Large-Scale Cooperative 3D Visual-Inertial Mapping in a Manhattan World

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 combined in a single map by imposing geometric constraints between commonly-observed point and line features. Furthermore, our formulation allows for modularity since new/old maps (or parts of them) can be easily added/removed with no impact on the remaining ones. Additionally, the proposed CM algorithm lends itself, for the most part, to parallel implementations, hence gaining in speed. Experimental results based on visual and inertial measurements collected from four users within two large buildings are used to assess the performance of the proposed CM algorithm.**

I. Introduction

A robust and tractable solution to the large-scale 3D mapping problem has many useful applications, such as human (or robot) indoor navigation, augmented reality, and search and rescue. Besides vision-only approaches, visual and inertial (rotational velocity and linear acceleration) measurements have also been used to create 3D maps (e.g., [1]), though the emphasis is on how to efficiently process a single dataset corresponding to all, or part, of an area of interest. In many practical applications, however, the device used for recording data (e.g., a cell phone or a wearable computer) may not have sufficient resources (e.g., storage space or battery) to collect data from a large area. Additionally, it may not be convenient for a single user to navigate the entire building at once. Furthermore, since existing algorithms [e.g., batch least-squares (BLS)] focus on creating a map out of *a single* dataset, every time a part of the map changes, or is deemed of insufficient quality or accuracy, the whole mapping process needs to be repeated.

The aforementioned limitations motivate investigating cooperative mapping (CM) methods which 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 poses (position and orientation) is not known.

Early work [2], as well as more recent cloud-based approaches (e.g., [3]), on CM focus on aligning the users' individual maps but without estimating the resulting combined map. In [4] and [5], all users' poses and maps are optimized by employing approximate algorithms such as the pose graph [6] and PTAM [7], respectively. On the other hand, [8] and [9] propose a distributed CM algorithm, the DDF-SAM, which creates constraints between multiple-user trajectories based on commonly-observed features. Specifically, each user summarizes its trajectory and non-common features by marginalizing them, and sends the resulting inferred measurements, relating *only* the commonly-observed features, to its neighbors. Then, every user optimizes its own trajectory and map by combining the received inferred measurements from its neighbors with its local measurements. Note that this is an approximate method, as the received inferred measurements cannot be relinearized when a better estimate is obtained through BLS iterations.

Furthermore, most works to date on localization and mapping have focused on processing only *point* features that are tracked through sequences of images. *Line* features, especially those aligned with the cardinal directions of a "Manhattan world", provide additional attitude information. In [10] and [11], both point and line features are used for improving the localization and mapping accuracy of filtering algorithms. In the context of BLS, [12] simultaneously estimates the camera's 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, the proposed solution decouples the camera motion into two rotations and two translations, then solves for each of them successively, rendering it sub-optimal. When lines are used for localization, line-loop-closure measurements are rarely used in practice, with the exception of [13]. Specifically, Zhang *et al.* [13] propose a method for detecting loop-closure line measurements, but under the assumption that the observed lines either reside within, or are perpendicular to, the plane corresponding to the floor.

Our proposed algorithm introduces an efficient, high-accuracy BLS solution that utilizes both consecutive and loop-closure observations of point and line features. In particular, the main contributions of this paper are:

- We formulate CM as a constrained optimization problem that is modular (i.e., maps or submaps can be added or removed) and lends itself to parallel implementation. In addition, the proposed algorithm is able to leverage each individual user's intermediate mapping results to reduce the processing cost.
- We provide a BLS solution utilizing points, "free lines" (lines not aligned with the cardinal directions of the building), and "Manhattan lines" to improve the estimation accuracy. Additionally, we provide a robust method

This work was supported by Google ATAP, Project Tango.

for detecting loop-closure line measurements.

- We validate our algorithm in two 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 [14].

The rest of the paper is structured as follows: In Section II, we define the CM problem and present an overview of our approach. In Section III, we provide the parameterization and measurement models for point, free-line, and Manhattan-line features. In Section IV, we present the geometric constraint that arises when two features, defined in two separate maps, correspond the same physical point. In Section 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 in detail our CM algorithm. In Section VI, we thoroughly evaluate the proposed method, both in terms of accuracy and computational complexity, and provide concluding remarks in Section VII.

II. Algorithm Overview

Consider multiple datasets consisting of visual and inertial measurements collected by several users with a camera and an inertial measurement unit (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 paper 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 the constraints that arise from commonly-observed point and line features.

Our formulation of the CM problem has two 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 add or remove 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 a substantial speed up.

III. System State and Measurement Models

In this section, we first describe the cost function introduced by the IMU measurements, and then present the feature parameterization and measurement model for processing

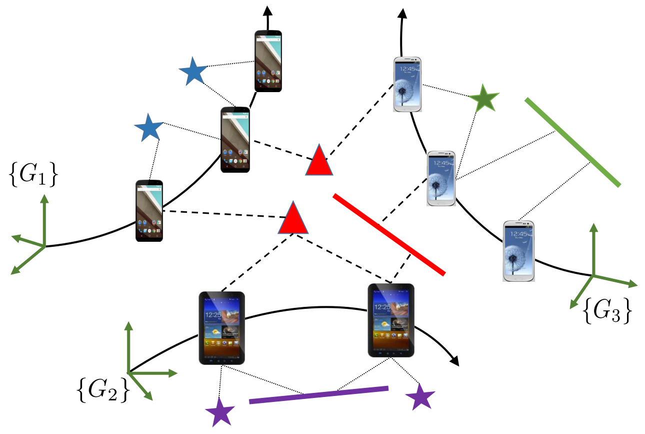

Fig. 1. Non-common (stars) and common (triangles) point features, as well as line features observed by one or more mobile devices.

point, free-line, and Manhattan-line features. Note that our system state includes the trajectory and map of each user, the transformation between users, and the orientation between the user and the Manhattan world frames.

