Relative Bundle Adjustment

Gabe Sibley

{gsibley@robots.ox.ac.uk}

Technical Report No. 2307/09

January 27, 2009

Robotics Research Group
Department of Engineering
University of Oxford
Parks Road, OX13PJ

Abstract

This report derives a relative objective function for bundle adjustment – driven by the desire for a truly large scale simultaneous localization and mapping algorithm that can operate incrementally in constant time. It is precisely the choice of a single privileged coordinate frame that makes bundle adjustment expensive to solve. This is especially true during loop closures, when the single frame approach necessitates adjusting all parameters in the loop. We give a relative formulation that is designed specifically to avoid the cost of optimizing all parameters at loop closure. Instead of optimizing in a single Euclidean space, relative bundle adjustment works in a metric-space defined by a connected Riemannian manifold. We find evidence that in this space, the global maximum likelihood solution can be found incrementally in constant time – even at loop closure.

1 Introduction

Bundle adjustment is the optimal solution to the so-called "full" simultaneous localization and mapping problem, in that it solves for the maximum likelihood solution given all measurements over all time. The goal in bundle adjustment is to minimize error between observed and predicted image-measurements of n 3D landmarks sensed from m 6D sensor poses (or frames) [31]. Measurements and parameter estimates are usually considered to be normally distributed, and the problem is typically tackled with non-linear least-squares optimization routines like Levenberg–Marquardt or the Gauss-Newton method. The linearized system matrix that appears in this process matches the form of the Fisher Information matrix, which in turn defines the Cramer Rao Lower Bound that is used to assess estimator consistency and optimality. It is not surprising therefore that bundle adjustment is the optimal non-linear least-squares simultaneous localization and mapping algorithm.

The cost of optimizing the bundle adjustment objective-function is cubic in complexity (in either m or n). For large and growing problems, this can quickly become prohibitive. This is especially true during loop-closure, when often all parameters in the loop must be adjusted. In a single coordinate frame, the farther the robot travels from the origin, the larger position uncertainty becomes. Errors at loop closure can therefore become arbitrarily large, which in turn makes it impossible to compute the *full* maximum likelihood solution in constant time (here the "full" solution is the one that finds the optimal estimates for all parameters).

It is not clear that it is *necessary* to estimate everything in a single coordinate frame – for instance most problems of autonomous navigation, such as path planning, obstacle avoidance or object manipulation, can be addressed within the confines of a *metric manifold*. Taking this route, we structure the problem as a graph of relative poses with landmarks specified in relation to these poses. In 3D this graph defines a connected Riemannian manifold of dimension six with a distance metric based on shortest paths. Note that this is not a sub-mapping approach – there are no distinct overlapping estimates, and there is only one objective function with a *minimal* parameter vector; similarly, this is not a pose-graph relaxation approach.

Together with an adaptive optimization scheme that only ever solves for a small sub-portion of the state vector, we find evidence that the full maximum likelihood solution in the manifold can be found using an incrementally constant time algorithm. Crucially, this appears true *even at loop closure*.

We stress at the outset that the relative solution is not equivalent to the normal Euclidean-space solution and it does not produce an estimate that can be easily embedded in a single Euclidean frame. Projecting from the relative manifold into a single Euclidean space is a difficult problem that we argue is best handled by external resources that do not have constant run-time requirements - e.g. by operator computers, not on the robot.

2 Related Work

There has been much interest in Gaussian non-linear least-squares solutions to SLAM based on "full-SLAM" or bundle adjustment [29][31][8][12][19], though the problem is an old one [3][22]. The full SLAM problem tries to optimize the joint vehicle trajectory and map structure simultaneously given all measurements ever made. There are approximate incremental solutions that only optimize a small local subset of the map [7], and there are methods that approximate the full solution with various forms of marginalization [19][27], or by ignoring small dependency information [30][21]. Recently some have successfully employed techniques from the linear algebra and numerical optimization communities to greatly reduce the cost of finding the full solution [17]. Many use key-frames to reduce complexity, though at the expense of accuracy [10][23][18]. All these techniques suffer from computational complexity issues during loop closures.

In the context of long term autonomy, roboticists recognize the need for online, real-time, navigation and mapping algorithms. This means that localization and mapping algorithms must operate incrementally within a constant-time budget. Driven by this need, many authors have recognized the benefit of relative representations [2][9][19][1][15][4][13][20]. The most common solution is probably sub-mapping [2][25][6][9], which breaks the estimation into many smaller mapping regions, computes individual solutions for each region, and then estimates the relationships between these sub-maps. Many difficult issues arise in sub-mapping, including map overlap, data duplication, map fusion and breaking, map alignment, optimal sub-map size, and consistent global estimation in a single Euclidean frame. The relative bundle adjustment we propose can be seen as a *continuous* sub-mapping approach that avoids these complications.

To solve large SLAM problems with many loops, the most successful methods currently are the pose-graph optimization algorithms. Instead of solving the full SLAM problem, these methods optimize a set of relative pose constraints [24][14]. This is attractive because using forward substitution it is possible to transform full SLAM into a generally sparse set of pose constraints [11][29], and even to make the resulting system of equations relative [19]. Note that, given the assumed Gaussian problem structure, this kind of forward substitution to a pose-graph is algebraically equivalent to marginalization; methods that marginalize landmark parameters onto pose parameters so as to define a pose-graph are executing the forward substitution phase of sparse bundle adjustment. In this light, pose-graph relaxation, which solves for the optimal path estimate, can be seen as one-half of one iteration of full SLAM, because full SLAM also back-substitutes for the map parameters, and iterates the procedure to convergence. Like other methods, pose-graph solvers have worst-case complexity at loop closure that scales with the length of the loop.

The work most similar to relative bundle adjustment is the relative formulations given by Eade [9] and Konolige [19]. The former is akin to sub-mapping methods with constraints to enforce global Euclidean consistency at loop closure; the latter formulates the cost function relative to a single Euclidean frame and then makes a series of approximations to produce a sparse relative pose-graph. Neither method derives the purely relative objective function (incrementally, both rely on some form of single-reference frame), neither formulates the objective function completely without privileged frames, and both methods carry the burden of finding a globally consistent estimate in a single Euclidean frame. Our approach is substantially different because of the completely relative underlying objective function that we derive.

Finally, a number of adaptive region approaches have been explored using global methods [28][26]. These techniques, together with all of the methods presented in this section, are at least linear in complexity in the length of their largest loop – none is constant time at loop closure, and all but one [2] solve for a solution in a single Euclidean space.

3 Problem Formulation

The goal in bundle adjustment is to minimize error between the observed and predicted image-measurements of n 3D landmarks sensed from m 6D sensor poses. Likewise, the method presented here will minimize the difference between predicted and measured values.

- Let $p=\{p_j,\ j=0,....,m\text{-}1\}$ be a set of m 6D relative vehicle poses. Each $p_j \in \mathbb{SE}3$ is a 6×1 vector that defines a 4×4 homogeneous transformation matrix, $T_{\alpha j}$ — this is "pose j in frame α".

- Let $l=\{\bar{l}_{jk},\ k=1,...,n,\ j\in\ [0,...,m\text{-}1]\}$ be a set of n 3D landmarks each parameterized relative to some *base-frame* j. Each $l_{jk} = \begin{bmatrix} \bar{l}_{jk} \\ 1 \end{bmatrix}$ is a 4×1 homogeneous point; the bar notation selects the 3D component.

