Implementation of the Pure Pursuit Path Tracking Algorithm

R. Craig Coulter
CMU-RI-TR-92-01

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January 1992

Table of Contents

List of Figures

Abstract

The main purpose of this technical report is to describe in detail the implementation of the pure pursuit path tracking algorithm. Given the general success of the algorithm over the past few years, it seems likely that it will be used again in land-based navigation problems. This report also includes a geometric derivation of the method, and presents some insights into the performance of the algorithm as a function of its parameters.

1.0 Introduction

The pure pursuit algorithm has been used at the Robotics Institute for a number of years. First on the Terragator, then on the NavLab and more recently, on the NavLab II (also called the HMMWV). Amidi[1] implemented and tested this algorithm under a variety of conditions, and found it to show the greatest promise as a general purpose tracking algorithm.

The main purpose of this technical report is to describe in detail the implementation of the pure pursuit algorithm. Given the general success of the algorithm over the past few years, it seems likely that it will be used again in land based navigation problems. This report also includes a geometric derivation of the method, and presents some insights into the performance of the algorithm as a function of its parameters.

1.1 History

The pure pursuit algorithm was originally devised as a method for calculating the arc necessary to get a robot back onto a path. This first application of the method came with the Terragator, a six wheeled skid steered robot that was used for outdoor vision experimentation in the early 80's. The standard references for the original derivations of the work go to Wallace[3].

When the work on the Terragator moved to the then new NavLab, the arc commanding algorithm followed. Throughout the NavLab project a number of path tracking algorithms were proposed and implemented, including the Quintic Polynomial approach and a "Control Theory" approach. Testing of all of these algorithms showed that the Pure Pursuit method was the most robust and reliable method going. Amidi[1]'s masters thesis contains the results of his comparison of the three aforementioned methods.

After the NavLab II (a.k.a. the HMMWV) was built we opted to use the pure pursuit tracker, based on its reliable performance. Our software team was busy developing other pieces of code for the planning, the dynamics, and the perception modules and we really didn't want to build a tracker from scratch. We copied some old pure pursuit tracking code onto the NavLab II and got it working and used it pretty steadily for about three months. We were a little disappointed in its performance, but found it acceptable. It would track most of the paths that we gave it, but occasionally lost a path completely.

The code that we had copied was experimental left-overs from the last NavLab project. It contained many tracking algorithms, one of which was pure pursuit. We had a few bugs in our system as a whole and couldn't discount the tracker as a possible culprit, so it fell to me to rewrite a tracker, with pure pursuit as the algorithm of choice. In performing this service I discovered that the code that we had been running had been executing with two separately defined lookahead distances.(You should read the next chapter for the definition and discussion of this parameter.) Parts of the code were running with a lookahead of 18 meters, and other parts were running with a lookahead of 4.5 meters. The reason that I bring this point up is that it amazed our group that the tracker had performed as well as it did given this fairly major error. We gained some additional respect for an algorithm that was robust enough to work when "purposely" maimed.

1.2. Description

What is pure pursuit?
Pure pursuit is a tracking algorithm that works by calculating the curvature that will move a vehicle from its current position to some goal position. The whole point of the algorithm is to choose a goal position that is some distance ahead of the vehicle on the path. The name pure pursuit comes from the analogy that we use to describe the method. We tend to think of the vehicle as chasing a point on the path some distance ahead of it - it is pursuing that moving point. That analogy is often used to compare this method to the way humans drive. We tend to look some distance in front of the car and head toward that spot. This lookahead distance changes as we drive to reflect the twist of the road and vision occlusions.

2.0 Theoretical Derivation

The pure pursuit approach is a method of geometrically determining the curvature that will drive the vehicle to a chosen path point, termed the goal point. This goal point is a point on the path that is one *lookahead distance* from the current vehicle position. An arc that joins the current point and the goal point is constructed. The chord length of this arc is the lookahead distance, and acts as the third constraint in determining a unique arc that joins the two points. Consider the lookahead distance to be analogous to the distance to a spot in front of a car that a human driver might look toward to track the roadway.

