Cooperative Parking for Self-Driving Cars

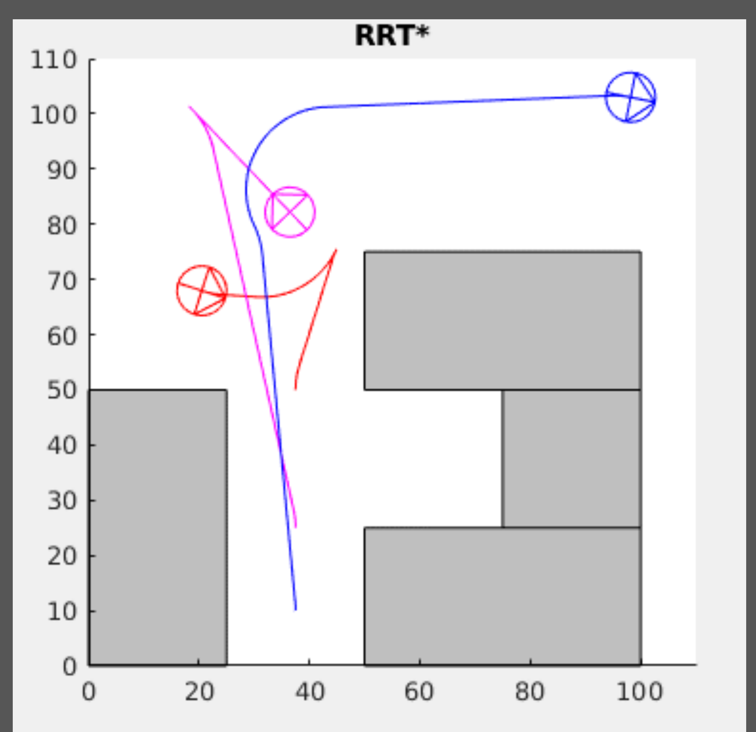
Avery Austin, Heath Camphire, Samuel Schmidgall
Anton Lukyanenko, Damoon Soudbakhsh
Mason Experimental Geometry Lab

Introduction

- Self-Driving cars are quickly being integrated into society, but there are still lacking any motion-planning algorithms that effectively find paths for multiple cars.
- In our work we aim to efficiently, with respect to time, produce a motion-path that near-optimally choreographs cars toward a specified location in our space. The purpose of designing this algorithm is to cooperatively park self-driving cars.
- On top of studying the properties of motion-planning algorithms with multiple cars, we also have developed simulations of the path-finding process and we have used the Mason Autonomous Robotics Labs Flockbots in order to simulate this physically.

Motion Planning Algorithm

- The state-of-the-art path-finding algorithm widely used in robotics is RRT* (Rapidly Exploring Random Trees *)
- RRT* is the path-finding algorithm that we used to predict optimal motion in our non-holonomic system.
- To fit our needs, we extended the RRT* Algorithm to work with with multiple cars while still maintaining convergence properties.
- The way that RRT* works is that it maintains a tree of randomly sampled points, and the index of the parent node that has the lowest path cost to that node. By iteratively sampling new points and adding them to the tree, we effectively generate a path toward every reachable position in our state space.

