Ackermann Steering Robot PD Controller

Akash Ravindra
Maryland Applied Graduate Engineering
University of Maryland
College Park, Maryland, USA.
aravind2@umd.edu

Manu Madhu Pillai
Maryland Applied Graduate Engineering
University of Maryland
College Park, Maryland, USA.
manump@umd.edu

***Abstract*—The Ackermann mechanism is a ubiquitous linkage used in automotive and robotics to aide in steering and control. They facilitate the control by allowing the wheels of the machine to pivot about the axis perpendicular to the motion, causing the wheels of the machine to point in a direction such that the resulting arc produces the desired motion. In this project, we develop a controller that controls the Ackermann mechanics such that a desired target heading and velocity can be achieved.**

***Index Terms*—Ackermann mechanism, Mobile robot, path planning, Forward kinematics, Inverse kinematics, C++, Agile Iterative process**

I. INTRODUCTION

The project aims to implement a controller for the newly developed ground robot developed by Acme Robotics which uses an Ackermann steering mechanism. Ackermann steering mechanism is superior to other steering mechanisms in terms of power usage, reduced tire slippage and better rough terrain performance. These advantages make the robot perform more consistently in various terrains and increases the longevity of the robot. The Controller takes inputs as desired heading and velocity and calculates steering angles and velocities of the front wheels to achieve the desired result. The wheelbase and track width are taken from the robot specifications. It is assumed that the motors used can output the exact angular velocities as provided by the controller.

II. TECHNICAL DETAILS

Test Driven Development will be used as the development process using pair programming. C++ 14 and above will be used as the programming language. The project will follow Google C++ Style Sheets for coding style. Cpplint will be used to verify coding style compliance and cppcheck will be used for static code analysis. Valgrind will be used to check for undefined behavior and memory leaks.

The project will undergo a series of unit test which will be implemented using the google test tool generating code stubs through Travis-CI. We will maintain a code coverage of 90% plus through the above mentioned tests and will verified on Coveralls for the same. We expect to use math, string and vector libraries which are a part of the standard C++ library and are open source.

III. METHODOLOGY

Consider figure 1 showing a skeleton of the robot. The center of the robot is assumed to be in the geometric center, between the two axles. The distance between the wheels, refered to here as the track width represented T_w and the distance between the axles, referred to here as the wheel base represented by L. θ_i and θ_o represent the relative steering angle of the inner and outer wheel respectively with the ICC (instantaneous center of curvature) with a radius of R.

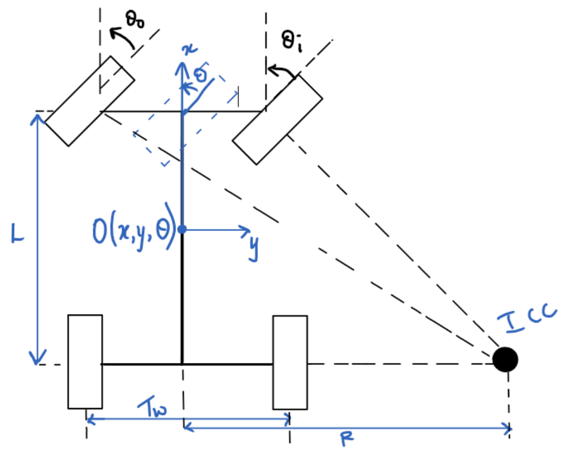

Fig. 1. Diagram defining the axis and terms used to describe the robot

The Ackerman steering equation can then be derived quite easily by considering the three triangles formed by the vertical side l and the side r plus or minus $\frac{w}{2}$.

$$\tan\theta = \frac{L}{R} \tag{1}$$

$$\tan\theta_i = \frac{L}{R - w/2} \tag{2}$$

$$\tan\theta_o = \frac{L}{R + w/2} \tag{3}$$

Evaluating the above equation we can express both θ_i and θ_o as a function of θ.

