Robotics Project

Final Report

Computer Science 5551

University of Minnesota

December 17, 2007

Peter Bailey, Matt Beckler, Thomas Bishop, and John Saxton

Abstract:

A solution of the parallel-parking problem has been created to work with the Pioneer 3 robotic system. Line detection and interpretation of SICK laser scanner data is used to provide an accurate estimate of parking spot location. A dual-circle navigation path approach has been implemented to execute the actual parallel-parking maneuver.

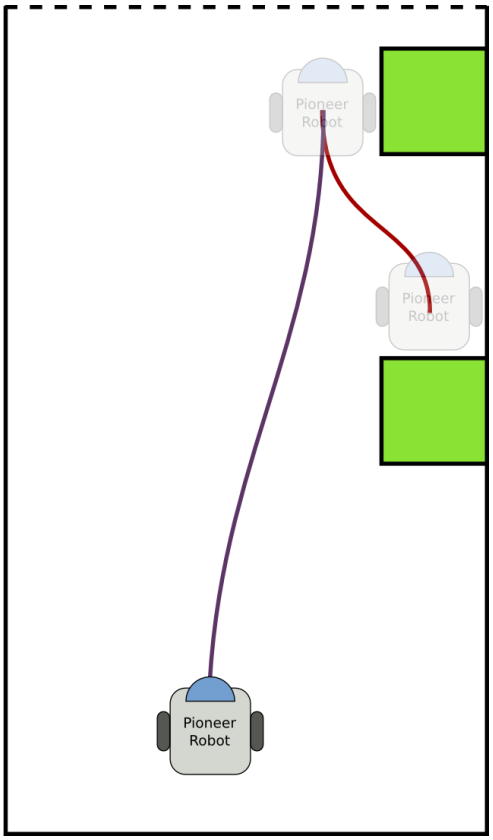
Introduction

Parallel parking a passenger vehicle is commonly held to be one of the most difficult maneuvers regularly performed by most drivers. For many drivers, it is a lengthy, error-prone, and time consuming task that most would rather not do. Our challenge is to develop a method of performing a parallel parking maneuver automatically, using a Pioneer 3 robot as our vehicle. We can extend this knowledge gained to real-life use relieving drivers from the daily hassle of executing the parallel parking maneuver themselves. To model the car-parking situation, we have restricted the turning radius of our robot, as well as using two robot-sized boxes to represent the parked vehicles defining our parking space.

Prior Work

Problem description

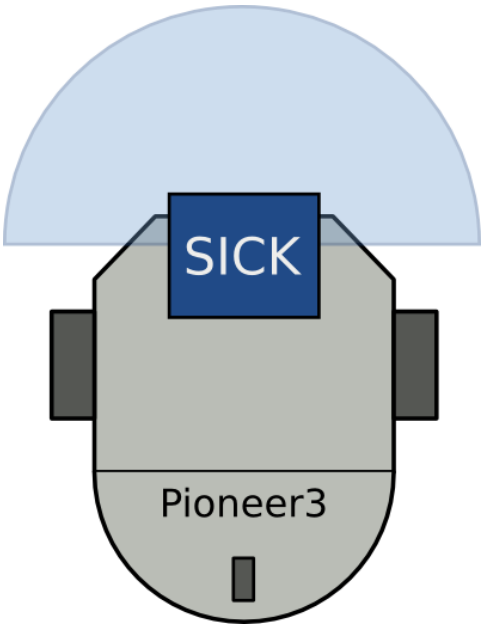
Our problem starts with the robot facing down the hallway, with two boxes defining the parking spot located along the right-hand wall. The robot should approach the parking spot by driving along the wall, much as a standard passenger vehicle drives down a street. The parking spot must be recognized and measured by the robot, to ensure that there is enough length and width to the parking spot for the robot to fit. Once the location and orientation of the parking spot has been determined, the



robot must navigate to a proper position and heading before executing the parking maneuver. The actual parallel-parking maneuver will be performed from just outside of the front obstacle. Our approach can be simplified into a series of discrete steps:

1. Detection of parking spot.
2. Wall following
3. Approach Alignment
4. Execution of parking maneuver

Equipment Specifications



We are working with the Pioneer 3 mobile robotics platform, created by ActivMedia Robotics, shown in the image at right. Our particular model has been outfitted with a SICK laser scanner, capable of measuring radial distances in a 180° arc. Using wheel encoders, the ARIA software API tracks the robot's position and heading as it navigates through the environment. ARIA also provides a suite of line detection and filtering routines that enable the SICK laser data to be directly interpreted and converted into lines with respect to the global coordinate system.

