# Motion-without-Structure: Real-time Multipose Optimization for Accurate Visual Odometry

Henning Lategahn, Andreas Geiger, Bernd Kitt, Christoph Stiller
Institute of Measurement and Control
Karlsruhe Institute of Technology
Karlsruhe, Germany
{henning.lategahn, geiger, bernd.kitt, stiller}@kit.edu

***Abstract*— State of the art visual odometry systems use bundle adjustment (BA) like methods to jointly optimize motion and scene structure. Fusing measurements from multiple time steps and optimizing an error criterion in a batch fashion seems to deliver the most accurate results. However, often the scene structure is of no interest and is a mere auxiliary quantity although it contributes heavily to the complexity of the problem. Herein we propose to use a recently developed incremental motion estimator which delivers relative pose displacements between each two frames within a sliding window inducing a pose graph. Moreover, we introduce a method to learn the uncertainty associated with each of the pose displacements. The pose graph is adjusted by non-linear least squares optimization while incorporating a motion model. Thereby we fuse measurements from multiple time steps much in the same sense as BA does. However, we obviate the need to estimate the scene structure yielding a very efficient estimator: Solving the non-linear least squares problem by a Gauss-Newton method takes approximately 1ms. We show the effectiveness of our method on simulated and real world data and demonstrate substantial improvements over incremental methods.**

## I. Introduction

Many environmental perception methods for robotics require a precise estimate of the vehicle's motion. Among these, are mapping algorithms [2], detection and tracking of moving objects [3], scene understanding [7] and many more.

Common choices for ego motion estimation include inertial sensors such as accelerometers and gyroscopes, Global Navigation Satellite System (GNSS) or wheel encoders. Inertial sensors providing highly accurate motion information may become prohibitively expensive for certain applications. GNSS may suffer from shadowing in street canyon like environments and wheel encoders may operate falsely on slippery ground.

Estimating ego motion from stereo cameras is an appealing alternative, especially since camera systems are already widely spread in robots and cars. They do not suffer from shadowing, are mostly independent of the ground structure and are low priced. Moreover, they make up an excellent complement to the aforementioned systems. Furthermore, many vision algorithms benefit from motion estimates derived from image streams since it supersedes the need for time synchronization between motion sensors and cameras which may become an art by itself.

Nowadays, state-of-the-art visual odometry systems use bundle adjustment (BA) [17], [1], [14], [11] or computationally efficient approximations thereof. All measurements of a sliding window are considered jointly and the ego motion and scene structure is estimated in a batch fashion. The key to accuracy lies in the joint optimization of measurements over multiple steps in time. Despite progress in exploiting sparsity patterns of the estimation problem and increasing computational power which led to first real time implementations it seems wasteful to estimate the entire scene structure for a mere motion estimate. BA like methods perform in real time only on extensive computing hardware whereas the proposed method potentially runs on low power computers due to its low complexity.

Herein, we propose a method that estimates the motion of a stereo camera by non-linear least squares optimization without estimating the scene structure. Measurements from multiple time steps are jointly optimized and ego motion is inferred. Our method does not needlessly estimate the 3D positions of any image features. We estimate pose displacements between each two frames in a sliding window by the method proposed in [6]. Figure 1 illustrates the pose displacement estimates graphically. The core of our method consists of interpreting pose displacement estimates as measurements and adjusting the thus induced pose graph by non-linear least squares optimization. Thereby we obviate the need to estimate scene structure which otherwise would heavily contribute to the complexity of the problem. Furthermore, we show how constraints imposed by the vehicle dynamics can be integrated into our estimation algorithm by considering a motion model.

Since pose displacement estimates are highly heteroscedastic we introduce a regression method that efficiently provides pose displacement estimation uncertainties. The regression is trained by a Monte Carlo (MC) simulation. The introduction of this regressor is another main contribution of our work.

Adjusting a pose graph without estimating scene structure is very efficient. Our implementation uses a Gauss-Newton optimization and takes approximately 1ms to converge. In experiments we show an increase in accuracy over a purely incremental baseline [6].

