Orientation Only Loop-closing with Closed-form Trajectory Bending

Gijs Dubbelman, Peter Hansen, Brett Browning and M. Bernardine Dias

***Abstract*— In earlier work closed-form trajectory bending was shown to provide an efficient and accurate *out-of-core* solution for loop-closing *exactly sparse* trajectories. Here we extend it to fuse exactly sparse trajectories, obtained from relative pose estimates, with absolute orientation data. This allows us to *close-the-loop* using absolute orientation data only. The benefit is that our approach does not rely on the observations from which the trajectory was estimated nor on the probabilistic links between poses in the trajectory. It therefore is highly efficient. The proposed method is compared against regular fusion and an iterative trajectory bending solution using a 5 km long urban trajectory. Proofs concerning optimality of our method are provided.**

I. Introduction

Estimating the pose of a moving system from on-board sensors is important for many application domains. One of the currently most researched methods is that of visual pose estimation or *visual odometry* (VO). In recent years visual odometry has gained significantly in accuracy, see for example [4], [14], [15], [17], [23]. Structural error build up, also known as *drift*, is inevitable however [5], [9] This is due to the fact that the VO solution is obtained from relative pose estimates to which no correction in terms of absolute pose information is applied. Often, accurate absolute pose information is available, be it at a significant lower rate than the visual data on which the relative poses are estimated. The goal of the presented method is to integrate this absolute pose information into the VO solution at the time it is registered, i.e. *online* such that it reduces drift and improves the estimated trajectory entirely (i.e. all poses from the first time-step up to the current time-step). Our method can do so without using the observations from which the relative poses were estimated. This makes it significantly more efficient than methods which rely on observations, such as e.g. *bundle adjustment* (BA) [21] or certain *Simultaneous Localization and Mapping* (SLAM) approaches [2], [3], [11].

In previous work [8] the absolute pose information was derived from *loop-detection* [1]. However, detecting loops is not always possible because moving in loops is not an effective strategy to reach a desired destination. Many realistic trajectories contain no loops at all, they are *loop-less*. In order to reduce drift in estimated loop-less trajectories one can use on-board absolute pose sensors. The most well known are based on the *global position system* (GPS). This is not always a viable solution as it, for example, requires line of sight to several satellites orbiting the earth. In this work we therefore focus on using an *attitude heading reference system* (AHRS) to provide absolute orientation data. An AHRS typically consists of *gyroscopes*, *accelerometers* and *magnetometers*. By fusing the information from these sensors an AHRS can deduce absolute orientation information. AHRS devices based on *micro-electro-mechanical systems* (MEMS) are affordable and widely available. These MEMS based AHRS systems are error prone when subjected to acceleration. When little acceleration is exercised however, they can provide accurate absolute orientation information.

The task is to use these accurate AHRS readings to not only update the current pose (as in regular fusion) but to update all previous poses as well without relying on visual observations or on the probabilistic links between poses (as in e.g. graph based SLAM). The method of closed-form trajectory bending can do so by applying constraint optimization on the manifold of rotations, basically bending the trajectory such that it ends in the desired orientation. The way the trajectory bends at each pose is determined by the uncertainty in the relative pose estimates. We show that our method has linear computational complexity in the number of poses while its memory requirements are constant for first-order memory (e.g. processor cash) and linear for second-order memory (e.g. flash). It can therefore compute its solution *out-of-core*.

Once improved orientation estimates are obtained, the positional estimates can be improved as well. This is because long term drift in VO trajectories are mainly driven by locally correlated relative rotational errors. This, in a sense, allows for *closing-the-loop* using absolute orientation data only. Here our approach is used *stand-alone* but its true utility may lie in initialization of more complex (non-linear) optimizers such as [12], [13], [22]. This will make them more efficient and more robust against local minima of the underlying (non-linear) objective function. Our approach can easily be incorporated into such optimization strategies and its source code is available at [7]. Furthermore, it can be applied to general trajectories and not only to those estimated by VO.

In Sec. II a detailed mathematical analysis of closed-form trajectory bending is provided. In this section we also discuss its closest related alternative: iterative trajectory bending [16]. Proofs concerning the convergence and optimality of our approach are also provided. These proofs were not present in [8] and they are an important contribution of this work. In Sec. III our closed-form approach, a regular fusion approach and an iterative approach are evaluated on a real 5 km long urban trajectory. This trajectory is estimated using

This report was made possible by the support of an NPRP grant from the Qatar National Research Fund. The statements made herein are solely the responsibility of the authors. The authors are with CMU's Robotics Institute, gijsdubb@cs.cmu.edu, phansen@qatar.cmu.edu, mbdias@ri.cmu.edu, brettb@cs.cmu.edu

binocular VO and its dataset is available at [7].

II. Trajectory bending

Consider a trajectory consisting of n relative pose estimates $\mathrm{M}_1, ..., \mathrm{M}_n$. Each relative pose is an element of $\mathrm{SE}(3)$ and consists of a rotation $\mathrm{R}_n \in \mathrm{SO}(3)$ and a translation $\mathbf{t}_n \in \mathrm{R}^3$. An estimate for the absolute pose at time-step n, denoted as A_n, is obtained by concatenating relative pose estimates with

$$\mathrm{A}_n = \mathrm{A}_0\mathrm{M}_1...\mathrm{M}_n = \mathrm{A}_0 \prod_{i=1}^{n} \mathrm{M}_i \ . \tag{1}$$

The absolute coordinate frame A_0 can be assumed to be I without loss of generality. Such a trajectory is exactly sparse as there are probabilistic links between successive poses only. These links are governed by the uncertainty in the relative pose estimates.

