2011 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
MODELING & SIMULATION, TESTING AND VALIDATION (MSTV) MINI-SYMPOSIUM
AUGUST 9-11 DEARBORN, MICHIGAN

DEVELOPMENT OF HIGH FIDELITY MOBILITY SIMULATION OF AN AUTONOMOUS VEHICLE IN AN OFF-ROAD SCENARIO USING INTEGRATED SENSOR, CONTROLLER, AND MULTI-BODY DYNAMICS

Paramsothy Jayakumar, PhD
William Smith
US Army RDECOM TARDEC
Warren, MI

Brant A. Ross, PhD, PE
MotionPort
Ypsilanti, MI

Rohit Jategaonkar
TASS Americas
Livonia, MI

Krystian Konarzewski
Safety Engineering Research, S.C
Warsaw, Poland

ABSTRACT

The mechanical behavior of a military vehicle during off-highway operation is complex and highly nonlinear. Some current vehicle concepts include added intelligence through the implementation of sensors and controllers to enable autonomous or semi-autonomous operations. Control systems have typically been developed with controls software where the mechanical plant and sensors are represented as simplified and often linearized blocks, resulting in a poor vehicle assessment. This paper describes the development of an integrated environment for a control system, mechanical system dynamics, and sensor simulation for an improved assessment of the vehicle system performance. The vehicle chosen is an autonomous robot that attempts to follow a prescribed path along an off-highway terrain. The effect of including a stability controller for vehicle mobility is assessed. The architecture of the integrated simulation environment is described and its potential to improve schedule and reduce risk of the development of mechatronic military vehicle systems is explored.

INTRODUCTION

Vehicles, whether intended for passengers or the military, undergo extensive testing and evaluation from the moment of conceptualization. Vehicle performance has been greatly improved through the use of computer simulation tools, which allow accurate feedback of vehicle behavior while still in the design phase. As vehicles become more autonomous, there is a greater need for simulation technologies which accurately evaluate the entire vehicle system. It is not enough to improve vehicle dynamic performance; we must also evaluate and improve the sensor suite employed on the vehicle, and the controller used to operate the vehicle given the current environment in a desired, predictable, and safe manner. A high-fidelity simulator for autonomous vehicles is needed in order to meet these goals.

Traversal of off-road terrains is often difficult. One approach for improved mobility is the use of track

assemblies rather than tires, whether for full-sized armored vehicles or smaller off-road robots. The improved mobility comes with increased mechanical complexity of the vehicle. Care is needed to properly assess the behavior of such a vehicle as it traverses uneven terrain typical for many missions because the assumption that the vehicle is able to smoothly respond to driver commands may not be valid. Vehicle response to the loss of traction is highly nonlinear and quick adjustments to the driving torque are needed.

Current vehicle concepts often include requirements for some level of autonomous or remote operation. A well-designed system of sensors and controllers is needed to adequately replace the immediate sensing and response of a local driver. There are two logical levels of sensing and control: first, a high-level path planning and execution process and second, the immediate control of vehicle motion given variation in terrain and soil conditions. The system of sensor and controller hardware in combination with planning, response, and communication software can be thought of as the intelligence subsystem of the vehicle.

A typical design process for the intelligence subsystem consists of 1) the specification of the sensors, control processors, and interfacing hardware; 2) the development of the controller software; 3) building a physical prototype; and 4) prototype testing.

Control simulation software is used to design each controller. Basic functions of the controller are represented with blocks and the physical plant (the mechanical vehicle in this case) is typically represented with a simplified function. Sensors and their local controllers may also be modeled with a simplified approximation. The simplification of the vehicle model and sensor function may result in predicted system behavior in the controller design software that does not match the combined behavior of the controller, vehicle dynamics, and sensor measurements. For example, the vehicle may only partially respond to control commands due to traction limitations. The response may also include an unexpected delay. Consequently there may be unanticipated problems with the controller software that are uncovered during the hardware prototype testing of the vehicle system.

OBJECTIVE

The first objective of this work is to develop and demonstrate a high-fidelity integrated simulation architecture that includes sensors, controllers, and a detailed vehicle model. Commercial off-the-shelf (COTS) software will be used as much as possible in order to minimize the schedule and provide maximum opportunity for high-fidelity modeling. Another benefit of using COTS software is a reduced burden of verification and validation (V&V), since the software vendors are responsible for the V&V of their respective product. Consequently the V&V effort for this project can be focused on the data interfaces between the software and the integrity of the data used for the models.

A second objective is to demonstrate a level of fidelity that could allow engineers to be able to assess the effect of specific changes to the vehicle system, such as a change in track design or in controller firmware. It is understood that simulation at this level of detail occurs much more slowly than real-time. The intent is to use this environment as an engineering tool rather than for training or other real-time applications.

TARDEC is interested in using this capability as a design tool that can efficiently improve the design of vehicles. It is important to have access to dynamic load information early in the design process. Also, the combination of force and motion data can provide fundamental information on the energy consumption of the vehicle. This information is especially useful for battery-powered vehicles. Insufficient battery power can lead to a “power failure” which has been shown to be significant in small unmanned ground vehicles [1].

