This manuscript titled “A multichannel visualization module for virtual manufacturing” has not been published already, nor is it under consideration for publication or in press elsewhere.

Word count: 6,111 words
A multichannel visualization module for virtual manufacturing

Yong-Sik Kim a, Jeongsam Yang b,* and Soonhung Han a

mk37do@icad.kaist.ac.kr, jyang@ajou.ac.kr, shhan@kaist.ac.kr

a Department of Mechanical Engineering
Korea Advanced Institute of Science & Technology
373-1, Gusong-Dong, Yusong-Gu,
Daejeon 305-701, Korea
Tel.: +82 42 869 3040
Fax: +82 42 862 9224

b,* Industrial & Information Systems Engineering
Ajou University
San 5, Wonchun-dong, Yeongtong-gu, Suwon 443-749, Korea
Tel.: +82 31 219 2335
Fax: +82 31 219 1610

* Corresponding author
A multichannel visualization module for virtual manufacturing
Abstract

Immersive virtual reality (VR) for manufacturing planning helps reduce product development time periods and improves the quality of the production. However, immersive VR equipment is generally expensive, both in terms of development and purchase. Users must also spend time to manually repair complex 3D shapes due to imperfect translations between the 3D engineering CAD models and the proprietary format of the VR system. In this paper, the proposed VR module uses a commercial virtual manufacturing system (VMS) as the viewer of an immersive VR system where a cluster of PCs adopts a modified simulation algorithm. The proposed module removes the data translation process and ensures good coherence among visual channels. For the experiments, the proposed module has been interfaced with the Delmia VMS. The clustering modules can reduce the cost of VR experiments while offering good performance.

Keywords: CAD; Immersive virtual reality; Multichannel visualization module; PC clusters; Virtual manufacturing system
1. Introduction

The recent combination of virtual reality (VR) technology and virtual manufacturing systems (VMSs) can enhance competitiveness in the automotive or shipbuilding industries. Designers can control the virtual machines inside a virtual factory and evaluate products before production begins. These evaluations or verifications can help eliminate unexpected errors, thereby reducing the overall processing time and cost.

Before the VMS has been introduced, decisions regarding job shops are made based on 2D shapes displayed on small monitors [1]. The process of abstracting 2D models from real factories is prone to errors due to the incomplete abstraction of the mathematical models and the limitations of the 2D model itself. Therefore, considerable experience and cost are required for accurate results [2,3].

As a countermeasure, companies expand into 3D VMSs that can accurately represent the existing or new factory layout [4,6]. This method, however, still cannot effectively deliver a sense of reality nor enable users to correctly grasp the situation because the large complex VMS results are displayed on a small, low resolution monitor. The experiments show that the editing process is easier with a mouse and keyboard, whereas error checking work is easier with immersive VR [5]. Because the main purpose of the VMS is to verify the proposed design and immersive VR has more information in quantity and quality, immersive VR is better than traditional monitors. Due to this advantage, companies that currently use VMS are shifting to immersive VR technology [7-9].
There are various immersive VR technologies such as head-mounted displays (HMDs), Reality Centers, and CAVEs (Computer Aided Virtual Environment). Practicing designers and researchers generally prefer CAVEs or Reality Centers to HMDs, because a large-scale screen is more convenient when communicating and cooperating with other departments.

A large-scale screen can be controlled in two ways: by using specialized graphic hardware, such as Onyx or Octane, or by using a cluster of PCs. Because the cluster of PCs is cheap and flexible enough to meet the need of various environments, many users prefer a cluster of PCs. However, an immersive VR with a cluster of PCs has a drawback: the simulation process of the VMS has a higher priority than the clustering process, so the clustering process cannot use enough CPU power to ensure the coherence of all the screens. The VMS also has a problem: when the immersive VR technology is applied to a commercial VMS, such as the Delmia of Dassault Systemes or the e-Factory of UGS, the conversion of data formats between the immersive VR system and the VMS causes data loss and distortion [12].