At time-step n the system receives additional information regarding its current absolute orientation. This *orientation update* can be fused with the current estimate, obtained from the relative pose estimates, resulting in an improved estimate for the current absolute orientation. This improved orientation estimate obtained after fusion is denoted as $\mathrm{D}_n \in \mathrm{SO}(3)$. It will automatically improve the positions and orientations for all all future poses, i.e. after n, in the trajectory.

The goal of trajectory bending is to also improve all previous orientations and positions, i.e. prior to n, using the orientation update D_n.

A. Iterative bending in $\mathrm{SO}(3)$

We start by considering an iterative *maximum-likelihood* (ML) solution. In spirit it is similar to the method proposed in [16], although in their work it was used for regular loop-closing and not for orientation-only loop-closing. We therefore modified their approach to work in the rotational subspace only.

When focusing on the rotational subspace, Eq. 1 reduces to

$$\mathrm{B}_n = \mathrm{B}_0\mathrm{R}_1...\mathrm{R}_n = \mathrm{B}_0 \prod_{i=1}^{n} \mathrm{R}_i \ , \tag{2}$$

where B_n is the absolute orientation of the nth pose in the trajectory and B_0 is again I. The goal is to update each $\mathrm{R}_1, ..., \mathrm{R}_n$ such that the final absolute orientation B_n becomes equal to D_n. When every relative orientation is improved the absolute positions can be improved by recomputing them with Eq. 1.

By using an intrinsic parametrization, see Appendix B, of the updates the constraint in the final pose can be expressed mathematically as

$$\| \log_{\mathrm{D}_n}(\prod_{i=1}^{n} \left(\exp_{\mathrm{R}_i}(\mathfrak{u}_i)\right))\|^2 = 0 \ . \tag{3}$$

This formula updates each relative orientation estimate by a *distributed correction term*, e.g. $\mathfrak{u}_i$, parametrized in their respective tangent spaces, e.g. $\exp_{\mathrm{R}_i}(\hat{\mathfrak{u}}_i)$. Then it computes the final absolute orientation of the trajectory by multiplying each updated relative orientation estimate. By mapping the final absolute orientation to the tangent space of the desired absolute orientation D_n and computing the squared length of the resulting tangent vector, the value of the constraint is obtained. It is zero only when the final orientation B_n is equal to the desired orientation D_n.

There are clearly infinitely many solutions for $\mathfrak{u}_1, ..., \mathfrak{u}_n$ which fulfill this constraint. The interest is however in obtaining a maximum likelihood solution. To this purpose the uncertainties of the relative orientation estimates $\mathrm{R}_1...\mathrm{R}_n$ are modeled as normal distributions with zero mean and covariances $\Sigma_1...\Sigma_n$ in their respective tangent spaces. The goal of ML trajectory bending is to find those distributed correction terms $\mathfrak{u}_1, ..., \mathfrak{u}_n$ for which their likelihood given these normally distributed uncertainties is maximized. This task is expressed by the ML objective function

$$f(\mathfrak{u}_1, ..., \mathfrak{u}_n) = \frac{1}{2} \sum_{i=1}^{n} \mathfrak{u}_i^\top \Sigma_i^{-1} \mathfrak{u}_i \ . \tag{4}$$

In spirit of the approach in [16] the objective function Eq.4 is minimized with respect to $\mathfrak{u}_1, ..., \mathfrak{u}_n$ under the non-linear constraint expressed by Eq. 3 by using *sequential quadratic programming*(SQP). Due to the non-linearities in the constraint this is a demanding optimization task which requires significant computation and is susceptible to local minima. For trajectories consisting out of many poses (i.e. $100 < n$) it is not possible to compute a solution online on current (portable) computational hardware.

Therefore, in order to provide a workable solution, a sub-mapping approach was employed in [16]. Basically, splitting the trajectory in several (i.e. 50) segments and only optimizing with respect to the relative poses connecting those segments. Such a sub-mapping approach does not offer the same level of optimality as optimization with respect to every pose of the trajectory. Nevertheless, in [16] it was shown to provide better results than loop-closing using EKF based trajectory-SLAM.

B. Closed-form bending in $\mathrm{SO}(3)$

We now proof that, when the uncertainties in the relative orientation estimates are isotropic, the ML constrained Eq. 3 and objective function Eq. 4 have a closed-form solution. These proofs extend the work in [8].

We start with considering the constraint which can be rewritten (see Appendix B) as

$$\| \log(\mathrm{D}_n^{-1} \prod_{i=1}^{n} (\mathrm{R}_i\mathrm{U}_i))\|^2 = 0 \ , \tag{5}$$

where each distributed correction term $\mathrm{U}_i = \exp(\mathfrak{u}_i)$. A possible solution for $\mathrm{U}_1, ..., \mathrm{U}_n$ that satisfies the constraint is setting all $\mathrm{U}_1, ..., \mathrm{U}_{n-1}$ to I and setting U_n to $\mathrm{B}_n^{-1}\mathrm{D}_n$. Indeed $\| \log(\mathrm{D}_n^{-1}\mathrm{R}_1...\mathrm{R}_{n-1}\mathrm{R}_n\mathrm{B}_n^{-1}\mathrm{D}_n)\| = \| \log(\mathrm{D}_n^{-1}\mathrm{B}_n\mathrm{B}_n^{-1}\mathrm{D}_n)\| = \| \log(\mathrm{I})\| = 0$. This solution basically leaves the trajectory unchanged and only adds the update $\mathrm{B}_n^{-1}\mathrm{D}_n$ at the end of the trajectory after R_n such that it ends in the desired orientation D_n.