The sensor software should have the ability to simulate a broad range of sensors, including radar, lidar, or laser scanners as well as vision-based sensors. The software should be adjustable to model several types of beam shapes, various scanner sampling ratios, and sensor scanning patterns. The virtual sensors should take occlusion into account and be sensitive to the detected object geometry and surface reflectivity. It should be possible to gather multiple outputs from each sensor, from range and field-of-view to sensor’s spatial accuracy and power loss of the reflected signal. With these capabilities the sensor software may be used to validate the standard performance of active scanners based on radar/laser sensor principles at a system level.

The control system software needs to be able to model control systems in the form of block diagrams as well as in the form of C-code (firmware).

The multibody dynamics (MBD) software should have the ability to model the mechanical components of the vehicle in detail, including track assemblies with suspensions. Software outputs should include the contact forces between the terrain and the track, the track and the roadwheels, idler and sprocket, and the connections to the suspension and to the chassis.

All of the software should have the ability to interface with other software using existing functionality.

LITERATURE REVIEW

The use of simulation to evaluate mechanical systems with controllers has been done for years. An initial application is robot simulators.

Some researchers develop their own robot simulator when evaluating a new control algorithm. This provides the researcher complete knowledge of the mathematics behind

the simulator, and allows for many simplifications necessary for proof of concept. In one example [2] the researchers focused on developing an optimal trajectory generation for rough terrains. Vehicle dynamics were simplified to a three degree of freedom model, and sensors were not modeled. In another example [3] the researchers focused on near-optimal navigation of mobile robots at high speed over rough terrain. Dynamic constraints were developed to limit the motion of the vehicle, while the control algorithm required a smooth terrain. Sensors were also not modeled.

Developing a robot simulator requires a significant time commitment, which does not necessarily benefit the main research goal. Several robot simulators are available today which do not require developing vehicle and sensor models, 3-dimensional environment, and data management.

USARSim [4] is an open source simulator which can support several sensor and robot types. Based on the Unreal Tournament game engine, users can add to the simulator's sensor and vehicle library. By using Unreal for visual rendering and physical modeling, the software allows users to focus on the sensors, control systems, and their interactions.

Webots [5] is a commercially available software by Cyberbotics which provides a library of sensors, robots, indoor and outdoor objects. The software also allows for physics-based simulations through the *Open Dynamics Engine*, an open source rigid-body dynamic simulator.

Microsoft Robotics Developer Studio [6] is a commercially available software which uses NVIDIA's PhysX technology to model physics behavior and the Microsoft XNA Framework to provide real-time 3D graphics rendering. The software includes pre-modeled robots, sensors, and environments.

Other robot simulators include *Gazebo* [7,8] and SIMORE [9]. Perhaps the simulator with the greatest focus on fidelity is *VANE* [10], a high fidelity simulation environment for ground robotics developed by the U.S. Army Engineer Research and Development Center. Sensor models incorporate internal and external noise to best reproduce real-world environments. All efforts are being made to represent the environment, sensors, and their interactions as accurately as possible. VANE is a standalone software which requires a supercomputer to handle its complex virtual environment modeling, and does not model complex vehicle dynamics as was done in this paper.

ARCHITECTURE

Figure 1 summarizes the target architecture. The boxes represent the different software and the labels by the arrows represent the data that flows between the software. This project represents the first time that the selected COTS software have been used together.

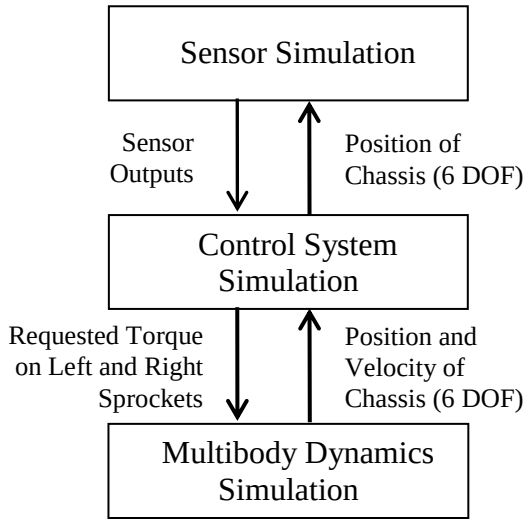

Figure 1. Software Architecture.

Sensor Simulation Software

The sensor simulation software, PreScan, was created as a development environment for Intelligent Vehicle (IV) systems, initially for on-road applications [11]. An IV module is a system with sensors that monitor the vehicle's surroundings and use this information to warn the driver or to intervene in the control of the vehicle. The software can be used to develop IV systems based on sensor technologies such as radar, laser, camera, GPS, and antennas for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication.

There are several modules that together provide everything an IV system developer needs. It works using four steps (see Figure 2). The dedicated PreScan graphical user interface allows users to build traffic scenarios using a database of road sections, infrastructure components (trees, buildings, traffic signs) and road users (cars, trucks, cyclists, pedestrians). Representations of real environments can be made by reading in information from Google Earth [12], Google 3D Warehouse [13], and/or a GPS navigation device. Weather conditions (rain, snow, fog) and light circumstances can also be specified.

