ISSN 2176-5480

22nd International Congress of Mechanical Engineering (COBEM 2013)
November 3-7, 2013, Ribeirão Preto, SP, Brazil
Copyright © 2013 by ABCM

METHODOLOGY TO INTEGRATE IMMERSIVE VIRTUAL ENVIRONMENTS WITH MACHINES ON-LINE REAL DATA

Heleno Murilo Campeão Vale
Claiton de Oliveira
São Carlos School of Engineering - University of São Paulo - Av. Trabalhador São-carlense, 400, Pq Arnold Schimidt São Carlos - SP/Brasil, CEP 13566-590
heleno@sc.usp.br
claiton@sc.usp.br

Arthur José Vieira Porto
São Carlos School of Engineering - University of São Paulo - Av. Trabalhador São-carlense, 400, Pq Arnold Schimidt São Carlos - SP/Brasil, CEP 13566-590
ajvporto@sc.usp.br

Abstract. *This paper presents a study which aims to integrate data from real machines in real time with a multiuser immersive virtual reality environment (CAVE) to allow improving the mechanical engineering simulations reliability. A case study was conducted based on the machining center ROMI D800 data capturing through the use of the open standard MTConnect. This case study proved to be possible to integrate a virtual ROMI D800 based on on-line real behaviors of a real machining center with pure virtual machines. They could communicate to each other inside a mechanical or production engineering simulation virtual environment improving the reliability of the simulations' results due to the trustworthy of the on-line real data being acquired. Furthermore by simulating on-line real machine behaviors, the virtual machines have trustworthy answers to users' commands as well as to other virtual machines data. With the help of stereoscopic glasses, the users can watch, navigate and interact with the operation of the virtual machines that instantly reproduce the movements of the real machines. At the end of execution or at any time of interest to users, data capture can be stopped and the real and virtual simulation data processed up to that point can be reverted or advanced to enable a specific study of a particular part of the process. With the use of an immersive virtual reality environment, mechanical or production engineering students, workers, managers or any other user, may take advantage of the benefits of complete immersion in the virtual environment as for example the navigation with six degrees of freedom around the machines and the shop-floor environment, zoom magnification to important machine details, adjustment the speed of execution of the previous processed data and so on. The simulation environment developed for the machining center ROMI D800 can be adapted to other types of environments and using external sensors it is possible to observe and analyze data normally invisible during the process such as temperature, force, stress and others. Finally, one of the main benefits of this research is to be a starting point to allow the simulation of many real expensive machines around the world together inside a unique virtual environment.*

Keywords: *Immersive Virtual Reality, MTConnect, CAVE, Machining Center*

1. INTRODUCTION

Recently, the manufacturing processes have been deeply influenced by globalization and increased competition in the market leads to a continuous improvement of manufacturing processes (Hernandez-Matias, *et al.*, 2008) requiring the minimization of costs and maximization of production. The technology of Virtual Reality (VR) has been widely applied, for example, to mechanical engineering manufacturing processes to aid different ways of simulating and analyzing the entire process and the possibilities of its improvement. Planning improvement, modeling, visualization and analysis of these systems are some examples of how the VR can contribute to the mechanical engineering processes (Lee, 2011). Other applications can be found in the work of Palma (2001), Chai (2010), Marinov (2004), Sung (2003), Wong (2010), among many others. The Multi-User Laboratory of Immersive Visualization (MLIV), located in the School of Engineering of São Carlos - EESC-USP owns a CAVE-like device with three projection screens, which was assembled, installed and configured mostly for researches in VR and manufacturing simulations. Undoubtedly, through immersion characteristics of the multi-projection system, it was possible improve the user's feeling of "being there" which contributes to the decision making process based on virtual reality visual simulations of real processes.

Besides the great benefits provided by an immersive virtual reality simulation, there are still many difficulties in the integration of on-line real data from manufacturing devices with information systems. There are several different machines which generate several different types and formats of data, delaying or even avoiding the proper integration of on-line data with a virtual reality simulation environment. The mechanical engineering and production engineering area does not standardized protocols and interfaces yet, as it is a very common subject in, for example, computer engineering area, via USB, Bluetooth and other standards.

ISSN 2176-5480

The remainder of this paper is organized as follows. In Section 2, the materials and methods used in this research are described. Section 3 presents the details of the developed methodology. Section 4 presents a case study and, finally, the next two sections address the conclusion and the future development, respectively.

2. MATERIALS AND METHODS

The development of this methodology used several materials and methods developed in the Multiuser Laboratory of Immersive Visualization (MLIV) of the School of Engineering of São Carlos (EESC), University of São Paulo (USP). Several exclusive tools and devices were produced to support immersive virtual reality applications like light wands devices, camera image acquisition methods, Bluetooth and infrared methods and devices etc. In the early years of development a CAVE-like device with three screens and six projectors was assembled, installed and configured. There are published papers of the same authors which address the MLIV like (Vale *et al.*, 2009), (Vale *et al.*, 2010) and (Oliveira *et al.*, 2009).

Besides the support to immersive visualization, a method of acquiring on-line real data from machining centers and using it inside a virtual reality environment with other virtual machines was developed. An open protocol called MTConnect was used as the basis to acquire several types of machines' information like axes position, information messages, door and alarm status, spindle speed etc.

2.1 Immersive Virtual Reality and CAVE-like devices

Advances in scientific simulation rapidly led to the generation of multi-terabyte sized datasets of unprecedented spatial and temporal resolution. The interactive visualization resources these datasets demand overwhelm the display capabilities of the typical desktop system, both in terms of raw graphics processing power and display resolution. In order to address this issue, researchers have explored the possibility of combining multiple standard resolution projectors into a single system to increase total display resolution. Such multi-projector display systems are currently under development at a number of research labs. An example of these systems is the CAVE (CAVE Automatic Virtual Environment) virtual reality/scientific visualization system (Cruz-Neira, 1995). Therefore, CAVE-like devices are virtual reality interfaces whose abstract design consists of a room whose walls, celling and floor surround a viewer with projected images improving the level of immersion of the virtual reality users. This topic describes the MLIV CAVE-like device, its hardware tools and software methods.

