The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Efficient Trajectory Bending with Applications to Loop Closure

Gijs Dubbelman[1,2] and Isaac Esteban[1,2] and Klamer Schutte[1]

Abstract— **In robotic applications the absolute pose is often obtained as the integral of successive relative rigid-body motions. As each relative rigid-body motion is typically the product of statistical inference, the integrated absolute pose will exhibit error build-up and the estimated trajectory will differ from the true trajectory undertaken by the system. Some application areas allow the system to receive additional information about its current absolute pose, for example from loop detection, which is more accurate than the integral of the relative rigid-body motions. The availability of this absolute information is usually less frequent than the information underlying the relative rigid-body motions. This contribution addresses an efficient closed form algorithm which minimally bends a trajectory such that the integrated pose is exactly equal to any particular desired pose. The manner in which the bending is distributed over the trajectory is controllable using weights. The proposed method will be compared against a maximum likelihood solution on simulated trajectories as well as on trajectories estimated from binocular and monocular data. The results indicate that the performance differences between the closed form approach and the maximum likelihood solution are negligible while the closed form approach is significantly more efficient.**

Index Terms— **loop closure, visual odometry, visual reconstruction.**

I. INTRODUCTION

Loop closure can conceptually be divided in three different phases, namely, loop detection, updating the current pose and updating the map and/or trajectory. This contribution focusses on the last phase, more precisely, updating the complete trajectory such that the final pose is exactly equal to a particular desired pose. This desired pose is typically derived from loop detection but it can, for example, also be attained from a Global Position System (GPS) and an Inertial Navigation System (INS). The proposed algorithm can be applied to any trajectory, the focus is however on trajectories estimated from visual data. In recent years visual-odometry has gained significantly in accuracy, for example Nister et al. [15], Dubbelman et al. [5], Dubbelman and Groen [6], Konolige et al. [12], Konolige and Agrawal [11] and Comport et al. [4]. To reduce the error growth in the estimated trajectory, techniques such as Simultaneous Localization and Mapping (SLAM), Thrun et al. [18], or Bundle Adjustment (BA), Triggs et al. [19], are frequently used. Given the current accuracy of visual-odometry, their main benefit is the ability to update the complete trajectory and/or map when additional (absolute) pose information becomes available. Even without considering their computational complexity, these methods are not without disadvantages. For example, BA theoretically provides a maximum likelihood (ML) solution by minimizing reprojection errors in all images. The ML solution is, however, challenging to obtain efficiently due to data which does not adhere to the assumptions made i.e. outliers. SLAM methods which use a Rao-Blackwellized particle filter, such as FastSLAM, Thrun et al. [18] and Montemerlo et al. [13], suffer from particle deprivation [18]. In short, as the uncertainty in the estimated trajectory and map grows over time, the fixed amount of particles no longer suffices to represent the complete pdf. adequately. As a consequence, there can be a point in time before which all particles share a common ancestor and therefore have exactly the same trajectory and map. When this occurs at loop closure, the particle filter can no longer update the trajectory nor map before this point. Furthermore, the possible trajectories during loop closure are limited by the particle distribution. When no particle, with a trajectory ending close to the desired pose exist, loop closing will be sub optimal. Considering SLAM approaches which use an EKF or an IKF, performing a trajectory or map update during loop closure by means of a filter update can yield unsatisfactory results, for an explanation see Castellanos et al. [3] and Baily et al. [2]. As a possible improvement Newman et al. [14] proposed an algorithm which bends an trajectory such that the final pose is equal to a desired pose. Their approach finds an ML solution by using non-linear iterative constrained optimization. This optimization scheme runs detached from the SLAM filter and its sole purpose is to update the trajectory at loop closure. When many probabilistic links between poses exist, using a method as proposed in [9] can be advantageous.

In this contribution the focus is on trajectories where the probabilistic links between absolute poses are only governed by the uncertainty in the relative rigid-body motion estimates i.e. the trajectory is exactly sparse. These trajectories are a typical product of visual odometry systems. In this setting an efficient closed form algorithm is proposed to update the complete trajectory at loop closure. When the uncertainties in the relative rigid-body motion estimates are known, they can be used to distribute the updates proportionally over the trajectory. The experiments on synthetic, monocular and binocular data indicate that satisfactory results are obtained when using the proposed closed form algorithm. When comparing the closed form algorithm against a ML solution, which in spirit is similar to that of Newman et al. [14], the observed performance differences are fractional while the proposed algorithm requires significantly less computation

Electro-Optical Systems[1], TNO Defence, Security and Safety, Oude Waalsdorperweg 63, 2509 JG The Hague, The Netherlands, {gijs.dubbelman, isaac.esteban,klamer.schutte}@tno.nl

Intelligent Systems Laboratory Amsterdam[2] (ISLA), University of Amsterdam, Science Park 107, 1098 XG Amsterdam, The Netherlands

time. For example, the proposed algorithm is 1000 times faster for trajectories consisting of 350 poses. This makes the proposed algorithm well suited for embedded applications where computational resources are typically limited. In this contribution the applicability of the proposed algorithm is demonstrated with respect to loop closure. In further research the proposed algorithm can be utilized to perform sensor fusion with absolute position and orientation sensors such as GPS/INS. Furthermore, the proposed closed form algorithm can be used to initiate further non linear optimization such as BA or can be integrated with existing SLAM approaches.