Modeling sensors is done by filling PreScan's generic sensor models with supplier-specific specifications. The interface with Matlab/Simulink [14] enables users to design and verify algorithms for data processing, sensor fusion, decision making, and control. This interface also allows for re-using existing Simulink algorithms or vehicle dynamics models from external software. Where appropriate it is possible to run Hardware-in-the-Loop simulations [15].

Multibody Dynamics Software

Commercial multibody dynamics was selected to provide the functionality for this project [16]. It is a general-purpose multibody dynamics software that also includes an application module for tracked vehicle modeling.

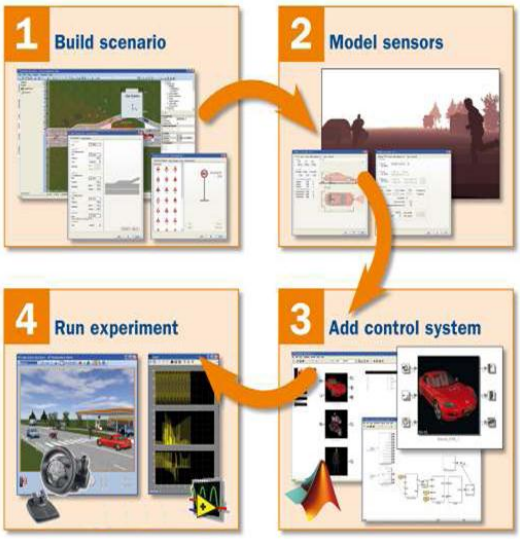

Figure 2. Four Stages of IV Development.

The track module gives the ability to simulate high mobility tracked vehicles, where the track shoes are typically defined as a combination of a steel frame and track pads. Band track with similar characteristics can also be modeled. The toolkit supports the definitions of sprockets, roadwheels, idlers, and the initial track shoe which is replicated along a user-defined path.

A subsystem capability allows each track assembly to be created and simulated separately, then integrated into the vehicle model. An integrated Bekker soil model [17] allows the user to define soils with the standard seven Bekker parameters or with a user-defined function.

Controls systems can be modeled in the RecurDyn environment, and will interact with the mechanical model. This plays an important role since design algorithms of a controller and the corresponding mechanical system ("plant") should be evaluated simultaneously when simulating a computer-controlled system.

There is also an interface that facilitates a co-simulation between RecurDyn and a control system defined in Matlab/Simulink.

Control System Simulation Software

The control simulation software, Matlab/Simulink, is well-known for its use in control system development. It is a logical selection since the select sensor and multibody dynamics software interface with it.

EXAMPLE MODEL

The application chosen for this project is a tracked autonomous robot that follows a path and navigates to avoid an obstacle.

Robot Model

The robot (shown in Figure 3) is a generic model that is used for evaluation purposes and does not have any properties of any commercial robot. This robot has four distinct track assemblies. Each of the main track assemblies consists of a sprocket, idler, 13 roadwheels and 4 carrier rollers. Each of the auxiliary front track assemblies consists of a sprocket, idler, 5 roadwheels, and 2 carrier rollers. The cleats of the track are functional and make contact with the terrain. This robot model is able to adjust the angle of the front track assemblies and climb stairs [18].

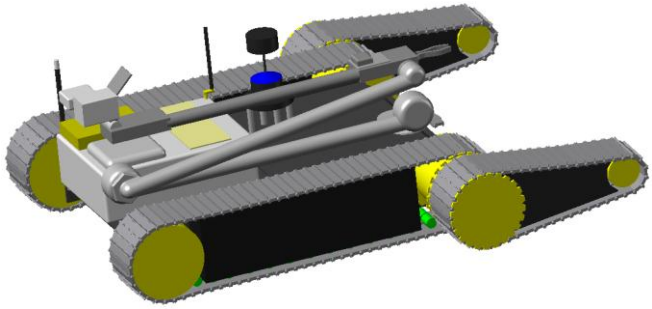

Figure 3. Example High Fidelity Robotic Vehicle Model

The inputs and outputs for the tracked robot model are presented in Figure 1. Specifically the output from the robot MBD model includes the positions (x,y,z), the orientation (x,y,z rotation angles), the forward velocity, and the yaw rate of the robot chassis.

A simplified version of the robot model was developed for use in setting up the controllers for the sensors and for vehicle stability. The track assemblies are replaced with four wheels (as shown in Figure 4) and the wheels on each side rotate together in the same way as any skid-steer vehicle.

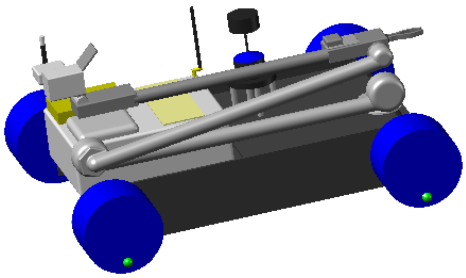

Figure 4. Simplified Robotic Vehicle Model

Scenario

The example scenario consists of the robot traversing a side slope around a hill and then having to navigate around an obstacle, as depicted in Figure 5. The vehicle controllers are needed to simply maintain the robot along a uniform path along the side of the hill. The controllers are further challenged by the need to control the robot around an obstacle that forces the robot to deviate from its target path.

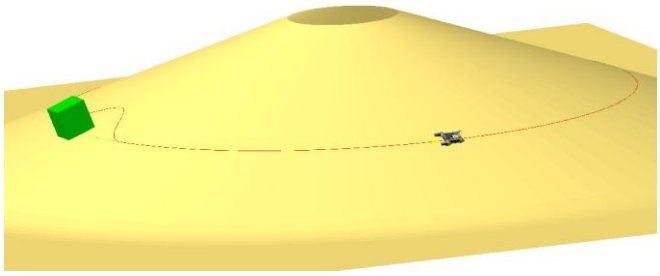

Figure 5. Example Terrain.

Controller

The vehicle uses a controller to turn the desired forward velocity and heading angle into torque commands. This is accomplished using two PID controllers, one for forward velocity and one for heading angle. The velocity controller provides a torque magnitude signal based on the error between the desired velocity and the measured forward velocity. The heading controller determines the torque scaling factor for the left and right sprockets based on an error heading signal that is sent from the navigation controller. The vehicle is capable of providing torque of different sign to each track in order to provide the highest possible turning rate. If a heading error is large enough the scaling factor can be greater than unity, so that the vehicle's turning rate is not limited by the velocity error. This allows the vehicle to turn sharply even during low velocity operation.

A stability controller is used to help the vehicle remain controllable, especially during sharp turns which can result during an obstacle avoidance maneuver. The stability controller monitors vehicle yaw rate, which can be measured using a gyroscopic sensor, and compares this value to an estimated desired yaw rate. Given the current vehicle forward velocity, and the distance and heading angle to the desired point, we can approximate the desired yaw rate. Knowing the desired change in angle (heading), we need the desired amount of time to complete the turn. This is approximated with a straight line from the current location to the new point, traveling at the current velocity. This approximation increases in accuracy as heading angle decreases and the waypoint distance increases. When the error between the desired and actual yaw rate exceeds a specified threshold, the desired velocity is reduced proportionally. The velocity and heading angle controllers are not changed by the stability controller.

The task accomplished by the stability controller is challenging for a tracked vehicle. Observation of tracked vehicle maneuvers by non-expert drivers suggests the difficulty of making smooth turns. With wheeled vehicles the steering of the wheels can be done smoothly with direct motion control. With tracked vehicles the driving torque is controlled and steering occurs when the static friction condition in the lateral direction is overcome. Once the static friction condition is overcome the turning resistance is reduced, thus tending to an overcorrection condition. These conditions lead to an unstable and highly nonlinear vehicle system.

The calculation of the effective coefficient of friction is based upon the relative velocity between the vehicle and the terrain, as shown in figure 6. There are thresholds of static and dynamic velocity as well as static and dynamic coefficients of friction. Transitions are made using a haversine function. When the relative velocity is less than the static velocity threshold:

$$\mu = \text{havsin}(v,0,0,v_s, \mu_s) \quad [1]$$

where μ smoothly transitions from 0 to μ_s as the velocity transitions from 0 to v_s

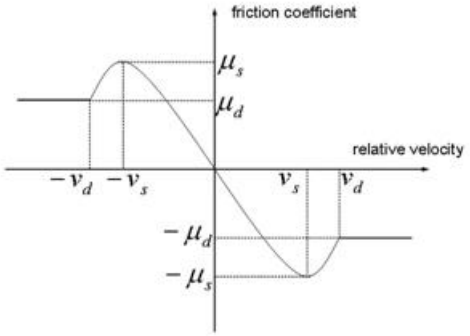

Figure 6. Friction Coefficient as a Function of Velocity.

Likewise, when the relative velocity is between the static velocity and dynamic velocity thresholds:

$$\mu = \text{havsin}(v, v_s, \mu_s, v_d, \mu_d) \quad [2]$$

where μ smoothly transitions from μ_s to μ_d as the velocity transitions from v_s to v_d.

A more advanced controller which takes into account the vehicle dynamics, terrain, and future course would lead to more accurate desired yaw rate estimation and better path following. The goal of this research was not to develop an advanced stability controller, but to show the benefits and potential uses of the robot simulator.

Sensors

The robot sensor suite is composed of nine generic sensors positioned horizontally in-planes at angles 0, +/-10, +/-30, +/-60 and +/-90 degrees from the robot's longitudinal axis (Figure 7).

Each of them fires a single pencil beam with effective range of 10 meters and sampling rate of 25Hz. If an object crosses a beam, information about range to the target is received. The generic sensor that is used is capable of accurate scanning of target geometry and takes into account surface reflectivity as well as possible occlusion. Surface reflectivity is defined by an Object Response Model (ORM) representing illuminated target's reflective cross section. Generic ORM are based on literature and associated with reflectivity in radar range, however, they can be modified to represent a reflectivity of a physical object, and also to a laser beam. It should be noted that the studied case is an ideal one, so no influence of atmospheric conditions or any kind of external noise is present, and the ORM was not specially adjusted.

