Minimizing Image Reprojection Error For Relative Scale Estimation In Visual Odometry

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

[1]Computer Vision and Geometry Lab, ETH Zurich
[2]Autonomous Systems Lab, ETH Zurich
fraundorfer@inf.ethz.ch, davide.scaramuzza@ieee.org, r.siegwart@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 locally optimal relative scales by minimizing the reprojection error using windowed bundle adjustment. We introduce a minimal parameterization of the bundle adjustment problem to get an efficient optimization procedure. We will present experimental results for scale estimation on challenging image data acquired by an onmidirectional camera mounted on a vehicle. The experiments show the high local accuracy of the scale and the robustness of the algorithm over long distances (∼1000m).**

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 inevitably 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 now robust and accurate relative scale estimation. To achieve this we propose a method that uses a non-linear minimization of the image reprojection error, bundle adjustment. Bundle adjustment [2] is known to produce optimal results in terms of image reprojections. It is in many cases used as a final offline algorithm. However, in [3] Engels et al. showed that windowed bundle adjustment can very well be used in online scenarios and even realtime scenarios. In this paper we describe a parameterization of the bundle adjustment to optimize the scale parameters only. Fewer parameters to optimize leads to a more efficient algorithm suitable for online use. 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 [4] Nister et al. describe a visual odometry system using a single camera. The method uses a combination of relative motion estimation with the 5pt 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 [3]. With a windowed bundle adjustment the cameras where 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.

In [5], 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 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

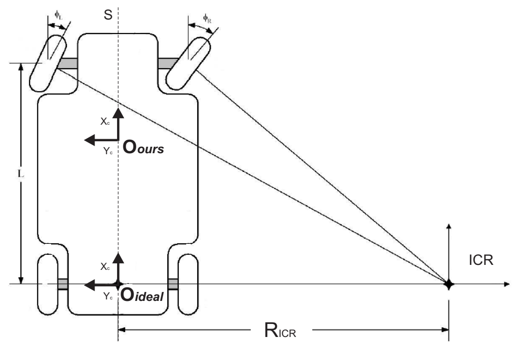


Fig. 1. General Ackermann steering principle (courtesy of Bjorn Jensen).

adjustment to compute the locally very accurate scale estimates to minimize drift. Bundle adjustment is parameterized in a way so that only the necessary parameters are refined.

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 that we were able to estimate the motion while simultaneously removing the outliers of the feature matching process with a very low computational cost. As a matter of fact, the motion estimation and outlier removal process took less 0.2 milliseconds with a normal DualCore laptop computer. The core of the algorithm was the exploitation of the nonholonomic constraints of wheeled vehicles, that is, they posses an Instantaneous Center of Rotation (see Fig. 1). In particular, we showed that, because of the existence of the ICR, every wheeled vehicle moves locally along a circumference and accordingly does also the camera. Although this local circular motion is just an approximation of the real motion of the vehicle, we showed that this model is actually satisfied very well by all vehicle mounted cameras. For car-like vehicles, indeed, the existence of the ICR is ensured by the Ackermann steering principle [6] (Fig. 1). As observed, circular motion can be described through only two parameters (i.e. the radius and the angle of curvature). Therefore, as pointed out in our previous work, one single feature correspondence suffices for computing the relative motion between two images. In case of multiple features, the best motion estimate and the inliers can be identified through histogram voting. In this section, 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 set ρ at 1. From this it follows that only two parameters need to be estimated and so only two image points are required.

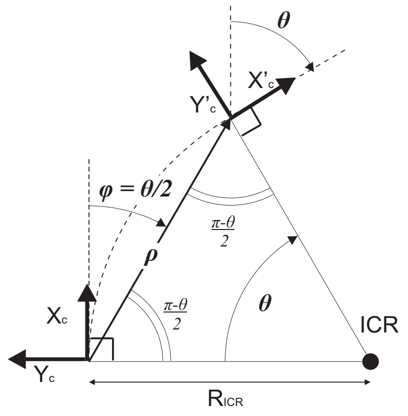


Fig. 2. Relation between camera axes in circular motion.

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 $\mathbf{E} = [\mathbf{T}]_\times \mathbf{R}$, 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:

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

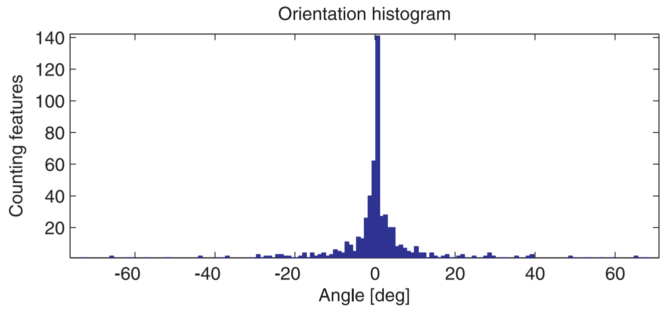


Fig. 3. An example histogram from feature correspondences.