The remainder of the paper is structured as follows. In Section II we review related work. The algorithm is detailed in Section III. Thereafter experimental results of simulated and real world data are given in Section IV before concluding

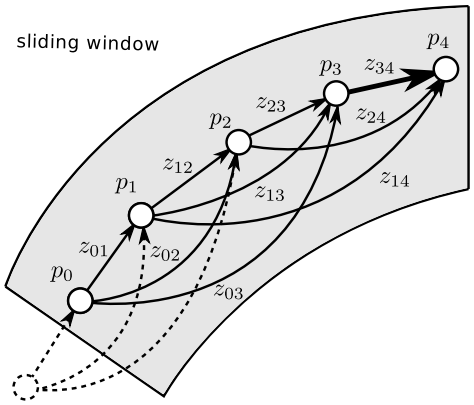


Fig. 1: Relative pose displacements $z_{ij}$ are computed from stereo vision between poses $p_i$ and $p_j$. Poses within a sliding window of length $M$ are thereafter adjusted by non-linear least squares optimization. $M = 4$ in this example graph. The bold arrow is the ego motion $h(\hat{p}_3, \hat{p}_4)$ which is the output of our algorithm after adjusting the graph.

in Section V.

## II. Related Work

Many approaches to estimating the vehicle's ego motion from stereo cameras [6], [8], [10] are incremental in nature. The motion is computed for each consecutive image pair and accumulated over time. Usually an error term describing the fit of motion and point matches is minimized. These methods are characterized by low computational complexity. However, their accuracy is somewhat limited and ego positions drift rather severely over time.

Better results are usually obtained by Simultaneous Localization and Mapping (SLAM) like approaches [15], [4], [13], [5]. SLAM is the problem of computing a map of previously unknown terrain while simultaneously localizing the robot within the map. Thereto, a set of landmarks (associated with salient features in the image), the current ego velocity and position are stacked into one single vector which is sequentially estimated from point correspondences by e.g. Extended Kalman Filters (EKFs). Fusing measurements from multiple time steps has increased the accuracy considerably. However, the use of EKFs has shown some disadvantages: The state covariance may diverge due to linearization errors [9] and landmark associations are irreversible. State updates of landmark miss associations are inevitably "baked" into the state vector causing catastrophic results. Therefore the computer vision front end which provides point matches is of utmost importance in practical implementations. In contrast, the system presented here merely requires feature correspondences between two frames and does not depend on feature tracks.

Today the best performing systems are bundle adjustment (BA) like methods [17], [1], [11]. The scene structure and ego motion is estimated such that the reprojection error of all point features onto all camera images is simultaneously minimized. Jointly optimizing scene structure and ego motion improves over the down sides of EKF like methods as mentioned above. Linearization errors are mitigated. The key seems to be the joint consideration of measurements of a given time window. However, the scene structure is often only a means to an end and discarded immediately thereafter. We propose to estimate motion by joint optimization of measurements over multiple time steps without computing scene structure, hence yielding a lean minimization problem. Using pose displacements induced by registering laser scans to build large maps has been recently proposed by Konolige and co-workers in [12]. The work aims at building large 2D maps of the environment and no focus is laid on motion estimation. In our work we are using stereo cameras exhibiting a completely different sensor characteristic. The displacements between poses are estimated by non-linear least squares optimization (see [6]). This however bears the problem of accurately and efficiently providing pose displacement uncertainties. We solve this issue by learning a covariance matrix data base from training imagery which we query during online computation. Unlike [12], we integrate motion constraints imposed by vehicle dynamics into our estimation process. Finally we compare different parameterizations of our method to an incremental baseline in simulations and real world situations.

## III. Ego Motion Estimation

Our method computes the ego displacements between each pair of poses within a sliding window of length $M$. Thus a connected pose graph as depicted in Figure 1 is induced. An edge of the graph labeled $z_{ij}$ is the ego displacement estimated between poses $p_i$ and $p_j$. After pairwise pose displacements are estimated within a window of length $M$ the pose graph is adjusted yielding the current ego motion. First an overview of our method is given before describing each part more thoroughly.

