Off Road Autonomous Vehicle Modeling and Repeatability Using Real World Telemetry via Simulation

Matthew P. Spencer[a] and Jeremy P. Bos[b]

[a]Keweenaw Research Center, 23620 Airpark Blvd, Calumet, MI, United States
[b]Michigan Technological University, 1400 Townsend Drive, Houghton, MI, United States

ABSTRACT

One approach to autonomous control of high mobility ground vehicle platforms operating on challenging terrain is with the use of predictive simulation. Using a simulated or virtual world, an autonomous system can optimize use of its control systems by predicting interaction between the vehicle and ground as well as the vehicle actuator state. Such a simulation allows the platform to assess multiple possible scenarios before attempting to execute a path. Physically realistic simulations covering all of these domains are currently computationally expensive, and are unable to provide fast execution times when assessing each individual scenario due to the use of high simulation frequencies ($>$ 1000Hz). This work evaluates using an Unreal Engine 4 vehicle model and virtual environment, leveraging its underlying PhysX library to build a simple unmanned vehicle platform. The simulation is demonstrated to successfully run at low simulation frequencies down to a lower threshold of 190Hz with minimal average cross-track-error and heading angle deviation when performing multiple real off road driving maneuvers. Real vehicle telemetry was used as input to drive the unmanned vehicle's integrated Pure Pursuit and PID autonomous driving control algorithms within the simulation and used as ground truth for comparison.

Keywords: autonomous, UE4, off road, simulation, real telemetry, low frequency

1. INTRODUCTION

This section details the problem and motivation for this paper along with a brief analysis of its results. A quick discussion of current work being performed in the field of autonomous off road vehicle simulations is presented, and past tools and systems are explained.

1.1 Overview

As the age of commercial and private autonomous vehicles looms over the horizon, many hours of research have been poured into a myriad of aspects relating to the autonomous vehicle (AV) industry. Governments and corporations over the past few decades have put significant effort into improving the capability of AV artificial intelligence, cameras, and sensor systems to reliably guide AVs through challenging terrain (e.g. mud, fields, dirt roads, etc.) regarding unstructured off road environments.[1,2] AVs operating in these environments cannot rely on pre-existing road signs, speed regulations, and detailed maps to successfully traverse to their destinations.

The applications that could utilize off road AVs are diverse, adding a substantial amount of variability to an already complicated task of structured on-road AV environments which use pre-existing landmarks, signs, and information while in operation.[3] Current methods involving path planning and object detection being used to achieve successful off road AV traversal are quickly changing and improving,[4] establishing the need for faster and cheaper ways of testing the implementation of newer methods. Computer vision algorithms used in Simultaneous Localization and Mapping (SLAM) techniques implementing modern camera and LiDAR technologies are being thoroughly explored to test AV operations in real time.[5] Simulations being utilized for these purposes are growing within the fields of digital image processing and artificial neural networks.[6] The incorporation of large data sets

Further author information:
M.P.S. : E-mail: mpspence@mtu.edu, Telephone: 906 231 7965

generated either within or outside of these simulations for the purpose of real world evaluation is becoming a specific key area of research and investment.[7]

Many well known 3D computer simulation tools and game engines such as ANVEL, Car-Sim, Unreal Engine, Gazebo, Unity, etc.[8] have been assessed to provide data precise enough to model AVs and test various autonomous scenarios. Most simulation systems are layered, requiring a federated approach incorporating multiple unique simulations specializing in different types of physics-based models running in parallel with visual modeling to achieve desirable results.[9] Federated systems coupled with high simulation frequencies leave legacy computer simulation architecture to require heavy computing power.[10] This generates long computational wait times between executing each step of the simulation, voiding their use within time-dependent real time applications. With the recent advancements in modern computers and technology, researchers have been pushing against these limitations by utilizing popular open-source software libraries to achieve realistic virtual worlds and simulation environments using a single simulation platform such as UE4.[11]

One way to speed up the execution time of a single simulation platform would be to decrease its simulation frequency. The set simulation frequency is also known as the simulation's frames-per-second (FPS).[12] After the engine computes the needed underlying physics and graphical calculations for the next time step of the simulation, the next image (frame) of the simulation is outputted. This process is called the "rendering pipeline".[13] The less calculations the simulation has to do (lowering the simulation frequency), the faster it performs in real time. The more calculations the simulation has to do (raising the simulation frequency), the slower the simulation will perform in real time. Higher simulation frequencies result in smoother and more accurate simulated results, which is why high simulation frequencies are chosen for modern AV simulation solutions. Lowering the simulation frequency is desirable for more time-pressed applications such as path planning though, where a greater number of paths need to be calculated preferably faster than real time.

Real world trace path telemetry for each maneuver is generated by capturing synchronized GPS and IMU trace data sampled at 400 Hz from a real world High Mobility Multipurpose Wheeled Vehicle (HMMWV) platform. The HMMWV platform was used due to the accuracy of the supplied virtual model measurements compared to a real HMMWV provided by the Keweenaw Research Center (KRC). A human driver performs each test maneuver within the environment that the virtual world is modeled to replicate. The UE4 game engine is programmed using C++ to control a virtual HMMWV vehicle model, recreating each driving maneuver within the simulated environment using the trace telemetry as input. The programming language C++ is used as a result of UE4 being natively built to interface with multiple C-based graphic API libraries such as OpenGL[14] and Vulkan[15] for better performance.[16] A modeled AV driver uses an integrated Pure Pursuit and PID controller to steer the vehicle along each path, moving along the specified coordinates at the desired target velocity.[17]

The UE4 simulation telemetry is recorded at a sampling rate corresponding to the set simulation frequency during each execution. After each simulated driving test, the recorded simulation telemetry is then compared to the original trace path telemetry. Total cross-track-error (CTE) and heading angle deviation for each maneuver is then averaged and plotted over the tested set of simulation frequencies. Comparisons between the average CTE and heading angle error against the real telemetry is done to see if lower simulation frequencies (< 1000Hz) can be deemed reliable, meeting a CTE and heading error as close to 0 as possible. Simulation step frequency is lowered for each test until a point of failure is observed and the current driving maneuver cannot be performed by the modeled HMMWV AV.

In the following sections, procedures for importing a real world environment into UE4 using colored geo-referenced point cloud data is explained. The KRC's outdoor vehicle test course was used to model the real-to-virtual UE4 world environment due to the high resolution geo-located point cloud data they provided*. The HMMWV model and its UE4 vehicle dynamic properties is then described along with the virtual AV driver using the UE4 Pure Pursuit and PID control C++ implementations.

1.2 Problem Statement

By integrating common AV velocity and steering control algorithms within a single visual and physics computer simulation platform, a low simulation frequency needs to be found that demonstrates reliable CTE and heading

*The KRC supplied the HMMWV model, geo-located 3D point cloud data, and the real HMMWV path telemetry used to drive the modeled AV for this paper.

angle error performance between the recorded path telemetry of a real vehicle platform and its virtual counterpart for effective path planning in real time. In order to understand the spatial and temporal resolution regarding modeled off road AV control within a single simulation platform using real world telemetry, this work evaluates multiple off road vehicle maneuvers performed by a real HMMWV platform[18] within the simulation. This paper then demonstrates a low overall average CTE (< 6.87cm) and overall average heading angle error (< 4.89 degrees) of a single simulation platform running at low simulation frequencies (< 1000Hz) when modeling multiple real AV off road path following maneuvers.

