CS393R Assignment 1: Obstacle Avoidance

Taijing Chen, Xuefei Zhao, Varshinee Sreekanth

September 2021

Video link: `https://youtu.be/rNg9FCGEwDc`

1 Initial Calculations

Let r_c be the y-coordinate of the center of turning with magnitude r.

r_c has a positive value if the car turns left, and it has a negative value if the car turns right.

Let $w_s := (w + 2 * \textit{safety-margin})$ be car width plus safety margins.

Let $l_s := (l + 2 * \textit{safety-margin})$ be car length plus safety margins.

Q1

$$\text{Maximum arc radius} = \text{outside front of the car} = \sqrt{(r + \frac{w_s}{2})^2 + (\frac{b + l_s}{2})^2} = \frac{1}{2}\sqrt{(2r + w_s)^2 + (b + l_s)^2}$$

Q2

$$\text{Minimum arc radius} = \text{inside rear of the car} = r - \frac{w_s}{2}$$

Q3

Let $r_q := (\sqrt{x^2 + (r_c - y)^2})$ be the distance from the center of turning to the point

r_q Must be larger than the radius of the arc traced by the inner back point of the car, and smaller than the arc traced by the inner front point of the car.

$$r - \frac{w_s}{2} \leq r_q \leq \frac{1}{2}\sqrt{(2r - w_s)^2 + (b + l_s)^2}$$

Q4

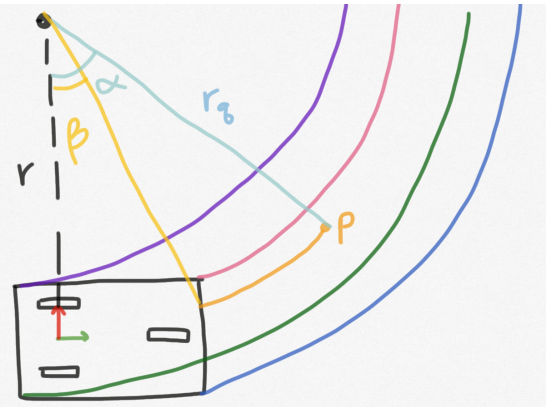
The distance from the point to the center of turning must be larger than the radius of the arc traced by the inner front point of the car, and smaller than the radius of the arc traced by the outer front of the car.

$$\frac{1}{2}\sqrt{(2r - w_s)^2 + (b + l_s)^2} \leq r_q \leq \frac{1}{2}\sqrt{(2r + w_s)^2 + (b + l_s)^2}$$

Q5

Never. The outer edge is always within the sweep volume because the rear outer side is always closer to the center of turning than the front outer side.

Q6



$$\alpha = \sin^{-1} \frac{x}{r_q}$$

Note $\alpha < 0$ if $x < 0$.

$$\beta = \begin{cases} \sin^{-1} \frac{(l_s+b)/2}{r_q}, & \text{if p hits car's front} \\ \cos^{-1} \frac{r-w_s/2}{r_q}, & \text{if p hits car's inner side} \end{cases}$$

if p will hit the car (p hits front OR p hits side):

$$\theta = \begin{cases} \alpha - \beta, & x >= 0 \text{ and } |y| \leq r \\ \pi - \alpha - \beta, & x >= 0 \text{ and } |y| > r \\ \pi - \alpha - \beta, & x < 0 \text{ and } |y| > r \\ 2\pi + \alpha - \beta, & x < 0 \text{ and } |y| \leq r \end{cases}$$

Otherwise, p won't hit the car. In this case, we return max value (free-path length is at maximum)

$$\theta = 2\pi$$

Therefore,

$$l_{\text{free-path}} = \theta \cdot r_q$$

Q7

$$v^2 = v_0^2 + 2ad$$

$$\text{stopping distance} = \frac{-v^2}{2a}$$

2 Additional Questions

2.1 What parameters does your algorithm require, and what is their impact?

- w_0: weight on free path length. By default, this is equal to 1. If $w_1 = 0$ and $w_2 = 0$, the robot will always take the curvature with longest free path length. When $w_1 \neq 0$ and $w_2 \neq 0$, the robot will take other factors into account to make decisions.
- w_1: weight on clearance. Increasing w_1 increases the robot's likelihood to drive through wider openings (bounded). By decreasing w_1, the robot becomes less likely to distinguish paths with narrows openings and wide openings.
- w_2: weight on $maxCurvature - |curvature|$. Increasing w_2 increases robot's likelihood to move straight ahead, and decreasing w_2 increases the the robot's tendency to take turns. We use this parameter to encourage the robot to go straight if the path ahead is unobstructed. Once we implement global navigation, this will change from curvature to distance to the goal.

2.2 How were the parameters tuned?

Tuning was primarily done through trial-and-error. Overall, we made educated initial guesses, and then adjusted the weights for our scoring function based on undesired behaviors we observed our robot exhibiting.

