Creating a Parallel Parking Robot

Nathan Moeller, Zhanglun Li, Baohua Sun, Jiawei Mo

December 16, 2014

1 Abstract

In the last two decades, automated parking technology has attracted significant attention in the automobile industry. Parallel parking for humans requires extensive skill, experience and practice. Even very experienced drivers may damage their vehicle due to carelessness or complexity of the environment. In this paper, we will discuss methods and techniques we used to achieve an automated mobile parallel parking robot. The key techniques in the automated robotic parallel parking problem include obstacle sensing, point stabilizing, trajectory calculation, trajectory tracking, and decision making based on the parking spot size limitations. The implementation of each step is discussed in this paper.

2 Introduction

Parallel parking is a difficult task for many drivers. It requires correct positioning and skilled turning ability that a lot of people do not possess. Luckily, parallel parking can be automated with an intelligent system that can read it's surroundings. To simulate this, we've created a parallel parking robot using a Pioneer 3 robot with a SICK laser scanner. Our simulation

is based off a real world parking scenario, meaning the turning radius of our robot has been limited to that of a real car, and cardboard boxes have been placed to represent parked cars.

3 Literature Review

Wang and Cartmell (1997) described the nonholonomic motion planning problem, i.e. the problem of finding the input needed to control a robot vehicle from a given initial configuration to a final configuration. [3] As described by Wang et al, the difficulty of this problem is being able to find a good travel curve that is able to satisfactorily simulate a human's driving behavior, and satisfy the steering limit of a real vehicle. The article provided 3 mathematical models. They are quintic polynomial, cubic polynomial and a triangular function for performing a parallel transfer maneuver. The paper also suggests that triangular function can be used to do the parking maneuver, which will generate a smooth and symmetric curve. The paper has compared simulations of the three mathematical models and concluded that the triangular function is the most efficient in terms of the final position of the vehicle in the x direction. This paper is useful for our work to determine which mathematical model to use to calculate a trajectory path for the robot to follow. We did not use the quintic polynomial and cubic polynomial model because of their inefficiency.

In the paper "Automatic Parallel Parking of Car-Like Mobile Robots Using Saturation Control and Bang-Bang Control", the parallel parking problem of wheeled mobile robots is considered [1]. This paper proposed two types of steering controllers for straight line tracking: bang-bang (optimal) control and SAT control. The bang-bang control described in this paper is exactly the method we are using. It uses the maximum steering angle to the left to finish half of the trajectory, it then uses the maximum steering angle on the opposite direction to complete the other half of the trajectory. The trajectory function is

the triangular function as described in the first paper. We did not end up using the SAT Control because it has a much more complicated math function. Also, the only advantage of SAT control is that it is continuous and the chattering is avoided. Since the chattering is out of our concern, we did not use SAT control model.

4 Problem Description

To simulate parallel parking in the real world, we used a Pioneer 3 robot with a SICK laser scanner to represent the car. The parked cars were represented as boxes. The problem is formalized as following: How can the robot find the parking spot while going forward along a wall and perform a parallel parking maneuver once a suitable spot is found? The following assumptions were made for our experiment:

1) The robot is initially moving parallel to the wall and the parking spot is on robot's right.

2) The robot is not moving faster than 0.1m/s so that laser scanner can have enough time to read in accurate data.

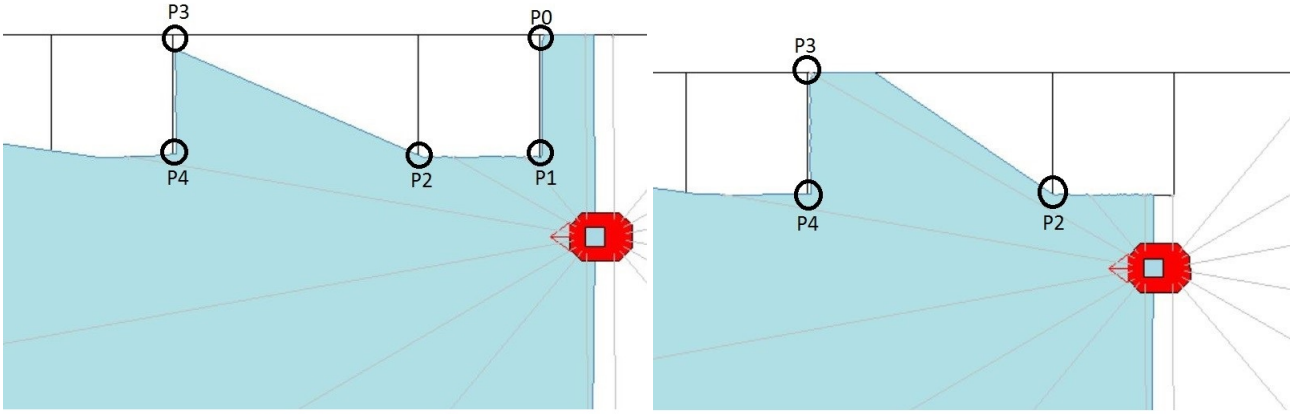
3) All boxes are cuboid which have obvious edges and corners. All boxes cannot be smaller or lower than robot. Otherwise, it may confuse the laser scanner.

