Virtual Reality
DOI 10.1007/s10055-009-0150-1

SI: MANUFACTURING AND CONSTRUCTION

Modeling and real-time simulation architectures for virtual prototyping of off-road vehicles

Manoj Karkee · Brian L. Steward ·
Atul G. Kelkar · Zachary T. Kemp II

Received: 24 June 2009 / Accepted: 17 December 2009

Abstract Virtual Reality-based simulation technology has evolved as a useful design and analysis tool at an early stage in the design for evaluating performance of human-operated agricultural and construction machinery. Detecting anomalies in the design prior to building physical prototypes and expensive testing leads to significant cost savings. The efficacy of such simulation technology depends on how realistically the simulation mimics the real-life operation of the machinery. It is therefore necessary to achieve 'real-time' dynamic simulation of such machines with operator-in-the-loop functionality. Such simulation often leads to intensive computational burdens. A distributed architecture was developed for off-road vehicle dynamic models and 3D graphics visualization to distribute the overall computational load of the system across multiple computational platforms. Multi-rate model simulation was also used to simulate various system dynamics with different integration time steps, so that the computational power can be distributed more intelligently. This architecture consisted of three major components: a dynamic model simulator, a virtual reality simulator for 3D graphics, and an interface to the controller and input hardware devices. Several off-road vehicle dynamics models were developed with varying degrees of fidelity, as well as automatic guidance controller models and a controller area network interface to embedded controllers and user input devices. The simulation architecture reduced the computational load to an individual machine and increased the real-time simulation capability with complex off-road vehicle system models and controllers. This architecture provides an environment to test virtual prototypes of the vehicle systems in real-time and the opportunity to test the functionality of newly developed controller software and hardware.

Keywords Real-time simulation · Distributed architecture · Virtual reality · Vehicle dynamics models · Multi-rate simulation

M. Karkee · B. L. Steward (✉)
Agricultural and Biosystems Engineering Department,
Iowa State University, 214B Davidson Hall, Ames,
IA 50011, USA
e-mail: bsteward@iastate.edu

A. G. Kelkar · Z. T. Kemp II
Mechanical Engineering Department,
Iowa State University, Ames, IA 50011, USA

1 Introduction

Globalization has put tremendous pressure on US manufacturers to reduce the cost of design and manufacturing products while maintaining quality and reliability. In particular, manufacturers of off-road machinery face a difficult challenge as the cost of design, development, and prototyping is very high due to the size and complexity of their products. A major portion of this cost involves the time taken to design and physically prototype machines and iterate through the design process based on the evaluation of the prototype. A long product development cycle time can cause significant delays in the introduction of a product to the market and a loss of the first-to-market position. In addition, modern off-highway machines are truly mechatronic machines in the sense that they are based on several engineering technologies including mechanisms, fluid power (hydraulics), electronics and embedded control. The design of such machine requires substantial amounts of systems integration, and thus early virtual prototyping and hardware-in-the-loop (HIL) testing is necessary for system design.

Advances in virtual reality (VR) technology offer exciting possibilities in this regard. In manufacturing, VR technology can be used to rapidly develop digital prototypes of the planned or existing products. VR-based digital prototyping, also called virtual prototyping, offers a realistic and flexible presentation of, and interaction with the digital prototypes (Sastry and Boyd 1998; Howard and Vance 2007). The main goal of virtual prototyping is to provide an immersive and interactive virtual environment for the design engineers and product users to examine the accuracy of and usability of the design, which is achieved through immersive displays, haptics, and other sensory input devices (Sastry and Boyd 1998). Virtual prototyping offers a faster and lower cost environment to test the new product designs or modifications to ensure that minimal problems occur at the production stage. The reduced physical prototyping and design cycle will substantially reduce the product development time and cost while improving the product quality (Savall et al. 2002; Howard and Vance 2007).

Models of product system dynamics and simulation are essential components of the virtual prototyping (De Sa and Zachmann 1999; Antonya and Talaba 2007; Howard and Vance 2007). To fully utilize virtual prototypes, these prototypes must be based on high-fidelity dynamic simulation, which not only includes, at times, a complete nonlinear model of the entire machine but also has realistic controller and operator interfaces. High-fidelity dynamic model simulation and realistic visualization helps design engineers understand the behavior of the new or existing off-road vehicle systems so that modifications or iterations through the product design and development cycles can be accelerated. In addition to rapid virtual prototyping, dynamics model simulation and VR visualization provide the benefits of easier and low-cost modeling and testing environment and convenience for studying user's perception and interaction with the new design (Cremer et al. 1996; Schulz et al. 1998; Kang et al. 2004; Castillo-Effen et al. 2005).

Various virtual prototyping applications using system dynamics model simulation and VR have been proposed in the literature to aid in the design, test and manufacture of various vehicle system products (Cremer et al. 1996; Sastry and Boyd 1998; Schulz et al. 1998; Fales et al. 2005; Antonya and Talaba 2007; Howard and Vance 2007). The National advanced driving simulator (NADS) facility at the University of Iowa (Cremer et al. 1996) provided an operator-in-the-loop simulation system which has been used in transportation studies, virtual prototyping and medical research. Schulz et al. (1998) developed an environment for visualizing car-body virtual prototypes. Cuadrado et al. (2004) developed a virtual prototype of a car to evaluate the trajectory responses and optimize the vehicle design. A multi-body system dynamics model was used to simulate the physical behavior of the system. Castillo-Effen et al. (2005) proposed a distributed architecture for modeling and visualization of multiple vehicles. Eberhard and Li (2006) also developed several simulation systems based on the VR and multi-body system dynamics. Antonya and Talaba (2007) developed a VR-based product evaluation and modification environment for mechanical systems.

In the area of virtual assembly, Johnson and Vance (2001) developed an assembly system using a physics engine to simulate the motion of virtual objects. Kim and Vance (2003) expanded this system with haptic capability, and Howard and Vance (2007) modified the system to develop a haptics-based desktop virtual assembly system. Similar virtual assembly systems were developed by Jayaram et al. (1999), Coutee and Bras (2002), Wan et al. (2004), and Zhu et al. (2004). VR has also provided an environment for simulating tele-operation and monitoring of robots and autonomous vehicles (Lin and Kuo 1997; Gracanin et al. 1999) and an interactive environment for vehicle controller design and performance evaluation (Eberhard and Li 2006). However, generality and flexibility of adding new devices or machines to the simulation system were not emphasized in most of these works, which may limit the wider applicability of the associated systems. In addition, it is important that off-road vehicle simulation and visualization systems include operator-and-controller-hardware-in-the-loop capability, which is essential to test the performance of new software and hardware controller devices in various field conditions that may be very difficult or expensive to find in the real world.

For hardware- and operator-in-the-loop simulation and VR visualization, real-time dynamics simulation is required to allow users to intuitively interact with the realistic virtual prototypes and to evaluate the performance of physical hardware. However, achieving real-time simulation goals becomes challenging when high-fidelity dynamic model simulation is required along with the realistic virtual reality visualization (Antonya and Talaba 2007; Howard and Vance 2007). In most of the systems mentioned earlier, real-time simulation was not achieved or was limited to moderately sized dynamic models. To meet these requirements and facilitate virtual prototyping of off-road vehicle systems, there is a need to develop a generic and flexible modeling, simulation and visualization system. The overall goal of this work was to develop a generic and flexible real-time VR-based simulation system for off-road dynamics models with operator and hardware in the loop configuration. The specific objectives were to:

- Develop a general-purpose, real-time, dynamic simulator for off-road vehicles in a VR environment with controller-hardware-and-operator-in-the-loop capability,

- Provide *plug and play* type functionality for simulation architecture requiring minimal effort to switch between models and controllers, and
- Enhance the real-time capability of the simulation using distributed and multi-rate simulation techniques.