Modeled AV path planning simulations that provide reliable CTE results exhibit evidence that the AV system can reach its target destination when following a selected path without accident.[19] CTE has commonly been used as a benchmark for path planning performance[20] and as an active input for path following control algorithms (e.g. Stanley method).[21] A common method for operating path finding systems is to quickly analyze multiple possible routes prior to path execution in real time within a replicated simulated world of an AV's environment.[8] By analyzing the CTE and AV's heading angle error between the simulation and real world vehicle, comparisons can be made between the virtual and real world models to observe the simulation's reliability at lowered simulation frequencies.

A recent build of Unreal Engine 4 (UE4) game engine with its underlying PhysX architecture[22] is used in this paper as the integrated visual and physics computer system for the AV simulation model. Telemetry gathered from a real vehicle platform is used as input to drive the virtual AV controllers within a modeled real world environment. The replicated outdoor virtual world allows performance comparisons between the simulation's path following accuracy to that of a vehicle platform in the real world. After all tested paths are executed within the simulation, average CTE and heading error is discussed and compared to the real world path telemetry. The data is quantiled and a two-term Gaussian fit-line applied. CTE Gaussian analysis is used commonly when measuring performance of AV algorithm control,[23][24] where an overall CTE close to 0 is desired to indicate that the AV stayed near the target input path. A low heading error also indicates stabilization between the simulation environment physics, e.g. gravity, and the AV's simulated controllers and vehicle dynamics.[25]

The end results demonstrate that future path following AV systems relying on a single simulation platform like UE4 can execute potential routes for real AV platforms running at lower simulation frequencies, resulting in low overall average CTE and heading error. Frequencies < 190Hz were observed to have exponentially growing average CTE and heading error, further evidenced by an exponential rise in the slope of the Gaussian best-fit-line assigned to each error data set. Simulation failure also became more common when executing at higher speeds during more complex off road vehicle maneuvers. Setting the simulation frequency > 190Hz, approaching the original input telemetry sampling frequency of 400Hz, produced minimal average CTE and heading error by comparison. Higher simulation frequencies resulted in simulations that were able to follow the supplied input path with greater accuracy compared to simulations set < 190Hz.

2. BACKGROUND

This section discusses background information on building blocks used to set up the foundation of the following computer simulation path following experiment. Previous literature is discussed along with differences between this research and aforementioned literature. The simulation software and framework is explained in context with off road autonomous vehicle research, along with the justification of using a Pure Pursuit controller for the virtual AV model.

2.1 Unreal Engine 4

UE4 is currently used as a visual simulation tool within the public and private AV research and development community[26].[27] There still remain questions about the robustness of its deterministic performance regarding the underlying physics implementation, such as object hit detection or computation lag,[7] but its popular use among video game developers provides a highly populated online community which regularly gives feedback and documentation alongside steady UE4 engine updates.[22] The source code is also made publicly available by the developers, allowing programmers with a good understanding of C++ and the necessary skills to develop engine specific tools and plugins to achieve specific simulation related goals.[16] The underlying physics framework,

PhysX, is also well known and documented,[28] offsetting some of the prior concerns when compared to other simulation tools such as "MuJo".[9]

Much work has been done experimenting in creating simulation scenarios of real world events created specifically for the UE4 game engine. Environments can be hand-made and created to represent believable scenarios to study various autonomous technologies involving neural network training,[5] pedestrian traffic avoidance[29],[30] specific LiDAR SLAM techniques,[6] and many more. The need and interest is therefore outgrowing the pace at which AV path planning and safety systems are being developed.

A recent UE4 build (v4.26) with the default UE4 mid-sized template vehicle model has been shown to autonomously drive within a virtual environment using the MUONS path planning stack developed at Michigan Technological University. It was determined that UE4 coupled with MUONS handled mid-sized vehicles with complex suspension systems well when compared to using a simpler HUSKY robot platform.[31] MUONS can also virtually incorporate the robot operating system (ROS) libraries in order to publish and subscribe to virtual sensors required to path plan adequately within complex unstructured environments, further indicating UE4's ability to be modified as needed. The environment created for the latter MUONS experiment was generated using a portion of real world point cloud data taken from the KRC's outdoor test course.[32] This emphasizes the value of the KRC course for correlating further real world to virtual world testing path planning experiments.

For past UE4 experiments mentioned in this section, all telemetry for path planning control and repeatable algorithm testing was created virtually inside of the simulation. Paths generated for an AV platform were executed virtually within the simulation, with no guarantee that the path could be executed on a real AV vehicle. This solely virtual limitation does not allow correlation between the virtual AV's predictive reliability and the real world vehicle platform it is trying to simulate. This paper implements this additional piece of real world information to make a realistic evaluation on repeatability when modeling potential AV scenarios using UE4 and PhysX.

2.2 Previous Simulation Work

Computer simulations to visually model AVs and their attempts at solving problems associated with real time path following date back to the early 1990s, when the United States Navy funded research into potential solutions for simulating autonomous underwater vehicles (AUVs).[33] It was shown that in order to achieve a complete 3D visual representation of the simulation, many computers had to be networked together to provide a successful AUV simulation. Although the complex simulation architecture is crude by today's standards, an elegant solution to finding a reliable simulation to model fast and comprehensive real time AV path planning scenarios is still being sought after today.[7]

Simulations are not only being developed in regards to AUVs and grounded AVs, but are also being developed within the field of aerial drone autonomy. In 2018, a drone flight controller was modeled and integrated within a virtually controlled drone, which was tested in a published study utilizing a UE4 C++ plugin called AirSim.[10] The UE4 simulation was fed prerecorded virtual world coordinate telemetry for path following, showing proof-of-concept for potential real world applications. The simulation frequency of the UE4 engine when sampling the drone path data was set to be 1000 Hz, providing waypoint path data 1ms apart from each point for playback within the AirSim controller. Lower threshold values regarding the simulation frequency rate were not considered, nor any attempt at testing the system on a real aerial AV platform outside of the simulation. These questions are brought to the forefront for this paper which uses real telemetry, and analyzes lower UE4 simulation frequencies within the context of AV simulation.

A real world example of a modeled AV UE4 simulation interacting with a real AV has been demonstrated to be a challenging endeavour as of late 2021.[34] A simple two-way road was modeled within UE4 representing a real two-way AV test site. The experiment used two virtual vehicles placed on the simulated road facing the same direction as each other. One of the simulated vehicles was controlled by a real AV operating on the actual test course, and the other simulated vehicle was controlled by a real user equipped with a driving wheel controller connected to the simulation. The purpose of this experiment was to test the real AV's crash avoidance system by having the human controlled virtual vehicle cut into the AV's lane. This was done in order to find a solution to prevent the risks involved when testing an AV safety system in real life. The experiment proved that this

virtual-to-real AV interaction was feasible, as the real AV was shown to be able to respond to the virtual vehicle and its simulated environment. However, the response from the real AV to avoid the simulated vehicle was found not reliable or accurate enough for use in testing actual AV safety systems.

What is not addressed by these experiments is the real computation time needed when running each simulation at a set simulation frequency, and how the simulation frequency impacts the simulation's overall performance. AV models have been created within UE4 and similar platforms,[35] virtual data has been programmed to be used as an input source during execution,[10] and real world AVs have been shown to be able to react to virtual input data,[34] but to draw further conclusions regarding repeatability of simulated models within the real world, a simulation would also need to use real world data outside of the virtual environment. This paper sets up the experiment and process to use UE4 alongside real world data to analyze AV simulation performance at different simulation frequencies.

2.3 Pure Pursuit

Pure Pursuit is an established autonomous steering control method, wherein the correct steering angle needed for the vehicle to move towards the next look-ahead waypoint is calculated.[17] The curvature between the current vehicle position and waypoint position is found, thus setting the vehicle's needed steering angle on this found curvature.[36] The calculated steering angle is set to achieve a more "human-like" smooth response compared to more simplistic alternatives such as the Carrot Chasing control algorithm,[37] which continually snaps the steering angle to face the next waypoint at a fixed look-ahead distance.

Pure Pursuit's ease in implementation while retaining the "human-like" driving characteristics was preferred over other more complicated path following algorithms such as the Stanley method, which involves calculating CTE and desired yaw rate to feed into the controller during operation for lateral stability.[38]

More modern methods to incorporate lateral and velocity input control have been integrated into the underlying Pure Pursuit algorithm for use on real world vehicles to find optimal look-ahead values. This is done by smoothing out turns by averaging angles needed to turn the vehicle to reach its waypoint destination.[39] The Pure Pursuit implementation in this paper relies on the additional lateral control combined with engine specific UE4 functions, wherein the look-ahead vector is shortened or lengthened depending on the upcoming steering angle needed to turn the vehicle. The look-ahead vector is shortened when approaching a heavy turn for example, and lengthened if following an estimated straight path.

3. SIMULATION SETUP

This section explains the general overview in setting up the UE4 simulation experiment. The terrain data importation method, Pure Pursuit algorithm implementation, and the process for controlling the virtual AV using real telemetry is explained. Each simulated driving test maneuver is also visualized, plotting the paths taken by the real HMMWV during each maneuver.

3.1 Terrain Importation

Specialized land surveying equipment was utilized by the KRC to create a geo-located point cloud LiDAR data set of the KRC's outdoor test course. Drone equipment was also used along with geo-located aerial images of the test course. The point cloud and texture data was then combined and transformed into a single .fbx file. This process was done internally at the KRC and provided for use within this paper. The .fbx file was then imported into the mesh editing software Blender.[40]

Each point in the original point cloud is zeroed using its minimum coordinate to place the origin of the transformed mesh at the (0,0) location. This zeroing process is done to ensure that the terrain is compatible with UE4's world coordinate system, orienting the origin point of the terrain at the simulated world's default virtual origin point, (0,0).

A Python script utilizing Blender's API is then executed on the mesh, scaling the terrain mesh and dividing the mesh into a user selected number of corresponding heightmap and texture tiles. Each tile produced has its heightmap file paired with the overlaid texture image of the mesh. This scaling was performed to use UE4's native landscape terrain type instead of a single point cloud mesh, trading overall mesh vertex resolution for

simulation performance. The tiles were loaded into UE4 using a custom proprietary C++ plug-in tool "KRC Terrain Creator" built using the UE4 editor API landscape creation functionality and internal engine function library seen in Fig. 1 (left). The resulting fully generated terrain landscape shown within the UE4 editor window is displayed in Fig. 1 (right). UE4 uses the metric system as its default measuring system for distance and length,[41] where 1 "unreal unit" equates to 1cm. Depending on the tile's location denoted by its (X,Y) filename, it is spawned and placed in its proper location within the virtual world.

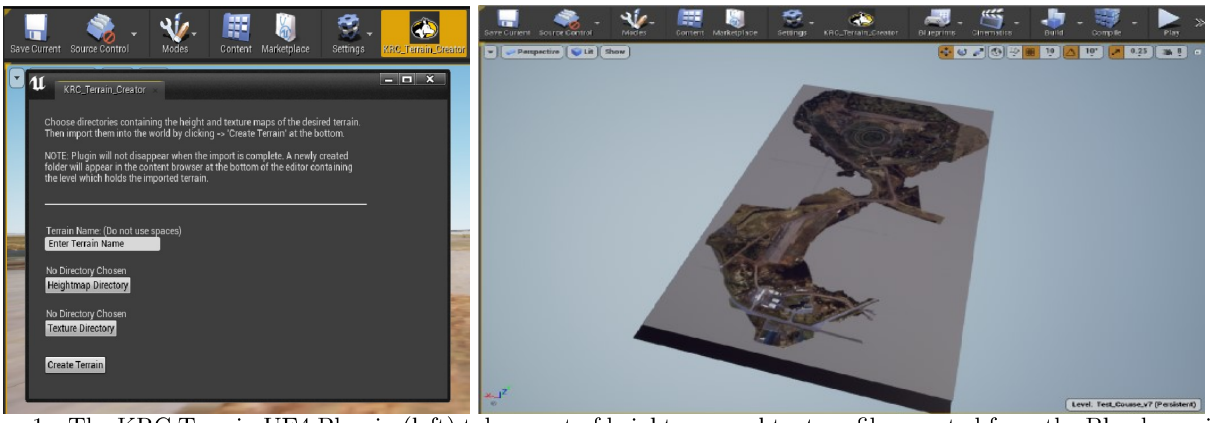

Figure 1. The KRC Terrain UE4 Plug-in (left) takes a set of heightmap and texture files created from the Blender script's geo-located LiDAR point cloud and fully generates the KRC terrain within a UE4 level as a landscape (right).

3.2 Vehicle Model

A HMMWV vehicle model supplied by the KRC was imported into UE4 seen in Fig. 2 and configured based on the default advanced vehicle template supplied by the engine. UE4 vehicle parameters were tuned alongside default UE4 vehicle PhysX model values[42] to correlate with real HMMWV properties[†].

Figure 2. A virtual HMMWV model imported into UE4 and assigned the default PhysX vehicle template. Basic modifications (such as tire width, weight, etc.) were made to the underlying PhysX model.

3.3 Real World Telemetry

A real HMMWV was equipped with GPS and IMU sensors to record a driver performing multiple different maneuvers at different speeds within an off-road environment. The data was recorded at rate of 400Hz, time synchronized across the necessary GPS and IMU devices. The resulting output data is processed into a MATLAB .mat file format, which was then converted into a CSV file for use as input for the UE4 simulation. 400Hz was chosen due to the need for additional simultaneous testing for an unrelated HMMWV's on-board controller needing a 400Hz signal refresh rate.

The KRC test course point cloud data was created in the NAD83 Northing and Easting coordinate system, while the GPS data for the HMMWV was captured in the WGS84 latitude and longitude coordinate system. Thus a conversion was made on each coordinate, utilizing the "PROJ" C++ library[43] and an up-to-date packaged

[†]Any HMMVW-style vehicle model using the default UE4 PhysX vehicle template can be used to recreate the experiment presented in this paper.

transform database, to convert WGS84 coordinates into required NAD83 coordinates for use as waypoints within the simulation. To match up the total number of path coordinates with the virtual world's origin point at (0,0), each path coordinate is re-zeroed and scaled with the same process used when importing the terrain.