In the rest 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. 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.

A. IMU state and measurement model

The pose (position and attitude quaternion) and velocity of the IMU-camera pair,[1] as well as the IMU biases at time step $k+1$, denoted by the 16×1 vector $\mathbf{p}_{k+1}$, can be computed from $\mathbf{p}_k$ by integrating the IMU's rotational velocity and linear acceleration measurements, $\mathbf{u}_k$. This process is described by the following equation [15], [16]:

$$\mathbf{p}_{k+1} = \mathbf{g}(\mathbf{p}_k, \mathbf{u}_k) + \mathbf{w}_k \qquad (1)$$

where $\mathbf{g}$ is the nonlinear function corresponding to the IMU measurement model and $\mathbf{w}_k$ is the IMU measurement noise, which is assumed to be zero mean, Gaussian with covariance $\mathbf{Q}_k$, computed through IMU characterization [17].

B. Point feature state and measurement model

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

$${}^{C_k}\mathbf{z} = \pi\left({}^{C_k}\mathbf{p}_{C_j} + {}^{C_k}_{C_j}\mathbf{C}\,{}^{C_j}\mathbf{x}_{P_i}\right) + \mathbf{n}_k \qquad (2)$$

where π represents the camera perspective-projection model and $\mathbf{n}_k$ is the measurement noise.

[1]To simplify the ensuing derivations, we assume the IMU and camera are co-located. In our experiments, we include the IMU-camera extrinsic calibration parameters (6 degrees of freedom transformations) of each user in the BLS problem formulation and estimate them concurrently with the trajectories and maps of the users.

C. Free-line feature state and measurement model

In this paper, we use the same 4 degrees of freedom (dof) free-line parameterization as in [11]. Consider the line $\boldsymbol{\ell}_i$ in Fig. 2 which is first observed by camera $\{C_j\}$. 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 $\{C_j\}$, x-axis is aligned with the line's direction $\boldsymbol{\ell}_i$, and z-axis points away from the origin of $\{C_j\}$. Then, the line is represented with respect to $\{C_j\}$ by the parameter vector ${}^{C_j}\mathbf{x}_{L_i} = \begin{bmatrix} {}^{C_j}\mathbf{q}_{L_i}^T & d_{L_i} \end{bmatrix}^T$. Defining ${}^{C_j}_{L_i}\mathbf{C} \triangleq \mathbf{C}({}^{C_j}\mathbf{q}_{L_i})$, the origin of the line frame in $\{C_j\}$ can be written as ${}^{C_j}\mathbf{p}_{L_i} = d_{L_i}{}^{C_j}_{L_i}\mathbf{C}\mathbf{e}_3$.

In the absence of noise, a line measurement $\mathbf{s}_k$ in $\{C_k\}$, which is defined as 2 dof unit vector perpendicular to the plane spanned by 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.,

$$\mathbf{s}_k^T {}^{C_k}_{C_j}\mathbf{C}{}^{C_j}_{L_i}\mathbf{C}\mathbf{e}_1 = 0 \tag{3}$$

$$\mathbf{s}_k^T \left({}^{C_k}_{C_j}\mathbf{C}{}^{C_j}\mathbf{p}_{L_i} + {}^{C_k}\mathbf{p}_{C_j} \right) = 0 \tag{4}$$

In the presence of noise, the measured normal vector is $\mathbf{s}_k' = \mathbf{C}\left(\mathbf{s}_k^{\perp}, n_1\right)\mathbf{C}\left(\mathbf{s}_k^{\perp\!\!\perp}, n_2\right)\mathbf{s}_k$, 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 axes $\mathbf{s}_k^{\perp}$ and $\mathbf{s}_k^{\perp\!\!\perp}$ perpendicular to it by angles of magnitude n_1 and n_2, respectively.

D. 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 $\{C_j\}$ is represented by the parameter vector ${}^{C_j}\mathbf{x}_{V_i} = \begin{bmatrix} \theta_{V_i} & d_{V_i} \end{bmatrix}^T$, where θ_{V_i} is the angle between ${}^{C_j}\mathbf{p}_{V_i}$ and the y-axis of the Manhattan world (i.e., $\mathbf{e}_2$), and d_{V_i} is the distance between the origins of $\{C_j\}$ and the Manhattan-line's frame $\{V_i\}$. Using this parameterization, ${}^{C_j}\mathbf{p}_{V_i}$ is expressed as:

$${}^{C_j}\mathbf{p}_{V_i} = d_{V_i}{}^{C_j}_{G}\mathbf{C}{}^{G}_{B}\mathbf{C}(\cos\theta_{V_i}\mathbf{e}_2 + \sin\theta_{V_i}\mathbf{e}_3) \tag{5}$$

where ${}^{G}_{B}\mathbf{C}$ is the rotation matrix expressing the orientation of the Manhattan world frame with respect to the global frame. Similar to (3) and (4), the geometric constraints corresponding to Manhattan lines are:

$$\mathbf{s}_k^T {}^{C_k}_{C_j}\mathbf{C}{}^{C_j}_{G}\mathbf{C}{}^{G}_{B}\mathbf{C}\mathbf{e}_1 = 0 \tag{6}$$

$$\mathbf{s}_k^T \left({}^{C_k}_{C_j}\mathbf{C}{}^{C_j}\mathbf{p}_{V_i} + {}^{C_k}\mathbf{p}_{C_j} \right) = 0 \tag{7}$$

IV. Common-Feature Constraints

In our problem formulation, if a feature is observed by multiple users, we define it as a different one in each map but 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 constraint for common point, free-line, and Manhattan-line features.