- Let $t=\{t_j,\ j=1,....,M\}$ be a set of M 6D transform estimates. Each $t_j \in \mathbb{SE}3$ is a 6×1 error-state parameter. The set contains m-1 edge

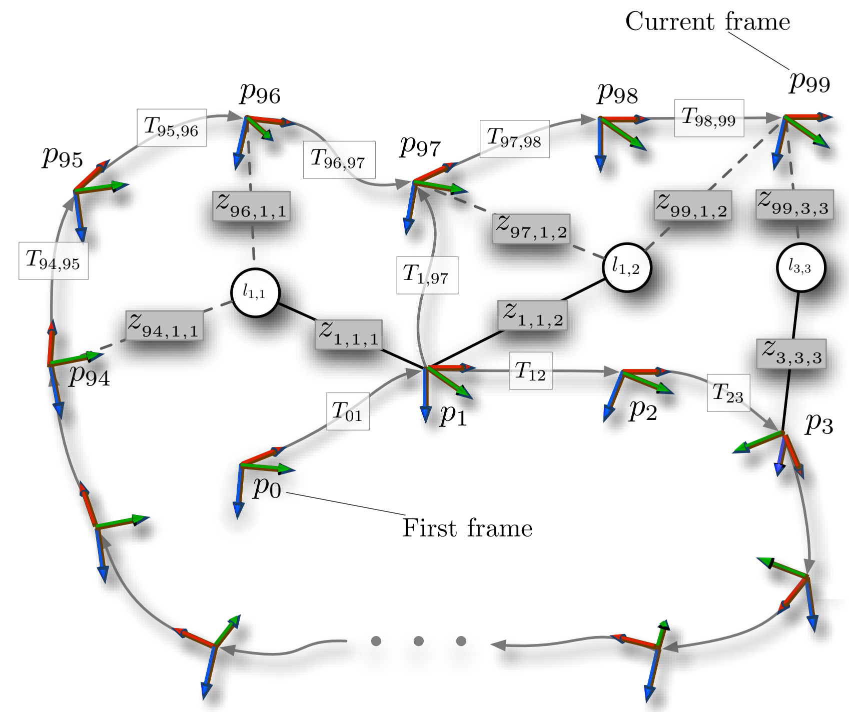


Figure 1: Example trajectory starting at the first frame, p_0, and ending at the current frame, p_{99}. The loop closure between frame 1 and frame 97 adds an extra edge to the graph. Landmark base-frames are indicated with solid lines. Each transform includes an infinitesimal delta-transform defined about $t_j = 0$ — that is, $T_{\alpha j}$=$\hat{T}_{\alpha j}T_{(t_j)}$, where $\hat{T}_{\alpha j}$ is the current estimate of the relative transform between frame α and frame j.

constraints along the vehicle path, and $\mathcal{L}$ loop-closure edge constraints (M=m-1+$\mathcal{L}$).

- The full state vector, $x = [l, t]^T$, grows as new frames and landmarks are added.
- Let z_{ijk} denote a measurement made in frame i of a landmark k that is stored relative to some *base-frame* j.

3.1 Relative Graph Representation

The relative transforms in this formulation are associated with edges in an undirected graph of frames. The graph is built incrementally as the vehicle moves through the environment, and extra edges are added during loop closure. The graph defines a connected Riemannian manifold that is by definition everywhere locally Euclidean, though globally it is not embedded in a single Euclidean space. The relationship between parent-frame α and child-frame j is defined by a 4×4 homogeneous transform matrix, $T_{\alpha j}{=}\hat{T}_{\alpha j}T_{(t_j)}$, where $\hat{T}_{\alpha j}$ is the current estimate. An example trajectory and graph with this notation is shown in Figure 1.

In order to predict a measurement of landmark l_{jk} in frame i, the landmark estimate must be transformed along the kinematic chain from j to i. This is defined by a composition of 4×4 homogeneous transforms

$$T_{ji} \quad = \quad \hat{T}_{j,j+1}T_{(t_{j+1})}\hat{T}_{j+1,j+2}T_{(t_{j+2})}, ..., \hat{T}_{i-1,i}T_{(t_i)}.$$

Traversal of kinematic chains like this is what distinguishes relative bundle adjustment from the traditional approach.

3.2 Sensor Model

The sensor model for a single measurement that describes how landmark k, stored relative to base-frame j, is transformed into frame i and projected into the sensor, is

$$\begin{aligned} h_{ijk}(x) &= Proj\left(T_{ji}^{-1} l_{jk}\right) \\ &= Proj\left(g_{ijk}(x)\right) \\ &= \mathcal{K}\left(\mathcal{M}(g_{ijk}(x)\right). \end{aligned}$$

The projection function $Proj$ is expanded in terms of the functions $\mathcal{K}$ and $\mathcal{M}$ to facilitate multi-camera rigs where the cameras are not coincident with the body of interest (but are instead related to it by a known vehicle-to-sensor transform, T_{is}). Further,

- $g_{ijk} : \mathbb{R}^{\dim(x)} \rightarrow \mathbb{R}^4$, $x \mapsto T_{ji}^{-1} l_{jk}$ transforms l_{jk} from base-frame j to the observation frame i.
- $\mathcal{M} : \mathbb{R}^4 \rightarrow \mathbb{R}^3$, $p \mapsto [\mathbf{M},\ 0]T_{is}^{-1}p$ transforms a point p from vehicle frame i to sensor frame s (useful for modeling multiple cameras and cameras not coincident with the vehicle frame, such as stereo systems). T_{is} is the sensor in the vehicle frame. The 3×3 matrix $\mathbf{M}$ is just a signed-permutation matrix that converts from the problem specific coordinate-frame (*fixed-frame xyz-Euler* angles in most robotics applications, which has $+x$ forward, $+y$ right and $+z$ down) to the computer-vision coordinate-frame (which has $+x$ right, $+y$ down and $+z$ forward). This function also projects from homogeneous to Euclidean coordinates.
- $\mathcal{K} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, is standard perspective projection via a 3×3 camera calibration matrix, K.

Assuming measurements z_{ijk} are normally distributed, $z_{ijk} \sim N(h_{ijk}(x), R_{ijk})$, then the non-linear least-squares objective function for relative bundle adjustment is

$$\begin{aligned} J &= \sum_{k=1}^{n} \sum_{i \in 1}^{m_k} \left(z_{ijk} - h_{ijk}(x)\right)^T R_{ijk}^{-1} \left(z_{ijk} - h_{ijk}(x)\right) \qquad (1) \\ &= \left\| z - h(x) \right\|_{R^{-1}}, \end{aligned}$$

which depends on the landmark estimate, l_{jk}, *and all the estimates* $t_{j+1}, ..., t_i$ *on the kinematic chain from the base-frame* j *to the measurement frame* i. This problem is solved using iterative non-linear least-squares Gauss-Newton minimization for the values of x that minimize re-projection error — this yields the *maximum likelihood estimate* (subject to local minima). Projecting via kinematic chains like this is novel, but as we will see it changes the sparsity patterns in the system Jacobian. Compared to normal bundle adjustment, this new pattern increases the cost of solving the sparse normal equations for updates δx to the state vector x — though the ultimate computational complexity is the same.

In general, there are multiple paths from frame j to frame i, which means that there might be multiple ways to project a landmark l_{jk} into frame i, and we therefore have to select which path to use. A simple solution is to always select the shortest path possible. If each edge has its weight set to the determinant of the edge transform uncertainty, then Dijkstra's algorithm will define the shortest chains with minimal transformation uncertainty.