### A. Overview

We use the constant turn rate and acceleration (CTRA) model which was shown to be the best performing model among the curve linear models investigated in [16]. Therefore one such pose is defined by its position, orientation, linear and angular velocity and linear acceleration, thus $p_i = (x_i, y_i, \theta_i, v_i, a_i, \omega_i)^T$. Hence our poses are two dimensional and contain no height information. The state transition equation of the CTRA model is denoted by the function $f(\cdot)$ and is given below. During motion estimation we exploit that pose $p_{i+1}$ should not deviate much from its predicted position $f(p_i)$.

The displacements are computed by the method proposed in [6] which is briefly reviewed in Section III-B. The ego displacement between pose $p_i$ and $p_j$ is denoted by $z_{ij} = (\Delta x_{ij}, \Delta y_{ij}, \Delta\theta_{ij})^T$ and is relative to pose $p_i$. For the ego displacement between poses $p_i$ and $p_j$ without error we have

$$\begin{pmatrix} \Delta x_{ij} \\ \Delta y_{ij} \\ \Delta\theta_{ij} \end{pmatrix} = \begin{pmatrix} R^T(\theta_i)\left[\begin{pmatrix} x_j \\ y_j \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix}\right] \\ \theta_j - \theta_i \end{pmatrix} \quad (1)$$

$$=: \quad h(p_i, p_j) \quad (2)$$

where $R(\theta_i)$ is a $2 \times 2$ rotation matrix of angle $\theta_i$. For two given poses $p_i$ and $p_j$ the function $h(p_i, p_j)$ computes the relative displacement according to the right hand side of (1). The rotation matrix $R^T(\theta_i)$ is required to transform global poses into displacements relative to $p_i$ as it is provided by the displacement estimator. We use the same nomenclature as [12]. After all pairwise ego displacements within a sliding window are estimated the ego motion estimation is launched. The goal is to find poses such that two poses $p_i$ and $p_j$ have a relative position $h(p_i, p_j)$ that does not deviate much from its estimated displacement $z_{ij}$.

All poses of a sliding window of length $M$ can now be estimated by considering the following error term consisting of error terms imposed by the motion model and error terms imposed by the estimated ego displacements

$$e'(p_0, \ldots, p_M) = \begin{pmatrix} p_1 - f(p_0) \\ p_2 - f(p_1) \\ \cdots \\ p_M - f(p_{M-1}) \\ z_{01} - h(p_0, p_1) \\ z_{02} - h(p_0, p_2) \\ \cdots \\ z_{0M} - h(p_0, p_M) \\ z_{12} - h(p_1, p_2) \\ \cdots \end{pmatrix} \quad (3)$$

with $x_0 = y_0 = \theta_0 = 0$ of pose $p_0$ fixed to the origin. Terms of the form $p_{i+1} - f(p_i)$ penalize motion that is incompatible to the motion model whereas terms of the form $z_{ij} - h(p_i, p_j)$ penalize deviations from the estimated displacement.

Assuming $e' \sim \mathcal{N}(0, C)$ leads to the following minimization problem

$$\hat{p}_0, \ldots, \hat{p}_M \quad (4)$$

$$= \arg\min_{p_0, \ldots, p_M} \left\{ \underbrace{e'(p_0, \ldots, p_M)^T C^{-1} e'(p_0, \ldots, p_M)}_{=E} \right\} \quad (5)$$

such that $x_0 = y_0 = \theta_0 = 0$. Any non-linear optimization method can be used to minimize the squared Mahalanobis distance $E$. Here we employ Gauss-Newton optimization. Determining the covariance matrix $C$ is crucial and its derivation is postponed until later. Our algorithm outputs the ego motion from the immediately preceding pose to the current pose which is $h(\hat{p}_{M-1}, \hat{p}_M)$, see also Figure 1.

### B. Point Matching and Ego Displacement

In order to obtain 3D poses between temporally adjacent stereo pairs, we follow the method proposed in [6], as it is easy to implement and efficient at the same time. First, interest points are extracted in two consecutive stereo pairs using blob and edge kernels in combination with non-maximal-suppression. Next, feature descriptors are built from local gradient patterns and matched efficiently between the four images using SIMD instructions. Since the stereo camera setup is assumed to be calibrated, we are able to project all features from the previous frame back into 3D via triangulation. The 6D rigid motion between the current and the previous frame is then recovered by the minimizer of the reprojection errors of the features in the current frame. To account for outliers the procedure is wrapped into a RANSAC sampling scheme with 50 iterations. Both, feature matching and pose estimation, take less than 50 ms on a single CPU core. For details we refer the reader to [6]. The 6D motion estimate is projected onto the road plane to obtain the pose displacement $(\Delta x_{ij}, \Delta y_{ij}, \Delta\theta_{ij})$.

