Visual SLAM: Why Bundle Adjust?

Álvaro Parra Bustos[1], Tat-Jun Chin[1], Anders Eriksson[2] and Ian Reid[1]

***Abstract*— Bundle adjustment plays a vital role in feature-based monocular SLAM. In many modern SLAM pipelines, bundle adjustment is performed to estimate the 6DOF camera trajectory and 3D map (3D point cloud) from the input feature tracks. However, two fundamental weaknesses plague SLAM systems based on bundle adjustment. First, the need to carefully initialise bundle adjustment means that all variables, in particular the map, must be estimated as accurately as possible and maintained over time, which makes the overall algorithm cumbersome. Second, since estimating the 3D structure (which requires sufficient baseline) is inherent in bundle adjustment, the SLAM algorithm will encounter difficulties during periods of slow motion or pure rotational motion.**

We propose a different SLAM optimisation core: instead of bundle adjustment, we conduct rotation averaging to incrementally optimise *only camera orientations*. *Given* the orientations, we estimate the camera positions and 3D points via a quasi-convex formulation that can be solved efficiently and *globally optimally*. Our approach not only obviates the need to estimate and maintain the positions and 3D map at keyframe rate (which enables simpler SLAM systems), it is also more capable of handling slow motions or pure rotational motions.

I. INTRODUCTION

Let $\mathbf{u}_{i,j}$ be the 2D coordinates of the i-th scene point as seen in the j-th image Z_j. Given a set $\{\mathbf{u}_{i,j}\}$ of observations, structure-from-motion (SfM) aims to estimate the 3D coordinates $\mathsf{X} = \{\mathbf{X}_i\}$ of the scene points and 6DOF poses $\{(\mathbf{R}_j, \mathbf{t}_j)\}$ of the images $\{\mathsf{Z}_j\}$ that agree with the observations. The bundle adjustment (BA) formulation is

$$\min_{\{\mathbf{X}_i\},\{(\mathbf{R}_j,\mathbf{t}_j)\}} \quad \sum_{i,j} \left\| \mathbf{u}_{i,j} - f(\mathbf{X}_i \mid \mathbf{R}_j, \mathbf{t}_j) \right\|_2^2, \tag{1}$$

where $f(\mathbf{X}_i \mid \mathbf{R}_j, \mathbf{t}_j)$ is the projection of $\mathbf{X}_i$ onto Z_j (assuming calibrated cameras). In practice, not all $\mathbf{X}_i$ are visible in every Z_j, thus some of the (i, j) terms are dropped. For ease of exposition, we follow [1] and regard the image set $\{\mathsf{Z}_j\}$ as inputs to BA, bearing in mind that the effective inputs are the observations $\{\mathbf{u}_{i,j}\}$ and the visibility matrix.

As a non-linear least squares problem, (1) is usually solved by gradient descent methods, e.g., Levenberg-Marquardt, which require initialisation for all unknowns. Thus, apart from the images $\{\mathsf{Z}_j\}$, the total inputs to a BA instance typically include the initial values for $\{(\mathbf{R}_j, \mathbf{t}_j)\}$ and X.

BA is justifiable in the maximum likelihood sense if the errors due to the uncertainty in localising the feature points $\{\mathbf{u}_{i,j}\}$ are Normally distributed. However, it is not obvious that available feature detectors satisfy this property [2], [3], [4]. While this does not reduce the usefulness of BA, its statistical validity should not be taken for granted.

[1]School of Computer Science, The University of Adelaide.

[2]School of Electrical Engineering and Computer Science, Queensland University of Technology

Algorithm 1 BA-SLAM (adapted from [1]).

1: $\mathsf{X} \leftarrow$ Initialise_points(Z_0).
2: **for** each keyframe step $t = 1, 2, \ldots$ **do**
3: $\quad s \leftarrow t - (\text{window size}) + 1$.
4: $\quad$ **if** a number of $n \geq 1$ points left field of view **then**
5: $\quad\quad \mathsf{X} \leftarrow \mathsf{X} \cup$ initialise_n_new_points(Z_t).
6: $\quad$ **end if**
7: $\quad \mathbf{R}_{s:t}, \mathbf{t}_{s:t}, \mathsf{X} \leftarrow \text{BA}(\mathbf{R}_{s:t}, \mathbf{t}_{s:t}, \mathsf{X}, \mathsf{Z}_{0:t})$.
8: $\quad$ **if** loop is detected in Z_t **then**
9: $\quad\quad \mathbf{R}_{1:t}, \mathbf{t}_{1:t}, \mathsf{X} \leftarrow \text{BA}(\mathbf{R}_{1:t}, \mathbf{t}_{1:t}, \mathsf{X}, \mathsf{Z}_{0:t})$.
10: $\quad$ **end if**
11: **end for**

A. BA-SLAM

Roughly speaking, monocular feature-based SLAM [5] (henceforth, "SLAM") is the execution of SfM incrementally to process streaming input images $\mathsf{Z}_{0:t}$, where

$$\mathsf{Z}_{0:t} = \{\mathsf{Z}_0, \mathsf{Z}_1, \ldots, \mathsf{Z}_t\}. \tag{2}$$

Several influential works [6], [7], [1], [8] have cemented the importance of BA in SLAM. Algorithm 1, which is adapted from [1, Table 1], describes a SLAM optimisation core based on BA over keyframes. Specifically:

- In Step 5, new scene points are "spawned" if the current frame Z_t does not adequately observe the map X.
- In Step 7 (a.k.a. *local mappping*), BA is used to estimate the camera trajectory and 3D map in the current time window. Often, local mapping is preceded by *camera tracking* to accurately initialise the current pose $(\mathbf{R}_t, \mathbf{t}_t)$. See [1, Sec. 5.3] or [8, Sec. V] for examples.
- In Step 9 (a.k.a. *loop closure*), a system-wide BA is executed to reoptimise all the variables and redistribute accumulated drift errors. Implicit in Algorithm 1 is the introduction of covisibility information between Z_t and older keyframes, prior to BA. Often, Step 9 is preceded by *pose graph optimisation* [9], [10], [11], [12] to give a more accurate initialisation of the poses.

