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

Chao X. Guo, Kourosh Sartipi, Ryan C. DuToit, Georgios 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 building 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 whole 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 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 poses (position and orientation) is not known.

Early work on CM focused on estimating the relative pose between users by aligning their individual maps but without estimating the resulting combined map [2]. Other approaches employ relative pose measurements between the users (or robots) to determine the transformation between their maps. Specifically, Kim *et al.* [3] proposed a pose-graph algorithm that optimizes over the users' trajectories, but not their maps. On the other hand, [4] and [5] optimize over both trajectories and maps, but also require the users to visit the same location at the same time for capturing inter-user measurements, a requirement that greatly reduces their applicability to general use cases.

To overcome this limitation, DDF-SAM [6] proposes a distributed CM algorithm that 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.[1] Then, each user optimizes its own trajectory and map by combining the received inferred measurements from its neighbors with its local measurements. The main limitation of DDF-SAM is that the resulting estimates from both algorithms are optimal with respect to only *local* information available to each neighborhood, but not with respect to all measurements.

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 aligning with the cardinal directions of a "Manhattan world", provide additional attitude information. In [8] [9], both point and line features are used for improving the localization and mapping accuracy of filtering algorithms. In the context of BLS, [10] 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, the proposed solution decouples the camera motion into two rotations and two translations, then solves for them in successively, rendering it sub-optimal. When lines are used for localization, line-loop-closure measurements are rarely used in practice, with the exception (to the best of our knowledge) of [11]. Specifically, Zhang *et al.* [11] 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:

[1] Since marginalization is computationally expensive, DDF-SAM 2.0 [7] provides an alternative summarization approach by reordering the system. Reordering, however, typically introduces additional fill-ins when solving the system, which increases the computational cost.

- 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.
- 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, to the best of our knowledge, we are the first to provide a robust method for detecting loop-closure line measurements.
- We validate our algorithm with two large-scale 3D experiments using datasets collected from multiple mobile devices.

The rest of the paper is structured as follows: In Section II, we define the CM problem, and present an overview of our proposed algorithm. In Section III, we provide the parameterization and measurement model for point, free-line, and Manhattan-line features, respectively. In Section IV, we present the geometric constraint that arises when two features, defined in two separate user maps, are physically the same feature. In Section V, we briefly review the BLS solution for a single user (trajectory and map), followed by an explanation of our method for finding the initial relative-pose estimate between users. We end this section with our efficient 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 there exists 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 can be divided into 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 visual measurements to 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.

By formulating our CM solution in this manner, our algorithm gains two desirable attributes: (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.

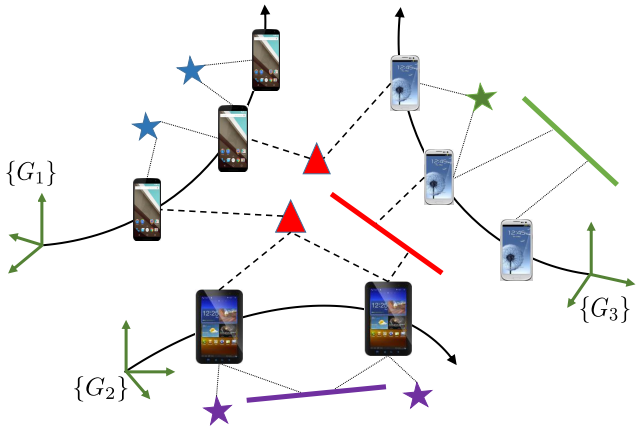

Fig. 1. Point (stars or triangles) or line measurements observed by one or more mobile devices.

III. System State and Measurement Models

In this section, we first describe the cost function introduced by the IMU measurements, and then present the measurement model for processing point, free-line, and Manhattan-line features that align with the principal axes of the building.

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 $\mathbf{e}_1 \triangleq \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$, $\mathbf{e}_2 \triangleq \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$, $\mathbf{e}_3 \triangleq \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$.

A. IMU state and measurement model

The pose (position and attitude quaternion) and velocity of the IMU-camera pair,[2] as well as the IMU biases at time step $i+1$, denoted by the 16×1 vector $\mathbf{p}_{i+1}$, can be computed from $\mathbf{p}_i$ by integrating the IMU's rotational velocity and linear acceleration measurements, $\mathbf{u}_i$. This process is described by the following equation:[3]

$$\mathbf{p}_{i+1} = \mathbf{g}(\mathbf{p}_i, \mathbf{u}_i) + \mathbf{w}_i \tag{1}$$

where $\mathbf{g}(\cdot,\cdot)$ is the nonlinear function corresponding to the IMU measurement model and $\mathbf{w}_i$ is the IMU measurement noise, which is assumed to be zero mean, Gaussian with

[2]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 DOF transformations) of each user in the BLS problem formulation and estimate them concurrently with the trajectories and maps of the users.

[3]The interested reader is referred to [12] and [13] for details on IMU integration.

covariance $\mathbf{Q}_i$, computed beforehand through IMU characterization [14].

B. Point feature state and measurement model

By defining the position of a point feature, $\mathbf{x}_{P_i}$, with respect to the first camera pose, $\{C_0\}$, that observes it as ${}^{C_0}\mathbf{x}_{P_i}$, the camera pose $\{C_k\}$ measures the bearing angle to the feature as:

$$ {}^{C_k}\mathbf{z} = \Pi\left({}^{C_k}\mathbf{p}_{C_0} + {}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{x}_{P_i}\right) + \mathbf{n}_k \tag{2} $$

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

C. Free-line feature state and measurement model

In this paper, we use the same 4 dof free-line parameterization as in [9]. Consider the line $\mathbf{l}_i$ in Fig. 2 which is first observed by camera pose $\{C_0\}$. 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 $\{C_0\}$, x-axis is aligned with the line's direction $\mathbf{l}_i$, and z-axis points away from the origin of $\{C_0\}$. Then, the line is represented with respect to $\{C_0\}$ by the parameter vector ${}^{C_0}\mathbf{x}_{L_i} = \begin{bmatrix} {}^{C_0}\mathbf{q}_{L_i}^T & d_{L_i} \end{bmatrix}^T$. Defining ${}^{C_0}_{L_i}\mathbf{C} \triangleq \mathbf{C}({}^{C_0}\mathbf{q}_{L_i})$, the origin of the line frame in the camera pose $\{C_0\}$ can be written as ${}^{C_0}\mathbf{p}_{L_i} = d_{L_i}\,{}^{C_0}_{L_i}\mathbf{C}\mathbf{e}_3$.

