In Proceedings of The 12th International Conference on Intelligent Autonomous Systems (IAS-12), Jeju Island, Korea, June 2012.

Fast 6D Odometry Based on Visual Features and Depth

Salvador Domínguez[1], Eduardo Zalama[2], Jaime Gómez García-Bermejo[2], Rainer Worst[3], and Sven Behnke[3,4]

[1] Cartif Foundation, Valladolid, Spain
[2] ITAP, University of Valladolid, Spain
[3] Fraunhofer IAIS, Sankt Augustin, Germany
[4] University of Bonn, Germany

Abstract. The availability of affordable RGB-D cameras which provide color and depth data at high data rates, such as Microsoft MS Kinect, poses a challenge to the limited resources of the computers onboard autonomous robots. Estimating the sensor trajectory, for example, is a key ingredient for robot localization and SLAM (Simultaneous Localization And Mapping), but current computers can hardly handle the stream of measurements. In this paper, we propose an efficient and reliable method to estimate the 6D movement of an RGB-D camera (3 linear translations and 3 rotation angles) of a moving RGB-D camera. Our approach is based on visual features that are mapped to the three Cartesian coordinates (3D) using measured depth. The features of consecutive frames are associated in 3D and the sensor pose increments are obtained by solving the resulting linear least square minimization system. The main contribution of our approach is the definition of a filter setup that produces the most reliable features that allows for keeping track of the sensor pose with a limited number of feature points. We systematically evaluate our approach using ground truth from an external measurement systems.

1 Introduction

Optimizing the use of the available resources in mobile robotics like energy or processing capabilities is important, because some robots do not have any external physical connection and have to work autonomously. One of the most important tasks a mobile robot must perform is self-localization with respect to a map. This task is, however, also one of the most resource-consuming when no global positioning system is used. When the robot pose is known, it can be used for navigation as well as other purposes such as mapping the environment. The measurements needed for localization are usually provided by expensive and heavy laser range sensors. The availability of affordable lightweight RGB-D cameras offers the possibility to acquire color images and depth maps at high frame rates which must be processed in real time in order to be used for robot control.

Finalist for Best Paper Award

In this paper, we propose a reliable method to obtain 6D odometry from color and depth images. Our method can be applied to estimate the sensor pose in real time, requiring limited computational resources.

We use a Kinect RGBD sensor as the only capturing device. The sensor driver provides color and depth images in a standard format. At each frame, we extract a set of visual features from the RGB image. They are filtered and mapped to 3D space using the associated depth image. We associate the features of consecutive frames in 3D and finally obtain the sensor pose increment. Key to our method is a carefully designed filtering process, which takes the most reliable features to estimate odometry. Finally, we calculate the transformation that minimizes the mean square error between two sets of associated features. Because our method processes only a limited number of features, it can be easily applied for pose estimation on mobile robots with limited computational resources, like e.g. a 1.6 GHz CPU with 1 GB RAM that we used for our experiments. Even with such a small computer, we achieve odometry rates near 20 Hz in low resolution, which makes it possible to use our method on a robot that moves at full speed (1 m/sec linear velocity and / or 45°/sec angular velocity in our case). Our method does not rely on any assumption about the kinematics of the movement. The odometry obtained can be used later as an input data stream for 6D-SLAM.

Raw detected features are intrinsically noisy and would produce low-quality odometry estimates if they were applied directly. The filtering reduces the number of features and consequently the time consumption of the overall process significantly. Instead of trying to match the features extracted from consecutive frames, we first stabilize the consistency of the current feature set with respect to a reference feature set. As explained below, such stabilization is performed by dynamically adapting the current set of features to a reference set.

The remainder of the paper is structured as follows. In Section II, we discuss related work. In Section III, we present our approach to the estimation of 6D odometry based on visual features. Finally, we demonstrate the advantages of our method using systematic experiments in Section IV.