The MLIV is located in the building of the Teaching and Research Automation and Simulation Center of the Department of Mechanical Engineering, in the School of Engineering of São Carlos. The MLIV physical structure comprises 10 meters long and 5 meters high building to accommodate the multi projection device. The room walls and ground are composed of materials suitable to aid absorption of unnecessary sound and light, reducing the level of reverberation and light noise during the immersive simulations. Two screens are perpendicular to the ground and forming an angle of 90 degrees between them. The third screen corresponds to the projection screen of the floor. The vertical screens materials are white and flexible and positioned in the shape of "L". The projection screen of the floor is also white but composed of wood to support the weight of the users. For each face there is a pair of projectors responsible for the projection of stereoscopic images. The projection of the images on the side walls is carried out by the back face of the screen, crossing it to be visualized by the user in its front face, while the ground receives a projection from the top with the aid of a tilted mirror to adjust the distance between the projector and the floor screen. The front and side projection screens have dimensions: 3.2m X 2.4m for front screen and 2.4m X 2.4m for the side screen. The rigid floor projection screen has dimensions of 3.2m X 2.4m. The structure that supports the projection screens is fixed and made of hardwood. The system consists yet of six projectors model Christie DS+60/3000 DW30/Matrix high light intensity and high resolution. Each projector is composed of a lighting system with two 300W lamps that offer high levels of brightness and contrast. The native resolution of each device is 1400 X 1050 pixels. For receiving the data sent by the outputs of the video cards, the connections between computers and projectors are established through DVI (Digital Visual Interface). The distance between the projectors and projection screens, as well as its position relative to the screen must be adjusted to obtain the best possible alignment of images divided in three screens. The projectors responsible for the floor screen are held in a metal frame attached to the top of the wooden structure which supports the three screens. To adjust the distance between the projectors and the ground, your lenses are facing a mirror set at an angle of approximately 45 degrees which reflects the images to the floor screen. Projectors responsible for front screen and side screen are kept in metal racks and positioned with its lens pointed directly at the vertical projection screens.

For each projection screen, two projectors with polarizing filters display a stereoscopic pair of images, which have an image with predominantly green color and the other with predominantly purple color, overlapping and blurred. Only through the use of passive stereoscopic glasses with polarizing filters of the same color as the projected images the viewer is able to see a three-dimensional image.

The development and implementation of applications for the multi projection device depends on a broad and ever evolving technology, which includes the communication infrastructure and the synchronized presentation of images by the three computers responsible for each one of the three screens. Each computer these three computers is equipped with an AMD Opteron 850 2.41 GHz, 4 GB RAM and video card Nvidia Quadro FX4500 with 512 MB of internal memory. One of the computers in the cluster also has a sound card PCI Creative SB X-Fi Titanium with 7.1 digital audio outputs. The MLIV has also a Home Theater Samsumg HT-C460 with 850W RMS which provides 5.1 surround sound system with five speakers and one subwoofer. The connection between the PC and the home theater is performed by an optical cable connected to the sound card output and the digital optical input of the home theater.

This sound system allows the use of standard audio (stereo or mono) or 3D audio. In addition, there are algorithms which allow the use of 5.1 surround sound controlled by motion trackers (head trackers), creating a full sound simulation (Soares, 2010). Some extra care should be taken to achieve better sound quality, for example, the use of carpet on the floor and lining the walls to eliminate inadequate reverberation. Too much reverb can confuse users regarding the correct positioning of the triggered sounds. All speakers are suspended, except for the subwoofer as it is recommended to keep it in the ground for better use of low-pitched sounds. The side front speakers are kept out of the projection area, positioned close to the joints of the front and side screens. The center front speaker is positioned below the projection area.

As a way of interacting with the virtual environment designed it is necessary to use some devices for data entry, which can be read and interpreted as user commands to perform some action in the 3D environment. Besides the most common devices such as mouse, keyboard and joystick, the researchers of the MLIV developed other exclusive devices and methods of interaction that are described in this section too. After several tests with devices depending on the Bluetooth connection, and an extensive unsuccessfully search for Bluetooth devices with higher range, it was decided to abandon the use of such technology and test the use of capture infrared light for video interaction. For this, we acquired low cost webcams and two high intensity infrared LEDs (Light-Emitting Diode). A simple webcam was dismantled and a light filter was attached to its lens. The light filter comprised photo film sheets in order to allow the passage of only infrared light, blocking the natural light. It was soon realized that infrared light was not good enough to enable correct detection of the movement of LEDs. Then, as an idea derived from the use of infrared LEDs and webcams, it was chose to test the capture of light from laser pointers and soon it was developed an exclusive MIVL 3 color light wand which interacts with the virtual scene through a simple webcam located in the intersection of the three screens at the floor screen. The webcam is positioned pointing to the users hands so it can capture the light of the wand and transform it in user's movements inside the virtual world or virtual objects' drag and drop operations. The camera position can be seen in Figure 1.

