Visual Odometry Pipeline

Pascal Buholzer, Fabio Dubois, Milan Schilling, Miro Voellmy

January 6, 2017

Symbols

Contents

1 Introduction

The aim of this mini project is the development of a visual odometry pipeline. This pipeline takes the consecutive gray-scale images of a single digital camera as input. Therefore the pipeline developed in this mini project is a monocular visual odometry pipeline.

The output of the pipeline is the position of the camera in relation to its initial position for each frame.

keywords: (VO, sequential, monocular, markov assumption)

2 Implementation

2.1 Framework

This pipeline was developed in MATLAB. Since the group consists of four students, a Git repository was used to be able to work on different files simultaneously, and to enable version control. (keywords: MATLAB, Git)

2.1.1 Coordinate Frames

In this mini project the coordinate frames were defined as shown in fig. 1. The camera coordinates are in a way oriented, that the x-y plane lies parallel to the image plane, while the z-axis is pointing towards the scenery. The world frame however is oriented in such a way that the x-y plane is parallel to the ground and the z-axis is pointing upwards. The origin of the world frame is at the same location as the origin of the first boot-strap image.

Transformation between frames are described by homogenous transformation matrices. T_{AB} maps points from frame B to frame A.

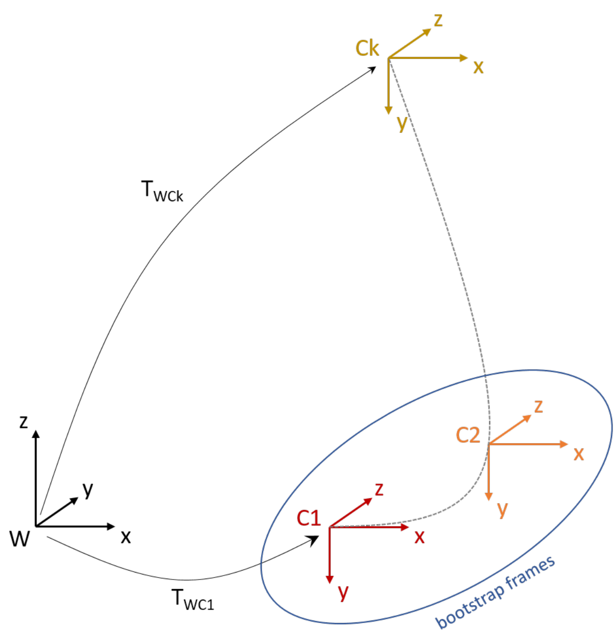

Figure 1: Coordinate Frames

2.1.2 Pipeline overview

As shown in fig. 2 the pipeline consists mainly of three parts, a bootstrap, the initialisation and the continuous operation. In section 2.2 and section 2.3 the initialisation and the continuous operation are described in detail.

2.1.3 Options and parameters

(keywords: parameter handling, GUI)

