Efficient Alignment of Visual-Inertial Maps

Kourosh Sartipi and Stergios I. Roumeliotis

Google Inc, CA, USA, 94043
{sartipi,stergiosr}@google.com

Abstract. In this paper, we address the problem of concurrently computing the transformation between multiple, gravity-aligned maps given common point feature observations. In particular, we formulate the problem as a minimization of the distances between shared features expressed with respect to different maps. We show that by marginalizing the maps' relative positions, the KKT conditions of the resulting minimization problem correspond to a system of multivariate polynomial equations (in the $\sin$ and $\cos$ of the relative yaw angles) which can be solved analytically for the case of a few maps. Furthermore, and to improve the efficiency when considering numerous maps, we present a fast, iterative process for determining the unknown relative map orientations.

1 Introduction

Creating an accurate 3D map within a GPS-denied area is required in a wide range of applications such as virtual and augmented reality. A camera and an inertial measurement unit (IMU) sensor pair is ideal for such scenarios because of their availability in most mobile devices. Their low computational resources, however, limit the size of the maps that one can create in real time using visual and inertial measurements. For this reason, many recent approaches to simultaneous localization and mapping (SLAM) employ a fast front-end based on a filtering algorithm (*e.g.*, [21]) while mapping is usually performed either offline or at a lower rate on a server. [1]

In order to reduce the dependence on pre-computed maps or the existence of a cloud infrastructure, recent work has focused on performing on-device mapping. One such popular method is ORB-SLAM [13], where the front-end tracks the camera position and orientation (pose) in real-time, while the back-end computes the local map and processes loop closures. To limit the time for optimizing over a large map, [15] divides it into multiple overlapping submaps, where each of them is optimized independently, and then aligned by iteratively optimizing only the variables involved in the inter-map measurements. In the aforementioned methods, new loop-closures could cause updates propagating through the entire system and introducing delays to the map optimization as the state size increases. These limitations motivate multi-map approaches which bound the time for creating each map.

Early such approaches (*e.g.*, [20]) employ variants of the extended Kalman filter (EKF) to estimate the device's pose and a local map of the surrounding features, while maintaining a global map containing the history of observed features. These methods, however, have quadratic, in the number of features, memory requirements due to storing the covariance matrix for the global map, rendering them unsuitable for mobile devices. To reduce the memory and processing cost, other approaches, such as [16] only update the local map in the exploration phase and postpone applying the loop-closure corrections between the maps. These updates, however, could potentially propagate through all the maps, delaying the response of the system.

[1] For a review of methods employing offline maps or maps from a remote server, please see [3, 7] and references therein.

Alternatively, Cunningham *et al*. [6] employ a smoothing approach to take advantage of the sparse structure of the problem in a multi-device setting, where they jointly optimize the common features and the transformations between the maps. As new measurements are processed, however, it is often required to update all the maps, which leads to significant delays. In [7], although each map is optimized independently (and hence in parallel), its last map-merging step has cost cubic in the number of features *common* to the maps, thus limiting the size of the area that can be mapped on a mobile device. A different process was introduced in [10, 12] where the concept of "anchor nodes" is employed to represent the transformations between different maps in a pose-graph setting. The system is then optimized incrementally for all the poses and map transforms, which is computationally expensive as it involves all the states in the system.

In [3], two distributed optimization schemes on pose graphs are proposed with different communication bandwidth requirements where each robot updates its states based on information from other robots, as well as its own observations. Employing this method for a centralized, multi-map setup on a mobile device is not desirable however, since all submaps require updates.

In the aforementioned approaches [6, 7, 10, 12], new measurements could potentially cause the estimates of all maps to be updated, thus incurring a high processing cost. As it was shown in [15], however, the local estimates of the maps are typically unaffected by the map-merging and joint optimization process. Hence, given accurate maps and their common feature correspondences, one only needs to compute the *transformations* between the maps, which will allow aligning them.

Estimating the 6 degrees of freedom (d.o.f.) transformation between *only* two maps was first solved in closed-form by Horn [9]. This method was extended to *multiple* maps in [1], where a least squares problem for minimizing the distance between common map features was formulated, then the relative map positions were eliminated and the resulting cost function of only map orientations was minimized in an iterative manner. Other methods such as [11, 19] employ a similar minimization approach by employing rotation matrices instead of unit quaternions to represent orientation. Alternatively, Chaudhury *et al*. [2] relax the non-linear, constrained least-squares minimization to a semidefinite program and show improved convergence properties. These algorithms, however, do not take advantage of the problem's structure for cases where the accelerometer measurements render the direction of gravity observable [8]. Specifically, the roll and pitch angles of all map frames are known, reducing the unknown degrees of freedom between relative maps to four (three for position and one for rotation about gravity). Furthermore, they do not incorporate the uncertainty of common feature constraints, which as we will show, improves accuracy. In summary, the main contributions of this paper are:

- We introduce a computationally efficient method for adjusting multiple gravity-aligned maps given common point feature constraints. Moreover, we incorporate the uncertainty in the feature estimates so as to improve accuracy, and introduce a relaxation to our approach to further reduce the processing cost.
- We validate the efficiency and accuracy of our algorithms quantitatively using room and building-scale datasets comprising multiple maps.

