2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

A Constricted Bundle Adjustment Parameterization for Relative Scale Estimation in Visual Odometry

Friedrich Fraundorfer[1], Davide Scaramuzza[2], and Marc Pollefeys[1]

[1]Computer Vision and Geometry Lab, ETH Zürich
[2]Autonomous Systems Lab, ETH Zürich
fraundorfer@inf.ethz.ch, davide.scaramuzza@ieee.org, marc.pollefeys@inf.ethz.ch

Abstract— **In this paper we address the problem of visual motion estimation (visual odometry) from a single vehicle mounted camera. One of the basic issues of visual odometry is relative scale estimation. We propose a method to compute the relative scales of a path by solving a bundle adjustment optimization problem. We introduce a constricted parameterization of the bundle adjustment problem, where only the distances between neighboring cameras are optimized, while the rotation angles and translation directions stay fixed. We will present visual odometry results for image data of a vehicle mounted onmidirectional camera for a track of 1000m length.**

I. Introduction

Vision based motion estimation (also called visual odometry) is a very challenging problem. Even more if it is attempted with a single camera only. Visual odometry requires the computation of camera rotation and translation between consecutive frames. It requires also to compute the relative scale between frames and finally to compute the absolute metric scale. For absolute scale one can use a stereo setup or in many cases it is sufficient to compute relative scale only. The relative motion (i.e. camera rotation, translation, and relative scale) should be computed online and with high accuracy as errors are accumulating over time. For accurate relative motion, good quality feature matches and feature tracks are necessary which are usually hard to get fully automatically. But even with robust algorithms there is an inevitable drift over time. Especially, relative scale seems to be most sensitive for even small inaccuracies. Even small deviations can accumulate very fast to large differences. In our previous work [1] we presented a method to compute rotation and translation robustly with low drift.

The main contribution of this paper is a constricted parameterization of the bundle adjustment problem [2] for relative scale estimation of planar motion. The optimization problem is formulated such that only the relative scales between camera poses get optimized. The rotation angles and translation directions are fixed. 3D points are re-computed on evaluating the cost function at every iteration, instead of being parameters of the optimization problem. Compared to full bundle adjustment for planar motion - where you would have 3 motion parameters per camera and 3 parameters per 3D point - our proposed parameterization has only 1 parameter per camera. Furthermore, our experiments show that with such a parameterization the optimization converges quicker to the final solution.

We show that our method produces locally very accurate results and shows little drift over long distances without any offline optimization step. We present visual odometry results on challenging image data of a vehicle mounted omnidirectional camera over long distances.

II. Related work

In [3], Nister et al. describe a visual odometry system using a single camera. The method uses a combination of relative motion estimation with the 5-point algorithm and 3D-2D pose estimation. From feature tracks, initial 3D points are computed by triangulation from camera poses that are computed by relative motion estimation. Additional poses are computed by pose estimation from 3D-2D matches. In the absence of multi-view feature tracks, the method continues with two-view relative motion estimation. In their paper a perspective camera was used, but the method would also apply to an omnidirectional camera.

In a later paper, Engels et al. describe a very similar system that is using an additional bundle adjustment step to refine the initial camera position [4]. With a windowed bundle adjustment, the cameras were locally optimized in an online fashion. However, the method was demonstrated only on an image sequence of an object on a turntable. In their bundle adjustment all the parameters were optimized. Visual odometry on a larger scale was presented by Mouragnon et al. [5], where they also use windowed bundle adjustment.

In [6], Tardif et al. proposed a new approach for the relative motion estimation, which decouples the rotation estimation from the translation. In particular, they compute the rotation from the epipolar geometry between the last and the new image and the remaining translation from the 3D map. Experiments were done using a PointGrey Ladybug omnidirectional cameras with high resolution (up to 10Mp) and in a urban scenario.