2 Related Work

Visual odometry is the process of calculating the increments of pose based on the analysis of images captured by a camera. One of the first works in visual odometry is attributed to Moravec [1]. In 1976 he managed to implement a visual odometry system on the Stanford Cart to perform corrections on the control of the vehicle autonomously. In that technical report, the basic concepts of tracking visual features and depth estimation were introduced.

We can identify two main types of visual odometry using cameras as the only capturing device: monocular and stereo. In monocular odometry depth information must be estimated from motion parallax as the camera moves through the environment. Interesting examples of monocular odometry implementation include Nister et al. [2] and Davison [3]. In stereo odometry, the camera does not have to move to calculate depth information. Some implementations of stereo

odometry are described by Moravec [1], Matties [4] and Nister et al. [2]. On the other hand, it is also possible to use additional devices for the acquisition of the depth like laser range scanners, time-of-flight cameras or infrared pattern projectors. The resulting input data stream for visual odometry estimation is in most cases a sequence of images with depth information

The tracking of the features plays an important role in visual odometry, because it is a process that consumes less time than extracting the features in every frame, and can be applied to several consecutive frames while as long as enough features are within the image boundaries. Barron [5] compiled a survey of optical flow algorithms for tracking, like the gradient-based technique proposed by Lucas and Kanade [6] (LK). LK is implemented in OpenCV and used in our approach. The selection of good features to track is also important in order to reduce resource consumption and increase accuracy. An interesting attention-driven feature selection is described by Einhorn [7].

3 Method

Figure 1 summarizes our approach for the estimatation of the odometry. No prior knowledge about the environment is assumed. We can distinguish two main loops, the extraction loop and the tracking loop. The extraction loop is only executed when a new set of features is needed (generated features GF), i.e. when there are not enough features to obtain a good odometry estimate. The tracking loop is executed in every frame and performs, among other things, tracking, filtering, and matching of the features to get the transformation. The key steps that make this approach suitable for real time application and less sensitive to noise are the filtering of the features and the consistency adaptation (marked with stars in Figure 1) which will be explained in detail below.

3.1 Visual features extraction.

Visual features are interesting parts of the image with properties that make them suitable for detection. Different steps to detect robust, distinctive invariant features are explained in Lowe [8]. There are different kinds of such features, e.g. corners and blobs. Because their extraction is computationally less demanding, we focus in this paper on the so called corners. Although corner features are not completely robust and stable, they can be stabilized by the 3D information, which in our case is available.

Corners are points which have two dominant directions in the luminosity gradient, in a local neighborhood. Some of the most important techniques for extracting corners are presented by Harris and Stephens [9] as well as Shi and Tomasi [10].

An evaluation of three features detectors (Harris, SIFT (Scale Invariant Feature Transform) and LKT (Lucas-Kanade-Tomasi)) is described by Klippenstein and Zhang [11]. They found LKT to have a better performance in general. We test our approach with two feature extractors: Harris corner detector and LKT

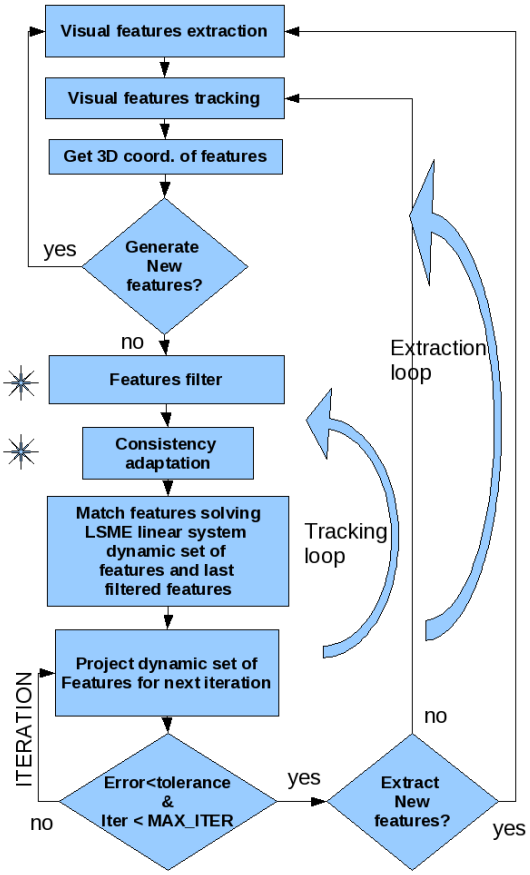


Fig. 1. Flow diagram of the overall process of odometry estimation

corner detector which are both included in the OpenCV library getting lightly better results with LKT.