2 Real-time simulation architecture

A system is said to be a real-time system if the correctness of the system responses depends on both the accuracy and timeliness of the computation (Stankovic 1988). For real-time systems, even though a computation generates the correct system responses, the responses could be regarded as unsuccessful if they lag physical time (Glinsky and Wainer 2002). A real-time VR-based simulation is required to enable the user to visualize and perceive the effect of their actions during the simulation in real-time (Antonya and Talaba 2007). A primary goal of this work was to provide a real-time simulation environment to the users, which is also flexible, extendable and scalable. As mentioned before, this goal of real-time VR-based simulation of off-road vehicles is challenging because of the heavy computational load required. To achieve this goal, a distributed vehicle simulation and visualization architecture (Fig. 1) was developed. The architecture consisted of three components: a vehicle dynamics model and model simulator, a VR geometric model and visualization system and an interface to controller and input hardware devices. The architecture provided the flexibility of distributing different components of the system across multiple processors to harness more computational capacity for the simulation.

All three components of the simulation architecture were implemented in such a way that the modeling, simulation and visualization systems were highly flexible to allow modifications without major programming efforts (Castillo-Effen et al. 2005). Existing technologies were surveyed to identify the best candidates for the implementation to meet the following three primary requirements.

1. *Modifiable* Easy to modify the visualization environment as well as swappability between available geometric and dynamic models
2. *Portable* across multiple environments
3. *Extendable* Easy to extend the system to include more dynamic models and additional hardware interfaces

2.1 Modeling and simulation environment and dynamics modeling

Matlab Simulink (The Mathworks, Natick, MA) was employed for dynamics system model development. Matlab Simulink offered several desirable features such as:

- Block-diagram–based modeling, which allowed to model a system with multiple hierarchical subsystems essential for cleaner and generalized modeling (e.g., swappable subsystems),
- Facility to include C/C++ code allowing easier socket programming and machine synchronization,
- Facility to automatically translate Simulink block diagrams into low level code and to build into executables, consequently accelerating the simulation speed.

In addition, SimMechanics, a physical modeling environment for mechanical systems, was used to develop some the modules included in the model library. To enable dynamic simulations to be seamlessly integrated into current and future VR applications, a modular environment was adopted for the vehicle dynamics modeling. This modeling environment was based on a system of data buses defining the interfaces between vehicle subsystems and the encapsulation of the vehicle subsystems into swappable model blocks. The I/O bus was designed to eliminate the need for the user of the system to manually make connections between dynamic model blocks. Through the data buses, all signals were routed using Simulink tags, which acted as *wireless* links between model blocks. Each model

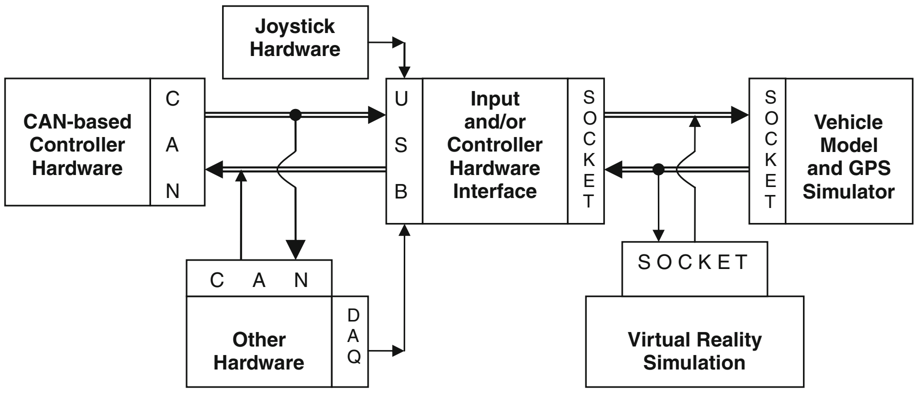

Joystick Hardware
CAN-based Controller Hardware
CAN
USB
Input and/or Controller Hardware Interface
SOCKET
SOCKET
Vehicle Model and GPS Simulator
CAN
Other Hardware
DAQ
SOCKET
Virtual Reality Simulation

Fig. 1 Generic distributed VR-based simulation architecture for off-road vehicle system dynamics models. The block diagram shows the communication paths and messages among the software and hardware components incorporated in system

block in this environment was a subsystem with a self-contained set of inputs and outputs connected to buses. Thus, this modeling environment fostered a modular environment so that users without detailed modeling expertise can still assemble vehicle dynamics models and run simulations. This simulation environment not only presented a cleaner appearance and simpler modeling environment, but also enabled model component swappability for dynamic system models of vehicles and components. Using the available set of subsystem model blocks with varying complexity, a user can easily optimize the model complexity to match available computational power and meet the need of the real-time simulation. Users can also develop their own models and add to the library with minimal effort.

Off-road vehicle subsystem models were developed and incorporated into a *model block library* (Fig. 2). The library included a preliminary set of subsystem models for a wheel loader, an agricultural tractor and towed implement, and a backhoe loader. The subsystem models included were chassis models, drive train models, linkage models, hydraulic models, navigation controller models, steering controller models, a VR interface, a hardware interface, and an internal data bus. The interfacing subsystems were developed using C-language code within the Simulink S-function builder blocks. Transmission control protocol/Internet protocol (TCP/IP) socket was used as the communication channel.

Alternative models developed for a particular subsystem were categorized into the common groups in the model block library. Different blocks from the same category can be swapped in and out of a vehicle system model without any changes to the system. Using the simulation architecture and the model block library, the set-up of the dynamics for an off-road vehicle simulation was a simple process consisting of the following three steps.

1. Select one instance of each category of required blocks and 'Setting/Communication Bus' block from the library.
2. Place each of these blocks in a Simulink Workspace.
3. Compile with Real-time Workshop and produce an executable.

2.2 VR visualization

The second component of the architecture was the VR visualization system. For an off-road vehicle visualization system to be widely adaptable, it should provide an environment with the following features:

- Seamless handling of low level details like culling, texture mapping, and rendering.
- Scalable from a desktop to a six-sided immersive VR visualization system.
- Feel of realism and interactivity such as immersion, and walking through the scene.
- Ease of manipulating the scene such as swapping different scene and vehicle models, setting initial location and orientation of the scene.

VR technology was used to visualize the off-road vehicle systems (Fig. 3a) because it offered all of the required features. The visualization system was based on a hierarchy of several programming tools and libraries (Fig. 3b). VR Juggler (Infiscape Corporation, Ames, IA) was the backbone of this visualization system, which provided a generic virtual reality development platform. OpenSceneGraph (OSG; AI2, Universidad Politecnica de

