Team Road Runner

Team 3: Arnav Dhamija, Luca Scheuer, Saumya Shah
Instructor: Rahul Mangharam(rahulm@seas)
ESE615 Autonomous Racing

Penn Engineering

Content

I. Milestone II approach
   - Pure pursuit
   - CMAES
   - Splines

II. Milestone III approach
   - RRT + Pure Pursuit
   - Advancements

III. Milestone IV approach
   - MPC
   - Lane switching

IV. Miscellaneous

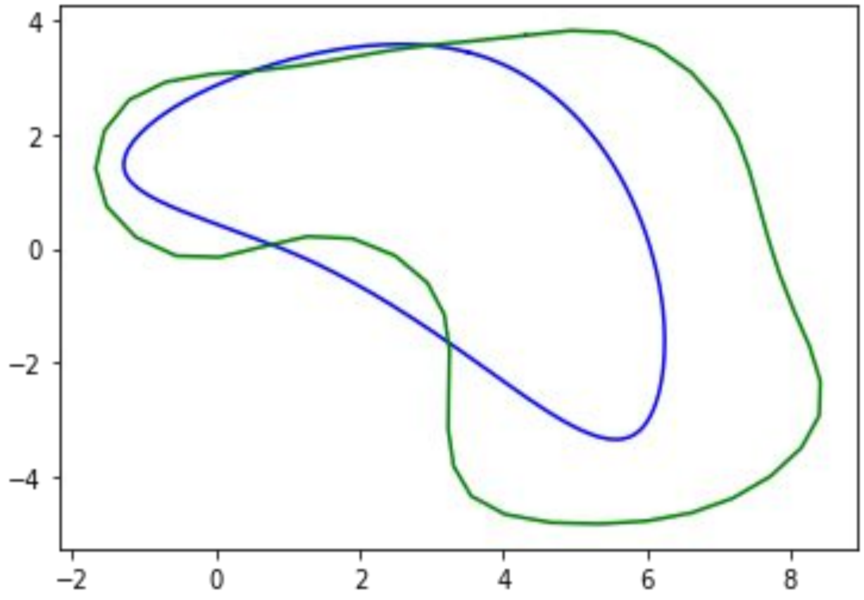
Milestone 2 approach

Optimal Trajectory

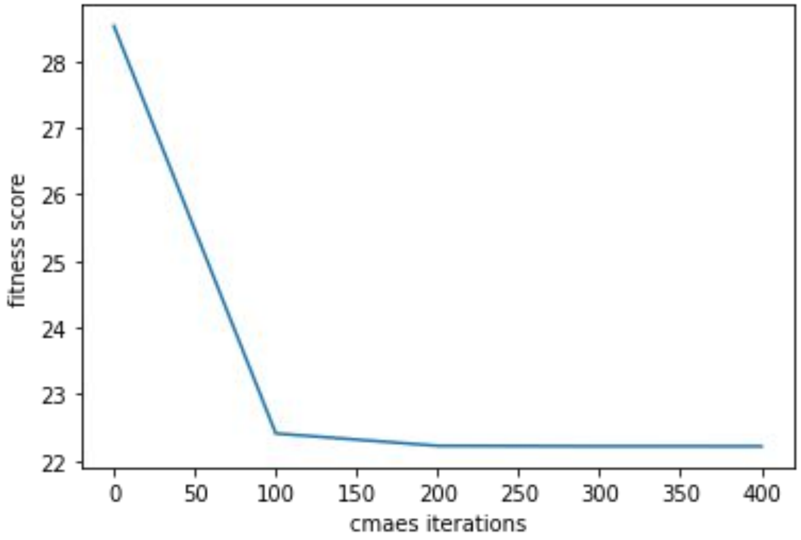
- CMAES for trajectory generation
  - Sparsely sampled centerline points for CMAES initialization
  - Polar constraints for CMA-ES solver
    - Solves for R_i , θ_i
    - No box transformations involved
    - Simplified constraints to bounds as required
  - Regularized spline fit on the sampled points
  - Fitness calculated as time to complete the loop with curvature dependent velocity profile (leads to approx. minimum curvature trajectory)
  - Regularized spline on final solution sampled for 500 points

