TUTORIAL

© DIGITAL VISION

Visual Odometry

Part II: Matching, Robustness, Optimization, and Applications

By Friedrich Fraundorfer and Davide Scaramuzza

Visual odometry (VO) is the process of estimating the egomotion of an agent (e.g., vehicle, human, and robot) using the input of a single or multiple cameras attached to it. Application domains include robotics, wearable computing, augmented reality, and automotive. The term VO was popularized in 2004 by Nister in his landmark article [1], but already appeared earlier, e.g., [88], [89]. The term was chosen for its similarity to wheel odometry, which incrementally estimates the motion of a vehicle by integrating the number of turns of its wheels over time. Likewise, VO operates by incrementally estimating the pose of the vehicle through examination of the changes that movement induces on the images of its onboard cameras. For the VO to work effectively, there should be sufficient illumination in the environment and a static scene with sufficient texture to allow apparent motion to be extracted. Furthermore, consecutive frames should be captured by ensuring that they have sufficient scene overlap.

Digital Object Identifier 10.1109/MRA.2012.2182810
Date of publication: 16 February 2012

The advantage of VO with respect to wheel odometry is that VO is not affected by wheel slip in uneven terrain or other adverse conditions. It has been demonstrated that compared to wheel odometry, VO provides more accurate trajectory estimates, with the relative position error ranging from 0.1 to 2%. This capability makes VO an interesting supplement to wheel odometry and, additionally, other navigation systems such as global positioning system (GPS), inertial measurement units (IMUs), and laser odometry (similar to VO, laser odometry estimates the egomotion of a vehicle by scan matching of consecutive laser scans). In GPS-denied environments, such as underwater and aerial, VO has utmost importance.

This two-part tutorial and survey provides a broad introduction to VO and the research that has been undertaken from 1980 to 2011. Although the first two decades witnessed many offline implementations, only during the third decade did real-time working systems flourish, which has led VO to be used on another planet by two Mars-exploration rovers for the first time. Part I presented a historical review of the first 30 years of research in this field, a discussion on camera modeling and calibration,

and a description of the main motion-estimation pipelines for both monocular and binocular schemes, outlining the pros and cons of each implementation [87]. Part II (this tutorial) deals with feature matching, robustness, and applications. It reviews the main point-feature detectors used in VO and the different outlier-rejection schemes. Particular emphasis is given to the random sample consensus (RANSAC) and the strategies devised to speed it up are discussed. Other topics covered are error modeling, loop-closure detection (or location recognition), and bundle adjustment. Links to online, ready-to-use code are also given. The mathematical notation and concepts used in this article are defined in Part I of this tutorial and, therefore, are not repeated here.

Feature Selection and Matching

There are two main approaches to find feature points and their correspondences. The first one is to find features in one image and track them in the following images using local search techniques, such as correlation. The second one is to independently detect features in all the images and match them based on some similarity metric between their descriptors. The former approach is more suitable when the images are taken from nearby viewpoints, whereas the latter is more suitable when a large motion or viewpoint change is expected. Early research in VO is opted for the former approach [2]–[5] while the works in the last decade concentrated on the latter approach [1], [6]–[9]. The reason is that early works were conceived for small-scale environments, where images were taken from nearby viewpoints, while in the last few decades, the focus has shifted to large-scale environments, and so the images are taken as far apart as possible from each to limit the motion-drift-related issues.

Feature Detection

During the feature-detection step, the image is searched for salient keypoints that are likely to match well in other images. A local feature is an image pattern that differs from its immediate neighborhood in terms of intensity, color, and texture. For VO, point detectors, such as corners or blobs, are important because their position in the image can be measured accurately.

A corner is defined as a point at the intersection of two or more edges. A blob is an image pattern that differs from its immediate neighborhood in terms of intensity, color, and texture. It is not an edge, nor a corner. The appealing properties that a good feature detector should have are: localization accuracy (both in position and scale), repeatability (i.e., a large number of features should be redetected in the next images), computational efficiency, robustness (to noise, compression artifacts, blur), distinctiveness (so that features can be accurately matched across different images), and invariance {to both photometric (e.g., illumination) and geometric changes [rotation, scale (zoom), perspective distortion]}.

The VO literature is characterized by many point-feature detectors, such as corner detectors (e.g., Moravec [2], Forstner [10], Harris [11], Shi-Tomasi [12], and FAST [13]) and blob detectors (SIFT [14], SURF [15], and CENSURE [16]). An overview of these detectors can be found in [17]. Each detector has its own pros and cons. Corner detectors are fast to compute but are less distinctive, whereas blob detectors are more distinctive but slower to detect. Additionally, corners are better localized in image position than blobs but are less localized in scale. This means that corners cannot be redetected as often as blobs after large changes in scale and viewpoint. However, blobs are not always the right choice in some environments—for instance, SIFT automatically neglects corners that urban environments are extremely rich of. For these reasons, the choice of the appropriate feature detector should be carefully considered, depending on the computational constraints, real-time requirements, environment type, and motion baseline (i.e., how nearby images are taken). An approximate comparison of properties and performance of different corner and blob detectors is given in Figure 1. Notice that SIFT, SURF, and CENSURE are not true affine invariant detectors but were empirically found to be invariant up to certain changes of the viewpoint. A performance evaluation of feature detectors and descriptors for indoor VO has been given in [18] and for outdoor environments in [9] and [19].

Every feature detector consists of two stages. The first is to apply a feature-response function on the entire image [such as the corner response function in the Harris detector or the difference-of-Gaussian (DoG) operator of the SIFT]. The second step is to apply nonmaxima suppression on the output of the first step. The goal is to identify all local minima (or maxima) of the feature-response function. The output of the nonmaxima suppression represents detected features. The trick to make a detector invariant to scale

	Corner Detector	Blob Detector	Rotation Invariant	Scale Invariant	Affine Invariant	Repeatability	Localization Accuracy	Robustness	Efficiency
Haris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
FAST	x		x	x		++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
SURF		x	x	x	x	+++	++	++	++
CENSURE		x	x	x	x	+++	++	+++	+++

Figure 1. Comparison of feature detectors: properties and performance.

changes consists in applying the detector at lower-scale and upper-scale versions of the same image [Figure 2(a)]. Invariance to perspective changes is instead attained by approximating the perspective distortion as an affine one.

SIFT is a feature devised for object and place recognition and found to give outstanding results for VO. The SIFT detector starts by convolving the upper and lower scales of the image with a DoG operator and then takes the local minima or maxima of the output across scales and space (Figure 2). The power of SIFT is in its robust descriptor, which will be explained in the following section. The SURF detector builds upon the SIFT but uses box filters to approximate the Gaussian, resulting in a faster computation compared to SIFT, which is achieved with integral images [90].

Feature Descriptor

In the feature description step, the region around each detected feature is converted into a compact descriptor that can be matched against other descriptors. The simplest descriptor of a feature is its appearance, that is, the intensity of the pixels in a patch around the feature point. In this case, error metrics such as the sum of squared differences (SSDs) or the normalized cross correlation (NCC) can be used to compare intensities [20]. Contrary to SSD, NCC compensates well for slight brightness changes. An alternative and more robust image similarity measure is the Census transform [21], which converts each image patch into a binary vector representing which neighbors have their intensity above or below the intensity of the central pixel. The patch similarity is then measured through Hamming distance.

