracting the constraint (26) corresponding to feature $\mathbf{x}_{P_1}$ from the ones for $\mathbf{x}_{P_i}, i = 2,\ldots,M$, results in:

$$ {}^{G_1}\mathbf{x}_{P_i} - {}^{G_1}\mathbf{x}_{P_1} = {}^{G_1}_{G_2}\mathbf{C}\left({}^{G_2}\mathbf{x}_{P_i} - {}^{G_2}\mathbf{x}_{P_1}\right) \tag{27} $$

which can be rewritten as:

$$ \mathbf{A}_i \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \triangleq \mathbf{A}_i\mathbf{w} = \mathbf{b}_i, \qquad i = 2,\ldots,M \tag{28} $$

where $\mathbf{A}_i$ and $\mathbf{b}_i$ are a 3×2 matrix and a 3×1 vector respectively, and both of them are functions of ${}^{G_1}\mathbf{x}_{P_i}$, ${}^{G_1}\mathbf{x}_{P_1}$, ${}^{G_2}\mathbf{x}_{P_i}$, and ${}^{G_2}\mathbf{x}_{P_1}$. By defining $\mathbf{A} = \left[\mathbf{A}_2^T,\ldots,\mathbf{A}_M^T\right]^T$ and $\mathbf{b} = \left[\mathbf{b}_2^T,\ldots,\mathbf{b}_M^T\right]^T$, $\mathbf{w}$ can be obtained by solving the following minimization problem:

$$ \begin{aligned} \mathbf{w}^* = \text{argmin } & \|\mathbf{A}\mathbf{w} - \mathbf{b}\|^2 \\ \text{s.t. } & \|\mathbf{w}\|^2 = 1 \end{aligned} \tag{29} $$

Note that (29) is a least-squares problem with a quadratic constraint, which can be solved by following the general methodology of [33]. Instead, in this work we provide a more efficient solution in Appendix A, which takes advantage of the fact that $\mathbf{w}$ is a 2×1 vector.

After solving for the yaw angle θ, we substitute ${}^{G_1}_{G_2}\mathbf{C}$ in (26) and obtain ${}^{G_1}\mathbf{p}_{G_2}$ as:

$$ {}^{G_1}\mathbf{p}_{G_2} = \frac{1}{M}\sum_{i=1}^{M}\left({}^{G_1}\mathbf{x}_{P_i} - {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{x}_{P_i}\right). \tag{30} $$

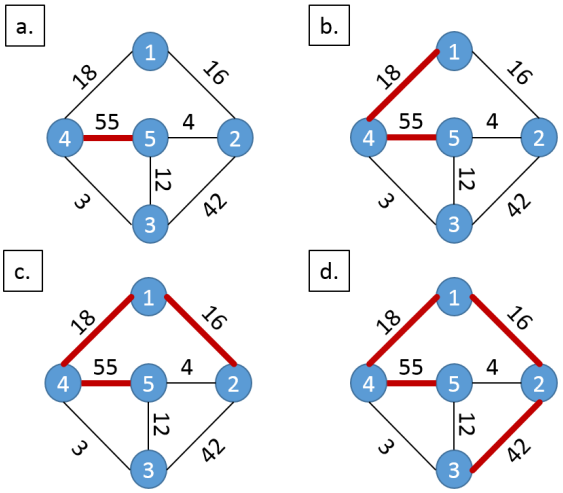


Fig. 4: Finding a maximum spanning tree (marked in red color) on the user graph. The numbers of commonly-observed features between pairs of users are written on the edges.

Both ${}^{G_1}\mathbf{p}_{G_2}$ in (30) and ${}^{G_1}_{G_2}\mathbf{C}$ corresponding to the $\mathbf{w}$ calculated in (29) will be used in Sect. V-C for initializing the CM algorithm. Note also that this transformation will be estimated by CM so as to improve the mapping accuracy.

The aforementioned process assumes that all matched point features are inliers and their 3D position estimates ${}^{G_1}\hat{\mathbf{x}}_{P_i}$, ${}^{G_2}\hat{\mathbf{x}}_{P_i}$, $i = 1,\ldots,M$, are accurate. In practice, we employ RANSAC to remove outliers. Specifically, the solution of (29) and (30) for $M = 2$ are used as the minimal solver for RANSAC.

2) Initial transformation between multiple users: A naive approach to obtain the relative pose between any two users is to compute the relative transformation between all possible pairs. Instead, we seek to compute the initial transformation between any two users indirectly by employing a chain of pairwise transformations.[5] To do so, we select the "best" (in the sense that they are computed using the maximum number of common features) among them that form a chain connecting *all* users. In order to solve this problem, we first construct a graph whose vertices correspond to each user and edges are assigned weights proportional to the number of commonly-observed features between the corresponding users. Then, we compute the maximum spanning tree (MST) of the graph following [34]. An example of finding the chain connection between users is illustrated in Fig. 4, where the transformations between five users are obtained by combining the transformation pairs (5, 4), (4, 1), (1, 2), (2, 3). Finally, in the CM problem, we will estimate the transformation between the user pairs that correspond to the edges in the resulting MST.

C. Cooperative Mapping

In this section, we first (Sect. V-C1) present the standard BLS formulation of the CM problem and briefly discuss some of its limitations. Subsequently (Sect. V-C2), we reformulate

[5]Note that only $n-1$ transformations should be included in the estimation problem, and they are sufficient for expressing all possible $n(n-1)/2$ transformations between users.

CM as a constrained optimization problem, and show the equivalence of its solution to that of the BLS formulation. The solution of the CM formulation is described in Sect. V-C3, while its advantages as compared to the standard BLS formulation of CM are discussed in Sect. V-C5.

1) BLS Formulation: As previously mentioned, linking the different users' maps requires using observations of common features. The standard BLS formulation when applied to the CM problem achieves this by modifying the camera measurement models for all three types of features to explicitly consider the transformation between the reference frames of the users observing the same features. Specifically, for a feature first observed by user 1 at camera pose $\{C_{\eta_1}\}$, its observation by user 2 at camera pose $\{C_{k_2}\}$ can still be expressed using the original point (6), free-line (9)-(10), or Manhattan-line (12)-(13) measurement model, but ${}^{C_{k_2}}_{C_{\eta_1}}\mathbf{C}$ and ${}^{C_{k_2}}\mathbf{p}_{C_{\eta_1}}$ need to be redefined using the transformation, $({}^{G_1}\mathbf{p}_{G_2}, {}^{G_1}_{G_2}\mathbf{C})$, between the two users as:

$$\begin{aligned} {}^{C_{k_2}}_{C_{\eta_1}}\mathbf{C} &= {}^{C_{k_2}}_{G_2}\mathbf{C}\,{}^{G_1}_{G_2}\mathbf{C}^T\,{}^{C_{\eta_1}}_{G_1}\mathbf{C}^T \\ {}^{C_{k_2}}\mathbf{p}_{C_{\eta_1}} &= {}^{C_{k_2}}_{G_2}\mathbf{C}\left({}^{G_1}_{G_2}\mathbf{C}^T({}^{G_1}\mathbf{p}_{C_{\eta_1}} - {}^{G_1}\mathbf{p}_{G_2}) - {}^{G_2}\mathbf{p}_{C_{k_2}}\right) \end{aligned} \tag{31}$$

This new camera model, for any *common* observation, can be written in a compact form as:

$$\mathbf{z}_c = \boldsymbol{\psi}(\mathbf{x}_a, \mathbf{f}_c, \mathbf{x}_\tau) + \mathbf{n}_c \tag{32}$$

where $\mathbf{x}_\tau$ is a vector of size $4(N-1)$ comprising the pairwise transformations between users initialized as described in Sect. V-B, $\mathbf{x}_a \triangleq \begin{bmatrix} {}^1\breve{\mathbf{p}}^T & \cdots & {}^N\breve{\mathbf{p}}^T \end{bmatrix}^T$ is the vector comprising all users' poses along with their velocities and IMU biases, $\mathbf{f}_c$ is the set of common features observed by two or more users, and $\mathbf{n}_c$ is the corresponding measurement noise of covariance $\mathbf{R}_c$.

Following the standard BLS formulation, we can construct the cost function for the CM problem by summing the following terms: (i) $\overline{\mathscr{C}}_j(\mathbf{x}_a, \mathbf{f}_a)$, the cost function of each individual user [see (22)] *after* removing all cost terms involving common features, where $\mathbf{f}_a$ denotes the features observed by only one user. (ii) The cost terms arising from each user's observations to common features [see (32)]. Thus, our objective function becomes:

$$\mathbf{x}_\tau^*, \mathbf{x}_a^*, \mathbf{f}_a^*, \mathbf{f}_c^* = \operatorname{argmin}\left(\sum_{j=1}^{N} \overline{\mathscr{C}}_j + ||\mathbf{z}_c - \boldsymbol{\psi}(\mathbf{x}_a, \mathbf{f}_c, \mathbf{x}_\tau)||^2_{\mathbf{R}_c}\right) \tag{33}$$

A standard BLS solution can be obtained following the same procedure as in (24), by applying Cholesky factorization on the Hessian matrix arising from all users' measurements. This strategy, however, treats the problem as one user collecting a large dataset, and ignores the fact that often there exist only few common observations between the different users. In the next section, we will present our alternative approach that first constructs a Hessian matrix for each user independently, and then employs the geometric constraints stemming from the common-feature observations to create correlations between the users' Hessians. As it will become evident later on, our formulation results in a solution that is modular, parallelizable, and its processing cost can be adjusted to improve efficiency.

2) CM Formulation: To introduce the proposed CM formulation, we employ the following theorem:

Theorem 1: The optimization problem (33) is equivalent to the following constrained optimization problem:

$$\begin{aligned} \mathbf{x}_\tau^*, \mathbf{x}_a^*, \mathbf{f}_a^*, \mathbf{f}_{c_1}^*, \cdots, \mathbf{f}_{c_N}^* = \operatorname{argmin} & \sum_{j=1}^{N} \mathscr{C}_j \\ \text{s. t.} \quad & \boldsymbol{\kappa}(\mathbf{x}_a, \mathbf{x}_\tau, \mathbf{f}_{c_{j\alpha}}, \mathbf{f}_{c_{j_\beta}}) = \mathbf{0}, \\ & j_\alpha, j_\beta = 1, \ldots, N, j_\alpha \neq j_\beta \end{aligned} \tag{34}$$

where $\mathscr{C}_j$ denotes the cost function for user j [see (22)], $\mathbf{f}_{c_{j\alpha}}$ and $\mathbf{f}_{c_{j_\beta}}$ are defined as the subset of common features $\mathbf{f}_c$ observed by user j_α and j_β, and $\boldsymbol{\kappa}(\mathbf{x}_a, \mathbf{x}_\tau, \mathbf{f}_{c_{j\alpha}}, \mathbf{f}_{c_{j_\beta}})$ denotes common-feature constraints as defined in (14), (18), (19), and (21).

Proof: See Appendix B.

Note the Hessian matrix corresponding to the cost function in the CM formulation (34) has a block-diagonal structure, where each block comprises the state of one user. As it will be explained in Sect. V-C5, solving CM as a constrained optimization problem takes advantage of this structure.

3) CM Solution: Since problem (34) is nonlinear, we solve it employing Gauss-Newton iterative minimization [35]. At each iteration, we focus on the following (linearized) constrained optimization problem:

$$\begin{aligned} \delta\mathbf{x}_\tau^*, \delta^1\mathbf{x}_u^*, \ldots, \delta^N\mathbf{x}_u^* = \operatorname{argmin} & \sum_{j=1}^{N} \left|\left|\mathbf{J}_j\delta^j\mathbf{x}_u - \mathbf{b}_j\right|\right|^2 \\ \text{s. t.} \quad & \mathbf{A}_\tau\delta\mathbf{x}_\tau + \sum_{j=1}^{N}\mathbf{A}_j\delta^j\mathbf{x}_u = \mathbf{r} \end{aligned} \tag{35}$$

where $\delta^j\mathbf{x}_u$ and $\delta\mathbf{x}_\tau$ are the error states of the user state, ${}^j\mathbf{x}_u$, and transformations between users, $\mathbf{x}_\tau$, respectively, while $\mathbf{J}_j$ and $\mathbf{b}_j$ are the corresponding Jacobian and residual of $\mathscr{C}_j$. $\mathbf{A}_j$, $\mathbf{A}_\tau$, and $\mathbf{r}$ are the Jacobians (corresponding to ${}^j\mathbf{x}_u$ and $\mathbf{x}_\tau$) and residual of the constraints.

The KKT optimality conditions [36] for (35) are:

$$\begin{aligned} & \mathbf{J}_j^T(\mathbf{J}_j\delta^j\mathbf{x}_u - \mathbf{b}_j) + \mathbf{A}_j^T\boldsymbol{\lambda} = \mathbf{0}, \quad j = 1, \ldots, N \\ & \mathbf{A}_\tau\delta\mathbf{x}_\tau + \sum_{j=1}^{N}\mathbf{A}_j\delta^j\mathbf{x}_u - \mathbf{r} = \mathbf{0} \\ & \mathbf{A}_\tau^T\boldsymbol{\lambda} = \mathbf{0} \end{aligned} \tag{36}$$

where $\boldsymbol{\lambda}$ is the Lagrange-multiplier vector.

To simplify notation, in what follows we present our algorithm for solving (36) for the case of two users, while its extension to three or more users is straightforward.