Note that Algorithm 1 is merely a "basic recipe" for SLAM. In practice, "*what will make or break a real-time SLAM system are all the (often heuristic) nitty-gritty details*" [13], e.g., how to select features/keyframes, how to update the covisibility graph, how to select/merge/prune 3D points, etc. However, since our focus is on optimisation, Algorithm 1 is sufficient to capture the core algorithmic elements of SLAM systems based on BA, such as ORB-SLAM [8].

B. Why do we want an alternative to BA-SLAM?

1) High system complexity: Besides computing the trajectory and map in the current vicinity, Step 7 in BA-SLAM plays a more basic role: incrementally estimating the variables in small BA "chunks" to serve as initialisation for the system-wide BA in Step 9. Note that since (1) is amenable to only locally optimal solutions, without good initial values for the large number of variables (poses and 3D points), Step 9 will converge to poor solutions.

Therefore, unavoidably all variables must be estimated as accurately as possible and updated at keyframe rate throughout the lifetime of the algorithm—we argue that this increases the complexity of SLAM systems. For example, while Algorithm 1 shows only the creation of new 3D points (Step 5), in a practical system (e.g., [8]) a host of other heuristics are required for map maintenance, e.g., map point selection, point culling, map updating and aggregation. Many of these heuristics contain a number of thresholds, which, if not tuned carefully, will lead to system failure.

2) Difficulties due to pure rotational motion: More fundamentally, since the estimation of 3D points (which require sufficient baseline) are essential, it is unavoidable that a system based on BA-SLAM will encounter numerical issues during periods of pure rotational motion or slow motion [14, Sec. 7.1], and will require special treatment to deal with this problem [15], [16], [17], [18]. This issue is particularly acute at the start of the sequence where the camera is usually slow moving[1]. For example, in ORB-SLAM, elaborate map initialisation heuristics [8, Sec. IV] (cf. Step 1 in Algorithm 1) are used to combat inaccuracies due to insufficient translations. However, experienced users of ORB-SLAM cite its difficulty to initialise on challenging image sequences.

II. L-INFINITY SLAM

Towards simpler visual SLAM systems, we propose a novel optimisation core called *L-infinity SLAM*; see Algorithm 2. A main distinguishing factor is that the online effort (Steps 4 and 8) are devoted to estimating only the camera orientations via *rotation averaging* [19], [20]. Given the orientations, a separate optimisation via the *known rotation problem* [21], [22] (Steps 5 and 9) is conducted to obtain the camera positions and 3D map—since rotation averaging can be done independently from position and map estimation, Steps 5 and 9 can be performed in a lower priority thread.

A. Rotation averaging and known rotation problem

Given a set of relative rotations $\{\mathbf{R}_{j,k}\}$ between pairs of overlapping images $\{\mathsf{Z}_j, \mathsf{Z}_k\}$, the goal of rotation averaging is to estimate the absolute rotations $\{\mathbf{R}_j\}$ that are consistent with the relative rotations (Sec. III-A will provide details on estimating relative rotations in our work). Following [20], we chose the chordal metric for rotations, which yields the

[1]Pioneers of monocular SLAM [5] call the deliberate motion to initialise a SLAM algorithm the "SLAM wiggle".

Algorithm 2 L-infinity SLAM.

1: **for** each keyframe step $t = 1, 2, \ldots$ **do**
2: $s \leftarrow t - (\text{window size}) + 1$.
3: $\{\mathbf{R}_{j,k}\}_{j,k\in\mathsf{N}_{\text{win}}} \leftarrow \text{relative\_rotation}(\mathsf{Z}_{(s-1):t})$.
4: $\mathbf{R}_{s:t} \leftarrow \text{rotation\_averaging}\left(\{\mathbf{R}_{j,k}\}_{j,k\in\mathsf{N}_{\text{win}}}\right)$.
5: $\mathbf{t}_{s:t}, \mathsf{X} \leftarrow \text{known\_rotation\_prob}(\mathbf{R}_{s:t}, \mathsf{Z}_{0:t})$.
6: **if** loop is detected in Z_t **then**
7: $\{\mathbf{R}_{j,k}\}_{j,k\in\mathsf{N}_{\text{sys}}} \leftarrow \text{relative\_rotation}(\mathsf{Z}_{0:t})$.
8: $\mathbf{R}_{1:t} \leftarrow \text{rotation\_averaging}\left(\{\mathbf{R}_{j,k}\}_{j,k\in\mathsf{N}_{\text{sys}}}\right)$.
9: $\mathbf{t}_{1:t}, \mathsf{X} \leftarrow \text{known\_rotation\_prob}(\mathbf{R}_{1:t}, \mathsf{Z}_{0:t})$.
10: **end if**
11: **end for**

rotation averaging formulation

$$\min_{\{\mathbf{R}_j\}} \quad \sum_{j,k\in\mathsf{N}} \left\| \mathbf{R}_{j,k} - \mathbf{R}_k \mathbf{R}_j^{-1} \right\|_F^2, \tag{3}$$

where N is the covisibility graph. In Step 3 of L-infinity SLAM, the covisibility graph N_{win} in the window is used, while in Step 8, the system-wide covisibility graph N_{sys} (updated to account for loop closure) is used. Sec. III will describe the specific algorithm for (3).

