

# VRMDS: an intuitive virtual environment for supporting the conceptual design of mechanisms

Juan Camilo Alvarez · Hai-Jun Su

Received: 18 June 2009 / Accepted: 30 October 2009  
© Springer-Verlag London Limited 2009

**Abstract** This paper presents Virtual Reality Mechanism Design Studio (VRMDS), an intuitive virtual environment for supporting the interactive design and simulation of mechanisms. The studio allows users to build spatial or planar mechanisms through intuitive operations and subsequently simulate their dynamic motion. Written in Python script language, VRMDS provides 3D stereoscopic immersive visualization, haptic enabled interaction, head and hand tracking and a user-friendly graphical user interface. A data model for organizing the data structure of links and commonly used mechanical joints is designed and implemented upon the basis of the Vizard Virtual Reality (VR) library. Within the virtual environment, the user can create links and assemble them into a mechanism by defining joints between links. Simultaneously, a corresponding MATLAB's SimMechanics model is automatically created at run time. The dynamics simulation of mechanisms is enabled by interfacing with the dynamics solver built-in SimMechanics. The user may choose to run the system in an immersive VR environment or a desktop environment. The result is a versatile mechanism design tool that is beneficial to the early stages of the design process. A case study of a spatial mechanism is provided to demonstrate the usefulness of this system in mechanism design.

**Keywords** Virtual reality · Conceptual design · Computer-aided design · Mechanism design ·

Multi-body dynamics simulation · Haptic interfaces · SimMechanics

## 1 Introduction

Conceptual design is one of the early stages in a design process. The main focus of the conceptual design is on innovation and creative designs rather than on detailed modeling. An ideal conceptual design tool shall allow engineers to explore a wide range of design alternatives, to evaluate their performance and to select the best one for further analysis and development without manual data conversion. Even though concept generation only constitutes less than 5% of the total development cost, over 75% of the production and manufacturing cost is committed by the end of the conceptual design phase (Ullman 2003). Therefore the importance of a conceptual design tool can never be overstated.

A kinematic mechanism or linkage is a collection of links connected with kinematic pairs or joints (McCarthy 2000). Although not always visible, mechanisms are widely embedded in engineered products and systems such as automobiles, aircrafts, robots, medical devices, biomechanical devices, as well as production systems for these products. Many mechanisms (e.g. industrial manipulators) in the real world are naturally three dimensional, i.e. producing a spatial motion. Simulating and visualizing the motion of these spatial mechanisms is an indispensable tool of the design process.

In the past decades, numerous contributions have been made by academia and industry in delivering powerful mechanism design and simulation tools. From the kinematics and mechanisms community, modern implementation of linkage synthesis software include LINCAGES

---

J. C. Alvarez · H.-J. Su (✉)  
Department of Mechanical Engineering, University of Maryland  
Baltimore County, 1000 Hilltop Circle, Baltimore,  
MD 21250, USA  
e-mail: haijun@umbc.edu

(Nelson and Erdman 1994), Sphinx (Larochelle et al. 1993), SPADES (Larochelle 1998) and Synthetica (Su et al. 2002). All of these mechanism design software focus on the dimensional synthesis of planar, spherical or spatial linkages. Even though they provide powerful kinematic synthesis solvers, they do not have interface for constructing mechanisms and simulating their motion. Commercial linkage design tools include WATT by Heron Technologies Inc., SyMech by symech.com and Ch Mechanism Toolkit by SoftIntegration Inc. (Cheng and Trang 2006). These tools mainly focus on the analysis and synthesis of planar 4-bar, 6-bar or 8-bars. Ch Mechanism Toolkit is an excellent for students learning mechanism design. Unfortunately, none of them provide a full dynamic simulation for general mechanisms.

The focus of commercial CAD software such as SolidWorks and Pro/E has been on tools for shape representation and manipulation, assembly and detail drawing creation, and integration with computer-aided manufacturing (CAM) and analysis packages. Even though many of them have already incorporated a robust kinematic constraint solver, they all lack of truly intuitive tools targeting the early design stages, which focus on creative and innovative design. Dynamic simulation packages such as WorkingModel (Design Simulation Technologies Inc.) and ADAMS (MSC Software Co.) provide versatile functionalities in modeling and simulating mechanisms. The former is an excellent tool for the conceptual design of planar mechanisms. Unfortunately, WorkingModel does not support spatial mechanisms, and it does not provide an advanced visualization and interaction functionality. Although ADAMS does support spatial mechanism design and simulation, it requires a commitment to a steep learning curve and it does not provide a user-friendly interface, which allows engineers to build and simulate a virtual mechanism through natural operations. MATLAB's SimMechanics toolbox provides a powerful multi-body dynamics solver. However, its building process is based on block diagram, which is not intuitive (see what is built), and is overly complicated. Its learning process can be long, as designers must know a great deal of parameters in order to build a valid model. Furthermore, its graphical visualization is primitive even when the MATLAB's VR toolbox is used.

During the last fifteen years, increasing affordability and fidelity of sensors and visualization equipments has made virtual reality (VR) (Burdea and Coiffet 2002) more and more commonly available for the product design and manufacturing process (Jayaram et al. 2001). The advent of abundant medium and high level development toolkits makes rapid prototyping of VR environments easier than ever before. Examples are VRJuggler (Bierbaum et al. 2001), Vizard Virtual Reality Toolkit by WorldViz LLC, Virtual Reality Peripheral Network (VRPN) (Taylor et al.

2001). In the aspect of virtual manufacturing, a great deal of work has focused on virtual prototyping, virtual assembly, product demonstration, training and education. For instance, Jayaram et al. (1999) have developed Virtual Assembly Design Environment (VADE) at Washington State University, Wan et al. (2004) have developed MI-VAS: A Multi-modal Immersive Virtual Assembly System. Seth et al. (2008) have developed SHARP: a Dual-Handed Haptic Assembly System, Brough et al. (2007) have develop a virtual environment for assembly training and Zhu (2008) has developed a haptic interface for Solidworks in supporting interactive assembly. However, relatively less work has been done for applying VR techniques in supporting interactive design especially the conceptual design stage. Antonya and Talaba (2007) 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. Stan et al. (2008) have developed a VR for a particular type of parallel robot by using the MATLAB's VR toolbox, which does not provide a 3d stereoscopic immersive visualization.

VR programs dedicated to the design of spherical mechanisms (Furlong et al. 1999) and spatial mechanisms (Vance et al. 2002) have been developed by Prof. Vance and her group at Iowa State University. Recently, these work have been extended to the design of compliant mechanisms (Seth et al. 2009). The immersive stereoscopic visualization and intuitive interaction provided by VR benefit the mechanism design in providing a better understanding of space position reasoning, interface checking and online evaluation. However, the focus of these work is on a particular kind of mechanisms, spherical 4R or spatial 4C, etc., not on providing an intuitive and user-friendly interface for interactively building mechanisms.