Optimal Trajectory

CMAES trajectory vs Centerline



Fitness curve vs optimization iters

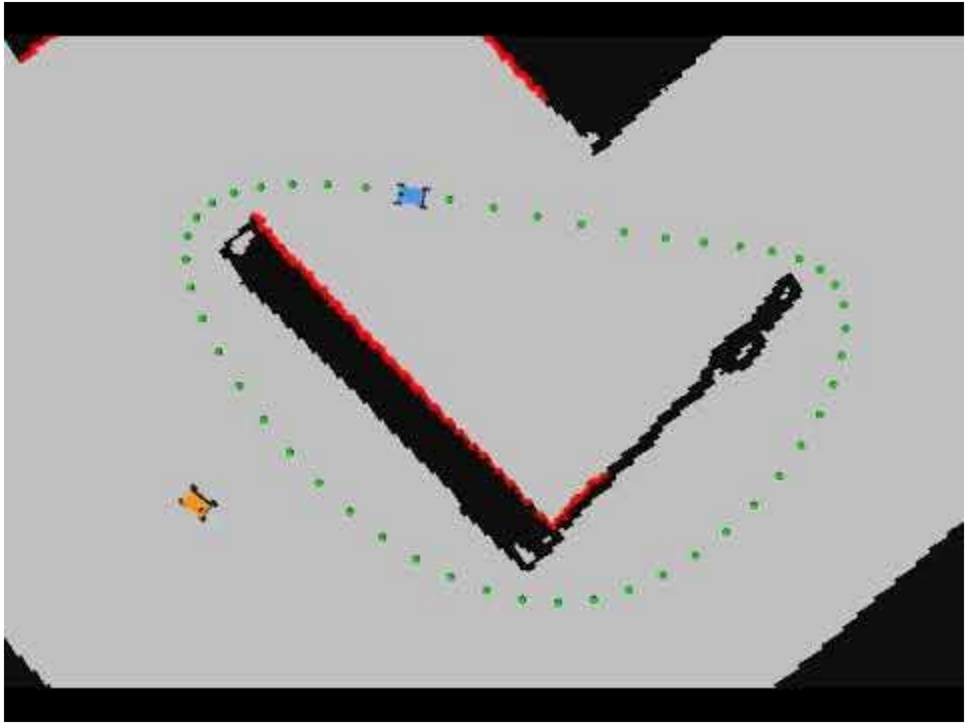


Penn Engineering

Optimal Trajectory

- Pure Pursuit
  - Used Pure Pursuit with lookahead to follow the way points at full speed
  - Cons: No drift control

