Visual Odometry Pipeline

Pascal Buholzer, Fabio Dubois, Milan Schilling, Miro Voellmy

January 6, 2017

Symbols

Contents

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)

Implementation

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)

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 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.

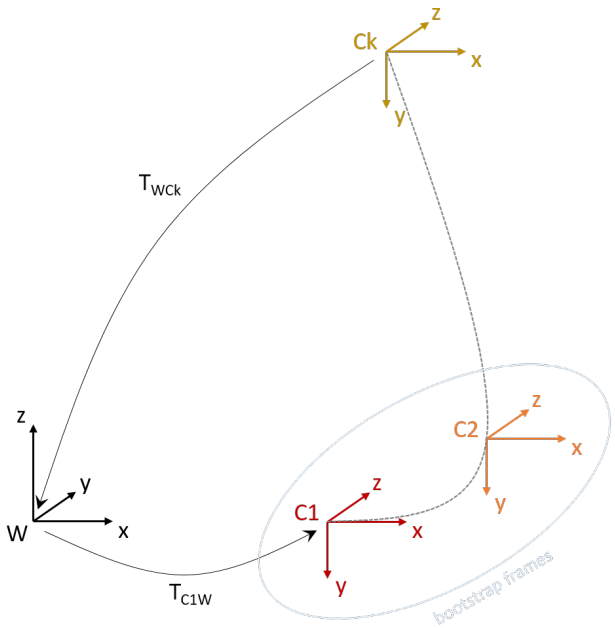

Figure 1: Coordinate Frames

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.

Options and parameters

(keywords: parameter handling, GUI)

The pipeline was designed in a modular way. A lot of functionality was wrapped into self-contained and self-explanatory functions, as described in the pipeline overview. Next to this 'functional programming' approach all the tuning variables (e.g. number of keypoints, bearing angle thresholds, etc.) were centrally aggregated in a parameter struct.
For further insight please consult the function `loadParameters.m`.

In order to run the visual odometry two launch procedures where implemented: For development and debug mode the main.m with default parameters can be executed. Numerous individual plots are displayed with

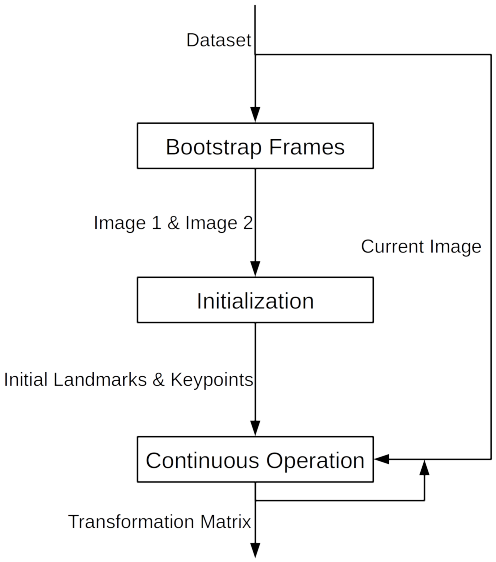

Figure 2: Rough Flow chart

insightful information about matching, inlier rejection and triangulation.

Out of performance reasons a more compact and user-friendly display of the pipeline output was created with a GUI designed with the Matlab GUIDE application, see fig. 3. Only the most crucial entities, like number of triangulated keypoints, are visualized for intuitive understanding and easy handling.
Please follow the steps below to run the visual odometry through the GUI environment:

1. adapt dataset paths in `loadParameters.m`
2. type into MATLAB command window: `gui_simple`
3. in *Parameters* panel select dataset to run on and toggle respective radio buttons
4. hit *Run* to launch the visual odometry

For more advanced parameter tuning resort to changing the default parameters in `loadParameters.m`.