Velocity of the vehicle model in UE4 is controlled by a hand-tuned C++ PID controller, which adjusts the acceleration and braking of the virtual vehicle until the recorded velocity is reached, as indicated by the real world trace telemetry at the given time step. This creates a virtual "cruise control" that adjusts the speed as needed. Default UE4 PhysX brake and engine torque is used to apply acceleration and brake pressure to meet the target velocity.

At each time step, the Pure Pursuit controller sets the navigated vehicle's steering angle to point towards the next destination waypoint as indicated by the input GPS waypoint telemetry running at the current simulation frequency. Missing the destination waypoint will result in the vehicle attempting to return to the trace path, causing the controller to sharply turn the wheel in an attempt to steer and reorient the vehicle back towards the missed waypoint. The resulting action causes the vehicle to endlessly spin around in circles when traveling at high velocities. This action is considered to be a point of failure for the simulation, and therefore the simulation is ended and noted as "Did-not-Finish" (DNF).

3.4 Simulation Tests

This section will outline the three sets of tests and their respective paths fixed at a simulation frequency, which is referred to as the simulation's frames-per-second (FPS) by UE4. The set frequencies for each test start at 60Hz and increase by 10Hz for each test until 400Hz is reached.

Fig. 3 (left) displays a straight path taken by Test 1 (*St5*) within the first set of calibration tests 1, 2, and 3. These tests are used to calibrate the PID controller for proper behavior, as well as make sure the Pure Pursuit controller's flexible look-ahead distance works as expected. The HMMWV travels in a straight line for *St5* at a low speed of 5mph. Fig. 3 (right) displays Tests 2 (*Left10*) and 3 (*Left15*), where a sharp left hand turn is done at the end of *St5* moving at slightly higher speeds of 10 and 15mph.

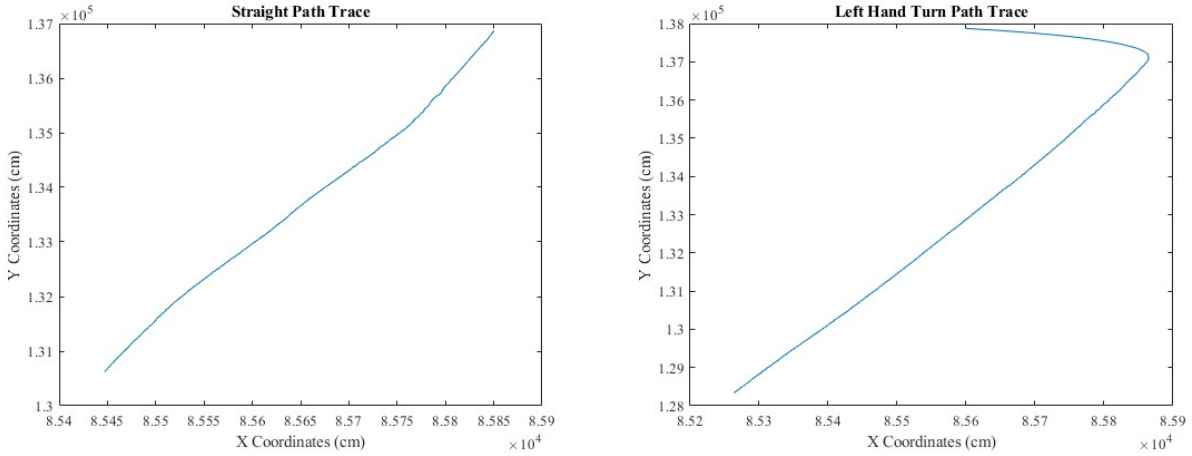

Figure 3. Straight path GPS trace (left) and left hand turn GPS trace (right) within UE4.

Fig. 4 (left) displays the path taken by the secondary set of tests 4 (*SS20*), 5 (*SS25*), and 6 (*SS30*). The HMMWV takes a wide sweeping left-hand turn called a Step Steer (SS) going at higher speeds of 20, 25, and 30mph. These tests demonstrate the simulation being able to execute a controlled sweeping turn of the vehicle 180 degrees while moving at higher velocities in an off road environment. Fig. 4 (right) displays the last maneuver taken by the final set of tests 7 (*DLC30*), 8 (*DLC35*), and 9 (*DLC40*). The HMMWV performs a Double Lane Change (DLC) moving the vehicle at higher speeds of 30, 35, and 40mph. These three tests demonstrate the ability for the simulation to execute a more difficult steering path taken at higher velocities within off road environments.

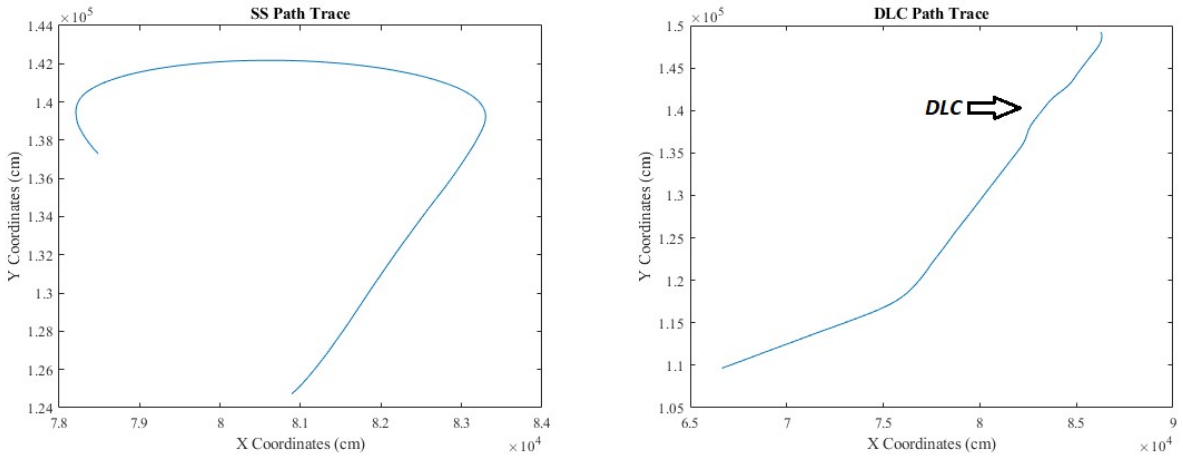

Figure 4. Step Steer path GPS trace (left) and Double Lane Change GPS trace (right) within UE4.

4. RESULTS

This section details the CTE and heading error results after simulating the three sets of maneuvers described previously in Section 3.4. Velocity error was not taken into consideration, as additional vehicle dynamic tuning aside from the default brake torque and engine RPM configuration supplied by UE4 was out of scope for this paper. Each test maneuver and resulting data set has a corresponding table and MATLAB generated line plots.