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

$$ \mathbf{s}_k^T\,{}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}_{L_i}\mathbf{C}\mathbf{e}_1 = 0 \tag{3} $$

$$ \mathbf{s}_k^T\left({}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{p}_{L_i} + {}^{C_k}\mathbf{p}_{C_0}\right) = 0 \tag{4} $$

In the presence of noise, the normal vector is defined as $\mathbf{s}_k = \mathbf{C}_{n_1}\mathbf{C}_{n_2}\mathbf{s}'_k$, where $\mathbf{C}_{n_1}$ and $\mathbf{C}_{n_2}$ express the rotational noise whose rotation axes are perpendicular to the measured normal vector $\mathbf{s}'_k$, and rotation angles are the noise magnitude.

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 degrees of freedom. 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 $\{C_0\}$ is represented by the parameter vector ${}^{C_0}\mathbf{x}_{V_i} = \begin{bmatrix} \theta_{V_i} & d_{V_i} \end{bmatrix}^T$, where θ_{V_i} is the angle between ${}^{C_0}\mathbf{p}_{V_i}$ and the y-axis (i.e., $\mathbf{e}_2$), and d_{V_i} is the distance between the origins of $\{C_0\}$ and the Manhattan-line's frame $\{V_i\}$. Using this parameterization, ${}^{C_0}\mathbf{p}_{V_i}$ is expressed as:

$$ {}^{C_0}\mathbf{p}_{V_i} = d_{V_i}\,{}^{C_0}_{G}\mathbf{C}\,{}^{G}_{B}\mathbf{C}(\cos\theta_{V_i}\mathbf{e}_2 + \sin\theta_{V_i}\mathbf{e}_3) \tag{5} $$

Similar to (3) and (4), the geometric constraints corresponding to Manhattan lines are:

$$ \mathbf{s}_k^T\,{}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}_{G}\mathbf{C}\,{}^{G}_{B}\mathbf{C}\mathbf{e}_1 = 0 \tag{6} $$

$$ \mathbf{s}_k^T\left({}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{p}_{V_i} + {}^{C_k}\mathbf{p}_{C_0}\right) = 0 \tag{7} $$

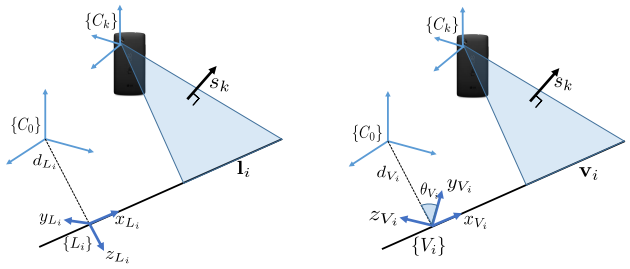

Fig. 2. The parameterization and measurement of free lines (left) and Manhattan lines (right).

IV. Common Features Constraints

In our problem formulation, if a feature is observed by multiple users, we define it as a different feature in each map but ensure geometric consistency by imposing constraints of the common features to be physically the same.

A. Point-feature constraint

Consider a point feature $\mathbf{x}_{P_i}$ observed by users $\{G_1\}$ and $\{G_2\}$, and expressed as ${}^{C_1}\mathbf{x}_{P_i}$ and ${}^{C_2}\mathbf{x}_{P_i}$ with respect to the first observing camera poses in the two user's maps. The geometric constraint between them is:

$$ {}^{C_1}\mathbf{x}_{P_i} - {}^{C_1}_{C_2}\mathbf{C}\,{}^{C_2}\mathbf{x}_{P_i} - {}^{C_1}\mathbf{p}_{C_2} = \mathbf{0} \tag{8} $$

where ${}^{C_1}_{C_2}\mathbf{C}$ and ${}^{C_1}\mathbf{p}_{C_2}$ are then expressed using the local camera poses and the transformation between the two maps:

$$ {}^{C_1}_{C_2}\mathbf{C} = {}^{C_1}_{G_1}\mathbf{C}\,{}^{G_1}_{G_2}\mathbf{C}\,{}^{C_2}_{G_2}\mathbf{C}^T \tag{9} $$

$$ {}^{C_1}\mathbf{p}_{C_2} = {}^{C_1}_{G_1}\mathbf{C}\left({}^{G_1}\mathbf{p}_{G_2} - {}^{G_1}\mathbf{p}_{C_1} + {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{p}_{C_2}\right) \tag{10} $$

B. Free-line constraint

Consider a free-line feature $\mathbf{l}$ observed by users $\{G_1\}$ and $\{G_2\}$, and expressed with respect to the first observing camera poses in the two maps. 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 geometric relation between frames $\{C_1\}$, $\{L_1\}$, $\{C_2\}$, and $\{L_2\}$:

$$ {}^{C_1}_{C_2}\mathbf{C}^T({}^{C_1}\mathbf{p}_{L_1} - {}^{C_1}\mathbf{p}_{C_2}) = {}^{C_2}\mathbf{p}_{L_2} + {}^{C_2}_{L_2}\mathbf{C}\,{}^{L_2}\mathbf{p}_{L_1} \tag{11} $$

where ${}^{L_2}\mathbf{p}_{L_1} = -d_c\mathbf{e}_1$. Defining $\mathbf{E}_{23} \triangleq \begin{bmatrix} \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix}^T$ and multiplying $\mathbf{E}_{23}\,{}^{C_2}_{L_2}\mathbf{C}^T$ to both sides of (11), we obtain the 2 dof constraint:

$$ \mathbf{E}_{23}\,{}^{C_2}_{L_2}\mathbf{C}^T\left({}^{C_1}_{C_2}\mathbf{C}^T({}^{C_1}\mathbf{p}_{L_1} - {}^{C_1}\mathbf{p}_{C_2}) - {}^{C_2}\mathbf{p}_{L_2}\right) = \mathbf{0} \tag{12} $$