3.3 Sparse Solution

The *normal equations* associated with the iterative non-linear least-squares Gauss-Newton solution to equation (1) are

$$H^T R^{-1} H \delta x = H^T R^{-1}(z - h(x)). \qquad (2)$$

where $H = \frac{\partial h}{\partial x}$ is the Jacobian of the sensor model, R is the block diagonal covariance matrix describing the uncertainty of the collective observation z (the stacked vector of all measurements). Referring to the example problem in Figure 3 we see that $H^T = \left[H_l^T H_t^T\right]$ and $\delta x = [\delta l, \delta t]^T$, which exposes the well known 2×2 block structure of equation (2),

$$\begin{bmatrix} V & W \\ W^T & U \end{bmatrix} \begin{bmatrix} \delta l \\ \delta t \end{bmatrix} = \begin{bmatrix} r_l \\ r_t \end{bmatrix}.$$

Here δl and δt are state-updates for the map and edge-transforms, and

$$r_l = H_l^T R^{-1}(z - h(x)),$$

$$r_t = H_t^T R^{-1}(z - h(x)),$$

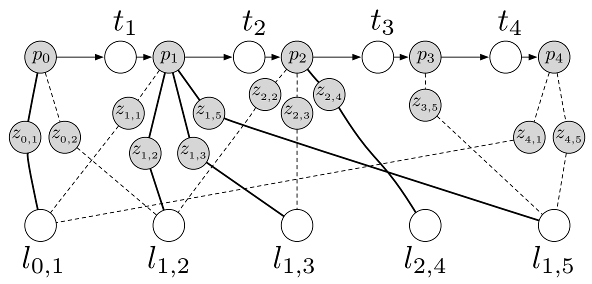


Figure 2: Graphical example for the sequence of 12 observations, z_{001}, z_{012}, z_{101}, z_{112}, z_{113}, z_{115}, z_{212}, z_{213}, z_{224}, z_{315}, z_{401}, and z_{415}. There are five poses, $p_{0,\ldots,4}$, four edge estimates $t_{1,..,4}$, and five landmarks l_{01}, l_{12}, l_{13}, l_{24} and l_{15}. This example has the Jacobian $H = \frac{\partial h}{\partial x}$ that is depicted in Figure 3. Bold lines from poses indicate which frames are base-frames.

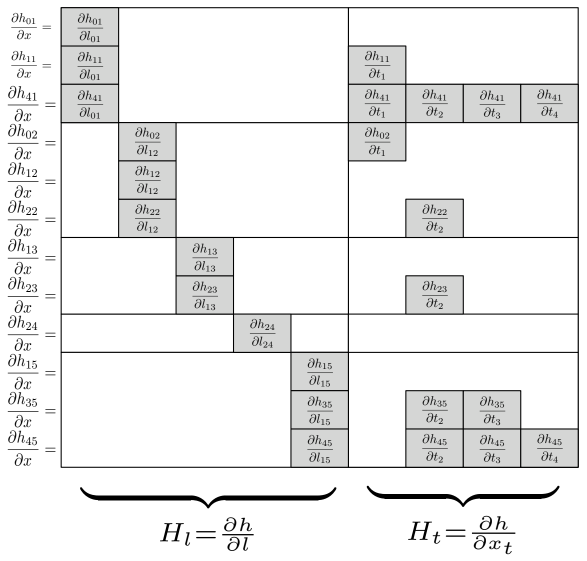

$\frac{\partial h_{01}}{\partial x} =$
$\frac{\partial h_{11}}{\partial x} =$
$\frac{\partial h_{41}}{\partial x} =$
$\frac{\partial h_{02}}{\partial x} =$
$\frac{\partial h_{12}}{\partial x} =$
$\frac{\partial h_{22}}{\partial x} =$
$\frac{\partial h_{13}}{\partial x} =$
$\frac{\partial h_{23}}{\partial x} =$
$\frac{\partial h_{24}}{\partial x} =$
$\frac{\partial h_{15}}{\partial x} =$
$\frac{\partial h_{35}}{\partial x} =$
$\frac{\partial h_{45}}{\partial x} =$
$\frac{\partial h_{01}}{\partial l_{01}}$
$\frac{\partial h_{11}}{\partial l_{01}}$
$\frac{\partial h_{11}}{\partial t_{1}}$
$\frac{\partial h_{41}}{\partial l_{01}}$
$\frac{\partial h_{41}}{\partial t_{1}}$
$\frac{\partial h_{41}}{\partial t_{2}}$
$\frac{\partial h_{41}}{\partial t_{3}}$
$\frac{\partial h_{41}}{\partial t_{4}}$
$\frac{\partial h_{02}}{\partial l_{12}}$
$\frac{\partial h_{02}}{\partial t_{1}}$
$\frac{\partial h_{12}}{\partial l_{12}}$
$\frac{\partial h_{22}}{\partial l_{12}}$
$\frac{\partial h_{22}}{\partial t_{2}}$
$\frac{\partial h_{13}}{\partial l_{13}}$
$\frac{\partial h_{23}}{\partial l_{13}}$
$\frac{\partial h_{23}}{\partial t_{2}}$
$\frac{\partial h_{24}}{\partial l_{24}}$
$\frac{\partial h_{15}}{\partial l_{15}}$
$\frac{\partial h_{35}}{\partial l_{15}}$
$\frac{\partial h_{35}}{\partial t_{2}}$
$\frac{\partial h_{35}}{\partial t_{3}}$
$\frac{\partial h_{45}}{\partial l_{15}}$
$\frac{\partial h_{45}}{\partial t_{2}}$
$\frac{\partial h_{45}}{\partial t_{3}}$
$\frac{\partial h_{45}}{\partial t_{4}}$
$H_l = \frac{\partial h}{\partial l}$
$H_t = \frac{\partial h}{\partial x_t}$

Figure 3: Example relative bundle adjustment Jacobian structure for the sequence of 12 observations in Figure 2. Grey indicates non-zero entries. The horizontal stripes in the right hand H_t term above correspond to projections that rely on transforming state estimates along kinematic chains from frame j to frame i. These stripes are the only difference in sparsity pattern between the relative formulation and traditional bundle adjustment.

$$V = H_l^T R^{-1} H_l,$$

$$W = H_l^T R^{-1} H_t,$$

and

$$U = H_t^T R^{-1} H_t.$$

Building these linearized equations is the dominant cost in solving each Gauss-Newton iteration, which makes it important to compute the sparse Jacobian of h efficiently.

3.4 Relative Jacobians

This section lists in detail how to compute the relative bundle adjustment Jacobians efficiently. Due to the functional dependence of the projection model on the kinematic chain of relative poses, the Jacobian in the relative formulation is very different from its Euclidean counterpart. With reference to Figure 4, focus for a moment on a single infinitesimal transform $T_{(t_c)}$ that is somewhere along the kinematic chain from frame i to j (note the direction). The individual terms shown in Figure 3 are

$$\frac{\partial h_{ijk}}{\partial \bar{l}_{jk}} = \frac{\partial \mathcal{K}}{\partial \mathcal{M}} \frac{\partial \mathcal{M}}{\partial g_{ijk}} \frac{\partial g_{ijk}}{\partial \bar{l}_{jk}},$$

and

$$\frac{\partial h_{ijk}}{\partial t_c} = \frac{\partial \mathcal{K}}{\partial \mathcal{M}} \frac{\partial \mathcal{M}}{\partial g_{ijk}} \frac{\partial g_{ijk}}{\partial t_c}.$$

Computing these requires $\frac{\partial \mathcal{K}}{\partial \mathcal{M}}$, $\frac{\partial \mathcal{M}}{\partial g_{ijk}}$, $\frac{\partial g_{ijk}}{\partial \bar{l}_{jk}}$, and $\frac{\partial g_{ijk}}{\partial t_c}$, which we derive next.