II. Bending of rigid-body trajectories

Let the relative rigid-body motion at time n be denoted as M_n. The absolute pose at time n, denoted as A, is the integral of all relative rigid-body motions i.e.

$$\mathrm{A} = \mathrm{M}_1....\mathrm{M}_n = \prod_{i=1}^{i=n} \mathrm{M}_i \ . \tag{1}$$

The desired absolute pose at time n is denoted as $\hat{\mathrm{A}}$. The goal of the algorithm is to update each M_n with an additional rigid-body transformation $\hat{\mathrm{U}}_n$ such that

$$\hat{\mathrm{A}} = \prod_{i=1}^{i=n} \left(\mathrm{M}_i \hat{\mathrm{U}}_i \right) \tag{2}$$

i.e. when integrating the relative rigid-body motions with their respective updates the trajectory ends in the desired pose. The rigid-body transformation U that brings A onto $\hat{\mathrm{A}}$ is obtained with $\mathrm{U} = \mathrm{A}^{-1}\hat{\mathrm{A}}$. Using the technique of sec. II-A allows this update motion U to be divided in n relative rigid-body motions such that $\mathrm{U} = \mathrm{U}_1....\mathrm{U}_n$. These updates can be concatenated to the rigid-body motions, which results in

$$\hat{\mathrm{A}} = \mathrm{M}_1....\mathrm{M}_n\mathrm{U}_1....\mathrm{U}_n \ . \tag{3}$$

The next step is to transform each of the interpolated update motions U_i to its equivalent $\hat{\mathrm{U}}_i$, because then they can be applied distributed over the trajectory as in eq. 2. Start by considering which transformation $\mathfrak{T}$ has to be applied to the final update U_n such that $\mathfrak{T}(\mathrm{U}_n) = \hat{\mathrm{U}}_n$ can be integrated after M_n and such that the final absolute pose remains the desired pose $\hat{\mathrm{A}}$, more formally,

$$\hat{\mathrm{A}} = \mathrm{M}_1....\mathrm{M}_{n-1}\ \mathrm{M}_n\mathfrak{T}(\mathrm{U}_n)\ \mathrm{U}_1....\mathrm{U}_{n-1} \ . \tag{4}$$

From eq. 4 it can be derived that the transformation is

$$\hat{\mathrm{U}}_n = \mathfrak{T}(\mathrm{U}_n) = (\mathrm{U}_1....\mathrm{U}_{n-1})\ \mathrm{U}_n\ (\mathrm{U}_1....\mathrm{U}_{n-1})^{-1}. \tag{5}$$

Also note from eq. 3 that by using the substitution $\mathrm{U}_1....\mathrm{U}_{n-1} = (\mathrm{M}_1....\mathrm{M}_n)^{-1}\hat{\mathrm{A}}\mathrm{U}_n^{-1}$ the transformation of eq. 5 can be written as

$$\mathfrak{T}(\mathrm{U}_n) = (\mathrm{M}_1....\mathrm{M}_n)^{-1}\hat{\mathrm{A}}\ \mathrm{U}_n\ \hat{\mathrm{A}}^{-1}(\mathrm{M}_1....\mathrm{M}_n). \tag{6}$$

Applying this transformation allows eq. 3 to be rewritten as

$$\hat{\mathrm{A}} = \mathrm{M}_1....\mathrm{M}_{n-1}\mathrm{M}_n\hat{\mathrm{U}}_n\mathrm{U}_1....\mathrm{U}_{n-1}. \tag{7}$$

Note that the final update U_n is transformed to $\hat{\mathrm{U}}_n$ such that it can be applied according to eq. 2 (i.e. after M_n) without changing the desired final absolute pose $\hat{\mathrm{A}}$. Now consider transforming the next update U_{n-1} such that

$$\hat{\mathrm{A}} = \mathrm{M}_1....\mathrm{M}_{n-2}\ \mathrm{M}_{n-1}\mathfrak{T}(\mathrm{U}_{n-1})\ \mathrm{M}_n\hat{\mathrm{U}}_n\mathrm{U}_1....\mathrm{U}_{n-2}. \tag{8}$$

Similarly to eq. 5 this transformation is

$$\mathfrak{T}(\mathrm{U}_{n-1}) = (\mathrm{M}_n\hat{\mathrm{U}}_n\mathrm{U}_1....\mathrm{U}_{n-2})\mathrm{U}_{n-1}(\mathrm{M}_n\hat{\mathrm{U}}_n\mathrm{U}_1....\mathrm{U}_{n-2})^{-1}. \tag{9}$$

Again note from eq. 7 that by using the substitution $(\mathrm{M}_n\hat{\mathrm{U}}_n\mathrm{U}_1....\mathrm{U}_{n-2}) = (\mathrm{M}_1....\mathrm{M}_{n-1})^{-1}\hat{\mathrm{A}}\mathrm{U}_{n-1}^{-1}$ the transformation of eq. 9 can be written as

$$\mathfrak{T}(\mathrm{U}_{n-1}) = (\mathrm{M}_1....\mathrm{M}_{n-1})^{-1}\hat{\mathrm{A}}\ \mathrm{U}_{n-1}\ \hat{\mathrm{A}}^{-1}(\mathrm{M}_1....\mathrm{M}_{n-1}). \tag{10}$$

When the same process is continued for all other updates, starting from U_{n-2} down to U_1, it can be derived that the general transformation $\mathfrak{T}$ is given by

$$\hat{\mathrm{U}}_m = \mathfrak{T}(\mathrm{U}_m) = (\prod_{i=1}^{i=m} \mathrm{M}_i)^{-1}\hat{\mathrm{A}}\mathrm{U}_m\hat{\mathrm{A}}^{-1}(\prod_{i=1}^{i=m} \mathrm{M}_i) \ . \tag{11}$$

The interesting property is that the transformation of each U_i only depends on the relative rigid body motions up to pose i i.e. $\mathrm{M}_1....\mathrm{M}_i$ and the desired absolute pose $\hat{\mathrm{A}}$. The solution to trajectory bending can therefore be obtained in closed form, has complexity $\mathrm{O}(n)$ and can be computed in any specific order. Furthermore, the amount of floating point operations per update is merely in the order of four hundred. For further sections it is useful to also define the inverse of $\mathfrak{T}$ as

$$\mathrm{U}_m = \mathfrak{T}^{-1}(\hat{\mathrm{U}}_m) = \hat{\mathrm{A}}^{-1}(\prod_{i=1}^{i=m} \mathrm{M}_i)\hat{\mathrm{U}}_m(\prod_{i=1}^{i=m} \mathrm{M}_i)^{-1}\hat{\mathrm{A}}. \tag{12}$$

.