Given the enormous innovation and advances in VR software and hardware technologies, the use of VR in industry is still limited in visualization and demonstration. One major obstacle is that it requires long and tedious procedures to convert CAD data set into the format usable by VR. Rather than the traditional way of bringing the CAD models into VR environment, recently much work (Barbieri et al. 2008; Bourdot et al. 2008; Fiorentino et al. 2006; Erlemeier 2006) has been done in integrating virtual reality into CAD, so-called VR-aided design (VRAD). The ultimate goal is to allow intuitive and direct 3D editing on native CAD models within virtual environment (Bourdot et al. 2008). However many issues such as space positioning precision and reduced perception in depth direction etc. have been identified and yet to be solved. See Fiorentino et al. (2006). Another drawback is that one has to work on a specific CAD system, meaning that the modules developed for one CAD system are not applicable to other

systems. Nevertheless, VRAD holds a big promise and deserves further attentions in the near future.

From the above literature survey, we can see that currently no mechanism design tool simultaneously has the following features: (1) focusing on the conceptual design stage, (2) providing an intuitive and user-friendly interface, (3) supporting arbitrary mechanism topologies and (4) providing full dynamic simulation capability. In this paper, we intend to fill this gap by developing a Virtual Reality Mechanism Design Studio (VRMDS) for supporting the interactive conceptual design and evaluation of spatial mechanisms. The program enables a fast concept generation and evaluation through the use of the powerful functionalities provided by Vizard VR library and MATLAB's SimMechanics toolbox.

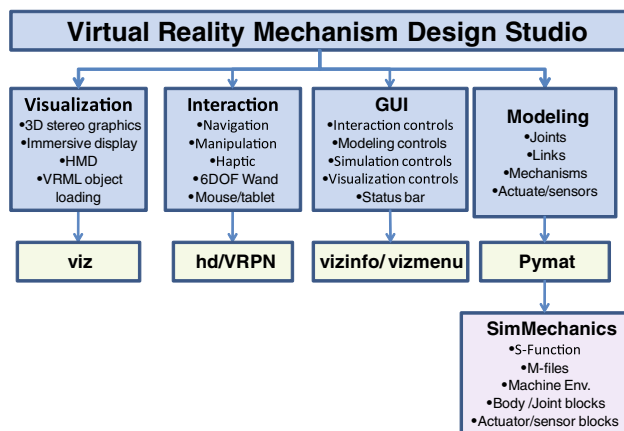
The rest of the paper is organized as follows. Section 2 presents the overall architecture and example uses of the system. Section 3 describes the data model of basic mechanism building elements such as links and joints. In Sect. 4, a single-joint pendulum is used to describe the basic steps for building and simulating a mechanism with the program. Section 5 provides a spatial mechanism example to demonstrate the use of this system for designing a more complicated mechanisms. Section 6 presents the conclusion and future work.

## 2 System overview

This section presents the overall architecture and major components of the system. Also we will describe example uses of the system for mechanism design.

### 2.1 Software architecture

Figure 1 shows the architecture of VRMDS, which incorporates four major modules: Visualization, Interaction,



**Fig. 1** The overall software architecture of virtual reality mechanism design studio (VRMDS)

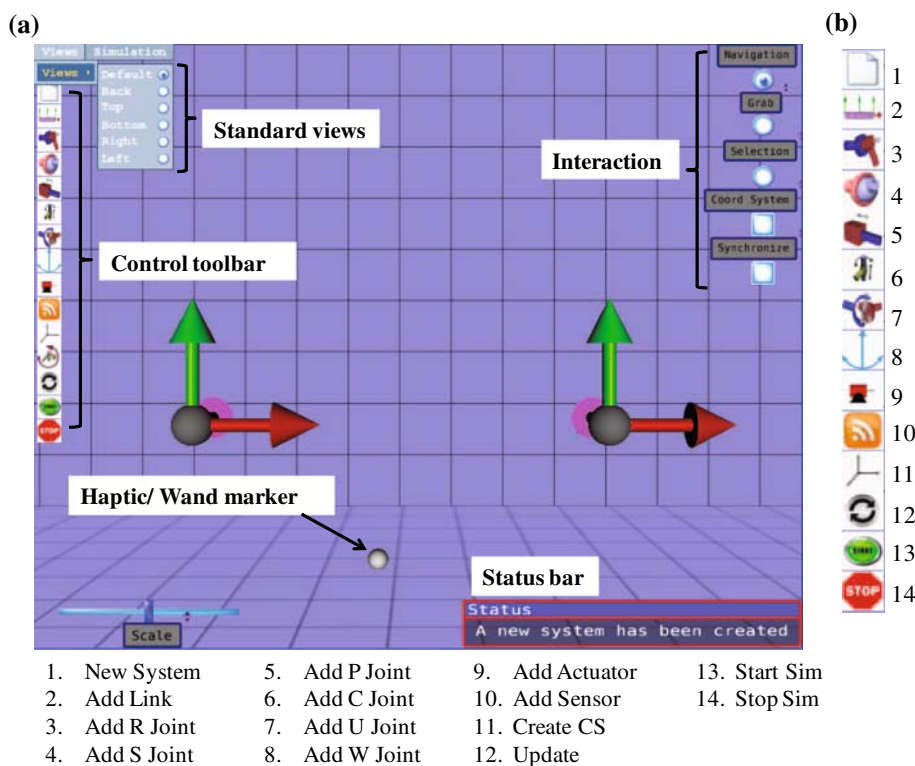
Graphical User Interface (GUI) and Modeling. We will discuss each in the following paragraphs.

The visualization module is based on Vizard's viz library, which use the open source OpenSceneGraph library as the 3D graphics engine. The 3D stereo and immersive visualization can be enabled/disabled by setting flags at the beginning of the Python program. The graphical representation of links and environmental objects such as floor and walls are imported into VRMDS by using the Vizard's built-in VRML loader. Six predefined standard views, "Front", "Back", "Top", "Bottom", "Left" and "Right" are employed for easy and quick navigation. It also supports head mounted display (HMD) for personal immersive visualization. These functionalities provided by Vizard library greatly simplifies coding and debugging, hence reduces the development cycle.

The interaction module provides functionalities such as scene navigation, object selection and manipulation (grabbing), which are necessary for interactive design activities. A set of flags along with a set of radio buttons are used to switch between these modes. When the program is under navigation mode, users can navigate through the scene for the best viewing angle by operating a mouse, a graphic tablet, a Sensable's PHANTOM Omni or 6DOF wand. When a PHANTOM Omni or wand is used, the user presses the stylus or wand buttons to navigate forward or backward along the direction of the PHANTOM stylus. Under the selection and manipulation mode, the user can either use a mouse, a PHANTOM Omni or 6DOF wand to pick and manipulate an object. Upon hd's collision detection implementation, the selected object's transparency will change indicating the user which of the objects is currently picked. Each object in VRMDS has a duplicate physics model which is created and updated on the fly. And the collision detection is based on vizard's physics library and checked by colliding the marker object with each object the physics scene. When a PHANTOM Omni is used, the user haptically feels the object being touched. He or she can then grab/drop an object by pressing/releasing a button.