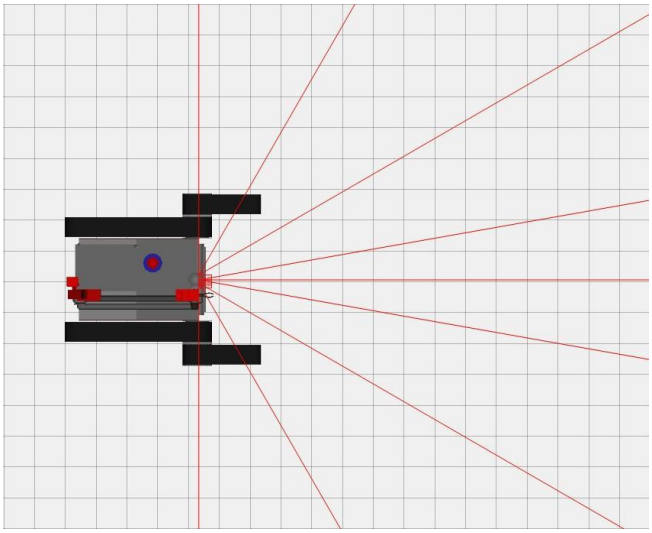

Figure 7. Sensor Definition

Object detection by one or more of the beam and range values gives the needed information to the path planning and avoidance modules that guide the robot.

The robot guidance combines trajectory following with path planning and avoidance algorithms composed of obstacle detection, avoidance, and return-to-the-path modules (see Figure 8).

The trajectory following module guides the robot from one point of the planned trajectory to the next. The trajectory is stored in a matrix, each point defined by its spatial position.

The obstacle avoidance module switches on when an obstacle is detected and the distance to the obstacle is lower than a pre-set value. This module also de-activates the module responsible for travelling along the pre-defined path and stores the last point of the path that has been successfully reached. The obstacle avoidance module utilises following components: 1) emergency stop, 2) obstacle detection, 3) detour from the path, 4) obstacle avoidance, and 5) search for the pre-defined path.

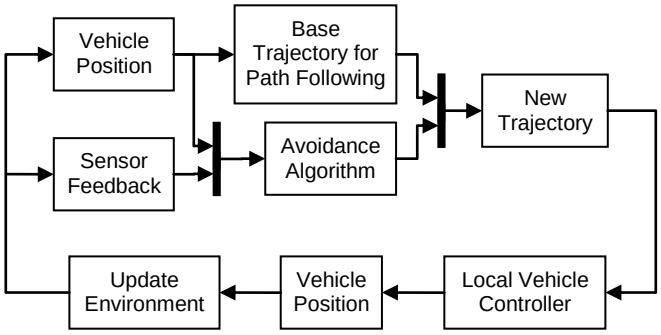

Figure 8. Path Planning and Avoidance Schematic

The emergency stop component is active all of the time and reduces robot velocity when the distance to an obstacle decreases (for example due to the malfunction of other components). If the distance is lower than a pre-set value, the robot stops and waits for external control (operator intervention), effectively ending the simulation.

The obstacle detection component is also permanently active. Based on readings from all sensors this component determines whether an obstacle blocks the vehicle's path. This component also assesses which side of the obstacle has more free space and if the obstacle is not leaning towards one side. Based on this information a decision on how to detour is made. It should be noted that the position of the pre-defined path is not taken into account. When distance to the obstacle becomes smaller than a certain value, detour from the path component is activated.

Detour from the path component forces the robot to take a sharp turn towards the side selected by the obstacle detection component. While detouring, the robot is using information from sensors to track distance to the obstacle. When the distance starts to increase, detouring ends and the next component is activated.

The obstacle avoidance component and the search for the path components work in parallel. Obstacle avoidance is set to keep the robot at a set distance to the obstacle and guides it to travel along the obstacle's side. At the same time the search for a pre-defined path component checks whether or not the front sensor is "crossing" the initial trajectory. If this virtual cross point is detected, travelling along the pre-defined path module activates, and the cross point becomes the next destination point of the robot.

The above sequence of action will repeat itself when a new obstruction is detected.

For the example terrain, obstacle, and robot described above, the avoidance module, linked to the sensor suite, takes control at a distance of 4 m away from the obstacle and guides the robot towards the edge with the shortest path to bypass an obstruction. If no edge is detected, the robot randomly turns left or right and follows an obstruction edge at a distance of 1.5 m. The robot follows the edge until it finds the initial trajectory at a distance of 10 m away. Then, the return-to-the-path module calculates the intersection point between the current heading and the pre-planned path. The robot is guided to the initial trajectory in a straight line and turns toward it as it approaches in order to enter the trajectory at a minimal angle. When it intersects with the initial trajectory at a new point, the avoidance module switches off and the robot begins to travel again along the pre-defined route.

RESULTS

Two vehicle configurations (high fidelity and simplified) and two terrain conditions (high and moderate friction) were considered. For the high fidelity model the high coefficient of friction between the track and the ground is 1.2. The moderate traction condition has an effective coefficient between the track and the ground of 0.7. For the simplified model the two friction values were 1.5 and 0.8, respectively.