2 Technical Approach

Consider N gravity-aligned 3D feature maps with known feature correspondences and unknown transformations between the maps created through a process similar to the algorithm of [7]. Let $\mathcal{F}_{ij}$ be the set of common features between maps i and j, ${}^1\mathbf{p}_i$

and ${}^1_i\mathbf{C}$ the position and orientation of map i with respect to map 1, respectively, where ${}^1_i\mathbf{C} = \mathbf{C}(\mathbf{e}_3, {}^1\theta_i)$ with $\begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix} = \mathbf{I}_3$ in which $\mathbf{I}_3$ is the 3×3 identity matrix, and ${}^i\mathbf{f}_m$ the position of feature m in map i. We define the cost function to be minimized as

$$\mathcal{C} = \sum_{i,j=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ij}} ||{}^1\mathbf{p}_j - {}^1\mathbf{p}_i + {}^1_j\mathbf{C}^j\mathbf{f}_m - {}^1_i\mathbf{C}^i\mathbf{f}_m||^2_{\boldsymbol{\Omega}_{ijm}} \tag{1}$$

where $\mathcal{F}_{ii}$ is an empty set and $\boldsymbol{\Omega}_{ijm}$ is the covariance associated with each cost term.

We start by stating that at its minimum, the cost function $\mathcal{C}$ in (1) will satisfy $\frac{\partial \mathcal{C}}{\partial {}^1\mathbf{p}_k} = 0$, for $k = 2, \ldots, N$, which results in

$$\sum_{i=2}^{N} \mathbf{A}_{ki} {}^1\mathbf{p}_i - \sum_{i=2}^{N} \mathbf{B}_{ki}\mathbf{K}{}^1\mathbf{u}_i - {}^1\mathbf{a}_k = \mathbf{0} \tag{2}$$

where

$$\mathbf{A}_{ki} \triangleq \begin{cases} -\sum_{\mathbf{f}_m \in \mathcal{F}_{ik}} \boldsymbol{\Omega}^{-1}_{ikm} & k \neq i \\ \sum_{i=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ik}} \boldsymbol{\Omega}^{-1}_{ikm} & k = i \end{cases} \tag{3}$$

$$\mathbf{B}_{ki} \triangleq \begin{cases} \sum_{\mathbf{f}_m \in \mathcal{F}_{ik}} \boldsymbol{\Omega}^{-1}_{ikm}\mathbf{L}({}^i\mathbf{f}_m) & k \neq i \\ -\sum_{l=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{lk}} \boldsymbol{\Omega}^{-1}_{lkm}\mathbf{L}({}^k\mathbf{f}_m) & k = i \end{cases} \tag{4}$$

$${}^1\mathbf{u}_i \triangleq \begin{bmatrix}\cos({}^1\theta_i) \\ \sin({}^1\theta_i)\end{bmatrix}, \quad \mathbf{K} \triangleq \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2\end{bmatrix}, \quad \mathbf{L}(\begin{bmatrix}x \\ y \\ z\end{bmatrix}) \triangleq \begin{bmatrix}x & -y & 0 \\ y & x & 0 \\ 0 & 0 & z\end{bmatrix} \tag{5}$$

$${}^1\mathbf{a}_k \triangleq \sum_{\mathbf{f}_m \in \mathcal{F}_{1k}} \boldsymbol{\Omega}^{-1}_{1km} {}^1\mathbf{f}_m + \sum_{i=2}^{N} \mathbf{B}_{ki}\mathbf{e}_3. \tag{6}$$

Defining $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{a}$ as the matrices and vector whose block elements are $\mathbf{A}_{ki}$, $\mathbf{B}_{ki}, \mathbf{K}$ and ${}^1\mathbf{a}_k$, respectively, it is easy to show that

$${}^1\mathbf{p}_k = {}^1\mathbf{t}_k + \sum_{i=2}^{N} \mathbf{T}_{ki} {}^1\mathbf{u}_i \tag{7}$$

where $\mathbf{T}_{ki}$ and ${}^1\mathbf{t}_k$ are 3×2 matrices and 3×1 vectors, respectively, and are defined as

$$\mathbf{A}^{-1}\mathbf{a} = \begin{bmatrix}{}^1\mathbf{t}_2 \\ \vdots \\ {}^1\mathbf{t}_N\end{bmatrix}, \quad \mathbf{A}^{-1}\mathbf{B} = \begin{bmatrix}\mathbf{T}_{22} & \ldots & \mathbf{T}_{2N} \\ \vdots & \ddots & \vdots \\ \mathbf{T}_{N2} & \ldots & \mathbf{T}_{NN}\end{bmatrix}. \tag{8}$$

