2021 NDIA GROUND VEHICLE SYSTEM ENGINEERING AND TECHNOLOGY SYMPOSIUM
SESSION NAME TECHNICAL SESSION
AUGUST 10-12, 2021 - NOVI, MICHIGAN

AUTONOMOUS VEHICLE ASSESSMENT FRAMEWORK

Mads R. Jeppesen[1], **Sigurd A. Madsen**[1], **Ole Balling, PhD**[1]

[1]Department of Mechanical and Production Engineering, Aarhus University, Denmark

ABSTRACT

Development and assessment of autonomous vehicle capability are relying on simulation software for time and cost efficiency. The value of such simulations are significantly dependent on minimizing the gap from simulation to real environment performance of systems. The simulations for off-road autonomous vehicle assessment are in particular challenging due to the complex nature of natural terrains and their virtual representations, vehicle-terrain interactions during soft soil maneuvering, and the integration of sensors and their output in virtual generated terrains. This paper presents the early development of a software tool aimed at simulating custom autonomous off-road scenarios generated from their real world counterparts. The effort is an important step in generating confidence in simulation based testing of autonomous systems as a forerunner for purely virtual generated scenarios for autonomous systems evaluation.

Citation: M.R. Jeppesen, S.A. Madsen, O. Balling, "Autonomous Vehicle Assessment Framework", In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 10-12, 2021.

1. INTRODUCTION

Development and assessment of autonomous vehicle capability are relying on simulation software for time and cost efficiency. The value of such simulations are significantly dependent on minimizing the gap from simulation to real environment performance of systems. The simulations for off-road autonomous vehicle assessment are in particular challenging due to the complex nature of natural terrains and their virtual representations, vehicle-terrain interactions during soft soil maneuvering, and the integration of sensors and their output in virtual generated terrains. This paper presents the early development of a software tool aimed at simulating custom autonomous off-road scenarios generated from their real world counterparts. The effort is an important step in generating confidence in simulation based testing of autonomous systems as a forerunner for purely virtual generated scenarios for autonomous systems evaluation.

The work presented here is the initial generation

of a virtual test-bed for autonomous vehicle system performance evaluation. Metrics for the evaluation of such systems can be implemented as pass/fail of obstacle recognition, collision avoidance, vehicle limit control, continous measure of deviation from expected vehicle path, velocity, and time to destination etc. While such specific metrics for the evaluation are not demonstrated in full, the work defines an example of a simulation environment and framework where a subset of such metrics are implemented. The objective of the work is autonomous stack performance evaluation on specific virtual implementations of vehicles, moving agents, sensors, terrain, obstacles and other environmental representations and behaviors. The Unity 3D game engine was chosen as the implementation platform due to its wide spread use for natural terrain generation as well as some built-in physics capability and fast collision detection, all governed by the demand for realtime performance. The specific vehicle simulation model and the environment are developed utilizing built in as well as external assets for advanced visualization capabilities and C# integration for creation of custom functionalities in the form of dynamic and sensor models.

Development of the virtual simulation framework is based on a specific use-case from the NATO Advanced Vehicle Technology Panel (AVT) Research Task Group 341 scenario team. The use-case consists of an autonomous off-road mission, a setup to demonstrate the capabilities of autonomous vehicle systems for convoy protection, reconnaissance and obstacle avoidance. The mission is set to take place at the Michigan Technological University's Keweenaw Research Center (KRC), where the digital information regarding the environment and the vehicle originate. The Fuel Efficiency Demonstrator (FED) Alpha is used as a reference vehicle for development of the vehicle model, with validation data supplied by the AVT-308 NATO Cooperative Demonstration of Technology (CDT) in the form of several individual vehicle dynamics and soft soil interaction test cases. The CDT demonstration was the culmination of the three year preceding work on Next Generation - NATO Reference Mobility Model [item 1]

2. VIRTUAL ENVIRONMENT

The modeling process of a virtual environment, based on a real-world location, is split up into a set of segmented layers based on the PEGASUS project definition as illustrated in Figure Figure 1, [item 2].

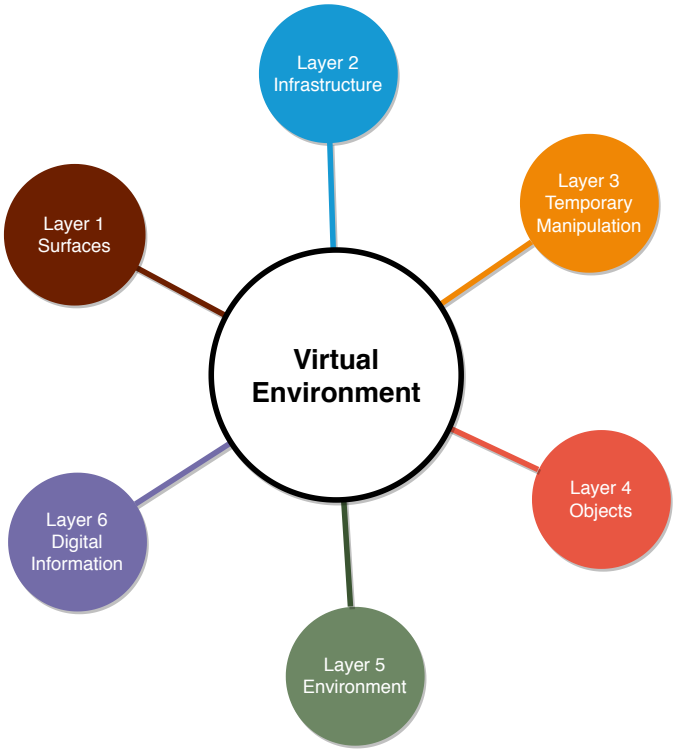
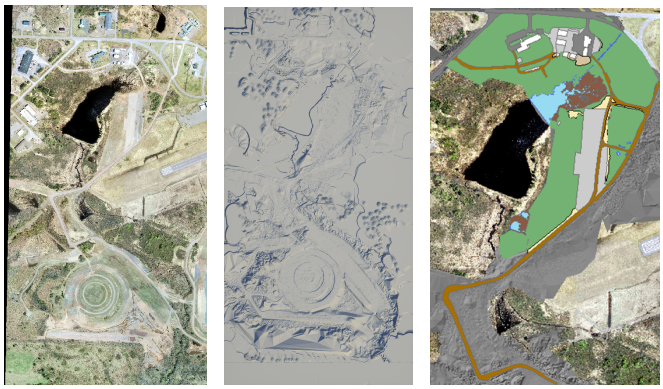


Figure 1: Virtual environment layer segmentation, [item 2].