4) No obstacles will appear during the whole parallel parking procedure.

5 Problem Solution

The first part of our solution was for the robot to detect a parking spot. We did this using the readings from the laser scanner, which gave us a distance for every angle. As the robot approaches a potential parking spot, the laser scanner is taking readings of its surroundings. In order for the robot to know what a spot is, we defined two configurations the robot could

be in. Configuration 1 was when the robot has not detected a potential parking spot, so it is simply driving forward. Configuration 2 was when the robot has passed the first box, and sees a potential spot to park. The idea for configurations was modified from "Robotics Project" [2]. The two configurations are shown in the graphic below.



Configuration 1 **Configuration 2**

Configuration 1 is defined as the robot being to the right of P0 and P1. To find P0, the robot first logs the distance from the robot to the wall. Then, looping from the angle straight to its right to straight forward, the robot looks at every angle's distance to the wall. We know that every distance should be increasing, because the robot is driving parallel to the wall. Once the robot sees a decreasing distance, the corner is detected and P0 is logged. A similar idea is used to find P1, but the robot looks for an increase in distance instead of a decrease. As the robot drives the angles for P0 and P1 are logged at each reading. We noticed that as robot drives closer to the first box, the angle for P0 approaches -90. Once it drives past the first box, P0 will still be detected but the angle dramatically increases towards 0. Once this happens, the robot knows that it is now in configuration 2, and it has actually detected P3. From here, we used the same idea to detect points P2 and P4 as we did to find points P0 and P1. Once the robot is in configuration 2, it knows the position of points P0 through P4 and it can calculate the size of the spot. To calculate the size of the spot, we converted P2 and P4 to Cartesian coordinates and subtracted their x-values. If

the difference is greater than 1 meter, the robot will pull forward to just beyond P4. If the spot is too small, our robot resets to configuration 1 and continues to look for a potential spot.

After finding the available spot, our robot can start parallel parking. Before stepping on the accelerator pedal, we need to veer the wheels. We used the 2015 Toyota Corolla as a model to get the parameters for our parking maneuver. A 2015 Toyota Corolla's turning radius is 10.85 meters and its width is 1.775 meters. Our robot, which is Pioneer 3-DX, has dimensions of 455mm long, 381 mm width, 237mm height. For convenience, we used the Toyota Corolla's center turning radius instead of its outer wheel turning radius. The Corolla's center turning radius is $(10.85 - 1.775/2) = 9.96m$. We then put the radius in terms of our robot:

$$\frac{.381}{1.75} = \frac{R}{9.96}, R = 2.168m \tag{1}$$

However in the automobile industry, the turning radius is a diameter instead of a radius. We simply divide R by 2 to get our final radius for our robot:

$$R = 1.084m \tag{2}$$

Unfortunately, the Pioneer 3-DXs wheels cannot change direction. Alternatively, we set different speeds for the left and right wheels to mimic turning the steering wheel.

There are two kinds of available spots based on the spot length. Case one is that the spot is long enough so the robot can reverse into it without multiple correcting maneuvers. The second case is that the spot is small, so the robot may need to perform multiple correcting maneuvers to get closer to the wall. Case 1 is the simple case, and can be seen in the figure below.