### C. Motion Model

Since we are interested in recovering camera poses of a car like robot, we incorporate prior knowledge in form of a motion model to stabilize the estimation results. A good overview of many curve linear models can be found in [16] and references therein. Errors resulting from highly dynamic driving maneuvers such as drifting can often be neglected as the typical operation mode is not in this dynamic range.

The CTRA model [16] assumes constant turn rates and accelerations. Violations of this model (i.e. change in acceleration and turn rate) are modeled by appropriate noise terms. The model predicts poses as

$$f(x_i, y_i, \theta_i, v_i, a_i, \omega_i) = \begin{pmatrix} x_i + \Delta x_i(\Delta t) \\ y_i + \Delta y_i(\Delta t) \\ \theta_i + \omega_i \Delta t \\ v_i + a_i \Delta t \\ a_i \\ \omega_i \end{pmatrix} \quad (6)$$

with

$$\Delta x_i(\Delta t) = \frac{1}{\omega_i^2}((v_i\omega_i + a_i\omega_i\Delta t)\sin(\theta_i + \omega_i\Delta t) + a_i\cos(\theta_i + \omega_i\Delta t) - v_i\omega_i\sin\theta_i - a_i\cos\theta_i) \quad (7)$$

and

$$\Delta y_i(\Delta t) = \frac{1}{\omega_i^2}((-v_i\omega_i - a_i\omega_i\Delta t)\cos(\theta_i + \omega_i\Delta t) + a_i\sin(\theta_i + \omega_i\Delta t) + v_i\omega_i\cos\theta_i - a_i\sin\theta_i) \quad (8)$$

where $\Delta t$ denotes the time lag. Hence the current vehicle state can be predicted from the preceding dynamic state of the vehicle. Note that $\Delta x_i(\Delta t)$ of (7) is substantially different from $\Delta x_{ij}$ of (1).

We assume the state of the vehicle to evolve according to

$$p_{i+1} = f(p_i) + \epsilon_i \quad (9)$$

with $\epsilon_i \sim \mathcal{N}(0, Q_i)$. Here, $\epsilon_i$ captures the change in turn rate and acceleration.

### D. Adjusting the Pose Graph

The pose graph as induced by the constraints of the motion model and the pose displacements $z_{ij}$ is adjusted to minimize the squared error $E$ in (5). To this end, the auxiliary function

$$\underbrace{e(v_0, a_0, \omega_0, p_1, \ldots, p_M)}_{=\mathbf{x}} = Se'(\mathbf{x}) \quad (10)$$

with $S^T S = C^{-1}$ being the Cholesky decomposition of the precision matrix $C^{-1}$ of $e'$ is introduced. The final estimate is the minimizing argument of

$$e(\mathbf{x})^T e(\mathbf{x}) \quad = \quad e'(\mathbf{x})^T C^{-1} e'(\mathbf{x}). \tag{11}$$

To minimize (11) we use the Gauss-Newton method. An initial guess $\mathbf{x}_{k=0}$ is used to linearize $e$ around $\mathbf{x}_k$ yielding

$$e(\mathbf{x}) \quad \approx \quad e(\mathbf{x}_k) + SE'(\mathbf{x}_k)\Delta\mathbf{x}_k \tag{12}$$

where $E'(\mathbf{x}_k)$ is the Jacobian of $e'$ evaluated at $\mathbf{x}_k$. The increment $\Delta\mathbf{x}_k$ minimizing the squared L2-norm of the linearized version of $e$ (12) is found by solving

$$E'(\mathbf{x}_k)^T S^T e(\mathbf{x}_k) \quad = \quad -E'(\mathbf{x}_k)^T C^{-1} E'(\mathbf{x}_k)\Delta\mathbf{x}_k \tag{13}$$

for $\Delta\mathbf{x}_k$. A new iteration starts by setting $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$ and we stop iterating when $||\Delta\mathbf{x}_k|| < \tau$ or when a maximum number of iterations is reached. Note that we do not require robust sampling schemes at this stage, since all correspondence outliers have already been eliminated during pose displacement estimation (see Section III-B).

