CHAPTER VI. IMPROVED CONTROLLER IMPLEMENTATION

To properly simulate the handling of a four-wheeled vehicle, the author performed an analysis of Ackermann steering geometry, and improved a packaged steering controller to more closely conform to Ackermann steering geometry.

The author first attempted to use the packaged steering controller. Because the ability to turn at a constant radius at speeds typical of vehicles participating in the 2004 and 2005 GCE is central to validating several of the conclusions reached throughout this research, to demonstrate the ability to limit the turning radius of the model to the radius of the representative challenge vehicle turning circle (37.7 ft or 11.49 m), the author generated a mesh of two circular walls with a height and width of 10 cm: the inner wall with a outer radius of 9.49 m and the outer wall with an inner radius of 13.49 m. A world file was generated to include this mesh and the representative challenge vehicle model located in the turning circle with its CG at a position offset by CG to rear axle x-dimension (1.265 m), and one-half the sum of the rear track width and section width (0.882 m).

When controlling the model with the `playerv` utility using the packaged controller, the author noted an unexpected “flattening” of the path of travel when the steering angle was at a maximum at full right extent. The steering angle was selected based on the representative challenge vehicle turning circle.

Problems encountered during development of the improved controller are documented by Appendix G.

VI.A. Independent steering wheel angle

The author analyzed the packaged steering controller and determined that setting the angle of both steering wheels to the same angle had the effect of "dragging" the rear of the model around at certain points on the path of travel, and proposed the observed behavior was due to the effect of friction caused by the angle of the outer steering wheel. Because the packaged steering controller set the angle of both steering wheels to the same angle, the outer steering wheel in any turn was exerting more force toward the center of the turning circle than it should have been exerting.

After a period of time during which the model would travel in a circular path as expected, the effect of friction would cause the front wheels to drag the front end of the model around using the inner rear wheel as a pivot. The motion of the model would return to normal for a while, then the effect of friction would cause the front wheels to drag the front end of the model around using the inner rear wheel again. This would continue as long as the model was traveling in a circle.

The author improved the steering controller to determine and set the angle of the steering wheels independently. This conforms to Ackermann steering geometry. The steering angles and angular velocities of the inside and outside wheels are calculated using the steering angle and angular velocity at model CG, track width, and wheelbase. As a result, when the steering angle is not equal to zero:

- The steering angle of the inside wheel (i.e., the wheel on the inside of the turn) is greater than the steering angle at model CG.
- The steering angle of the outside wheel is less than the steering angle at model

CG.

VI.B. Odometry

Because several of the simulation targets require the ability to determine the position of the model in relation to detected obstacles, the author implemented odometry in the improved controller in a manner similar to the implementation in the packaged position controller for a robot using differential drive.

The position of the model is determined based on the distance traveled by the rear wheels, which is calculated using the angular velocity and radius of the rear wheels. One weakness of this method is that the reported position of the model is independent of the actual position of the model when the wheels are not touching the ground because the controller continues to calculate the distance traveled by the rear wheels.

However, if this is a problem in simulation it may also have been a problem during the 2004 and 2005 GCE. Thirteen of 25 teams in 2004 and seven of 23 teams in 2005 referred to the use of encoders as navigation sensors on each wheel, rear wheels only, or the drive axle to provide instantaneous velocity.

In addition, the error in reported position of the model increases as the distance traveled from the initial position of the model increases due to the accumulation of errors in calculation over thousands of simulation cycles. Because most of the tests performed by the author were over relatively short distances, the author did not attempt to resolve this problem.

The author attempted to use the reported position to determine the exact location at which rollover occurred to be able to analyze the path of the model until the onset of

rollover. However, attempts to determine the onset of rollover were unsuccessful. See paragraph VII.E.

VI.C. Additional features of the improved controller

In addition to the parameters described above, the author also implemented several features to increase the usability and realism of the improved controller:

- Gas pedal

The author implemented a "gas pedal" by using the `cmdVelocity.pos.x` value returned by improved controller function `GetPositionCmd` to scale the maximum constant acceleration calculated by the controller. The maximum constant acceleration was determined by values for the final velocity and time to reach the final velocity. These values are read from the model XML file.

The `cmdVelocity.pos.x` value returned by function `GetPositionCmd` is between -0.1 and 0.5, depending on how far the user drags the cursor to the left or right, respectively. The minimum and maximum values were compiled into the `playerv` utility and may be modified by making changes to function `position2d_servo_vel` in file `pv_dev_position2d.c`. To minimize the number of changes required, the author decided not to revise the `playerv` utility to modify the minimum and maximum values but to use these values to scale acceleration via the improved controller.

