FORMULA ELECTRIC AT BERKELEY

Autonomous Software Recruitment Project

**Note: This is an individual project and must be completed independently. You are, however, encouraged to get help from outside resources! This is meant to be a learning experience - feel free to reach out to us, your peers, read research papers, and have fun with it! Of course, please let us know if you have any questions.*

***Note**: This project is **difficult**. Do not let that stop you from trying! Building an autonomous car is incredibly difficult, but nobody on this team knew what they were doing when they first started. In order to do extraordinary things, you must start by doing something difficult, and realizing just how much you are truly capable of. We all started on a daunting recruitment project, and have come a long way since then… It's time for you to start your journey!*

Office Hours

Office Hours: OH Sheet

Note: These office hours are for both weeks, and please contact us privately to determine an alternative if these office hours are not possible for you. These times are meant to be open for you to get any help that you need on the project, discuss your progress, and work through any questions that you should have. We will NOT debug your code, but we will give perspective, explain some of the basic concepts, help learn to use solvers, help understand the project, challenge your understanding, give comfort in your time of need, etc.

Program Objectives

Our goal is to make the FEB car one of the first FSAE cars in the US to race autonomously. Our European FSAE counterparts (Formula Student Germany, Formula Student Italy, Formula Student Austria, and so on) have mandated that all teams must be driverless since the start of this past year's competition. As such, FEB's objective is to be on a development timeline that is ahead of the curve in comparison to the majority of our fellow US teams (bar fellow visionaries such as CMU Autonomous). By starting sooner rather than later, we open up the possibility for a greater competitive advantage once the inevitable rules switch takes place here in the US.

We currently have an **entire software pipeline** working in the EUFS simulator! We've been spending all summer running tests and benchmarks to prepare us to run the car autonomously, and as safely as possible. While we have a functioning pipeline, there's much room for improvement in both efficiency and accuracy in our controls, state estimation, perception, and hardware.

We're looking for team members who can not only design future software to be faster and more accurate, but will also see their software through to the finish line, and implement it onto the car, to interact with the cars mechanical systems and connect in with the rest of the pipeline.

We do not expect you to be able to have the knowledge and skills to be able to do this right away, but we do expect you to be passionate, ask questions, dedicate lots of time, and constantly learn more and more about autonomous driving. Even our leads are constantly researching new algorithms and methods for state estimation, perception, etc.

Project Components within the Autonomous Driving R&D Team

Perception - Ensure accurate, fast, and robust perception during high speed racing, which includes sensors like camera and LiDAR, while dealing with noisy, uncertain, and incomplete information.

SLAM (Simultaneous Localization and Mapping)/Localization/Mapping/State Estimation - Responsible for accurately estimating vehicle pose and vehicle kinematic state using a combination of GPS, IMU, camera, etc.

Path Planning/Controls - Develop multi-agent planning algorithms to maneuver around other vehicles at similar speeds. Use traditional control as well as model predictive control to find safe (yet aggressive) acceleration and steering commands.

Simulation - Responsible for designing and executing varied and comprehensive race scenarios, and unit testing across the entire system.

Integration - Ensure the integrity and dependability of our entire infrastructure, including ROS, environment, GCP, testing.

Position Background & Overview

The Autonomous Driving R&D Team's role is critical to the success of the team and requires a significant time commitment. This (non-exhaustive) list details the requirements & responsibilities we are looking for in potential candidates:

- An ability to commit at least 8-10 hours a week to FEB
- A demonstrated ability to design and implement part of or an entire autonomous driving system
- A wide knowledge base across all facets of an autonomous driving system
- Creative, open-minded, and curious with the will to learn, expand your skills and push yourself
- Not afraid to explore new concepts within a set of rules
- A strong passion for building things from scratch
- Great teamwork, leadership, documentation, and communication skills
- Individuals that have demonstrated they can take point on and OWN a project from concept phase to final product ready to be used on-vehicle

- Strong software engineering aspects (debugging/profiling, version control, code modularity, maintainability, etc.)
- Project Timeline Development and Maintenance
- Design and Specification goals development
- Research: Well aware of current developments in the FSD space, and provide resources to members
- Potential (Non-Exhaustive) Skill Set - **We by no means expect any candidates to have all the experiences and knowledge base outlined below**:
    - Strong knowledge of probability theory
    - Familiarity with state estimation concepts, such as Kalman Filters, importance sampling, and Bayesian State Estimation
    - Familiarity with some optimization concepts/problems, such as gradient descent, loss functions, regularization
    - Experience with C++ programming, Python programming, Linux, TensorRT, ROS, OpenCV, PyTorch, TensorFlow, MXNet, Git, Docker (Containerization), Numpy, Pandas, etc.
    - Experience with CAN protocol
    - Experience with deep learning frameworks
    - Experience with YOLO and other object detection algorithms
    - Understanding of advanced driver assistance systems (ADAS) sensors such as radar, stereo camera, ultrasonic, LIDAR, IMUs, etc. along with an understanding of measurement, data-reduction, target identification.
    - Knowledge of neural networks
    - Familiarity with signal processing (LTI filtering, outlier rejection, reasoning in both time and frequency domains)
    - Experience with control loop design and tuning for vehicles, using kinematic models
    - Experience with implementation of advanced control algorithms (e.g. LQR, MPC)
    - Familiarity with basic computer vision concepts (intrinsic and extrinsic calibrations, homogeneous coordinates, projection matrices, epipolar geometry, visual odometry, etc.)
    - Experience working in software/hardware integrations
    - Knowledge of vehicle kinematics/dynamics
    - Knowledge of how to drive stick-shift
    - Knowledge of how to tie a knot with a cherry stem and your tongue
    - Defense Against the Dark Arts (Mandatory)
    - These jokes are to say, we absolutely do not expect these skills, but they are what you will learn at FEB!