3.2 Tracking of visual features.

The tracking of features is a crucial step of visual odometry. Shi and Tomasi [10] propose a matching algorithm for tracking visual features based on a model of affine image changes. This technique has been adopted in our approach. For the tracking process, a pair of images is used: the most recently captured image and the previous one (Figure 2). The visual features corresponding to the previous image can be obtained either by feature extraction or as a result of the previous tracking iteration. We apply Shi-Tomasi's method to track the features in the first image, so we know where to expect them in the second image. If we cannot

establish correspondence for a feature in the second image, it is marked as "non-paired" feature.

Fig. 2. Features in one frame (previous) are tracked during the tracking process to the next frame (tracked features TF). Some of the correspondences are shown in magenta.

3.3 Generation of new features.

Because we are looking for a disambiguous solution to the linear error minimization system (see paragraph III-E), a minimum of 4 features are needed so the number of equations is larger or equal to the number of unknowns. When no such solution can be found we seek to generate a new set. There are several reasons for this to happen:

- the scene has no interest points to be easily distinguished like flat colored surfaces with few corners.
- when moving the sensor, the features will likely change their positions in the image and some of them can disappear on the image borders.
- the distribution of the features can lead to clustering in some parts of the image, for example when the camera moves backwards and distance between tracked features reduces and some of the features fuse together.

Moreover, the grouping of features decreases the precision of the movement estimate. For better performance, the visual features should be distributed roughly uniformly in the image. In order to deal with this problem, we measure the distance traveled and the rotated angles of the sensor from the last feature extraction. When either the distance or the angle, exceeds a given threshold, we trigger the generation of a new set of features.

3.4 Projection of visual features in 3D.

RGB-D sensors like MS Kinect provide the depth information at most pixels in the color image. However, sometimes the physical properties of the surface at some positions are not suitable for the measurement of depth. Many of the visual features lie on object boundaries where depth information is often missing. Moreover, visual features might jump from the front edge of an object to the background side.

If the entire neighborhood of the feature has similar depth values, then the depth of the feature point can be obtained by averaging these values. It could happen that the feature does not have depth information or that there is a large difference among the neighbors depths, probably because the feature is lying at the jump edge caused by a border. In this case, we utilize border ownership, which means that we average the depth of the closest neighboring points.

Averaging has also the effect of reducing the noise of the depth measurements. It can happen that no points in the neighborhood have depth information and it will not be possible to calculate the 3D depth of the feature. Those features will be discarded in the filtering process.

Preferring closer points is also advantageous for the measurement precision of triangulation sensors like Kinect because the error increase quadratically with the distance. The optimal radius of the neighborhood depends on the image resolution. In our experiments with a 640x480 resolution, a radius of 3 pixels has been proven to work suitably. Because our neighborhood is defined as a square, this implies a neighborhood of 49 pixels.

3.5 Filtering and consistency adaptation

In order to retain only the most reliable features for pose estimation, we designed a set of rules for discarding features. Firstly, we filter out the features with missing depth, then the consistency of the distribution is checked. We analyze the relative 3D positions of the last tracked features TF compared to the respective positions of the generated features GF, as is shown in Figure 3.