A. Relative interpolation on SE(3)

Clearly there are infinitely many combinations of n relative rigid-body motions which when integrated result in the update motion $\mathrm{U} = \mathrm{A}_n^{-1}\hat{\mathrm{A}}_n$. It seems natural however to bend the trajectory minimally, furthermore, control over the distribution of bending over the trajectory is desired. These intuitions will be described more formally in sec. II-B. In this section a technique is presented that divides U in n relative rigid-body motions of minimal magnitude. Furthermore, the magnitude of each update is controllable by weights $w_1, ..., w_n$ with $w_1{+}w_2{+}w_3, ..., {+}w_n = 1$. For this purpose it is convenient to parameterize an element of $SE(3)$, i.e. the group of rigid-body motions, using a unit quaternion $\mathbf{q}$ and a translation vector $\mathbf{t}$. A unit quaternion consists of a one dimensional real part q and a three dimensional spatial part $\vec{\mathbf{q}} = (q_i, q_j, q_k)$, thus $\mathbf{q} = (q, \vec{\mathbf{q}})$. The inverse of a quaternion is denoted as $\mathbf{q}^{-1}$ and multiplication simply as $\mathbf{q}_1\mathbf{q}_2$. See Grassia [8] for more information on quaternion algebra and its relation to rotation. To shorten notation a rigid-body motion is denoted with $\mathrm{M} = (\mathbf{q}, \mathbf{t})$. One can define the identity, group multiplication and inversion, denoted as I, $\mathrm{M}_1\mathrm{M}_2$ and M^{-1}, for this parametrization straightforwardly.

Or the chosen parametrization can be transformed back and forth to the usual four by four homogenous matrix representation if preferred. Note that the formulas of sec. II are independent from the choice of parametrization. For the purpose of interpolating between rotations the following mappings on unit quaternions are useful

$$\mathfrak{q} = \log(\mathbf{q}) = \begin{cases} \arccos(q)\frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|} , & q \neq 1 \\ (0,\ 0,\ 0) , & q = 1 \end{cases} , \tag{13}$$

$$\mathbf{q} = \exp(\mathfrak{q}) = \begin{cases} (\cos(\|\mathfrak{q}\|)\ ,\ \sin(\|\mathfrak{q}\|)\ \frac{\mathfrak{q}}{\|\mathfrak{q}\|}), & \|\mathfrak{q}\| \neq 0 \\ (1,\ 0,\ 0,\ 0) , & \|\mathfrak{q}\| = 0 \end{cases} , \tag{14}$$

Grassia [8]. Note that $\|\mathfrak{q}\|$ is half the angle of rotation and the direction of $\mathfrak{q}$ is the rotation axis. For optimality considerations it is important that the interpolation is performed over the minimal length trajectory from A_n to $\hat{\mathrm{A}}_n$. In that respect we recall that the intrinsic left invariant metric between two rigid body motions is expressed as

$$d(\mathrm{M}_1, \mathrm{M}_2)_\alpha = \sqrt{\|\log({\mathbf{q}_1}^{-1}\mathbf{q}_2)\|^2 + \|\alpha(\mathbf{t}_2 - \mathbf{t}_1)\|^2} . \tag{15}$$

Park [16]. Note the α in eq. 15 which specifies the weighting between the rotational and translational differences. The magnitude of a rigid-body motion M is then defined as $d(\mathrm{I}, \mathrm{M})_\alpha$. Considering this metric the following mappings[1] on elements of $SE(3)$ are defined

$$\mathfrak{m} = \log(\mathrm{M}) = (\log(\mathbf{q}), \alpha\mathbf{t}) , \tag{16}$$

$$\mathrm{M} = \exp(\mathfrak{m}) = (\exp(\mathfrak{q}), \frac{1}{\alpha}\mathbf{t}) . \tag{17}$$

They are based on the double geodesic approach, Altafini [1], and apply the mappings eq. 16 and eq. 17 on the rotational part while scaling the translational part. The mappings can be extended over the entire manifold of $SE(3)$ by right multiplication i.e. $\log_{\mathrm{M}_1}(\mathrm{M}_2) = \log(\mathrm{M}_1^{-1}\mathrm{M}_2)$ and $\exp_{\mathrm{M}_1}(\mathrm{M}_2) = \mathrm{M}_1 \exp(\mathrm{M}_2)$. The reason for this construct is that the minimal length trajectory between two rigid-body motions M_1 and M_2 becomes a straight line in $\mathcal{T}_{\mathrm{M}_1}$ i.e. the tangent space of M_1 (and also in $\mathcal{T}_{\mathrm{M}_2}$ for that matter). Therefore, the intrinsic distance between two rigid-body motions M_1 and M_2 can now be expressed conveniently as

$$d(\mathrm{M}_1, \mathrm{M}_2)_\alpha = \left\|\log_{\mathrm{M}_1}(\mathrm{M}_2)\right\| = \left\|\log_{\mathrm{M}_2}(\mathrm{M}_1)\right\| \tag{18}$$

i.e. it is the Euclidean length of the tangent vector representing M_2 in the tangent space of M_1 and vice versa. Following the minimal length trajectory from A_n towards $\hat{\mathrm{A}}_n$ can therefore be performed by scaling the tangent vector representing $\hat{\mathrm{A}}_n$ in the tangent space of A_n i.e.