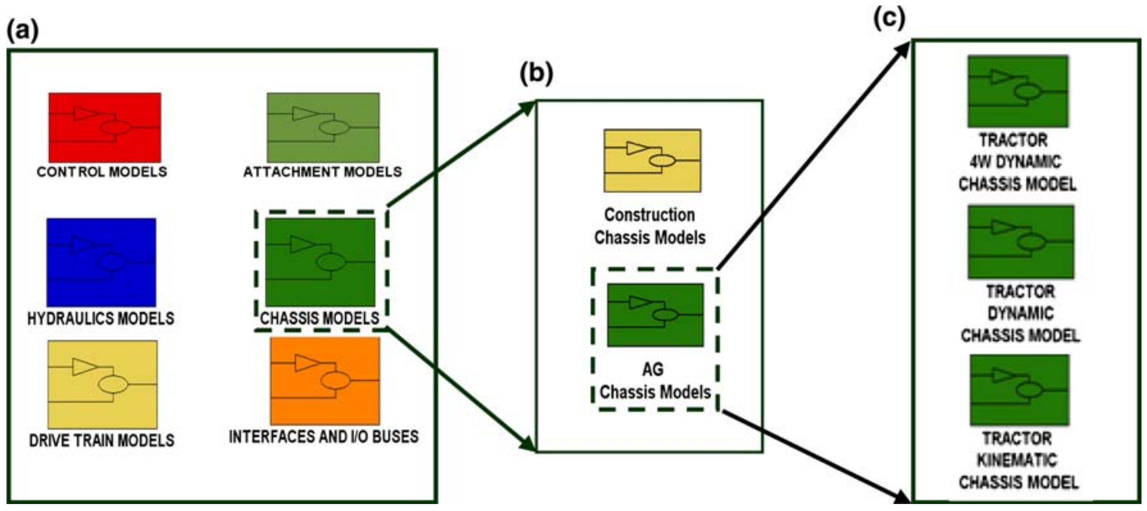
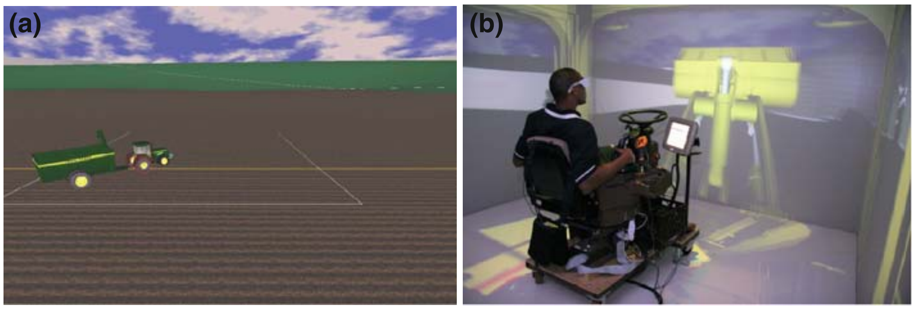


Fig. 2 Generic modeling and simulation environment for off-road vehicle systems; **a** library of generic model blocks, **b** available vehicle chassis models, and **c** available chassis models for a tractor



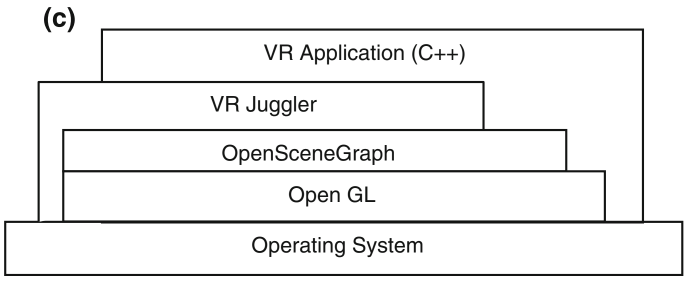


Fig. 3 Off-road vehicle VR visualization system, **a** the virtual farm scene with an example machine, **b** operator in the loop interface for real-time simulation with a four-sided CAVE VR visualization (Virtual Reality Application Center, Iowa State University), and **c** the software hierarchy

Valencia, Spain) was used as the computer graphics engine. A generic application structure was developed for the VR visualization based on an XML parameter file. Visualization and communication-related parameters were read from a user-defined parameter file, which provided the flexibility and simplicity in selecting the required vehicle geometries and setting the initial position and orientation of vehicles in the virtual scene. The capability to switch geometric models of vehicle parts, implements, and attachments during run-time was also implemented so that the design reviews can be performed conveniently. A shared application data structure was implemented to synchronize the communication between dynamics simulator and the clustering nodes used in the immersive VR.

Incorporation of real world terrain into the VR visualization system was essential for studying vehicle dynamic behavior and safety scenarios in realistic terrains. Publicly available digital elevation models (DEM) provided by the United States Geological Survey (USGS) were incorporated into the virtual world. DEMs represent the elevation data using regularly gridded points over the field. ArcGIS (ESRI, Redlands, CA) was used to convert gridded elevations into polygonal representation in VRML format before importing it to the OSG. The VR application implemented terrain following by collision detection of each vehicle wheel with the virtual world terrain.

2.3 Hardware interface

The third component of the simulation architecture was the interface between the vehicle model simulator and controller and operator input hardware. Visual C++ was used to develop the hardware interface application on a Windows platform. The interface was capable of communicating with controller area network (CAN)-based hardware. CAN is an asynchronous serial network used in vehicle systems, which was designed to allow controllers and other electronic units to communicate among each other without requiring a host computer (Bosch 1991). Because the CAN bus is a common resource to all the nodes connected to it, a priority-based arbitration and scheduling scheme, also called carrier sense multiple access with collision avoidance (CSMA/CA), is used to meet the deadlines of each unit in spite of the competition for the communication medium (Livani et al. 1999). The CAN protocol for data transfer requires an arbitration or identifier field to specify a source address and global priority among the nodes. The specification allows 0–8 byte CAN message transmitted in the data field (Fig. 4).

The CAN protocol allows only message-based communication, meaning that messages are broadcasted to the bus without specifying any destination address. A CAN node scans through all messages on the bus to find useful messages. The standard supports up to 1Mbits/s data transfer rate (Pazul 2009). The interface program developed in this work simulated a virtual CAN node to read and write required CAN messages. The interface also incorporated a data acquisition (DAQ) system to sense various physical hardware signals. Joystick devices were also interfaced to provide additional user inputs. The interface communicated to the system dynamics simulator through a TCP/IP socket and to the hardware through the USB ports. A CAN to USB adaptor (LAWICEL AB, Sweden) was used to provide a hardware interface to the CAN-based hardware.

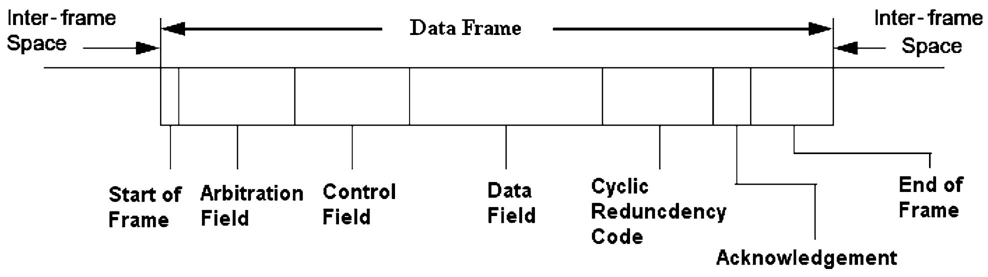


Fig. 4 CAN data frame structure includes an arbitration field, also known as identifier, used to resolve the conflict when two or more nodes in the network send messages at the same time

2.4 Communication scheme

The three components of the distributed simulation architecture communicated to each other using a client–server TCP/IP sockets. The system dynamics model simulator was designated as the server to establish socket communication as this component was required to communicate with both the hardware interface and VR visualization system. Two server sockets were implemented in the vehicle system model blocks; one responsible to receive data from the hardware interface and the other from the VR visualization system. One client socket was created each in the hardware interface and the VR visualization systems. A generic packet structure was defined for each communication channel (model simulator to VR, VR to model simulator, model simulator to hardware interface, and hardware interface to model simulator) to provide a common communication protocol required to simulate various off-road vehicle systems.

At the start of a simulation, the vehicle model simulator started a server socket and waited for the connection request from the hardware. Similarly, another server socket waited for the VR visualization system. Once the connections were made, handshaking was performed to identify and initialize the data communication among the components. Then the information produced by the vehicle model simulator (e.g., position, attitude, yaw-rate) was sent to both the hardware interface and the VR visualization system. Using this information, the interface generated messages in the format that CAN-based hardware recognizes and sent them over the CAN network to the hardware. The interface was also responsible to acquire user inputs through hardware data acquisition systems and forward the signals to the vehicle model simulator.

The VR visualization system received the required information from the model simulator and applied corresponding transformations/mapping to the appropriate objects in the virtual environment. The VR visualization system was also responsible for estimating and sending vehicle attitude information back to the vehicle model simulator to simulate the vertical, roll and pitch chassis dynamics. In this communication scheme, each component in the architecture sent one data packet every 20 ms. This speed was sufficient to communicate with user input devices and other hardware whose practical frequency/data-rate, often, was about 5 Hz. The speed was also enough to visualize the system with reasonable refresh rate ($\sim$50 frames per second). As the data rate required was not very demanding, the Ethernet communication medium used in this architecture provided the required bandwidth without any problem.

2.5 Summary

The real-time simulation architecture is well suited for real-time vehicle system simulation when the subsystems could be represented as individual modules. Both physical modeling and mathematical modeling approaches can be incorporated together to achieve a system level simulation. Furthermore, various hardware and software modules can be connected to this environment as long as they satisfy the communication protocols. This flexibility can be utilized to integrate technologies such as reconfigurable computing to speed up the dynamic model simulation.

The limitations associated with the modeling framework follow from limitations inherent with the Matlab/Simulink and other causal, block diagram-based modeling tools currently used. Because these tools require causal relationships to be assigned through signal flow directions and port directions on functional blocks, the topology of the architecture was unnecessarily constrained (Fritzson and Bunus 2002). To be able to add new subsystem models into this framework, the input/output signaling convention must strictly be followed. Newer acausal, physical modeling tools will address these limitations and are being investigated. Also, because the simulation architecture was based on TCP/IP socket protocol, the available bandwidth may limit the application of this architecture to higher speed systems.

3 Case studies

The proposed distributed real-time simulation architecture was used to develop various off-road vehicle simulation and visualization applications. Two of the candidate applications are discussed in this section to demonstrate the real-time simulation capability and flexibility of the simulation architecture. The first case study focuses on the multi-rate simulation, whereas the second case study focuses on the hardware and operator in the loop simulation capability.

3.1 Multi-rate simulation

Real-time simulation is necessary for virtual prototyping in off-road vehicle systems design and development. The distributed simulation architecture harnessed the computational power of multiple processors to increase the real-time simulation capability of off-road vehicle models. However, as the model complexity is increased in the pursuit of higher accuracy and fidelity, the dynamic model simulation alone may limit the real-time simulation capability.

Various strategies were investigated to reduce the computational burden on the dynamics model simulation so that real-time simulation can be achieved for increasingly complex models. Simulation speed can be increased by simplified rigid body dynamics, constraint maintenance, and velocity slow down compensation, but the model accuracy and/or fidelity will be compromised (Howard and Vance 2007). Selecting a suitable simulation time step was also found to be critical to achieve a good trade-off between the simulation speed and the accuracy of the simulation results.

When modeling and simulating the dynamics of mechatronic systems typical of off-road vehicles, models will tend to consist of subsystem models which have slower and faster dynamics and are often lightly coupled. One way to improve the real-time performance of these stiff dynamical systems is to simulate different subsystems at different rates, so that unnecessary computational burden is reduced (Fig. 5). The modular simulation architecture developed in this work offered a highly favorable modeling and simulation environment to implement multi-rate simulation. Multi-rate simulation also provided an opportunity to divided subsystem models into different machines to further increase the computational efficiency and thus the real-time capability.

Multi-rate simulation was implemented by creating two different model blocks containing relatively faster and slower dynamics in each model. Socket communication provided the required communication between two models. In this simulation approach, three different time steps must be determined; the time step for slower dynamics (T), the time step for faster dynamics (t), and rate at which the two models communicate (c). Because faster dynamics require a smaller time step, the relationship $t < T$ is always true. It can also be assumed that the relation $t < c < T$ will hold

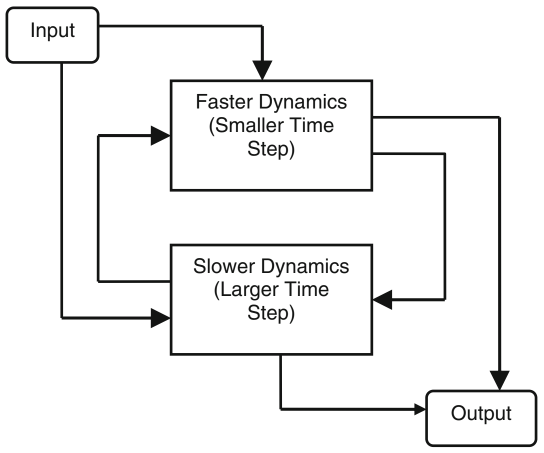

Input
Faster Dynamics (Smaller Time Step)
Slower Dynamics (Larger Time Step)
Output

Fig. 5 Multi-rate simulation scheme. Faster and slower dynamics were separated and simulated with different time steps

for the communication time step c. There is a trade-off in selecting these time steps. Larger time steps might lead to instability or loss of accuracy, while smaller time steps might lead to slower simulations. The choice of t and T also depends on the solver or integrator used for each model. It is preferable to use the best possible solver for the faster dynamics model and a moderately accurate solver for the slower dynamics model. However, the solvers can be changed if they do not give rise to either the required accuracy or stability or real-time speed.

To choose the three time steps, first, a maximum value of a time step s was found with which the whole model ran without instabilities. This value of s was used as the reference point for all the three time steps t, c and T. The initial estimates for t was anywhere between $s/10$ and s/5, and c and T were set equal to s. If the simulation was numerically unstable with these time steps, the value of t was decreased until the system became stable. If the simulation was stable, c and T were increased together until the system became marginally stable. The time step c was fixed at that point and T was further increased until real-time simulation was reached without affecting the stability. If the real-time goal was not achieved, a larger time step t and reduced order hydraulics solver were investigated.

In this case study, a backhoe loader model was developed using the library blocks developed in this work (Fig. 7a). The model consisted of a rear linkage dynamics, linkage hydraulics dynamics, interface to VR visualization, interface to the operator inputs, and vehicle ID. The user input interface program was developed to establish communication with USB-based joysticks and received signals from the joysticks and sent this information back to the vehicle model via a TCP/IP socket. These joystick signals were used to provide steering, drive, brake and lift inputs required for the dynamic simulation. *Vehicle ID* block was required to inform the VR visualization system of the type of machine being simulated so that appropriate geometric models can be loaded and simulated.

For the loader model, the linkage and the linkage hydraulics were the two main components offering the possibility of the multi-rate simulation. The backhoe loader rear linkage multi-body dynamics model was developed using SimMechanics toolbox. There were 16 bodies in the linkage model consisting of the swing, boom, crowd, bucket, upper link, lower link, two swing cylinders, two swing pistons, one each of the boom, bucket and crowd cylinders, and one each of the boom, bucket and crowd pistons. Assuming the generalized coordinates to be the positions and orientations of the center of mass (or some point) of each body, there were 96 generalized co-ordinates and so 96 degrees of freedom before constraints.

