

**WINVR2010-3739**

## **ENABLING A HIGH FIDELITY DYNAMICS SIMULATION OF CAD ASSEMBLIES IN A VIRTUAL ENVIRONMENT FOR MACHINE DESIGN**

**Cong Yue, Hai-Jun Su\* and Juan Camilo Alvarez**

Department of Mechanical Engineering  
University of Maryland, Baltimore County  
Baltimore, Maryland 21250  
Email: haijun@umbc.edu

**Qiaode Jeffrey Ge**

Department of Mechanical Engineering  
Stony Brook University  
Stony Brook, NY 11794-2300  
Email: Qiaode.Ge@stonybrook.edu

### **ABSTRACT**

This paper presents a software architecture that enables VR-MDS (Virtual Reality Mechanism Design Studio) to simulate multi-body dynamics of computer-aided design (CAD) assemblies. VRMDS is a recently developed virtual environment dedicated to the conceptual design of mechanisms and machines. It allows users to build spatial or planar mechanisms through intuitive operations. In this paper, we develop Python's parsing modules that import CAD assembly models in either XML or MDL format files into VRMDS and visualize them through the use of WRL or OSG geometry files. CAD assembly models consist of parts as well as kinematic constraints among them. These parts and constraints can be translated into links and kinematic joints of mechanisms and machines. The dynamics simulation for the assembly is achieved by MATLAB SimMechanics solver that communicates with VRMDS through a dedicated Pymat interface and M-script files. Finally, two case studies are provided to demonstrate the feasibility and validity of using assembly models in this virtual reality system for mechanism design. The high fidelity of the SimMechanics dynamics solver makes the simulation justified scientifically. The result is a highly integrated virtual reality design environment that is dedicated to both the concept design and virtual prototyping of machines.

### **1 Introduction and motivations**

Virtual Reality (VR) [1] is an emerging engineering design tool [2] that can shorten the development time, cut costs and im-

prove usability and quality of products. VR technology permits testing of virtual prototypes through high fidelity interactive simulation in order to reduce expensive and time-consuming physical prototyping. The advent of abundant medium and high level development toolkits such as VRJuggler [3], Vizard Virtual Reality Toolkit by WorldViz LLC [4], Virtual Reality Peripheral Network (VRPN) [5] makes rapid prototyping of VR environments easier than ever before.

In this paper, our focus is on the use of VR in machine and mechanism design. Machine or mechanisms are key components of numerous engineered products and systems. They are often the motion components that interact with external environment. Many authors have contributed to increase the usability of VR technology to mechanism design. For instance Prof. Vance and her group at Iowa State University have developed VRSpherical [6] and VRSpatial [7] that are specialized for spherical and spatial mechanism design. Recently these works have been extended to the design of compliant mechanisms [8]. However they do not provide dynamics simulation functionality. [9] have developed a virtual environment (VE) for design evaluation of mechanical systems. Their focus is not on the construction of mechanical systems but on the dynamic simulation aspect. [10] have developed a VR for a particular type of parallel robot.

One indispensable functionality for machine design in a VR environment is a high fidelity dynamics simulation, in particular multi-body dynamics. Even though a large number of physics engines including Open Dynamics Engine (ODE), Ageia PhysX, Bullet, Newton etc. [11] available, they are mostly designed for gaming not for scientific simulation. Furthermore, these en-

---

\*Address all correspondence to this author.