In this paper, the above problems are investigated, and a multichannel visualization system is proposed to improve the performance of immersive VR systems where VMSs are combined. Two approaches to improve the performances are proposed for the multichannel visualization system. In the first approach, a commercial VMS is used as the viewer by adding immersive VR modules into the VMS. This approach does not require a data conversion process, thus enabling the data loss and distortion errors that occur during the conversion of the data from the VMS to the VR systems to be eliminated. In the second approach, a modified simulation algorithm that provides the coherence of the VMSs in a
cluster of PCs is proposed. Because the simulation engine in the VMS has the exclusive right to use the CPU, the cluster module does not have priority access to the VMS that provides coherence during the simulation. The modified simulation algorithm divides the simulation into short-time intervals and the cluster module can provide coherence between the intervals of the short-time simulations. Because short-time simulations appear to be a seamless simulation, the discrepancy between the screens is too small for users to recognize.
2. Related work

2.1. Review of previous studies

Much research on the VMS has focused on proving the usefulness of VR. Table 1 compares the related studies and the method proposed in this paper. Bazargan-Lari [10] sketched a layout of a virtual factory on a 2D monitor and found the optimized layout for equipment by using mathematical algorithms. Although the proposed method can express the layout numerically, 3D objects are projected onto the 2D plane and it can’t handle 3D layout.

Weyrich et al. [4] developed a workbench for a virtual manufacturing factory. The workbench is a virtual working table that enables 3D projections of virtual objects. The user can observe the 3D factory from the workbench by wearing LCD shutter glasses and can interact with the workbench system using a six-dimensional input device. The workbench system, which comprises hierarchical object structures, can handle large amounts of shape data, level-of-detail (LOD), and texture mapping. Weyrich et al. used the SGI’s Onyx system as the hardware for the immersive environment and their own libraries for the visualizations. Weyrich et al.’s study focused on expanding the application fields of VR. Additional procedures, such as data conversion, are required if data created by other VMSs is to be used within the workbench system.

To arrange the manufacturing cell equipment, Korves et al. [5] developed an interactive layout planning system using an HMD. This system uses an immersive interface in which
the necessary equipment is selected and moved from the database to the workspace and
where the working environment is arranged according to the user’s choice. Whenever the
user violates a predefined constraint, audiovisual feedback is provided. To show the
usefulness of VR, Korves et al. compared immersive VR using an HMD and a 3D mouse
with a nonimmersive VR using monitors and a conventional mouse. According to the
comparison, when the user has experience with layout planning and is familiar with
immersive VR, the immersive VR environment proved to be better at error checking than
the nonimmersive environment.

Ng et al. [6] used a virtual environment to investigate the automated design and planning
of a cable harness. They designed automated cable routes using a method of artificial
intelligence to infer the 3D space. The user observed the 3D images through an HMD, used
a 3D mouse for navigation and input. It also provided haptic feedback with audiovisual
expressions to detect collisions.

Choi et al. [11] developed a modeling and simulation solution, VMFactory, for virtual
factory applications. Based on OpenGL, VMFactory provides functions such as the
modeling of virtual equipment and the factory layout; a command language for model-
based simulation; and schedule simulations, which are connected to the manufacturing
execution system (MES). However, VMFactory does not consider immersive VR.
Table 1  Comparison of the related studies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
<td>2D monitor</td>
<td>Workbench, LCD shutter glasses (stereoscopy)</td>
<td>SGI Indigo2</td>
<td>HP Workstation</td>
<td>2D monitor</td>
<td>Multichannel visualization system</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>N/A</td>
<td>IRIX</td>
<td>IRIX</td>
<td>HP-UX</td>
<td>Windows</td>
<td>Windows</td>
</tr>
<tr>
<td><strong>Input device</strong></td>
<td>Keyboard</td>
<td>Space joystick, dataglove, 3D mouse</td>
<td>HMD, 2D or 3D mouse</td>
<td>HMD, 3D mouse</td>
<td>2D or 3D mouse, keyboard</td>
<td>Keyboard</td>
</tr>
<tr>
<td><strong>Visualization method</strong></td>
<td>2D display module</td>
<td>Proprietary library</td>
<td>N/A</td>
<td>N/A</td>
<td>OpenGL</td>
<td>Delmia viewer</td>
</tr>
<tr>
<td><strong>VMS interface</strong></td>
<td>N/A</td>
<td>Partially handles layout issues in a virtual environment N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Provides programming environment for VMFactory</td>
<td>Interfaces with Delmia</td>
</tr>
<tr>
<td><strong>Data format</strong></td>
<td>N/A</td>
<td>-Uses CAD formats such as DXF and IGES after repairing</td>
<td>N/A</td>
<td>-Additional repairing is required during conversion</td>
<td>-Proprietary data format</td>
<td>Converts to VRML format but additional repairing is required</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td>-Mathematical approach for layout optimization -Focuses on finding effective algorithms for equipment layout</td>
<td>-Focuses on building a hardware interface and virtual environment -Applicable to the level of detail -Uses Multigen to modify 3D shapes</td>
<td>N/A</td>
<td>-Focuses on the usefulness of virtual reality for layout planning</td>
<td>-Focuses on the cable harness design and planning within virtual environments</td>
<td>-Provides modeling of virtual equipment and factory layouts, command language for model-based simulations, and schedule simulations</td>
</tr>
</tbody>
</table>
2.2 Commercial VMSs

