Z_∞ - Monocular Localization Algorithm with Uncertainty Analysis for Outdoor Applications

Elmar Mair and Darius Burschka
Technische Universität München (TUM)
Germany

1. Motivation

Localization is an essential task for a meaningful application of a robotic system. The knowledge about its 6DoF *pose* represented as a 3D position and the orientation in space allows a system to execute tasks defined in the 3D workspace and to prevent collisions using the knowledge about obstacles from a model of the environment. A system needs to be capable of estimating its absolute location in the world and to track the changes in the position during its motion. We speak in this context of global and relative localization capabilities. The localization task performs a registration of the sensor perception to a reference. The two above localization alternatives differ in the way how this reference is defined. The global localization requires an a-priori knowledge about the environment stored usually as a geometric model or as an image database that needs to be registered to the current sensory perception, see Thrun (2002) for a comprehensive survey.

Since early work in the 1970's, such as SRI's Shakey (Nilsson, 1984) and Moravec's Cart (Moravec, 1983), there have been great strides in the development of vision-based navigation methods for mobile robots operating both indoors and outdoors. Much of the efficiency and robustness of the recent systems can be attributed to the use of special purpose architectures and algorithms that are tailored to exploit domain specific image cues. For example, road followers rely on finding the road boundary and lane markers (Dickmanns & Graefe, 1988; Hebert et al., 1995) or landmarks (Fennema et al., 1990; Kuhnert, 1990; Lazanas & Latombe, 1992; Levitt et al., 1987) whereas mobile robots navigating in hallways have exploited uniform texture of the floor (Horswill, 1992), floor/wall features (Kriegman et al., 1989; Kim & Navatia, 1995), and overhead lights (Fukuda et al., 1995). Although these domain specializations lead to impressive performance, they do so by imposing particular sensor cues and representations on low-level navigation. As a result, a system that works in one domain may require substantial redesign before it can be used in another.

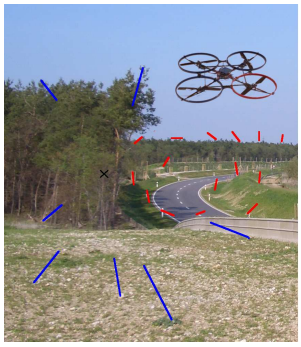


Fig. 1. The Z_∞ algorithm estimates the rotation (red) and the translation up to scale (blue) based on a monocular camera in an efficient way. Therefore, it fits well for resource-limited systems like e.g. flying mobile robots.

One interesting localization approach for mobile navigation on predefined paths are the *Vision-Based Control* approaches. In many cases it is not necessary to use sophisticated map-based systems to control the paths of the robot—instead a simple teaching phase may be sufficient to specify the robot's nominal pathways. Consider for example mobile robots performing delivery in office environments, serving as AGV's in an industrial setting, acting as a tour guide, or operating as a security guard or military sentry. In these situations, the robot repeatedly follows the same nominal path, except for minor perturbations due to control/sensing errors, slippage, or transient/dynamic obstacles. This is in fact on of our goals: to develop a system that can be *walked* in a teaching step through the environment. During this teaching phase the robot learns the path based on sensor perception and then later repeats this path using the same sensors together with previously stored information. Teaching (showing) a robot its nominal pathways has been considered by others including (Matsumoto et al., 1999; 1996; Ohno et al., 1996; Tang & Yuta, 2001). One approach is to use a stored two or three-dimensional representation (map) of the environment together with sensing. A learned path can be replayed by first constructing a map during training and then continuously localizing the robot with respect to the map during playback. However, it is not clear that building a metrically accurate map is in fact necessary for navigation tasks which only involve following the same path continuously. Another approach would be to use no prior information, but rather to generate the control signals directly from only currently sensed data. In this case no path specification at all is possible. An approach based on an Image Jacobian was presented in (Burschka & Hager, 2001).

On the other hand, relative localization approaches track merely the incremental changes in the pose of the robot and do not necessarily require any a-priori knowledge. In relative localization approaches, the initial or a sensor perception from the previous time step is used as a reference.

There are several methods, how the *pose* of a camera system can be estimated. We can distinguish here: multi-camera and monocular approaches. Since a multi-camera system has a calibrated reference position of the mounted cameras, stereo reconstruction algorithms (Hirschmüller, 2008; Brown et al., 2003) can be used to calculate the three-dimensional information from the camera information and the resulting 3D data can be matched to an a-priori model of the environment. Consequently, there has been considerable effort on the problem of mobile robot localization and mapping. This problem is known as simultaneous localization and mapping (SLAM) and there is a vast amount of literature on this topic (see e.g., (Thrun, 2002) for a comprehensive survey). SLAM has been especially succesful in indoor structured environments (Konolige, 2004; Gonzalez-Banos & Latombe, 2002; Tards et al., 2002).

Monocular navigation needs to solve an additional problem of the dimensionality reduction in the perceived data due to the camera projection. A 6DoF pose needs to be estimated from 2D images. There exist solutions to pose estimation for 3 point correspondences for most traditional camera models, such as for example orthographic, weak perspective (Alter, 1994), affine, projective (Faugeras, 1993; Hartley & Zisserman, 2000) and calibrated perspective (Haralick et al., 1994). These approaches constrain the possible poses of the camera to up to four pairs of solutions in the case of a calibrated perspective camera. At most one solution from each pair is valid according to the orientation constraints and the other solution is the reflection of the camera center across the plane of the three points.

In the work from Nister (Nister, 2004), an approach sampling for the correct solution along the rays of projection solving an octic polynomial to find the actual camera pose is presented

that is limited to exactly 3 points neglecting any possible additional information. A solution provided by Davison consists of building a probabilistic 3D map with a sparse set of good landmarks to track (Davison, 2003). Klein was able to achieve even more accurate results as the EKF based approach of Davison by efficiently separating the tracking and the mapping routines (Klein & Murray, 2008).
In this chapter we address the problem of the robust relative localization with monocular video cameras. We propose a localization algorithm that does not require any a-priori knowledge about the environment and that is capable not only of estimation of the 6 motion parameters together but also the uncertainty values describing the accuracy of the estimates. The output of the system is an abstraction of a monocular camera to a relative motion sensing unit that outputs the motion parameters together with the accuracy estimates for the current reading. Only this additional accuracy information allows a fusion of the sensor output in SLAM and other filtering approaches that are based on Kalman Filters.

2. Z_∞: monocular motion estimation

One problem in estimating an arbitrary motion in $3D$ from a real sensor with noise and outliers is to quantify the error and to suppress outliers that deteriorate the result. Most known approaches try to find all six degrees of freedom simultaneously. The error can occur in any dimension and, therefore, it is difficult in such methods to weight or isolate bad measurements from the data set. The erroneous data can be detected and rejected more effectively if the error is estimated separately along all parameters instead of a global value. Thus, a separation of the rotation estimation from the translation simplifies the computation and the suppression of error prone data immensely. This is one major advantage of our Z_∞ algorithm presented in this chapter. We use the usually undesirable quantization effects of the camera projection to separate translation-invariant from translation-dependent landmarks in the image. In fact, we determine the rotational component of the present motion without ever considering the translational one. Fast closed form solutions for the rotation and translation from optical flow exist if the influences are separable. These allow an efficient and globally optimal motion estimation without any a-priori information.
In this section, we describe how we detect translation-invariant landmarks for rotation estimation and how the whole algorithm is applied to an image sequence.