Consider Figure 1. The vehicle is pictured, with the axes of the vehicle's coordinate system drawn. The X axis passes through the rear axel of the vehicle. Shin[2] shows that propulsion and steering are geometrically decoupled if the vehicle's coordinate system is placed at the rear differential with the x-axis colinear to the rear axel.

The point (x,y), which is one lookahead distance l from the origin, is also shown. The point (x,y) is constrained to be on the path. The objective is to calculate the curvature of the arc that joins the origin to (x,y) and whose chord length is l.

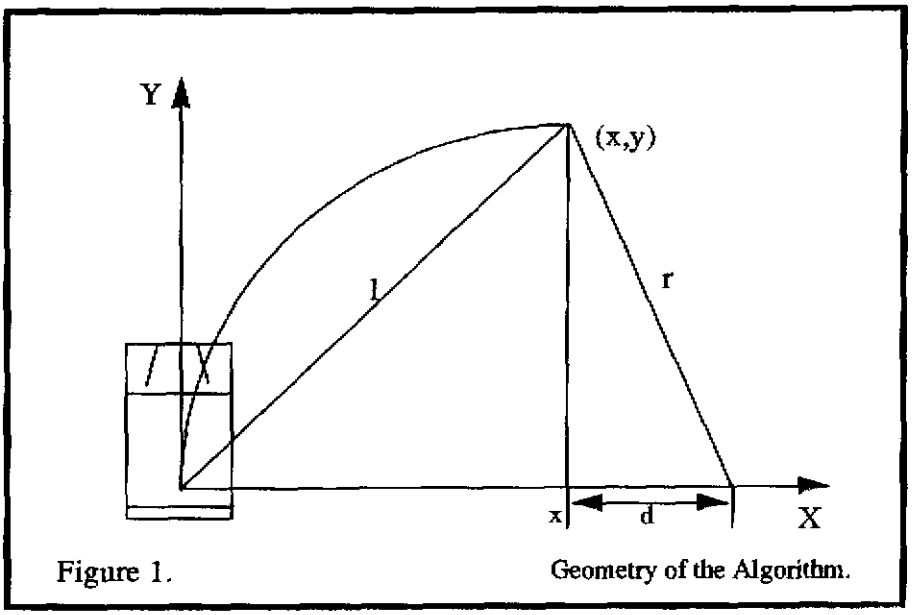

Figure 1. Geometry of the Algorithm.

The following two equations hold. The first is from the geometry of the smaller right triangle in Figure 1. The second from the summing of line segments on the x axis.

$$x^2 + y^2 = l^2 \qquad (2.1)$$

$$x + d = r \qquad (2.2)$$

- Equation (2.1) describes the circle of radius l about the origin. This is the locus of possible goal points for the vehicle.

- Equation (2.2) describes the relationship between the radius of the arc that joins the origin and the goal point, and the x offset of the goal point from the vehicle. This equation simply states that the radius of the arc and the x offset are independent and differ by d.

The next series of equations relate the curvature of the arc to the lookahead distance. The algebra is straightforward and requires no further explanation.

$$d = r - x$$

$$(r-x)^2 + y^2 = r^2$$

$$r^2 - 2rx + x^2 + y^2 = r^2$$

$$2rx = l^2$$

$$r = \frac{l^2}{2x}$$

$$\gamma = \frac{2x}{l^2}$$

The curvature has been related to the x offset of the goal point from the origin by the inverse square of the lookahead distance l. There is some similarity in form to a proportional controller where the gain is 2 times the inverse square of l. However the "error" in this form is the x offset of a point ahead of the vehicle.

3.0 Implementation

The method itself is fairly straightforward, as is the implementation. The only real implementation problems lie in deciding how to deal with the path information (communication, graphics, updating the path with new information from the planner), and even that isn't too bad.

