# GOLDENEYE

*APTIV VISIT, FRIDAY APRIL 20, 2018*

# *INTRO*

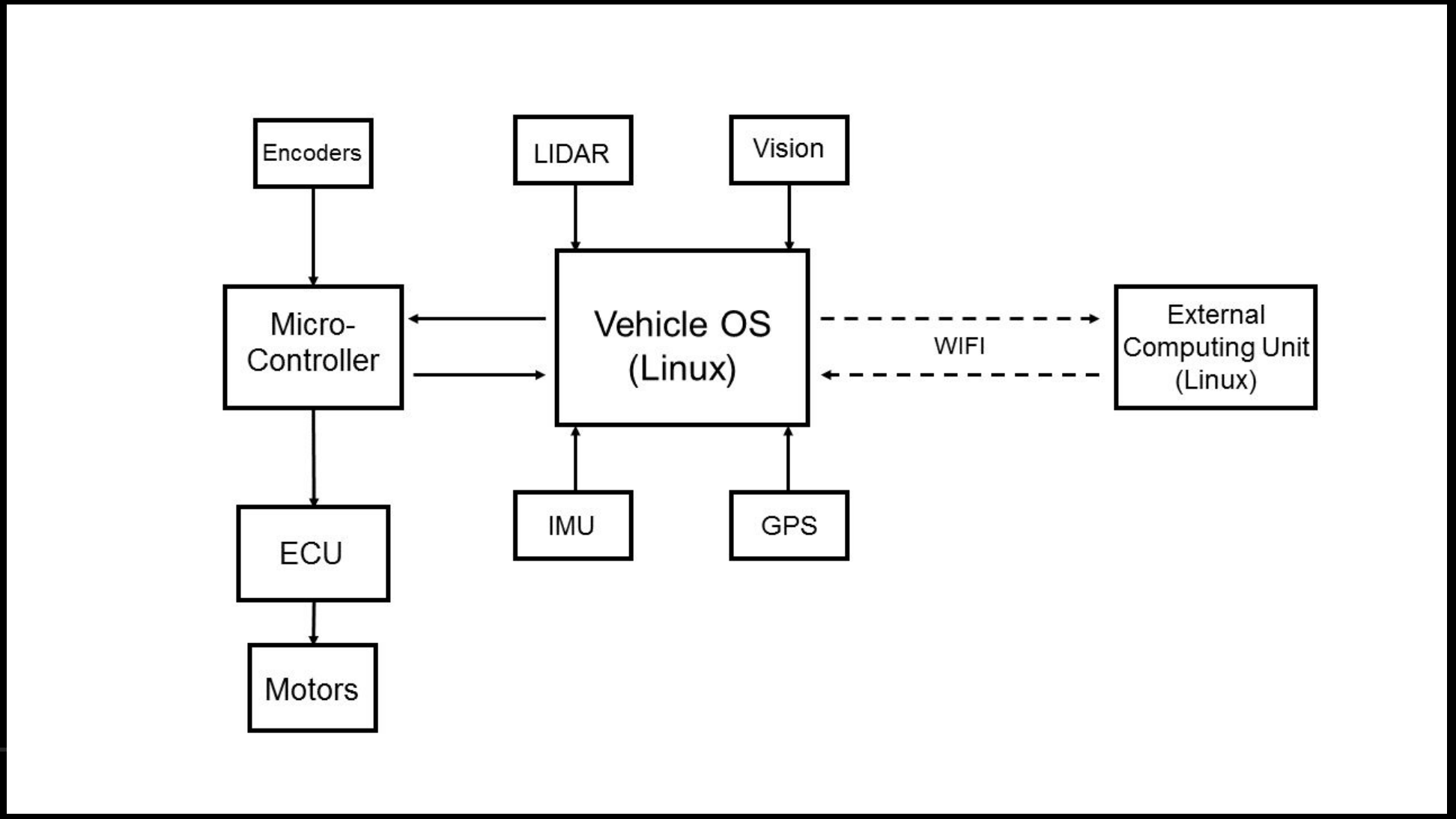
- Berkeley Student Team
- Won prize in NASA Aerospace Paper Competition
- With help from Ashish Krupadanam, a new project was created
- Working with Prof. Borrelli's MPC lab, 2 1/10 scale RC cars were built.
- Goal: develop applications in predictive driving