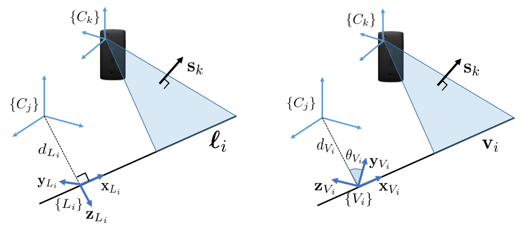

Fig. 2. The parameterization and measurement of free lines (left) and Manhattan lines (right).

A. Point-feature constraint

Consider a point feature $\mathbf{x}_{P_i}$ observed by users $\{G_1\}$ and $\{G_2\}$, and expressed as ${}^{C_{j1}}\mathbf{x}_{P_i}$ and ${}^{C_{j2}}\mathbf{x}_{P_i}$ with respect to the first observing camera poses in the two users' maps. The geometric constraint between them is:

$${}^{C_{j1}}\mathbf{x}_{P_i} - {}^{C_{j1}}_{C_{j2}}\mathbf{C}{}^{C_{j2}}\mathbf{x}_{P_i} - {}^{C_{j1}}\mathbf{p}_{C_{j2}} = \mathbf{0} \tag{8}$$

where ${}^{C_{j1}}_{C_{j2}}\mathbf{C}$ and ${}^{C_{j1}}\mathbf{p}_{C_{j2}}$ are then expressed using the local camera poses and the transformation between the two maps:

$${}^{C_{j1}}_{C_{j2}}\mathbf{C} = {}^{C_{j1}}_{G_1}\mathbf{C}{}^{G_1}_{G_2}\mathbf{C}{}^{C_{j2}}_{G_2}\mathbf{C}^T \tag{9}$$

$${}^{C_{j1}}\mathbf{p}_{C_{j2}} = {}^{C_{j1}}_{G_1}\mathbf{C}\left({}^{G_1}\mathbf{p}_{G_2} - {}^{G_1}\mathbf{p}_{C_{j1}} + {}^{G_1}_{G_2}\mathbf{C}{}^{G_2}\mathbf{p}_{C_{j2}} \right) \tag{10}$$

B. Free-line-feature constraint

Consider a free-line $\boldsymbol{\ell}_i$ observed by two users, and expressed with respect to the first observing camera poses $\{C_{j_1}\}$ and $\{C_{j_2}\}$ 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 $\{C_{j_1}\}$, $\{L_{i_1}\}$, $\{C_{j_2}\}$, and $\{L_{i_2}\}$:

$${}^{L_{i_2}}\mathbf{p}_{L_{i_1}} = {}^{C_{j2}}_{L_{i_2}}\mathbf{C}^T \left[{}^{C_{j1}}_{C_{j2}}\mathbf{C}^T ({}^{C_{j1}}\mathbf{p}_{L_{i_1}} - {}^{C_{j1}}\mathbf{p}_{C_{j_2}}) - {}^{C_{j2}}\mathbf{p}_{L_{i_2}} \right] \tag{11}$$

where ${}^{L_{i_2}}\mathbf{p}_{L_{i_1}} = -d_c\mathbf{e}_1$. To remove d_c from (11), we define $\mathbf{E}_{23} \triangleq \begin{bmatrix} \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}^T$ and multiply with it both sides of (11) to obtain the 2 dof constraint:

$$\mathbf{E}_{23}{}^{C_{j2}}_{L_{i_2}}\mathbf{C}^T \left({}^{C_{j1}}_{C_{j2}}\mathbf{C}^T ({}^{C_{j1}}\mathbf{p}_{L_{i_1}} - {}^{C_{j1}}\mathbf{p}_{C_{j_2}}) - {}^{C_{j2}}\mathbf{p}_{L_{i_2}} \right) = \mathbf{0} \tag{12}$$

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:

$$\mathbf{E}_{23}\left({}^{C_{j1}}_{L_{i_1}}\mathbf{C}\mathbf{e}_1 - {}^{C_{j1}}_{C_{j_2}}\mathbf{C}{}^{C_{j2}}_{L_{i_2}}\mathbf{C}\mathbf{e}_1 \right) = \mathbf{0} \tag{13}$$

C. Manhattan-line-feature constraint

The common Manhattan-line features also satisfy (11), 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:

$${}^{C_{j2}}_{V_{i_2}}\mathbf{C}{}^{V_{i_2}}\mathbf{p}_{V_{i_1}} = -d_c{}^{C_{j2}}_{B}\mathbf{C}\mathbf{e}_1 \tag{14}$$

Then, similar to (12), the 2 dof translational common Manhattan line constraint can be written as:

$$\mathbf{E}_{23}{}^{C_{j2}}_{B}\mathbf{C}^T \left({}^{C_{j1}}_{C_{j2}}\mathbf{C}^T ({}^{C_{j1}}\mathbf{p}_{V_{i_1}} - {}^{C_{j1}}\mathbf{p}_{C_{j_2}}) - {}^{C_{j2}}\mathbf{p}_{V_{i_2}} \right) = \mathbf{0} \tag{15}$$

Note also that since the Manhattan lines align with the building's cardinal directions, the orientation constraint corresponding to (13) is automatically satisfied.