2.2 Initialization

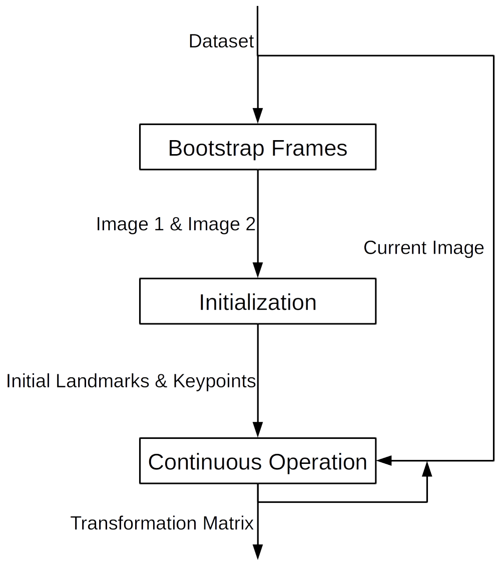

Figure 2: Rough Flow chart

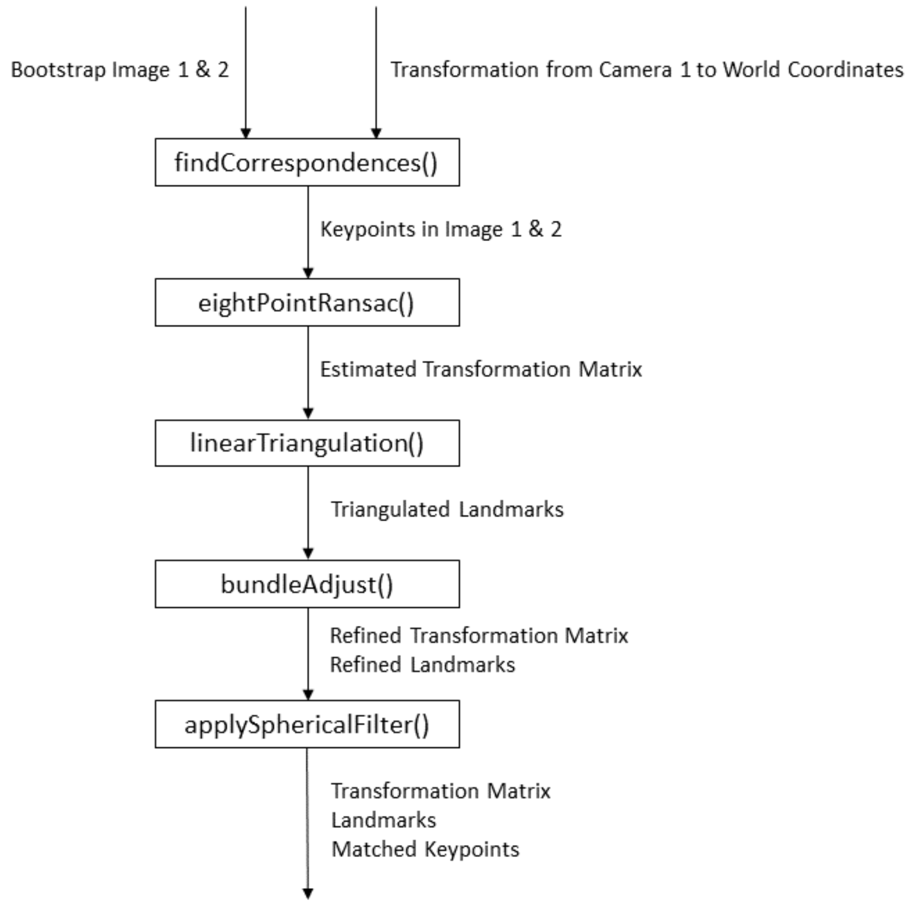

Figure 3: Init Flow chart

2.3 Continuous Operation

Continuous operation of the VO pipeline is implemented in the "process frame" function. It tracks keypoints with corresponding landmarks over several frames while estimating the pose difference between successive frames. Further, a keypoint tracker finds new candidate keypoints which will become new landmarks if a candidate keypoint was tracked war enough and achieved "good" trianguability. This ensures to never run out of landmarks and keypoints if the image changes over time.

2.3.1 Conventions

- Index of previous frame: i
- Index of current frame: j
- Pose difference between previous to current frame: $T_{C_i C_j}$

2.3.2 p3p_dlt_Ransac oder Pose difference estimation

To estimate the pose difference $T_{C_i C_j}$ from frame i to j we use the p3p ransac algorithm also used in exercise 5. If wished the ransac can also use dlt pose estimation. Using these ransac algorithm ensures to remove outliers from our landmarks. We don't use DLT refinement after the p3p ransac since the best guess from p3p often gave better results.

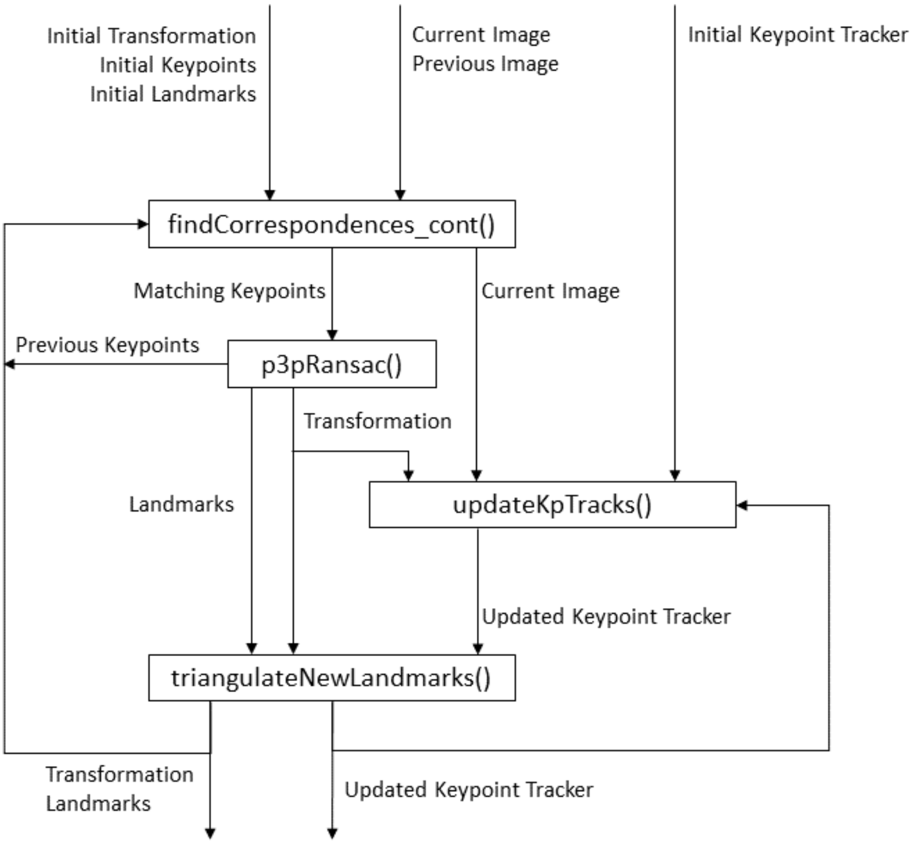

Figure 4: Cont Flow chart

3 Results

3.1 Overall performance

(keywords: Real time ness, comparison to groundtruth, compare different datasets Impact of features)

4 Discussion

What have we learned, what worked?

Possible future work, improvements (loop closure, ...)

5 Conclusion