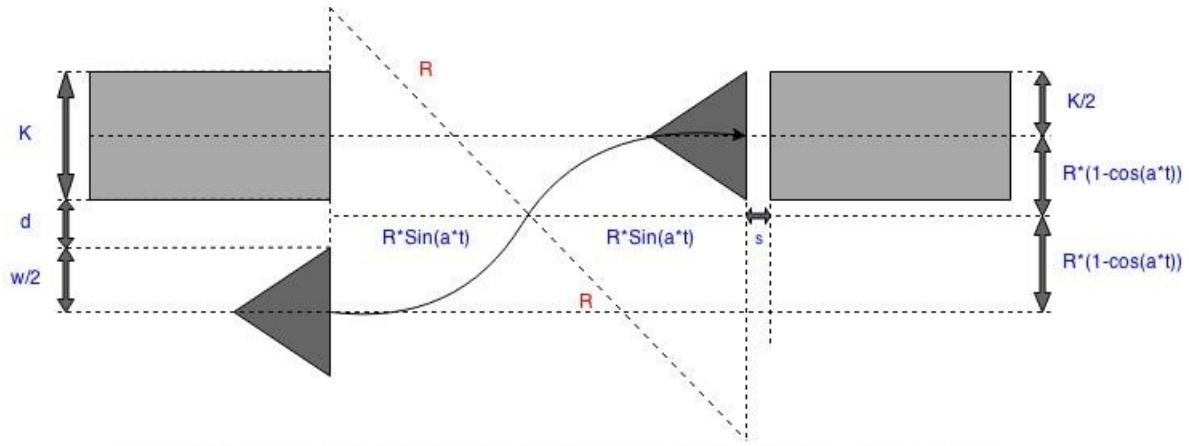


Figure 1: K(box width), R(Rotation radius), a(angular velocity), t(halfway time), s(safe distance to back), d(distance to box), w(robot width)

Every variable in figure 1 above is either known, or can be calculated. For instance, we can calculate time (t) using the formula:

$$t = \frac{2R * arccos(1 - \frac{K+w+2d}{4R})}{2.426 * Vi} \tag{3}$$

We can calculate the distance of the spot for case 1 using this formula:

$$D = Rsin(at) + Rsin(at) + s \tag{4}$$

The robot will back into the spot, and will stop when it is a distance of s away from the box.

When the length of spot is less than D of (4) but still available for parallel parking, we have to maneuver the robot to get closer to the wall step by step until the distance is within a threshold we set.

This is case 2. It is obvious from figure 2 to see that after every maneuver, the robot is 2R(1-cos(at)) closer to the wall. To be safe, we reduced the speed difference of the two wheels by 0.95 and let it move half of t in (3). This ensures it stays within the spot and doesn't hit the wall or the boxes.

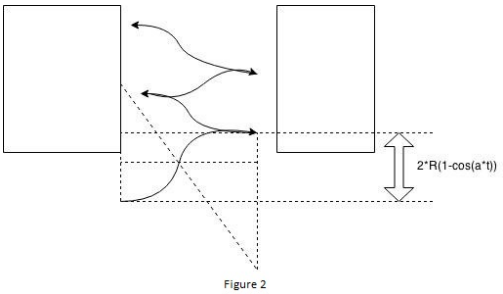


Figure 2

6 Results

Our threshold for a valid spot size is 1.0m, but it turns out is not exactly 1.0m. This error is caused by the way we do detecting. When we are scanning the spot size from a relatively far position, the data from laser scanner is slightly inaccurate. This will lead to a theta close to 0. So according to the equation $p.x = distance * cos(\theta)$, p.x is changing very fast when theta close to 0, which will cause a large error in the laser data. This error is still acceptable in our experiment. The table below shows our results in detecting a spot.

	Calculated Size	**Actual Size**	**Robot Reaction**
1	1.277m	1.3m	Accept
2	1.197m	1.2m	Accept
3	1.101m	1.1m	Accept
4	0.923m	1.0m	Decline
5	0.911m	0.9m	Decline

We achieved successful results for the parallel parking maneuver. When the available spot size is large enough (larger than 1.3m in our experiment), the robot completes a one-time parking maneuver (case 1). This is the easiest way to ensure the body of robot is completely inside the spot. For smaller spots that still qualified, the robot takes multiple maneuvers to gradually park into the suitable position (case 2). The robot stops when the distance to wall is less than 0.2m and larger than 0.1m. Our results can be seen in the table below.

	Initial Distance to Wall	Ending Distance to Wall	Spot Size	Park Time	Maneuver
1	0.8m	0.4m	2m	6.6s	One-time
2	0.8m	0.4m	1.5m	6.6s	One-time
3	0.8m	0.4m	1.3m	6.6s	One-time
4	0.8m	0.15m	1.1m	6.3s	Multiple
5	1.0m	0.18m	1.1m	7.8s	Multiple
6	1.2m	0.13m	1.1m	9.5s	Multiple

7 Conclusion

We were extremely satisfied we our results. Our robot successfully parallel parked for various sized spots. However, many more improvements could be added. One needed addition would be to get rid of the the assumption that the robot is driving parallel to the wall. This would be a required improvement to generalize this algorithm for the real world. Another improvement would be to add obstacle detection. If there is anything blocking our robot from parking, it will currently fail. In the real world there may be obstacle preventing the robot from parking, so this would be a needed addition.

References

[1] D. Lubomir P. Plamen. Automatic parallel parking of car-like mobile robots using saturation control and bang-bang control. *Recent Advances in Circuits, Systems, Telecommunications and Control*, 2009.

[2] Thomas Bishop Peter Bailey, Matt Beckler and John Saxton. Robot project. 2007.

[3] Yongji Wang and M. P. Cartmell. Autonomous vehicle parallel parking design using function fitting approaches. *Robotica*, 16(2):159–170, 1998.