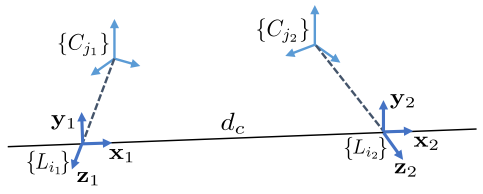

Fig. 3. Depiction of line constraints.

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 (Section V-B) describe our approach to find an initial estimate for the relative poses between users. Subsequently, we introduce our CM algorithm in Section V-C.

A. Single-user batch least-squares

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

$$\mathscr{C}_j = ||\bar{\mathbf{p}}_j - \mathbf{g}(\bar{\mathbf{p}}_j, \mathbf{u}_j)||^2_{\mathbf{Q}_j} + \left|\left|\mathbf{z}_j - \mathbf{h}(\bar{\mathbf{p}}_j, \mathbf{f}_j, {}^{G_j}\mathbf{q}_B)\right|\right|^2_{\mathbf{R}_j} \tag{16}$$

where the first term corresponds to the cost function arising from IMU measurements [see (1)], while the second one is due to visual observations of point [see (2)], free-line [see (3)-(4)], and Manhattan-line [see (6)-(7)] features. Also in (16), $\bar{\mathbf{p}}_j$ denotes the user's poses, $\mathbf{f}_j$ is the vector of all features, ${}^{G_j}\mathbf{q}_B$ is the quaternion representing the orientation between the user and Manhattan world frames, $\mathbf{u}_j$ and $\mathbf{z}_j$ comprises all IMU and visual measurements, while $\mathbf{Q}_j$ and $\mathbf{R}_j$ are the covariance matrices describing their measurement noises.

The cost function (16) can be minimized by employing Gauss-Newton iterative minimization. In particular, by expressing the error states of $\bar{\mathbf{p}}_j$, $\mathbf{f}_j$, and ${}^{G_j}\mathbf{q}_B$ with $\tilde{\mathbf{p}}_j$, $\tilde{\mathbf{f}}_j$, and ${}^{G_j}\tilde{\theta}_B$, respectively, and defining $\delta\mathbf{x}_j \triangleq \begin{bmatrix}\tilde{\mathbf{p}}_j^T & \tilde{\mathbf{f}}_j^T & {}^{G_j}\tilde{\theta}_B^T\end{bmatrix}^T$, we have the linearized cost function:

$$\mathscr{C}'_j = ||\mathbf{J}_j\delta\mathbf{x}_j - \mathbf{b}_j||^2 \tag{17}$$

where $\mathbf{J}_j$ and $\mathbf{b}_j$ are the Jacobian and residual. In each Gauss-Newton iteration, (17) is solved very efficiently by using the Cholmod algorithm [18]. 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 (17) with respect to $\delta\mathbf{x}_j$ as follows:

$$\begin{aligned} &\mathbf{J}_j^T\mathbf{J}_j\delta\mathbf{x}_j = \mathbf{J}_j^T\mathbf{b}_j \Leftrightarrow \mathbf{G}_j\mathbf{G}_j^T\delta\mathbf{x}_j = \mathbf{J}_j^T\mathbf{b}_j \\ \Leftrightarrow \quad &\mathbf{G}_j\delta\mathbf{y}_j = \mathbf{J}_j^T\mathbf{b}_j, \quad \text{with } \delta\mathbf{y}_j = \mathbf{G}_j^T\delta\mathbf{x}_j \end{aligned} \tag{18}$$

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

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 trajectory, $\bar{\mathbf{p}}_j$, and map, $\mathbf{f}_j$, expressed in its own reference frame. These estimates and the Choleskey factor, $\mathbf{G}_j$, will be provided to the CM algorithm (see Section 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

As shown in [19], 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 (${}^{G_1}_{G_2}\mathbf{C}$ and ${}^{G_1}\mathbf{p}_{G_2}$) between any two users has 4 dof: One corresponding to their relative yaw angle and three corresponding to their relative position.

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

$${}^{G_1}\mathbf{x}_{P_i} = {}^{G_1}\mathbf{p}_{G_2} + {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{x}_{P_i} \tag{19}$$

where ${}^{G_1}\mathbf{x}_{P_i}$, ${}^{G_2}\mathbf{x}_{P_i}$ are the point feature $\mathbf{x}_{P_i}$'s 3D positions expressed in users' frames $\{G_1\}$ and $\{G_2\}$, respectively, and ${}^{G_1}_{G_2}\mathbf{C} = Rot_z(\phi_{12})$.

As we prove in [20], two common points suffice for computing ϕ_{12} and ${}^{G_1}\mathbf{p}_{G_2}$. Furthermore, we employ the two-point minimal solver in conjunction with RANSAC [21] for detecting outliers. Lastly, we use all inliers to find in closed-form the least-squares relative transformation between maps. Due to space limitations, we refer to [20] for the details of the initialization process.

C. Cooperative mapping

In this section, we first present the standard BLS formulation of the CM problem and then reformulate it as an equivalent constrained optimization problem, whose solution takes advantage of the problem's structure.