The next step is to distribute this one-step update $\mathrm{B}_n^{-1}\mathrm{D}_n$ over the entire trajectory such that every orientation improves. To this purpose we first segment $\mathrm{B}_n^{-1}\mathrm{D}_n$ in n relative orientations $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$ such that

$$\mathrm{B}_n^{-1}\mathrm{D}_n = \prod_{i=1}^{n} \hat{\mathrm{U}}_i. \tag{6}$$

The elements $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$ are the *local correction terms* and again $\| \log(\mathrm{D}_n^{-1}\mathrm{R}_1...\mathrm{R}_n\hat{\mathrm{U}}_1...\hat{\mathrm{U}}_n\| = 0$; the constraint is satisfied.

These local correction terms are obtained by interpolating $\mathrm{B}_n^{-1}\mathrm{D}_n$ with

$$\hat{\mathrm{U}}_j = \mathfrak{I}(\sum_{i=1}^{j-1} w_i)^{-1}\mathfrak{I}(\sum_{i=1}^{j} w_i), \tag{7}$$

where

$$\mathfrak{I}(t) = \exp_{\mathrm{B}_n}(t \log_{\mathrm{B}_n}(\mathrm{D}_n)). \tag{8}$$

The weights $w_1, ..., w_n$ with $w_1+w_2+...+w_n = 1$ determine how much of the update $\mathrm{B}_n^{-1}\mathrm{D}_n$ will be distributed to a particular pose in the trajectory. The larger the weight w_i the more bending will occur at the ith pose in the trajectory. In Sec. II-C we show that a ML solution is obtained by setting these weights proportional to the variance in the relative orientation estimates.

So far we have obtained the local correction terms $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$. The final step is distributing them over the trajectory, i.e. specifying a mapping $\mathfrak{T}$ which takes the local correction terms $\hat{\mathrm{U}}_1...\hat{\mathrm{U}}_n$ to the distributed correction terms $\mathrm{U}_1...\mathrm{U}_n$ of Eq. 5 such that

$$\left(\prod_{i=1}^{n} \mathrm{R}_i\right)\left(\prod_{i=1}^{n} \hat{\mathrm{U}}_i\right) = \prod_{i=1}^{n}\left(\mathrm{R}_i\mathfrak{T}(\hat{\mathrm{U}}_i)\right) = \mathrm{D}_n. \tag{9}$$

This will add a correction term, e.g. $\mathfrak{T}(\hat{\mathrm{U}}_i)$, to each pose in the trajectory assuring it ends in D_n.

In essence this mapping $\mathfrak{T}$ is nothing more than a change of basis applied to each of the local correction terms $\hat{\mathrm{U}}_1...\hat{\mathrm{U}}_n$. The difficultly however is that the change of basis is not unique. This is due to the fact that we have a choice in which order to process the local correction terms. For n local correction terms there are $n!$ distinct orders. Given a general order, the mapping $\mathfrak{T}$ for a single correction term $\hat{\mathrm{U}}_j$ depends on other correction terms as well. This prevents efficient computation, as it requires recomputing the trajectory after each single correction term has been processed by $\mathfrak{T}$. Here we are interested in a mapping $\mathfrak{T}$ for which such reintegration is not required and therefore is significantly more efficient. It turns out that such a mapping exists, it is provided by the following theorem.

Theorem 1: A relation between the local correction terms $\hat{\mathrm{U}}_1...\hat{\mathrm{U}}_n$ and the distributed correction terms $\mathrm{U}_1...\mathrm{U}_n$ obeying the constraint Eq. 5 is provided by

$$\begin{aligned} \mathfrak{T}(\hat{\mathrm{U}}_j) &= \left(\prod_{i=1}^{j} \mathrm{R}_i\right)^{-1} \mathrm{D}_n\hat{\mathrm{U}}_j\mathrm{D}_n^{-1}\left(\prod_{i=1}^{j} \mathrm{R}_i\right) \\ &= \mathrm{B}_j^{-1}\mathrm{D}_n\hat{\mathrm{U}}_j\mathrm{D}_n^{-1}\mathrm{B}_j \\ &= \mathrm{U}_j \end{aligned} \tag{10}$$

The proof for this theorem is provided in Appendix A and guarantees that our method returns a solution which satisfies the constraint (for any set of normalized weights $w_1, ..., w_n$).

The interesting property is that the mapping $\mathfrak{T}$ for each $\hat{\mathrm{U}}_j$ only depends on the original absolute orientation B_j and the desired final absolute orientation D_n. The solution to trajectory bending can therefore be obtained in closed form and has computational complexity $\mathrm{O}(n)$. As the mapping can be computed independently for each $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$ the memory requirements are constant for first-order memory (e.g. processor cache) and linear in n for second-order memory (e.g. flash). This allows for out-of-core processing and makes our trajectory bending algorithm significantly more efficient than alternative (ML) approaches such as that of [16].

For further sections it is useful to define the inverse of $\mathfrak{T}$ as

$$\hat{\mathrm{U}}_j = \mathfrak{T}^{-1}(\mathrm{U}_j) = \mathrm{D}_n^{-1}\mathrm{B}_j\mathrm{U}_j\mathrm{B}_j^{-1}\mathrm{D}_n\ . \tag{11}$$

It maps distributed correction terms back to local corrections terms.

C. proof of optimality

In this section we derive that our algorithm provides a ML solution when the uncertainties in the relative orientation estimates are isotropic and the weights $w_1, ..., w_n$ are set proportional to these uncertainties.