3.4.1 Perspective Projection Jacobian

The perspective projection function using the camera calibration matrix, K, is

$$K\mathcal{M} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & s_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathcal{M}_1 \\ \mathcal{M}_2 \\ \mathcal{M}_3 \end{bmatrix},$$

which is scaled by w to get the final 2D pixel location

$$\mathcal{K}(\mathcal{M}) = \begin{bmatrix} u \\ v \end{bmatrix} / w = a(\mathcal{M})/b(\mathcal{M}) = \begin{bmatrix} (f_x\mathcal{M}_1 + s_x\mathcal{M}_2 + c_x\mathcal{M}_3) \\ (f_y\mathcal{M}_2 + c_y\mathcal{M}_3) \end{bmatrix} / \mathcal{M}_3.$$

The Jacobian of $\mathcal{K}(\mathcal{M})$ is thus

$$\frac{\partial \mathcal{K}}{\partial \mathcal{M}} = \frac{b(\mathcal{M})a'(\mathcal{M}) - a(\mathcal{M})b'(\mathcal{M})}{b(\mathcal{M})^2}$$

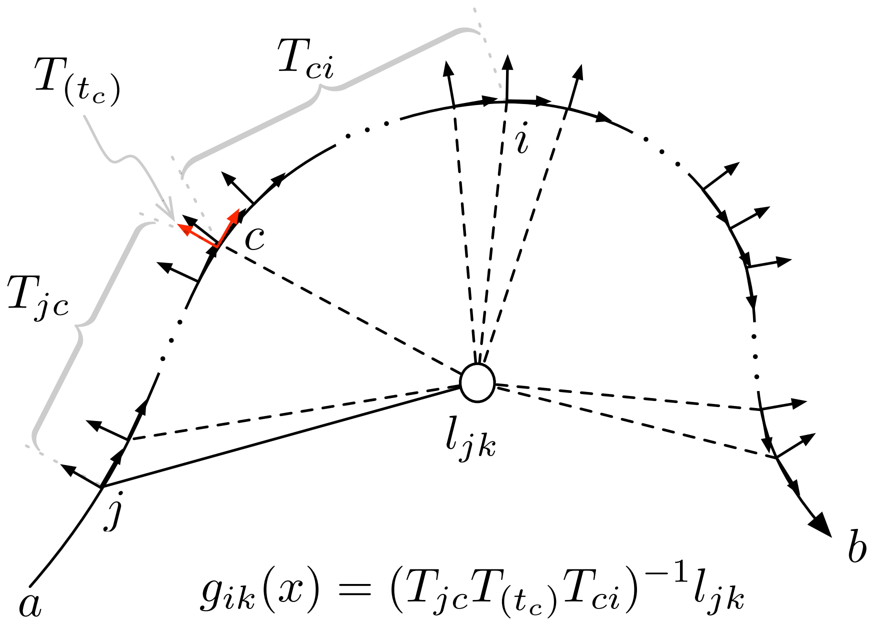


Figure 4: This diagram shows the sensor following a path from a to b while making measurements of landmark l_{jk} (indicated with dashed lines). Landmark k is stored relative to frame j (indicated by a solid line). To compute the projection of landmark k in frame i, we evaluate $h_{ijk} = Proj(g_{ijk}(x))$, where $g_{ijk}(x) = T_{ji}^{-1}l_{jk} = \left(T_{j,j+1}T_{(t_{j+1})}T_{j+1,j+2}T_{(t_{j+2})}, ..., T_{i-1,i}T_{(t_i)}\right)^{-1} l_{jk}$ encapsulates projection along the kinematic chain between frame j and frame i. To help understand how the relative formulation Jacobian is computed, this diagram focuses on the the delta transform $T_{(t_c)}$ indicated in red. The state-vector terms of interest when computing derivatives are 1) the transform parameters t_c, and 2) the landmark parameters l_{jk}.

where

$$a'(\mathcal{M}) = \frac{\partial a}{\partial \mathcal{M}} = \begin{bmatrix} f_x & s_x & c_x \\ 0 & f_y & c_y \end{bmatrix} \qquad \text{and } b'(\mathcal{M}) = \frac{\partial b}{\partial \mathcal{M}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

leading to

$$\begin{aligned}
\frac{\partial \mathcal{K}}{\partial \mathcal{M}} &= \left(\mathcal{M}_3 \begin{bmatrix} f_x & s_x & c_x \\ 0 & f_y & c_y \end{bmatrix} - \begin{bmatrix} (f_x\mathcal{M}_1 + s_x\mathcal{M}_2 + c_x\mathcal{M}_3) \\ (f_y\mathcal{M}_2 + c_y\mathcal{M}_3) \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \right) / \mathcal{M}_3^2 \\
&= \left(\begin{bmatrix} f_x & s_x & c_x \\ 0 & f_y & c_y \end{bmatrix} - \begin{bmatrix} 0 & 0 & u \\ 0 & 0 & v \end{bmatrix} \right) / \mathcal{M}_3 \\
&= \begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix} / \mathcal{M}_3. \qquad (3)
\end{aligned}$$

3.4.2 Convention-Configuration Jacobian

The Jacobian of $\mathcal{M}$ is trivial; we list it for completeness:

$$\frac{\partial \mathcal{M}}{\partial g_{ijk}} = [\mathbf{M},\ 0] T_{is}^{-1}. \qquad (4)$$

3.4.3 3D Point Jacobian

The 4×3 Jacobian of g_{ijk} with respect to a 3D point $\bar{l}_{jk}$ is

$$\begin{aligned}
\frac{\partial g_{ijk}}{\partial \bar{l}_{jk}} &= T_{ij}^{-1}[1,1,1,0]^T \\
&= \begin{bmatrix} R_{ji}^T \\ 0 \end{bmatrix}. \qquad (5)
\end{aligned}$$

where R_{ij} is the rotation matrix in T_{ij}.

3.4.4 Transform Jacobian

The 4×6 Jacobian $\frac{\partial g_{ijk}}{\partial t_c}$ has three cases that depend on the direction of the transform $T_{(t_c)}$ on the path from frame frame i to j (note the direction)

$$\frac{\partial g_{ijk}}{\partial t_c} = \begin{cases} T_{ic} \frac{\partial T_{(t_c)}}{\partial t_c} T_{cj} l_{jk} & \text{if } T_{(t_c)} \text{ points towards } j \\ T_{ic} \frac{\partial T_{(-t_c)}}{\partial t_c} T_{cj} l_{jk} & \text{if } T_{(t_c)} \text{ points towards } i \\ 0 & \text{if } i = j \end{cases}$$

and $\frac{\partial T_{(t_c)}}{\partial t_c}$ are the canonical generators of $\mathbb{SE}3$ (a $4 \times 4 \times 6$ tensor, see Appendix A). This expands to

$$\frac{\partial g_{ijk}}{\partial t_c} = \begin{cases} T_{ic}\begin{bmatrix} I & [\bar{l}_{ck}]_\times \\ 0 & 0 \end{bmatrix} & \text{if } T_{(t_c)} \text{ points towards } j \\ -T_{ic}\begin{bmatrix} I & [\bar{l}_{ck}]_\times \\ 0 & 0 \end{bmatrix} & \text{if } T_{(t_c)} \text{ points towards } i \\ 0 & \text{if } i = j \end{cases} \tag{6}$$

Where $\bar{l}_{ck}$ is just the point $\bar{l}_{jk}$ transferred to frame c, and the operator $[v]_\times$ maps a 3-vector $v = [v_1, v_2, v_3]^T$ to a 3×3 skew symmetric matrix,

$$[v] = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}.$$

3.4.5 Computing the Jacobians Efficiently

Putting equations (3), (4), (5) and (6) together,

$$\begin{aligned} \frac{\partial h_{ijk}}{\partial t_c} &= \frac{\partial \mathcal{K}}{\partial \mathcal{M}}\frac{\partial \mathcal{M}}{\partial g_{ijk}}\frac{\partial g_{ijk}}{\partial t_c} \\ &= \pm\begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix}/\mathcal{M}_3[\mathbf{M},\ 0]T_{is}^{-1}T_{ic}\begin{bmatrix} I & [\bar{l}_{ck}]_\times \\ 0 & 0 \end{bmatrix} \\ &= \pm\begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix}/\mathcal{M}_3\mathbf{M}R_{is}^T R_{ic}\begin{bmatrix} I & [\bar{l}_{ck}]_\times \end{bmatrix} \end{aligned} \tag{7}$$

$$\begin{aligned} \frac{\partial h_{ijk}}{\partial \bar{l}_{jk}} &= \frac{\partial \mathcal{K}}{\partial \mathcal{M}}\frac{\partial \mathcal{K}}{\partial g_{ijk}}\frac{\partial g_{ijk}}{\partial \bar{l}_{jk}} \\ &= \begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix}/\mathcal{M}_3[\mathbf{M},\ 0]T_{is}^{-1}\begin{bmatrix} R_{ji}^T \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix}/\mathcal{M}_3\mathbf{M}R_{is}^T R_{ji}^T. \end{aligned} \tag{8}$$