1) CM problem formulation: As previously mentioned, linking the different users' maps requires using observations of common features. One way to achieve this would be to modify the camera measurement model for all three types of features to explicitly consider the transformation between the reference frames of the users observing the same features. This new camera model, for any common feature observation, can be written in a compact form as:

$$\mathbf{z}_c = \mathbf{s}(\mathbf{x}_a, \mathbf{f}_c, \mathbf{x}_\tau) + \mathbf{n}_c \tag{20}$$

where $\mathbf{x}_\tau$ is a vector of size $4(N-1)$ comprising the pairwise transformations between users computed as described in Section V-B, $\mathbf{x}_a = \begin{bmatrix}\bar{\mathbf{p}}_1^T, {}^{G_1}\mathbf{q}_B^T, \ldots, \bar{\mathbf{p}}_N^T, {}^{G_N}\mathbf{q}_B^T\end{bmatrix}^T$ is the vector comprising all users' poses and their orientations in the Manhattan world, $\mathbf{n}_c$ is the corresponding measurement noise of covariance $\mathbf{R}_c$. Additionally, we split the set of features, $\mathbf{f}$, into two subsets, $\mathbf{f}_c$ and $\mathbf{f}_a$, comprising the set of features observed by only one or multiple users, respectively.

Following this formulation and renaming as $\overline{\mathscr{C}}_i$ the cost function of each user [see (16)] after removing the cost terms involving common features [see (20)], CM requires solving:

$$\mathbf{x}_a^*, \mathbf{f}_a^*, \mathbf{f}_c^*, \mathbf{x}_\tau^* = \operatorname{argmin}\left(\sum_{i=1}^{N}\overline{\mathscr{C}}_i + ||\mathbf{z}_c - \mathbf{s}(\mathbf{x}_a, \mathbf{f}_c, \mathbf{x}_\tau)||^2_{\mathbf{R}_c}\right) \tag{21}$$

To improve the modularity and efficiency of CM, we employ the following theorem:

Theorem 1: The optimization problem (21) is equivalent to the following constrained optimization problem:

$$\mathbf{x}_a^*, \mathbf{f}_a^*, \mathbf{f}_{c_1}^*, \dots, \mathbf{f}_{c_N}^*, \mathbf{x}_\tau^* = \operatorname{argmin} \sum_{i=1}^{N} \mathscr{C}_i \tag{22}$$

$$\text{s. t.} \quad \kappa(\mathbf{x}_a, \mathbf{x}_\tau, \mathbf{f}_{c_i}, \mathbf{f}_{c_j}) = \mathbf{0}, \quad i, j = 1, \dots, N, i \neq j \tag{23}$$

where $\mathscr{C}_i$ denotes the cost function for user i [see (16)], $\mathbf{f}_{c_i}$ is defined as the subset of $\mathbf{f}_c$ observed by user i, and κ denotes common-feature constraints as defined in (8), (12), (13), and (15).

Sketch of the proof: Substituting the constraint (23) in (22) and rearranging terms results in (21). ■

As explained in Section V-C.3, the constrained formulation of (22)-(23) has several advantages over that of (21).

2) CM solution: Since the cost function of (22) is nonlinear, we employ Gauss-Newton iterative minimization [22]. At each iteration, we focus on the following (linearized) constrained BLS problem:

$$\delta\mathbf{x}_\tau^*, \delta\mathbf{x}_1^*, \dots, \delta\mathbf{x}_N^* = \operatorname{argmin} \sum_{i=1}^{N} ||\mathbf{J}_i \delta\mathbf{x}_i - \mathbf{b}_i||^2$$

$$\text{s. t.} \quad \sum_{i=1}^{N} \mathbf{A}_i \delta\mathbf{x}_i + \mathbf{A}_\tau \delta\mathbf{x}_\tau = \mathbf{r} \tag{24}$$

where $\delta\mathbf{x}_i$ is the error state of user i (comprising $\bar{\mathbf{p}}_i$, $\mathbf{f}_{c_i}$, $\mathbf{f}_{a_i}$, and ${}^{G_i}\mathbf{q}_B$), $\delta\mathbf{x}_\tau$ is the error state of $\mathbf{x}_\tau$, while $\mathbf{J}_i$ and $\mathbf{b}_i$ are the corresponding Jacobian and residual of $\mathscr{C}_i$ in (22). $\mathbf{A}_i$, $\mathbf{A}_\tau$, and $\mathbf{r}$ are the Jacobians (corresponding to $\mathbf{x}_i$ and $\mathbf{x}_\tau$) and residual of the constraints, respectively.

The KKT optimality conditions [23] for (24) are:

$$\begin{aligned} &\mathbf{J}_i^T(\mathbf{J}_i \delta\mathbf{x}_i - \mathbf{b}_i) + \mathbf{A}_i^T \lambda = \mathbf{0}, \quad i = 1, \dots, N \\ &\sum_{i=1}^{N} \mathbf{A}_i \delta\mathbf{x}_i + \mathbf{A}_\tau \delta\mathbf{x}_\tau - \mathbf{r} = \mathbf{0} \\ &\mathbf{A}_\tau^T \lambda = \mathbf{0} \end{aligned} \tag{25}$$

where λ is the Lagrange-multiplier vector.

To simplify notation, in what follows, we present our algorithm for solving (25) for the case of two users, while its extension to three or more users is straightforward.