Milestone 2

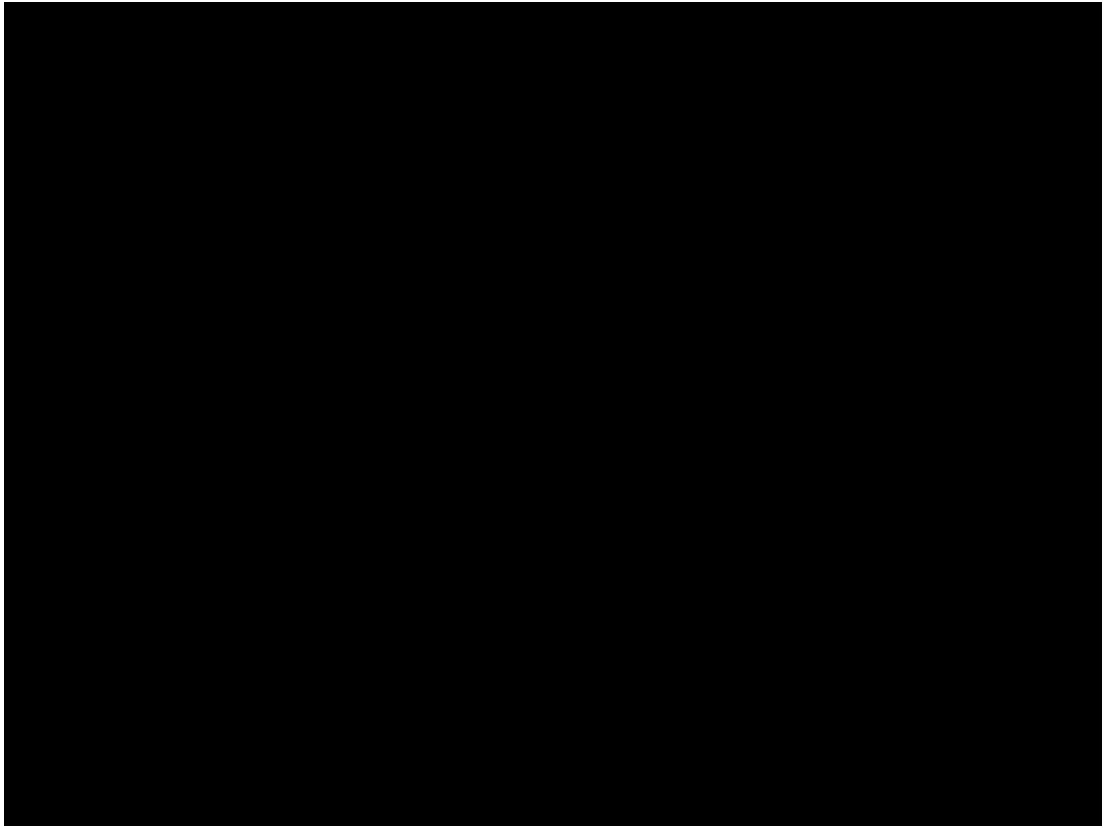


Milestone 3 approach

RRT*

- RRT* was used for Milestone 3 with following features:
  - Angle Constraints in steering function
  - Velocity calculated based on path length and speed
  - Early stopping - direct to goal
  - Path Freezing until collision detection or completion
  - Occ-grid dilation
- Pure pursuit used to set goals for RRT* with some lookahead
- Issues with tracking, single-threaded approach

Result:



Milestone 4 approach

Overall idea

- Trajectory Selection
  - Local Trajectories Generation
  - Local Path selection to track Global Path

- MPC controller
  - Reference Tracking controller

- Lane Switching

Trajectory Selection

- Global Path
  - CMA-ES trajectory + Alternate parallel lanes

- Local Paths
  - Generated constant curvature paths offline
  - Drift minimal curves with fixed velocity

Mini-HRHC paths

Trajectory Selection

- Path selection
  - Selected the local reference path which maximizes the objective

- Objective:

Closeness to the global path	+	maximizing progress	+	incremental index
(prevents moving far from global)		(promotes going far along global)		(prevents reversal)

MPC Controller

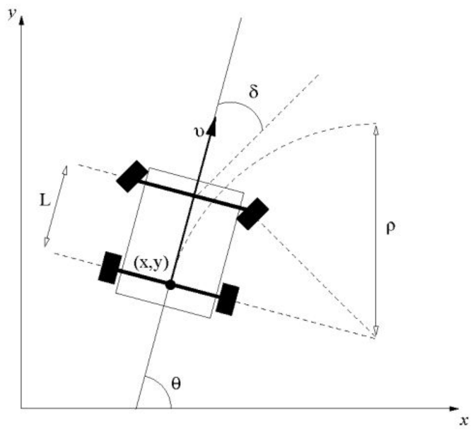
- Cost function
  - Quadratic Cost

$$\sum_{k=0}^{N-1} (\mathbf{x}_k - \mathbf{x}_{ref,k})^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_{ref,k}) + (\mathbf{u}_k - \mathbf{u}_{ref,k})^T \mathbf{R} (\mathbf{u}_k - \mathbf{u}_{ref,k})$$

  - Large state costs (x,y; orientation not penalized)
  - Small input costs (higher for orientation, near zero for v)
    - Desired input held at a constant high speed

MPC Constraints

- Dynamics of System
  - Kinematic Model with x,y,ϴ
  - Linearized using Forward Euler Discretization (proved to be accurate if t≃ 200ms)