Delmia of Dassault Systemes allows the problems that can occur during the manufacturing process to be evaluated through simulations based on a virtual factory. When human-like avatars are included in the process, Delmia provides an environment where the avatars imitate the worker’s actions.

e-Factory of UGS can arrange and analyze equipment in a virtual factory. e-Factory comprises FactoryCAD for designing the factory layout and FactoryFlow for analyzing the factory layout. FactoryCAD can provide the optimal layout by selecting information from the shape libraries of the cranes; conveyors, and robots. It also provides purchase costs, power, cycle times, and connection times. FactoryFlow helps the user understand the process of layouts in a virtual factory and provides ideas for improvement by graphically showing costs and usages, enabling the user to better analyze the flow of products during the manufacturing process.

Delmia and e-Factory usually visualize the process simulations on desktop monitors. To build an immersive VR, expensive dedicated hardware, such as SGI’s Onyx system is required. To visualize the models created in other VMSs, the data must be converted into a neutral format such as VRML. The process of converting the data can introduce distortions and data loss depending on the complexity of the shapes [12]. For the kinematics information, the user needs to input additional information to avoid incomplete conversions. For instance, the expected movement of a camera is along the axis, but after conversion, the camera is found to behave differently than expected.
Mun et al. [13] introduced a methodology that can convert military data using Synthetic Environment Data Representation and Interchange Specification (SEDRIS). However, the standardization of SEDRIS is underway, and there are not enough utility tools at present.

2.3 Cluster-based VR libraries

There are VR libraries that allow PC clusters to produce synchronized views of the VR environment frame-by-frame. The NAVERLIB library, which is based on the OpenGL Performer, has the advantages of easy-to-use editing functions and compatibility with multimedia [14]. CAVELib, which runs on the SGI computers, supports VR clustering [15] and shares fixed data through the cluster, such as the navigational matrix and input device values. The DIVERSE library supports a networking infrastructure; however it does not explicitly support clustering [16,17]. DIVERSE provides a framework for networked shared memory that can be used to connect input devices to a VR application or to construct shared virtual environments. VR Juggler, a VR library similar in function to the CAVELib, provides two extensions that support clustering: Net Juggler and Cluster Juggler [18]. In both cases, a full copy of the VR Juggler application runs on each render node of the cluster. The applications share input events and are synchronized at the end of each drawn frame. Furthermore, some VR systems like R3-Interactive [20,21] focus only on controlling the display devices to produce seamless displays. If overlapping or curved screens are used, this kind of system is useful.
Table 2 compares the four cluster-based VR libraries. For visualizations, these libraries use graphic libraries such as OpenGL or OpenGL Performer. Some of the VR libraries can be connected to immersive VR equipment or have network functions that support immersive VR features. The VR libraries cannot directly read engineering data such as CAD; therefore, the data must be converted into a neutral format such as VRML.

