Monocular Visual Odometry and object identification

Ángel Lorente Rogel, *a.lorenterogel@student.utwente.nl, s2413159, MSc EE*

***Abstract*—This report presents the development of the final project of the course *2D and 3D scene analysis* of the University of Twente. It explains a process to determine relative spatial information and object identification in the form of two separate algorithms. The first one is a visual odometry algorithm to position the camera and roughly reconstruct the environment. And a second one that serves to detect one object in the image sequence.**

***Keywords*—Monocular visual odometry, object identification.**

I. Introduction

MONOCULAR Visual Odometry is a process for extracting spatial information from images taken by a single camera. This topic is being widely studied by the image processing and computer vision community and there are several methods and theories developed. This project was made using the libraries and tools mentioned in the materials section focusing in the methods with lower computation costs but maintaining a decent performance.

II. Methods and Materials

A. Materials

- Matlab image processing and computer vision toolbox
- Python v3.7.5
- NumPy v1.18.2
- OpenCV v3.4.4 with opencv_contrib library

B. Methods

1) Analysis: For this project we want to to reconstruct the trajectory and pose of the camera given only the video recorded as an input and analyzed frame by frame. The output of the system will be two arrays giving the translation and rotation of the camera for each frame, relative to the first position that will be set in the origin [0, 0, 0] in Euclidean coordinates.

2) Camera Calibration: Every camera has a nonlinear distortion produced by the imperfections present in the camera system/ lens system. For this reason it is necessary to model this behaviour by what is called the intrinsic parameters of the camera and use them to rectify the images. This parameters will also serve in the other methods for computing the extrinsic parameters and estimate the relative pose of the camera.

The intrinsic parameters are modeled as a matrix of the form:

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where f_x and f_y are the x and y focal lengths, c_x and c_y are the x and y coordinates of the coordinates of the optical center of the image and γ is the skew between the axes.

For computing the intrinsic parameters at least 4 points are needed to be known in the 3D world and and its respective 2D projection at the camera, for this purpose the chessboard pattern method is used. This method (Algorithm 1) uses a chessboard pattern to establish the 3D world coordinates and its respective 2D points in the image by knowing the size of the chessboard squares. In this project the camera intrinsic parameters were computed using the calibrator app of the Matlab image processing and computer vision toolbox.

Algorithm 1: Chessboard pattern calibration method.

input : Set of calibration images
output: Intrinsic camera parameters

1 Capture several images of the chessboard from different angles;
2 Find the 2D and 3D coordinates of the corners in the board;
3 Refine corner detection;
4 Compute intrinsic parameters;

3) Feature extraction: Once the intrinsic parameters were computed, the images are rectified and the following step is to extract the characteristic features of the image. For this task the ORB detector was selected due to its enhanced performance compared with its parents FAST and BRIEF. ORB is a FAST detector and a BRIEF descriptor which is up to ten times faster than a SURF detector. Therefore, the detectors SURF and SIFT were discarded due to its heavier computation costs.

4) Feature tracking: Once the keypoints and descriptor are obtained the next step is to find those in the following frame. For this purpose feature tracking is used, and the algorithm selected is the KLT tracker, which it is also implemented in OpenCV. On a first instance it was proposed to use again the ORB detector with feature matching but after several tries the KLT algorithm performed better and produced an easier implementation.

5) Camera pose estimation: For estimating the position and rotation of the camera as the video is recorded it is needed to compute the essential matrix. The essential matrix is a 3x3 matrix that relates the points in the first frame with the ones in the second frame, defined as $p_1 E p_2 = 0$, where p_1 and p_2 are the homogeneous points in normalized image coordinates of the first and second frames. For computing this matrix it is needed to have a minimum of 5 correspondences between

Algorithm 2: Feature tracking with KLT algorithm.

input : Two frames and keypoints of the first frame
output: Keypoints matched in the second frame

1 Compute possible directions of the given interest points;
2 Interest point detection in the second frame;
3 Compute errors between first and second frame points;
4 Perform local optimization;
5 Return points with minimum error;

the first and second frame, but as correspondences are not usually perfect, a random based algorithm is used, in this case RANSAC.