Assumptions

In the real world, parking spots are normally well defined. In our case, we will also ensure that the parking spot is well defined. Our robot's world will consist of a hallway in the EECS building with parallel walls. The robot will park between

two flat-sided boxes placed adjacent to the right-hand wall. All movement and object detection will be on a horizontal plan, at a height determined by the mounting height of the SICK laser scanner. The boxes can be different sizes, and at varying distances apart. The robot does not need to be initially aligned with the wall, and it has no requirements for initial distance from the parking spot.

The Pioneer 3 robot has only two drive wheels, with a small caster wheel behind to provide balance. This robot is holonomic, meaning the number of controllable degrees of freedom is greater than or equal to the total number of degrees of freedom. For this problem, movement and rotation in a plane, there are three degrees of freedom in total, namely the x-position, y-position, and angle of rotation. With only two wheels, we can independently control all three of these degrees of freedom. An example of a non-holonomic robot would be a standard car, where we are unable to independently control all three degrees of freedom. Since the non-holonomic case is more interesting and challenging, we will be restricting our robot to execute the parallel parking maneuver using a minimum turning radius of 450 mm, a distance of about 18 inches.

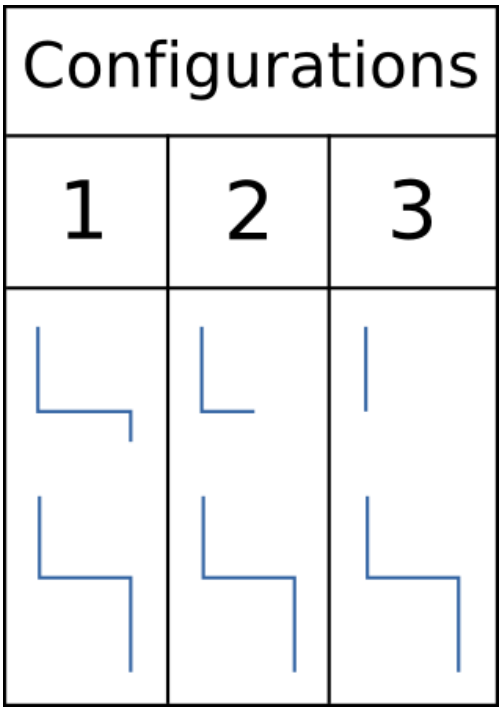
Initial Experimental Results

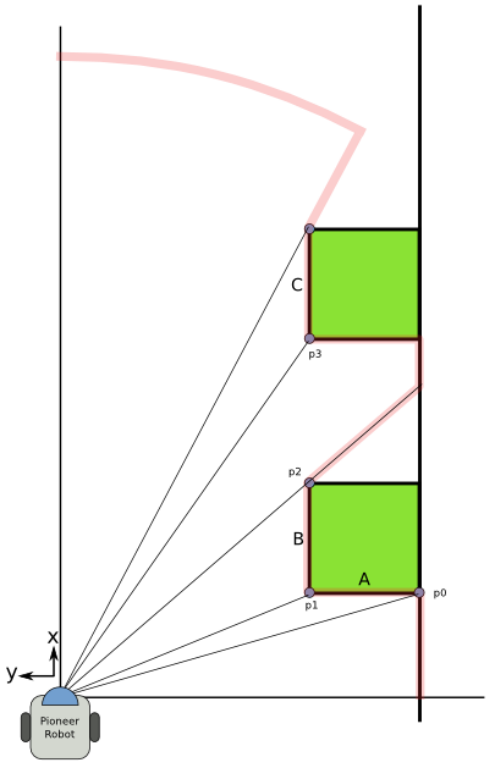
This is a description of our initial work, that we developed and perfected using the simulator. It did not work as well as hoped with the actual robot, and was redeveloped. The new system is detailed in the next section.

The first step of the original parking algorithm is to locate straight lines. Rather than write our own algorithm to perform this task, we used the line detection functions built into the ARIA API. Not only does this algorithm detect lines, but it is also intelligent enough to discard smaller lines that may not actually

correspond to a line.

The next step of our algorithm is to extract the positions of the boxes from the line data. This is complicated by the fact that not only will the magnitudes and angles of the lines change as a function of position, but the number of lines will also change. As such, the first thing we did was come up with three possible line configurations. When we are far away from the parking spot, we will only see the left hand side of the far box. We called this configuration 3. As we approach the parking spot, the front of the far box becomes visible. We called this configuration 2. It is also possible that we would see the entire front side of the far box and part of the wall. This is called configuration 1. All three configurations are shown in the figure to the right. This is important, because if we know which configuration we are in, we can find which endpoints of which lines correspond to p0-p3 (see left hand figure).





Writing code to determine which of the three configurations we were currently in was fairly straightforward. Each configuration is defined by how the lines are aligned relative to each other: it is a function of angles, not length. Using this

observation, we can conclude that each configuration will have its own “fingerprint.” With this is mind, one can generate a fingerprint for each configuration and observe that each configuration has its own unique fingerprint.