The terrain surfaces are described by geo-referenced aerial images (GeoTIFFs), terrain height maps in the form of Triangular Irregular Network (TIN) based on LiDAR scans and geo-referenced shape files containing additional surface information, all of which can be loaded into

Unity through any type of Geospatial Information System (GIS) software (Figure Figure 2).



(a) GeoTIFF (b) TIN (c) Shapefiles

Figure 2: KRC Terrain surface data.

Boundaries and no-go regions from the data set are considered infrastructure in the segmentation and can be; either physical or digital restrictions. The Unity environment can be modified and manipulated, acting as temporary manipulation of the environment. Change in topography or surface quality (e.g. rain) which can lead to reduced friction, as an example. Objects, buildings, and vegetation are placed based on shape files (Figure Figure 3), which describe their position and type. Shape files can also be used to specify surface types and qualities. Smaller vegetation (grass, bushes, etc.) are purely visual and does not act as physical obstacles for the vehicle. Larger vegetation (trees, rocks, etc.) and other objects (light posts, buildings, vehicles, etc.) all act as both visual and physical obstacles in the environment.

Figure 3: Object shapefiles

Figure Figure 4 shows a part of the modeled Keweenaw Research Center as an example of the populated and textured environment.

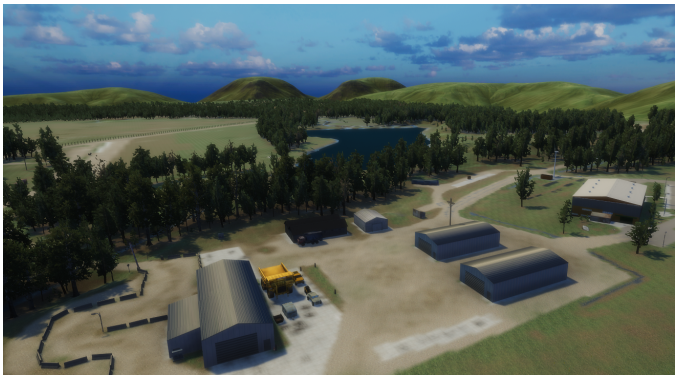


Figure 4: Populated and textured environment.

The climate and ambiance can be manipulated in many ways; implemented methods include fog, vegetation health (drought), time of day, year and location (sun position), and lighting (Figure Figure 5).

(a) Day (b) Night

Figure 5: Day and night manipulation.

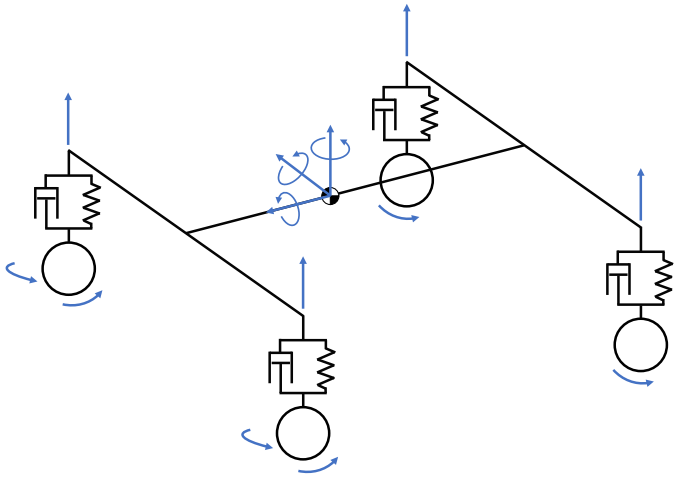


Figure 6: Dynamic Model DOF's illustration.

3. VEHICLE MODEL

As the purpose of the simulation framework is to assess autonomous systems and not vehicle dynamics, the primary concern is not achieving full dynamic fidelity, however, a certain degree of accuracy is necessary, especially for higher-level autonomy. The focus of the vehicle model is real-time performance and adaptability. The possibility of creating a vehicle based on an engine, transmission, mass distribution, wheel configuration, and suspension allows for a simple vehicle setup.

The model is based on a primary body and four wheels, resulting in 15 total degrees of freedom (DoF). 6 DoF's for the main body, 2 DoF's for steering of the front wheels, 4 DoF's for each individual wheel and 4 for the suspension of each wheel (Figure 6). A slip friction model is used to calculate the traction forces and a simple power train model for the applied wheel torque [7, 10]. The vehicle model handles three inputs: Throttle, Steering, and Brake. The two components of the vehicle model then calculate the output torque and return the forces exerted onto the vehicle by the wheels. Tyre interaction with the terrain is handled by a modified Unity wheel collider (Figure 7), containing both the slip-friction model and a linear suspension model.

The FED Alpha is used as a reference vehicle with the supplied data to demonstrate, verify, and validate the dynamic model. Several unit tests have been conducted within the simulation per the US Army TOP documentation [Bracamonte2017] used by the CDT. The tests include; Wall-to-Wall Turning Diameter, Acceleration, Coast Down, Gear Shift, Braking, Constant Cornering, Half-Round, Vertical Step, and V-Ditch.

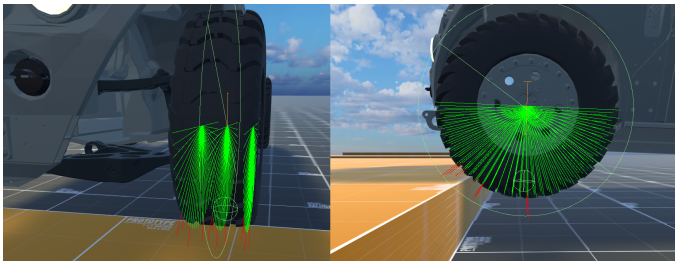


Figure 7: 3-dimensional tire model.

The simple 15-DOF model, using slip-friction based traction, allows for high performance while maintaining the main vehicle dynamic aspects. The FED Alpha reference vehicle, used as the primary use-case, was modeled using the base vehicle model, with good simulation results, compared to the reference data with some deviations, primarily in

the high fidelity tests, such as measured vertical acceleration in half-round tests.

3.1 Soft Soil Integration

Tools have been developed for the simulation to support limited regions with deformable terrain (Soft Soil) while maintaining real-time simulation capabilities (Figure Figure 8). The regions consist of a continuous fine mesh and use a tire stencil and a basic Bekker's Pressure Sinkage relation [Wong2010] to calculate deformations. Equilibrium between the terrain deformation and the vertical wheel forces is at each time step is calculated based on the pressure sinkage relation in [item 5]:

$$p = (k_c/b + k_\phi)\, z^n \tag{1}$$

At each time step, a fraction of the required deformation for equilibrium is applied to the terrain, based on a damping factor c_z. For simplicity, the terrain deformation is considered plastic, leading to no springback of the soil. To account for the burrowing effect of the tire rut during slip, an additional slip-sinkage scaling (s_s) is included. This scaling follows a linear approximation, where 100% slip increases the static sinkage by a specified factor (k_s) as defined in [item 9].

The change in deformation needed for equilibrium at a given iteration (i) is calculated from an inverse Bekker pressure and the previous iteration's deformation.

$$\Delta z_i = ((z_{Bekker} \cdot s_s) - z_{i-1}) \cdot \frac{\Delta t}{\Delta t \cdot c_z} \tag{2}$$

In soft soil regions the slip-friction models are still used for calculation of the traction, as methods for determining the traction in soft soils was found incompatible with real time performance. This is recognized as an area of needed improvement of this framework for improved realtime tractive related mobility performance.

The soft soil regions include memory, such that it can be deformed, saved, and used for a later simulation of another vehicle. Likewise, multiple vehicles can do multiple passes on the same soil, having their driving characteristics affected by the passage of a prior tire.

Figure 8: Deformed terrain.

4. AUTONOMY COMPATIBILITY

The primary purpose of the framework is to assist the development and assessment of autonomous system mobility performance. To facilitate this, three areas need to be addressed. The first part is creating tools and/or sensors that allow an autonomy stack to observe the digital representation of the environment. These sensors must produce outputs similar to that of a physical sensor to be compatible with the autonomous algorithms. The second component is to allow communication between the simulation and an externally developed autonomy stack in terms of sensor output and steering input. The final element is developing autonomous algorithms native to the simulation software to act as simple independent agents and as part of an autonomy stack when testing smaller subsystems.

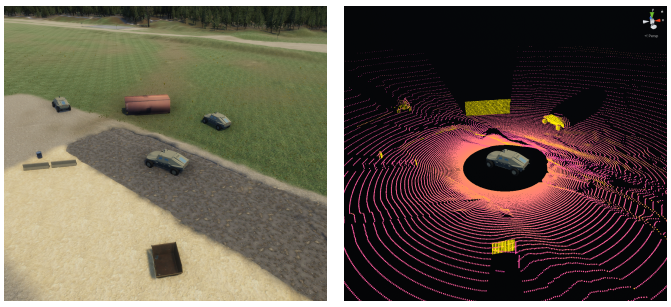
4.1 Sensors

Ideal sensor models are implemented in the software allowing the autonomy stack a perception of the virtual environment. Through the sensors, it is possible to locate objects, measure distances, and otherwise perceive the environment. LiDAR,

camera, and position sensors are implemented. These are some of the primary sensors within vehicle autonomy.

A virtual camera is used in the simulations as a native component in Unity, giving an autonomous algorithm a visual representation of the environment. The Unity camera can be directly set up using physical parameters such as resolution, Field of View (FoV), skewness, aperture, exposure, lens distortion, focal distance, etc. A custom depth camera model combines depth data with the visual representation to yield an enhanced data output.

The LiDAR is a primary sensor used in many autonomous systems, which allows for a 3-dimensional representation of an area by distance measurements. LiDAR sensors often consist of a set of vertically spaced rays mounted in a rotating housing [item 8]. This allows for 360-degree representation of the surrounding environment in the form of a point cloud.



(a) Virtual environment (b) LiDAR output

Figure 9: LiDAR data visualization.

The LiDAR sensor is modeled in Unity using raycasts [item 12], which, similarly to the physical sensor, casts a ray from a position in a direction for a given length and returns information based on collisions. For an object to interact with the LiDAR raycast, and thereby detectable, it must be assigned a collider, which specifies the boundaries at which collisions occur. The collider and raycast interaction can be modified to filter unwanted data (terrain, leader vehicle, etc.).

4.2 ROS Integration

The Robotic Operating System (ROS) framework [item 14] is used as the primary communication to and from the simulation. The framework is an open source system containing several tools and libraries for both robot behavior and standardized communication. Any autonomy stack supporting ROS-based communication can navigate and control the simulated vehicle using one of several developed drivers (Way-point-, Twist-, Throttle/Steer/Brake-based).

The ROS# library [item 13] allows unity running on Windows to publish and subscribe to ROS topics through a Rosbridge WebSocket [item 11]. Setting up a combination of custom publishers and subscribers allows for any given combination of sensors for the required autonomy stack inputs.

4.3 Virtual Agents

Autonomous systems comes in many types concerning complete system performance, specific skill levels for certain functionality and widely different application purposes. Examples of skills are localization, object detection, global and local route planning, and stability control, several of which function as subsystems in a full autonomy stack. Some scenarios require different independent agents (leader-follower, dynamic obstacles, etc.). For the developed framework to assist the development and assessment of autonomous driving systems, a level of built-in autonomy is required to fill out the remaining autonomy stack and act as independent agents. Several developed autonomous driving algorithms can work as parts of an incomplete autonomy stack. The developed way-point based autonomous driver and an adjustable obstacle avoidance allow for creating predetermined paths for various driving agents that act based on predetermined events. Manual control of a vehicle (or more) can similarly act as independent agents.

5. DEMONSTRATIONS

The following demonstrations will highlight the strengths of the simulation software as part of the development and test of autonomous algorithms and tools for evaluation. The demonstrations utilize the ROS communication capabilities and incorporate two different autonomous algorithms, first individually and then combined. The final demonstration is done using the simulation software's autonomy and driving a mission inspired by proposals made by the NATO AVT-341 task group [item 4] for the KRC terrain.

5.1 Navigation Stack

A local route-planner was tested in the simulation framework in collaboration with parallel work at Aarhus University on the development of a ROS based system for autonomous driving of an unmanned small-scale tracked vehicle, [item 15]. This work implemented a driver model for a tracked vehicle based on navigation stack implemented in ROS. This system uses a point cloud and vehicle odometry to build a local and global map of the environment. Based on a user-input desired global end position, the navigation stack publishes a twist message moving the vehicle toward the position. Using updated odometry message (and point cloud), the navigation stack recalculate twist outputs and the maps continuously. An in-depth overview of the navigation stack can be found in [item 15].

The ROS Node structure of this demonstration contains three parts. The simulation software running on a Windows PC, the ROS WebSocket running on a Virtual Machine on the same PC, and finally, the ROS navigation stack running on an Nvidia Jetson, which the navigation stack has been developed on. The two systems are connected using a switch. An overview of the structure can be seen in figure Figure 10.

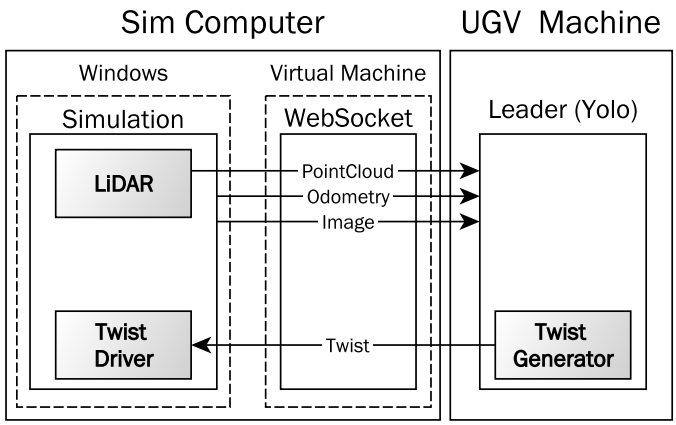


Figure 10: ROS Navigation Stack demonstration network overview.

The simulation software publishes the LiDAR point cloud along with a camera feed and odometry data. The vehicle in the simulation is controlled by the twist message published by the navigation stack.

The subscribed topics on the navigation stack, as well as the generated map, can be shown using ROS rviz, which is a $3D$ visualization tool for ROS. An example from parts of the demonstration can be seen in figure Figure 11. A $2D$ navigation goal (way-point), can be selected inside rviz, the autonomy stack calculates a global (and local) route towards the target and starts publishing twist commands.

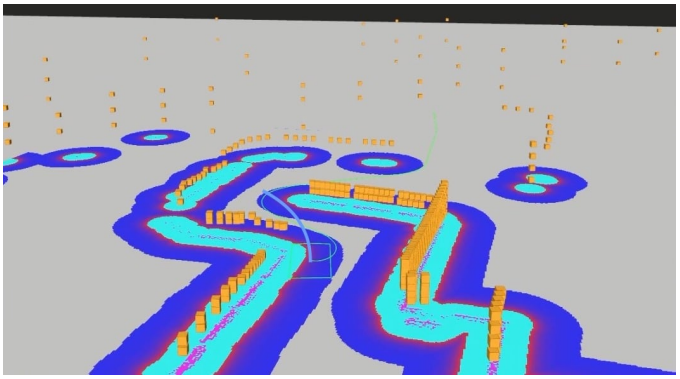


Figure 11: ROS rviz - Navigation Stack example.

The entire purpose of the simulation framework comes to light in this demonstration, as the original

driver model, developed by [item 15] was made for a tracked vehicle and as such has significantly different control characteristics. The new driver of the navigation stack was tuned and modified using the simulated data. Driving the virtual vehicle and observing the output and responses from the autonomy stack based on the sensor and vehicle data gives quick access to a highly customizable environment for both testing and development. The simulation framework has two tasks in this demonstration. First, simulating the vehicle behavior based on the twist commands published by the navigation stack. This is achieved using a twist driver. The second task is publishing the vehicle state and sensor observations back to the autonomy stack. An example from the virtual environment can be seen in figure Figure 12, which is the same view as in figure Figure 11.

Figure 12: Simulation Environment - Navigation Stack example.

The computational requirements of a high-density point cloud can significantly halter performance, as well as increase the required data transfer rates. The computational capabilities of the Nvidia Jetson are impressive for its size, though limited when it comes to the large data packages. To overcome the computational power of the Nvidia Jetson, as well as increasing the speed of the simulation, the point cloud resolution of the LiDAR data during testing was kept to a limited size. Connecting the autonomy stack to the simulation showed the possibilities and advantages of using the virtual environment for testing and development of the autonomy stack. Instead of relying on a physical vehicle and environment, the simulation software gave access to a highly customizable environment for both testing and development. During the collaboration, the simulation framework helped discover aspects of the autonomy stack which functioned unintentionally and in that way furthered the development. High velocities as well as larger and more complex environments made it possible to push the functionalities and capabilities of the autonomy stack even further. A video from the simulation are located at: `www.youtube.com/watch?v=f5mlVAAPj9U`

5.2 Leader-Follower Stack Implementation

This demonstration is generated in collaboration with another parallel effort developing a stereo camera based tracker system [item 16]. This work developed a ROS based tracker using computer vision and object detection on a depth augmented stereo camera feed. The algorithm uses location of the detected object and segmentation of the point cloud to determine the 3D position of the object. Once the object has been localized a position is published relative to the detected object. The use case of this is a leader-follower scenario, where the algorithm will detect a leader vehicle from the follower's camera feed, and publish way-points for the follower to keep up with the leader. An in-depth overview of the detection algorithm can be found in [item 16].

The node structure of this demonstration contains two parts. The simulation software running on a Windows PC and the ROS based computer vision algorithm running externally. This time, instead of running the Rosbridge WebSocket on a virtual machine, the WebSocket runs on the same system running the computer vision algorithm, which is installed on a native Linux system. The computer

vision algorithm subscribes to three topics; an image along with a PointCloud2 from a depth camera and the vehicle odometry. The two systems are connected using a switch. An overview of the structure can be seen in figure Figure 13.

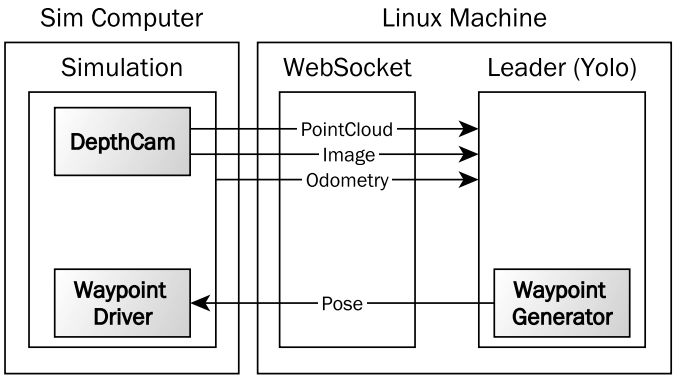


Figure 13: ROS Twist network overview.

Two versions of the system were run. The first was performed with the full density point cloud with a resolution of $672x376$. The purpose of the test was to verify and validate the performance of the computer vision algorithm and its ability to detect and correctly extract the position of the leader. The computer vision algorithm requires the full horizontal and vertical resolution to function, so publishing a less dense point cloud was not possible. As with the previous test, the large data packages contained in the point cloud message proved even more problematic with the dense point cloud. The point cloud generated from the depth camera, along with the camera feed and odometry can be seen illustrated in ROS rviz in figure Figure 14. The output from the algorithm is a way-point published as a pose. The position of the way-point is based on the desired relative offset from the detected object, in this case the leader.

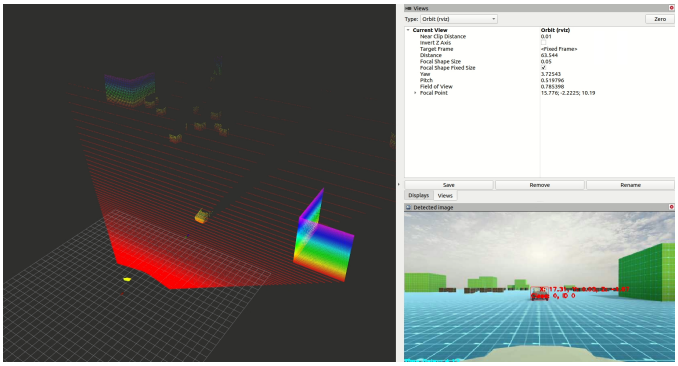


Figure 14: ROS rviz - Test scene, full density point cloud.

For the second setup, only the pose of the leader was published, along with the camera feed and a sparse depth camera point cloud. That way surpassing the large data package necessary for the full density point cloud. The computer vision algorithm will still only publish a new way-point when and if the algorithm has detected the leader in the camera feed. The published data from the second test can be seen illustrated in ROS rviz in figure Figure 15, with the traversed path marked in red.

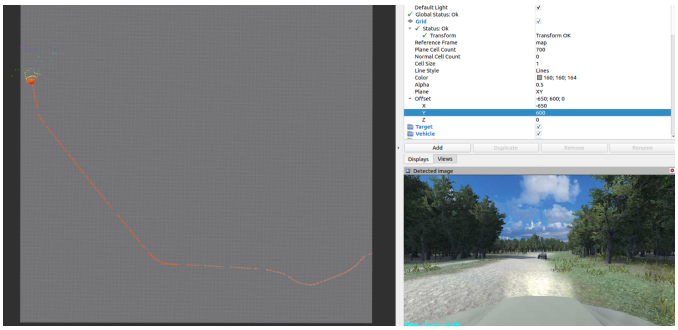


Figure 15: ROS rviz - KRC scene, sparse point cloud, traversed path marked in red.

The simulation framework has three tasks in this demonstration. First, the leader vehicle will follow a predetermined set of way-points using a developed way-point driver based on three parts, velocity control, steering control, and waypoint approval. The

second task is publishing the follower odometry data and camera feed along with the depth measurements or leader pose. The final task is updating the follower's way-point by subscribing to the computer vision algorithm's published way-points. The target velocity of the follower will be set equal to the leader's current velocity, as the leader-follower algorithm does not publish a target velocity. The follower will then use the same way-point driver that the leader is using, however with a non-zero value for the distance proportional gain to allow for the follower to catch up with the leader. An image from the tests with the sparse point cloud and leader pose publish can be seen in figure Figure 16.

Figure 16: Simulation Environment - KRC scene leader-follower test.

Similar to the previous demonstration the utilities of the simulation framework allow for creating a virtual test environment in which the parameters of an autonomy stack can be tested and tuned. However in this case the graphical fidelity of the virtual environment is of high importance, as the computer vision algorithm is based on the image feed generated in the virtual environment. For the algorithm to be able to reliably detect the leading vehicle, the representation of the environment and the leader vehicle must be sufficiently represented in relation to the resolutions of the actual cameras used and the models of these. A validation study in model representation of sensors and virtual environment should be performed to qualitatively conclude on the validity of the results. In conclusion, the computer vision algorithm successfully detected and positioned the leader using the point cloud and image feed generated by a virtual depth augmented stereo camera. From the newly positioned leader, a published way-point was received by the simulation and the follower vehicle successfully followed the leader vehicle. Generating and publishing the full point cloud from the virtual depth camera resulted in a very slow simulation, however, once confidence is gained in the computer vision algorithm, the sparse point cloud and leader position were used, resulting in a significantly faster simulation. Vehicle speeds of up to 40 km/h and maneuvers including sharp turns were tested without problems Using the sparse sensor data. A video from the demonstration can be seen in `https://youtu.be/-Z5703kIbsg`

5.3 Full Autonomy Leader-Follower Stack Demonstration

The purpose of this demonstration is to verify the combined efforts of the previous demonstrations. The demonstration includes tracking of a leader vehicle and publishing of way-points to the following vehicle. The navigation stack uses the way-points to calculate a route and publish the twist message. The simulation framework will drive the leader vehicle using the way-point driver and a predetermined path. The follower vehicle will be driven from the published twist messages.

The node structure of the full autonomy stack is a combination of the previous two node structures. The simulation framework is running on a Windows PC and the ROS based systems (computer vision, navigation stack, and Rosbridge) on a native Linux system illustrated in figure Figure 17. The detection algorithm subscribes to three topics; an image from the depth camera, the follower's odometry, and the leader's pose, and in turn publishes a way-point. The navigation stack subscribes to three topics; the way-point from the detection algorithm, the follower's odometry, and the LiDAR point cloud. The navigation stack will finally publish the twist messages for the follower.

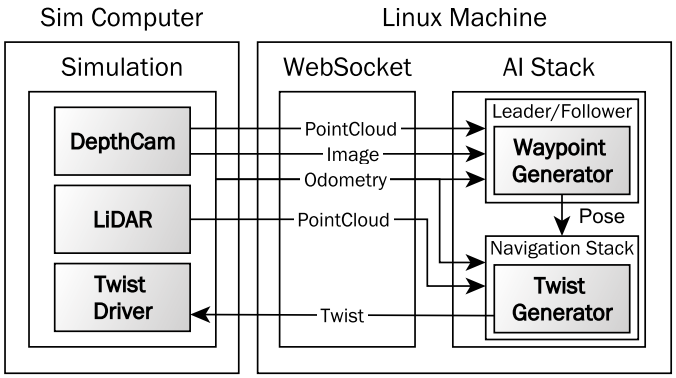


Figure 17: ROS full autonomy stack network.

The full autonomy stack contains the computer vision algorithm generating the way-point and the navigation stack generating the outputs for the vehicle. The computer vision algorithm uses the sparse depth camera model, only subscribing to the leader's pose and publishing the way-point whenever the leader is successfully detected. An example from the demonstration can be seen in figure Figure 18.

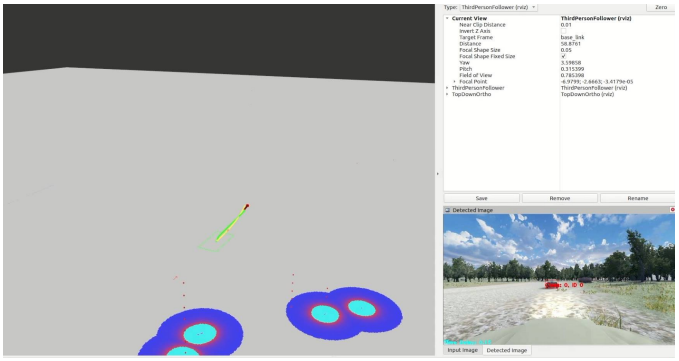


Figure 18: ROS full autonomy stack example.

In this demonstration, the simulation framework drives the leader vehicle following a predetermined route. Along the route an obstacle will appear behind it, blocking the follower's path. The simulation framework generates and publishes the sensor data from the follower vehicle and drives it using the published twist message and twist driver. An example from the demonstration can be seen in figure Figure 19.

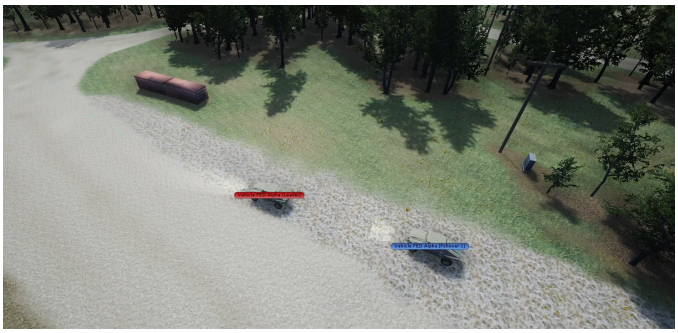


Figure 19: Simulation Environment full autonomy stack example.

The combined functionalities of the full autonomy stack successfully detected and followed the leader using the virtually generated sensor data and camera feed. An obstacle appeared between the leader and follower on a section of the route, to which the autonomy stack successfully maneuvered around and caught up with the leader. A video from the simulation is located at `https://www.youtube.com/watch?v=dYPs3-39VwQ`

5.4 Full Mission Profile Demonstration

The final demonstration is based on the simulation software's autonomy, path and way-point capabilities. The UGV task is based on proposed mission suggestions put forth by the AVT-341 [item 4]. The task, along with the dynamic events, will be handled by the path master and the way-point events. The details of the event and path master are omitted here, but these are scripting tools developed for ease of setting up a mission profile in the developed framework.

The mission is split up into four parts as illustrated in figure Figure 20: Departure (Green), Force Protection (Yellow), Reconnaissance (Red) and Withdrawal (Turquoise). Included in the mission are four vehicles of which; two are Manned Ground Vehicles (MGVs) and two are Autonomous Ground Vehicle (AGVs). In the simulation demonstration, the MGV is autonomously controlled, however, the objectives are altered to fit the proposed tasks. Along

the mission route, specific events or tasks will take place which are further described in each section.