# *CONTENT:*

1. Team
2. Vehicle Architecture
3. Lane Keeping
4. Low Level Control
5. Point to Point navigation
6. Obstacle Avoidance
7. Sensor Fusion
8. Current and Future Work
9. Final Remarks

# *TEAM*

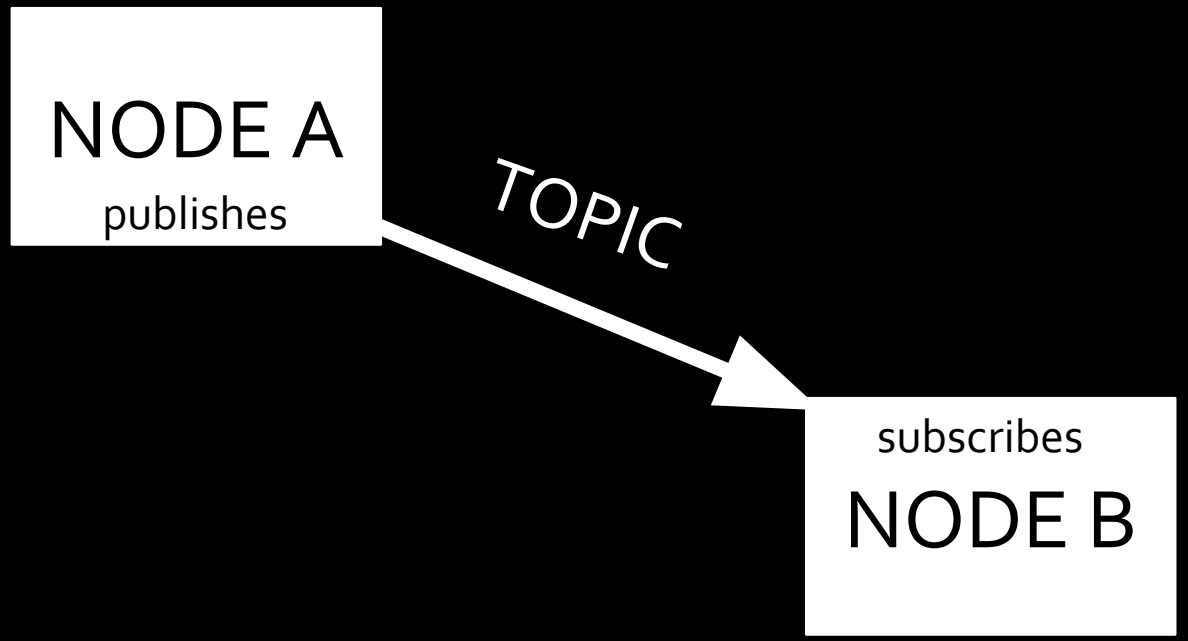
# HARDWARE ARCHITECTURE



# SOFTWARE ARCHITECTURE

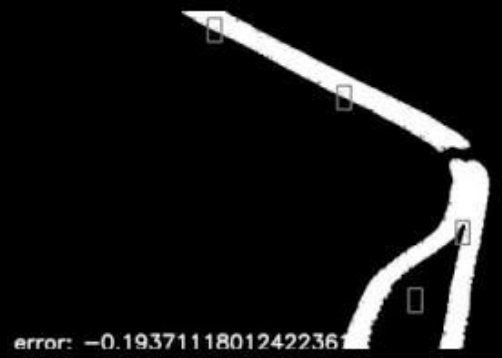
Use Robotic Operating System (ROS)

- Flexible, robust development framework
- Provides pipeline for communication between programs running in parallel
- Instantiate Nodes, representing processes in system
- Nodes communicate over Topics, through pre-specified data types (messages)
- Framework allows for computation on multiple machines

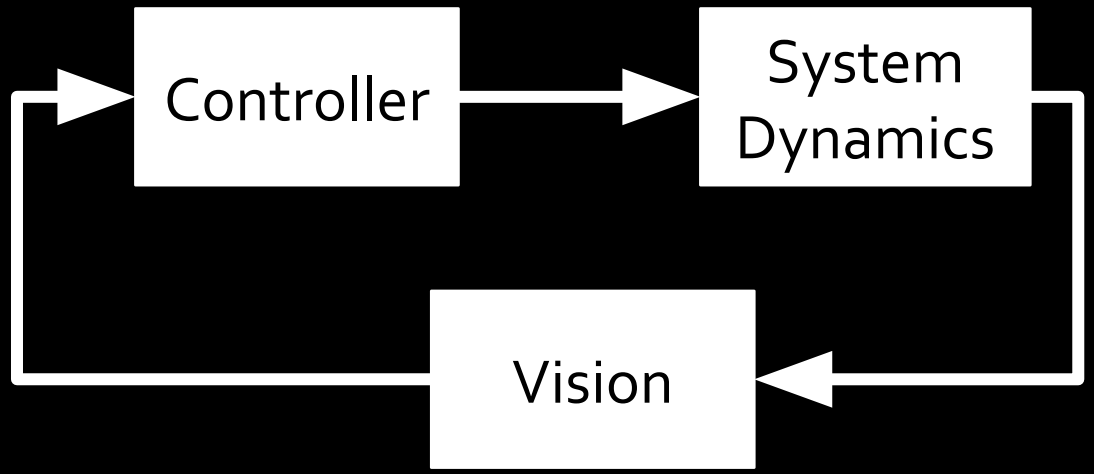



# LANE KEEPING

## Vision Algorithm




## SISO Controller



# *LANE KEEPING*

## Vision Algorithm

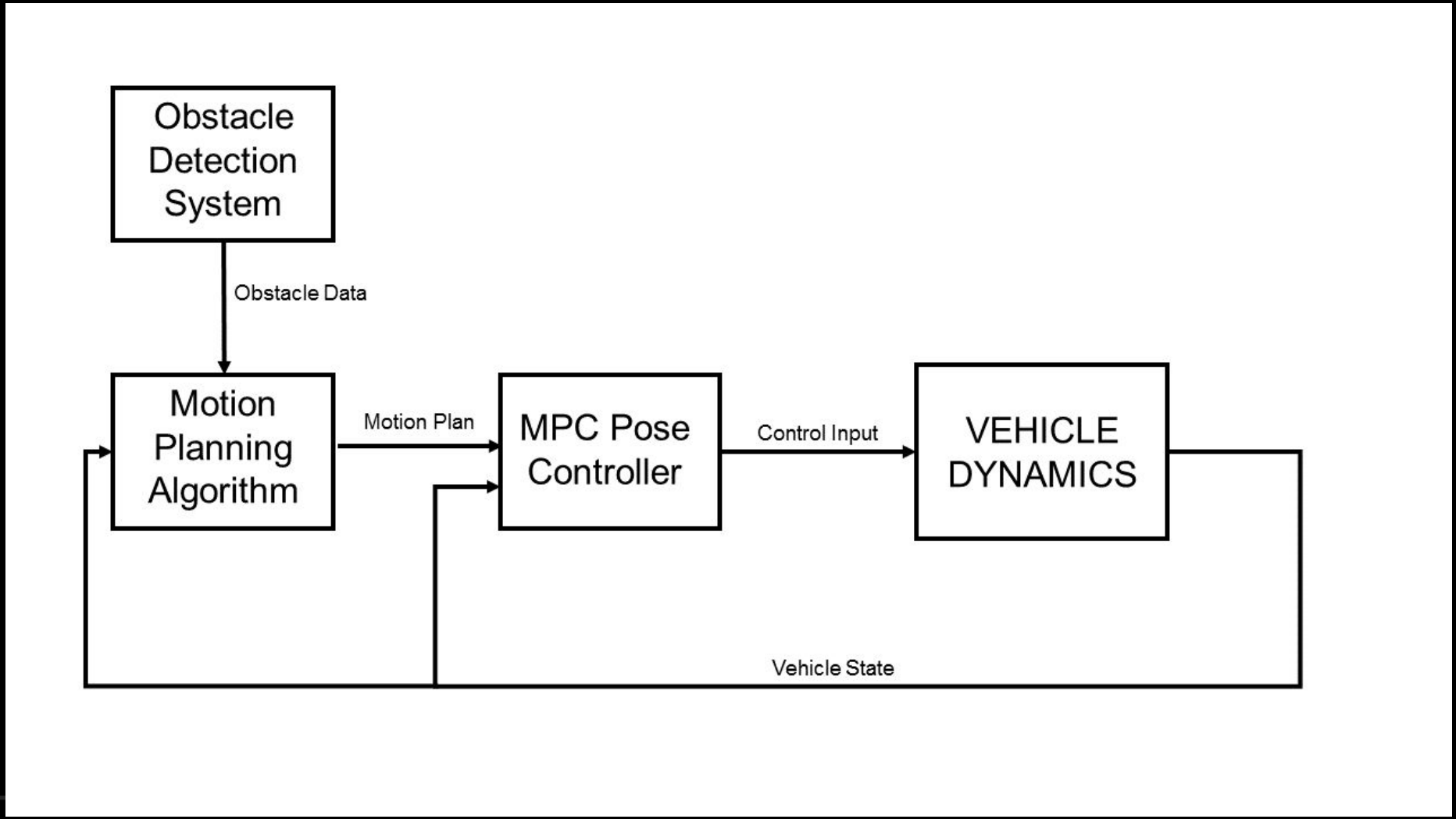
- Thresholding followed by detection of y-coordinate of maximum contrast
- Prioritizing computational efficiency on the odroid for real time image processing means we can only look at 10 rows of the image to determine error

## SISO Controller

- PI controller maps the error to an actuation command
- Feedback control makes the system relatively robust to imperfect plant and control constants
- Controller takes into account delay to compensate for high lookahead of camera

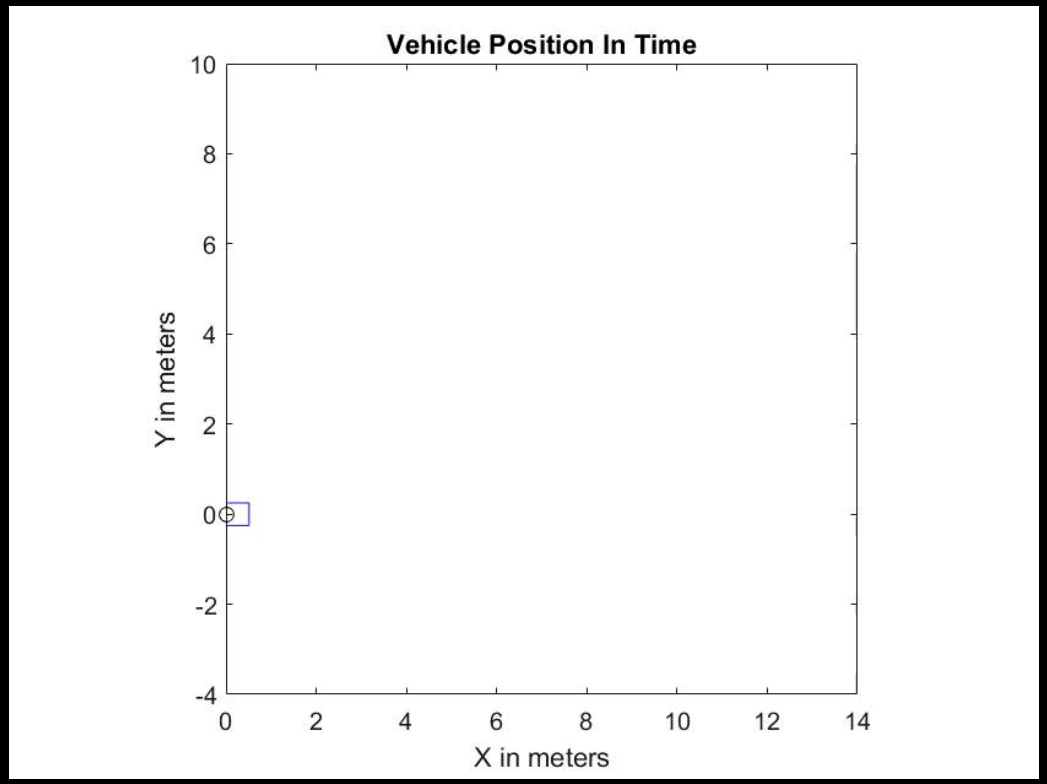
# *LANE KEEPING*

# POINT TO POINT NAV



# *Model Predictive Controller*

- System dynamics are known
- In discrete time, state vector $\vec{x}_{[k+n]}$ depends on $\vec{x}_{[k]}$ and control input $\vec{u}_{[k]}$ , $\vec{u}_{[k+1]}$,…, $\vec{u}_{[k+n-1]}$
- Formulate and solve optimization problem at each timestep



# *Model Predictive Controller*

- Single target point
- Brute force trajectory generation over finite receding horizon
- Evaluate cost function over all trajectories, execute optimal

Euclidean distance cost function → Aggressive Actuation

# *MOTION PLANNING*

- Motion planner plans path before initiating motion
- Path represented by set of points [xi,yi]

Planner is active in feedback loop:

- Regulates waypoint sent to MPC controller

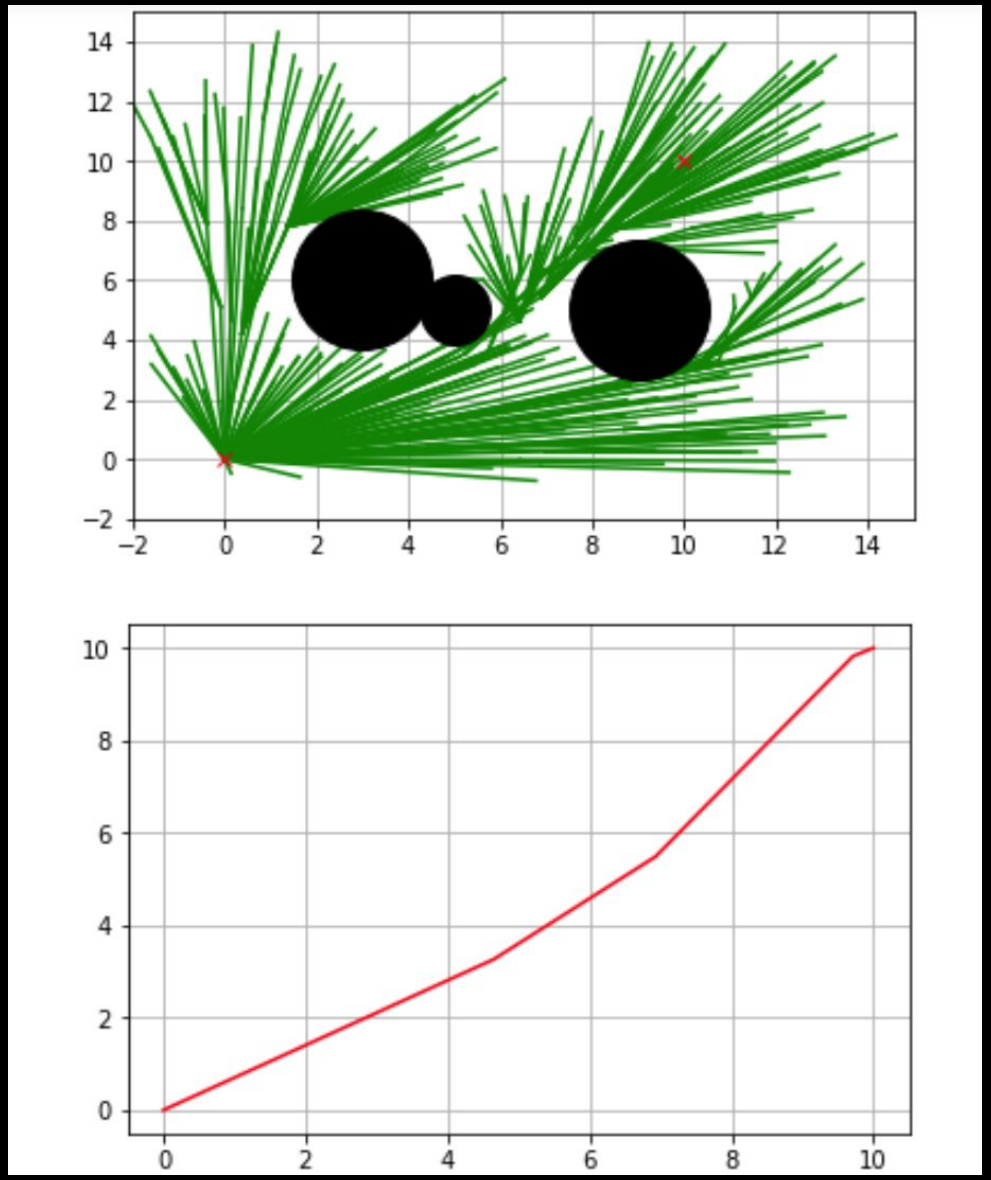
Approach guarantees feasible solution in real time, low computational cost.

# MOTION PLANNING

## RRT* Algorithm

Key Observation: Creating a feasible trajectory from scratch is hard, checking whether a given trajectory is feasible is easy.

Developed by Prof. Emilio Frazzoli et al. at MIT, current CTO at Nutonomy (Aptiv acquisition)



# MOTION PLANNING

## Geometric Algorithm

Works for navigation through field of obstacles.

Generates sparse set of waypoints, relies on MPC controller to navigate in a smooth manner.

Capitalizes on knowledge of turning radius

## Greedy Search:

1. current location = start point
2. While(goal != reached):
   a. obs = find nearest obstacle within range of motion
   b. on circle between current location and obs : sample feasible points
   c. find optimal point
   d. append optimal point to path and set current location to optimal point

# *MOTION PLANNING*

# Sensor Fusion

- Camera provides high resolution in detecting detail, though fails to perceive depth
- Planar LIDAR data generates depth information through 2D-point cloud, however lacks detail to distinguish objects
- Goal: Capitalize on strengths of respective sensor feeds and generate accurate estimates of object location

# *Sensor Fusion*

- Prior sensor fusion projects generate object detection scheme by immediately fusing entire image and LIDAR-point maps
- Given robust vision object detection schemes, same goal can be achieved by associating regions of interest (ROI) in an image with regions of lidar data

# *Sensor Fusion*

- Other methods derive a geometric mapping between pixels and LIDAR points
- *Given that some mapping exists, could we train a neural network to implicitly encode said mapping between image pixels and lidar points on its own?*
- LIDAR, camera, and GPS data collected during training to associate vehicle position in space relative to that of another object
- This is done by driving two cars within each other's field of view. GPS data is referenced to generate ground truth relative coordinates

# *Sensor Fusion*

# *Sensor Fusion*

- Results of data set

# *Current Work*

- Motion Planning
  - Combining methods like RRT* for global path planning while using local optimizers like Trajopt for smoother short-distance paths
  - Defining allowed and disallowed regions of sampling as continuous blocks
  - Using longer horizon MPC controllers to penalize extreme actuation
- SLAM
  - Using in-build ROS SLAM to localize the car indoor and detect allowed regions using LIDAR
  - Using EKF to track other vehicles using LIDAR and arriving at probabilistically accurate estimates of position, velocity, acceleration and predicted paths

# *Future Work*

- Racing Inference and Planning
  - Using both cars together around a track to infer racing strategies of observed vehicles around a racetrack
  - Adapting to opponents strategy using an MPC controller further optimized with Q-Learning
- Simulator
  - A reasonably accurate simulator of the car is essential to making deep learning or Q-Learning based approaches feasible
  - Will facilitate exploration of these techniques given sparse data collected in the real world, can explore meta-learning and fine-tuning of simulator trained models

# *Final Remarks*