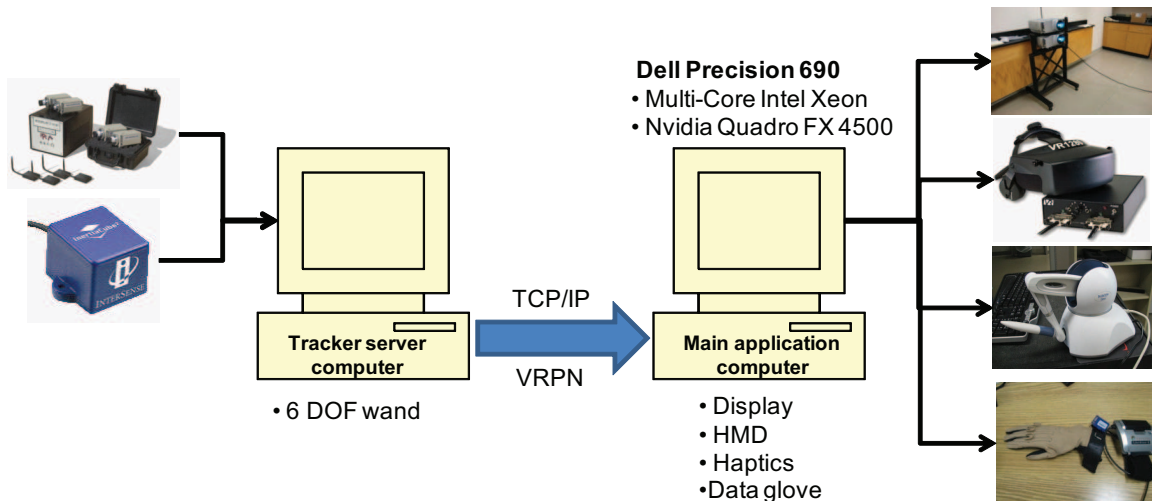


Figure 1. The hardware architecture of the system

gines are not robust when simulating machines with loop linkages. On the other hand, most modern CAD software is bundled with computer-aided engineering (CAE) tools that provide a high fidelity physics simulation. One representative product for machine design is the ADAMS software by MSC Software Co. MATLAB's SimMechanics toolbox also provides a powerful multi-body dynamics solver. It allows user adjust the solver parameters and tolerances. However these high fidelity dynamics solvers are not accessible to most VR systems as they are not open sources. Pöhland et.al [12] have developed a software library for real time simulation of multibody systems (MBS) and a library for Virtual Reality (VR) applications.

Recently many authors have attempted to integrating virtual reality into CAD systems [13–18]. This is so called VR-aided design (VRAD). The ultimate goal is to allow intuitive and direct 3D editing on native CAD models within virtual environment [15]. Several issues [16] concerning space positioning precision and reduced perception in depth direction etc. have been identified and yet to be solved.

Recently Alvarez and Su (2009) [19] have developed VR-MDS (Virtual Reality Mechanism Design Studio) for supporting interactive conceptual design of mechanisms and machines. VR-MDS allows user create machines through VR devices such as motion trackers and haptic devices. Subsequently it allows dynamics simulation through a dedicated interface that communicates with the Matlab SimMechanics toolbox for solving multi-body dynamics. However it lacks of the capability of importing and simulating complex CAD assemblies which is important for virtual prototyping stage in a product design cycle.

In this paper, we will develop a set of Python routines for parsing MDL and XML models. These models are exported from CAD software such as Pro/Engineer and Solidworks. These

modules allow us simulate the multi-body dynamics of highly complicated CAD assemblies in VRMDS. These new modules will enrich the functionalities of VRMDS and enable both the concept design and virtual prototyping in the same environment.

The rest of the paper is organized as follows. Section 2 presents the overall architecture. Sections 3 and 4 describes the implementation details. Section 5 presents the interaction and communication mechanism used in the system. Section 6 provides two case studies to demonstrate the use of this system for complex CAD assemblies. Section 7 presents the conclusion and future work.

## 2 System architecture

This section presents the overview of the system architecture.

### 2.1 Hardware architecture

VRMDS was developed at the Virtual Reality and Mechanisms Laboratory (VRML) at UMBC. The application is mainly supported by two computers and a combination of peripheral devices that enhance the immersion and sense of telepresence. Head and hand tracking is achieved by a combining a PPTX4 tracking system and InertiaCubes. This devices run on a server computer where their real-time sensory data is processed in World Viz' PPT studio. The real-time position and orientation data is then transferred to the main application computer via TCP/IP VRPN7 (virtual reality peripheral network) protocols. Figure 1 shows the hardware architecture of VRMDS.

Currently Virtual Reality & Mechanism Lab of UMBC is accommodating a 10 feet by 7.5 feet VisDuo-SX dual projector display and a VR 1280 SXGA Head Mounted Display. A video