2.1 Principle of Z_∞

Talking about the projective transformation and the pixel discretization of digital cameras, these characteristics are usually considered to be the barriers for image based motion estimation. However, splitting the motion in a rotational and translational part is based on just these effects - Z_∞ uses the projective transformation and pixel discretization to segment the tracked features into a translation-invariant and a translation-dependent component.
We see from the basic camera projection equation

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix} \tag{1}$$

with f representing the focal length, that the X and Y, being $3D$ world coordinates of the imaged point, are both divided by the landmark's distance Z.
According to the projective transformation and the motion equation, which applies a rotation R and a translation T to a point P_i,

$$P_i' = R^T P_i - T \tag{2}$$

the camera motion affects the landmarks mapped onto the image plan as follows.
Each point in the image of a calibrated camera can be interpreted as a ray from the optical center to the 3D landmark intersecting the camera image at a point.

$$P_i \;=\; \lambda_i \vec{n}_i \tag{3}$$

The length λ_i of the vector $\vec{n}_i$ to the landmark i is unknown in the camera projection.
Applying a rotation to the camera appears to rotate the displayed landmarks in an opposite direction by the angle of rotation . The rotation is not affected by the distance to a landmark. This is not the case for translations. A translational motion of the camera results in a different motion for each landmark depending on its distance to the camera.

$$\lambda_i' \vec{n}_i' \;=\; R\,(\lambda_i \vec{n}_i \;+\; \vec{T}) \tag{4}$$

Due to the pixel discretization, the translation cannot be measured anymore if a landmark exceeds a certain distance to the camera. This distance

$$z_\infty = \frac{f}{s_m}\, T_m \tag{5}$$

depends on the translational component parallel to the camera plane, the focal length f and the pixel size s_m. T_m and s_m are the translation between frames of the video sequence and pixel-size. Depending on the translational motion between two images z_∞ can be quite close to the camera, e.g., if we move a standard camera with a focal length of $8.6\,mm$ a pixel size of $9.6\,\mu m$ and a framerate of $30\,Hz$ with a translational component parallel to the image plane of about $2\,km/h \rightarrow z_\infty = 16.6\,m$. Fig. 2 illustrates these characteristics of the camera projections.
Therefore, the typical observed motion T_m is smaller than the value assumed above for a motion observed by a camera looking parallel to the motion vector T. This would correspond to a camera looking to the side. An important observation is that T_m is not only scaled by the distance to the observed point but the angle ψ is important for the actual observability of a point motion. We see from Equation 6 directly that a radial motion as it is the case for points along the optical center but also any other motion along the line of projection lets a point become a part of the $\{P_{k\infty}\}$ set of points, from which the translation cannot be calculated (Fig. 3).

$$\begin{gathered}
\Delta T_{s_m} = \cos\psi_m \Delta T \\
\Delta T_m = \tfrac{T_{s_m}}{\cos\varphi_m} = \tfrac{\cos\psi_m}{\cos\varphi_m}\,\Delta T \\
\Delta T_x \;=\; \tfrac{\cos\psi_x}{\cos\varphi_x}\,\Delta T \qquad \wedge \qquad \Delta T_y \;=\; \tfrac{\cos\psi_y}{\cos\varphi_y}\,\Delta T
\end{gathered} \tag{6}$$

Therefore, we have shown that image features can be split into a set which is translation-invariant and a set which is translation-dependent. Tests on outdoor pictures have shown that the contrast of the sky to the ground at the horizon generates many good features to track. Indeed, an average percentage of 60 % of the selected features lies in outdoor images at the horizon.
But how can we identify which feature corresponds to which set? We have no information about the rotation, the translation or the distance of the landmarks visible in the image. The solution is provided in the algorithm described in the next section.

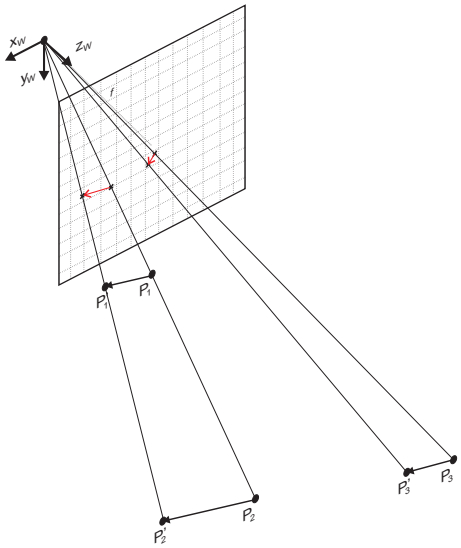


Fig. 2. Optical flow vector 1 (P_1, P_1') is shorter than vector 2 but it is closer to the optical center. Therefore, both vectors result in the same projection. Vector 3 has the same length as vector 1, but it is further from the image plane - therefore, the projection is much shorter. It becomes so small, that the projection of P_3 and P_3' lie within the same pixel. If such a projection results only from a translation then it would be translation-invariant, because the translation has no influence on it.

2.2 RANSAC revisited

The **Ran**dom **Sa**mpling **C**onsensus algorithm (RANSAC) is an iterative framework to find parameters for a given model from an data-set including outliers (Fischler & Bolles, 1981). In each iteration, an as small as possible data subset is randomly chosen to calculate the model parameters. This model is than applied to the residual data set and the elements are split into fitting elements (inliers) and non-fitting elements (outliers). This step is repeated several times and the best model, according to the number of inliers and their residual error, is chosen. Preemptive RANSAC is a variant which allows the algorithm to leave the loop in case that a certain quality criterion applies also before the maximum number of iterations is reached.
In our case, the estimated model is a rotation matrix for which the entries have to be calculated. Therefore, we take three corresponding points from the two subsequent images and we estimate the rotation matrix based on them, as it is explained in Section 2.3. The estimated rotation is applied to the residual data set. In case that the initial three correspondences were from points in the world at a distance further than z_∞ and so represent only the rotational part, the true rotation is estimated. Otherwise, the translational component of the OF-vectors results in a wrong rotation matrix. Applying the resulting rotation on the remaining data set shows if the initial points where truly translational invariant.
In the case that we find other sets calculating the same rotation we can assume that the first three vectors where translation-independent and we found a first estimate for a possible rotation. Another indication for a correct rotation estimate is that the remaining translational vectors after compensation of rotation in all corresponding points need to intersect in one point.

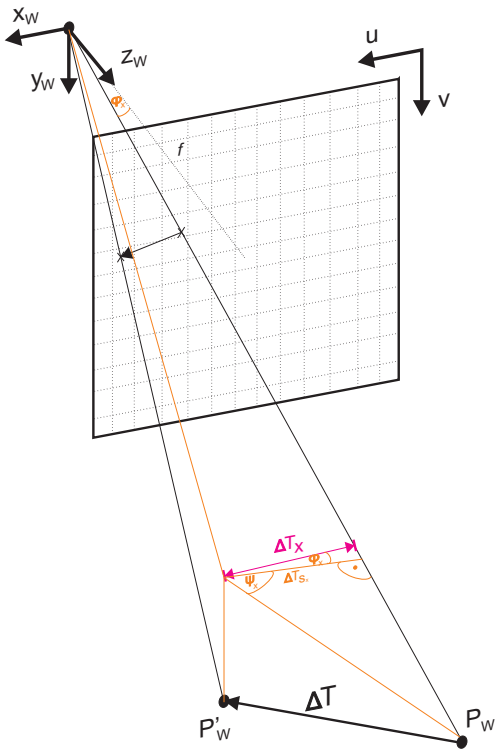


Fig. 3. This drawing visualizes how the measurable translational component ΔT_x can be calculated. In general the optical flow vectors become longer the more parallel they become to the image plane and the closer they get to the image border. The orange auxiliary line illustrates the projections onto the x-z plane.

