ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ROBOTICS &
PERCEPTION
GROUP

Mini Project Report

Vision Algorithms for Mobile Robotics 2018

Autumn Term 2019

Supervised by:

Authors:
Flavio De Vincenti
Lionel Gulich
Victor Klemm
Tommaso Macrí

Contents

Chapter 1

Introduction

In this report we present the workings and results of the Vision for Mobile Robotics 2018 mini-project, which consisted of building a visual odometry pipeline. Our pipeline was developed in *MATLAB* and uses as many of the pre-written functions of the *Computer Vision Toolbox* as possible. As an additional feature we present the application of the pipeline on a challenging dataset from the Ascento robot, and fusion of its wheel odometry with the visual odometry.

Chapter 2

Implementation

2.1 Bootstrapping

Goal of the initial bootstrapping is to provide an initial estimate of the camera pose and a set of landmarks and their respective keypoints on the image plane. The bootstrapping was subdivided into 4 parts:

Frame selection Two frames are selected for determining the relative camera pose between them.

Keypoint extraction Using the Harris keypoint extractor on each frame the n most distinctive keypoints are extracted.

Patch matching We match the keypoints between the two frames by extracting the respective block descriptors and comparing their euclidean distances.

Pose estimation Using the 8-point algorithm, RANSAC and the matched keypoints, an estimate of the Fundamental Matrix is obtained. This can then be used to obtain a pose estimate of the camera.

Landmark triangulation Lastly the keypoints from both frames are triangulated using the camera pose estimate to obtain the keypoints position in 3D space.

2.2 Continuous Operation

To let the VO pipeline run continuously over time, the Markov design suggested in the mini-project statement was implemented. At each frame, the function *processFrame* takes the previous state, the previous image and the current one as inputs, and returns the updated state as well as the updated camera pose as outputs.
The continuous operation was subdivided into 2 parts:

Pose estimation The current pose was estimated only using the tracked keypoints and landmarks from the previous frame. KLT was used for tracking and P3P for estimating the pose, alongside RANSAC to refine the calculation.

New landmarks triangulation In order to cope with the landmarks "leakage", new ones have to be continuously triangulated over time. To do this, the candidate keypoints from the previous frame are tracked in the current one. Then, the corresponding 3D points are triangulated, and if the angle between the bearing vectors from the triangulated landmarks to the camera positions corresponding to their current and first observations, respectively, is higher than a certain threshold, the considered keypoints and landmarks are added to the state.
New Harris corners are subsequently detected and added to the list of the candidate keypoints, such that these are not redundant with the existing ones.

2.3 Additional Feature

As an additional feature we present the application of the pipeline on a dataset recorded on the Ascento robot. Furthermore, we show the use of wheel encoders for refining the visual and overall pose estimation.

Figure 2.1: Current prototype of the Ascento robot.

Ascento is a two-wheeled inverted pendulum robot. As can be seen in Figure 2.1, it combines wheels and legs, which allows it to not only move around on flat grounds, but also overcome obstacles through jumping. Its main body includes a forward pointing matrix vision mvBlueFox global shutter camera on which the dataset was recorded. The dataset consists of an exploratory drive through the *Maschinenhalle* of the *ML* building at ETH Zurich.

The dataset contains multiple difficulties:

Lighting Since the dataset was recorded after dark, all light sources are artificial. Generally the light conditions in the ML building are rather dark and also far from homogeneous.

Environment structure The dataset's near environment consists mainly of plain floor and walls of factory cubicles, which do not offer a lot of structure. Thus detection of keypoints, in the near field of view is hard. These keypoints are especially essential for acquiring good translation estimates.

Camera movement As the robot is essentially a two-wheeled inverted pendulum its locomotion inherently adds additional movement to the camera as compared to a camera mounted on a 4-wheeled vehicle.

