ETH Zürich

Vision Algorithms for Mobile Robotics

Visual Odometry Mini-Project

Authors:	*Student Number:*
Patrick Pfreundschuh	13-940-903
Tim Franzmeyer	14-943-427

January 6, 2019

Abstract

In the following mini-project, a monocular visual odometry (VO) pipeline is implemented and tested on different datasets. The VO pipeline consists of the four blocks that were also suggested in the problem statement: 1.Bootstrapping of initial pose estimate and landmarks, 2. continuous key point tracking between subsequent frames, 3. continuous pose estimation using 2D keypoint $\leftrightarrow$ 3D landmark correspondences and 4. continuously adding new landmarks.

The bootstrapping of initial keypoints and landmarks is achieved using Harris corner feature correspondences from different frames. For continuous tracking of keypoints, the Kanade-Lucas-Tomasi Tracker (KLT) is used in subsequent frames. Continuous pose estimation is then achieved with the p3p algorithm using the corresponding landmark and keypoint pairs in subsequent frames. Since keypoints and thereby landmarks, continuously fall out of the range of view of the camera or can not be tracked correctly, and thus can not be used for the pose estimation anymore, new landmarks are triangulated continuously if the fulfill different criteria.

The proposed VO pipeline was implemented in MATLAB 2018b. With differently tuned parameters, the pipeline was successfully run on the three proposed problem datasets (KITTI 00, Malaga 07, parking garage). The pipeline videos were recorded with an INTEL(R) CORE I7 quad-core 3.2 GHz processor.

Contents

1 Introduction **3**

2 Result Videos **3**

3 Visual Odometry Pipeline **3**

3.1 Bootstrap 3
3.2 Continuous Operation 4

4 Pipeline Modules **4**

4.1 Feature detection 4
4.2 Keypoint tracking 4
4.3 Fundamental matrix estimation 4
4.4 Essential matrix calculation 5
4.5 Disambiguate relative poses 5
4.6 p3p algorithm 5
4.7 Landmark generation 5
4.7.1 Keypoint candidates 5
4.7.2 Triangulation 6
4.7.3 Feature distribution 6
4.8 Re-bootstrap 6

5 Results and Discussion **7**

5.1 Parking 7
5.2 KITTI 7
5.3 Malaga 8
5.4 Tuning parameters 9

6 Key insights **10**

7 Conclusion **10**

7.1 Future work 10

1 Introduction

A Matlab implementation of a visual odometry pipeline is proposed and explained in the following report. This VO pipeline was built in the context of the Visual Odometry mini-project of the ETHZ class Vision Algorithms for Mobile Robotics and builds on methods and algorithms that were covered in this class.

The task of the mini-project was to implement a VO pipeline to continuously estimate the camera pose, given only the frames of single monocular camera videos. The videos that should be covered are the following well-known datasets: KITTI 00, Malaga 07, parking garage.

2 Result Videos

The result videos can be found in the following Youtube Playlist:

https://www.youtube.com/playlist?list=PLtgbkSCeR57u5hr-EI-8XFBsm3c4lRoy

3 Visual Odometry Pipeline

3.1 Bootstrap

Since the p3p algorithm(sec. 4.6) used for continuous pose estimation requires 2D keypoint - 3D landmark correspondences, an initial set of landmarks and keypoints has to be generated. Since the given datasets are already undistorted, a preprocessing of the images was not necessary. Harris Corners and feature matching are used to find keypoint correspondences between a frame and the third following frame. Since these correspondences may contain outliers, RANSAC is used to separate inliers and outliers. Using the inliers, the fundamental matrix F is calculated (sec. (sec. 4.3) and since the K matrices of each camera are known, the essential matrix E can be calculated (sec. 4.4). E can then be decomposed to find the translation T and rotation R of the camera in the second bootstrap frame in relation to the first one . 3D landmarks can now be triangulated, using R,T and the 2D inlier keypoint correspondences (sec. 4.5).