A degenerate intersection point is also the infinity, where all resulting vectors are parallel in the image. This is a result of a motion parallel to the image. If there are no further OF-pairs agreeing with the calculated rotation, we can expect, that we had at least one translation-dependent element. In case of success, a more accurate and general rotation is estimated on all inlier found by the first estimate. The average error of the back-projection of the vector-endpoints according to the result and the number of found inlier gives an accuracy measure and permits a comparison of the results of each iteration.
The vectors identified as rotation-inlier overlap after the compensation of rotation in both images and are, therefore, translation-invariant. The rotation-outliers represent translation-dependent optical flow vectors and real outliers. Again, we use RANSAC to suppress outliers for a robust translation estimation as described in Section 2.4.
The probability to find an initial set of three translation-invariant optical flow elements depends on the percentage of such vectors in the data set. As described above, tests have shown, that an average of 60% of the features are far enough to be translation-invariant. The lower 5% quantile was about 30%. These results in approximately 37 iterations necessary to assure that in 95% of the cases RANSAC can determine the rotation and divide the data set. Usually

several rotation inliers can also be tracked in the next image. Such features are then preferred for rotation estimation, because it can be assumed that these landmarks still are further than z_∞. Thus, the number of RANSAC iterations for rotation estimation is reduced to one - in practice a few iterations are done to improve the robustness.

2.3 Rotation estimation

The rotation of a point cloud can be estimated in closed form based on the direction vectors to the different landmarks. Therefore, three different registration methods exist: the rotation can be calculated using quaternions (Walker et al., 1991), by singular value decomposition (SVD) (Arun et al., 1987) or Eigenvalue decomposition (EVD) (Horn, 1987). We use the so called Arun's Algorithm, based on SVD. It is analogue to the more familiar approach presented by Horn, which uses an EVD.

The corresponding points used for rotation estimation belong all to translation-invariant points. Therefore, Equation 4 can be abbreviated to

$$P_i' = \lambda_i' \vec{n}_i' = R * P_i = R\,\lambda_i \vec{n}_i = \lambda_i R\,\vec{n}_i, \quad \textbf{with} \quad \lambda_i' \approx \lambda_i \Rightarrow \vec{n}_i' = R\,\vec{n}_i \tag{7}$$

Thus, we must solve following least squares problem

$$min \sum \left\| R P_i - P_i' \right\|^2 \tag{8}$$

This is achieved by the Arun's Algorithm which works as follows. The input for the algorithm are the two corresponding point clouds $\{P_i\}$ and $\{P_i'\}$, which are only rotated and whose rotation we want to estimate. First the origin of the coordinate frame has to be moved to the center of the point cloud:

$$\bar{P} = \frac{1}{n} \sum_{i=1}^{n} P_i, \quad \bar{P}' = \frac{1}{n} \sum_{i=1}^{n} P_i' \tag{9}$$

$$P_i^* = P_i - \bar{P}, \quad P_i'^* = P_i' - \bar{P}' \tag{10}$$

Now, the non scaled sample cross-covariance matrix for these point clouds is calculated to

$$\tilde{M} = \sum_{i=1}^{n} P_i'^* \, P_i^{*T} \tag{11}$$

Therefore, $\frac{1}{n}\tilde{M}$ is the sample cross-covariance matrix between $\{P_i\}$ and $\{P_i'\}$. It can be shown that the rotation matrix which minimizes Equation 8 also fulfills

$$\tilde{R} = argmax_{\tilde{R}}\ tr(R\tilde{M}) \tag{12}$$

Is $(\tilde{U}, \tilde{\Sigma}, \tilde{V})$ the SVD of $\tilde{M}$

$$\tilde{M} \ = \ (\tilde{U}, \tilde{\Sigma}, \tilde{V}) \tag{13}$$

then $\tilde{R}$ can be calculated to

$$\tilde{R} = \tilde{V}\,\tilde{U}^T \tag{14}$$

$\tilde{R}$ is orthonormal, symmetric and positive definite. However, it can happen that all features lie in a plane. In that case not the rotation matrix, put a mirroring matrix is calculated. Such a

result can be recognized by the determinant of R, if $det(\tilde{R}) = -1$ instead of $+1$. The rotation matrix can then be calculated to

$$\tilde{R}' = \tilde{V}'\tilde{U}^T, \quad \text{with } \tilde{V}' = \begin{pmatrix} v_1 \\ v_2 \\ -v_3 \end{pmatrix} \tag{15}$$

v_1, v_2 and v_3 are the rows in V corresponding to the singular values λ_1, λ_2 and λ_3, whereas $\lambda_1 > \lambda_2 > \lambda_3 = 0$.
The uncertainty estimate of the rotation estimation consists in the average reprojection error and the percentage of rotation inliers in the data set.
Actually, the Arun's Algorithm is thought to estimate both, rotation and translation, but to estimate latter the origin of both point clouds must be the same, which is not the case for a moving camera.

2.4 Translation estimation

Compared to the rotation estimation, the calculation of the translation is rather trivial. All the points in $\{P'_i\}$ are rotated back by R^T and consist afterwards only of the translational part of the motion.

$$\widehat{P'} = R^T P' = R^T(R(P + \vec{T})) = P + \vec{T} \tag{16}$$

$\vec{T} \neq! 0$ because only translation-dependent features are used.
We know from epipolar constraints that these back-rotated optical flow vectors $\widehat{P'}$ meet all in the epipole in theory. Due to noise, approximation and discretization issues this will not be the case in real data. Therefore, we calculate the point cloud of all intersections. The centroid of this point cloud is supposed to be the epipole. However, there are also several short vectors, which from very distant points which contain only a small observable translational component. These vectors are inaccurate for direction indication. Further, there are several almost parallel vectors which are also ill-conditioned to calculate their intersection. It seems reasonable to weight the intersection points by a quality criterion resulting from the angle between the rays which form the intersection and their length.
As uncertainty value for the translation the euclidean distance between the calculated epipole and the optical flow vectors is used, as well as the weights used to calculate the intersections' centroid.