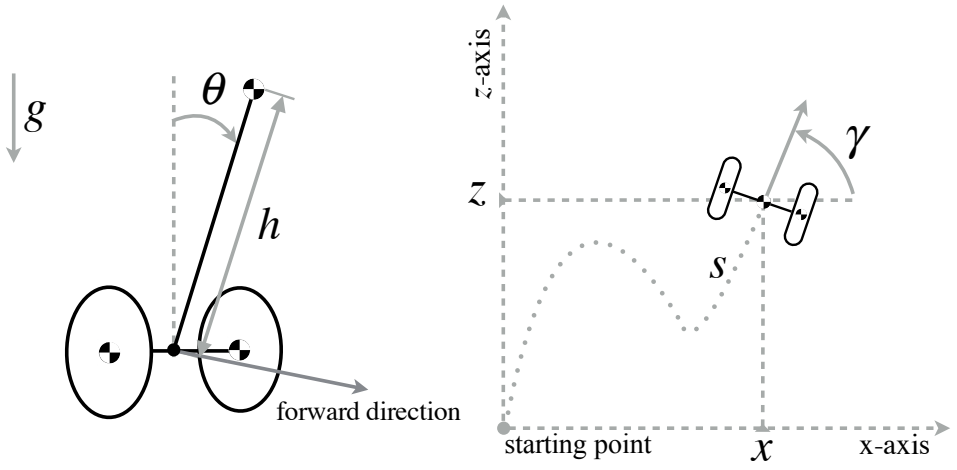

Figure 2.2: Relevant kinematic quantities of the Ascento robot model.

In addition to its camera, the Ascento robot is equipped with two incremental wheel encoders, whose measurements are used in conjunction with a kinematic model to estimate its position on the plane 2.2. When recording the dataset, it was made sure that all relevant sensor measurements (e.g. wheel encoder position and tilt angle by an inertial measurement unit) and their respective timestamps were recorded along the image stream.

This dataset was then imported as a *ROS* (robot operating system) bag file into the *MATLAB* environment, where all measurements were read in using a custom made *ROS* message format. As the camera is equipped with a fish-eye lens the equidistant distortion model had to be used. For rectifying the images the C++-library *undistorter*[1] was used. The results will be further discussed in chapter 4, but to motivate the following work, a few things need to be noted:

As monocular visual odometry is used, no absolute scale can be determined. Furthermore, as the dataset is very challenging due to the reasons discussed above, the visual odometry pipeline is affected by a severe scale drift. These two reasons make the estimate of the visual odometry not (yet) usable for a global robot pose estimation, although the direction and orientation estimated by the visual odometry matches the wheel odometry prediction closely.

Out of these shortcomings the idea of the additional feature directly emerged: Using an extended Kalman filter (EKF) the absolute scale and scale drift of the visual

[1] `https://github.com/schneith/undistorter/`

odometry could be corrected through the wheel odometry data, while simultaneously fusing the two measurement sources into a global robot pose estimate.
The implementation of this additional feature can be roughly subdivided into the following 4 parts:

Initialization The extended Kalman filter is initialized and the initial robot pose transformations are inferred for both the visual and the wheel odometry.

Gather robot poses Out of the current camera transformation matrix a current robot pose is calculated. This is done by using the default transformation of the camera frame in the robot's world frame (centered at its wheels) read out from the robot's CAD model. This transformation is then adjusted through the current pitch angle of the robot and projected onto the floor to infer the current pose. Additionally, also new values for the wheel odometry are read in and adjusted to the frame of reference.

Extended Kalman filter The robot pose differences between the current and the last iteration for both the visual and wheel odometry are fed into the extended Kalman filter and the prior and measurement update steps are executed. This results in an estimated global robot pose difference and the visual scale drift correction.

Plot integrated robot poses The robot pose differences are integrated to form a current robot pose for the visual, the wheel encoder and world estimate each. These are then visualized as robot models in a 3d-space and as a global trajectory showing the path of each estimate.

A test of the additional feature on the custom dataset including a visualization can be found in chapter 4.

Chapter 3

Tuning

All parameters are specific to the dataset and were carefully tuned by the students.

3.1 Tuning of bootstrapping

The most important parameters of the initialization process are the selected frames as well as the parameters for the RANSAC. The frames have to be chosen such that the movement between them is large enough for triangulating the landmarks accurately while still being small enough such that keypoints still can be matched precisely. For the filmed-out-of-car datasets Kitti and Malaga frames 1 and 3 proved to be a good measure. For the parking and Ascento dataset 1 and 5 were chosen as they have a slower camera motion.

For the 8-point algorithm RANSAC a confidence of 85% and distance threshold of 10^{-4} have lead to good results.

3.2 Tuning of continuous operation

We found that small changes in the parameters, especially in the initialization, led to completely different tracked trajectories. Yet in all cases the pose estimates were locally consistent.

It is worth mentioning an issue that, once solved, resulted in considerably better estimates. In *MATLAB*, the location of the detected corners is returned as a single precision value, while the tracking, triangulation and pose estimation operations can be computed up to double precision. Thus, after having performed the necessary type casts, the trajectories became much more accurate, and their consistency had a much broader scope.