3.1. Path Representation

A path is represented as a set of discrete points. (It has to be to be stored in memory.) Typically a path point is of some PATH_TYPE that is a struct containing the following information:

- x location in global coordinates.
- y location in global coordinates.
- heading in global coordinates.
- curvature of the path at this point.
- distance (along a straight line) of this point from the beginning of the path.

3.2. Communication and Path Management

The tracker usually runs on one machine while the planner runs on another. The idea is that during a navigation cycle the planner finds a path segment through the newly perceived terrain.

Meanwhile the tracker is driving the vehicle along the old path. When the planner completes its part of the cycle it must send the new path to the tracker. This new path probably partially overlaps the old path, so the planner must also tell the tracker what part of the old path can be overwritten with new information. Specific data management and communications techniques are beyond the scope of this technical report, the intent here is only to note that they are necessary and non-trivial portions of a real tracker implementation.

We must also note in this section that some provision must be made for interfacing the tracker with the sensor modules on the vehicle. In our implementation the tracker has an interface to a central vehicle controller which can inform the tracker of the current vehicle pose. The tracker can send its driving requests to this central controller for vehicle execution.

3.3. Pursuit Algorithm

The implementation of the pure pursuit algorithm itself is fairly straightforward. The pure pursuit algorithm can be outlined as follows:

- Determine the current location of the vehicle.
- Find the path point closest to the vehicle.
- Find the goal point.
- Transform the goal point to vehicle coordinates.
- Calculate the curvature and request the vehicle to set the steering to that curvature.
- Update the vehicle's position.

1) *Determine the current location of the vehicle.* Both the HMMWV and NavLab have a central vehicle controller that provides functions which report the vehicle's current position as (x,y,heading). The position is reported with respect to the vehicle's position at initialization time. This original position is the global reference frame for the run.

2) *Find the path point closest to the vehicle.* In the geometric derivation it was stated that the goal point would be within one lookahead distance of the vehicle. It is possible that there are multiple points one lookahead distance from the vehicle's current location. The vehicle should steer toward the closest point one lookahead distance from its current location. Therefore, the path point closest to the vehicle will first be found, and the search for a point 1 lookahead distance away from the vehicle will start at this point and commence up the path.

3) *Find the goal point.* The goal point is found by moving up the path and calculating the distance between that path point and the vehicle's current location. Path point locations are recorded in the global frame; this calculation is done in global coordinates.

4) *Transform the goal point to vehicle coordinates.* Once the goal point has been found, it must be transformed to the vehicle's local coordinates. The geometric derivation for the curvature was done in vehicle coordinates and curvature commands to the vehicle make sense in vehicle coordinates.

5) *Calculate the curvature*. Using the curvature equation derived in the last section, calculate the desired vehicle curvature. The curvature is transformed into steering wheel angle by the vehicle's on board controller.

6) *Update the vehicle's position*.During simulation, it is necessary to determine what effects the command has upon the vehicle's position and heading.

4.0 Properties of the Algorithm.

4.1. Effects of Changing the Lookahead Distance

There is one parameter in the pure pursuit algorithm, the lookahead distance. The effects of changing the lookahead distance must be considered within the context of one of two problems:

1) Regaining a path; i.e. the vehicle is a "large" distance from the path and must attain the path.

2) Maintaining the path; i.e. the vehicle is on the path and wants to remain on the path.

The effects of changing the parameter in the first problem are easy to imagine using the analogy to human driving. Longer lookahead distances tend to converge to the path more gradually and with less oscillation. The response of the pure pursuit tracker looks similar to the step response of a second order dynamic system (Figure 2.), and the value of l tends to act as a damping factor.