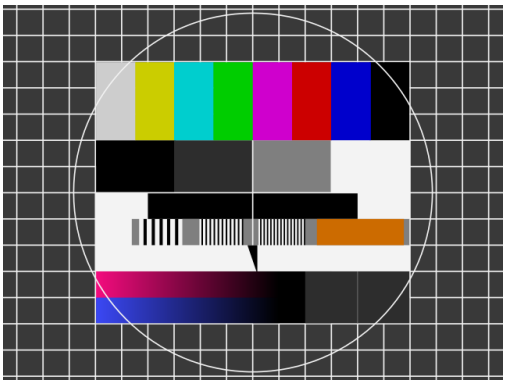

Figure 3: Graphical user interface

Initialization

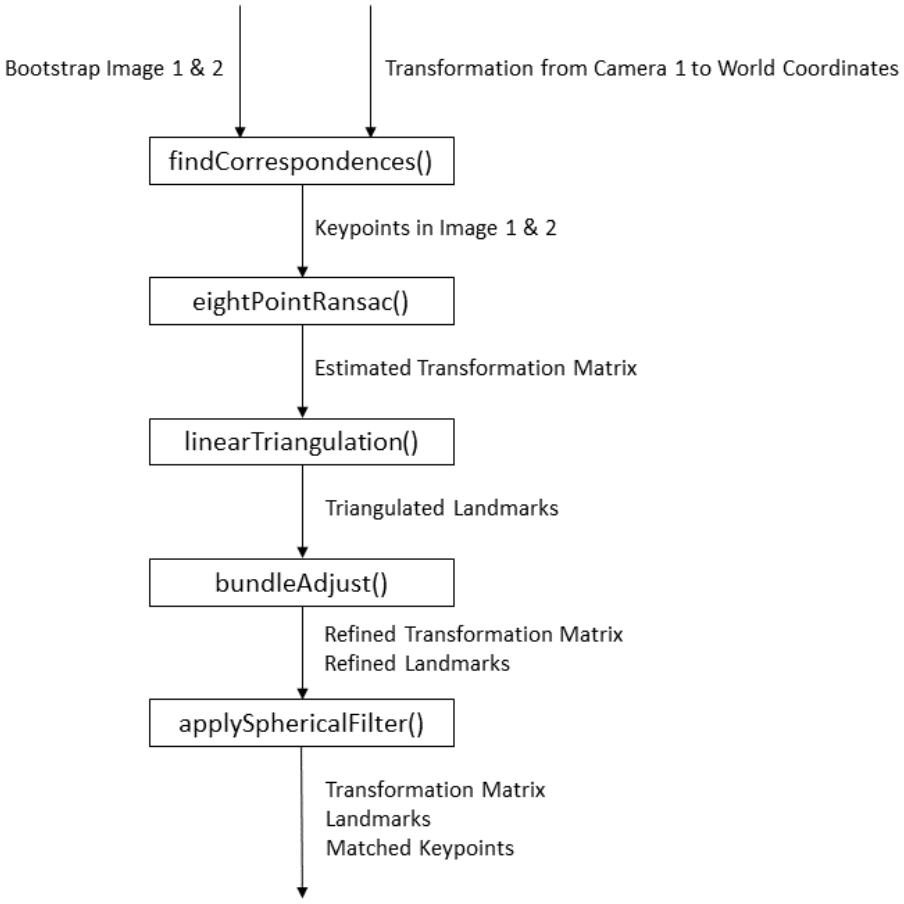

Figure 4: Init Flow chart

Continuous Operation

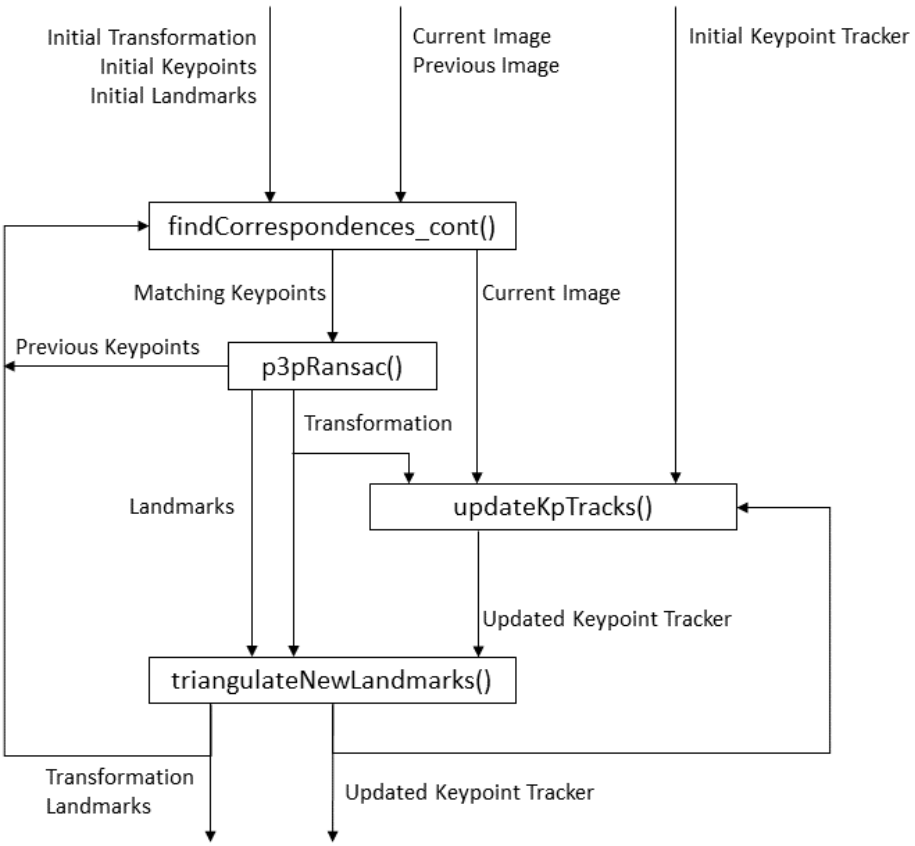

Figure 5: Cont Flow chart

Results

Overall performance

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

Discussion

What have we learned, what worked?
Possible future work, improvements (loop closure, ...)

Conclusion