Writing (25) in a compact form yields:

$$\underbrace{\begin{bmatrix} \mathbf{J}_1^T\mathbf{J}_1 & & \mathbf{A}_1^T & \\ & \mathbf{J}_2^T\mathbf{J}_2 & \mathbf{A}_2^T & \\ \mathbf{A}_1 & \mathbf{A}_2 & & \mathbf{A}_\tau \\ & & \mathbf{A}_\tau^T & \end{bmatrix}}_{\mathbf{H}_{CM}} \begin{bmatrix} \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \\ \lambda \\ \delta\mathbf{x}_\tau \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1^T\mathbf{b}_1 \\ \mathbf{J}_2^T\mathbf{b}_2 \\ \mathbf{r} \\ \mathbf{0} \end{bmatrix} \tag{26}$$

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 [24], can be employed to solve (26), we propose an alternative approach that takes advantage of $\mathbf{H}_{CM}$'s structure and the Cholesky factors previously computed by each user as follows:

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

$$\mathbf{H}_{CM} = \begin{bmatrix} \mathbf{G}_1 & & & \\ & \mathbf{G}_2 & & \\ \mathbf{K}_1^T & \mathbf{K}_2^T & \mathbf{T}_{11} & \\ & & \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{G}_1^T & & \mathbf{K}_1 & \\ & \mathbf{G}_2^T & \mathbf{K}_2 & \\ & & -\mathbf{T}_{11}^T & \mathbf{T}_{21}^T \\ & & & -\mathbf{T}_{22}^T \end{bmatrix} \tag{27}$$

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 (27), and employing the structure of $\mathbf{H}_{CM}$ in (26) yields the following system of equations:

$$\mathbf{G}_i\mathbf{K}_i = \mathbf{A}_i^T, \quad i = 1, 2 \tag{28}$$

$$\mathbf{T}_{11}\mathbf{T}_{11}^T = \sum_{i=1}^{2} \mathbf{K}_i^T\mathbf{K}_i \tag{29}$$

$$\mathbf{T}_{11}\mathbf{T}_{21}^T = \mathbf{A}_\tau \tag{30}$$

$$\mathbf{T}_{22}\mathbf{T}_{22}^T = \mathbf{T}_{21}\mathbf{T}_{21}^T \tag{31}$$

To find the $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, and $\mathbf{T}_{22}$ that satisfy (28)-(31), we first compute $\mathbf{K}_i$, $i = 1, 2$, by solving a linear equation corresponding to each of the columns of $\mathbf{K}_i$ [see (28)].

Defining $\mathbf{K} = \left[\mathbf{K}_1^T \mathbf{K}_2^T\right]^T$, it is easy to show that $\sum_{i=1}^{2} \mathbf{K}_i^T\mathbf{K}_i = \mathbf{K}^T\mathbf{K}$ is a positive definite matrix. Thus, we compute $\mathbf{T}_{11}$ as the Cholesky factor of $\mathbf{K}^T\mathbf{K}$ that satisfies (29). Given $\mathbf{T}_{11}$, we determine $\mathbf{T}_{21}$ using triangular back-substitution according to (30).

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 (31)]. ■

Once all the block matrices in (27) are obtained, (26) is efficiently solved by employing two back-substitutions involving triangular matrices.

Now, we briefly discuss the computational complexity of computing each block in (27). The Cholesky factors $\mathbf{G}_i$ 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 $\mathbf{G}_i$ 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 $\mathbf{K}_i$ matrices involves triangular back-substitution according to (28), which has low computational cost for two reasons: (i) the $\mathbf{A}_i$ matrices are very sparse (less than 0.01% nonzero elements); (ii) each column of the $\mathbf{K}_i$ 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.

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 typically a tall matrix, $\mathbf{K}^T\mathbf{K}$ is generally a *small dense* square matrix with size equal to the number of commonly-observed feature constraints. Thus, computing $\mathbf{T}_{11}$ has *cubic* processing cost with respect to the number of constraints.

Lastly, both $\mathbf{T}_{21}$ and $\mathbf{T}_{22}$ are very small matrices, and take little time to compute. Once all the block matrices are computed, solving the linear system requires only two sparse back triangular substitutions.

As we will show in the experimental results, computing $\mathbf{K}$ and $\mathbf{K}^T\mathbf{K}$ are the most computationally demanding parts of our CM algorithm, and the time they take depends on the number of commonly-observed feature constraints. Fortunately, both these two operations are parallelizable.

3) CM solution advantages: Formulating and solving CM as a constrained optimization problem has the following advantages:

Parallelization: In (21), due to the features observed by multiple users, many of the 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 (22) are parallelizable (e.g., computing the $\mathbf{K}_i$ and $\mathbf{G}_i$ matrices). This is of particular importance when mapping very large areas such as airports, museums, shopping malls, etc.

Modularity: In (21), 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 (22) common features always use the same measurement model as other features and the transformation between maps is never required, hence the feature measurement model is uniform. Moreover, adding or removing users' trajectories and maps does not affect the Jacobian matrices of the other users. Instead, we simply add the corresponding constraints. This is especially convenient when expanding the map or updating pre-existing maps.

Efficiency: The Cholesky factor of each individual user's Hessian matrix can be reused in (22) to speed up processing. Moreover, the partitioning of (22) reduces the memory requirements of the Cholesky factorization.

VI. Experiment Results

In what follows, we briefly describe the experiment setup, show the estimated trajectories and maps, compare our result to ground truth, and provide computation times for each step of the algorithm. **An interactive visualization of the experimental results is available online [14].**

A. Dataset collection

The visual and inertial measurements used in our CM tests were collected using a Project Tango developer phone and tablet [25].[2] Greyscale images, with resolution 640×480, were saved at 15 Hz, along with consumer-grade, MEMS-based, IMU data at 100 Hz. Two buildings were mapped, the Keller Hall and Walter Library at the University of Minnesota (UMN) campus. Keller Hall contains four datasets of approximately 1300, 1000, 800, and 500 m, while Walter Library comprises of four datasets with length about 1000, 400, 400, and 100 m.