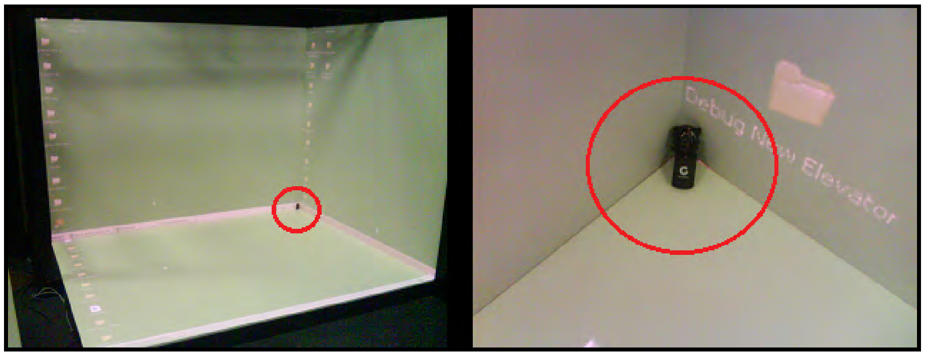


Figure 1. Webcam in the intersection of the three screens

The programming platform used in MIVL is the "Visual C++ 2005 Express Edition" the VRJuggler 2.2.1 framework. For the import of 3D models and the creation of the scene graph the library OpenSG 1.6.0 is used. For the development of interaction solutions for motion detection or camera data capture the library OpenCV 2.0 is used and, finally, the manipulation of sound is done through the use of the library OpenAL 2.0. VR Juggler is a research project of the Virtual Reality Applications Center at Iowa State University, which includes a collection of technologies, which provide the necessary tools for the development of VR applications. It is basically a platform for the development and implementation of applications based on VR, it is composed by a set of C++ libraries. These libraries provide a set of interrelated classes that make programming for VR easier. The platform allows the application to be independent of the hardware architecture and operating system through the use of general configuration files. Also it makes possible the configuration of dedicated hardware by configuring their attributes without the need to modify the application code. This provides a unified operating environment ensuring application portability between different resources. Every action which occurs in a virtual environment controlled by VR Juggler is bound to its kernel component. The kernel acts as an interface between the application and the platform, providing hardware abstraction.

OpenSG is a scene graph manager for the development of high performance graphics applications. A scene graph is an arrangement of 3D objects in a tree structure which specifies the contents of a virtual universe and indicates how it will be rendered. In it, you can set sound, light, orientation, geometry, location and visual appearance of objects.

Describe a scene using a scene graph is a simpler task than building the scene from the command line to specify graphical primitives such as OpenGL (Open Graphics Library). The scene graph allows storing data in a hierarchical manner where parent nodes items with lower branches affect their child nodes items with its ramifications, thus, the root node groups all objects of the tree. The OpenSG provides an object-oriented framework where, in a lower level of abstraction, the OpenGL library is working. OpenSG is compatible with applications which support OpenGL. Its function is to free the developer from implementing and optimizing low level graphics calls, helping the fast development of graphics applications and saving the developer to program complex routines of computer graphics and 3D rendering.

The Open Source Computer Vision - OpenCV - is a programming library, open source, initially developed by Intel Corporation. OpenCV has a variety of tools for image interpretation, from simple operations such as a noise filter, even complex operations such as motion analysis, pattern recognition and 3D reconstruction. The OpenCV library was designed with the goal of making computer vision accessible to users and developers in areas such as human-computer interaction and real-time robotics. An OpenCV program invokes an automatic DLL (Dynamic Linked Library) which detects the processor type and loads the appropriate DLL. The library IPL (Image Processing Library) is provided in the OpenCV package because it depends partly on the IPL. The OpenCV library is divided into groups of functions:

- Image processing;
- Structural analysis;
- Analysis of motion and object tracking;
- Pattern Recognition;
- Camera Calibration and
- 3D reconstruction.

OpenAL (Open Audio Library) is a software interface to audio hardware. In general, it is a way to generate audio for three-dimensional simulations. This interface consists of several functions which affect the CPU and sound hardware allowing developers to create multichannel output for three-dimensional sound around the user. The OpenAL library was created in the late 90s, as a complement to the OpenGL library following the same coding style.

2.2 MTConnect protocol

There are still many difficulties in the integration of data from manufacturing devices with information systems. Several machines generate several different types and formats of data delaying or even preventing the proper integration of data into a simulation or other kind of research method. The mechanical engineering and the production engineering area does not standardized protocols and interfaces yet, as it is very common, for example, in computer engineering standards like USB, Bluetooth etc. The standard offered by MTConnect protocol provides a common means for communication between shop floor devices and can be deployed in order to simplify the collection and evaluation of these data. The protocol is based on XML (eXtensible Markup Language), a generic markup language, which has as main purpose to facilitate the sharing of information through the Internet. XML uses a hierarchical method for organizing data. The main components of the MTConnect protocol are the adapter and the agent.

The adapter is the element responsible for controlling all communication with the device and the Agent. It is a TCP/IP server and while no agent is connected, the adapter works on standby without communicating with the device. When an agent is connected, the adapter attempts to connect to the device. In case of error the adapter considers that the device is turned off and sends this information to the Agent. On the other hand, if the connection is successfully completed, the adapter starts collecting data periodically, sending new data to the agent and avoiding overhead in network communication.

The Agent is the application that implements the interface MTConnect controlling the storage of data collected by the adapter and implementing an HTTP server for handling requests from client applications. An agent can represent more than one device depending on the availability of the machines in the plant. All received data is saved in a buffer of configurable size and a sequence number is created to identify each received data. Thus, it is possible, through the HTTP interface to request the data history of certain components or the entire device. If the buffer capacity is reached, the oldest data is lost and so the application can save new data. The end user can also modify the Agent to save the values in a database if necessary therefore all data will be available at any time. Figure 2 shows the relation among the application software, the machining center and the MTConnect adapter and agent.