The overall quality of the tracked landmarks decreased over time, and for long datasets, like the Kitti one, this often led to an insufficient number of inliers for the estimation of the camera pose. To cope with this inconvenience, a strategy was to substantially increase the number of detected keypoints in each frame; however, this usually led to poor estimates, although no error was thrown for the whole duration of the video.

3.3 Tuning of the additional feature

The only quantities that can be tuned within the additional feature are found in the extended Kalman filter:

Initial state estimate and variance As the evaluated dataset is rather short with 499 images, the initial estimates were tuned for fast convergence, to show the filter in action instead of it converging to the correct values.

Model noise As the model noise is an empirical quantity, it was tuned manually to achieve stable and reasonable results. As there is only one dataset at our disposal, one might argue that these parameters are overfit to the current one.

Measurement noise To determine the measurement noise of the measurement sources, we would need to determine their variance in relation to a ground truth, which is unfortunately not available. To illustrate the workings of the filter, the variances of the rectified visual odometry and the wheel odometry were chosen as equal.

Chapter 4

Results

4.1 Kitti

The kitti dataset was for our group the most challenging. In particular we noticed that the obtained trajectories were very parameter sensitive and we had to commit a lot of time in order to tune the algorithms we used not only to obtain a satisfying result, meaning not only a reasonable pose estimation but also the completion of the dataset. After several trials, the result was much more accurate, as the final trajectory is very similar to the one obtained with *pose refinement*, considered as reference.

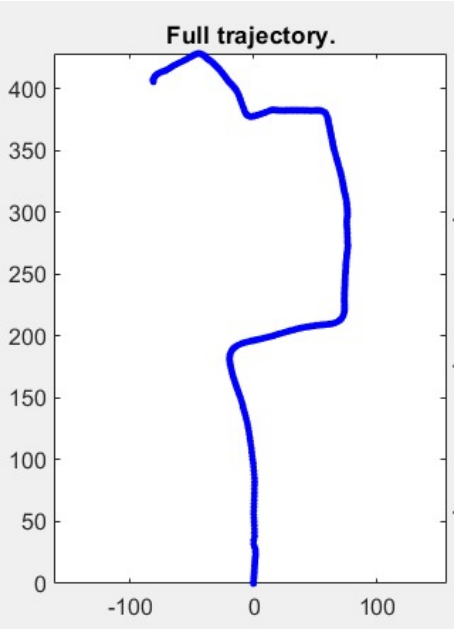

Figure 4.1: Full trajectory of the kitti dataset.

4.2 Malaga

The Malaga dataset is another available outdoor traffic environment dataset and even if it is considered to be more difficult than the kitti one, we could obtain

satisfying results with less parameter tuning. This happened both because we concentrated especially on the kitti dataset and because we were lucky with the initial guess of the parameters.

4.3 Parking

With the parking dataset, a simulated video of an indoor garage environment, we encountered some problems as the Visual Odometry provided a very poor estimation; the reference was clear, the motion should have been a simple straight line. Probably, the parameters we used for that dataset were optimized for a motion in the z direction, on the other side the Garage dataset consists only of a translation in x-direction.

4.4 Ascento

The Ascento dataset is the one we used to apply the additional feature on, but we also tested our pure visual odometry algorithm on it. We obtained at first a very poor result, as the environment is particularly challenging. Only then have we realized that because the camera on board of Ascento is equipped with a fisheye lens, we had to rectify the images using the equidistant distortion model. Further we extracted the Harris corners only from a rectangular central area of the frames and the trajectory improved a lot.

