Visual Odometry with Semantic Segmentation for non-holonomic vehicles

Vincenzo Polizzi

Abstract

One of the main assumption when implementing a Visual Odometry (VO) pipeline is that the environment is considered to be static, the only moving object is the camera. In order to overcome this limitation what is proposed in this report is the use of semantic information to avoid to track features on moving objects. Semantics can also improve feature matching [1] reducing RANSAC iterations and resulting in better performance for the VO pipeline. On the other hand one can argue that in order to retrieve the semantic classes from an image the computational effort is too high, nevertheless it makes sense to use semantic only when already used on the system for other purposes.

1 Introduction

To develop this project a modified version of the Duckietown Gym [2] that includes the semantic annotated images is used. The semantic classes chosen are nine and they are annotated according to a different gray scales as follow:

- LANE EDGE 255
- SIGN HORIZONTAL 204
- VEHICLE 178
- LANE 153
- PEDESTRIAN 127
- BUILDING 102
- NATURE 76
- SIGN VERTICAL 51
- OTHER 0

The classes VEHICLE, PEDESTRIAN or NATURE are considered as out of interest since them refer to moving and changing objects. In particular, not considering NATURE can improve Place Recognition, in the case one wants to implement a SLAM system [3] .

In the following chapters the used method, the data cleaning approach and the results are shown.

2 Method

The following pipeline is implemented:

1. Feature Detection, tracking and cleaning
2. Robot pose and Point Cloud estimation
3. Triangulation with respect to the point cloud

2.1 Feature Detection, tracking and cleaning

In order to make the pipeline light, Shi-Tomasi corners are used in this project. After extracting the features a binary semantic descriptor is found. The semantic descriptor is defined as a list of zeros and ones indicating the presence of a semantic class in a patch around each corner. In particular an order is chosen to create the list, in this specific case "other", "vertical signs", "buildings", "lane", "horizontal signs", "lane edges". This means that if in the patch around the corner the class "other" and the class "lane edges" are present the descriptor is [1, 0, 0, 0, 0, 1]. The patch is represented by a 3 pixels square centered on the corner.

When each feature has its semantic descriptor the Lucas Kanade feature tracker is used [4] to get the correspondences features in the next frame. To the resulting tracked points is then assigned a semantic descriptor too. In order to have a semantic consistency among the correspondences, the semantic descriptors of each feature are compared. Two points are considered as match only if they have the same semantic descriptor, else they are rejected.

2.2 Robot pose and Point Cloud estimation

After the matches are available, the 5-points algorithm is used to retrieve the Essential matrix [5]. Having the essential matrix then the pose of the camera is estimated performing the cheirality check, namely the 3D points need to have a positive depth, in this way it is possible to get the correct orientation of the robot. Once the Rotation matrix and the translation vector are given by the previous passages the 2D points are projected in the 3D world in a resulting point cloud.

2.3 Triangulation with respect to the point cloud

The obtained point cloud is needed to triangulate the robot using non-iterative solution to the PnP problem namely EPNP [6]. When the correspondences between a frame and the point cloud are below a certain threshold the Shi-Tomasi corners are calculated again and the point cloud is reinitialized.

3 Dataset

The dataset is recorded using the open source Duckietown Simulator. It is a gym enviroment that reproduces typical Duckietown cities and Duckiebots.

3.1 Data acquisition

While the robot moves autonomously around the environment the generated images and poses are saved. The semantic ground truth is given thanks to an ad-hoc modified version of the simulator.

4 Results

Unfortunately the code seems not to work really well. In my opinion this can be due to the fact that the tracked features are not enough, since the simulator doesn't show an enough complex texture. What could be done is to test it on a real scenario. Another issue is given by the semantic images, indeed to have good segmented images the edges need to be sharp. This means that while simulating the environment the renderer doesn't have to perform any kind of interpolation. Anyway so far this is one of the main issue with the modified version of the Simulator used. This problem causes the rejection of correct matches.

References

[1] R. Arandjelović and A. Zisserman, "Visual vocabulary with a semantic twist," 11 2014, pp. 178–195.

[2] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull, "Duckietown environments for openai gym," https://github.com/duckietown/gym-duckietown, 2018.

[3] A. Mousavian, J. Košecká, and Jyh-Ming Lien, "Semantically guided location recognition for outdoors scenes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4882–4889.

[4] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," 1999.

[5] D. Nistér, "Nist??r, d.: An efficient solution to the five-point relative pose problem. ieee-tpami 26(6), 756-770," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, pp. 756–77, 07 2004.

[6] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem." *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv81.html#LepetitMF09