Our method mainly differs from previous works in that we separate relative motion and scale estimation and put an additional focus on consistent scale estimation. The relative motion estimation is especially designed for planar motion, which leads to a very efficient and extremely robust algorithm. For relative scale estimation, we finally use bundle

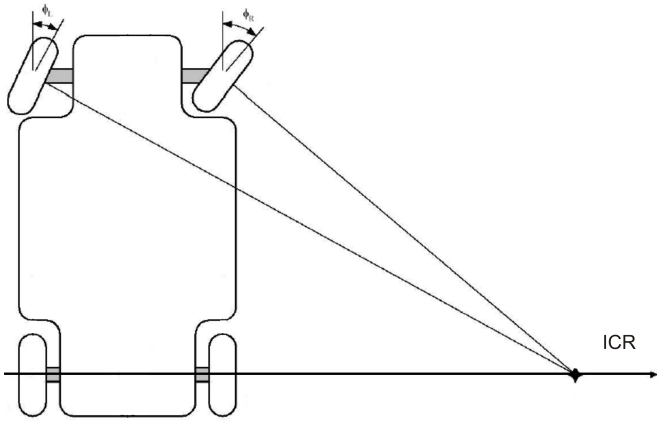

Fig. 1. General Ackermann steering principle.

adjustment with a constricted parameterization to compute the relative scales all at once.

III. Computing rotation and translation

In our previous work [1] we described an algorithm to estimate the relative motion between two camera images taken from a single vehicle-mounted camera. The main contribution of that work was to make use of the fact that many wheeled vehicles move with piece-wise circular motion, assuming the vehicle reference coordinate system is attached to the rigid non-steering axle. For car-like vehicles, this motion is ensured by the Ackermann steering principle [7] (Fig. 1). For a giving steering angle, the wheels are turned in a way so that the car will go around in a circle. Each of the front wheels is turned in a way so that both rotate around the same center, the so called Instantaneous Center of Rotation (ICR). The path of a car therefore follows a piecewise circular trajectory, because every change in the steering angle will result in a new circular trajectory. A camera mounted over the rear axle of the vehicle will also follow circular motion. For visual odometry, we showed in [1] that in practice the motion between to camera frames is nicely circular. This was used to derive a novel motion estimation algorithm for the planar motion case that can compute rotation and translation between two frames from a single point correspondence instead of two point correspondences [8]. This minimal parameterization can be used for robust motion estimation using RANSAC [9]. Because only one point correspondence is necessary the number of random samples is lower than in the 2-point case and the motion estimation is much faster. As a matter of fact, the motion estimation and outlier removal process took less 0.2 milliseconds with a normal DualCore laptop computer. In the following, we will shortly summarize the algorithm introduced in our previous work.

A. The Essential Matrix

Under planar motion, the two relative poses of a camera can be described by three parameters, namely the yaw angle θ and the polar coordinates (ρ, ϕ) of the second position relative to the first position (Fig. 2). Since when using only one camera the scale factor is unknown, we can arbitrarily

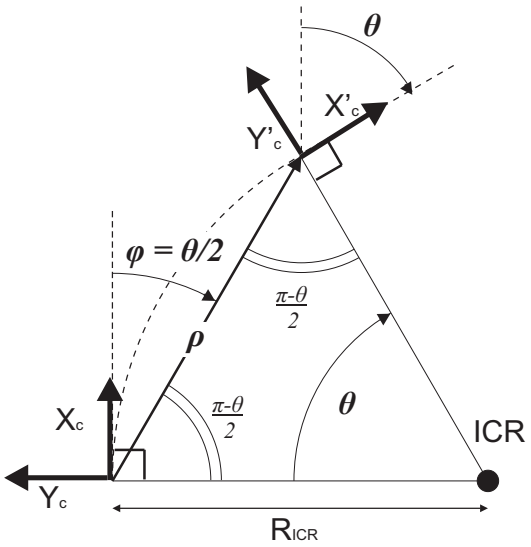

Fig. 2. Relation between camera axes in circular motion.

set ρ to 1. From this it follows that only two parameters need to be estimated and so only two image points are required. However, if the camera moves locally along a circumference (as in Fig. 2) then we have $\phi = \theta/2$; thus, only θ needs to be estimated and so only one image point is required. Observe that straight motion is also described through our circular motion model; in fact in this case we would have $\theta = 0$ and thus $\phi = 0$.

