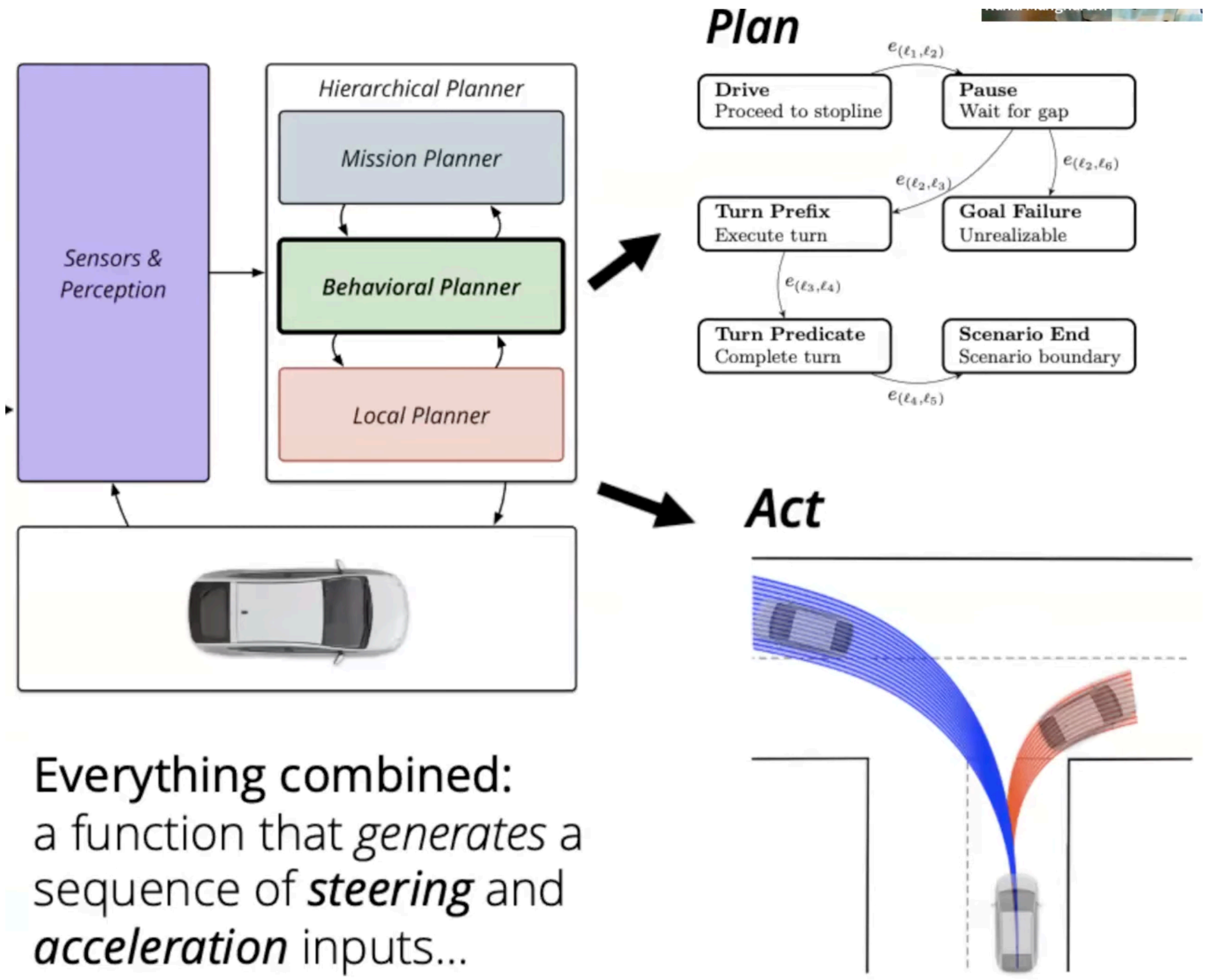
Lecture 1

- Lecture 1: https://www.youtube.com/watch?v=v6w_zVHL8WQ
- Written by Milad Abdi with some AI assistance.
- These notes are meant as a reference and cannot replace watching the lecture.
- **Skill Stack**
  - Base: Perception, planning, and control
  - **Machine Learning layer** (vision and perception algorithms): Used when physics-based models are insufficient (e.g., drifting)
- Formal Methods: To prove safety mathematically
- F1-Tenth uses NVIDIA Jetson Xavier NX is a GPGPUs (General-Purpose computing on Graphics Processing Units):
  - Supports running and programming applications **directly on the device**.
  - Can connect a **monitor, keyboard, and mouse** (no external laptop required).
  - Can also be accessed **remotely via SSH over Ethernet**.
  - Enabled Workflow
    - Develop locally (Mac/Windows, Docker or not)
    - Push to GitHub
    - SSH into Jetson
    - Pull repo
    - Build/run **in the Jetson's Linux environment**
- **Lesson Plan**
  - Learn to drive: Avoid Crashing (Reactive planning)
  - Learn to race: Follow the Raceline (Purse pursuit), Race & Overtake (Graph planner)
- **Balancing Performance and Safety**: Current AV (autonomous vehicles) technology still **struggles in non-cooperative scenarios** like merging due to **competing objectives**:
  - Max performance: Negotiate the merge without delay or hesitation
  - Max safety: Avoid catastrophic failures and crashes
  - **Racing (autonomously) highlights this performance safety tradeoff. This is why we race.**
    1. **Detecting vehicle limits**: Unstructured environment; different tracks and conditions; different vehicle setups.
    2. **Decision making at the vehicle limits**: High prediction uncertainty; strategy, energy, overtaking.
    3. **Handling at the vehicle limits**: High speeds and acceleration; high planning horizon necessary; small reaction times

        - If we only trained in civilian driving then we would be too safe and not push the limits of AV technology. The civilian cars trickle down from the race cars.
- **AV Perception, Planning, Control (PPC) pipeline**:
    - Sensors & Perception: To begin navigating
        - Camera feeds (RGB images: color + texture). Texture is beyond color e.g., asphalt is rough and grainy but walls are smooth or brick.
        - Lidar feeds (3D point clouds: precise distance measurements)
        - Capture depth views (per-pixel distance from camera/LiDAR)
        - Capture optical flow views (relative motion of pixels between frames for **speed & direction**)
        - Object detection (what objects are present and where they are)