3.2 Continuous Operation

To estimate the pose in every frame, keypoints from the previous frame are tracked in the new frame using KLT feature tracking. Given the pixel coordinates of these features and their corresponding 3D landmarks, the p3p algorithm is used together with MSAC to find outliers and the pose estimate simultaneously. Since keypoints also continuously get lost as the camera moves along (since they fall out of the field of view or since KLT can not track them), new features and landmarks have to be added continuously. To do so, keypoint candidates are continuously added. If a keypoint candidate fulfills certain criteria, it is added as a keypoint together with its triangulated landmark.

4 Pipeline Modules

4.1 Feature detection

Keypoints are detected using Harris Corner Detection (HCD). HCD is applied on every frame.

The MATLAB functions *detectHarrisFeatures* and *extractFeatures* was used to implement this functionality.

4.2 Keypoint tracking

Keypoints are tracked using the Kanade-Lucas-Tomasi feature tracker. Keypoints that could not be tracked reliably are deleted from the keypoint set, together with their corresponding landmark.

The MATLAB *vision.PointTracker System object* was used to implement this functionality.

4.3 Fundamental matrix estimation

The fundamental matrix F is calculated with the normalized 8-point algorithm together with RANSAC. RANSAC searches for the set of keypoints (inliers) that results in a Fundamental Matrix that has the largest set of points that fit to the fundamental matrix. Outliers are removed from the keypoint set.

The MATLAB function *estimateFundamentalMatrix* was used to implement this functionality.

4.4 Essential matrix calculation

The essential matrix E can be calculated from the fundamental matrix F, since the intrinsic matrix K is given for all cameras: $E = K^T * F * K$

The MATLAB function *relativeCameraPose* was used to implement this functionality.

4.5 Disambiguate relative poses

Four different R, t configurations can be calculated from the Essential Matrix E. Since all landmarks should lie in front of both cameras (due to the field of view), the configuration that has the most landmarks in front of both cameras is chosen.

The MATLAB function *relativeCameraPose* was used to implement this functionality.

4.6 p3p algorithm

The P3p algorithm together with MSAC is used to robustly calculate the camera pose based on the 2D keypoint - 3D landmark correspondences. MSAC gives the set of landmarks, that results in a pose that has the largest set of points with a reproduction error below a certain threshold. Outlier keypoint-landmark correspondences are removed from the keypoint-landmark set.

The MATLAB function *estimateWorldCameraPose* was used to implement this functionality.

4.7 Landmark generation

4.7.1 Keypoint candidates

To add new landmarks, features are detected using Harris corner detection in every frame and saved in a set of keypoint candidates. These keypoint candidates, are tracked from frame to frame using the KLT tracker. A non-maximum suppression radius is used, to remove feature candidates, that are close to already existing candidates or existing keypoints.

4.7.2 Triangulation

Using the 2D-2D keypoint correspondences of the feature candidates, their corresponding landmarks are triangulated in every frame. If the landmarks fullfil the following criteria, they are added as a landmark:

1. Reprojection error is below a tuned threshold
2. The bearing angle as proposed in the problem statement is beyond a tuned threshold

The MATLAB function *triangulate* was used to implement this functionality.

4.7.3 Feature distribution

Too many features may be tracked in a specific part of the picture, as certain objects in the picture may contain more distinctive features than other ones. However, this imbalance can lead to inaccurate pose estimates. To account for this and equally distribute both feature- and candidate points, we introduced a grid that is overlayed. This grid consists of 5 blocks in the x-direction, and 2 blocks in the y-direction, hence 10 in total. If new points (keypoints or candidate points) are added, the algorithm computes the optimal number of points, that each square should contain to maintain an equal distribution, depending on the total number of points that shall be added. Then, for every square, the algorithm enforces that the maximum number does not exceed 1.5 times the optimal number. The factor 1.5 ensures, that still sufficient keypoints are added, if some squares do not contain any.

4.8 Re-bootstrap

Re-bootstrapping is done, if the pipeline loses to much features. If the number of landmarks falls below a threshold, the same bootstrapping as in the beginning is done. This means, that all former points are discarded. The pose is concatenated such that it is consistent with the last pose. However, since the baseline in the re-bootstrap is different than the initial baseline, the scale is different, which means that the motion does not have the same speed as before the re-bootstrap. Thus, after re-bootstrapping, the trajectory is only locally consistent, but not globally.