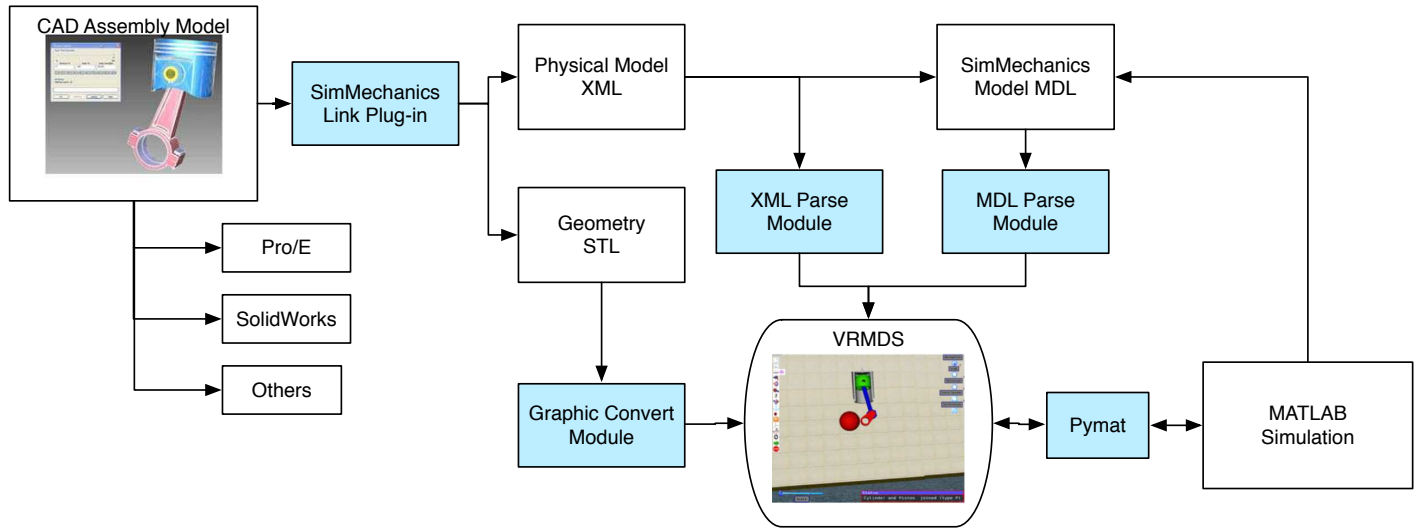


Figure 2. Software architecture of the system

splitter is used to deliver the identical image to both the power wall and the HMD. The head tracking is enabled by a PPT X4 optical position tracker and a InertiaCube orientation tracker. A dedicated computer is employed to work as a server for providing real time tracking data through VRPN [5] to the main program running on another computer over an internal network. The main computer is a DELL 690 Precision workstation equipped with a Nvidia's Quadro FX4500 professional graphics card.

## 2.2 Software architecture

VRMDS supports 3D stereoscopic visualization, haptic interaction, head mounted display (HMD), head and hand tracking. Also, VRMDS has accomplished four major functions: Visualization, Interaction, Graphical User Interface (GUI) and Modeling by previous work. The visualization module provides a virtual reality display environment. The interaction module provides functionalities such as scene navigation, object selection and manipulation (by grabbing), which are necessary for interactive design activities. The Graphical User Interface (GUI) is implemented using the `vizinfo` and `vizmenu` modules of Vizard. Finally, the modeling module allows us to create and modify a parallel SimMechanics model in the background and enable the real-time data flow from and to VRMDS. These works have been done and described in [19].

Based on these works, we extend the functionality of VRMDS to make it not only able to create and modify mechanism, but also able to import existing physical models from XML or MDL model files into VRMDS. The advantage is that we can exploit the existing functionalities provided in VRMDS.

Figure 2 gives the architecture view of how this new VR-

MDS function works. It can be concluded into following steps:

1. Obtaining CAD assembly model. These models can be created by many common used 3D modeling software such as Pro/ENGINEER, SolidWorks and etc.
2. Generating XML format model file from assembly model. We use SimMechanics Links plug-in which is provided by MathWork<sup>TM</sup> Inc. to export assembly model to XML model. It also converts geometry from the CAD assembly into .stl extension geometry file needed for visualizing the bodies in the model.
3. Importing XML model into VRMDS. By using XML parse module, the model can be interpreted and imported into VRMDS.
4. Attaching geometry into VRMDS. Graphic convert module would be called automatically to convert .stl file into .osg file and import it into VRMDS. After that, geometry graphics would be visualized in VRMDS.
5. Transferring data to SimMechanics environment. After receiving information from the XML model, VRMDS not only creates assembly model in its environment, but also transfers all the data of bodies and joints into MATLAB's SimMechanics environment through Pymat. We use Pymat to transfer data to some pre-built m-files, which are used to create block diagrams in SimMechanics.
6. Dynamic simulation and real-time interaction. Finally, after all the steps are done, we can run the simulation in VRMDS. At this time, the identical SimMechanics system would run in the background, and dynamics solver of MATLAB would run in real-time. All the position and orientation data are collected by two sensors. Then, through S-Function all

the output would send to Pymat, which would transfer them back to VRMDS, so that VRMDS would do real-time simulation by continuously reading data from Pymat and sending changes to Pymat.

As Figure 2 implies, this architecture also provide utility that enables VRMDS to import MDL files. MDL files can be obtained from XML model files or earlier work done in SimMechanics system. It is another optional method to import physical model into VRMDS. MDL parse module is developed to collect the information needed by VRMDS. After the model is imported into VRMDS, we can simulate the dynamics of the model.