### Table 2  Comparison of four cluster-based VR libraries

<table>
<thead>
<tr>
<th></th>
<th>NAVERLIB</th>
<th>Net Juggler</th>
<th>CAVELib</th>
<th>DIVERSE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS</strong></td>
<td>Redhat Linux,</td>
<td>SGI IRIX, Linux,</td>
<td>SGI IRIX, Linux,</td>
<td>SGI IRIX, Linux</td>
</tr>
<tr>
<td></td>
<td>Windows</td>
<td>Windows</td>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td><strong>VR devices</strong></td>
<td>CAVE-like system,</td>
<td>Trackers,</td>
<td>Variety of VR</td>
<td>HMD, tracker,</td>
</tr>
<tr>
<td></td>
<td>I/O device</td>
<td>datagloves, I/O</td>
<td>equipment</td>
<td>joystick, motion base</td>
</tr>
<tr>
<td><strong>Graphic library</strong></td>
<td>-Supports</td>
<td>OpenGL-based</td>
<td>OpenGL Performer</td>
<td>OpenGL Performer–based</td>
</tr>
<tr>
<td></td>
<td>multichannel</td>
<td>visualization</td>
<td>or OpenGL-based</td>
<td>visualization</td>
</tr>
<tr>
<td></td>
<td>functions</td>
<td></td>
<td>visualization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-OpenGL Performer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>or OpenGL-based</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>visualization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Data format</strong></td>
<td>Compatible with</td>
<td>N/A</td>
<td>Compatible with</td>
<td>Compatible with</td>
</tr>
<tr>
<td></td>
<td>OpenGL Performer</td>
<td></td>
<td>OpenGL Performer</td>
<td>OpenGL Performer</td>
</tr>
<tr>
<td><strong>Features</strong></td>
<td>-Easily manipulated with XML script</td>
<td>- Supports the cluster for VR Juggler</td>
<td>-Suitable for a multichannel system</td>
<td>- Supports CAVE and Immersive Desk</td>
</tr>
<tr>
<td></td>
<td>- Supports rendering functions of 3D video signals</td>
<td></td>
<td></td>
<td>- Implements high-end applications such as a VR Juggler</td>
</tr>
<tr>
<td></td>
<td>- Enables a cluster of PCs to be built and synchronized</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. Design of multichannel visualization module

3.1 System requirements

To build a multichannel visualization system based on a cluster of PCs, the time for the screen switching at animation needs to be synchronized and the consistency of the property information ensured so that several small screens can function logically as one large screen.

3.1.1 Time synchronization for screen switching

When a model is displayed with multichannel screens, each screen displays different parts of the model; therefore, there are differences in the graphic load that each PC needs to handle. The graphic load differences cause time delays in completing the rendering of certain frames on each PC. As a result, some PCs with a heavy load can cause total system to be slow. The user only sees screens that appear disjointed instead of logically synchronized images. Fig. 1 shows a model of landing gear on three screens. Differences in the rendering time occur because the graphic load of the left screen is greater than that of the middle and right screens, thereby causing time delays. The rendering speed is 7 milliseconds (ms) for the left screen, 4 ms for the middle screen, and 2.8 ms for the right screen; and these differences cause the model to appear irregularly. Network traffic problems may also disrupt the synchronization of the screen switching: sending large amounts of visualization data can slow the network down because the network’s bandwidths are limited. The refreshed data should be transmitted to the slave PCs after
being confirmed by the master PC each time a frame is displayed.

![Diagram showing differences in rendering time due to the graphic load.](image)

**Fig. 1.** Differences in rendering time due to the graphic load.

When a cluster of PCs is used to display a single image, all screens connected to the PCs must be refreshed simultaneously as if they form a single screen. The images should not be projected on the screen immediately after being drawn on the frame buffers; rather, the images should be synchronized with the other frames. Therefore, to display the images
from each PC on several independent screens, the connected screens need to be synchronized.

To allow the simultaneous display of frames after the corresponding images have been displayed on slave PC, the master PC sends a signal notifying each slave PC that the display of frames has been completed. When all the images are drawn from the frame buffers of all the PCs, the simultaneous swap buffering allows the entire image to be synchronized. Although the time taken for each image data to be drawn from each frame buffer may differ, the screen display can be performed simultaneously.

3.1.2. Consistency of property information

To achieve a single logical display, the property information also needs to be synchronized; for example, color properties. The field of view (FOV) also needs to be synchronized from various angles. Fig. 2 shows an example of inconsistent property information. The left scene is what the user wants to see. The image transformation by the user’s input influences the left picture but not the right picture. This inconsistent property occurs because the color information held in both slave PCs were not updated consistently.
For a virtual factory with a multichannel system, various situations should be considered, such as the dynamic movement of equipment, color changes that reflect the condition of the equipment, and changes in the FOV. The changes in properties should not be limited but should be synchronized to allow consistent display under a multichannel system.

3.2 Module design

To construct an immersive VR environment, the commercial VMS Delmia has been interfaced with a multichannel visualization system based on a cluster of PCs.

3.2.1 Design of the control module
As shown in Fig. 3, the multichannel visualization system consists of a master PC, which controls the system, and a cluster of slave PCs.

