**Simulations of Curve Tracking using Curvature Based Control Laws**

Graduate Student: Robert Sizemore

Undergraduate Students: Marquet Barnes, Trevor Gilmore, Megan Harper, Thomas Le

Louisiana State University

Math 4020

December 3, 2015

**Introduction**

The purpose of this project was to create a feedback system using steering control law that enables an autonomous vehicle to track a simple, closed curve using information about the curve at the closest point. The prototype problem is to consider how a vehicle subject only to curvature can follow a boundary with collision avoidance. The course provided an opportunity for research into a concept of “Boundary Following using Gyroscopic Control” presented in a paper written by F. Zhang, E.W. Justin, and P.S. Krishnaprosed through the Institute for Systems Research at the University of Maryland College Park. The project required group members to learn and understand the published paper, the features of MATLAB, and their role in furthering the information provided on how a vehicle could track a curve by reaching critical points and staying a steady distance from the object on the curve. Some topics that were covered during this project were differential equations, curvature, class functions, and graphical user interfaces (GUIs). The control law we studied is applicable for the special case of smooth curves with constant curvature (e.g. circles and lines). Such control laws as the one we studied in this project are crucial for the design of self-driving cars and autonomous mobile robots.

The paper “Boundary Following using Gyroscopic Control”, the researchers speak to having the closest point and the vehicle being treating as a pair of interacting particles. The program reported that the vehicle would only experience shifting, also referred to as steering, when the user would apply curvature values, which is subject to our control. The curvature is chosen so that the rho and phi dynamics converge to zero. Convergence implies the curve tracking idea where rho is the distance to the closest point on the curve, phi is the angle between the tangent at the closest point, and the direction the vehicle is facing. As the project was presented, our group played the roles of the engineers, assuming the control works to achieve curve tracking and leaving the justification to be found in the original paper by F. Zhang and his fellow researchers. We simulated a curve-tracking situation and verified empirically that our control works.

**First Thought**

One of the first attempts the group made to tackle the project of having the vehicle track the curve was to create a function to move the car. Using curvature, the function created was called curvatureMove(), and it would allow the car to move clockwise for a fixed amount of time. It took into account negative curvature by moving counterclockwise and zero curvature by moving in a line. Each iteration would return a new car with an updated position and velocity. The problem that was presented itself was that since the function was dependent on curvature, it could lead to getting a very tiny circle since the time to move was fixed so the car could travel in small circles multiple times.

The next piece to the project was doing calculations using parameterization that would lead us to the curvature values we would be using. Through the given parameterization, one can find the tangent and unit tangent, which leads you in three directions. The first value that can be derived is the length of the tangent and from the unit tangent; one can find the unit normal and the derivative of the unit tangent. The derivate of the unit tangent has a length, which is used in the final calculation of curvature, which is the length of the derivate of the unit tangent divided by the length of the tangent.

For this first part of the project, we used the calculated values for steering control u2 which is showed in Figure 1 on the next page.