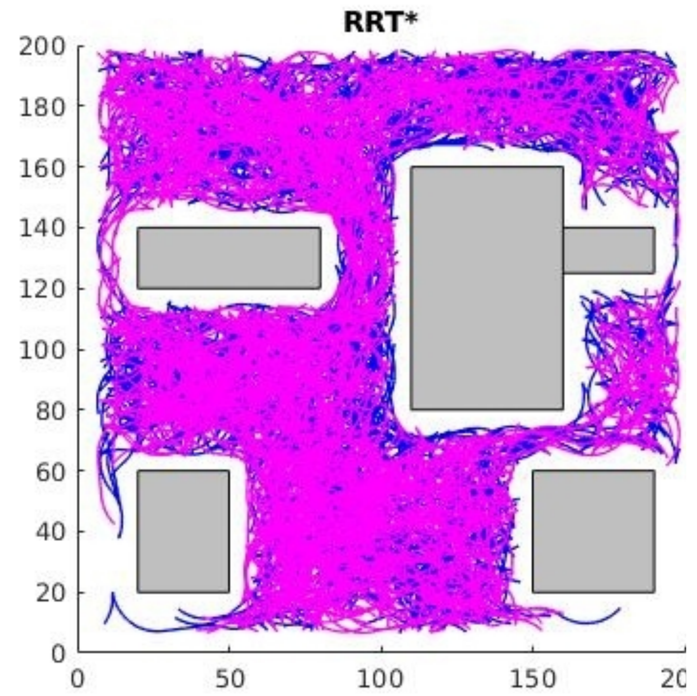


State: A vector in a 3*n dimensional space where n is the number of cars.

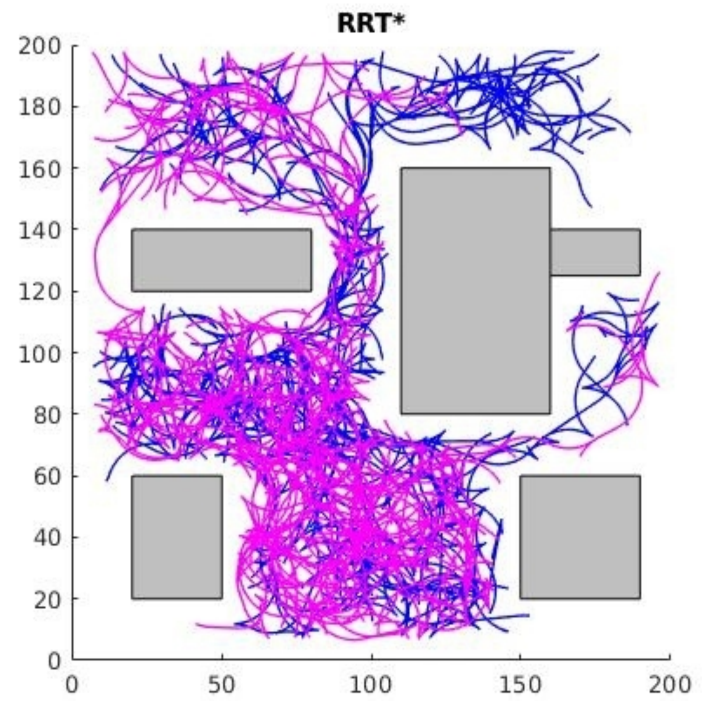
Position: A vector in a 3 dimensional space where the elements of the vector represent x, y and theta respectively

Tree: A collection of states each connected to the respective closest state in the tree_

2 Car Reeds-Shepp Path RRT* Trees



10000 Iterations
1400 Nodes in Tree



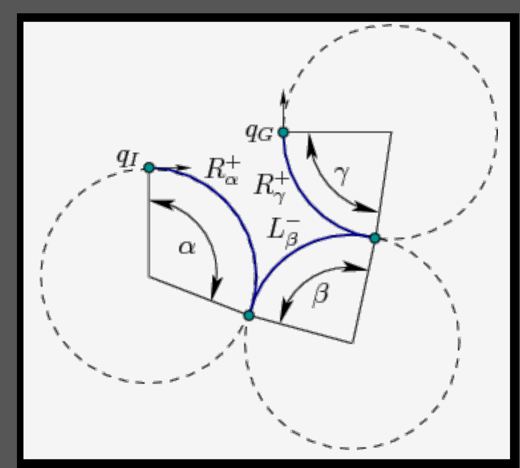
5000 Iterations
350 Nodes in Tree

Computing Optimal Paths

- If you are given two points in an N dimensional space and a car that must follow that path, how can you generate the most optimal path from the starting point to the end point given the differential constraints imposed on the car?
- Reeds and Shepp proposed in 1990 that there are no more than 48 different motion sequences in forward and backward vehicles that provide the optimal path from one point to another in paths that are constrained by turning radius.