The controller and MBD software was used with the high fidelity vehicle model while all three software packages were used with the simplified vehicle model. Figure 9 shows the motion of the high fidelity robot as it follows a predefined avoidance path around an obstacle. Figure 10 shows the motion of the simplified robot as it autonomously adapts its path and negotiates around an obstacle that blocks it from following the initial path.

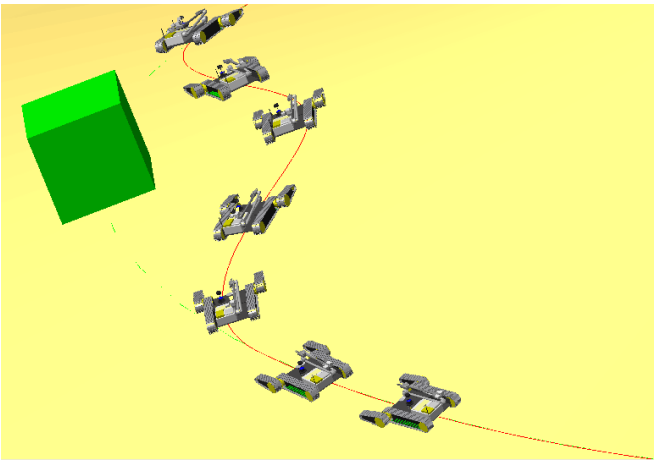

Figure 9. Robot motion around the obstacle in a predefined avoidance maneuver.

Figures 11 and 12 depict example sensor outputs during an arbitrary obstacle avoidance maneuver. The output of each sensor is zero when no obstacle is detected and is the distance to the obstacle when an obstacle is detected.

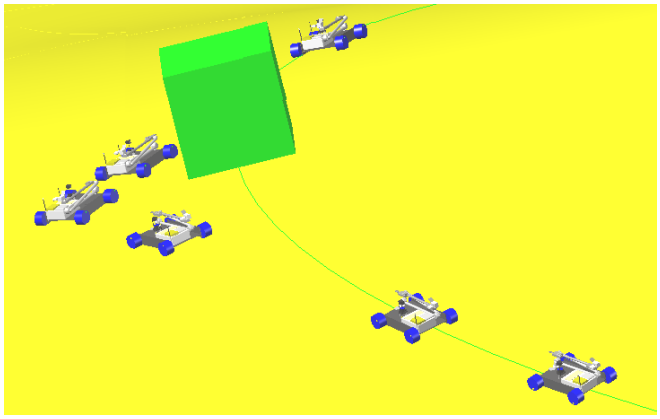

Figure 10. Robot motion around the obstacle in an autonomous avoidance maneuver.

The importance of the scanning from -90 to +90 degrees can be seen from the figures. While a single obstacle to the side may not justify a path change, if an obstacle that lies straight ahead invokes an avoidance maneuver, it is important to know if a turn will take the vehicle towards another obstacle.

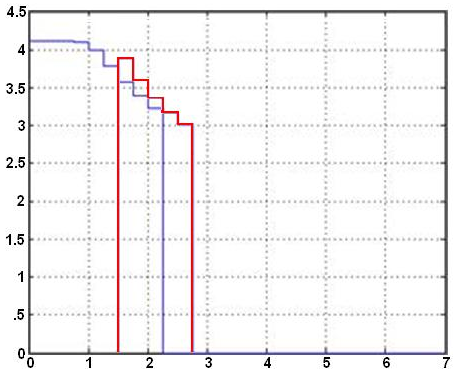

Figure 11. Sensor readings at 0 (red) and -10 (blue) degrees (distance to the obstacle) vs. simulation time

Figure 13 shows the path of the vehicle chassis as viewed from directly above the terrain, with two curves for each friction cases for the high fidelity robot. The difference in the curves reflect the additional difficulty in controlling the turning of the vehicle on the side slope of the terrain with a lower coefficient of friction.

Figure 14 contrasts the torque that was applied to right sprockets for the high fidelity and the simplified robotic vehicle. The torque profiles reflect the difference in the vehicle models as well as the different path selected by the

sensor software in order to avoid the obstacle. The magnitude of the torque values is a function of the rate of change of the heading information from the sensor software and the conversion of the heading input into torque inputs by the local vehicle controller. Careful design of the algorithm that determines the heading angle is needed in order to prevent rapid change in the requested heading angle when obstacle avoidance states change.

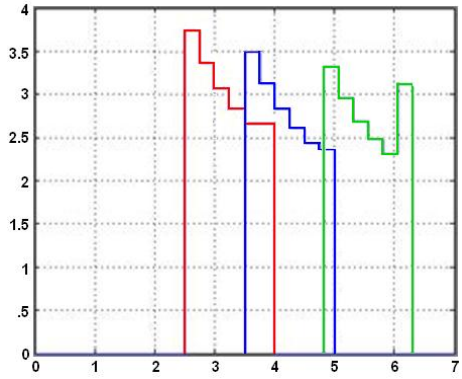

Figure 12. Sensor readings (distance to the obstacle) vs. simulation time (-30 (red), -60 (blue) and -90 (green) degrees).