This leads to the following efficient solution for both $\frac{\partial h_{ijk}}{\partial \bar{l}_{jk}}$ and $\frac{\partial h_{ijk}}{\partial t_c}$

1. Compute the 2×3 matrix $A = \begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix}/\mathcal{M}_3\mathbf{M}R_{is}^T$
2. Compute the 2×3 matrix $\frac{\partial h_{ijk}}{\partial \bar{l}_{jk}} = AR_{ji}^T$
3. Compute the 2×6 matrix $\frac{\partial h_{ijk}}{\partial t_c} = \pm AR_{ic}\begin{bmatrix} I, & [\bar{l}_{cj}]_\times \end{bmatrix}$

where the sign in step 3 depends on the direction of $T_{(t_c)}$. As an aside, note that this approach simplifies dealing with multi-camera systems, because the only thing that changes is T_{is}, the vehicle-to-sensor transform.

3.5 Complexity of Computing the Sparse Relative Solution

algorithm 1 Build linear system. Computes U, V, W, r_t, and r_l in $O(m^2n)$

```
Clear $U$, $V$, $W$, $r_t$, and $r_l$
for all landmarks $k$ do
  for all key-frames $i$ with a measurement of landmark $k$ do
    Compute $\frac{\partial h_{ijk}}{\partial l_{jk}}$
    $e_{ijk} = z_{ijk} - h_{ijk}(x)$
    $w_{ijk} = R_{ijk}^{-1}$
    $V_k = V_k + \frac{\partial h_{ijk}}{\partial l_{jk}}^T w_{ijk}^{-1} \frac{\partial h_{ijk}}{\partial l_{jk}}$
    $r_{l_k} = r_{l_k} + \frac{\partial h_{ijk}}{\partial l_{jk}}^T w_{ijk}^{-1} e_{ijk}$
    for all $p \in Path(i,j)$ do
      Compute $\frac{\partial h_{ijk}}{\partial t_p}$
      $r_{t_P} = r_{t_P} + \frac{\partial h_{ijk}}{\partial t_p}^T w_{ijk} e_{ijk}$
      $W_{kp} = W_{kp} + \frac{\partial h_{ijk}}{\partial l_{jk}}^T w_{ijk} \frac{\partial h_{ijk}}{\partial t_p}$
      for all $q \in Path(p,j)$ do
        Compute $\frac{\partial h_{ijk}}{\partial t_q}$
        $U_{pq} = U_{pq} + \frac{\partial h_{ijk}}{\partial t_p}^T w_{ijk} \frac{\partial h_{ijk}}{\partial t_q}$
        $U_{qp} = U_{qp} + \frac{\partial h_{ijk}}{\partial t_q}^T w_{ijk} \frac{\partial h_{ijk}}{\partial t_p}$
      end for
    end for
  end for
end for
```

Similar to sparse bundle adjustment, the following steps are used to exploit the structure of H to compute the *normal equations* and state-updates efficiently:

1. *Build linear system*, computing the terms U, V, W, r_t, and r_l. Complexity is $O(m^2n)$ using key-frames or $O(m^3n)$ using all images.
2. *Forward substitute*, computing $A = U - W^T V^{-1} W$, and $b = r_t - W^T V^{-1} r_l$. Complexity is $O(m^2n)$.
3. *Solve reduced system* of equations, $A\delta t = b$ for the update δt. Complexity is $O(m^3)$.
4. *Back substitute* to solve for the map update, $\delta l = V^{-1}(r_l - W\delta t)$. Complexity is $O(mn)$.

The first step is completely different in the relative framework and is described in more detail in Algorithm 1. The overall complexity is $O(m^3)$, which matches traditional sparse bundle adjustment. Note that it is easy to convert Algorithm 1 into a robust m-estimator by replacing the weights, w_{ijk}, with robust weight kernels, $w_{ijk} = R_{ijk}^{-1}\mathcal{W}(e_{ijk})$ — for example we use the Huber kernel [16].

Focusing on a single transform c in the chain from j to i, note that equation (7) can be split into a left part, L_c, and right part, R_c, as

$$\begin{aligned} \frac{\partial h_{ijk}}{\partial t_c} &= \pm \begin{bmatrix} f_x & s_x & c_x - u \\ 0 & f_y & c_y - v \end{bmatrix} / m_3 M R_{is}^T R_{ic} \begin{bmatrix} I & [l_{ck}]_\times \end{bmatrix} \\ &= \pm L_c \frac{\partial T_{(t_c)}}{\partial t_c} R_c. \end{aligned}$$

This is useful because the terms L_c and R_c can be computed incrementally as we traverse from node i to node j in Algorithm 1. Finally, notice that if feature tracks are contiguous over numerous frames (which they typically are), then the sparsity pattern in W will be the same in the relative-formulation as it is in traditional one – hence the relative-formulation cost of forward-substitution, solving the reduced system, and back-substitution (steps 2-4) should be approximately equivalent.

4 Adaptive Optimization

To reduce computation, it is important to optimize only those parameters that might change in light of new information [26][28]. Below we outline one approach to limit the parameters that are actively optimized.

A breadth-first-search from the most recent frame is used to discover local parameters that might require adjustment. During the search, all frames in which the average re-projection error changes by more than a threshold, $\Delta\epsilon$, are added to an *active region* that will be optimized. The search stops when no frame being explored has a change in re-projection error greater than $\Delta\epsilon$. Landmarks visible from active frames are activated, and all non-active frames that have measurements of these landmarks are added to a list of static frames, which forms a slightly larger set we call the static region. Measurements made from static frames are included in the optimization, but the associated relative pose-error parameters are not solved for. The spanning tree, active region, and static region are shown in Figure 5.

5 Results

The iterative non-linear least-squares solution that exploits the sparse relative structure and the four steps in section 3.5 results in the run-time break-down shown in Figure 6. This illustrates that building the sparse system of equations is the dominant cost.