We start with the ML constraint Eq. 3. This constraint is specified using the distributed correction terms. In Sec. II-B we showed that this constraint can also be satisfied using the local correction terms $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$ together with the mapping $\mathfrak{T}$ defined in Eq. 10. The constraint Eq. 3 is therefore conceptually similar to the constraint

$$\mathrm{B}_n^{-1}\mathrm{D}_n = \prod_{i=1}^{n} \hat{\mathrm{U}}_i. \tag{12}$$

When this constraint is satisfied, then so is the original constraint of Eq. 3. Our approach even enforces this constraint within machine precision.

Now we focus on the ML objective function Eq. 4. In the case of isotropic and inhomogeneous noise in SO(3) the covariance matrices in Eq. 4 are of the form

$$\Sigma_i = \sigma_i^2\mathrm{I} \tag{13}$$

and the ML objective function reduces to

$$f(\mathfrak{r}_1, ..., \mathfrak{r}_n) = \frac{1}{2}\sum_{i=1}^{n} \frac{\|\mathfrak{r}_i\|^2}{\sigma_i^2} = \frac{1}{2}\sum_{i=1}^{n} \frac{\|\log(\mathrm{U}_i)\|^2}{\sigma_i^2}. \tag{14}$$

The following well known equality, e.g. see [18], is important

$$\|\log(\mathrm{U}_j)\| = \left\|\log(\mathrm{R}\mathrm{U}_j\mathrm{R}^{-1})\right\|, \tag{15}$$

with R and U_j elements of $\mathrm{SO}(3)$. It is a direct consequence of the fact that the norm on rotations is left and right invariant with respect to a change of basis in $\mathrm{SO}(3)$. The importance of this equality is that we are allowed to apply any change of basis in SO(3) to each $\mathrm{U}_1, ..., \mathrm{U}_n$ in the ML objective function Eq. 14. We are therefore also allowed to apply the change of basis $\mathfrak{T}^{-1}$ defined in Eq. 11. This change of basis maps a distributed correction U_i term back to its related local correction term $\hat{\mathrm{U}}_i$. The ML objective function can therefore be rewritten as

$$\frac{1}{2}\sum_{i=1}^{n}\frac{\|\log(\mathfrak{T}^{-1}(\mathrm{U}_i))\|^2}{\sigma_i^2} = \frac{1}{2}\sum_{i=1}^{n}\frac{\|\log(\hat{\mathrm{U}}_i)\|^2}{\sigma_i^2}, \qquad (16)$$

What we have gained is that, just as the constrained Eq. 12, the ML objective is now specified using local correction terms $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$ instead of distributed correction terms $\mathrm{U}_1, ..., \mathrm{U}_n$.

Thus, when the noise in the relative rotational estimates is isotropic, minimizing the ML objective Eq. 4 wrt. $\mathrm{U}_1, ..., \mathrm{U}_n$ under the constraint Eq.3 is similar to minimizing

$$\frac{1}{2}\sum_{i=1}^{n}\frac{\|\log(\hat{\mathrm{U}}_i)\|^2}{\sigma_i^2} \quad \text{with constraint} \quad \mathrm{B}_n^{-1}\mathrm{D}_n = \prod_{i=1}^{n}\hat{\mathrm{U}}_i \qquad (17)$$

wrt. $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$. This optimization task basically states: finding a path through $\mathrm{SO}(3)$, made up of steps $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$, starting at I and ending in $\mathrm{B}_n^{-1}\mathrm{D}_n$ such that the summed squared lengths of these segments is minimal with respect to $\sigma_1^2, ..., \sigma_n^2$.

In order for the summed squared step lengths to be minimal, the path itself must be the minimal length path from I to $\mathrm{B}_n^{-1}\mathrm{D}_n$. Such paths through $\mathrm{SO}(3)$ are well known and called minimizing geodesic [6]. As our interpolation function Eq. 8 involves the Riemannian logarithm and exponent the path traced out by $\hat{\mathrm{U}}_1, ..., \hat{\mathrm{U}}_n$ follows this minimizing geodesic for any set of normalized weights $w_1, ..., w_n$.

The remaining question is therefore: which particular values to use for the weights $w_1, ..., w_n$ such that an optimal solution is obtained. Consider that the length of the path from I to $\mathrm{B}_n^{-1}\mathrm{D}_n$ is $\|\log(\mathrm{B}_n^{-1}\mathrm{D}_n)\|$. The length of each step is

$$\|\log(\hat{\mathrm{U}}_i)\| = w_i\|\log(\mathrm{B}_n^{-1}\mathrm{D}_n)\|. \qquad (18)$$

Plugging the right hand side of this equality into the ML objective of Eq. 17 we get

$$\frac{1}{2}\sum_{i=1}^{n}\frac{w_i^2}{\sigma_i^2} \qquad (19)$$

(where we also took out $\|\log(\mathrm{B}_n^{-1}\mathrm{D}_n)\|^2$ because it is a constant). This new objective function must be minimized with respect to $w_1, ..., w_n$ under the constraint

$$1 = \sum_{i=1}^{n} w_i. \qquad (20)$$

(a) (b)

Fig. 1. Representative images from the 5 km long binocular urban data set made available at [7].

By using the method of Lagrange multipliers we find that the optimal value for each weight is provided by

$$w_j = \frac{\sigma_j^2}{\sum_{i=1}^{n}\sigma_i^2}. \qquad (21)$$

Each weight and therefore the length of each step over the geodesic is proportional to the uncertainty, i.e. the variance, in its corresponding relative orientation estimate. The more uncertainty in the relative orientation estimate the more bending will occur at its location in the trajectory. This clearly agrees with an intuitive understanding of statistically informed trajectory bending.