### 3 Implement importing assembly model from model file

This section presents the implementation of importing XML and MDL model file. In both VRMDS and SimMechanics, a machine system is comprised of multiple links (or called bodies) which are connected by joints. Links are constructed according to reference coordinate systems (CS) and center of gravity (CG). And joints are created according to CSs of both base link and follower link. Thus, when reading in XML and MDL files, we must get link (or body) list and get joint list separately. Although the data structures are different between XML and MDL files, the overall procedure are almost the same as shown in Figure 3. The process can be described by following steps:

1. Read in model file and decide whether it is XML file or MDL file. And calling corresponding parse module.
2. Scan the file for the link list and joint list. The link list includes body name, CS(s) and CG information. The joint list keeps all the joints information such as joint name, joint type, joint axis, base link CS and follower link CS.
3. After the above work, both lists are sent to link class and joint class in order to create links and joints. With all these data, system can be built in VRMDS and SimMechanics.

Details on parsing MDL and XML model files are described in the following sections.

#### 3.1 Using MDL model file

File with .mdl extension are Simulink model file, which contains all the information for building block diagrams used in SimMechanics environment in MATLAB.

We have developed Python routines to parse the input MDL model file and create a new SimMechanics system. Please note this process is necessary since the original MDL file does include the following information that is necessary for dynamics simulation in VRMDS: 1) Sensor blocks for data transfer between VRMDS and SimMechanics. 2) The original CS data is not relative to the reference CS, but to the world. 3) VRMDS

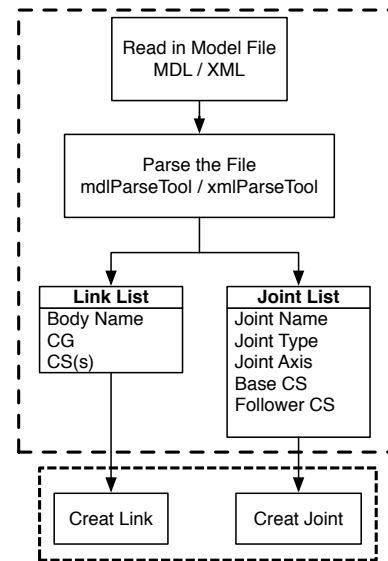


Figure 3. The general process of importing model files

needs a method to reuse the MDL file. 4) In the future, more work can be done through the modification of MDL file directly and efficiently without starting a MATLAB session.

Figure 4 shows the hierarchical structure of a typical MDL file. Generally, all the bodies and joints attributes are included in several different children blocks and “Line” block contains connection information of the system.

Base on the work done by Fauske [20], we developed a mdl- ParseTool() module package that can parse mdl file into nested list object in Python. This module aims at building functionalities which are similar to the command functions provided by Simulink.

By developing several functions, it provides us with the ability of obtaining list of bodies and joints, as well as their attribute information such as position, orientation, relative CS and connection port information.

#### 3.2 Using XML model file

The Extensible Markup Language (XML) is one of the most widely-used formats for representing structured information. Figure 5 represents the typical structure of XML model file. All the bodies and joints information are stored in between <bodies> and <joints> elements respectively. In each <Body> node, it contains many <Frame> elements which represent CS or CG of one body. Typical element syntax could be like <Frame ref='1'> and the element should be closed with </Frame>. In this example, “Frame” is the tag name of this element, and “ref='1' ” represents an attribute and its value. The value of “ref” is an useful index for finding the con-

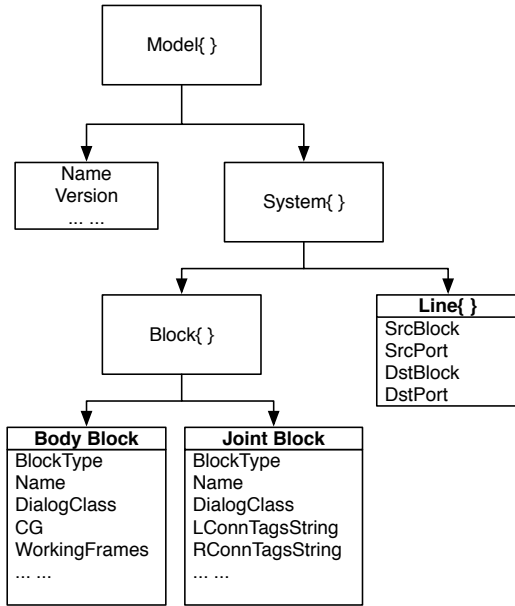


Figure 4. The structure of MDL model file

nection relation between joints and bodies.

