Course Seminar

Stere visual odometry - KITTI dataset

- **Exercise goals**
  Study and implementation of a stereo visual odometry algorithm for road vehicles. Using the KITTI dataset for experimental evaluation and validation.
- **Preparation**
  Exercise can be done in any programming language and environment. In order to successfully complete the exercise, study the corresponding lectures on feature detection and tracking (matching), multiview geometry, stereoscopic reconstruction and visual odometry. Feel free to study online materials, open source visual odometry implementations, and use available OpenCV or Matlab libraries - but develop your own pipeline.

Introduction

Localization constitutes a fundamental building block of any autonomous system. This is especially emphasized for autonomous vehicles that participate in urban traffic and need to maintain highly accurate estimates of their pose for navigation purposes. The localization can rely on proprioceptive sensors, like the wheel odometry and inertial measurement units, and exteroceptive sensors like cameras, laser range sensors and even radars. Indeed, localization of autonomous vehicles typically relies on the fusion of most of the aforementioned sensors, but, nevertheless, each modality should operate as accurately as possible to produce a reliably functioning autonomous system.

Visual localization is one of the key actors in modern localization systems for mobile robots and autonomous vehicles. Cameras offer a rich source of information at a fraction of the price of other competitive hardware. All this makes cameras an attractive sensor modality and visual localization has been a subject of research for several decades now [1, 2, 3]. Specifically, in this excercize we will focus on the stereo visual odometry, which is a task of estimating the trajectory of the vehicle using stereo images which can provide depth of the scene by using stereoscopic reconstruction. Stereo odometry for mobile robots and autonomous vehicles still offers many challenges and researchers are continuously striving to develop ever more accurate and reliable methods or leverage novel visual modalities, even though to determine the relative pose between two views only 3 non-colinear 3D points are required.

In the exercise we will use measurements obtained by the vehicle shown in Fig. 1 that was used to record the KITTI dataset developed for benchmarking autonomous vehicle localization algorithms. The KITTI dataset was recorded in urban, rural, and highway scenarios by two stereo pairs (color and grayscale) set at a baseline of 0.54 m. Image rate was 10 Hz, with a resolution somewhat smaller than the original 1392×512 pixels due to rectification. High accuracy OXTS3003 GPS/IMU unit was employed for recording the ground truth trajectory. Odometry part of the dataset contains eleven train tracks with the ground truth, plus eleven test tracks without the ground truth. Test tracks can be evaluated online by uploading resulting trajectories to a designated server. Evaluation results contain translation and rotation error and they can be published and compared on the official KITTI scoreboard.

Since the vehicle was also equipped with a differential GPS setup, which enables high accuracy measurements, we can use this data to evaluate the stereo odometry accuracy. This dataset was and still is an important factor in driving the process of obtaining accurate and robust odometries [4], and has been acting as a public benchmark for road vehicles since 2012. Multiple stereo vision methods currently achieve less than 1% translation error [5, 6, 7, 8, 9, 10, 11, 12], demonstrating the ability of cameras to produce highly accurate road vehicle trajectories. However, do not expect to achieve this level of accuracy in the excercize, as this is a long-term process.

Exercise

As stated earlier, the KITTI dataset is divided in two groups of tracks: the traing and test tracks. For

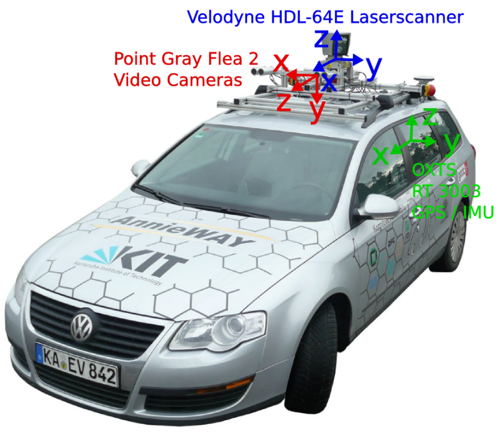

Figure 1: Vehicle used for recording the KITTI dataset developed to benchmark autonomous vehicle localization algorithms

the exercise, pick one of the tracks from the train set and work on it during the exercise. Note that the dataset contains rectified stereo images that were obtained through the calibration process, thus for camera parameters, use the default values given in the dataset. You do not need to rectify the images yourself.

You will be working with stereo images, where $I_{l,k}, I_{r,k}$ represent the left and right image, respectively. The goal will be to reconstruct the pose of the camera C_k, given a series of estimated relative transformations using the odoemtry $T_k, T_{k-1}, \dots, T_1$. To estimate the relative transformation consider two groups of approaches: 2D-3D approaches and 3D-3D approaches.

The 3D-3D approach estimates motion from 3D-to-3D feature correspondences - it is also called the point cloud registration problem). To do this, one first must triangulate the 3D feature points (e.g., using stereoscopic reconstruction or some other depth estimation algorithm, see also semi-global matching (SGM)). The minimal-case solution in this case requires 3 non-colinear correspondences. In general, the goal is to find the T_k that minimizes the following distance between the two 3D features sets

$$\text{argmin}_{T_k} \sum_i ||X_k^i - T_k X_{k-1}^i||^2, \tag{1}$$

where the supersript i denotes the i-th feature, and X_k, X_{k-1} are the matched 3D points. There are multiple approaches to solving the point cloud registration, e.g., the "Least-Squares Fitting of Two 3-D Point Sets", published by A. S. Arun, T. S. Huang and S. D. Blostein in 1987, or see any of the available iterative closest point (ICP) algorithm implementations in OpenCV or Matlab libraries.