Then, since the x-axes of frames $\{L_1\}$ and $\{L_2\}$ are both defined according to the line direction, we have the additional 2 dof orientation constraints:

$$ \mathbf{E}_{23}\left({}^{C_1}_{L_1}\mathbf{C}\mathbf{e}_1 - {}^{C_1}_{C_2}\mathbf{C}\,{}^{C_2}_{L_2}\mathbf{C}\mathbf{e}_1\right) = \mathbf{0} \tag{13} $$

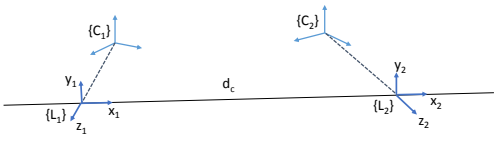

Fig. 3. line constraints

C. Manhattan-line constraint

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

$$ {}^{C_2}_{L_2}\mathbf{C}\,{}^{L_2}\mathbf{p}_{L_1} = -d_{cB_2}\,{}^{C_2}\mathbf{C}\mathbf{e}_1 \tag{14} $$

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

$$ \mathbf{E}_{23}\,{}^{C_2}_{B_2}\mathbf{C}^T\left({}^{C_1}_{C_2}\mathbf{C}^T\left({}^{C_1}\mathbf{p}_{L_1} - {}^{C_1}\mathbf{p}_{C_2}\right) - {}^{C_2}\mathbf{p}_{L_2}\right) = \mathbf{0} \tag{15} $$

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

V. Algorithm Description

In what follows, we briefly review the BLS method for determining the trajectory and map of each user based on only its measurements; then we describe our approach to find the initial estimate for the relative poses between users. Finally, we introduce our CM algorithm.

A. Single-user batch least-squares

Computing the BLS estimate, under the assumptions of Gaussian independent noise and Markovian motion models, is equivalent to the maximum likelihood estimate (MLE), which requires minimizing the following non-linear cost function:

$$ \mathscr{C} = ||\mathbf{p} - \mathbf{g}(\mathbf{p},\mathbf{u})||^2_{\mathbf{Q}} + ||\mathbf{z} - \mathbf{h}(\mathbf{p},\mathbf{f})||^2_{\mathbf{R}} \tag{16} $$

where the first term corresponds to the cost function arising from IMU measurements, and the second term corresponds to the function arising from visual measurements (including point, free-line and Manhattan-line features). $\mathbf{p} \triangleq \left[\mathbf{p}_1^T,\ldots,\mathbf{p}_L^T\right]^T$ denotes all the user poses, $\mathbf{f}$ denotes all the features, $\mathbf{u}$ includes all the IMU measurements, $\mathbf{z}$ includes all the visual measurements, $\mathbf{Q}$ and $\mathbf{R}$ are the covariance matrices of the corresponding observations.

Analytically determining the minimum of (16) is, in general, computationally intractable. For this reason, we employ Gauss-Newton iterative minimization [15]. In particular, expressing the error states of $\mathbf{p}$ and $\mathbf{f}$ with $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{f}}$, respectively, and defining $\delta\mathbf{x} \triangleq \left[\tilde{\mathbf{p}}^T \quad \tilde{\mathbf{f}}^T\right]^T$, we have the linearized system:

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

where $\mathbf{J}$ and $\mathbf{b}$ are the Jacobian and residual. The linear system of equations resulting from minimizing (17) can be solved using either QR factorization of the Jacobian matrix $\mathbf{J}$, which is numerically more stable, or Cholesky factorization of the Hessian matrix $\mathbf{J}^T\mathbf{J}$, which has lower computational complexity. Since we use inertial measurements, the improved condition number and the sparsity (less than 1% nonzero elements) of the Hessian allows us to employ the very efficient Cholmod algorithm [16]. Defining $\mathbf{G}$ as the Cholesky factor of the Hessian, i.e., $\mathbf{G}\mathbf{G}^T = \mathbf{J}^T\mathbf{J}$, we minimize (17) with respect to $\delta\mathbf{x}$ as follows:

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

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

At this point, we should 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, $\mathbf{p}$, and map, $\mathbf{f}$, expressed with respect to each user's local reference frame. These estimates and the Cholesky factor, $\mathbf{G}$, 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' frames of reference is necessary. This initialization process using visual observations of common 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 [17] for outlier rejection and/or to find 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 [18], 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 has four DOF: one corresponding to their relative yaw angle and three corresponding to their relative position. Defining the two users' frames of reference as $\{G_1\}$ and $\{G_2\}$, we need 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 equals:

$$ {}^{G_1}_{G_2}\mathbf{C} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{19} $$

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

$${}^{G_1}\mathbf{x}_{P_i} = {}^{G_1}\mathbf{p}_{G_2} + {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{x}_{P_i} \tag{20} $$

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 $\{G_1\}$ and $\{G_2\}$, respectively.

Subtracting the constraint (20) corresponding to feature $\mathbf{x}_{P_1}$ from the constraints (20) corresponding to $\mathbf{x}_{P_j}, j = 2,\ldots,M$, results in:

$$ {}^{G_1}\mathbf{x}_{P_j} - {}^{G_1}\mathbf{x}_{P_1} = {}^{G_1}_{G_2}\mathbf{C}\left({}^{G_2}\mathbf{x}_{P_j} - {}^{G_2}\mathbf{x}_{P_1}\right) \tag{21} $$

which can be rewritten as:

$$ \mathbf{A}_j \begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix} \triangleq \mathbf{A}_j\mathbf{w} = \mathbf{b}_j, \qquad j = 2,\ldots,M \tag{22} $$

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

$$ \underset{\mathbf{w}}{\text{argmin}} \ \|\mathbf{A}\mathbf{w}-\mathbf{b}\| \qquad \text{s.t.}\ \|\mathbf{w}\|^2 = 1 \tag{23} $$