To simulate a gas pedal, the author calculated forward velocity using a scaled acceleration equal to constant maximum acceleration multiplied by a factor of `cmdVelocity.pos.x/0.5`, and reverse velocity using a scaled acceleration equal to constant maximum acceleration multiplied by a factor of `cmdVelocity.pos.x/0.1`.

As a result, dragging the cursor farther to the left or right from center is analogous to depressing the gas pedal harder while in “Reverse” or “Drive”, respectively.

- Brake pedal

The author implemented a “brake pedal” by reducing the velocity by three times the scaled acceleration calculated above when velocity was greater than zero. In practice, this reduction in velocity would have to be fitted to the braking profile of the simulated challenge vehicle.

- Elimination of redundant classes

The packaged controller used three classes to represent wheels: a base class (“Wheel”), and two derived classes (“DriveWheel” and “FullWheel”). The member variables and functions for these classes were very similar. The author collapsed the three classes into a single class (“Wheel”) by defining an additional member variable: `type`, which is assigned when the wheels are created by reading the type from the model XML file. Three types are supported: `DRIVE`, `STEER`, and `FULL`.

- Revise the steering controller to increase use of parameters

The packaged controller used non-parameter private class member variables. Most Gazebo classes use parameters in lieu of private class member variables. The author revised the steering controller to make increased use of parameters for values used by the controller to more closely conform to other Gazebo classes, but did not eliminate the use of private class member variables to store values calculated from the parameters in use.

VI.D. Parameters in use by the improved controller

In addition to the characteristics required to accurately model the representative challenge vehicle, the author implemented the following parameters for the purposes of evaluating the simulation targets:

- `useSwaybars`

When parameter `useSwaybars` is TRUE, the improved controller will attempt to compensate for "up" and "down" forces in each joint in a manner similar to anti-sway bars by applying a counter force to each joint. The counter force is calculated using parameters `swayForce` and `swayForceLimit`. When FALSE, the controller does not attempt to compensate.

- `swayForce`

When parameter `useSwaybars` is TRUE, parameter `swayForce` defines the force used to determine the moment applied in each joint due to displacement of the joint. The moment is the product of the force and the displacement.

- `swayForceLimit`

When parameter `useSwaybars` is TRUE, parameter `swayForceLimit` defines the maximum moment used to compensate for "up" and "down" forces in each joint. Forces which would result in moment greater than the sway force limit are not applied to the joint.

- `useConstantVelocityMode`

The improved controller reads velocity commands from the `playerv` utility. When parameter `useConstantVelocityMode` is TRUE, the controller will maintain

a constant velocity. When FALSE, the controller causes the model to coast to a stop.

- `useConstantSteeringAngleMode`

The improved controller reads steering commands from the `playerv` utility. When parameter `useConstantSteeringAngleMode` is TRUE, the controller will use parameter `constantSteeringAngle` to override the steering angle sent from the `playerv` utility. Steering angle at model CG will equal parameter `constantSteeringAngle`. When FALSE, the controller accepts steering commands sent from the `playerv` utility.

- `constantSteeringAngle`

When parameter `useConstantSteeringAngle` is TRUE, parameter `constantSteeringAngle` defines the steering angle at model CG and overrides steering commands from the `playerv` utility.

- `useSafeVelocity`

When parameter `useSafeVelocity` is TRUE, limits the maximum velocity at model CG to the maximum allowed by representative challenge vehicle and course geometry. When FALSE, the maximum velocity at model CG is equal to the final velocity. The final velocity is set using the `<velocityFinal>` declaration and, with the `<velocityFinalTime>` declaration, is used to calculate acceleration for the model, which is a constant.

- `velocityOffset`

When parameter `useSafeVelocity` is TRUE, parameter `velocityOffset` defines the amount by which to increase the calculated maximum velocity at model CG.

- `useTurnRadius`

When parameter `useTurnRadius` is TRUE, the calculation of the maximum steering angle, and maximum velocity and angular velocity at model CG, is based on parameter `turnRadius`. When FALSE, the calculation is based on representative challenge vehicle geometry and characteristics, specifically turning circle, track width, and section width.

- `turnRadius`

When parameter `useTurnRadius` is TRUE, parameter `turnRadius` defines the turn radius used to calculate the maximum steering angle, and maximum velocity and angular velocity at model CG.

VI.E. Validation of the improved controller

Output from the steering controller was logged to confirm the values calculated conformed to Ackermann steering geometry and other design decisions implemented by the author.