4.5 Additional Feature

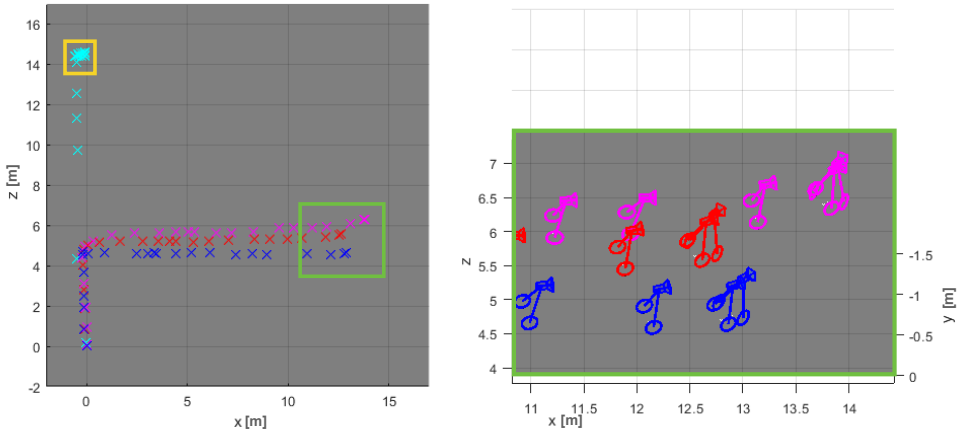

Figure 4.2: *Left:* Estimated trajectory. In *cyan* the output of the visual odometry pipeline, in *blue* the rectified visual odometry, in *magenta* the wheel odometry estimate and in *red* the overall most probable estimate. *Right:* Enlarged view of the area in the green rectangle of the left image. The measured and estimated robot poses are visualized, whereby the colors correspond to the ones on the left.

The results of the scale corrected and fused estimations 4.2 *left* seem reasonable: The scale of the rectified visual odometry matches the scale of the wheel encoders in absolute terms and the 90°turn is quite accurately depicted by both measurement

sources. It is to be noted that the plots in 4.2 and 4.3 are only updated every 15 frames for clarity of the results.

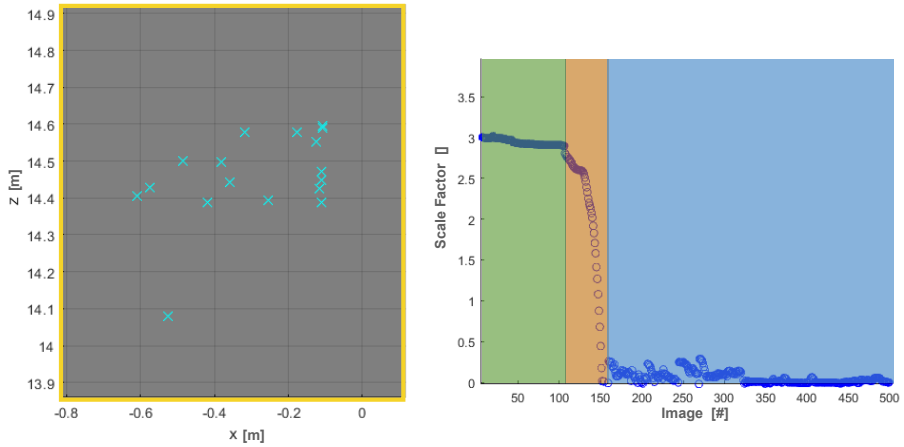

Figure 4.3: *Left:* Enlarged view of the area in the yellow rectangle in 4.2 *left*. To be noted is the severe scale drift induced during and after the turning maneuver of the visual odometry. *Right:* Estimate of the scale factor in x-direction. The shaded green, orange and blue areas correspond to before, during and after the turning maneuver.

Internally, severe scale drift experienced during and after the turning maneuver 4.3 *left* leads to an extremely small absolute scale. The extended Kalman filter successfully detects this change 4.3 *right*, whereby the estimated scale factor is large with nearly no scale drift during the straight part of the dataset and then begins to drop significantly while the robot is turning. When going straight again, it converges towards a small scale factor, representing the extremely small absolute scale experienced by the visual odometry after the turn.
Further, a comparison to an unfortunately non-existent ground truth would be required to quantify the results, although qualitative evaluation of position and orientation 4.2 *right* seems to match the dataset in both cases to content.

Appendix A

Appendix

A.1 Used pre-written functions

- *detectHarrisFeatures*, getKeypoints.m, triangulateNewLandmarks.m
- *cornerPoints.selectStrongest*, getKeypoints.m, triangulateNewLandmarks.m
- *extractFeatures*, getDescriptors.m
- *matchFeatures*, getMatchedKeypoints.m
- *estimateFundamentalMatrix*, getRelativePose.m
- *relativeCameraPose*, getRelativePose.m
- *cameraPoseToExtrinsics*, getLandmarks.m
- *cameraMatrix*, getLandmarks.m
- *triangulate*, getLandmarks.m, isNewKeypoint.m
- *estimateWorldCameraPose*, estimateCurrentPose.m
- *vision.PointTracker*, estimateCurrentPose.m, triangulateNewLandmarks.m
- *rotx*, tf2RobotPose.m, plotRobotPose.m, additionalFeature.m
- *roty*, plotRobotPose.m
- *rot2Eul*, tf2RobotPose.m