Due to the high performance of today’s PCs, network load is the key factor that influences performance. When the model data are located only on the master PC, the master PC must transfer both instructions and data. On the other hand, if the data are located on every slave PCs, only the instructions and indicators to the data modified by the instructions need to be transmitted and this enables the master PC to control all slave PCs faster. For example, the WireGL, a centralized multichannel library, requires such a high network load so much that its animations appear to be slow. Therefore, a distributed system could reduce the network load and accelerate the overall system speed. Thus, to improve the system speed, a distributed system has been adopted. This type of system requires that both the model data and the VMS be installed on each slave PC.
The master PC comprises a user interface module and a global control module. The user interface module instructs the global control module so that the multichannel visualization system can respond to the user. The global control module supervises the synchronization of time and properties by controlling the cluster of slave PCs.

Each slave PC comprises a local control module and a translator module, and has local information about its relative geometrical position that allows coherent images to be displayed. The local control module supervises the translator module in executing commands received from the global control module, and it sets the local view by combining the local position with the received data. The translator module translates the commands so that they can be executed in the VMS.
The global control module, which synchronizes the screen switching and maintains the consistency of the property information, supervises the slave PCs in real time. After checking the user input data and the script data for dynamic movement, a signal that controls the screen is sent to the slave PCs if any of the data are modified. The script data comes from the VMS and controls the model’s dynamic movement.

When the screen has been completely updated, the local control module sends a completion signal to the global control module. The global control module waits for the signals from slave PCs until the completion signals are received from all slave PCs. Although differences in the screen refreshing rate may occur due to the graphic load and network traffic, the control module can overcome the differences. When the screen has been completely refreshed, the global control module signals the next frame.

Fig. 4 shows the relationship and data flow between the global control module of the master PC, and the local control module and translator module of each slave PC. The arrows between the modules show that the altered image data inside the corresponding frame, as well as the related data such as user commands, are transmitted between the modules. The unchanged parts of the image are not transmitted because the VMS of the slave PC already has the data. At the start of every frame, the global control module confirms which factors—such as the user input, previously recorded dynamic data, and time—may cause an alteration, and it sends the data that require alterations to each slave PC. After receiving the data from the local control module, the translator module converts the data into the appropriate commands that are then processed by the VMS.
3.2.2 Design of the translator module

The translator module converts the data received from the local control module into a command that can be processed by the VMS. The content of the converted data includes the FOV data that changes with the user location: the data for starting, stopping, and pausing the animated movement, the data for screen refreshing, and the data for the transmission of signals to show the completion of the refreshing process. For the multichannel visualization system, Dassault Systemes’s Delmia, a commercial VMS, was used. To access the
simulation engine of Delmia, a translator module using command line interpreter (CLI) [19] has been implemented. By externally approaching the simulation engines, the CLI, which is provided as the API of Delmia, can control the commands related to the visualization.
4. Implementation and experiment

4.1 Implementation environment

Table 3 shows the environment in which the proposed method is implemented. Windows 2000 Professional was used as the operating system. The global control module and the local control modules were implemented on the basis of MS-Win32 API, and Microsoft Foundation Class (MFC) was used for the user interface. The translator module connected to the VMS was implemented with the aid of the CLI to provide access to Delmia’s simulation engine. The Nettools program was used to control the network between the master PC and the slave PCs, and, instead of using TCP/IP, the faster UDP/IP was used to guarantee network stability.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Windows 2000 Professional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global &amp; local control modules</td>
<td>C++, MS-Win32 API, and MFC</td>
</tr>
<tr>
<td>Translator module</td>
<td>C++, CLI, and Nettools</td>
</tr>
<tr>
<td>Networking</td>
<td>UDP/IP and Nettools</td>
</tr>
<tr>
<td>VMS</td>
<td>Delmia of Dassault Systemes</td>
</tr>
</tbody>
</table>

As shown in Fig. 5, the multichannel visualization system based on a cluster of PCs has five channels that provide the user with a sense of immersion. Each channel has a 120 inch
screen, a slave PC, and a beam projector. To improve the immersive experience, the screens surround the user, and the beam projectors were suspended from the ceiling to give frontal projections, which give brighter images with lower costs. The surrounding screens provide a similar experience to that of the Reality Center. It is convenient for cooperative work in the manufacturing industry and provides a wider FOV than wall type display screens. After considering the performance of the equipment, the conditions of the ambient environment, and the size constraints, the system was arranged to give the user a 45 degree horizontal FOV and a 45 degree vertical FOV for each screen. The overall horizontal FOV is the sum of the horizontal FOVs of each screen.