Substituting (7) into (1) yields a cost function in the orientations *only*:

$$\mathcal{C} = \sum_{i,j=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ij}} ||\mathbf{r}_{ijm} + \sum_{k=2}^{N} \mathbf{Q}_{ijmk} {}^1\mathbf{u}_k||^2_{\boldsymbol{\Omega}_{ijm}} \tag{9}$$

with

$$\mathbf{r}_{ijm} \triangleq {}^1\mathbf{t}_j - {}^1\mathbf{t}_i + \mathbf{L}({}^j\mathbf{f}_m)(\mathbf{e}_3 + \delta_{1j}\mathbf{e}_1) - \mathbf{L}({}^i\mathbf{f}_m)(\mathbf{e}_3 + \delta_{1i}\mathbf{e}_1) \tag{10}$$

$$\mathbf{Q}_{ijmk} \triangleq \mathbf{T}_{jk} - \mathbf{T}_{ik} + \delta_{jk}\mathbf{L}({}^j\mathbf{f}_m)\mathbf{K} - \delta_{ik}\mathbf{L}({}^i\mathbf{f}_m)\mathbf{K} \tag{11}$$

where δ_{ij} is the Kronecker delta and we defined ${}^1\mathbf{t}_1 \triangleq \mathbf{0}_{3\times1}$ and $\mathbf{T}_{1k} \triangleq \mathbf{0}_{3\times2}$. Furthermore, expanding (9) will result in an expression of the form:

$$\mathcal{C} = \mathcal{C}_0 + \bar{\mathbf{v}}^T\bar{\mathbf{u}} + \bar{\mathbf{u}}^T\bar{\mathbf{W}}\bar{\mathbf{u}} \tag{12}$$

where $\bar{\mathbf{u}} \triangleq \left[{}^1\mathbf{u}_2^T \dots {}^1\mathbf{u}_N^T\right]^T$, $\mathcal{C}_0 \triangleq \sum_{i,j=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ij}} \|\mathbf{r}_{ijm}\|^2_{\mathbf{\Omega}_{ijm}}$ and

$$\bar{\mathbf{v}} \triangleq \begin{bmatrix} \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_N \end{bmatrix}, \quad \bar{\mathbf{W}} \triangleq \begin{bmatrix} \mathbf{W}_{22} & \dots & \mathbf{W}_{2N} \\ \vdots & \ddots & \vdots \\ \mathbf{W}_{N2} & \dots & \mathbf{W}_{NN} \end{bmatrix} \tag{13}$$

$$\mathbf{v}_k \triangleq 2 \sum_{i,j=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ij}} \mathbf{Q}_{ijmk}^T \mathbf{\Omega}_{ijm}^{-1} \mathbf{r}_{ijm} \tag{14}$$

$$\mathbf{W}_{kl} \triangleq \sum_{i,j=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ij}} \mathbf{Q}_{ijmk}^T \mathbf{\Omega}_{ijm}^{-1} \mathbf{Q}_{ijml}. \tag{15}$$

Note that (12) can be minimized *analytically* subject to the constraints ${}^1\mathbf{u}_k^T {}^1\mathbf{u}_k = 1, k = 2, \dots, N$, at a cost exponential in the number of maps. Specifically, by defining $\nu_{2i-3} = \cos({}^1\theta_i)$ and $\nu_{2i-2} = \sin({}^1\theta_i)$, for $i = 2, \dots, N$, it is straightforward to show that (12) becomes a quadratic with quadratic constraints of the form $\nu_{2i-3}^2 + \nu_{2i-2}^2 = 1, i = 2, \dots, N$. Formulating the Lagrangian $\mathcal{L} = \mathcal{C} + \sum_{i=2}^{N} \omega_i(\nu_{2i-3}^2 + \nu_{2i-2}^2 - 1)$ of (12) and computing the KKT conditions will result into a set of quadratic equations whose solution has complexity exponential in the number of maps N [5].

Thus, following such an approach is of practical use for only a small number of maps. Alternatively, we employ the first-order Taylor series expansion, which results in $\bar{\mathbf{u}} \simeq \hat{\bar{\mathbf{u}}} + \mathbf{G}\tilde{\boldsymbol{\psi}}$ in which $\hat{\bar{\mathbf{u}}}$ is the evaluation of vector $\bar{\mathbf{u}}$ around the current estimate obtained analytically from pairwise map matches, and

$$\tilde{\boldsymbol{\psi}} \triangleq \begin{bmatrix} {}^1\tilde{\theta}_2 \\ \vdots \\ {}^1\tilde{\theta}_N \end{bmatrix}, \quad \mathbf{G} \triangleq \begin{bmatrix} \mathbf{J}^1\hat{\mathbf{u}}_2 & & & \\ & \mathbf{J}^1\hat{\mathbf{u}}_3 & & \\ & & \ddots & \\ & & & \mathbf{J}^1\hat{\mathbf{u}}_N \end{bmatrix}, \quad \mathbf{J} \triangleq \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{16}$$

where ${}^1\tilde{\theta}_k = {}^1\theta_k - {}^1\hat{\theta}_k, k = 2, \dots, N$, denotes the error between the true map's, ${}^1\theta_k$, and estimated, ${}^1\hat{\theta}_k$, orientations. Substituting (16) in (12) results in the error cost function:

$$\mathcal{C}' = \mathcal{C}_0' + (\bar{\mathbf{v}} + 2\bar{\mathbf{W}}\hat{\bar{\mathbf{u}}})^T \mathbf{G}\tilde{\boldsymbol{\psi}} + \tilde{\boldsymbol{\psi}}^T \mathbf{G}^T \bar{\mathbf{W}} \mathbf{G}\tilde{\boldsymbol{\psi}} \tag{17}$$