B. Data preparation and initial estimate

First, each user solves its own single-map (SM) estimation problem via BLS to determine its own trajectory and map based on the following information:

(SM 1) An initial estimate for its trajectory and map: In our case, this is computed using a variant of the multi-state constrained Kalman filter (MSC-KF), based on [19]. Each MSC-KF operates on the IMU measurements and feature tracks collected by each user. Feature tracks correspond to Harris-corners [26] extracted from each image and tracked using the Kanade-Lucas-Tomasi (KLT) algorithm [27]. The subset of these feature tracks that pass the 2pt-RANSAC [28], are used by the MSC-KF to improve the estimated trajectory of each user. These tracks, however, are not used to detect loop closures.

(SM 2) Point-feature loop-closure detection (intra-dataset): To determine if a user has revisited an area, we follow a bag-of-words approach using ORB feature descriptors [29] and employ our implementation of Nistér's vocabulary tree [30]. These matches are confirmed after they pass a 3pt+1-RANSAC [31] geometric-consistency test.

(SM 3) Line tracking: Line segments are extracted from images using the Line Segment Detection (LSD) algorithm [32]. The line tracking process first creates hypotheses for each 3D line's orientation and position by considering all possible line vector pairs $\left({}^{1}\mathbf{s}_i, {}^{2}\mathbf{s}_j\right)$ between the first and the second image. Then, line segments from the third image are used to determine valid hypotheses. Line segment triplets that satisfy the three-image constraints are then considered as a line track, and are used to estimate a 3D line's parameters (see [20] for details). Finally, the line segments from image 1 that were not assigned to any 3D lines are discarded, while the unassigned line segments from images 2 and 3 are used to create new hypotheses that are validated using the segments from image 4. Note that the line segments of image 4 are first tested against the current set of tracks, and the remaining segments are used for hypothesis testing. This process is then repeated as new images are considered.

Once the line-tracking process is completed, the subset of line tracks corresponding to the cardinal directions of the building are identified using a RANSAC-based vanishing point estimator [33]. Lastly, we use the trajectory and line estimates of each user's BLS to find loop-closure line features by accepting free or Manhattan lines at poses where loop-closure point features have previously been found. In particular, if any of these 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 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), and determine the matched images and corresponding common landmarks across all datasets.

(CM 2) Inter-dataset line-feature loop-closure detection: We follow a process similar to (SM 3) using the resulting CM trajectory.

(CM 3) Relative transformation initialization: Once common point features are identified, we follow the approach of Section V-B to compute an initial estimate for the un-

[2]We use the Tango devices for collecting visual and inertial data; all algorithms described in this paper were implemented at UMN.

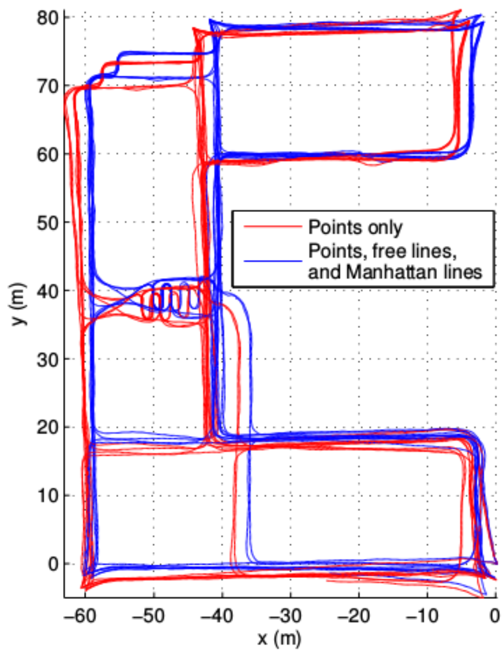

Fig. 4. Trajectories of all users in Keller Hall using points-only versus points, free lines, and Manhattan lines.

Measurements used	Points only	Points and lines
Average target distance error	61 cm	57 cm
Percent error	0.81%	0.69%

TABLE I
PAIRWISE DISTANCE ERROR BETWEEN APRILTAGS IN KELLER.

known 4-dof transformation between user pairs.

Given the above information, we finally employ the algorithm described in Section V-C to solve the CM as a constrained optimization problem. The results from our experiments are summarized in the following section.

C. CM algorithm evaluation

In Figs. 4 and 5, we present the estimated trajectories of all users for the Keller Hall and Walter Library datasets, respectively. The achieved accuracy of the CM algorithm can be qualitatively assessed by examining the $x-z$ view of Fig. 5, and observing that the z (height) estimated for all users' trajectories remains about the same despite the fact that they have travelled for hundreds of meters across multiple floors. Moreover, the effect of using free and Manhattan lines can be observed in Fig. 4, where the yaw error of the trajectory is corrected.