        - Object tracking (maintaining object identity and motion over time)
        - Semantic Segmentation (pixel-level classification of scene elements). Make every pixel assigned a class label e.g., this pixel is road, this pixel is car, this pixel is pedestrian.
    - Hierarchical Planner: Different levels of planner that repeats 20-100 times a second.
        - Note that each planner repeats at different speeds. Mission planner doesn't repeat nearly as much as local planner.
        1. Mission Planner: Main goal. E.g., Go from Science World to SFU, what streets to go on.
        2. Behavioural Planner: Choices at each step. E.g., Drive or Pause
        3. Local Planner: Act. E.g., this is the best trajectory -> generate sequence of steering and acceleration inputs to follow that path



- **Pipeline Continuation**: Hardware (sensors) -> Software (PPC) -> Hardware (race car)
- **Perception**:
  - Perception-only based navigation: No bigger plan for where to go, just driving based off reactions (just reactions to sensor input).
  - Localization: Estimating the vehicle's pose (position + orientation) in the world
  - Detection: Identifying objects and obstacles in the environment
- **Planning**:
  - Prediction: Estimating future motion / path
  - Behavior: High-level decisions (e.g., how to handle other drivers or sequence turns)
  - Trajectory: Generating the right trajectories that minimize cost while adhering to dynamic systems constraints.
- **Control**:
  - Path & Velocity: Computing steering and speed commands to follow a planned path
  - PID control: Feedback controller using proportional, integral, and derivative terms to reduce error
  - Modern predictive control: Optimization-based control (e.g., MPC) that accounts for future states
- **Types of Failures**:

    - False Positive
    - False Negative
- **F1Tenth Gym**:
    - Lightweight 2D simulator built in Python
    - Asynchronous
    - Faster than real time execution (30x realtime)
    - Realistic vehicle simulation and collision
    - Publishes laser scan and odometry data
    - Built for fast prototyping
- **LG SVL Simulator**:
    - 3D simulator with unity based environments
    - **Distributed cloud simulation**
    - Bridges: ROS, ROS2, Apollo, Autoware, Python
    - Sophisticated physics simulation
    - Sensor integration
- A **bridge** connects **different systems or software**, allowing them to communicate seamlessly.
    - Connects the simulator (e.g., LG SVL) to nodes (e.g., from ROS2/Autoware).
    - Translates **sensor data** (LiDAR, camera, GPS, IMU) from the simulator into **ROS2 messages**.
    - Sends **control commands** from the software back to the simulator.