When considering the argumentation concerning optimality in [8], then the fact that the combined norm on $\mathbb{R}^3 \times \mathrm{SO}(3)$ is not invariant with respect to a change of basis in SE(3), e.g. see [18], prevents rewriting the ML objective as in Eq. 16. Therefore, in contrast to Eq. 16, the objective function in [8] is not related to the original ML objective function.

III. Evaluation

In this section our closed-form approach is compared against regular fusion and against its closest related alternative: iterative trajectory bending. The comparison is based on a 5 km long urban trajectory estimated by binocular VO. This dataset consists out of approximately ten thousand stereo image pairs from which 2000 poses are estimated. The dataset is accompanied by D-GPS based ground truth. It is one of the most challenging publicly available VO datasets. This is mainly due to many stop-and-go traffic situations, e.g. see Fig. 1, during which large regions of the image are occupied by independent moving objects.

Our experimental setup, incorporating closed-form trajectory bending, comprises the following:

1) *Running visual odometry*: We use the approach of [10] with automatic key-framing as in [23] and inlier-outlier tracking to suppress independent moving objects. Besides the relative poses themselves we also estimate their uncertainties.

2) *Detecting time-windows of low acceleration*: The absolute orientation data is provided by an affordable *micro-electro-mechanical* based AHRS. This AHRS is used as a black-box and we directly work with its fused output. The

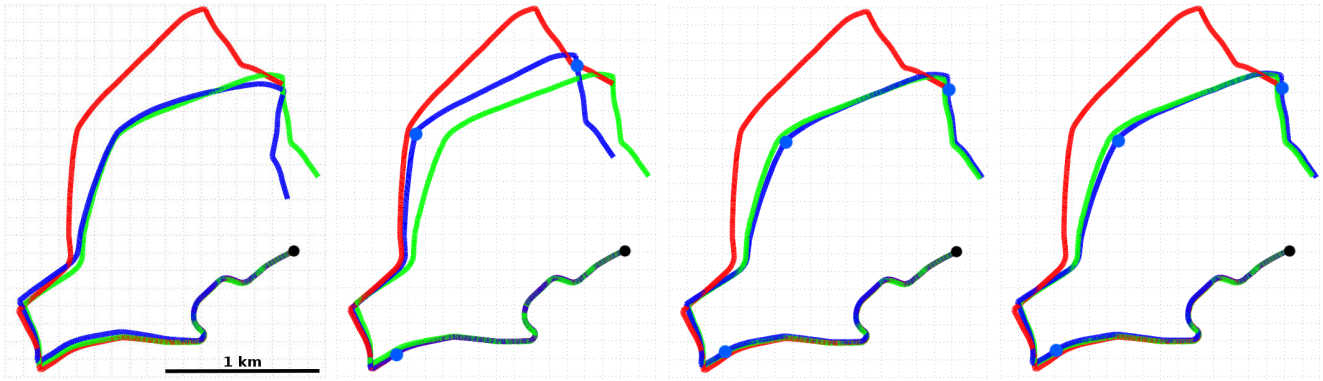


Fig. 3. Results of the four tested approaches: regular fusion (a), with time-window selection (b), closed-form trajectory bending (c), iterative trajectory bending (d). The VO trajectory is depicted in red, the D-GPS based ground truth in green and the results obtained after applying each approach in blue. The selected time-windows of low acceleration are depicted by the blue dots on the trajectories.

AHRS is error prone when subjected to (angular) acceleration. In case of low acceleration, it provides relatively reliable absolute orientation readings. We therefore choose to only use the AHRS readings within time-windows of low acceleration. They are detected by first measuring the acceleration from the VO estimates (we use the VO estimates because they exhibit high short-term accuracy). Thresholds are applied to the measured accelerations and we only use time-windows of at least 5 seconds long. The threshold on the translational acceleration is set to pass below 20 mm/s^2 and the threshold on the angular acceleration is set to pass below 0.02 deg./s^2. For the 5 km long dataset only three time-windows of low acceleration are detected.

3) *Fusing the VO absolute orientation estimate with the AHRS readings*: When a suitable time-window is detected the ARHS orientation readings are fused with estimates obtained from VO thereby providing the update D_n. We choose to do so by computing an intrinsic weighted mean [19] of all visual odometry and AHRS estimates falling within the time-window. These visual odometry and AHRS estimates provide a Monte-Carlo representation of their underlying uncertainties. This fusion approach is therefore not limited to Normal distributions neither requires linearization. The intrinsic mean is that point in $\mathrm{SO}(3)$ which provides a ML estimate given the uncertainties. As there are typically significantly more AHRS readings (it operates at 60 Hertz) within a time-window than VO estimates, the VO estimates are re-weighted such that the total VO weight $\alpha = 0.65$ times that of the total AHRS weight. After the update is obtained, the AHRS is reset to equal D_n. This suppresses drift of the fusion inside the AHRS itself.

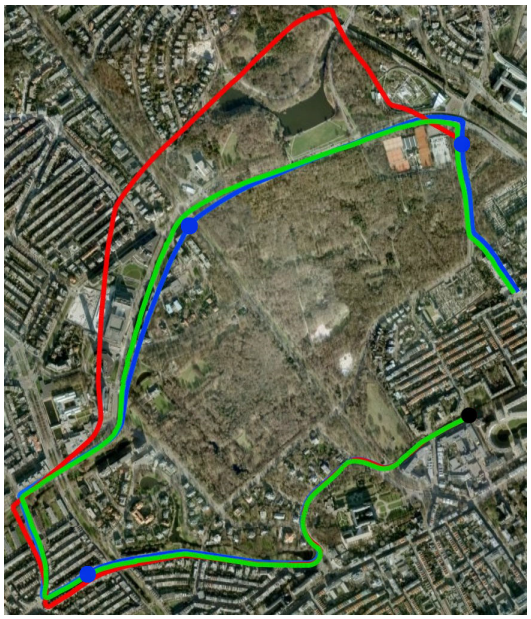
4) *Applying trajectory bending*: At this point we only have an update D_n for the current time-step. Trajectory bending uses this update to improve all relative VO orientation estimates up to the previous detected time-window. This assures that the trajectory ends in the desired absolute orientation when recomputing it from the updated relative poses with Eq. 1.