4.1 Overall Results

Tab. 1 in the Appendix displays the quantiles for the overall average CTE ($\overline{CTE}$) and Heading error ($\overline{Head.Err.}$) for all tested simulation frequencies during each maneuver seen in Fig. 5. The frequency before the first $\overline{CTE}$ and $\overline{Head.Err.}$ value above Q3 was used as the threshold frequency, which was both 190Hz. A two-term Gaussian curve was chosen as the best fit line for analysis. Both the $\overline{CTE}$ and $\overline{Head.Err.}$ is seen to grow linearly until an exponential growth point happens in its slope below the chosen 190Hz lower threshold. Frequencies below 110Hz were not taken into consideration due to the *DLC40* simulation failing below 110Hz.

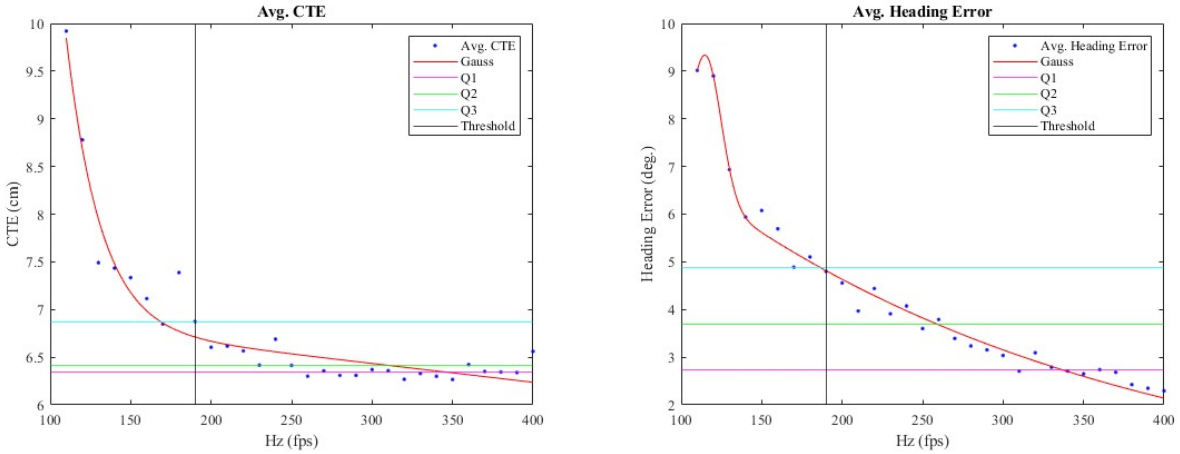

Figure 5. Total averaged cross-track-error (left) and heading angle error (right) for each tested maneuver over the specified frequency from 60Hz to 400Hz.

4.2 Calibration Tests

Fig. 6 plot the $\overline{CTE}$ and $\overline{Head.Err.}$ of each test on the same graphs for analysis. Looking at Fig. 6 (left), a low $\overline{CTE}$ beneath Q1 for test speeds of 5, 10, and 15mph is demonstrated. A slight exponential growth can be seen

developing along with a higher overall $\overline{CTE}$ within Test *Left15* moving at the highest speed of 15mph compared to the low $\overline{CTE}$ of 2-3cm from tests *St5* and *Left10*.

A larger difference is viewed between the straight run $\overline{Head.Err.}$ of Test *St5* and the two tests with a sharp left hand turn seen in Fig. 6 (right). A more pronounced exponential increase is viewed in test *Left10* and *Left15* running at 10 and 15mph as the frequency is pushed lower.

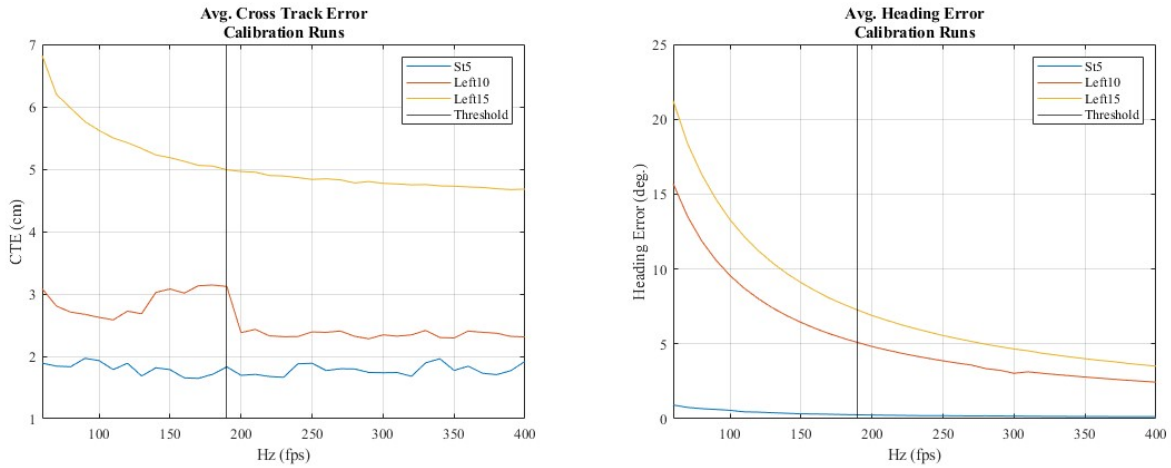

Figure 6. Average cross-track-error (left) and heading angle error (right) for each calibration test *St5*, *Left10* and *Left15* maneuver over the specified frequency from 60Hz to 400Hz.

4.3 SS Tests

Fig. 7 plots the $\overline{CTE}$ and $\overline{Head.Err.}$ of each test on the same graphs for analysis.A higher $\overline{CTE}$ is observed again in Fig. 7 (left) when the vehicle moves at higher velocities, this time when performing a more gradual SS turn. Test *SS30* moving at 30mph shows a sharp increase in $\overline{CTE}$ when simulating at less than 100Hz. The sharp increase follows the simulation failing at 70Hz for tests *SS25* and *SS30*, and at 60Hz for Test *SS20*. These DNFs are caused by the vehicle missing the next waypoint.

The simulation AV model shows greater deviation from the trace path at lower frequencies, which can again be seen by the greater $\overline{Head.Err.}$ data shown in Fig. 7 (right) for tests *SS20*, *SS25*, and *SS30*. As the frequency of the simulation is lowered, the vehicle starts to deviate the original trace path. This is similar to the increase in $\overline{Head.Err.}$ seen at higher speeds with low simulation frequency in Tests *Left10* and *Left15*.

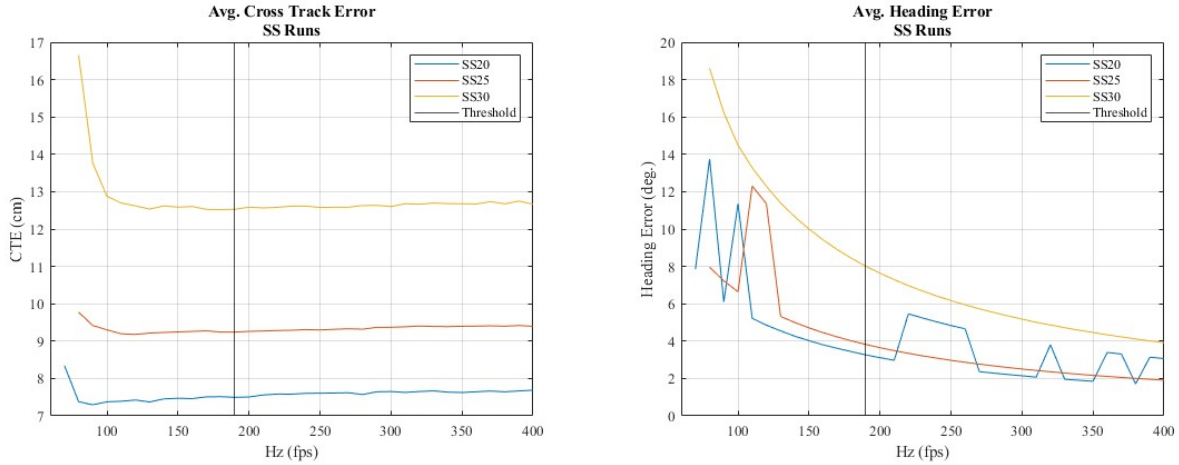

Figure 7. Average cross-track-error (left) and heading angle error (right) for each step steer test *SS20*, *SS25* and *SS30* maneuver over the specified frequency from 60Hz to 400Hz.

This deviation from the trace path is further evidenced by comparing the $\overline{CTE}$ during Test *SS30* running at 400Hz and 80Hz seen in Fig. 8. The 80Hz $\overline{CTE}$ shows the vehicle straying farther from the input path during the SS turn between 15 and 30 seconds compared to the more controlled 400Hz $\overline{CTE}$ during execution.

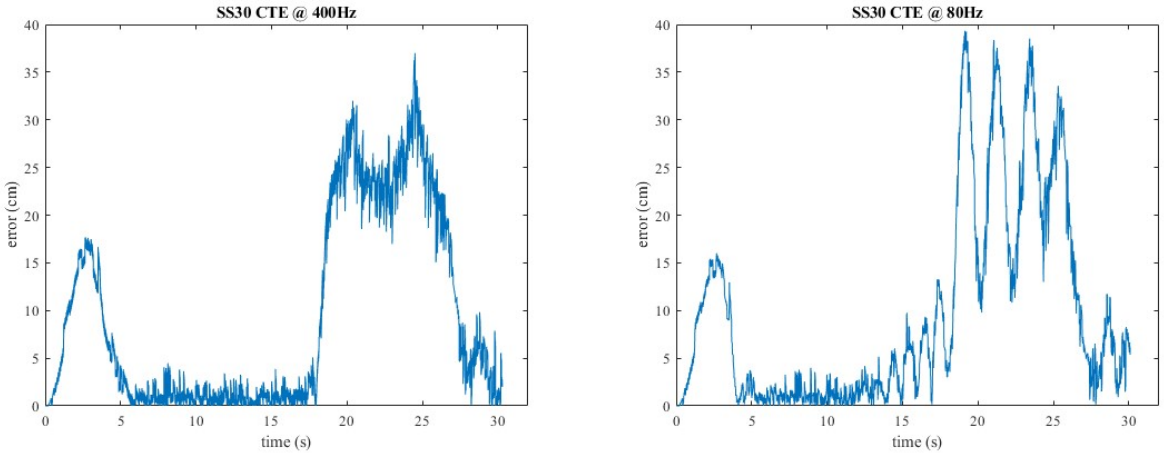

Figure 8. Cross-track-error comparison over the duration of the *SS30* test running at a simulation frequency of 400Hz (left) and 80Hz (right).

4.4 DLC Tests

Fig. 9 plots the $\overline{CTE}$ and $\overline{Head.Err.}$ of each test on the same graphs for analysis. Test *DLC40*, as with the previous tests performing their selected maneuvers at the highest speed, shows a greater $\overline{CTE}$ for each frequency displayed in Fig. 9 (left). Due to the more complex DLC maneuver and the higher speeds of each test the simulation fails at all frequencies lower than 100 Hz, whereas the Calibration and SS Tests successfully complete simulations at frequencies lower than 100 Hz. Test *DLC40* is shown to fail even sooner at 110 Hz. Even though Test *DLC30* appears to show a higher $\overline{CTE}$ at 100Hz, this is due to Test *DLC40* failing at that frequency. An additional cross-track-error comparison over the duration of *DLC40* similar to Fig. 8 is shown in the Appendix.

The simulation AV model deviates greater when traveling at higher speeds and performing a more complex driving maneuver, which is further evidenced observing the Test *DLC30* $\overline{Head.Err.}$ in Fig. 9 (right). Tests *DLC30* and *DLC35* show a steady increase in $\overline{Head.Err.}$ while descending to lower frequencies, while Test *DLC40* is greater and less predictable deviating from the trace path at the highest speed.

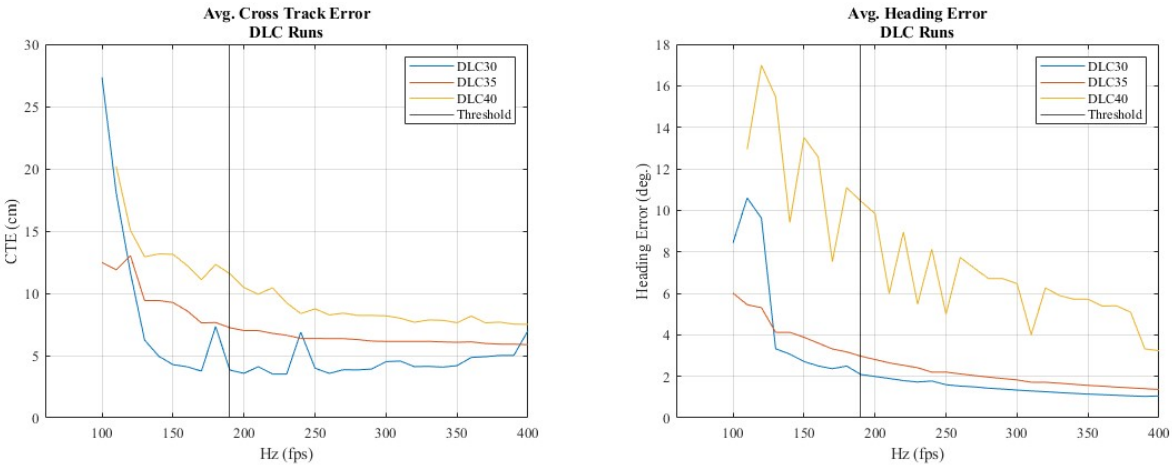

Figure 9. Average cross-track-error (left) and heading angle error (right) for each double lane change test *DLC30*, *DLC35* and *DLC40* maneuver over the specified frequency from 60Hz to 400Hz.

5. DISCUSSION

This chapter sets to draw conclusions from observations of the $\overline{CTE}$ and $\overline{Head.Err.}$ data from each set of off road simulation tests. A determination is made about whether or not a baseline low frequency can be successfully demonstrated when repeating and modeling real world off road AV scenarios. Additional simulation improvements outlined in this experiment are discussed, along with future research moving forward.