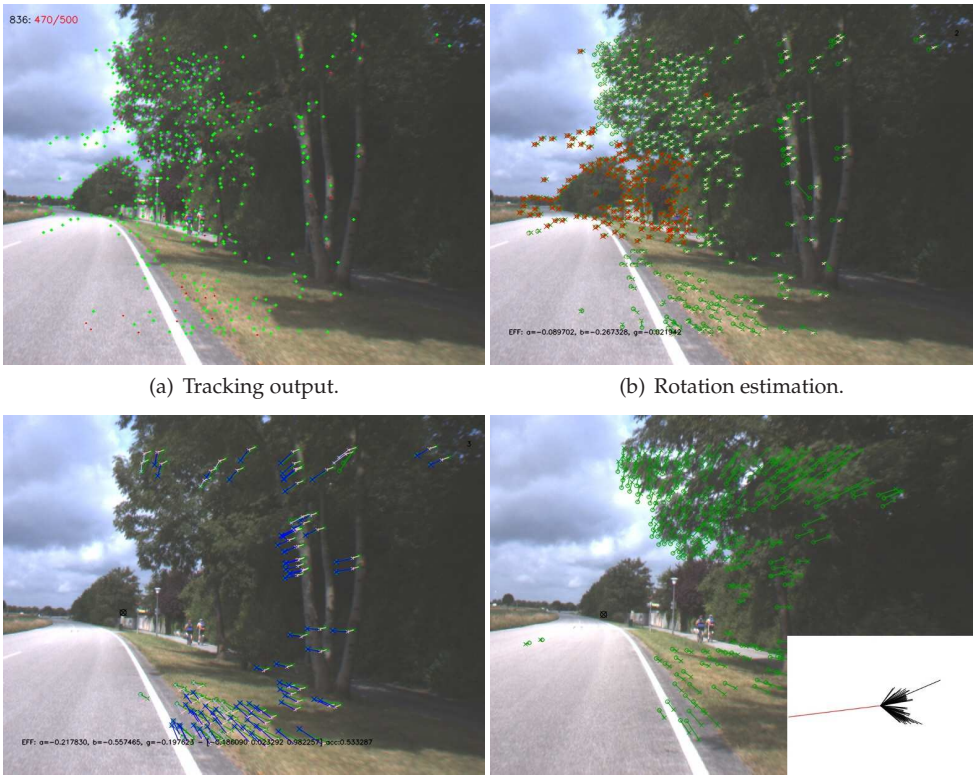
3. Experiments

In the following section some simulation and experimental results are presented. First, some explanations to the experimental setup and the visualization of the Z_∞ output are given. Then, a few simulation results are shown and finally two experiments with real data are demonstrated. Amongst others, the method is compared to an other visual localization approach and to estimates from an inertial measurement unit (IMU).

3.1 Explanation of the Visualization Encoding

In the following, the color encoding of the visualization is briefly explained to improve the reader's understanding. The landmarks are marked green if they could be tracked from the last image or red if they were lost and replaced by new good features to track (see Fig. 4(a)). The optical flow is represented as green vectors, where the original position of the feature is marked by a circle and the tracked location by a cross. The results of the rotation estimation

are colored red. The endpoints of the optical flow vectors are rotated back according to the computed rotation – they are marked as crosses too. Thereby, translation-invariant features are represented by red vectors, while the outlier of the rotation computation are magenta (see Fig. 4(b)). The translation estimation is illustrated blue. Similar to the rotational result, the vectors, which were used for epipole estimation, are dark blue, while the outlier and therefore wrong tracked features are light blue. The black star (circle and cross) represents the computed epipole (see Fig. 4(c)). A result of the obstacle avoidance is shown in Fig. 4(d) and consists in two parts: the main image shows the optical flow vectors which are used for obstacle detection and the small sub-image shows the computed obstacle map. The vectors are mapped regarding their angle to the calculated epipole and a red line illustrates the suggested swerve direction (swerve angle) (please refer also to Section 3.6).



(a) Tracking output. (b) Rotation estimation.

(c) Translation estimation. (d) Obstacle detection.

Fig. 4. Visualization example for the Z_∞ output.

3.2 Simulation results

The Z_∞ algorithm has also been tested extensively with artificial, controllable data. Matlab has been used as simulation environment. Six parameter sets have been chosen to show the

algorithms reliability and insensibility to white noise and outlier (see Table 1). The conditions of this test are as follows:

	white noise (σ^2)	outlier (%)	pixel-discretization
parameter set 1	0	0	no
parameter set 2	0.05	0	no
parameter set 3	0.2	0	no
parameter set 4	0	5	no
parameter set 5	0	0	yes
parameter set 6	0.05	2	yes

Table 1. Characteristics of the parameter sets used for simulation.

A standard camera with $8.6\,mm$ focal length, $8.6\,\mu m$ pixel size and a resolution of 768×576 pixel is simulated. The parameters are transformed to a unit focal camera model first. The simulated translation can randomly vary between -0.05 to $+0.05\,m$ in X- and Y-direction and twice as much along the optical axis. Further, a rotation of up to 0.5° per axis is allowed. At maximum 100 points are simulated in the field of view and they are located within a distance of $1\,km$ and $100\,m$ altitude. The camera is simulated to be in $50\,m$ altitude. The translation computation is executed every 10^{th} frame. Fig. 5 and 6 show an example of such a rotation and translation estimation on artificial data.
The simulated white noise is added after projection onto the camera plane and the outlier can have a size of up to 10 pixels – a local tracker would not find any further correspondences neither. Each parameter set is tested on a simulated image sequence of 2000 images. The most important parameters during the simulation are:

- maximum number of iterations for rotation estimation: 40
- maximum number of iterations for translation estimation: 20
- maximum number of steps per translation estimation: 5

	average rotational error (°)	average transl. dir. error (°)	failed rotation calculations	failed transl. calculations
parameter set 1	0.0109	1.71	0	0
parameter set 2	0.0150	2.59	0	1
parameter set 3	0.0295	4.74	1	1
parameter set 4	0.0098	3.82	2	10
parameter set 5	0.0188	2.82	0	2
parameter set 6	0.0213	6.10	1	1

Table 2. Average rotation and direction of translation error and the number of failed rotation and translation computations for the simulation of the six parameter sets of Table 1.

The results of the simulation are shown in Table 2. While the rotation estimation is rather robust against noise and outliers, the translation computation suffers from these characteristics. Too many outlier even prevent a successful translation estimation.
Also other ill-conditioning circumstances could be simulated. In the following we depict a few insights which could be verified based on the results of the simulations:

- The estimation of the rotation about the optical axis is ill-conditioned. This results in an approximately twice as large error compared to the other two axes.

- While the rotation estimation does not depend at all on the translation, an error in the computation of the rotation will also be reflected in the translational result. Large errors at the rotation estimation even prevent the translation estimation – this is also a reason for the large number of failed translation computations in Table 2.
- If the translation is quite small it is more probably to misguide the rotation estimation because the translational component is within the error tolerance of the rotation computation. This fact is also noticeable in the experiment of Section 3.4.
- If the camera motion is along the optical axis, the estimation of the direction of translation is conditioned best (see also Fig. 6). If the camera translation becomes perpendicular to the direction of view, small errors in the rotation estimation, noise or a few outliers may yield a large error. An exact parallel motion would imply that the translational optical flow vectors meet at infinity, which is computationally ill-conditioned. This fact should be highly recommended when choosing a camera setup for the Z_∞ algorithm. It works best if the cameras are mounted in the direction of motion, if such a preference exists.

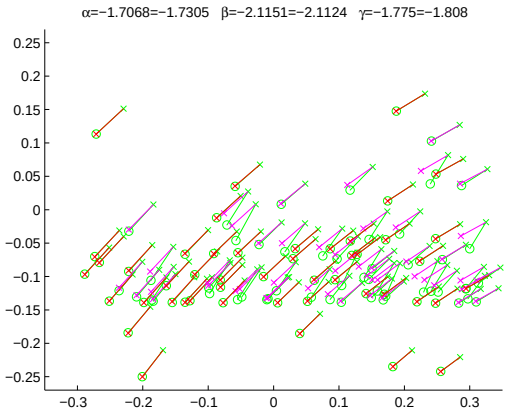


Fig. 5. Simulation of the rotation estimation in Matlab.

3.3 Experimental setup

To test our algorithm, first feature correspondences have to be found. Therefore, the Kanade-Lucas-Tomasi (KLT) tracker has been used (Lucas & Kanade, 1981). Good features are selected according to (Shi & Tomasi, 1994). The tracker is a local tracker with subpixel accuracy. It has been chosen due to its performance and robustness. Fig. 4(a) shows an example output of the tracker with 500 features. However, also global tracker, like SURF have been successfully tested (see Section 3.7).

