Virtual Reality Simulator For Robotics Learning

Raúl Crespo, René García, Samuel Quiroz
Mechatronics Department
Tecnológico de Monterrey, Campus Ciudad de México
rcrespo@itesm.mx, samuel.quiroz.lima@gmail.com, albrto.cz@gmail.com

***Abstract*— The overall purpose of this project is to test the impact and potential benefits of Virtual Reality technology by creating a virtual simulator for the operation of the Mitsubishi Movemaster RV-M1 Robot.**

The project aims to be a proposal for education innovation by increasing the student's interactivity with the robot's features inside a virtual environment, developed using the game engine Unity and the Oculus Rift headset for the virtual visualization.

By using this virtual simulator, on site operation time is decreased, mainly by allowing off-line programming of the equipment. It also allows universities without or with limited industrial machinery to provide their students a way to learn and practice on industrial automation and robotics simulation topics without inconvenience.

The virtual simulator is designed to decrease the student's learning curve by displaying a complete virtual environment where the tridimensional model of the robotic arm can be visualized and programmed according to the real model's parameters and specifications.

Keywords—Virtual Reality; Education Innovation; Robotics Trainning; Oculus Rift; Unity

I. INTRODUCTION

Interactions between students and technology are clearly in almost all daily activities. At school, students are most of their time surrounded by on-line devices such as cellphones, computers and tablets. Those devices have become new tools for educational activities, since they are often used to watch a tutorial video or to watch real time streaming classes from different parts of the world. The current challenge depends on implementing new technology to achieve new educational experiences. Time to achieve this challenge can be substantially reduced if educational contents and tools are designed and develop using the same tools used to program and develop media and entertainment products such as videogames or 3D movies technology.

Moreover, students are more active in their learning process. Often they need to watch and do things, always surrounded by technology. Meanwhile the teacher needs to capture the students' attention to actively involve them in learning. Teacher's main goal should be to create learning environments in education processes that really transform the student's life and finally into a competent professional.

The development of VR applications is a current trending phenomenon and according to Pantelidis [1] its use in education can be considered as one of the natural evolutions of computer-assisted instruction (CAI) or computer-based training (CBT). Pantelidis [1] asserts that at every level of education, VR has the potential to make a difference mainly by motivating, encouraging and exciting students. Virtual reality is not appropriate for every instructional objective. Pantelidis [2] makes the following suggestions on when to use and when not to use virtual reality in education. Use or consider using virtual reality when:

- A simulation could be used.
- Teaching or training using the real thing is dangerous, impossible, inconvenient, or difficult.
- A model of an environment will teach or train as well as the real thing.

The learning process is enhanced by using immersive virtual reality technology in educational applications by increasing the students' focus on the theoretical and practical features of their assignments by promoting independent learning and also collaborative work and concurrent use of limited resources such as real machines and robots becomes feasible.

Therefore, this paper presents the potential benefits of implementing immersive virtual reality technology in an educational application. The particular scope of this publication is to integrate an immersive virtual reality technology such as the Oculus Rift with the videogame engine Unity 3D to produce a virtual simulator of a five degree of freedom robotic manipulator. The simulator includes an interactive user interface to provide assisted instruction to the user. This means that the simulator will prevent any wrong or out of range movement programmed by the user by displaying on the virtual reality world alert messages.

The simulator is part of a bigger system which aims to reproduce the off-line simulated movement sequences into the real word. This task is achieved by transferring the generated files of the simulation to a real Mitsubishi Movemaster RV-M1 robot. The complete system architecture is shown in Fig 1.

The complete system intends to enhance the students' learning experience, since they are not restricted to design and simulate movement sequences, but they are allow to test their simulation in the real model and evaluate if the movement sequence works as they planned.

It is important to state that the described system does not seek to replace the instructor, but to optimize the use of the real model and allow the student to have a learning experience as realistic as possible.

