Int J Adv Manuf Technol (2009) 43:620–630
DOI 10.1007/s00170-008-1736-6

ORIGINAL ARTICLE

Dynamic simulation of virtual prototypes in immersive environment

Fabio Bruno · Francesco Caruso · Kezhun Li · Alessandro Milite · Maurizio Muzzupappa

Received: 26 February 2008 / Accepted: 28 August 2008 / Published online: 19 September 2008

Abstract Virtual reality (VR) became a very common mean during the development of the industrial products. One of the main applications of VR in the industrial field is the validation of virtual prototypes (VP). A virtual prototype should be able to reproduce, as realistically as possible, the behaviour of the product from any point of view. In this paper we propose an unexploited approach to the simulation of a VP in VR. A high level software library for the inter-process communication has been developed to let the multi-body solver communicate with the VR environment. Such approach allows the designer to use different software frameworks for the simulation and the visualisation. The test case provided regards an excavator machine. It is possible to simulate the action of the actuators to move arms and bucket, and also perform visibility analyses discovering the viewing volume of the operator.

Keywords Virtual reality · Numerical analysis · Multi-body analysis · Visibility analysis

1 Introduction

The main target of the virtual reality (VR) in industry is the preview and the discovery of any possible design flaws before the realisation of physical prototypes. The aid provided by VR is noticeable, since the user can interact with the virtual prototype in a very natural way. The earliest industrial VR applications merely provided the visualisation of the virtual prototype (VP) in VR, in order to perform an aesthetical validation of the product. VP visualisation in VR is crucial to exploit the overall appearance of the product, but it can also be useful in order to perform some analyses like parts reachability, cabling, design, assembly, validation, etc.

More recent applications, on the other hand, try to simulate the behaviour of the industrial products in VR, i.e. they try to simulate the virtual prototype. In this way it is possible to use VR not only for aesthetical purposes, but also to discover any functional flaw. An overview of the possible utilisation of VR in the industrial field has been presented in [1], in which a taxonomy of all the employments of VR in industry is presented.

However, as depicted in the previously mentioned work, there is a lack of software tools able to support designers in the development of interactive and fully functional virtual prototypes. This lack is probably one of the main obstacles in the diffusion of VR techniques for the product behaviour simulation. Engineers in fact use specialised simulation software to design the industrial product and, at the moment it is not possible to evaluate the models developed in these simulation packages directly in VR. As pointed out in [2] it is necessary to work out a specific solution for the several problems occurring during the integration between VR and the other applications of the product development process (PDP), including the simulation. One of the few approaches devoted to the integration of simulation packages and VR has been presented by Kirner et al. [3] [4] who developed the VR-SIM, an object oriented C++ library, able to incorporate a real-time systems (RTS) simulator with VR technologies. The use of VR-SIM includes the creation of the system to be validated and of a virtual environment related to this system.

F. Bruno (✉) · F. Caruso · M. Muzzupappa
Department of Mechanical Engineering, University of Calabria,
87030 Arcavacata of Rende (CS), Italy
e-mail: f.bruno@unical.it

K. Li
Digital Prototyping and Simulation, Case New Holland,
Lake Forest, IL, USA

A. Milite
VR&CAE, Elasis ScPA,
Pomigliano d'Arco (NA), Italy

The case-study described in [3] is a robot arm coupled to an automatic transport-belt, used in a factory for piling up boxes. This work demonstrates that VR technology is applicable and useful to support RTS simulations, as a form used to evaluate the correctness of such systems. But the VR-SIM is a tool addressed to software engineers responsible for the development of real-time, process control systems; it requires code development for the implementation of the virtual product, and it is not suitable to be used by industrial or mechanical engineers in the PDP.

A similar work has been presented by Sánchez et al. [5] who developed the Easy Java Simulation (Ejs) [6], a software tool designed to create interactive simulations in Java using models created with Simulink. Basically, Ejs is able to communicate with the Simulink model and to tightly control its execution. The communication takes place through a DLL library written in C and some utility Java classes. The main advantage of this work is that Ejs creates Java applets that are independent and multi-platform, and can be visualised using any Web browser, can read data across the net and be controlled using scripts from within html pages. Ejs can also be used with Java 3D to create interactive virtual products in 3D, but it has been conceived mainly for educational purposes and it cannot be efficiently integrated into a PDP because Java 3D is not suitable for the visualisation of complex models. In [8], an innovative framework for numeric co-simulation and three-dimensional visualisation has been presented. This last paper is addressed, above all, to the simulation of mechatronic products, in which different physical domains have to be simulated. It provides an approach to achieve a co-simulation using several heterogeneous solvers and, at the same time, visualise the results in a three-dimensional environment. In [9], an environment for the behavioural simulation of CAD assemblies is presented. The global model is formed by several component objects. Each component object has a behaviour (simulation model) and a form (CAD model). To achieve the composition of behavioural models, the authors introduce a port-based modelling paradigm to describe the interactions among the objects. The framework extracts data directly from the CAD model to compute physical properties and define the constraints of the assemblies, and then perform a numerical simulation to predict the behaviour, but no visualisation of the CAD model occurs. In [15] the Daimler–Chrysler research group presented a research work for the simulation of one-dimensional flexible part. The solver is integrated in a virtual reality software for use in applications such as cable routing and assembly simulation.

In [10] an innovative fork-lift simulator, suited for training in industrial environments is presented. This simulator aims to improve skills in driving and handling materials using a fork-lift. The system can reply inertial feedback on the operator and allows the user to control all the tasks of a fork-lift, such as driving and handling materials. The numerical model is modelled and simulated using the ODE [11] C++ library, hence any modification to the model implies the modification of the application code.

The use of VR together with simulation has proven to be useful also for the creation of a virtual simulation environment for manufacturing tasks [13]. In this work, a training operator environment for a numerical control milling machine is presented. The user can learn how to control the CNC machine in a safe way, and it is even possible to use a remote location through the web.