The hydraulic system actuating the backhoe loader rear linkage consisted of two inlet valves, two compensation valves, two return valves, two cylinder volumes, and a load-sensing pump (Fig. 6) for each cylinder load actuation. The flow rates through the valve openings were calculated using the following orifice equation,

$$Q = A_O(x) C_d \sqrt{\frac{2|P_{in} - P_{out}|}{\rho}}\, \mathrm{sgn}(P_{in} - P_{out}) \tag{1}$$

where Q is volume flow-rate, $A_0(x)$ is orifice area which is a function of joystick position x controlling the valve spool, C_d is coefficient of discharge, P_{in} is inlet pressure, P_{out} is outlet pressure, and ρ is fluid density. For the inlet valves,

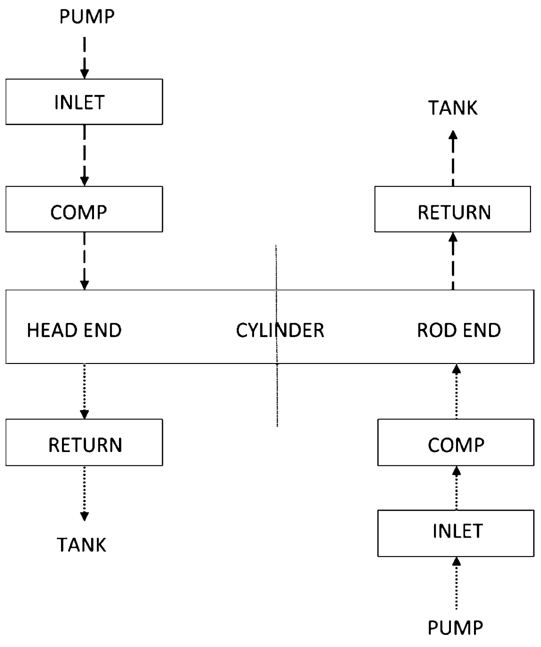


Fig. 6 Fluid flow schematic of the backhoe loader hydraulic system

A_0 was determined using a metering curve translating the user joystick inputs to the area of the valves.

The area of the compensator valve was modeled as,

$$\frac{dA_c}{dt} = K(P_c - P_l) \tag{2}$$

where A_c is area of compensator valve, K is a constant, P_c is compensator pressure and P_l is load sense pressure.

The pressure dynamics at the pump outlet were represented by,

$$\frac{dP_p}{dt} = \frac{1}{2\pi}\left(-Q_v - P_p K_{LP} + \frac{\omega_p D_p}{2\pi}\right) \tag{3}$$

where P_p is the pump outlet pressure, Q_v is the net flow-rate, K_{LP} is the leakage flow coefficient, ω_p is the pump rotational velocity, and D_p is the pump displacement.

The pressure dynamics in the cylinder volume were represented by,

$$\frac{dP_c}{dt} = \frac{\beta}{V}\left(Q_v - {dV}/{dt}\right) \tag{4}$$

where P_c is the cylinder pressure, β is the fluid bulk modulus and V is the cylinder volume.

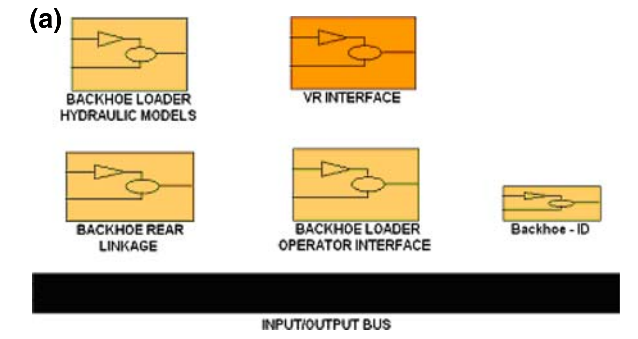
The cylinder force was then calculated as,

$$F = P_h A_h - P_r A_r - bv \tag{5}$$

where P_h and P_r are the pressures in head and rod ends of cylinder calculated using 4, A_h is piston area, A_r is effective area in the rod end, b is the viscous damping coefficient, and v is the piston velocity.

This single cylinder hydraulic system consisted of four dynamics states, four orifice equations and two lookup tables for the valve metering curves. Because there were total of five cylinders consisting of two swing cylinders, and one each of the boom, bucket and crowd cylinders, there were 20 dynamic states, 20 orifice equations and 10 one-dimensional lookup tables in the backhoe loader rear linkage hydraulic system. The hydraulic model was challenging to achieve a real-time simulation goal because it consisted of high frequency dynamics requiring very small integration time step and also highly nonlinear states.

Because the hydraulics dynamics required a smaller time step to run when compared to the loader linkage dynamics, a multi-rate simulation approach was implemented to improve the real-time simulation capability for this model (Fig. 7b). As mentioned before, there were two solvers and three time steps to be chosen to perform this multi-rate simulation. A fifth order solver/integrator (Simulink *ode5*) was suitable for the hydraulics block, while a third order solver (Simulink *ode3*) was suited for the linkage block. The simulation time steps s, t, c and T were 0.005 s, 0.001 s (ode5 solver), 0.002 s and 0.008 s (ode3 solver), respectively.


(a)
BACKHOE LOADER HYDRAULIC MODELS
VR INTERFACE
BACKHOE REAR LINKAGE
BACKHOE LOADER OPERATOR INTERFACE
Backhoe - ID
INPUT/OUTPUT BUS

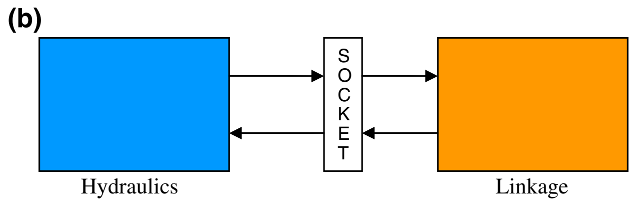


Fig. 7 A Backhoe loader model developed using the model block library; **a** top level view and **b** multi-rate simulation with socket communication

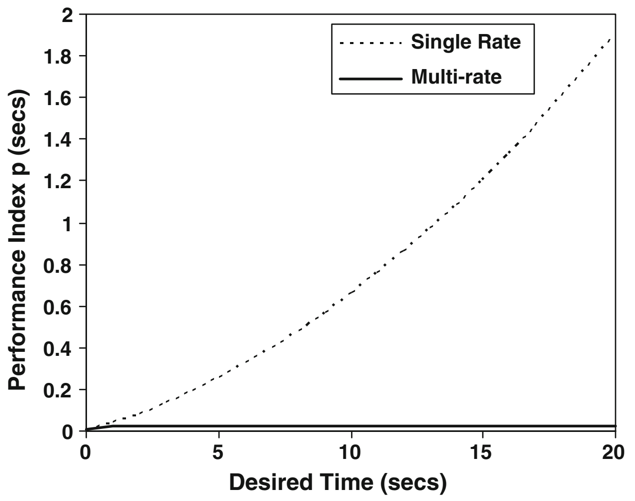


Fig. 8 Real-time simulation performance with the single-rate- and multi-rate simulation techniques

3.1.1 Discussion

Multi-rate simulation, in addition to the computational efficiency provided by the distributed architecture, provided a method to reduce the burden on the dynamics simulator due to unnecessarily small integration time step could be reduced. The following metric was used to compare the real-time performance of the single-rate and multi-rate simulations.

$$p(t_a) = \frac{1}{T}\int_{o}^{T} (t_d - t_a)dt_a \qquad (6)$$

where $p =$ performance index, $T =$ total desired time, $t_a =$ actual time, and $t_d =$ desired time. *Actual time* is the computational time required by the microprocessor to complete the simulation for a single integration cycle. *Desired time* is the integration time step required by the system dynamics. The simulation will run in real time if the actual time is smaller than or equal to the desired simulation time. Multi-rate simulation showed a potential to increase the computational speed over the single-rate simulation and thus the potential to achieve the real-time simulation goal for complex vehicle system models. The result indicated that the time deficit of the single-rate simulation to meet the real-time goal was increasing exponentially with simulation time, whereas that of the multi-rate simulation remained constant at a very low value over the entire simulation period (Fig. 8).