978-1-5090-0240-5/15/$31.00 ©2015 IEEE 9-11 December 2015, Tecnológico de Monterrey, Mexico City Campus, Mexico City
2015 International Conference on Interactive Collaborative and Blended Learning (ICBL)

II. BACKGROUND

A. Oculus Rift Best Practices

If VR experiences ignore fundamental best practices, they can lead to simulator sickness. Historically, many of these problems have been attributed to sub-optimal VR hardware variables, such as system latency. The Oculus Rift represents a new generation of VR devices, one that resolves many issues of earlier systems [3]. Even with a flawless hardware implementation, improperly designed content can still lead to an uncomfortable experience.

Foregoing the Heads-Up Display (HUD) and integrating information into the environment would be ideal. In general, Oculus discourages the use of traditional HUDs. Instead, Oculus best practices encourage developers to embed that information into the environment itself. Although certain old conventions can work with thoughtful re-design that is mindful of the demands of stereoscopic vision simply porting over the HUD from a non-VR game into VR content introduces new issues that make them impractical or even discomforting [3].

Consider building informational devices into the environment itself. Remember that users can move their heads to glean information in a natural and intuitive way that might not work in traditional video games.

What's important is presenting information in a clear and comfortable way that does not interfere with the player's ability to perceive a clear, single image of the environment or the information they are trying to gather.

B. The Sixense SDK

The Sixense Core API allows game developers to create games and applications that make use of six-degree-of-freedom motion tracking of position (X,Y,Z) and orientation (pitch, roll, yaw) with Sixense-powered controllers [4].

The Sixense Core API is a cross-platform minimal C interface for getting data and sending commands to and from Sixense-powered devices. A Unity plugin is also included for easy integration, with support for Unreal Engine 4 and other game engines.

III. METHODOLOGY

The methodology implemented in this project is based on subsystems that includes an input stage, a processing unit and an output stage. Communication between each stage is shown in Fig.1. The functionality of each one is described in the following sections.

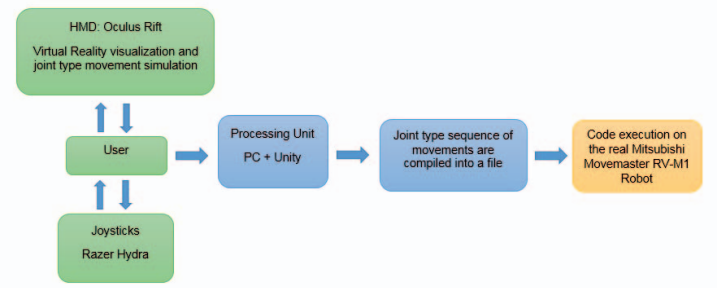

Fig. 1. System Block Diagram

A. System Components

System Components are enlisted below:

Hardware:

- Oculus Rift
- Razer Hydra Joysticks
- Mitsubishi Movemaster RV-M1 Robot

Software:

- Oculus SDK
- Sixense SDK
- Denford Robot Communications
- Unity 3D
- Sketch-Up and Blender

Unity Assets:

- Takohi Inverse Kinematics
- APEX Game Tools

B. Process Analysis

The first step in order to develop the VR application was to analyze the process for virtualization, in this case the simulation of the Mitsubishi robot manipulator, identifying the links of the robot, each degree of freedom (DOF), and the tools involved (gripper), and measure them to create the 3D model replica of each one of them, as shown in Fig. 2. This step also involved the analysis of the teach pendant programming mode in order to measure the motion limits of each DOF.

C. Oculus Rift Setup

Oculus Rift setup required the Oculus Runtime for Windows V0.4.4-beta using DK2 device with 960 x 1080 HD resolution per eye, 72 Hz refresh rate.

System Requirements are listed below:

- Unity pro 4.5 or higher version.
- Oculus SDK for Windows Runtime v0.4.4

Results of the initial setup of the Oculus Rift is shown in Fig. 3.

D. Oculus Rift User Interface

The User Interface (UI) for the Oculus Rift was designed using the Oculus Unity integration package 'OVR' and some of its data structures such as the OVRCameraController prefab, the OVRCameraRig and the OVRManager. Getting started requires the same basic setup as using the OvrCameraController prefab: creating a scene and then adding the Oculus integration package into the project's assets.