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 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 [7].

IV. Computing the Scale

Bundle adjustment (BA) minimizes the image reprojection error to refine camera poses and 3D points [2], [8]. Standard method is to refine camera and point parameters by Levenberg-Marquard optimization [8]. Usually bundle adjustment requires a good initial solution so that the optimization process does not get stuck at a local minimum. Having more parameter to optimize makes it harder to find a good solution.A naive approach to compute the scale would be to optimize all camera and point parameters and then compute the scale from the distance between the cameras. Instead we describe a way to parameterize the bundle adjustment problem to solve for the scale directly. This simplifies the optimization problem as less parameters, only scale, needs to be optimized.

The scale estimation algorithm (see outline in Fig. 6) takes a sequence of initial cameras poses and features tracks as input. Only rotation and translation vector of the camera poses are known, the length of the translation vector, the scale, is to be estimated. Our windowed BA takes a sequence of n cameras $P_i, P_{i+1}, .., P_{i+n-1}$ and the corresponding feature tracks. In the first iteration the scale of the first 2 cameras P_0, P_1 is fixed (by convention this is usually set to 1). 3D point features are estimated from these two cameras. The scale of the remaining cameras is set to the same scale as for the first 2 cameras, that is 1 (see Fig. 4(a)). Then the scale of the remaining $n-2$ cameras is estimated with BA while keeping the first scale fixed (see Fig. 4(b)). For the next iteration the BA window is shifted by one. Now we already have scale estimates for all but the last camera. It is initialized with the scale of the previous camera. The 3D points for new feature tracks are initialized by triangulation using the first 2 cameras. 3D points from feature tracks that were used in the previous iteration are carried over. These 3D points are now kept fixed in the optimization to fix the overall scale. This allows us to optimize all the camera positions including the first two. This process is now iterated for the full camera sequence (see Fig. 4(c)).

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,

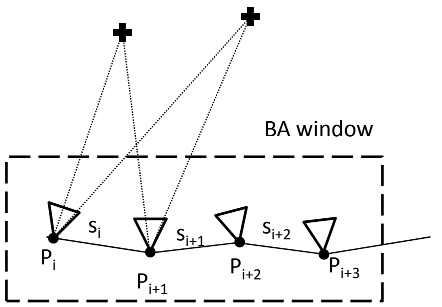
$$e_r = \ln(1 + \frac{e^2}{\sigma^2}), \quad (7)$$

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 don't have to much influence. This robust cost function was reported to be successful in [3]. As we are interested in the scale only, we simplified the BA problem so that only the minimum number of parameters need to be optimized, which leads to an efficient algorithm.

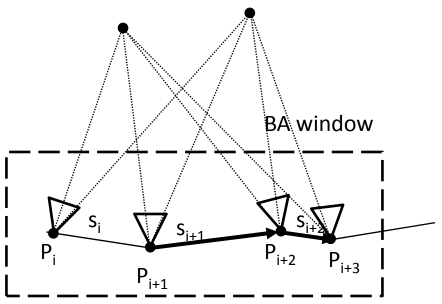
A known problem of bundle adjustment are the initial camera and point estimates [2]. Bundle adjustment will produce good results if the initializations are sufficiently accurate. However, it is not guaranteed that the optimization process finds the global minimum. In fact, it is rather likely that the optimization process gets stuck in a local minimum. In our approach we initialize the scale of a new camera with the scale from the previous camera. This is by no means an accurate initialization. However, the result in [3] show that good results are possible even with poor initialization. In our case where the new camera is described by only one parameter it is even possible to sample a range of initialization values. This allows to cope with situations where the current scale differs largely from the previous scale.

V. Parameterization of the BA problem

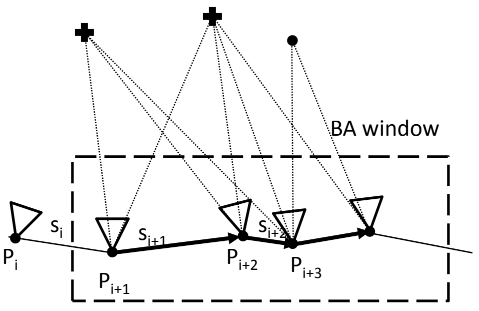
Each camera position P_{i+1} is written as the previous camera position P_i plus a vector to P_{i+1} of length s_i (see



(a)



(b)



(c)

Fig. 4. Scale estimation steps. (a) Initialization. All the scales are set to 1 and initial 3D points are computed from the first two cameras and feature tracks. (b) Initial scale estimate. BA is used to find the remaining scales and refine the 3D points. (c) Iteration. The BA window is shifted to include the next camera position.All scales and new 3D points are estimated. 3D points from the previous iteration are fixed to propagate the scale.

