MVG - II, III Assignment Questions

1 [MVG] Visual Odometry

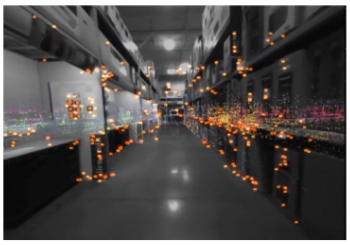
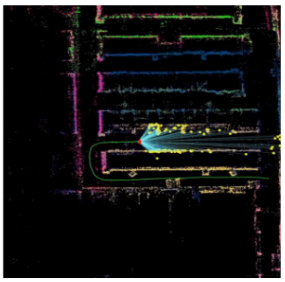


Image sequence (or video stream)
from one or more cameras attached to a moving vehicle



Camera trajectory (3D structure is a plus)

$R_0, R_1, \ldots, R_i$

$t_0, t_1, \ldots, t_i$

1.1 Introduction

Visual odometry is the process of recovering the egomotion (the trajectory) of an agent using only the input from the camera or a system of cameras attached to the agent. This is a well-studied problem in robotic vision and is a critical part of many applications such as Mars rovers and self-driving cars for localization. In this assignment, you will implement a basic monocular visual odometry algorithm. All the theory needed for attempting this problem is covered in Lectures 32–37 from Cyrill's *Mobile Sensing and Robotics II* (MSR-II) playlist: [YouTube – MSR-II].

To begin, download all the required data from the drive linked - (click here). It contains a sequence of images from the KITTI dataset (you'll have to use `ffmpeg` on kitti-sample.mp4 and extract the 801 source frames). The ground truth pose of each frame (in row-major order) and the camera parameters are provided as well in the parent directory.

1.2 Procedure

The following is an overview of the entire algorithm:

1. Find the corresponding features between frames I_k and I_{k-1}.
2. Using these feature correspondences, estimate the essential matrix between the two images within a RANSAC scheme.
3. Decompose the essential matrix to obtain the relative rotation R_k and translation t_k, and form the transformation T_k.
4. Scale the translation t_k with the absolute or the relative scale.
5. Concatenate the relative transformation by computing $C_k = C_{k-1}T_k$, where C_{k-1} is the previous pose of the camera in the world frame.
6. Repeat the above steps for the remaining pairs of frames.

The main task in computing visual odometry is to compute the relative transformation T_k from each pair of images I_k and I_{k-1} and then concatenate these transformations to recover the full trajectory of the camera.

There are two broad approaches to compute the relative motion T_k:

- **Appearance-based (or direct) method**: uses the intensity information of all the pixels in the input images.
- **Feature-based method**: uses salient and repeatable features extracted and tracked across the images.

You will be implementing a feature-based method.

1.3 Implementation Details

For every new image I_k, the first step consists of detecting and matching 2D features with those from the previous frame. These 2D features are locations in the image that can be reliably found and matched across multiple images. To find these features, use the inbuilt SIFT OpenCV implementation.

The main task is motion computation. Using these feature correspondences, implement the 8-point algorithm for fundamental matrix estimation. Implement it inside a RANSAC scheme to get rid of outliers.

Then, compute the essential matrix, and decompose it into the relative R and t using `cv2.recoverPose`. Note that this function returns the R and t of the first camera with respect to the second, and not the other way around.

Since the absolute scale of the translation cannot be computed from just two images, the function only returns the direction of t as a unit vector. Use the ground truth translation to get the absolute scale, and multiply your unit translation with this scale. Then concatenate your transformations and repeat for the next pair of frames to recover the full absolute trajectory.

1.4 Deliverables

- A `.txt` file containing the estimated poses, provided in the same format as the ground truth.
- A plot of the estimated trajectory along with the ground truth trajectory.
- Report the obtained trajectory error. Use EVO for this:

```
pip install evo --upgrade --no-binary evo
evo_traj kitti ground-truth.txt your-result.txt -va --plot --plot_mode xz
```

- Comment on the performance of your algorithm. When does it work well or fail, and why?

2 [Bonus/Research] In-The-Wild Visual Re-localization

You are tasked with Monocular 6-DoF Camera Re-localization that must work in any general scenario. Id est, you've been to a place before and have had the chance to capture, with your phone's (singular!) camera, the scene in detail (you already have a powerful SLAM algorithm embedded in the phone you carry!). You now need to develop an algorithm that can deduce the full 6-DoF (rotation and metric translation) of a new device that enters this location at a different time. There's various reasons why such a system would be useful - for robot localization, displaying 3D visualizations within a shared augmented reality platform. The algorithm needs to work very generally given any scene - so you may not use markers or any other sensors placed withing the environment. The solution must purely depend on being able to process the original (mapping) camera run and being able to robustly register your current (query) camera run. Feel free to make any assumptions.

- What front-end/feature extraction would you use ?
- How would the registration algorithm look like ?
- Are there any published works that already tackle this ? Collate these resources.

Good luck with the assignment!