$$\theta_i = \arctan\frac{2l\sin\theta}{2l\cos\theta - w\sin\theta} \tag{4}$$

$$\theta_o = \arctan\frac{2l\sin\theta}{2l\cos\theta + w\sin\theta} \tag{5}$$

A. Forward Kinematics

Now that we have a relation that related the various parameters of the robot to the its central position and orientation, the forward kinematics can be used to predict the future state given the current configuration. Considering the state of the vehicle is represent as a shown in the figure 2 where θ is the angle of the wheel and ϕ is the heading and (x, y) is the cartesian position of the center of the robot. Given the speed of the wheel as s we can derive the equation as,

$$\dot{x} = s\cos\phi \tag{6}$$

$$\dot{y} = s\sin\phi \tag{7}$$

$$\dot{\phi} = \frac{s}{l}\tan\theta \tag{8}$$

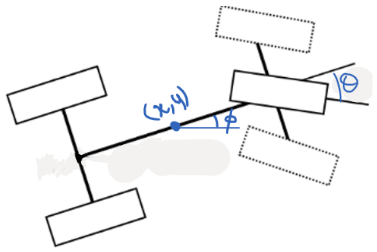

Fig. 2. Forward kinematic sketch

Using the above equation as the equation of the plant, we can construct a closed loop controller as shown in figure 3. Where K_p and K_d are the proportional and derivative gains respectively which need to be tuned to produce good response when a given reference value is provided. The Odometer is a sensor that will be used to capture the real time output i.e position and velocity of the plant. This is then used to close the loop and provide feedback to the controller.

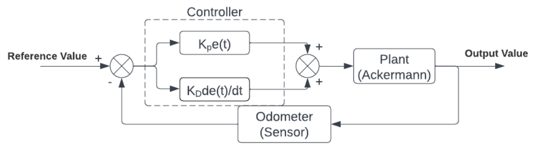

Fig. 3. Closed loop setup

IV. Risk and Unknowns

One of the major unknowns is the frequency of the inputs. Hence, it introduces a latency issue. A solution to this issue is to queue the inputs such that the system reads the input from the queue whenever it is ready. Another unknown is the precision and time delay of the odometer sensor, which is an essential input to the overall closed loop controller.

We assume that the robot frame of reference to be at the physical centre of the robot, that the front wheels have the freedom to turn about an angle of 45 degrees, the latency in passing to the updated values of velocities and steering to the required actuators after calculation to be minimal. The wheelbase and track width were obtained from the specification sheet of the Acme robot. Any deviation from the assumptions made would risk facing issues.

The calculated velocities for the two drive wheels and calculated the steering angle to reach the destination will be the outputs of the system. Positive steering angle indicates that the steering is turning left and negative steering angle indicates that the steering is turning right.

All units considered by the system will be in compliance with the International System of Units.

V. CONCLUSION

Controller for an Ackermann kinematic model with a maximum steering angle constraint (e.g. $<$ 45 degrees) which takes inputs robot target heading and velocity and gives output steering and the two drive wheel velocities, demonstrating convergence to the set points using a simulator.

VI. Deliverables

The deliverables of the project are listed below:

- Code Repository on Github with a detailed ReadMe and commit history
- Short Video explaination
- UML Diagram
- Detailed documentation
- Unit test results integrated with Travis-CI and Coveralls for code coverage.
- Results of cppcheck performing static code analysis and cpplint checking Google C++ Sytle sheets compiliance.
- Result of Valgrind performing undefined behavior check and memory leak check.

References

[1] Jonathan Vogel."Tech Explained: Ackermann Steering Geometry" https://www.racecar-engineering.com/articles/tech-explained-ackermann-steering-geometry/

[2] Eisele, Robert. "Ackerman Steering." Open source XARG http://www.xarg.org/book/kinematics/ackerman-steering/

[3] "Pair Programming." Wikipedia, http://en.wikipedia.org/wiki/Pair_programming/