(23) is a least-squares problem with a quadratic constraint, and its solution can be found in [19]. After solving for the yaw angle θ, we substitute ${}^{G_1}_{G_2}\mathbf{C}$ in (20) and obtain ${}^{G_1}\mathbf{p}_{G_2}$ as:

$$ {}^{G_1}\mathbf{p}_{G_2} = \frac{1}{M}\sum_{j=1}^{M}\left({}^{G_1}\mathbf{x}_{P_j} - {}^{G_1}_{G_2}\mathbf{C}\,{}^{G_2}\mathbf{x}_{P_j}\right). \tag{24} $$

Both ${}^{G_1}\mathbf{p}_{G_2}$ in (24) and ${}^{G_1}_{G_2}\mathbf{C}$ corresponding to the $\mathbf{w}$ calculated in (23) will be used in Section V-C for initializing the CM BLS. Note also that this transformation will be incorporated in the BLS problem so as to improve its accuracy.

The aforementioned process assumes that all matched point features are inliers and their 3D position estimates ${}^{G_1}\hat{\mathbf{x}}_{P_j}$, ${}^{G_2}\hat{\mathbf{x}}_{P_j}$ are accurate. In practice, we employ RANSAC to remove outliers. Specifically, (23) and (24), for $M = 2$, are used as the minimal solver for RANSAC.

2) Initial transformation between multiple users: A naive [$O(n^2)$ cost in the number of users n] 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.[4] To do so, we select the "best" pairwise transformations (in the sense that they are computed using the maximum number of common landmarks) 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 inversely proportional to the number of commonly-observed features between the corresponding users, and then compute the minimum spanning tree following [20].

C. Cooperative Mapping

In this section, we present our proposed CM algorithm which can reuse the available Cholesky factor of each user's Hessian matrix. We start by formulating the CM problem as a *constrained* BLS minimization problem. As it will become evident later on, this formulation has two main advantages:

i) Modularity: Maps (or parts of them) computed by different users at different times can be added or removed without altering the structure of the problem or the Jacobians involved. This is especially convenient when expanding the map or updating pre-existing maps.

ii) Parallelization: By taking advantage of the structure of the constrained BLS problem, a significant part of the calculations can be performed in parallel. This would be of particular importance for mapping very large areas such as airports, museums, shopping malls, etc.

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 corresponding to 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 all common observations can be written in a compact form as:

$$ \mathbf{z}_c = \mathbf{s}(\mathbf{x}_a, \mathbf{f}_c, \mathbf{x}_\tau) + \mathbf{n}_c \tag{25} $$

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, $\mathbf{x}_a = \left[\mathbf{x}_1^T,\ldots,\mathbf{x}_N^T\right]^T$ is the vector comprising all users' poses, $\mathbf{f}_c$ is the set of features observed by two or more users, and $\mathbf{n}_c$ is the corresponding measurement noise of covariance $\mathbf{R}_c$.

Following this formulation and renaming as $\overline{\mathscr{C}}_i$ the cost function of each user [see (16)] after removing the cost terms corresponding to common features [see (25)], requires solving:

$$ \underset{\mathbf{x}_a,\mathbf{f}_a,\mathbf{f}_c,\mathbf{x}_\tau}{\text{argmin}} \left(\sum_{i=1}^{N}\overline{\mathscr{C}}_i + ||\mathbf{z}_c - \mathbf{s}(\mathbf{x}_a,\mathbf{f}_c,\mathbf{x}_\tau)||^2_{\mathbf{R}_c}\right) \tag{26} $$

where $\mathbf{f}_a = \left[\mathbf{f}_1^T,\ldots,\mathbf{f}_N^T\right]^T$ are the features observed by only one user.

To improve the modularity and efficiency for solving the CM problem, we employ the following theorem:

Theorem 1: The optimization problem (26) is equivalent to the following constrained optimization problem:

$$ \underset{\mathbf{x}_a,\mathbf{f}_a,\mathbf{f}_c,\mathbf{x}_\tau}{\text{argmin}} \sum_{i=1}^{N}\mathscr{C}_i \tag{27} $$

$$ \text{s. t.}\quad \mathbf{k}(\mathbf{x}_a,\mathbf{x}_\tau,\mathbf{f}_{c_i},\mathbf{f}_{c_j}) = \mathbf{0}, \quad i,j = 1,\ldots,N, i\neq j \tag{28} $$

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 $\mathbf{k}$ denotes common-feature constraints as defined in (8), (12), (13), and (15). Note that both $\mathbf{f}_{c_i}$ and $\mathbf{f}_{c_j}$ are optimization variables here.

Sketch of the proof: Substituting the constraint (28) in (27) and rearranging terms results in (26). We prove this equivalence for the case of point feature measurements in Appendix-I.■

Formulating and solving (27) is better than (26) for several reasons:

[4]Note that only $n-1$ transformations should be included in the estimation problem, including all possible $n(n-1)/2$ would result into an overdetermined problem.

- In (27), the correlations between users in the cost function are removed, thus we can parallelize most of the required computations. In (26), though due to features observed by multiple users, many of the off-diagonal blocks in the resulting Hessian matrix become dense. Thus, there is no straightforward way to parallelize computations.
- As discussed later on, the Cholesky factor of each individual user's Hessian matrix can be reused by (27) to speed up processing.
- In contrast to (26), in (27) the measurement model for the common features does *not* need to be modified. Thus, all Jacobian matrices can be built without knowing whether features are observed by one or multiple users. Moreover, adding or removing users' trajectories and maps does not affect the Jacobian matrices of the other users. We simply need to add the corresponding constraints.

