Author: Polato Anna, 2007061
Scarpa Mattia, 2005826

Homework 3: Object recognition and tracking

This project required to detect and track a set of objects in a video.
As first step, we loaded the video and we took the first frame to detect some planar objects of interest and, in order to do this, we used some reference images representing the desired objects. Firstly, we found SIFT features from both reference images and frame, defined by keypoints coordinates and corresponding descriptors. Then using a brute force matcher (implemented by the function cv::BFMatcher in OpenCV), we identified corresponding features matches resorting to the KNN method and we kept only good matches applying Lowe's ratio test between the best two matches for each keypoint.

The choice of the Lowe's test threshold is relevant to get the minimum number of outliers during the optical flow estimation for an object, however, it is also necessary to have enough data points to maintain a good tracking.

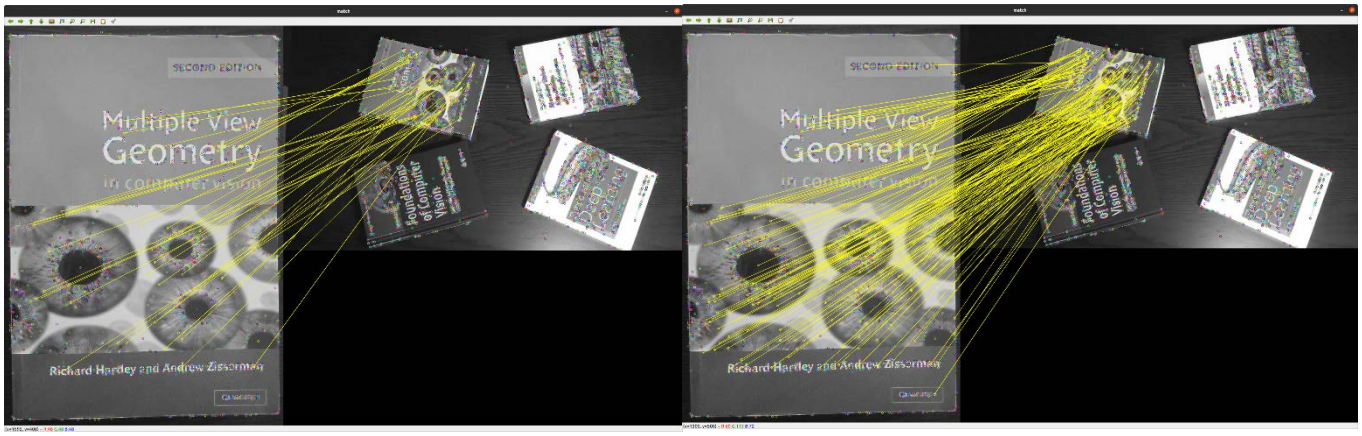

Figure 1: a) features matching with Lowe's threshold = 0.45; b) features matching with Lowe's threshold = 0.8

We concluded the detection process drawing a bounding box around each detected object in the frame. For this reason, we used good matches coordinates to compute the homography matrix (using the function cv::findhomography()) of objects of interest so that it was possible to find their corresponding corners in the frame, and use them to outline the bounding boxes.
To make the perspective transformation more robust, we used the RANSAC method to discard the outliers and, a mask indexing their positions, provided by the used function, allowed preserving only inliers.

At this point we started the tracking process on the objects of interest and we performed it for all subsequent frames of the video.
We tracked the found inliers using the LK algorithm which is implemented by the function cv::calcOpticalFlowPyrLK().
Given two subsequent frames and the matched features points from the first image, the algorithm finds their position in the second image. Moreover, the LK algorithm return a status vector that allowed us to discard all the features whose optical flow has not been found.

To conclude the project, we updated the bounding boxes, at each step, computing an homography from subsequent frames to estimate the rotation and translation of the objects.

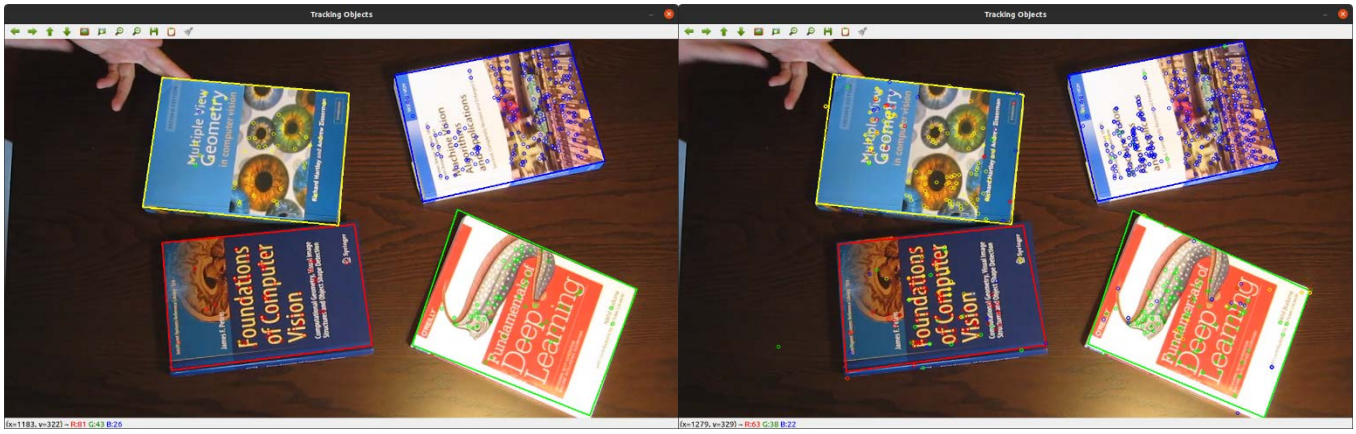

Figure 2: feature used for optical flow estimation with matches obtained from: a) Lowe's threshold =0.45; b) Lowe's threshold =0.8

SCRIPT NOTES:

The full project has been developed by both the groups member.

In the build folder it has been configured and generated a solution for a Linux platform, and the executable is built as DetectAndTrack. To run it is sufficient to type `./DetectAndTrack` in the command line from the build folder and follow the given instruction. To generate again the solution (or generate it for a different platform) create a different folder or erase the file CmakeCache.txt.

To set up the data files (video and objects images) it is strongly recommended to place the video file in the "*data/*" folder and the images with objects to be detected following the path "*data/objects/*". However, a command line parser has been implemented allowing to specify the path for the used video file or the objects images (e.g., `./DetectAndTrack -v=videoPath/`). It is also possible to manually select the corner for each object if requested ad the execution passing the specific command in the command line. Type `./DetectAndTrack -h` or `./DetectAndTrack –help` to get the information for the parameters allowed.