ISSN 2176-5480

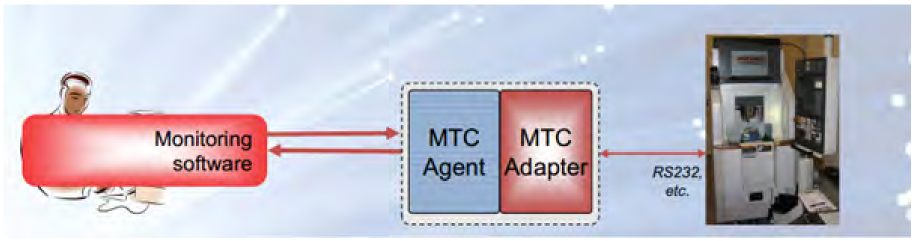


Figure 2. Application software, machining center and MTConnect adapter and agent communication

The concepts involved in agent services are:

- Header: information regarding the protocol;
- Components: parts of a device;
- Data Items: describe the data available on a device;
- Samples: a measure of an item in constant change at a given time;
- Events: state changes and alerts;
- Streams: set of sample or event data;
- Alarms: indicates abnormal behavior.

The initialization structure of a generic agent is shown in figure 3.

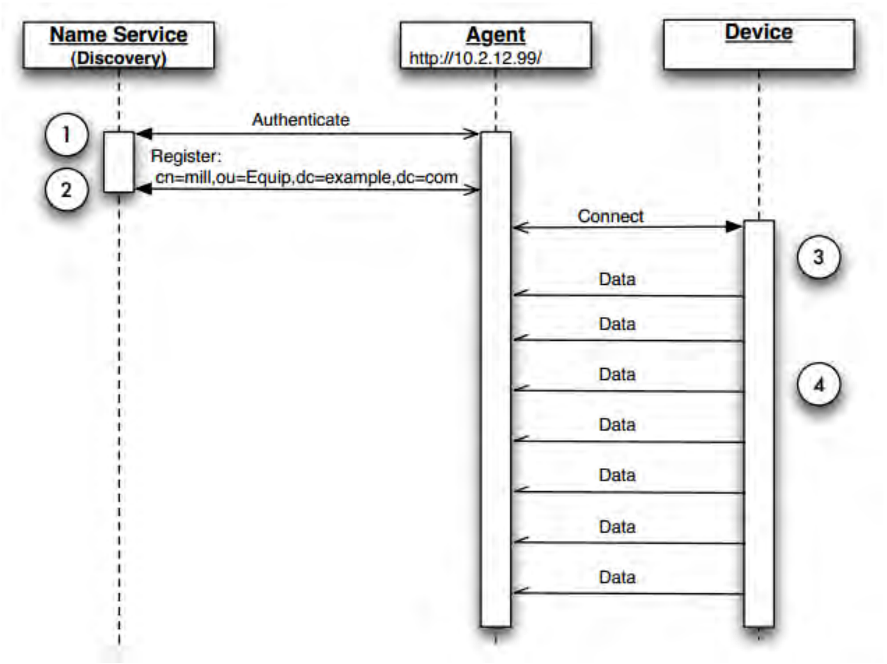


Figure 3. Initialization structure of an MTConnect agent

First the agent connects and authenticates with the Discovery service (Name Service). Discovery is a service which allows an application to find agents for a particular device. Then a generic agent registers its URL so it can be located. The agent connects to the device using the proper APIs and the device sends data to the agent or the agent requests data to the device through the use of an adapter not shown in previous figure.

MTConnect agents receive three different kinds of requests:

- Probe - returns the components and items of a given device;
- Current - returns a snapshot of the most recent values for each item;
- Sample - returns data samples and sequential events.

Generally, the first request made is a probe request in order to determine all items of each device. Then a current request is done to get the most current values of each data item and especially to get the next sequence number so that the application can get sample requests from its sequence number on. Generally the instance identification value is also stored in order to restore the application in case of failure of the agent.

3. METHODOLOGY

Machine tools produce several different kinds of data during the execution of their operations. Therefore, to acquire data from these machines is not a trivial task, especially because most of them were not prepared for supplying data to external applications. So, the first step of this research was to define the data types used based on the MTConnect protocol standard types. Furthermore, the data types were chosen based on their importance for immersive virtual reality simulations, for example, starting and stopping the execution and components behavior visual data. The immersion and realism of objects, movement, interaction and animation is an important part of VR applications and simulations. Therefore data with large visual distinction, as the current state of the door (open, closed or unlocked) and alert warnings, among others, were judged as essential, both due to its importance to the operation of the machines and their native visual aspect. The main data selected for the development of this methodology and supported by MTConnect protocol is listed and described below:

- Door_State: current status of the door of the machine tool. This may vary among only three values: open, unlocked and closed;
- Emergency_Stop: current status of the emergency stop actuator. This can vary between only two values: armed and triggered;
- Execution: operating status of the controller which can vary between five values: ready, active, interrupted, feed_hold and stopped;
- Part_Count: an integer which represents the number of parts produced to date
    - All: count all pieces produced;
    - Good: count only parts produced correctly;
    - Bad: count only parts produced with errors.
- Part_Id: identification number of the actual part in production;
- Position: position of a movable component of a device;
- Rotary_Velocity: rotation speed of a component.

In this research it was defined that devices and conveyors would be linked together according to the order of its addition to the virtual scene. The order of entry is defined in the parameters creation module, which allows the user to position each element of the scene and define their relationship. For example, the device identified with the number "1" relates only to the transporter numbered with the number "2". The device numbered "3" is related to the conveyors numbered with the values "2" and "4" and so forth. Based on the relationship defined by the user, the elements of the scene are queued in a string of run parameters of the main application and at the start of execution the devices are instantiated and neighboring devices are related to each other through their classes attributes. An object receives the object data stored in its "previous object" variable and sends data to the object stored in its "next object" variable. Update methods, specific to each device, constantly check the value of Part_Count (All) in an XML file, created by the MTConnect agent, and increase the value of its integer internal variables. The variable Part_Id is directly related to Part_Count_Good and Part_Count_Bad in order to inform the user which parts were recorded as Bad (defective part) or Good (part successfully produced). Each device has a thread for reading data from its XML file available through private network or Internet. The position data is read continuously by the same thread that handles other kind of data. Translating and rotating data are stored in two vectors of information which act as a buffer between the download of XML files and the animation and translation functions.

After defining the data types which would be used in the methodology, a hierarchical tree structure was set to be adopted by the classes and subclasses of the methodology. This step aimed to enhance the extensibility of devices and data types used to provide flexibility and allow its adaptation to other application areas. The main classes and hierarchy adopted for devices and components are shown below:

- Shop-Floor singleton class: the main class of the application;
- Device abstract class
    - DeviceMTC subclass: a virtual device supplied by MTConnect online data from real machines;
    - DeviceVirtual subclass: a pure virtual device;
    - DeviceSimulator subclass: a device supplied by online simulators information;
- Component abstract class
    - ComponentDoor subclass: includes a door component to a device;
    - ComponentAxes subclass;
        - ComponentAxesLinear leaf class: includes a linear movable component to a device;
        - ComponentAxesRotary leaf class: includes a rotary movable component to a device;

ISSN 2176-5480

- ComponentFactory class: used to create all components;
- Conveyor abstract class
  - ConveyorRobot subclass: a robot arm used to carry objects from one machine to another;
  - ConveyorWorker subclass: a human worker used to carry objects;
  - ConveyorStart subclass: internal hidden class to mark and supply the first device;
  - ConveyorEnd subclass: internal hidden class to mark and receive data from the last device.

An abstract class cannot be instantiated. Therefore the classes Device, Conveyor and Component work only as abstractions or generalizations of their respective concrete subclasses. These abstractions are used to make the software design more flexible, because with the use of references to abstract classes, it is possible to treat any type of concrete classes in the same manner inside the application. Thus, it is possible to extend the number of types of concrete classes without the need to change the main part of the source code. ShopFloor class is the main class of this methodology. It has several responsibilities including the instantiation of other classes and startup the scene graph (OpenSG library – a tree which stores objects in the virtual scene). This class is not abstract, so it can be instantiated. However, as it is a major class of control which should be instantiated only once, it was decided to implement it as a singleton class. Thus, in a future extension of this work, where several applications should access the same virtual environment, all of them would share the ShopFloor class properly. Some specific details of sharing data through threads (parallel processing) will not be treated in this paper. The ComponentFactory class is a concrete class which simplifies the insertion of new components to the whole system, as well as their creation. This class allows a user to create components without the need of knowing their concrete subclasses. The user requests the instantiation of a component and the ComponentFactory decides which concrete class to instantiate, based on the parameters passed by the user.

For modeling 3D objects it was used a specific software able to import and export VRML files with extension "WRL". Any modeling software which exports VRML files through native functions or plug-ins can be used for the production of objects in this approach. Only the lights of the environments were created directly on the C++ code, the rest of the scene was imported with the help of VRML files. A library of basic virtual shop-floor objects was created with the following components:

- Lathe;
- CNC Lathe;
- Drill;
- Milling;
- Conveyor;
- Operator;
- Robotic Arm;
- Scenario for a generic shop-floor.

This library can be extended by new programmers or users of the proposed methodology just following the standards described in this paper. It was decided that all static parts of virtual objects would be kept in a single group named "STATIC". It was also stipulated that each movable member of each object would be grouped under the name of its movable axis of translation or rotation, for example, “X”, “Y” or “Z” for linear movements and “C” for rotary movements. Also it is possible to name groups of combined movements like "XY" for example. When importing VRML files from the modeling software to the main application it is important to observe the need to interchange Z and Y axes like showed in Figure 4.

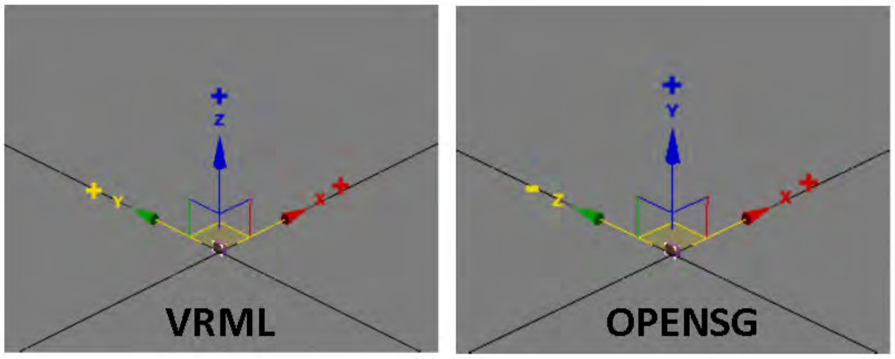
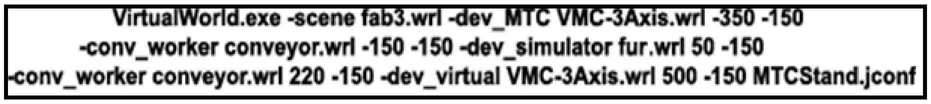


Figure 4. Case study for ROMI D800 with Fanuc CNC

The center of the created objects must be kept exactly in the center of the axis of coordinates in the modeling software, i.e. at the "0x0x0" or intersection of “X”, “Y” and “Z”. This form of standardization is important for the main application, since the rotation operation is applied based on the center axis of the object. Objects which have their center located away from the center of the coordinate axis, when exported to VRML format behave as if there were an invisible bounding box surrounding it and forcing it to stay at the intersection of the axes. Besides ensuring the correct positioning and orientation of the final group, it is important to ensure that the size of the created objects is consistent with the stipulated proportion to the main application. The methodology stipulates that 200 generic units correspond to 2 meters.