The OVRCameraController prefab is included in the Oculus Integration assets and it contains the stereo cameras used for Rift integration.

To display the UI using Oculus Rift both left eye and right eye must include a camera component and both prefabs must be child components of the OVRCameraRig prefab.

The UI is created with a C# script included in the Oculus Unity integration package named OVRMainMenu, which contains objects and functions to render out a UI that allows major interactivity with the virtual world. OVR data structures

such as OVRGUI, OVRCrosshair, RectTransform, Canvas, RenderMode and RenderTexture allow to create UI elements that can be displayed in the Oculus Rift. Results of the UI rendering are shown in Fig. 4.

E. Integration of the Sixense SDK for Razer Hydra

The Razer Hydra provides a controller for each hand that may report its absolute position and rotation, along with a series of standard gamepad controls.

Before you can make use of the Razer Hydra controller in Unity 3D you will need to download the Sixense SDK from the Unity Asset Store and export it to your project's assets. This will import all scripts and example scenes available for the Razer Hydra to your current project's assets folder, including the sixense.dll library which is an important element to avoid any possible malfunction.

The unique feature of the Razer Hydra is being able to detect each the left and right controller's absolute position and rotation in space. When it comes to generating position input events, there are two options available. The first option is to receive each x, y and z axis update as separate events.

The second option is to receive only a single position event that includes each axis as a separate parameter.

Mapping of the functions into the Razer Hydra are shown in Fig.5.

F. Programming the virtual model in Unity 3D

Once in Unity, each part of the robot was assigned to a different GameObject and scale was adjusted individually to achieve a final scale factor of 0.3. Finally the robot was constructed by embedding each GameObject in a hierarchical way according to the hierarchical order of the DOF's links in the real world robot.

The robot's virtual model was programmed using built-in functions of Unity, mainly the Transform function, which allows to store and manipulate the position, rotation and scale of the GameObject in a scene. Each GameObject movement is done having a point of origin as a home reference, which is where the robot is initialized when the program starts.

By programming the degree and direction of rotation of each degree of freedom of the robot, according to the manufacturer specifications, many things are accomplished. First, rotation angles information can be displayed to a GUI in order to assist the user when a high precision task is required to grab an object or if user is assigned a task to configure the degrees of rotation necessary to move an object form one place to another.

The control of each DOF is done using the Razer Hydra controls, which allows the user to efficiently manipulate each link of the robot as it is done in the real model using a complicated teach pendant. The controls also allow the user to display the UI when specific information of each DOF may be required. Assignment of the control triggers that allow as well the generation of the instructions program that will be downloaded to the real world robot. This file is generated programmatically from a C# script and it will instruct the robot which degree of freedom, angle and direction to move.

IV. RESULTS

The application is programmed using a videogame engine, Unity 3D, using the built-in function Transform and methods embedded to work with local Euler angles. The corresponding scripts are coded using C#. The Takohi Inverse Kinematic asset is used to program the correct joint displacement in the virtual world. At this stage, the robot joints angles are displayed in a GUI. In order to program the robot to follow the most optimal movement sequence in a 2D space, the APEX Game Tools asset was used. The implementation of the collision-free motion planning algorithm produced successful results in two dimensions, where the robot's end effector moves from one point to another on the x/y plane. As a result, collision-free paths and optimal movement sequence are achieved as shown in Fig.6.

In the final version of the simulator the user interacts with the Oculus Rift, the virtual version of the Mitsubishi Movemaster RV-M1 Robot and the Razer Hydra controls in two possible modes: teach pendant mode or automatic motion planning mode.

Different test scenarios were designed to ascertain the functionality and precision of the programs generated in the virtual world. First the robot was calibrated to make sure the movements translated to the real robot were the same, joint programming had the best results since the movements were all replicated exactly by the real robot, so it was decided to use joint programs as a reference for the positions programming. Generating different sequences, evaluations consisted in verifying the complete program execution and contrasting the positions with the positions obtained with the joint program.

The obtained results were: Positions programming had a slight error, when compared with positions from the robot in the virtual application, of 2% in the x and y axes and 5% in the z axe.

With this data, the next evaluation was to test the tasks the application could do when used by students. The first task was to move both the real robot to three positions, previously marked, students first manipulated the real robot and then repeated the same task in the virtual application. A second task, using the same procedure to use both robots, was to manipulate the robot to grab an object and displace it to a set location. Evaluation consisted on verifying execution, measuring the completion time on both robots and the perceived experience by users.

Results showed that students took less time completing the tasks in the virtual application, this because of the difficulty of handling the real robot compared to the easy comprehension on the controllers of the application; finally students were asked to give their feedback, from where it was obtained the following:

- 53% think it is faster and 20% think it is attractive
- 80% believe that the system will allow better understanding of the robot operation
- 60% suggest that it is easy to enter a sequence robot in the virtual application
- In 70% of cases the sequence was executed with precision in the real model

- 100% of participants think that the system can be a beneficial tool for learning

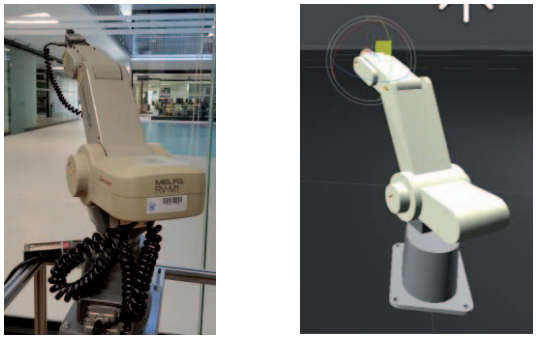

Fig. 2. (a) Real world Mitsubishi Movemaster RV-M1; (b) 3D model

Fig. 3. Oculus Rift Configuration Helper.

Fig 4. Virtual visualization using the Oculus Rift.

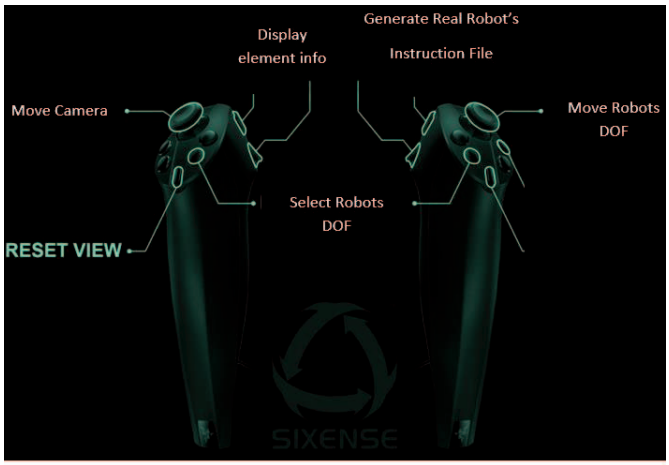

Fig. 5. Mapping of the functions of each element of the Razer Hydra controls.

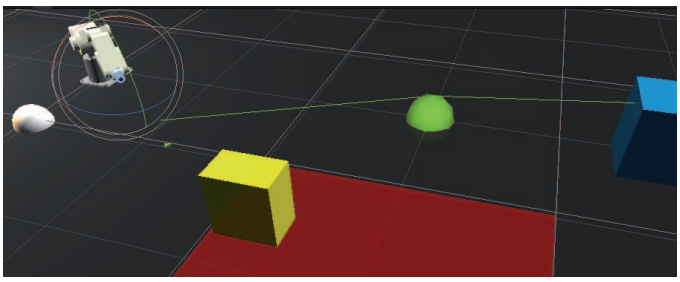

Fig 6. Pathfinding implementation between two points.

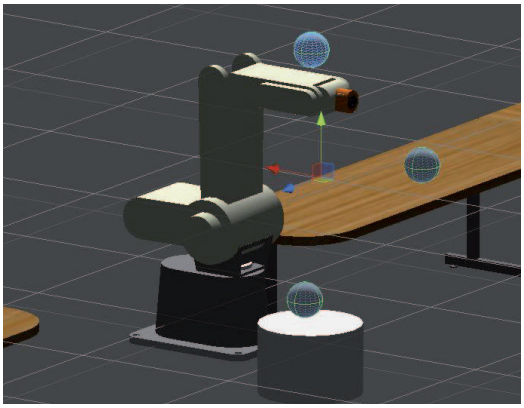

Fig 7. Testing scenario for the virtual environment.