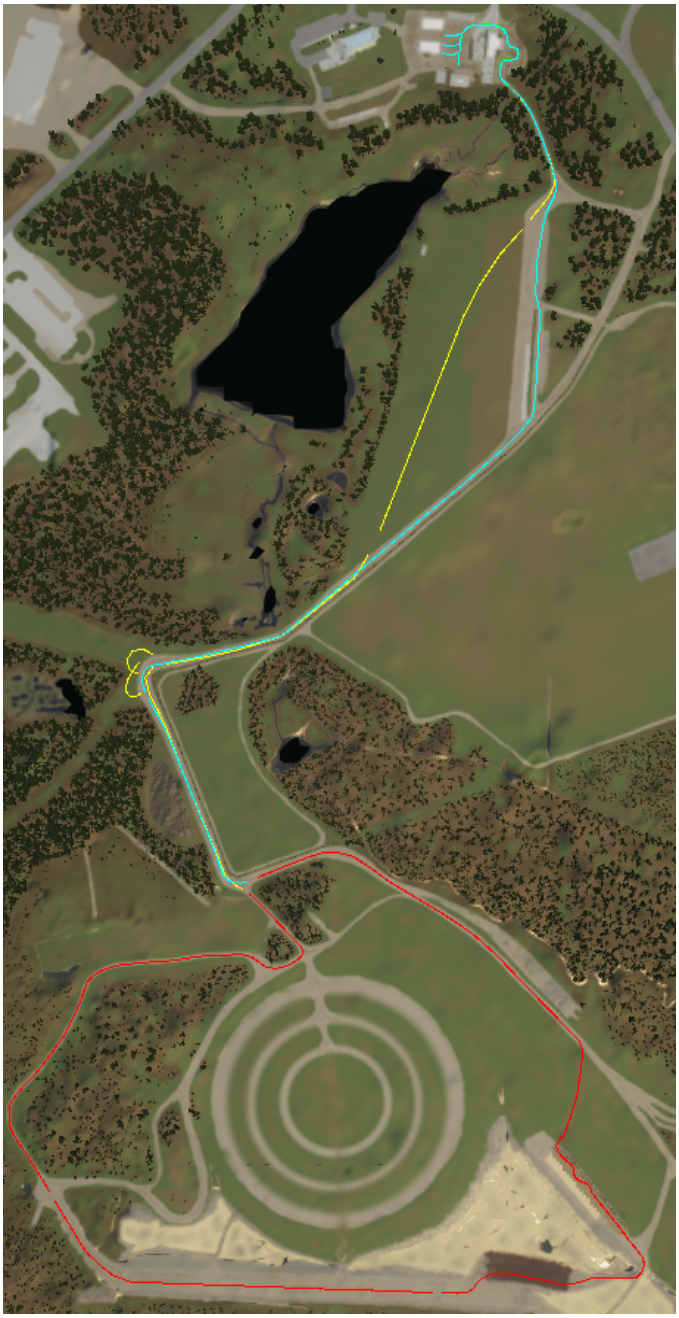
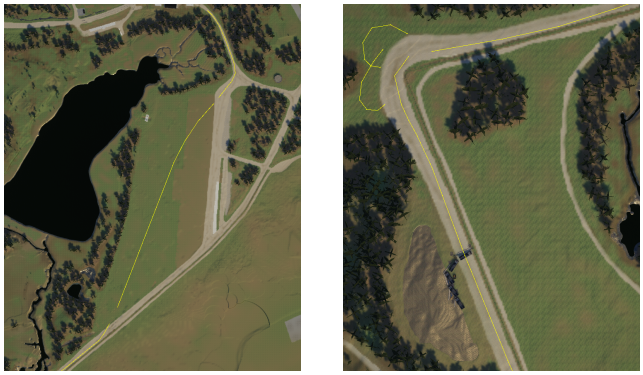


Figure 20: Complete mission overview. Departure (Green), Force Protection (Yellow), Reconnaissance (Red) and Withdrawal (Turquoise).

Departure: The vehicles start from designated positions and orientations. The departure from the forward operating base (FOB) goes through a narrow jersey barrier passage, which is negotiated using leader-follower column formation.

Figure 21: Departure from FOB.

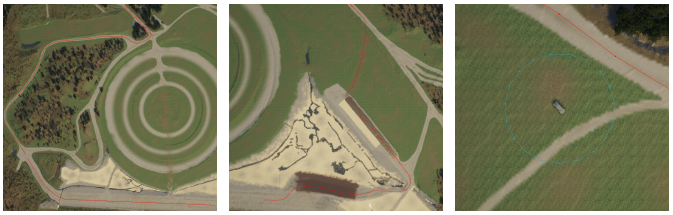
Force Protection: After exiting from FOB in column formation, the convoy swaps to diamond formation crossing the field with an AGV in front and back (Figure 22a). Before reentry to the road, the AGVs move forward in side-by-side formation to scout the road with MGVs following behind. MGVs wait at the assembly area, while AGVs continue the autonomous missions (Figure 22b). Upon meeting unexpected obstacles on the road, AGVs detour into soft soil pit avoiding the leader's tracks. One AGV waits behind at end of the road while the leader AGV continues with reconnaissance ahead.



(a) Field and road (b) Assembly & obstacle

Figure 22: Force protection phase.

Reconnaissance: The autonomous leader goes on with reconnaissance of the terrain, along the road, over the RMS track (Figure 23a), through the soft soil track and back on the road (Figure 23b). Along the road a disabled vehicle is placed, which should be imaged from all sides to gather information (Figure 23c).



(a) RMS track (b) Soft soil track (c) Disabled Veh.

Figure 23: Reconnaissance phase.

Withdrawal: The two AGVs meet up and return to the assembly area regrouping with the awaiting MGVs. Here the obstacle previously on the road has been moved (Figure 24a). The convoy withdraws toward the FOB in column formation. Upon getting a UAV threat detected signal, the withdrawal route should divert into a vegetated route for air concealment (Figure 24b). On reentry to FOB, the convoy should negotiate the narrow passage and park at a designated position (Figure 24c).

(a) Free passage (b) UAV diversion (c) Parking

Figure 24: Withdrawal phase.

Setting up the complete mission profile with all four phases is done using the developed path master tool. The mission is segmented into smaller pieces for a better overview of individual tasks. The triggered event-routes show up as individual tasks, such as U-turns, disabled vehicle reconnaissance, and route deviation. The event system built into the path master handles tasks and signals during the mission and toggles the use of the event-routes. Furthermore, it is also used for handling changes in convoy formation and dynamic events, such as the road obstacle.

Formation changes are handled by the leader, which uses fixed way-points. When the leader reaches a way-point where a change in formation is designated to happen, it swaps the formation protocol, using a reference to the current leader and follower as well as a specific relative position. This allows for linking leaders and followers when using column-based formations and following one leader in diamond formations. The formations can be customized according to what is suited for a specific mission.

Dynamic events are dynamic changes to static objects or states, such as moving a static object to act as an obstacle or activating a UAV threat. The dynamic event handling allows for toggling of any type of event. A dynamic event could be an enemy vehicle traversing a specific path, or a human or animal moving along a prescribed path or directed by an external AI engine.

Assessment criteria Certain assessment criteria can be used When evaluating an autonomous vehicle system's performance. These can vary from mission to mission, as specific mission related tasks should be included in the assessment. The criteria can be split into the following categories, following AVT-341 [item 4]:

- **Task successful**
- **Deviations**
- **Timings**
- **Identification**
- **Response**

A task's success criteria is evaluated on a yes or no basis if possible in comparison with a physical test. Tasks could be way-point reached, obstacle avoided, convoy formation obtained, and similar. The deviation criteria are distance metrics describing the mean deviations from the set route. Examples are the leader distance from the planned route or a follower vehicle's distance from a set convoy formation configuration. The timing criteria measures how fast the mission can be accomplished. The identification criteria rate performance to correctly identify specific objects, such as obstacles, humans, vehicles, etc. on a correct or not correct basis. The response criteria evaluate the system response compared to the expected response given different scenarios, such as a disabled vehicle, obstacle, or UAV threat. The full specified mission is simulated successfully in the simulation framework. A video of the full demonstration is located at `https://www.youtube.com/watch?v=-Ad1tzMIYhM`

Figure 25: Autonomy Stack Assessment Demonstration.

The simulation acts as a demonstration of the simulation framework's capabilities. There was no exact mission acceptance criteria defined so only an example of result metrics are listed in Table 1.

Table 1: Simulation result metrics from KRC mission demonstration.

Category	Metric	Result
Tasks	Exit FOB	Yes
	Diamond formation	Yes
	Side-by-side formation	Yes
	Negotiate soft soil pit	Yes
	Return to base	Yes
Deviations	Leader	Unavailable
	Formation	Unavailable
Timings	Departure	00:01:01
	Force Protection	00:02:44
	Recon	00:06:24
	Withdrawal	00:03:45
Identification	Obstacle	Yes
	Disabled vehicle	Yes
Response	Avoid obstacles	Yes
	Recon disabled vehicle	Yes
	UAV threat diversion	Yes

6. SUMMARY AND FUTURE WORK

Development and testing of autonomous systems is a complex task with many concurrent influencing factors that renders real world testing

insurmountable. A script based configurable digital simulation tool can significantly aid this process.

The paper presented the results of an implementation of a framework and general approach to creating a virtual environment in Unity3D based on a real-world geo-spatial represented of terrain and objects. The framework demonstrated an implementation of communication with existing autonomous ROS-based autonomous algorithms. A dynamic vehicle model and several cameras and sensors allow autonomous navigation and control within the virtual environment.

The autonomous vehicle assessment framework described here is the prototype of a simulation framework used to assess the questions and research tasks arising in the AVT-341 research group. Several areas of the framework, needs further development and testing to achieve higher accuracy and functionalities depending on the mission requirements. Areas of improvement can be in the areas of improved fidelity power train, suspension model, tire model, sensor noise models, soft-body mechanics of vegetation collision and overcoming, and additional types of sensors.

The KRC mission profile demonstration video can be found at `https://www.youtube.com/watch?v=-Ad1tzMIYhM`.

7. REFERENCES

1. Balling, O., Bradbury, M., Bruce, J., Cammarere, M., Choi, K., Dasch, J., ... Wojtysiak, B. (2019). AVT-248 Next Generation NATO Reference Mobility Model (NG-NRMM) Development. NATO Science and Technology Organisation (STO).

2. Mazzega, J. (2019). PEGASUS presentation @ NATO AVT 44th Panel Business Meeting, ET-194. Trondheim, Norway.

3. McCullough, M., Jayakumar, P., and Dasch, J. (2017). The Next Generation NATO Reference mobility model development. Journal of Terramechanics, 73, 49-60.

4. Advanced Vehicle Technology Panel, Research Task Group 341. Mobility Assessment Methods and Tools for Autonomous Military Ground Systems. NATO Science and Technology Organization, 2020.

5. J.Y. Wong, "Terramechanics and Off-Road Vehicle Engineering", 2nd edition, Butterworth - Heinemann, Carleton University, Ottawa, 2010.

6. Lazaro F. Bracamonte and Raymond G. Fontaine, "US Army Test Operations Procedure (TOP)", Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center, Automotive Directorate, 2017.

7. J. Y. Wong, "Theory of Ground Vehicles", Wiley, 2001.

8. Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain, "A survey on LiDAR scanning mechanisms", Electronics (Switzerland), 9(5), 2020.

9. M. Lyasko, "Slip sinkage effect in soil–vehicle mechanics", Journal of Terramechanics, 47(1):21–31, 2010.

10. B. P. Minaker, "Fundamentals of Vehicle Dynamics and Modelling: A Textbook for Engineers With Illustrations and Examples", Automotive Series, Wiley, 2019.

11. R. Codd-Downey and P.M. Forooshani and A Speers and H Wang and M. Jenkin, "From ROS to unity: Leveraging robot and virtual environment middleware for immersive teleoperation", York Centre for Field Robotics and Lassonde School of EngineeringYork University, Toronto, Ontario, M3J 1P3, Canada, 2014.

12. Unity3d.com "Unity Scripting, Raycast" https://docs.unity3d.com/ScriptReference/Physics.Raycast.html, Accessed July 2021.

13. https://github.com/siemens/ros-sharp "ROS#" Accessed July 2021.

14. https://www.ros.org "ROS Operating System", Accessed July 2021

15. M. Skovby and F.Ø. Skafsgaard, "A ROS Based System for Autonomous Driving of Unmanned Ground Vehicles", MSc Thesis, Aarhus University, 2021.

16. J.V.K. Sørensen and T.F.K.Sørensen, "Stereo Camera based Tracker System", MSc Thesis, Aarhus University, 2021