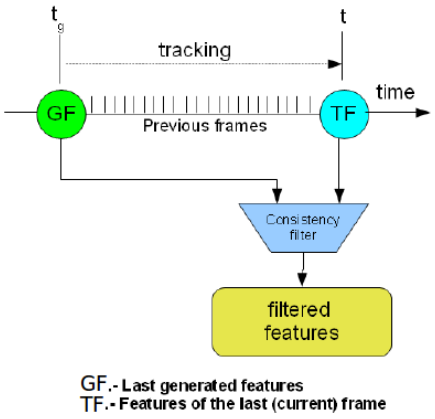


Fig. 3. Filtering process. The last extracted features (GF) and the last tracked features (TF) are used to filter the features.

This filter is based on the idea that every feature has to correspond to a fixed point in the real world. This way, the position of the TF in global coordinates

must not differ too much from the global position of GF during the tracking process. In our approach, we model the set of features as a cloud of 3D points where each point is connected to the rest by elastic links.

To move a point from its equilibrium point in such elastic net, it is necessary to apply a force (see Figure 4). If we consider the GF as the balanced state where the deformation energy is zero, then the TF has an energy accumulated because the relative position of the features is not exactly the same due to non-perfect tracking. The force required at every node to move it from its equilibrium point to its current position, can be calculated through a vectorial sum of the forces in every link connecting that node to the rest. Every link can only apply a force in its longitudinal direction and can be calculated because we know the 3D coordinates of every feature in both sets of features. The condition to pass the filter is that the displacement force must stay below a threshold value.

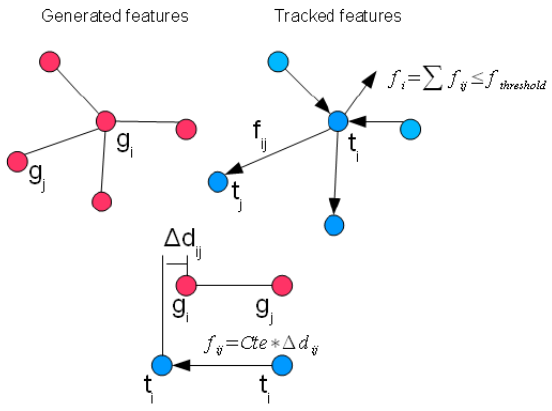


Fig. 4. The force in each feature is determined as a vectorial sum of the deformation forces in each link. The force in every link (i,j) is proportional to the difference of distance between that link in TF and GF.

The consistency filter discards features that moved more than a threshold distance from their equilibrium position. However, probably most of the rest of the features are still noisy in depth because the sensor accuracy decreases on the square of the distance to the point in the real world. Noisy depth values lead to decreased odometry accuracy, especially at distances above 2 meters using the MS Kinect sensor. To solve this problem we apply a filter to adapt and stabilize the consistency of the TF. This is what we call consistency adaptation in Figure 1. This filter works on so-called adaptive features TF^*. This new set of features is the result of the gradual consistency adaptation from set TF to GF. It can be seen as a stable version of the TF that we use for calculating the odometry transformation.

Figure 5 represents the process of consistency adaptation of the features. TF^* have the properties of having consistency similar to GF and a global pose similar to TF. Little changes in the relative average positions of the tracked features in 3D will be adapted progressively to the equilibrium position. TF^* and GF are finally used to calculate the transformation of the odometry leading to stable odometry.

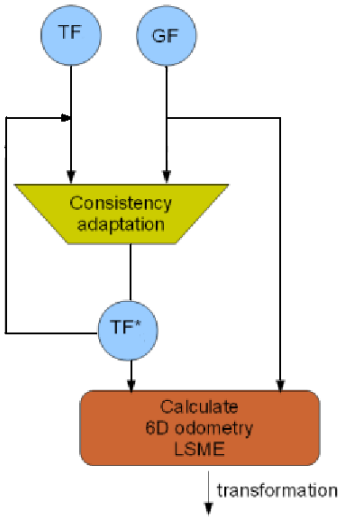


Fig. 5. Calculation and use of the adaptive features.

The adaptation of the consistency has positive effects over the noise in the estimated position. Figure 6 shows the difference without and with adaptation, respectively. The ground truth in this experiment was composed by perpendicular straight lines following X, Y and Z axes. When no adaptation is applied, the noise increases with the distance to the features, on the contrary with adaptation the noise is less dependent on that distance.