When using regular fusion we do not detect time windows of low acceleration. Instead, we repeatedly fuse after each 10th pose. The method of fusion is exactly the same as in step 3, except for the duration of the time-window which is now fixed to 200 milliseconds, furthermore, α is set to 20 to account for the fact that we are using significantly less reliable AHRS readings. We also show the results when using this regular fusion approach together with the time-window selection mechanism of step 2. For this experiment α is again set to 0.65 because we are using accurate AHRS readings only. For these two approaches no trajectory bending is performed after fusion. When using the iterative bending approach of Sec. II-A, we follow exactly the same steps 1 to 4 but replace our closed-form trajectory bending approach with the iterative approach using 50 sub-maps as in [16]. This assures that a solution of good accuracy is obtained within a reasonable time budget.

A. Results and discussion

The results of the four tested approaches are visualized in Fig. 3. The methods which include bending are also depicted in Fig. 2 on top of aerial imagery.

From Fig. 2.a it can observed that closed-form trajectory bending is able to significantly reduce drift in the VO trajectory on basis of orientation data only. To verify that this is due to trajectory bending, we can compare its result in Fig. 3.c with that of Fig. 3.b. The only difference between these two methods is using (Fig. 3.c) or not using (Fig. 3.b) trajectory bending. In the latter case, the drift is marginally reduced because orientations at selected time-windows are improved but not back-propagated by trajectory bending. When using regular fusion, see Fig. 3.a, drift is also reduced. However, only up to the point that the AHRS readings become erroneous (i.e. significantly outside the range of their expected operational uncertainty). From Fig. 3.d and Fig. 2.b it can be observed that using the ML bending approach has no additional benefit over our closed-form approach, both approaches practically provide the same solution. However, the closed-form approach has a computation time of only 0.3



(a)

(b)

Fig. 2. Results of closed-form trajectory bending (a) and a close-up at the final pose with the results of iterative bending (b). The VO trajectory is depicted in red, the D-GPS based ground truth in green, the results after closed form bending in blue and that of iterative bending in orange. The selected time-windows of low acceleration are depicted by the blue dots on the trajectory.

seconds for the total of 2000 poses (in Matlab) whereas the iterative approach takes around 3 seconds. Our closed-form approach also is significantly easier to implement on small embedded processors than the SQP based iterative approach.

The positional error (including height) in the final pose reduces from 550 m to 17 m when applying our closed-form method. This is 0.35 percent of the 5 km of total distance traveled. This is a remarkable performance as no absolute positional data is used. The benefit of our approach is that we can be highly selective on the quality of the AHRS readings and, at the same time, are still able to improve the entire trajectory to high accuracy. This makes our approach more tolerant against erroneous readings of affordable AHRS systems.

IV. Conclusions

A novel trajectory bending algorithm was proposed to incorporate absolute orientation data into trajectories obtained by visual odometry. This allows for closing-the-loop on orientation data only. Contrary to an iterative optimizer, the proposed algorithm operates in closed-form, has linear computational complexity in the number of poses and can be computed out-of-core. It is therefore significantly more efficient than this iterative counterpart while its accuracy is shown to be comparable under realistic conditions. The results obtained on a 5 km long and challenging urban trajectory extend the current state-of-art in pose estimation from on-board sensors.

References

[1] A. Angeli, D. Filliat, S. Doncieux, and J. Meyertrech. A Fast and Incremental Method for Loop-closure Detection Using Bags of Visual Words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 2008.

[2] T. Bailey and H. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *Robotics and Automation Magazine*, June 2006.

[3] T. Bailey and H. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II State of the Art. *Robotics and Automation Magazine*, 13(3):108–117, Sept. 2006.

[4] A. I. Comport, E. Malis, and P. Rives. Accurate Quadrifocal Tracking for Robust 3D Visual Odometry. In *IEEE International Conference on Robotics and Automation*, pages 40–45, Apr. 2007.

[5] K. Cornelis, F. Verbiest, and L. J. V. Gool. Drift Detection and Removal for Sequential Structure from Motion Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:10, 2004.

[6] M. P. do Carmo. *Riemannian Geometry*. Mathematics: Theory and Applications. Birkhäuser, 1st edition, Jan. 1992.

[7] G. Dubbelman, L. Dorst, K. Schutte, and F. C. A. Groen. TNO/UvA The Hague Binocular Data Sets, www.gijsdubbelman.com, 2011.

[8] G. Dubbelman, I. Esteban, and K. Schutte. Efficient Trajectory Bending with Applications to Loop Closure. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–7, Taipei, Taiwan, 2010.

[9] G. Dubbelman and F. Groen. Bias Reduction for Stereo Based Motion Estimation with Applications to Large Scale Visual Odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, MIAMI, USA, June 2009.

[10] 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*, pages 3914–3920, Nice, France, 2008.

[11] P. Elinas, R. Sim, and J. J. Little. σSLAM: Stereo Vision SLAM Using the Rao-Blackwellised Particle Filter and a Novel Mixture Proposal Distribution. In *IEEE International Conference on Robotics and Automation*, pages 1564–1570, 2006.