Fig. 7 shows the processing times of a C++-Implementation of the algorithm on a 1.6 GHz Intel Core Duo processor T2300. The same parameters as in the simulations (Section 3.2) were used. The code is not optimized for performance and the processing is only single-threaded. The time was measured with the "gprof" profiling tool.

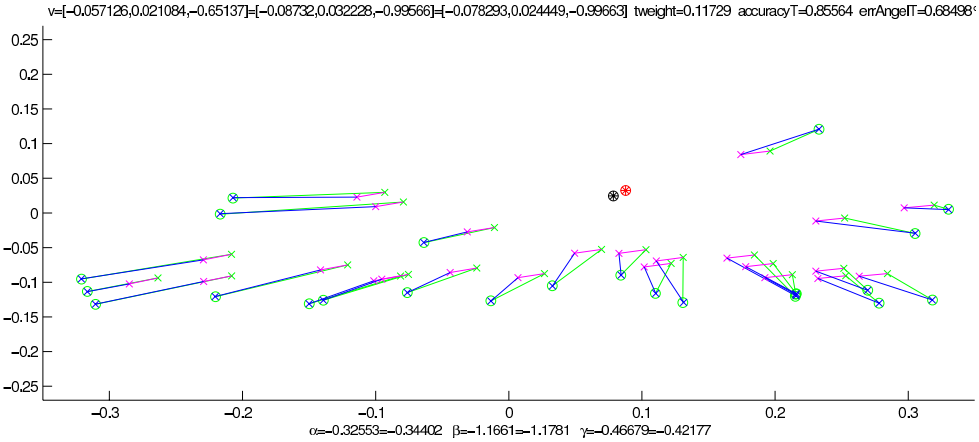


Fig. 6. Simulation of the translation estimation in Matlab.

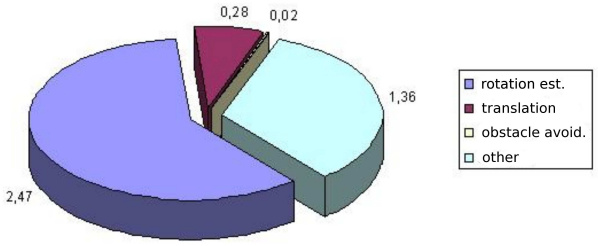


Fig. 7. The computation time for each module of the Z_∞ algorithm in milliseconds. In total an average time of 4.13 ms on a 4 years old Notebook is required.

3.4 Comparison of Z_∞ to stereo-supported visual localization (SSVL)

In this experiment the Z_∞ algorithm is compared to an algorithm, which has proven to be even accurate enough for short-range 3D-modeling (Strobl et al., 2009). The method has been designed for such short-distance applications, but it has been shown to be also accurate and reliable for mobile robots (Meier et al., 2009). Comparing this indoor and outdoor localization algorithm, we want to highlight the advantages and disadvantages, the bottlenecks and the limits of both approaches.

In the next section the modules of this SSVL approach are briefly described to improve the understanding of the reader.

3.4.1 Stereo-supported visual navigation

This visual localization method is indeed also based on monocular image processing. However, usually it is supported by a second camera. In a nutshell: A patch-based KLT tracker is used to select good features and track them from image to image. Those features have to be initialized for the subsequent pose estimation. Therefore, three different initialization possibilities exist: structure from motion, structure from reference or structure from stereo. In order to get the proper scale and to be independent from any modifications of the environment, a

fast subpixel-accurate stereo initialization has been invented and is typically used. The pose is calculated by the so called robustified visual GPS (RVGPS) algorithm (Burschka & Hager, 2003). Every time the features get lost or are moving outside of the field of view, new features are initialized using the stereo images. However, the old feature sets are not dropped, but maintained by an intelligent feature set management. As soon as the camera comes back to a location it has already been, the features are refound and reused for tracking. Thus, the accumulated bias gets reduced again and the error is kept as small as possible. This method does not provide any probabilistic filtering like Kalman or particle filters. Nevertheless, any of these methods can be used to smooth the localization output and reject outlier. For further details to this algorithm please refer to (Mair et al., 2009).

3.4.2 Comparison results: Z_∞ - SSVL

For this experiment two Marlin F046C cameras from Allied Vision have been mounted on a stereo rig with $9\,cm$ baseline on the top of a Pioneer3-DX from MobileRobots Inc. (Fig. 8). The robot has been computed to follow a circular path with $3\,m$ diameter. During the run 1920 stereo image pairs were acquired. The cameras have been slightly tilted to the ground to increase the close areas in the images, while keeping the horizon and the structures at infinity well visible. Although, the odometry is rather imprecise, the Pioneer has been able to close the circle with only a few centimeters of error.

Fig. 8. The Pioneer3-DX carrying the two Marlin cameras.

Fig. 9 shows the trajectory computed by the stereo supported localization. It is apparent that there is no bundle adjustment applied: wrong initialized features lead to a wrong pose estimation, but are usually immediately removed by the M-estimator in RVGPS. However, if there are too many wrong features, it can last some images until they get removed from the current feature set. Such a behavior is visible in the lower half of the circle, where the computed trajectory leaves the circular path and jumps back again after a while. Nevertheless, such outlier could also be suppressed by a probabilistic filter, like e.g. a Kalman filter.
To provide an accurate stereo initialization with this baseline, all landmarks have to lie within a range of approximately $10\,m$. This image sequence is therefore ill conditioned for RVGPS-based localization due to several facts: only features in the lower half of the image can be used for pose estimation, due to the large rotation the feature sets are leaving the field of view quickly and the ground on which the robot is operating has only little structure which eases

miss-matches and the loss of features during tracking (see Fig. 11).
The large rotational component of the motion compared to the amount of translation prevents the Z_∞ algorithm to determine the direction of translation accurately. The small translation yields in a z_∞-range of 10 cm per image. Therefore, the translation can only be evaluated after several images, when it gets accumulated to a measurable length. Further, the small translational component in the optical flow vectors yields the features to be tested as translational invariant and therefore to be misleadingly used for rotation estimation and to be rejected from further translation estimation. A large number of images for translation accumulation increases also the probability of the features to get lost by the tracker – this is also favored by the poor structured ground.

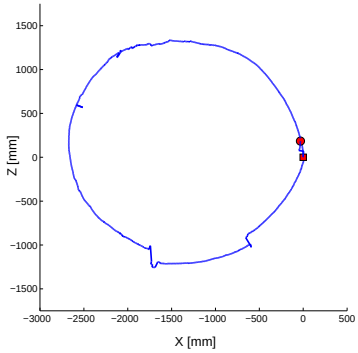


Fig. 9. The computed trajectory by the SSVL algorithm. The red square is the starting point and the red circle the endpoint of the path.

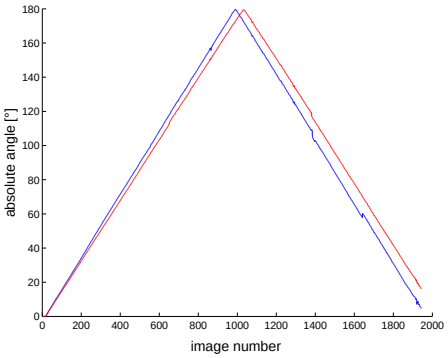


Fig. 10. These figure compares the absolute angle estimated by the Z_∞ (red) and the SSVL (blue) algorithm.