2) CM Solution: As the cost function of (27) is nonlinear, we again employ Gauss-Newton iterative minimization. At each iteration, we focus on the following (linearized) constrained BLS problem:

$$\begin{aligned} &\underset{\delta\mathbf{x}_\tau,\delta\mathbf{x}_{c_1},\ldots,\delta\mathbf{x}_{c_N}}{\operatorname{argmin}} \sum_{i=1}^{N} ||\mathbf{J}_i\delta\mathbf{x}_{c_i}-\mathbf{b}_i||^2 \\ &\text{s. t.} \quad \sum_{i=1}^{N}\mathbf{A}_i\delta\mathbf{x}_{c_i}+\mathbf{A}_\tau\delta\mathbf{x}_\tau=\mathbf{r} \end{aligned} \tag{29}$$

where $\delta\mathbf{x}_{c_i}$ is the error state of $\mathbf{x}_{c_i}$ (comprising $\mathbf{x}_i$, $\mathbf{f}_i$, and the commonly-observed features $\mathbf{f}_{c_i}$), $\delta\mathbf{x}_\tau$ is the error state of $\mathbf{x}_\tau$, while $\mathbf{J}_i$ and $\mathbf{b}_i$ are the corresponding Jacobian and residual. $\mathbf{A}_i$, $\mathbf{A}_\tau$, and $\mathbf{r}$ are the Jacobians (corresponding to $\mathbf{x}_{c_i}$ and $\mathbf{x}_\tau$) and residual of the constraints, respectively.

The KKT optimality condition [21] for (29) is:

$$\begin{aligned} &\mathbf{J}_i^T(\mathbf{J}_i\delta\mathbf{x}_{c_i}-\mathbf{b}_i)+\mathbf{A}_i^T\lambda=\mathbf{0}, \quad i=1,\ldots,N \\ &\sum_{i=1}^{N}\mathbf{A}_i\delta\mathbf{x}_{c_i}+\mathbf{A}_\tau\delta\mathbf{x}_\tau-\mathbf{r}=\mathbf{0} \\ &\mathbf{A}_\tau^T\lambda=\mathbf{0} \end{aligned} \tag{30}$$

where λ is the Lagrange-multiplier vector.

To simplify notation, in what follows we present our algorithm to solve (30) for the case of two users. This method can be easily extended to three or more users.

Writing (30) in a compact form yields:

$$\underbrace{\begin{bmatrix} \mathbf{J}_1^T\mathbf{J}_1 & & \mathbf{A}_1^T & \\ & \mathbf{J}_2^T\mathbf{J}_2 & \mathbf{A}_2^T & \\ \mathbf{A}_1 & \mathbf{A}_2 & & \mathbf{A}_\tau \\ & & \mathbf{A}_\tau^T & \end{bmatrix}}_{\mathbf{H}_{CM}} \begin{bmatrix} \delta\mathbf{x}_{c_1} \\ \delta\mathbf{x}_{c_2} \\ \lambda \\ \delta\mathbf{x}_\tau \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1^T\mathbf{b}_1 \\ \mathbf{J}_2^T\mathbf{b}_2 \\ \mathbf{r} \\ \mathbf{0} \end{bmatrix} \tag{31}$$

Due to the zeros in $\mathbf{H}_{CM}$'s (3, 3) and (4, 4) block-diagonal elements, it is not positive definite. Thus, Cholesky factorization cannot be applied. Although other methods, such as diagonal pivoting [22], can be employed to solve (31), we propose an alternative approach that takes advantage of the Cholesky factors previously computed by each user. Specifically, we employ the following theorem (triangular factorization) that takes advantage of $\mathbf{H}_{CM}$'s structure:

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

$$\mathbf{H}_{CM}=\begin{bmatrix} \mathbf{G}_1 & & & \\ & \mathbf{G}_2 & & \\ \mathbf{K}_1^T & \mathbf{K}_2^T & \mathbf{T}_{11} & \\ & & \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{G}_1^T & & \mathbf{K}_1 & \\ & \mathbf{G}_2^T & \mathbf{K}_2 & \\ & & -\mathbf{T}_{11}^T & \mathbf{T}_{21}^T \\ & & & -\mathbf{T}_{22}^T \end{bmatrix} \tag{32}$$

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 (32), and employing the structure of $\mathbf{H}_{CM}$ in (31) yields the following system of equations:

$$\mathbf{G}_i\mathbf{K}_i=\mathbf{A}_i^T, \quad i=1,2 \tag{33}$$

$$\mathbf{T}_{11}\mathbf{T}_{11}^T=\sum_{i=1}^{2}\mathbf{K}_i^T\mathbf{K}_i \tag{34}$$

$$\mathbf{T}_{11}\mathbf{T}_{21}^T=\mathbf{A}_\tau \tag{35}$$

$$\mathbf{T}_{22}\mathbf{T}_{22}^T=\mathbf{T}_{21}\mathbf{T}_{21}^T \tag{36}$$

To find matrices $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{T}_{11}$, $\mathbf{T}_{21}$, $\mathbf{T}_{22}$ that satisfy (33)-(36), 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 (33)].

Defining $\mathbf{K}=\begin{bmatrix}\mathbf{K}_1^T\mathbf{K}_2^T\end{bmatrix}^T$, it is easy to see that $\sum_{i=1}^{2}\mathbf{K}_i^T\mathbf{K}_i=\mathbf{K}^T\mathbf{K}$ is a positive definite matrix. Therefore, we select $\mathbf{T}_{11}$ as the Cholesky factor of $\mathbf{K}^T\mathbf{K}$ that satisfies (34). Once matrix $\mathbf{T}_{11}$ is obtained, $\mathbf{T}_{21}$ is computed using triangular back substitution according to (35).

$\mathbf{T}_{21}\mathbf{T}_{21}^T$ can also be shown to be positive definite. Thus, according to (36), $\mathbf{T}_{22}$ can be selected as the Cholesky factor of $\mathbf{T}_{21}\mathbf{T}_{21}^T$. ■

Once all the block matrices in (32) are obtained, (31) is efficiently solved by employing two back substitutions involving triangular matrices.

