Virtual Sensing for Autonomous Vehicle Simulation in Chrono

by

Asher Elmquist

A thesis submitted in partial fulfillment of the requirements for an Honor’s in Research in Mechanical Engineering

at the

University of Wisconsin-Madison

December, 2016
Virtual Sensing for Autonomous Vehicle Simulation in Chrono

by

Asher Elmquist

Under the supervision of Professor Dan Negrut
at the University of Wisconsin-Madison

Approved by

Dan Negrut

Date

1/26/2017
Abstract

Due to the continued development and implementation of autonomous vehicles, the necessity for safe testing of these systems has increased. There is push-back to Uber bringing self-driving taxis to the streets as well as uncertainty with the safety of autonomous algorithms after the high profile Tesla autopilot crash early in 2016. Physical testing has begun to be implemented but virtual simulation capabilities could reduce the cost as well as increase the flexibility and safety of autonomous vehicle testing. Software developed at the University of Wisconsin-Madison has created an opportunity to simulate many vehicle, high-fidelity systems to model autonomous-autonomous, autonomous-human, and autonomous-environment interactions. By implementing realistic virtual sensors in Chrono, self-driving algorithms can be tested in edge cases including rain, snow, and fog. These sensors will provide accurate data based on physical noise models and affects from the environment.

Keywords: autonomous vehicles, virtual sensing
## Contents

1 Introduction 3

2 Virtual Sensing 4
   2.1 LiDAR ........................................... 5
   2.2 GPS ............................................. 6
   2.3 IMU ............................................. 8
   2.4 Camera ......................................... 9

3 CAVE Project 10
   3.1 Chrono C-axis: Network Interface .................. 11
   3.2 Chrono A-axis: Virtual Sensors Simulation .......... 12
   3.3 Chrono V-axis: Multi-Vehicle Support .............. 13
   3.4 Chrono E-axis: Virtual World Support .............. 14

4 Demonstration of Technology 15
   4.1 Single Vehicle Proof of Concept .................... 15
   4.2 Demonstration of Multi-Vehicle Capability and Scaling 17
   4.3 CAVE Demonstration: Multi-Client Simulation Across Network 19

5 Conclusion 20

6 Acknowledgments 21
1 Introduction

In 2014, 26% of the 6,870 million metric tons of CO2 equivalent produced were from the transportation sector, according to the US EPA [16]. In the following year, 140.43 billion gallons of gasoline were consumed in the United States. While there is research being conducted on improving fuel consumption, one solution car makers have moved toward is development of autonomous vehicles [8]. This is seen by many as a possible way to reduce traffic and decrease crashes due to human error. For the average commuter in Los Angeles, about 81 hours per year are wasted in traffic which increases the likelihood of injuries and fatalities [6]. This amount of wasted time is similarly high in other populous cities across the United States including Washington, DC, San Francisco, Houston, and New York City [6].

While some are welcoming the emergence of autonomous vehicles, many people see high risk involved in the proliferation of such technology. Uber has recently announced it will be bringing self-driving taxis to the Pittsburgh area. This is seen as dangerous by some because of a lack of testing. These concerned citizens would like to see extensive testing of the vehicles before consumers are asked to trust these autonomous drivers [12]. This concern was reinforced when, early in 2016 a Tesla vehicle crashed while in autopilot mode, killing the driver [10]. While Tesla maintains the autopilot system is not a replacement for attentive driving and is not a fully autonomous vehicle, many found this a rude awakening to the era of autonomous vehicles and believe more testing needs to be conducted before anyone relies on self-driving cars in their daily lives.

While testing has been done to understand autonomous vehicles, the extent and breadth of this testing falls short of consumer expectation. Although the University of Michigan recently created a 32 acre autonomous vehicle testing facility called MCity [11] in an effort to understand the response of autonomous vehicles within a changing environment, their capabilities are limited due to the high cost of set-up, time consumption of physical testing, and restricted access to this testing facility.

At the University of Wisconsin-Madison, the capability and expertise exists to create and develop a virtual autonomous vehicle testing framework which would assist autonomous
vehicle manufacturers and programmers by providing scenarios in which to test their vehicles. A virtual framework would allow for users to understand how their vehicles and algorithms hold up in numerous scenarios that are not achievable in any current test. In the simulation framework of project Chrono, full vehicle dynamics and physically based virtual worlds can be simulated in-silico to provide vehicles and their autonomous divers with realistic scenarios. The scenarios can be changed in such a virtual environment to encompass environmental effects such as snow, rain or fog, as well as human effects by adding in pedestrians and other human drivers. But more importantly, this simulation capability would provide for safe testing.

The basis of an autonomous vehicle simulation framework is project Chrono. Chrono is a simulation platform created by Professor Tasora in 1997 and has since been used and developed by the Simulation Based Engineering Lab at the University of Wisconsin-Madison. This is an open-source C++ library with the goal of supporting simulation in numerous engineering applications including robotics, granular materials, and vehicle dynamics especially when massive, scaling systems are required [5, 9]. To address the need for virtual autonomous vehicle testing, Chrono has been, and will continue to be, augmented to provide sensing capabilities, dispersed simulation of autonomous algorithms which will protect proprietary aspects of vehicle development, and expansive virtual world support.

2 Virtual Sensing

Because autonomous vehicles make decisions purely off of sensor feedback from the world around the vehicle, creating virtual sensors is a key component of any autonomous vehicle simulation platform. Sensing in physical systems can be difficult and is the source of error when path planning and obstacle detection [10]. This is why the sensing capability of Chrono is of utmost importance. It is vital the virtual sensors provide a realistic image of the surroundings back to the vehicle and can replicate noise and error observed by their physical counterparts. Some current robotics simulation platforms incorporate sensor data
simulation such as Gazebo and V-REP, but these simplify the noise models by assuming a Gaussian distribution. While this represents uncertainty, many sensors are dependent on environmental conditions such as rain, fog, or snow which is vital in autonomous vehicle simulations.

The sensor module in Chrono will be responsible for generating and recording data representing the data accumulated by various sensors. Example sensors include cameras, LiDAR, GPS, IMU, force sensors, etc. This will allow Chrono users to add robotic controllers to the simulation and integrate decision making into the process. Noise models are key to the simulation process as this is where error is introduced to the computer’s algorithms and can cause variation in decision making.

2.1 LiDAR

In more detail, the LiDAR, which has been successfully implemented in Chrono, is able to detect any collision object using ray tracing collision detection to create a range for each of the rays sent out by the LiDAR. When queried, the LiDAR passes back distances last seen by each ray of the LiDAR similar to the string of data which a physical LiDAR would send. All parameters of this sensor can be customized by setting update rates, resolutions, minimum and maximum ray distances, number of rays, and angles for which the LiDAR is active, to accurately represent the physical LiDAR desired. An example LiDAR is shown in Figure 1. This demonstrates how a LiDAR sends out rays which will return a finite distance when obstacles are detected. Based on these distances, computer algorithms can determine proper reactions, including obstacle avoidance, and readjust the planned path.

Because a LiDAR emits light and measures the time that light takes to return to the sensor, various materials reflect certain amounts of that light and scatter differently. This contributes to a material-based noise which should be mimicked by the simulation engine to provide the most realistic information back to the computer. Emitted light from a LiDAR will also scatter in rain and fog resulting in erroneous and limited data returning to the computer. In a test of any robot, especially an autonomous vehicle, the effects of this varying
Figure 1: Basic Two Dimensional LiDAR. A LiDAR may be used on an autonomous vehicle to detect other cars or obstacles.

data on performance and decision making will allow for accelerated software development of artificially intelligent drivers. In Chrono, the LiDAR noise models will be based on physical effects rather than simplified noise distributions.

2.2 GPS

One of the most important sensors used in autonomous cars is the Global Positioning System (GPS). The GPS in a vehicle enables the vehicle to track its position and monitor destinations. GPS modules send back a set of data containing time, longitude, latitude, number of satellites, and altitude. To mimic the physical sensor, the virtual sensor determines the position of the parent body, will apply a given noise model dependent on sensors parameters, and will return the GPS equivalent position in standard form. A simple GPS has been implemented in Chrono but will require further work to introduce physically accurate noise.

