Controller for an Ackermann Kinematic Model

Charu Sharma
Robotics Engineering, ENPM
University of Maryland
College Park
charu107@umd.edu

Vivek Sood
Robotics Engineering, ENPM
University of Maryland
College Park
vsood@umd.edu

Task— **Acme Robotics is launching a brand-new robotics-based product next year, but their small development team is falling behind. They've reached out to help them get back on track.**

The task is to design a new module in one of their robotics-based components using high-quality software engineering practices so their team can take your deliverable, finish development and integrate it into the product.

- **Controller for an Ackermann kinematic model with a maximum steering angle constraint having the inputs as robot target heading and velocity whereas the outputs are the steering and the two drive wheel velocities, demonstrating convergence to the set points**

I. INTRODUCTION

A simple approximation to perfect Ackermann steering geometry may be generated by moving the steering pivot points inward so as to lie on a line drawn between the steering kingpins and the center of the rear axle. The steering pivot points are joined by a rigid bar called the tie rod, which can also be part of the steering mechanism, in the form of a rack and pinion for instance. With perfect Ackermann, at any angle of steering, the center point of all of the circles traced by all wheels will lie at a common point. Note that this may be difficult to arrange in practice with simple linkages, and designers are advised to draw or analyze their steering systems over the full range of steering angles.

The model (Image 1) represents a vehicle with two axles separated by the distance, WheelBase T. The state of the vehicle is defined as a four-element vector, [x y theta psi], with a global xy-position, specified in meters. The xy-position is located at the middle of the rear axle. The vehicle heading, theta, and steering angle, psi is specified in degrees. The vehicle heading is defined at the center of the rear axle. Angles will be computed in degrees.

To compute the time derivative states for the model, we use the derivative function with input steering commands and the current robot state.

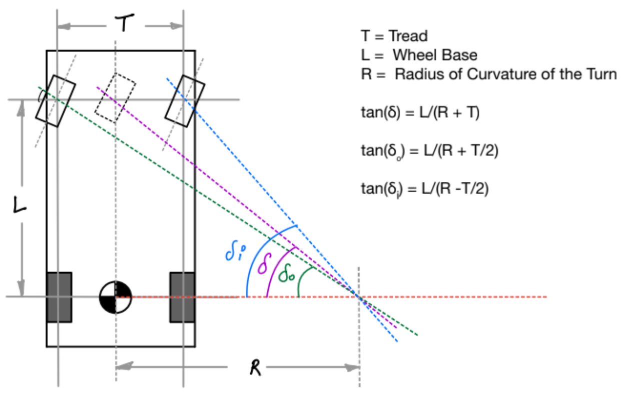

T
L
R
δ_i
δ
δ_o
T = Tread
L = Wheel Base
R = Radius of Curvature of the Turn
$\tan(\delta) = L/(R + T)$
$\tan(\delta_o) = L/(R + T/2)$
$\tan(\delta_i) = L/(R - T/2)$

Image 1: Ackermann steering model

II. PROBLEM DEFINITION

The main objective of our project is to create a Controller that uses the Ackermann Kinematic model with maximum steering angle of 45 degrees.

We plan to have a working model which works in aligning the robot to the target heading and with a target velocity simultaneously. For this we would have an Ackermann controller and a PID controller for the tasks of getting to the target heading and target velocity simultaneously.

III. METHODOLOGY

We need to build a controller to navigate a 4 wheeled robot with an Ackermann steering. The controller consists of a PID algorithm which ensures that the velocity converges to the set point, and a steering algorithm that helps the robot turn.

We have the PID algorithm that calculates the error and applies corrections through proportional, integral and derivative gains. The steering algorithm is developed to turn the robot. We plan to use these in our project with slight differences in operations.

For our tentative plan, we first calculate the length of the arc between the current robot heading and the target heading. Next is to calculate the inner and out Ackermann's angles. We then need the time dt in which the movements can be made, and we move towards the target heading for this time. Here we have the PID controller to manage the Ackermann steering's orientation. Next, simultaneously we have the PID algorithm in a closed loop to maintain the gradual velocity over time till we reach the target velocity.