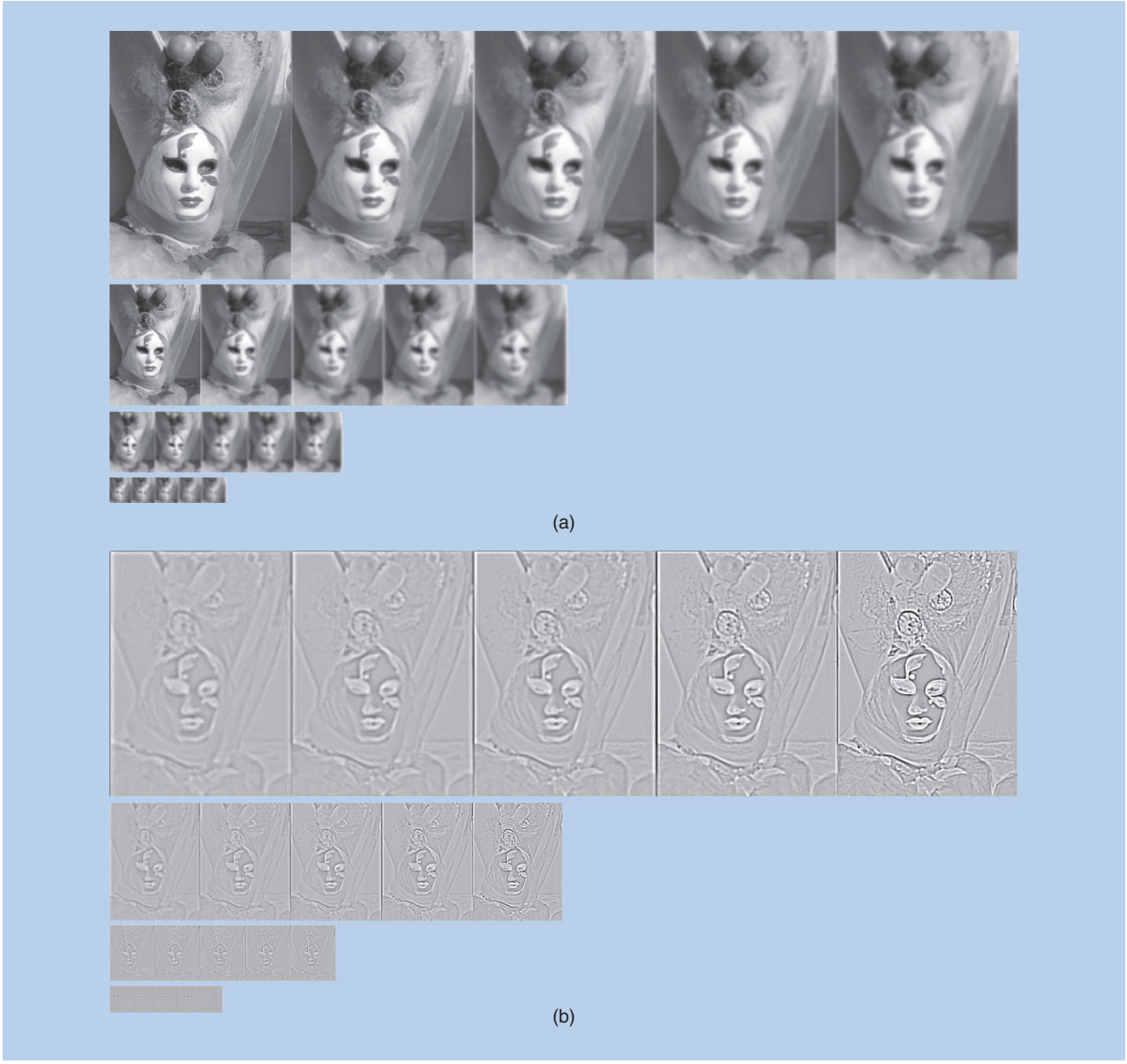

Figure 2. The original image (a, left) is smoothed with four Gaussian filters with different sigmas, and this is repeated after downsampling the image of a factor two. Finally, (b) DoG images are computed by taking the difference between successive Gaussian-smoothed images. SIFT features are found as local minima or maxima of DoG images across scales and space.

In many cases, the local appearance of the feature is not a good descriptor of the information carried by the feature because its appearance will change with orientation, scale, and viewpoint changes. In fact, SSD and NCC are not invariant to any of these changes, and, therefore, their use is limited to images taken at nearby positions. One of the most popular descriptors for point features is the SIFT. The SIFT descriptor is basically a histogram of local gradient orientations. The patch around the feature is decomposed into a 4×4 grid. For each quadrant, a histogram of eight gradient orientations is built. All these histograms are then concatenated together, forming a 128-element descriptor vector. To reduce the effect of illumination changes, the descriptor is then normalized to unit length.

Figure 3. SIFT features shown with orientation and scale.

The SIFT descriptor proved to be stable against changes in illumination, rotation, and scale, and even up to 60° changes in viewpoint. Example of SIFT features are shown in Figure 3. The orientation and scale of each feature is shown. The SIFT descriptor can, in general, be computed for corner or blob features; however, its performance will decrease on corners because, by definition, corners occur at the intersection of edges. Therefore, its descriptor won't be as distinctive as for blobs, which, conversely, lie in highly textured regions of the image.

Between 2010 and 2011, three new descriptors have been devised, which are much faster to compute than SIFT and SURF. A simple binary descriptor named BRIEF [22] became popular: it uses pairwise brightness comparisons sampled from a patch around the keypoint. While extremely fast to extract and compare, it still exhibits high discriminative power in the absence of rotation and scale change. Inspired by its success, ORB [23] was developed, which tackles orientation invariance and an optimization of the sampling scheme for the brightness value pairs. Along the same lines, BRISK [24] provides a keypoint detector based on FAST, which allows scale and rotation invariance, and a binary descriptor that uses a configurable sampling pattern.

Feature Matching

The feature-matching step searches for corresponding features in other images. Figure 4 shows the SIFT features matched across multiple frames overlaid on the first image. The set of matches corresponding to the same feature is called *feature track*. The simplest way for matching features between two images is to compare all feature descriptors in the first image to all other feature descriptors in the second image. Descriptors are compared using a similarity measure. If the descriptor is the local appearance of the feature, then a good measure is the SSD or NCC. For SIFT descriptors, this is the Euclidean distance.

Mutual Consistency Check

After comparing all feature descriptors between two images, the best correspondence of a feature in the second image is chosen as that with the closest descriptor (in terms of distance or similarity). However, this stage may result with features in the second image matching with more than one feature in the first image. To decide which match to accept, the mutual consistency check can be used. This consists in pairing every feature in the second image with features in the first image. Only pairs of corresponding features that mutually have each other as a preferred match are accepted as correct.

Constrained Matching

A disadvantage of this exhaustive matching is that it is quadratic in the number of features, which can become impractical when the number of features is large (e.g., several thousands). A better approach is to use an indexing structure, such as a multidimensional search tree or a hash table, to rapidly search for features near a given feature. A faster feature matching is to search for potential correspondences in regions of the second image where they are expected to be. These regions can be predicted using a motion model and the three-dimensional (3-D) feature position (if available). For instance, this is the case in the 3-D-to-2-D-based motion estimation described in Part I of this tutorial. The motion can be given by an additional sensor like IMU, wheel odometry [25], laser, and GPS, or can be inferred from the previous position assuming a constant velocity model, as proposed in [26]. The

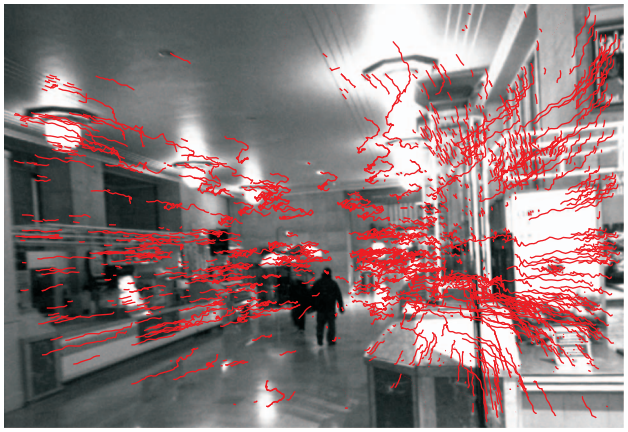

Figure 4. SIFT-feature tracks.

predicted region is then calculated as an error ellipse from the uncertainty of the motion and that of the 3-D point.