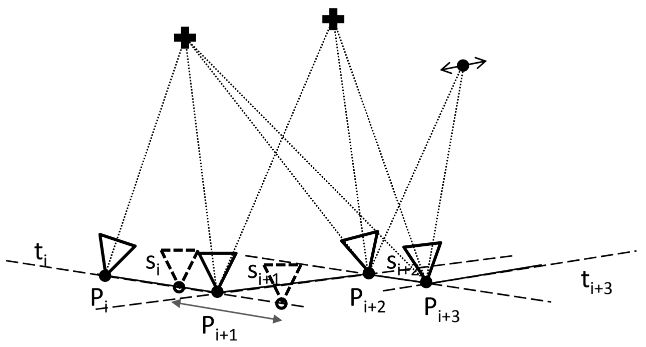


Fig. 5. By optimizing the scale parameters the cameras $P_i, \ldots, P_{i+n}$ are only allowed to move on the original translation vectors t_i. Rotation and translation vectors are thus unchanged. The scale is fixed by fixing 3D points computed in the previous iteration (crosses). New 3D points are initialized by triangulation and get refined in the optimization (dots).

Fig. 5 for an illustration).

$$P_{i+1} = P_i + s_i t_i \tag{8}$$

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. Within a BA window P_j to P_{j+n} each camera position P_i is expressed in terms of all the previous cameras.

$$P_i = P_j + s_j t_j + s_{j+1} t_{j+1} + \ldots + s_{i-1} t_{i-1} \tag{9}$$

The rotation of the camera is not effected by this scale change and needs no update. 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. A 3D point is described as usual by 3 parameters. The total number of parameters to be optimized depends on the number of cameras and number of 3D points. For a BA window with n cameras and p 3D points we optimize $n + 3p$ parameters.

A. Outlier detection in n-view feature tracks

At least one feature track over n cameras is necessary to compute all the n scales. To be robust to image noise we seek the least-squares estimate of multiple feature points, rather than using a single track. However computing the scale for a single track can be used for outlier removal. For every individual feature track we compute an independent scale estimate. The optimization problem has only $n+3$ parameters and is thus quickly solved. We then look at the reprojection error and discard the ones above a threshold as outliers. In addition we keep using a robust cost function (described in previous section) to cope with any remaining outliers.

VI. Results

In this section we present visual odometry results on a challenging image data. 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. 7. 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

- INITIALIZATION
  - Extract n-view feature tracks and two-view matches from first frames
  - Compute R,t from two-view matches
  - Outlier rejection by BA for each individual feature track
  - Compute scale from feature tracks (fix scale of first two cameras)
  - Store 3D points for next iteration
- ITERATION
  - Advance BA window by one
  - Extract n-view feature tracks and two-view matches
  - Compute R,t from two-view matches
  - Outlier rejection by BA for each individual feature track
  - Compute scale from feature tracks (fix 3D points of previous iteration)
  - Store 3D points for next iteration

Fig. 6. Outline of the visual odometry algorithm

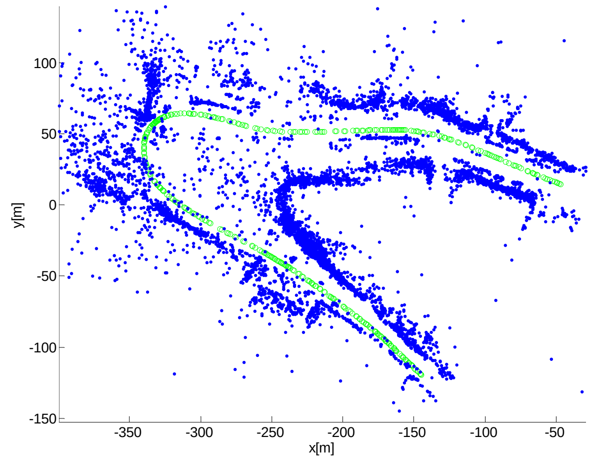


Fig. 8. Vehicle path computed from our visual odometry method. Green circles are the vehicle locations. Blue dots are triangulated feature points.