Once one realizes that each configuration has its own unique fingerprint, writing the algorithm becomes easy. One first iterates through all the lines, generating a list consisting of the angles between each line. Next, one iterates over this list searching for a fingerprint corresponding to one of the three configurations we defined. If we find one, we extract the points p0-p3. If not, we blindly drive forward, hoping that this will bring us closer to the parking spot.

This method worked perfectly in the simulator. Our algorithm correctly identified which configuration we were in, and it provided us with accurate data for p0-p3. We then drove to the parking spot and successfully executed the parking maneuver. However, our algorithm rarely worked in practice. We simply couldn't determine what configuration we were in. After some debugging, we realized this was because we were having difficulty detecting the sides of the boxes. This is probably a combination of us having a poor angle on the sides of the boxes and the fact that the boxes seemed to be moderately reflective. Regardless of the cause of the problem, we concluded that this algorithm wasn't robust enough for our purposes and we set out to write something better.

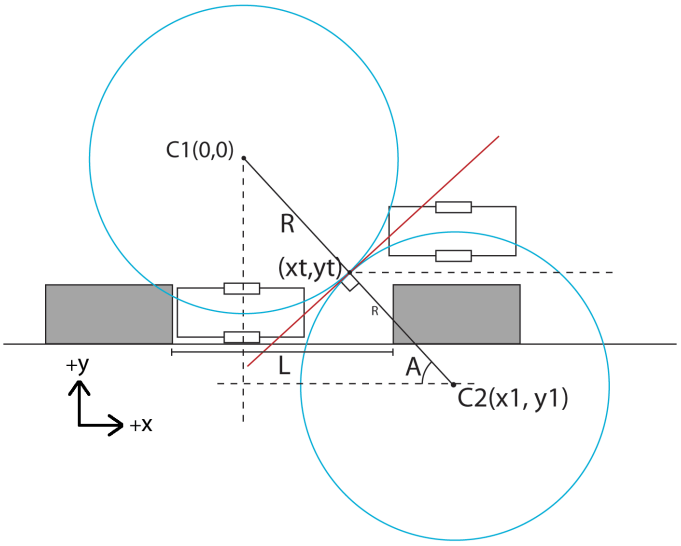
Our Second Attempt

We concluded that if we wanted to get the robot to work properly, we couldn't count on being able to see the far box properly. As such, we settled upon using a method that initially only scanned the first box. In this algorithm we use

the SICK laser scanner to try to find the first box. We use the same fingerprint principle to try to find the first box. Once we find the first box, we drive slightly past it and scan for the second box. If we find a second box, we know we have found a suitable place to park, and we get into position to execute the parking maneuver.

Parking Maneuver

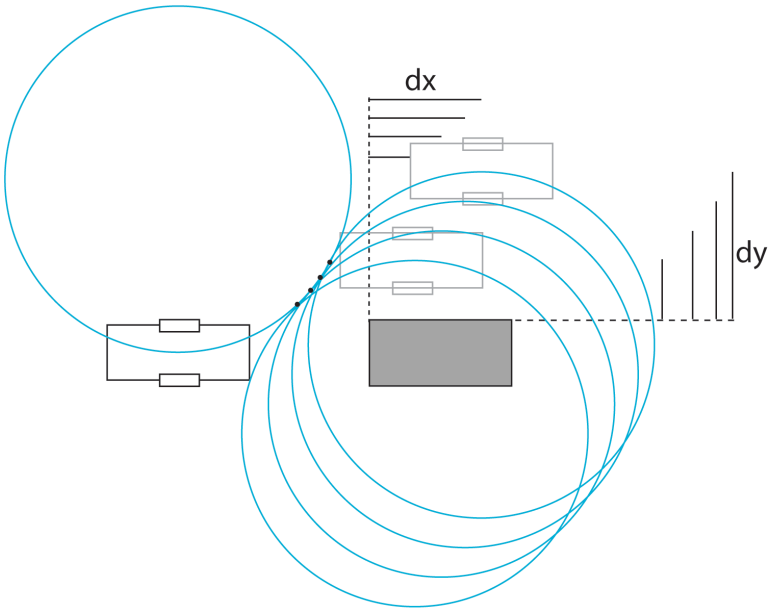
The algorithm decided upon is based on the concept that a parallel parking procedure is comprised of two arcs of equal length by identical circles, C_1 and C_2. C_1 is R distance in only the positive y-direction from the goal parking position. C_2 is the circle relating to the initial position whose center is located a distance R in the negative y direction. In both the initial and desired positions, the line connecting the center of the robot to its respective circle center is perpendicular to the x-axis. The initial position begins at the center of C_2 minus a distance R in the y direction. These circles are equal in size, with radius R, the tightest turning radius possible. In the case with a dual wheel robot a constraint was placed on the wheels to emulated a four wheeled car with turning radius similar to that of a physical car. For the Pioneer 3, the minimum turning radius was decided to be 450 mm, which is slightly longer then the width of the robot.



