Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ROBOTICS & PERCEPTION GROUP

Programming Project Report

Autumn semester 2021

Giacomo Manzoni – Roberto Pellerito – Lorenzo Piglia

Contents

Implementation Overview:

The pipeline has been developed following the theoretical concepts introduced in the lectures. The main techniques used came from the ideas developed in the recitations, however, they were often applied from a different perspective and combined freely to better suit the project.

Aiming for maximum performance MATLAB built in functions (C*omputer Vision Toolbox*) have been used as much as possible.

Both direct and indirect methods have been used, inspired the SVO workflow.

The main building blocks are the *initialization* and *processFrame* functions acting as the bootstrapping and continuous operation phase.

Other additional functions have been implemented like *displayTrajectory* to add additional features.

To obtain a cleaner and easier to tune code all the parameters have been placed into one single struct, accessible via *getParams.*

Bootstrapping:

The initialization phase uses the classical indirect features matching approach on the first and 4th camera views. Originally bootstrapping with direct features tracking has been considered as an option but it produced poor results, in our opinion because frames are to far apart for the Taylor expansion to hold.

The bootstrapping frames had to be chosen carefully to ensure sufficient baseline for accurate triangulation of the point-cloud, while still being small enough such that keypoints can be matched properly.

Various features detectors and descriptors have been tried, settling for the Shi-Tomasi detector in order to get the most accurate features localization and therefore initial pose estimation. Sift detector has been considered, in order to return even more robust results but successive frames are reasonably similar in terms of viewpoint and illumination changes, not to cause significative difference.

The filter size has been manually tuned to get the highest features quality while keeping the computation as light as possible.

8 point RANSAC has been used to retrieve the initial pose of the camera wrt the first camera which acts as the world frame. After computing the fundamental matrix it is decomposed to get R|t given the intrinsic matrix K.

Now it is only a matter of linearly triangulating the landmarks, keeping only the ones with reasonable accuracy (reprojection error).

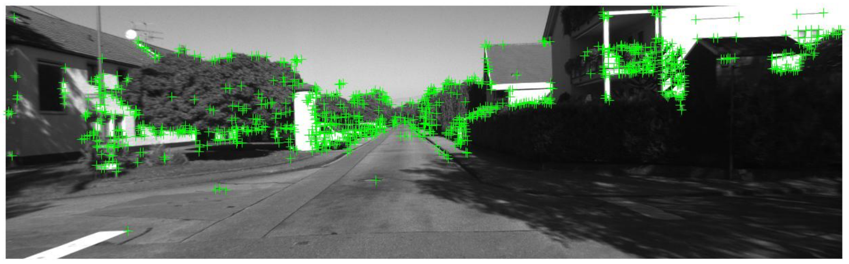

Figure 1: Shi-Tomasi features in KITTI bootstrapping

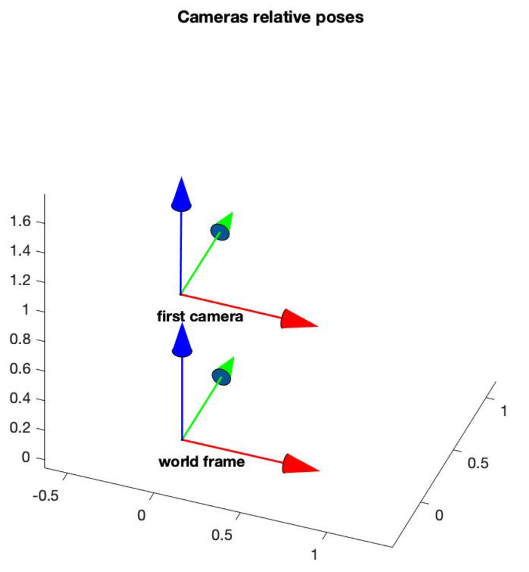

Figure 2: Poses bootstrapping for KITTI dataset

Continuous Operation:

After having initialized the Visual Odometry pipeline with the first poses, the goal is a continuous trajectory retrieval.

The fastest and more accurate approach to feature tracking is via direct methods and here is when KLT came into hand. To ensure better performance semi-dense tracking have been utilized.

One issue with KLT has been that it returns float values for the px coordinates since it applies a warping function. It caused many problems with features tracking but once recognized it has been fixed with a simple rounding.