The Graphical User Interface (GUI) is implemented using the vizinfo and vizmenu modules of Vizard. A screen shot of the design interface is shown in Fig. 2a. The example in the figure represents the interaction through the use of a PHANTOM Omni represented by a haptic marker inside the virtual design environment. The user can also use a 6DOF wand, in which case the marker object is dragged by the wand. The interaction controls consist of a group of radio buttons that make the navigation, manipulation and selection modes mutually exclusive. As far as modeling and simulation controls are concerned, VRMDS provides a toolbar with buttons that trigger the different actions (Fig. 2b). Each one of the button events is

**Fig. 2** a A screen shot of graphical user interface and b the control toolbar



associated with either a SimMechanics command or one of our built-in M-files besides its main action inside our virtual environment. Following each of those events, multiple calls are made to retrieve model or simulation information from the real-time data written in the MATLAB workspace by our S-Function. As a result, not only VRMDS simplifies the SimMechanics modeling process, but also it provides enhanced 3-D stereo graphics and interaction with the model. A status bar is implemented to prompt the user with useful information relevant to the model's current stage of design or simulation as well as violations or errors.

Finally, the modeling module allows us to create and modify a parallel model in Simulink by using the Python to MATLAB interface PyMat. Pymat acts as a dynamic link library (DLL) enabling the real-time data flow from and to VRMDS. As a result, any attempted design in VRMDS will also be simultaneously created, simulated and validated in SimMechanics, a Simulink toolkit that enables users to model mechanical systems. In the design process, GUI events will trigger not only the specific actions in VRMDS but also their equivalent in SimMechanics by using our built in M-files for system creation, modification, and validation, as well as S-Functions employed for real-time simulation data sharing. All the SimMechanics modeling will run internally in MATLAB and will never be seen by the user. We will discuss the details about mechanism modeling in Sect. 3.

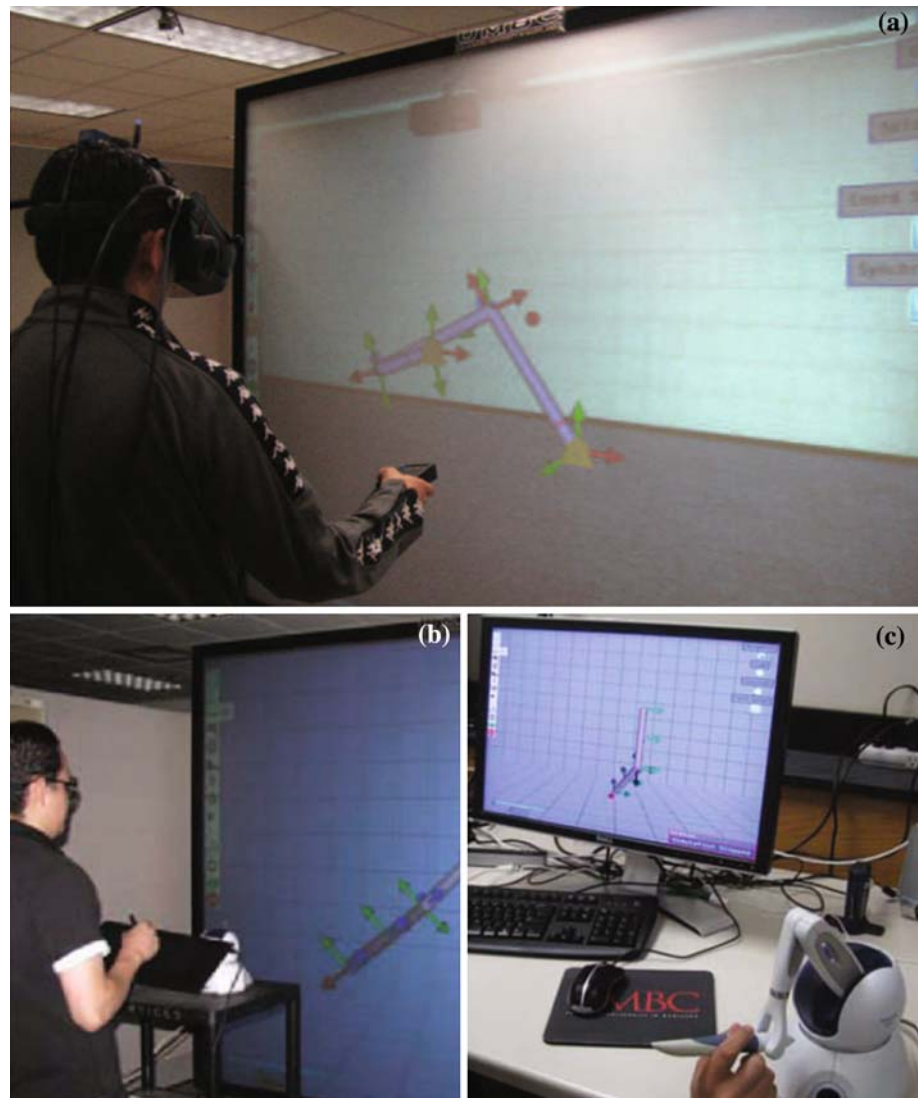
## 2.2 Hardware settings and example uses

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 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 (Taylor et al. 2001) 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.

Figure 3 shows three typical hardware settings for using VRMDS in mechanism design. Figure 3a shows an immersive display environment where the user wears a HMD with a head tracking for immersive visualization and uses a 6DOF wand for object manipulation and picking.

Figure 3b shows the use of a graphic tablet for GUI interaction, and haptic device is used for picking and manipulation. The graphic tablet option is ideal for replacing a mouse in an immersive VR environment. The immersive visualization can be enabled by setting a 3D stereo flag and using a Head-Mounted Display (HMD) or a power wall system with polarized glasses. It should be pointed out that

**Fig. 3** The use of VRMDS with **a** a 6DOF wand, HMD with PPT tracker and a power wall **b** a power wall display, a Wacom graphic tablet and a PHANTOM Omni, **c** a desktop LCD monitor and a PHANTOM Omni



using a PHANTOM Omni within this environment can distract the designer's attention. The workspace volume of the haptic device is also a limiting factor. Nevertheless, we recommend this setting for team discussion and brainstorming which are essential for the conceptual design stage.

Figure 3c shows a desktop environment which is comprised of a LCD monitor and a desktop PHANTOM Omni haptic device. This personal setting is ideal for daily design practices. Within this environment, the designer focuses on the design itself and explore different ideas and evaluate their feasibility in a timely matter. Other hardware settings are under investigation for using VRMDS for mechanism design.

### 3 Mechanism modeling and simulation

Tsai (2000) defines a mechanism or linkage as a collection of interconnected components, individually called "links".

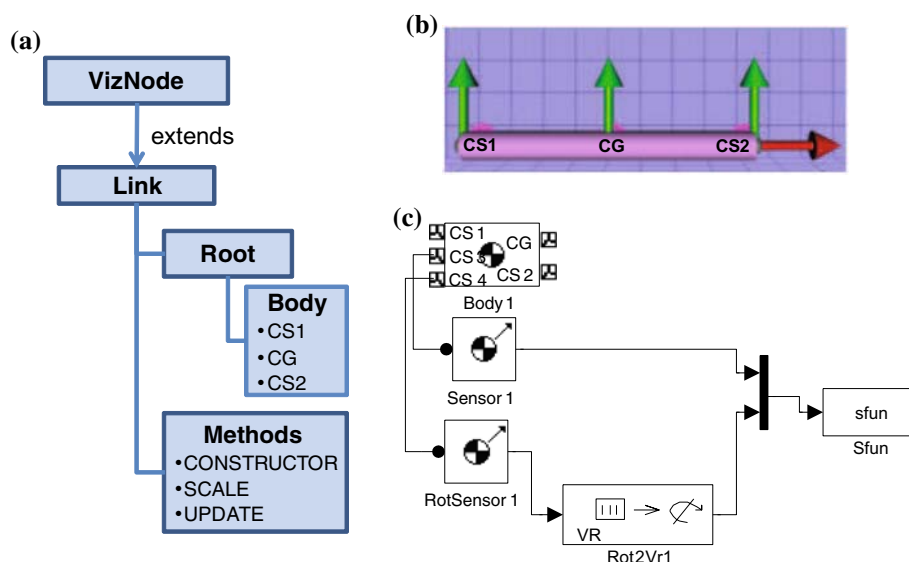
The physical connections between links are called "joints" or "kinematic pairs". In this section, we describe how links and joints are modeled and how the dynamic motion is simulated in VRMDS.