The initial guess $\mathbf{x}_0$ is found by accumulating the pose displacements $z_{i\ i+1}$ for $i = 0, \ldots, M-1$ starting from pose $(x_0, y_0, \theta_0) = (0, 0, 0)$ which is the solution of the incremental method. Accelerations and linear velocities are set to zero whereas the angular velocities are set to $\omega_i = 0.0001$. This initialization speeds up the solver by one iteration compared to a zero initial guess. In practice we note that our optimization always returns the global optimizer $\mathbf{x}$ using this initialization.

In the following we describe the assembly of the covariance matrix $C$ of $e'$. $C$ is block diagonal with the following entries

$$C \quad = \quad \begin{pmatrix} Q & 0 \\ 0 & P \end{pmatrix} \tag{14}$$

$$Q \quad = \quad \begin{pmatrix} Q_0 & 0 & \cdots & \\ 0 & Q_1 & \cdots & \\ \vdots & \vdots & \ddots & \\ & & & Q_{M-1} \end{pmatrix} \tag{15}$$

$$P \quad = \quad \begin{pmatrix} P_{01} & 0 & \cdots & \\ 0 & P_{02} & \cdots & \\ \vdots & \vdots & \ddots & \\ & & & P_{M-1\ M} \end{pmatrix} \tag{16}$$

For pose transition $p_{i+1} = f(p_i)$ the covariance $Q_i$ captures the model violation (cf. (9)). The pose displacement between poses $p_i$ and $p_j$ is given with accuracy $P_{ij}$ and is derived from the point matches between the two images (details given below). The diagonal elements of $C$ describe the certainty of the elements of (3). The block diagonal structure of $C$ can be exploited when computing the Cholesky decomposition $S$ of $C^{-1}$.

### *E. Noise Covariances $Q_i$ and $P_{ij}$*

The pose transition covariance $Q_i$ is computed by

$$Q_i \quad = \quad F_i D F_i^T \tag{17}$$

where $F = \frac{\partial}{\partial x} f$ denotes the Jacobian of $f$ evaluated at $p_i$ and $D = \mathrm{diag}(\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_v^2, \sigma_a^2, \sigma_\omega^2)$ is a diagonal matrix. $\sigma_x^2, \sigma_y^2, \sigma_\theta^2$ and $\sigma_v^2$ are kept very low (however non-zero to assure non-singularity of $Q_i$) whereas $\sigma_a^2$ and $\sigma_\omega^2$ are vehicle specific, in our case $\sigma_a = 1m/s^2\Delta t$ and $\sigma_\omega = 30°\pi/180°\Delta t$ with $\Delta t$ being the time lag.

The certainty of the ego displacement $z_{ij}$ can vary considerably depending on the characteristic of the point matches between the two images. Therefore it is crucial to provide a faithful approximation of the measurements covariance $P_{ij}$. Otherwise uncertain ego displacements contribute as much as certain displacements, hence polluting the estimate.

Since the computation of the ego displacements from point matches involves iterations, error propagation from point match uncertainties to ego displacement uncertainties can not be employed. We address this problem by running MC simulations to build a covariance data base. We store these covariance matrices together with feature vectors extracted from the image. Regression based on these features is then used to retrieve an appropriate matrix from the data base for every pair of frames during online computation.

We use one sequence for training and one for testing. From the training sequence we compute the pose displacement covariance for a few thousand representative pairs of poses. For one such displacement between poses $i$ and $j$ we compute the point matches between the images of poses $i$ and $j$ denoted by $\mathcal{M}$. Thereafter many thousand copies $\mathcal{M}_0, \ldots, \mathcal{M}_K$ of $\mathcal{M}$ are produced and all feature matches are disturbed by isotropic Gaussian noise with $\sigma = 0.5px$. Finally the ego displacement $z_k$ for each such copy $\mathcal{M}_k$ is computed. The covariance matrix stored for this particular pair of poses is the empirical covariance of $z_0, \ldots, z_K$.