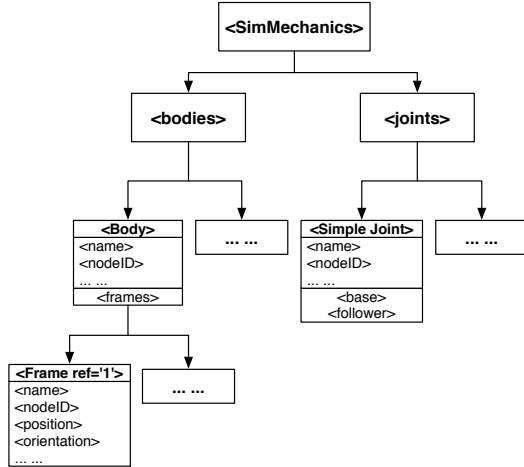


Figure 5. The structure of XML modeling file

Python provides a useful module for parsing XML file called `minidom` which is a light-weight implementation of the Document Object Model (DOM) interface [21]. It can parse XML file into a document object for representing the content of the XML. Based on it, we defined our own `xmlParseTool` module for fetch-

ing a list of information of bodies and joints.

#### 4 Model generation and data translation

Before building a model, an M-file called "new\_sys" is executed at the initialization stage, resulting a default template of SimMechanics MDL file created in the background. Figure 6 shows the contents of the template MDL file. It consists of a machine environment where the solver parameters, gravity vector, and visualization configuration parameters are specified. In addition, a ground link defined as Link0 is fixed to a ground block. Finally, two Mux bars are pre-built, and two blocks connect to them. One is S-function block used for transfer sensor for model analysis, the other block is for sharing data to main program.

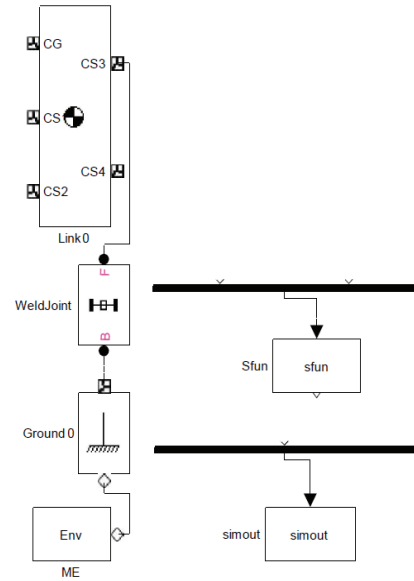


Figure 6. The template system file

After initialization, VRMDS will load two classes named Link class and Joint class and build physical model by using the link list and joint list generated from either XML or MDL file. General speaking, these two classes do two things. On the one hand, they build physical model in VRMDS, on the other hand they send links and joint data to Pymat in order to create SimMechanics model behind the scene. The structures of link class and joint class are described in [19]. However, the previous version of classes can only work for building model within VRMDS environment. For that reason, we rewrite the classes to include new features and new functionalities which are described below.

## 4.1 Link class

Link class includes main functions as below:

1. Translate position and orientation of CS and CG.
2. Get/Set reference CS.
3. Add/Get CS.
4. Attach geometry of links to CS.
5. Update link information to SimMechanics environment.

As described above, in VRMDS system, we have a default template pre-built before creating a new mechanism. In this template (show it in example), there is a basic link named “Link0”, which is regarded as root ground and all the other bodies and joints are created related to it. However, the position and orientation of 3D model represented in XML or MDL files are using world coordinate system by default. As a result, we need to recalculate the transform matrix of CS and CG transform matrix relative to the new reference CS.

After translation, a geometry model will be attached to a designated CS and displayed in the VR environment. Since VRMDS only accepts .wrl or .osg files, we use an open source library called *osgconv* to convert STL file into OSG file automatically. Next, link class will call the constructor function, which calls a MATLAB .m script named “newlink” with CS and CG information via Pymat. As a result, body diagram will be created in SimMechanics environment.

## 4.2 Joint class

The Joint class has two main functionalities:

1. Create joints between base link and follower link
2. Update joint information to SimMechanics environment.

To create a joint, the base link and follower link information must be provided. This could be done by either input joint list information imported from XML or MDL files, or be detected when picking bodies in VR environment. After that, joints between bodies could be built.

Then through update function, like links, joints data will be transferred to a M-file script, named “newjoint”, and a joint diagram will be created in SimMechanics environment.