Let us now derive the expression for the essential matrix using the considerations above. Let $\mathbf{R}$ and $\mathbf{T}$ be the unknown rotation and translation matrices which relate the two camera poses. Then, we have

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; \mathbf{T} = \rho \cdot \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \\ 0 \end{bmatrix} \quad (1)$$

because we considered the motion along the xy plane and the rotation about the z-axis. Then, let $\mathbf{p} = [x, y, z]^T$ and $\mathbf{p}' = [x', y', z']^T$ be the image coordinates of a scene point seen from the two camera positions. Observe that to make our approach independent of the camera model, we use spherical image coordinates; therefore $\mathbf{p}$ and $\mathbf{p}'$ are the image points back projected onto a unit sphere (i.e. $\|\mathbf{p}\| = \|\mathbf{p}'\| = 1$). This is always possible once the camera is calibrated.

As known in computer vision, the two unknown camera positions and the image coordinates must verify the epipolar constraint

$$\mathbf{p}'^T \mathbf{E} \mathbf{p} \; = \; 0, \quad (2)$$

where E (called *essential matrix*) is defined as $\mathbf{E} = [\mathbf{T}]_\times \mathbf{R}$,

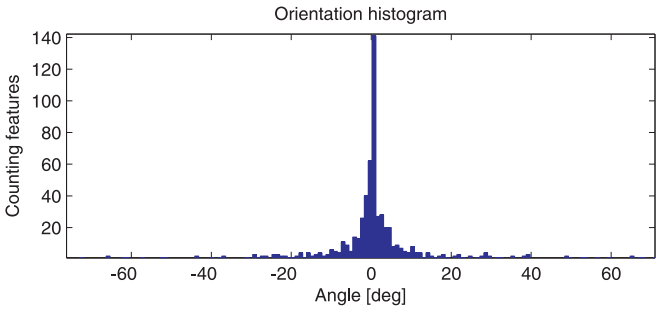

Fig. 3. An example histogram from feature correspondences.

where $[\mathbf{T}]_\times$ denotes the skew symmetric matrix

$$[\mathbf{T}]_\times = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}. \quad (3)$$

Then, using (1), (3), and the constraint $\phi = \theta/2$, we obtain the expression of the essential matrix for planar circular motion:

$$\mathbf{E} = \rho \cdot \begin{bmatrix} 0 & 0 & \sin(\frac{\theta}{2}) \\ 0 & 0 & -\cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) & 0 \end{bmatrix} \quad (4)$$

B. Recovering θ

By replacing (4) into (2), we can observe that every image point contributes to the following homogeneous equation:

$$\sin\left(\frac{\theta}{2}\right) \cdot (x'z + z'x) + \cos\left(\frac{\theta}{2}\right) \cdot (y'z - z'y) = 0 \quad (5)$$

Given one image point, the rotation angle θ can then be obtained from (5) as:

$$\theta = -2\tan^{-1}\left(\frac{y'z - z'y}{x'z + z'x}\right) \quad (6)$$

C. Outlier removal in 2-view motion estimation

The possibility of estimating the motion using only one feature correspondence allows us to implement a very efficient algorithm for removing the outliers, which is based on histogram voting. First, θ is computed from each feature correspondence using (6); then, a histogram H is built where each bin contains the number of features which count for the same θ. A sample histogram built from real data is shown in Fig. 3. When the circular motion model is well satisfied, the histogram has a very narrow peak centered on the best motion estimate θ^*, that is $\theta^* = argmax\{H\}$. As the reader can perceive, θ^* represents our motion hypothesis; knowing it, the inliers can be identified by using reprojection error.

Once the outliers are identified, we refine the motion estimate from all remaining inliers through the 2-point algorithm described in [8].

IV. Computing the scale

To compute the scale of a trajectory we set up a bundle adjustment problem to solve for all the scales at once. Bundle adjustment (BA) minimizes the image reprojection error to refine camera poses and 3D points [2], [10]. The standard method is to refine camera and point parameters by Levenberg-Marquard optimization [10]. Usually bundle adjustment requires a good initial solution so that the optimization process does not get stuck at a local minimum. Having more parameters to optimize makes it harder to find a good solution, therefore here we simplify the optimization problem to have less parameters, namely only the scale.

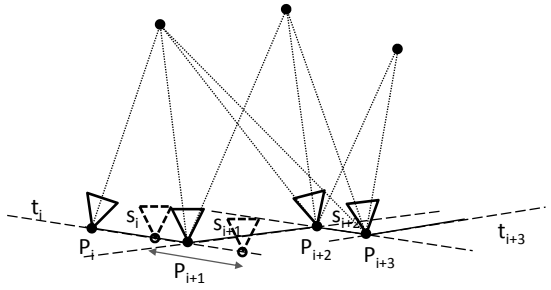

Fig. 4. By optimizing the scale parameters, the cameras $P_i, \ldots, P_{i+n}$ are only allowed to move on the original translation vectors t_i. Rotation and translation vectors are thus unchanged.