In addition to the qualitative results, we present a ground-truth comparison for Keller Hall. Specifically, we placed four AprilTags [34] in the far corners of a single floor within the building and used the building's blueprints to find the true distance (ground truth) between any pair of AprilTags. Then, to compute the estimated distance between AprilTags, we employ the PnP algorithm of [35] to find an observed AprilTag's position expressed with respect to the camera's frame, and use the CM estimate of the camera frame to

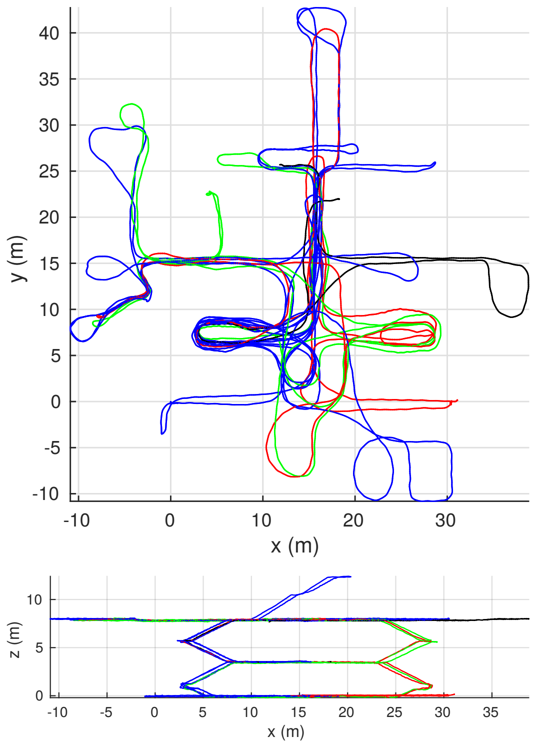

Fig. 5. Trajectories of all users in Walter Library using points, free lines, and Manhattan lines. Each user's trajectory is depicted with a different color.

Calculation and Time			
$\mathbf{G}$	9.7 s	$\mathbf{T}_{21}$	48.4 ms
$\mathbf{T}_{11}$	4.1 s	$\mathbf{K}^T\mathbf{K}$*	255.1 s / 45.1 s
$\mathbf{T}_{22}$	0.2 ms	Back-solve	320.0 ms
Time per iteration	261.5 / 51.5 s	(6 required)	

* Times reported from sequential / parallel implementations.

TABLE II
COOPERATIVE MAPPING PROCESSING TIMES FOR KELLER DATASETS.

express the AprilTag with respect to the global frame. Lastly, we average the estimated positions of each AprilTag across all observations and compute the norm of their differences.[3] Errors in the pairwise distance between AprilTags found from this method when using only points versus when using points, free lines, and Manhattan lines are reported in Table I.

Next, we present the times to compute our CM result on a desktop computer with an Intel® Xeon® E5-1650 v2 processor (3.5 GHz). Our implementation uses the Cholmod [18] algorithm from SuiteSparse to find Cholesky factors, while back-substitution is performed using the Eigen library [36]. In the four Keller Hall datasets, we processed 131,010 points, 1,589 free lines, and 2,582 Manhattan lines. Of these, 1,823 points, 8 free lines, and 33 Manhattan lines were common to two or more datasets. The CM processing times are shown in Table II. As previously mentioned, computing $\mathbf{K}^T\mathbf{K}$ is the most computationally demanding part of the algorithm taking 255.1 s for sequential processing. Exploit-

[3]The distance from the camera to the AprilTag is relatively small as compared to the distance between AprilTags (about 0.3 m compared to over 80 m) so any error in the PnP estimate will negligibly affect the result.

ing the inherent parallelism of our proposed algorithm, the time for computing $\mathbf{K}^T\mathbf{K}$ is reduced to 45.1 s using Intel's "Threading Building Blocks" library [37] (i.e., 17.7% of the sequential processing time).

VII. Conclusion

In this paper, we introduced a cooperative mapping (CM) algorithm for combining visual and inertial measurements collected using mobile devices by multiple users at different times across large indoor spaces. We considered the most general case, where the users' relative transformation is unknown. Our formulation of CM as a Batch Least Squares (BLS) constrained-optimization problem offers significant advantages when processing multiple maps: (i) Modularity as maps (or submaps) can be added and removed with minimal effort; (ii) Computational gains as partial results regarding the trajectory and map of each user can be re-used; (iii) Significant speed ups from parallelizing not only each users's BLS-solution, but also many of the proposed CM algorithm's steps. Two large-scale CM experiments were conducted demonstrating the performance of the proposed algorithm in terms of accuracy and processing speed when using point, free-line, and Manhattan-line visual features.

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 – June 7 2014, pp. 3638–3643.

[2] 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.

[3] 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.

[4] 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.

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

[6] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1066–1077, Oct. 2008.

[7] 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.

[8] 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.

[9] 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.

[10] 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, Sept. 22–25 2002, pp. 513–516.

[11] 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.

[12] 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.

[13] 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.

[14] [Online]. Available: http://onionmaps.info

[15] A. B. Chatfield, *Fundamentals of High Accuracy Inertial Navigation*. American Institute of Aeronautics and Astronautics, 1997, vol. 174.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] C. X. Guo, K. Sartipi, R. DuToit, G. 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," University of Minnesota, Tech. Rep., Mar. 2015. [Online]. Available: http://www-users.cs.umn.edu/~chaguo/pdfs/CM_line.pdf

[21] 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, June 1981.

[22] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[24] 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.

[25] Google, "Project Tango," https://www.google.com/atap/projecttango/.

[26] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of the Alvey Vision Conference*, Manchester, UK, Aug. 31 – Sept. 2 1988, pp. 147–151.

[27] 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.

[28] 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.

[29] 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.

[30] 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, June 17 – 22 2006, pp. 2161–2168.

[31] 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, 2012.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[37] [Online]. Available: https://www.threadingbuildingblocks.org/