The use of VR and simulation for training tasks has been also tried in the medical field [14].

In [12] the overall architecture of the immersive virtual product design framework is presented. This framework is still at an infancy stage. Theoretically, it enables users to navigate and interact with 3D peripherals with the display system. Participants may passively view the virtual world revolving around them while others control navigation and simulations (CFD and FEA). They may also interact with other participants, e.g. to pass an object from one embodied manikin to another. The authors do not explain how to perform numerical simulations within the virtual environment, in fact they consider this topic an unsolved and challenging problem. Another example of virtual prototype simulation framework is presented in [16], in which a virtual prototype of a hybrid electric vehicle is created within the virtual test bed (VTB) environment, which has been developed for modeling, simulation, analysis and virtual prototyping of large-scale multi-technical dynamic systems. VTB allows each component of a large-scale multi-technical system to be described in the most appropriate language (e.g., SPICE for electronic components, ACSL, advanced continuous simulation language, for dynamic systems, and SABER for power electronic circuits). Further, it provides advanced visualisation of simulation results, including full-motion animation of mechanical components, and imaginative mappings of computed results onto the system topology.

None of the previously mentioned papers mentions the simultaneous use of VR and numerical simulations for validation tasks of industrial products.

In our opinion, the union of both these techniques, can lead to great improvements in this direction. Further, it enhances the concept of VR, commonly used for visualisation purposes, with the integration of numerical simulations. Moreover, in almost all the previously mentioned work, a unique software environment is employed for both simulation and visualisation. Our approach, instead, is based on the inter-process communication among different software modules, and relies on a middleware for the software communication. This approach make possible to freely choose the software to be employed, either for the

simulation and either for the visualisation. In this way engineers can use their favourite software during the product development phase.

2 Dynamic simulations in VR

The aim of this work is the development of an environment for the evaluation and the validation of both aesthetical and functional features of the industrial products in VR. VR environments are certainly one of the best solutions for the validation of aesthetical features. Using VR applications, the user is fully immersed in a three-dimensional synthesised environment and can perceive the appearance of the virtual objects as if they were real. VR applications, in fact, provide a high quality, immersive visual representation of virtual prototypes, and the user can easily evaluate aesthetical qualities and/or discover any styling defect. However, a VR software is conceived mainly to reproduce only the visual appearance, i.e. to be a kind of high-end visualisation system. For this reason, in fact, the term *visual reality* was neologised, in order to more appropriately describe the VR environment. However, at the moment, the validation of the functional features is not as easy to achieve as the aesthetical one. Some of the modern VR software allow one to define a behaviour for each object in the virtual world, providing built-in physics simulators, oriented, above all, to video-games, in order to simulate mainly collisions and rigid bodies dynamics. Such simulators can provide fast results, but they are not as accurate and flexible as the simulation software used by engineers. As a consequence, the behaviour we obtain is an animation, and not a simulation and it cannot be considered as a robust validation tool. It is not possible, for example, to validate the correct dimensioning of an actuator, because the dynamics is not taken into account. Furthermore, a kind of "translation" of the behaviour from the simulation software into the VR software is necessary. The core of our work is the idea of creating a VR environment, able to use the results of the simulators which are commonly used by engineers. To achieve such goal, VR software and simulation software have to interact in order to obtain a simulation in VR. The numerical model used for the VR simulation is exactly the model created by the engineer during the design phase.

The idea is to use the VR environment as a back-end for simulations. The simulation software computes the displacements and the rotations of the mechanical parts, and sends the results to the visualisation environment, which moves the parts in accord with the simulation results. Further, any control logic is simulated by the simulation environment.

Our approach offers the following advantages:

- The integration with numerical solvers allows one to simulate a very wide range of physical domains, while in the previously mentioned approaches (re-creation of the animation within the VR software) only kinematics can be simulated.
- It allows the hardware in loop (HIL) technique, since some numerical simulation software, e.g.: Simulink, can perform an HIL simulation. This is a noticeable advantage, because it is possible to test the real control logic hardware (Fig. 1).
- There is no need to convert and re-describe the control logic behaviour in the VR software, thus allowing us to save time and avoid design bugs due to errors or imprecision in the conversion.

The integration of the VR environment with a simulation environment enhances the VR and makes it closer to the real environment. As a consequence, VP simulation improves and become more effective, because it is closer to the simulation with physical prototypes. Obviously, in order to improve the overall availability, the quality of the user interaction with the virtual prototype is crucially important. For this reason the interface of product should be reproduced as realistically as possible, and the user should be able to interact with it in the most natural way. As regards the interaction, there are two possible scenarios. The first case is when the interface to reproduce is physically available (Fig. 2) or it can be physically simulated, e.g. a joystick used to simulate a command lever. In this scenario, the user sends the commands directly to the simulation environment. The latter processes the user's actions and computes the positions of the VP parts, in keeping with the mathematical model implemented by the designer, or, in other words, it computes the current configuration of every degree of freedom (DOF) of the assembly. Subsequently, the simulation software sends data about every DOF in the 3D assembly to the VR software that updates the status of the 3D models. So the VP is continuously updated by the simulation model.

The other scenario (Fig. 3) is when the interface has to be virtually simulated. In this case, it is common practice to use VR specialised devices, like data-gloves, tracking and haptic devices. The choice of the VR devices is a crucial and challenging issue, as described in [7]. In this scenario VR application recognises the user's generated commands and it sends these events to the simulation package that, as previously described, computes the new configuration of the model and consequently sends the displacements of each moving part to the virtual reality environment.

To achieve the aimed integration, a communication channel between the two environments is obviously necessary. In this channel, the VR software receives data from the simulations, and updates the configuration of the three-dimensional model.