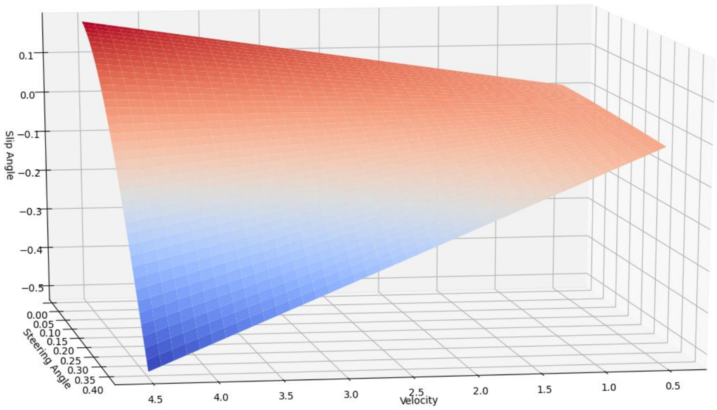


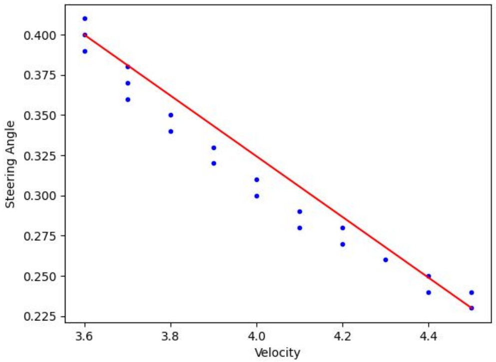
$$\dot{x} = vcos(\psi)$$
$$\dot{y} = vsin(\psi)$$
$$\dot{\psi} = vtan(\delta)/C_L$$

$$x = [x, y, \psi]^T$$
$$u = [v, \delta]^T$$

MPC Constraints

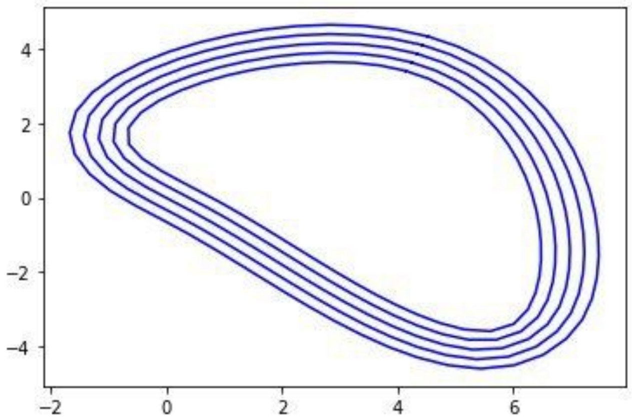
- Steering vs Velocity Constraints
  - Non-linear, but kind of close
  - Set to infinity in final race parameters





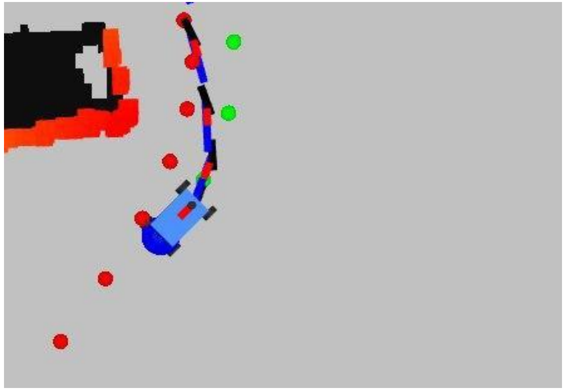
Lane Changing backup

- Why?
  - Current CMA-ES trajectory might be blocked by opponent
- Our solution
  - Keep 5 backup concentric paths which serve as lanes and switch tracking them on the fly
  - Use lookahead for collision on current path