[12] 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.

[13] G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *IEEE International Conference on Robotics and Automation*, 2011.

[14] K. Konolige and M. Agrawal. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, Oct. 2008.

[15] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey. Outdoor mapping and navigation using stereo vision. In *Proceedings of the International Symposium on Experimental Robotics*, 2006.

[16] P. Newman, D. Cole, and K. Ho. Outdoor SLAM using Visual Appearance and Laser Ranging. In *IEEE International Conference on Robotics and Automation*, 2006.

[17] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, Jan. 2006.

[18] F. C. Park. Distance Metrics on the Rigid-Body Motions with Applications to Mechanism Design. *Transactions of the ASME*, 117:48–54, Mar. 1995.

[19] X. Pennec. Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127–154, July 2006.

[20] J. M. Selig. *Geometrical Methods in Robotics*. Springer, 1996.

[21] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzibbon. Bundle adjustment - a modern synthesis. *In Vision Algorithms: Theory and Practice, LNCS*, pages:298–375, 1999.

[22] R. Wagner, O. Birbach, and U. Frese. Rapid Development of Manifold-Based Graph Optimization Systems for Multi-Sensor Calibration and SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[23] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. S. Sawhney. Ten-fold Improvement in Visual Odometry Using Landmark Matching. In *IEEE International Conference on Computer Vision*, pages 1–8, Oct. 2007.

APPENDIX A

Proof: The proof of our theorem is based on *induction*. To allow for compact formulas we switch to the following notation style:

$$\begin{array}{rclcrcl} \mathrm{X}_{j:k} & = & \prod_{i=j}^{k} \mathrm{X}_i & , & \mathrm{X}_{j:k}^{-1} & = \left(\prod_{i=j}^{k} \mathrm{X}_i\right)^{-1} & = \prod_{i=k}^{j} \mathrm{X}_i^{-1} \quad , \\ (\mathrm{XY})_{j:k} & = & \prod_{i=j}^{k} \mathrm{X}_i\mathrm{Y}_i \quad \text{and} & & (\mathrm{XY})_{j:k}^{-1} & = \left(\prod_{i=j}^{k} \mathrm{X}_i\mathrm{Y}_i\right)^{-1} & = \prod_{i=k}^{j} \mathrm{Y}_i^{-1}\mathrm{X}_i^{-1} \end{array} \tag{22}$$

(note that these products are constructed from left to right as in Eq. 1). By using this notation we can express constraint Eq. 9 by the *proposition*

$$P(k): \quad \mathrm{R}_{1:n-k}\,(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-k} = \mathrm{D}_n, \tag{23}$$

where $\mathfrak{T}$ is defined as in Theorem 5.1 with

$$\mathfrak{T}(\hat{\mathrm{U}}_j) = \mathrm{R}_{1:j}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_j\mathrm{D}_n^{-1}\mathrm{R}_{1:j} = \mathrm{U}_j. \tag{24}$$

For our theorem 5.1 to be correct, the proposition $P(k)$ has to be valid for all $1 \leq k \leq n$. For $k = n$ the proposition reads $P(n) : (\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{1:n} = \mathrm{D}_n$, i.e. when all correction terms are distributed over the trajectory it ends in the desired pose. This is the goal of our bending algorithm.

The correctness of the *basis* $P(k = 1)$ is provided by the equalities

$$\begin{array}{rcrcl} P(1) & : & \mathrm{R}_{1:n-1}\mathrm{R}_n\mathfrak{T}(\hat{\mathrm{U}}_n)\hat{\mathrm{U}}_{1:n-1} & = & \mathrm{R}_{1:n}\mathfrak{T}(\hat{\mathrm{U}}_n)\hat{\mathrm{U}}_{1:n-1} \\ & & & = & \mathrm{R}_{1:n}\mathrm{R}_{1:n}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_n\mathrm{D}_n^{-1}\mathrm{R}_{1:n}\hat{\mathrm{U}}_{1:n-1} \\ & & & = & \mathrm{D}_n\hat{\mathrm{U}}_n\mathrm{D}_n^{-1}\mathrm{R}_{1:n}\hat{\mathrm{U}}_{1:n-1} \\ & & & = & \mathrm{D}_n\hat{\mathrm{U}}_n(\mathrm{R}_{1:n}\hat{\mathrm{U}}_{1:n})^{-1}\mathrm{R}_{1:n}\hat{\mathrm{U}}_{1:n-1} \\ & & & = & \mathrm{D}_n\hat{\mathrm{U}}_n\hat{\mathrm{U}}_{1:n}^{-1}\mathrm{R}_{1:n}^{-1}\mathrm{R}_{1:n}\hat{\mathrm{U}}_{1:n-1} \\ & & & = & \mathrm{D}_n\hat{\mathrm{U}}_n\hat{\mathrm{U}}_{1:n}^{-1}\hat{\mathrm{U}}_{1:n-1} \\ & & & = & \mathrm{D}_n\hat{\mathrm{U}}_n\hat{\mathrm{U}}_n^{-1} \\ & & & = & \mathrm{D}_n. \end{array} \tag{25}$$

The *inductive hypothesis* states that

$$P(k) : \mathrm{R}_{1:n-k}\,(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-k} = \mathrm{D}_n \tag{26}$$

is true. We now proof the correctness of the *inductive step*, i.e. the correctness of $P(k+1)$ when assuming that the inductive hypothesis $P(k)$ holds. Let us start with rewriting $P(k)$ by singling out $\hat{\mathrm{U}}_{n-k}$, i.e.