3.6 Estimation of pose increments

Let us consider that we have tracked a set of N visual features p from the previous frame to the last one producing the set q whose features are paired one by one with those in p. These features have passed the filtering process and consistency adaptation so they are stable.

We are looking for a rigid transformation $(\Re, t)$ that minimizes the value of the following expression

Without adaptation With adaptation

Fig. 6. Effect in the adaptation of the consistency. The distance from the sensor to the features increases in the direction of the arrow. Notice that the dispersion of the position increases with the distance to the sensor when no adaptative consistency is used

$$J(\Re, t) = \sum_{N} \|p_i - \Re q_i - t\|^2 \tag{1}$$

where p_i and q_i represent the 3D position of the paired feature i in the previous frame and the current frame, respectively.

This is a typical problem of Linear Least Square Minimization. The linear system is represented by 3×N equations (N features × 3 coordinates) with 12 unknowns (9 for the rotation matrix $\Re$ and 3 for the translation vector t). Therefore, to have a unique solution we need at least 4 features.

Equation (1) can be represented in a different way, being equivalent to minimize the following expression

$$\begin{aligned} &\sum_{N} \|p_i - A(q_i) * \wp\|^2 \rightarrow p_i \approx A(q_i) * \wp \\ &A(q_i)^T * p_i = A(q_i)^T * A(q_i) * \wp \\ &\wp = (A(q_i)^T * A(q_i))^{-1} * A(q_i)^T * p_i = R(q_i) * p_i \end{aligned} \tag{2}$$

In this expression, $\wp$ is the 12×1 vector of unknowns which is composed by elements of the rigid transformation $(\Re, t)$ and $R(q_i)$ is a 12×3N matrix which depends on the coordinates of the features in the last frame q_i, and can be determined by classical methods such as QR or Cholesky factorization [12].

The error of the transformation is checked by projecting every feature in the last frame to the previous frame using the transformation $(\Re, t)$. If the error for the feature i ϵ_i is less or equal to a threshold δ the feature will be used for the next iteration (3).

$$\epsilon_i = \|p_i - \Re q_i - t\| \leq \delta \tag{3}$$

The process of pose estimation stops when

$$\sum_{N} \|p_i - \Re q_i - t\| \leq threshold \tag{4}$$

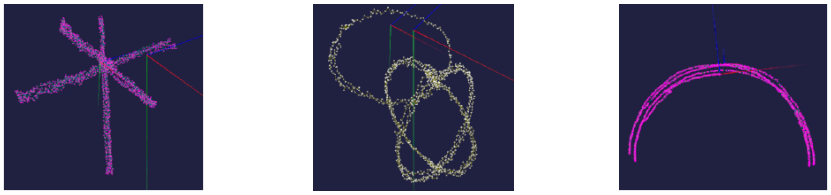
4 Experiments and Results

To evaluate the quality of our method, we have used datasets with a known ground truth. The datasets were recorded at Gelsenkirchen University of Applied Sciences using an industrial Kuka KR60HA robot (Figure 7). One example of the scene seen by the sensor is shown in Figure 8.

Fig. 7. Robot used for ground truth experiments

Fig. 8. Scene seen by the camera for experiments 1 and 2

The sensor trajectories used were geometric paths such as lines and arcs (see Figure 9) traced at about 15 cm/sec. The number of visual features always was bigger than 10 and the distance to the features covered a range between 2 and 5 m. Every test starts and finishes in the same position so that the difference between the initial position and the final one will measure the total error. We also pay attention to the maximum distance of the estimated position to the theoretical trajectory.



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3

Fig. 9. Trajectories obtained on the experiments.

Experiment 1: Linear translation along X, Y, and Z directions. The traveled distance along each direction was 1 m, and was repeated three times (see Figure 9a). This way, the total traveled length was 18 m. The final error found in the position was 11 cm in translation and 1.5° in inclination, which correspond to a 0.6% accumulated position error. The maximum deviation from the theoretical trajectory was only 6 cm.