#### 3.1 Data structure of links

In this paper, a link is considered a uniform rigid body. According to the number of incident links, a link can be categorized into binary, ternary, quaternary and so on. For instance, a binary link connects to only two other links via joints. Currently, VRMDS supports links with arbitrary number of incident links. The user will create a default binary link and afterward adds any number of points to the link.

Python's object-oriented programming language capabilities were employed to develop our own data structure. A Python class `Link` is extended from the Vizard's `VizNode` class and implemented to define a rigid link. See Fig. 4a for the class hierarchy.

**Fig. 4** The data model of a binary link: **a** the structure of the Python class Link, **b** the graphical representation, **c** the body blocks and reference ports created in SimMechanics



As seen in the figure, each instance of the `Link` class is composed of a root and a body objects. The root object is an empty layer used to group multiple nodes together. Such nodes are by default the body object, its associated coordinate systems (CS) and any other point or coordinate system the user might want to define along the design process for assembly or other purposes. Figure 4b shows the graphical representation of a binary link, where the local coordinate systems CS1 and CS2 define two reference points/lines for joining other links and CG represents its center of gravity. For binary links, CS1 and CS2 are by default set at the two ends while CG is placed at the middle of the link. This particular arrangement was developed to keep consistency between VRMDS bodies and primitives and SimMechanics blocks. Through a call to the link constructor, the user creates an object that is directly tied to a SimMechanics block. The body block's initial conditions in SimMechanics are dependent on the data send upon calling the update method after interaction has taken place in VRMDS. Likewise, the link object's positions and orientations are dependent on the block's sensors data retrieved in real time through PyMat during simulation or assembly.

The initiation method of the class takes as parameters the PyMat handle and the geometrical information of the link. In each call to the constructor, a SimMechanics body block, a position and rotation sensor are created (Fig. 4c). The sensors connect to a copy of the first coordinate system CS1 and measure the rotations and positions relative to the adjoining coordinate system, which in most cases is a ground point or CS2 from the adjacent block. The output of the rotation sensor is directed to a Simulink VR toolbox block that converts the measured rotation matrix into a rotation axis–angle vector. The axis and angle of rotation

and the position output from the first sensor are combined into a single vector through the use of a Mux block that combines multiple signals or vectors into a single vector. The output of the Mux block is then directed to our S-Function block.

The scale method applies a local uniform transformation to the selected body object and returns the scaling factor, which is used as a parameter for our update method. The update method computes the vectors that fully define all the bodies' coordinate systems and then makes all of the necessary adjustments in the SimMechanics blocks for the model's consistency.

The user can add any number of CS to a link. To change its position and orientation of a CS, one can manipulate the selected CS with mouse, haptic device or wand. To enable the collision detection, VRMDS creates a duplicate physics model for all CS and link objects in a mechanism model. The collision between a marker object and any mechanism object will be detected at run time by calling the build-in viz.phys library. Once a collision is detected, a collision event is triggered and the collided object will be subject to picking and manipulation. Once any object the scene is modified, the physics model and SimMechanics model will be updated accordingly.

### 3.2 Modeling of joints

Joints are physical connection between links. Our interface currently supports revolute (R), prismatic (P), spherical (S) and cylindrical (C), universal (U) joints and weld (W) joints. The S,C and U joints are considered special assemblies of the basic R and P joints. The weld joint is used to rigidly fixing two bodies, i.e. no relative motion between the incident links.

Every joint type modeled in VRMDS simply represents a constraint between two bodies' relative motions. When defining a joint the user must specify the type, source body, destination body, as well as the location of placement. For example, for defining a revolute joint, the user should pick a source and destination axes from the coordinate systems of two different bodies. As a result, the two axes will be aligned, and the degrees of freedom of the destination body will be restrained by imposing the newly defined constraint relative to the source block. As shown in Fig. 4b, each CS represents a local reference position and orientation on a body. The red, green and blue arrows along with the gray sphere represent the  $x$ ,  $y$ ,  $z$  axis and the origin, respectively, in each one of our coordinate systems. Upon the addition of a joint primitive, one of those components will change color to black to indicate the presence of a joint and the way the constraint will be applied.

To demonstrate the assembly process, let us take a spherical joint as an example. Before the assembly, two unconstrained links "Body1" and "Body2" are created. To join them with a S joint, the user selects a CS on the source body (CS2 of Body1) and picks another CS on the destination body (CS1 of Body1). Once they are picked, the color of the origin of the selected CS is changed. The user will then click the "Add S Joint" icon on the control toolbar. VRMDS will create a joint block called "Spherical" in the SimMechanics model during these operations

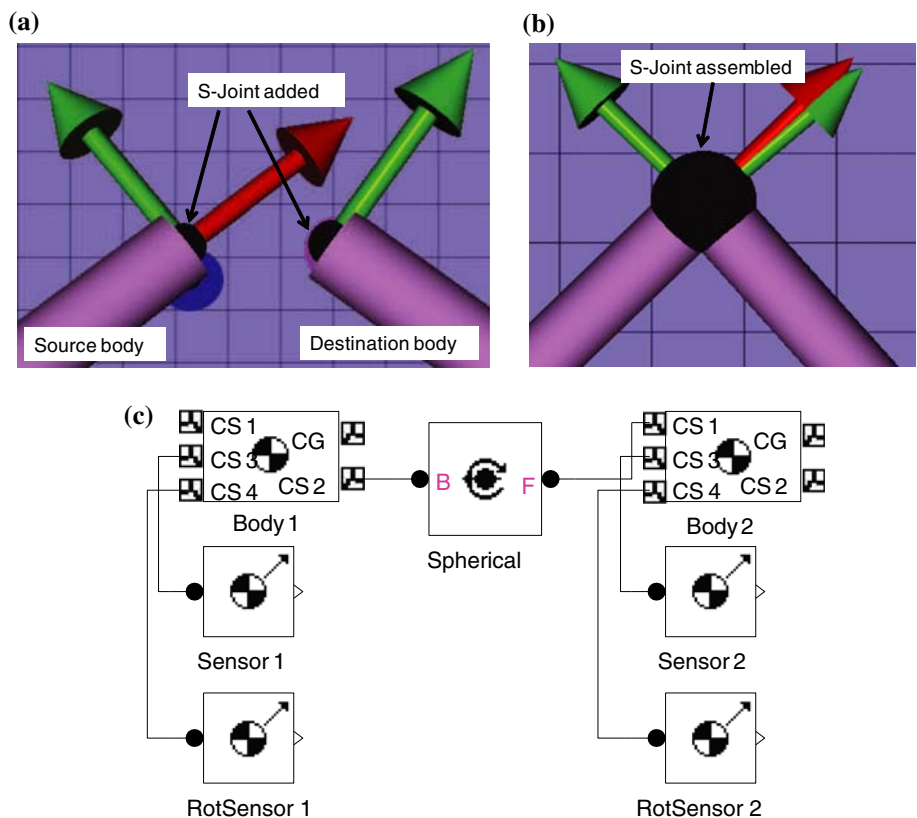
(see Fig. 5c). After that, a system update command (which calls the SimMechanics's constraint solver) is executed to keep consistency between the SimMechanics model and VRMDS model. Figure 5b shows the graphical representation in VRMDS after the assembly.

### 3.3 Mechanism assembly

By default, each mechanism created will have a ground link which is rigidly fixed to a ground point. Assembling mechanisms typically starts from this ground link and continues by defining joints between links. We have implemented disassembled joint and actuators for constructing and actuating mechanisms with complex topologies. Reference points are represented by a single coordinate system and can be attached to a link or to the world. If reference points are attached to a link, their motion follows that of the link.

To build loop or closed chain mechanisms, disassembled joints must be used. These types of joints will be used in cases where user's errors during interactions will result in constraint or any other type of errors or violations. When a link is attached to the ground link, we call the link "adjoining" to the ground. If the follower link is already adjoining to the ground, a corresponding disassembled R/P/C/S joint will be used in order to form a loop linkage. By using a disassembled joint, the system takes care of

**Fig. 5** Two links are connected with a spherical joint **a** before assembly **b** after assembly **c** the link and joint blocks created in SimMechanics



interaction errors and typically adjusts joint parameters to make the assembly feasible. If any tolerance or constraint error shall arise, our error checking will indicate that modifications should be made in the model's topology and assembly should be re-attempted.

### 3.4 Dynamic simulation of mechanisms

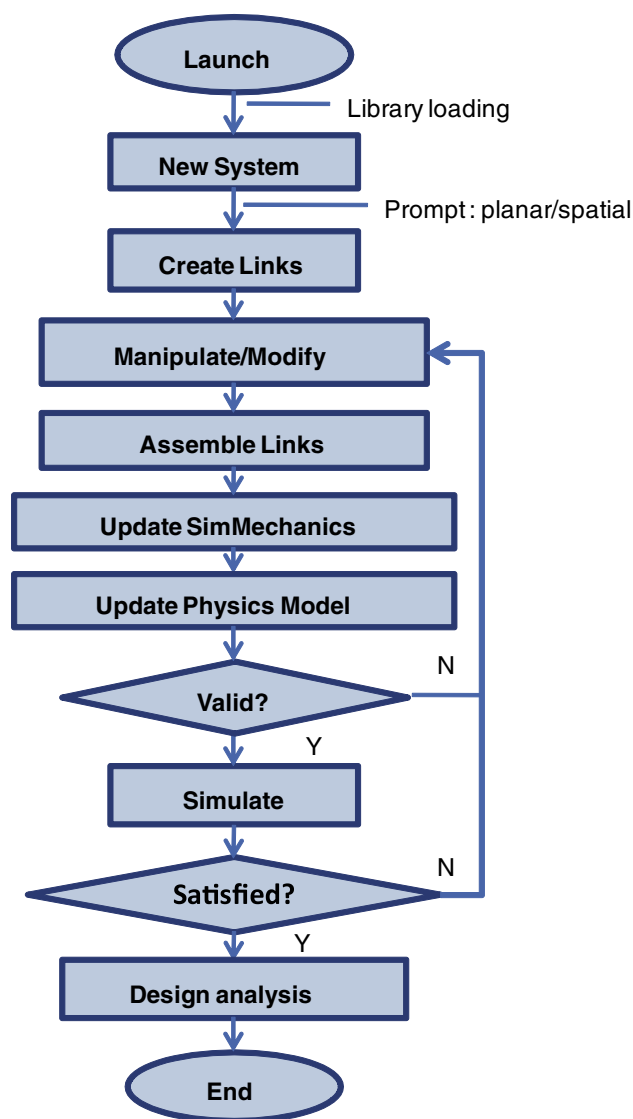
To fully encapsulate the SimMechanics functionality and eliminate the programming and construction of block diagrams, our interface contains a series of built-in MATLAB script files that create bodies, joints, actuators, sensors and set the relevant parameters after interaction takes place. Typically, each M-file is called after a GUI event takes place in VRMDS. For example when the user clicks on the new system button, he will be asked to choose a mechanism type (planar or spatial). Furthermore, the built in M-file associated with that event will create a new SimMechanics system that contains a machine environment with some default settings such as the gravity vector, constraint solver and tolerances among others. Other script files include creation of links, joints, coordinate systems, sensor, actuators and model update.

Our S-function is the most critical component for the communication among all VRMDS modules. Built on top of a level-1 M-file S-function template provided by the Simulink library, our S-function writes in real time our model outputs in the MATLAB workspace. The model outputs are the data retrieved by the position and rotation sensors connected to each of our body blocks. By means of a declaration of a global variable in MATLAB and a synchronization timer in Vizard, the simulation or assembly data become available for real-time simulation rendering inside VRMDS.

## 4 The design process

Figure 6 shows the basic process flow using VRMDS for mechanism design. The process flow follows three major steps: link creation, mechanism assembly and mechanism simulation which are iterated when necessary. The details are described as the following.

The VRMDS design procedure begins by loading dependent plug-ins and libraries including viz, hd, PyMat, etc. for handling visualization, haptic interaction and interfacing with MATLAB. Once the user is ready to model, he/she will have to click on the new system button. As a result, a new SimMechanics default environment gets created and all the respective Simulink, SimMechanics and VR toolbox libraries are loaded, also the designer will be prompted to choose the mechanism type. Next, the process continues by defining multiple bodies in the virtual space;



**Fig. 6** The process flow using VRMDS for mechanism design

simultaneously, the corresponding SimMechanics blocks will be created. All the SimMechanics operations are done in the back scene without interrupting the operations in main program.

After that, the designer would interact with the virtual objects by rotating, translating, scaling and joining bodies to one another as desired. Subsequently, calls to the assemble or update functions should be made to make the changes of interaction effective in both our virtual representation of the model and the SimMechanics system. Also, the physics model will be updated by reading the current data of each object in the model. At the same time, our try and catch built in M-file for error checking will be called to evaluate the model feasibility and a flag will be raised in case a violation occurs. If a violation occurs, the user should go back into the manipulation mode and make the appropriate changes. On the other hand, if the assembly

is successful, the user should proceed to define the sensors or actuators that suit his needs and simulate the system. If the analysis is satisfactory, the designer would exit the design loop and the main program; if not, he/she should re-iterate through the loop either by modifying the current design or creating a new system.

