CS 3630 : Assignment 5: Visual Odometry

Deadlines:

- **Check point (Part 1):** Thurs, March 26th 11:55pm via T-Square.
- **Final Assignment Due (Part 2)**: Thurs, April 2nd 11:55pm via T-Square.
- Assignment has to be done in groups of **2**.
- Mention your **group member names** in the submission.

It is absolutely not allowed to share your source code with anyone in the class as well as to use code from the Internet. If you have any questions, ask on Piazza or in class. **Do not give out answers on Piazza!**

Introduction

In the previous assignment, we used a combination of odometry measurements and range-bearing measurements to estimate the robot trajectory and location of the AprilTags. Instead of solving the SLAM problem in this assignment, we are going to use the camera measurements to compute the odometry of the robot without the use of the vehicle commands. This process of estimating the motion of a camera is known as **visual odometry**.

1 Data Collection (Checkpoint)

In the first part of the assignment, you'll drive the robot and record some datasets in a natural environment (no AprilTags). Record two or three different datasets and make sure you get at least a couple where you have a bit of curve. Since visual odometry is computed using only the pictures taken by the robot, make sure that you take enough pictures along the trajectory to accurately estimate the robot's motion. You can use **logDataCalico.py** to log the data. An example dataset is also attached with the assignment. Some other things to keep in mind are:

1. Ensure that consecutive images should have enough features in common to estimate the robot motion.
2. As suggested in the previous assignment, the pictures should be taken when the robot is not moving to avoid rolling shutter.
3. Constant illumination across images helps feature matching.

1.1 Writeup

1. Make a video of the trajectory taken by the robot (in real environment) when collecting the datasets.
2. Provide a plot of the robot motion using just the motor commands for each trial. You can use these for comparison to the visual odometry solutions. Code from last assginment's EKF motion prediction step can be used to plot robot's trajectory given motor commands.

2 Visual Odometry (Final Submission)

Visual odometry consists of the following main steps: (1) finding features in each image and matching the corresponding features between consecutive images, (2) verifying the correspondences and removing the outliers using RANSAC algorithm, (3) converting the resulting fundamental matrix to essential matrix using the intrinsic calibration parameters of the Fluke camera, (4) estimating relative transformation from essential matrix and (5) finally integrating the transformations to infer the robot trajectory (similar to EKF motion update step). Each of the steps are described in detail below.

2.1 Feature Matching

It is the problem of finding the pixel coordinates in two different images that correspond to the same point in the world. Many of the pixels like cloud, walls are insufficiently distinct. So, we find interest points which are repeatable across view point and illumination changes and and match those interest points between two images. Feature descriptors are used to describe the region around the interest points and are used in the matching algorithm. The general feature matching algorithm follows the structure:

1. For each feature in the first image
   (a) Find the nearest neighbor (closest) feature descriptor in the second image
   (b) If the distance to the nearest neighbor is less than a threshold, consider it as a match.

2.2 Geometric verification using RANSAC

In general threshold based correspondence estimation contains many outliers and we use RANSAC algorithm to remove the outliers and estimate the **fundamental matrix**. Estimating a fundamental matrix requires eight points so we randomly choose eight candidates correspoinding points and estimate F to create a model. This model is tested against all the other candidate pairs and those that fit, vote for this model. The process is repeated a number of times and the model that had the most supporters (the consensus) is returned. The feature-pairs that support the model aare termed inliers and those that do not are outliers.

2.3 Estimating essential matrix

Given the fundamental matrix F, we can estimate the essential matrix E using the equation:

$$E = K^T F K$$

where K is the intrinsic matrix.

Parameter	Value
f_x	1211.2959
f_y	1206.00512
s	0.0
u	657.15924
v	403.17667

Table 1: Intrinsic Parameters

You can use the values mentioned in Table 1 to specify the intrinsic matrix. Or you can estimate the intrinsic parameters using the camera calibration toolbox (`http://www.vision.caltech.edu/bouguetj/calib_doc/`). Calibration is recommended since the intrnsic parameters might slightly differ from the one specified in Table 1.

2.4 Estimating relative transformation from Essential Matrix

Given the essential matrix, we have to estimate the relative transformation between the consecutive images for it to be useful for visual odometry. Section **14.2.2** of the textbook (Peter Corke) discusses this in detail.

2.5 Integrating relative transformations

Relative transformation can then be integrated using EKF motion update equations as used in the last assignment.

Deliverable

1. Implement and discuss each of the above steps in detail and show the corresponding results for each step. Provide screenshots of your visualizations to support your discussion
2. Compare the trajectory given by visual odometry to the trajectory given by using the motor commands and give reasons behind the difference.