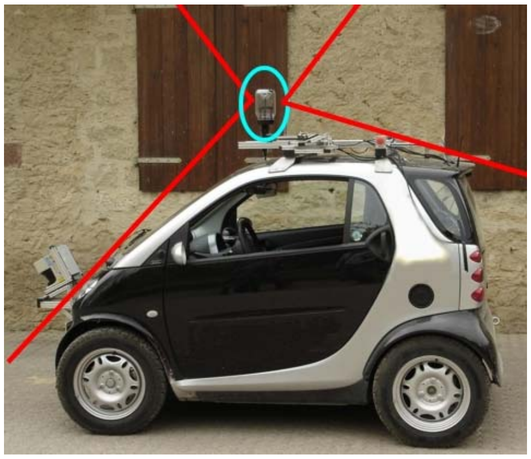


Fig. 7. The vehicle used in our experiments equipped with the omnidirectional camera (in the circle). The vertical field of view is indicated by the lines.

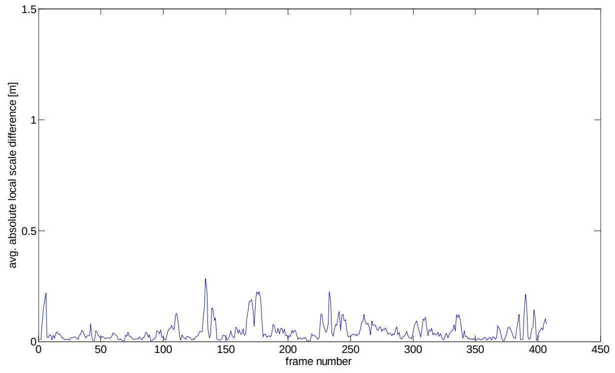


Fig. 9. Local accuracy of the scale estimation. The graph shows the avgerage absolute difference between visual odometry scale and wheel odometry scale for a 4 frame window. The overall average difference is only 3.98cm.

camera was installed as shown in Fig. 7. The camera system was calibrated using the toolbox from Scaramuzza [9], [10]. 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.

Fig. 8 shows the result of our visual odometry method for a part of the data set. The recovered path has a length of 427m. Green 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 successive added. The scale was estimated with a 4-view BA window. The rather low framerate of the camera lead to short feature tracks which did not allow to use larger BA windows. Both, the 2-view matches and the n-view feature tracks were computed by SIFT feature matching [11]. Fig. 9 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 difference in a 4-view window for each frame. From the graph it can be seen that the visual odometry is very close to the wheel odometry. The overall average scale difference is only 4.1cm (std. dev. 4.5cm). The average distance between frames in the processed dataset is 1.04m.

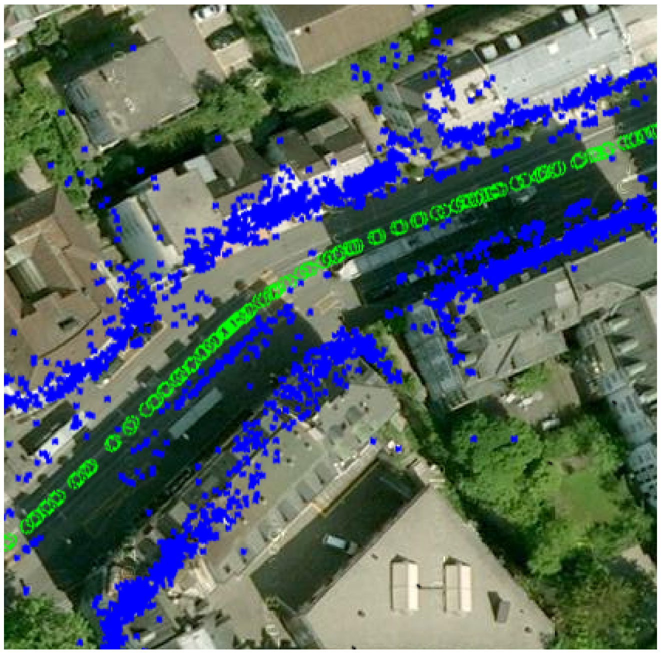


Fig. 10. The estimated vehicle locations and triangulated feature points. The result overlays nicely with satellite map data.

VII. 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 windowed bundle adjustment that complements a very robust relative motion estimation. We provided a parameterization of the bundle adjustment problem that optimizes directly for the relative scale. The cost function that is minimized is the image reprojection error. We demonstrated results on a challenging image dataset acquired in a city during peak time. The estimated scale is comparable to the scale of the car's wheel odometry. Bundle adjustment demonstrated to be able to compute locally accurate scales, however errors still accumulate. This can only be solved by including loop closing constraints. Our approach could in principle also be used as an offline method to optimize the motion estimates including loop closing constraints. This will be investigated in future research.

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.

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

[4] 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.

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

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

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

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

[9] 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.

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

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