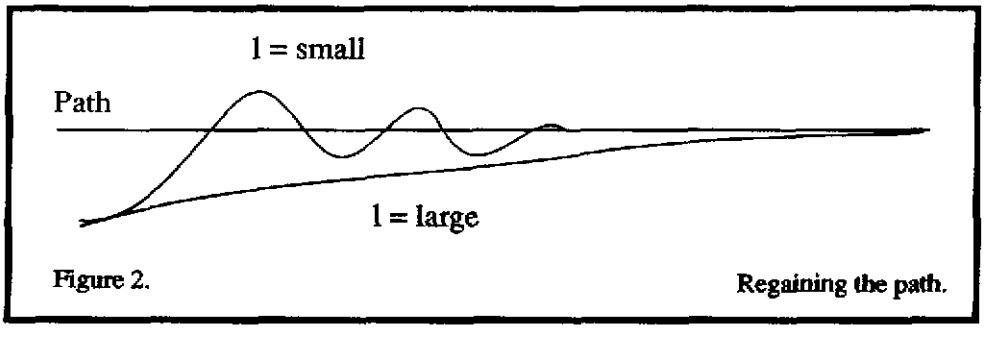

Figure 2. Regaining the path.

In the second problem, the longer the lookahead distance, the less "curvy" of a path that can be followed. The algorithm is calculating a curvature so that the vehicle can drive an arc. If the path between the vehicle and the goal point is sufficiently "curvy" then there is no single arc that joins the two points; any driven arc will induce error.

4.2. Non Unique Lookahead for a Given Path Curvature.

The path tracking problem that we most often have to address is that of staying on a path, rather than getting onto a path. We felt that it would be very useful to find a closed form relationship between the curvature of the path and an optimal lookahead distance. We sought to find a solution for the optimal lookahead for a circle of arbitrary curvature. Once this relationship was found, then the lookahead could be changed as the curvature of the path changes.

Finding a solution to this problem implies that a one to one relationship between the lookahead

distance and the curvature exists. The circle in Figure 3 is a path of constant curvature and therefore should have one lookahead distance that can be used for path following.

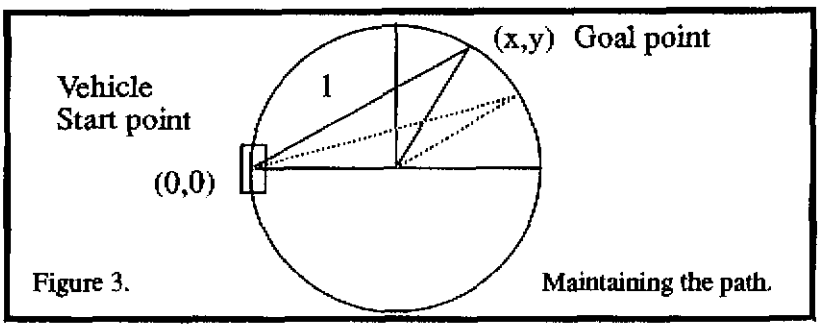

Figure 3. Maintaining the path.

An isosceles triangle within the circle shows the lookahead distance as the base of the triangle extending from the starting point to the goal point, the sides of the triangle are simply the circle radius. A lookahead distance that can form this isosceles triangle will satisfy the conditions on curvature stated earlier and therefore define the curvature of this circle. But it can be easily be seen in the diagram that many lookahead distances will satisfy the curvature conditions. Given a lookahead distance we can define a curvature, but given a curvature the lookahead distance is indeterminate. In fact, lookahead distances ranging between 0 and 2r are all equally admissable.

4.3. Comments.

During the past year of use, we have learned some of the limitations of this path tracking method. The two major limitations are related to the effects of dynamics. The method does not model the capability of the vehicle or of its actuators, and so assumes perfect response to requested curvatures. This causes two problems:

1) A sharp change in curvature can be requested at a high speed, causing the vehicle's rear end to skid.

2) The vehicle will not close on the path as quickly as desired because of the first order lag in steering.

5.0 References

[1] Amidi, O. "Integrated Mobile Robot Control.", Masters Thesis, Dept. Of Electrical and Computer Engineering, CMU, May, 1990.

[2] Shin, D.H "High Performance Tracking of Explicit Paths by Roadworthy Mobile Robots.", Doctoral Thesis, Dept. of Civil Engineering, CMU, May, 1990.

[3] Wallace, R. et al. "First Results in Robot Road-Following", report within CMU-RI-TR-86-4.