Alternatively, if only the motion model is known but not the 3-D feature position, the corresponding match can be searched along the epipolar line in the second image. This process is called *epipolar matching*. As can be observed in Figure 5, a single 2-D feature and the two camera centers define a plane in the 3-D space that intersect both images into two lines, called *epipolar lines*. An epipolar line can be computed directly from a 2-D feature and the relative motion of the camera, as explained in Part I of this tutorial. Each feature in the first image has a different epipolar line in the second image.

In stereovision, instead of computing the epipolar line for each candidate feature, the images are usually rectified. Image rectification is a remapping of an image pair into a new image pair where epipolar lines of the left and right images are horizontal and aligned to each other. This has the advantage of facilitating image-correspondence search since epipolar lines no longer have to be computed for each feature: the correspondent of one feature in the left (right) image can be searched across those features in the right (left) image, which lie on the same row. Image rectification can be executed efficiently on graphics processing units (GPUs). In stereovision, the relative position between the two cameras is known precisely. However, if the motion is affected by uncertainty, the epipolar search is usually expanded to a rectangular area within a certain distance from the epipolar line. In stereovision, SSD, NCC, and Census transform are the widely used similarity metrics for epipolar matching [91].

Feature Tracking

An alternative to independently finding features in all candidate images and then matching them is to detect features in the first image and, then, search for their corresponding matches in the following images. This detect-then-track approach is suitable for VO applications where images are taken at nearby locations, where the amount of motion and appearance deformation between adjacent frames is small. For this particular application, SSD and NCC can work well.

However, if features are tracked over long image sequences, their appearance can undergo larger changes.

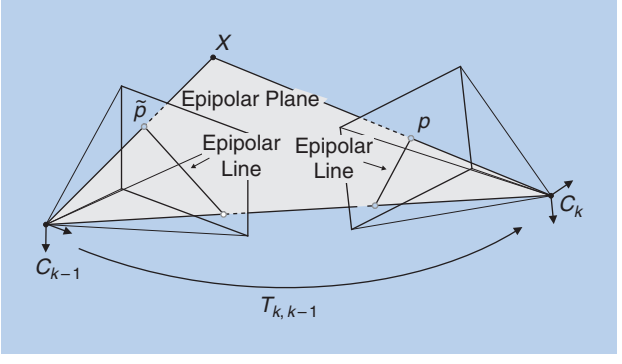

Figure 5. Illustration of the epipolar constraint.

In this case, a solution is to apply an affine-distortion model to each feature. The resulting tracker is often called *KanadeLucasTomasi* (KLT) tracker [12].

Discussion

SIFT Matching

For SIFT feature matching, a distance-ratio test was proposed by the authors initially, for use in place and object detection [14]. This distance-ratio test accepts the closest match (the one with minimum Euclidean distance) only if the ratio between the closest and the second closest match is smaller than a user-specified threshold. The idea behind this test is to remove matches that might be ambiguous, e.g., due to repetitive structure. The threshold for the test can only be set heuristically and an unlucky guess might remove correct matches as well. Therefore, in many cases, it might be beneficial to skip the ratio test and let RANSAC take care of the outliers as explained in the "Outlier Removal" section.

Lines and Edgelets

An alternative to point features for VO is to use lines or edgelets, as proposed in [27] and [28]. They can be used in addition to points in structured environments and may provide additional cues, such as direction (of the line or edgelet), and planarity and orthogonality constraints. Contrary to points, lines are more difficult to match because lines are more likely to be occluded than points. Furthermore, the origin and end of a line segment of edgelet may not exist (e.g., occlusions and horizon line).

Number of Features and Distribution

The distribution of the features in the image has been found to affect the VO results remarkably [1], [9], [29]. In particular, more features provide more stable motion-estimation results than with fewer features, but at the same time, the keypoints should cover the image as evenly as possible. To do this, the image can be partitioned into a grid, and the feature detector is applied to each cell by tuning the detection thresholds until a minimum number of features are found in each subimage [1]. As a rule of the thumb, 1,000 features is a good number for a 640×480-pixel image.

Dense and Correspondence-Free Methods

An alternative to sparse-feature extraction is to use dense methods, such as optical flow [30], or feature-less methods [31]. Optical flow aims at tracking, ideally, each individual pixel or a subset of the whole image (e.g., all pixels on a grid specified by the user). However, similar to feature tracking, it assumes small motion between frames and, therefore, is not suitable for VO applications since motion error accumulates quickly. Another alternative is feature-less motion-estimation methods, such as [31]: all the pixels in the two images are used to compute the relative motion using a harmonic Fourier transform. This method has the advantage to work especially with low-texture images but

is computationally extremely expensive (can take up to several minutes), and the recovered motion is less accurate than with feature-based methods.

Outlier Removal

Matched points are usually contaminated by outliers, that is, wrong data associations. Possible causes of outliers are image noise, occlusions, blur, and changes in viewpoint and illumination for which the mathematical model of the feature detector or descriptor does not account for. For instance, most of the feature-matching techniques assume linear illumination changes, pure camera rotation and scaling (zoom), or affine distortion. However, these are just mathematical models that approximate the more complex reality (image saturation, perspective distortion, and motion blur). For the camera motion to be estimated accurately, it is important that outliers be removed. Outlier rejection is the most delicate task in VO. An example VO result before and after removing the outliers is shown in Figure 6.

RANSAC

The solution to outlier removal consists in taking advantage of the geometric constraints introduced by the motion model. Robust estimation methods, such as M-estimation [32], case deletion, and explicitly fitting and removing outliers [33], can be used but these often work only if there are relatively few outliers. RANSAC [34] has been established as the standard method for model estimation in the presence of outliers.

The idea behind RANSAC is to compute model hypotheses from randomly sampled sets of data points and then verify these hypotheses on the other data points. The hypothesis that shows the highest consensus with the other data is selected as a solution. For two-view motion estimation as used in VO, the estimated model is the relative motion (R, t) between the two camera positions, and the data points are the candidate feature correspondences.

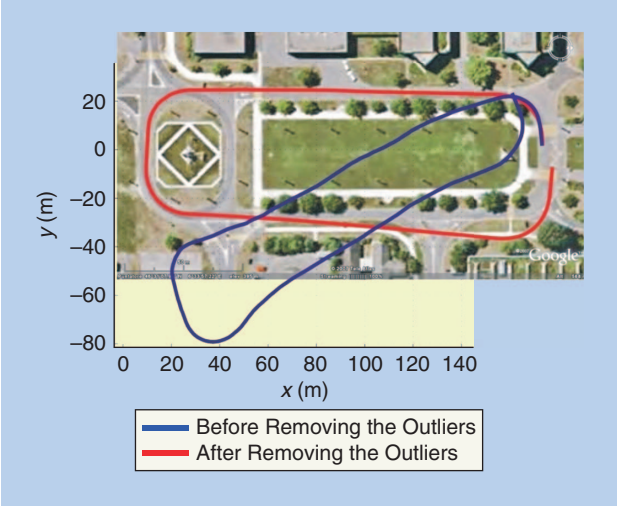

Figure 6. Comparison between VO trajectories estimated before and after removing the outliers. (Photo courtesy of Google Maps © 2007 Google, © 2007 Tele Atlas.)