A. The Constricted Parameterization of the BA problem

Each camera position P_{i+1} is written as the previous camera position P_i plus a vector to P_{i+1} of length s_i (see Fig. 4 for an illustration).

$$P_{i+1} = P_i + s_i t_i \quad (7)$$

The vector t_i is a unit vector from the current camera to the next camera. For the bundle adjustment we allow the camera positions to move along their vectors. This means only one parameter per camera, i.e. s_i, will be optimized. In the optimization, the cameras are not allowed to move independently. A change in P_i directly influences the position of P_{i+1}. This ensures that the direction of the translation vector t between two cameras remains unchanged. From Eq. 8 it is visible that each camera position P_i is expressed in terms of all the previous cameras.

$$P_i = P_1 + s_1 t_1 + s_2 t_2 + \ldots + s_{i-1} t_{i-1} \quad (8)$$

In our implementation we assume planar motion, thus our camera position is described by two parameters t_x, t_y and the rotation can be described by one parameter r. The rotation of the camera will not be optimized, it will be kept fixed throughout the whole optimization. It would be possible to include also the rotation, which would however double the number of parameters to optimize. A 3D point is described as usual by three parameters.

B. Optimization

The scale estimation algorithm takes a sequence of initial camera poses $P_1, P_2, .., P_n$ and features tracks as input. The

camera poses are represented by the position of the first camera C_1, translation vectors $t_1, ..., t_{n-1}$, rotations $r_1, ..., r_n$ and scales $s_1, ..., s_{n-1}$. The scale values are not known and are therefore initially set to 1. During the optimization only the scale parameters are optimized. For a path with n cameras we optimize n parameters.

The cost function which is minimized is a robustified image reprojection error. We use a robust function to deal with outliers in the data. The robust reprojection error e_r is computed using the Cauchy-function as robustifier,

$$e_r = \ln(1 + \frac{e^2}{\sigma^2}), \tag{9}$$

where σ is the expected standard deviation of the image features and e is the image reprojection error. For reprojection errors larger than the threshold σ, the cost function flattens significantly so that the large errors from outliers do not have to much influence. This robust cost function was reported to be successful in [4].

In contrast to standard BA, we do not give the 3D position of feature points as input. To compute the cost function, the 3D points are re-computed for each iteration of the optimization to evaluate the reprojection function. This is done because the 3D points from the initial solution deviate largely from the solution with the correct scale. During optimization, huge changes in 3D point position would be necessary, which in practice very often leads to a convergence to a local minimum.

C. Outlier detection in n-view feature tracks

Although we use a robust cost function, gross outliers should be removed for the BA to achieve optimal results. To detect outliers in the n-view feature tracks, we perform a 3D reconstruction for each track from it's first 2 views. Then, we run our scale optimization on the single feature track only. We estimate the unknown scales and if it converges we compute the average reprojection error of all the views. If the reprojection error is higher than a certain threshold, the feature track is marked as an outlier and not used in the optimization.

D. Comparison to parameterization in x and y

We compare the parameterization in section IV-A with a parameterization where each camera position is parameterized in its x and y coordinates. This is basically the standard form. Here we allow the camera to move in x, y direction while we still keep the rotation fixed. In contrast to the previous approach - where each camera position is directly linked to its previous camera by Eq. 8 - here the camera positions are not directly linked. Each of them can be optimized independently, however they are still linked by the 3D points. But we will show that this indirect link will not be able to propagate scale changes efficiently.