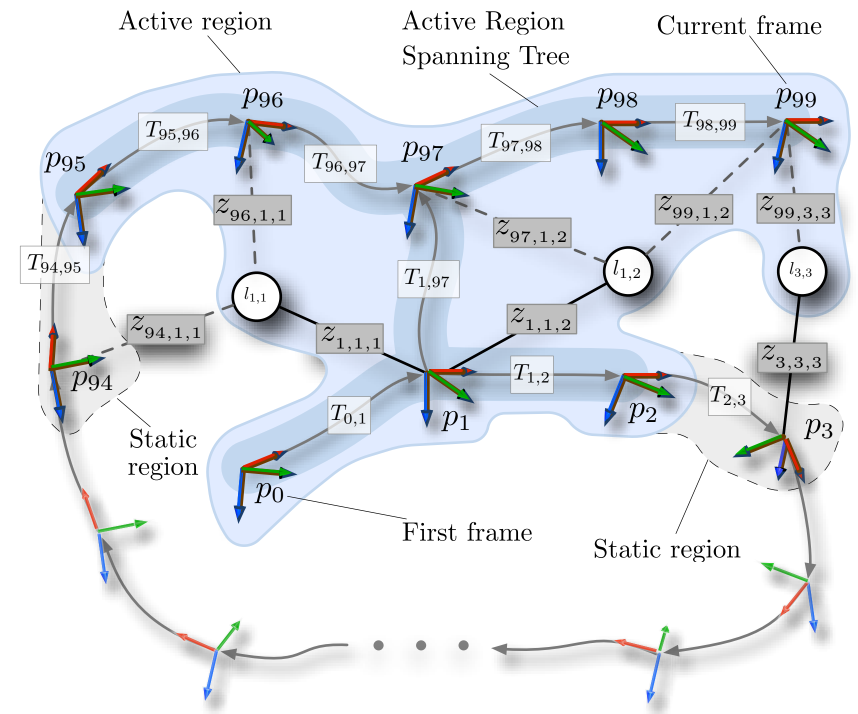


Figure 5: Discovery of local active region. In this example, re-projection errors have changed by more than $\Delta\epsilon$ in the local frames p_{95}, p_{96}, p_{97}, p_{98}, p_{99}, p_0, p_1, and p_2. This local active region is discovered by a weighted breadth-first-search starting at the current frame, p_{99}. All landmarks visible from active frames are optimized for. Any non-active frames that have measurements of active landmarks are added to the static region. Measurements from the static region contribute to the objective function, but the associated edges are not solved for (the frames are fixed).

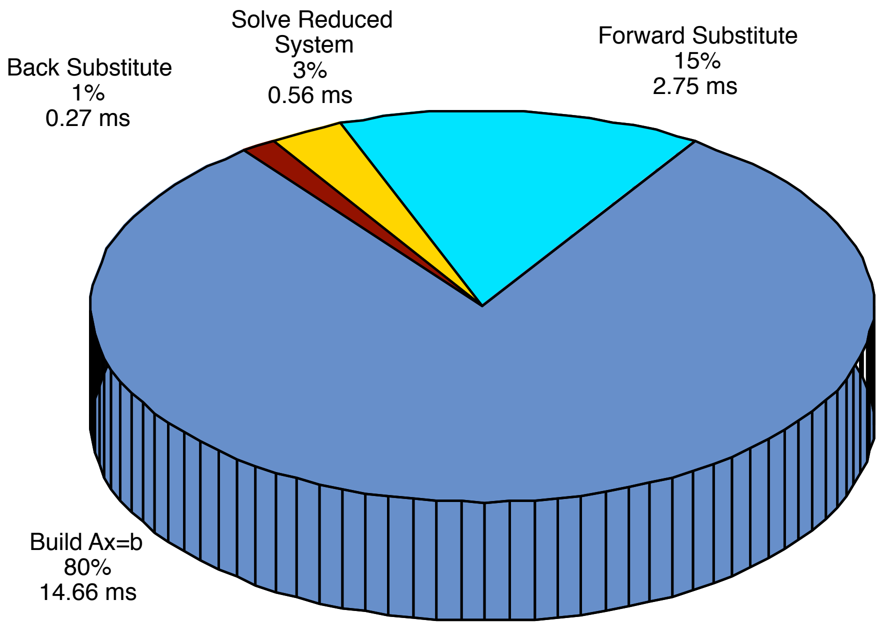


Figure 6: Average run-times for the main steps of relative bundle adjustment. The average adaptive region from the Monte Carlo simulation was 4.6 frames. Note that it is the cost of building the linear system of equations that dominates the cubic complexity of solving for the adaptive region of poses.

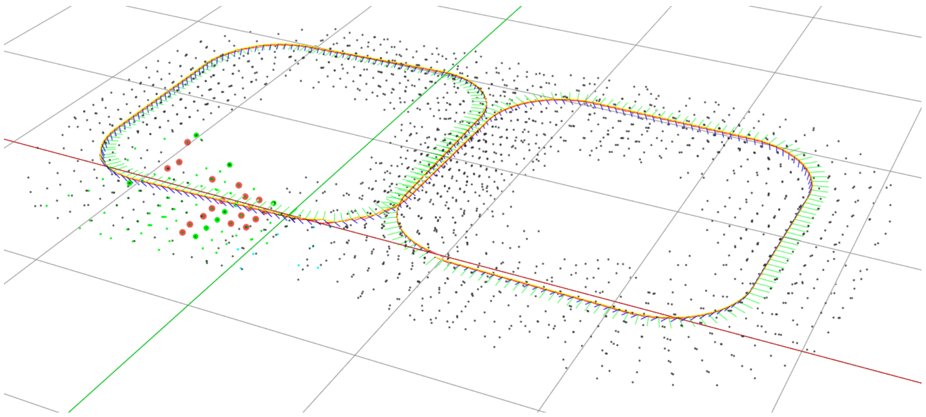


Figure 7: Figure-of-eight sequence used in Monte-Carlo simulation. This sequence has 288 frames and 3215 landmarks and 12591 measurements with 1 pixel standard deviation Gaussian measurement noise added. The large red landmarks indicate a loop closure.

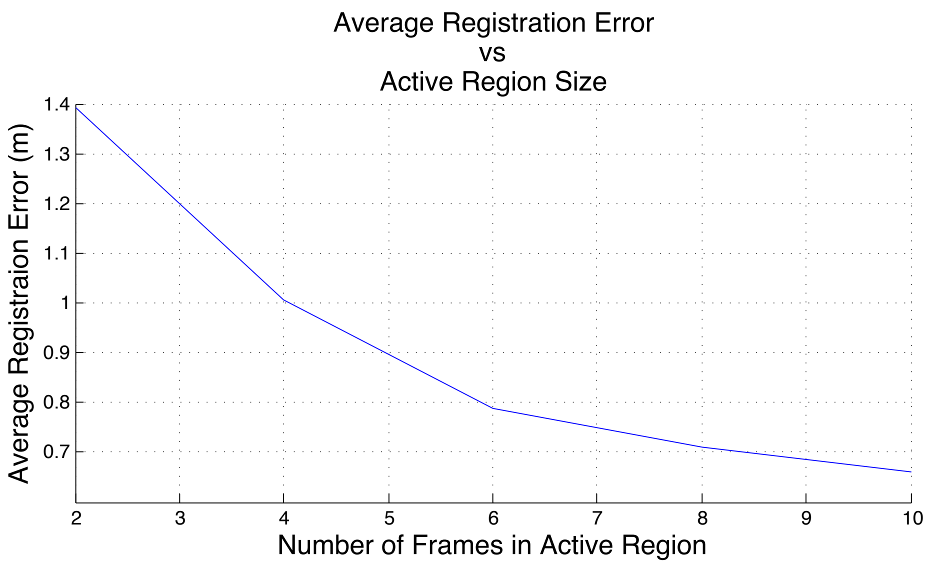
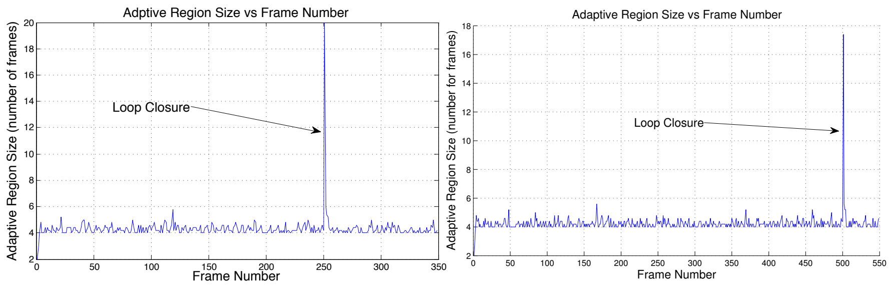


Figure 8: Average Registration Error vs. Number of Frames being updated. In the relative formulation, as the local region grows the average RMS error drops quickly toward the same as when computed with all frames active. This motivates the use of an adaptive region that allows parameters to vary only if it has an effect on the cost function.