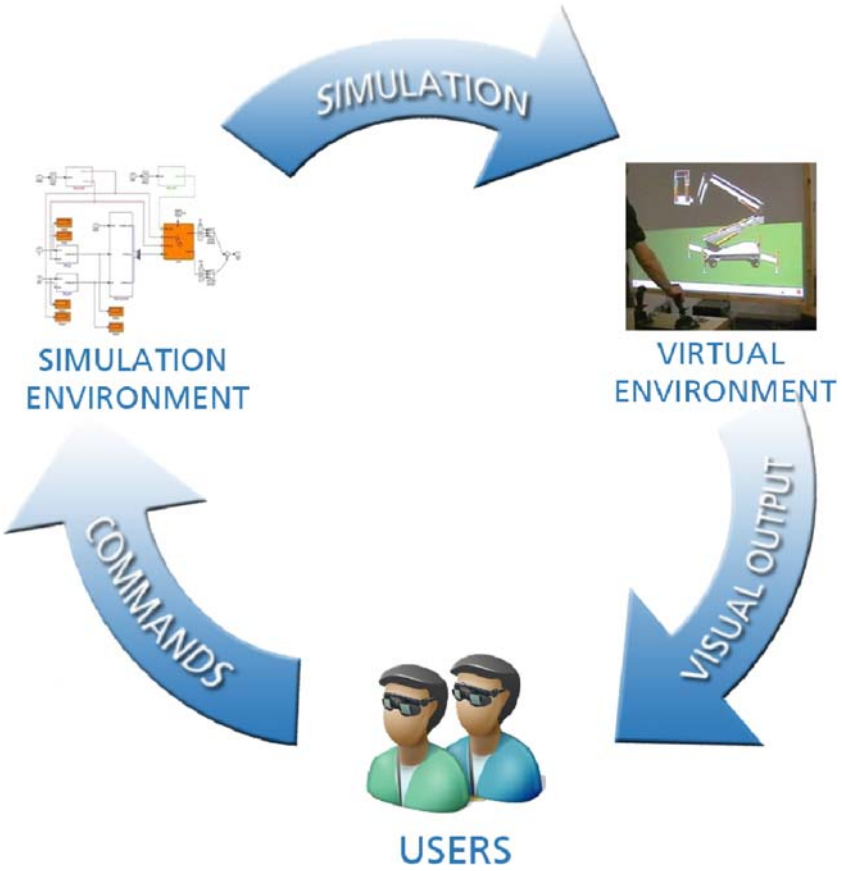

Fig. 1 A possible working loop of a dynamic simulation in VR

To achieve this target, it is obviously important to consider both the geometrical model and the behavioural model of the product and let them interact with each other. The geometric model is the three-dimensional representation of the product.

The behavioural model considers the dynamic features of the product, due to its mechanical and electronics properties. From a mathematical point of view, it is a differential equations system, but it is modelled and simulated using a Computer-Aided Control Engineering (CACE) software and/or a multi-body solver. Through this integration, the virtual environment becomes active and closer to the real environment. Virtual reality becomes an

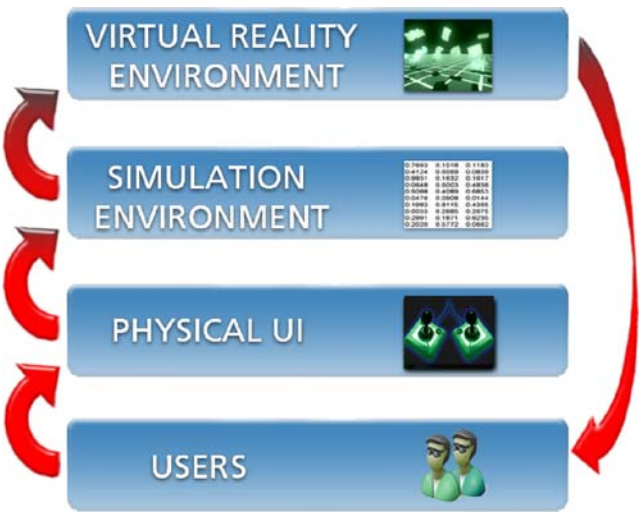

Fig. 2 A possible interaction scenario using physical devices for the user interface

Fig. 3 A possible interaction scenario using VR devices for the user interface

environment for the experimentation on virtual prototypes, which helps engineers to better understand the results of numerical simulations.

3 Development of the simulation framework

In order to achieve our purpose, a high level software library for the inter-process communication (IPC) has been developed. This library allows the simulation software to communicate with the VR environment (Fig. 4).

We have chosen Matlab/Simulink as a simulation software. This environment is almost a standard for the general purpose simulation. It is very diffused and versatile. Further, a lot of optional packages (called toolboxes) are present. These packages provide further sets of high level operations for a specific task. In this work the SimMechanism toolbox has been used as multi-body solver. With the SimMechanism toolbox it is possible to model and simulate a mechanic assembly, specifying properties for bodies and joints.

The software used as a virtual environment is Division MockUp by PTC. It provides several virtual environments visualisation and exploration functionalities and, further, every application can be customised. Division MockUp, in fact, provides an SDK for the plug-in development, and a scripting language to create virtual environments.

Virtual environments are re-created using a VDI file, i.e. a plain ASCII file containing the description of the virtual scene. It contains both the hierarchy of the assembly and the attributes of each part of the assembly. Division is made up by two software modules: a low-level core called dVS, and a high level interface, called dVISE. It is also possible to use the EC library, i.e. an API to dVISE, in order to easily achieve the following tasks:

- Define new action functions to customise the assemblies' behaviour;
- Create assemblies, cameras, reference points and annotations;
- Execute functions on the assembly's hierarchy.