Writing (36) in a compact form yields:

$$\underbrace{\begin{bmatrix} \mathbf{J}_1^T\mathbf{J}_1 & & \mathbf{A}_1^T & \\ & \mathbf{J}_2^T\mathbf{J}_2 & \mathbf{A}_2^T & \\ \mathbf{A}_1 & \mathbf{A}_2 & & \mathbf{A}_\tau \\ & & \mathbf{A}_\tau^T & \end{bmatrix}}_{\mathbf{H}_{CM}} \begin{bmatrix} \delta^1\mathbf{x}_u \\ \delta^2\mathbf{x}_u \\ \boldsymbol{\lambda} \\ \delta\mathbf{x}_\tau \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1^T\mathbf{b}_1 \\ \mathbf{J}_2^T\mathbf{b}_2 \\ \mathbf{r} \\ \mathbf{0} \end{bmatrix} \tag{37}$$

Note that due to the zeros in the (3, 3) and (4, 4) block-diagonal elements, $\mathbf{H}_{CM}$ is *not* positive definite. Thus, Cholesky factorization cannot be applied. Although other methods, such as diagonal pivoting [37], can be employed to solve (37), we propose an alternative approach that takes advantage of $\mathbf{H}_{CM}$'s structure and the Cholesky factors previously computed by each user based on the following theorem:

Theorem 2: $\mathbf{H}_{CM}$ can be factorized into the product of a lower-triangular and an upper-triangular matrix as:

$$\mathbf{H}_{CM} = \begin{bmatrix} \mathbf{G}_1 & & & \\ & \mathbf{G}_2 & & \\ \mathbf{K}_1^T & \mathbf{K}_2^T & \mathbf{T}_{11} & \\ & & -\mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{G}_1^T & & \mathbf{K}_1 & \\ & \mathbf{G}_2^T & \mathbf{K}_2 & \\ & & -\mathbf{T}_{11}^T & \mathbf{T}_{21}^T \\ & & & \mathbf{T}_{22}^T \end{bmatrix} \tag{38}$$

where $\mathbf{G}_1$ and $\mathbf{G}_2$ are the Cholesky factors of the users' Hessian matrices $\mathbf{J}_1^T\mathbf{J}_1$ and $\mathbf{J}_2^T\mathbf{J}_2$, respectively, and $\mathbf{T}_{11}$ and $\mathbf{T}_{22}$ are lower-triangular matrices.

Proof: Multiplying the two triangular matrices in (38), and employing the structure of $\mathbf{H}_{CM}$ in (37) yields the following system of equations:

$$\mathbf{G}_j\mathbf{K}_j = \mathbf{A}_j^T, \quad j = 1,2 \tag{39}$$

$$\mathbf{T}_{11}\mathbf{T}_{11}^T = \sum_{j=1}^{2} \mathbf{K}_j^T\mathbf{K}_j \tag{40}$$

$$\mathbf{T}_{11}\mathbf{T}_{21}^T = \mathbf{A}_\tau \tag{41}$$

$$\mathbf{T}_{22}\mathbf{T}_{22}^T = \mathbf{T}_{21}\mathbf{T}_{21}^T \tag{42}$$

To find the matrices $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, and $\mathbf{T}_{22}$ that satisfy (39)-(42), we first compute $\mathbf{K}_j, j = 1,2$, by solving a linear equation corresponding to each of the columns of $\mathbf{K}_j$ [see (39)].

Moreover, by defining $\mathbf{K} = \left[\mathbf{K}_1^T\mathbf{K}_2^T\right]^T$, it is easy to see that

$$\sum_{j=1}^{2} \mathbf{K}_j^T\mathbf{K}_j = \mathbf{K}^T\mathbf{K} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix} \begin{bmatrix} (\mathbf{G}_1\mathbf{G}_1^T)^{-1} & \mathbf{0} \\ \mathbf{0} & (\mathbf{G}_2\mathbf{G}_2^T)^{-1} \end{bmatrix} \begin{bmatrix}\mathbf{A}_1^T \\ \mathbf{A}_2^T\end{bmatrix}$$

is a positive definite matrix because $\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}$ has full row rank [i.e., each common-feature constraint appears in (34) only once]. Thus, we compute $\mathbf{T}_{11}$ as the Cholesky factor of $\mathbf{K}^T\mathbf{K}$ that satisfies (40). Given $\mathbf{T}_{11}$, we determine $\mathbf{T}_{21}$ in (41) using triangular back-substitution.

Lastly, $\mathbf{T}_{21}\mathbf{T}_{21}^T$ is also positive definite, and thus $\mathbf{T}_{22}$ is selected as the Cholesky factor of $\mathbf{T}_{21}\mathbf{T}_{21}^T$ [see (42)]. ■

Once all the block matrices in (38) are obtained, (37) is efficiently solved by employing two back-substitutions involving triangular matrices.

4) CM Computational Complexity: Now, we briefly discuss the processing cost for computing each block in (38). The Cholesky factors $\mathbf{G}_j$ do *not* require any calculation in the first Gauss-Newton iteration, because they have already been computed by each user. Starting from the second Gauss-Newton iteration, the $\mathbf{G}_j$ matrices need to be re-computed, which can be done in parallel, at a cost that depends on the structure of the Hessian.

Computing the $\mathbf{K}_j$ matrices involves triangular back-substitution according to (39), which has low computational cost for two reasons: (i) The $\mathbf{A}_j$ matrices are very sparse (less than 0.01% nonzero elements); (ii) Each column of the $\mathbf{K}_j$ matrices can be computed in parallel. Note also that since the number of columns of $\mathbf{K}$ is equal to the number of commonly-observed feature constraints, the time for computing $\mathbf{K}$ grows *linearly* with the number of constraints.

Defining each row in $\mathbf{K}$ as $\mathbf{k}_m^T$, $\mathbf{K}^T\mathbf{K}$ is computed as $\sum \mathbf{k}_m\mathbf{k}_m^T$, where all the terms in the summation are calculated in parallel. Since the size of $\mathbf{k}_m$ equals the dimension of the commonly-observed feature constraints, the overall computational complexity increases *quadratically* with the number of constraints.

The $\mathbf{T}_{11}$ matrix is the Cholesky factor of $\mathbf{K}^T\mathbf{K}$. Although $\mathbf{K}$ is sparse (about 1% nonzero elements), since it is, in general, a tall matrix, $\mathbf{K}^T\mathbf{K}$ is typically a *dense* square matrix with size equal to the number of commonly-observed feature constraints. Thus, computing $\mathbf{T}_{11}$ has, in general, *cubic* processing cost with respect to the number of constraints.

Lastly, both $\mathbf{T}_{21}$ and $\mathbf{T}_{22}$ are very small matrices (their size depends on the number of users/maps), and thus take very little time to compute. Once all the block matrices are computed, solving the linear system in (37) requires only two sparse, triangular back-substitutions.

As we will show in the experimental results, when the number of common features is relatively small, the most computational demanding part of our CM algorithm is computing $\mathbf{G}_j$. On the other hand, in the case of many common-feature observations, the processing cost for computing $\mathbf{K}$, $\mathbf{K}^T\mathbf{K}$, and $\mathbf{T}_{11}$ (linear, quadratic, and cubic, respectively) becomes dominant. Note though that, besides computing $\mathbf{T}_{11}$, all these operations are parallelizable. Additionally, as explained in Sect. V-D, the CM algorithm provides a straightforward mechanism for trading processing cost for accuracy by reducing the number of commonly-observed feature constraints imposed.

5) CM Solution Advantages: Formulating and solving CM as a constrained optimization problem has the following advantages:

Parallelization: In (33), since some of the features are observed by multiple users, the corresponding off-diagonal blocks in the resulting Hessian matrix are nonzero. Thus, there is no straightforward way to parallelize computations. In contrast, and as described above, most operations required for solving (34) are parallelizable (e.g., computing the $\mathbf{K}_j$ and $\mathbf{G}_j$ matrices). This is of particular importance when mapping very large areas (e.g., airports, museums, shopping malls) using multiple devices.

Memory Efficiency: The BLS formulation (33) requires applying Cholesky factorization on the Hessian created from all the users' data. In contrast, the proposed formulation [see (34)] applies Cholesky factorization on the (smaller-size) Hessians corresponding to each user's data. Since the memory requirements of Cholesky factorization depend heavily on the problem size, solving (33) is significantly more memory demanding.

Modularity: In (33), the feature measurement model changes if a common feature is already defined in another map, in which case the transformation between the maps needs to be involved. In contrast, in (34) common features always use the same measurement model as the rest of the features (i.e., it does not involve the maps' transformations), thus ensuring uniformity. Moreover, adding (dropping) a user's trajectory and map does not affect the Jacobian matrices of the other users. Instead, we simply add (remove) the corresponding Jacobian and constraints. This is especially convenient when expanding the map or updating pre-existing maps.

6) CM Covariance: For the BLS formulation of (33), we can find the uncertainty of the estimated state, i.e., covariance matrix, by inverting the system's Hessian matrix. On the other hand, since we formulate CM as a constrained optimization problem, the estimate's covariance cannot be computed following the same procedure. Instead, in Appendix C we present

the derivation for efficiently computing the CM's covariance by taking advantage of the intermediate resulting matrices $\mathbf{G}_j$, $\mathbf{K}_j$, $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, and $\mathbf{T}_{22}$.

D. Commonly-observed Feature Sparsification: Resource-aware CM

As described in Sect. V-C3, the computational cost of some of the matrices employed by the proposed CM algorithm increases quadratically to cubically with the number of commonly-observed features. Therefore, the CM algorithm is best suited for cases when the number of inter-dataset loop-closures (i.e., feature correspondences) is relatively small. To address scenarios where the users may observe a large number of common features, in what follows, we seek to retain the commonly-observed feature constraints corresponding to only a subset of them, $\bar{\mathbf{f}}_{c_1},\ldots,\bar{\mathbf{f}}_{c_N}$, where $\bar{\mathbf{f}}_{c_j}\subseteq\mathbf{f}_{c_j}$, $j=1,\ldots,N$, so as to lower the CM's processing cost. By doing so, (34) can be written as:

$$\begin{aligned}\mathbf{x}_\tau^*,\mathbf{x}_a^*,\mathbf{f}_a^*,\mathbf{f}_{c_1}^*,\cdots,\mathbf{f}_{c_N}^* = &\operatorname{argmin}\sum_{j=1}^{N}\mathscr{C}_j\\ &\text{s. t.}\quad \boldsymbol{\kappa}(\mathbf{x}_a,\mathbf{x}_\tau,\bar{\mathbf{f}}_{c_{j_\alpha}},\bar{\mathbf{f}}_{c_{j_\beta}})=\mathbf{0},\\ &\qquad j_\alpha,j_\beta=1,\ldots,N, j_\alpha\neq j_\beta \end{aligned}\tag{43}$$

Note that during this constraint "sparsification" process, we do not drop or change any feature measurement. Instead, we relax the optimization problem by assuming that a feature common to two or more maps corresponds to different physical points or lines in each user's map. Additionally, since no spurious information is gained through this approximation, this feature-sparsification method is consistent.

At this point, we should note that the optimal solution to the problem of selecting the most informative (in terms of expected CM accuracy) commonly-observed features has computational cost significantly higher than solving CM using all available common features. For this reason, we introduce a heuristic method to efficiently select commonly-observed features that are (i) evenly distributed across the physical space, and (ii) accurately estimated in each map, so as to improve the rigidity and accuracy of the resulting merged map.

In indoor environments, features appear on the ceiling, walls, and floor of each level while there are rarely any between the ceiling and the floor of consecutive levels. By taking advantage of these gaps in the vertical distribution of the features, it is fairly easy to partition the 3D space into volumes, each corresponding to a building floor. To do so, we project all features on the global (aligned with gravity) z axis and employ K-means [38] to cluster them into the corresponding levels. Subsequently, we project each floor's features on the $x-y$ plane represented as a uniform grid. Based on this partitioning, we generate a sparse subset of commonly-observed features by selecting at most 2 point features per cell, prioritized by the number of times a feature is observed. Note that two is the minimum number of features required for computing the transformation between pairs of maps (see Sect. V-B). Moreover, our experiments have shown that the positioning accuracy of a feature typically increases with the number of observations.

The $x-y$ distribution of common features before and after sparsification using grids of various sizes is illustrated in Fig. 9. This common-feature sparsification method is shown to be effective in our experimental results. For example, in a building-size dataset, after reducing the number of common features from more than four thousand to about two hundred, the root-mean-square difference between the estimated user trajectories is only 13.5 cm.

VI. Experiment Results

In this section, we first describe our experimental setup, and then introduce the methods we employ for tracking point and line features within a single and across multiple user maps. Subsequently, we present a detailed experimental evaluation and analysis of our CM algorithm on two large-scale datasets collected in Keller Hall and Walter Library at the University of Minnesota. Specifically, the Keller Hall dataset illustrates a scenario where the users' trajectories along the building's corridors overlap a lot and hence they observe a large number of common features. In this dataset, we demonstrate the accuracy improvement when employing line features, in addition to point features, as well as the significant processing savings, with only minimal accuracy loss, resulting from the proposed feature-sparsification technique (see Sect. V-D). On the other hand, the Walter Library dataset represents the typical case where the users cover, for the most part, different areas of the building visiting only few common locations. In this dataset, we show that the CM algorithm is significantly more efficient, in terms of processing cost and memory usage, as compared to the standard BLS solution. Finally, we provide a quantitative evaluation of the CM algorithm on three additional datasets collected in the RSS 2015 and CVPR 2016 conference sites, and the MIT Stata Center. **An interactive visualization of the experimental results on these five buildings, as well as on multiple other locations, is available online at [24].**

A. Dataset Preparation and Algorithm Implementation

The visual and inertial measurements used in our CM tests were collected using Project Tango developer phones and tablets [39].[6] Greyscale images, of resolution 640×480, were saved at 15 Hz, along with consumer-grade, MEMS-based, IMU data at 100 Hz. All the reported timing results are obtained by running the algorithms on a desktop computer with an Intel® Xeon® E5-1650 v2 Processor (3.5 GHz). The parallelization is implemented using Intel's threading building blocks (TBB) library [40]. The matrix operations are performed utilizing the Eigen library [41], while the Cholesky factors (i.e., the matrices $\mathbf{G}_j$) are computed by employing the Cholmod [31] algorithm from the SuiteSparse library [42].[7] Based on our tests, Cholmod is faster than other Cholesky factorization algorithms, such as the Simplicial Cholesky in Eigen and CSparse [44] in SuiteSparse. Cholmod is a sequential algorithm, but it is able to utilize a basic linear algebra subprograms (BLAS) library to perform low-level matrix operations (e.g., matrix multiplication) in parallel. For this reason, we compared the speed of three of the most

[6]We only use the Tango phones and tablets for collecting visual and inertial data. All the algorithms described in Sect. V were implemented by the authors of this paper.

[7]Note that the SuiteSparse library is also used in Google's standard non-linear least-squares solver, Ceres [43].

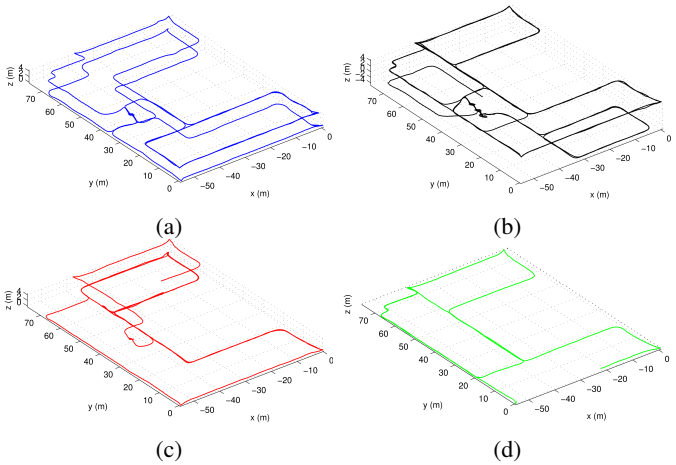


Fig. 5: Keller Hall dataset: Trajectories of the four users.

widely used BLAS libraries: OpenBLAS [45] (developed from GotoBLAS2 [46]), EigenBLAS (the BLAS supported by the Eigen library), and Intel MKLBLAS [47]. MKLBLAS turns out to be the fastest one, and it is the one we chose to use.

In the proposed algorithm, as shown in Sect. V-A, every user first solves its own single-map (SM) estimation problem via BLS to determine its local trajectory and map. To do so, each user requires the following information:

(SM 1) An initial estimate for its trajectory and map: In our case, this is computed using an extension of the multi-state constrained Kalman filter (MSC-KF) [48], based on [16] and [49]. Each MSC-KF operates on the IMU measurements and feature tracks collected by each user. Feature tracks correspond to Harris-corners [50] extracted from each image and tracked using the Kanade-Lucas-Tomasi (KLT) algorithm [51]. The subset of these feature tracks that pass the 2pt-RANSAC [52], are used by the MSC-KF, along with the IMU data, to estimate the trajectory of each user. These tracks, however, are not employed for loop-closure detection.

(SM 2) Intra-dataset point-feature loop-closure detection: To determine if a user has revisited an area, we follow a bag-of-words approach using ORB feature descriptors [53] and employ our implementation of Nistér's vocabulary tree [54]. These matches are confirmed after they pass a 3pt+1-RANSAC [55] geometric-consistency test.

(SM 3) Line tracking and intra-dataset line-feature loop-closure detection: Algorithm 1 summarizes the main steps of the line-tracking process. First, from each image $\mathscr{I}_k$, $k = 1,\ldots,K$, line segments are extracted using the line segment detection (LSD) algorithm [56], and the corresponding line normals are assigned to the set $\mathscr{F}_k$ (see lines 2-4 of Algorithm 1). Then, for $k \geq 3$, each line normal $\mathbf{s}_k \in \mathscr{F}_k$ is tested against the current (accepted as valid) 3D line hypotheses $\mathscr{L}$ (initially an empty set) by computing the re-projection errors in (9) and (10). Note that each hypothesis comprises the triangulated line parameters and the corresponding set of pairs of image indices and line-normal measurements. The line normals which pass the test are added to the list of their accepting hypothesis and are removed from $\mathscr{F}_k$ to avoid re-processing (see lines 6-12 of Algorithm 1). Additionally, the method for computing the 4-dof line parameters based on measurements [see $Meas(h)$ in line 8 of Algorithm 1] corresponding to each hypothesis h is described in Appendix D.

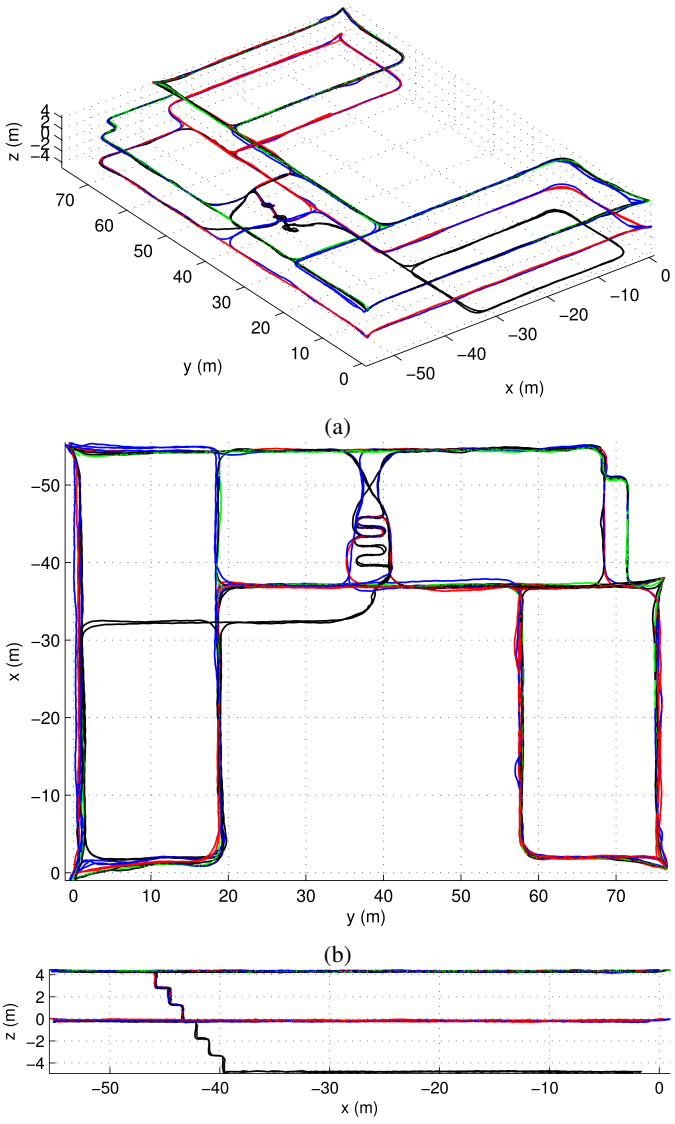


Fig. 6: Keller Hall dataset: Merged trajectories of all users: (a) 3D, (b) y-x, and (c) x-z views. Each user's trajectory is depicted with a different color.

Afterwards, as shown in lines 13-19 of Algorithm 1, the line-tracking process creates a new set $\mathscr{H}$ of candidate 3D lines by assuming each possible pair of line normals $\mathbf{s}_{k-2} \in \mathscr{F}_{k-2}$ and $\mathbf{s}_{k-1} \in \mathscr{F}_{k-1}$ corresponds to measurements of a single 3D line observed at poses $k-2$ and $k-1$, respectively.

Subsequently, each hypothesis $h \in \mathscr{H}$ is validated by computing the re-projection error of each line normal $\mathbf{s}_k \in \mathscr{F}_k$. If this is smaller than a threshold (currently set to 0.01), h is updated with $\mathbf{s}_k$ and is added to $\mathscr{L}$. These steps are shown in lines 20-27 of Algorithm 1.

Once the line-tracking process is completed, the subset of line tracks corresponding to the cardinal directions of the building (i.e., Manhattan lines) are identified using a RANSAC-based vanishing-point estimator [57]. Lastly, we use the trajectory and line BLS estimates of each user to search for loop-closure line features at poses where loop-closure

Input: $\mathscr{I}_k,\ k=1,\ldots,K$ the set of images viewed by each pose k of the user j; the camera poses ${}^j\check{\mathbf{p}}$ of the user j
Output: The set of line tracks $\mathscr{L}$

$\mathscr{L} \leftarrow \{\}$; // Initially empty set of 3D lines
// Extract line normals of images
for $k=1:K$ **do**
  $\mathscr{F}_k \leftarrow LSD\_Extract(\mathscr{I}_k)$
end
for $k=3:K$ **do**
  // Test current line measurements against previous hypotheses
  foreach $h \in \mathscr{L}$, $\mathbf{s} \in \mathscr{F}_k$ **do**
    if $ReprojectionLineTest(h, \mathbf{s}, {}^j\check{\mathbf{p}})$ **then**
      $Meas(h) \leftarrow Meas(h) \cup \{(k,\mathbf{s})\}$
      $Param(h) \leftarrow TriangulateLine(Meas(h), {}^j\check{\mathbf{p}})$
      $\mathscr{F}_k \leftarrow \mathscr{F}_k - \{\mathbf{s}\}$
    end
  end
  // New hypotheses generation
  $\mathscr{H} \leftarrow \{\}$; // Potential line hypotheses
  foreach $\mathbf{s}_{k-1} \in \mathscr{F}_{k-1}$, $\mathbf{s}_{k-2} \in \mathscr{F}_{k-2}$ **do**
    $h \leftarrow \varnothing$
    $Meas(h) \leftarrow \{(k-2,\mathbf{s}_{k-2}),(k-1,\mathbf{s}_{k-1})\}$
    $Param(h) \leftarrow TriangulateLine(Meas(h), {}^j\check{\mathbf{p}})$
    $\mathscr{H} \leftarrow \mathscr{H} \cup \{h\}$
  end
  // Test current line measurements against new hypotheses
  foreach $h \in \mathscr{H}$, $\mathbf{s} \in \mathscr{F}_k$ **do**
    if $ReprojectionLineTest(h, \mathbf{s}, {}^j\check{\mathbf{p}})$ **then**
      $Meas(h) \leftarrow Meas(h) \cup \{(k,\mathbf{s})\}$
      $Param(h) \leftarrow TriangulateLine(Meas(h), {}^j\check{\mathbf{p}})$
      $\mathscr{F}_k \leftarrow \mathscr{F}_k - \{\mathbf{s}\}$
      $\mathscr{L} \leftarrow \mathscr{L} \cup \{h\}$
    end
  end
end

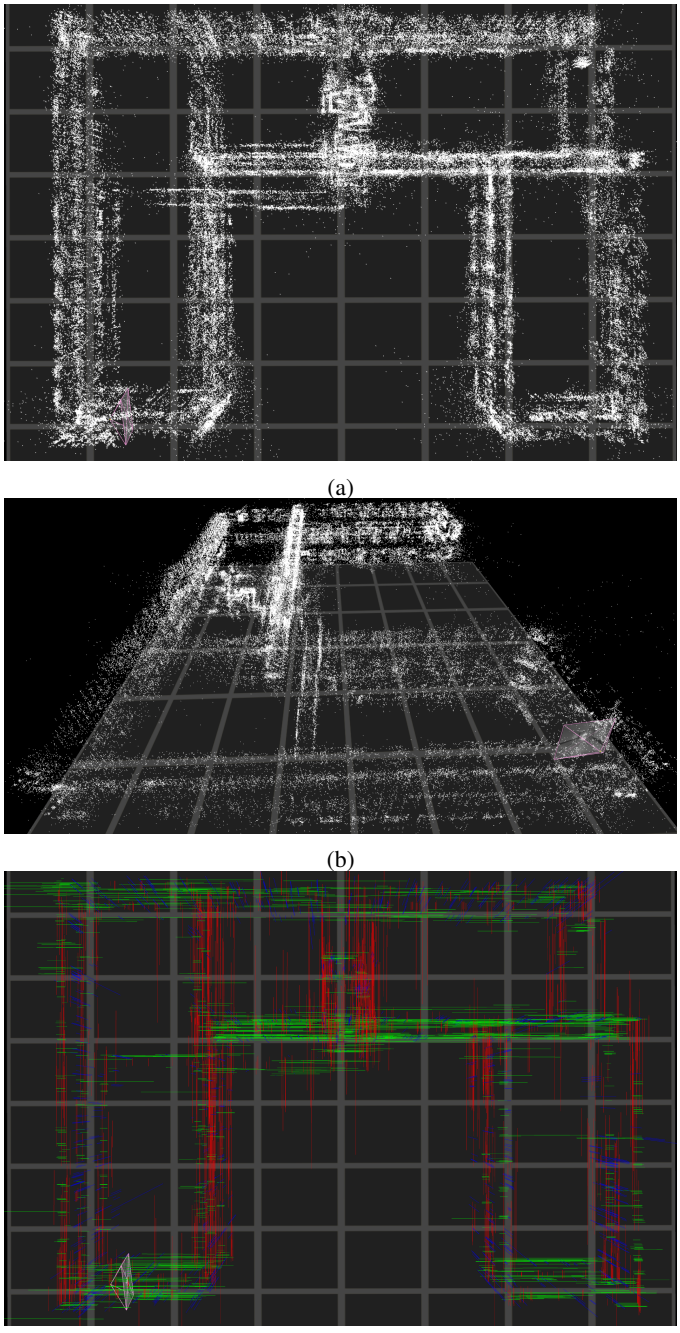
Algorithm 1: Line tracking process. Each line track consists of a set of line normals viewed by different poses.

point features have previously been found.[8] In particular, if any free or Manhattan lines are close to each other (i.e., the difference of their distance and direction parameters are within one degree and 15 cm, respectively), they are accepted as loop-closure measurements.

