Project 3: Visual-Inertial SLAM

*Collaboration in the sense of discussion is allowed, however, the assignment is **individual** and the work you turn in should be entirely your own. It is absolutely forbidden to copy or even look at anyone else's code. Please acknowledge **in writing** people you discuss the project with. See the collaboration and academic integrity statement here:* https://natanaso.github.io/ece276a.

Submission

Please submit the following files on **Gradescope** by the deadline shown at the top right corner.

1. **Programming assignment**: upload all code you have written for the project (do not include the provided datasets) and a README file with a clear, concise description of the main file and how to run your code.
2. **Report**: upload your report in pdf format. You are encouraged but not required to use an IEEE conference template[1] for your report.

Project Description

The goal of this project is to implement visual-inertial simultaneous localization and mapping (SLAM) using an extended Kalman filter (EKF). You are provided with measurements from an inertial measurement unit (IMU) and a stereo camera as well as the intrinsic camera calibration and the extrinsic calibration between the two sensors, specifying the transformation from the left camera frame to the IMU frame. We will use data obtained by Clearpath Jackal robots navigating on MIT's campus in Cambridge, MA (see Fig. 1). The following data are provided at https://ucsdcloud-my.sharepoint.com/:f:/g/personal/natanasov_ucsd_edu/ElKB_H_Rfu9GuA38wucC_UEBMB0HKK4Bs-339Mq-p-F5Ew:

- **IMU measurements**: linear velocity $\mathbf{v}_t \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$ of the body with coordinates expressed in the body frame of the IMU.
- **Time stamps**: τ_t in UNIX time (seconds since January 1, 1970).
- **Camera measurements**: grayscale images with pixel length and width (600×480) are provided separately in datasetXX_imgs.npy.
- **Visual feature measurements**: pixel coordinates $\mathbf{z}_t \in \mathbb{R}^{4\times M}$ of detected visual features from M point landmarks with precomputed correspondences between the left and the right camera frames (see Fig. 2). Landmarks i that were not observable at time t have measurement $\mathbf{z}_{t,i} = \begin{bmatrix}-1 & -1 & -1 & -1\end{bmatrix}^\top$, indicating a missing observation.

  Note: There are many features in each dataset and not all may be useful for SLAM. You may holistically come up with an algorithm that filters out "bad" points and think of ways to keep the computational complexity manageable.

  Videos showing the feature tracking over time (just for illustration purpose) are provided in the drive. It may be helpful to use the footage as a reference for the ground truth landmarks.
- **Extrinsic calibration**: transformation ${}_CT_I \in SE(3)$ from the IMU frame to the left and right camera. The IMU frame is oriented as x = forward, y = left, z = up.
- **Intrinsic calibration**: values for the camera calibration matrix for the left and right camera:

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}.$$

  The stereo baseline must be deduced from the given extrinsic calibration matrices.

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html

Figure 1: Clearpath Jackal robots on MIT's campus.

Note: You may notice that the stereo camera setup may not be perfect horizontal. This means when you project your landmark points to the optical frame, there will be a very small margin of error. Although negligible, you may consider individually projecting the landmark points to each camera separately using their respective camera calibration matrix and extrinsic pose instead of constructing a stereo-camera projection matrix K_s.

Implement an EKF prediction step based on $SE(3)$ kinematics with IMU measurements and an EKF update step based on stereo-camera observation model with visual-feature observations to perform SLAM. In detail, you should complete the following tasks.

1. [15 pts] **IMU localization via EKF prediction**: Implement an EKF prediction step based on the $SE(3)$ kinematics equations and the given linear velocity and angular velocity measurements from the IMU to estimate the pose $T_t \in SE(3)$ of the IMU over time t.

2. [10 pts] **Feature detection and matching** (**Optional part for extra credit**): Use dataset02, which does not provide features. You need to perform feature detection and tracking on your own.