While an ideal GPS is a simple conversion of sensor position into a new coordinate frame, a virtual sensor that provides realistic information is an intricate process. GPS relies on satellites transmitting data to the sensor. The sensor picks up the signals and, based on the satellites’ position and the time of transmission, calculates the sensor’s own position. The
noise associated with a standard GPS module can be relatively high, but can also depend on
the weather and number of satellites the receiver can "see" [2]. Many modern GPS receivers
use differential correction to increase the precision drastically [7]. As Figure 2 depicts, this
technology works by implementing a stationary reference receiver at a known location which
receives GPS signals and relays a correction factor based on the accuracy of its own position.

Figure 2: Networked Differential GPS [7]. This shows how a differential GPS may use
reference stations located at known positions to correct for satellite positional uncertainties.

In Chrono, the GPS sensor must be able to distinguish between standard and differential
GPS modules as well as add noise based on weather effects. Having realistic GPS sensors in
Chrono will allow for vehicles and robots placed in a Chrono simulation to base their decisions
in a manner similar to the physical world.
2.3 IMU

An Inertial Measurement Unit (IMU) can contain an accelerometer, gyroscope, and compass, and is used to provide acceleration and orientation information. Figure 3 shows an example set of IMU data that a vehicle would be able to utilize based on the ISO standard vehicle coordinate system. Specifically, it would be able to see its $x, y,$ and $z$ axis accelerations as well as the pitch (rotation forward-backward), roll (rotation side-to-side), and yaw (rotation around the vertical axis).

Figure 3: IMU Sensor. An IMU is capable of returning acceleration and angular orientations to a central computer for decision making.

In the simulation software, this information is queried from the parent object to which the sensor is attached, processed with a given noise model, and returned. The simple or
"ideal" sensor has been implemented in Chrono to return the exact values but noise still needs to be understood. The noise model for an IMU consists of a constant offset, a moving bias or drift, and wide-band sensor noise. The wide-band sensor noise can be assumed to be a Gaussian distribution while the drift should be dependent on sensor parameters including sampling frequency [1].

2.4 Camera

For many systems, the primary form of information comes from a camera module. In the case of a vehicle, this can be used to maintain the vehicle's position within a lane, or distinguish and understand road signs. Because cameras are not managed by the physics engine, this sensor is dependent on the render engine. The camera will be implemented in the render engine to convert a known scene into an image. This image will be processed, have noise applied, then be returned to the simulation loop where it can be accessed by a vehicle driver, or other intelligent algorithm. It also provides a way to monitor a simulation even when sensor feedback is not required. The camera can be placed with a view of all, or a specific portion of the simulation and can be strung together to give a visual representation of what is, or has happened, to the objects in question.

Noise on a camera is not consistent with a Gaussian distribution as seen in current robotic simulation engines. In a camera module, noise is generated in a number of locations between the object and the output. These variations are due to the distribution of photons arriving in the lens, the sensor reading, and then the analog-to-digital conversion [3]. These effects are shown in Figure 4. Initially the camera sensor will only include the left two components, specifically the pre-amplifier noise which consists of photon noise and read noise. The photon noise is linearly dependent on pixel intensity, while read noise is purely a product of sensor characteristics and is independent of intensities. Read noise will be implemented by determining a Gaussian distribution and adding that to each pixel independently. The photon noise will be implemented by applying the product of a second Gaussian distribution and the pixel's own intensity to each pixel in the frame.
The camera sensor will focus on the first two components: photon noise and sensor read noise which are associated with amount of light and the translation of this noise into signals respectively.

3 CAVE Project

The Connected Autonomous Vehicle Emulator (CAVE) is a framework for providing a virtual environment to test autonomous vehicles. It combines modules currently implemented in project Chrono with three new modules. These new modules will be; a sensors module to provide realistic sensing to the autonomous vehicles, a virtual world/environment in which the vehicles will be immersed, and a network by which these autonomous vehicle will connect to this framework. These modules would work alongside any current module required for the desired vehicle and environment scenario. The network will be responsible for handling all server-client interactions to allow for simulation connection while the sensors and virtual world would work together with Chrono::Vehicle to provide detailed information for all drivers and agents participating in this framework. Other modules of Chrono will be used on an as needed basis. For vehicles, Chrono::Vehicle can be used to model the dynamics and receive driving inputs.

The simulation flow for a single vehicle is shown in Figure 5 and can be extrapolated out for hundreds or thousands of autonomous and human-driven vehicles. The server will be hosted by the Simulation Based Engineering Lab in Madison, WI but individual vehicles and vehicle simulations will happen across the country. For example, car manufacturer A could log into the server with a vehicle and relay information to the server about its current position and the server will send back the position information of each other vehicle currently
on the server. In this way, the computation will be distributed across numerous computers with no one computer simulating the physics of the entire world.

Figure 5: CAVE Simulation Flow. All calculations will take place on the CAVE Client with the CAVE Daemon holding and distributing world and other client information.

### 3.1 Chrono C-axis: Network Interface

The CAVE project will provide a means by which many autonomous and human driven vehicle can interact in a virtual environment. To enable this, a reliable and skinny server is necessary. The server will know about all the clients in the virtual world but will not perform any of the simulation - all calculations will happen on the remote clients for speed. The server will allow clients to connect and insert a vehicle, pedestrian, bicycle, or other world object into the environment to interact with other clients. In this way, atypical and unpredictable scenarios will be presented to various autonomous vehicles.

The server creates an individual thread for each client to handle all communication
independently. Once the dedicated listener thread recognizes a new client, it creates this thread and continues to listen for new clients [4].

Using Boost as the underlying library, the networking module implements a TCP design but could be changed to UDP to allow for better scaling in the future [4]. This scaling ability will be important as the target is to allow for hundreds of vehicles to be connected to the server at any given time. Another way the server will maintain speed is to restrict how much of the world each client sees. Since a client does not need to receive information about another client a mile away, the server will divide the clients up and send only the relevant information each update step.

The server would hold the official simulation time and have a fixed time step. It would require all clients to remain in lock-step with the server but allow for fast computing clients to reduce the time step for more accurate simulations. In this respect, each client would be allowed to have its own time step given a constant time budget for each step taken by the server. This would allow for clients to choose high accuracy simulations while maintaining mass amounts of data and driven miles by the clients who wish to achieve such results. The server and client interaction along with the message passing system is detailed further in SBEL Technical Report 2016-06 [4].

A future development of the networking component will be adding "connected" simulation capabilities. Local ad-hoc networks will be simulated to represent vehicle-to-vehicle communication. This is important since vehicles will need to pass information directly to each other without just sensing the other vehicles. As this technology is created, accurate simulation of network failures needs to reflect physical phenomena in order to provide the autonomous vehicles with realistic scenarios in which developers can study the reaction.

3.2 Chrono A-axis: Virtual Sensors Simulation

The CAVE A-axis represents support for simulating the "autonomous" component of self-driving vehicles and provides sensor information on which the driver algorithms will be able
to base its decision making. In order to test autonomous driving algorithms with accuracy, these sensors need to provide physically realistic data. The sensors being implemented in Chrono, as discussed in section 2, are currently idea sensors, meaning they provide exact and unbiased/precise data, but will be augmented with physics based noise models dependent on sensor parameters, characteristics, and trends, as well as environmental impact. This impact is most evident and intuitive in light dependent sensors such as camera or LiDAR. A LiDAR emits light and measures the time until reflection. This reflection is clearly dependent on reflective material as well as the air quality; a LiDAR scan during snow-fall will scatter and return partial data, skewing the information and providing bogus data in some cases.

The goal of this Chrono module is to provide realistic sensor data to the vehicle algorithms to understand edge cases where autonomous vehicles may not have a perfect representation of the surrounding (i.e. rain or fog). In close development with the virtual world, these sensors will need to understand the material effects and be able to simulation atmospheric conditions. They should also be able to provide "false" information as in a case where the sensor itself fails. These will provide much needed edge cases on which algorithms can be stress tested, and improved.

3.3 Chrono V-axis: Multi-Vehicle Support

The V-axis of this project is physics based vehicle support. In order to study the behavior of a vehicle without compromising safety, the true physics of the vehicle must be modeled to limit the abilities to depict real world scenarios. For a vehicle approaching a stop sign in the snow, the decision making should take into account slippage and poor braking. With the physics based approach, the vehicle will have appropriate limitations and these accurate driver studies can occur.