## 5 Simulation and interaction

Figure 7 represents the communication between VRMDS and MATLAB solver, and the implementation of simulation. Pymat plays an important role in the communication. For the model build-up stage, it transfers data from Link class and Joint class to corresponding built-in MATLAB scripts, which then build diagram blocks in the SimMechanics system.

During the simulation stage, S-function continuously sends simulation data to Pymat which collects the value and send them back to VRMDS. In the mean time, VRMDS starts a real-time

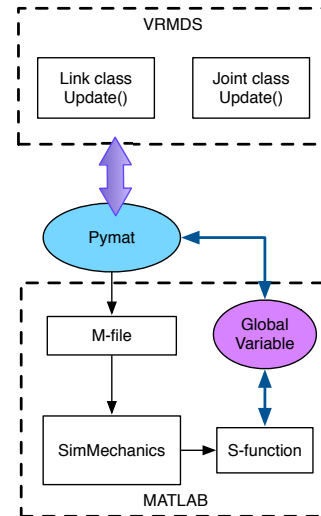


Figure 7. The communication for simulation

simulation through calling time event instance that constantly read them and update the state of links and joints. As a result, VRMDS implement an real-time simulation synchronically.

When parameters are modified in VRMDS during the run-time, the modifications are also conveyed by Pymat to MATLAB. These modifications then cause variable changes in S-function and simulation solver will receive them and do some changes immediately.

In the VR environment, we can use devices, such as WAND or PHANTOM to navigate, pick and manipulate the assemblies. Currently, the dynamics simulation must be stopped to allow object picking and manipulating. We are investigating methods that allow users to interact with the model during the simulation.

## 6 Case studies

In this section, two cases will be provided to demonstrate the validity of dynamics simulation of CAD assemblies in VRMDS.

### 6.1 Internal combustion engine model

This example will show the process to import MDL model file into VRMDS. Figure 8 shows the original MDL file that we want to import into VRMDS system. It is a internal combustion engine which is a demo from MATLAB. Basically this is a crank-rock mechanism, it contains five parts: cylinder, piston, connecting rod, crank and crank shaft. And it has five joints: one weld joint, two revolute joints, one prismatic joint and one cylindrical joint. And all the parts have .wrl geometry file. Figure 9 shows the assembly model that has been imported into VRMDS. Figure 10 shows the MDL file created in the background. One

can see that extra sensor blocks, CS ports and mux are added to communicate with VRMDS. Compared with the visualization of Matlab, VRMDS provides an immersive visualization and intuitive interaction that is critical to design review and online modification.

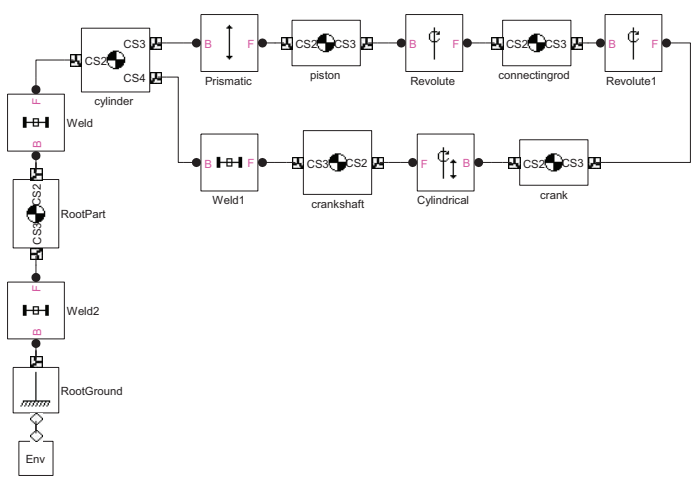


Figure 8. Diagram block of internal combustion engine in MDL file

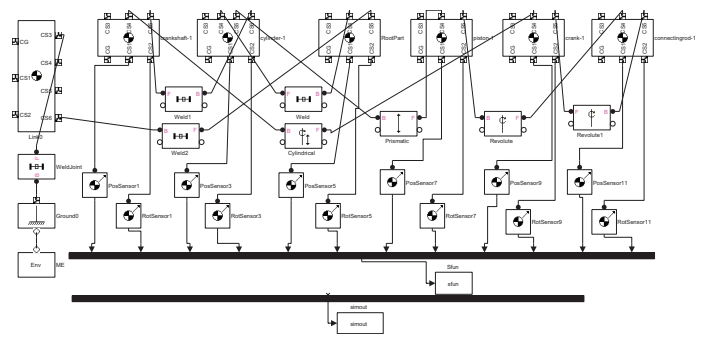


Figure 10. MDL file of internal combustion engine created by VRMDS

### 6.2 Aircraft engine

This example imports a relatively more complicated model from XML file into VRMDS. Figure 11 shows a 3D aircraft engine assembly model created from Pro/ENGINEER. This assembly model has five pistons which are connected by a crankshaft. It is important to mention that each of those five pistons is a single closed-loop mechanism which until now cannot be simulated by physics solvers such as open dynamics engine (ODE).

