Ackermann kinematic Controller

Overview:

What you are doing – We are planning to implement Ackermann bicycle kinematics controller on a mobile robot/vehicle with 4 four wheels where steering is attached to the front wheels and rear wheels are assumed to be fixed. The controller will control the steering angle and wheel velocities to reach a desired location and orientation. The inputs given will be heading and velocity to the mobile robot and output will be in the form of steering angle and velocities of both wheels. In order to prevent slippage between the tires the maximum angle constraint will be given as 45 degrees.

The precise controller is required in order to solve the problem of over-steer and under-steer leading wrong trajectory planning of the robot/vehicle. Having a closed loop PD controller would be ideal to mitigate the shortcomings. An open source visualizer will be used to verify the functionality of the controller.

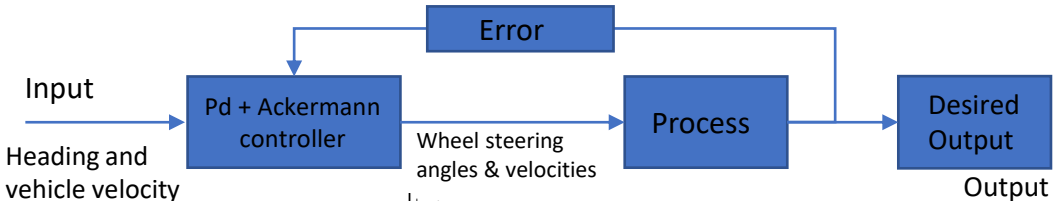
By using this control module software, Acme mobile robots can get better efficiency with their path planners and applications in real world scenario.

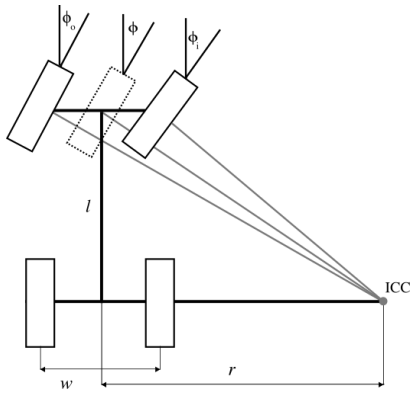
Methodology:

This project will use test driven pair programming development and google tests to ensure the software functions as designed.

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, and Valgrind will be used to check for memory leaks and undefined behaviors. OpenCV will be used to visualize the performance of the controller. Cpp-boilerplate will be used to set up the basic layout of the software.

Risks involved: Designing our own PD controller will increase the complexity of our project which leads to increase schedule and performance risk. There is also risk in working with OpenCV as neither partner has a lot of experience using it. This risk is mitigated by the large knowledge base associated with OpenCV.





Expected Timeline :

Task	Duration	Start Date	End Date
Initialize github repo and update badges	1d	10/03/20	10/03/20
Creating project back-log	2d	10/04/20	10/05/20
UML diagram and class declarations	2d	10/06/20	10/07/20
Populating the classes and applying algorithms for controller with regular commits with cpp-check and cpplint	5d	10/08/20	10/12/20
Writing google unit tests, testing and updating algorithms as needed with cpp-check and cpplint	4d	10/10/20	10/13/20
Check for memory leaks and debug using valgrind	2d	10/14/20	10/15/20
Implement visualization	4d	10/16/20	10/19/20

Expected Results:

- Accurate orientation and wheel velocity to reach desired given values with correct trajectory
- Simulation comprehensive enough for the user to visualize.