As an example, a single-link single-joint system was created to simulate a simple pendulum oscillating under the influence of gravity. The binary link is created and loaded in its default position and orientation. Then, it is fixed to the world by selecting the  $z$  axis of CS1 and defining a revolute joint. Finally, the system is simulated and the motion is shown in Fig. 7a. Figure 7b shows the equivalent SimMechanics system block diagram. It can be appreciated that by two button events of our interface we can create a trivial model more intuitively than by making use of other software such as SimMechanics.

### 5 Case study: a RSSP spatial loop linkage

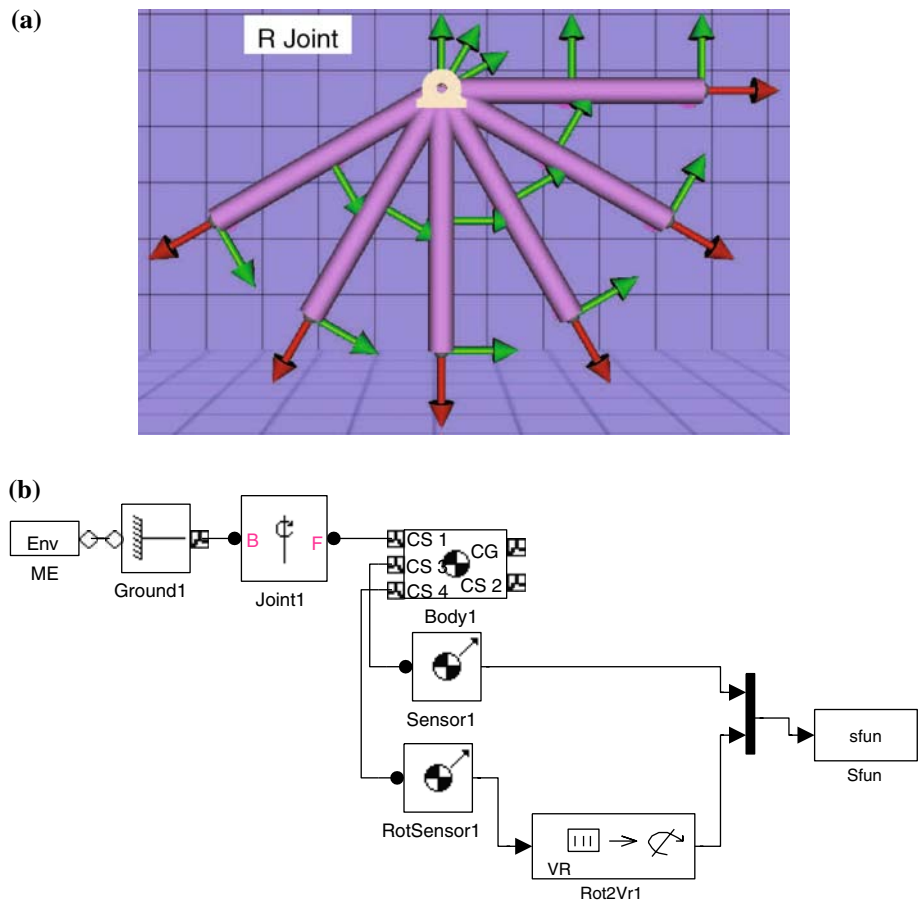
As a case study, we will examine a spatial RSSP four bar linkage whose schematic is shown in Fig. 8. This mechanism consists of a crank link of length  $l_1$  which is fixed to

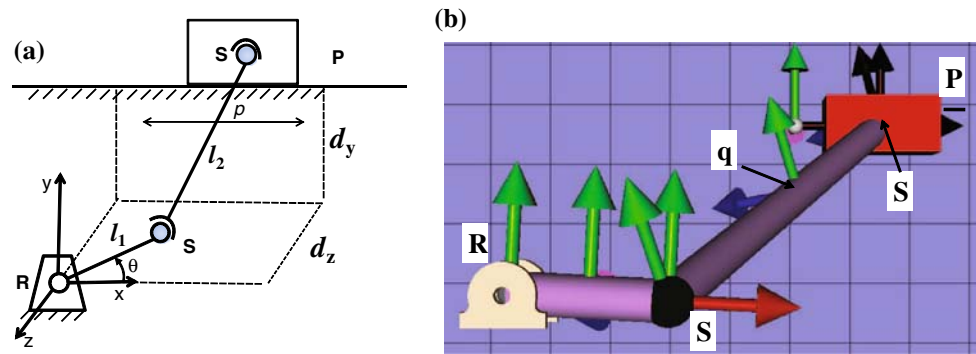
the ground by a revolute joint rotating about the global  $z$  axis. Also, a coupler link of length  $l_2$  is connected to the crank and a sliding link by spherical joints. As stated earlier, all links are assumed to have a uniform shape. Finally, the loop closure joint is a prismatic joint defined at an offset point  $(0, d_y, d_z)$  from the global coordinate system allowing only one translational degree of freedom ( $P$ ) along the slider's local  $x$  axis.

In VRMDS, we begin by defining a new spatial mechanism system. Then, we initialize three of our binary link objects of different geometries. Link 1 and 2 are cylindrical, and link 3 is cubical. Subsequently, the links are sized, manipulated and assembled as mentioned previously. Link 1 is grounded to the world by selecting the  $z$  axis of its CS1 coordinate system. Then, links 1 and 2 are joined by selecting their CS2 and CS1, respectively, and clicking the spherical joint button. Similarly, the joint between link 2 and link 3 is created. At last, we define a coordinate system in space and make use of the loop closure SimMechanics disassembled prismatic joint to take care of the assembly accordingly. Figure 9 illustrates the VRMDS model of the linkage during simulation.

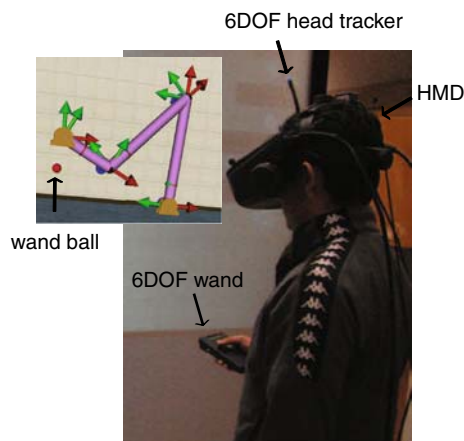
After creating the mechanism model, VRMDS also provides the functionalities of importing environment