Given a set of absolute camera orientations $\{\mathbf{R}_j\}$, the known rotation problem (KRot) [21] optimises the camera positions $\{\mathbf{t}_j\}$ and 3D points $\{\mathbf{X}_i\}$ as

$$\min_{\{\mathbf{X}_i\},\{\mathbf{t}_j\}} \quad \max_{i,j} \quad \|\mathbf{z}_{i,j} - f(\mathbf{X}_i \mid \mathbf{R}_j, \mathbf{t}_j)\|_2, \tag{4}$$

subject to cheirality constraints (details in Sec. III). Observe that unlike (1) which minimises the sum of squared reprojection errors, (4) minimises the maximum reprojection error, which can be viewed as the ℓ_∞-norm of the vector of reprojection errors (leading to the name "L-infinity SLAM").

At this juncture, it is vital to note that (4) is quasi-convex, which is amenable to *efficient global solution* [21], [22]. In our work, a novel variant of KRot is proposed specifically for the loop closure optimisation in Step 9; details in Sec. III.

B. Benefits of L-infinity SLAM

1) Simplicity: As alluded to above, tracking and loop-closing in L-infinity SLAM estimate only orientations. Since positions and 3D map are obtained via an independent optimisation problem that can be solved globally optimally, the results of Steps 5 and 9 do not affect the results of subsequent instances[2]. Therefore, there is no need to accurately calculate positions and 3D map on-the-fly and maintain/propagate them. Note that in Algorithm 2, Steps 5 and 9 are shown mainly to make the overall functionality of L-infinity SLAM equivalent to BA-SLAM. Contrast this to the equivalent steps in BA-SLAM (Steps 7 and 9 in Algorithm 1), whose resulting quality are vital at all times to ensure correct operation.

A signficant advantage of the processing flow of L-infinity SLAM is that many tasks related to map maintenance (e.g., feature/map point selection, point culling, map updating

[2]Their results can be used to warm start the subsequent instances, but this is an optional computational consideration.

and aggregation) can be done in a low priority thread, or even offline if there is no need for on-the-fly position and map estimation (e.g., the application in [23]). This has the potential to significantly simplify visual SLAM systems.

2) Handling pure rotation motion: It is well-known that under epipolar geometry, camera orientation can be estimated independently from the translation [24]. Hence, since the online routines in L-infinity SLAM estimate orientations only, a real-time system based on L-infinity SLAM is less likely to encounter difficulties due to pure or close-to-pure rotational motions. Potential numerical issues due to insufficient baselines between camera views can be handled in the low-priority thread that estimates position and 3D map. Sec. IV-B will provide results that illustrate this advantage of L-infinity SLAM over BA-SLAM.

C. Concerns on global optimality and outliers

Some readers may find it disconcerting that in L-infinity SLAM the estimation of the variables are detached. *First, note that we have not claimed that L-infinity SLAM is globally optimal in all variables.* Second, there is ample evidence [25], [20] that rotation averaging algorithms are capable of producing highly-accurate orientation estimates, independently from positions and 3D points. Since the quasiconvex estimation for positions and 3D points is globally optimal, the overall quality of L-infinity SLAM will be high, as we will demonstrate in Sec. IV.

Also, a common impression of ℓ_∞ estimation is its sensitivity to outliers. Note, however, that both the ℓ_∞ and ℓ_2 norms have a breakdown point of 0 [26], hence both norms are equally susceptible to outliers. In practical BA-SLAM systems, a typical remedy is to pass the ℓ_2 residual through an isotropic robust norm (e.g., Cauchy norm). Likewise, there are efficient and theoretically justified techniques to identify and remove outliers in ℓ_∞ estimation [27], [28]. Hence, outliers do not present a problem for L-infinity SLAM.

III. Algorithmic details

In this section, we describe the details of the core optimisation routines in L-infinity SLAM. Consider a calibrated camera with $\mathbf{K}$ the $\mathbb{R}^{3\times3}$ camera intrinsic matrix. Let

$$\mathbf{P}_j = \mathbf{K}[\mathbf{R}_j \ \mathbf{t}_j] \tag{5}$$

be the projection matrix of the j-th image with assumed known rotation matrix $\mathbf{R}_j$ in $SO(3)$ and unknown translation vector $\mathbf{t}_j$ in $\mathbb{R}^3$. For simplicity, we assume $\mathbf{K} = \mathbf{I}_{3\times3}$. For an arbitrary $\mathbf{K}$, derivations are still valid if camera extrinsics are recovered by applying $\mathbf{K}^{-1}$ to $\mathbf{R}_j$ and $\mathbf{t}_j$, which now form the three first columns and the last column of $\mathbf{P}_j$.

A. Estimating relative motions

L-infinity SLAM estimates camera rotations from relative camera rotations $\mathbf{R}_{j,k}$ in the covisibility graph N_{win}. We simply estimate $\mathbf{R}_{j,k}$ from the essential matrix which can be decomposed into $\mathbf{R}_{j,k}$ and a relative translation direction $\mathbf{t}_{j,k}^{(E)}$ ($\|\mathbf{t}_{j,k}^{(E)}\| = 1$). A weakness of this decomposition is the need of images with sufficient displacement; however, other methods can be used to estimate relative motions [24], [29], [30]. In the case of low displacement, we estimated $\mathbf{R}_{j,k}$ by rotationally aligning backprojected feature rays by using a rotation only variant of Trimmed ICP [31].