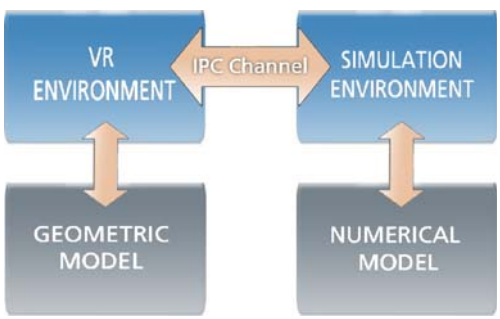

Fig. 4 Framework architecture

The communication between the two applications can be synchronous or asynchronous. The sender (i.e. Matlab/Simulink) sends data deriving from the simulation in asynchronous mode using non-blocking sockets, and does not take care of the reception of the visualisation environment. In this way, the simulator computes the simulation results also for the next simulation cycles, without stopping the computation.

On the other hand, Division adopts the synchronous mode communication, using the blocking sockets. It stops its execution until it gets the message from Matlab/Simulink. Having received the message, it moves the 3D model using the obtained information, and it sends the request for the new data. As it is easy to understand, a new thread takes care of the communication. In this way the user can still interact with the 3D scene (Fig. 5).

Resuming, Matlab/Simulink does not send new data computed for the next simulation cycles until a Division request occurs. Whereas Division stops the execution until it has received all the data from Matlab/Simulink, then it sends a data request to the simulator.

3.1 The SimLib library

In order to simplify the development of such applications, it is necessary to implement a high level communication library. This library should provide an easy to use IPC channel for the communication between the VR environment and the solver. Therefore, we implemented the SimLib library. It is a versatile library that can be easily adapted to all the possible test cases. SimLib uses TCP socket, therefore it is possible to run the simulator and the VR application on different machines.

The SimLib library is quite easy to use. It has few functions, implementing the code for the TCP/IP communication and synchronisation. Therefore, the developer must not take care of sockets and threads.

The functions of the library are:

- SimLib_Channel* **SIMServerOpen**(unsigned short port, int connections, u_long non_blocking).
- SimLib_Channel* **SIMClientOpen**(const char* host, unsigned
- int **SIMClose**(SimLib_Channel* s)
- SimLib_Channel* **SimLib_Accept**(SimLib_Channel* server)
- int **SimLib_synchroSend**(SimLib_Channel* to, SimLib_Data* r)
- int **SimLib_synchroReceive**(SimLib_Channel* s, SimLib_Data* r)
- void **SimLib_SendNextData**(SimLib_Channel* s)

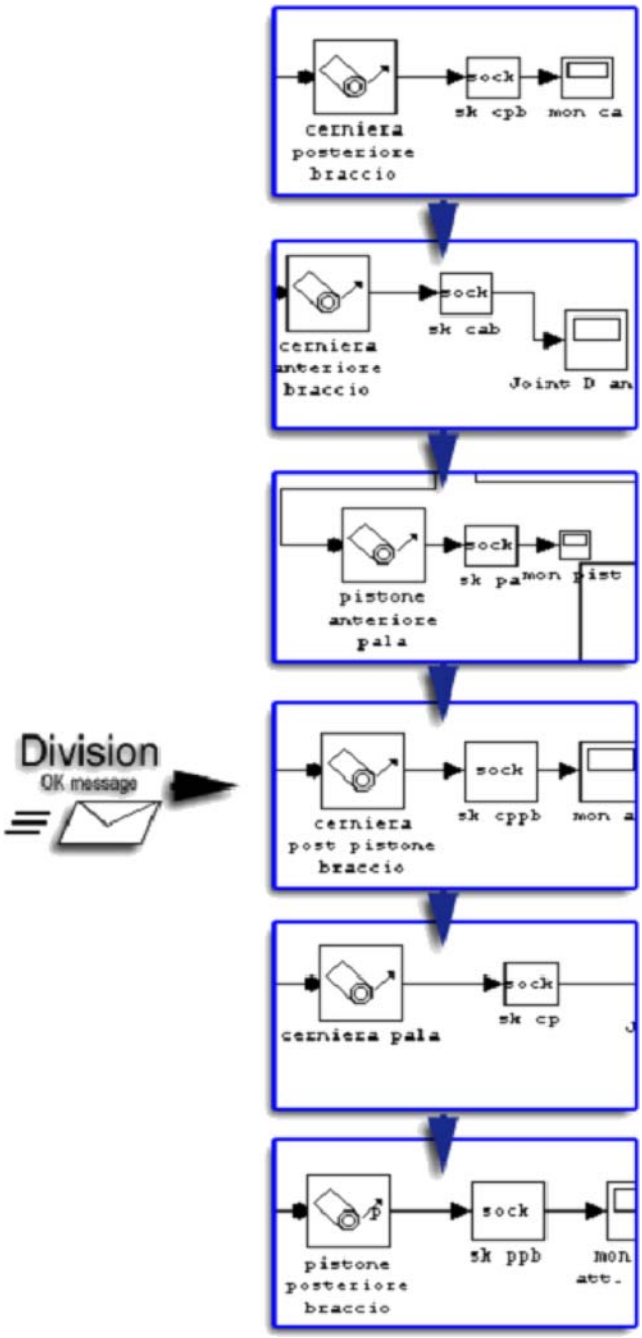

Fig. 5 Asynchronous receiving of the Division OK message

- void **SimLib_StartListener**(SimLib_Channel* server, void (*ptActionCB)(const SimLib_Data))
- void **SimLib_StopListener**()

The SIMServerOpen function creates a server communication channel listening on the specified port. It is possible to set it as a blocking or non-blocking communication channel.

The SIMClientOpen function creates a client communication channel, which attempts to connect to the server specified by the parameter. It is possible to set it as a blocking or non-blocking communication channel.

The SIMClose function closes and removes a communication channel from the memory.

SimLib_Accept function establishes a connection between the server specified as a parameter and the connecting client. It returns a new communication channel which will be used by the client for the communication.