The feature vector is composed of the bins of a histogram computed from all disparity values of the source image $i$. For larger displacements nearby points are not matched anymore which leads to filling up the lower bins of the histogram. Smaller displacements are characterized by larger disparity values in the source image. The overall number of points also contributes to the certainty of the displacement estimate. For the experiment a histogram size of 20 was used.

As regression we refrain from using computationally expensive method such as Gaussian Processes and use nearest neighbor regression for computational efficiency. For a query feature vector, we take the covariance associated with the most similar feature vector in the training set. As similarity metric we employ the L2 norm. Thereby the time to query a feature vector is negligible for a moderate sized (approx. 2k samples) training set.

## IV. Experiments

We have conducted two types of experiments. First a simulation was run to assess the proposed method under ideal conditions. Second, our method was tested on real data.

### *A. Simulation Experiments*

From (9) it is easy to generate a simulated trajectory from any given initial state $p_0$. Figure 2 shows one such generated

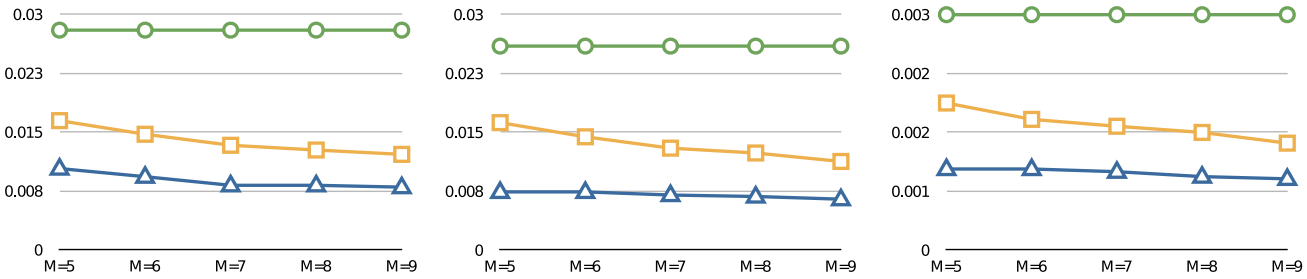


Fig. 3: The standard deviations of the motion error of a simulated trajectory are given. From left to right: $\sigma_{\Delta x}, \sigma_{\Delta y}, \sigma_{\Delta \theta}$. The window length $M$ is varied. The green curve (circle) shows the error of the incremental method. The yellow curve (squaress) show the error of our method without a motion model whereas the blue curve (triangles) depicts the error of our method with a motion model. The numbers are given in meters, meters and radians (left to right).

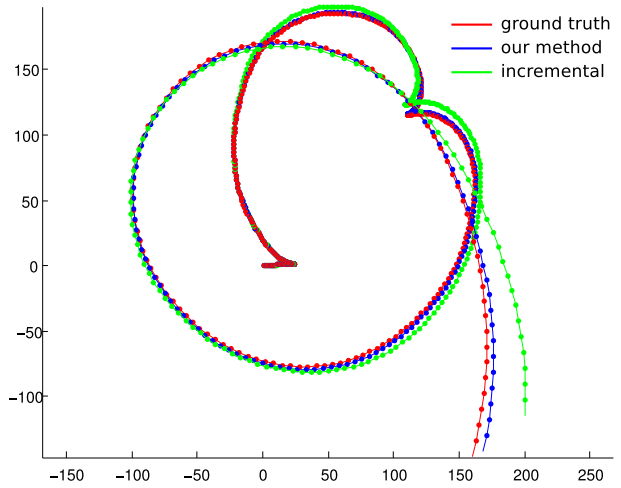


Fig. 2: A simulated trajectory of approximately 1.4 km. Red denotes ground truth, blue denotes the trajectory as estimated by our method and green is the accumulation of consecutive ego displacement estimates (without pose adjustment).