**Fig. 7** A single pendulum created by VRMDS: **a** dynamic motion **b** the SimMechanics model





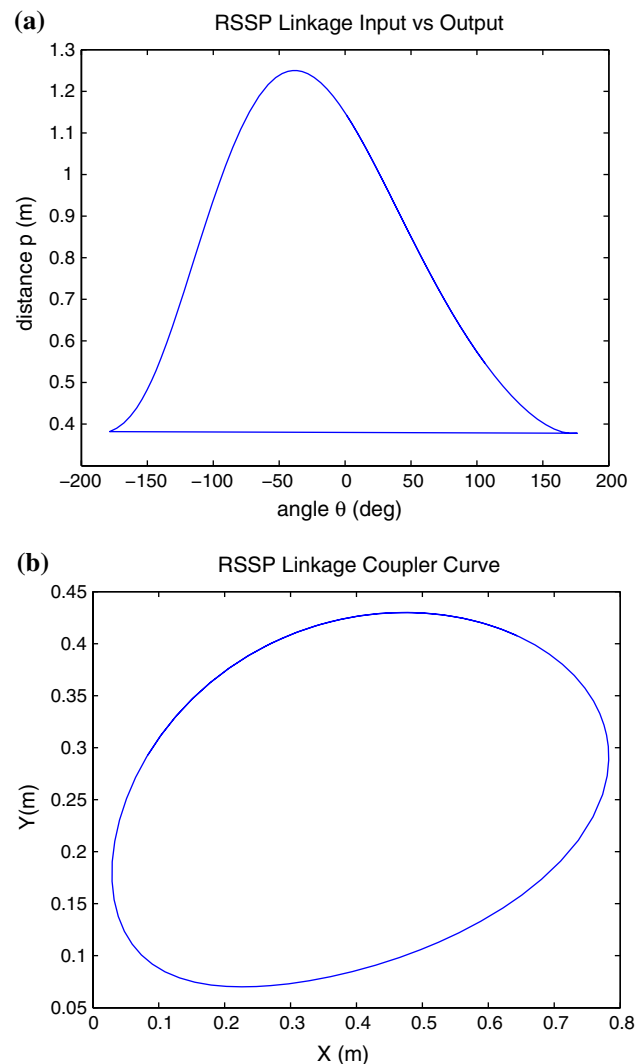
**Fig. 8** **a** The schematic of a RSSP spatial linkage, **b** The RSSP spatial linkage created in VRMDS



**Fig. 9** A spatial linkage created in an immersive virtual room environment

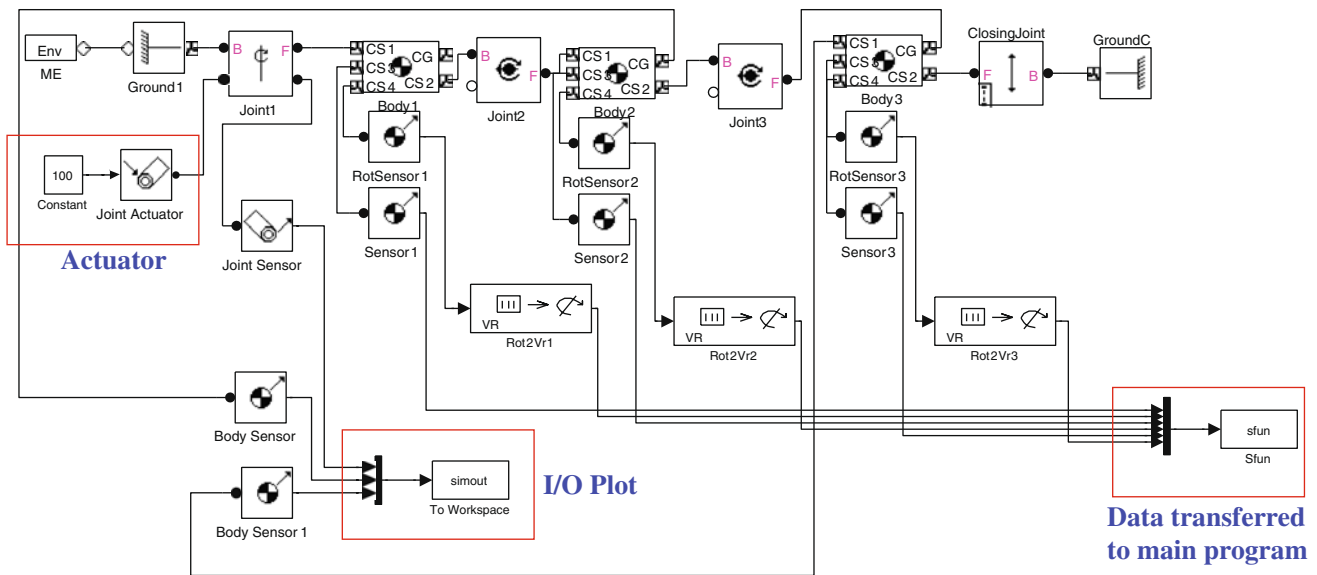
models for further evaluation. Very often in mechanism design, the allowable design space is bounded by the surrounding environment. Understanding the spatial relationship between the mechanism and the environment is therefore critical for the success of the produced design concepts. Figure 9 shows a typical use of the VRMDS in an immersive environment where a HMD with a 6DOF head tracker and a 6DOF wand for picking and manipulation. The mechanism can be placed in a realistic room simulating the workshop/factory environment. This is very useful for checking space relation and possible interference between the mechanism and its surrounding environment during the simulation. And the user can walk around the environment to inspect the mechanism motion from different perspectives. The other cooperators can also inspect the mechanisms through the power wall display.

For our design analysis, a constant torque joint actuator was defined to the R joint. In addition, by selecting CS2 of the coupler link (point q in Fig. 9), the sliding box coordinate system and clicking on our sensor button we can obtain the positions, velocities and accelerations reached by the bodies during simulation. Every call to the add sensor function will take either a coordinate system or a joint



**Fig. 10** The design analysis of the RSSP linkage created in MATLAB. **a** The plot of the input  $\theta$  vs. the output  $p$ , **b** the  $x$ - $y$  plot of the trajectory of the CG of the coupler link

object to measure the desired parameters, and the results will be saved into the MATLAB workspace. Figure 10a shows the relationship between the input angle  $\theta$  and the



**Fig. 11** The SimMechanics model for the RSSP spatial mechanism created by the main program

sliding distance  $p$ . Figure 10b shows the  $x$  and  $y$  coordinates of the center of mass of the coupler link. The  $z$  coordinates of this spatial curve are not plotted for the sake of conciseness.

Figure 11 shows the complete SimMechanics block diagram of our RSSP linkage. In addition to the already introduced body block sets and joints, we can observe how the sensor and actuator capabilities are implemented. The user can retrieve any data from any of the defined bodies or their coordinate systems or any coordinate system that might be defined as a point of interest. This simulation data will be available in the MATLAB workspace for analysis after simulation is completed. The actuator is added by selecting a joint or body object and a constant torque or generalized force will be applied depending on the object to be actuated. Finally, the data retrieved during simulation by our body sensors and other blocks is sent to a special type of sink that in conjunction with our S-Function allows for real-time data sharing to the main program.

It is evident that adding the multiple blocks, making the right connections and setting all the different parameters correctly make the modeling a complex procedure. Any designer who might want to use this powerful tool would have to go through an instructive process in which the learning curve could be very long.

## 6 Discussion and conclusion

This paper describes a Virtual Reality Mechanism Design Studio (VRMDS) for supporting the interactive design and

simulation of general mechanisms. The program takes advantage of Vizard library for VR visualization and interaction and MATLAB's SimMechanics toolbox for multi-body dynamics simulation. The functionalities of the system include (a) the support of VR equipments immersive displays, haptic devices and tracking systems, (b) convenient graphical user interfaces for facilitating interactive design, (c) dynamic simulation of mechanisms and (d) realistic surrounding environment simulation. The design process of using this system consists of three main steps: creation, assembly and simulation, which can be iterated when needed. Through the use of VRMDS, a designer can quickly create and evaluate a design concept by just a few simple operations.