Future Module Insights

- **Race 1: Reactive Methods**
    - **Pose Representation and Transforms**:
        - Each sensor provides measurements in the frame of reference specific to that sensor.
        - **Sensor Fusion Algorithms**: Combine data from multiple sensors to get a **more accurate and reliable understanding of the environment** than any single sensor alone.
        - Outcome: Coordinate frames, Rigid body transforms.
    - **Multiple Reference Frames**: World makes sense if we put laser scans in the **global map frame** instead of the **laser frame** -> Gives more useful info when planning in **global frame** (a consistent world-fixed coordinate system) than in the **observation frame** (the frame where each individual sensor observation is interpreted).
    - **Electronic Speed Control**: Actuate vehicle via commands in physics units.

        - Outcome: PID (Proportional-Integral-Derivative controller for precise speed/position control), motor control (low-level signals sent to the motor to achieve desired motion), etc.
    - **Walk Following**: How we can drive the car around the track.
        - Outcome: Basics of PID, how to compute error, and failure modes.
    - **Obstacle performance**: How can we avoid obstacles.
        - Outcome: Basics of reactive navigation, avoidance on both static and dynamic obstacles.
- **Race 2: Map-based Methods**
    - **Scan Matching**: Where is the robot with respect to the previous frame.
        - Outcome: Iterative closest point algorithm, implementing a real research paper.
        - Scan matching is a fundamental localization algorithm, and is used in most of the modern SLAM algorithms.
        - **SLAM (simultaneous localization and mapping)**: Method that lets you build a map and localize your vehicle in that map at the same time. Used for path planning and obstacle avoidance. Learns the map -> Exploits the learned map for speed
    - **Simultaneous Localization and Mapping with Cartographer**: How to use state-of-the-art tools for map building
        - Outcome: Understand the Cartographer paper and how it relates to scan matching.
        - **Localization**: Estimate the robot's pose within a map using sensor data, often by matching LiDAR scans or other observations to the map.
    - **Localization: Particle Filter**: Given a map of the world and multiple sensor observations, what is the pose of my robot?
        - Outcome: Understanding particle filter, which is a version of a bayesian filter.
    - **Pure Pursuit**: How to track a reference trajectory given a map and the ability to localize?
        - Outcome: Closed form geometric approach and alternatives.
- **Race 3: Head-to-Head**
    - **Motion Planning**: How do we combine the capabilities of map based methods while being able to avoid obstacles
        - Outcome: Understanding search-based motion planning, probabilistic planning methods, RRT and its variants.
        - **Occupancy Grid**: Approximating the real world with a discrete representation, also relates back to SLAM lecture
        - **Planning in discrete space**: With search-based planning methods (A*, Dijkstra's)

    - **Planning in continuous space**: With probabilistic planning methods (RRT, RRT*)
- **Model Predictive Control**: Create dynamically feasible trajectories for overtaking.
    - Outcome: Trajectory optimization & Sampling based MPC
- **Learning-based CV (Computer Vision): Detection and Pose Estimation**: Where is the other car without using fiducial markers?
    - Outcome: Understanding the multi-view geometry, the epipolar constraint, stereo vision, and using Convolution Neural Network detectors.
- **Classical CV: Detection and Pose Estimation**: Where is the other car?
    - Outcome: Understanding camera model, single view geometry, Homography, detecting features, and prediction.
- **RL (Reinforcement Learning)**: How to learn from human drivers (RL can be more than this but this course is focused on human imitation).
    - Outcome: Understand imitation learning and implement it.