The author used parameters `useConstantSteeringAngleMode` and `useSafeVelocity` to limit the motion of the model to travel in a circle based on the representative challenge vehicle turning circle of 11.491 m at a constant steering angle of -0.376337 radians (i.e., a right turn at a constant steering angle of 21.56 degrees), launched Gazebo, and let the simulation run for 60 s. The author validated the controller by confirming:

- The radius used to calculate maximum velocity, maximum angular velocity, and maximum steering angle at model CG (“calculated radius”) was 6.63 m,

corresponding to one-half the sum of the model's turning circle, rear track width, and section width.

- The maximum velocity of the model was 8.72 m/s, corresponding to the calculated radius of 6.63 m and SSF of 1.17.
- The maximum angular velocity at model CG was 1.316 radians/s, corresponding to a maximum velocity of 8.72 m/s and circumference of 41.6 m.
- The maximum steering angle at model CG was 0.376337 radians, corresponding to a wheelbase of 2.619 m and calculated radius of 6.63 m.

The model completed twelve complete rotations in 60 s, with an average velocity of 8.71 m/s. Because the model accelerated from a complete stop, the average velocity was less than the maximum velocity at model CG. At a controller update rate of 50 Hz, the radius of the turning circle of the model ("reported radius") was 6.52 m. The error in the reported radius was -0.11 m (approximately 1.7 percent of the calculated radius). The author proposes this error may be due to the centripetal force applied by the steering wheels toward the center of the turning circle. No significant eccentricity was noted.

The author concluded the improved controller used representative challenge vehicle and course geometry, such as turning circle and SSF, to correctly calculate internal variables used to limit the maximum velocity and steering angle at left or right extent.

For example, to evaluate the rollover condition, in particular 2004 GCE course segment 2570-2571-2572, it was necessary to verify the improved controller calculated

maximum angular velocity at model CG correctly using SSF and turning circle. Using a SSF of 1.17 and turning circle of 11.491 m, the author confirmed the controller calculated the correct maximum angular velocity of 22.0632 rad/s for all wheels when steering angle was equal to zero, corresponding to a velocity of 18.2 mph, which is consistent with the Team 2005-06 challenge vehicle.

Problems encountered while validating the improved controller are documented in Appendix G.

The author then used parameter `velocityOffset` to determine how much the maximum velocity could be increased before rollover occurred. Table I summarizes the results:

Table I. Approximate simulation time at which rollover occurs at velocity offset.	
Velocity offset (m/s)	**Simulation time (s)**
0.10	-
0.20	-
0.22	-
0.24	-
0.25	48
0.26	19
0.27	14
0.28	12
0.29	10
0.30	9
0.31	8
0.32	6
0.33	5
0.34	4
0.35	3
0.36	2

With a velocity offset of 0.36 m/s, rollover occurred almost immediately after the model accelerated to maximum velocity, so the author discontinued the trials, concluding that increasing the velocity offset would not result in a reduction in the simulation time at which rollover occurs since constant acceleration was calculated based on parameters in use by the controller.

The simulation was then allowed to run for 300 s simulation time with a velocity offset of 0.24 m/s with no rollover event.

The information summarized by Table I is visually represented by Figure 1:

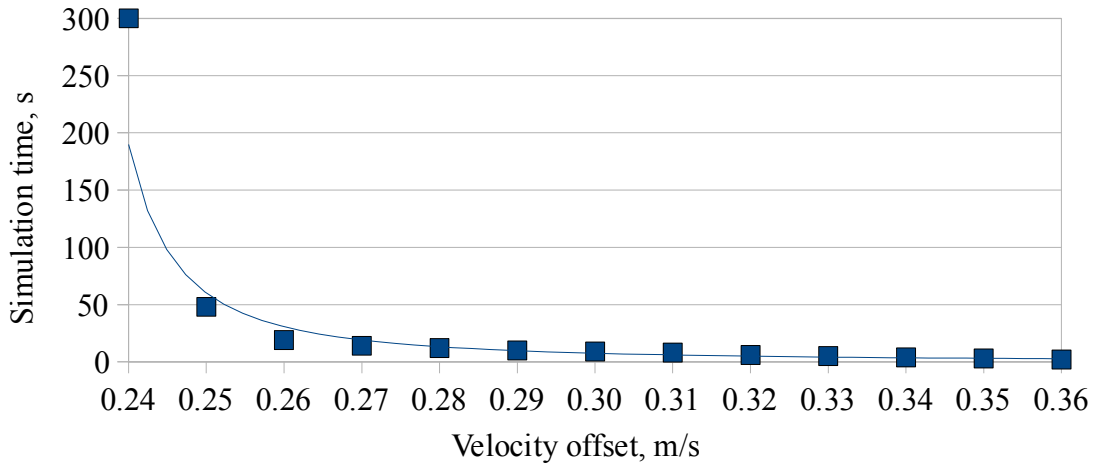

Figure 1. Approximate simulation time at which rollover occurs versus velocity offset.