To start the application it was created a separated module which allows the user to position and link each object. This module internally creates a command line with tags and parameters, objects and positions like showed in Figure 5.



```
VirtualWorld.exe -scene fab3.wrl -dev_MTC VMC-3Axis.wrl -350 -150
-conv_worker conveyor.wrl -150 -150 -dev_simulator fur.wrl 50 -150
-conv_worker conveyor.wrl 220 -150 -dev_virtual VMC-3Axis.wrl 500 -150 MTCStand.jconf
```

Figure 5. Example of an internal command line to start the application

The potential values for tags are: “Scene” for the application scenario, “–dev_mtc” for instantiate a Device_MTC (MTConnect) type which the name without its extension will be used also for the MTConnect agent requisitions, “-dev_virtual” to instantiate a pure virtual device, “-dev_simulator” to use any online data simulator, including the MTConnect institute free simulator and “–conv_worker” and “-conv_robot” to instantiate conveyors.

To develop the MTConnect adapter, first it was defined what kind of data would be acquired from the available machine, in this case a ROMI D800 with Fanuc CNC. Next, the developer needs to find the specific API (Application Programming Interface) for each device. Then he can use an available adapter framework together with the APIs and create his own adapter, specific for his needs.

The first communication made between the application and the MTConnect agent is a current request which check the capabilities of the device before start requesting samples of data. This check returns an XML file “Streams” which comprises all items and components of a device. However, the most important attribute to be observed by the application is the “nextSequence” attribute which is used as input data for the first request of data samples. After obtaining the “nextSequence” number through a current request, the device object instantiated begins performing sample requests to acquire position or controller data. The “nextSequence” attribute is updated every XML file received. After acquiring data through XML files, they are initially stored in buffers for use almost immediately, or online, inside the virtual scene. The buffer consists of two string vectors, which are used together and serially to avoid concurrent access and enable continuous reading of data. Synchronization is arranged through semaphores and threads.

Behavior data are displayed in the virtual scene by the functions named “translateObject” and “rotateObject” to be called in the main function named “preFrame” which holds the main looping of the application. The functions “translateObject” and “rotateObject” require as input data increments on all axes “X”, “Y” and “Z”. These values are obtained directly from the buffer vector for each device. Each pass through the preFrame looping updates the values of all devices instantiated. The buffer vectors store four types of data simultaneously: data, axis, component number and sequence number of the data. When updating the values of the buffers in the virtual scene, inside the “preFrame” looping, all these data are evaluated so the application can determine which component should have its data updated and which of their motion axes should be updated. The “preFrame” performs the reading of data from each device sequentially based on the sequence number stored along with each data point. The “readXML” function is responsible for keeping the buffer vectors ordered before reaching “preFrame”. Besides the use of semaphores to control access, download and read XML files, the synchronizing of buffer vectors reading and writing uses also a boolean variable called “firstVector” which informs the functions “readXML” and “preFrame” about which vector should be read or written at that moment of execution.

4. CASE STUDY

A case study was conducted based on the machining center ROMI Discovery 800 with CNC GE-FANUC Series 21i. It was developed a specific MTConnect adapter module for CNC GE-FANUC based on the structure provided by the MTConnect institute and the previous work of Ferraz (2012). The adapter acquires data and converts them into a format understood by the agent, through the use of the library FOCAS2 specific for Fanuc CNCs. The complete set of elements for this case study consists of the machining center and its support computer, the computer data server (agent) and the master node of the computer cluster (CAVE) as illustrated in the Figure 6.

ISSN 2176-5480

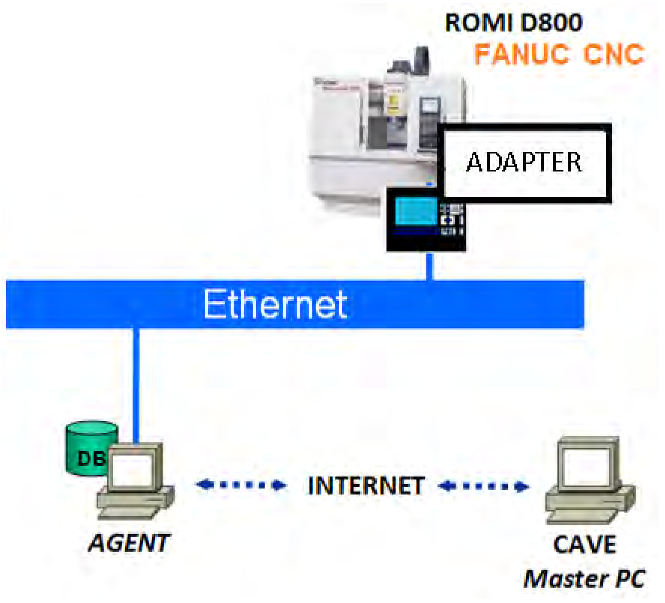
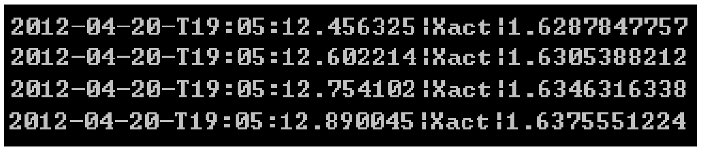


Figure 6. Elements of the case study for ROMI D800 with Fanuc CNC

The machining center ROMI D800 is not fully compatible with the MTConnect protocol, thus there is the need to install an adapter in the support computer of the machine. This adapter can also be embedded directly into the HMI PC (Human Machine Interface) of the machine. Data packages are sent from the adapter to the agent, located on a server. Figure 7 shows a data package acquired by an MTConnect adapter.