5.1 Conclusions

This paper has successfully demonstrated that a low simulation frequency (<1000Hz) can be used for repeatable $\overline{CTE}$ and $\overline{Head.Err.}$ performance when analyzing the recorded path telemetry of a real vehicle platform and its virtual counterpart. Integrated C++ Pure Pursuit and PID control algorithms running at the set simulation frequency were used to drive a HMMWV military vehicle modeled within UE4 (a single simulation platform) through multiple different off road driving maneuvers performed by a real driver at varying low to original input velocities within a virtual representation of the KRC test course.

A trade off between the set simulation frequency and the virtual AV's deviation from the real input trace path telemetry can be seen when observing each set of tests. The $\overline{CTE}$ and $\overline{Head.Err.}$ for each test remained lowest when running at or near the original input trace path sampled frequency, and slowly increased linearly until hitting a point in which the simulation started deviating from the supplied trace path exponentially. This was seen in the $\overline{CTE}$ for *SS30* running at 400Hz viewed in Fig. 8 (left) compared to running at 80Hz seen in Fig. 8 (right). The relationship between $\overline{CTE}$ and maximum AV speed was positive, as maximum speed increased $\overline{CTE}$ increased. This was also true for the relationship between $\overline{Head.Err.}$ and AV speed.

Based on the overall average results for both $\overline{CTE}$ and average $\overline{Head.Err.}$, a lower boundary simulation frequency of 190Hz is recommended for path following when modeling and repeating the stated straight, left, SS, and DLC real world off road driving maneuvers up to the tested maximum speeds. Frequencies lower than 190Hz introduce noticeable path deviation, increasing the risk of simulation failure especially when the driving maneuver was performed at higher velocities. The greater deviation is also shown in the exponential increase found in the Gaussian fit line's slope when applied to the overall average results for both the $\overline{CTE}$ and $\overline{Head.Err.}$ below 190Hz viewed in Fig. 5.

5.2 Future Work

More complex autonomous control methods besides Pure Pursuit could be implemented and improved upon to control the vehicle model using the created UE4 simulation environment. An example of a more complex controller is the Stanley control method, which includes lateral acceleration, yaw rate, and current CTE in its steering angle calculations. A similar experiment would be performed for each additional control method at the same lower frequencies presented in this paper to see if greater stability can be achieved below 190Hz.

Once a path planning control method is chosen, an open-loop control system within the virtual world theoretically could be constructed between the simulation and a real world AV platform to test the viability of future AV control using the simulated environment. Data compared between the simulated path and actual path executed on the real AV could be analyzed to see if a low $\overline{CTE}$ holds at a simulation frequency of 190Hz. Any instability caused by introducing systems outside of an ideal virtually simulated environment could also be observed (e.g., communication over a wired or wireless network medium such as Ethernet or 802.11p).

A basic default set of virtual off road vehicle dynamics and characteristics (e.g. suspension system, brake torques, lateral slip coefficients, etc.) were chosen to achieve path following. Characteristics of the vehicle, such as the slip and yaw rate, still need to be researched and evaluated within the simulation using UE4 and PhysX. Outside of the default physical interactions between the PhysX vehicle and UE4 landscape environment using the integrated Pure Pursuit and PID controllers, definite approximations of what is possible between real and virtual world vehicle dynamics is still unknown.

Even though the research in this paper has demonstrated that low simulation frequency AV modeling and repeatability using real world telemetry is possible, more research is needed before any definitive statements can be made regarding the full scope of testing real time control or path planning AV systems using a single visual and physics simulation platform like UE4.

APPENDIX A. ADDITIONAL FIGURES AND TABLES

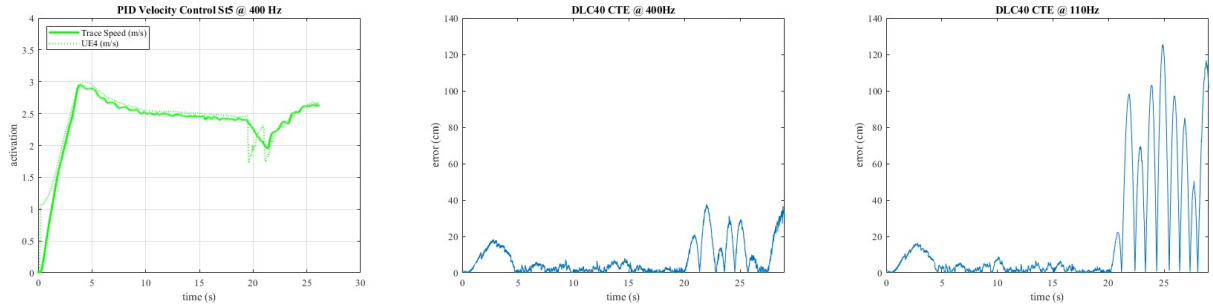

Figure 10. PID calibration control test for St5 running at 400Hz (left), followed by a cross-track-error comparison over the duration of the *DLC40* test running at 400Hz (center) and 110Hz (right).

Table 1. Overall Average Error Quantiles

Quantile	$\overline{CTE}(cm)$	$\overline{Head.Err.}(deg.)$
Q1	6.3389	2.7372
Q2	6.4189	3.6950
Q3	6.8727	4.8862

ACKNOWLEDGMENTS

I thank Jesus Christ, family, friends, and my Grandma Spencer whose love and support makes work like this possible. I would like to specifically acknowledge Derek Chopp of the Keweenaw Research Center for building the Blender script used in this experiment to initially parse the point cloud terrain data. I would also like to acknowledge all the students and full-time employees at the Keweenaw Research Center for their work in data acquisition regarding the terrain point cloud, texture map, and HMMWV telemetry. A large amount of gratitude is also owed to the university and professors who continue to impart on me the required wisdom and knowledge to do research work and further contribute to the field of electrical and computer engineering.

REFERENCES

[1] A. Singh, D. H., "Modeling off-road rollover using terramechanics for real time driving simulator," in [*2014 NDIA Ground Vehicle Systems Engineering and Technology Symposium*], (2014).

[2] Aquilio, A., [*A Framework for Dynamic Terrain with Application in Off-road Ground Vehicle Simulations*], Georgia State University, Atlanta, GA (February 2006).

[3] Brummelen, J., Gruyer, D., and Najjaran, H., "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies* **89**, 384–406 (2018).

[4] Yuan, X., Huang, G., and Shi, K., "Improved adaptive path following control system for autonomous vehicle in different velocities," in [*IEEE Transactions on Intelligent Transportation Systems*], **21**(8), 3247–3256 (2020).

[5] Pollok, T., L, J., Ruf, B., and Schumann, A., "Unrealgt: Using unreal engine to generate ground truth datasets," in [*Advances in Visual Computing*], *Lecture Notes in Computer Science* **11844**, 670–682 (2019).

[6] Espineira, J. P., Robinson, J., Groenewald, J., Chan, P. H., and Donzella, V., "Realistic lidar with noise model for real-time testing of automated vehicles in a virtual environment," in [*IEEE Sensors Journal*], **24**(8), 9919–9926 (2021).

[7] Chance, G., Ghobrial, A., McAreavey, K., Lemaignan, S., and T. Pip, K. E., [*On Determinism of Game Engines used for Simulation-based Autonomous Vehicle Verification*], Cornell University, Ithaca, NY (April 2021).