Fig. 10 illustrates the computed angle of the Z_∞ and SSVL algorithm and Table 3 shows some key-values regarding the result. The SSVL method proves its accuracy with only 3.8° absolute error, even though the ill-conditioned images. On the other hand, Z_∞ accumulates an absolute error of 16° during the sequence. This seems to be a large error before taking into account two facts: The difference between both rotation estimations is not zero-mean as expected. This can be easily explained reminding again the problem to detect translation-variant landmarks at such a small translational motion. Close landmarks are also used for rotation estimation, as shown in Figure 12, and because the translation is always done in the same direction on a circular path, the estimation error corresponds to a measurement bias.
The other advantage of the SSVL approach is its bias reduction strategy. As explained in Section 3.4.1, the pose estimation is based on initialized feature sets. The distance of each feature is estimated by stereo triangulation and is not updated anymore. The poses are always calculated respective to the image, where the landmarks were initialized. Thus, the only time, where a bias can be accumulated is when SSVL switches to a new feature set. The memory consumption grows continuously if the camera does not operate in a restricted environment. That prevents the SSVL algorithm to be used on platforms operating in large workspaces like outdoor mobile robots. In this experiment 116 sets were necessary. On the other hand, the Z_∞ method does not need to provide a feature management. It estimates the pose from image

to image, which makes it adequate for outdoor applications of resource-limited platforms without environment restrictions. As drawback it suffers from an error accumulation like all the non-map-based algorithms without a global reference. Considering the relative error in the sense of error per accumulation step, we achieve a much smaller error for the Z_∞ algorithm than the SSVL approach. For the Z_∞ method with 1920 steps, an average error of 0.008° arises. Despite of the error accumulated by the mistakenly used translation-variant features, this is almost 4 times less than with SSVL, where 116 steps yield an error of 0.03° each. In Fig. 13 the computed angles for each frame are plotted. Only a few outlier can be identified, whereas Z_∞ has less outlier than SSVL. This is not so problematic for SSVL due to its global pose estimation approach as described above. Fig. 14 shows again the differences between the estimated rotations of both methods for each image. The mean difference corresponds to the explained bias of -0.02° with a standard deviation of 0.14°.

Fig. 11. A screenshot of the SSVL algorithm at work. Only close landmarks are used.

Fig. 12. The small translations in that example yield also close features to be used for rotation estimation.

	abs. X-rot. error (°)	abs. Y-rot. error (°)	abs. Z-rot. error (°)	no. of error accumulations	aver. Y-rot. error per acc. (°)	average rotation (°)
SSVL	3.8	2.87	-0.01	116	0.03	0.21
Z_∞	16.0	-1.14	1.37	1920	0.008	0.19

Table 3. Keyvalues of the comparison between Z_∞ and SSVL for the circle experiment. The absolute rotation error for all three axes, the number of error accumulations of each method, the average rotational error about the Y-axis for each error accumulation and the average rotation per frame are listed.

3.5 Comparison of Z_∞ to an inertial measurement unit

For this experiment the inertial measurement unit (IMU) "MTi" from Xsens has been mounted on a Guppy F046C camera from Allied Vision (see Fig. 15). The IMU provides the absolute rotation, based on the earth's magnetic field and three gyroscopes, and the acceleration of the device. The camera images and the IMU data were acquired with $25\,Hz$. Before the data can be evaluated, a camera to IMU calibration is necessary.

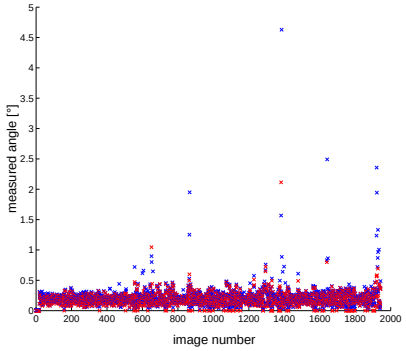


Fig. 13. This figures shows the angles computed for each frame - red the Z_∞ and blue the SSVL results. The crosses mark the angle about the Y-axis. Only these marks can be identified in the plot.

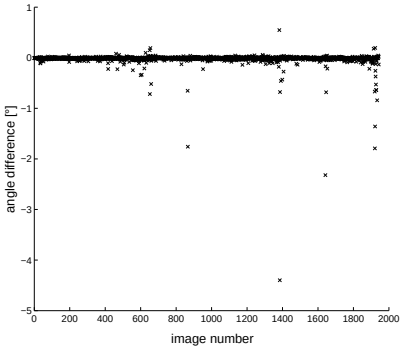


Fig. 14. This plot shows the difference between the Z_∞ and SSVL rotation estimates along the Y-axis. Only a few outlier can be identified, which are detected by the SSVL's M-estimator. Hence, they do not have any effect on the final result.

3.5.1 Camera to IMU calibration

Again, we benefit from the Z_∞ algorithm – this time to calibrate the camera to the IMU. To speed up the calibration procedure we use a global tracker, namely SURF. The camera-IMU setup was rotated along all the axes while logging the sensor's data. From that data four images were chosen for the final calibration (see Fig. 3.5.1). The relative rotations R_{imurel} and R_{camrel} were computed and the rotation between the two coordinate frames ${}^{imu}R_{cam}$ can be determined using the Euler axis-angle representation. The axis of the rotation between the frames is the cross-product of the rotation axes of the two relative rotations, while the angle corresponds to their dot-product.
An other way to calibrate the two coordinate frames would be to solve following equation

$$ {}^{imu}R_{cam}\, R_{camrel}\, {}^{imu}R_{cam}^{T} = R_{imurel} \tag{17}$$

Fig. 15. The Xsens IMU is mounted on a Guppy camera. The data sets were acquired by holding the camera-IMU system out of a car.

This problem is known as $AX = XB$ problem and is not trivial – therefore, the upper approach has been used.

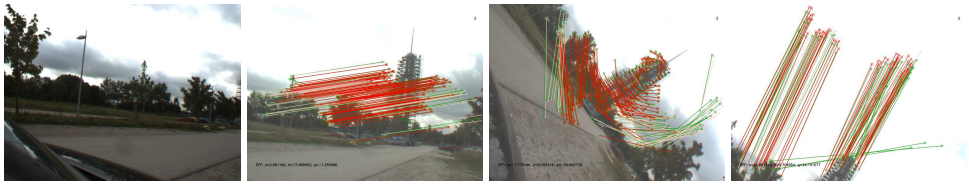


Fig. 16. These four images were used for camera to IMU calibration.