RANSAC is an iterative algorithm, which at every iteration it randomly samples five points from the set of correspondences, estimates the Essential Matrix, and then checks if the other points are inliers when using this essential matrix. The algorithm terminates after a fixed number of iterations, and the Essential matrix with which the maximum number of points agree is used.

Once the the Essential matrix is obtained, the following step is to obtain the rotation matrix and the translation vector from it. The relation between the essential matrix, the rotation matrix and the translation vector is $E = R[t]_x$ where R is the rotation matrix and $[t]_x$ is the representation of the cross product with the translation vector t. The trajectory of the camera is stored in a matrix for a later representation.

Algorithm 3: New camera pose estimation.

input : Point correspondences, Intrinsic parameters
output: Disparity Map

1 Select minimal point correspondences at random;
2 Compute the solutions for E;
3 Determine inliers;
4 Compute R and t from E;
5 Update new camera direction as $R_{new} = RR_{new}$;
6 Update new camera position as $t_{new} = t + tR_{new}$;

6) Object detection: The second algorithm that was developed it is an object detection method that detects an object available in the house, in this case a banana. For this task a Haar cascade was used, due to its fast performance and relatively easy implementation. Haar cascade is a machine learning algorithm that serves to train a cascade function to detect objects in images. A Haar cascade it is trained by selecting positive and negative images (as it happens in other machine learning methods) and it need from several images containing the object that we want to detect and way more without the object to obtain a good classifier. For this project an already trained Haar cascade function is used [1].

III. Results

After the camera calibration was performed the intrinsic parameters were obtained as:

$$K = \begin{bmatrix} 1000.50 & 0 & 634.57 \\ 0 & 998.38 & 489.96 \\ 0 & 0 & 1 \end{bmatrix}$$

The results with the feature detection algorithm were plotted into the respective frame of the video to show its performance detecting and tracking the features. In Figure 1, it can be seen the features that were detected in the first frame (upper left image) and the ones that were tracked in the following frames.

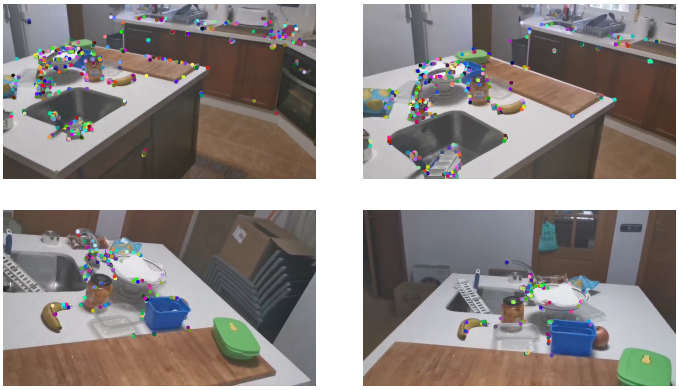

Fig. 1: Feature detection and tracking in different frames.

Using the keypoints detected in Figure 1 the Essential matrix was computed for each trajectory as explained in *Method 5*, giving as result the estimated trajectory seen in Figure 3 and Figure 2.

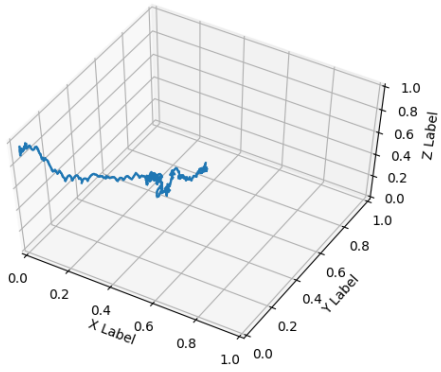

Fig. 2: 3D plot of the camera trajectory.

Regarding the object detection the results seen in Figure 4 were obtained. Further results can be seen in the videos provided with the report to see the feature detection and tracking, and the object detection in action frame by frame.