Chrono::Vehicle is the basis of this thrust and is well developed and validated against test data and other software. Chrono::Vehicle provides template based support for vehicle modeling to allow for a broad range of on and off road tests of vehicle characteristics. Within this toolkit, Chrono allows for the generation of high fidelity vehicle models based on their
subsystems, each of which is simulated completely from power-train and engine, to suspension and steering, as well as tires for vehicle-road interaction. This high-fidelity system allows for the simulation of traffic scenarios both on-road as well as off-road in the case of fording mud traversal. Examples of Chrono::Vehicle simulations can be viewed on the Simulation Based Engineering Lab’s website [14].

Chrono::Vehicle, its capabilities, and underlying techniques are laid out in full in [13].

3.4 Chrono E-axis: Virtual World Support

To provide realistic scenarios to test autonomous controllers, a fully immersive virtual world is required. This virtual world must mimic, to a high degree of accuracy, the physical world in which an autonomous car would be operating in order to test edge cases with fidelity. The sensors will be developed to match the noise and trends of physical sensors, but if the information being sent to the sensor is cartoonish or inaccurate, these sensors will not be able to convey data as desired.

The virtual world will provide a way to load and maintain world components in Chrono. It would create a way to define buildings, roads, sidewalks, signs, and other roadway features. Having a way to generate roadways based on regions in the physical world would be beneficial as it would allow any autonomous cars that are using maps and route planning to work seamlessly within the virtual world as well.

This virtual world must be able to load and unload chunks of the world to limit the resource usage. Dynamic agents within this virtual world will not need to know about objects far away and thus should not have those distant objects loaded into the simulation, thereby wasting memory and computation. Instead, as the agent approaches these distant objects and has a chance for collision or recognition from its sensors, these distant objects should be added to agent’s simulation.
4 Demonstration of Technology

4.1 Single Vehicle Proof of Concept

In order to demonstrate the capability to simulation autonomous vehicles, a proof of concept was developed consisting of a single autonomous vehicle. This included a dynamic vehicle and world simulated in Chrono with sensor simulation and rendering in Gazebo since the functionality was not yet developed within the Project Chrono software. As can be seen in Figure 6, the simulation depended on having Chrono and Gazebo tightly integrated. The world information was shared between Chrono and Gazebo with Chrono calculating physics and Gazebo supporting the sensor simulation.

Figure 6 shows the flow of the simulation shared between Chrono and Gazebo. The driver received sensor data from gazebo and, after processing it, returned driver inputs into Chrono::Vehicle. Both Chrono and Gazebo then output the simulation results in the form of dynamic and state data from Chrono and sensor data and rendering from Gazebo for visualization. By using Chrono to simulate the physics, it allowed for greater flexibility in vehicle definitions, and additional modules. The expandability of Chrono to handle fluid-solid interaction, deformable terrain, and many-body systems in parallel is crucial for future vehicle dynamics.

![Simulation Flow](image)

Figure 6: The initial simulation flow when using Chrono-Gazebo integration.

A key feature of the Chrono component of this simulation is demonstrated in Figure 7.
This image was captured as the autonomous vehicle navigated a ditch in the road with dynamic obstacles at the bottom. All physics were simulated in Chrono and demonstrated the ability for Chrono to handle many-body systems and collisions. It also showed the simulation frameworks ability to have the vehicle interact with the environment rather than the environment simply being fixed around the vehicle. A simulation with the goal of understanding self-driving cars should also understand the limitations of that vehicle’s physics.

![Vehicle navigating ditch with dynamic obstacles.](image)

Figure 7: Vehicle navigating ditch with dynamic obstacles.

By using Gazebo and its sensor capability, an autonomous driver was able to maintain the vehicle’s position on the road. It processed images produced by the camera and, finding the center stripe on the road, steered the vehicle toward the center of the road some distance ahead. This simple driver also demonstrated the ability to use Lidar data to prevent a collision. Based on the information received by the Lidar, the vehicle was able to detect obstacles and slow down as it approached these obstacles. Below, in Figure 8, the vehicle can be seen navigating a curve in the road denoted by the center stripe. The Lidar data is also shown and the Jersey Barriers can be seen as individual points in the displayed Lidar data.
4.2 Demonstration of Multi-Vehicle Capability and Scaling

The second proof of concept was demonstrating a use case and validation for a multi-vehicle autonomous vehicle simulation framework. It also demonstrated the need for a scalable and flexible solution. In 2007, a traffic jam study was conducted in Japan to examine the propagation of traffic shock waves backwards along a road. For this experiment, 22 vehicles were positioned around a circular road with circumference of 230 m. These drivers were told to maintain a velocity of 30 km/hr and a safe minimum distance [15]. Once the vehicles were driving, a traffic jam appears on the circle and begins to propagate backwards around that circuit.