B. Rotation averaging

Several methods exist to solve (3) [32], [19], [33], [20]. Here we adopt the robust method of [33] which uses an iteratively reweighted least-squares approach in $SO(3)$. The method in [33] is simpler than BA as, for example, no linearisation and no estimation of a damping factor is required.

C. Known rotation problem

By referring to $\mathbf{R}_j^{(1:2)}$ as the first two rows of $\mathbf{R}_j$, and to $\mathbf{R}_j^{(3)}$ as the third row of $\mathbf{R}_j$ (similarly for $\mathbf{t}^{(1:2)}$ and $\mathbf{t}^{(3)}$), the projection of $\mathbf{X}_i$ onto the j-th image is given by

$$f(\mathbf{X}_i \mid \mathbf{R}_j, \mathbf{t}_j) := \frac{\mathbf{R}_j^{(1:2)}\mathbf{X}_i + \mathbf{t}_j^{(1:2)}}{\mathbf{R}_j^{(3)}\mathbf{X}_i + \mathbf{t}_j^{(3)}}, \tag{6}$$

and the known rotation problem (4) can be rewritten by adding an extra variable γ as

$$P_0: \quad \min_{\{\mathbf{X}_i\},\{\mathbf{t}_j\}} \ \gamma \tag{7a}$$

$$\text{subject to} \quad \frac{\left\| \mathbf{A}_{i,j} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix} \right\|_2}{\mathbf{b}_j^\top \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix}} \le \gamma, \quad \forall i,j, \tag{7b}$$

$$\mathbf{b}_j^\top \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix} \ge 0, \quad \forall i,j, \tag{7c}$$

$$\gamma \ge 0, \tag{7d}$$

where

$$\mathbf{A}_{i,j} = \begin{bmatrix} \mathbf{S}_{i,j} & \mathbf{I}_{2\times2} & \mathbf{z}_{i,j} \end{bmatrix}, \quad \mathbf{b}_j = \begin{bmatrix} \mathbf{R}_j^{(3)} & 0 & 0 & 1 \end{bmatrix}^\top, \quad \text{and} \tag{8}$$

$$\mathbf{S}_{i,j} = \mathbf{R}_j^{(1:2)} - \mathbf{z}_{i,j}\mathbf{R}_j^{(3)}. \tag{9}$$

Cheirality constraints (7c) impose to $\mathbf{X}_i$ to lie in front of the cameras in which $\mathbf{X}_i$ is visible.

Intuitively, γ defines the sublevel sets of the objective in (4), i.e., the maximum over the LHS of (7b). For a fixed γ,

$$\left\| \mathbf{A}_{i,j} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix} \right\|_2 - \gamma \mathbf{b}_j^\top \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix} \ \le \ 0 \tag{10}$$

defines a convex set hence the objective is quasi-convex and (P_0) is a quasi-convex problem. For a detailed proof of (P_0) being a quasi-convex problem the reader can refer to [34].

1) Solving the known rotation problem: (P_0) can be rewritten as

$$P_1: \quad \min_{\{\mathbf{X}_i\},\{\mathbf{t}_j\}} \ \gamma \tag{11a}$$

$$\text{subject to} \quad \left\| \mathbf{A}_{i,j} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix} \right\|_2 \le \gamma \, \mathbf{b}_j^\top \begin{bmatrix} \mathbf{X}_i \\ \mathbf{t}_j \end{bmatrix}, \ \forall i,j, \tag{11b}$$

$$\gamma \ge 0, \tag{11c}$$

in which the Cheirality constraints (7c) are implicit in (11b) as both the LHS of (11b) and γ are non-negative. If γ is fixed, constraints (11b) became second-order cones which allows to solve (P_1) by using bisection trough SOCP feasibility tests [21]. Several other methods solve (P_1) through SOCP sub-problems. [35] shown that Gugat's algorithm [36] outperforms among this type of methods (bisection, Brent's algorithm[37], and Dinkelbach's algorithm [38], [22]). Recently, Zhang et al. [39] presented a method, named Res-Int, which outperformed existent methods by alternating between pose estimation and triangulation to efficiently partition the problem into small sub-problems without compromising global optimality. As a result, Res-Int solves (P_1) in about 3 seconds for moderate size input (around 15 images and 3000 3D points).

We use Res-Int as the known_rotation_prob routine in Line 5 in Algorithm 2 since its superior performance. Although Res-Int converges in a few seconds for moderate problems (as those optimised for a moving window in Algorithm 2), its performance is still inadequate for medium to large size problems ($> 10,000$ 3D points, > 100 images) that Algorithm 2 optimises when detecting a loop. For such problems, Res-Int could take from a few minutes to hours to converge (see [39, Table 2]). To efficiently address loop closure, we propose a new formulation that incorporates relative camera translation directions (obtained from the essential matrix) to alleviate the size of the problem but still produce an accurate result.

D. Known rotation problem with translation direction constraints

Solving KRot in Step 9 in Algorithm 2 can be excessively time-consuming as a loop can be detected at an advanced stage generating a large input size. Instead, we propose to address loop closure over a sample of the input with a formulation that incorporates camera translation directions.