$$\mathfrak{I}(s) = \exp_{\mathrm{A}_n}(s \log_{\mathrm{A}_n}(\hat{\mathrm{A}}_n)) , \tag{19}$$

where $0 \leq s \leq 1$. The relative motions $\mathrm{U}_1..\mathrm{U}_n$ can therefore be obtained with

$$\mathrm{U}_m = \mathfrak{I}(\sum_{i=1}^{i=m-1} w_i)^{-1}\mathfrak{I}(\sum_{i=1}^{i=m} w_i). \tag{20}$$

Note that indeed $\mathrm{A}_n^{-1}\hat{\mathrm{A}}_n = \mathrm{U}_1..\mathrm{U}_n$ and that the since the metric eq. 15 is left invariant the magnitudes of $\mathrm{U}_1..\mathrm{U}_n$ are proportional to the weights $w_1....w_n$.

B. Maximum likelihood

It is interesting to consider for which optimization criteria the proposed closed form algorithm attains a maximum likelihood solution. To specify the appropriate objective function and constraint function each update $\hat{\mathrm{U}}_i$ is parameterized by a six dimensional vector $\hat{\mathrm{u}}_i$. This vector is a tangent vector of SE(3) at the location of its respective relative motion estimate M_i i.e. $\hat{\mathrm{u}}_i \in \mathcal{T}_{\mathrm{M}_i}$. The uncertainty of M_i is also expressed in its tangent space and it is denoted as the six by six dimensional matrix Σ_i. Generally, the uncertainty in the motion estimates is anisotropic (not of equal magnitude for each DoF.) and inhomogeneous (different for each motion estimate). The translational and rotational subspaces of $SE(3)$ also are on different units for which no natural weighting exist, hence the α in eq. 15. The uncertainty in the rotational part is modeled with $\mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{q}i})$ and is independent from the translational part which is modeled with $\mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{t}i})$ (therefore Σ_i is block diagonal). The maximum likelihood solution to the trajectory bending task can be found by minimizing the objective function

$$f(\mathrm{u}_1, ..., \mathrm{u}_n) = \sum_{i=1}^{i=n} \hat{\mathrm{u}}_i^\top \Sigma_i^{-1} \hat{\mathrm{u}}_i \tag{21}$$

with respect to $\hat{\mathrm{u}}_i....\hat{\mathrm{u}}_n$ under the non linear constraint

$$\log_{\hat{\mathrm{A}}_n}(\prod_{i=1}^{i=n} \left(\exp_{\mathrm{M}_i}(\hat{\mathrm{u}}_i)\right)) = \mathbf{0} \tag{22}$$

i.e. when the updates are applied to their relative motions the trajectory ends in the desired pose $\hat{\mathrm{A}}_n$. Apart from the non linearity of the constraint, the challenge is that the objective function is expressed as a sum over all individual tangent spaces. Due to the fact that $SE(3)$ does not admit a bi-invariant metric, Park [16], the probability of the same update is different in each tangent space (even when the noise would be isotropic and homogeneous). In other words the magnitude of a motion is not preserved when applying eq. 11. Therefore, to obtain the ML solution one does not only have to consider the constraint and the uncertainty distributions but also the local geometry of the trajectory. Intuitively speaking, the local geometry of the trajectory has to be exploited to find an ML solution. Solving this optimization task requires computational intensive non-linear optimization techniques. The remainder of this section can be skipped at first reading. The question arises: which simplifications have to be made to the objective function and the assumptions such that the closed form algorithm can be proven to provide a maximum likelihood solution. In that respect, consider the case where the uncertainty in the relative motion estimates is inhomogeneous but isotropic i.e. $\mathcal{N}(\mathbf{0}, \sigma_{\mathbf{q}i}^2\mathrm{I})$ and $\mathcal{N}(\mathbf{0}, \sigma_{\mathbf{t}i}^2\mathrm{I})$ and the ratio between $\sigma_{\mathbf{q}i}$ and $\sigma_{\mathbf{t}i}$ is similar for each motion estimate. Because then α can be set to $\frac{\sigma_{\mathbf{q}i}}{\sigma_{\mathbf{t}i}}$. This ensures isotropy i.e. $\sigma_{\mathbf{q}i} = \sigma_{\mathbf{t}i} = \sigma_i$ and

therefore all tangent vectors of equal length share the same probability in their respective tangent spaces. In this setting consider the following objective function

$$f(\mathrm{u}_1, ..., \mathrm{u}_n) = \sum_{i=1}^{i=n} \frac{\left\| \log(\mathfrak{T}_i^{-1}(\exp(\hat{\mathrm{u}}_i))) \right\|^2}{\sigma_i^2} \qquad (23)$$

which again is minimized with respect to $\hat{\mathrm{u}}_1 .... \hat{\mathrm{u}}_n$ under the same non linear constraint of eq. 22. The main difference is that this objective function is expressed in the tangent space of a common reference frame instead of over all tangent spaces individually. Now consider the solution obtained by the closed form algorithm where the weights are set according to

$$w_i = \frac{\sigma_i^2}{\sum_{i=1}^{i=n} \sigma_i^2} \ . \qquad (24)$$

Due to the isotropy a decrease in the eq. 23 can only be obtained by reducing the length of a tangent vector. The reduction causes a residual in the constraint which must be compensated by the other updates. Since the length of these compensations are measured in the same common reference frame, and since the update U is of minimal magnitude, the sum of the lengths of these compensations is equal to the length of the residual. Due to the quadratic nature of eq. 23 this cause an increase of the objective function. Therefore, the solution obtained by the closed form algorithm when using weights according to eq. 24 is the global minima of eq. 23 under constraint eq. 22 for isotropic noise. Following a similar argumentation it is clear that the solution is also unique.