The following assumptions are made when entering into this phase:

1. The robot is parallel to the wall and boxes.

2. The distance between the boxes is wider then the length R plus the distance from the center of the robot to its rear.
3. The target location is known
4. dy, the y-distance from the edge of the front box to the edge of the robot is known.



Because the position of C_1 and d_y is static, d_y is the only unknown. Dx must be found to enable C_1 and C_2 to osculate creating a tangent line between them. Once this condition is accomplished the arc procedure may begin. For every d_y there is a specific corresponding d_y that places the circles in location. The center of C_1 is assumed from here on to be the origin. The robots coordinates are then based off of this origin, plus some padding, (x_{robot}, y_{robot}). Using angle A, the angle that is part of the triangle connecting C_1, C_2 and the y-axis, trigonometry is used to obtain the distance the center of C_2, (x_1, y_1) is from the origin. Calculating C_2 (x_1, y_1) uses the following equations:

$$\sin(A)=\frac{y_1}{2\mathrm{R}}$$
$$A=\sin^{-1}(\frac{y_1}{2\mathrm{R}})$$
$$\cos(A)=\frac{x_1}{2\mathrm{R}}$$
$$x_1=2\mathrm{R}\cos(A)$$

Now that x_1 is known, the robot is repositioned along the x-axis to align C_2 For each possible configuration of the two circles, the turning point is going to create a symmetric divide forcing the two arclengths along each circle to have the same length. The arclength is determined using angle A through the following equation: $Arclen=R(\frac{\pi}{2}-A)$. The robot then uses the helper function, moveInCircle() to move along both arclengths and into its final position. The parameters of the function moveInCircle() are an angle, a radius, and a clockwise boolean. It always performs movements in the backward direction because that was all that was required within the scope of this project.

Future Work

While we have created a system to successfully parallel park our robot, there are many areas of improvement. The foremost would be general improvements to the algorithms we use for detection and parking. For the parking algorithm, we used a simple dual-circle path planning algorithm, and it would be interesting to experiment with alternative algorithms, such as a sinusoid path, or continuously updated path planning using a PID controller. Another area for improvement would be in the detection of the parking spot from the laser data. Our current approach gives a reasonable amount of error, since it only uses one reading. If we were to take multiple readings, we would be able to minimize the

error, and provide more precise positioning data.

In terms of useful features, we would like to add the ability to use the laser scanner to watch for obstacles while we are backing up. If there were sonar devices mounted to the rear of the robot, it would also be beneficial to use them to detect the vehicle to the rear, to provide last-minute warnings of a collision when backing up. Also, once a parking spot has been identified, it would be useful, especially for a real-world agent, to use the laser scanner to identify any obstacles in, and fully characterize the parking spot on approach. Our current agent only uses the laser scanner initially to identify the first box, and then to identify the second box once it has reached the first box. Gathering data during the driving from first box to the second box would be useful, and would help improve both our estimate of the robot's position, as well as the overall accuracy and precision of the parking.

To finish the parallel-parking problem, it would be relatively simple to add the ability to pull-out and leave a parking spot, using the laser scanner to identify the dimensions of the vehicle in front of the robot. We could then calculate the path to leave the parking space without colliding with either of the bounding vehicles.

Currently, after executing the dual-circles maneuver to back into the parking spot, the robot pulls forward a distance of 1/3 the total length of the parking spot, as calculated by the laser data. This is certainly non-ideal, and it would be better to actually calculate the best distance to pull forward, in order to leave the robot evenly spaced between the two boxes.

Overall, our work on this project relied on a number of assumptions that

would not necessarily apply to a real-life agent in the real world. We use a very simple algorithm to count detected lines from immediately to the right of the robot's laser scanner, which requires that the first thing to the right of the robot's laser arc is the right wall. If this is not the case, it will have trouble aligning to the wall, and will be unable to identify the boxes. In the future, it would be good to remove this restriction, perhaps by also looking to the left wall of the hallway for alignment data.

References

1. Paromtchik, Igor E. and Laugier, Christian. “Autonomous Parallel Parking of a Nonholonomic Vehicle”, Proceedings of the 1996 IEEE Intelligent Vehicle Symposium.
2. Murray, Richard M. and Sastry, S. Shankar. “Nonholonomic Motion Planning: Steering Using Sinusoids”, IEEE Transactions On Automatic Control, Vol. 38, No. 5, May 1993.
3. Paromtchik, Igor E. and Laugier, Christian. “Automatic Parallel Parking and Returning to Traffic Maneuvers”, Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems.