Multi-Modal Distributed Simulation Combining Cars, Bicyclists, and Pedestrians

Joseph K. Kearney, PhD
Professor
Department of Computer Science
University of Iowa

David A. Noyce, PhD
Professor
Department of Civil and Environmental Engineering
University of Wisconsin – Madison

Multi-Modal Distributed Simulation Combining Cars, Bicyclists, and Pedestrians

Joseph K. Kearney, PhD
Professor
Department of Computer Science
University of Iowa
https://orcid.org/0000-0002-0443-8152

Soumyajit Chakraborty
Graduate Research Assistant
Department of Computer Science
University of Iowa
https://orcid.org/0000-0003-4318-0681

David A. Noyce, PhD
Professor
Department of Civil and Environmental Engineering
University of Wisconsin – Madison
https://orcid.org/0000-0001-5887-8391

Yuanyuan Jiang, PhD
Assistant Professor
Department of Computer Science
California State University – San Marcos
https://orcid.org/0000-0002-3070-811X

Kelvin R. Santiago-Chaparro, PhD
Assistant Researcher
Department of Civil and Environmental Engineering
University of Wisconsin – Madison
https://orcid.org/0000-0001-6897-0351

A Report on Research Sponsored by

SAFER-SIM University Transportation Center

Federal Grant No: 69A3551747131

September 2018

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

Table of Contents

List of Figures

List of Tables

Abstract

A collaborative project between the University of Wisconsin-Madison and the University of Iowa was conducted to understand the feasibility of conducting driving and pedestrian simulator experiments that involve multiple agents (humans) in the simulation. As part of the collaborative project, a distributed driving simulation environment was created that can be used to conduct driving simulation experiments with multiple human agents participating in the simulation as a pedestrian or as a driver. The distributed simulation environment was created using the Unity game engine. A pedestrian can participate in the experiment using a virtual reality headset, while the driver can participate using a regular computer connected to a screen and a steering wheel control system. The distributed nature of the environment, i.e., the ability to have subjects be part of the same experiment from geographically distant locations, is achieved by relying on the built-in networking functionality provided by the Unity game engine. As will be described in the report, the simulation environment was created by relying on the tools provided by the integrated development environment used by the Unity game engine and by creating supplemental modules created to support common tasks in driving simulation experiments.

1 Introduction and Background

Driving, bicycling, and pedestrian simulators have proven to be valuable tools for investigating the underlying causes of crashes and testing various engineering and educational countermeasures to reduce the risk of crashes. However, most research efforts have focused on studying how individual drivers, bicyclists, or pedestrians perform in a specific scenario where experiment participants are exposed only to scripted (static or dynamic) actors; little has been done to study the interactions of drivers, bicyclists, and pedestrians in controlled simulator experiments.

By connecting a group of human-in-the-loop simulators, multiple participants (e.g., drivers and pedestrians) can share the same virtual environment and be placed in complex and hazardous conditions that cannot be studied in the real world due to the difficulty of creating the necessary controlled conditions and due to the ethical issues raised by placing participants in danger. Unfortunately, the availability of research tools that make the aforementioned type of research possible is limited.

1.1 Highlights of Previous Research

There is a long history of research and development in distributed simulation connecting remote sites into a common, simulated world. Much of the early work in distributed simulation was aimed at military applications beginning in the 1980s and continuing through today [1]. Two approaches emerged for managing network communication and data consistency across distributed simulations: client-server and peer to peer (P2P).

Multi-participant connected driving simulators have been developed to examine the real-time interactions of drivers at intersections and in collision avoidance situations (e.g., two drivers encountering each other at the crest of a hill) [2]. Connected driving simulations have also been used to study the impact of new vehicle technologies, including driver assistance systems and platooning. We know of only one study that used connected-simulation technology to link a pedestrian and driving simulator [3]. This study examined driver yielding behavior in different pedestrian-crossing situations and

found a difference in driver behavior depending on whether the pedestrian was a real person in a connected simulator vs. an autonomously controlled pedestrian agent.

Numerous research teams are exploring ways to use the latest generation of gaming platforms to build driving, bicycling, and pedestrian simulators. For example, the Hank Lab at the University of Iowa has built pedestrian and bicycling simulators using Unity3D as the simulation engine and rendering platform. The Unity3D platform provides support for network-based multiplayer gaming through Unity's High-Level API (HLAPI) system. The HLAPI is based on a client-server architecture. The server manages the communication and maintains consistency across the network. One computer can act as a dedicated server. Alternatively, one of the clients can take the role of the server (called a host).

1.2 Objectives

The aim of this project was to explore the potential of the Unity3D platform for building a connected driving/pedestrian simulation system. The primary goal was to build and test a prototype system linking a driving simulator at the University of Wisconsin to a pedestrian simulator at the University of Iowa using the Unity3D HLAPI for network communication. The pedestrian simulator uses the HTC Vive head-mounted virtual reality display to create an immersive experience.

Timing tests were conducted to determine the responsiveness of the system. In addition, the project examined the use of teleportation to allow multiple encounters between driver and pedestrian. One of the basic problems in connected driving/pedestrian simulation is structuring the scenario so that the driver and pedestrian have repeated interactions. In the real world, once a driver passes a pedestrian, he/she must loop back to re-encounter the pedestrian. In a virtual environment, participants can be teleported to new locations to create repeated interactions. However, it is unclear if such discontinuous jumps in location will be disruptive to the experience and if it will lead to increased propensity for simulator sickness. Our goal was to implement a scenario

with repeated encounters using teleportation and conduct informal tests of participant tolerance for such jumps.

1.3 Collaborative Nature of Project

The project described in this report was a collaborative effort between the University of Iowa and the University of Wisconsin-Madison. The University of Iowa is home to state-of-the-art bicycling and pedestrian simulators and numerous driving simulators, including the National Advanced Driving Simulator. The University of Wisconsin-Madison team has experience with the development of driving simulator scenarios and data analysis procedures. The work by the University of Iowa and the University of Wisconsin-Madison focused on a simulation platform created using the Unity game engine. A parallel effort took place at the University of Massachusetts-Amherst and focused on connecting multiple simulators using a Realtime Technologies platform.

2 Creation of a Simulator Environment for Pedestrians

The University of Iowa Hank Lab has two large-screen pedestrian and bicycling simulators, as well as head-mounted display (HMD) virtual reality (VR) systems. For this project, we decided to use the HTC Vive HMD as our main platform. The Vive and similar HMD systems (e.g., the Oculus Rift) are gaining popularity among the simulation and VR communities due to their affordability, portability, and ease of deployment. We used the Unity3D game engine for building the simulation. The Vive HMD has a plugin available on Unity3D asset store (called SteamVR Plugin) that integrates the Vive into Unity applications with pre-made camera view frustums and automatic updates of head position based on the Lighthouse tracking system.

The virtual neighborhood was created using the Unity3D asset Simple City.

Figure 3.1 – The Simple City Unity3D asset used to build our road database.

We modified the town model layout and created a long/straight road with 24 intersections for the car to drive on. The pedestrian view-frustum object, which contains cameras for the user's two eyes and a bounding box representing the motion-tracked space, was placed at the first intersection when the application was initialized. The position and orientation of the pedestrian's cameras were controlled by the Vive Headset motion-tracking input streamed in via the SteamVR plugin. When the user physically moved around in the tracked space, the cameras also moved, providing a consistent view of the virtual world. When the user moved close to the tracking space boundary, a

blue grid appeared to inform the user to stop walking. When the user stood at the edge of a curb, he/she saw cars coming from the left side.

A leading driverless vehicle was created in the virtual world starting 2 meters ahead of the car that was controlled by a real driver. The lead vehicle movesdat a constant speed of 25 MPH (11.176 m/s). The driver was instructed to follow the lead vehicle at a safe following distance. The pedestrian was instructed to cross between the lead car and the driver-controlled car if he/she could safely do so. When the two cars passed the pedestrian, the Pedestrian Vive Node was teleported to another intersection in the path of the driver and lead vehicle. Thus, the vehicle re-encountered the pedestrian.

A 3D virtual character model from the Unity3D asset store was used to represent the pedestrian as a virtual avatar. Its position was controlled by SteamVR Vive HeadSet input. Thus, the avatar tracked the movement of the pedestrian user. The pedestrian could look down and see his/her virtual body. The driver user can saw a virtual character move when the pedestrian user moved. A motion-captured walking animation was triggered when the pedestrian's position changed, and a standing idle animation i\was triggered when the pedestrian stopped moving.

2.1 Similarities Between Pedestrian and Bicycle Simulators

We built pedestrian and driver interaction simulation in this project. The system and lessons learned apply to a bicycle simulator. The Hank Lab has a bicycle set-up in the HTC Vive tracking field when running bicycle simulations. The bike's steering angle and moving speed are streamed into the Unity3D engine via a USB port. We still use the SteamVR Plugin to get a "Bike Vive Node" object that is similar to the "Pedestrian Vive Node" object. The main difference is that when a user rides a bike, the whole Bike Vive Node moves, instead of only the cameras. The Bike Vive Node's updated location and orientation are calculated from bike steering angle input and speed on every simulation frame. When a pedestrian walks around in the tracking field, his eye positions move within the tracking bounding box. When a user rides the bike, the relative position between the user's eyes and the tracking bounding box remains the same, but the whole

bounding box, together with cameras, needs to move around in the virtual world for the rider to move around virtually. A bike model is used instead of a pedestrian model. We still need to work on mapping the real bike's steering and pedaling to the virtual bike.

A bicycle simulator is very similar to the pedestrian simulator otherwise. A bike model moves across the intersection when the rider decides to cross between the leading car and the driver-controlled car. The Bike Vive Node gets teleported to following intersections for more encounters.

3 Creation of a Simulator Environment for Driving

Multiple driving simulator platforms exist on the market today, and researchers have successfully used these platforms to understand safety issues associated with the existing transportation system. Unfortunately, existing platforms often have limitations that prevent researchers from ultimately achieving the research goals outlined for a project without significant effort. In the experience of the authors, these efforts are often related to the underlying architecture of the existing simulator platforms. For example, traffic signal systems or vehicle generation models used by some of the existing driving simulator platforms are not aligned with the theory used by transportation engineers to model the characteristics and behaviors observed in transportation systems. Because of these limitations, researchers are often forced to spend significant time tweaking the system and creating additional modules to be able to study the questions associated with a proposed system.

Based on the experience of the authors with existing driving simulator platforms, the research team realized that using the existing driving simulator platforms to create a distributed simulation environment would require a significant amount of work and modifications. Furthermore, these modifications to the platform would only address issues associated with the distributed environment to simultaneously study the interactions of multiple human subjects across different transportation modes. Therefore, other limitations of existing platforms such as those related to signal systems and traffic generation would go unaddressed. As a result, the research team decided to pursue an approach in which a streamlined version of a driving simulation environment would be created from “scratch” using modern tools, thus providing a platform that can grow in the future and support research.

3.1 Selection of Unity as a Development Platform

To create a streamlined version of a driving simulator platform that can be used for creating a distributed simulator environment, the Unity game engine was used. The Unity game engine was selected by the research team due to the wide support that is

available for the platform online and offline. There are numerous online support forums, tutorials, and examples, which simplifies the process of starting a project from zero. Offline, numerous books have been published that can be used as training material and supplemented with the online resources.

3.2 Overview of Unity-Based Driving Simulation Platform

Like the pedestrian simulation component, the driving simulation component of the distributed simulation environment was created using the Unity game engine. At its core, the Unity game engine provides for a powerful platform that can be used as a visualization tool for 3D models of a roadway environment in the form of a game. Included with the engine are physics models, camera views, and object control tools that can be used to build sophisticated simulators. For example, the physics model can be configured to resemble the behavior of a vehicle. Treating a “game” created with the Unity game engine as a driving simulation tool requires the addition of supplemental modules that allow control of the environment with a steering wheel as well as the collection of data. The sections ahead describe additional components created to turn a visualization tool created in the Unity game engine into a driving simulation environment.

3.3 Creation of Supplemental Tools

While there is built-in support for game controls through the game engine as well as debugging capabilities, the unique challenges of a driving simulator, the expected uses of the platform, and the potential expansion opportunities require the creation of additional tools specifically designed to support a driving simulation system. Additional tools developed include an independent interface capable of connecting to multiple steering wheel systems, a virtual control interface used for testing and debugging purposes, and a data visualization and playback mechanism. Additional details about these tools are presented in the sections ahead.

3.3.1 Steering Wheel Integration

When the Unity game engine is used to create a driving simulation system, a wide range of options are available that enable control of the simulation. For example, one of the features of the integrated development environment provided by the Unity game engine is the ability to map different inputs to events in the simulation. However, throughout the project the research team found that integration with different steering wheels proved difficult. For example, the standard steering wheel controller shown in Figure 3.1 did not operate as expected without the use of custom code.

Figure 3.1– Sample Game Controller Compatible with Driving Simulator

To facilitate integration with different controls that include a steering wheel and corresponding pedals, a custom module was created. The custom module made it possible for the research team to demonstrate how a driving simulation environment in Unity can be created to interact not only with existing control interfaces but also with custom ones. As will be shown in the sections ahead, the research team demonstrated the ability to use the vehicle cabin of a full-scale driving simulator to control the Unity-based simulation environment.

3.3.2 Virtual Control Module Integration

One of the challenges that existing simulator platforms present when creating and testing scenarios is the lack of detailed controls for testing and debugging the simulation. As part of the creation of the streamlined driving simulator platform described in this report, the research team created a control interface for the simulator platform that can be used in lieu of a steering wheel controller. The interface allowed the research team to quickly test the environment and participate in debugging sessions with collaborating institutions without the need to assemble a small desktop simulator every time collaboration was needed. A screenshot of the control interface is shown ahead in Figure 3.2.

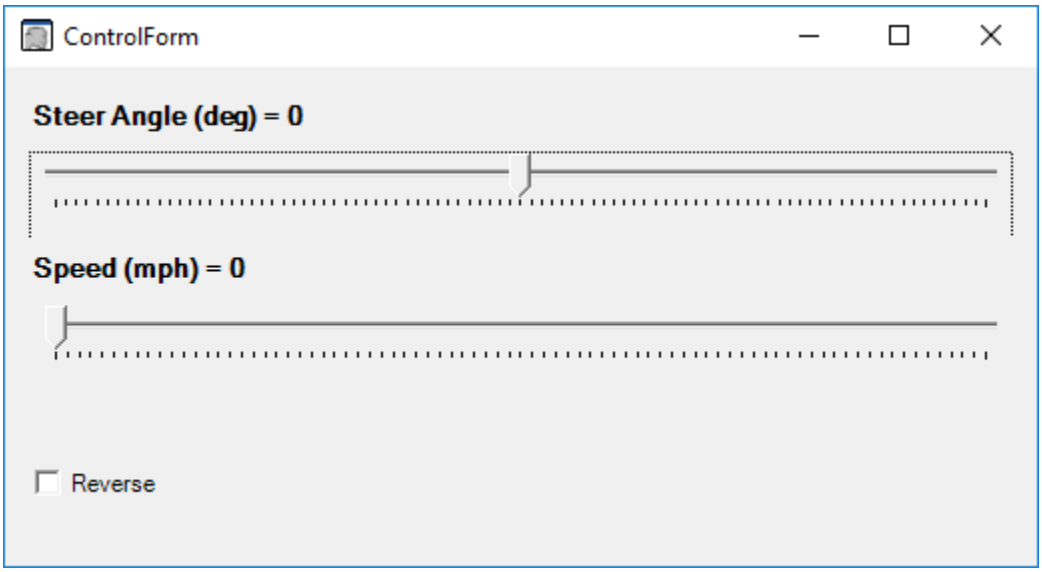

Figure 3.2 – Control Interface for Debugging Purposes

The control interface shown was created as a dynamic link library (DLL) that the simulator platform can access for instructions about the vehicle speed and steering. Steering and speed values are exposed as public variables to the simulator platforms that load the DLL. The DLL was created in C# and requires the use of the *UnityEngine* library to facilitate interaction with the Unity-based simulator.

3.3.3 Data Recording and Visualization System

Without data collection capabilities, a driving simulation tool is simply a visualization tool built using the Unity game engine. Therefore, a data collection module was added to the system. The data collection module can be used to log the position and orientation of the pedestrian and vehicle along with input values such as steering and brake pedal position. A timestamp is attached to each of the aforementioned measurements, and the frequency at which data is logged is a function of the frame rate that can be achieved in the simulation based on the computer hardware on which the simulation is executed. Data collected can be stored in a format suitable for analysis using tools commonly available to research teams in the field of driving simulation.

An additional module was created to explore how the research team can take advantage of the visualization functionality of the Unity game engine to allow visualizing data collected during a driving simulation experiment. The created module allows end users to select the 3D model used for the simulation as well as the data collected for a user during the simulation in order to “replay” the simulation. Having a tool that allows researchers to replay a simulation after the experiment is over provides added value to collected data that is key to quantifying and analyzing the behaviors observed during the experiments.

3.4 Additional Challenges and Limitations

One of the main challenges associated with creating a driving simulator platform from “scratch” is the need to define a vehicle dynamics model that is sufficiently accurate for the research being conducted. As part of the research project, the team explored different alternatives for a vehicle dynamics model that could be included in the simulation. However, after identifying the immediate research goals, the team decided to treat the vehicle dynamics component as an extremely modular component of the project and isolate it from the simulation environment. This design decision allowed the research team to create a simplified vehicle dynamics model that met the immediate

needs of the project while providing an opportunity for future expansion as the driving simulator platform grows in terms of fidelity.

3.4.1 Vehicle Dynamics Model

The vehicle dynamics model is a key component of any driving simulation platform. However, for many research projects conducted in driving simulation environments, the characteristics of the vehicle dynamics model do not play a significant role in the behaviors captured during the simulation (e.g., when the primary focus of the project is on pedestrian behavior). As a result, the research team focused on the non-vehicle-dynamics components of the simulation environment, such as the control and data collection interfaces, and decided to use a simplified model for vehicle dynamics. Steering behavior of the vehicle is achieved by having the vehicle follow a circular path when turning; the radius followed is determined based on the steering angle. The speed of the vehicle was modeled as a linear function of the position of the gas pedal.

4 Integration of Environments, Network Testing, and Lessons Learned

The different components that make up the driving simulator environment and the pedestrian simulator environment were combined into a single system. The combined version of the system is what defines the distributed simulation platform described in this report. A simple experimental scenario for testing purposes was created and expanded with network capabilities that make the distributed nature of the platform possible.

4.1 Scenario Creation for Test Experiment

The scenario consists of a grid network of two-lane roads. The pedestrian is initially situated at an intersection, and the driver's vehicle is positioned behind an autonomously controlled vehicle on the same road as the pedestrian. The driver's task to follow the autonomous vehicle at a safe following distance. The lead vehicle is programmatically controlled to approach the location of the pedestrian, and the pedestrian is instructed to cross between the two vehicles. The goal is to look at the responsiveness of the driver as the pedestrian initiates his/her movement to cross.

Once the vehicle passes the pedestrian's location, the pedestrian is teleported to another intersection ahead of the driver's vehicle. This allows repeat encounters between the driver and the pedestrian. A message is displayed in the participant's headset asking him/her to direct his/her gaze across the intersection. The goal is to minimize any visual disruption that may lead to simulator sickness. A countdown clock warns the participant that a teleport jump is coming.

4.2 Adding Networking Capabilities to Unity-Based Simulations

The Unity3D engine supports distributed multi-user interaction via network APIs. We used the HLAPI in this project. The HLAPI supports a drag and drop style of programming to create networked games. One user application is set as network host, and other users can join the game as network clients. All the non-player objects, like the driverless car in this project or enemies in a shooting game, are created by adding HLAPI networked components onto the game objects. The position and orientation of

those game objects are first calculated by the host application and automatically synchronized to all the client applications by the HLAPI. The player objects directly controlled by user input (e.g., the pedestrian and the driver-controlled car in this project) are managed by the corresponding client application, and their positions and orientations are synchronized to other client applications by the HLAPI.

When all the users use the same player object (e.g., if all users were drivers), the player objects can be managed by the HLAPI with a drag and drop style that requires no scripting. In this project, we use different player objects in two applications (one pedestrian and one driver-controlled vehicle), so we fused a modified version of the Unity HLAPI network manager acquired through Git and made some modifications to complete the task of sending pedestrian and vehicle information across the network. We created an empty game object called “Network Manager” and added the modified network manager scripting components. Both the pedestrian and driver-controlled vehicle objects have network identity and network transform scripts from the HLAPI attached. They are created as prefab objects and linked to the Network Manager.

When starting the pedestrian simulation, the programmer selects “Is pedestrian” from the network interface and starts the game as host. The pedestrian virtual character will appear on an empty road in the Unity3D application. Once the host has been initiated, the driver simulation can join by entering the IP address of the pedestrian simulator and selecting the “connect as client” button on the graphical user interface. When the network connection is established, the pedestrian will see the lead vehicle and the driver-controlled vehicle behind it; the driver will see the pedestrian virtual character waiting to cross at an intersection. Scripts that control the behavior of the player objects reference variables from the Network Manager component to determine whether they are running as a pedestrian simulation or driving simulation. Based on this determination, the scripts enable or disable the appropriate sections of code. For example, the Driver Vive Node will disable the driver camera that follows the driver-controlled vehicle when the simulation is running in pedestrian mode. The driver-

controlled vehicle's movement script will also stop taking input from the driver simulation hardware and stop being controlled by the HLAPI orientation and position updates.

4.3 Latency Tests and Results

Latency is the time delay between the updates of state variables from one site to another on a distributed network. For example, when a driver turns on a turn signal, the state of the turn signal must be communicated and registered on remote simulations. The difference in the time when the turn signal state is registered on the local and remote simulators is the latency. For geographically distributed sites, latency is a fact of life because of the time it takes to communicate from one place to another. Latency is caused by network communication time and whatever additional time it takes to update state.

One of the difficulties in calculating latency is having synchronized clocks at multiple sites to determine send and receive times on a consistent basis. We avoided this by estimating the round-trip time for a call and response between two sites. The timing interval began when site A set a state variable to true. As soon as site B detected the change, it set another state variable to true. When site A detected the change in state made by site B, the time interval ended. Thus, the time interval represented the time to communication from A to B and B to A. The latency is computed as half of this value. Table 4.1 contains roundtrip times for 10 tests of call and response latency. The average roundtrip latency was 200 ms. Thus, the average one-way latency was calculated to be approximately 100 ms. A photo of the environment used by the research team during testing is shown in Figure 4.1.

Table 4.1 – Latency Results

Test Number	Latency Measured (ms)
1	258
2	224
3	170
4	191
5	189
6	213
7	214
8	194
9	172
10	170

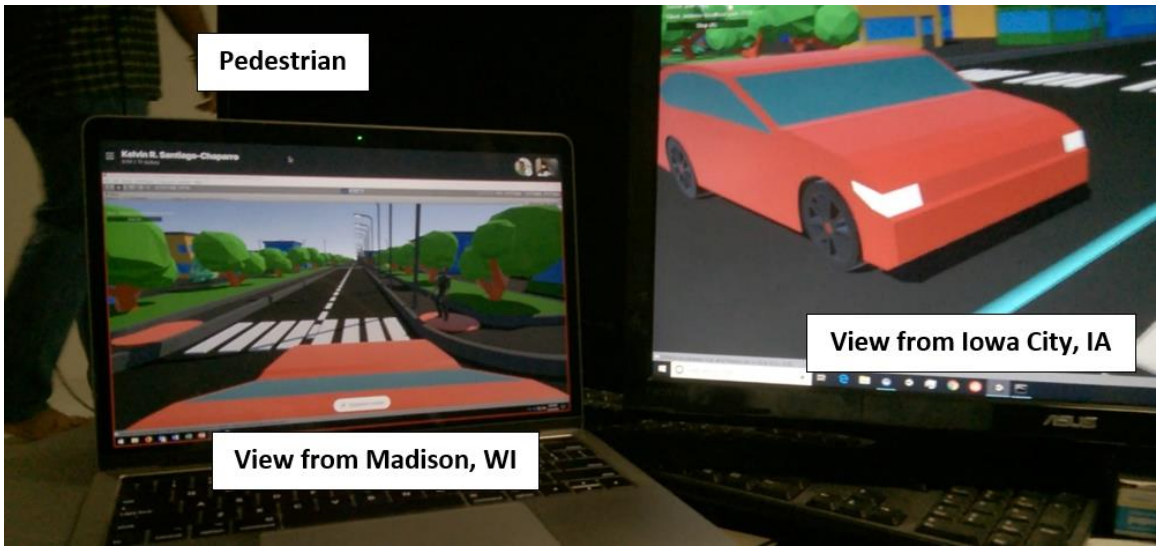

Figure 4.1 – Typical environment setup during latency tests

4.4 Feasibility of Integration with Other Platforms

As suggested throughout the report, one of the key design decisions in the creation of the driving simulator component was modularity. The research team goal was to create a simulation tool that could be expanded and integrated with other platforms to facilitate future research. As such, an input module was created that allows receiving speed, steering, break, and gas pedal values from an external platform.

4.4.1 Development of Generic Input Listener for Driving Control

The input module created works in a similar way to that of the virtual control module created for simplifying the development and testing process and previously described in Section 3.3.2. However, instead of exposing data obtained from user interactions with buttons and other control objects, the data that is exposed to the simulator platform is obtained by listening for packages via UDP. This approach makes it possible for any other control system (that can be configured to export vehicle performance and driver control data over UDP) to act as a control interface for the simulator platform created as part of the research described. The research team tested the feasibility of controlling the simulator platform using the vehicle cabin that is part of an RTI-based driving simulator. A photo taken during the successful test is shown in Figure 4.2.

Figure 4.2 – Integration of Unity-based simulator and RTI vehicle cabin

During the test shown in Figure 4.2, a module was added to the RTI-based simulator to export brake and gas pedal information, along with steering wheel position, via UDP to the IP address of a computer running the Unity-based driving simulator created as

part of the research project described. An input module developed in C# listened for UDP messages and translated the messages received into the corresponding input values for steering, braking, and gas pedal.

5 Conclusions

This report describes a distributed simulation environment that makes it possible to study pedestrian and vehicle real-time interactions using two human participants – one driving a simulated vehicle on a desktop simulator; the other acting as a pedestrian wearing a VR headset. The simulation environment was built using the Unity3D game engine. The environment created makes it possible for the two human participants in an experiment to be geographically distant by relying on the built-in networking functionality provided by the Unity game engine. Latency tests conducted during the project suggest that the built-in functionality of Unity provides sufficiently low latency for conducting human-in-the-loop experiments with the distributed simulation environment created.

The use of the Unity game engine to create the distributed simulation environment makes it possible for researchers to create custom scenarios that are difficult to create using existing simulation platforms. For example, more detailed controls on the behaviors of the simulation are possible due to the use of C# for programming and the availability of a more robust application programming interface compared to those of driving simulation platforms typically used.

5.1 Future Work and Potential Applications

This project demonstrates the feasibility of using a general-purpose, off-the-shelf game engine to build a distributed multi-modal simulator. While this project linked only two simulation sites, the underlying network communications system supports multi-site connections, making it possible to link many different simulators.