The relationship between the simulation time at which rollover occurs and velocity offset suggests a power regression, but the data collected by the author is approximate. The author used the `playerv` utility to control the velocity of the model. As a result, there was a delay between starting the simulation and starting to accelerate the model of approximately 1.5 seconds.

Based on the results, the author concluded merely exceeding the maximum safe velocity may not result in rollover. The actual onset of rollover may be delayed, provided the model recovers before rollover occurs. As a result, the author proposed that the model could safely exceed the maximum velocity at model CG without rollover for a

limited time without adverse consequences, and that the time by which the model could safely exceed the maximum velocity at model CG without rollover would decrease as the velocity offset increased.

The author calculated the effective SSF of the model using the maximum velocity at which the model was able to successfully complete the test (9.08 m/s). The effective SSF of the model was 1.27 corresponding to a turn radius of 6.63 m, which well exceeds the representative challenge vehicle SSF of 1.17.

In general, suspension and tire effects contribute to a reduction of up to ten percent in SSF, meaning the increase over the effective SSF of the representative challenge vehicle of 1.07 was approximately 20 percent.

Because one of the purposes of high-fidelity simulation is to model real-world interaction which it may not be possible to evaluate in the real world, the author is not confident this is not a valid result:

- The model is an unrealistically rigid body.

A rigid-body model cannot predict time-dependent details of rollover such as those observed above.

- The accumulation of error in joints over time may result in bodies drifting away from their expected positions.

The ODE Manual states: “There is a mechanism to reduce joint error: during each simulation step each joint applies a special force to bring its bodies back into correct alignment. This force is controlled by the *error reduction parameter* (ERP), which has a value between 0 and 1.” and “The ERP specifies what proportion of the joint error will be

fixed during the next simulation step." ([44]).

Setting the ERP equal to 1.0 was not recommended. Setting the ERP to a value between 0.1 and 0.8 was recommended (0.2 is the default). An ERP of 0.8, the maximum recommended, was selected. This value conforms to the ERP in use in several packaged world files and models, and specifically world file "simplecar.world" which was used to evaluate the packaged steering controller.

- The representative challenge vehicle model does not model suspension and tire effects.

ODE provides a mechanism to control the "springyness" of joints, and which could be used to model challenge vehicle suspension and tire effects: Constraint Force Mixing (CFM). Setting the CFM to be less than zero was not recommended. The ODE Manual states: "If CFM is set to zero, the constraint will be hard. If CFM is set to a positive value, it will be possible to violate the constraint by 'pushing on it' (for example, for contact constraints by forcing the two contacting objects together). In other words the constraint will be soft, and the softness will increase as CFM increases." ([44]).

A CFM of 0.00001 (10^{-5}) was selected. This value conforms to the CFM in use in several packaged world files and models, and specifically world file "simplecar.world" which was used to evaluate the packaged steering controller.

- An effective SSF of 1.41 was calculated during evaluation of 2004 GCE course segment 2570-2571-2572.

This value exceeds the effective SSF calculated during validation of the improved controller. The author concluded the effective SSF of the model was independent of

model geometry, but may be a function of force due to lateral acceleration:

$$F = ma = \frac{mv^2}{r}, \text{therefore } a = \frac{v^2}{r}$$

$$\text{The rollover condition is: } \frac{v^2}{rg} > SSF$$

Force due to lateral acceleration, *a*, is balanced by centripetal force (the sum of forces exerted by the tires toward the center of the turning circle) at the tire-surface interface. ODE calculates tangential forces at the tire-surface interface ("contact point") using a "friction pyramid" approximation of the Coulomb friction model.

As a result, the author proposed selection of ERP and CFM in combination with friction approximation may result in an effective SSF of the model which exceeds the effective SSF of the representative challenge vehicle, and concluded additional testing in simulation and using the representative challenge vehicle would be required to effectively "tune" the model.

The model was not revised to have an SSF of 1.17 or an effective SSF of 1.07 because it would have required altering the geometry of the model, which was based on the representative challenge vehicle.