ENPM673 - Perception for Autonomous Robots

Project 5

Prateek Arora, Abhinav Modi, Samer Charifa

Due date: 4^{th} *May 2020, 11:59 p.m.*

Submission Guidelines

- The homework is to be completed in group and there should be a single submission per group.
- Your submission on ELMS/Canvas must be a zip file, following the naming convention **YourDirectoryID_proj_5.zip**. If you email ID is abc@umd.edu or abc@terpmail.umd.edu, then your DirectoryID is **abc**. Remember, this is your directory ID and **NOT** your UID. Please provide detailed instructions on how to run your code in README.md file.
- For each section of the homework, explain briefly what you did, and describe any interesting problems you encountered and/or solutions you implemented. Your report should preferable be typeset in LaTeX.
- Please submit only the python script(s) you used to compute the results, the PDF report you generate for the project and a detailed README file (refer to the directory structure below).
- Include sample outputs in the your report.
- The video outputs are to submitted as a separate link in the report itself. The link can be a YouTube video or a google drive link (or any other cloud storage). Make sure that you provide proper sharing permission to access the files. **If we are not able to access the output files you will be awarded 0 for that part of the project**.
- Please don't include the dataset provided to you and your video results in your submission, rather provide the link to your video output in the report.
- Disallowed function:
  - any inbuilt function from scipy or sklean that directly implements a tracker.
  - any other inbuilt function that solves the question in less than 5 lines.

The file tree of your submission SHOULD resemble this:

```
YourDirectoryID_hw1.zip
├── Code
│   ├── .py files
│   └── any subdirectories that you may have
├── Report.pdf
└── README.md
```

Data

Link to the Dataset

Visual Odometry

Visual Odometry is a crucial concept in Robotics Perception for estimating the trajectory of the robot (the camera on the robot to be precise). The concepts involved in Visual Odometry are quite the same for SLAM which needless to say is an integral part of Perception.

In this project you are given frames of a driving sequence taken by a camera in a car, and the scripts to extract the intrinsic parameters. Your will implement the different steps to estimate the 3D motion of the camera, and provide as output a plot of the trajectory of the camera.

Data Preparation

First you have to prepare the data using the following steps:

1. The input images are in Bayer format from which you can recover the color images using the demosaic function with GBRG alignment. That is, convert the Bayer pattern encoded image img to a color image using:

   ```
   color image = cv2.cvtColor(img, cv2.COLOR_BayerGR2BGR)
   ```

2. Extract the camera parameters using ReadCameraModel.py as follows:

   ```
   fx, fy, cx, cy, G camera image, LUT = ReadCameraModel('./model')
   ```

3. Undistort the current frame and next frame using UndistortImage.py as follows:

   ```
   undistorted image = UndistortImage(original image,LUT)
   ```

The basic Pipeline [100 Points]

To estimate the 3D motion (translation and rotation) between successive frames in the sequence, implement the following steps below. A good description of the individual steps can be found at this page (steps 3.1-3.5.1) :

1. Find point correspondences between successive frames using a keypoint algorithm of your choice.
2. Estimate the Fundamental Matrix using the Eight-Point Algorithm within RANSAC
   (a) Center the image data at the origin and scale the data according to the 8-Point Algorithm
   (b) Estimate the Fundamental matrix F via RANSAC from point correspondences.
   (c) Enforce the rank 2 constraint on F.
3. Estimate the Essential E matrix from the Fundamental matrix F by accounting for the calibration parameters.
4. Decompose E into a translation T and a rotation R. There will be four possible solutions.
5. Find the correct T and R from the depth positivity. That is, estimate for each of the four solutions the depth of all points linearly using the cheirality equations. Choose the R and T which gives the largest amount of positive depth values.
6. Plot the position of the camera center (for each frame) based on the rotation and translation parameters between successive frames.

You should implement the functions to estimate the Essential Matrix and also to recover the rotation/translation matrices and NOT use opencv's built-in functions or any third party code.

Comparison [extra credit upto 20 points]

Compare your result against the rotation/translation parameters recovered using cv2.findEssentialMat and cv2.recoverPose from opencv. Plot both the trajectories and report the accumulated drift in trajectory from your implementation versus the one you computed with the use the opencv functions.

Additional Steps in the Pipeline [extra credit 30 Points]

For a better estimate, solve non-linearly for the depth and 3D motion using the so-called algebraic error.

1. First, write code that estimates the depth using non-linear triangulation (Section 3.5.1 in this description.)
2. Second, write the code that estimates the rotation and translation non-linear using the Nonlinear PnP. (Section 3.6.3. in this description.)

Useful Resources

A video lecture on the Estimation of the Fundamental and the Essential Matrix

Acknowledgement

The Dataset used is by courtesy of Oxford's Robotics Institute.