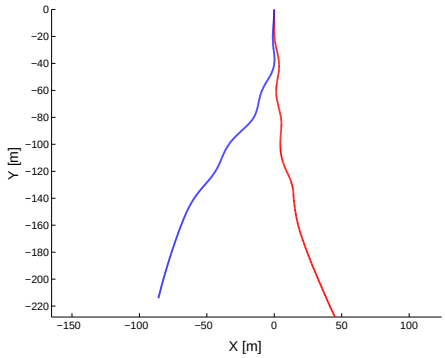
3.5.2 Comparison results: Z_∞ - IMU

An IMU provides accurate information about the rotational speed. Combined with a magnetic field sensor, which measures the earth's magnetic field, it becomes a powerful angle sensor. The Xsens IMU comes up with an internal Kalman filter which fuses these two sensors. However, the accelerometer is a bad translation sensor, due to several facts. First, the measured acceleration has to be integrated twice in order to get the translational motion and second, this integration has to start when the device is static in order to get the proper velocity. An other problem are the forces which are also measured by an accelerometer like the earth's gravitation and the centrifugal force in curves.
Fig. 17 visualizes two data sets[1] acquired while holding the camera-IMU system out of a car. The left image shows a sinuous trajectory on a slightly left bent street. The right image describes the path through a turnaround. Again, this street is slightly left bent. Both runs prove the Z_∞ based direction of translation estimation to outperform the IMU based acceleration integration. The main problem are the centrifugal forces, which, ones integrated, they are not removed from the velocity vector anymore and cause the estimated pose to drift away.
One of the main disadvantages of the Z_∞ translation estimation is the lack of scale. Tests, where we tried to keep at least the scale constant over a run, failed because of the numerical conditions of the problem. Therefore, in this experiment the same scale resulting from the IMU measurement is used for weighting the Z_∞ based translation vector.
Fig. 18 shows the angles measured along the axis perpendicular to the street, when driving the sinuous line illustrated in the left image of Fig. 17. Even though the IMU has a Kalman filter integrated, the Z_∞ algorithm provides a much smooth measurement with less outlier.
However, both, Z_∞ and IMU, have the problem to miss very small rotations, because the pixel displacement is to small to be tracked or the rotation lies within the noise of the gyroscopes respectively. This sensitivity problem can be solved at least for the Z_∞ approach by simply keeping the same image as reference frame instead of changing it with each iteration. The algorithm can then swap respective to the magnitude of the measured rotation.

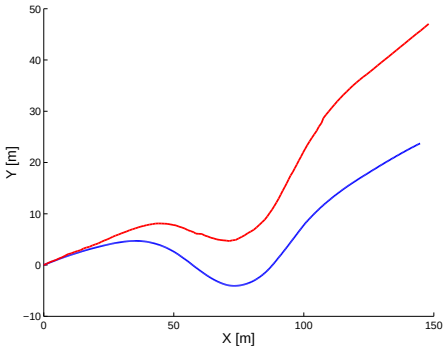
3.6 Obstacle Avoidance

Potential obstacles can be detected by evaluating the optical flow. Long vectors correspond to close or/and fast objects. The closer or the faster an object is moving respective to the camera, the larger the optical flow vectors become. For obstacle avoidance this relation is advantageous in a twofold manner. A certain length is identified, depending on the camera and the

[1] The pictures at the right hand-side are "Google maps" screen-shots: http://maps.google.com



(a) Driving a sinuous line.



(b) Passing a turnaround.

Fig. 17. Two trajectories measured by the Z_∞ algorithm (red) and the IMU (blue). In the pictures at the right-hand side the driven trajectories are sketched in black.

application, which defines a "dangerous" object. In a static environment, where only the camera is moving, such objects should be detected earlier if the motion of the camera is fast. This gives enough time to consider the obstacle for trajectory planning. In a dynamic environment, fast objects are much more dangerous than slow ones and, therefore, these should be detected earlier.

The question remains how to determine where the objects are located. Without a global reference it is only possible to describe them relative to the camera. Therefore, the calculated epipole is used and objects are detected respective to the direction of translation. An obstacle map shows the characteristics of obstacles: their angle respective to the epipole, their level of danger as length and a suggested swerve direction for obstacle avoidance. Fig. 19 shows the output of the obstacle avoidance module on an image sequence acquired by the Pioneer equipped with a Marlin F046C camera as in the experiment of Section 3.4. The robot was programmed to move straight forward for the time of acquisition.

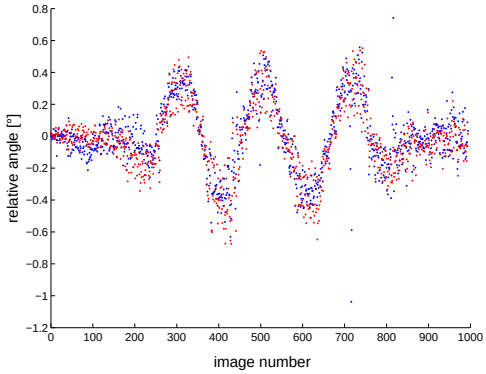
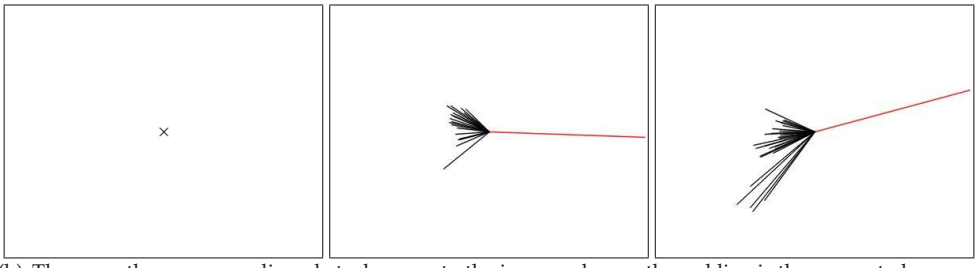


Fig. 18. This image shows the relative angle measured during the same run as in the left image of Fig. 17. The Z_∞ rotation estimation is red, while the IMU data is blue.

(a) Input images for the obstacle detection: the black star is the epipole and the green vectors are the optical flow.



(b) These are the corresponding obstacle maps to the images above - the red line is the suggested swerve direction for obstacle avoidance.

Fig. 19. The Z_∞ algorithm provides also an obstacle detection and their avoidance according to the built obstacle map.

3.7 SURF based Z_∞

The Z_∞ algorithm has also been successfully tested with the global tracker SURF. Like every global tracker, SURF also provides much more outlier and the accuracy is in general worse than with local trackers. We used SURF when processing images acquired by an Octocopter, taking 10 megapixel pictures of the church in Seefeld, Germany. Fig. 20 shows the rotation and

translation estimation result for one such image pair. The images where made in quite large distances. Thus, SURF features have been extracted and the rotation and also translation was computed for each feature set. Despite the high percentage of outlier and the poor accuracy, the Z_∞ algorithm could successfully process the image sequence.

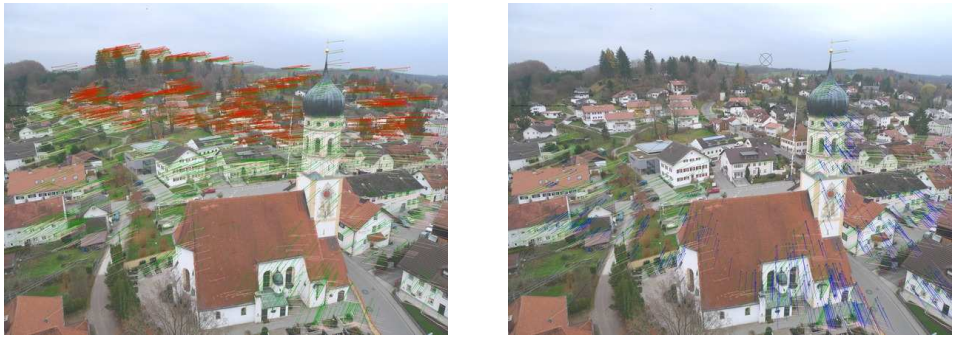


Fig. 20. Motion estimation based on SURF features.