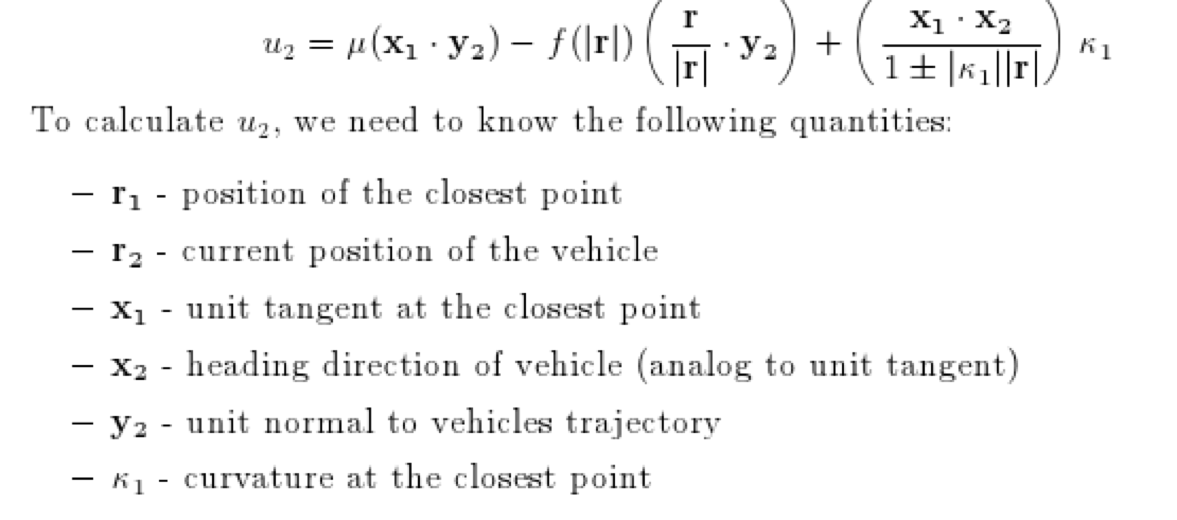


Figure 1: Steering Control Law 1

To give a little background to the function featured above, one would use arc-length parameterization to show the boundary curve, which is described by,

**r’1 = x1**

**x’1 = y1k1**

**y’1 = -x1**κ**1,**

where κ**1** is the plane curvature function for the boundary curve. The boundary following model considers the vehicle to be moving at unit speed. The next step figuring out how to store all the information we were going to put in and all the data coming out. The team created a convenient “class object” that could store all potential information such as vehicle location, vehicle direction vector, closest point on the curve, and curvature at the closest point. This class object could be easily modified and all the current information could be retrieved. The last aspect we discussed before the midterm presentation was using the MATLAB Guide feature to create a graphical user interface as the medium to show the vehicle tracking the boundary curve and be able to manipulate the data input all in one centralized location.

**The Second Thought**

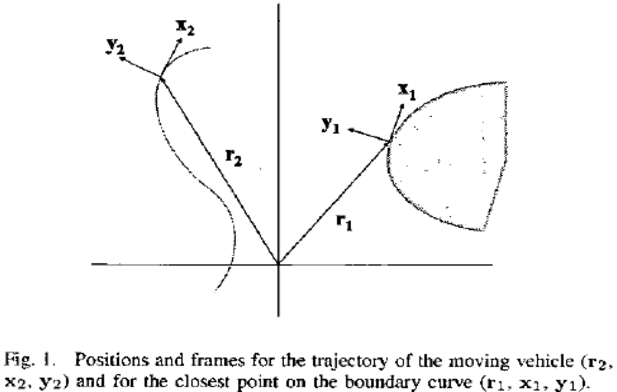
Following the midterm presentation, the design team decided to take another look at how we were approaching the project and if the information we had gathered was going to adequately track the boundary curve. So by furthering investigating the original paper, the Frenet- Serret frames, seen in figure 3, allowed for us to derive the velocity v of the closest point as a function of the distance from the vehicle to the closest point, ρ. Also we were able to derive the curvature at the closest point,κ, and the angle, φ, between the tangent to the curve at the closest point and the heading direction of the vehicle.



Figure 2: Velocity

Figure 3: Frenet- Serret Frames

Mathematically, we say that our vehicle tracks the curve if the relative distance ρ between the vehicle and the closest point converges to a constant ρ0, some defined “safe distance” from the boundary. Also if the angle φ between the heading direction of the vehicle x1, and the tangent to the curve at the closest point x2 converges to 0. By redefining the equations, we were able to find the corresponding dynamical system governing the relative motions of the vehicle and the closest point on the boundary curve given by:

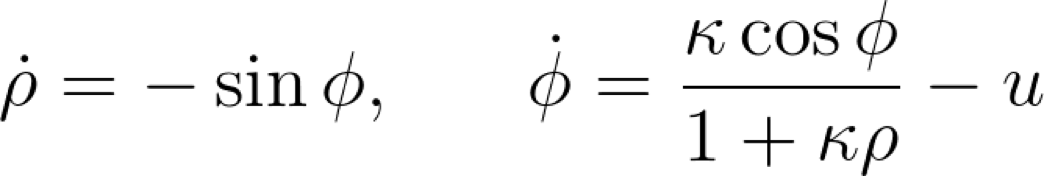


Figure 4: Dynamical System of Relative Motions

The steering control u should be chosen such that for solutions of the above systems, ρ converges to ρ0, and φ converges to φ0. The authors proposed another format of the steering control law u, which we implemented and can be seen below as figure 5.