Inlier points to a hypothesis are found by computing the point-to-epipolar line distance [35]. The point-to-epipolar line distance is usually computed as a first-order approximation—called *Sampson distance*—for efficiency reasons [35]. An alternative to the point-to-epipolar line distance is the directional error proposed by Oliensis [36]. The directional error measures the angle between the ray of the image feature and the epipolar plane. The authors claim that the use of the directional error is advantageous for the case of omnidirectional and wide-angle cameras but also beneficial for the standard camera case.

The outline of RANSAC is given in Algorithm 1.

Algorithm 1. VO from 2-D-to-2-D correspondences.

1) Initial: let A be a set of N feature correspondences2) Repeat
2.1) Randomly select a sample of s points from A
2.2) Fit a model to these points
2.3) Compute the distance of all other points to this model
2.4) Construct the inlier set (i.e. count the number of points whose distance from the model $< d$)
2.5) Store these inliers
2.6) Until maximum number of iterations reached
3) The set with the maximum number of inliers is chosen as a solution to the problem
4) Estimate the model using all the inliers.

The number of subsets (iterations) N that is necessary to guarantee that a correct solution is found can be computed by

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}, \tag{1}$$

where s is the number of data points from which the model can be instantiated, ϵ is the percentage of outliers in the data points, and P is the requested probability of success [34]. For the sake of robustness, in many practical implementations, N is usually multiplied by a factor of ten. More advanced implementations of RANSAC estimate the fraction of inliers adaptively, iteration after iteration.

As observed, RANSAC is a probabilistic method and is nondeterministic in that it exhibits a different solution on different runs; however, the solution tends to be stable when the number of iterations grows.

Minimal Model Parameterizations

As can be observed in Figure 7, N is exponential in the number of data point s necessary to estimate the model. Therefore, there is a high interest in using a minimal parameterization of the model. In Part I of this tutorial, an eight-point minimal solver for uncalibrated cameras was described. Although it works also for calibrated cameras, the eight-point algorithm fails when the scene points are

coplanar. However, when the camera is calibrated, its six degrees of freedom (DoF) motion can be inferred from a minimum of five-point correspondences, and the first solution to this problem was given in 1913 by Kruppa [37]. Several five-point minimal solvers were proposed later in [38]–[40], but an efficient implementation, based on [39], was found only in 2003 by Nister [41] and later revised in [42]. Before that, the six- [43], seven- [44], or eight- solvers were commonly used. However, the five-point solver has the advantage that it works also for planar scenes. (Observe that eight- and seven-point solvers work for uncalibrated, perspective cameras. To use them also with omnidirectional cameras, the camera needs to be calibrated. Alternatively, n -point solvers for uncalibrated omnidirectional cameras have also been proposed [45]–[47], where n depends on the type of mirror or fish eye used. Lim et al. [48] showed that, for calibrated omnidirectional cameras, 6 DoF motion can be recovered using only two pairs of antipodal image points. Antipodal image points are points whose rays are aligned but which correspond to opposite viewing directions. They also showed that antipodal points allow us to independently estimate translation and rotation.)

Despite the five-point algorithm represents the minimal solver for 6 DoF motion of calibrated cameras, in the last few decades, there have been several attempts to exploit different cues to reduce the number of motion parameters. In [49], Fraundorfer et al. proposed a three-point minimal solver for the case of two known camera-orientation angles. For instance, this can be used when the camera is rigidly attached to a gravity sensor (in fact, the gravity vector fixes two camera-orientation angles). Later, Naroditsky et al. [50] improved on that work by showing that the three-point minimal solver can be used in a four-point (three-plus-one) RANSAC scheme. The three-plus-one stands for the fact that an additional far scene point (ideally, a point at infinity) is used to fix the two orientation angles. Using their four-point RANSAC, they also show a successful 6 DoF VO. A two-point minimal solver for 6-DoF VO was proposed by Kneip et al. [51], which uses the full rotation matrix from an IMU rigidly attached to the camera.

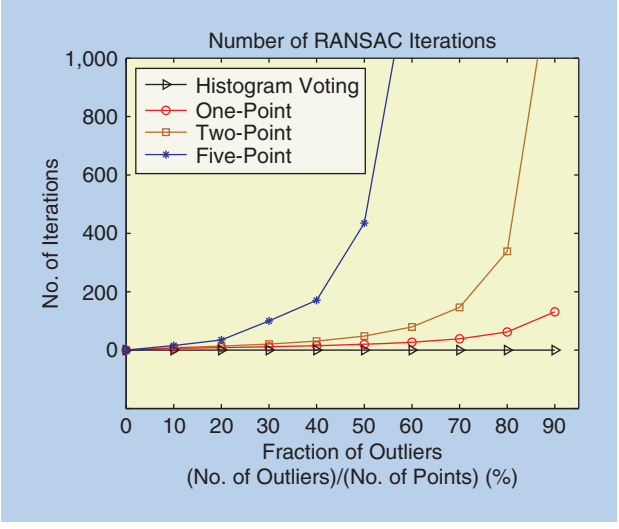

Figure 7. Number of RANSAC iterations versus fraction of outliers.

In the case of planar motion, the motion model complexity is reduced to 3 DoF and can be parameterized with two points as described in [52]. For wheeled vehicles, Scaramuzza et al. [9], [53] showed that the motion can be locally described as planar and circular, and, therefore, the motion model complexity is reduced to 2 DoF, leading to a one-point minimal solver. Using a single point for motion estimation is the lowest motion parameterization possible and results in the most efficient RANSAC algorithm. Additionally, they show that, by using histogram, voting outliers can be found in a small, single iteration. A performance evaluation of five-, two-, and one-point RANSAC algorithms for VO was finally presented in [54].

To recap, the reader should remember that, if the camera motion is unconstrained, the minimum number of points to estimate the motion is five, and, therefore, the five-point RANSAC (or the six-, seven-, or eight-point one) should be used. Of course, using the five-point RANSAC will require less iterations (and thus less time) than the six-, seven-, or eight-point RANSAC. A summary of the number of minimum RANSAC iterations as a function of the number of model parameters s is shown in Table 1 for the eight-, seven-, five-, four-, two-, one-point minimal solvers. These values were obtained from (1), assuming a probability of success $P = 99\%$ and a percentage of outliers $\epsilon = 50\%$.

Table 1. Number of RANSAC iterations.

Number of points (s):	8	7	6	5	4	2	1
Number of iterations (N):	1,177	587	292	145	71	16	7

Reducing the Iterations of RANSAC

As can be observed in Table 1, with $P = 99\%$ and $\epsilon = 50\%$, the five-point RANSAC requires a minimum of 145 iterations. However, in reality, the things are not always so straightforward. Sometimes, the number of outliers is underestimated and using more iterations increases the chances to find more inliers. In some cases, it can even be necessary to allow for thousands of iterations. Because of this, several works have been produced in the endeavor of increasing the speed of RANSAC. The maximum likelihood estimation sample consensus [55] makes the measurement of correspondences more reliable and improves the estimate of the hypotheses. The progressive sample consensus [56]

ranks the correspondences based on their similarity and generates motion hypotheses starting from points with higher rank. Preemptive RANSAC [57] uses preemptive scoring of the motion hypotheses and a fixed number of iterations. Uncertainty RANSAC [58] incorporates feature uncertainty and shows that this determines a decrease in the number of potential outliers, thus enforcing a reduction in the number of iterations. In [59], a deterministic RANSAC approach is proposed, which also estimates the probability that a match is correct.