III. Experiments and results

From a practical point of view it is interesting to investigate the performance differences of the closed form and ML algorithms and relate it to their computational efficiency. Both artificial as well as real data will be used to verify the applicability of the proposed algorithm. While the noise is generally anisotropic it is still desirable to choose the weights $w_1..w_n$ such that they are proportional to the uncertainty in their respective relative motion estimates $\mathrm{M}_1..\mathrm{M}_n$. A convenient value for α is given by

$$\alpha = \frac{\sum_{i=1}^{i=n} \sqrt{\mathrm{trace}(\Sigma_{\mathbf{q}i})}}{\sum_{i=1}^{i=n} \sqrt{\mathrm{trace}(\Sigma_{\mathbf{t}i})}} \ , \qquad (25)$$

where the trace function provides the sum of the diagonal elements of a matrix. The weights can then be computed with

$$w_i = \frac{\mathrm{trace}(\Sigma_{\mathbf{q}i}) + \alpha^2 \mathrm{trace}(\Sigma_{\mathbf{t}i})}{\sum_{i=1}^{i=n} \mathrm{trace}(\Sigma_{\mathbf{q}i}) + \alpha^2 \sum_{i=1}^{i=n} \mathrm{trace}(\Sigma_{\mathbf{t}i})} \ . \qquad (26)$$

For isotropic noise this automatically reduces to eq. 24. The ML optimization task is solved by employing a Sequential Quadratic Programming (SQP) algorithm. Since the desired final pose is typically a result from statistical inference, it is of little practical value to enforce it to within machine precision (as is the case for the closed form algorithm). Therefore, the SQP stops iteration when the residual of the constraint is within the one standard deviation ellipsoid of the update uncertainty. Typically it takes in the order of 5 iterations to do so. Despite the few iterations the computation time of the SQP algorithm is unfavorable for trajectories consisting of many poses. This was already noted by Newman et al. [14] for their particular approach. Their solution was to divide the trajectories in segments and only optimize with respect to those poses which link the segments together. In essence it can be seen as a sub-map approach. The same approach will be used for the ML algorithm in this research.

A. Synthetic data

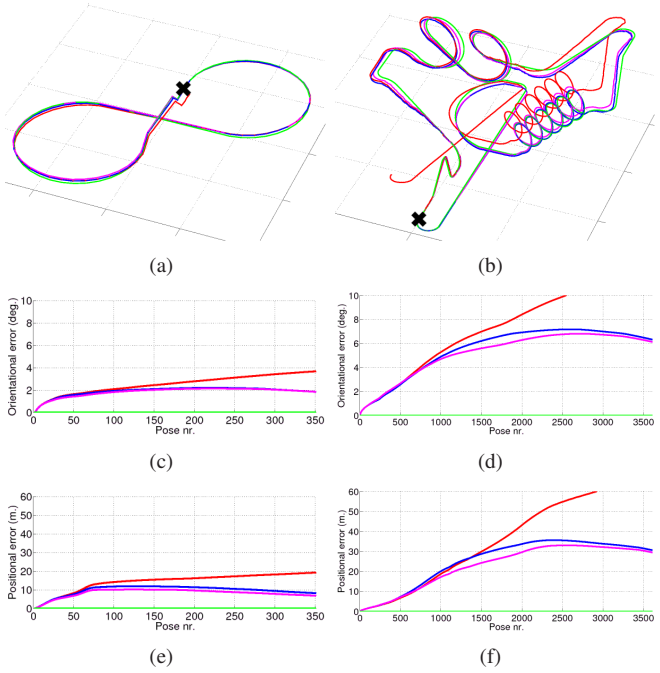
Two artificially generated trajectories will be used to compare the proposed closed-form and ML algorithms. The first trajectory consists of 350 poses, the second trajectory consists of approximately 3500 poses. For both trajectories the first pose is equal to the last pose. To each relative rigid-body motion in these trajectories inhomogeneous and anisotropic noise is added, see sec. II-B. The closed form algorithm is applied on all poses of both trajectories. When applying the ML algorithm, the second trajectory consisting of 3500 poses, is divided in 350 sub-maps of 10 poses each. The mth ground truth absolute pose is denoted as $\bar{\mathrm{A}}_m = (\bar{\mathbf{q}}_m, \bar{\mathbf{t}}_m)$. The mth absolute pose, obtained from integrating the simulated erroneous relative rigid-body motions, is denoted by $\mathrm{A}_m = (\mathbf{q}_m, \mathbf{t}_m)$. The positional error at pose number m i.e. $\mathcal{P}_m$ and the orientational error $\mathcal{O}_m$ are defined as the running means

$$\mathcal{P}_m = \frac{1}{m} \sum_{i=1}^{i=m} \| t_m - \bar{t}_m \| \qquad (27)$$

and

$$\mathcal{O}_m = \frac{180}{m\pi} \sum_{i=1}^{i=m} \left\| 2 \log(\bar{q}_m^{-1} q_m) \right\| \ . \qquad (28)$$