This case study demonstrated that the multi-rate dynamics modeling can be used successfully to reduce the computational burden of the real-time VR-based simulation of off-road vehicle systems.

3.2 Operator and controller hardware in the loop simulation

The ability to test the performance of various software and hardware controllers in the virtual environment can reduce the development time and cost for the new controllers. Using the real-time VR-based simulation architecture, a controller-hardware-in-the-loop (CHIL) simulation was implemented to incorporate CAN-based navigation system and automatic steering controller hardware. In contrast to the classical concept of hardware-in-loop (HIL) system, which involves simulating a controller and putting a real machine in the loop, the CHIL system simulates the machine and puts the hardware controller in the loop. This scheme enabled the simulation study of the performance of newly developed controller hardware and software. The scheme also enabled the study of the vehicle dynamics models with the current controller hardware. The generic simulation architecture (see Sect. 2) was used to perform this case study with the navigation and universal steering controller hardware units in the loop (Fig. 9).

Various component models were selected from the model block library (see Sect. 3.2) to construct a tractor and towed implement system model with CHIL configuration. The vehicle system model included a chassis model, a drive train model, an interface to the navigation controller hardware (including GPS simulator), an interface to the steering controller hardware, an interface to the VR visualization, an Input/Output bus and a Vehicle ID (Fig. 10). An implement model was also added to the system. The *Vehicle ID* block was required to inform the VR

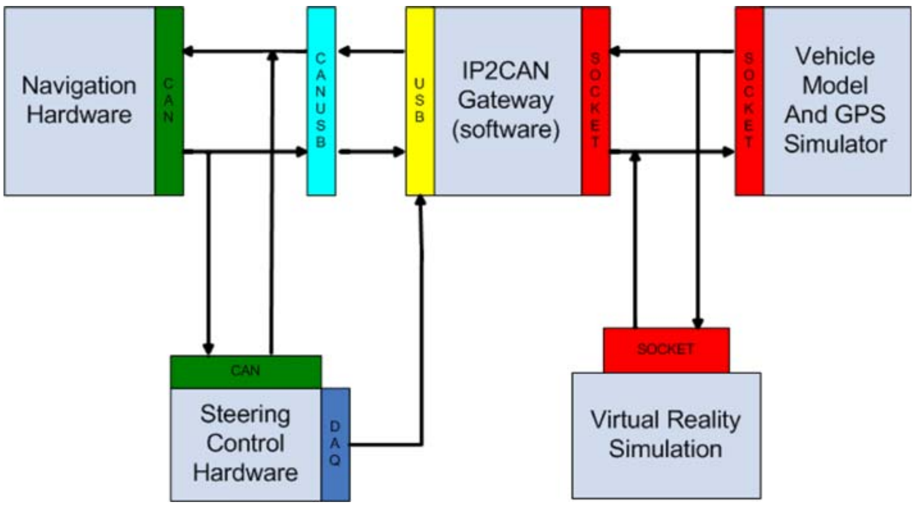

Navigation Hardware
CAN
CANUSB
USB
IP2CAN Gateway (software)
SOCKET
SOCKET
Vehicle Model And GPS Simulator
CAN
Steering Control Hardware
DAQ
SOCKET
Virtual Reality Simulation

Fig. 9 An implementation of controller-hardware-in-the-loop architecture using automatic guidance navigation and steering controller hardware

visualization of the type of machine being simulated so that appropriate geometric models can be loaded and simulated. The model was very simple at the top level (Fig. 10a) showing just the subsystem blocks consisting of the vehicle system model. The actual mathematics and the required interactions between subsystems were hidden inside the subsystem blocks and the data bus block that could be exposed as the individual blocks are explored (Fig. 10b, c).

Three different tractor chassis models of varying fidelity namely a kinematic model, a dynamic model and a four-wheel dynamic model were developed and included in the model block library. The kinematic model considered only the geometric variables and thus consisted of only the positional states. The dynamic model consisted of tire side slip states and velocity states in addition to the positional states as represented by the 7–12 below. The lateral velocity state equation is:

$$\dot{v} = -\frac{C_{\alpha,f}\alpha_f}{m} - \frac{C_{\alpha,f}\alpha_r}{m} - u\gamma \tag{7}$$

where v is the lateral velocity of vehicle CG, m is the vehicle mass, $C_{\alpha,\mathrm{f}}$ and $C_{\alpha,\mathrm{r}}$ are the front and rear tire cornering stiffnesses, α_f and α_r are the front and rear tire side slip angles, u is the forward velocity, and γ is the yaw -rate. The yaw-rate state equation is:

$$\dot{v} = -\frac{aC_{\alpha,f}\alpha_f}{I} + \frac{bC_{\alpha,r}\alpha_r}{I} \tag{8}$$

where a is the distance between the front axle and the vehicle CG, b is the distance between the rear axle and the vehicle CG, and I is the yaw moment of inertia at the vehicle CG.

The front and rear tire side slip angle state equations were given by (Karkee and Steward 2008)

$$\dot{\alpha}_\mathrm{f} = \frac{v}{\sigma_\mathrm{f}} + \frac{a\gamma}{\sigma_\mathrm{f}} - \frac{u}{\sigma_\mathrm{f}}\delta - \frac{u}{\sigma_\mathrm{f}}\alpha_\mathrm{f} \tag{9}$$

and

$$\dot{\alpha}_\mathrm{r} = \frac{v}{\sigma_\mathrm{r}} - \frac{b\gamma}{\sigma_\mathrm{r}} - \frac{u}{\sigma_r}\alpha_\mathrm{r} \tag{10}$$

where σ_f and σ_r are the front and rear tire relaxation lengths, and δ is the front steering angle.

The vehicle trajectory was calculated using the two equations:

$$\dot{x} = u\cos\varphi - v\sin\varphi \tag{11}$$

$$\dot{y} = u\sin\varphi + v\cos\varphi \tag{12}$$

where (x,y) is the vehicle CG position in the world coordinate system.

The four-wheel dynamic model was developed using SimMechanics instead of deriving differential equations to represent the system. Using the model block library and the simulation environment, the three chassis models were swapped in and out alternatively to observe the simulation speed with each chassis model.

A global positioning system (GPS) simulator was included in the vehicle model block, which converted the vehicle position into latitude and longitude format. The vehicle position, attitude and yaw-rate data produced by the model simulator were next sent to the interface software through a TCP/IP socket. An interface to CAN called TCP/IP socket-to-CAN or the IP2CAN gateway was developed to communicate with the CAN-based hardware. The IP2CAN gateway managed the communication between the dynamic model and the hardware. The gateway received the vehicle position, attitude and yaw velocity information from the dynamic model simulator, converted it to the format that navigation hardware understands, and sent it out to the controller hardware over a CAN bus. The gateway was also responsible to receive the off-tracking and heading errors from the navigation controller hardware, convert the messages into the form the