V. Results

In this section we present visual odometry results on a challenging image data set. The images were acquired by a car equipped with an omnidirectional camera driving through a city. A picture of our vehicle (a Smart) is shown in Fig. 5. The omnidirectional camera is composed of a hyperbolic mirror (KAIDAN 360 One VR) and a digital color camera (SONY XCD-SX910, image size 1280×960 pixels). The camera was installed as shown in Fig. 5. The camera system was calibrated using the toolbox from Scaramuzza [11], [12]. Images were taken at an average framerate of 10Hz. Due to the sharing of memory resources with other sensors the framerate did not stay constant but could drop to as low as 5Hz. The vehicle's speed could range from 0 to 45km/h. The varying and rather low framerate is one of the main challenges of the image set and complicates feature tracking. The distance between feature locations of matches in consecutive frames can be very high, which leads to differently warped image patches because of the omnidirectional camera. The data was collected in real traffic during peak time. Therefore many other moving objects, cars, busses, pedestrians, etc. are present. The route not only goes through nice urban canyons but also leads to open places that lack structure for feature tracking. Feature tracks were computed by SIFT feature matching [13]. For motion estimation only 2-view tracks were used. For scale estimation features tracks up to 10 frames were used.

Fig. 5. The vehicle used in our experiments equipped with the omnidirectional camera (in the circle). The vertical field of view is indicated by the lines.

Fig. 7 shows the result of our visual odometry method for a part of the data set. The recovered path has a length of 960m. Black circles represent the vehicle's location and the blue dots are triangulated feature points. Rotation and translation of the motion are computed from 2-view feature matches and successively added. The scale was estimated with the proposed approach. Fig. 8 shows a comparison of the visual odometry scale with wheel odometry from the car. We compared the local accuracy of our method to the wheel odometry. The graph shows the average absolute scale difference for a sliding window of three-frames length. The

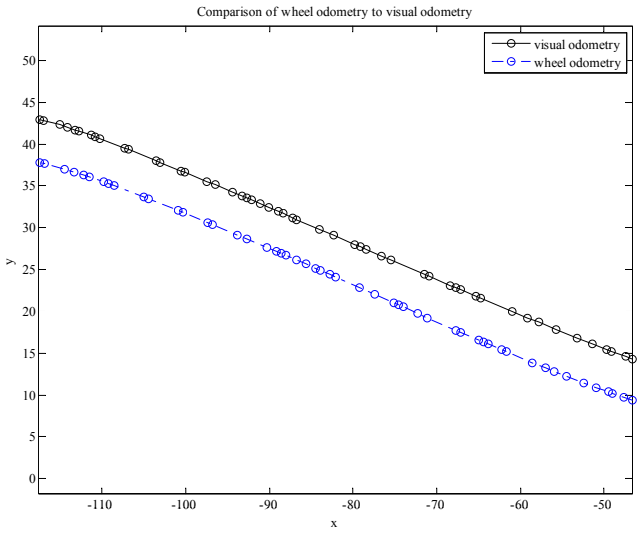

Fig. 6. Comparison of the visual scale estimate to the scale from wheel odometry. Only 50 frames of the sequence are shown to see the details of the scale distribution. The visual scale estimates match the wheel odometry nicely. (The two trajectories are intentionally offset for easy comparison.)

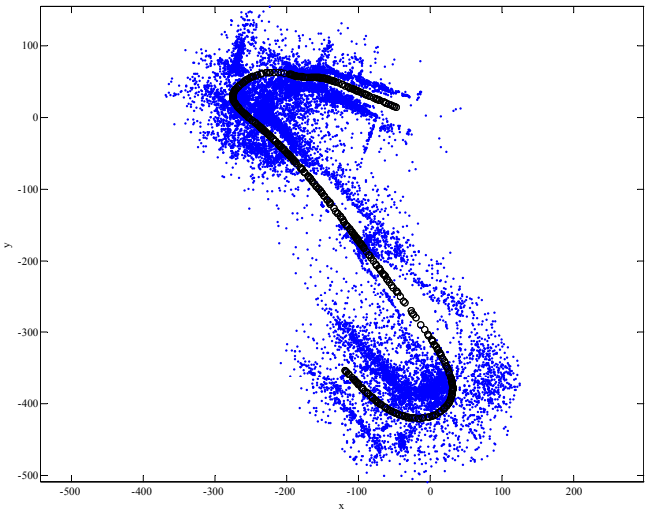

Fig. 7. Vehicle path computed from our visual odometry method. Black circles are the vehicle locations. Blue dots are triangulated feature points.