The pipeline of a 3D-3D stereo odometry might have the following form

1. Capture two stereo image pairs $I_{l,k-1}, I_{r,k-1}$ and $I_{l,k}, I_{r,k}$
2. Extract and match features between $I_{l,k-1}, I_{l,k}$ (this way you will know correspondences between two stereo frame features)
3. Match and triangulate matched features for each stereo pair
4. Compute the relative transformation T_k from 3D feature sets X_{k-1} and X_k
5. Concatenate camera transformation by computing $C_k = C_{k-1}T_k$.

The 3D-2D approach on the other hand, has been found to yield more accurate results than 3D-3D correspondences because it minimizes the image reprojection errorinstead of the 3D-to-3D feature position error. In this case, the relative transformation T_k is computed from the 3D-to-2D correspondences X_{k-1} and p_k, where p_k represents the set of 2D image points at frame k. In our case, the 3D set X_{k-1} can be computed from the stereo image pair. In general, we are tasked with finding T_k that minimizes the so-called image reprojection error

$$\text{argmin}_{T_k} \sum_i ||p_k^i - \hat{p}_{k-1}^i||^2, \tag{2}$$

where $\hat{p}^i_{k-1}$ is the reprojection of the 3D point X^i_{k-1} into image I_k according to the transformation T_k. For the stereo camera, we can reproject the 3D points from frame $k-1$ to either left or right images from frame k. This problem is known as perspective from n points (PnP), and again there are many different solutions in the literature. The minimal case involves three 3D-to-2D correspondences and this is called perspective from three points (P3P) and returns four solutions that can be disambiguated using one or more additional points. P3P is the standard method for robust motion estimation in the presence of outliers.

The pipeline of a 3D-2D stereo odometry might have the following form

1. Capture two stereo image pairs $I_{l,k-1}, I_{r,k-1}$ and $I_{l,k}, I_{r,k}$
2. Extract and match features between $I_{l,k-1}, I_{l,k}$
3. Match and triangulate matched features for $I_{l,k-1}, I_{r,k-1}$
4. Compute the relative transformation T_k using PnP from 3D-to-2D matches of X_{k-1} and $\hat{p}_{k-1}$ in $I_{l,k}$
5. Concatenate camera transformation by computing $C_k = C_{k-1}T_k$.

For your stereo odometry implementation pick one of the two methods. You can use the available libraries and functions for feature detection and matching, robust essential matrix estimation, depth reconstruction, robust P3P etc. But you have to implement your own pipeline, it is not allowed to use an existing function for stereo odometry estimation.

By concatenating the relative transforms you will reconstruct the trajectory of the vehicle. Then the next step is to evaluate the accuracy of your algorithm by using the ground truth data. For evaluation study the following odometry and SLAM evaluation toolboxes and use either of the two: evo and rpg_trajectory_evaluation.

Exercise submission

Write a report explaining your approach and commenting the obtained results. The report in single-column style should be maximum 10 pages long. Submit it along with a zip file containing your source code and instructions for running your package.

Good luck!

Bonus stuff

If you feel inspired to do so, you can additionally try to solve some the following challenges:

1. Implement the bundle adjustement algorithm on a window of camera poses (e.g., 3 to 5 poses)
2. Additionally to the camera poses, build the map of the environment
3. Some tracks contain the so-called loop closures. Detect them using an approach of your choice; commonly the so-called bag-of-words approach is used.
4. To have simulatenous localization and mapping algorithm, after detecting the loop closure (in this case you can opt to do it by hand), run a so-called pose graph optimization algorithm that will based on the introduced constraint optimize over all the previous poses and you should obtain a more accurate trajectory. Libraries for this step include GTSAM, iSAM2, g2o and Ceres.
5. Try to implement a monocular odometry using only left camera images. In this case your results will be up to scale and usually during the evaluation process a least squares fit to the ground truth is executed in order to align the scale as best as possible to the ground truth before computing the trajectory error.

References

[1] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[2] D. Scaramuzza and F. Fraundorfer, “Visual odometry: Part I: The First 30 Years and Fundamentals,” *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[3] F. Fraundorfer and D. Scaramuzza, "Visual odometry part II: matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[5] I. Cvišić and I. Petrović, "Stereo odometry based on careful feature selection and tracking," *2015 European Conference on Mobile Robots, ECMR 2015 - Proceedings*, pp. 0–5, 2015.

[6] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 3903–3911.

[7] J. Zhu, "Image gradient-based joint direct visual odometry for stereo camera," in *IJCAI International Joint Conference on Artificial Intelligence*, 2017, pp. 4558–4564.

[8] K. Lenac, J. Ćesić, I. Marković, and I. Petrović, "Exactly Sparse Delayed State Filter on Lie groups for Long-term Pose Graph SLAM," *International Journal of Robotics Research*, vol. 37, no. 6, pp. 585–610, 2018.

[9] I. Cvišić, J. Ćesić, I. Marković, and I. Petrović, "SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs," *Journal of Field Robotics*, vol. 35, no. 4, pp. 578–595, 2018.

[10] M. Buczko, V. Willert, J. Schwehr, and J. Adamy, "Self-Validation for Automotive Visual Odometry," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 573–578.

[11] P. Bénet and A. Guinamard, "Robust and Accurate Deterministic Visual Odometry," in *International Technical Meeting of the Satellite Division of The Institute of Navigation (ION + GNSS)*, 2020, pp. 2260 – 2271.

[12] M. Ferrera, A. Eudes, J. Moras, M. Sanfourche, and G. Lebesnerais, "OV^{2}SLAM : A Fully Online and Versatile Visual SLAM for Real-Time Applications," *IEEE Robotics and Automation Letters*, 2021.