Cooperative Parking for Self-Driving Cars

Avery Austin, Carlos Guerra, Samuel Schmidgall Heath Camphire, Anton Lukyanenko, Damoon Soudbakhsh

Mason Experimental Geometry Lab

GEORGE MASON UNIVERSITY

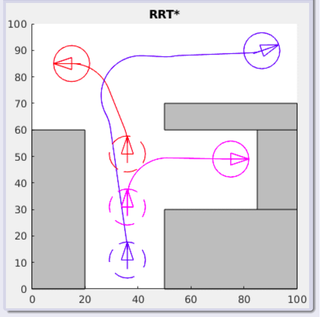
Introduction

- 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.

Path Finding Algorithm

RRT* is an algorithm that retains a tree structure of nodes where the connections between nodes are geometric paths between points in the space. To add a new node to the tree, points are sampled from the state space and then that point is connected to its nearest neighbor. Then, the newly added node is checked for whether or not it makes a more optimal parent compared to the node's parents around it. If so, it becomes the new parent for those nodes.

Three Car Path



Norm Value comparison

- We compared how generating cost using different norm values effected the paths generated
- It was decided that the ideal norm value is L1 and that path optimally decreases with higher norm values up to L_∞

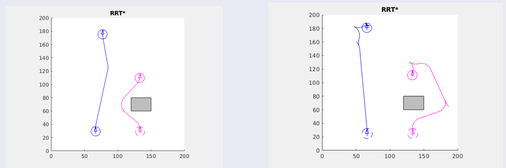


Figure: Comparison of norms: L_1(left) L_∞(right)

Modulus Sampling

Modulus Sampling is a method of sampling that was empirically shown to increase the speed of convergence.

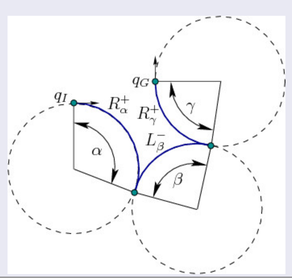
Algorithm 1: Prime Modulus Sampling

$ModulusArray \leftarrow [Prime_1, Prime_2, ...Prime_n]$
Such that n = number of cars and $Prime_m$ is unique
for $i \leftarrow 0$ **to** *iterations* **do**
  positionSample $\leftarrow$ []
  for $j \leftarrow 0$ **to** *numCars* **do**
    if i *mod ModulusArray[j]* $== 0$ **then**
      positionSample[j] = ($goal_j$X, $goal_j$Y, $goal_j\theta$)
    else
      positionSample[j] = (randX, randY, randθ)
  end
end

Reeds-Shepp Metric Space

- 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.
- 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.

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



Robotics

- After the simulations yielded good results we tested the generated paths using robots called Flockbots.
- First, we developed a simple global differential drive line following control system. We then tuned the PID controller and various tolerances.
- Lastly, we replicated the paths for various shapes and RRT* paths using FlockBots.

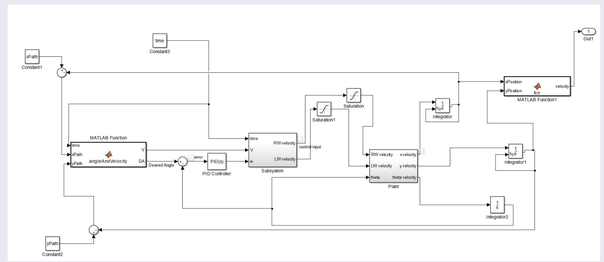


Figure: Differential drive control system

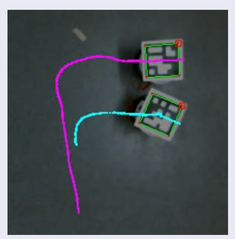


Figure: RRT* Path

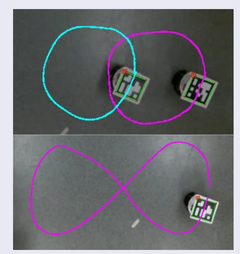


Figure: Other Paths

Future Work

- Integrate simulation code with FlockBot control code.
- Implement real-time obstacle detection and path generation.
- Geometrically prove optimal global parking garage configurations
- Learn the Reed Shepp geometry regions to reduce path generation complexity