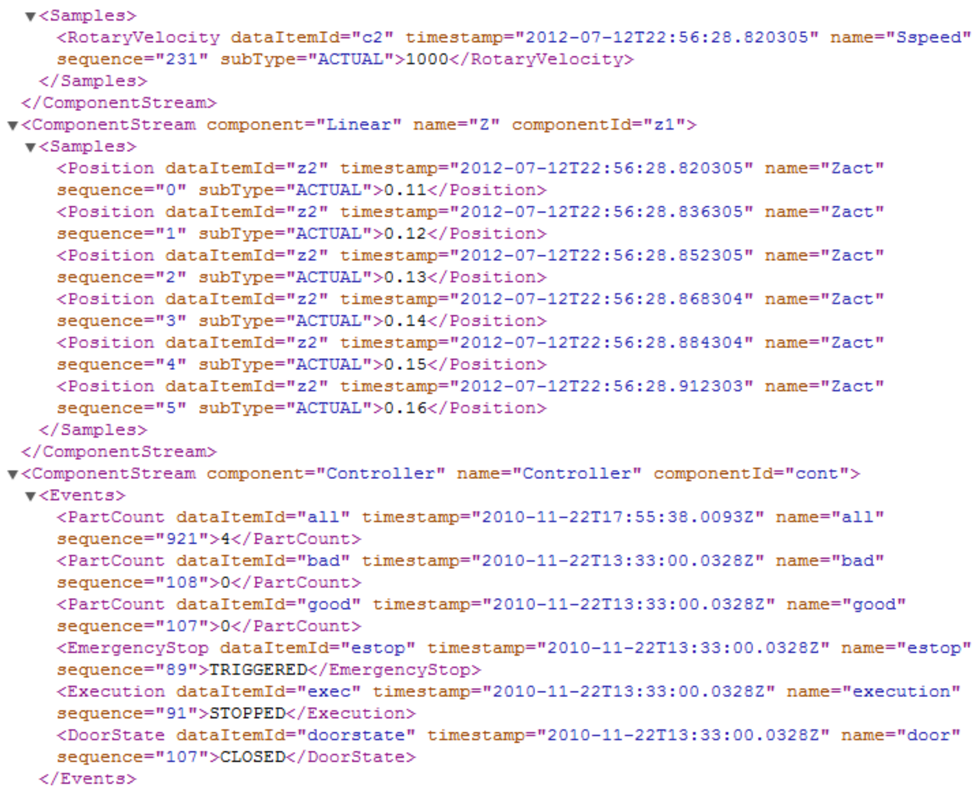


```
2012-04-20-T19:05:12.456325|Xact|1.6287847757
2012-04-20-T19:05:12.602214|Xact|1.6305388212
2012-04-20-T19:05:12.754102|Xact|1.6346316338
2012-04-20-T19:05:12.890045|Xact|1.6375551224
```

Figure 7. Data package acquired by an adapter

The function of the agent is to receive the adapter data and turn them to default MTConnect standard, so they can be sent in XML files to the applications as they requests data from the main computer of the CAVE (master node). Figure 8 shows an example of an agent XML file which should be sent to the virtual environment with position and controller information.

ISSN 2176-5480



```
  ▼<Samples>
     <RotaryVelocity dataItemId="c2" timestamp="2012-07-12T22:56:28.820305" name="Sspeed"
     sequence="231" subType="ACTUAL">1000</RotaryVelocity>
   </Samples>
 </ComponentStream>
▼<ComponentStream component="Linear" name="Z" componentId="z1">
  ▼<Samples>
     <Position dataItemId="z2" timestamp="2012-07-12T22:56:28.820305" name="Zact"
     sequence="0" subType="ACTUAL">0.11</Position>
     <Position dataItemId="z2" timestamp="2012-07-12T22:56:28.836305" name="Zact"
     sequence="1" subType="ACTUAL">0.12</Position>
     <Position dataItemId="z2" timestamp="2012-07-12T22:56:28.852305" name="Zact"
     sequence="2" subType="ACTUAL">0.13</Position>
     <Position dataItemId="z2" timestamp="2012-07-12T22:56:28.868304" name="Zact"
     sequence="3" subType="ACTUAL">0.14</Position>
     <Position dataItemId="z2" timestamp="2012-07-12T22:56:28.884304" name="Zact"
     sequence="4" subType="ACTUAL">0.15</Position>
     <Position dataItemId="z2" timestamp="2012-07-12T22:56:28.912303" name="Zact"
     sequence="5" subType="ACTUAL">0.16</Position>
   </Samples>
 </ComponentStream>
▼<ComponentStream component="Controller" name="Controller" componentId="cont">
  ▼<Events>
     <PartCount dataItemId="all" timestamp="2010-11-22T17:55:38.0093Z" name="all"
     sequence="921">4</PartCount>
     <PartCount dataItemId="bad" timestamp="2010-11-22T13:33:00.0328Z" name="bad"
     sequence="108">0</PartCount>
     <PartCount dataItemId="good" timestamp="2010-11-22T13:33:00.0328Z" name="good"
     sequence="107">0</PartCount>
     <EmergencyStop dataItemId="estop" timestamp="2010-11-22T13:33:00.0328Z" name="estop"
     sequence="89">TRIGGERED</EmergencyStop>
     <Execution dataItemId="exec" timestamp="2010-11-22T13:33:00.0328Z" name="execution"
     sequence="91">STOPPED</Execution>
     <DoorState dataItemId="doorstate" timestamp="2010-11-22T13:33:00.0328Z" name="door"
     sequence="107">CLOSED</DoorState>
   </Events>
```

Figure 8. Agent XML data

For this case study an open source agent supplied by the MTConnect Institute was used. Figure 9 shows a virtual machine working with off-line data and another virtual machine working with online real data captured from a real ROMI D800 machine located in another building of the same university.