Once each user has solved its own SM problem, they communicate to the CM algorithm their estimated trajectories, maps, Cholesky factors, and all available visual-inertial measurements. At this point, another step of preprocessing is required to compute the following quantities:

(CM 1) Inter-dataset point-feature loop-closure detection: To achieve this, we follow the same procedure as in (SM 2),

[8]This requirement (i.e., to concurrently have a point-feature loop-closure) is imposed so as to increase robustness to line mismatching.



(a) (b) (c)

Fig. 7: Keller Hall dataset: (a)-(b) 3D point cloud from two viewing directions. (c) 3D Manhattan-line features following the cardinal directions colored with red (x), green (y), and blue (z), respectively.

and determine the matched images and corresponding common features across all users' datasets.

(CM 2) Relative-transformation initialization: Once common point features are identified, we follow the approach of Sect. V-B to compute an initial estimate for the unknown, 4-dof transformation between each user pair.

(CM 3) Inter-dataset line-feature loop-closure detection: We follow a process similar to (SM 3) using the CM camera pose estimates.

Once the above pre-processing steps are completed, we employ the CM algorithm of Sect. V-C to estimate the trajectories of all users, as well as the combined building map. The results from our experiments are summarized in the following sections.

B. Evaluation Results on the Keller Hall Dataset

This dataset is acquired by four users navigating through the building. The CM estimated user trajectories are shown in Fig. 5 and Fig. 6. Figs. 5 (a)-(d) correspond to trajectories of approximately 1300, 1000, 800, and 500 m, respectively, from which 181,140 points, 945 free lines, and 4,511 Manhattan lines are processed by the CM algorithm. Of these features, 4,243 points, 18 free lines, and 148 Manhattan lines are common to two or more datasets. Additionally, we present the estimated 3D point cloud and lines in Fig. 7.

1) CM with Point and Line Features: The achieved accuracy of the CM algorithm can be qualitatively assessed by observing the CM estimated trajectories of all users for the Keller Hall datasets shown in Fig. 6. Note that we intentionally instructed the users collecting data to keep the camera at about the same height, and walk in the middle of fairly narrow corridors. Correspondingly, the z (height) estimated for all users' trajectories remains about the same in the $x-z$ view of the trajectory estimates [see Fig. 6 (c)], despite the fact that they have travelled for hundreds of meters across multiple floors. Moreover, in the $x-y$ view [see Fig. 6 (b)], the user trajectories on different floors overlap almost exactly.

In addition to the qualitative results, we also present a ground-truth comparison. Specifically, we placed four AprilTags [58] in the far corners of a single floor within the building and used the building's blueprints to determine the true distance (ground truth) between any pair of AprilTags. On the other hand, to compute the estimated distance between AprilTags, we first employ the PnP algorithm of [59] to find the observing camera's pose, $({}^{A_n}\mathbf{p}_{C_k}, {}^{A_n}_{C_k}\mathbf{C})$, with respect to each AprilTag's frame, and then use the CM estimate of the camera's global pose, $({}^{G}\mathbf{p}_{C_k}, {}^{G}_{C_k}\mathbf{C})$, to express each AprilTag with respect to the global frame $\{G\}$ as:

$$ {}^{G}\mathbf{p}_{A_n} = {}^{G}\mathbf{p}_{C_k} - {}^{G}_{C_k}\mathbf{C}\,{}^{A_n}_{C_k}\mathbf{C}^{T}\,{}^{A_n}\mathbf{p}_{C_k} \tag{44} $$

Next, we average the estimated positions of each AprilTag across all camera observations and compute the estimated pairwise distances between AprilTags. For example, the estimated distance between two AprilTags, A_{n_1} and A_{n_2}, is computed as $\|{}^{G}\mathbf{p}_{A_{n_1}} - {}^{G}\mathbf{p}_{A_{n_2}}\|$. Lastly, we assess the CM algorithm's accuracy by comparing the difference between the estimated and true distances between all AprilTags. Note the distance from the camera to the AprilTag is relatively small as compared to the distance between AprilTags (about 0.4 m compared to over 80 m) so any error in the PnP estimate will negligibly affect the result.

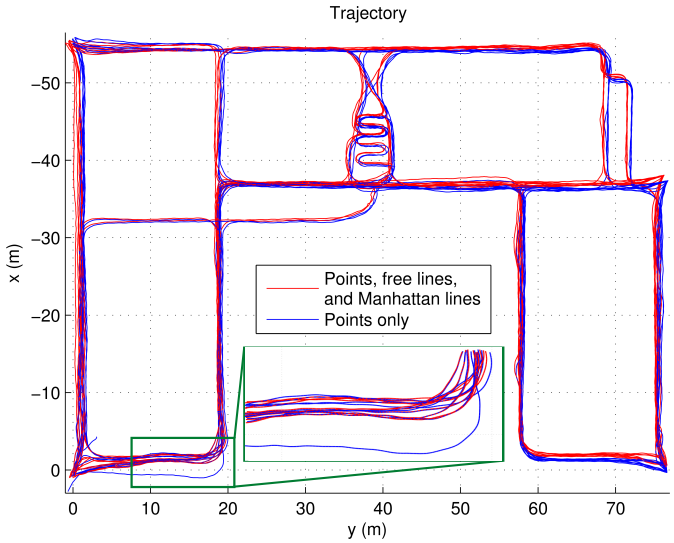


Fig. 8: Trajectories of all users in Keller Hall using points-only versus points, free lines, and Manhattan lines.

Algorithm	Initialization	8 m grid	4 m grid	1 m grid	Exact
Constraint dimension	0	1050	2063	5877	16912
Feature count	0	223	488	1507	4243
Position RMSD (cm)	258.3	13.5	8.8	5.7	0

TABLE I: Keller Hall dataset: Number of common point features, the dimension of common point, free-line, and Manhattan-line feature constraints, and position RMSD of the approximate CM solutions as compared to the exact CM.

The effect of using free and Manhattan lines is depicted in Fig. 8. In particular, the absolute/relative errors in the pairwise distances between AprilTags estimated by the CM algorithm when using only points versus when using points, free-lines, and Manhattan-lines are 63 cm/0.84% and 48 cm/0.64%, respectively. Furthermore, at the part of the trajectory shown as zoomed-in at the bottom-left of Fig. 8, one of the users explores a new area where no loop-closure information is available. In this case, when only points are used, the estimated user trajectory deviates about 1.7 m from the corridor, while this error is corrected by processing line features. In addition to the increased number of measurements when processing lines, the improved accuracy is attributed to the availability of orientation information (from Manhattan lines) about the users with respect to a single, common building frame. Lastly, the trajectory estimated when employing only point features has a small, but noticeable yaw error. This error, as expected, is corrected when Manhattan lines are also used since they provide attitude information between the user's and the building's frames, which makes the yaw angle observable.

2) Sparse CM: Hereafter, we compare the exact CM algorithm with the following four approximations: (i) The initialization (zero-th iteration) of the CM algorithm; that is, the maps and trajectories resulting when we align the BLS-estimated single-user maps using the inter-dataset transforma-

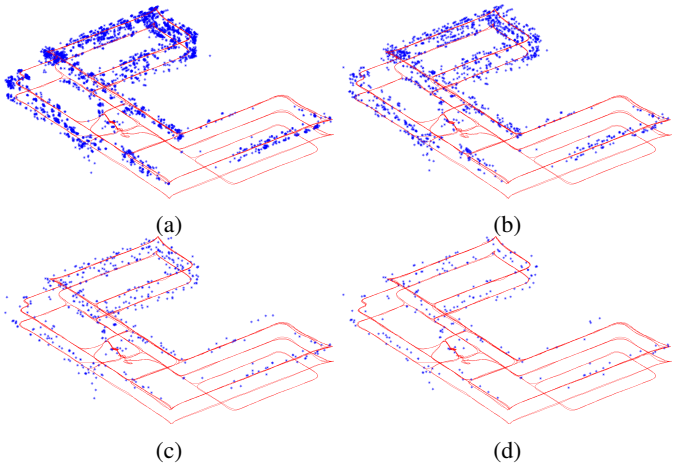


Fig. 9: Common point features in the Keller Hall dataset: (a) Original CM, sparse CM using grid cells of size (b) 1×1 m^2, (c) 4×4 m^2, (d) 8×8 m^2.

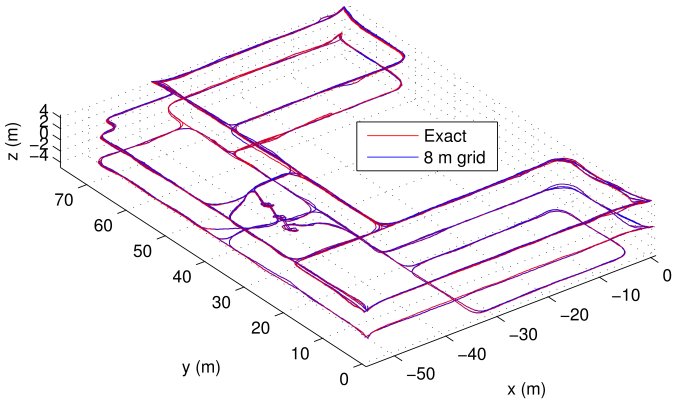


Fig. 10: Keller Hall dataset: Trajectories estimated from the original CM and sparse CM with common-feature sparsification using grid cells of size 8×8 m^2.

tion computed by the method of Sect. V-B; (ii)-(iv) Sparse CM, i.e., CM with commonly-observed feature sparsification (see Sect. V-D), using grid cells of size 1×1 m^2, 4×4 m^2, and 8×8 m^2, respectively. Since the number of common free-line and Manhattan-line features is small, we only sparsified the point features. The common point features before and after sparsification are shown in Fig. 9.

The resulting number of common point features, the dimension of the total common point, free-line, and Manhattan-line feature constraints [see (43)], as well as the Root Mean Square Difference (RMSD) of the position with respect to the exact CM estimates of (i)-(iv) are shown in Table I. In order to visually inspect the impact of this level of position RMSD on the estimated user trajectories, we depict the ones computed from the sparse CM using grid cells of size 8×8 m^2 against those found by the exact CM in Fig. 10. Notice that the sparse CM preserves about 5% of the common-feature constraints, while obtaining an estimate that has very small position difference from that of the exact CM. The intuition behind this result is that since only two point features are required to compute the transformation between two maps (see Sect. V-B), a few hundred (instead of several thousand) common-feature constraints suffice for creating an accurate merged map.

Algorithm	8 m grid	4 m grid	1 m grid	Exact
$\mathbf{G}_j$	13.5	13.5	13.5	13.5
$\mathbf{K}_j$	4.8	9.7	29.3	85.7
$\mathbf{K}^T\mathbf{K}$	1.2	4.5	36.4	334.2
$\mathbf{T}_{11}$	0.04	0.2	5.0	114.5
CM total	19.5	27.9	84.2	547.9
BLS	32.9	34.4	37.4	41.6

TABLE II: Keller Hall dataset: Timing comparison between CM and BLS with different number of commonly-observed features (sec/iteration).

The timing results of the exact and sparse CM algorithms are reported in Table II.[9] Note that all the algorithms typically converge after 5-10 Gauss-Newton iterations. For comparison, we also report the timing results for solving the equivalent BLS formulation [see (33)] using the Cholmod Cholesky-factorization algorithm.[10] As evident from Table II, the sparse CM takes significantly less time compared to the exact CM solution. Moreover, when the number of common features is in the order of a few hundred (as shown in Table I, 233 for 8 m grid and 488 for 4 m grid), the proposed CM solver is significantly faster than the equivalent BLS solver. Furthermore, since the Cholesky factors ($\mathbf{G}_j$ matrices) can be reused from each single-user BLS in the first iteration, additional time can be saved by the CM algorithm. At this point, we should note that, as shown in Table II, the time for computing the matrices $\mathbf{G}_j$ is constant, while finding $\mathbf{K}_j$, $\mathbf{K}^T\mathbf{K}$, and $\mathbf{T}_{11}$ has complexity linear, quadratic, and cubic, respectively, in the dimension of the common-feature constraint. This result verifies the computational complexity analysis of Sect. V-C3, and confirms that the processing cost of the CM algorithm can be controlled by adjusting the number of commonly-observed features.

Lastly, in Table III we compared the peak memory usage between the proposed CM algorithm, when computing the Cholesky factorizations corresponding to each dataset sequentially, and the equivalent BLS solver. The CM solver maximizes its memory use when computing the Cholesky factor, $\mathbf{G}_j$, of the Hessian matrix corresponding to the *largest* dataset, which is independent of the number of commonly-observed features. Analogously, the peak memory usage of the BLS solver occurs during the Cholesky factorization of the Hessian corresponding to the *merged* dataset, which is about 4 times larger than the CM's Hessian corresponding to the largest dataset. Since the memory requirements of the Cholesky factorization increase almost linearly (for sparse matrices) with the size of the Hessian, the BLS requires approximately 4 times more memory than the CM algorithm. Furthermore, this memory usage grows with the number of common features, as these introduce correlations between different datasets. Note that while the results in this paper pertain to running on a desktop PC, many of the potential use cases require running CM on mobile devices. On such

[9]The timings for CM initialization is not listed because it does not perform the same computations as the other methods, but instead it directly aligns the users' trajectories and maps.

[10]In all cases, the *same* common-feature constraints are employed in the CM and BLS formulations. Hence, the CM and BLS solvers produce the same solutions (within numerical precision) but with different timings.

Algorithm	8 m grid	4 m grid	1 m grid	Exact
CM	1.37	1.37	1.37	1.37
BLS	4.41	4.72	4.97	5.60

TABLE III: Keller Hall dataset: Peak memory usage comparison between the proposed CM algorithm and the BLS formulation (in GB).