(a) 50m loop with closure at frame 250. (b) 100m loop with closure at frame 500.

Figure 9: This figure shows how the number of frames in the adaptive region fluctuates over time and during loop closure. During loop closure the size of the adaptive region jumps to accommodate all the local frames that have been added to the active region, as well as any neighboring frames that will be affected. Notice that errors do not propagate all the way around the loop, and only a fraction of the state vector needs to be updated. Loop closure at 250 and 500 frames induces updates in approximately the same number of parameters, which strongly indicates that optimization at loop closure will remain constant time, independent of loop size. Before loop closure, the average metric position error is over 75cm for the 500 frame loop. Using the same adaptive region criteria, Euclidean bundle adjustment would require adjusting *all* parameters in the loop - whereas the adaptive relative approach only adjusts 20 poses.

5.1 Simulation

To determine the performance of the relative framework, a batch of Monte Carlo simulations were run. The sequence contains a realistic trajectory, landmark distribution, and a 1 pixel standard deviation Gaussian measurement noise (see Figure 7).

We compute error in the following way: for each pose in the trajectory, we register that pose to its ground truth counterpart, and then *localize* the rest of the relative trajectory in that frame. Note that "localizing" the relative trajectory is done with a breadth-first-search that computes each frame's pose in the coordinate system of the root frame; this process projects from the relative manifold into a single Euclidean frame, and may cause "rips" to appear at distant loop closures. Finally, the total trajectory registration error is computed as the average Euclidean distance between ground truth and the localized frames. The average of all frames and all registrations is the error plotted. Not surprisingly, initial results in Figure 8 indicate that error reduces towards the full solution (in the relative space) as the local region increases in size.

The results here use an adaptive region threshold of $\Delta\epsilon = 0.05$ pixels. With this threshold we find that the discovery of new frames to include in the active region quickly drops to between 4 and 5 poses, except at loop closure where it jumps to accommodate the larger region of poses found by the weighted breadth-first-search. Figure 9 shows the adaptive region size discovered for two different loop closures, one 50m long and another 100m long. The point to note is that the discovered adaptive region is independent of loop size, and errors do not propagate around the loop even though loop closure error is ~75cm on average for the 500 frame sequence. Using the same adaptive region criteria, Euclidean bundle adjustment would require adjusting *all* parameters in the loop - whereas the adaptive relative approach adjusts just 20 poses.

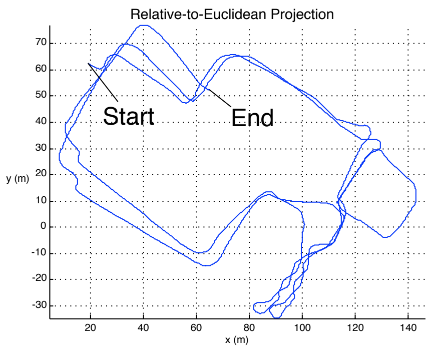
Our adaptive strategy for discovering the active region is designed to have a rippling effect: when parameter estimates change, it affects the re-projection error in nearby frames, which, if greater than $\Delta\epsilon$, will add those parameters to the active region, potentially causing them to change...etc. A key result of the relative formulation is that these *errors stop propagating* and balance out with distance from the new information - that is, the network of parameters is critically damped.

6 Discussion

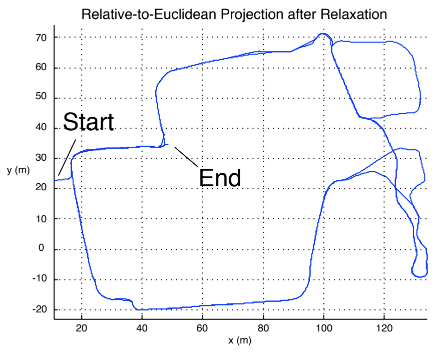
The privileged-frame approach and the relative formulations are very different; their objective functions are different and they solve for different quantities. The former embeds the trajectory in a single Euclidean space; the latter in a connected Riemannian manifold. At first reading it may appear that the lack of a simple Euclidean distance metric between two points, and the fact that we cannot render the solution very easily, is a disadvantage of the relative formulation. Note however that the manifold is a metric space, and distance between two points can be computed from shortest paths in the graph. With this in mind, the relative representation should *still* be amenable to planning algorithms which are commonly defined over graphs in the first place. Furthermore, because the manifold is (by definition) locally Euclidean, algorithms that require precise *local* metric estimates, such as obstacle avoidance or object manipulation, can operate without impediment.

We posit that a *topometric* relative formulation is sufficient for many mobile robot navigation tasks, and that a single global Euclidean representation is rarely necessary. Certainly the benefits afforded by incrementally constant time performance are tremendous, and in the light of that, some inconvenience may be acceptable. If a unified global Euclidean picture is deemed essential by a particular external application or technique, our choice would be to push responsibility for generating the single Euclidean embedding into that process - for example undertaking fast approximate pose-graph relaxation in order to render consistent results in a user interface [24, 14].

As an example, Figure 10 shows the result of transforming a large relative state estimate into a single Euclidean frame using pose-graph relaxation. Note that even this state-of-the art global Euclidean estimate fails to discover the true rectilinear structure. Arguably the best way to improve the map would be to *schedule* new measurements across the diagonal of the map, thereby considerably constraining the solution. While this interventionist approach is used



(a) Before global relxation



(b) After global relaxation

Figure 10: To view relative estimates in a consistent fashion (single global frame) we have to transform from the relative representation to a single Euclidean coordinate system. This relative-to-Euclidean projection can be viewed as a relative pose-graph relaxation problem that takes loop closure constraints into account and produces consistent global estimates. The sequence here has 23K poses over 1.08 kilometers which makes the conversion computationally expensive – in particular it *cannot* be solved in constant time. Therefore, the relative-to-Euclidean projection is designed to run on the user interface, *not* on the robot. The robust relative-framework stereo SLAM system that produced these trajectories runs live at 20-40hz on 512x384 grey-scale images and has real-time automatic loop closure detection using FABMAP [5].

extensively in surveying, we are not comfortable with placing such a requirement on a mobile platform — ideally navigation and mapping should be a quiet background task producing estimates for consumption by any interested client process. With this example in mind, perhaps accurate global Euclidean state estimates are the wrong goal to aim for — what matters is relative metric accuracy and topological consistency — all of which can be attained with a relative manifold approach.

7 Conclusion

The fact that the variables in bundle adjustment are defined relative to a single coordinate frame has a large impact on the algorithm's iterative convergence rate. This is especially true at loop closure, when large errors must propagate around the entire loop to correct for global errors that have accumulated along the path. As an alternative, we have presented an adaptive relative formulation that can be viewed as a *continuous* sub-mapping approach – in many ways our relative treatment is an intuitive simplification of previous sub-mapping methods. Furthermore by solving all parameters within an adaptive region, the proposed method attempts to match the full maximum likelihood solution within the metric space defined by the manifold. In stark contrast to traditional bundle adjustment, our evaluations and results indicate that state updates in the relative approach are constant time, and crucially, remain so even during loop closure events.