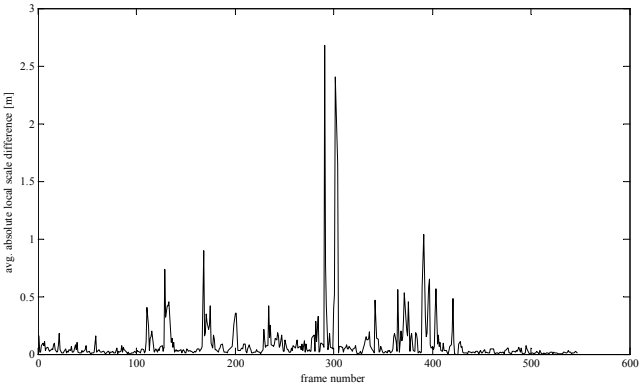

Fig. 8. Local accuracy of the scale estimation. The graph shows the average absolute difference between visual odometry scale and wheel odometry scale for a 3 frame window. The overall average difference is 8.4cm. The large spikes in the plot indicate frames were scale jumps occur. At spots like this the scale estimate might be off by more than 1m.

overall average scale difference is only 8.4cm (std. dev. 21cm). The average distance between frames in the processed dataset is 1.04m. Fig. 6 shows a comparison of the visual scale estimate to the scale from wheel odometry. In this figure, 50 frames of the sequence are shown to see the details of the scale distribution. The visual scale estimates match the wheel odometry.

The plot in Fig. 9 shows the deficiencies of the standard x, y parameterization as described in section IV-D. With this parameterization the scale changes do not propagate well. After optimization, the path still overlaps the initial solution, only locally the scale got adapted. With the constricted parameterization the scale changes get propagated. The shape of the trajectory was therefore changed towards matching the ground truth trajectory as shown in the figure.

Fig. 10 shows a comparison of the visual odometry to the wheel odometry. Fig. 11 shows a part of the trajectory with triangulated features as an overlay onto a satellite image. The triangulated features fit nicely to the map. Our approach is designed as a batch process and thus will not run in realtime. The number of parameters to optimize determines the amount of computer memory needed to solve the problem. Here, by reducing the number of parameters to optimize larger problems can be solved with the same amount of memory.

VI. Conclusions

In this paper we described a new method for computing visual odometry for a single vehicle mounted camera. The main contribution is a relative scale estimation method based on a constricted parameterization of a bundle adjustment problem. The relative scales for a sequence of camera positions are solved all at once. The proposed parameterization keeps the optimization problem small, compared to a standard BA problem. Only the scale is estimated, rotation and translation is kept fixed during optimization. We demonstrate that this approach is able to propagate scale changes better compared to a parameterization of the camera positions with the x and y coordinates. We showed results on an image dataset taken with an omnidirectional camera on a vehicle while driving through a city. The estimated scales are very close to a ground truth measured from wheel odometry. Our scale estimates show only small drift which could be completely eliminated if loops were closed. Loop constraints can easily be integrated in our optimization approach which will be one of our future steps.

References

[1] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *Proc. IEEE International Conference on Robotics and Automation, Kobe, Japan*, 2009.

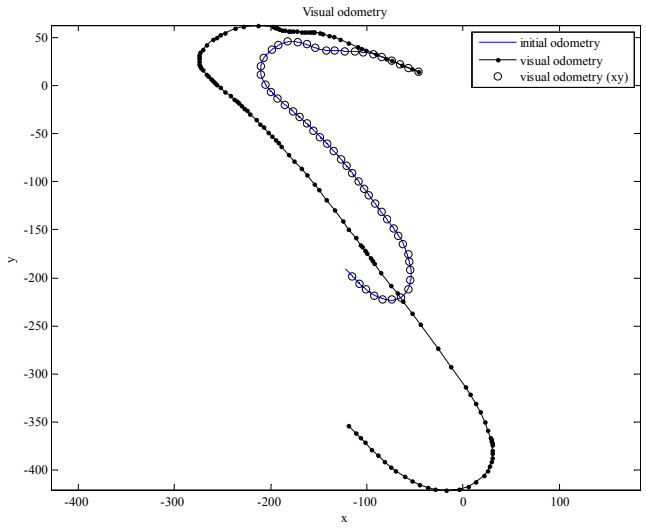

Fig. 9. This plot shows the deficiencies of the standard x, y parameterization. The optimized solution does not propagate the scale changes. After optimization the solution still overlaps with the initial solution, only locally the scale got adapted. With the constricted parameterization the scale changes get propagated. The shape of the trajectory therefore was changed towards the ground truth trajectory.