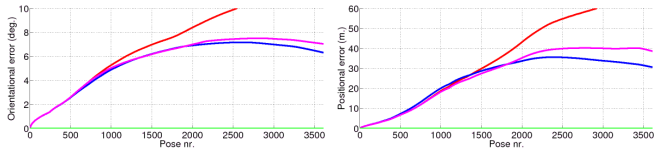
For a single experiment the trajectories are plotted in fig. 1.a and 1.b. The average positional and orientational errors over one hundred experiments are plotted in fig. 1.c,d,e, and 1.f. From the results plotted in fig. 1 it can be seen that both methods reduce the overall error for both trajectories. As expected the ML solution is more accurate, the performance differences are fractional however. The run-time of the closed-form algorithm for the first and second trajectory is 0.09 s. and 0.9 s. respectively. For the ML algorithm when using 350 poses/sub-maps the run-time is 100 s. Clearly, for an equal amount of poses the closed-form algorithm is significantly more efficient e.g. a factor 1000 times faster when using 350 poses. The computation time of the ML algorithm can be reduced by lowering the



(a) (b) (c) (d) (e) (f)

Fig. 1. Results on simulated trajectories, trajectory consisting of 350 poses (a), trajectory consisting of 3500 poses (b). Ground truth (green), no loop closure (red), closed form algorithm (blue) and ML solution when using 350 sub-maps (magenta). Running mean errors averaged over one hundred experiments (c,d,e) and (f). The black cross depicts the loop-closing point.

number of sub-maps. For example, when using five sub-maps the computation time reduces to 1.3 s. In this setting, the performance of the ML algorithm on the trajectory consisting of 3500 poses reduces significantly, as can be observed from fig. 2. The average computation times of both algorithms are summarized in table I. All implementations are based on Matlab and ran on a single core of an Intel Xeon (2.66GHz) processor. Considerable speed-ups can therefore be expected when implementing both algorithms in a more efficient programming language. When doing so, the relative differences in computation time between both algorithms are expected to remain similar.



(a) (b)

Fig. 2. Results on the simulated trajectory consisting of 3500 poses (fig. 1.b). Ground truth (green), no loop closure (red), closed form algorithm (blue) and ML algorithm when using 5 sub-maps (magenta).

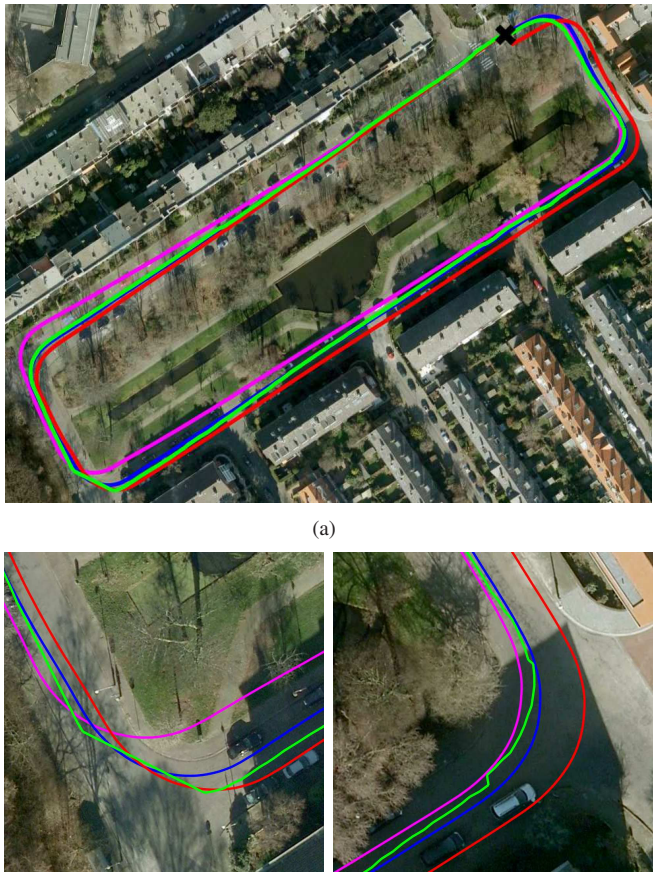
TABLE I

AVERAGE COMPUTATION TIME ON SYNTHETIC DATA

Algorithm	350 poses	3500 poses	350 sub-maps	5 sub-maps
Closed form	0.09 s.	0.9 s.		
ML	100 s.	+10 h.	100 s.	1.3 s.

B. Binocular data

The binocular data set, consisting of approx. 1200 images, is recorded using a stereo camera with a baseline of 0.4 m., a resolution of 640 by 480 pixels and a FoV. of 45 deg. The stereo camera was mounted on a vehicle which drove an urban trajectory of approximately 600 m. The relative rigid-body motions are estimated using the EM-SE(3) algorithm of Dubbelman et al. [5]. This algorithm applies intrinsic robust clustering on motion hypotheses which are generated using a fundamental subset strategy. The interesting property of this algorithm is that besides a robust estimate of the relative rigid-body motion, it also returns a monte carlo estimate of the covariance matrix which is expressed in the tangent space of the robust estimate. These covariance matrices are used during trajectory bending. The estimated trajectory exist of approx. 1200 poses. When using the ML approach the trajectory is again segmented in 350 sub-maps. The results are plotted in fig 3. It can be seen that both approaches



(a)

(b) (c)