Appendix A: Rotation Derivatives

For reference, here are some useful facts about transformations, rotations and their derivatives. Let θ=$[r, p, q]$ represent roll, pitch and yaw. The associated rotation matrix using the fixed-frame xyz-Euler angle convention is

$$\begin{aligned}
R &= R_q R_p R_r \\
&= \begin{bmatrix} \cos(q) & -\sin(q) & 0 \\ \sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(p) & 0 & \sin(p) \\ 0 & 1 & 0 \\ -\sin(p) & 0 & \cos(p) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(r) & -\sin(r) \\ 0 & \sin(r) & \cos(r) \end{bmatrix} \\
&= \begin{bmatrix} \cos(p)\cos(q) & -\cos(r)\sin(q)+\sin(r)\sin(p)\cos(q) & \sin(r)\sin(q)+\cos(r)\sin(p)\cos(q) \\ \cos(p)\sin(q) & \cos(r)\cos(q)+\sin(r)\sin(p)\sin(q) & -\sin(r)\cos(q)+\cos(r)\sin(p)\sin(q) \\ -\sin(p) & \sin(r)\cos(p) & \cos(r)\cos(p) \end{bmatrix}.
\end{aligned}$$

The derivatives of R with respect to infinitesimal rotations (θ=0) are

$$\begin{aligned}
\frac{\partial R}{\partial r} &= R_q R_p \frac{\partial R_r}{\partial r} \\
& \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \cos'(r) & -\sin'(r) \\ 0 & \sin'(r) & \cos'(r) \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial R}{\partial p} &= R_q \frac{\partial R_p}{\partial p} R_q \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos'(p) & 0 & \sin'(p) \\ 0 & 0 & 0 \\ -\sin'(p) & 0 & \cos'(p) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix},
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial R}{\partial q} &= \frac{\partial R_q}{\partial q} R_p R_r \\
&= \begin{bmatrix} \cos'(q) & -\sin'(q) & 0 \\ \sin'(q) & \cos'(q) & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
& \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.
\end{aligned}$$

The individual terms for the derivative of a transformation matrix $T_{(t)}$ with respect to $t = [x, y, z, r, p, q]$ are

$$\frac{\partial T_{(t)}}{\partial x} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = G_1, \ \frac{\partial T_{(t)}}{\partial y} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = G_2,$$

$$\frac{\partial T_{(t)}}{\partial z} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = G_3, \ \frac{\partial T_{(t)}}{\partial r} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = G_4$$

$$\frac{\partial T_{(t)}}{\partial p} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = G_5, \ \frac{\partial T_{(t)}}{\partial q} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = G_6$$

which is a $4 \times 4 \times 6$ tensor

$$\frac{\partial T_{(t)}}{\partial t} = \begin{bmatrix} G_1 & G_2 & G_3 & G_4 & G_5 & G_6 \end{bmatrix}.$$

These are the canonical generators of SE(3). When right multiplied by a 4×1 homogeneous vector $v = [x, y, z, 1]^T$

$$\begin{aligned} \frac{\partial T_{(t)}}{\partial t} v &= -\begin{bmatrix} G_1 & G_2 & G_3 & G_4 & G_5 & G_6 \end{bmatrix} v \\ &= \begin{bmatrix} I & [\bar{v}]_\times \\ 0 & 0 \end{bmatrix} \end{aligned}$$

where $[\bar{v}]_\times$ is the 3×3 skew symmetric matrix,

$$[\bar{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}.$$

Recall that the inverse of a homogeneous transformation matrix is

$$\begin{bmatrix} R & \bar{v} \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T \bar{v} \\ 0 & 1 \end{bmatrix}.$$

The derivative of an inverse transform is thus

$$\begin{aligned}
\frac{\partial T_{(t)}^{-1}}{\partial t} &= \frac{\partial}{\partial t}\begin{bmatrix} R^T & -R^T\bar{v}^T \\ 0 & 1 \end{bmatrix} \\
&= -\begin{bmatrix} G_1 & G_2 & G_3 & G_4 & G_5 & G_6 \end{bmatrix} \\
&= -\frac{\partial T_{(t)}}{\partial t} \\
&= \frac{\partial T_{(-t)}}{\partial t}
\end{aligned}$$

because $R = I$ when we take the Jacobian w.r.t. $[x, y, z]$ and $[x, y, z] = 0$ when we take the Jacobian w.r.t. R.

References

[1] J. Blanco, J. Fernandez-Madrigal, and J. Gonzalez. Toward a unified bayesian approach to hybrid metric–topological SLAM. *IEEE Transactions on Robotics and Automation*, 24(2):259–270, 2008.

[2] M. Bosse, P. Newman, J. Leonard, and S. Teller. An Atlas framework for scalable mapping. Technical report, MIT Marine Robotics Laboratory, 2002.

[3] D.C. Brown. A solution to the general problem of multiple station analytical stereotriangulation. Technical report, RCP-MTP Data Reduction Technical Report No. 43, Patrick Air Force Base, Florida (also designated as AFMTC 58-8), 1958.

[4] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardos. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*, 2007.

[5] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665, 2008.

[6] A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Realtime single camera SLAM. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 29(6):1113–1139, 2007.

[7] M. C. Deans. *Bearings-Only Localization and Mapping*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2005.

[8] F. Dellaert. Square root SAM. In *Proceedings of Robotics: Science and Systems*, pages 1181–1203, Boston, June 2005.

[9] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular SLAM. In *Proceedings British Machine Vision Conference*, September 2008.

[10] C. Engels, H. Stewenius, and D. Nister. Bundle adjustment rules. In *Photogrammetric Computer Vision*, 2006.

[11] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Robotics: Science and Systems*, pages 57–64, 2005.

[12] A. W. Fitzgibbon and A. Zisserman. *Automatic Camera Recovery for Closed or Open Image Sequences*. Springer, Freiburg, Germany, June 2004.

[13] U. Frese and T. Duckett. A multigrid approach for accelerating relaxation-based SLAM. In *Proceedings IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR 2003)*, pages 39–46, Acapulco, Mexico, August 2003.

[14] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings Robotics: Science and Systems*, Atlanta, June 2007.

[15] J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, June 2001.

[16] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(2):73–101, 1964.

[17] M. Kaess. *Incremental Smoothing and Mapping*. PhD thesis, Georgia Institute of Technology, 2008.

[18] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision*, Marseille, October 2008.

[19] K. Konolige and M. Agrawal. FrameSLAM: from bundle adjustment to realtime visual mapping. *IEEE Transactions on Robotics and Automation*, 24(5):1066–1077, 2008.

[20] A. Martinelli, V. Nguyen, N. Tomatis, and R. Siegwart. A relative map approach to SLAM based on shift and rotation invariants. *Robotics and Autonomous Systems*, 55(1):50–61, 2007.

[21] P. F. McLauchlan. The variable state dimension filter applied to surface-based structure from motion. Technical report, University of Surrey, 1999.

[22] E. M. Mikhail. *Observations and Least Squares*. University Press of America, Washington, D.C, 1983.

[23] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyse, and P. Sayd. Real time localization and 3d reconstruction. In *Proceedings of Computer Vision and Pattern Recognition*, New York, New York, June 2006.

[24] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2262–2269, Orlando, May 2006.

[25] P. Pinies and J. D. Tardos. Scalable slam building conditionally independent local maps. In *IEEE conference on Intelligent Robots and Systems*, 2007.

[26] A. Ranganathan, M. Kaess, and F. Dellaert. Loopy SAM. In *International Joint Conferences on Artificial Intelligence*, pages 2191–2196, 2007.

[27] G. Sibley, L. Matthies, and G. Sukhatme. A sliding window filter for incremental SLAM. In Ville Kyrki and Danica Kragic, editors, *From features to actions - Unifying perspectives in computational and robot vision*, volume 8, chapter 7, pages 103–112. Springer Lecture Notes in Electrical Engineering, 2007.

[28] D. Steedly and I. Essa. Propagation of innovative information in non-linear least-squares structure from motion. In *ICCV01*, pages 223–229, 2001.

[29] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[30] S. Thrun, D. Koller, Z. Ghahmarani, and H. Durrant-Whyte. SLAM updates require constant time. In *Workshop on the Algorithmic Foundations of Robotics*, Nice, France, December 2002.

[31] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372. Springer-Verlag, London, UK, 2000.