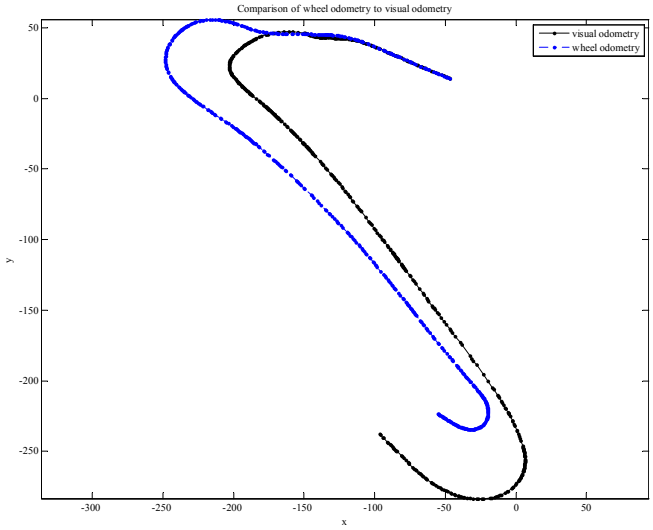

Fig. 10. Visual odometry compared to wheel odometry. At the end of the sequence after the second turn a larger scale drift occurs.

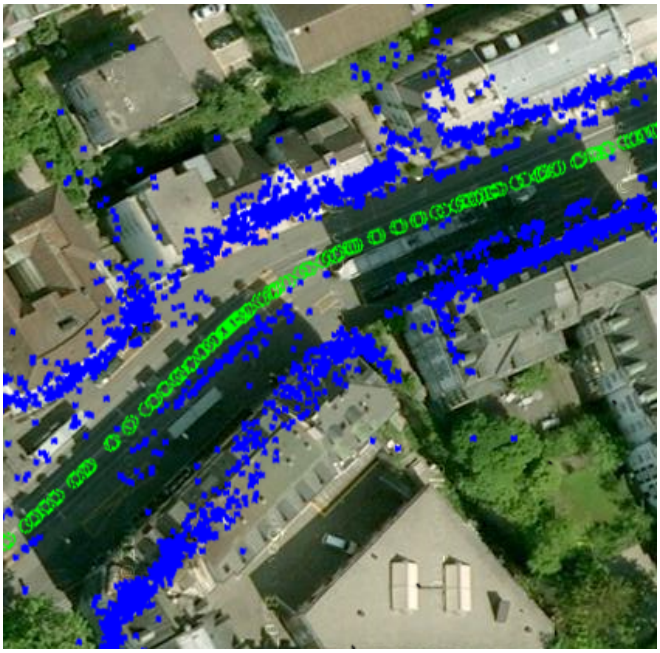

Fig. 11. The estimated vehicle locations and triangulated feature points. The result overlays nicely with satellite map data. Only features points with a reprojection error < 1 pixel are plotted.

[2] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment: A modern synthesis," in *Vision Algorithms Workshop: Theory and Practice*, 1999, pp. 298–372.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, 2004, pp. I: 652–659.

[4] C. Engels, H. Stewenius, and D. Nister, "Bundle adjustment rules," in *Proc. Photogrammetric Computer Vision 2006. ISPRS – Commission III Symposium, Bonn, Germany*, 2006.

[5] E. Mouragnon, F. Dekeyser, P. Sayd, M. Lhuillier, and M. Dhome, "Real time localization and 3d reconstruction," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, New York City, New York*, 2006, pp. I: 363–370.

[6] J. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *IEEE IROS'08*, 2008.

[7] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[8] D. Ortín and J. M. M. Montiel, "Indoor robot motion based on monocular images," *Robotica*, vol. 19, no. 3, pp. 331–342, 2001.

[9] M. A. Fischler and R. C. Bolles, "RANSAC random sampling concensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of ACM*, vol. 26, pp. 381–395, 1981.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, 2000.

[11] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easy calibrating omnidirectional cameras," in *IEEE International Conference on Intelligent Robots and Systems (IROS 2006)*, oct 2006.

[12] D. Scaramuzza, "Ocamcalib toolbox: Omnidirectional camera calibration toolbox for matlab," 2006, google for "*ocamcalib*".

[13] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.