Totally, this model contains 27 parts, 18 weld joints, 10 cylindrical joints and 5 revolute joints. By using SimMechanics Link plug-in, we can get a XML model file for this assembly and 8 STL files for geometry. Then we import the file into VRMDS. Figure 12 shows that the engine together with an environment model is finally imported into VRMDS. In behind, a MDL model is created, which contains more than 200 blocks, so we cannot show it. The assembly model look the same as it is in 3D modeling software. Figure 13 shows the aircraft engine simulated in the VR environment. While in the VR environment, you can walk around to investigate the assembly from different angles of view. Also you can use devices like WAND or PHANTOM to manipulate each part of the assembly model.

In addition, users can add controllers to the model to analyze the machine system. Note this provide a platform for testing controllers.

### 6.3 Performance analysis

The dynamic simulation is rely on the powerful simulation solver of SimMechanics, so the accuracy and performance of dynamic simulation in VRMDS environment are also basically depend it. Other important factors that may influence the performance are the latency of data transfer between VRMDS and MATLAB, and computer hardware configuration. Right now the data transfer is within the same computer through Pymat interface and the data is not huge, so the performance in VRMDS can fully satisfy our needs. In our case, there are 12 different data for each body to be transferred every time, for the complex aircraft engine model, they have less than 30 parts and that is only about

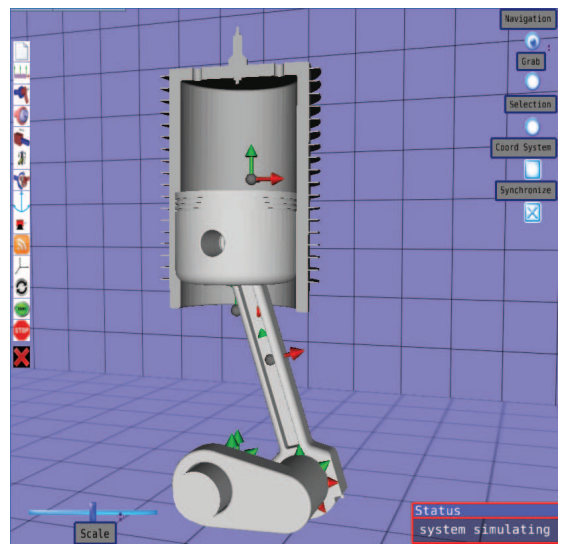


Figure 9. Internal combustion engine simulated in VRMDS

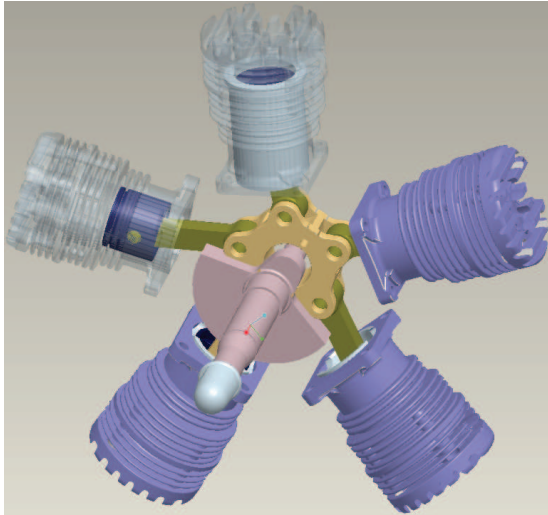


Figure 11. Aircraft engine assembly model



Figure 12. Aircraft engine assembly imported in VRMDS



Figure 13. Simulation and interaction in the VR environment

ously establishes a data communication with the main program through Pymat library. We have used two Pro/Engineer assembly models to demonstrate the use of the proposed framework. This framework enables VRMDS importing and simulating highly complex CAD assemblies in a VR environment without much efforts.

### Acknowledgements

The authors acknowledge the support of National Science Foundation under Collaborative Research grants to University of Maryland at Baltimore County (Hai-Jun Su, grant CMMI-0900517) and Stony Brook University (Qiaode Jeffrey Ge, grant CMMI-0856594).