The SimLib_synchroSend and Sim-Lib_synchroReceive functions send and receive data using a common protocol for the IPC. The developer must specify the SimLib_Data data structure used for the communication. After which, the structure is sent and received through the IPC channel. The data structure can also be very complex, but in this case the communication speed is slower. In our implementation, we defined the SimLib_Data structure as follows:

- typedef struct {
- char partName[128]; // name of the part to move
- int partId; // id of the part to move
- float tx; //x-axis translation value
- float ty; //y-axis translation value
- float tz; //z-axis translation value
- float rx; //x-axis rotation value
- float ry; //y-axis rotation value
- float rz; //z-axis rotation value
- } SimLib_Data;

The SimLib_StartListener and SimLib_StopListener functions respectively create and destroy the listener thread. The listener thread can be created by a server communication channel in order to handle the connection with a client. If there is a listener, the server communication channel must be in blocking mode. A listener thread calls another thread (called Answer thread) for each connecting client. The answer thread receives data from the client, and then calls a callback function (if specified) which can use data received from the client (Fig. 6).

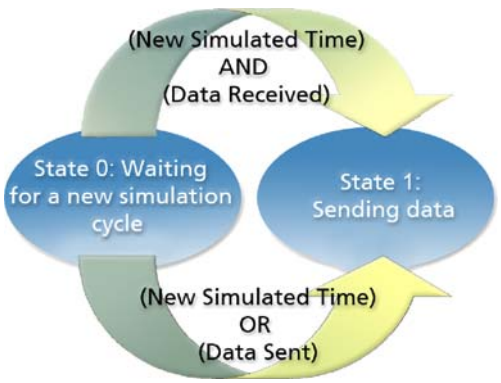

Fig. 6 Finite State Diagram of the overall behaviour

3.2 The Simulink S-Function for the communication

The Matlab/Simulink environment can be extended by the development of user defined S-Functions (). These ones can be used within a Simulink model as a conventional Simulink building block, with a user defined behaviour and set of actions. In our study, the S-Function is responsible for the communication between Matlab/Simulink and Division. As it is easy to understand, the communication uses the SimLib library and relies on the IPC channel provided by the library itself. The main task of this S-Function is the asynchronous data sending to Division. In other words, the S-Function sends simulation data to Division, without stopping the simulation. In order to obtain a consistent visualisation in fact, it is necessary to send all data of each time-step. Since in the model there is more than one part governed by the simulator, it is important not to send data of different time-steps to achieve correct visualisations. To achieve this target, we used the information from the Matlab ssGetT function, which returns the simulation time. As explained in the next image, once the OK message from Division is obtained, all the blocks send data in a sequential order.

The configuration of each communication block is quite easy. It is possible in fact, to set the network parameters and the part name via GUI as shown in Figs. 7, 8 and 9.

The parameters needed by each block are:

- IP and port of the host to which data are sent
- The name of the part which is governed by the block
- Number of the blocks in the Simulink model

3.3 The divison plug-in

The plug-ins are software tools used to link Division to customised applications.

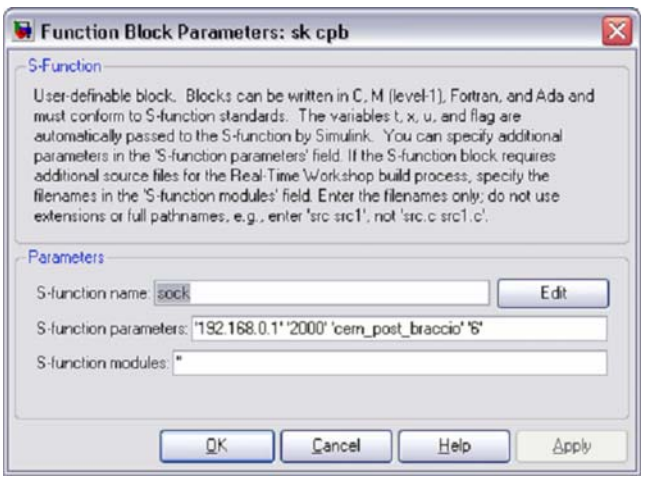

Fig. 7 GUI for the S-Function configuration

In our purpose, the plug-in has to read simulation data coming from Matlab, and move the parts of the assembly in accord with the received data. There is no bi-directional communication, since

Firstly, during the plug-in initialisation, a server communication channel is created. Then a new thread for the communication is created using a SimLib_listener. The call-back functions of the latter are *applyAction* and *isReceived*. The *applyAction* function implements the modification of the assembly in accord with received data, while the *isReceived* function sends the signal of end reception.

It is necessary to save the references to the assemblies to be moved in order to animate the model. These references in fact, are used by the applyAction call-back to animate the model. The _saveRef function has been implemented, in order to save the references to the assemblies into a hash table. It is important to save the reference to the assembly with the same name specified in the Matlab S-Function. As a matter of fact, the name of the assembly contained in the SimLib_Data sent by the S-Function is the key search within the hash table.

4 Test case

The test case regards the simulation and relative visualisation of a tractor. In particular we focused on the simulation of the movements of arm and bucket. This test case required six communication blocks in the Simulink model. These blocks send the orientation of:

- Lower hinge of the arm
- Lower hinge of the support cylinder of the arm
- Lower hinge of the bucket
- Lower hinge of the support cylinder of the bucket

And the translations of:

- Piston of the arm
- Piston of the bucket

These communication blocks are highlighted in the figure below.

Each of these communication blocks receives data concerning the position/orientation of a joint. The first three ports (starting from the upper side of the block) are used for the rotations in the three axes, and the last three ports are used for the translations on the three axes.

As regards the 3D model of the tractor, it is necessary to find and to register the assembly corresponding to the Simulink blocks. Once the part has been found, it must be renamed using the same name as the parameter in the Simulink communication block, as previously described. In the 3D model, there is a hierarchy of the several parts forming the assembly. In other words, it is possible to

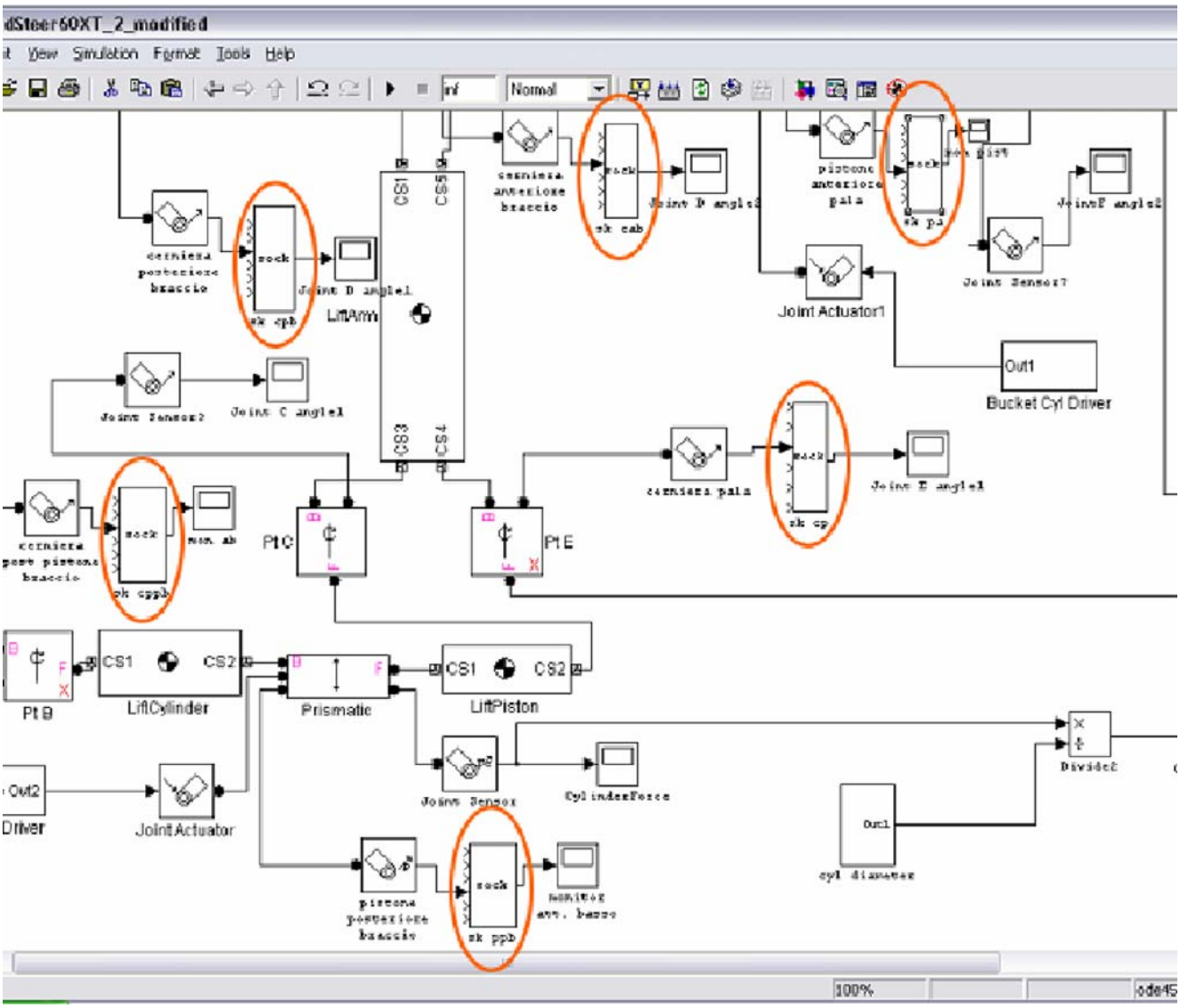

Fig. 8 Building block for data communication within the Simulink model

specify a parent/child relationship between two parts. Once this relationship has been defined, the child will follow the same movements as its parent. Therefore, if one moves a part, all the children of this part will follow the same movements. This feature is useful for our task, because once we have defined the relationship between the parts, it is sufficient to move the parent of a kinematic chain to obtain the movement of all its children. In the 3D model of the tractor e.g., a parent/child relationship between cylinder and piston of a hydraulic actuator has been defined.

With the use of a common joystick, it is possible to govern the actuators, like in a real tractor. The Matlab/Simulink environment in fact, provides the interface with such devices via virtual reality toolbox. During the simulation, it is possible to notice a delay between the command and the response. This delay is due only to the complexity of the 3D model. Division, in fact, has to complete the rendering for each position. After having simplified the 3D model, and deleted all the parts not affected by the simulation, the response of the system was really high.

5 Visibility analysis

As a further investigation, in this test case, we have implemented an application for visibility analysis. Quite often, in fact, visibility requirements are unavoidable constraints for the final product.