Inspired by the quasi-convex approach of Sim and Hartley [40] to estimate the camera translations from $\mathbf{t}_{j,k}^{(E)}$ and known camera rotations, we constrained camera positions

$$\mathbf{C}_j = -\mathbf{R}_j^\top \mathbf{t}_j \tag{12}$$

in the known rotation problem (P_1) to agree up to an angular threshold

$$\angle(\mathbf{t}_{j,k}, \mathbf{C}_k - \mathbf{C}_j) \leq \alpha \quad \forall j,k, \tag{13}$$

to

$$\mathbf{t}_{j,k} = (\mathbf{K}^{-1}\mathbf{R}_j)^\top \mathbf{t}_{j,k}^{(E)}, \tag{14}$$

which is the relative translation direction in world coordinates (we explicitly apply $\mathbf{K}^{-1}$ to $\mathbf{R}_j$ in (14) in case $\mathbf{K} \neq \mathbf{I}_{3\times 3}$ and therefore $\mathbf{R}_j$ is not a rotation matrix).

We observed that the method of [40] was unable to produce satisfactory results (see results in Sec. IV) for loop closure, arguably since no structural information is optimised thus camera positions were not sufficiently constrained. On the other hand, adding angle constraints (13) to (P_1) it allows to efficiently solve loop closure (< 100 s) with a sparse set of 3D points (300 points for result in Fig. 2b). Our proposition yields the following problem.

$$P_3: \quad \min_{\{\mathbf{X}_i\},\{\mathbf{C}_j\}} \quad \gamma \tag{15a}$$

$$\text{subject to} \quad \left\| \mathbf{B}_{i,j} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{C}_j \end{bmatrix} \right\|_2 \leq \gamma\, \mathbf{c}_j^\top \begin{bmatrix} \mathbf{X}_i \\ \mathbf{C}_j \end{bmatrix}, \ \forall i,j, \tag{15b}$$

$$\left\| \mathbf{D}_{j,k} \begin{bmatrix} \mathbf{C}_j \\ \mathbf{C}_k \end{bmatrix} \right\|_2 \leq \mathbf{e}_{j,k}^\top \begin{bmatrix} \mathbf{C}_j \\ \mathbf{C}_k \end{bmatrix}, \ \forall j,k, \tag{15c}$$

$$\gamma \geq 0, \tag{15d}$$

where

$$\mathbf{B}_{i,j} = \begin{bmatrix} \mathbf{S}_{i,j} & -\mathbf{S}_{i,j} \end{bmatrix}, \qquad \mathbf{c}_j = \begin{bmatrix} \mathbf{R}_j^{(3)} & -\mathbf{R}_j^{(3)} \end{bmatrix}, \tag{16}$$

$$\mathbf{D}_{j,k} = \begin{bmatrix} \mathbf{Z}_{j,k}^{(1:2)} & -\mathbf{Z}_{j,k}^{(1:2)} \end{bmatrix}, \quad \mathbf{e}_{j,k} = \tan(\alpha) \begin{bmatrix} \mathbf{t}_{j,k}^\top & -\mathbf{t}_{j,k}^\top \end{bmatrix}^\top, \tag{17}$$

and $\mathbf{Z}_{j,k}$ is a rotation matrix such that

$$\mathbf{Z}_{j,k}\mathbf{t}_{j,k} = [0\ 0\ 1]^\top. \tag{18}$$

Similarly to the method in [40], (P_3) is valid for $\alpha < 90°$. For derivation details of the camera translation direction constraints (15c) please refer to [40].

IV. Results

Here we compare L-infinity SLAM (Algorithm 2) against BA-SLAM (Algorithm 1) on real data with precise ground truth. The used dataset, provided by Maptek[3], was captured with a system equipped with a high precision INS (refer to [23] for system's details). Mounted on a truck, a forward looking camera captured a video together with inertial measurements providing the ground truth for the camera poses.

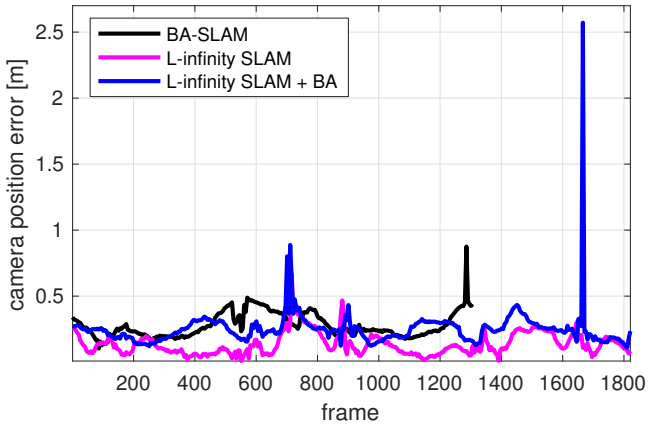
Experiments were run on a PC with a quad-core 2.5GHz Intel core i7 CPU and 16GB of RAM. We implemented L-infinity SLAM and BA-SLAM in MATLAB with the following optimisation routines:

- BA: implemented in C++ using the Ceres solver [41].
- rotation_averaging: code provided in [33].
- known_rotation_prob: code provided in [39].
- krot_tdc: (P_3) implemented in MATLAB using SeDuMi [42].

