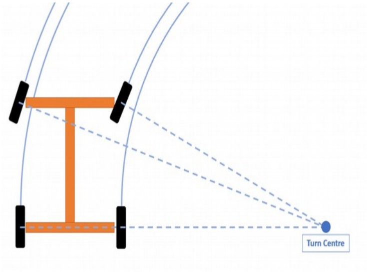
Markose Jacob

Ackermann Steering Controller

Pooja Kabra

Overview:

We propose to implement an Ackermann Steering Controller for a 4-wheeled robot. Tyres tend to slip when a car tries to make a turn, if all four wheels were moving at the same velocity and orientation. The geometrical solution to this is for all wheels to have their axles arranged as radii of circles with a common centre point.

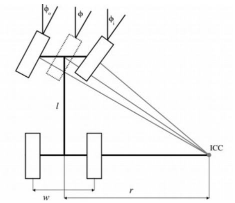


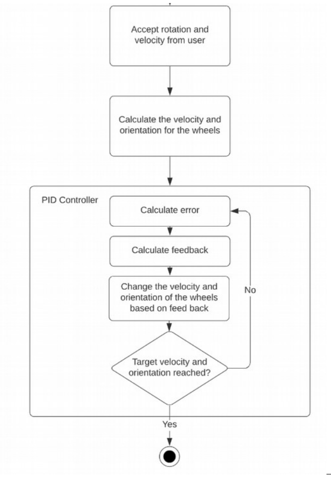
As the rear wheels are fixed, the the steering geometry must steer the inside front tyre at a larger angle than the outside front. Also the outer tyre would cover a larger path than the inner tyre in the same time and their velocities need to be adjusted accordingly.

Based on target heading and velocity, Ackermann controller will compute the ideal turning angle and wheel velocity for each front wheel. These values will be forwarded to low level motor controllers that drive the hardware motor output.

Methodology:

The flowchart describes the intended implementation of the controller.





Accept rotation and velocity from user

Calculate the velocity and orientation for the wheels

PID Controller

Calculate error

Calculate feedback

Change the velocity and orientation of the wheels based on feed back

Target velocity and orientation reached?

No

Yes

The programming language C++ 14 or above will be used for development of the software. Travis-CI will be used to verify successful builds of our GitHub repository, Coveralls will report the total coverage of the program. Visual Studio Code will be used as the Integrated Development Environment. Valgrind will be used to check for memory leaks and undefined behaviors. This project will use Test Driven Development and Google tests to ensure the software functions are designed as desired.

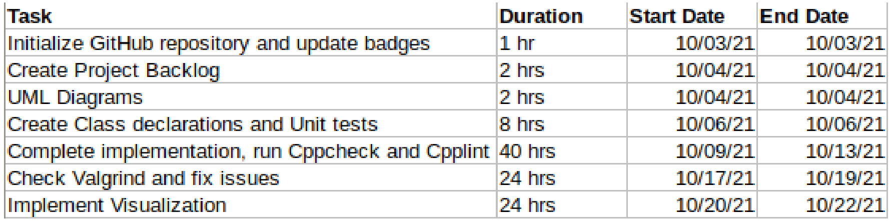
Goals:

Highly accurate orientation and wheel velocity to reach desired given values

Robot closely traces expected trajectory

Comprehensive Visualization of results

Estimated Timeline:



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	8 hrs	10/06/21	10/06/21
Complete implementation, run Cppcheck and Cpplint	40 hrs	10/09/21	10/13/21
Check Valgrind and fix issues	24 hrs	10/17/21	10/19/21
Implement Visualization	24 hrs	10/20/21	10/22/21