What all the mentioned algorithms have in common is that the motion hypotheses are directly generated from the points. Conversely, other algorithms operate by sampling the hypotheses from a proposal distribution of the vehicle motion model [60], [61].

Among all these algorithms, preemptive RANSAC has been the most popular one because the number of iterations can be fixed a priori, which has several advantages when real-time operation is necessary.

Is It Really Better to Use a Minimal Set in RANSAC?

If one is concerned with certain speed requirements, using a minimal point set is definitely better than using a nonminimal set. However, even the five-point RANSAC might not be the best idea if the image correspondences are very noisy. In this case, using more points than a minimal set is proved to give better performance (in terms of accuracy and number of inliers) [62], [63]. To understand it, consider a single iteration of the five-point RANSAC: at first, five random points are selected and used to estimate the motion model; second, this motion hypothesis is tested on all other points. If the selected five points are inliers with large image noise, the motion estimated from them will be inaccurate and will exhibit fewer inliers when tested on all the other points. Conversely, if the motion is estimated from more than five points using the five-point solver, the effects of noise are averaged and the estimated model will be more accurate, with the effect that more inliers will be identified. Therefore, when the computational time is not a real concern and one deals with noisy features, using a nonminimal set may be better than using a minimal set [62].

Error Propagation

In VO, individual transformations $T_{k,k-1}$ are concatenated to form the current pose of the robot C_k (see Part I of this tutorial). Each of these transformations $T_{k,k-1}$ has an uncertainty, and the uncertainty of the camera pose C_k depends on the uncertainty of past transformations. This is illustrated in Figure 8. The uncertainty of the transformation $T_{k+1,k}$ computed by VO depends on camera geometry and the image features. A derivation for the stereo case can be found in [3].

In the following, the uncertainty propagation is discussed. Each camera pose C_k and each transformation $T_{k,k-1}$ can be represented by a six-element vector containing the position (x, y, z) and orientation (in Euler angles ϕ, θ, ψ). These six-element vectors are denoted by $\vec{C}_k$ and $\vec{T}_{k,k-1}$, respectively, e.g., $\vec{C}_k = (x, y, z, \phi, \theta, \psi)^\top$. Each transformation $\vec{T}_{k,k-1}$ is represented by its mean and covariance $\Sigma_{k,k-1}$. The covariance matrix $\Sigma_{k,k-1}$ is a 6×6 matrix. The camera pose $\vec{C}_k$ is written as $\vec{C}_k = f(\vec{C}_{k-1}, \vec{T}_{k,k-1})$, that is a function of the previous pose $\vec{C}_{k-1}$ and the transformation $\vec{T}_{k,k-1}$ with their covariances Σ_{k-1} and $\Sigma_{k,k-1}$, respectively. The combined covariance matrix $\vec{C}_k$ is a 12×12 matrix and a compound of the covariance matrices $\Sigma_{k,k-1}$ and Σ_{k-1}. $\vec{C}_k$ can be computed by using the error propagation law [64], which uses a first-order Taylor approximation; therefore,

$$\Sigma_k = J \begin{bmatrix} \Sigma_{k-1} & 0 \\ 0 & \Sigma_{k,k-1} \end{bmatrix} J^\top \tag{2}$$

$$= J_{\vec{C}_{k-1}} \Sigma_{k-1} J_{\vec{C}_{k-1}}{}^\top + J_{\vec{T}_{k,k-1}} \Sigma_{k,k-1} J_{\vec{T}_{k,k-1}}{}^\top, \tag{3}$$

where $J_{\vec{C}_{k-1}}$ $J_{\vec{T}_{k,k-1}}$ are the Jacobians of f with respect to $\vec{C}_{k-1}$ and $\vec{T}_{k,k-1}$, respectively. As can be observed from this equation, the camera-pose uncertainty is always increasing when concatenating transformations. Thus, it is important to keep the uncertainties of the individual transformations small to reduce the drift.

Camera Pose Optimization

VO computes the camera poses by concatenating the transformations, in most cases from two subsequent views at times k and $k-1$ (see Part I of this tutorial). However, it might also be possible to compute transformations between the current time k and the n last time steps $T_{k,k-2}, \ldots, T_{k,k-n}$, or even for any time step $T_{i,j}$. If these transformations are known, they can be used to improve the camera poses by using them as additional constraints in a pose-graph optimization.

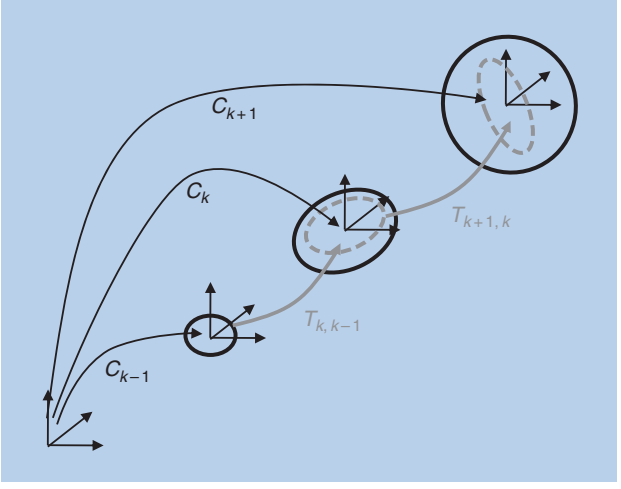

Figure 8. The uncertainty of the camera pose at C_k is a combination of the uncertainty at C_{k-1} (black solid ellipse) and the uncertainty of the transformation $T_{k,k-1}$ (gray dashed ellipse).

Pose-Graph Optimization

The camera poses computed from VO can be represented as a pose graph, which is a graph where the camera poses are the nodes and the rigid-body transformations between the camera poses are the edges between nodes [65]. Each additional transformation that is known can be added as an edge into the pose graph. The edge constraints e_{ij} define the following cost function:

$$\sum_{e_{ij}} \left\| C_i - T_{e_{ij}} C_j \right\|^2, \tag{4}$$

where $T_{e_{ij}}$ is the transformation between the poses i and j. Pose graph optimization seeks the camera pose parameters that minimize this cost function. The rotation part of the transformation makes the cost function nonlinear, and a nonlinear optimization algorithm (e.g., Levenberg-Marquardt) has to be used.

Loop Constraints for Pose-Graph Optimization

Loop constraints are valuable constraints for pose graph optimization. These constraints form graph edges between nodes that are usually far apart and between which large drift might have been accumulated. Commonly, events like reobserving a landmark after not seeing it for a long time or coming back to a previously mapped area are called *loop detections* [66]. Loop constraints can be found by evaluating visual similarity between the current camera images and past camera images. Visual similarity can be computed using global image descriptors (e.g., [67] and [68]) or local image descriptors (e.g., [69]). Recently, loop detection by visual similarity using local image descriptors got a lot of attention and one of the most successful methods are based on the so-called visual words [70]–[73]. In these approaches, an image is represented by a bag of visual words. The visual similarity between two images is then computed as the distance of the visual word histograms of the two images. The visual word-based approach is extremely efficient to compute visual similarities between large sets of image data, a property important for loop detection. A visual word represents a high-dimensional feature descriptor (e.g., SIFT or SURF) with a single integer number. For this quantization, the original high-dimensional descriptor space is divided into nonoverlapping cells by k-means clustering [74], which is called the *visual vocabulary*. All feature descriptors that fall within the same cell will get the cell number assigned, which represents the visual word. Visual-word-based similarity computation is often accelerated by organizing the visual-word database as an inverted-file data structure [75] that makes use of the finite range of visual vocabulary. Visual similarity computation is the first step of loop detection. After finding the top-n similar images, usually a geometric verification using the epipolar constraint is performed and, for confirmed matches, a rigid-body transformation is computed using wide-baseline feature matches between the two images. This rigid-body transformation is added to the pose graph as an additional loop constraint.