$$P(k) : \mathrm{R}_{1:n-k}\,(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-k} = \mathrm{R}_{1:n-k}\,(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)}\hat{\mathrm{U}}_{n-k} = \mathrm{D}_n. \tag{27}$$

Observe from the last equality in Eq. 27 that the term

$$(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)} = \mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}^{-1} \tag{28}$$

when assuming $P(k)$ is true. We can therefore rewrite the inductive hypothesis $P(k)$ to

$$P(k): \mathrm{R}_{1:n-k}(\mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}^{-1})\hat{\mathrm{U}}_{n-k}(\mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}^{-1})^{-1}(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)} = \mathrm{D}_n. \tag{29}$$

Now observe that the term

$$\begin{array}{rcl} (\mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}^{-1})\hat{\mathrm{U}}_{n-k}(\mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}^{-1})^{-1} & = & \mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}^{-1}\hat{\mathrm{U}}_{n-k}\hat{\mathrm{U}}_{n-k}\mathrm{D}_n^{-1}\mathrm{R}_{1:n-k} \\ & = & \mathrm{R}_{1:n-k}^{-1}\mathrm{D}_n\hat{\mathrm{U}}_{n-k}\mathrm{D}_n^{-1}\mathrm{R}_{1:n-k} \\ & = & \mathfrak{T}(\hat{\mathrm{U}}_{n-k}), \end{array} \tag{30}$$

it is our mapping function $\mathfrak{T}$ applied to $\hat{\mathrm{U}}_{n-k}$. This allows reformulating proposition $P(k)$ as

$$P(k): \quad \mathrm{R}_{1:n-k}\mathfrak{T}(\hat{\mathrm{U}}_{n-k})(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)} = \mathrm{D}_n \tag{31}$$

By rewriting its equality we find that

$$\begin{array}{ll} \mathrm{R}_{1:n-k}\mathfrak{T}(\hat{\mathrm{U}}_{n-k})(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)} & = \mathrm{D}_n \\ \mathrm{R}_{1:n-(k+1)}\mathrm{R}_{n-k}\mathfrak{T}(\hat{\mathrm{U}}_{n-k})(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-k+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)} & = \mathrm{D}_n \\ \mathrm{R}_{1:n-(k+1)}(\mathrm{R}\mathfrak{T}(\hat{\mathrm{U}}))_{n-(k+1)+1:n}\,\hat{\mathrm{U}}_{1:n-(k+1)} & = \mathrm{D}_n \\ P(k+1) & = \mathrm{D}_n. \end{array} \tag{32}$$

This validates the correctness of the inductive step. The proof of our theorem 5.1 then follows from induction.

APPENDIX B

In this appendix we provide the intrinsic statistical functions for rotation matrices. For more information on intrinsic statistics in general we recommend [19].

The Riemannian exponential map wrt. the identity I for a tangent vector $\mathfrak{r}$ is

$$\mathrm{R} = \mathrm{Exp}(\mathfrak{r}) = \mathrm{I} + \sin(\|\mathfrak{r}\|)[\frac{\mathfrak{r}}{\|\mathfrak{r}\|}]_\times + (1-\cos(\|\mathfrak{r}\|))[\frac{\mathfrak{r}}{\|\mathfrak{r}\|}]_\times^2, \tag{33}$$

where

$$\left[(s_x, s_y, s_z)^\top\right]_\times = \begin{bmatrix} 0 & -s_z & s_y \\ s_z & 0 & -s_x \\ -s_y & s_x & 0 \end{bmatrix}. \tag{34}$$

It coincides with the Rodrigues rotation formula in that it produces a rotation matrix R which encodes a rotation around an axis $\frac{\mathfrak{r}}{\|\mathfrak{r}\|}$ with angle $\|\mathfrak{r}\|$.

To express the Riemannian logarithmic map wrt. the identity I we need the notation

$$[\mathrm{S}]_\uparrow = \begin{bmatrix} 0 & -s_z & s_y \\ s_z & 0 & -s_x \\ -s_y & s_x & 0 \end{bmatrix}_\uparrow = (s_x, s_y, s_z)^\top \tag{35}$$

to map a skew symmetric matrix S into a vector. With this notation the logarithmic map is

$$\mathfrak{r} = \mathrm{Log}(\mathrm{R}) = \begin{cases} \left[\theta\frac{(\mathrm{R}-\mathrm{R}^\top)}{2\sin(\theta)}\right]_\uparrow & , \quad \theta \neq 0 \\ (0,0,0)^\top & , \quad \theta = 0 \end{cases} \tag{36}$$

with

$$\theta = \arccos\left(\frac{\mathrm{trace}(\mathrm{R})-1}{2}\right) \tag{37}$$

[20]. For a rotation matrix R expressing a rotation around normalized axis $\mathbf{r}$ with angle α it effectively produces the tangent vector $\mathfrak{r} = \alpha\mathbf{r}$.

The Riemannian mappings for general rotation matrices are provided by

$$\mathrm{R}_2 = \mathrm{Exp}_{\mathrm{R}_1}(\mathfrak{r}_2) = \mathrm{R}_1(\mathrm{Exp}_{\mathbf{e}}(\mathfrak{r}_2)) \tag{38}$$

$$\mathfrak{r}_2 = \mathrm{Log}_{\mathrm{R}_1}(\mathrm{R}_2) = \mathrm{Log}_{\mathbf{e}}(\mathrm{R}_1^{-1}(\mathrm{R}_2)) \tag{39}$$

and the intrinsic distance between rotations can then be expressed with

$$\left\|\mathrm{Log}_{\mathrm{R}_1}(\mathrm{R}_2)\right\|. \tag{40}$$