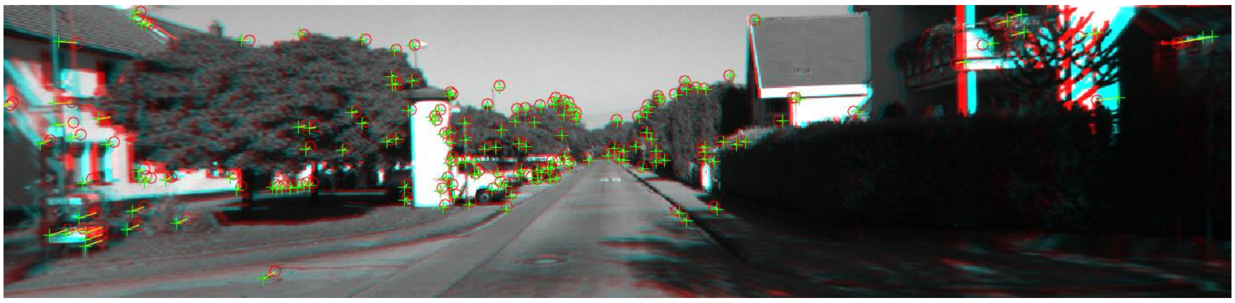
Figure 3: KLT features tracking in KITTI dataset

During the tracking process some keypoints get lost, therefore the point-cloud needs to be trimmed accordingly.

Loosing keypoints has been a big issue in the early stages of development because it prevented the algorithm to run pose estimation and eventually die.

During successive stages of development a balance in the number of features has been found by fine hand tuning.

From here the camera pose can be retrieved via P3P RANSAC.

The classical VO requires keyframe extracting however following the statement suggestion here this has been implemented in a continuous fashion to satisfy the Markovian convention.

extractKeyframes:

This is the function responsible for mimicking the keyframe extraction, the core of the algorithm.

Firstly candidate keypoints have been tracked with the same direct approach as above (*S.C*). Accordingly the candidates pose history have been updated (*S.T* and *S.F*).

Secondly the current state needs to be augmented. The discerning factor is the “angle of the tracking sequence”, as suggested in the statement.

Keypoints passing the angle test have been triangulated and only those with a reasonable reprojection error (1px) have been added to the point-cloud.

In order for the process to be sustainable the candidate list needs to be updated too. This happens similarly to the initialization phase by features extracting. Even here Shi-Tomasi detector have been used and only the strongest corners have been kept.

To make feature tracking more robust a re-initialization function has been introduced. Therefore the pipeline can also cope with pure rotation.

Additional feature:

The additional feature to the assignment has been chosen to be the *determination of the ground truth trajectory by extrapolation of the scaling factor*.

The technique used here has been to recognize in the video an object of known dimensions, from which the real world translation can be deduced.

The datasets are all based on a car driving around, therefore the object in question has been chosen to be a car of known dimensions.

Knowing the real world length of a particular car part, and knowing for how many frame it is visible in the video, the real translation is trivially determined.

In particular in our code this happens in kitti frame 40, malaga frame 95 and parking frame 101. The effect is visible in the trajectory plot as it gets rescaled immediately.

In our implementation this has been hard-coded but ground for future development could be utilizing *template matching* to automatically localize the car in the sequence.

An even more advanced solution, which could lead to online scale retrieval, would be to utilize panoptic segmentation to determine when a generic known car is present in the scene and applying the same reasoning from here to get the scale factor.

Final Results:

Our final implementation is accurate enough to ensure great local consistency even without optimization methods.

Here are some of our results:

