Project Proposal: Ackermann Steering Controller for Autonomous Vehicle Module

Overview

This proposal outlines the design and development of a control module for an Ackermann Steering Controller to be used in an autonomous vehicle system. The control module's primary responsibility is to compute steering angles with a maximum constraint of 45 degrees while controlling the drive wheel velocities. The core functionality will involve calculating the appropriate steering and velocity outputs based on the input target heading and velocity, ensuring convergence to the desired set points in an efficient and error-minimizing manner.

Assumptions

1. The target heading and velocity will be provided as input by the navigation module.
2. The module will primarily focus on steering control and wheel velocity, with velocity control for translational movement being handled separately.
3. The system will include a sensor suite (e.g., encoders, IMUs) to provide real-time feedback on the robot's position and orientation, which the control module will rely upon for error calculation.

Design and Development Processes

The development process will follow Agile methodologies, ensuring iterative progress with regular feedback loops. The development will include several key phases:

1. **Requirements Gathering**: This includes specifying interface requirements with the motor control and navigation modules.
2. **Design**: We will create detailed UML diagrams and architecture documentation to describe the module's structure, class relationships, and methods.
3. **Development**: Implementation will follow test-driven development (TDD) practices, ensuring high code quality and robustness.
4. **Testing**: Unit tests will validate all functional components of the system, and system-level tests will ensure that the module works effectively within the robot's control architecture. GitHub Actions will be used for continuous integration and deployment (CI/CD), running tests on each commit.
5. **Documentation**: Throughout the project, comprehensive documentation will be created to explain the system design, API, and any external dependencies.

Technologies to be Used

The controller will be implemented in C++, leveraging the following technologies:

- **ROS2 (Robot Operating System)** for communication between modules and subsystems.
- **Google Test (gtest)** for unit testing.
- **GitHub Actions** for CI/CD.

Component Functionality

The main functionalities of the Ackermann Steering Controller include:

- **Steering Control**: Adjusting the front wheel steering angles based on the target heading while adhering to the maximum steering angle constraint.
- **Velocity Control**: Calculating the drive velocities for the two wheels based on the robot's target velocity and heading.
- **Error Minimization**: Using the PID controller to minimize the error between the target and actual trajectories.

Risk and Mitigation

Technical Risks

1. **Integration Challenges**: The controller needs to interface seamlessly with other subsystems. Rigorous integration testing and clear API documentation will mitigate this risk.
2. **Physical Limitations**: While the system can compute steering and velocity commands, the actual performance of the vehicle will depend on its mechanical design. Simulations and testing in controlled environments will help validate the system before deployment.

Project Risks

1. **Team Coordination**: Pair programming will be used extensively to ensure knowledge transfer and minimize the impact of any team member's absence. Code reviews and documentation will further mitigate this risk.
2. **Time Constraints**: The project timeline is tight, but using Agile sprints will ensure progress is closely monitored, and tasks are reprioritized as needed.

Final Deliverables

1. **Ackermann Steering Controller Module**: Complete source code for the controller, following all coding standards and best practices.
2. **UML Diagrams**: Documenting the structure of the module and its interaction with other components.
3. **Unit and System Tests**: Comprehensive test suite ensuring the correctness and robustness of the module.
4. **Documentation**: Detailed API and integration documentation, ensuring smooth handover to the Acme Robotics team.

Team Members and Pair Programming

The project will be developed by a team consisting of Dhairya Shah and Harsh Senjaliya. Pair programming will be executed for all core functionality, with both developers working together to ensure shared responsibility and knowledge. Each pair programming session will be documented, with code commits reflecting the joint effort.