in which $\mathcal{C}_0' = \mathcal{C}_0 + \mathbf{v}^T\hat{\bar{\mathbf{u}}} + \hat{\bar{\mathbf{u}}}^T\bar{\mathbf{W}}\hat{\bar{\mathbf{u}}}$ is constant. To minimize (17), we set $\frac{\partial \mathcal{C}'}{\partial \tilde{\boldsymbol{\psi}}} = \mathbf{0}$, then

$$\mathbf{G}^T\bar{\mathbf{W}}\mathbf{G}\tilde{\boldsymbol{\psi}} = -\mathbf{G}^T\left(\bar{\mathbf{W}}\hat{\bar{\mathbf{u}}} + \frac{1}{2}\bar{\mathbf{v}}\right). \tag{18}$$

After finding $\tilde{\boldsymbol{\psi}}$ from (18) and updating the current estimates of the map orientations ${}^1\theta_i^{\oplus} = {}^1\theta_i^{\ominus} + {}^1\tilde{\theta}_i, i = 2, \dots, N$, we compute the map positions from (7) and initiate a new iteration until convergence ($||\tilde{\boldsymbol{\psi}}||/(N-1) \leq 10^{-5}$). Note that solving (18) requires $\mathcal{O}(N^3)$ operations, hence the overall algorithm's complexity is set by computing the $\bar{\mathbf{W}}$ matrix in (12), which is $\mathcal{O}(N^4M)$ with $M \triangleq \max|\mathcal{F}_{ij}|$. The matrix $\bar{\mathbf{W}}$ and vector $\bar{\mathbf{v}}$, however, depend only on feature positions and term covariances, and hence need to be computed only *once* in the iterative minimization process.

Note that in the special case where all the covariances are equal to the same scalar times identity (*i.e.*, $\mathbf{\Omega}_{ijm} = \sigma^2\mathbf{I}_3$), we can simplify the algorithm further, achieving lower processing at the expense of accuracy. Specifically, the cost function to minimize

becomes

$$\mathcal{C}' = \sum_{i,j=1}^{N} \sum_{\mathbf{f}_m \in \mathcal{F}_{ij}} \left\| {}^1\mathbf{p}_j - {}^1\mathbf{p}_i + {}^1_j\mathbf{C}^j \mathbf{f}_m - {}^1_i\mathbf{C}^i \mathbf{f}_m \right\|^2. \tag{19}$$

Following the same process as for minimizing (1), it is straightforward to show that

$${}^1\mathbf{p}_k = \sum_{l=2}^{N} \gamma_{kl} {}^1\mathbf{a}'_l - \sum_{i,l=2}^{N} \gamma_{kl} {}^1_i\mathbf{C}^i \mathbf{b}'_l \tag{20}$$

where

$${}^i\mathbf{a}'_k \triangleq \sum_{f_m \in \mathcal{F}_{ik}} {}^i\mathbf{f}_m, \quad {}^i\mathbf{b}'_k \triangleq \begin{cases} -{}^i\mathbf{a}_k & i \neq k \\ \sum_{j=1}^{N} {}^i\mathbf{a}_j & i = k \end{cases}, \quad \lambda_{ki} \triangleq \begin{cases} -|\mathcal{F}_{ki}| & k \neq i \\ \sum_{j=1}^{N} |\mathcal{F}_{jk}| & i = k \end{cases} \tag{21}$$

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_{22} & \dots & \lambda_{2N} \\ \vdots & \ddots & \vdots \\ \lambda_{N2} & \dots & \lambda_{NN} \end{bmatrix}, \quad \boldsymbol{\Lambda}^{-1} \triangleq \begin{bmatrix} \gamma_{22} & \dots & \gamma_{2N} \\ \vdots & \ddots & \vdots \\ \gamma_{N2} & \dots & \gamma_{NN} \end{bmatrix}. \tag{22}$$

Equation (20) can be further simplified by defining ${}^1\mathbf{t}'_k \triangleq \sum_{l=2}^{N} \gamma_{kl} {}^1\mathbf{a}'_l$ and ${}^i\mathbf{t}'_k \triangleq -\sum_{l=2}^{N} \gamma_{kl} {}^i\mathbf{b}'_l$:

$${}^1\mathbf{p}_k = {}^1\mathbf{t}'_k + \sum_{i=2}^{N} {}^1_i\mathbf{C}^i \mathbf{t}'_k. \tag{23}$$

Computing ${}^i\mathbf{a}'_k$ and ${}^i\mathbf{b}'_k$ requires $\mathcal{O}(N^2 M)$ operations, while finding $\boldsymbol{\Lambda}^{-1}$ has $\mathcal{O}(N^3)$ cost. Note, however, that as compared to computing $\mathbf{A}_{ki}$ and $\mathbf{B}_{ki}$ in (2), which involve 3×3 matrix multiplications and additions, determining ${}^i\mathbf{a}'_k$ and ${}^i\mathbf{b}'_k$ involves only vector additions. Furthermore, calculating $\boldsymbol{\Lambda}^{-1}$ requires inverting a $(N-1) \times (N-1)$ matrix, compared to $\mathbf{A}^{-1}$ in (3) which is a $3(N-1) \times 3(N-1)$ matrix.