	$\mathbf{G}_j$	$\mathbf{K}_j$	$\mathbf{K}^T\mathbf{K}$	$\mathbf{T}_{11}$	CM total	BLS
Time	7.94	3.47	0.52	0.07	11.95	16.33

TABLE IV: Walter Library dataset: Timing comparison between CM and BLS (sec/iteration).

hardware, the memory footprint of a mapping algorithm is of particular importance due to limited available RAM (e.g., the Project Tango tablet has 4 GB of RAM out of which 1 GB is used by the Android operating system).

C. Evaluation Results on the Walter Library Dataset

This dataset is collected by three users when walking for approximately 600, 600, and 650 m. The CM-estimated users' trajectories are shown in Fig. 11 and Fig. 12. Note that the height change of the trajectory depicted in black [see Fig. 12 (c)] is because of the two ramps on the first floor. Since the users' trajectories have limited overlap, only 200 features (see Fig. 13), amongst the 78,868 point features processed by the CM, are commonly observed by more than one user. For this reason, no common-feature constraint sparsification is necessary in this case. Instead, we will focus our analysis on the exact CM algorithm's computation and memory requirements as compared to the standard BLS. The timings for computing the CM and BLS solutions are reported in Table IV, and their required memory usage are 0.93 GB and 2.44 GB, respectively. Note that the CM algorithm has a 37% speedup and 163% memory savings. Moreover, recall that the processing bottleneck of the CM solution for the Keller Hall dataset was computing the matrices $\mathbf{K}_j$, $\mathbf{K}^T\mathbf{K}$, and $\mathbf{T}_{11}$. As these costs decrease with the number of common features, computing the Cholesky factors $\mathbf{G}_j$ corresponding to each user's dataset, becomes the most expensive operation.

For the purpose of completeness, we also provide the sparse CM memory, timing, and accuracy results in Table V. As compared to the Keller Hall dataset, since there are fewer common features, and thus a lower percentage of them is pruned, the sparse CM accuracy loss is also smaller.

Algorithm	Initialization	8 m grid	4 m grid	1 m grid	Exact
Constraint dimension	0	99	174	369	600
Feature count	0	33	58	123	200
CM time (sec/iter)	0	8.8	9.2	10.7	12.0
BLS time	0	15.4	15.7	15.9	16.3
CM memory (GB)	0	0.9	0.9	0.9	0.9
BLS memory	0	2.1	2.2	2.3	2.4
Position RMSD (cm)	44.0	7.9	5.3	2.7	0

TABLE V: Walter Library dataset: Three user trajectories of approximately 600, 600, and 650 m.

Algorithm	Initialization	8 m grid	4 m grid	1 m grid	Exact
Constraint dimension	0	222	591	1821	4647
Feature count	0	73	194	598	1534
CM time (sec/iter)	0	31.1	35.3	58.4	158.7
BLS time	0	46.4	48.5	60.9	63.2
CM memory (GB)	0	2.9	2.9	2.9	2.9
BLS memory	0	5.5	5.7	6.9	7.2
Position RMSD (cm)	60.2	10.0	8.7	4.5	0

TABLE VI: RSS 2015 conference site dataset: Three user trajectories of approximately 700, 800, and 400 m.

Algorithm	Initialization	8 m grid	4 m grid	1 m grid	Exact
Constraint dimension	0	711	1638	5472	12129
Feature count	0	222	514	1695	3780
CM time (sec/iter)	0	26.8	40.3	122.7	433.3
BLS time	0	44.6	47.7	49.5	54.5
CM memory (GB)	0	1.7	1.8	1.8	1.8
BLS memory	0	4.8	4.9	5.4	5.9
Position RMSD (cm)	68.4	29.3	13.4	9.1	0

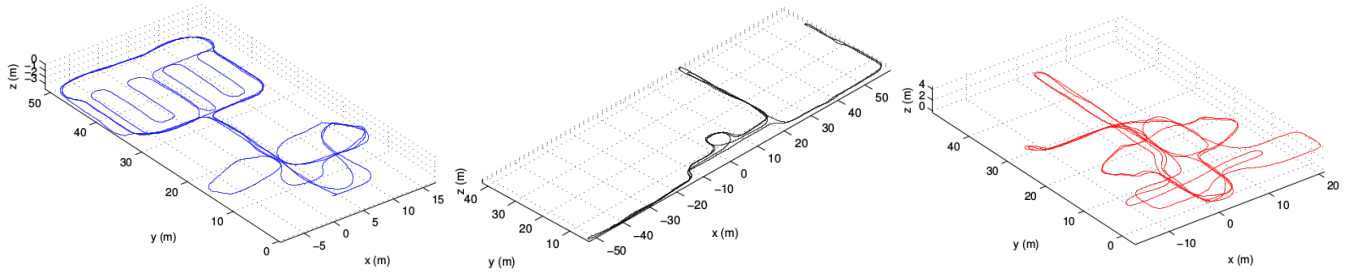
TABLE VII: CVPR 2016 conference site dataset: Three user trajectories of approximately 950, 500, and 1000 m.

D. Evaluation Results on Additional Datasets

In this section, we quantitatively evaluate the exact and sparse CM algorithms on three more datasets: RSS 2015 conference site, CVPR 2016 conference site, and MIT Stata Center, which are included in our online interactive visualization [24]. The corresponding timing results, position RMSD, peak memory usage, as well as number of common-feature constraints are shown in Tables VI, VII, and VIII, respectively. Similar to the conclusion drawn before, in all these three datasets, the sparse CM algorithm has typically lower processing and memory requirements as compared to the BLS solver.

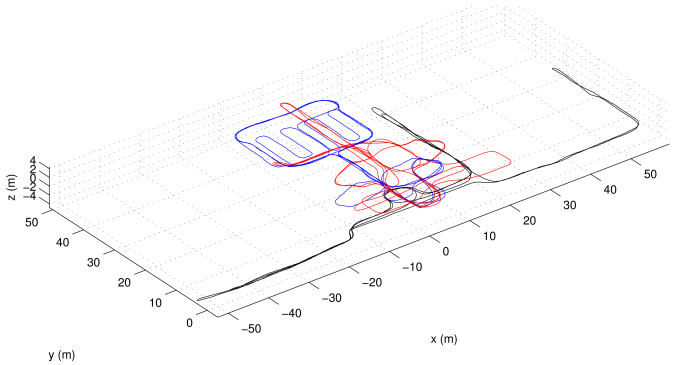
One interesting finding is that the position RMSD of the CM initialization is much larger on the longest testing dataset (the MIT Stata Center) as compared to the others. This is because when aligning different users' trajectories, the position RMSD introduced by their relative orientation error increases proportionally to the trajectory radius. For the same reason, in this dataset, the position error of the exact CM solution, as well as the position difference between the exact and sparse CM solutions, is also expected to be larger. Another important thing to note is that the mapping accuracy and processing cost also strongly depend on the building's size. For example, as compared to the Keller Hall, the CVPR 2016 conference site is a significantly larger building (i.e., contains more features), while the total trajectory length in the collected dataset is shorter (i.e., less coverage of each corridor). As a result, both the CM and BLS solutions are more expensive on this dataset, while the resulting position RMSD is still higher.

In summary, as compared to the standard BLS, the CM algorithm is significantly more efficient in terms of both computational cost and memory usage when the number of common features is small. On the other hand, as the number of feature constraints increases, the CM solution remains less memory demanding but requires more computations. In such cases, we have the option of incurring a minimal loss

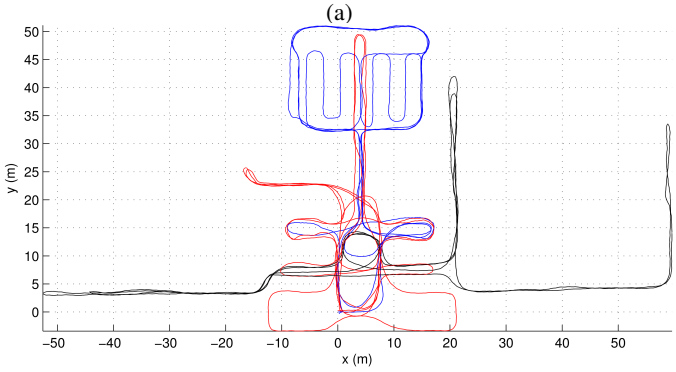


(a) (b) (c)

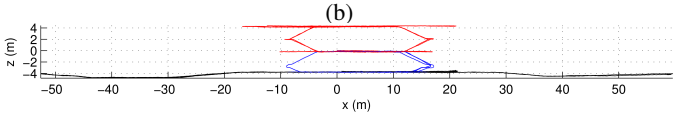
Fig. 11: Walter Library dataset: Trajectories of the three users.



(a)



(b)



(c)

Fig. 12: Walter Library dataset: Merged trajectories of all users from (a) 3D, (b) x-y, and (c) x-z views. Each user's trajectory is depicted with a different color.

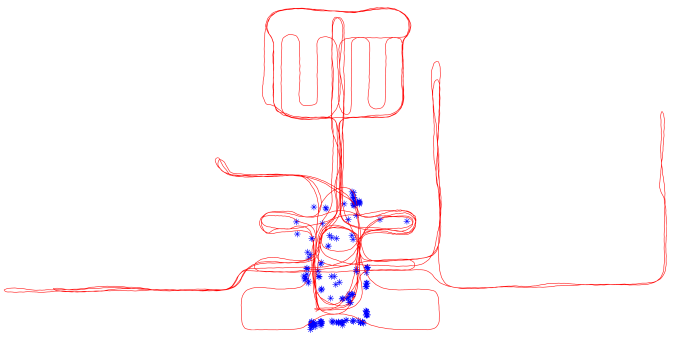


Fig. 13: Common point features in the Walter Library dataset.

Algorithm	Initialization	8 m grid	4 m grid	1 m grid	Exact
Constraint dimension	0	831	1905	5853	10179
Feature count	0	221	526	1652	2892
CM time (sec/iter)	0	23.7	30.4	81.7	191.3
BLS time	0	58.2	60.6	64.7	65.5
CM memory (GB)	0	1.3	1.3	1.3	1.3
BLS memory	0	6.1	6.6	7.5	7.7
Position RMSD (cm)	974.3	48.1	28.9	17.0	0

TABLE VIII: MIT Stata Center dataset: Seven user trajectories of approximately 1050, 1050, 1550, 150, 400, 1300, and 700 m.

of estimation accuracy for a significant gain in speed by processing only a subset of the common features selected as described in Sect. V-D. Finally, we note that as a highly parallelizable algorithm, CM will have even larger processing time savings compared to BLS if running on a more powerful computer, e.g., on the cloud.

VII. Conclusion and Future Work

In this paper, we introduced a cooperative mapping (CM) algorithm for combining visual and inertial measurements collected using multiple mobile devices at different times across large indoor spaces. We considered the most general case, where the users' relative transformation is not known and cannot be inferred by directly observing each other. Our formulation of CM as a batch least-squares (BLS)-equivalent constrained-optimization problem offers significant advantages when processing multiple maps: (i) Resource awareness, as the computational cost can be adjusted by selecting a subset of commonly-observed feature constraints; (ii) Computational gains, as the most computationally intensive operations are parallelizable; (iii) Memory savings, as Cholesky factorization is applied on the Hessian of each user's data instead of the Hessian corresponding to the merged data from all users. Furthermore, in addition to point features, we utilized free-line and Manhattan-line features for improving the estimation accuracy by making the yaw angle observable. Five large-scale CM experiments were conducted in order to assess the performance of the proposed algorithm, while more qualitative evaluations can be found in our online interactive visualization [24].

As part of our future work, we plan to further investigate the impact of each common feature on the mapping accuracy. In particular, we seek to find geometric criteria for efficiently determining the common-feature constraints that, when removed, will cause minimum increase in the resulting CM covariance.

APPENDIX A
SOLUTION TO THE LEAST-SQUARES PROBLEM WITH A QUADRATIC CONSTRAINT

The KKT optimality conditions [36] for (29) are:

$$\mathbf{A}^T(\mathbf{A}\mathbf{w}-\mathbf{b})+\lambda\mathbf{w}=\mathbf{0} \tag{45}$$

$$\mathbf{w}^T\mathbf{w}-1=0 \tag{46}$$

Expressing $\mathbf{w}$ with respect to λ from (45) and substituting into (46) yields:

$$\mathbf{b}^T\mathbf{A}(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-2}\mathbf{A}^T\mathbf{b}=1 \tag{47}$$

Next, we define:

$$\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I}=\begin{bmatrix}a_{11}+\lambda & a_{12}\\ a_{12} & a_{22}+\lambda\end{bmatrix},$$

compute its inverse analytically

$$(\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I})^{-1}=\frac{1}{(a_{11}+\lambda)(a_{22}+\lambda)-a_{12}^2}\begin{bmatrix}a_{22}+\lambda & -a_{12}\\ -a_{12} & a_{11}+\lambda\end{bmatrix}$$

and substitute it in (47) to get the following equation:

$$\begin{aligned}&\mathbf{b}^T\mathbf{A}\begin{bmatrix}a_{22}+\lambda & -a_{12}\\ -a_{12} & a_{11}+\lambda\end{bmatrix}^2\mathbf{A}^T\mathbf{b}\\ &=\left[(a_{11}+\lambda)(a_{22}+\lambda)-a_{12}^2\right]^2\end{aligned} \tag{48}$$

Note that (48) is quartic in λ and can be solved in closed form. Once λ is obtained, $\mathbf{w}$ can be computed through back-substitution using (45).

APPENDIX B
PROOF OF THEOREM 1

We prove the theorem for the simple case of a point feature commonly observed by two users. The extension to different types of features and multiple users is straightforward.