### REFERENCES

- [1] Burdea, G., and Coiffet, P., 2002. *Virtual Reality Technology Second Edition*. John Wiley & Sons Inc., Hoboken, New Jersey.
- [2] Jayaram, S., Vance, J., Gadh, R., Jayaram, U., and Srinivasan, H., 2001. "Assessment of vr technology and its applications to engineering problems". *ASME J. Comput. Info. Sci. Eng.*, **1**(1), pp. 72–83.
- [3] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C., 2001. "Vr juggler: A virtual platform for virtual reality application development". In *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)*, IEEE Computer Society, p. 89.
- [4] Vizard vr toolkit. WorldViz LLC. [www.worldviz.com](http://www.worldviz.com).
- [5] Taylor, II, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T., 2001. "Vrpn: a device-independent, network-transparent vr peripheral system". In

360 different data to be transferred, also due to the multi-core of CPU, the computation and transfer speed can be satisfied.

### 7 Discussion and Conclusion

This paper presents a general framework for enabling a high fidelity multi-body dynamics simulation of complex CAD assemblies in a virtual environment that is dedicated to machine and mechanism design. We described the overview and implementation details of the system which are implemented in Python and Matlab scripts. To use the system, users can export a CAD assembly into a XML file or a MDL file which is then imported into VRMDS. VRMDS automatically generates a corresponding Matlab SimMechanics model in the back scene while simultane-

- VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology, ACM, pp. 55–61.
- [6] Furlong, T. J., Vance, J. M., and Larochelle, P. M., 1999. “Spherical mechanism synthesis in virtual reality”. *ASME Journal of Mechanical Design*, **121**(4), pp. 515–520.
- [7] Vance, J. M., Larochelle, P. M., and Dorozhkin, D. V., 2002. “VRSpatial: Designing spatial mechanisms using virtual reality”. In Proceedings of ASME IDETC/CIE 2004.
- [8] Seth, U., Su, H.-J., and Vance, J. M., 2009. “A virtual reality interface for the design of compliant mechanisms”. In Proceedings of ASME IDETC/CIE 2009.
- [9] Antonya, C., and Talaba, D., 2007. “Design evaluation and modification of mechanical systems in virtual environments”. *Virtual Real.*, **11**(4), pp. 275–285.
- [10] Stan, S.-D., Manic, M., Maties, V., and Balan, R., 2008. “A novel virtual reality robot interface for isoglide3 parallel robot”. In ICIRA (1), pp. 1265–1275.
- [11] Boeing, A., and Bräunl, T., 2007. “Evaluation of real-time physics simulation systems”. In GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia, ACM, pp. 281–288.
- [12] Pöhland, K., Berssenbrügge, J., Krumm, H., and Ebbesmeyer, P., 2002. “Virtual reality-based dynamics simulation for virtual prototypes”. *ASME Conference Proceedings*, **2002**(36215), pp. 95–101.
- [13] Joshi, H., Jayaram, S., Jayaram, U., and Varoz, L., 2008. “An open architecture for embedding vr-based mechanical tools into cad applications”. *ASME Conference Proceedings*, **2008**(43277), pp. 1625–1635.
- [14] Barbieri, L., Bruno, F., Caruso, F., and Muzzupappa, M., 2008. “Innovative integration techniques between virtual reality systems and cax tools”. *The International Journal of Advanced Manufacturing Technology*, **38**(11), pp. 1085–1097.
- [15] Bourdot, P., Convard, T., Picon, F., Ammi, M., Touraine, D., and Vzien, J.-M., 2008. “Vr-cad integration: Multimodal immersive interaction and advanced haptic paradigms for implicit edition of cad models”. *Computer-Aided Design*, **In Press, Corrected Proof**, pp. –.
- [16] Fiorentino, M., Monno, G., and Uva, A., 2006. “Cad interfaces in virtual reality: Issues and solutions”. In Proceedings of the workshop on virtual reality in product engineering and robotics: Technology and applications.
- [17] Erlemeier, T. A., 2006. “The development of a virtual reality based cad system for design review”. Ms thesis, Iowa State University, Ames, Iowa, USA.
- [18] Kim, S.-R., and Weissmann, D., 2006. “Middleware-based integration of multiple cad and pdm systems into virtual reality environment”. *Computer-Aided Design & Applications*, **3**(5), pp. 547–556.
- [19] Alvarez, J. C., and Su, H.-J., 2009. “Vrmds: an intuitive virtual environment for supporting the conceptual design of mechanisms”. *Virtual Reality, SI: Manufacturing and Construction*, pp. 1–12.
- [20] K. m. fauske. <http://www.fauskes.net/nb/parsing-simulink/>.
- [21] Rossum, G. V., 2000. *Python Library Reference*. Iuniverse Inc., Bloomington, Indiana.