   We are going to prepare the feature tracks that will be used for landmark mapping. The goal is to obtain the pixel coordinates $\mathbf{z}_t \in \mathbb{R}^{4\times M}$ of detected visual features with correspondences between the left and the right camera frames, and then track the detected features in the left image over time (see Fig. 2). For each timestep t, create a matrix $\mathbf{z}_t \in \mathbb{R}^{4\times M}$, where M is the total number of features in all of the images and each column corresponds to a specific feature. Formatted as follows:

$$\mathbf{z}_t = \begin{bmatrix} l_{x,1} & l_{x,2} & l_{x,3} & \dots & l_{x,M} \\ l_{y,1} & l_{y,2} & l_{y,3} & \dots & l_{y,M} \\ r_{x,1} & r_{x,2} & r_{x,3} & \dots & r_{x,M} \\ r_{y,1} & r_{y,2} & r_{y,3} & \dots & r_{y,M} \end{bmatrix}$$

   where

   - $l_{x,j}$ is the x-pixel-coordinate of the j-th feature of the left camera
   - $l_{y,j}$ is the y-pixel-coordinate of the j-th feature of the left camera
   - $r_{x,j}$ is the x-pixel-coordinate of the j-th feature of the right camera
   - $r_{y,j}$ is the y-pixel-coordinate of the j-th feature of the right camera

   Of course, some features will not be present in all of the frames. Therefore, if the j-th feature is not in the t-th frame, then set $l_{x,j} = l_{y,j} = r_{x,j} = r_{y,j} = -1$. Furthermore, you may want to filter out some feature points in order to improve the observation model when performing the EKF SLAM.

