Ackermann Steering Controller - Proposal

Pooja Kabra
University of Maryland
College Park, Maryland
pkabra@umd.edu

Markose Jacob
University of Maryland
College Park, Maryland
markj11@umd.edu

I. Introduction

It is absolutely crucial to ensure that there is no slippage in the wheels as the robot makes turns. Slippage can cause the robot to drift out of control and produce undesired results. One way to ensure there is no slippage is by using an Ackermann Steering method along with a PID controller. Ackermann method calculates the heading and velocity needed at the inner and outer wheels of the robot so as to ensure there is no slippage.

A. Project Overview

In this project we are going to use the Ackermann steering method to calculate the velocity and heading needed for each wheel to avoid slippage and then use a PID to change the velocity and heading of each wheel gradually to ensure that the wheels don't slip and cause the robot to drift out of control. The basic idea of an Ackermann controller is that the inner wheels should rotate slower than the outer wheels and the angle of steer should be larger compared to the outer wheel. This will prevent the wheels from slipping sideways as the robot makes a turn. The code we deliver can be integrated with the motor control of the robot. The inputs needed will be the target heading and target velocity for the robot and the code calculates the heading and velocity for each wheel which will then use a PID controller to send inputs to the hardware.

B. Project Deliverables

At the end of this project we will provide you with a code written in C++ which will take in the target heading and target velocity of the robot and the output will be signals from a PID controller to gradually change the velocity and heading for the inner and outer wheels so as to avoid slippage. The code can be directly integrated with motor control of the robot.

C. Evolution of Software Project Management Plan

Our current plan to provide you with a solution to avoid slippage in a four wheeled robot with rear wheel drive and front wheel steering.

D. Reference Materials

[1] Ackermann's formula

[2] Luca Bascetta, Davide A. Cucci, Matteo Matteucci **Kinematic trajectory tracking controller for an all-terrain Ackermann steering vehicle.** 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016: Leipzig,Germany

[3] Introduction: PID Controller Design

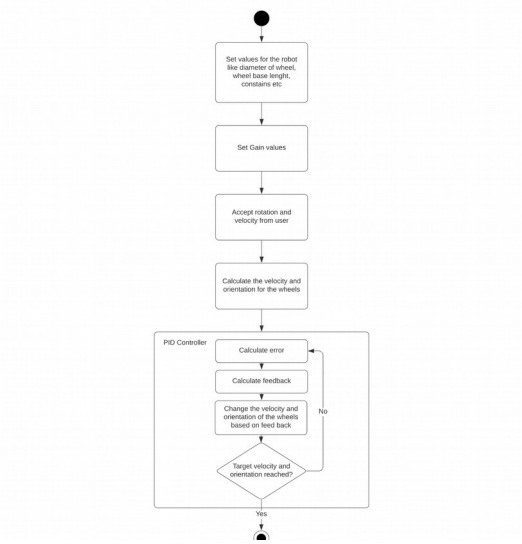
Definitions and Acronyms

PID - Proportional-integral-derivative

II. Project Organization

A. Process Model

Once the set velocity and rotation is input, the PID controller compares it to the actual state of the robot and calculates the error. The error is processed to compute feedback to drive the output closer to the desired value. The updated output is again checked against the desired and processed further. This is repeatedly performed till the target state of the robot is attained.



B. Organization Structure

The project workflow will be conducted using Agile methodology. This will ensure that the product meets all requirements and delivered on time after doing proper testing. This also allows us to incorporate any changes in requirement if it were to come up.

C. Organizational Boundaries and Interfaces

The work products will include an interface that would accept system variables (wheel radius, track length, wheelbase) in addition to controller gains and target rotation and velocity, and display graph visualization of the controller realizing desired values. The client will be able to view 2D simulation of the car making a turn around a constant radius of curvature.

D. Project Responsibilities

Acme Robotics is the client for the project. Project Management, Execution and Review will be the onus of our

team.

III. Managerial Process

A. Management Objectives and Priorities

The management would oversee that the deliverables are released on the agreed timelines and ensure that the software development process follows high-quality software engineering practices so the client can take the deliverable and easily integrate it into their product.

B. Assumptions, Dependencies and Constraints

We assume a rear wheel drive and front wheel steering for the four-wheeled robot. No foreseeable dependencies/constraints at the moment.

C. Risk Management

There have been known challenges in using OpenCV with Cpp. If 2D visualization using OpenCV fails, we will still provide graph based visualizations of state variables to demonstrate performance.

D. Monitoring and Controlling Mechanisms

We will make use of Git for version control. Continuous integration will be handled using Travis CI and code coverage analysis using Coveralls. These tools will be synced with our GitHub Repository and the status will be continuously available to monitor. We will resort to Test-Driven Development as a testing technique and GoogleTest will be our unit testing framework.

E. Staffing Plan

A two resource team is necessary and sufficient. We will switch among roles to drive the project to completion.

IV. Technical Process

A. Methods, Tools and Techniques

This project implementation will follow Agile Workflow. We will be switching the driver and navigator role, For the start the driver will be Markose Jacob and Pooja Kabra will be Navigator. The programming language we plan on using is C++ as it is very fast and hence makes it more suitable for practical applications. The environment is going to be Linux based Ubuntu 18.04 and the build system will be “make”. We also plan to use tools like Travis CI for unit testing, Coveralls to check the coverage of our unit test, Cppcheck and Cpplint to ensure that our product is top quality. We will be following the Google style guide and generating documentation so that the code can easily be understood. Lastly, we will be providing you with a readme file with the instructions on how to build and run the code.

B. Software Documentation

Full Doxygen documentation will be provided for the code.

V. Work Packages

A. Resource requirements

The application to be developed does not require heavy computation. Two generic workstations with Intel Core i5 or low end i7, 4GB/8GB of RAM and 128GB SSD would be required.

B. Schedule

Below is a quick overview of timelines.

Task	Duration	Start Date	End Date
Initialize GitHub repository and update badges	1 hr	10/03/21	10/03/21
Create Project Backlog	2 hrs	10/04/21	10/04/21
UML Diagrams	2 hrs	10/04/21	10/04/21
Create Class declarations and Unit tests	2.5 hrs	10/06/21	10/06/21
Complete implementation, run Cppcheck and Cpplint	4.5 hrs	10/09/21	10/13/21
Check Valgrind and fix issues	2.5 hrs	10/17/21	10/17/21
Implement Visualization	4 hrs	10/20/21	10/21/21

The product with its complete functionality will be delivered on October 25, 2021.