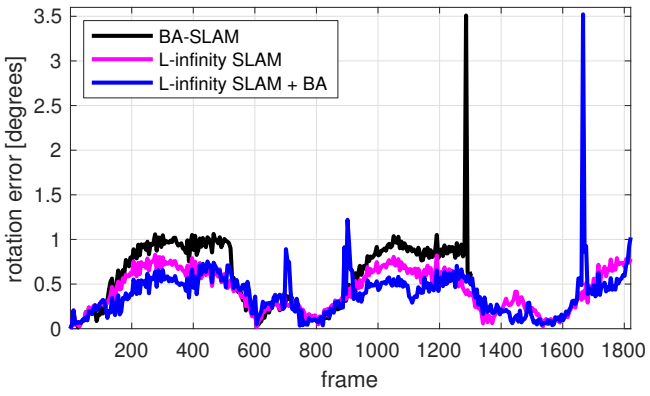
A. Results for the Maptek dataset

We sampled the full sequence (1833 frames) into 358 keyframes. We detected the occurrence of a loop by using provided ground truth in both BA-SLAM and L-infinity SLAM. Since the moving camera describes a two-loop sequence (see the ground truth in Fig. 2), after completing the first loop (at frame 790), a loop is detected for each consecutive keyframe. To solve loop closure in L-infinity SLAM, we fed krot_tdc with 300 uniformly sampled feature tracks (we used the same sample size for loop closure in BA-SLAM). krot_tdc accurately solved loop closure (see Fig 1) in 90.54 s in a MATLAB single-thread implementation. Since its quasi-convex nature, krot_tdc does not have to be invoked

[3] https://www.maptek.com/



(a)

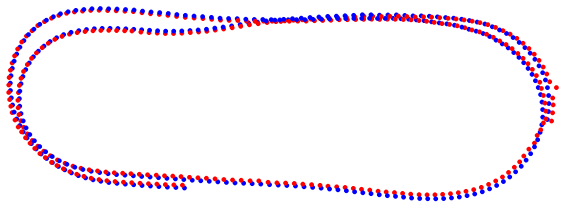


(b)

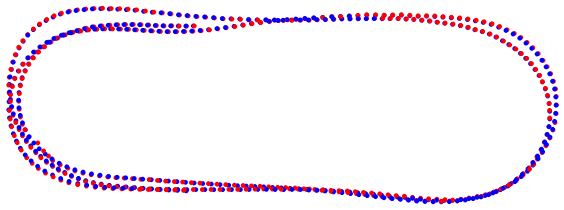
Fig. 1: (a) Camera position error, and (b) camera rotation error for BA-SLAM and L-infinity SLAM in the Maptek dataset. The comparison includes the result of BA after krot_tdc fed with same feature tracks.

at each loop detection; here we invoked krot_tdc at the last keyframe only. On the contrary, BA-SLAM could be incapable of fixing the drift produced if invocations of BA are skipped (e.g., failing in detecting a loop on a real-world system). We set a window size equal to 10.

To compare BA-SLAM and L-infinity SLAM, Fig. 1 plots the camera position error and camera rotation error for both methods. BA-SLAM was unable to complete the sequence –the camera got disconnected at frame 1356, i.e., no feature track existed for the visited keyframe. This disconnection was a consequence of the outlier removal heuristic used for BA-SLAM (removing a feature track if the distance of the camera position to any visible 3D point is above a threshold; 150 m for this experiment) which failed by eliminating inliers when BA was unable to produce an accurate result. We observed that BA is prone to fail on reduced data input, hence BA-SLAM needs to keep a large number of feature tracks. The l_∞ optimisation approach of L-infinity SLAM admits less risky outlier removal strategies (e.g. eliminating the support set) without the need of keeping a large number of tracks. As result, BA-SLAM achieved lower camera position ($< 0.67\ m$) and camera rotation ($< 0.83°$) errors than BA-SLAM.



(a) BA-SLAM



(b) L-infinity SLAM with krot_tdc

Fig. 2: Estimated camera positions (red dots) superposed with the ground truth (blue dots) for (a) BA-SLAM, and (b) L-infinity SLAM solving loop closure with the proposed krot_tdc.

Fig. 3: Camera positions obtained with method in [40].

We also ran BA after krot_tdc on same feature tracks. The camera position and rotation error tend to be lowered; however, BA produced significant error in several camera poses as depicted in Fig. 1.

1) krot_tdc vs the camera recovering method in [40]: We ran the method in [40] with the same relative translation directions $\mathbf{t}_{j,k}^{(E)}$ and camera rotations used to solve loop closure. As depicted in Fig. 3, recovering camera positions is unachievable from $\mathbf{t}_{j,k}^{(E)}$ measurements only. In addition, Fig. 4 shows known_rotation_prob failed on producing an accurate result when solving loop closure on the same tracks we used with krot_tdc.

2) Map reconstruction: Since we reconstructed 300 scene points only when solving loop closure, we used the quasi-convex method in [39] to triangulate all scene points. Fig. 6 shows the scene reconstruction and the camera positions.