Now, we will briefly discuss the computational complexity of computing each block in (32). 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 (33), which has low computational cost for two reasons: (i) the $\mathbf{A}_i$ matrices are very sparse (less than 0.01% nonzero elements); (ii) the $\mathbf{K}_i$ matrices can be computed in parallel. However, 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 $\sum\mathbf{K}^T\mathbf{K}$. Although $\mathbf{K}$ is sparse (about 1% nonzero elements), since it is a (very) tall matrix, $\sum\mathbf{K}^T\mathbf{K}$ is generally a *dense* square matrix with size equal to the number of commonly-observed-feature

constraints. Thus, computing $\mathbf{T}_{11}$ has *cubic* processing 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 $\sum \mathbf{K}^T\mathbf{K}$ are the most computationally demanding part of our CM algorithm, and the time it takes depends on the number of commonly-observed feature constraints. Fortunately, these two operations are parallelizable.

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 estimated trajectories and point cloud is available online [23].**

A. Dataset Collection

The visual and inertial measurements used in our CM algorithm were collected using a Project Tango developer phone and tablet [24].[5] Greyscale images, with resolution 640×480, were saved at 15 Hz, along with consumer-grade, MEMS-based, IMU data at 100 Hz. Two separate buildings were mapped, Keller Hall and Walter Library at the University of Minnesota campus. Keller Hall contains four datasets of approximately 1300, 1000, 800, and 500 m, while Walter Library comprises four datasets with length about 1000, 400, 400, and 100 m.

B. Data Preparation and Initial Estimate

Before employing the proposed CM algorithm, users solve their own single-map (SM) estimation problem via BLS to determine their own trajectory and map. Each user requires 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 the approach of [18]. Each MSC-KF operates on the IMU measurements and feature tracks collected by each user. Feature tracks correspond to Harris-corners [25] extracted from each image and tracked using the Kanade-Lucas-Tomasi (KLT) algorithm [26]. The subset of these feature tracks that pass the 2pt-RANSAC [27], 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 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 [28] and employ our implementation of Nistér's vocabulary tree [29]. These matches are confirmed after they pass a 5pt-RANSAC [30] geometric-consistency test.

[5]We only use the Tango phone and tablet for collecting visual and inertial data; all the algorithms described in this paper were implemented by us.

(SM 3) Line tracking. Line segments are extracted from images using the LSD algorithm [31]. The line tracking process first creates hypotheses for each 3D line's orientation and position by considering all possible line vector pairs $\left({}^1s_i, {}^2s_j\right)$ 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 constraints are then used to create hypotheses for a 3D line's rotation and orientation (Further details are described in the Appendix-II). 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. This process is then repeated as new images are considered. Once the line tracking is finished, the subset of line tracks corresponding to the cardinal directions of the building are identified using a RANSAC-based vanishing point estimator [32]. We then 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. If any of these free or Manhattan lines are close to one another, in the sense that the difference of their distance and direction parameters were small (within one degree of orientation and 15cm distance), they are accepted as loop-closure measurements.

With each user having solved its own SM problem, they communicate to the CM their estimated trajectories, maps, Cholesky factors, and all 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. To detect inter-dataset loop closures for free and Manhattan lines, a similar process to (SM 3) is executed using the resulting CM trajectory.

(CM 3) Relative transformation initialization. Once the common landmarks are identified, we follow the approach of Section V-B to compute an initial estimate for the unknown 4 DOF transformations between pairs of users.

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

C. Experimental Results

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

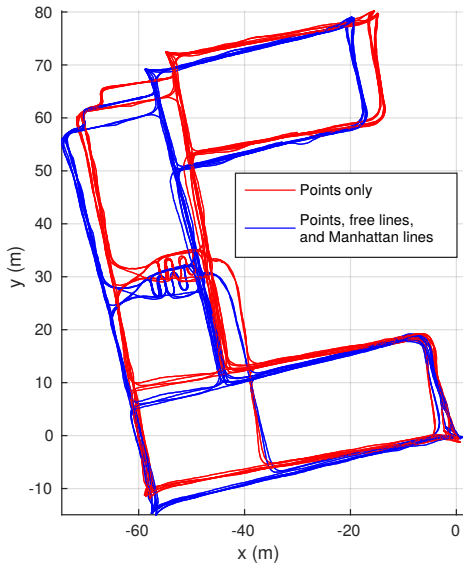

Fig. 4. Trajectory of all users in Keller Hall using points only versus points, free lines, and Manhattan lines.

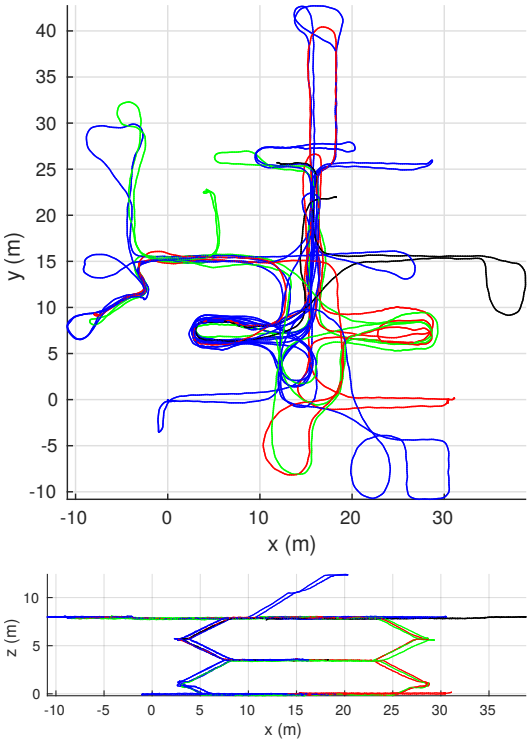

Fig. 5. Trajectories of all users in Walter Library using points, free lines, and Manhattan lines. Each user's trajectory is colored separately.

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 THE KELLER DATASETS.

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/Iteration*	261.5 s / 51.5 s	Iterations	6

* These times are reported from sequential / parallel implementation.

TABLE II

COOPERATIVE MAPPING PROCESSING TIMES FOR KELLER DATASETS.

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, where we placed four AprilTags [33] in the far corners of a single floor within the building. Then, we utilize the building's blueprints to find the true distance between any pair of AprilTags. With the ground truth found, we identify the estimated distance between AprilTags with the following process: First, we employ the PnP algorithm [34] 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 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.[6] Errors in the pairwise distance between AprilTags found from this method when using only points and when using points, free-line, and Manhattan-line measurements are reported in Table I

Next, we present the times to produce our CM result. All times are reported from a desktop computer with an Intel® Xeon® E5-1650 v2 Processor (3.5 GHz). Our implementation uses the Cholmod [16] algorithm from SuiteSparse to find Cholesky factors, while back-substitution is performed using the Eigen library [35].