Windowed (or Local) Bundle Adjustment

Windowed bundle adjustment [76] is similar to pose-graph optimization as it tries to optimize the camera parameters but, in addition, it also optimizes the 3-D-landmark parameters at the same time. It is applicable to the cases where image features are tracked over more than two frames. Windowed bundle adjustment considers a so-called window of n image frames and then performs a parameter optimization of camera poses and 3-D landmarks for this set of image frames. In bundle adjustment, the error function to minimize is the image reprojection error:

$$\arg\min_{X^i, C_k} \sum_{i,k} \left\| p_k^i - g(X^i, C_k) \right\|^2, \tag{5}$$

where p_k^i is the ith image point of the 3-D landmark X^i measured in the kth image and $g(X^i, C_k)$ is its image reprojection according to the current camera pose C_k.

The reprojection error is a nonlinear function, and the optimization is usually carried out using Levenberg-Marquardt. This requires an initialization that is close to the minimum. Usually, a standard two-view VO solution serves as initialization. The Jacobian for this optimization problem has a specific structure that can be exploited for efficient computation [76].

Windowed bundle adjustment reduces the drift compared to two-view VO because it uses feature measurements over more than two image frames. The current camera pose is linked via the 3-D landmark, and the image feature tracks not only the previous camera pose but also the camera poses further back. The current and $n-1$ previous camera poses need to be consistent with the measurements over n image frames. The choice of the window size n is mostly governed by computational reasons. The computational complexity of bundle adjustment in general is $O((qM + lN)^3)$ with M and N being the number of points and cameras poses and q and l the number of parameters for points and camera poses. A small window size limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible. It is possible to reduce the computational complexity by just optimizing over the camera parameters and keeping the 3-D landmarks fixed, e.g., if the 3-D landmarks are accurately triangulated from a stereo setup.

Applications

VO has been successfully applied within various fields. It is used for egomotion estimation for space exploration (e.g., computing the egomotion of Mars Rovers [25] and that of a planetary lander in the decent phase [77]) and can also be found in consumer hardware, e.g., the Dacuda scanner mouse [78].

Table 2. Software and data sets.

Author	Description	Link
Willow Garage	OpenCV: A computer vision library maintained by Willow Garage. The library includes many of the feature detectors mentioned in this tutorial (e.g., Harris, KLT, SIFT, SURF, FAST, BRIEF, ORB). In addition, the library contains the basic motion-estimation algorithms as well as stereo-matching algorithms.	http://opencv.willowgarage.com
Willow Garage	Robot operating system (ROS): A huge library and middleware maintained by Willow Garage for developing robot applications. Contains a VO package and many other computer-vision-related packages.	http://www.ros.org
Willow Garage	Point cloud library (PCL): A 3-D-data-processing library maintained from Willow Garage, which includes useful algorithms to compute transformations between 3-D-point clouds.	http://pointclouds.org
Henrik Stewenius et al.	Five-point algorithm: An implementation of the five-point algorithm for computing the essential matrix.	http://www.vis.uky.edu/~stewe/FIVEPOINT/
Changchang Wu et al.	SiftGPU: Real-time implementation of SIFT.	http://cs.unc.edu/~ccwu/siftgpu
Nico Cornelis et al.	GPUSurf: Real-time implementation of SURF.	http://homes.esat.kuleuven.be/~ncorneli/gpusurf
Christopfer Zach	GPU-KLT: Real-time implementation of the KLT tracker.	http://www.inf.ethz.ch/personal/chzach/opensource.html
Edward Rosten	Original implementation of the FAST detector.	http://www.edwardrosten.com/work/fast.html
Michael Calonder	Original implementation of the BRIEF descriptor.	http://cvlab.epfl.ch/software/brief/
Leutenegger et al.	BRISK feature detector.	http://www.asl.ethz.ch/people/lestefan/personal/BRISK
Jean-Yves Bouguet	Camera Calibration Toolbox for MATLAB.	http://www.vision.caltech.edu/bouguetj/calib_doc
Davide Scaramuzza	OCamCalib: Omnidirectional Camera Calibration Toolbox for MATLAB.	https://sites.google.com/site/scarabotix/ocamcalib-toolbox
Christopher Mei	Omnidirectional camera calibration toolbox for MATLAB	http://homepages.laas.fr/~cmei/index.php/Toolbox
Mark Cummins	Fast appearance-based mapping: Visual-word-based loop detection.	http://www.robots.ox.ac.uk/~mjc/Software.htm
Friedrich Fraundorfer	Vocsearch: Visual-word-based place recognition and image search.	http://www.inf.ethz.ch/personal/fraundof/page2.html
Manolis Lourakis	Sparse bundle adjustment (SBA)	http://www.ics.forth.gr/~lourakis/sba
Christopher Zach	Simple sparse bundle adjustment (SSBA)	http://www.inf.ethz.ch/personal/chzach/opensource.html
Rainer Kuemmerle et al.	G2O: Library for graph-based nonlinear function optimization. Contains several variants of SLAM and bundle adjustment.	http://openslam.org/g2o
RAWSEEDS EU Project	RAWSEEDS: Collection of data sets with different sensors (lidars, cameras, and IMUs) with ground truth.	http://www.rawseeds.org
SFLY EU Project	SFLY-MAV data set: Camera-IMU data set captured from an aerial vehicle with Vicon data for ground truth.	http://www.sfly.org
Davide Scaramuzza	ETH OMNI-VO: An omnidirectional-image data set captured from the roof of a car for several kilometers in a urban environment. MATLAB code for VO is provided.	http://sites.google.com/site/scarabotix

VO is applied in all kinds of mobile-robotics systems, such as space robots, ground robots, aerial robots, and underwater robots. But probably, the most popular application of VO has been on NASA Mars exploration rovers [25], [79]. NASA's VO has been used since January 2004 to track the motion of the two NASA rovers Spirit and Opportunity as a supplement to dead reckoning. Their stereo VO system was implemented on a 20-MHz central processing unit and took up to three minutes for a two-view structure-from-motion step. VO was mainly used to approach targets efficiently as well as to maintain vehicle safety while driving near obstacles on slopes, achieving

difficult drive approaches, performing slip checks to ensure that the vehicle is still making progress.

VO is also applied onboard of unmanned aerial vehicles of all kinds of sizes, e.g., within the Autonomous Vehicle Aerial Tracking and Reconnaissance [80] and Swarm of Micro Flying Robots (SFLY) [81] projects. Within the SFLY project, VO was used to perform autonomous take-off, point-to-point navigation, and landing of small-scale quadrocopters.

Autonomous underwater vehicle is also a domain where VO plays a big role. Underwater vehicles cannot rely on GPS for position estimation; thus, onboard sensors need to be used. Cameras provide a cost-effective solution; in addition, the ocean floor often provides a texture-rich environment [82], which is ideal for computer vision methods. Applications range from coral-reef inspection (e.g., the Starbug system [82] to archaeological surveys [83].

VO also plays a big role in the automotive industry. Driver assistance systems (e.g., assisted braking) already rely on computer vision and digital cameras. VO for automotive market is in development, and its first demonstrations have been successfully shown, e.g., within the Daimler 6-D-Vision system [84] or as part of the VisLab autonomous vehicle [85]. Driving the development of this technology is the low cost of vision sensors as compared to Lidar sensors, which is an important factor for the automotive industry.

Available Code

Some algorithms that can be used to build a VO system are made publicly available by their authors. Table 2 points the reader to a selection of these resources.

Conclusions

Part II of the tutorial has summarized the remaining building blocks of the VO pipeline: specifically, how to detect and match salient and repeatable features across frames and robust estimation in the presence of outliers and bundle adjustment. In addition, error propagation, applications, and links to publicly available code are included. VO is a well understood and established part of robotics.

VO has reached a maturity that has allowed us to successfully use it for certain classes of applications: space, ground, aerial, and underwater. In the presence of loop closures, VO can be used as a building block for a complete SLAM algorithm to reduce motion drift. Challenges that still remain are to develop and demonstrate large-scale and long-term implementations, such as driving autonomous cars for hundreds of miles. Such systems have recently been demonstrated using Lidar and Radar sensors [86]. However, for VO to be used in such systems, technical issues regarding robustness and, especially, long-term stability have to be resolved. Eventually, VO has the potential to replace Lidar-based systems for egomotion estimation, which are currently leading the state of the art in accuracy, robustness, and reliability. VO offers a cheaper and mechanically easier-to-manufacture solution for egomotion estimation, while, additionally, being fully passive. Furthermore, the ongoing miniaturization of digital cameras offers the possibility to develop smaller and smaller robotic systems capable of egomotion estimation.

Acknowledgments

The authors thank Konstantinos Derpanis for his fruitful comments and suggestions.

References

[1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2004, pp. 652–659.

[2] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford University, Stanford, CA, 1980.

[3] L. Matthies and S. Shafer, "Error modeling in stereo navigation," *IEEE J. Robot. Automat.*, vol. 3, no. 3, pp. 239–248, 1987.

[4] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo, "Rover self localization in planetary-like environments," in *Proc. Int. Symp. Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, 1999, pp. 433–440.

[5] C. Olson, L. Matthies, M. Schoppers, and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, pp. 453–458.

[6] M. Lhuillier, "Automatic structure and motion using a catadioptric camera," in *Proc. IEEE Workshop Omnidirectional Vision*, 2005, pp. 1–8.

[7] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2006, pp. 363–370.

[8] J. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2008, pp. 2531–2538.

[9] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 4293–4299.

[10] W. Forstner, "A feature based correspondence algorithm for image matching," *Int. Archiv. Photogram.*, vol. 26, no. 3, pp. 150–166, 1986.

[11] C. Harris and J. Pike, "3d positional integration from image sequences," in *Proc. Alvey Vision Conf.*, 1987, pp. 233–236.

[12] C. Tomasi and J. Shi, "Good features to track," in *Proc. CVPR*, 1994, pp. 593–600.

[13] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conf. Computer Vision*, 2006, vol. 1, pp. 430–443.

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 20, no. 2, pp. 91–110, 2003.

[15] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. ECCV*, 2006, pp. 404–417.

[16] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extremas for realtime feature detection and matching," in *Proc. European Conf. Computer Vision*, 2008, pp. 102–115.

[17] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, MIT Press, 2011.

[18] A. Schmidt, M. Kraft, and A. Kasinski, "An evaluation of image feature detectors and descriptors for robot navigation," in *Proc. Int. Conf. Computer Vision and Graphics*, 2010, pp. 251–259.

[19] N. Govender, "Evaluation of feature detection algorithms for structure from motion," Council for Scientific and Industrial Research, Pretoria, Technical Report, 2009.

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Englewood Cliffs, NJ, Prentice Hall, 2007.

[21] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proc. European Conf. Computer Vision*, 1994, pp. 151–158.

[22] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. European Conf. Computer Vision*, 2010, pp. 778–792.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf (pdf)," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Barcelona, Nov. 2011, pp. 2564–2571.

[24] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proc. Int. Conf. Computer Vision*, 2011, pp. 2548–2555.

[25] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers: Field reports," *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.

[26] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. Int. Conf. Computer Vision*, 2003, pp. 1403–1410.

[27] T. Lemaire and S. Lacroix, "Vision-based SLAM: Stereo and monocular approaches," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 343–364, 2006.

[28] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. European Conf. Computer Vision*, 2008, pp. 802–815.

[29] H. Strasdat, J. Montiel, and A. Davison, "Real time monocular SLAM: Why filter?" in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2657–2664.

[30] B. Horn and B. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, 1981.

[31] A. Makadia, C. Geyer, and K. Daniilidis, "Correspondence-free structure from motion," *Int. J. Comput. Vis.*, vol. 75, no. 3, pp. 311–327, 2007.

[32] P. Torr and D. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," *Int. J. Comput. Vis.*, vol. 24, no. 3, pp. 271–300, 1997.

[33] K. Sim and R. Hartley, "Recovering camera motion using l_∞ minimization," *IEEE Conf. Computer Vision and Pattern Recognition*, 2006, pp. 1230–1237.

[34] M. A. Fischler and R. C. Bolles, "RANSAC sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[35] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, MA, Cambridge Univ. Press, 2004.

[36] J. Oliensis, "Exact two-image structure from motion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 12, pp. 1618–1633, 2002.

[37] E. Kruppa, "Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung," in *Proc. Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. IIa.*, 1913, vol. 122, pp. 1939–1948.

[38] O. Faugeras and S. Maybank, "Motion from point matches: Multiplicity of solutions," *Int. J. Comput. Vis.*, vol. 4, no. 3, pp. 225–246, 1990.

[39] J. Philip, "A non-iterative algorithm for determining all essential matrices corresponding to five point pairs," *Photogram. Rec.*, vol. 15, no. 88, pp. 589–599, 1996.

[40] B. Triggs, "Routines for relative pose of two calibrated cameras from 5 points," INRIA Rhone-Alpes, Tech. Rep., 2000.

[41] D. Nister, "An efficient solution to the five-point relative pose problem," *Proc. CVPR03*, 2003, pp. II: 195–202.

[42] H. Stewenius, C. Engels, and D. Nister, "Recent developments on direct relative orientation," *ISPRS J. Photogram. Remote Sens.*, vol. 60, no. 4, pp. 284–294, 2006.

[43] O. Pizarro, R. Eustice, and H. Singh, "Relative pose estimation for instrumented, calibrated imaging platforms," in *Proc. DICTA*, 2003, pp. 601–612.

[44] R. Sturm, "Das problem der projektivitaet und seine anwendung auf die flaechen zweiten grades," *Math. Annal.*, vol. 1, no. 4, pp. 533–573, 1869.

[45] C. Geyer and H. Stewenius, "A nine-point algorithm for estimating paracatadioptric fundamental matrices," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'07),* June 2007, pp. 1–8, 17–22.

[46] P. Sturm and J. Barreto, "General imaging geometry for central catadioptric cameras," in *Proc. 10th European Conf. Computer Vision*, Marseille, France, 2008, pp. 609–622.

[47] P. Sturm, S. Ramalingam, J. Tardif, S. Gasparini, and J. Barreto, "Camera models and fundamental concepts used in geometric computer vision," *Foundat. Trends Comput. Graph. Vis.*, vol. 6, no. 1–2, pp. 1–183, 2010.