The demonstration project focused on showing how a driver and a pedestrian can be placed in the same virtual environment and allowed to interact at a crosswalk. However, the potential of the research approach demonstrated in this project goes beyond understanding how pedestrians interact with vehicles at a crosswalk. Some examples of possible study areas that could be considered as next steps for the procedures demonstrated are listed in the sections ahead.

5.1.1 Understanding of Interactions Between Drivers and Pedestrians

The technology demonstrated can be used to study more complex vehicle-pedestrian interactions that are key to better understanding driver and pedestrian decision making and to reducing crashes. The importance of non-verbal communication between drivers and pedestrians through gestures, head nods, and eye contact can be examined.

In addition, the results can inform microscopic traffic simulations. For example, the decision-making process associated with left turns or right turns at signalized intersections (when a pedestrian is present) can be studied in detail, and the findings can be used to improve the existing agent-based simulation tools used by traffic engineers.

5.1.2 Understanding of Interactions between Multiple Drivers

One of the challenges with existing driving simulation experiments is the use of a single human driver in the environment. Having more than one human driver can provide insights into the collective behavior of drivers in a controlled environment. A detailed understanding of driver interactions can open the door to developing better risk-taking models for use in the traffic safety field.

5.1.3 Understanding of Interactions between Drivers and Autonomous Vehicles

Training the machine learning models that will be key to a successful deployment of autonomous vehicles (and acceptance by the public) will require datasets that accurately describe the interactions between these vehicles and human drivers. These datasets can be easily obtained through on-road tests for normal driving conditions and scenarios. However, from an ethical and safety perspective, obtaining datasets about the interaction of other drivers with these vehicles under unsafe scenarios is difficult. And while simulations can be created based on existing knowledge to create artificial training datasets, having experimental data obtained in a controlled environment can be key to improving the quality of the models used by autonomous vehicles.

It is conceivable that by using technology similar to the one demonstrated here, the "brains" of an autonomous vehicle could be used to control the artificial traffic in the simulation while one or more human drivers are introduced into the mix. Under such a scenario, drivers could be placed in an unsafe scenario and data collected about the interactions between the drivers and the autonomous vehicles. The data obtained could then be used to improve the driver behavior models used by autonomous vehicles, thus resulting in safer interactions with non-autonomous traffic.

5.2 Summary of Student Involvement

This project involved students at both the graduate and undergraduate level. Students were involved in the planning and execution of the research project. The nature of the typical iterative process that characterizes research, and which inherently results in critical thinking because of the trial and error approach, means that students acquired skills that are critical to their professional development.

5.2.1 Workforce Development

In addition to gaining improved research skills, the students had significant exposure to the use of the Unity3D game engine. The Unity3D game engine is a widely used tool across the gaming and visualization industry. Therefore, an indirect result of the project was that students developed skills that can make them competitive in the growing field of game-related technologies.

5.3 Technology Transfer

As a part of the project requirement, the team will pursue dissemination of the underlying research work via webinars and other presentations. A webinar will allow the authors to discuss the approach used to develop the simulation environment in a dynamic environment in which questions can be discussed with participants. The webinar will be recorded and made available for future use, thus providing an important complement for this report.

References

1. Wilcox, P.A., Burger, A.G., & Hoare, P. (2000). Advanced distributed simulation: a review of developments and their implication for data collection and analysis. *Simulation Practice and Theory, 8*(3-4): 201-231.
2. Hancock, P.A., & de Ridder, S.N. (2003). Behavioural accident avoidance science: understanding response in collision incipient conditions. *Ergonomics, 46*(12): 1111-1135.
3. Lehsing, Karcke, & Bengler (2015)
4. Maag, C., Muhlbacher, D., Mark, C., & Kruger, H.P. (2012). Studying effects of advanced driver assistance systems (ADAS) on individual and group level using multi-driver simulation. *IEEE Intelligent Transportation Systems Magazine, 4*(3), pp.45-54.
5. Sawyer, B.D., & Hancock, P.A. (2012). Development of a linked simulation network to evaluate intelligent transportation system vehicle to vehicle solutions. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 56*(1): 2316-2320.