Consider a point feature, ${}^{C_{\eta_1}}\mathbf{x}_P$, defined with respect to the camera pose $\{C_{\eta_1}\}$ of user 1, and also observed by user 2 at camera pose $\{C_{k_2}\}$. From (33), the BLS formulation of this problem is:

$$\begin{aligned}&\mathbf{x}_\tau^*,\mathbf{x}_a^*,\mathbf{f}_a^*,{}^{C_{\eta_1}}\mathbf{x}_P^*\\ &=\text{argmin}\left(\sum_{j=1}^{2}\overline{\mathscr{C}}_j+\|{}^{C_{k_2}}\mathbf{z}-\boldsymbol{\pi}\left({}^{C_{k_2}}\mathbf{p}_{C_{\eta_1}}+{}^{C_{k_2}}_{C_{\eta_1}}\mathbf{C}\,{}^{C_{\eta_1}}\mathbf{x}_P\right)\|_{\mathbf{R}}^2\right)\end{aligned} \tag{49}$$

where the relative camera pose $({}^{C_{k_2}}\mathbf{p}_{C_{\eta_1}},\ {}^{C_{k_2}}_{C_{\eta_1}}\mathbf{C})$ can be expressed as a function of the system states $\mathbf{x}_\tau$ and $\mathbf{x}_a$ as:

$$\begin{aligned}{}^{C_{k_2}}\mathbf{p}_{C_{\eta_1}}&={}^{C_{k_2}}_{G_2}\mathbf{C}\left({}^{G_1}_{G_2}\mathbf{C}^T\left({}^{G_1}\mathbf{p}_{C_{\eta_1}}-{}^{G_1}\mathbf{p}_{G_2}\right)-{}^{G_2}\mathbf{p}_{C_{k_2}}\right)\\ {}^{C_{k_2}}_{C_{\eta_1}}\mathbf{C}&={}^{C_{k_2}}_{G_2}\mathbf{C}\,{}^{G_1}_{G_2}\mathbf{C}^T\,{}^{G_1}_{C_{\eta_1}}\mathbf{C}\end{aligned}$$

On the other hand, the proposed CM formulation is:

$$\mathbf{x}_\tau^*,\mathbf{x}_a^*,\mathbf{f}_a^*,{}^{C_{\eta_1}}\mathbf{x}_P^*,{}^{C_{\eta_2}}\mathbf{x}_P^*=\text{argmin}\sum_{j=1}^{2}\mathscr{C}_j \tag{50}$$

$$\text{s. t.}\quad {}^{C_{\eta_2}}\mathbf{x}_P={}^{C_{\eta_2}}_{C_{\eta_1}}\mathbf{C}\,{}^{C_{\eta_1}}\mathbf{x}_P+{}^{C_{\eta_2}}\mathbf{p}_{C_{\eta_1}} \tag{51}$$

where the same feature also appears as ${}^{C_{\eta_2}}\mathbf{x}_P$ with respect to a camera pose $\{C_{\eta_2}\}$ of user 2. Note that in this case, the relative user transformation $({}^{G_1}\mathbf{p}_{G_2},\ {}^{G_1}_{G_2}\mathbf{C})$ appears in the constraint (51), instead of the cost function. Specifically, the relative pose $({}^{C_{\eta_2}}\mathbf{p}_{C_{\eta_1}},\ {}^{C_{\eta_2}}_{C_{\eta_1}}\mathbf{C})$ between the two camera poses with respect to which this feature is expressed in the two users' maps is:

$$\begin{aligned}{}^{C_{\eta_2}}\mathbf{p}_{C_{\eta_1}}&={}^{C_{\eta_2}}_{G_2}\mathbf{C}\left({}^{G_1}_{G_2}\mathbf{C}^T\left({}^{G_1}\mathbf{p}_{C_{\eta_1}}-{}^{G_1}\mathbf{p}_{G_2}\right)-{}^{G_2}\mathbf{p}_{C_{\eta_2}}\right)\\ {}^{C_{\eta_2}}_{C_{\eta_1}}\mathbf{C}&={}^{C_{\eta_2}}_{G_2}\mathbf{C}\,{}^{G_1}_{G_2}\mathbf{C}^T\,{}^{G_1}_{C_{\eta_1}}\mathbf{C}\end{aligned}$$

As mentioned earlier, in CM, we employ the nominal camera measurement model [see (6)] to express a feature's observation by user 2. Thus, the corresponding cost term for this measurement appearing in $\mathscr{C}_2$ [see (50)] is:

$$\|{}^{C_{k_2}}\mathbf{z}-\boldsymbol{\pi}\left({}^{C_{k_2}}\mathbf{p}_{C_{\eta_2}}+{}^{C_{k_2}}_{C_{\eta_2}}\mathbf{C}\,{}^{C_{\eta_2}}\mathbf{x}_P\right)\|_{\mathbf{R}}^2 \tag{52}$$

Substituting the constraint (51) into the cost term (52) results in:

$$\begin{aligned}&\|{}^{C_{k_2}}\mathbf{z}-\boldsymbol{\pi}\left({}^{C_{k_2}}\mathbf{p}_{C_{\eta_2}}+{}^{C_{k_2}}_{C_{\eta_2}}\mathbf{C}\,{}^{C_{\eta_2}}\mathbf{x}_P\right)\|_{\mathbf{R}}^2\\ =\;&\|{}^{C_{k_2}}\mathbf{z}-\boldsymbol{\pi}\left({}^{C_{k_2}}\mathbf{p}_{C_{\eta_2}}+{}^{C_{k_2}}_{C_{\eta_2}}\mathbf{C}\left({}^{C_{\eta_2}}_{C_{\eta_1}}\mathbf{C}\,{}^{C_{\eta_1}}\mathbf{x}_P+{}^{C_{\eta_2}}\mathbf{p}_{C_{\eta_1}}\right)\right)\|_{\mathbf{R}}^2\\ =\;&\|{}^{C_{k_2}}\mathbf{z}-\boldsymbol{\pi}\left({}^{C_{k_2}}\mathbf{p}_{C_{\eta_1}}+{}^{C_{k_2}}_{C_{\eta_1}}\mathbf{C}\,{}^{C_{\eta_1}}\mathbf{x}_P\right)\|_{\mathbf{R}}^2\end{aligned} \tag{53}$$

which is the same as the corresponding cost term in the BLS formulation (49). Thus, the CM (34) and BLS (33) formulations are equivalent in this case. ■

APPENDIX C
COMPUTING THE COVARIANCE MATRIX OF THE CM SOLUTION

We hereafter present how to compute the covariance of the CM estimates as a function of the matrices $\mathbf{K}_j$, $\mathbf{G}_j$, $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, and $\mathbf{T}_{22}$ [see (39)-(42)] for the case of two users.

Permuting the third and fourth block rows and columns of (37) yields the following equivalent linearized system:

$$\underbrace{\begin{bmatrix}\mathbf{J}_1{}^T\mathbf{J}_1 & \mathbf{0} & \mathbf{0} & \mathbf{A}_1^T\\ \mathbf{0} & \mathbf{J}_2{}^T\mathbf{J}_2 & \mathbf{0} & \mathbf{A}_2^T\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_\tau^T\\ \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_\tau & \mathbf{0}\end{bmatrix}}_{\mathbf{C}}\begin{bmatrix}\delta^1\mathbf{x}_u\\ \delta^2\mathbf{x}_u\\ \delta\mathbf{x}_\tau\\ \lambda\end{bmatrix}=\begin{bmatrix}\mathbf{J}_1^T\mathbf{b}_1\\ \mathbf{J}_1^T\mathbf{b}_2\\ \mathbf{0}\\ \mathbf{r}\end{bmatrix} \tag{54}$$

As shown in [60], the covariance of the error-state vector in (54), $\begin{bmatrix}\delta^1\mathbf{x}_u^T & \delta^2\mathbf{x}_u^T & \delta\mathbf{x}_\tau^T\end{bmatrix}^T$, which is the same as the one computed when solving the unconstrained optimization problem, is equal to the top-left 3×3 block of $\mathbf{C}^{-1}$. To derive $\mathbf{C}^{-1}$, we first partition $\mathbf{C}$ as:

$$\mathbf{C}=\begin{bmatrix}\mathbf{J}^T\mathbf{J} & \mathbf{Y}\\ \mathbf{Y}^T & \Theta\end{bmatrix} \tag{55}$$

where

$$\mathbf{J}^T\mathbf{J}\triangleq\begin{bmatrix}\mathbf{J}_1{}^T\mathbf{J}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{J}_2{}^T\mathbf{J}_2\end{bmatrix},\quad \mathbf{Y}\triangleq\begin{bmatrix}\mathbf{0} & \mathbf{A}_1^T\\ \mathbf{0} & \mathbf{A}_2^T\end{bmatrix},\quad \Theta\triangleq\begin{bmatrix}\mathbf{0} & \mathbf{A}_\tau^T\\ \mathbf{A}_\tau & \mathbf{0}\end{bmatrix} \tag{56}$$

Note also that based on the definition of the $\mathbf{G}_j$ and $\mathbf{K}_j$ matrices in (39), we have:

$$\mathbf{J}^T\mathbf{J}=\mathbf{G}\mathbf{G}^T,\quad \mathbf{G}^{-1}\mathbf{Y}=\begin{bmatrix}\mathbf{0} & \mathbf{K}\end{bmatrix} \tag{57}$$

Next, we partition $\mathbf{C}^{-1}$ as:

$$\mathbf{C}^{-1}\triangleq\begin{bmatrix}\mathbf{E}_{2\times2} & \mathbf{B}_{2\times2}^T\\ \mathbf{B}_{2\times2} & \mathbf{D}_{2\times2}\end{bmatrix} \tag{58}$$

and derive the expressions of $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{B}$ in terms of $\mathbf{K}_j$, $\mathbf{G}_j$, $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, and $\mathbf{T}_{22}$. In particular, by employing the block matrix-inversion lemma on $\mathbf{C}$, $\mathbf{D}$ can be expressed as:

$$\begin{aligned}\mathbf{D}&=\left(\begin{bmatrix}\mathbf{0} & \mathbf{A}_\tau^T\\ \mathbf{A}_\tau & \mathbf{0}\end{bmatrix}-\mathbf{Y}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{Y}\right)^{-1}\\ &=\underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{A}_\tau^T\\ \mathbf{A}_\tau & -\mathbf{K}^T\mathbf{K}\end{bmatrix}^{-1}}_{\mathbf{M}}\triangleq\begin{bmatrix}\mathbf{M}_{11} & \mathbf{M}_{12}\\ \mathbf{M}_{12}^T & \mathbf{M}_{22}\end{bmatrix}\end{aligned} \tag{59}$$

where based on the block matrix-inversion lemma the elements of matrix $\mathbf{M}$ are computed as:

$$\mathbf{M}_{11} = (\mathbf{A}_\tau^T(\mathbf{K}^T\mathbf{K})^{-1}\mathbf{A}_\tau)^{-1}$$
$$\mathbf{M}_{12} = (\mathbf{A}_\tau^T(\mathbf{K}^T\mathbf{K})^{-1}\mathbf{A}_\tau)^{-1}\mathbf{A}_\tau^T(\mathbf{K}^T\mathbf{K})^{-1}$$
$$\mathbf{M}_{22} = (-\mathbf{K}^T\mathbf{K})^{-1} + (\mathbf{K}^T\mathbf{K})^{-1}\mathbf{A}_\tau(\mathbf{A}_\tau^T(\mathbf{K}^T\mathbf{K})^{-1}\mathbf{A}_\tau)^{-1}\mathbf{A}_\tau^T(\mathbf{K}^T\mathbf{K})^{-1}$$

The above expressions can be further simplified using the definitions of $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, and $\mathbf{T}_{22}$ [see (40)-(42)] as:

$$\mathbf{M}_{11} = (\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1} \tag{60}$$
$$\mathbf{M}_{12} = (\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\mathbf{T}_{21}\mathbf{T}_{11}^{-1} \tag{61}$$
$$\mathbf{M}_{22} = -(\mathbf{T}_{11}\mathbf{T}_{11}^T)^{-1} + \mathbf{T}_{11}^{-T}\mathbf{T}_{21}^T(\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\mathbf{T}_{21}\mathbf{T}_{11}^{-1} \tag{62}$$

Similar to $\mathbf{D}$, the matrices $\mathbf{E}$ and $\mathbf{B}$ are also determined by applying the block matrix-inversion lemma on $\mathbf{C}$ [see (58)], i.e.,

$$\begin{aligned}\mathbf{E} =& (\mathbf{G}\mathbf{G}^T)^{-1} + (\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{Y}\mathbf{M}\mathbf{Y}^T(\mathbf{G}\mathbf{G}^T)^{-1} \\ =& (\mathbf{G}\mathbf{G}^T)^{-1} + \begin{bmatrix}\mathbf{0} & \mathbf{G}^{-T}\mathbf{K}\end{bmatrix}\begin{bmatrix}\mathbf{M}_{11} & \mathbf{M}_{12}\\ \mathbf{M}_{12}^T & \mathbf{M}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{0}\\ \mathbf{K}^T\mathbf{G}^{-1}\end{bmatrix} \\ =& (\mathbf{G}\mathbf{G}^T)^{-1} + \mathbf{G}^{-T}\mathbf{K}\mathbf{M}_{22}\mathbf{K}^T\mathbf{G}^{-1}\end{aligned} \tag{63}$$

$$\begin{aligned}\mathbf{B} =& -\begin{bmatrix}\mathbf{M}_{11} & \mathbf{M}_{12}\\ \mathbf{M}_{12}^T & \mathbf{M}_{22}\end{bmatrix}(\mathbf{G}^{-1}\mathbf{Y})^T\mathbf{G}^{-1} = -\begin{bmatrix}\mathbf{M}_{12}\mathbf{K}^T\mathbf{G}^{-1}\\ \mathbf{M}_{22}\mathbf{K}^T\mathbf{G}^{-1}\end{bmatrix} \\ =& \begin{bmatrix}-(\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\mathbf{T}_{21}\mathbf{T}_{11}^{-1}\mathbf{K}^T\mathbf{G}^{-1}\\ -\left[\mathbf{T}_{11}^{-T}\mathbf{T}_{21}^T(\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\mathbf{T}_{21}\mathbf{T}_{11}^{-1} - (\mathbf{T}_{11}\mathbf{T}_{11}^T)^{-1}\right]\mathbf{K}^T\mathbf{G}^{-1}\end{bmatrix}\end{aligned} \tag{64}$$

With the derived $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{B}$ matrices [see (59), (63), and (64)], the covariance of the CM estimates, i.e., the top-left 3×3 block of matrix $\mathbf{C}^{-1}$ is obtained as:

$$\mathbf{C}^{-1}(1:3,1:3) = \begin{bmatrix}\mathbf{E} & \mathbf{B}(1,1)^T\\ \mathbf{B}(1,1) & \mathbf{D}(1,1)\end{bmatrix} =$$

$$\begin{bmatrix}(\mathbf{G}\mathbf{G}^T)^{-1} + \mathbf{G}^{-T}\mathbf{K}\mathbf{M}_{22}\mathbf{K}^T\mathbf{G}^{-1} & -\mathbf{G}^{-T}\mathbf{K}\mathbf{T}_{11}^{-T}\mathbf{T}_{21}^T(\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\\ -(\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\mathbf{T}_{21}\mathbf{T}_{11}^{-1}\mathbf{K}^T\mathbf{G}^{-1} & (\mathbf{T}_{22}\mathbf{T}_{22}^T)^{-1}\end{bmatrix}$$

It can be easily shown that for the case of three or more users, the expression of the CM estimates' covariance has the same structure as above, with the size of the matrices $\mathbf{G}$ and $\mathbf{K}$ increasing with the number of users.

