# 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. See the collaboration and academic integrity statement here:* https://natanaso.github.io/ece276a. *Books, websites, and papers may be consulted but not copied from. It is absolutely forbidden to copy or even look at anyone else's code. Please acknowledge **in writing** people you discuss the project with and provide references for any books or papers you use.*

## 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.

## Problems

In square brackets are the points assigned to each part.

1. Implement visual-inertial simultaneous localization and mapping (SLAM) using an extended Kalman filter (EKF). You are provided with synchronized 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. The following data are provided.
   - **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.
   - **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. 1). Landmarks $i$ that were not observable at time $t$ have a measurement of $\mathbf{z}_{t,i} = \begin{bmatrix}-1 & -1 & -1 & -1\end{bmatrix}^\top$, indicating a missing observation.
   - **Time stamps**: $\tau_t$ in unix time (seconds since January 1, 1970).
   - **Intrinsic calibration**: stereo baseline $b$ in meters and camera calibration matrix:
   $$K = \begin{bmatrix} fs_u & 0 & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix}.$$
   - **Extrinsic calibration**: transformation ${}_I T_C \in SE(3)$ from the left camera to the IMU frame. Note that the IMU sensor is placed **upside down** on the vehicle (see Fig. 2) so the IMU frame is oriented as x = forward, y = right, z = down. When estimating the IMU trajectory, you may leave the IMU frame orientation as is or rotate around the x-axis to obtain a regular coordinate frame.

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

Figure 1: Visual features matched across the left-right camera frames (left) and across time (right).

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.

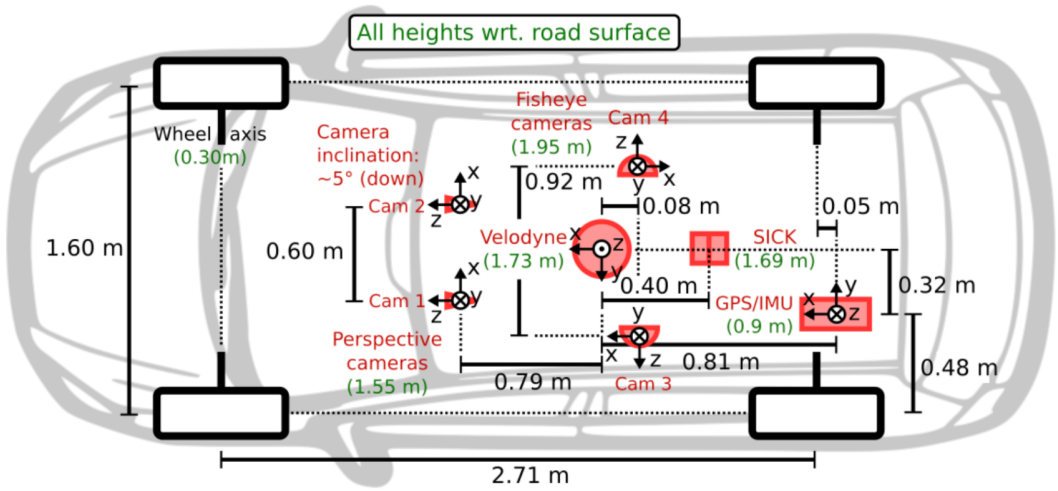
(a) [15 pts] **IMU localization via EKF prediction**: Implement an EKF prediction step based on the $SE(3)$ kinematics equations and the linear and angular velocity measurements from the IMU to estimate the pose $T_t \in SE(3)$ of the IMU over time $t$.

(b) [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 meaasurements 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 $\mathbf{m}$. 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.

(c) [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.

2. 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 but 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, do include them in your submission zip file and refer to them in your report.




As illustrated in the figure, the coordinate systems of the sensors are defined as follows:

- Perspective cameras (`image_00`, `image_01`): x = right, y = down, z = forward
- Left fisheye camera (`image_02`): x = forward, y = down, z = left
- Right fisheye camera (`image_03`): x = backward, y = down, z = right
- Velodyne: x = forward, y = left, z = up
- GPS/IMU: x = forward, y = right, z = down

Figure 2: Sensor Configurations