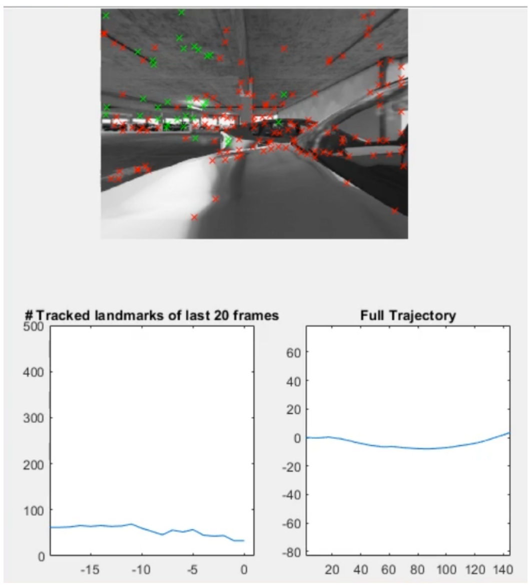
5 Results and Discussion

5.1 Parking

Difficulties: In the parking dataset, there are both really close objects like cars in the first row and remote objects, like cars in the background. Furthermore, features in the background often get out of the field of view of the camera, since they are blocked from cars in the background. To deal with this problems, the grid takes care that features are tracked in each section of the image, and thus the pipeline thus neither only depend on features in the foreground, nor in the background.

Achievements: As shown in the result video, the pipeline can locally consistent follow the ground truth trajectory, with a small drift in Z direction. The trajectory is also smooth, which corresponds to the motion of a car.

Figure 1: Parking Trajectory

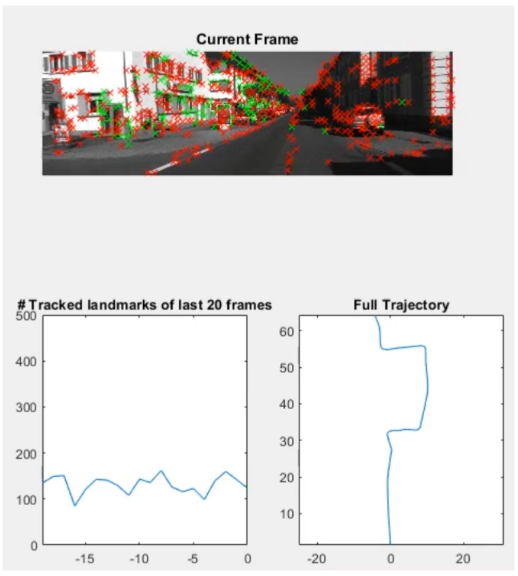


5.2 KITTI

Difficulties: The KITTI dataset offers several challenges for the pipeline. Due to the sunlight, some frames have a high dynamic range and fast changes from bright areas to dark areas. Furthermore, there are sharp turns in the dataset, which makes it harder to track the trajectory, since landmarks fall out of the range of view quickly. There are also moving objects in the dataset, which can confuse the pipeline.

Achievements: As shown in the result video, the pipeline can track the motion locally consistent. Tracking of the trajectory through sharp turns works reliably and the pipeline can also handle bright and dark regions. The first four turns can be tracked globally consistent with a relative low scale drift. However, as the camera moves along, the scale starts to drift, which leads to a high loss of features. Eventually, the pipeline has to re-bootstrap. After the bootstrap, the trajectoy is locally consistent. However, due to the different scale after the re-bootstrap, the trajectory is not globally consistent anymore.