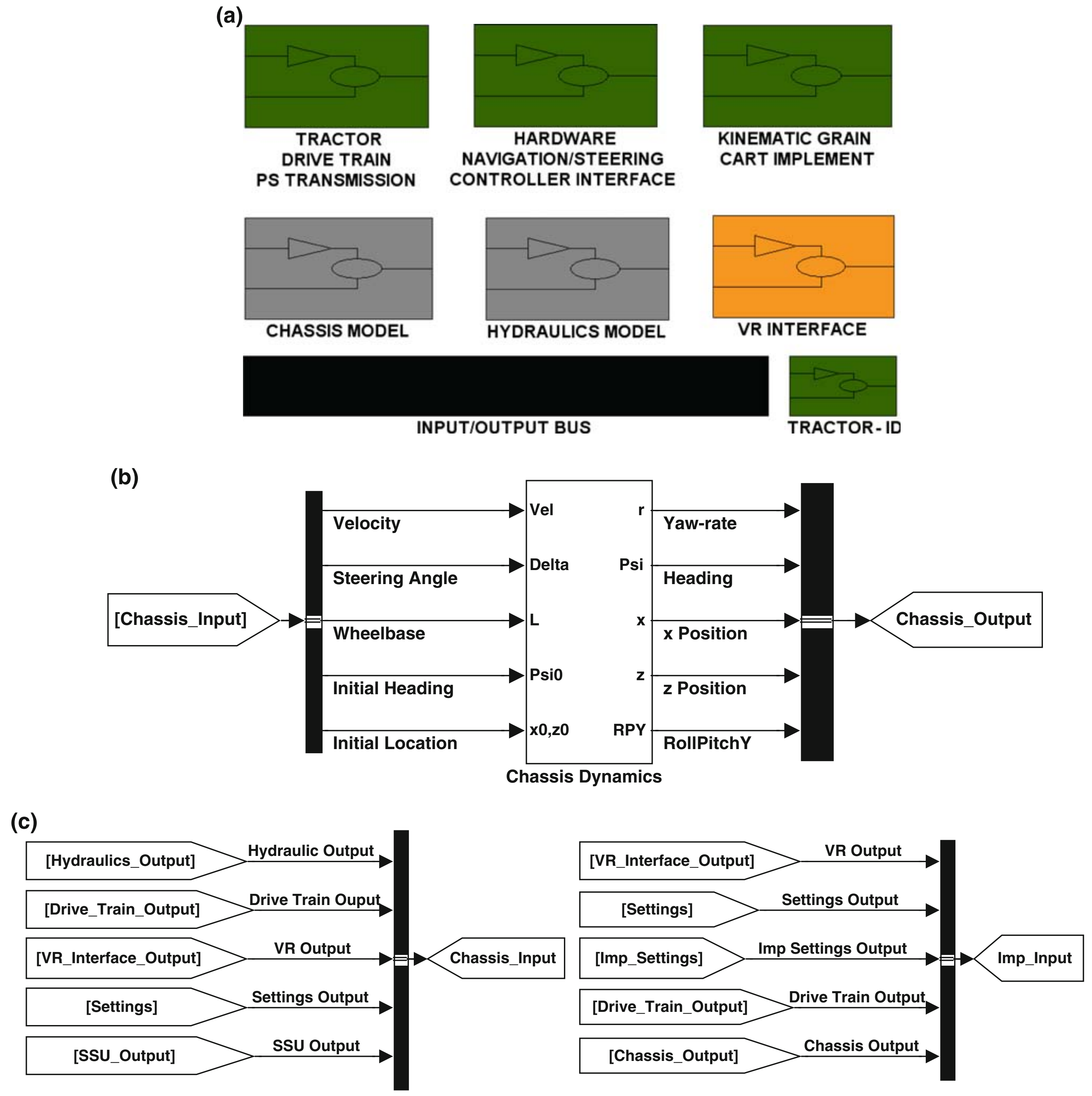


Fig. 10 Controller-hardware-in-the-loop simulation of an agricultural tractor, **a** Top level view of a Simulink model with interfaces to the controller hardware and steering unit, **b** chassis subsystem input and output signals and **c** interconnectivity represented by input/output bus

vehicle dynamics simulator understands and sent them out to the dynamic simulator over the TCP/IP socket. The gateway program implemented a state logic structure to produce the required CAN messages to enable automatic steering function on the navigation controller hardware. A number of additional messages were necessary to establish a connection and interact with the navigation controller hardware through a simulated GPS receiver. If all the required messages were present in the CAN network, the navigation controller enabled the automatic steering functionality and started generating navigation error messages (off-tracking error and heading error). These errors were calculated and broadcasted over the CAN bus once for each set of position and attitude messages received by the navigation controller. The steering controller unit read in and parsed these errors and applied them to generate the proper steering angle based on the controller implemented in the hardware.

The IP2CAN gateway was also responsible for reading in the steering angle from the hardware steering controller and sending that over to the model, which was used as an input steering angle to the vehicle dynamics. A hardware data acquisition system was developed to acquire the steering angle signal. The vehicle model simulator, on the other end, received the steering angle and other input signals and applied them to the vehicle system model, thus completing the simulation loop.

3.2.1 Discussion

The CHIL simulation was successfully implemented using the distributed architecture and modular dynamics modeling and simulation environment. The simulation demonstrated that the architecture can be used to perform various controller and operator hardware in the loop real-time VR-based simulations (Fig. 10c), which will be valuable to interactively test the performance of newly developed controllers for various off-road vehicle subsystems in various field conditions that may be very difficult or expensive to find in the real world. The VR visualization included field boundaries and the predefined path that an autonomous off-road vehicle is going to follow. The actual path followed by the vehicle can also be inserted. These path lines enabled a user to visually evaluate the performance of an autonomous vehicle guidance algorithm.

As mentioned before, the CHIL simulation was performed with tractor chassis models of varying fidelity and complexity. The models used were a kinematic model, a two-wheel dynamic model and a four-wheel dynamic model. The kinematic model and the four-wheel dynamic model were, respectively the simplest and the most complex models among the three models. As the model complexity increased, the computational time for each simulation cycle was increased substantially (Fig. 11). A desktop computer (XPS 400, Dell, Round Rock, TX) with Pentium D, 3.2 GHz, microprocessor running with 2 GB system memory and running Windows XP took approximately 100 times longer to finish a single simulation cycle of the four-wheel dynamic chassis model when compared to that of the kinematic model.

The comparison helped to identify an appropriate chassis model to achieve a real-time simulation with the available computational resources. The Dell XPS 400 desktop computer achieved real-time simulation goal for all three chassis models when the other subsystems used in the vehicle system model were computationally light (Table 1). In this case, the four-wheel dynamic chassis model was used in the simulation. However, when realistic steering valve dynamics were added to the system, the real-time simulation goal was not achieved with the four-wheel dynamic chassis model. Because the steering valve dynamics desired a very small simulation time step, the computational power available in the desktop system did not meet the real-time simulation goal. A bicycle dynamic model was the choice in this case to achieve the real-time simulation.