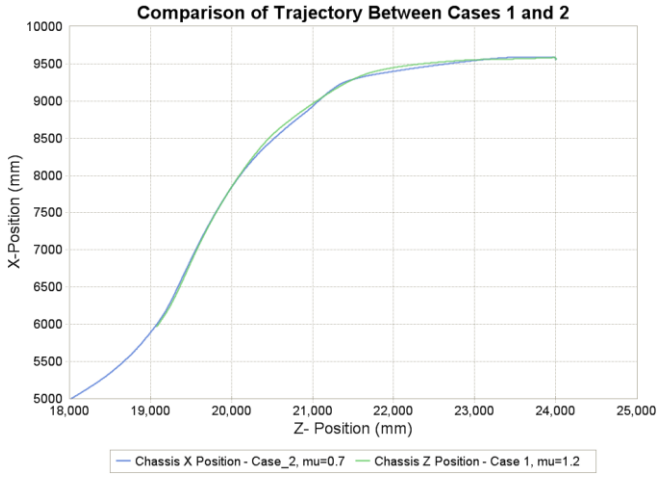

Figure 13. Comparison of the trajectory followed by the chassis of the tracked vehicle for the two cases.

Figure 15 compares the forward velocities of the vehicle chassis and the track of the left and right track assemblies for a portion of the high fidelity, high friction case. The track velocity is calculated by multiplying the sprocket velocity by the radius of the band track at the sprocket. Unlike Ackerman-type vehicles, large slip rates are necessary to make the vehicle turn. This is further complicated for low friction operation.

Figure 16 compares the vertical loads at the sprockets and idlers during the simulation. The difference in values reflects changes in fore-aft and side-to-side weight distribution due to the terrain and the motion of the vehicle.

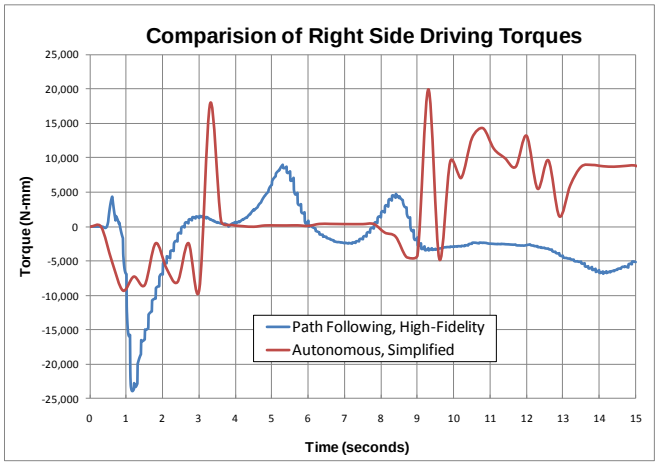

Figure 14. Right Side Driving Torques

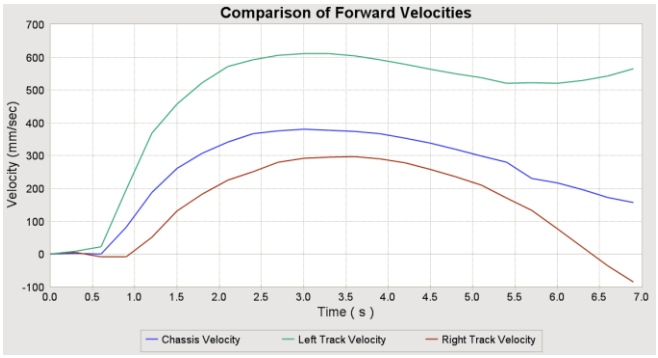

Figure 15. Comparison of the Forward Velocities of the Vehicle Chassis and the Left and Right Tracks.

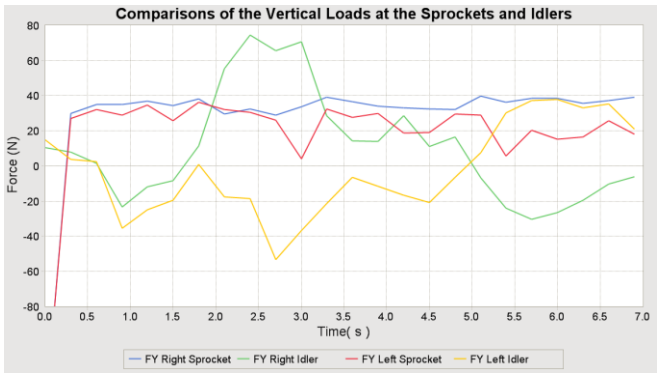

Figure 16. Vertical loads at the Sprockets and Idlers

CONCLUSIONS

The architecture of the integrated simulation environment has been described and its potential to improve schedule and reduce risk of the development of mechatronic military vehicle systems was explored.

The presented work proposes a solution that can integrate both detailed traction and dynamics modeling of a robot with a controller working on sensor inputs. The proposed approach allows engineers to model robot motion in various environments with high fidelity. The fact that sensor readings and their influence on control (and thus the dynamical behavior) are taken into account increases this fidelity. Methods that allow easy building of complicated environments and include the influence of various conditions on overall robot performance may be essential in the development of robust control algorithms, as well as operating procedures and planning. Further, high fidelity models allow the engineer to see the effect of changes to detailed suspension and other mechanical system properties.