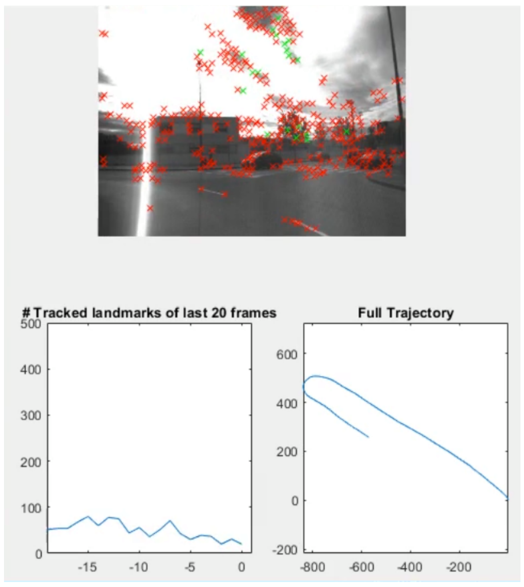
Figure 2: KITTI Trajectory



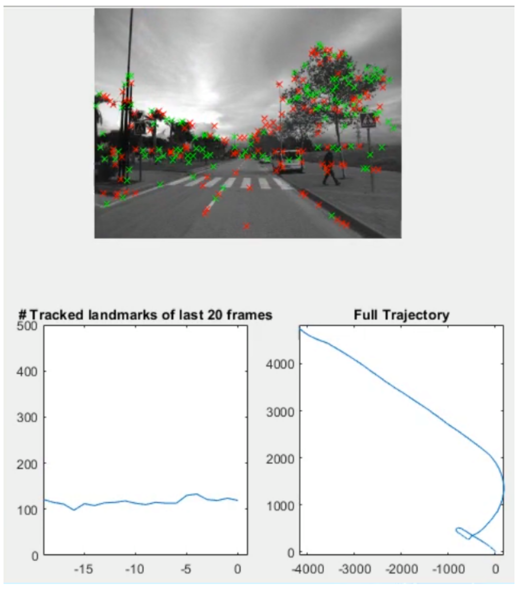
5.3 Malaga

Difficulties: The Malaga dataset basically consists of 2 types of sections: Long straight sections, and 180° curves. There are also frames, that are facing the sun, and thus have sun-flares and high illumination changes. There are dynamic objects in the dataset as well.

Achievements: As shown in the result video, the pipeline can track the motion locally consistent. The straight sections, can be recognized as straight sections, and the turns are close to half circles, which corresponds to a 180° turn in a roundabout. However, the tracking does not work through the frame which is directly facing the sun. Thus, the pipeline has to re-bootstrap and looses the scale.
This can be clearly seen in Figure 3)



(a) Path before re-bootstrap



(b) Path after re-bootstrap

Figure 3: Malaga Trajectory

5.4 Tuning parameters

Parameter tuning was one of the main tasks during the project. The main parameters are shown in 1. During the testing of the pipeline, the following parameters were the most influencal: *Minimum Harris quality* for the HCD, *Non-maximum suppression radius* for the feature selection, *Bearing angle threshold* and *Maximum Reprojection error* for triangulation and *Pixel tolerance* in the p3p algorithm.

Table 1: Tuning parameters

Tuning parameter	**Parking**	**KITTI**	**Malaga**
Minimum Harris quality	$1e-4$	$1e-4$	$1e-4$
Non-maximum sup. radius	$15px$	$10px$	$10px$
Bearing angle threshold	$0.2°$	$0.1°$	$0.1°$
Repr. error threshold	$5px$	$10px$	$2px$
Pixel tolerance	$1px$	$1.5px$	$0.7px$

6 Key insights

Eventhough the same pipeline is used on all three datasets, the parameters used in each set are really different from each other. Also the performance drastically changes between the datasets. While the latter was expected, we did not expect that tuning parameters will be so sensible. Also, the way in which parameters had to be tuned was not straight forward, since we realized that it is not always possible, to tune parameters indepently from each other. This was especially true for the pixel reprojection error and the RANSAC distance thresholds, since these had a big influene on each other. We also realized, that more tracked keypoints do not neccessarily mean a better trajectory, since they can also destabilize the RANSAC estimate.

7 Conclusion

The proposed monocular visual odometry pipeline implements the basic functionalities to track a trajectory given video frames and the camera intrinsics.

7.1 Future work

The proposed pipeline could be improved in several ways: A landmark filter that removes landmarks that are triangulated to far away or behind the camera, could stabilize the trajectory and decrease drift. Furthermore, a scale compensation after the re-bootstrap could lead to a globally more consistent pipeline. The threshold parameters could still be improved or adapted dynamically according to the current amount of keypoints and to decrease the runtime. Additionally, bundle-adjustment or loop-close techniques could be applied, to get rid of drift. Parallel computing and an improved code implementation could also speed up the runtime.