APPENDIX D
LINE FEATURE TRIANGULATION

We hereafter present the method we employ for triangulating line features (i.e., computing the rotation and distance of the line with respect to its first observing camera pose), both for the minimal and least-squares cases. Assuming that the line is first observed by the camera frame $\{C_k\}$, we define the x and z axes of the line frame as:

$$\boldsymbol{\ell} = {}^{C_k}\mathbf{x}_L \triangleq {}^{C_k}_{L}\mathbf{C}\mathbf{e}_1 \tag{65}$$
$${}^{C_k}\mathbf{z}_L \triangleq {}^{C_k}_{L}\mathbf{C}\mathbf{e}_3 \tag{66}$$

Using these definitions and equations (9) and (10), the line constraints resulting from the observation $\mathbf{s}_{k+m}$ of a camera $\{C_{k+m}\}$ are:

$$\mathbf{s}_{k+m}^T\,{}^{C_{k+m}}_{C_k}\mathbf{C}\boldsymbol{\ell} = 0 \tag{67}$$
$$\mathbf{s}_{k+m}^T\left({}^{C_{k+m}}_{C_k}\mathbf{C}\,{}^{C_k}\mathbf{p}_L + {}^{C_{k+m}}\mathbf{p}_{C_k}\right) = 0 \tag{68}$$

where (67) and (68) constrain the line's direction, $\boldsymbol{\ell}$, and position, ${}^{C_k}\mathbf{p}_L$, respectively.

In this Appendix, for the sake of clarity, we represent all camera poses in the frame $\{C_k\}$, i.e., the first observing camera of the line.

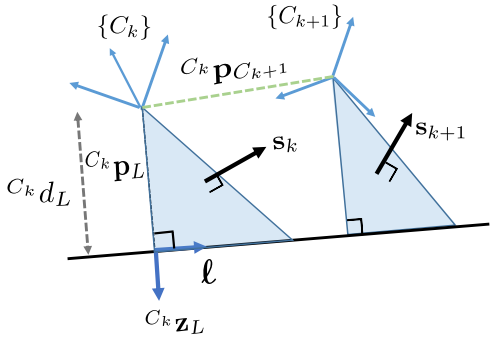


Fig. 14: Depiction of the geometric constraints between the line $\boldsymbol{\ell}$ and the camera poses from which it is observed.

A. Minimal case

First, we show how the line parameters are extracted using observations from two different known camera poses C_k and C_{k+1}. Substitution of the line direction $\boldsymbol{\ell}$ and the observing camera parameters in (67) results in:

$$\mathbf{s}_k^T\boldsymbol{\ell} = 0 \tag{69}$$
$$\mathbf{s}_{k+1}^T\,{}^{C_{k+1}}_{C_k}\mathbf{C}\boldsymbol{\ell} = 0 \tag{70}$$

Since $\boldsymbol{\ell}$ is perpendicular to both $\mathbf{s}_k$ and ${}^{C_{k+1}}_{C_k}\mathbf{C}^T\mathbf{s}_{k+1}$, the direction of the line is computed as:

$$\boldsymbol{\ell} = \gamma\lfloor\mathbf{s}_k\rfloor\,{}^{C_{k+1}}_{C_k}\mathbf{C}^T\mathbf{s}_{k+1} \tag{71}$$

where

$$\begin{aligned}\gamma^{-1} &= \left\|\lfloor\mathbf{s}_k\rfloor\,{}^{C_{k+1}}_{C_k}\mathbf{C}^T\mathbf{s}_{k+1}\right\|_2 \\ &= \sqrt{\mathbf{s}_{k+1}^T\,{}^{C_{k+1}}_{C_k}\mathbf{C}(\mathbf{I}-\mathbf{s}_k\mathbf{s}_k^T)\,{}^{C_{k+1}}_{C_k}\mathbf{C}^T\mathbf{s}_{k+1}} \\ &= \sqrt{1-(\mathbf{s}_{k+1}^T\,{}^{C_{k+1}}_{C_k}\mathbf{C}\mathbf{s}_k)^2}\end{aligned} \tag{72}$$

The remaining 2 dof of the 3D line corresponding to the distance ${}^{C_k}d_L$ to the line and the unit vector ${}^{C_k}\mathbf{z}_L$, which is normal to the line (see Fig. 14), are determined as follows:

From the geometry of the problem, we have:

$${}^{C_k}\mathbf{z}_L = \lfloor\boldsymbol{\ell}\rfloor\mathbf{s}_k \tag{73}$$

Next, substituting

$${}^{C_k}\mathbf{p}_L = {}^{C_k}d_L\,{}^{C_k}\mathbf{z}_L = {}^{C_k}d_L\lfloor\boldsymbol{\ell}\rfloor\mathbf{s}_k = -{}^{C_k}d_L\lfloor\mathbf{s}_k\rfloor\boldsymbol{\ell} \tag{74}$$

in (68) for $m=1$, i.e.,

$$\mathbf{s}_{k+1}^T\left(-{}^{C_k}d_L\,{}^{C_{k+1}}_{C_k}\mathbf{C}\lfloor\mathbf{s}_k\rfloor\boldsymbol{\ell} + {}^{C_{k+1}}\mathbf{p}_{C_k}\right) = 0 \tag{75}$$

yields[11]

$${}^{C_k}d_L = \frac{\mathbf{s}_{k+1}^T\,{}^{C_{k+1}}\mathbf{p}_{C_k}}{\mathbf{s}_{k+1}^T\,{}^{C_{k+1}}_{C_k}\mathbf{C}\lfloor\mathbf{s}_k\rfloor\boldsymbol{\ell}} = -\gamma\,\mathbf{s}_{k+1}^T\,{}^{C_{k+1}}\mathbf{p}_{C_k} \tag{76}$$

Finally, the triangulated line parameter vector is $\left[{}^{C_k}\mathbf{q}_L^T \quad {}^{C_k}d_L\right]^T$, where ${}^{C_k}\mathbf{q}_L$ is the quaternion representation of the rotation matrix ${}^{C_k}_{L}\mathbf{C} = \left[\boldsymbol{\ell} \quad \mathbf{s}_k \quad {}^{C_k}\mathbf{z}_L\right]$.

[11] Please note that we choose the direction of $\boldsymbol{\ell}$ such that ${}^{C_k}d_L > 0$.

B. Least-squares case

Given $K > 1$ observations of the line normals from known camera poses, the line parameters are computed by minimizing the sum of squares of the re-projection errors corresponding to (67) and (68). Specifically, we first seek to find the unit vector $\boldsymbol{\ell}$ that minimizes the cost function

$$\mathscr{C}_{\boldsymbol{\ell}} = \sum_{m=0}^{K-1} ||\mathbf{s}_{k+m}^T {}^{C_{k+m}}_{C_k}\mathbf{C}\boldsymbol{\ell}||_2^2 \tag{77}$$

$$= \boldsymbol{\ell}^T \left(\sum_{m=0}^{K-1} {}^{C_{k+m}}_{C_k}\mathbf{C}^T \mathbf{s}_{k+m}\mathbf{s}_{k+m}^T {}^{C_{k+m}}_{C_k}\mathbf{C} \right) \boldsymbol{\ell} \tag{78}$$

$$= \boldsymbol{\ell}^T \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \boldsymbol{\ell} \tag{79}$$

As evident, (79) is minimized by selecting as $\boldsymbol{\ell}$ the eigenvector corresponding to the minimum eigenvalue of the symmetric positive definite matrix $\boldsymbol{\Gamma}^T\boldsymbol{\Gamma}$ where

$$\boldsymbol{\Gamma} = \begin{bmatrix} \mathbf{s}_k^T \\ \mathbf{s}_{k+1}^T {}^{C_{k+1}}_{C_k}\mathbf{C} \\ \vdots \\ \mathbf{s}_{k+K-1}^T {}^{C_{k+K-1}}_{C_k}\mathbf{C} \end{bmatrix}. \tag{80}$$

Next, given the optimum value $\boldsymbol{\ell}^*$ of (79), we seek to find ${}^{C_k}\mathbf{p}_L = {}^{C_k}d_L {}^{C_k}\mathbf{z}_L$ that minimizes the cost function

$$\mathscr{C}_L = \frac{1}{2} \sum_{m=0}^{K-1} ||\mathbf{s}_{k+m}^T \left({}^{C_{k+m}}_{C_k}\mathbf{C} {}^{C_k}\mathbf{p}_L + {}^{C_{k+m}}\mathbf{p}_{C_k} \right)||_2^2 \tag{81}$$

$$= \frac{1}{2} ||\boldsymbol{\Gamma} {}^{C_k}\mathbf{p}_L - \boldsymbol{\xi}||^2 \tag{82}$$

$$\text{s.t.} \quad \boldsymbol{\ell}^{*T} {}^{C_k}\mathbf{p}_L = 0 \tag{83}$$

where

$$\boldsymbol{\xi} = - \begin{bmatrix} 0 \\ \mathbf{s}_{k+1}^T {}^{C_{k+1}}\mathbf{p}_{C_k} \\ \vdots \\ \mathbf{s}_{k+K-1}^T {}^{C_{k+K-1}}\mathbf{p}_{C_k} \end{bmatrix}. \tag{84}$$

To do so, we employ the Lagrange multiplier method [61], which results in the following system of linear equations:

$$\begin{bmatrix} \boldsymbol{\Gamma}^T\boldsymbol{\Gamma} & \boldsymbol{\ell}^* \\ \boldsymbol{\ell}^{*T} & 0 \end{bmatrix} \begin{bmatrix} {}^{C_k}\mathbf{p}_L \\ \lambda \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Gamma}^T\boldsymbol{\xi} \\ 0 \end{bmatrix} \tag{85}$$

Once ${}^{C_k}\mathbf{p}_L$ is found from (85), ${}^{C_k}\mathbf{z}_L$ and ${}^{C_k}d_L$ are computed as ${}^{C_k}d_L = ||{}^{C_k}\mathbf{p}_L||_2$, ${}^{C_k}\mathbf{z}_L = \frac{1}{{}^{C_k}d_L} {}^{C_k}\mathbf{p}_L$.

NOMENCLATURE

j	User index
i	Feature index
$\{G\}$	Global frame
$\{C_\eta\}$	Frame of the camera pose first observing a feature
$\{C_k\}$	Frame of the camera pose taking a feature measurement
$\{G_j\}$	Global frame of user j
$\{C_{\eta_j}\}$	Frame of camera pose of user j first observing a feature
$\{C_{k_j}\}$	Frame of camera pose of user j taking a feature measurement
$\{B\}$	Manhattan-world frame
$\{L_i\}$	Frame of free line $\boldsymbol{\ell}_i$
$\{V_i\}$	Frame of Manhattan line $\mathbf{v}_i$
$\{L_{i_j}\}$	Frame of free line $\boldsymbol{\ell}_i$ defined by user j
$\{V_{i_j}\}$	Frame of Manhattan line $\mathbf{v}_i$ defined by user j
${}^2\mathbf{p}_1$	Position of frame $\{1\}$ in frame $\{2\}$
${}^2_1\mathbf{C}$	Orientation of frame $\{1\}$ in frame $\{2\}$
${}^j\mathbf{x}_u$	State vector of user j
$\mathbf{x}_\tau$	Pairwise transformations between users' frames
${}^j\check{\mathbf{p}}_k$	Pose, velocity, and IMU biases of user j at step k
${}^j\check{\mathbf{p}}$	All poses, velocities, and IMU biases of user j
${}^j\mathbf{f}$	Features observed by user j
${}^j\mathbf{q}_B$	Orientation of user j in the Manhattan world
$\mathbf{x}_a$	All users' poses, velocities, and IMU biases
$\mathbf{f}_c$	All common features
$\mathbf{f}_a$	All features observed by only one user
$\mathbf{f}_{c_j}$	Common features observed by user j

ACKNOWLEDGMENT

This work was supported by Google LLC, Project Tango, and performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-JRNL-751808).

REFERENCES

[1] E. D. Nerurkar, K. J. Wu, and S. I. Roumeliotis, "C-KLAM: Constrained keyframe-based localization and mapping," in *Proc. of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, May 31 – Jun. 7 2014, pp. 3638–3643.