As regards tractors, the ISO 5721 normative describes different visibility requirements for different types of agricultural tractors. In particular, there are three main visibility concerns in most agricultural tractors: visibility

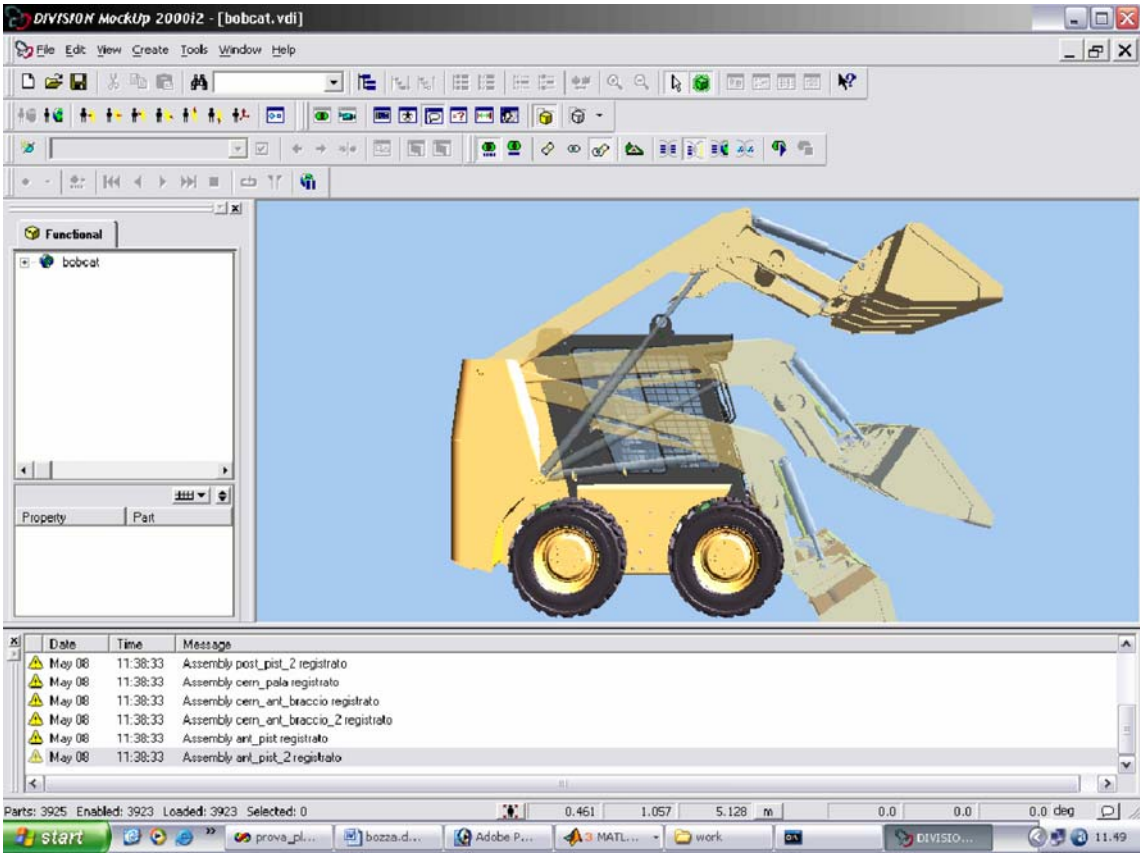

Fig. 9 The division environment during the simulation

forward, including the view over the hood, the view forward and down to the ground and tyres, and the view forward and up; visibility to the side, including the view over side consoles and fenders; and visibility rearward, especially the view to the drawbar, pickup hitch, and PTO.

Tractors that have to be roaded, especially small and medium-size tractors, must often meet minimum forward over-hood visibility requirements. Market standards and requirements should be consulted to ensure conformity with such requirements.

Large articulated and non-articulated tractors used primarily for drawbar-type work may have less demanding visibility requirements for certain areas of the visual field, such as the view to the hitch and to the side.

The visibility requirements for row-crop tractors may be the greatest. High importance should be given to the view

Fig. 10 The operator's view inside the cabin

Fig. 11 Visualisation of the viewing cone of the operator

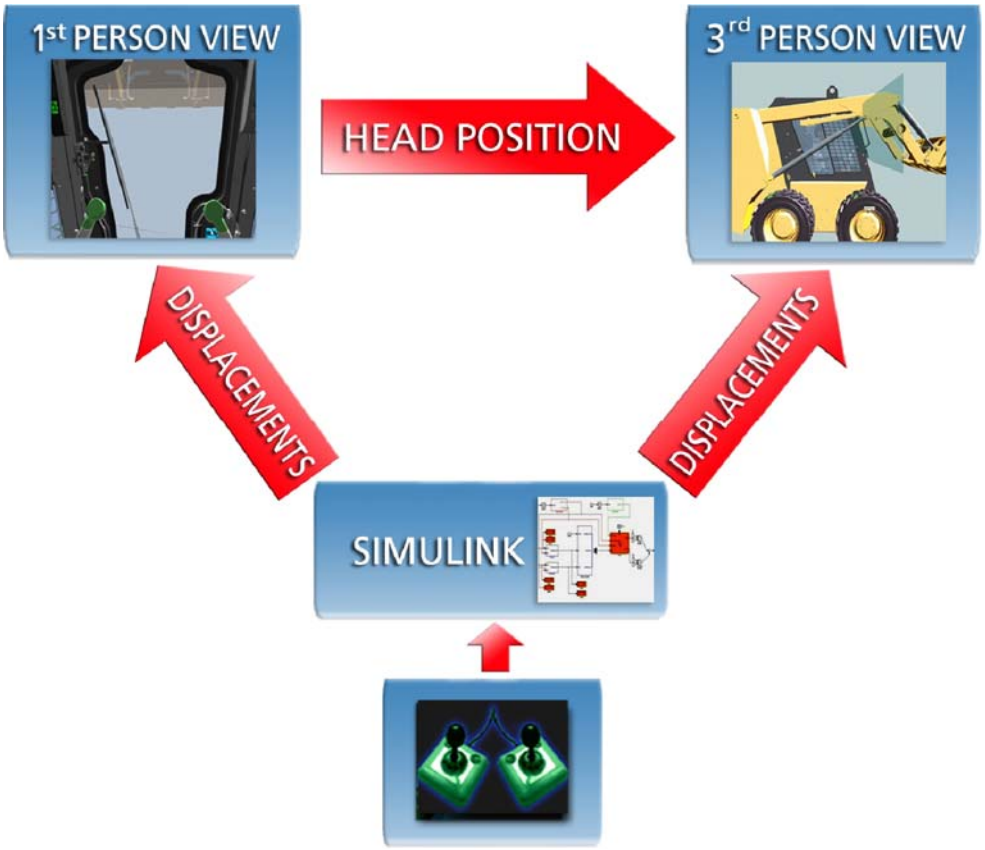

Fig. 12 Visualisation of both the first person and third person views and the simulation model

down to the ground inside and behind the front tyres, the view over the side and the view out the back down to the hitch.