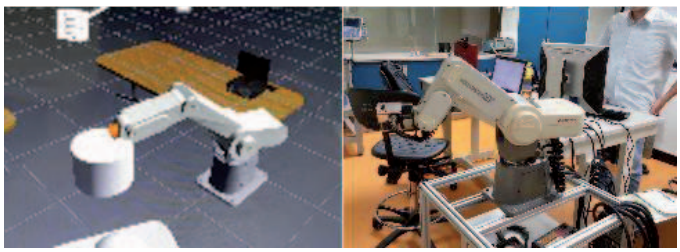

Fig 8. Engineering students accomplished their lab practice.

Fig 9. Engineering students testing the virtual reality simulator.

V. DISCUSSION

Some benefits of using Virtual Reality for robotics learning include efficient use of real machinery and flexibility on building different practice scenarios. It can also benefit learning programs that depend or benefit from efficiency and flexibility of the learning hours.

Learning users can make use of the virtual reality environment and have interaction with it, chances to make mistakes are allowed, since simulation prevents user or equipment damage. Users can also have flexibility of the hours dedicated to the learning process, as the system requirements are minimum and users can make use of it simultaneously and in desired hours, not depending on the schedule of a real robot.

The virtual reality allows users to have realistic simulations, which results in a high reliability of the system.

Using Virtual Reality elements for controlling the Mitsubishi robot manipulator, along with the automatic creation of programs that are used to reproduce the movements made in the virtual environment, have proved satisfactory results by minimizing robot programming time and by easing the routines' implementation. The resulting application encourages immersive interaction by using the Oculus Rift and enhance learning through a realistic simulation of the Mitsubishi Movemaster RV-M1 Robot.

VI. CONCLUSION

Based on the obtained results and tests, it can be determined that the application has a high potential to become a very reliable engineering learning tool. The system is fully functional and it is presented in a stage from which it can be modified to be used with different platforms, operative systems, controllers and different robot models. Future modifications can be made to the application easily using the Unity Game Engine.

The further stage of research will be concentrated on the implementation of more efficient pathfinding algorithms and on improving the user interface in order to make it more friendly and useful according to the user's requirements.

REFERENCES

[1] V. S. Pantelidis, "Reasons to Use Virtual Reality in Education and Training Courses and a Model to Determine When to Use Virtual Reality", Themes in science and technology education, Special Issue, Klidarithmos Computer Books Vol. 2 No 1-2, 2009, pp 59-70.

[2] Pantelidis, V. S. (1996). Suggestions on when to use and when not to use virtual reality in education. VR in the Schools, 2(1), 18. Retrieved from http://vr.coe.ecu.edu/vrits/2-1Pante.htm

[3] Oculus Best Practices. Oculus VR, 2015, pp 30-35. http://static.oculus.com/documentation/pdfs/intro-vr/latest/bp.pdf

[4] Sixense Core API. Sixense, 2013. sixense.com/sixensecoreapi

[5] J.C. Latombe, Robot motion planning, Kluwer Academic Publishers, London, 1993.

[6] K. Foit, Automatic programming and generation of collision-free paths for the Mitsubishi Movemaster RV-M1 robot, Journal of Achievements in Materials and Manufacturing Engineering 47/1 (2011) 57-65.

[7] Pantelidis, V. S. (1995). Reasons to use virtual reality in education. VR in the Schools, 1(1), 9. Retrieved from http://vr.coe.ecu.edu/vrits/1-1pante.htm

[8] M.E. Macías, E. D. Guridi, "Emulation of Real Processes to Improve Training in Automation", Int. J. Engng Ed., Vol 25, 2, March 2009, pp 358-364.

[9] Muñoz L., Rudomin I. Laboratorios virtuales asistidos. Tesis de maestría. Instituto Tecnológico y de Estudios Superiores de Monterrey campus Estado de México. 2003