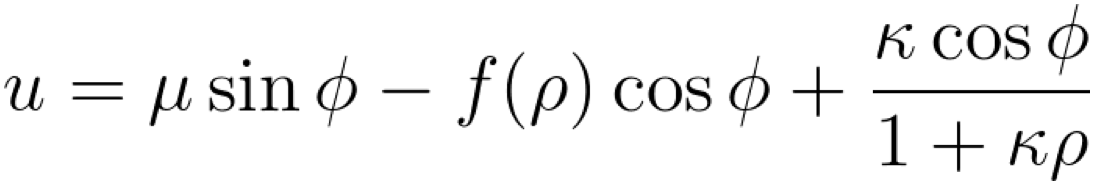


Figure 5: Steering Control Law

In this instance, ρ and φ are easily computable from position, tangent, and normal vectors for the vehicle and closest point. Similarity, the curvature of a parameterized function is easily computable.

After getting a clearer picture of the mathematical description of the paper, our group was able to better create a tracker graphical user interface (GUI), which can be seen in figure 6.



Figure 6: Tracker Graphical User Interface

Through the MATLAB’s GUIDE tool, the boundary curve was hardcoded into the GUI and the curve was specified by parametric equations x (t) and y (t), which in this demonstration, we considered the unit circle. The vehicle’s trajectory is plotted by iterating along arcs of circles with curvature given by the control law. The tracker object stores the vehicle’s position and velocity between iterations. The user is able to specify the vehicle’s initial position and direction, the total number of iterations, and the desired separation. The boundary curve, the vehicle’s trajectory, and the equilibrium curve are plotted in a window.

**Conclusion**

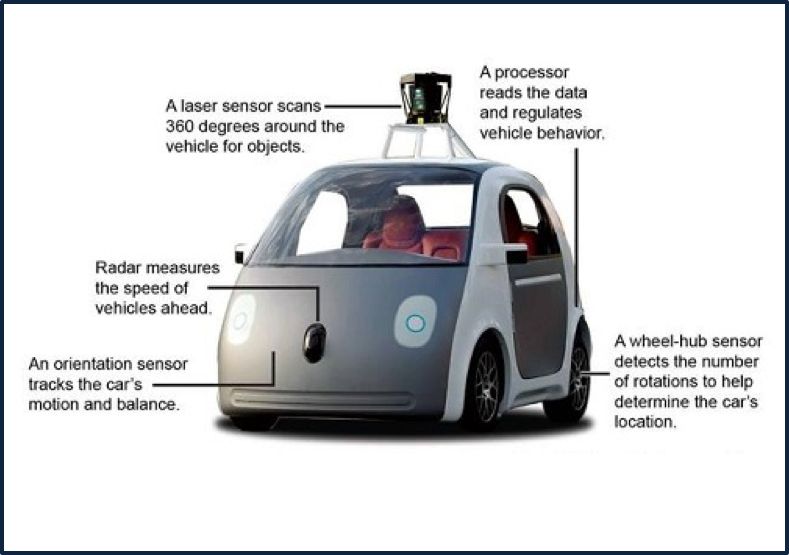
 In this research project, we were able to use the control law to achieve curve tracking in the case of constant curvature. The graphical user interface that was created featured a hardcoded curve, the unit circle, which leads to the issue that further testing would more than likely involve a recreation of the coding without the unit circle being present. The logical next step would be testing the control law for tracking more general parameterized curves with nonconstant curvature, or even user-drawn curves, where these is no guarantee of success. This step is often what is being used by current engineers, seen in figure 7, to create the newest technology.

Figure 7: An Autonomous Vehicle

The author of the paper that was used for this project, Fumin Zhang, also wrote a second paper that was looked over in the beginning of the semester. The paper called “Curve Tracking Control for Autonomous Vehicles with Rigidly Mounted Range Sensors” featured a furthering option of our project. A more sophisticated switching control law is discussed that can achieve curve tracking in the more general case of nonconstanst curvature.

**References**

“Boundary following using gyroscopic control” 2004 F. Chang, E.W. Justh, P.S. Krishnaprasad

“Curve Tracking Control for Autonomous Vehicles with Rigidly Mounted Range Sensors” 2009 Jonghoek Kim, Fumin Zhang, Magnus Egerstedt