Virtual reality is certainly a powerful mean in order to discover any possible design flaw as regards visibility. Therefore, we tried to exploit VR capabilities also in the visibility analysis field.

This application allows us to simultaneously run two executions of dv/Mockup. One execution is directly connected to the input devices and it runs the simulation in first person. The operator wears the head mounted display and controls the virtual tractor through a pair of joysticks (Figs. 10, 11 and 12).

The second execution allows engineers to view in third person what happens in the simulation. And, in particular, they may also analyse the viewing cone of the operator that controls the tractor.

6 Conclusions

This paper describes an experimental environment for the validation of functional virtual prototypes. Two environments are needed in order to achieve this task: a virtual reality environment and a numerical simulation one. The link between them is an inter-process communication library developed for this purpose. The idea is to enrich the standard VR environment with numerical simulations. In this way, VR becomes closer to a real environment, because the overall behaviour can be simulated and predicted by the numerical simulation software and then visualised in VR.

In the test case we used Division as a VR environment and Matlab/Simulink as a numerical solver, but the methodology we employed can be used with other software environments, even if the development of the interfaces between each software and the IPC library is needed. The effort for the interface development represents the main drawback of the proposed methodology.

In the test-case, the real interface, formed by two command levers, has been replicated by two joysticks. The numerical simulator computes the displacements of the parts composing the assembly. Certainly, the simulation can be more complex, and it is possible to increase the overall realism by enriching the simulation model, e.g. adding the collision detection simulation to prevent the penetration of the parts.

Another drawback is represented by the part names. As a matter of fact, in order to animate the model, the simulator sends the name of the part together with its displacement to the VR software. Therefore, there must be consistency between part names of the 3D assembly and part names in the simulator, but. the achievement of this consistency can be a time-consuming and error-prone operation.

Moreover, in the test case, a new approach to the visibility analysis is proposed, which is a quite important task for almost all kinds of machines. With the help of this tool, designers can easily discover all the non-visible areas during the validation.

Finally, as it is easy to understand, the validation environment can be easily converted and used within a training operator environment, since the user can learn the effects of the commands in the virtual environment.

References

1. Jimeno A, Puerta A (2006) State of the art of the virtual reality applied to design and manufacturing processes. Int J Adv Manuf Technol doi:10.1007/s00170-006-0534-2
2. Barbieri L, Bruno F, Caruso F, Muzzupappa M (2007) "Innovative integration techniques between Virtual Reality systems and CAx tools". Int J Adv Manuf Technol doi:10.1007/s00170-007-1160-3
3. Kirner TG, Kirner C (2005) Simulation of real-time systems: an object-oriented approach supported by a virtual reality-based tool. In Proceedings of 38th Annual Simulation Symposium, 4–6 April, San Diego, California
4. Gimenez AM, Kirner TG (1997) Validation of real-time systems using a virtual reality simulation tool, Systems, Man, and Cybernetics, ''Computational Cybernetics and Simulation'., 1997 IEEE International Conference on Volume 2, 12–15 Oct. 1997 Page(s):1586–1591 vol.2 doi:10.1109/ICSMC.1997.638225
5. Sánchez J, Esquembre F, Martín C, Dormido S, Dormido-Canto S, Canto RD, Pastor R, Urquía A (2005) Easy java simulations: An open-source tool to develop interactive virtual laboratories using MATLAB/Simulink. Int J Eng Educ 21(5):798–813
6. http://fem.um.es/Ejs
7. Stefani O, Karaseitanidis I (2004) Designing 3D Input Devices for Immersive Environments, 2004 IEEE International Conference on Systems, Man and Cybernetics
8. Bruno F, Caruso F, Muzzupappa M, Stork A (2007) An experimental environment for the runtime communication among different solvers and visualisation modules, Proceedings of 19th European Modeling and Simulation Symposium, Genova, October 2007
9. Sinha R, Paredis C, Koshla P (2000) Integration of Mechanical CAD and Behavioural Modeling, 2000 IEEE/ACM International Workshop on Behavioural Modeling and Simulation (BMAS'00) p. 31
10. Bergamasco M, Perotti S, Avizzano CA, Angerilli M, Carrozzino M, Facenza G et al (2005) Fork-lift truck simulator for training in industrial Environment, Conference on Emerging Technologies and Factory Automation doi:10.1109/ETFA.2005.1612593
11. www.q12.org
12. Bao JS, Jin Y, Gu MQ, Yan JQ, Ma DZ (2002) Immersive virtual product development. J Mater Process Technol 129(1):592–596 (5), 11 October 2002
13. Acal AP, Lobera AS (2007) Virtual reality simulation applied to a numerical control milling machine. International Journal of Interactive Design and Manufacturing 1(3):143–154, (August)
14. Grantcharov TP (2006) Virtual reality simulation in training and assessment. Eur Clin Obstet Gynaecol doi:10.1007/s11296-006-0054-5
15. Grégoire M, Schöme E (2007) Interactive simulation of one-dimensional flexible parts. Comput-aided Des 39(8):694–707, (August 2007)
16. Gökdere LU, Benlyazid K, Dougal RA, Santi E, Brice CW (2002) A virtual prototype for a hybrid electric vehicle. Mechatronics 12 (4):575–593 doi:10.1016/S0957-4158(01)00009-5