Because the weights are all relative to each other, we always fix w_0 to be 1.0. Then we tested our robots on different maps with different characteristics, and adjusted the weights accordingly. To see if the robot made reasonable path decisions, we visualized all path options (curvatures and free-path lengths), best path options (curvatures and free-path lengths), clearance, and the robot itself with safety-margins on the map. To tune weights, first, on an empty map, we made sure the car moved straight ahead when $w_0, w_1, w_2 \neq 0$. Then, we put the robot in rooms with doors, adjusting the weights so that it could go through the openings. If our robot seemed to curve too much, we increased the weight on curvature. If the robot avoided narrow hallways, we decreased the weight on clearance. After multiple iterations of testing and adjustment, we landed on a set of finely-tuned values where our robot's performance is close to optimal. Lastly, we tested these parameters on various maps and starting locations to see if robot can plan and execute reasonable paths.

2.3 What challenges did you encounter, and how did you overcome them?

1. Our primary challenge was ensuring that our free path length was calculated correctly, as this was the single most important factor in our scoring function. In doing this, we encountered multiple challenges where the robot would attempt to go through walls, or not see obstacles on its right, etc. To overcome this, we recalculated all of our math and realized that there were many more edge cases than we

initially thought when calculating the angle between the car and the end of the free path. Once we found all these special cases, our free path length calculation became correct and our robot stopped trying to pursue paths where there were obstacles or walls.

2. Our second challenge involved our parameter tuning process. Initially, when our robot made an incorrect move, it was very difficult to determine which weight was incorrect. Therefore, we developed an experimental methodology where we reduced all weights to zero and slowly increased only one weight at a time, which allowed us to determine whether the weight on free path length, curvature, or clearance was the problem.

3. Another challenge we had was calculating the clearance. Initially, we thought the algebra would be the same as the one for free path length, but it turned out to be more complex. The points could be anywhere, which makes our prior assumption false. Therefore we redid the math and did many trails on the simulator to ensure we covered all the edge cases.

4. Our final challenge was tuning on the real car. Our code worked perfectly on the simulator, but when we tested on our real car, it kept hitting the wall. After some investigation and debugging, we realized there were some noises in our observation which made the point cloud overlapping with our car. Thus, our code ignored the obstacles that were very close to the front of the car. We included the noises on the point cloud and the car turned out to run elegantly.

2.4 How could you further improve upon your results?

Right now, our robot is generally able to navigate through obstacles and doors without stopping at them. However, there are some inherent limitations of our algorithm that allow some room for improvement. For instance, our robot's performance on very narrow tracks could be improved, because the nature of our algorithm (where we give weight to clearance) means that when a turn is extremely narrow, the robot is slightly more likely to get stuck. This behavior is exactly as intended on normal maps, but on narrow maps with tight turns, the model is not quite as performant. To fix this, we need to implement the extra credit portion of the assignment (U-turns) in order for the robots to get itself unstuck on narrow turns.

Additionally, another improvement we could make is to account for drift and bias in the robot's motion. Right now, if the robot plans to make a turn without too much clearance and drifts into a wall, there's no way to account for that. This can only be improved by implementing Assignment 2, where we account for errors in the robot's location when compared to the planned path and adjust direction accordingly.

2.5 GitHub Repo (Will be made public on 9/21):

`https://github.com/varshinees/cs393r-autonomous-robots`

2.6 Team Member Contributions

We always did pair programming when working, so everyone has the same contribution.

Taijing Chen - Attended work sessions nearly every day, attended in-person work sessions to test out code on the car, spent many hours debugging both with the team and individually.

Xuefei Zhao - Attended work sessions nearly every day, attended in-person work sessions to test out code on the car, spent many hours debugging both with the team and individually.

Varshinee Sreekanth - Attended work sessions nearly every day, attended in-person work sessions to test out code on the car, spent many hours debugging both with the team and individually.

3 Appendix

Referencing the image from Question 6:

Radius calculations:

purple $= r - \frac{w_s}{2}$

pink $= \frac{1}{2}\sqrt{(2r - w_s)^2 + (b + l_s)^2}$

green $= r + \frac{w_s}{2}$

blue $= \frac{1}{2}\sqrt{(2r + w_s)^2 + (b + l_s)^2}$

purple $<$ green, pink $<$ blue

purple $<$ pink, green $<$ blue

Case where green arc is to the left of the pink arc (Ignored in our implementation as per Piazza @28):

$$
\begin{aligned}
r + \frac{w_s}{2} &< \frac{1}{2}\sqrt{(2r - w_s)^2 + (b + l_s)^2} \\
4r + w_s^2 &< (2r - w_s)^2 + (b + l_s)^2 \\
4r + w_s^2 &< 4r^2 - 4rw_s + w_s^2 + b^2 + 2bl_s + l^2 \\
0 &< -4rw_s + b^2 + 2bl_s + l^2 \\
4rw_s &< (b + l_s)^2 \\
2\sqrt{rw_s} &< b + l_s
\end{aligned}
$$