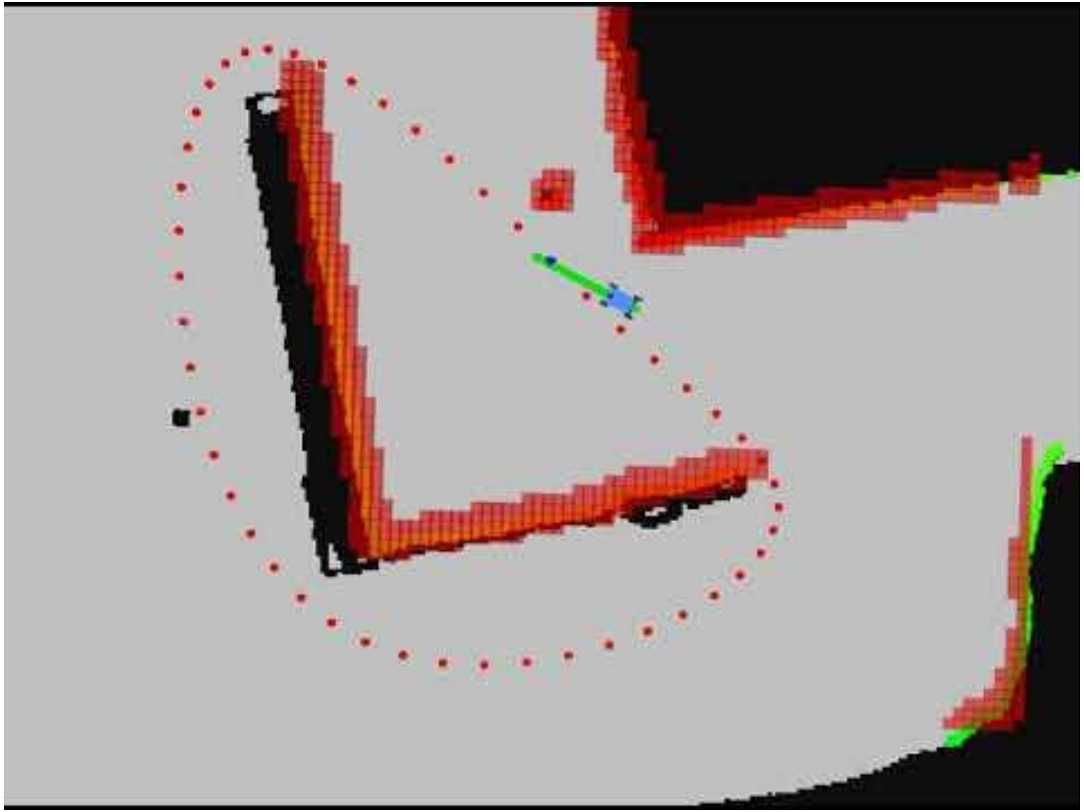


Other features

- Multithreading
  - Have a detached thread publishing on /drive in an infinite loop at 50Hz
  - Use remainder of previous MPC solution outputs before current MPC finishes solving
    - Anytime Algorithm!
- Visualizations
  - Car positions as calculated by QP solution
  - Lane changes



Results



Miscellaneous

Testing Methodology

- First test in 1-P simulator when running in debugger
  - Find edge cases and fix them
  - Use pose estimate and place vehicle in different positions
  - Add obstacles to course
- Run in 2-P Gym simulator
  - Use gdb attach if problems show up
- Race against our old agents
  - Issues with passing slower agents or unpredictable ones

Cutting Room Floor - Unused Material

- Prediction
  - Using Gaussian Process model trained over a bundle of trajectories generated using CMA-ES
  - Predicts dx/dt and dy/dt in real time for half the loop
  - Shortcoming: needs several priors about opponent trajectory, model size > 1.5GB, difficult to incorporate with planner
- Half space constraints for MPC
  - Couldn't figure out a suitable formulation to use for collision prevention

Cutting Room Floor - Unused Material

- RRT* as MPC Backup
  - Used regularized cubic splines to interpolate between trajectory points for smoothing
  - Applied randomized shortcutting algorithm to reduce path length safely
  - Issues: Couldn't get MPC to track the trajectory reliably

Fast Smoothing of Manipulator Trajectories using Optimal Bounded-Acceleration Shortcuts

Kris Hauser* and Victor Ng-Thow-Hing†
* School of Informatics and Computing, Indiana University, Bloomington, IN 47405 USA
† Honda Research Institute, Mountain View, CA 94041 USA

***Abstract*—This paper considers a shortcutting heuristic to smooth jerky trajectories for many-DOF robot manipulators subject to collision constraints, velocity bounds, and acceleration bounds. The heuristic repeatedly picks two points on the trajectory and attempts to replace the intermediate trajectory with a shorter, collision-free segment. Here, we construct segments that interpolate between endpoints with specified velocity in a time-optimal fashion, while respecting velocity and acceleration bounds. These trajectory segments consist of parabolic and straight-line curves, and can be computed in closed form. Experiments on reaching tasks in cluttered human environments demonstrate that the technique can generate smooth, collision-free, and natural-looking motion in seconds for a PUMA manipulator and the Honda ASIMO robot.**

I. Introduction

Autonomous robots that interact with humans or human environments must have the capability to quickly generate safe and natural-looking motion. So far, it has been a challenge to simultaneously satisfy the three objectives of speed, safety, and esthetics for high-DOF robots performing complex tasks in unstructured environments. Sample-based planners (e.g., PRM, RRT, etc., see Chapter 7 of [1]) are widely used to plan collision-free paths for high-DOF robots. They are often fast, but they produce jerky, unnatural paths. This paper presents a fast smoothing algorithm that postprocesses paths to produce a dynamic trajectory that respects velocity and acceleration bounds and avoids collision.