trajectory. After trajectory generation, the relative poses between pose couples within a sliding window can be computed by (1). Setting $z_{ij} = h(p_i, p_j) + \delta_{ij}$ with $\delta_{ij} \sim \mathcal{N}(0, P_{ij})$ finally provides simulated pose displacement measurements. We set $P_{ij} = P$ constant for all $i, j$ in our simulation.
$P$ was computed by projecting a set of 3D points into two consecutive frames and running a MC simulation as detailed in Section III-E. Thereby $P$ is a faithful approximation of a pose displacement uncertainty.
As depicted in Figure 2 the trajectory can be recovered by our proposed method. The red path indicates ground truth, blue is the estimated trajectory by adjusting the pose graph and green is the accumulation of consecutive pose displacements. Green is the trajectory as it would have been estimated by a pure incremental motion estimator. The pose adjusted trajectory (blue) is more accurate and exhibits a lower drift than the non-adjusted counter part (green).

To validate our method more thoroughly we have run the simulation with 1000 poses and varying parameters and compared the estimated *motion* to the ground truth motion. The motion error is expressed by the empirical standard deviations $\sigma_{\Delta x}, \sigma_{\Delta y}, \sigma_{\Delta \theta}$ of the displacement error. Moreover, we have run the simulation without a motion model. That is, we have only considered the bottom part of (3) which is associated with the pose displacements only. The results for different lengths $M$ of the sliding window are given in Figure 3. The error of the incremental method is given in green (circle). Since no poses are adjusted this error is independent of the window length $M$. The error of our method using no motion model is shown in yellow (squares) whereas the error of our proposed method including a motion model is shown in blue (triangles).
The results show that increasing the number of poses which are adjusted increases overall accuracy. This holds true for the case with and without a motion model. Including the proposed motion model further improves the accuracy. In all cases, we clearly outperform the incremental baseline.

### B. Real World Data

Our method was tested on a sequence of a circular trajectory of approximately 500m. The end of the trajectory matches the beginning of the trajectory. Our test vehicle is equipped with a stereo camera rig with a base length of 57.5 cm. The image size after rectification is $1350 \times 370$ pixels. The opening angle of the cameras are 90 degrees. Image points are detected and matched as detailed in [6]. We have matched at most 100 points between any two image pairs. The ego displacement is computed as described in Section III-B.
Accumulating the ego displacements of all consecutive frames leads to the trajectory shown in blue in Figure 4. This accumulation exhibits a heavy drift. It seems that the rotation angles are insufficiently estimated by the method. The red path shows the reconstructed trajectory obtained by using the proposed method with $M = 4$ and no motion model. Incorporating the motion model constraint leads to the result shown in green. The results are in accordance to the simulated data. Adjusting the pose graph improves the outcome considerably. Integrating a motion model further improves results.
Finally we have run the system with motion model but without regressing the covariance matrix. Instead the pose displacement covariance was set to a fixed pose displacement

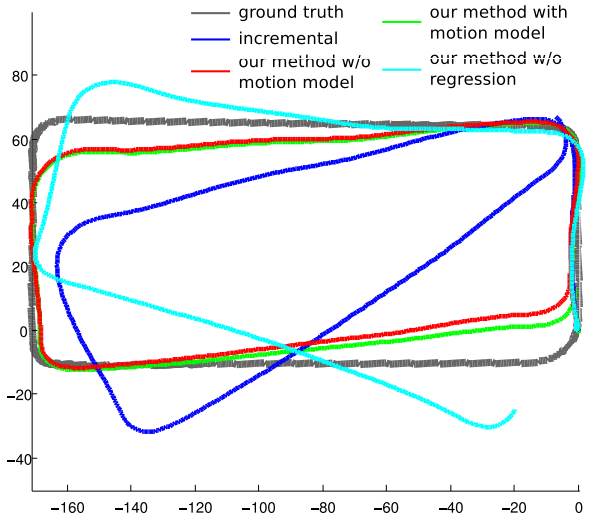


Fig. 4: Results of a real trajectory of approximately 500m. The incremental motion estimator is shown in blue. Green denotes the trajectory as computed by our method. Using no motion model yields the red trajectory. The cyan trajectory is obtained by our method when not using the regression for the pose displacement covariances. Ground truth is underlaid in gray.

covariance for all frame couples. The resulting trajectory is depicted in cyan in Figure 4. This clear deterioration over the use of the regression stems from the fact that uncertain pose displacements are overrated. The ground truth is underlaid in gray.