Base	α	β	γ	d
$C_\alpha \mid C_\beta \mid C_\gamma$	$[0,\pi]$	$[0,\pi]$	$[0,\pi]$	—
$C_\alpha \mid C_\beta C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha C_\beta \mid C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha S_d C_\gamma$	$[0,\pi/2]$	-	$[0,\pi/2]$	$(0,\infty)$
$C_\alpha C_\beta \mid C_\beta C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha \mid C_\beta C_\beta \mid C_\gamma$	$[0,\beta]$	$[0,\pi/2]$	$[0,\beta]$	—
$C_\alpha \mid C_{\pi/2} S_d C_{\pi/2} \mid C_\gamma$	$[0,\pi/2]$	-	$[0,\pi/2]$	$(0,\infty)$
$C_\alpha \mid C_{\pi/2} S_d C_\gamma$	$[0,\pi/2]$	-	$[0,\pi/2]$	$(0,\infty)$
$C_\alpha S_d C_{\pi/2} \mid C_\gamma$	$[0,\pi/2]$	-	$[0,\pi/2]$	$(0,\infty)$

48 motion sequences in Reeds-Shepp cars



Example of CCC Motion Sequence

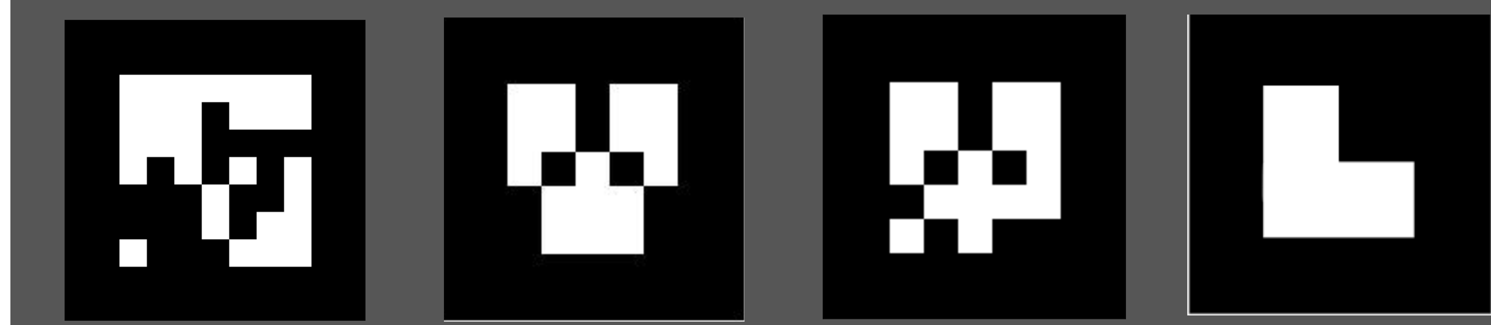
Symbol	Gear: u_1	Steering: u_2
S^+	1	0
S^-	-1	0
L^+	1	1
L^-	-1	1
R^+	1	-1
R^-	-1	-1

The six motion primiitives

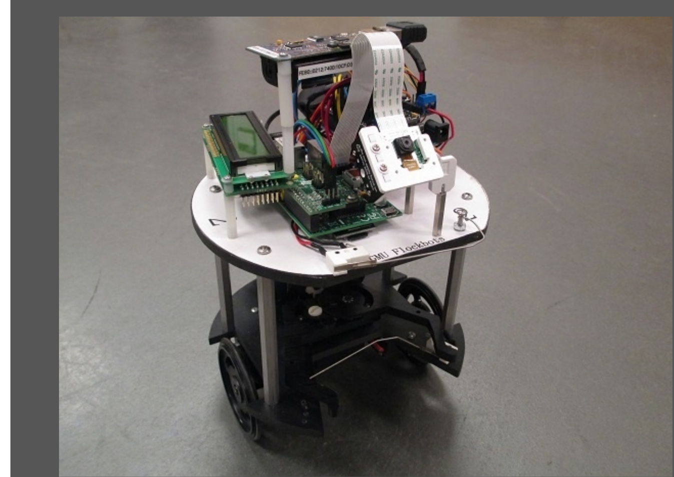
- Although Reeds-Shepp curves provide an optimal path, they are expensive to compute. A large focus of our project was speeding up our code in order to balance the runtime.