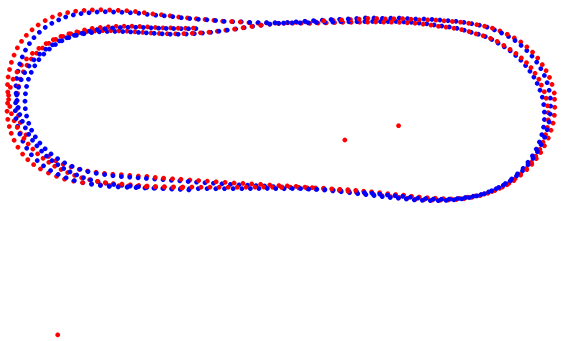
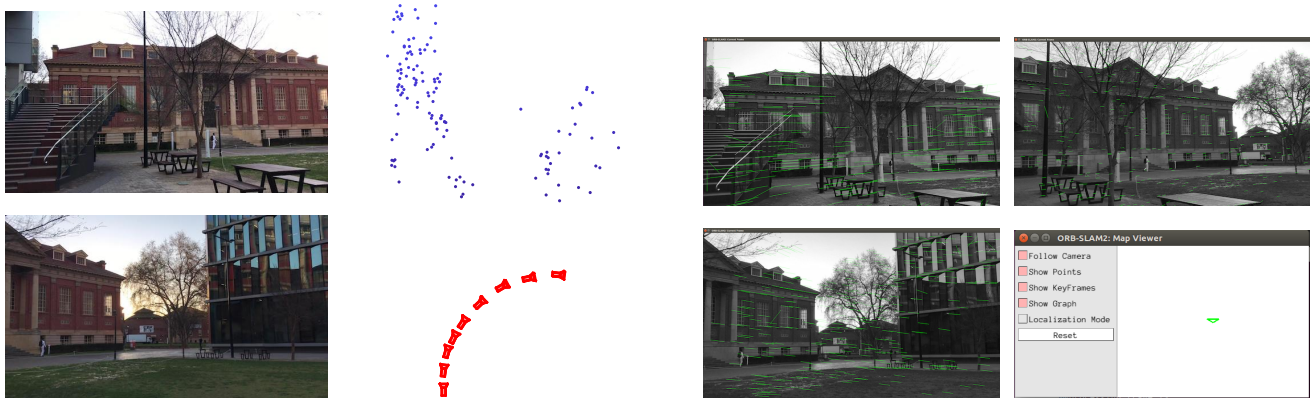


Fig. 4: Superposition of known_rotation_prob camera positions (red dots) with the ground truth (blue dots).



(a) Input (b) L-infinity SLAM (c) ORB-SLAM

Fig. 5: A pedestrian recorded a scene while walking and rotating a smart-phone camera. (a) Frame samples of the input video. (b) L-infinity SLAM scene reconstruction. (c) ORB-SLAM failed to initialise hence no reconstruction was possible. Green lines indicate unsuccessful initialisation. The bottom right screenshot displays result (blank) at the end of the sequence.

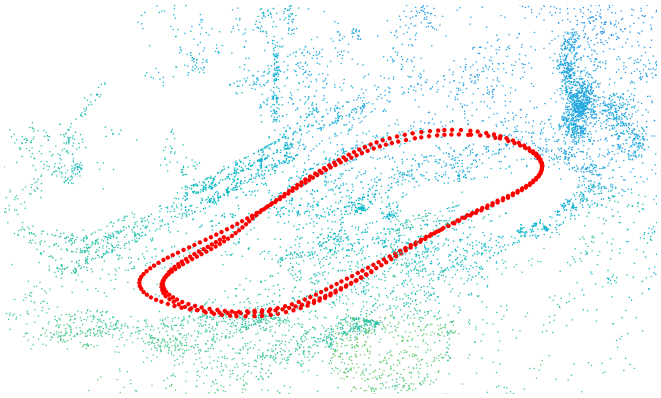


Fig. 6: L-infinity SLAM reconstruction.

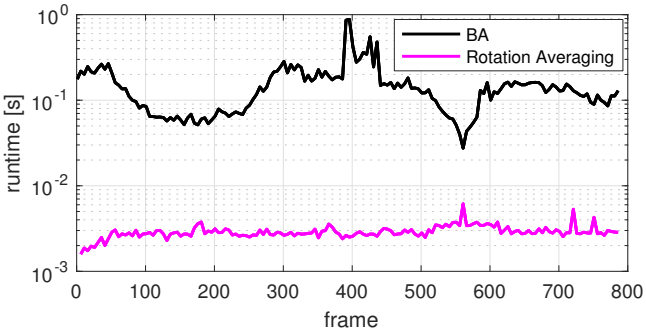


Fig. 7: Runtime comparison of incremental rotation averaging and BA.

B. *Low speed and high rotational motion*

We tested L-Infinity SLAM against ORB-SLAM with a video recorded with a smart-phone by a pedestrian while turning right; Fig. 5a displays two frames of the video. ORB-SLAM failed to build an initial map (see Fig. 5c), arguably, since the low baseline of frames from the walking speed sequence with rotational motion (see camera poses in Fig. 5b). Unlike ORB-SLAM, the quasi-convex formulation of L-Infinity SLAM does not required any initial map to track camera motions. As result, only L-infinity SLAM produced a reconstruction (see Fig. 5b).

C. *Runtime of online routines*

To compare the efficiency L-infinity SLAM against BA-SLAM for rotation only camera motions, we measured the runtime of incremental rotation averaging and incremental BA, which are the fundamental optimisation routines for this problem. We used a window size equal to 10 and plotted the runtimes for the first loop of the Maptek dataset in Fig. 7 (in log scale). Rotation averaging is an order of magnitude faster than BA which indicates the superior efficiency of L-infinity SLAM over BA-SLAM for rotation only problems.

V. CONCLUSIONS

We presented L-infinity SLAM to be a simpler alternative to SLAM systems based on bundle adjustment. Driven by globally optimal quasi-convex optimisation, there is no need to maintain an accurate map and camera motions at key-frame rate as demanded by systems based on bundle adjustment. Instead, the online effort is devoted to efficiently estimating camera orientations through rotation averaging. To efficiently solve loop closure, we proposed a variant of the known rotation problem which incorporates relative translation directions to accurately solve camera drifts when optimising over a sample of feature tracks. Also, L-infinity SLAM is a simple and efficient alternative for applications requiring estimating slow motions or only rotational motions. We hope L-infinity SLAM can motivate future research on quasi-convex optimisation in the SLAM community.