As for the backup our agenda is to make sure the robot is aligned to the target heading and then made to reach the target velocity. We first calculate the length of the arc followed by inner and outer Ackermann's angles. Then we need the time for which the movement will take place. Using this time definition will have a closed loop controller to manage the steering. We continue till we reach the target heading. After this we have the PID controllers to achieve the target velocity.

The components for the controller include the robot classifier/component (*TwoWDRobot*), visualization module (*Visualization*), PID controller (*PID*), and steering module (*Ackermann*). The visualization module has an aggregatory relation with the robot classifier hence is not essential for the existence of the *TwoWDRobot* class. PID and the steering module have a composite relationship with the robot classifier hence they are essential for the existence of the *TwoWDRobot* class (Image 2)

[Robot_Contoller/ClassDependency.pdf at main · viveksood97/Robot_Contoller · GitHub]

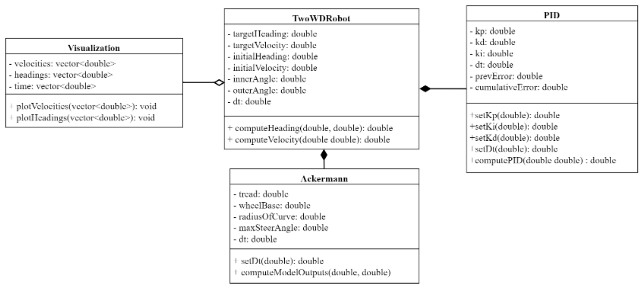


Image 2: UML Diagram

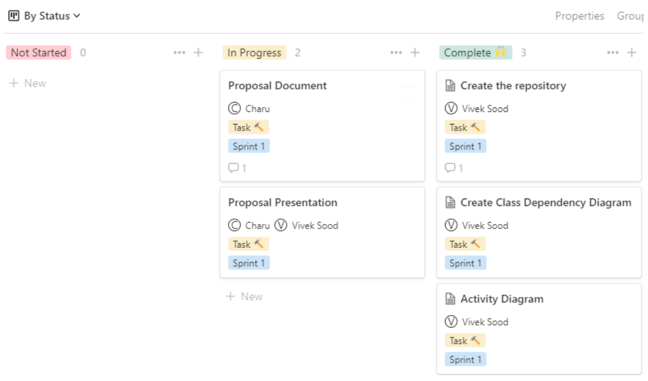
We also plan to show the convergence of set points in real time on GNUPlot which is a plotting utility that can be used in cpp.

IV. PROCESS PLANNING

We are employing Agile Software Development Cycle for the project. For this we are making use of the tool called Notion (Image 3).

[Controller for an Ackermann Kinematic Model | By Status (notion.so)]

For the current proposal we have 1 Sprint and 5 Tasks. This is under single Epic. We plan to have multiple Epics for subsequent phases of the project.


By Status
Properties Grou

Not Started 0
+ New

In Progress 2

Proposal Document
Charu
Task
Sprint 1
1

Proposal Presentation
Charu Vivek Sood
Task
Sprint 1

+ New

Complete 3

Create the repository
Vivek Sood
Task
Sprint 1
1

Create Class Dependency Diagram
Vivek Sood
Task
Sprint 1

Activity Diagram
Vivek Sood
Task
Sprint 1

Image 3: Roadmap page

V. CHALLENGES AND MITIGATION PLAN

There are few challenges that we are anticipating while doing this project:

- We plan to employ the PID controller for the velocity method. But while working with the steering method we intend to have the proportional, integral and derivative gain values of the PID that work in tandem to converge at the targeted setpoints. So, if we are unable to get that we plan to implement a basic closed loop controller for target heading and PID for target velocity

- we plan to have a real time plotting on GNUPlot, which if failed we intend to just plot the graph at the end of execution

References

[1] https://en.wikipedia.org/wiki/Ackermann_steering_geometry

[2] https://www.mathworks.com/help/robotics/ref/ackermannkinematics.html

[3] Tech Explained: Ackermann Steering Geometry - Racecar Engineering (racecar-engineering.com)

[4] (1) Roadmap | Sprints (notion.s