Robotics

- Differential drive robots were used to provide a real world representation of the paths generated by our RRT* model
- Each differential drive robot was given a path generated by the model, then made to follow the path using the differential drive kinematic equations accounting for minimum turning radius
- The robots used AR tags to calculate their position based on a global reference frame to provide feedback in the closed loop path following system.



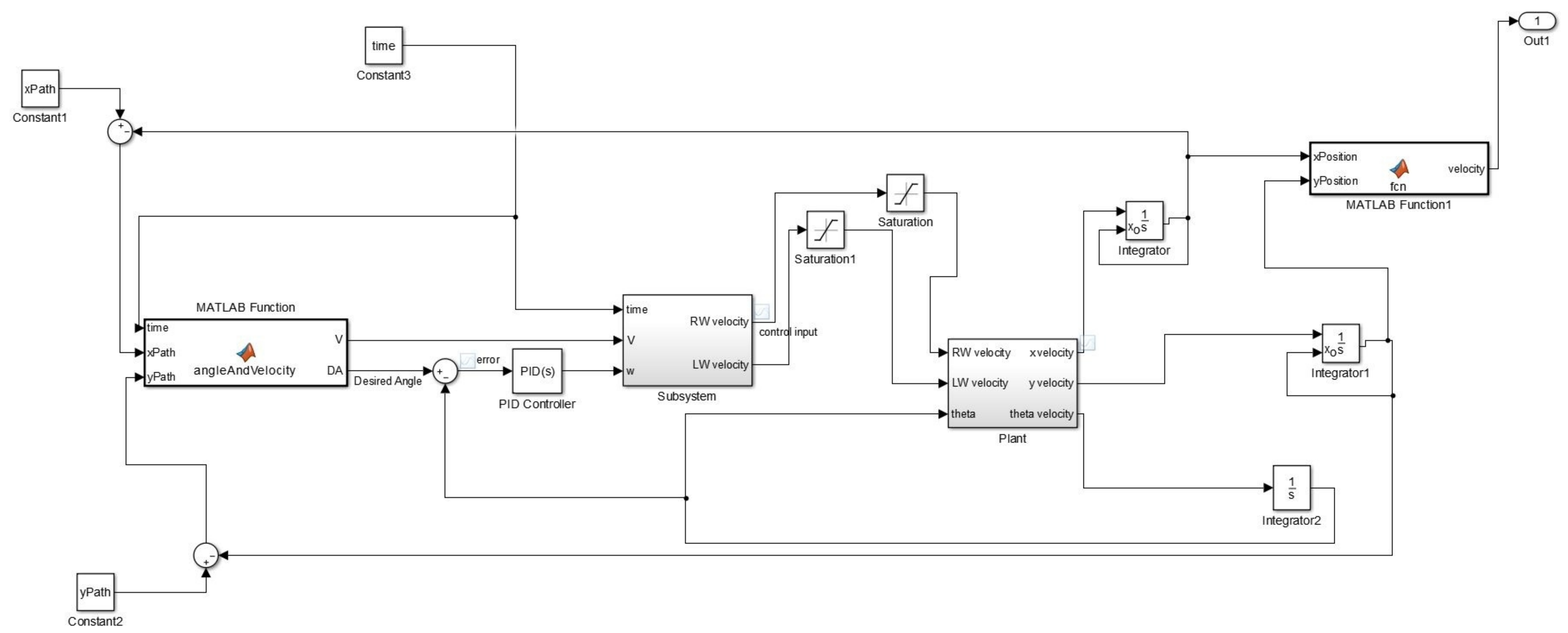
Four AR Aruco Tags



Flockbot Robot

$$\dot{x} = \frac{r}{2}(u_l + u_r)\cos\theta$$
$$\dot{y} = \frac{r}{2}(u_l + u_r)\sin\theta$$
$$\dot{\theta} = \frac{r}{L}(u_r - u_l).$$

Kinematic Equations which govern Differential Drive Robotic motion



Closed Loop Block Diagram representation of the Kinematic Model used to Control Differential Drive Robot.