Fig. 5. Hardware arrangement of the multichannel visualization system.
A translator module and a local control module were installed on each slave PC. The master PC had a user interface module for the system operation and a global control module used to exchange instructions with the slave PCs. For the cluster of PCs, general desktop PCs with a 2.4 GHz Pentium processor CPU were used, and they were connected through a 100 Mbps Ethernet network.

Refreshing 20 to 30 frames per second is required for the user to naturally recognize moving pictures. The time required to refresh a single frame is less than 30 ms. According to the hardware performance test results, the time required to visualize a single frame on a PC with a 75 Hz refresh rate is less than 13 ms, and the network transmission time between PCs is less than 0.1 ms. Therefore it is found that the multichannel visualization system can maintain a speed above the required level.

4.2 Implementation of the multichannel visualization modules

4.2.1 User interface module

The user interface module has been implemented using the MFC of the Windows environment. With the aid of a keyboard, the user can send data to the user interface module that is necessary for adjusting the observer location, starting and stopping the simulation, and controlling the simulation. The user interface module then sends the data to the global control module.

4.2.2 Global control module
To enable the global control module to work with Delmia or other VMSs after minor modifications, Winsock of MS-Win32 API is used for the network of the global control module. Many VMSs, including Delmia, are blocked from outer networks during the simulation. As a result, the synchronization of screen switching is difficult due to the blocking mode of the VMS. This implies that the global control module needs to control the VMS through the slave PCs using another method while the VMS is executing the simulations. To overcome this constraint, the global control module split the entire simulation process into many short simulation sub-processes. The global control module executes the short sub-processes continuously and controls the slave PCs of the VR cluster between those sub-processes. The interweaving process time is too short for a user to perceive something unusual during the simulation. To accomplish this process, the global control module should handle two cases: one case where dynamic movement occurs and the other where it does not occur.
Fig. 6 shows the process by which the dynamic and static data is handled inside the global control module. When the VMS is processing a simulation, the global control module appears to simultaneously handle the user’s requirements for view control and the simulation data. The global control module divides the entire simulation process into many short sub-processes. The global control module lets all slave nodes process the simulation in short bursts and also updates the slave nodes with view control information between the short simulation sub-processes. This fast switching or interweaving process enables the user to feel that system is interactive during the simulation. If the status is static, the global
control module only checks for the user’s requirements for view control. The user’s requirements are then reflected in the slave PCs.

4.2.3 Translator module and local control module

The translator module receives the user’s requirements from the local control module and translates them to be suitable for the VMS installed inside the slave PC. Using the converted data, the local control module is connected to the VMS. The converted data comprises the user location data and auxiliary data for additional functions. The user location data contain the location and direction in six degrees of freedom. The data are updated and sent to the slave PCs every frame. The auxiliary data contain the information for file loading and resetting, and for starting, stopping, and pausing the simulation. This data are only sent to the slave PCs only when required.

The translator module is designed as a console mode for the efficient management of the system resources. The translator module depends on the VMS. If the VMS is replaced with another VMS, only the translator module needs to be modified or replaced. The translator module converts the received data into a form suitable for the VMS. The data received from the local control module include predefined commands and variables for commands. The translator module converts these command sets into commands of the VMS. Then, these commands are then sent to the VMS Delmia via Nettools, which is the network interface for Delmia, and these commands are executed in Delmia.

Some APIs of VMSs are not always sufficient for the VR environment. Some VMSs, including Delmia, have limited viewing functions. We can solve this problem by combining
several commands. For example, Delmia does not have an API command for the direct view control; but this can be solved indirectly by using the camera mounting API. The indirect method inserts a special primitive into Delmia and adds the camera node to this primitive. By controlling this primitive, the direct view can be controlled.

The local control module is responsible for setting the initial environmental variables for the slave PCs and connecting the global control module with the translator modules. Setting the initial environmental variables involves setting the global display functions that are related to the direction of the FOV; this type of setting ensures that the physical arrangement makes several screens appear as a single logical screen. Other initial environmental variables include the local display functions that are related to the horizontal FOV of each screen. Because these settings are different for each slave PC, each local control module is not the same.