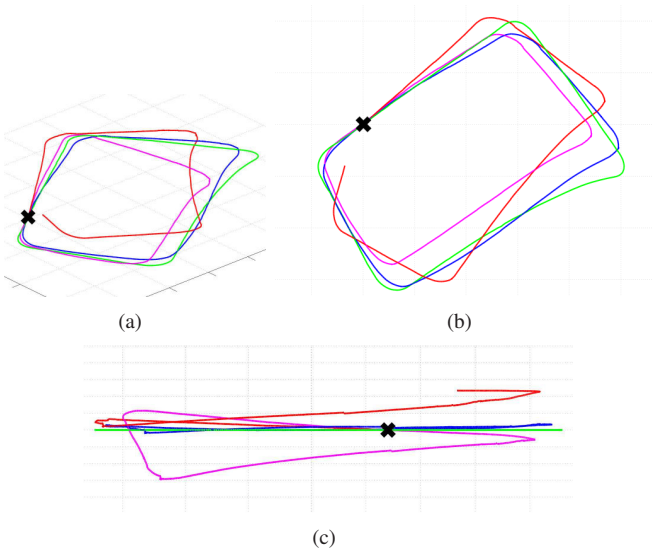
Fig. 3. Results on trajectory consisting of approx. 1200 poses estimated from binocular data, arial image overlay (a), close-up of lower left corner (b), close-up of lower right corner (c). Binocular odometry (red), closed form algorithm (blue), ML solution (magenta) and differential GPS (green). The black cross depicts the loop-closing point.

close the loop satisfactory, however, the trajectory obtained with the ML approach differs significantly from the ground truth. In contrary to the simulation of sec. III-A where the noise characteristics were known exactly, here the true noise

characteristics of each relative motion is unknown and is approximated by a Gaussian pdf. Recent research indicates that the noise in motion estimates is typically biased (or skewed) instead of pure Gaussian [6]. The approximation of the true unknown noise characteristics by a Gaussian pdf influences the ML solution significantly more than is does influence the closed form approach. It seems that the cost function in eq. 23 optimized by the closed form approach is better suited for the underlying error structure of visual-odometry. Further research on this topic is required. It is interesting to observe that the accuracy of differential GPS is limited for certain segments of the trajectory. A quantitative comparison between the closed form and ML solution on basis of differential GPS is therefore omitted. The computation time of the closed form algorithm is approximately 0.4 s. for the ML solution using 350 sub-maps it is again 100 s.

C. Monocular data

The monocular odometry data set is recorded using a camera with a resolution of 640 by 480 pixels and a FoV. of 45 deg. A total approx. 2000 images are recorded covering a total distance of approx. 900 m. The camera motion is estimated robustly using RANSAC, where the normalized 8-point algorithm [10] is used to generate the fundamental subset hypotheses. That hypothesis which has the largest number of inliers, based on the Sampson error, is selected. From the inliers of this best hypothesis a new motion estimate is obtained. A local iterative refinement step, which minimizes reprojection errors, is used to optimize both the camera pose and the camera calibration. This approach offers a favorable tradeoff between satisfactory accuracy and computational efficiency. The results on the monocular data set are plotted in fig. 4. It can be seen that the monocular odometry algorithm induces a drift in the trajectory that accumulates over time. By using the proposed trajectory bending algorithm this drift is reduced. This is particulary visible in fig. 4.c where the estimated height of each pose is plotted. Note that while the ML algorithm (again using 350 sub-maps) closed the loop satisfactory, its trajectory differs significantly from the ground-truth. Again it seems that the cost function of the closed form approach is better suited for the underlying error structure of visual-odometry. Furthermore, the difference in computation time was significantly larger, i.e. 0.6 s. for the closed form algorithm and 300 s. for the ML algorithm. Next the applicability of the proposed closed form algorithm with respect to monocular 3D reconstruction will be illustrated. To this purpose 85 images are recorded with a hand held Digital Reflex Camera (Canon 350D) mounted with a 10 mm. wide angle lens. The images have a resolution of 1728 by 1152 pixels and cover a trajectory of approx. 80 m. Two example images of this data set are shown in fig. 5.c and d. Once the camera poses are recovered, by the same monocular odometry procedure as described earlier, a 3D model is estimated using [7]. By employing the closed form algorithm the camera poses can be improved, after which, the model can be re-estimated. In fig. 5 the reconstructed model with and without closed form trajectory bending is



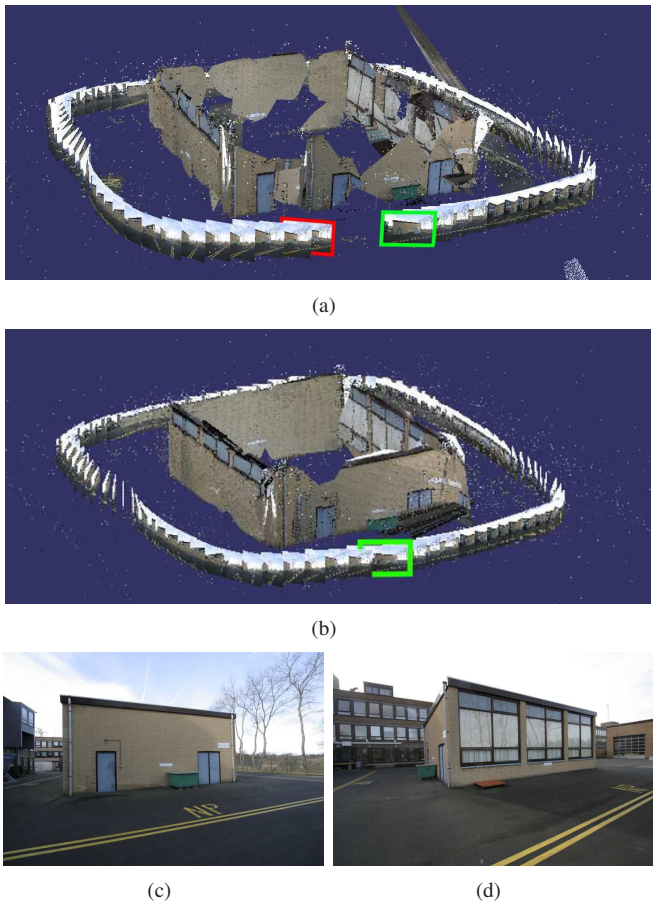
(d)