The main challenge of the project was to couple three software packages responsible for modeling different aspects of the robot – traction, control, and sensing. This has been successfully achieved, though not without some shortcomings. One of the problems is the simulation time, especially with a very detailed model of a tracked robot. Right now it is not possible to do the simulation in real time, however that leveled of performance can be approached with a simplified model.

A second challenge was dealing with the unstable and highly nonlinear steering response of a tracked vehicle. The need for sophisticated control systems was clearly identified.

Some valuable lessons have been learned. First, control systems can be defined in all three of the software packages used in this study. At the same time the concept of distributed control has become common in vehicle systems. Therefore some decisions had to be made regarding the proper distribution of control in the simulated vehicle system. In this case we separated the sensor controller from the local stability controller used for vehicle mobility. The ability to logically segment the controller function within the simulation helps provide insights about the controllers early in the development process.

Second, control system logic is generally developed in an environment where the plant (in this case the vehicle dynamics) can be evaluated very quickly. However, fast evaluation is not possible with a high-fidelity dynamics model. The resolution is to develop a simplified version of the vehicle model that can be used when the overall control logic is developed. The simplified version of a tracked vehicle is a skid-steer vehicle with four tires. The tires should be located at the sprocket and idler positions and the mass properties of the simplified vehicle should be the same as those of the fully-defined tracked vehicle. Once the general logic of the control system is established, the high fidelity vehicle model should be included in the simulation and the final tuning should be done.

FUTURE WORK

This approach shows a range of future possibilities. Control algorithms can be developed and tested together virtually with a realistically responding robot model. This will surely decrease development time and cost. Moreover, the presented approach can be also used to virtually test the applicability of a robot to a specific task, as well as to test "in the lab" operating procedures of new equipment. Additional model complexities, such as soft-soil modeling using Bekker equations instead of a rigid ground assumption, can further improve the accuracy of the results. A simplified model that is tuned using the data from the high-fidelity simulation, running in real time, may be used for additional testing and training of operators.

ACKNOWLEDGEMENTS

The authors of this paper would like to acknowledge this work would not have been possible without the contributions made by Robin van der Made of TNO Automotive Safety Solutions (TASS), Marcin Miroslaw and Konrad Kamieniecki of Safety Engineering Research (SEARCH), and Andrzej Chmielniak, PhD, Faculty of Power and Aeronautical Engineering from the Warsaw University of Technology.

DISCLAIMER

Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA, and shall not be used for advertising or product endorsement purposes.

REFERENCES

[1] Carlson, J. and Murphy, R. R. (2005). How UGVs physically fail in the field. *IEEE Transactions on Robotics*, Vol. 21, No. 3.

[2] Howard, M. and Kelly, A. (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, Vol. 26, pp. 141-166.

[3] Iagnemma, K., Shimoda, S., and Shiller, Z. (2008). Near-optimal navigation of high speed mobile robots on uneven terrain. *IROS 2008. IEEE/RSJ International*

Conference on Intelligent Robots and Systems, pp.4098-4103.

[4] Carpin, S., Lewis, M., Wang, J., Balakirsky, S., and Scrapper, C. (2007). USARSim: a robot simulator for research and education. *IEEE International Conference on Robotics and Automation*, pp.1400-1405.

[5] Cyberbotics Ltd: Webots (2011), http://www.cyberbotics.com/products/webots/

[6] Microsoft: Robotics developer studio 2008 (2011), http://www.microsoft.com/robotics

[7] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[8] Rusu, R., Maldonado, A., Beetz, M., and Gerkey, B. (2007). Extending player/stage/gazebo towards cognitive robots acting in ubiquitous sensor equipped environments. *ICRA Workshop for Networked Robot Systems*, Rome, Italy.

[9] Kotthäuser, T. and Mertsching, B. (2010). Validating vision and robotic algorithms for real world environments. *Simulation, Modeling, and Programming for Autonomous Robots, Lecture Notes in Computer Science*. Springer Berlin / Heidelbert, Vol. 6472, pp. 97 – 108.

[10] Cummins, C., Durst, P., Gates, B., Goodin, C., and Priddy, J. (2010). High fidelity sensor simulations for the virtual autonomous navigation environment. *Simulation, Modeling and Programming for Autonomous Robots, Lecture Notes in Computer Science*. Springer Berlin / Heidelbert, Vol. 6472, pp. 97 – 108.

[11] Tideman, M. (2010). Scenario based simulation environment for assistance-systems. *ATZ Magazine*, Vol. 112 [Online]. Available: http://www.atzonline.com/

[12] Google Earth: http://earth.google.com/

[13] Google 3D Warehouse: http://sketchup.google.com/3dwarehouse/

[14] Matlab/Simulink: http://www.mathworks.com/

[15] dSPACE ControlDesk: http://www.dspace.com/

[16] RecurDyn Help Manual, FunctionBay Inc., Seoul, Korea, 2011.

[17] Bekker, M. G. (1956). Theory of land locomotion. *The University of Michigan Press*, Ann Arbor, MI.

[18] Ross, B. A. (2006). Dynamic operational limits for a stair-climbing, tracked, urban robot. *SAE 2006 World Congress and Exposition*, 06MV-5.