The benefits of VRMDS are described as the following. First of all, the design process is intuitive, i.e. seeing what you build. As long as the designer has a basic understanding of mechanisms, he/she can pick up this design tool fairly quickly. This shortens the conceptual design process or allow more ideas explored in the same period of time. This tool can also be used as a classroom tool for teaching mechanism design. Second, the system can run on both a desktop environment or an immersive VR environment while most of VR programs can only run on expensive multi-wall environment. The former environment is well suitable for daily based design activities. And the latter environment is recommended for conceptual design activities such as team discussion and brainstorming. Third, VRMDS provides a full immersive visualization which is very important for designing spatial mechanisms. Finally, VRMDS provides a full dynamic simulation of mechanisms with almost arbitrary topologies. This versatility is

especially important to the conceptual design stage as the designer may have to explore and evaluate a large number of design alternatives.

In future versions of VRMDS, we intend to support more joint types and machine elements. Motion control actuators as well as fully embedding the analysis mode will also be attempted. Other ideas include exploring both SimMechanics and Vizard CAD assembly files importing/exporting capabilities to support the analysis, visualization and interaction of more complicated mechanisms. Finally we plan to conduct an useability study to measure the usefulness of the system.

**Acknowledgements** The authors acknowledge the support of the National Science Foundation grant no. 0900517.

## References

- Antonya C, Talaba D (2007) Design evaluation and modification of mechanical systems in virtual environments. *Virtual Real* 11(4):275–285. doi:[10.1007/s10055-007-0074-6](https://doi.org/10.1007/s10055-007-0074-6)
- Barbieri L, Bruno F, Caruso F, Muzzupappa M (2008) Innovative integration techniques between virtual reality systems and cax tools. *The Int J Adv Manuf Technol* 38(11):1085–1097. doi:[10.1007/s00170-007-1160-3](https://doi.org/10.1007/s00170-007-1160-3)
- Bierbaum A, Just C, Hartling P, Meinert K, Baker A, 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, Washington, DC, USA, p. 89. ISBN 0-7695-0948-7
- Bourdot P, Convard T, Picon F, Ammi M, Touraine D, Vzien JM (2008) Vr-cad integration: multimodal immersive interaction and advanced haptic paradigms for implicit edition of cad models. *Computer-aided design in press corrected proof*. doi:[10.1016/j.cad.2008.10.014](https://doi.org/10.1016/j.cad.2008.10.014). <http://www.sciencedirect.com/science/article/B6TYR-4TW6HV4-1/2/ca24badbdaa9cb7f9a0dcb872c9ddb77>
- Brough JE, Schwartz M, Gupta SK, Anand DK, Kavetsky R, Pettersen R (2007) Towards the development of a virtual environment-based training system for mechanical assembly operations. *Virtual Real* 11(4):189–206. doi:[10.1007/s10055-007-0076-4](https://doi.org/10.1007/s10055-007-0076-4)
- Burdea G, Coiffet P (2002) *Virtual reality technology second edition* Wiley, Hoboken
- Cheng H, Trang T (2006) Object-oriented interactive mechanism design and analysis. *Eng Comput* 21(3):237–246. doi:[10.1007/s00366-005-0008-4](https://doi.org/10.1007/s00366-005-0008-4)
- Erlemeier TA (2006) The development of a virtual reality based cad system for design review. Ms thesis, Iowa State University, Ames, Iowa, USA
- Fiorentino M, Monno G, 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*
- Furlong TJ, Vance JM, Larochelle PM (1999) Spherical mechanism synthesis in virtual reality. *ASME J Mech Des* 121(4):515–520. doi:[10.1115/1.2829491](https://doi.org/10.1115/1.2829491). <http://link.aip.org/link/?JMD/121/515/1>
- Jayaram S, Jayaram U, Wang Y, Tirumali H, Lyons K, Hart P (1999) Vade: a virtual assembly design environment. *IEEE Comput Graph Appl* 19(6):44–50. doi:[10.1109/38.799739](https://doi.org/10.1109/38.799739)
- Jayaram S, Vance J, Gadh R, Jayaram U, Srinivasan H (2001) Assessment of vr technology and its applications to engineering problems. *J Comput Info Sci Eng* 1(1):72–83. doi:[10.1115/1.1353846](https://doi.org/10.1115/1.1353846)
- Larochelle PM (1998) Spades: Software for synthesizing spatial 4c linkages. In: *Proceedings of the ASME DETC'98*, Sept. 13–16, Atlanta, GA
- Larochelle PM, Dooley J, Murray A, McCarthy J (1993) Sphinx: software for synthesizing spherical 4r mechanisms. In: *Proceedings of the 1993 NSF design and manufacturing systems conference*. University of North Carolina at Charlotte
- McCarthy JM (2000) *Geometric design of linkages*. Springer, New York
- Nelson L, Erdman AG (1994) Recent enhancements to the linkage-6 synthesis package, including circuit rectification. In: *Proceedings of the ASME design technical conference, mechanism synthesis and analysis, DE*, vol. 70, no. 3. pp. 263–272
- Seth A, Su HJ, Vance JM (2008) Development of a dual-handed haptic assembly system: sharp. *ASME J Comput Inf Sci Eng* 8(4):044502. doi:[10.1115/1.3006306](https://doi.org/10.1115/1.3006306). <http://link.aip.org/link/?CIS/8/044502/1>
- Seth U, Su HJ, Vance JM (2009) A virtual reality interface for the design of compliant mechanisms. In: *Proceedings of ASME IDETC/CIE 2009*, August 30–September 2. San Diego, CA
- Stan SD, Manic M, Maties V, Balan R (2008) A novel virtual reality robot interface for isoglide3 parallel robot. In: *ICIRA (1)*, pp. 1265–1275
- Su HJ, Collins CL, McCarthy JM (2002) An extensible java applet for spatial linkage synthesis. In: *ASME international design engineering technical conferences*, Sept.29–Oct.2, Montreal, Canada
- Taylor II RM, Hudson TC, Seeger A, Weber H, Juliano J, Helser AT (2001) Vrpn: a device-independent, network-transparent vr peripheral system. In: *VRST '01: Proceedings of the ACM symposium on virtual reality software and technology*. ACM, New York, pp. 55–61. ISBN 1-58113-427-4. doi:[10.1145/505008.505019](https://doi.org/10.1145/505008.505019)
- Tsai LW (2000) *Mechanism design: enumeration of kinematic structures according to function*. CRC Press, Boca Raton
- Ullman DG (2003) *The mechanical design process*. McGraw Hill, New York
- Vance JM, Larochelle PM, Dorozhkin DV, Spatial VR (2002) Designing spatial mechanisms using virtual reality. In: *Proceedings of ASME IDETC/CIE 2004*, September 29–October 2. Montreal, Canada
- Wan H, Gao S, Peng Q, Dai G, Zhang F (2004) Mivas: a multi-modal immersive virtual assembly system. In: *ASME 2004 design engineering technical conferences and computers and information in engineering conference*. Salt Lake City
- Zhu W (2008) A methodology for building up an infrastructure of haptically enhanced computer-aided design systems. *J Comput Inf Sci Eng* 8(4):041004. doi:[10.1115/1.2988340](https://doi.org/10.1115/1.2988340). <http://link.aip.org/link/?CIS/8/041004/1>