[8] Fields, M., Brewer, R., Edge, H., Pusey, J., Weller, E., and et al., "simulation tools for robotics research and assessment," in [*Medical Imaging: Image Processing*], *Proc. SPIE* **9837** (2016).

[9] Erez, T., Tassa, Y., and Todorov, E., "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode, and physx," in [*2015 IEEE International Conference on Robotics and Automation*], *IEEE ICRA*, 4397–4404 (2015).

[10] Shah, S., Dey, D., Lovett, C., and Kapoor, A., "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in [*Field and Service Robotics*], Hutter, M. and Siegwart, R., eds., 621–635, Springer International Publishing, Cham (2018).

[11] Games, E., "Unreal Engine." Unreal Engine, 2021, https://www.unrealengine.com/en-US/. (Accessed: 04 Novemeber 2021).

[12] Swierad, O., "Measuring Performance." Unreal Art Optimization, 2019, https://unrealartoptimization.github.io/book/process/measuring-performance/. (Accessed: 03 March 2022).

[13] Messaoudi, F., Ksentini, A., and G. Simon, P. B., "Performance analysis of game engines on mobile and fixed devices," *ACM Transactions on Multimedia Computing, Communications, and Applications* **13**(4), 1–28 (2017).

[14] Group, K., "OpenGl Headline News." Online, 2021, https://www.opengl.org/. (Accessed: 20 March 2022).

[15] Overvoorde, A., "Vulkan Tutorial." Vulkan, 2020, https://www.mathworks.com/help/nav/ref/controllerpurepursuit-system-object.html. (Accessed: 20 March 2022).

[16] Games, E., "Programming in C++." Unreal Engine v4.27, 2022, https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/. (Accessed: 20 March 2022).

[17] Coulter, R., [*Implementation of the Pure Pursuit Path Tracking Algorithm*], Carnegie Mellon University, Pittsburgh, PA (January 1992).

[18] Military.com, "High Mobility Multi-Purpose Wheeled Vehicle (HMMWV)." 2021, https://www.unrealengine.com/en-US/. (Accessed: 04 Novemeber 2021).

[19] Chen, Y., Chen, S., Zhang, T., Zhang, S., and Zheng, N., "Autonomous vehicle testing and validation platform: Integrated simulation system with hardware in the loop*," in [*2018 IEEE Intelligent Vehicles Symposium (IV)*], *IEEE IV*, 949–956 (2018).

[20] Mashadi, B. and Majidi, M., "Global optimal path planning of an autonomous vehicle for overtaking a moving obstacle," in [*Latin American Journal of Solids and Structures*], **11** (2014).

[21] Hoffmann, G. M., Tomlin, C. J., Montemerlo, M., and Thrun, S., "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in [*2007 American Control Conference*], 2296–2301 (2007).

[22] Games, E., "Unreal Engine Forms." UE4v4.27, https://forums.unrealengine.com/. (Accessed: 04 Novemeber 2021).

[23] Rathinam, S. and et al., "Autonomous searching and tracking of a river using an uav," in [*2007 American Control Conference*], 359–364 (2007).

[24] Kamran, D., Zhu, J., and Lauer, M., "Learning path tracking for real car-like mobile robots from simulation," in [*2019 European Conference on Mobile Robots (ECMR)*], *ECMR*, 1–6 (2019).

[25] Hu, C., Wang, R., Yan, F., and Chen, N., "Should the desired heading in path following of autonomous vehicles be the tangent direction of the desired path?," in [*IEEE Transactions on Intelligent Transportation Systems*], **16**(6), 3084–3094 (2015).

[26] D. Talwar, S. Guruswamy, N. R. and Eirinaki, M., "Evaluating validity of synthetic data in perception tasks for autonomous vehicles," in [*2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*], *IEEE AITest*, 73–80 (2020).

[27] Loze, S., "Autonomous vehicle development and training meets user experience at VI-grade." Epic Games, Novemeber 11, 2021, https://www.unrealengine.com/en-US/spotlights/autonomous-vehicle-development-and-training-meets-user-experience-at-vi-grade. (Accessed: 22 February 2022).

[28] NVIDIA, "NVIDIA PhysX SDK 4.1 Documentation." April 18, 2021, https://gameworksdocs.nvidia.com/PhysX/4.1/documentation/physxguide/Index.html. (Accessed: 04 Novemeber 2021).

[29] Tran, T. T. M., Parker, C., and Tomitsch, M., "A review of virtual reality studies on autonomous vehicle–pedestrian interaction," in [*IEEE Transactions on Human-Machine Systems*], **51**(6), 641–652 (2021).

[30] Schmitt, P. and et al., [*nuReality: A VR environment for research of pedestrian and autonomous vehicle interactions*], Cornell University, Ithaca, NY (January 2022).

[31] Jeffries, Z., Majhor, C., Carter, J., Kysar, S., and Bos, J. P., "Muons path planning performance for a vehicle with complex suspension in unreal," in [*Autonomous systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2021*], *Proc. SPIE* **11748** (2021).

[32] Young, P., Kysar, S., and Bos, J., "Unreal as a simulation environment for off-road autonomy," in [*Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2020*], *Proc. SPIE* **11415** (2020).

[33] Brutzman, D. P., Kanayama, Y., and Zyda, M. J., "Integrated simulation for rapid development of autonomous underwater vehicles," in [*Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology*], 3–10 (1992).

[34] Che, X., Li, C., and Zhang, Z., "A test method for self-driving vehicle based on mixed reality," in [*2021 IEEE International Conference on Smart Internet of Things (SmartIoT)*], *IEEE SmartIoT*, 401–405 (2021).

[35] Peirt, A., Karpman, E., and et. al., "Modelling of off-road wheeled vehicles for real-time dynamic simulation," *Journal of Terramechanics* **97**, 45–58 (2021).

[36] Snider, J., [*Automatic Steering Methods for Autonomous Automobile Tracking*], Carnegie Mellon University, Pittsburgh, PA (February 2009).

[37] Niu, H., Lu, Y., Savvaris, A., and Tsourdos, A., "Efficient path following algorithm for unmanned surface vehicle," in [*OCEANS 2016 - Shanghai*], 1–7 (2016).

[38] Lu, Z., Shyrokau, B., Boulkroune, B., van Aalst, S., and Happee, R., "Performance benchmark of state-of-the-art lateral path-following controllers," in [*2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*], *IEEE AMC*, 541–546 (2018).

[39] Wang, S., Fu, S., Li, B., and Wang, S., "Path tracking control of tracked paver based on improved pure pursuit algorithm," in [*2021 40th Chinese Control Conference (CCC)*], *IEEE CCC*, 4187–4192 (2021).

[40] Foundation, T. B., "Blender." https://www.blender.org/. (Accessed: 04 Novemeber 2021).

[41] Mowerin, N., "Scale and Measurement Inside Unreal Engine 4." techarthub, 2022, https://www.techarthub.com/scale-and-measurement-inside-unreal-engine-4/. (Accessed: 16 February 2022).

[42] NVIDIA, "User's Guide, Vehicles." PhysX SDKv3.4.0 Documentation, 2022, https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/Vehicles.html. (Accessed: 18 February 2022).

[43] Project, O. G., "PROJ." https://proj.org/index.html. (Accessed: 04 Novemeber 2021).