In the four Keller datasets, we processed 131,010 points, 1,589 free lines, and 2,582 Manhattan lines. Of these features, 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 reported, computing $\mathbf{K}^T\mathbf{K}$ is the most computationally demanding part of the algorithm, which takes 255.1 s for sequential operation. Exploiting 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 [36], which is 17.7% of the sequential operation 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 not known and cannot be inferred by directly observing each other. Our formulation of CM as a Batch Least Squares

[6]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.

(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 the individual users' BLS-solutions, but also many of the steps of the proposed CM algorithm. Two large-scale CM experiments are conducted to assess the performance of the proposed algorithm in terms of accuracy and processing speed when using point, free-line, and Manhattan-line visual measurements. As part of our future work, we plan to employ additional feature types (e.g., planes) for improving accuracy and use the resulting map for localizing users within the building.

APPENDIX-I

To simplify notations in this proof, we assume the sensor directly measures the point feature position. The extension to the camera measurement model is straightforward.

If a landmark is observed at camera poses $\{C_{i0}\}$ and $\{C_{i1}\}$ by user $\{G_i\}$, and camera poses $\{C_{j0}\}$ and $\{C_{j1}\}$ by user $\{G_j\}$, we define its position with respect to the first camera pose when a user observes it as ${}^{C_{i0}}\mathbf{x}_{P_k}$ and ${}^{C_{j0}}\mathbf{x}_{P_k}$. Then in our problem formulation (27), the measurement models corresponding to camera poses $\{C_{i1}\}$ and $\{C_{j1}\}$ are: [7]

$$\mathbf{h}_1 = {}^{C_{i1}}\mathbf{x}_{P_k} = {}^{C_{i1}}_{C_{i0}}\mathbf{C}\big({}^{C_{i0}}\mathbf{x}_{P_k} - {}^{C_{i0}}\mathbf{p}_{C_{i1}}\big) \tag{37}$$

$$\mathbf{h}_2 = {}^{C_{j1}}\mathbf{x}_{P_k} = {}^{C_{j1}}_{C_{j0}}\mathbf{C}\big({}^{C_{j0}}\mathbf{x}_{P_k} - {}^{C_{j0}}\mathbf{p}_{C_{j1}}\big) \tag{38}$$

and the following constraint is added to the optimization problem:

$$\begin{aligned} {}^{G_j}\mathbf{x}_{P_k} &= {}^{G_j}_{G_i}\mathbf{C}\,{}^{G_i}\mathbf{x}_{P_k} + {}^{G_j}\mathbf{p}_{G_i} \\ \Leftrightarrow {}^{G_j}_{C_{j0}}\mathbf{C}\,{}^{C_{j0}}\mathbf{x}_{P_k} + {}^{G_j}\mathbf{p}_{C_{j0}} &= {}^{G_j}_{G_i}\mathbf{C}\big({}^{G_i}_{C_{i0}}\mathbf{C}\,{}^{C_{i0}}\mathbf{x}_{P_k} + {}^{G_i}\mathbf{p}_{C_{i0}}\big) + {}^{G_j}\mathbf{p}_{G_i} \end{aligned} \tag{39}$$

By contrast, in the problem formulation (26), although the measurement model corresponding to $\{C_{i1}\}$ stays the same, the measurement model corresponding to $\{C_{j1}\}$ is changed to:

$$\mathbf{s}_2 = {}^{C_{j1}}\mathbf{x}_{P_k} = {}^{C_{j1}}_{G_j}\mathbf{C}\big({}^{G_j}\mathbf{x}_{P_k} - {}^{G_j}\mathbf{p}_{C_{j1}}\big) \tag{40}$$

where

$${}^{G_j}\mathbf{x}_{P_k} = {}^{G_j}_{G_i}\mathbf{C}\big({}^{G_i}_{C_{i0}}\mathbf{C}\,{}^{C_{i0}}\mathbf{x}_{P_k} + {}^{G_i}\mathbf{p}_{C_{i0}}\big) + {}^{G_j}\mathbf{p}_{G_i} \tag{41}$$

Since substituting the constraint (39) into the measurement (38) results in (40), the two optimization problems are mathematically equivalent.

APPENDIX-II

In this section, we present the method of triangulating lines, both for the minimal and least squares cases. The goal of this section is to compute the rotational matrix corresponding to the line frame, and distance of the line to the origin. Assuming the line is defined in the frame of camera $\{C_0\}$, we define the following:

$${}^{C_0}\mathbf{x}_L \triangleq {}^{C_0}_{L}\mathbf{C}\mathbf{e}_1 \tag{42}$$

$${}^{C_0}\mathbf{z}_L \triangleq {}^{C_0}_{L}\mathbf{C}\mathbf{e}_3 \tag{43}$$

Using these definitions, the line constraints can be written as:

$${}^{C_k}\mathbf{s}_k^T\,{}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{x}_L = 0 \tag{44}$$

$${}^{C_k}\mathbf{s}_k^T\left({}^{C_k}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{P}_L + {}^{C_k}\mathbf{P}_{C_0}\right) = 0 \tag{45}$$

A. Minimal case

Given two observations of line normals in two different camera frames with known poses, we can compute the 4 dof line parameters using the equations (44) and (45). If the two normal observations are made from the cameras C_1 and C_2, we have:

$${}^{C_1}\mathbf{s}_1^T\,{}^{C_1}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{x}_L = 0 \tag{46}$$

$${}^{C_2}\mathbf{s}_2^T\,{}^{C_2}_{C_0}\mathbf{C}\,{}^{C_0}\mathbf{x}_L = 0 \tag{47}$$