4. Conclusion

We have presented a localization algorithm which is based on a monocular camera. It relies on the principle of separate estimation of rotation and translation, which significantly simplifies the computational problem. Instead of solving an octal polynomial equation, like it is the case with the 8-point algorithm, the SVD of a 3×3 matrix and the calculation of the intersection point cloud and its centroid are sufficient for motion estimation. Further, this separation allows an own uncertainty feedback for both motion components. No aprori knowledge is required for the algorithms nor any information needs to be carried between the image pairs. The algorithm is requires very little processing time time and is very memory-efficient. Nevertheless, we achieve comparable results to other localization methods and we have shown its accuracy in simulations and real world experiments. The drawback of the algorithm is its restriction to outdoor applications. However, especially for such applications the processing time and memory consumption is crucial and the proposed approach seems to fit the requirements for outdoor localization on mobile resource-limited platforms very well. Although, the rotation and translation is calculated analytically, the algorithm to separate these components, RANSAC, is an iterative method. Nevertheless, as explained in Section 2.3, from the second motion estimation on, the number of iterations necessary to split the data into translation-dependent and translation-invariant is reduced to a few. Therefore, Z_∞ is a motion estimation algorithm based on a monocular camera for outdoor mobile robots applications. Unlike other state of the art approaches, this image based navigation can run also on embedded, resource-limited systems with small memory and little processing power, keeping a high level of accuracy and robustness.

In the future we plan to adapt the algorithm to work also indoor by down-sampling the images and provide a hierarchical, iterative rotation estimation. An other point of research is the fusion of IMU and camera data, because experiments have shown that a camera and an IMU are two sensors which complement each other.

5. Acknowledgment

We thank the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) for making available a Pioneer 3-DX and a Xsens IMU for our experiments. Further, we want to acknowledge the permission to process their image sequence of the church of Seefeld, Germany.

6. References

Alter, T. (1994). 3D Pose from 3 Points Using Weak Perspective., *IEEE PAMI, Vol. 16, No. 8* pp. 802–808.

Arun, K. S., Huang, T. S. & Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets, *IEEE Trans. Pattern Anal. Mach. Intell.* **9**(5): 698–700.

Brown, M. Z., Burschka, D. & Hager, G. D. (2003). Advances in Computational Stereo, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(8): 993–1008.

Burschka, D. & Hager, G. (2001). Dynamic composition of tracking primitives for interactive vision-guided navigation, *Proc. of SPIE 2001, Mobile Robots XVI*, pp. 114–125.

Burschka, D. & Hager, G. D. (2003). V-GPS - image-based control for 3d guidance systems, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera, *International Conference on Computer Vision*, Vol. 2.

Dickmanns, E. D. & Graefe, V. (1988). Dynamic monocular machine vision, *Machine Vision and Applications* **v**: 223–240.

Faugeras, O. (1993). *Three-Dimensional Computer Vision*, The MIT Press.

Fennema, C., Hanson, A., Riseman, E., Beveridge, J. & Kumar, R. (1990). Model-directed mobile robot navigation, *IEEE Trans. on Robotics and Automation* **20**(6): 1352–69.

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM* **24**(6): 381–395.

Fukuda, T., Ito, S., Arai, F. & Yokoyama, Y. (1995). Navigation system based on ceiling landmark recogntion for autonomous mobile robot, *IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 150–155.

Gonzalez-Banos, H. H. & Latombe, J. C. (2002). Navigation Strategies for Exploring Indoor Environments, *International Journal of Robotics Research, 21(10-11)* pp. 829–848.

Haralick, R., Lee, C., Ottenberg, K. & Nölle, M. (1994). Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem, *International Journal on Computer Vision, 13(3)* pp. 331–356.

Hartley, R. & Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*, Cambridge University Press.

Hebert, M., Pomerleau, D., Stentz, A. & Thorpe, C. (1995). Computer vision for navigation; the cmu ugv project, *Proceedings of the Workshop on Vision for Robots*, IEEE Computer Society Press, pp. 87–96.

Hirschmüller, H. (2008). Stereo processing by semiglobal matching and mutual information, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(2): 328–341.

Horn, B. K. P. (1987). Closed-formsolution of absolute orientation using unit quaternion, *Journal of the Optical Society of America A, Vol. 4, page 629-642* .

Horswill, I. (1992). Polly: A vision-based artificial agent, *AAAI*, pp. 824–829.

Kim, D. & Navatia, R. (1995). Symbolic navigation with a generic map, *Proceedings of the Workshop on Vision for Robots*, IEEE Computer Society Press, pp. 136–145.

Klein, G. & Murray, D. (2008). Improving the agility of keyframe-based SLAM, *10th European Conference on Computer Vision (ECCV'08)*, pp. 802–815.

Konolige, K. (2004). Large-scale Map-making, *In Proceedings of the National Conference on AI (AAAI)* .

Kriegman, D. J., Triendl, E. & Binford, T. O. (1989). Stereo vision and navigation in buildings for mobile robots, *IEEE Trans. on Robotics and Automation* **5**(6): 792–803.

Kuhnert, K.-D. (1990). Fusing dynamic vision and landmark navigation for autonomous driving, *IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 113–119.

Lazanas, A. & Latombe, J.-C. (1992). Landmark-based robot navigation, *Proc. Am. Assoc. Art. Intell.*

Levitt, T. S., Lawton, D. T., Chelberg, D. M. & Nelson, P. C. (1987). Qualitative landmark-based path planning and following, *Proceedings of AAAI-87*, Morgan Kaufman, Los Altos, pp. 689–694.

Lucas, B. D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision, *Proceedings of the International Joint Conference on Artificial Intelligence*.

Mair, E., Strobl, K. H., Suppa, M. & Burschka, D. (2009). Efficient camera-based pose estimation for real-time applications, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Matsumoto, Inaba, M. & Inoue, H. (1996). Visual navigation using view-sequenced route representation, *IEEE Conf. on Robotics and Automation*, pp. 83–88.

Matsumoto, Y., Ikeda, K., Inaba, M. & Inoue, H. (1999). Visual Navigation using Omnidirectional View Sequence, *IEEE Int. Workshop on Intelligent Robots and Systems*, Kyongju, Korea, pp. 317–322.

Meier, W., Mair, E., Burschka, D. & Steinbach, E. (2009). Visual homing and surprise detection in cognitive mobile robots using image-based environment representations, *Proceedings of the IEEE International Conference on Robotics and Automation*.

Moravec, H. P. (1983). The Stanford cart and the CMU rover, *Proceedings of the IEEE* **71**(7).

Nilsson, N. J. (1984). Shakey the robot, *Technical Report 323*, SRI Int.

Nister, D. (2004). A Minimal Solution to the Generalised 3-Point Pose Problem, *CVPR* .

Ohno, T., Ohya, A. & Yuta, S. (1996). Autonomous Navigation for Mobile Robots Referring Pre-recorded Image Sequences, *IEEE Int. Workshop on Intelligent Robots and Systems*, Osaka, Japan, pp. 672–679.

Shi, J. & Tomasi, C. (1994). Good features to track, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Strobl, K. H., Mair, E., Bodenmüller, T., Kielhöfer, S., Sepp, W., Suppa, M., Burschka, D. & Hirzinger, G. (2009). The self-referenced DLR 3D-modeler, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Tang, L. & Yuta, S. (2001). Vision Based Navigation for Mobile Robots in Indoor Environments by Teaching and Playing-back Scheme, *ICRA*, pp. 3072–3077.

Tards, J. D., Neira, J., Newman, P. M. & Leonard, J. J. (2002). Robust Mapping and Localization in Indoor Environments using Sonar Data, *International Journal of Robotics Research, 21(4)* pp. 311–330.

Thrun, S. (2002). Robotic mapping: A survey, *In G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann* .

Walker, M., Shao, L. & Voltz, R. (1991). Estimating 3-D location parameters using dual number quaternions, *CVGIP: Image Understanding* **54**(3): 358–367.