[48] J. Lim, N. Barnes, and H. Li, "Estimating relative camera motion from the antipodal-epipolar constraint," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, no. 10, pp. 1907–1914, 2010.

[49] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, "A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles," in *Proc. European Conf. Computer Vision*, 2010, pp. 269–282.

[50] O. Naroditsky, X. S. Zhou, J. Gallier, S. I. Roumeliotis, and K. Daniilidis, "Two efficient solutions for visual odometry using directional correspondence," *IEEE Trans. Pattern Anal. Machine Intell.*, no. 99, p. 1, 2011.

[51] L. Kneip, M. Chli, and R. Siegwart, "Robust real-time visual odometry with a single camera and an imu," in *Proc. British Machine Vision Conf.*, 2011.

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

[53] D. Scaramuzza, "1-point-RANSAC structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *Int. J. Comput. Vis.*, vol. 95, no. 1, pp. 74–85, 2011.

[54] D. Scaramuzza, "Performance evaluation of 1-point ransac visual odometry," *J. Field Robot.*, vol. 28, no. 5, pp. 792–811, 2011.

[55] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 138–156, 2000.

[56] O. Chum and J. Matas, "Matching with prosac—Progressive sample consensus," in *Proc. CVPR*, 2005, pp. 220–226.

[57] D. Nister, "Preemptive ransac for live structure and motion estimation," *Machine Vis. Applicat.*, vol. 16, no. 5, pp. 321–329, 2005.

[58] R. Raguram, J. Frahm, and M. Pollefeys, "Exploiting uncertainty in random sample consensus," in *Proc. ICCV*, 2009, pp. 2074–2081.

[59] P. McIlroy, E. Rosten, S. Taylor, and T. Drummond, "Deterministic sample consensus with multiple match hypotheses," in *Proc. British Machine Vision Conf.*, 2010, pp. 1–11.

[60] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for ekf filtering: Application to real-time structure from motion and visual odometry," *J. Field Robot.*, vol. 27, no. 5, pp. 609–631, 2010.

[61] D. Scaramuzza, A. Censi, and K. Daniilidis, "Exploiting motion priors in visual odometry for vehicle-mounted cameras with non-holonomic constraints," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2011, pp. 4469–4476.

[62] E. Rosten, G. Reitmayr, and T. Drummond, "Improved ransac performance using simple, iterative minimal-set solvers," University of Cambridge, Tech. Rep., arXiv:1007.1432v1, 2010.

[63] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Proc. DAGM-Symp.*, 2003, pp. 236–243.

[64] R. C. Smith, P. Cheeseman. (1986). On the representation, and estimation of spatial uncertainty, *Int. J. Robot. Res.*, [Online], vol. 5, no. 4, pp. 56–68. Available: http://ijr.sagepub.com/content/5/4/56.abstract

[65] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. ICRA*, 2006, pp. 2262–2269.

[66] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II. State of the art," *IEEE Robot. Automat. Mag.*, vol. 13, no. 3, pp. 108–117, 2006.

[67] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 2000, pp. 1023–1029.

[68] M. Jogan and A. Leonardis, "Robust localization using panoramic view-based recognition," in *Proc. ICPR*, 2000, vol. 4, pp. 136–139.

[69] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1–2, pp. 43–72, 2005.

[70] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006, pp. 1180–1187.

[71] M. Cummins and P. Newman, (2008). "FAB-MAP: Probabilistic localization mapping in the space of appearance," *Int. J. Robot. Res.* [Online], vol. 27, no. 6, pp. 647–665. Available: http://ijr.sagepub.com/cgi/content/abstract/27/6/647

[72] F. Fraundorfer, C. Engels, and D. Nistér, "Topological mapping, localization and navigation using image collections," in *Proc. IEEE/RSJ Conf. Intelligent Robots and Systems*, 2007, pp. 3872–3877.

[73] F. Fraundorfer, C. Wu, J.-M. Frahm, and M. Pollefeys, "Visual word based location recognition in 3d models using distance augmented weighting," in *Proc. 4th Int. Symp. 3D Data Processing, Visualization, and Transmission*, 2008, pp. 1–8.

[74] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, New York, Wiley, 2001.

[75] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, New York, 2006, pp. 2161–2168.

[76] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment a modern synthesis," in *Proc. Int. Workshop Vision Algorithms: Theory and Practice (ICCV'99)*, 2000, pp. 298–372.

[77] G. Sibley, L. Matthies, and G. Sukhatme. (2010). Sliding window filter with application to planetary landing. *J. Field Robot.*, [Online], vol. 27, no. 5, pp. 587–608. Available: http://dx.doi.org/10.1002/rob.20360

[78] Dacuda AG. (2011). Dacuda scanner mouse. [Online]. Available: http://www.dacuda.com/

[79] Y. Cheng, M. W. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," *IEEE Robot. Automat. Mag.*, vol. 13, no. 2, pp. 54–62, 2006.

[80] J. Kelly and G. S. Sukhatme. (2007. Sept.). An experimental study of aerial stereo visual odometry. In *Proc. IFAC –Int. Federation of Automatic Control Symp. Intelligent Autonomous Vehicles*, Toulouse, France. [Online]. Available: http://cres.usc.edu/cgi-bin/print.pub.details.pl?pubid=543

[81] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments," *J. Field Robot.*, vol. 28, no. 6, pp. 854–874, 2011.

[82] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke, "A hybrid AUV design for shallow water reef navigation," in *Proc. IEEE Int. Conf. Robotics and Automation, ICRA*, Apr. 2005, pp. 2105–2110.

[83] B. P. Foley, K. DellaPorta, D. Sakellariou, B. S. Bingham, R. Camilli, R. M. Eustice, D. Evagelistis, V. L. Ferrini, K. Katsaros, D. Kourkoumelis, A. Mallios, P. Micha, D. A. Mindell, C. Roman, H. Singh, D. S. Switzer, and T. Theodoulou, "The 2005 chios ancient shipwreck survey: New methods for underwater archaeology," *Hesperia*, vol. 78, no. 2, pp. 269–305, 2009.

[84] A. G. Daimler. (2011). 6d vision. [Online]. Available: http://www.6d-vision.com/

[85] M. Bertozzi, A. Broggi, E. Cardarelli, R. Fedriga, L. Mazzei, and P. Porta, "Viac expedition toward autonomous mobility [from the field]," *IEEE Robot. Automat. Mag.*, vol. 18, no. 3, pp. 120–124, Sept. 2011.

[86] E. Guizzo. (2011). How Google's self-driving car works. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/howfl-google-self-driving-car-works

[87] D. Scaramuzza and F. Fraundorfer, "Visual odometry," *IEEE Robotics Automat. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[88] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Stereo ego-motion improvements for robust rover navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2001)*, 2001, vol. 2, pp. 1099–1104.

[89] M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett, "Honeybee navigation en route to the goal: Visual flight control and odometry," *J. Exp. Biol.*, vol. 199, 1996, pp. 237–244.

[90] P. Viola and M. Jones, "Robust real time object detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2001.

[91] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, no. 1–3, pp. 7–42, Apr.–June 2002.

Friedrich Fraundorfer, Institute of Visual Computing, Department of Computer Science, ETH Zurich, Switzerland. E-mail: fraundorfer@inf.ethz.ch.

Davide Scaramuzza, Robotics and Perception Lab, Department of Informatics, University of Zurich, Switzerland. E-mail: davide.scaramuzza@ieee.org.