In the Chrono-Gazebo framework, this same study was created with similar driver parameters. The vehicles were autonomously driven by a Chrono Driver which could read in the vehicle’s speed in order to maintain the 30 km/hr target and received LiDAR data so the drivers could maintain a safe minimum distance between themselves and the vehicle ahead. Figure 9 shows a snapshot of the vehicles from the side as well as from a birds-eye view. In this top view, the traffic jam can clearly be seen at the top right with a vehicle density much higher than the rest of the road.

While this shows that a traffic jam was seen in both the physical and virtual models, it
Figure 9: Vehicles driving in a circle demonstrating traffic jam propagation properties.

is important to understand the similarity of the results. By plotting the vehicle position vs time in Figure 10, the changes in traffic density can clearly be seen. On the left in Figure 10a, the physical experiment is plotted with a traffic propagation velocity of -20 km/hr. Figure 10b shows the same graph for the virtual setup. This setup saw similar propagation velocity at -17 km/hr.

(a) Physical Traffic Circle Experiment [15]  (b) Virtual Traffic Circle Experiment

Figure 10: Comparison of Physical and Virtual Traffic Jam Experiments
This demonstration was instrumental in showing the validity and use of a multi-vehicle autonomous simulator. It showed that this system can represent physical behavior seen in traffic jams. The major takeaway from the framework was a need for improved scalability and distribution. This simulation could not be directly parallelized or run on a cluster with the overhead from Gazebo as the camera rendering required a screen context. Additionally, by using Gazebo and Chrono, much of the physics are duplicated resulting in a waste of computational resources.

4.3 CAVE Demonstration: Multi-Client Simulation Across Network

In accordance with the CAVE framework described previously, an initial demonstration was created to show the capability that this could afford. This framework allows for both autonomous and human driven vehicles to connect and operate in the same world while being simulated on separate computers. To illustrate this network capability, a human driver logged into the server followed by two separate autonomous vehicles. The autonomous vehicles were programmed to follow the vehicle ahead of it on the road. Figure 11 demonstrates this scenario. This is a human-led autonomous convoy. The leader of the pack is human driven by connecting a steering wheel game console to a computer running Chrono vehicle. The two autonomous vehicles are following the center point of the vehicle ahead of it while maintaining a minimum distance using the LiDAR that has been added to Chrono. These are rudimentary autonomous vehicles as the goal of the framework is not to create the drivers but rather to allow driver developers to use this platform.

This demonstration provides proof of concept for the network capabilities and the autonomous support via sensors but also shows the work needed for improved rendering and virtual world.
Figure 11: Demonstration of follow the leader autonomous convoy. Each vehicle is simulated on separate client.

5 Conclusion

This project is still in full development, but currently allows for autonomous vehicle testing based on a physics based simulation engine. For CAVE, vehicle support is well implemented with support for deformable terrain, template based wheeled and tracked vehicles, and fluid solid interaction such as fording. The network, or CAVE Daemon, allows the connectivity of vehicles independent of driver type to accommodate autonomous vehicle developers and human drivers alike. The autonomous support in the form of Chrono Sensors currently supports LiDAR, GPS, and IMU. These sensors are “ideal” sensors without noise models. The virtual world is handled internally by Chrono and has not seen significant development.

There is significant future work in three main areas of CAVE. This work is laid out from
most to least developed. CAVE network will implement a heartbeat to maintain simulation times constant across all clients on the server. This will preserve physics accuracy and valid interaction between vehicles. This network will also implement a system to simulate vehicle connectivity via ad-hoc networks that would send agent data to augment decision making from sensor readings. Chrono Sensors and CAVE autonomy will see further implementation of current LiDAR, GPS, and IMU to add physically accurate and dependent noise by which autonomous vehicle would realistically base decisions. A camera will also be added to provide visuals to these drivers. The camera will also see development of realistic noise models based on each level of image processing and acquisition.

6 Acknowledgments

This project was supported in part by the Faustin Prinz Undergraduate Research Fellowship and by the Simulation Based Engineering Lab at the University of Wisconsin, Madison.
References