Substituting (23) into (19) and rearranging terms results in the following cost function involving only orientations:

$$\mathcal{C}' = \sum_{i,j=1}^{N} \sum_{f_m \in \mathcal{F}_{ij}} ||\mathbf{r}'_{ijm} + \sum_{l=2}^{N} {}^1_l\mathbf{C}^l \mathbf{q}_{ijm}||^2 \tag{24}$$

where

$$\mathbf{r}'_{ijm} \triangleq {}^1\mathbf{t}'_j - {}^1\mathbf{t}'_i + \delta_{j1} {}^j\mathbf{f}_m - \delta_{i1} {}^i\mathbf{f}_m \tag{25}$$

$${}^l\mathbf{q}_{ijm} \triangleq {}^l\mathbf{t}'_j - {}^l\mathbf{t}'_i + \delta_{jl} {}^j\mathbf{f}_m - \delta_{il} {}^i\mathbf{f}_m \tag{26}$$

and we have defined ${}^1\mathbf{t}'_1 \triangleq \mathbf{0}_{3\times 1}$. Note that the third element of the 3×1 vectors in (24) will not be affected by the rotation matrices since all rotations are around the z-axis. Hence, we can disregard the last element of the vectors $\mathbf{r}'_{ijm} + \sum_{l=2}^{N} {}^1_l\mathbf{C}^l \mathbf{q}_{ijm}$ in (24) to reduce computations. This is equivalent to defining $\bar{\mathbf{r}}_{ijm} \triangleq \mathbf{K}^T \mathbf{r}'_{ijm}$, ${}^l\bar{\mathbf{q}}_{ijm} \triangleq \mathbf{K}^T {}^l\mathbf{q}_{ijm}$ and ${}^1_l\mathbf{R} \triangleq \mathbf{K}^T {}^1_l\mathbf{C}\mathbf{K}$ and minimizing the following modified cost function instead:

$$\mathcal{C}'' = \sum_{i,j=1}^{N} \sum_{f_m \in \mathcal{F}_{ij}} ||\bar{\mathbf{r}}_{ijm} + \sum_{l=2}^{N} {}^1_l\mathbf{R}^l \bar{\mathbf{q}}_{ijm}||^2 \tag{27}$$

which can be written as

$$\mathcal{C}'' = \mathcal{C}_0'' + \mathbf{g}^T \bar{\mathbf{u}} + \bar{\mathbf{u}}^T \bar{\mathbf{H}} \bar{\mathbf{u}} \tag{28}$$

where we have employed the identity ${}^1_k\mathbf{R}^T {}^1_k\mathbf{R} = \mathbf{I}_2$ and defined

$$\mathcal{C}_0'' \triangleq \sum_{i,j} \sum_{f_m \in \mathcal{F}_{ij}} \left\{ \|\bar{\mathbf{r}}_{ijm}\|^2 + \sum_{l=2}^{N} \left\| {}^l\bar{\mathbf{q}}_{ijm} \right\|^2 \right\} \tag{29}$$

$$\mathbf{v}_l' \triangleq 2 \sum_{ij} \sum_{f_m \in \mathcal{F}_{ij}} \bar{\mathbf{Q}}_{ijml}^T \bar{\mathbf{r}}_{ijm}, \quad \bar{\mathbf{Q}}_{ijml} \triangleq \mathbf{K}^T \mathbf{L}({}^l\mathbf{q}_{ijm}) \mathbf{K} \tag{30}$$

$$\mathbf{H}_{kl} \triangleq \begin{cases} \sum_{ij} \sum_{f_m \in \mathcal{F}_{ij}} \bar{\mathbf{Q}}_{ijmk}^T \bar{\mathbf{Q}}_{ijml} & k \neq l \\ \mathbf{0}_{2\times 2} & k = l \end{cases} \tag{31}$$

$$\mathbf{g} \triangleq \begin{bmatrix} \mathbf{v}_2'^T & \dots & \mathbf{v}_N'^T \end{bmatrix}^T, \quad \bar{\mathbf{H}} \triangleq \begin{bmatrix} \mathbf{H}_{22} & \dots & \mathbf{H}_{2N} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{N2} & \dots & \mathbf{H}_{NN} \end{bmatrix}. \tag{32}$$

Lastly, we employ an iterative approach by first linearizing $\bar{\mathbf{u}}$ around its current estimate, and obtain $\tilde{\boldsymbol{\psi}}$ from:

$$\mathbf{G}^T \bar{\mathbf{H}} \mathbf{G} \tilde{\boldsymbol{\psi}} = -\mathbf{G}^T \left(\bar{\mathbf{H}} \hat{\bar{\mathbf{u}}} + \frac{1}{2} \mathbf{g} \right). \tag{33}$$

As before, the solution to (33) is used to update the orientation estimates, compute the positions, and initiate a new iteration until convergence.

