UNIVERSITY OF WATERLOO
Department of Systems Design Engineering

ME 597: Assignment 1

prepared by

Sarah Elliott Leigh Pauls
November 1, 2013

1.0 Bicycle Model

The bicycle model has a couple of

2.0 Carrot Planner

The robot needs to track a rectangular path. The path A is defined by 4 points.

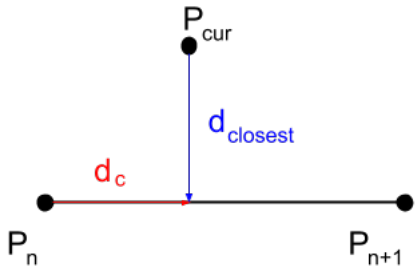
$$P = [\ p_1 p_2 p_3 p_4\ ] \tag{1}$$

To find the closest point on the path you need to know which part of the path is closest. We can imagine the path as having 4 segments, one for each side of the rectangle. Segment $S(p_n, p_{n+1})$ is the segment with start and end points at p_n and p_{n+1} respectively, where $n = 1, 2, 3$. For each S, we need to find the closest point on that segment. Then we will compare the closest points on each segment to see which of the segments is the closest overall.

Let p_{cur} be the current point. Let $r_{p_n/p_{n+1}}$ be the distance between those two points. This notation is extended to all other distances between points. The distance calculation is fairly trivial and not included in this report. Let's define the distance from the first waypoint, p_n, of the current segment to the closest point on the line as d_c.

$$d_c = r_{p_{cur}/p_n} * \frac{r^2_{p_{n+1}/p_{cur}} - r^2_{p_n/p_{cur}} - r^2_{p_{n+1}/p_n}}{-2 * r_{p_n/p_c ur} * r_{p_{n+1}/p_n}} \tag{2}$$

The figure below shows an example of d_c. Other point configurations are possible. To



find the closest point on the line we have to consider a number of cases. If the points are oriented as above, finding the closest point is fairly straight forward. You just add d_c to the location of p_n. However, what if p_{cur} is closest to the top edge of the rectangle and moving counterclockwise? This would mean you need to subtract d_c from the location of p_n. Other cases include if d_c is negative or greater than r_{p_{n+1}/p_n}; for example if p_{cur} is outside the rectangle. To take all these situations into account:

$$p_{closest} = \begin{cases} p_n & d_c < 0 \\ p_n + d_c * \frac{p_{n+1} = p_n}{r_{p_{n+1}/p_n}} & r_{p_{n+1}/p_n} > d_c > 0 \\ p_{n+1} & d_c > r_{p_{n+1}/p_n} \end{cases}$$

This is repeated for every line segment and the line segment where the distance between p_{cur} and $p_{closest}$ is chosen. Then that $p_{closest}$ is the overall closest point. To set the actual desired carrot position, check how mush distance is left on the line until the next waypoint. If the

distance left is less than r =*1* then the next waypoint is the carrot position. Otherwise, it computes the carrot position as follows:

$$p_{carrot} = \begin{cases} p_{n+1} & r > r_{p_{n+1}/p_{closest}} \\ p_{closest} + r * \frac{p_{n+1}+p_{closest}}{r_{p_{n+1}/p_{closest}}} & r < r_{p_{n+1}/p_{closest}} \end{cases}$$

To steer towards that position we find the the vector distance between the current position and carrot position, $r_{p_{carrot}/p_{current}}$. Then the desired heading is:

$$h_{desired} = arctan(\frac{r_{p_{carrot}/p_{current}x}}{r_{p_{carrot}/p_{current}y}}) \tag{3}$$

$$h_{error} = \begin{cases} h_{error} - 2 * \pi & h_{error} > \pi \\ (h_{desired} - h_{t-1})\%2 * \pi & h_{error} < \pi \end{cases}$$

We make the steering angle using proportional control (in this case we multiply our steering angle error by 1 to get a new steering angle).

$$\delta = 1 * h_{error} \tag{4}$$

Then we just feed that steering angle into the bike model from the previous question. The resulting plot of the robot driving around the rectangle is shown below. It starts with an abitrary position at (10, 1) with a heading of 90°.