Standard sample-based planners compute piecewise linear paths that can be executed precisely by stopping the robot at

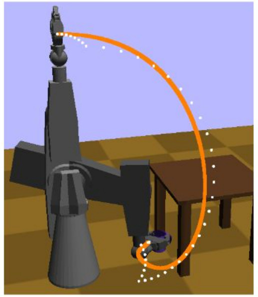
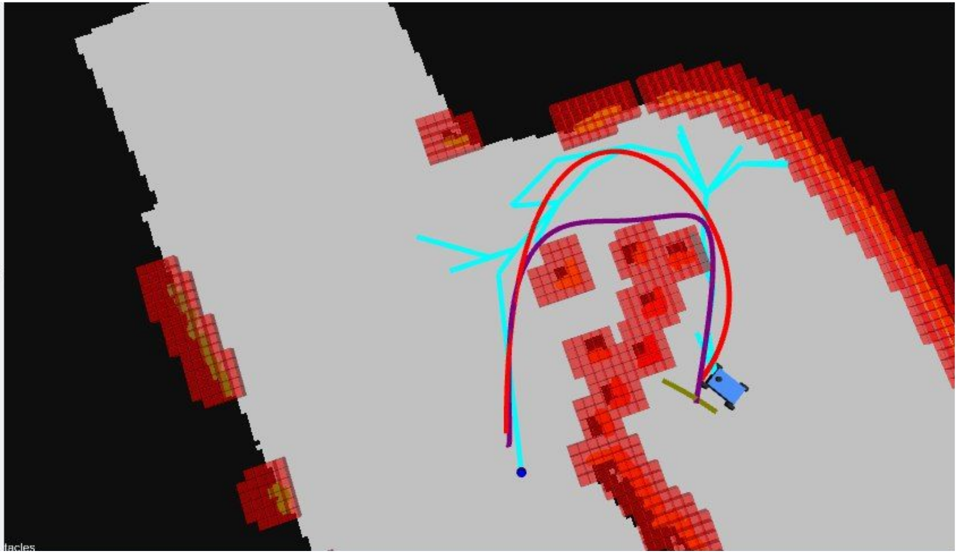


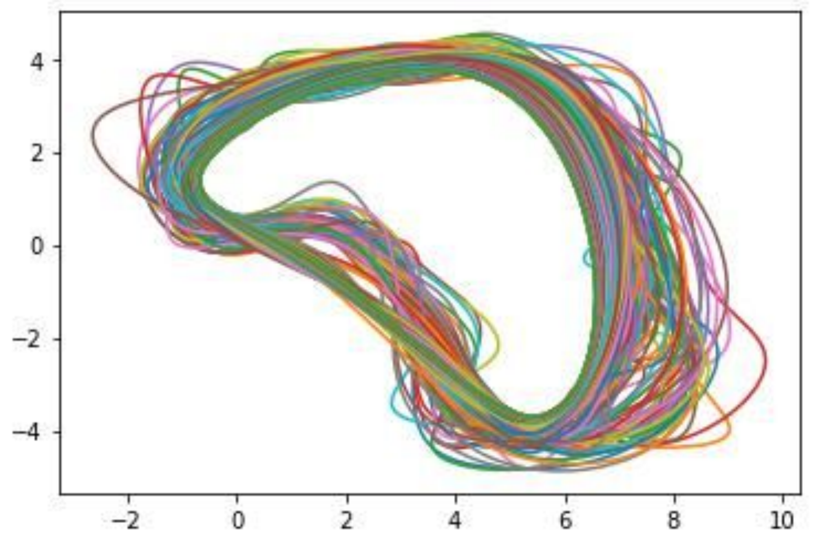
Fig. 1. A manipulator reaches under a table to grasp a cup. The white dotted curve depicts the original end effector path. The orange curve depicts the smoothed path after 100 randomly-attempted shortcuts. Execution time is reduced from 9.4s to 4.0s.

the algorithm is a smooth trajectory that respects collision, velocity, and acceleration constraints.