Experiment 2: Circles traced on the XY, XZ, and YZ planes (see Figure 9b). Each circle was 50 cm in diameter. Every circle was repeated three times. The total length of the trajectory was 14.13 m and the final error was 4.5 cm and 1.2° in inclination, that is less than 0.3 % of the total length of the trajectory. The maximum deviation from the theoretical curve was 7 cm.

Experiment 3: Translation and rotation along an arc of 180° and 1 m radius, with the camera pointing along the radial direction (see Figure 9c). Each movement was repeated three times, so the total traveled distance was 18.85 m and the total angle rotated was 1080°. The final pose of the camera had an error of 36 cm in position and 10° in orientation, which corresponds to a 1.9 % linear error and 0.9 % rotation error.

5 Conclusion

We have presented a method for estimating the trajectory of a moving RGB-D sensor using 6D odometry based on visual features obtained from the RGB images, which are mapped to 3D using their measured depth. Key to the success of our method is a filtering process that removes the features that are not reliable, thus making our solution robust. The consistency adaptation of the features improves the stability and precision of the result. Our algorithm is computationally very efficient, because only a limited number of points is processed every frame.

We evaluated our method by moving the camera with an industrial arm, which provides ground truth for the trajectories. The obtained results are satisfactory for visual odometry considering the limitations of the sensor with regards to depth noise.

Our 6D visual odometry is in use as one of the key parts of the 3D SLAM system for the ground robot used in NIFTi project which is equipped with a small computer with a processor of 1.6 GHz. Working at 640×480 and 15-20 fps the CPU usage stays near 35 % on that computer.

6 Acknowledgments

We would like to thank to both Cartif Foundation and Fraunhofer IAIS for having made this work possible. This work has been funded by the European Commission under contract number EU FP7 NIFTi / ICT-247870 and also supported by Junta de Castilla y León (CCTT/10/VA/0001) and by the Science and Innovation Spanish ministry (Pr.Nb.DPI2008-06738-C02-01/DPI). Thanks also to the NIFTi team of IAIS, especially to Thorsten Linder, Viatcheslav Tretyakov and Nenad Biresev for their support in the experiments and helpful suggestions.

References

1. H Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Tech. report and doctoral dissertation, Robotics Institute, Carnegie Mellon University, Stanford University, 1980. CMU-RI-TR-80-03

2. D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. Journal of Field Robotics, 23(1), 2006
3. A. J. Davison. Real-time simultaneous localization and mapping with a single camera. In Proceedings of the International Conference on Computer Vision, Nice, 2003
4. L. Matties. Dynamic Stereo Vision. PhD thesis, Dept. of Computer Science, Carnegie Mellon University, 1989. CMU-CS-89-195
5. J. L. Barron, D. J. Fleet and S.S. Beauchemin. Performance of Optical Flow Techniques. The International Journal of Computer Vision 12(1):43-77. 1994.
6. B.D. Lucas, and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proceedings of 7th International Joint Conference on Artificial Intelligence, 674-679. 1981.
7. E. Einhorn, Ch. Schrter, H.-M. Gross-M. Can't Take my Eye on You: Attention-Driven Monocular Obstacle Detection and 3D Mapping. IROS 2010
8. D. G. Lowe. Distinctive image features from scale invariant keypoints. International Journal of Computer Vision, 60(2):91110, 2004.
9. C. Harris and M. Stephens. A Combined Corner and Edge Detector. Proc. of The Fourth Alvey Vision Conference, Manchester, pp. 147-151. 1988.
10. J. Shi and C. Tomasi. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition, Seattle, 1993
11. J. Klippenstein and H. Zhang. Quantitative Evaluation of Feature Extractors for Visual SLAM. J Department of Computing Science University of Alberta Edmonton, AB, Canada T6G 2E8. 2007
12. G. H. Golub and, Ch. F. Van Loan. Matrix Computations (3rd ed.), Johns Hopkins, ISBN 978-0-8018-5414-9, 1996