Thus, ${}^{C_0}\mathbf{x}_L$ is perpendicular to both ${}^{C_1}_{C_0}\mathbf{C}^{TC_1}\mathbf{s}_1$ and ${}^{C_2}_{C_0}\mathbf{C}^{TC_2}\mathbf{s}_2$, thus we can compute the direction of the line as:

$${}^{C_0}\mathbf{x}_L = \left({}^{C_1}_{C_0}\mathbf{C}^{TC_1}\mathbf{s}_1\right) \times \left({}^{C_2}_{C_0}\mathbf{C}^{TC_2}\mathbf{s}_2\right) \tag{48}$$

Finding ${}^{C_0}\mathbf{x}_L$, fixed 2 dof of the line, and the remaining 2 dofs are extracted by computing ${}^{C_0}\mathbf{P}_L$. Using the relation ${}^{C_0}\mathbf{x}_L^T\,{}^{C_0}\mathbf{P}_L = 0$, we write ${}^{C_0}\mathbf{P}_L = \alpha\,{}^{C_0}\mathbf{x}_L^{\perp} + \beta\,{}^{C_0}\mathbf{x}_L^{\perp\perp}$, in which ${}^{C_0}\mathbf{x}_L^{\perp}$ and ${}^{C_0}\mathbf{x}_L^{\perp\perp}$ are any pair of orthogonal vectors for which $\{{}^{C_0}\mathbf{x}_L, {}^{C_0}\mathbf{x}_L^{\perp}, {}^{C_0}\mathbf{x}_L^{\perp\perp}\}$ is an orthonormal set. Thus:

$$\begin{bmatrix} {}^{C_1}\mathbf{s}_1^T\,{}^{C_1}_{C_0}\mathbf{C} \\ {}^{C_2}\mathbf{s}_2^T\,{}^{C_2}_{C_0}\mathbf{C} \end{bmatrix} \begin{bmatrix} {}^{C_0}\mathbf{x}_L^{\perp} & {}^{C_0}\mathbf{x}_L^{\perp\perp} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = - \begin{bmatrix} {}^{C_1}\mathbf{s}_1^T\,{}^{C_1}\mathbf{P}_{C_0} \\ {}^{C_2}\mathbf{s}_2^T\,{}^{C_2}\mathbf{P}_{C_0} \end{bmatrix} \tag{49}$$

Both α and β can be easily obtained by solving the equation (49). Then:

$${}^{C_0}d_L = \sqrt{\alpha^2 + \beta^2} \tag{50}$$

$${}^{C_0}\mathbf{z}_L = \frac{\alpha\,{}^{C_0}\mathbf{x}_L^{\perp} + \beta\,{}^{C_0}\mathbf{x}_L^{\perp\perp}}{{}^{C_0}d_L} \tag{51}$$

To compute the rotational parameter of the line, we then compute the vector ${}^{C_0}\mathbf{y}_L = {}^{C_0}\mathbf{z}_L \times {}^{C_0}\mathbf{x}_L$ to construct a rotation matrix.

B. Least squares case

Given more than two observations of line normals from known camera poses, the line parameters can be extracted using a similar method as the minimal case. First, we extract the ${}^{C_0}\mathbf{x}_L$ vector, then using the orthogonality constraint we compute both ${}^{C_0}\mathbf{z}_L$ and ${}^{C_0}d_L$ parameters. At this stage, we construct a rotation matrix using ${}^{C_0}\mathbf{x}_L$ and ${}^{C_0}\mathbf{z}_L$ vectors and compute the corresponding quaternion specifying the line orientation with respect to it's first observing camera frame. Assuming the cameras $C_1, C_2 \ldots C_k$ observe the normals

[7] 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.

${}^{C_1}\mathbf{s}_1, {}^{C_2}\mathbf{s}_2 \ldots {}^{C_k}\mathbf{s}_k$ to the line, we define the matrix $\mathbf{M}$ and vector $\mathbf{t}$ as follows:

$$\mathbf{M} \triangleq \begin{bmatrix} {}^{C_1}\mathbf{s}_1^T {}^{C_1}_{C_0}\mathbf{C} \\ {}^{C_2}\mathbf{s}_2^T {}^{C_2}_{C_0}\mathbf{C} \\ \vdots \\ {}^{C_k}\mathbf{s}_k^T {}^{C_k}_{C_0}\mathbf{C} \end{bmatrix}, \mathbf{b} \triangleq - \begin{bmatrix} {}^{C_1}\mathbf{s}_1^T {}^{C_1}\mathbf{P}_{C_0} \\ {}^{C_2}\mathbf{s}_2^T {}^{C_2}\mathbf{P}_{C_0} \\ \vdots \\ {}^{C_k}\mathbf{s}_k^T {}^{C_k}\mathbf{P}_{C_0} \end{bmatrix} \tag{52}$$

It is easy to see that, in absence of noise, $\mathbf{M}^{C_0}\mathbf{x}_L = 0$. Thus, in presence of noise, the least squares solution for ${}^{C_0}\mathbf{x}_L$ is the eigenvector of $\mathbf{M}$ corresponding to its smallest eigenvalue. Having computed ${}^{C_0}\mathbf{x}_L$, we again use the decomposition ${}^{C_0}\mathbf{P}_L = \alpha {}^{C_0}\mathbf{x}_L^{\perp} + \beta {}^{C_0}\mathbf{x}_L^{\perp\perp}$. It is easy to see that α, β satisfy the following system of equations:

$$\mathbf{M} \begin{bmatrix} {}^{C_0}\mathbf{x}_L^{\perp} & {}^{C_0}\mathbf{x}_L^{\perp\perp} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \mathbf{b} \tag{53}$$

After finding α, β using equation (53), the distance and orientation parameters are computed similar to the previous section.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[16] 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.

[17] 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.

[18] 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.

[19] W. Gander, "Least squares with a quadratic constraint," *Numerische Mathematik*, vol. 36, pp. 291–307, 1981.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press Cambridge, 2001.

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

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, June 2004.

[31] 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.

[32] 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.

[33] 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.

[34] B. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and analysis of solutions of the three point perspective pose estimation problem," *International journal of computer vision*, vol. 13, no. 3, pp. 331–356, Dec. 1 1994.

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

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