ELECTRICAL AND COMPUTER ENGINEERING
ACADEMIC YEAR 2023/2024
INSTITUTO SUPERIOR TÉCNICO

Image Processing and Vision

PIV Report

Localization using camera footage

Authors:	*Email:*	*Number:*
João Mateus	joao.c.mateus@tecnico.ulisboa.pt	99969
João Góis	joao.f.gois@tecnico.ulisboa.pt	99972
Rodrigo Francisco	rodrigofrancisco@tecnico.ulisboa.pt	100073
Tomás Marques	tomas.luis.marques@tecnico.ulisboa.pt	100104

Group 15

2023/2024

1 Objectives

The objective for Part 1 was to obtain localization on a (given) map, via homographies.

To compute the homographies, a structured pipeline has been established. The process involves utilizing *process_video.py* to extract features from the video frames. With the obtained keypoints, homographies are then calculated between frames of the input video. Finally, all the homographies are exported to an output file.

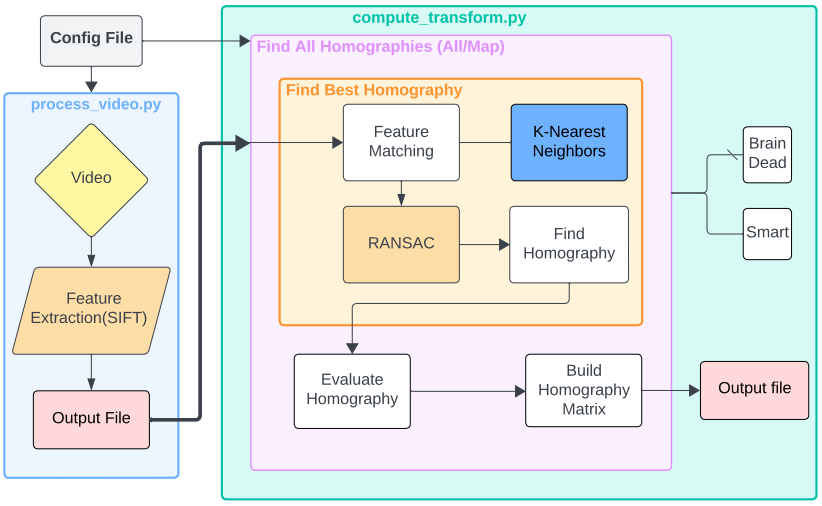

Figure 1: Project Pipeline

2 Methods

2.1 Feature Selection

In order to obtain the keypoints for the calculations of the homographies, the SIFT package was used to find all the most relevant features and to be used on the calculation of the homographies. Also, SIFT was run with gray scale images.

To get better quality over quantity in the keypoints, the images were downscaled, this ensures that the limited keypoints aren't redundant or as easily confused with each other.

2.2 Feature Matching

The feature matching was done using the KNN package. SIFT, used in section 2.1, provides us with descriptors for the keypoints. These descriptors are exploited to match the features in pairs.

To ensure the matching pairs are top-notch, a distance threshold was implemented. Pairs with a distance bigger than the threshold will be filtered out. Doing so, we can guarantee that all the features paired are actual matches and not just random pairs that ended up together because they were close enough to be aggregated by KNN. Furthermore, the program is prepared to raise the distance threshold to bigger values if not enough couples were found.

2.3 Ransac

The RANSAC algorithm was used to find the plane in the images with the biggest number of points, so that the best possible homography is obtained. In addition, the keypoint pairs returned by the Feature Matching maybe be mismatched, and these are taken care of by RANSAC.

The threshold to decide if a keypoint is inlier or not, will depend on the degree to which the keypoints follow the overall transformation between frames. To make the threshold adapt to each video, it was made proportional to the standard deviation of the distance error from KNN, obtained in the previous section.

2.4 Compute Homographies

The homography computed from one frame to another is derived from the inliers obtained through RANSAC. This is achieved by providing all the inliers to a function that returns the eigenvector corresponding to the homography matrix for our inliers.

To compute homographies between distant frames (e.g., m and n), we initially adopted a sequential multiplication approach, where consecutive homographies from frame m to n were multiplied (as seen in Figure 1, the Brain Dead block). While effective for smaller videos, this method exhibited error propagation, resulting in larger errors for distant frame pairs.

In response, we introduced an improved method (Smart Block, in Figure 1). This new approach computes homographies for non consecutive frames directly when possible. This is determined by the quality of the homography via mean squared distance between keypoint pairs. After all the possible direct jumps are obtained, the algorithm finds the smallest sequence of homography multiplications to the rest of the frames, this strategy minimizes error propagation.

3 Extras

3.1 Constants

The project has a lot of important constants, that help us control the data and the output. For instance, `Step Size` defines how many frames are jumped/ignored. `Frame Limit` allows us to define a limit for the number of processed frames.

3.2 Piv Lib

The project contains a folder called *piv_lib*. In this folder it is possible to find three scripts. The first one is $config.py$, a parser for the config files. Secondly, *cv.py*, where are located handmade versions of popular algorithms and functions like RANSAC, warpPerspective, findHomography and more. The last one, *utils.py* has some useful functions. The most helpful throughout the project was $showTransformations$, because it plots the keypoints pairs, before and after ransac, which was crucial for debugging. Likewise, $showHomography$ is another standout function, which as the name implies, plots the homography.

3.3 Run Project

To provide an enhanced testing experience, a file called *run_project.py* was devised. This script runs *process_video*, $compute\_transform$ and an additional file, *homography_probe*, that provides a user interface to examine the homographies. By choosing one of the available modes, and the frame numbers, it's possible to visualize all the plots and information one could desire.