3 Experimental Results

The visual and inertial data used in our tests were collected and processed online using two Google Pixel phones. Greyscale images of resolution 640×480 were saved at 30 Hz, along with consumer-grade, MEMS-based, IMU data at 200 Hz. We have divided our datasets into two categories: (i) Room-scale datasets with VICON ground-truth, and (ii) Building-scale datasets for which we have used the optimal Batch Least-Square (BLS) estimate as the ground-truth.

The visual and inertial data are first processed by the square root inverse filter [21], which operates on IMU data and feature tracks extracted from the images. The latter are produced by extracting FAST [17] corners and tracked by matching their corresponding ORB [18] descriptors in consecutive images. To detect loop-closures, we follow a bag-of-words approach using the ORB descriptors and employ the 3pt+1-RANSAC [14] for outlier-detection. As data are processed by the filter, feature tracks, loop-closures, IMU measurements, and the state estimates of the filter are passed to the BLS process.

Multiple maps are created by splitting temporally a dataset into smaller maps with equal temporal lengths. This process breaks the feature tracks spanning more than one map, but has the advantage that the common (among maps) features are known and no extra processing is required for determining them. We start by computing the BLS estimate of each map, and store the IMU-camera's trajectory and the feature positions, along with their covariances $\mathbf{P}_{im}$. The latter are approximated by inverting the 3×3 diagonal blocks of the Hessian matrix corresponding to each feature. Furthermore, we confirm the inliers among the maps' common feature pairs by passing them through the

2pt+1-RANSAC. Specifically, given the features ${}^1\mathbf{f}_1, {}^1\mathbf{f}_2$ and ${}^2\mathbf{f}_1, {}^2\mathbf{f}_2$, the minimal solver of the 2pt+1-RANSAC computes the relative yaw ${}^1\theta_2$ and position ${}^1\mathbf{p}_2$, by minimizing (1) analytically for the case of two maps.

To compute an *initial estimate* for the relative yaws, a naive solution would be to directly use the result of the 2pt+1 RANSAC in the outlier rejection step between maps 1 and $i, i = 2, \ldots, N$. There is, however, no guarantee that more than one common features exist between map 1 and every other map, hence this approach is prone to failure. Instead, we construct a graph where the nodes represent maps and the edges' weights are equal to the number of common features between the maps. We then compute the Maximum Spanning Tree (MST) of this graph following [4] which computes a tree with the maximum weights connecting *all* maps. Subsequently, for each map, we traverse the MST from the map's node to the root of the tree (*i.e.*, map 1), and compute the relative yaw by using only the relative transformation in the path to the root.

Given the inlier common feature constraints and an initial estimate for the map transformations, we only need to evaluate the common feature constraint covariance $\boldsymbol{\Omega}_{ijm}$. To do so, we assume that in each map i, the feature m's error state ${}^i\tilde{\mathbf{f}}_m$ is drawn from a zero-mean Gaussian distribution with covariance $\mathbf{P}_{im}$. Next, we define the noise of the cost term resulting from the observation of feature m by maps i and j as

$$\mathbf{n}_{ijm} = {}^1\mathbf{p}_j - {}^1\mathbf{p}_i + {}^1_j\mathbf{C}\,{}^j\mathbf{f}_m - {}^1_i\mathbf{C}\,{}^i\mathbf{f}_m \tag{34}$$

Where $\mathbf{n}_{ijm}$ is a zero mean Gaussian noise with covariance $\boldsymbol{\Omega}_{ijm}$. By considering the maps' transformations as (unknown) deterministic variables, $\mathbf{n}_{ijm} \simeq {}^1_i\mathbf{C}\,{}^i\tilde{\mathbf{f}}_m - {}^1_j\mathbf{C}\,{}^j\tilde{\mathbf{f}}_m$ and thus its covariance is $\boldsymbol{\Omega}_{ijm} = {}^1_i\mathbf{C}\mathbf{P}_{im}{}^1_i\mathbf{C}^T + {}^1_j\mathbf{C}\mathbf{P}_{jm}{}^1_j\mathbf{C}^T$. Note that although this approximate covariance is smaller (in the positive-definite sense) than the true covariance of the cost terms, our results show that using it significantly improves the accuracy.

Specifically, for our evaluations, we employ the proposed map-alignment process (see Sect. 2), to compute the map transformations and create a fused map along with the IMU-camera aligned trajectories. The latter are then compared to the ground truth (VICON for room-scale, BLS estimate for building-scale). Note that the errors between the fused trajectory and the ground truth are caused by (i) errors in the map transformations (*e.g.*, due to outliers or approximate covariances), and (ii) the map's local trajectory inaccuracies due to processing each map independently.