Fig. 4. Results on trajectory consisting of approx. 2000 poses estimated from monocular data (a), top-view (b), side-view (c), arial image overlay (d). Monocular odometry (red), closed form algorithm (blue), ML solution (magenta) and differential GPS (green). The black cross depicts the loop-closing point.

shown. From fig. 5 it is clear that the closed form algorithm improved the trajectory estimates such that the loop is closed. Consequently, this improvement in pose estimates enables the 3D reconstruction algorithm to estimate a more accurate model.

IV. Conclusion

A closed form algorithm is presented which bends a trajectory consisting of relative rigid body motions such that the final absolute pose is exactly equal to a desired pose. Within this contribution its applicability with respect to loop closure is investigated. It is compared against a maximum likelihood solution based on sequential quadratic programming. From experiments using synthetic, binocular and monocular data it can be concluded that the proposed



(a) (b) (c) (d)

Fig. 5. Monocular 3D reconstruction, without closed form trajectory bending (a), with closed form trajectory bending (b). First and last frame of trajectories are outlined in red and green respectively. Example images used for reconstruction, a front-view image (c), a side-view image (d).

closed form approach attains satisfactory results on large trajectories while its computation time is fractional. On basis of experiments using synthetic data it can be concluded that the maximum likelihood solution is in essence more accurate than the closed form approach. Its computational complexity, however, prevents it from being applied to each pose for realistically large trajectories. When the maximum likelihood solution is applied using a sub-map strategy it can become less accurate than the closed form approach while still requiring more computation time. This makes the proposed algorithm for many applications the algorithm of choice for trajectory optimization during loop closure. Furthermore, its computational efficiency enables its use for embedded applications where computational resources are typically limited.

REFERENCES

[1] C. Altafini, "The de casteljau algorithm on se(3)," *Lecture Notes in Control and Information Sciences*, vol. 258, pp. 23–34, 2000.

[2] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and . Eduardo Nebot. IROS, "Consistency of the ekf-slam algorithm," in *IEEE/RSJ Conference on Itelligent Robots and Systems*, 2006.

[3] J. A. Castellanos, J. Neira, and J. D. Tardos, "Limits to the consistency of ekf-based slam," in *5th IFAC Symp. on Intelligent Autonomous Vehicles*, July 2004.

[4] A. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3d visual odometry," in *IEEE International Conference on Robotics and Automation*, April 2007, pp. 40–45.

[5] G. Dubbelman, W. van der Mark, and F. C. A. Groen, "Accurate and robust ego-motion estimation using expectation maximization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2008, pp. 3914–3920.

[6] G. Dubbelman and F. Groen, "Bias reduction for stereo based motion estimation with applications to large scale visual odometry," in *International conference on Computer Vision and Pattern Recognition*, June 2009, pp. 1–8.

[7] I. Esteban, J. Dijk, and F. Groen., "Fit3d toolbox: multiple view geometry and 3d reconstruction for matlab," in *SPIE International Symposium on Security & Defence Europe*, 2010.

[8] F. S. Grassia, "Practical parameterization of rotations using the exponential map," *Journal of Graphics Tools*, vol. 3, no. 3, pp. 29 – 48, 1998.

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

[10] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, p. 580593, 1997.

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

[12] K. Konolige, M. Agrawal, and J. Solà, "Large scale visual odometry for rough terrain," in *International Symposium on Research in Robotics*, 2007.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[14] P. Newman, D. Cole, and K. Ho, "Outdoor slam using visual appearance and laser ranging," in *IEEE International Conference on Robotics and Automation*, 2006.

[15] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, January 2006.

[16] F. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Transactions of the ASME*, vol. 117, pp. 48–54, March 1995.

[17] J. M. Selig, *Geometrical Methods in Robotics*, D. Gries and F. B. Schneider, Eds. Springer, 1996.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[19] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *International Workshop on Vision Algorithms: Theory and Practice*, 1999, pp. 298 – 372.

[1]The terminology of logarithmic map and exponential map used in this paper is related to Riemannian geometry. Similar and related terminology exist in the theory of Lie algebra. For some algebraic elements, such as quaternions, these mappings are the same within Riemannian geometry and Lie algebra. Generally however these mappings are not necessarily related. Intuitively speaking, the mappings in Riemannian geometry are used to transfer points, in an intrinsic distance and direction preserving manner, between the manifold and the tangent space attached to a specific development point on the manifold. They do however not necessarily share algebraic properties with the usual exponent and logarithm on natural numbers. The Lie algebraic mappings do focus more on the group algebraic properties, however they are no longer necessarily related to intrinsic distance and direction [16]. An example is the Lie algebraic logarithm for elements for $SE(3)$, Selig [17], which can not be used to specify the intrinsic distance of eq. 15 as the Euclidean length of the tangent vector (i.e. element of the Lie algebra). When using these Lie algebraic mappings for interpolating between rigid-body motions A_n and $\hat{\mathrm{A}}_n$ one obtains an Ad-invariant curve starting at A_n and ending at $\hat{\mathrm{A}}_n$ [1]. This curve does not describe the minimal length trajectory in $\mathbb{R}^3$, therefore, using the Lie algebraic mappings is unfavorable for the application of trajectory bending.