Cutting Room Floor - Unused Material

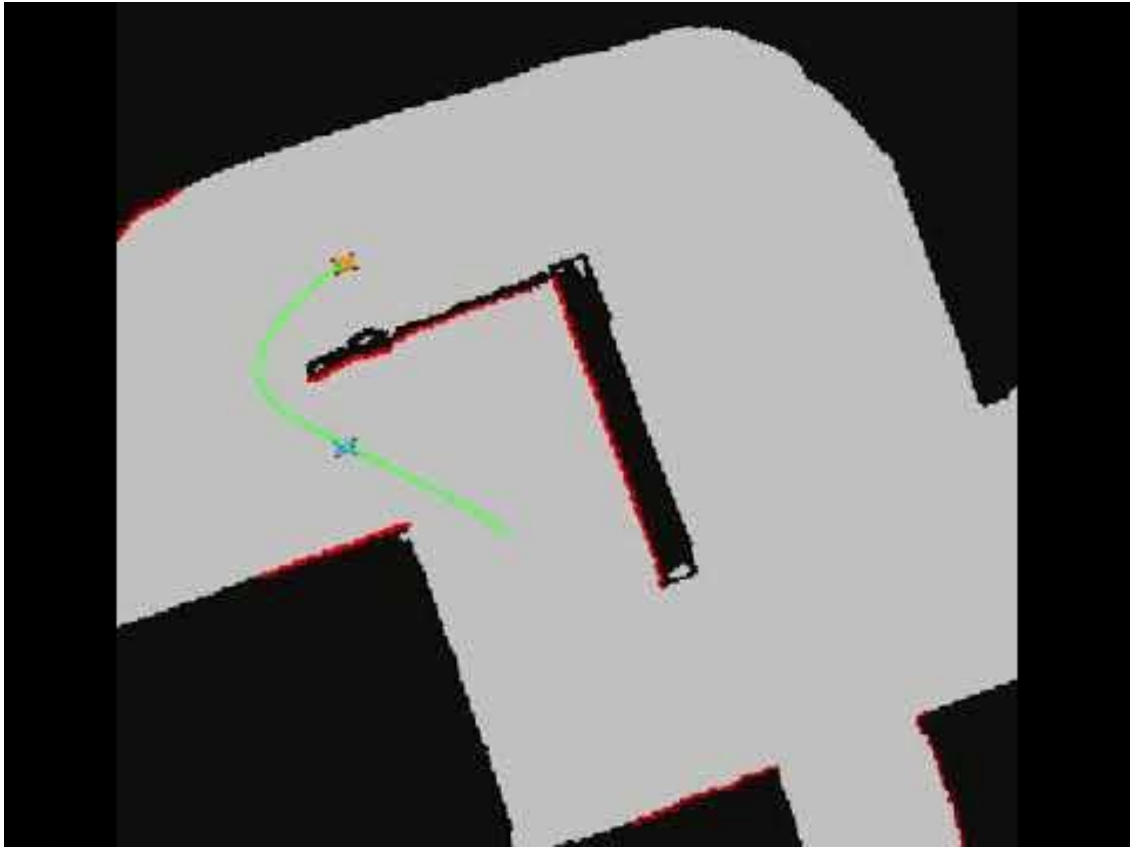


Red - Splined RRT, Purple - Shortcut & Splined RRT**



Trajectories sampled for training a Gaussian Process

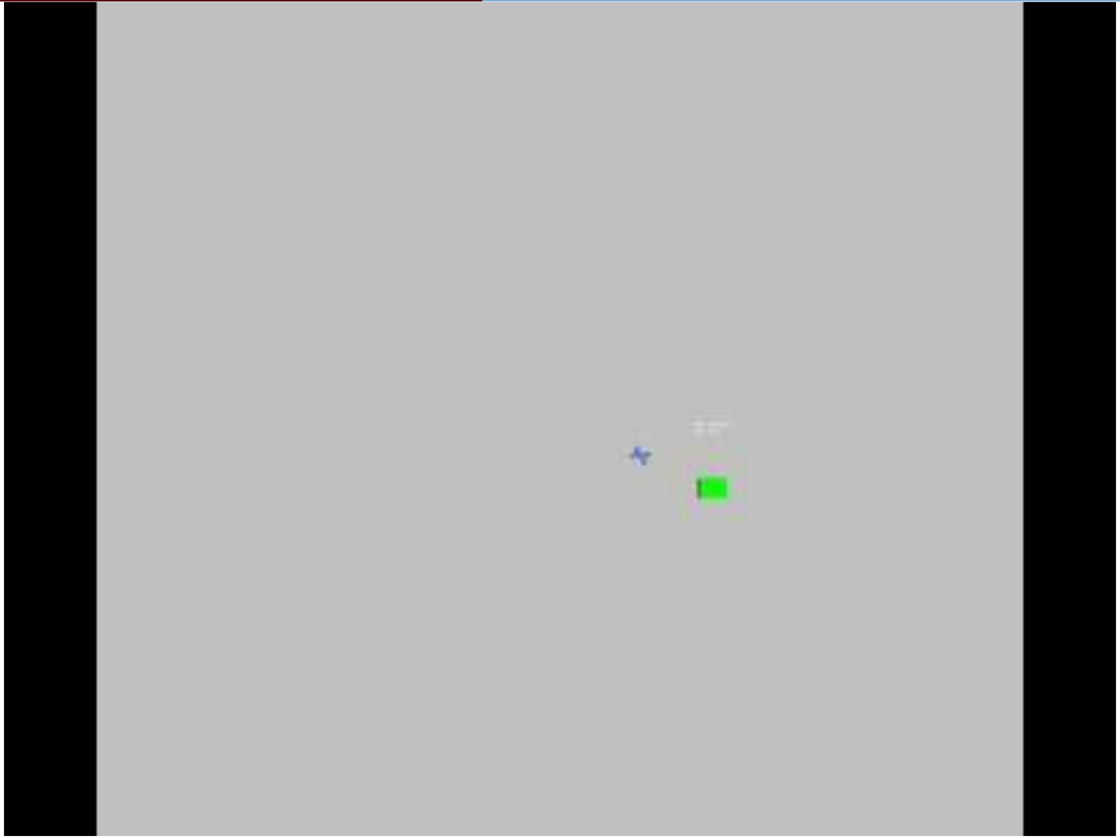
Cutting Room Floor - Unused Material



Learning Experiences

- (Dynamically feasible) RRT* is hard to get right
- Start simple before complicated solutions
- Use classes and objects liberally for C++ codebase
- Always use run with GDB when trying new things!
- Using visualizations helps a lot with debugging
- Learn to use the right tool for programming and collaborating
- Unit tests **would have** saved a LOT of time
- "Brilliance is knowing when to stop" - _anon (2020)

Something Silly



Acknowledgements

Thank you TAs for all your hard work for making this learning experience possible and Prof. Rahul for this opportunity.

"We may win and we may lose, but we will never be defeated."

- _anon (2020)

THANK YOU

Penn Engineering

“Questions, Comments, and Concerns?”

References

1. MPC Lecture slides: https://docs.google.com/presentation/d/1TcleCBkHEoWMcvqR1ykKvhhl7tlENiC6Xy_UVFI3ppl/edit#slide=id.p30
2. Alexander Domahidi Alexander Liniger and Manfred Morari. Optimization-based autonomous racing of 1:43 Scale rc cars. November 2017
3. Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path.International Journalof Control, 87(6):1297–1311, 2014
4. Achin Jain Joseph Auckley Kim Luong Matthew O'Kelly, Hongrui Zheng and Rahul Mangharam. Tech report:Tunercar: A superoptimization toolchain for autonomous racing. September 2019.
5. Goli, S. A., Far, B. H., & Fapojuwo, A. O. (2018). Vehicle Trajectory Prediction with Gaussian Process Regression in Connected Vehicle Environment. IEEE Intelligent Vehicles Symposium, Proceedings, 2018-June(Iv), 550–555. https://doi.org/10.1109/IVS.2018.8500614
6. Hauser, K., & Ng-Thow-Hing, V. (2010). Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. Proceedings - IEEE International Conference on Robotics and Automation, 2493–2498. https://doi.org/10.1109/ROBOT.2010.5509683