In this work, we used five datasets to evaluate our method in terms of accuracy and efficiency (see Table 1). For datasets L1 and L2, which are building size, we use as ground truth the optimal BLS solution. Datasets S1, S2, S3, on the other hand, are room size where the device is moved inside a VICON room while collecting data. A visualization of the fused trajectories against the ground truth for L1 and S1 is depicted in Fig. 1 and Fig. 2, respectively. Moreover, the position root mean square error (RMSE), the total length of the trajectory and the computation times are provided in Table 1 for the case of using covariances or without them (*i.e.*, $\boldsymbol{\Omega}_{ijm} = \sigma^2\mathbf{I}_3$), denoted by *WC* and *WOC*, respectively. To compare the accuracy and the processing time of our methods to alternative approaches that compute the 6 d.o.f. transformations between the maps, we have employed the Gauss-Newton algorithm to minimize a cost function similar to (1) where the rotations are no longer gravity-aligned. The Gauss-Newton approach is denoted by *GN* in Table 1. Additionally, we have provided the number of maps and maximum number of common features between maps. Comparing the RMSE values of GN and WC shows that our method converges to the optimal map-alignment solution.

As previously stated (Sect. 2), while our proposed methods' processing time is quartic in the number of maps, it only grows linearly in the number of common features. This is clearly shown when the number of maps grows by a factor of four, the time to com-

pute the transformations grows by a factor of three hundred, while varying the number of common features has smaller impact. Note though, even in this case the total time is only a fraction of a second, which verifies the efficiency of our approach. Additionally, WC times are between 10%-60% faster than GN times. We have also included the times required to optimize the entire system in BLS (denoted by BLS Time), and the average time of solving each map individually (denoted by Ind. Time in Table 1) so as to highlight the advantage of using multiple maps in an online setting.

Besides the processing cost savings, Table 1 also demonstrates that our methods are, in general, more accurate than GN estimating the maps' 6 d.o.f. poses. This is due to the gravity-alignment of the maps, *i.e.*, the roll and pitch of the trajectory are accurately estimated in the visual-inertial BLS, hence estimating them in the multi-map alignment could lead to overfitting on the data and less accurate estimates. In some cases though, small errors in the roll and pitch of different maps, as in dataset S3, causes the RMSE of the 4 d.o.f. estimate to be slightly higher than that of the 6 d.o.f.

Dataset	WC-RMSE (cm)	WOC RMSE (cm)	GN-RMSE (cm)	WC-Time (ms)	WOC-Time (ms)	GN-Time (ms)	Num. Maps	Max Common Features	Dataset Length (m)	BLS Time (sec)	Ind. Time (sec)
L1	**30.1**	38.1	**30.1**	371.8	**201.5**	515.2	21	276	752	-	6.59
L2	**28.3**	46.4	28.5	206.5	**133.9**	289.1	15	243	323	209	3.66
S1	**13.0**	14.6	13.1	0.7	**0.6**	1.2	5	18	112	82	25.5
S2	**5.0**	5.4	5.2	1.8	**0.9**	3.2	5	70	104	123	34.7
S3	8.0	10.0	**7.7**	1.2	**0.7**	1.9	5	28	104	102	18.9

Table 1: Table of RMSE errors (cm) and the calculation times. We were unable to compute the BLS time of dataset L1 as the phone ran out of memory.