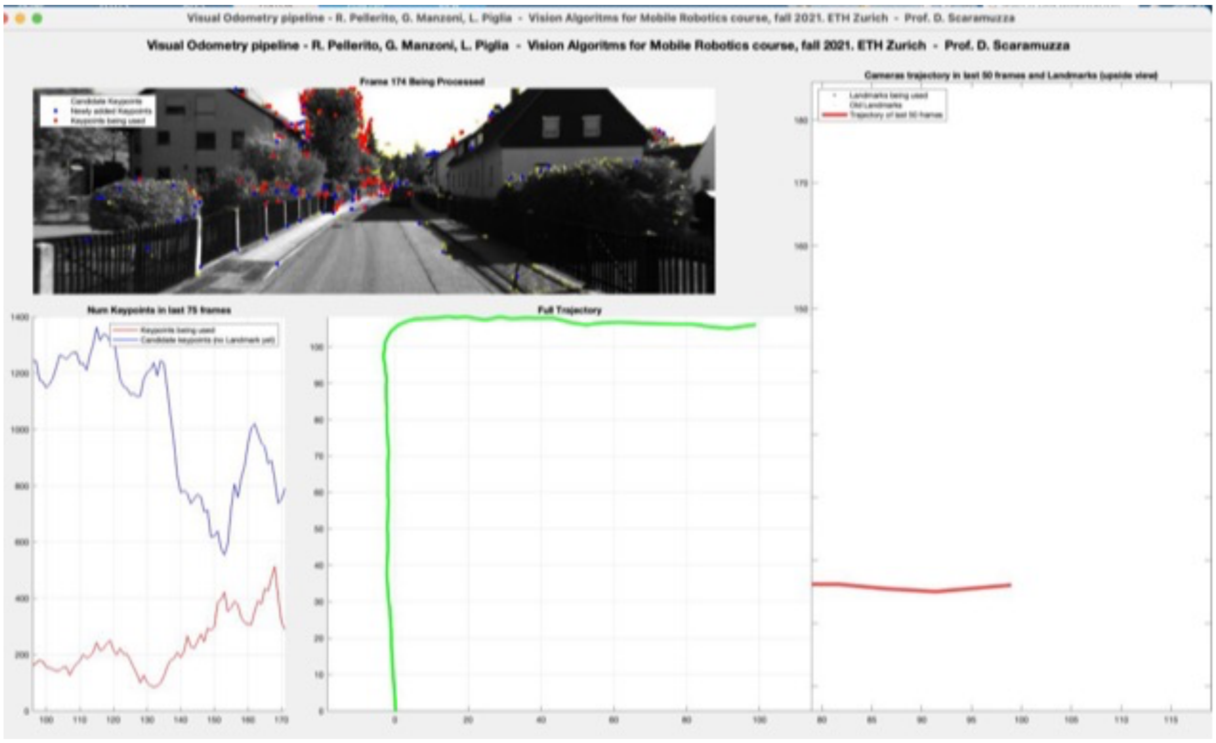
Visual Odometry pipeline - R. Pellerito, G. Manzoni, L. Piglia - Vision Algoritms for Mobile Robotics course, fall 2021. ETH Zurich - Prof. D. Scaramuzza
Frame 174 Being Processed
Candidate Keypoints
Newly added Keypoints
Keypoints being used
Cameras trajectory in last 50 frames and Landmarks (upside view)
Landmarks being used
Old Landmarks
Trajectory of last 50 frames
Num Keypoints in last 75 frames
Keypoints being used
Candidate keypoints (no Landmark yet)
Full Trajectory

Figure 4: See how the trajectory is smooth even in the sharp curve of kitti

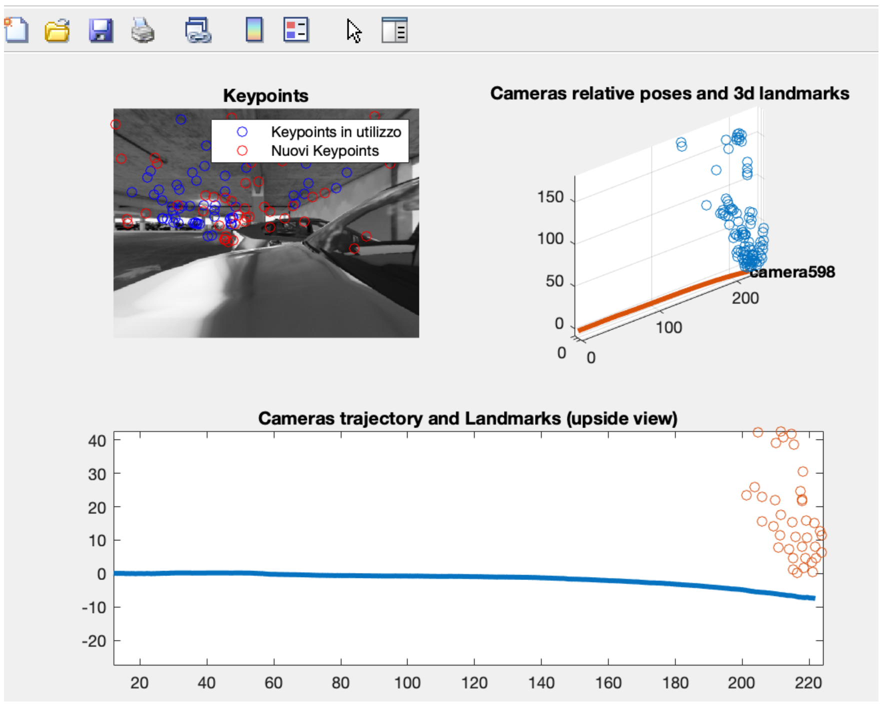
Keypoints
Keypoints in utilizzo
Nuovi Keypoints
Cameras relative poses and 3d landmarks
camera598
Cameras trajectory and Landmarks (upside view)

Figure 5: Trajectory in the parking dataset. The algorithm accumulates little drift, which could be corrected by integration of external sensors data