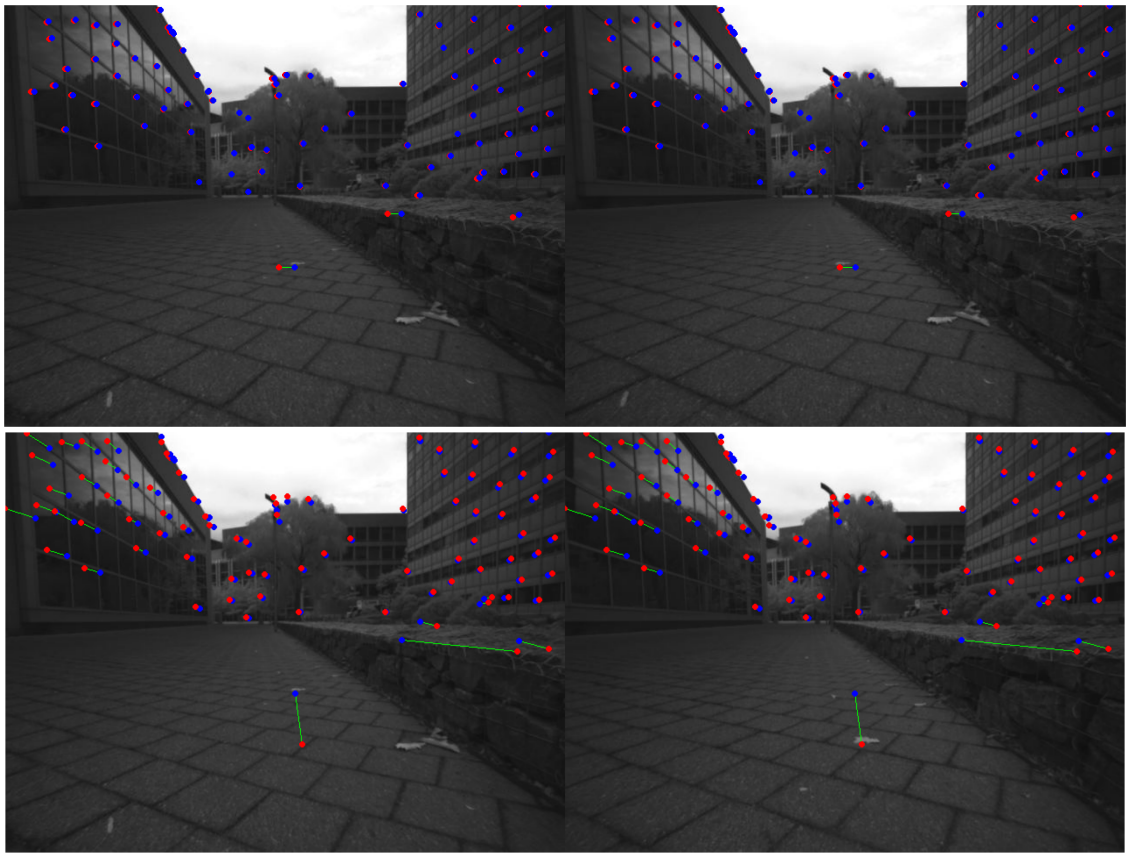

Figure 2: Visual features matched across the left-right camera frames (top) and across time (bottom). Blue features correspond to the left image and red features correspond to the right image (top) and otherwise (bottom) blue corresponds to features at time t and red corresponds to features at $t + 15$. The green lines represent the translation between the features.

Use OpenCV to track features that are common between the two frames across different time stamps. Feel free to use any function from OpenCV to aid your algorithm. Some useful ones are Shi-Tomasi Corner Detection and Optical Flow. Below are some suggested specific steps.

(a) [5 pts] Perform stereo matching using the features detected on the left image. Use optical flow to track the features in the left image to the right image to find correspondences. You are allowed to use the OpenCV function *goodFeaturesToTrack* and *calcOpticalFlowPyrLK*. You are also allowed to refer to *image_processor.cpp*[2], which is the feature tracking part implemented in C++. However, you are *not* allowed to directly copy from any other Python repositories on the internet. Performing RANSAC is optional.

(b) [5 pts] Perform temporal feature tracking using the features detected in the left image to the left image in the next time instance, using optical flow. You are allowed to use the OpenCV function *calcOpticalFlowPyrLK*.

3. [15 pts] **Landmark mapping via EKF update**: Assume that the predicted IMU trajectory from part (a) is correct and focus on estimating the landmark positions $\mathbf{m} \in \mathbb{R}^{3\times M}$ of the landmarks observed in the images. There are many visual feature measurements in the dataset but you do not need to use all to obtain good results. You can think of ways to keep the computational complexity manageable. You should implement an EKF with the unknown landmark positions $\mathbf{m} \in \mathbb{R}^{3\times M}$ as a state and perform EKF update steps using the visual observations $\mathbf{z}_t$ to keep track of the mean and covariance of $\mathbf{m}$.

[2] https://github.com/KumarRobotics/msckf_vio/blob/master/src/image_processor.cpp

We are assuming that the landmarks $\mathbf{m}$ are static, and so it is not necessary to implement an EKF prediction step for them. Since the sensor does not move sufficiently along the z-axis, the estimation of the z coordinate of the landmarks will not be very good. This is expected and you should not worry about it. Focus on estimating the landmark x and y coordinates well.

Since you are initializing a very large and sparse matrix for your landmark covariance, it might be useful to use the `csr_matrix` or `lil_matrix` class from SciPy to accelerate matrix calculation.

4. [20 pts] **Visual-inertial SLAM**: Combine the IMU prediction step from part (a) with the landmark update step from part (b) and implement an update step for the IMU pose $T_t \in SE(3)$, based on the stereo-camera observation model, to obtain a complete visual-inertial SLAM algorithm.

   It is your job to tune the noise in order to obtain the best possible trajectory. The IMU trajectory will only be a decent estimate, and you want to use the estimated landmarks to correct the IMU positions as you perform SLAM!

Project Report

Write a project report describing your approach to the visual-inertial SLAM problem. Your report should include the following sections.

- [5 pts] **Introduction**: Discuss what the problem is, why it is important, and present a brief overview of your approach.
- [10 pts] **Problem Formulation**: State the problem you are trying to solve in mathematical terms. This section should be short and clear and should rigorously define the quantities you are interested in, i.e., data, input, and desired output. However, this section should not present your solution.
- [20 pts] **Technical Approach**: Describe your approach to visual-inertial SLAM.
- [15 pts] **Results**: Present your results, and discuss them — what worked, what did not, and why. Make sure your results include plots clearly showing the estimated IMU trajectory as well as the estimated x and y positions of the visual features. If you have videos, please include them in your submission zip file and refer to them in your report.