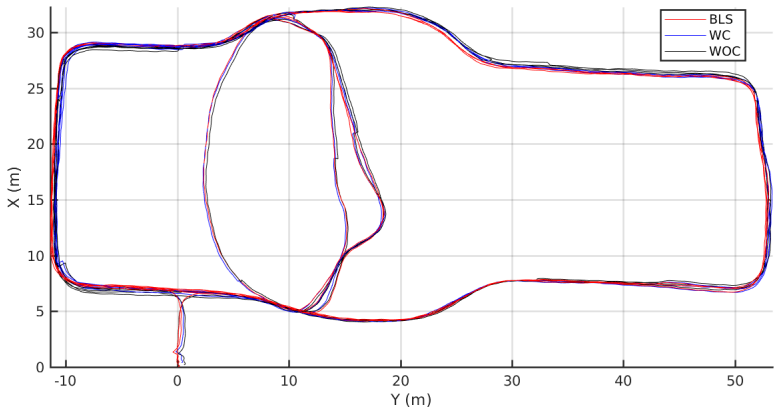

Fig. 1: Dataset L1

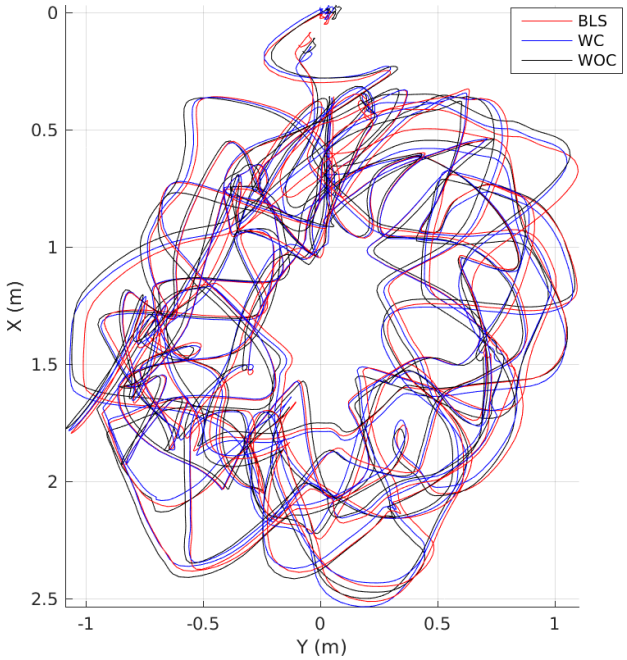

Fig. 2: Dataset S1

4 Conclusion and Future Work

In this work, we introduced an efficient method for computing the 4 d.o.f. transformation between multiple maps given 3D point feature correspondences. We cast the map alignment problem as a least squares minimization of the Mahalanobis distance between common features, and solve it efficiently by eliminating the maps' relative positions and computing their relative yaw angles first. We have experimentally evaluated our method in datasets of different scales (room-size and building-size) and have verified its accuracy and speed. As part of our future work, we seek to investigate methods that split the maps based on criteria other than time so as to ensure high accuracy. Additionally, we aim to further reduce the processing cost by efficiently selecting and reprocessing the most informative features within each, as well as across multiple maps.

References

1. Benjemaa, R., Schmitt, F.: A solution for the registration of multiple 3D point sets using unit quaternions. In: Proc. of the European Conference on Computer Vision. pp. 34–50. Freiburg, Germany (Jun 2–6 1998)
2. Chaudhury, K.N., Khoo, Y., Singer, A.: Global registration of multiple point clouds using semidefinite programming. SIAM Journal on Optimization 25(1), 468–501 (2015)
3. Choudhary, S., Carlone, L., Nieto, C., Rogers, J., Christensen, H.I., Dellaert, F.: Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. The International Journal of Robotics Research 36(12), 1286–1311 (2017)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press Cambridge (2001)

5. Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra. Springer-Verlag New York, Inc., Secaucus, NJ (2007)
6. Cunningham, A., Indelman, V., Dellaert, F.: DDF-SAM 2.0: Consistent distributed smoothing and mapping. In: Proc. of the IEEE International Conference on Robotics and Automation. pp. 5220–5227. Karlsruhe, Germany (May 6–10 2013)
7. Guo, C.X., Sartipi, K., DuToit, R.C., Georgiou, G.A., Li, R., O'Leary, J., Nerurkar, E.D., Hesch, J.A., Roumeliotis, S.I.: Large-scale cooperative 3D visual-inertial mapping in a manhattan world. In: Proc. of the IEEE International Conference on Robotics and Automation. pp. 1071–1078. Stockholm, Sweden (May 16–21 2016)
8. Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I.: Consistency analysis and improvement of vision-aided inertial navigation. IEEE Trans. on Robotics 30(1), 158–176 (Feb 2014)
9. Horn, B.K.: Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America 4(4), 629–642 (1987)
10. Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., Teller, S.: Multiple relative pose graphs for robust cooperative mapping. In: Proc. of the IEEE International Conference on Robotics and Automation. pp. 3185–3192. Anchorage, AK (May 3–8 2010)
11. Krishnan, S., Lee, P.Y., Moore, J.B., Venkatasubramanian, S.: Global registration of multiple 3D point sets via optimization-on-a-manifold. In: Proc. of the Eurographics Symposium on Geometry Processing. SGP '05, Aire-la-Ville, Switzerland (Jul 4–6 2005)
12. McDonald, J., Kaess, M., Cadena, C., Neira, J., Leonard, J.J.: Real-time 6-DOF multi-session visual SLAM over large-scale environments. Robotics and Autonomous Systems 61(10), 1144–1158 (Oct 2013)
13. Mur-Artal, R., Montiel, J., Tardós, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans. on Robotics 31(5), 1147–1163 (2015)
14. Naroditsky, O., Zhou, X.S., Roumeliotis, S.I., Daniilidis, K.: Two efficient solutions for visual odometry using directional correspondence. IEEE Trans. on Pattern Analysis and Machine Intelligence 34(4), 818–824 (Apr 2012)
15. Ni, K., Steedly, D., Dellaert, F.: Tectonic SAM: Exact, out-of-core, submap-based SLAM. In: Proc. of the IEEE International Conference on Robotics and Automation. pp. 1678–1685. Rome, Italy (Apr 10–14 2007)
16. Piniés, P., Paz, L.M., Gálvez-López, D., Tardós, J.D.: CI-Graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system. Journal of Field Robotics 27(5), 561–586 (2010)
17. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. Proc. of the European Conference on Computer Vision pp. 430–443 (May 7–13 2006)
18. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: Proc. of the IEEE International Conference on Computer Vision. pp. 2564–2571. Barcelona, Spain (Nov 6–13 2011)
19. Williams, J., Bennamoun, M.: Simultaneous registration of multiple corresponding point sets. Computer Vision and Image Understanding 81(1), 117–142 (Jan 2001)
20. Williams, S.B., Dissanayake, G., Durrant-Whyte, H.: Towards multi-vehicle simultaneous localisation and mapping. In: Proc. of the IEEE International Conference on Robotics and Automation. vol. 3, pp. 2743–2748. Washington, DC (May 11–15 2002)
21. Wu, K.J., Ahmed, A.M., Georgiou, G.A., Roumeliotis, S.I.: A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In: Proc. of Robotics: Science and Systems. Rome, Italy (Jul 12–16 2015)