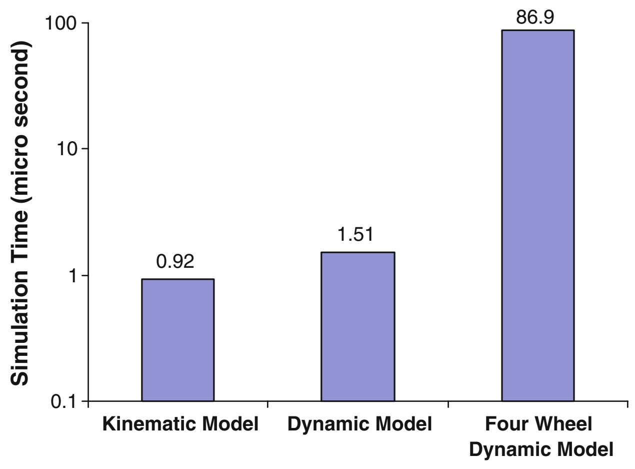


Fig. 11 Computational time for each simulation cycle of different tractor chassis model. The simulation time of the four-wheel dynamic model was substantially higher than that of the kinematic and dynamic models

4 Conclusions

Real-time simulation has become an essential element of VR-based simulation architecture with operator and hardware in the loop simulation. A flexible, scalable, and extendable off-road vehicle modeling, simulation and visualization architecture was developed for operator-and-hardware-in-the-loop off-road vehicle system dynamics models. The distributed architecture and multi-rate modeling technique were used to enhance the computational speed of the dynamic simulation with visualization in an immersive VR environment. The simulation architecture

Table 1 Assessment of the real-time potential of different chassis models with and without a steering valve dynamics model

Chassis model	Integration time step		Real time
	t_d (μs)	t_a (μs)	
Kinematic model	1,000	0.9	Yes
Dynamic model	1,000	1.5	Yes
Four-wheel dynamic model	1,000	86.9	Yes
Dynamic model with steering valve dynamics	10	6.1	Yes
Four-wheel dynamic model with steering valve dynamics	10	92.2	No

was designed to handle diverse interfacing hardware and software platforms. The architecture also provided a generic and cleaner modeling, simulation and visualization environment. A plug and play modeling environment saved considerable set-up time between different simulation runs. Model block libraries were also developed for various off-road vehicle systems, which allowed users to select models with appropriate complexity to achieve real-time simulation with the available computations resources. Moreover, the case studies presented in this paper demonstrated the practicality, flexibility and versatility of the real-time VR-based hardware-and-operator-in-the-loop simulation architecture, which will be helpful in interactive product design evaluation and modification using virtual prototypes.

Models of different vehicle subsystems such as chassis, power train and guidance controller were developed and added to a model block library. The architecture allowed users to develop their own models and add to the library without much difficulty. Various blocks from the library can easily be selected to build a new vehicle system model. The blocks can be swapped in and out to a vehicle system model so that altering a vehicle system model becomes highly flexible. Some of the complex models were developed with multi-rate simulation capability. This distributed real-time VR-based simulation system can effectively be used in virtual prototyping of off-road vehicles and controllers, which can facilitate:

- Reducing number of product development and test cycles,
- Virtual testing of various controller hardware and software,
- Reducing risk associated with extended field tests/ experiments particularly in difficult terrains,
- Reducing post processing and communication effort on modeling and simulation, and
- Developing methodologies for understanding customer behavior and perceptions.

Acknowledgments This research of the Iowa Agriculture and Home Economics Experiment Station, Ames, Iowa, Project No. 3612, was supported by Hatch Act and State of Iowa funds. The authors would like to thank Deere & Co. for their technical and financial support.

References

Antonya C, Talaba D (2007) Design evaluation and modification of mechanical systems in virtual environments. Virtual Real 11:275–285

Bosch R (1991) CAN specification version 2.0. Rober Bousch GmbH, Postfach 300240, D-7000 Stuttgart 30

Castillo-Effen MW, Castillo C, Moreno WA, Valavanis KP (2005) Modeling and visualization of multiple autonomous heterogeneous vehicles. In: Proceedings of the IEEE international conference on systems, man and cybernetics. The Hague, the Netherlands, 10–12 Oct 2005

Coutee AS, Bras B (2002) Collision detection for virtual objects in a haptic assembly and disassembly simulation environment. In: Proceedings of ASME design engineering technical conferences and computer and information in engineering conference, Montreal, QC, Canada

Cremer J, Kearney J, Papelis Y (1996) Driving simulation: challenges for VR technology. IEEE Comput Graph Appl 16(5):16–20

Cuadrado J, Gonzalez M, Gutierrez R, Naya MA (2004) Real time MBS formulations: towards virtual engineering. In: Product engineering, eco-design technologies and green energies. Springer, Heidelberg, pp 253–272

de Sa AG, Zachmann G (1999) Virtual reality as a tool for verification of assembly and maintenance processes. Comput Graph (Pergamon) 23:389–403

Eberhard P, Li Z (2006) Virtual reality simulation of multibody systems. In: Proceedings of EUROMECH colloquium 476, Ferrol

Fales R, Spencer E, Chipperfield K, Wagner F, Kelkar A (2005) Modeling and control of a wheel loader with a human-in-the-loop assessment using virtual reality. J Dyn Syst Meas Control 127:415–423

Fritzson P, Bunus P (2002) Modelica, a general object-oriented language for continuous and discrete-event system modeling and simulation. In: Proceedings of the 35th annual simulation symposium, San Diego, CA, April 2002

Glinsky E, Wainer G (2002) Definition of real-time simulation in the CD++ toolkit. In: Proceedings of the summer computer simulation conference, 14–18 July, San Diego, CA

Gracanin D, Matijasevic M, Tsourveloudis NC, Valavanis KP (1999) Virtual reality testbed for mobile robots. In: Proceedings of the IEEE international symposium on industrial electronics. Bled, Slovenia, 12–16 July 1999

Howard BM, Vance M (2007) Desktop haptic virtual assembly using physically based modelling. Virtual Real 11:207–215

Jayaram S, Jayaram U, Wang Y, Tirumali H, Lyons K, Hart P (1999) VADE: a virtual assembly design environment. IEEE Comput Graph Appl 19:44–50

Johnson TC, Vance JM (2001) The use of the voxmap pointshell method of collision detection in virtual assembly methods planning. In: Proceedings of the ASME design engineering technical conference, Pittsburgh, PA

Kang HS, Abdul Jalil MK, Mailah M (2004) A PC-based driving simulator using virtual reality technology. In: Proceedings of ACM SIGGRAPH international conference on virtual reality continuum and its applications in industry, Singapore, 16–18 June 2004

Karkee M, Steward BL (2008) Open and closed loop system characteristics of a tractor and an implement dynamic model. ASABE Paper No. 084761. St. Joseph, ASABE, MI

Kim CE, Vance JM (2003) Using VPS (Voxmap PointShell) as the basis for interaction in a virtual assembly environment. In: SME design engineering technical conferences and computers and information in engineering conference, Chicago, IL, United States

Lin Q, Kuo C (1997) Virtual tele-operation of underwater robots. In: Proceedings of the IEEE international conference on robotics and automation, Albuquerque, New Mexico, 21–27 April 1997

Livani MA, Kaiser J, Jia W (1999) Scheduling hard and soft real-time communication in a controller area network. Control Eng Pract 7:1515–1523

Pazul K (2009) Controller area network (CAN) basics. Microchip technology, Inc. AN713. Available at: http://www.cl.cam.ac.uk/research/srg/HAN/Lambda/webdocs/an713.pdf. Accessed: 16 June 2009

Sastry L, Boyd DRS (1998) Virtual environments for engineering applications. Virtual Real 3:235–244

Savall J, Borro D, Gil JJ, Matey L (2002) Description of a haptic system for virtual maintainability in aeronautics. In: Proceedings of 2002 IEEE/RSJ international conference on intelligent robots and systems, Lausanne, Switzerland

Schulz M, Reuding T, Ertl T (1998) Analyzing engineering simulations in a virtual environment. IEEE Comput Graph Appl 18(6):46–52

Stankovic J (1988) Misconceptions about real time computing: a serious problem for next generation systems. IEEE Computer 21(10):10–19

Wan H, Gao S, Peng Q, Dai G, Zhang F (2004) MIVAS: a multi-modal immersive virtual assembly system. In: Proceedings of the ASME design engineering technical conference, Salt Lake City, UT

Zhu Z, Gao S, Wan H, Luo Y, Yang W (2004) Grasp identification and multi-finger haptic feedback for virtual assembly. In: Proceedings of the ASME design engineering technical conference, Salt Lake City, UT