State Estimation Recruitment Project

This project will help you learn about *localization*: figuring out where the car is at any given point in time. While position is inherently simple for humans, robots have no idea where they are in space! Using things like GPS and wheel rotation measurements can help determine the position, but all of these sensors are inherently noisy, like Sixers fans when they're down by 30. To be able to acquire an accurate position of the car, we have to work some magic. In our recruitment project, we want to see a rough outline of how you would get a position estimate for the car, and maybe even improve the estimate of the cones around it! We have resources available to help you understand how to approach this problem and are available through office hours to clarify any questions on the project.

Resources (Conceptual): There are many different ways of approaching this problem, with pros and cons to each method. Here are just a few (explore on your own!) methods of solving the problem. We recommend watching these 2 videos first, and then start to go more in depth on which solution you find yourself leaning towards. Do not get married to one solution though!

SLAM Intro, Kalman Filters Intro

SLAM (Simultaneous Localization and Mapping):
Note: Saying you're doing SLAM is like a company saying they "use AI". It tells us absolutely nothing. There's many different forms of SLAM (Particle, Graph, EFK, Visual, LiDar). Explore them all and be prepared to justify your choice!
Matlab intro - Short and pretty easy to understand. Goes over the varieties of SLAM in a broad sense.
Berkeley Research Paper part I - A little more technical. Goes over the history and uses of SLAM. Also a brief intro to two of the main types of SLAM.
Berkeley Research Paper part II - Even more technical. Goes into the specifics of how different SLAM algorithms are implemented.
German paper - Very long. Not overly technical. Goes into how SLAM is used in formula competitions. The most important part is the differences between EKF SLAM and Graph Slam.
Stanford Paper - Short but very technical. Goes over how FastSLAM works in depth.

Kalman Filters/probabilistic state estimation:

Kalman, Bayesian, and Particle Filters are methods for fusing many sources of noisy data into a better state estimate using statistics. For linear systems with Gaussian noise, Kalman filters are mathematically optimal. They're the state estimation complement of the LQR in the LQG problem.
Here are some resources to learn about these kinds of algorithms:
Kalman & Bayesian Filters in Python - An implementation-focused approach to learning about probabilistic filtering. Very long, but well explained.
Software bible - See chapter 9.1 then chapter 8. Epic book on robotics.
EKF paper - A case study for applying the Extended Kalman Filter in competition robots. Has a good statistical primer (although it doesn't derive the Kalman equations so much as present them) and is an application familiar to anyone who did robotics in high school.
MATLAB Video Series - A nice series on Kalman filtering

Clever Shortcuts:
Sometimes the best solution is a simple solution! Different sensors are more accurate, and sometimes you can get away with cheeky things. We encourage you to use your own intuition and explore different ways of approaching this problem.

Machine Learning:
If you feel confident in your abilities to train a model, come to OH and tell us about your plan, and we can generate more testing data!

Resources for Optimization:
Many techniques include numerical optimization in some form. If this is your first time encountering this topic, it can be quite opaque, due to the need for very optimized solver code (so it's all ancient and written in C or Fortran) and the complex mathematics. Here are some resources aimed to lower that barrier to entry:

SciPy for Beginners - Most basic starting video
KKT conditions & IPM - more mathematical foundation for optimization and Interior Point Methods

Project Steps

- Step 1: Design and Plan!

    - Requirement:
        - Read the Positional Estimation resources, use first principles reasoning to decide which you would use and why. Be specific - LiDAR vs. visual, 2D vs 3D, filtering approach, etc. Then address how you would solve the problems a Formula team faces with your chosen version.
        - Come to OH and talk through your thinking, we will challenge your choice and ask you follow-up questions. This isn't meant to deter you from your implementation, just to solidify *why* you chose the implementation you did.

    - Deliverable:
        - You will be expected to come prepared with an explanation as to why you chose the implementation you chose. We expect many will use a powerpoint to do this, but this is not a requirement, as long as you can clearly communicate your thought process.
        - Be able to explain your thought process that led to your decision. We really want to see how you can learn new things, analyze results, and use that to problem solve.
        - There is no explicit correct answer, we are just interested in seeing how you would come to a solution.
        - Be prepared to answer questions.
- Step 2: Implement!
    - Deliverable:
        - You will implement the chosen implementation to estimate the car's state!
        - You will be given a CSV filled with time stamped messages from sensors (noisy sensors!) and have to use these messages to estimate your position and orientation!
            - Message Types:
                - GPS: positional estimate from GPS of where the car is
                    - Latitude and Longitude
                - IMU: Acceleration and orientation data of the car
                    - Orientation and linear acceleration
                - Cones: Perception spotted cone positions in polar coordinates
                - Wheel Speeds: Wheel speeds of the car
                    - Gives each wheels rotation
                        - Diameter is 0.505 meters

        - Speed is in RPM
    - Note: You absolutely are not required to use all of these message types, our current implementation does not use all of these (we use defense against the dark arts).

CSV Notes:

stationary.csv - csv of the car while its not moving, made to give you an idea of the sensors in a stationary setting

movement.csv - csv of the car while its moving, predict where it is at during different times