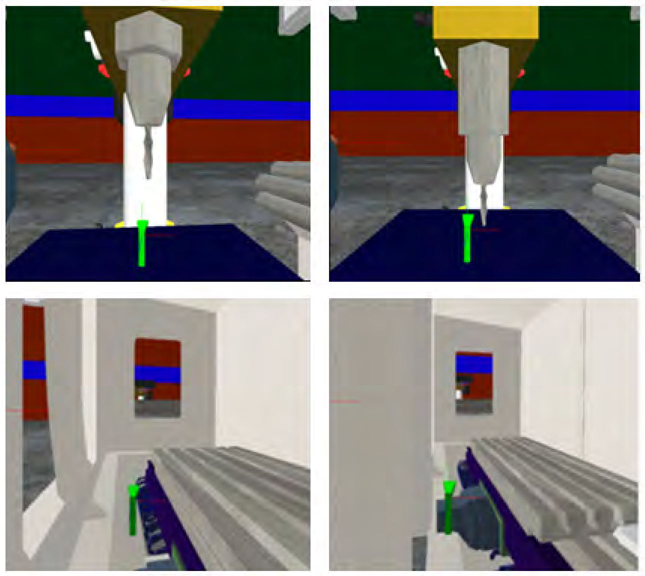


Figure 9. Virtual machines working with pure virtual data and online captured real data

5. CONCLUSION

This research presented a methodology to allow the use of online real data inside immersive virtual reality environments, such as CAVE-like devices. The main focus of this research was to take the first step to improve reliability on mechanical engineering simulations and to allow a new kind of simulation where many remote real machines can collaborate together inside a single virtual environment at the same time. The MTConnect protocol and its adapter and agent frameworks were used to acquire online manufacturing data from machining centers and communicate it to virtual environments. Standardization of data acquisition and communication methods as well as objects creation using a standard and hierarchical structure provides extensibility and flexibility to a methodology, simplifying future work developments and the use of it in other areas. Finally, this methodology was tested and validated through case studies and implemented based on open source software libraries and low-cost devices.

6. FUTURE DEVELOPMENT

This methodology is only a first step to achieve more reliable mechanical engineering or production engineering simulations. There is a lot to be done yet. Immersive virtual reality improves the common virtual reality results based on the level of immersion provided to users. But, in this research, besides using an immersive device, the main focus was to create a viable methodology to use online real data within immersive virtual environments. There are big efforts to be done to improve the realism of the immersive scenes. Therefore, to achieve practical mechanical engineering simulations and great decision-making results with this methodology, the improvement of quality of immersive virtual scenes is necessary and proposed as a future work. Also, besides the quality of the virtual objects and processing time, we propose a special study of 3d sounding systems for mechanical engineering shop-floor environments.

The MTConnect protocol can be better explored in a future work by developing new classes of objects which inherit the structure described in this paper. Hence, the methodology described here was created with special computer programming techniques which allow its extension in a simplified and viable manner.

Finally, this research can even be used as basis for a study about production lines teleoperation through immersive virtual environments although not all machine tools work with opened CNCs, which could allow the input of remote data from the CAVE to the machines.

7. ACKNOWLEDGEMENTS

The authors thank FAPESP and CNPQ for financial support.

8. REFERENCES

Chai Y., Zhou Y., Wang K., 2010, "E-Blocks of Computer Disk Drives based on Virtual Manufacturing". Third International Workshop on Advanced Computational Intelligence.

Cruz-Neira, C., 1995, "Virtual Reality Based on Multiple Projection Screens: The Cave and its Applications to Computational Science and Engineering", *PhD. Thesis - University of Illinois*, Chicago, IL, USA.

Ferraz F. Jr., Silva E. J., Rondon L. C., Giuffrida A. B., Abujamra R. C., Valente M. O., 2012, "Supervisão remota da usinagem com o padrão MTConnect". *Máquinas e Metais*, Aranda, n.556, 2012.

Hernandez-Matias, J.C., Vizan, A., Perez-Garcia, J. e Rios, J., 2008, "An integrated modeling framework to support manufacturing system diagnosis for continuous improvement", *Robotics and Computer-integrated manufacturing*, 24, 2, 187-199.

Lee H., Banerjee A. A, 2011 "Self-configurable large-scale virtual manufacturing environment for collaborative designers", *Manufacturing And Construction Virtual Reality* 15:21–40.

Marinov V.P., Seetharamu S., 2004, "Virtual machining operation: a concept and an example", *The International Society for Optical Engineering.*

Oliveira C., Sena D. C., Vale H. M. C., Carvalho H. J. R., Porto, A., J., V., 2009, "Low-cost gesture detection as a form of interaction in a virtual reality multi-projection system", *Innovative Developments in Design and Manufacturing: Advanced Research in Virtual and Rapid Prototyping*". London: Taylor & Francis Ltd, v., p. 615-618.

Palma J. G., 2001, "Metamodelo para a Modelagem e Simulação de Sistemas a Eventos discretos, baseado em Redes de Petri e Realidade Virtual. Uma Aplicação em Sistema de Manufatura". *Doctorate Thesis.* Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

Soares L. P. et al, 2010, "Designing a Highly Immersive Interactive Environment: The Virtual Mine", *Computer Graphics Forum*, Volume 29, number 6 pp. 1756–1769.

Sung W.T., Ou S.C., 2003, "Using Virtual Reality technologies for manufacturing applications", *International Journal of Computer Applications in Technology* (IJCAT), Vol. 17, No. 4.

Vale H. M. C., Oliveira C., Sena D. C., Porto A., J., V., 2010, "Construção de Ambientes de Multiprojeção Imersivos para Tratamento de Fobias no Ambiente de Trabalho". *Innovare*, v. 9, p. 84-100, 2010.

ISSN 2176-5480

Vale H. M. C., Oliveira C., Sena D. C., Porto A., J., V., 2009, "Construction of immersive multi-projection environments for treatment of phobia of heights", *Innovative Developments in Design and Manufacturing: Advanced Research in Virtual and Rapid Prototyping*. London: Taylor & Francis Ltd, v., p. 619-623.

Wong S.F., Yang Z.X., Cao N.; Ho W.I., 2010, "Applied RFID and Virtual Reality Technology in professional training system for manufacturing", *Proceedings of the 2010 IEEE*, IEEM, 2010.

9. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.