4.3 Experiments

Fig. 7 shows the landing gear module of airplanes on the multichannel visualization system. The data of the landing gear module includes the landing gear box, the landing gear, the landing gear support and suspension system, as well as the corresponding kinematics data. This module simulates the cover of the landing gear box opening and the landing gear unfolding. This type of simulation requires an exact time sequence, so that ordinary graphics browsers cannot satisfy this criterion. Other than the multichannel system, the simulation of Delmia is identical with that of Delmia on a single PC.
Fig. 7. Experiment 1: Simulation of the landing gear module

Fig. 8. Experiment 2: Simulation of the spot-welding robot for BIW

Fig. 8 shows a section of an automotive assembly line. This simulation shows the moving station of an automotive BIW (body in white) through the assembly line with two spot-welding robots and jigs of the welding robots. Process layout designers can check the welding position of the robots through the virtual factory that provides variable size graphic
images of the assembly line, and they can analyze problems that may occur during the assembly process.

Fig. 9 is the tire attachment process in an automotive assembly line and it shows the usefulness of a life-size immersive VR. The worker in the form of an avatar selects and mounts a tire on a life-size vehicle that enters the assembly line on a conveyor belt. By analyzing the motion path of the worker and calibration by using the real size, the system allows ergonomic evaluations of the work environment.

![Fig. 9. Experiment 3: Simulation of the tire attachment.](image)

4.4 Comparison with existing systems
There are two existing ways to visualize the VMS data in an immersive VR system: (1) using a graphic server that can run the VMS and immersive VR software simultaneously, which requires high costs for VMS and VR software development costs, and (2) converting the geometric and kinematics data created from the VMS into a neutral or other format compatible with the legacy VR software. The converting processes cannot ensure data integrity and may not be cheap or fast. Furthermore, the complex structure can cause flaws.

Instead, in the study, a commercial VMS has been used as the viewer of the immersive VR system. This can reduce the cost and eliminate data integrity problems. Table 4 compares the proposed method with the SGI Onyx hardware, a high performance graphic server dedicated to Delmia, and traditional immersive VR software with data conversion. The SGI’s Onyx hardware architecture is designed to support a multichannel display and an immersive VR environment; even though it has excellent speed, its inflexible hardware configuration does not easily meet the diverse needs of diverse users.
Table 4  Comparison with existing systems

<table>
<thead>
<tr>
<th></th>
<th>Proposed method</th>
<th>SGI Onyx hardware dedicated to Delmia</th>
<th>VR viewers with data conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Low price desktop PC</td>
<td>Exclusive hardware</td>
<td>Low price desktop PC</td>
</tr>
<tr>
<td>Construction costs</td>
<td>Inexpensive</td>
<td>Expensive</td>
<td>Inexpensive</td>
</tr>
<tr>
<td>Data accuracy</td>
<td>No need for data conversion</td>
<td>No need for data conversion</td>
<td>Data loss due to data conversion</td>
</tr>
<tr>
<td>Execution speed</td>
<td>Reduced speed due to the network</td>
<td>High speed with the system bus</td>
<td>Reduced speed due to the network</td>
</tr>
<tr>
<td>System expansion</td>
<td>Expandable</td>
<td>Expensive to expand from the initial design specifications</td>
<td>Expandable</td>
</tr>
</tbody>
</table>
5. Conclusion

An immersive VR module has been proposed that can be interfaced with a virtual manufacturing system (VMS). The module uses the VMS as the graphic viewer of the immersive VR system with a cluster of PCs. Using the VMS as the graphic viewer for the VR system can avoid the data translation process that may generate flaws in the translated data. Removing the data translation process also reduces to reduce the preparation time and ensures the reliability of the data.

Because the vision is higher priority than other sensor of human, the proposed module has been realized mainly on the visualization part. Although the interactive devices in the proposed method are keyboard and mouse only, other immersive device such as 3D mouse and position tracker can be applied to the proposed method. And because Reality Center is good for cooperation in the manufacturing industry; the realized system can match the performance of the Reality Center. By changing some options we can make it CAVE system.

The proposed module can display the shapes and simulation processes of manufacturing equipments. This can provide the user immersive experience, and the life-size of the equipment brings reality to users such as designers or experienced workers.

This immersive VR module has been tested with a commercial VMS, Delmia. Users can see the same models as in a single PC and can obtain more information according to the number of screen. Using the Delmia VMS and the proposed immersive VR module, the experiments show that the immersive VR module can give a more realistic immersive
experience to the users without any changes in data when the module is in a static state or dynamic simulation.

References