It shall be noted here that the method to compute ego displacements produces much better results when matching more image points (few thousands as apposed to one hundred). Results are given in [6]. However, herein we demonstrate the improvements by fusing multiple ego displacements. The image processing front end consisting of interest point detection and matching as well as the ego displacement computation is much faster with fewer points. Computing ego displacements for 100 points takes 3 ms. For each new image frame $M$ ego displacements need to be computed to set up the pose graph (see Figure 1). Adjusting the pose graph of the specified size requires only 0.5 ms even when including the motion model. Note that this is independent of the feature matching front end. For each frame the pose graph needs to be adjusted only once. All times are achieved on a single CPU core. State-of-the-art BA methods achieve frame rates ($\widetilde{\approx}$100ms) at best on heavily parallelized computing hardware. The bottle neck of our computation right now is the image processing front end. We intend to improve upon this in the future.

## V. Conclusion and Future Research

Herein we have demonstrated how pairwise ego displacements between frames within a sliding window can be used to compute the ego motion of a robot or vehicle. To this end the motion estimation was cast into a non-linear least squares problem incorporating a CTRA motion model. The covariance of the ego displacement estimate was provided by nearest neighbor regression.

Experiments with simulated and real world data from our testing vehicle show that the pose adjustment increase accuracy. Moreover, the motion model further improves upon the results. Adjusting a pose graph takes 0.5ms on a single CPU core while BA runs much slower. This makes our approach particularly interesting for mobile applications where low power consumption is crucial.

Developing a point matching front end that is tightly coupled with the motion estimator is ongoing research. The motion estimated for the current pose may be used to restrict the search space of point matches of foregoing images. Moreover, the large displacement of image features for large displacements needs to be addressed. Furthermore we intend to extend the proposed motion estimator into a full SLAM system akin to [12].

## References


[1] M. Agrawal and K. Konolige, "Rough terrain visual odometry," in *Proceedings of the International Conference on Advanced Robotics (ICAR)*, 2007.

[2] H. Badino, U. Franke, and R. Mester, "Free space computation using stochastic occupancy grids and dynamic programming," in *Workshop on Dynamical Vision (ICCV)*, 2007.

[3] A. Barth and U. Franke, "Estimating the driving state of oncoming vehicles from a moving platform using stereo vision," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 4, pp. 560–571, 2009.

[4] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1052–1067, 2007.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[6] A. Geiger, J. Ziegler, and S. C., "Stereoscan: Dense 3d mapping in real-time," in *IEEE Intelligent Vehicles Symposium*, June 2011.

[7] A. Geiger, M. Lauer, and R. Urtasun, "A generative model for 3d urban scene understanding from movable platforms," in *Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, USA, June 2011.

[8] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. Ieee, 2008, pp. 3946–3952.

[9] S. Julier and J. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4238–4243.

[10] M. Kaess, K. Ni, and F. Dellaert, "Flow separation for fast and robust stereo odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 3539–3544.

[11] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," *Robotics Research*, pp. 201–212, 2011.

[12] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Sparse pose adjustment for 2d mapping," *IROS, Taipei, Taiwan*, vol. 10, p. 2010, 2010.

[13] T. Lemaire, C. Berger, I. Jung, and S. Lacroix, "Vision-based slam: Stereo and monocular approaches," *International Journal of Computer Vision (IJCV)*, vol. 74, no. 3, p. 364, 2007.

[14] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," 2004.

[15] P. Piniés and J. Tardós, "Scalable SLAM building conditionally independent local maps," in *IEEE conference on Intelligent Robots and Systems (IROS)*, 2007.

[16] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *Information Fusion, 2008 11th International Conference on*. IEEE, 2008, pp. 1–6.

[17] G. Sibley, C. Mei, I. Reid, and P. Newman, "Vast-scale Outdoor Navigation Using Adaptive Relative Bundle Adjustment," *The International Journal of Robotics Research*, 2010.