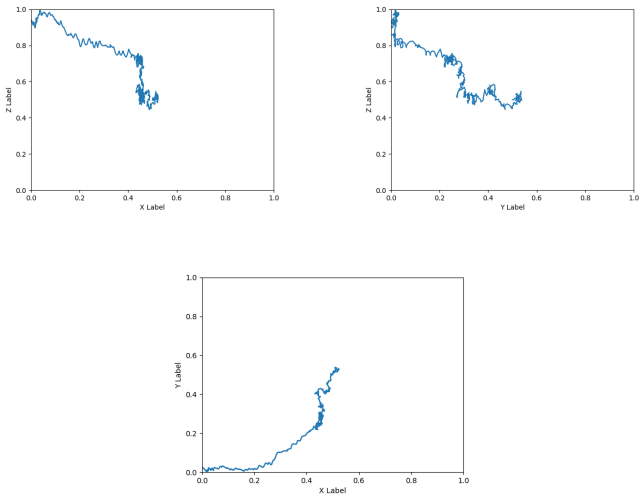

Fig. 3: Camera trajectory seen in the XZ (Upper left) YZ (Upper right) and XY (Bottom) planes.

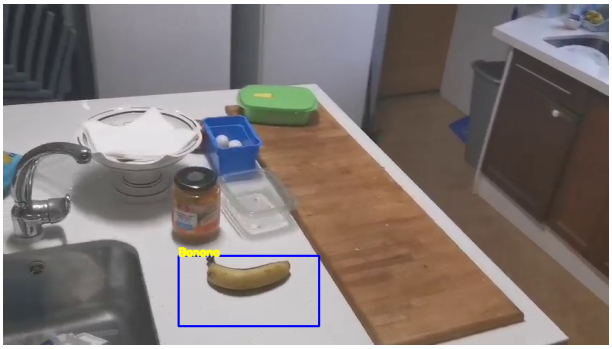

Fig. 4: Banana detection.

IV. Discussion

Starting with the *feature detection* and *feature tracking* results it is seen in Figure 1 that the features detected in the first frame are tracked along the video independently of the rotation and translation of the camera along the trajectory. This shows that the KLT algorithm performs very well in this task and lets very good results for the Essential matrix computation. Nevertheless, the way this method is implemented has a disadvantage, only the points that were initially detected will be tracked in the following steps as only the keypoints of the current and following frame are taken into account. For this reason it can be seen that in the bottom-right picture there are less features detected that in the first one. For solving this problem it could be implemented a local map taking into account the position of the camera rather than the actual method of just comparing 2 sequential frames.

Regarding the *camera pose estimation* it is seen if compared with the video that the trajectory is well recomposed. In the XZ and YZ planes (Upper figures in Figure 3) it is seen that as the camera moves to higher values of X and Y, the Z position gets lower as it happens in the video. The same if we look at the XY plane of the reconstructed trajectory (Bottom picture), it performs a curve in increasing values of X and Y, please note that as it was said the initial position it is stated as [0, 0, 0]. These results could be further improved by using bundle adjustment to eliminate the noise present in the curves, but as it happens to the feature detection issue discussed in the previous paragraph, this wasn't implemented due to time constrains.

Finally, looking at the results in Figure 4, the Haar cascade performs very well once it has a well trained detector. Nevertheless, it can be seen in the video that the detection is not always as perfect as it is shown in Figure 4. Sometimes there are false positives and other times, specially depending in the perspective, the banana it is not correctly detected or not detected at all.

V. Conclusion

In conclusion, the implemented algorithms perform decently and carry out well the tasks of camera trajectory estimation and object recognition. It is seen in the results that the computed trajectory reflects well the reality of the trajectory when filming the video. In addition, the object recognition algorithm detects well the banana in a good part of the times. Overall This project serves to show the potential of these techniques in computer vision and with some further development it could serve in even more complex situations.

VI. Bibliography

[1] https://github.com/mrnugget/opencv-haar-classifier-training/blob/master/trained_classifiers/banana_classifier.xml

[2] ORB-SLAM: a Versatile and AccurateMonocular SLAM System. Raúl M. et al. 2015. IEEE transactions on robotics. DOI: 10.1109/TRO.2015.2463671

[3] OpenCV docs