[2] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Proc. of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9–13 2011, pp. 3281–3288.

[3] D. Strelow and S. Singh, "Motion estimation from image and inertial measurements," *International Journal of Robotics Research*, vol. 23, no. 12, pp. 1157–1195, Dec. 2004.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 6, pp. 314–334, Dec. 2015.

[5] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation," in *Proc. of the International Conference on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 27–31 2003, pp. 212–217.

[6] T. Cieslewski, S. Lynen, M. Dymczyk, S. Magnenat, and R. Siegwart, "Map API - scalable decentralized map building for robots," in *Proc. of the IEEE International Conference on Robotics and Automation*, Seattle, WA, May 26–30 2015, pp. 6241–6247.

[7] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *Proc. of the IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 3–8 2010, pp. 3185–3192.

[8] X. S. Zhou and S. I. Roumeliotis, "Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case," in *Proc. of the International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9–15 2006, pp. 1785–1792.

[9] L. A. Andersson and J. Nygards, "C-SAM: Multi-robot SLAM using square root information smoothing," in *Proc. of the IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 19–23 2008, pp. 2798–2805.

[10] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using constrained factor graphs," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 18–22 2010, pp. 3025–3030.

[11] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6–10 2013, pp. 5220–5227.

[12] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov. 13–16 2007, pp. 225–234.

[13] L. Riazuelo, J. Civera, and J. M. M. Montiel, "C2TAM: A cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, pp. 401–413, Apr. 2014.

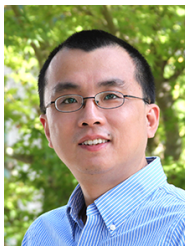
[14] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3D mapping in real-time with low-cost robots," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, Apr. 2015.
[15] C. Forster, S. Lynen, L. Kneip, and D. Scaramuza, "Collaborative monocular SLAM with multiple micro aerial vehicles," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 3 – 7 2013, pp. 3963–3970.
[16] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. on Robotics*, vol. 30, no. 1, pp. 158–176, Feb. 2014.
[17] M. Bosse, R. Rikoski, J. Leonard, and S. Teller, "Vanishing points and 3D lines from omnidirectional video," in *Proc. of the IEEE International Conference on Image Processing*, Rochester, New York, Sep. 22–25 2002, pp. 513–516.
[18] D. G. Kottas and S. I. Roumeliotis, "Exploiting urban scenes for vision-aided inertial navigation," in *Robotics: Science and Systems Conference*, Berlin, Germany, Jun. 24–28 2013.
[19] F. Camposeco and M. Pollefeys, "Using vanishing points to improve visual-inertial odometry," in *Proc. of the IEEE International Conference on Robotics and Automation*, Seattle, Washington, May 26–30 2015, pp. 5219–5225.
[20] R. Kawanishi, A. Yamashita, T. Kaneko, and H. Asama, "Parallel line-based structure from motion by using omnidirectional camera in textureless scene," *Advanced Robotics*, vol. 27, no. 1, pp. 19–23, 2013.
[21] J. H. Lee, G. Zhang, J. Lim, and I. H. Suh, "Place recognition using straight lines for vision-based SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6–10 2013, pp. 3799–3806.
[22] G. Zhang, D. H. Kang, and I. H. Suh, "Loop closure through vanishing points in a line-based monocular SLAM," in *Proc. of the IEEE International Conference on Robotics and Automation*, Saint Paul, Minnesota, May 14–18 2012, pp. 4565–4570.
[23] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithm, and Software*. New York, NY: John Wiley and Sons, 2004.
[24] [Online]. Available: http://onionmaps.info
[25] C. X. Guo, K. Sartipi, R. C. DuToit, G. A. Georgiou, R. Li, J. O'Leary, E. D. Nerurkar, J. A. Hesch, and S. I. Roumeliotis, "Large-scale cooperative 3D visual-inertial mapping in a manhattan world," in *Proc. of the IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, May 16–21 2016.
[26] A. B. Chatfield, *Fundamentals of High Accuracy Inertial Navigation*. American Institute of Aeronautics and Astronautics, 1997, vol. 174.
[27] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep., March 2005.
[28] R. O. Allen and D. H. Change, "Performance testing of the systron donner quartz gyro (qrs11-100-420); sn's 3332, 3347 and 3544," JPL, Tech. Rep., 1993.
[29] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, Oct. 2008.
[30] D. G. Kottas and S. I. Roumeliotis, "Efficient and consistent vision-aided inertial navigation using line observations," in *Proc. of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6 – 10 2013, pp. 1532 – 1539.
[31] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate," *ACM Transactions on Mathematical Software*, vol. 35, no. 3, pp. 22:1–22:14, Oct. 2008.
[32] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.
[33] W. Gander, "Least squares with a quadratic constraint," *Numerische Mathematik*, vol. 36, pp. 291–307, 1981.
[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press Cambridge, 2001.
[35] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.
[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[37] J. R. Bunch and B. N. Parlett, "Direct methods for solving symmetric indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 8, no. 4, pp. 639–655, Dec. 1971.
[38] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
[39] Google, "Project Tango," https://www.google.com/atap/projecttango/.
[40] [Online]. Available: https://www.threadingbuildingblocks.org/
[41] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.
[42] [Online]. Available: http://faculty.cse.tamu.edu/davis/suitesparse.html
[43] [Online]. Available: http://ceres-solver.org/
[44] T. A. Davis, *Direct Methods for Sparse Linear Systems*. SIAM, 2006.
[45] [Online]. Available: http://www.openblas.net/
[46] [Online]. Available: https://www.tacc.utexas.edu/research-development/tacc-software/gotoblas2
[47] [Online]. Available: https://software.intel.com/en-us/node/520724
[48] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 10–14 2007, pp. 3482–3489.
[49] C. X. Guo, D. G. Kottas, R. C. DuToit, A. Ahmed, R. Li, and S. I. Roumeliotis, "Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps," in *Proc. of Robotics: Science and Systems*, Berkeley, California, Jul. 12-16 2014.
[50] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of the Alvey Vision Conference*, Manchester, UK, Aug. 31 – Sep. 2 1988, pp. 147–151.
[51] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the International Joint Conference on Artificaial Intelligence*, Vancouver, British Columbia, Aug. 24–28 1981, pp. 674–679.
[52] L. Kneip, M. Chli, and R. Siegwart, "Robust real-time visual odometry with a single camera and an IMU," in *Proc. of The British Machine Vision Conference*, Dundee, Scotland, Aug. 2011, pp. 1–11.
[53] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. of the IEEE International Conference on Computer Vision*, Barcelona, Spain, Nov. 6–13 2011, pp. 2564–2571.
[54] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, New York, NY, Jun. 17 – 22 2006, pp. 2161–2168.
[55] O. Naroditsky, X. S. Zhou, S. I. Roumeliotis, and K. Daniilidis, "Two efficient solutions for visual odometry using directional correspondence," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 818–824, Apr. 2012.
[56] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a line segment detector," *Image Processing On Line*, vol. 2, pp. 35–55, Mar. 2012.
[57] F. M. Mirzaei and S. I. Roumeliotis, "Optimal estimation of vanishing points in a manhattan world," in *Proc. of the IEEE Internatiional Conference on Computer Vision*, Barcelona, Spain, Nov. 6 – 12 2011, pp. 2452 – 2461.
[58] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9–13 2011, pp. 3400–3407.
[59] J. A. Hesch and S. I. Roumeliotis, "A direct least-squares (DLS) method for PnP," in *Proc. of the IEEE Internatiional Conference on Computer Vision*, Barcelona, Spain, Nov. 6–13 2011, pp. 383–390.
[60] E. Kostina and O. Kostyukova, "Computing covariance matrices for constrained nonlinear large scale parameter estimation problems using Krylov subspace methods," *International Series of Numerical Mathematics*, vol. 1, pp. 197–212, 2012.
[61] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

Chao X. Guo received the B.Eng. and M.Eng. in Information Engineering from the Zhejiang University, China, in 2007 and 2010, respectively, and the M.S. and Ph.D. in Computer Science from the University of Minnesota, MN, USA, in 2016. Since then, he has been working on motion tracking, mapping, and sensor calibration technologies at Google LLC for Project Tango (3D sensing and perception on smartphones), ARCore (Android AR platform), and WorldSense (Daydream VR platform).

Kourosh Sartipi received the B.S. degree in Physics in 2011 from Isfahan University of Technology, Isfahan, Iran and M.S. in Computer Science from Sharif University of Technology, Tehran, Iran in 2013. He is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests include localization, mapping and sensor fusion.

Ryan C. DuToit received the B.S. degree in computer science, physics, and mathematics in 2013 from the College of Arts and Sciences, Drake University, Des Moines, IA, USA and the M.S. degree in computer science in 2016 from the College of Science and Engineering, University of Minnesota, MN, USA, where he is working toward the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include localization and navigation on visual-inertial systems, as well as enabling augmented and virtual reality experiences on mobile devices at a massive scale.

Georgios A. Georgiou received the B.S. in Computer Engineering from University of Cyprus, Cyprus, in 2012 and the M.S. degree in Computer Science and Engineering from the University of Minnesota, Twin Cities, in 2016. He is currently a Senior Software Engineer at GM Cruise LLC in San Francisco, CA, focusing on vehicle localization, calibration, and large-scale mapping. His research interests include long-term autonomy of ground vehicles, sensor fusion, and numerical linear algebra and optimization.



Ruipeng Li received the B.S. degree in Computer Science from Beijing University of Posts and Telecommunications, Beijing, China, in 2006, the M.S. degree in Computer Science from Arkansas State University, AR, USA, in 2008, and the Ph.D. degree in Computer Science from University of Minnesota, MN, USA, in 2015. From 2015 to 2018, he was a postdoctoral researcher at the Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, where he is currently a computer scientist. His research interests include sparse matrix computations, parallel computing, iterative methods for solving linear systems and eigenvalue problems, and multilevel preconditioning techniques.

John O'Leary received a Bachelor's in Computer Science in 2016 from the College of Science and Engineering, University of Minnesota (UMN), Minneapolis, MN, USA. His focus was in human-computer interaction and data systems management. Mr. O'Leary received the Barry Goldwater Scholarship in 2015.

Esha D. Nerurkar received the B.Eng. degree in electronics engineering in 2004 from the University of Mumbai, India. She received her M.S. degree in computer science in 2010, and Ph.D. degree in computer science in 2016 from the College of Science and Engineering, University of Minnesota (UMN), Minneapolis, MN, USA. In 2013, she joined Google and is currently the technical lead for the mobile perception group that ships motion tracking technologies for Project Tango-enabled phones, smartphone AR for Android (ARCore), and head tracking for Daydream VR headsets (WorldSense).

Dr. Nerurkar received the 2012 University of Minnesota Doctoral Dissertation Fellowship. She was the finalist for the Best Student Paper Awards for ICRA 2009 and IROS 2013, and the 2010 Google Anita Borg scholarship.

Joel A. Hesch received the B.Eng. degree in computer engineering in 2004, the M.S. degree in computer science in 2008, and the Ph.D. degree in computer science in 2016 from the College of Science and Engineering, University of Minnesota (UMN), Minneapolis, MN, USA. In 2012 he moved to Google, where he was a founding member of Project Tango - developing and shipping perception technologies for the world's first depth-enabled smartphones (Tango), the Android AR platform (ARCore), and the Daydream VR platform (WorldSense). In 2017 he moved to Facebook, where he leads a perception group.

Dr. Hesch received the 2011 UMN Doctoral Dissertation Fellowship. In 2010, he was selected as a Honeywell International Innovator Scholar. He also received the National Institutes of Health Neuro-physical-computational Sciences Training Fellowship (2007-2009) in Computational Neuroscience. In addition, he received the 2007, CSE Excellence in Research Award.

Stergios I. Roumeliotis (IEEE Fellow 2016) received the Diploma in Electrical Engineering from the National Technical University of Athens, Greece, in 1995, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Southern California, CA in 1999 and 2000 respectively. From 2000 to 2002 he was a Postdoctoral Fellow at the California Institute of Technology, CA. Between 2002 and 2013 he was first an Assistant and then an Associate Professor with the Department of Computer Science and Engineering, University of Minnesota (UMN), MN, where he is currently a Professor and affiliate member of the Electrical Engineering and Mechanical Engineering departments. Between 2009 and 2015 S.I. Roumeliotis was the Associate Director for Research of the Digital Technology Center (DTC) at UMN, while since Jan. 2017, he is the Chief Scientist for Visual and Inertial Systems at Google Daydream. His research interests include visual-inertial state and environment estimation and representation for autonomous vehicles (space, aerial, ground, underwater) and mobile devices with special emphasis on augmented reality (AR) applications.

S. I. Roumeliotis is the recipient of the Guillermo E. Borja Award (2009), the National Science Foundation (NSF) Presidential Early Career Award for Scientists and Engineers (PECASE) (2008), the NSF CAREER award (2006), the McKnight Land-Grant Professorship award (2006-08), the ICRA Best Reviewer Award (2006), and he is the co-recipient of the One NASA Peer award (2006), and the One NASA Center Best award (2006). Papers he has co-authored have received the King-Sun Fu Best Paper Award of the IEEE Transactions on Robotics (2009), the Robotics Society of Japan Best Journal Paper award (2007), the ICASSP Best Student Paper award (2006), the NASA Tech Briefs award (2004), and five of them were Finalists for the IROS Best Student Paper Award (2013), the RSS Best Paper Award (2009), the ICRA Best Student Paper Award (2009), the IROS Best Paper Award (2006), and the ICRA Best Vision Paper Award (2017). S.I. Roumeliotis served as Associate Editor for the IEEE Transactions on Robotics between 2006 and 2010.