ACKNOWLEDGEMENT

This work was supported by ARC Grants DP160103490 and CE140100016.

REFERENCES

[1] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual SLAM: why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[2] Y. Kanazawa and K. Kanatani, "Do we really have to consider covariance matrices for image features?" in *ICCV*, 2001.

[3] K. Kanatani, "Uncertainty modeling and model selection for geometric inference," *IEEE TPAMI*, vol. 26, no. 10, pp. 1307–1319, 2004.

[4] ——, "Statistical optimization for geometric fitting: theoretical accuracy bound and high order error analysis," *IJCV*, vol. 80, pp. 167–188, 2008.

[5] A. J. Davison, I. D. Reid, N. M. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM," *IEEE TPAMI*, vol. 29, no. 6, pp. 1–16, 2007.

[6] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," *Photogrammetric Computer Vision*, vol. 2, 2006.

[7] G. Klein and D. W. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, 2007.

[8] R. Mur-Artal, J. Montiel, and J. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE TRO*, vol. 31, no. 5, pp. 1147–1163, 2015.

[9] G. Grisetti, C. Stachniss, and W. Burgard, "Non-linear constraint network optimization for efficient map learning," *IEEE Trans. on Intelligence Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[10] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: a general framework for graph optimization," in *ICRA*, 2011.

[11] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dallaert, "iSAM2: incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, 2012.

[12] L. Carlone, D. Rosen, G. Calafiore, J. J. Leonard, and F. Dellaert, "Lagrangian duality in 3D SLAM: verification techniques and optimal solutions," in *IROS*, 2015.

[13] J. Engel, "Tutorial on geometric and semantic 3D reconstruction, CVPR 2017," https://people.eecs.berkeley.edu/~chaene/cvpr17tut/.

[14] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Trans. on Computer Vision and Applications*, vol. 9, no. 16, 2017.

[15] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer, "Live tracking and mapping from both general and rotation-only camera motion," in *ISMAR*, 2012.

[16] C. Pirchheim, D. Schmalstieg, and G. Reitmayr, "Handling pure camera rotation in keyframe-based SLAM," in *ISMAR*, 2013.

[17] C. Herrera, K. Kim, J. Kannala, K. Pulli, and J. Heikkilä, "DT-SLAM: deferred triangulation for robust SLAM," in *3DV*, 2014.

[18] C. Tang, O. Wang, and P. Tan, "GSLAM: initialization-robust monocular visual SLAM via global structure-from-motion," in *3DV*, 2017.

[19] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *IJCV*, vol. 130, no. 3, pp. 267–305, 2013.

[20] A. Eriksson, C. Olsson, F. Kahl, and T.-J. Chin, "Rotation averaging and strong duality," in *CVPR*, 2018.

[21] F. Kahl, "Multiple view geometry and the L_∞-norm," in *ICCV*, 2005.

[22] C. Olsson, A. Eriksson, and F. Kahl, "Efficient optimization for l_∞-problems using pseudoconvexity," in *ICCV*, 2007.

[23] A. Khosravian, T.-J. Chin, I. Reid, and R. Mahony, "A discrete-time attitude observer on SO(3) for vision and GPS fusion," in *ICRA*, 2017.

[24] L. Kneip and H. Li, "Efficient computation of relative pose for multi-camera systems," in *CVPR*, 2014, pp. 446–453.

[25] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization," in *ICRA*, 2015.

[26] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John Wiley and Sons, 1987.

[27] Q. Ke and T. Kanade, "Quasiconvex optimization for robust geometric reconstruction," *TPAMI*, vol. 29, no. 10, 2007.

[28] K. Sim and R. Hartley, "Removing outliers using the L_∞ norm," in *CVPR*, 2006.

[29] J. Ventura, C. Arth, and V. Lepetit, "An efficient minimal solution for multi-camera motion," in *ICCV*, 2015, pp. 747–755.

[30] H. Ha, T.-H. Oh, and I. S. Kweon, "A closed-form solution to rotation estimation for structure from small motion," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 393–397, 2018.

[31] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *ICPR*, vol. 3. IEEE, 2002, pp. 545–548.

[32] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the Weiszfeld algorithm," in *CVPR*, 2011.

[33] A. Chatterjee and V. Madhav Govindu, "Efficient and robust large-scale rotation averaging," in *ICCV*, 2013.

[34] F. Kahl and R. Hartley, "Multiple-view geometry under the L_∞-norm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1603–1617, 2008.

[35] S. Agarwal, N. Snavely, and S. M. Seitz, "Fast algorithms for l_∞ problems in multiview geometry," in *CVPR*, 2008.

[36] M. Gugat, "A fast algorithm for a class of generalized fractional programs," *Management Science*, vol. 42, no. 10, pp. 1493–1499, 1996.

[37] R. Brent, *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[38] W. Dinkelbach, "On nonlinear fractional programming," *Management science*, vol. 13, no. 7, pp. 492–498, 1967.

[39] Q. Zhang, T.-J. Chin, and H. M. Le, "A fast resection-intersection method for the known rotation problem," in *CVPR*, 2018.

[40] K. Sim and R. Hartley, "Recovering camera motion using L_∞ minimization," in *CVPR*, 2006, pp. 1230–1237.

[41] S. Agarwal and K. Mierle, *Ceres Solver: Tutorial & Reference*, Google Inc.

[42] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1-4, pp. 625–653, 1999.