Virtual Reality techniques for planning the Offshore Robotizing

Felipe Carvalho, Alberto Raposo
TecGraf Institute
PUC-Rio
Rio de Janeiro, Brazil
{kamel,abraposo}@tecgraf.puc-rio.br

Ismael Santos, Mauricio Galassi
CENPES
Petrobras
Rio de Janeiro, Brazil
{ismaelh, mauricio.galassi}@petrobras.com.br

Abstract— **Remote locations such as ultra-deep water reservoirs (400 m or deeper) have been increasing the production complexity and logistic costs for Oil & Gas industry. In such conditions, oil exploration feasibility depends on new technologies to optimize production efficiency. One possible solution to this challenge is to increase the degree of automation in production units. New design concepts consider the use of robotic devices in such scenarios. In this paper we present the use of virtual reality techniques in a robotics framework, SimUEP-Robotics (Robotics Simulator for Stationary Production Units), aimed to enable planning the offshore platform robotizing. SimUEP-Robotics has an integrated Virtual Reality Engine specially tailored to provide realistic visualization of large offshore scene models in an immersive environment. Through the use of those visualization tools it is possible, for example, to better understand the quality of the planned robot trajectory and propose new algorithms that can be further evaluated in the virtual environment. We argue that the validation process in an immersive virtual environment reduces risks and costs of real operation tests scenarios.**

Keywords—virtual reality; robot simulation;offshore platforms.

I. INTRODUCTION

The vast majority of new Oil & Gas fields present high production complexity and logistic costs, in addition to several other challenges. In Brazil, exploration in the pre-salt layer represents a mark in the oil industry, breaking production records in deep and ultra deep waters. The pre-salt platforms operate far away from the coasts at distances that can exceed 300 km, increasing significantly transportation costs, logistics infrastructure and response time, especially in emergency situations. Under such conditions, oil exploration feasibility depends on new technologies to optimize production efficiency.

Robotics and automation can greatly improve the operational quality and productivity of offshore Oil & Gas installations (From 2010). Robots can work non-stop without the need for intervals and pauses, are less prone to errors, and also more reliable than human workers. This can increase the operation of the platform by reducing down-time and production stops, which are main concerns for the platform operators. Introducing robotic and autonomous solutions will also decrease costs related to commissioning and logistics, particularly by avoiding flying human operators to distant fields. (S. O. Johnsen 2007).

The next generation of offshore platforms will present high degree of automation. New design concepts consider the use of robotic systems to accomplish autonomous or teleoperated tasks on production plants, even though there are several technological challenges that should be overcome to have feasible robotics technology available offshore. In this paper, we propose an offshore robotics framework as a first step to enable the planning of offshore platform robotizing, using auxiliary virtual reality tools and techniques aiming to improve the understanding of the planned task throughout its execution in an immersive environment.

The SimUEP-Robotics is formulated concerning the offshore requirements. It is a flexible framework that allows rapid prototyping development of new applications through the addition of customized robots and devices in the target scenario by the use of an interactive editor. The proposed framework also simplifies the development and validation of the control algorithms by integrating virtual and real robots, sensors and different scenarios.

Virtual Reality (VR) technologies can improve the understanding of any engineering project. In the present work those techniques are used for visualizing the offshore platform robotizing activities. Immersive visualization environments for robotics simulation have the potential to reduce the complexity and difficulty in visualizing and validating simulations of operations performed by robots on a real Stationary Production Unity (SPU). The visualization tools manage data and present visual information that enables professionals to observe and acquire a better understanding of problems, such as robot motion planning. Without visualization, it is hard to understand the quality of the solution found for the movement of the robot based only on simulation results. Validation reduces risks and costs of testing operations in real scenarios.

II. RELATED WORKS

Different robotic frameworks and simulation platforms have been developed both for research purposes and industrial applications. However some important functionality present in our proposed framework is absent in the existing ones. Before starting the development of SimUEP-Robotics we made a thoroughly comparison among the most advanced tools available in the market at that time (2011), such as Webots (Cyberbotics 1996), Microsoft Robotics Developer Studio (Microsoft Corp. 2010), V-REP (Coppelia Robotics 2010) and ROS (Quigley 2009); and other industrial application tools such as RobotStudio (ABB 2001) and ROS industrial (ROS-Industrial™ Consortium 2012).

Webots is a fast prototyping and simulation software for mobile robotics. Complex robotic scenarios can be designed, with different robots that can be equipped with simulated sensors and actuators. The advantages are that the robot controller can be programmed with other development environments and that it is available on any operation system (OS). The main disadvantage is that it requires a paid license to have full capabilities and an annual renewal license for software update. The ability to build a customized scenario is a feature also presented in SimUEP-Robotics. In our solution, the robot controller can also be programmed by external Robotics Simulators as long as they are compatible with ROS (Figure 1).

Robotics Developer Studio (RDS) is a Windows-based environment for robot control and simulation developed by Microsoft. RDS is based on Concurrency and Coordination Runtime (CCR) and Decentralized Software Services (DSS) concepts. The first one is a .NET-based concurrent library implementation that manages asynchronous parallel tasks. The second is a lightweight state-oriented service model which allows multiple services to run on a single node achieving complex behaviors. The current version, RDS 4, is free. However, it only works with Windows. In our solution we also use ROS and its ecosystem's components.

V-REP is a virtual robot experimentation platform that has an integrated development environment and is based on a distributed control architecture. In this way, objects can be controlled via an embedded script, a plugin, a ROS node, a remote API client or a custom solution. V-REP is open-source, but requires payment to have all functionalities. We also perceived the V-REP integration with ROS as a promised solution that we decided to pursuit.

RobotStudio is a simulation and offline programming software that was developed by ABB Robotics. RobotStudio is built on top of the ABB VirtualController, which is an exact copy of the software that runs on the robots from ABB. In this way tasks like training can be performed without disturbing production. The program requires a paid license to have all functionalities and is restricted to ABB models.

The Robot Operating System (ROS) is a multiplatform meta OS which establishes a communication layer and some services and applications provided through a package framework. It is a software framework for robot software development, providing operating system-like functionality on a heterogeneous computer cluster. Willow Garage maintains the core system with an open-source free software license. An extensive worldwide community efficiently expands it. As an OS it provides hardware abstraction, low-level device control, user transparent inter process connection and communication based on the message passing paradigm. The process nodes are identified by its corresponding IP address, so network nodes integration is automatically implemented. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to a "robot framework".

For robotics simulation, ROS ecosystem integrates the Gazebo project, which is a three dimensional multi robot simulator providing sensor data and physical interaction between objects. The Gazebo integration with ROS inspired very much our solution indeed. In fact, we decided to create SimUEP-Robotics due to the lack of powerful VR tools and techniques in the ROS ecosystem.

When considering industrial applications, ROS-Industrial is a BSD-licensed software library which extends the capabilities of the ROS software to industrial applications. It is a recent initiative from Southwest Research Institute (SwRI), being supported by the ROS-IndustrialTM Consortium (RIC).

III. SimUEP-Robotics Architecture

In robotics applications, the classic programming approach of writing a specific code to each project is being modified. The new programming paradigm is the use of multiple reusable subtask programs to accomplish current and future tasks. This not only decreases the rework of researchers but also reduce the project development time. In this context, a flexible architecture that permits rapid prototyping of different scenarios and applications is a valuable tool for the industry. Additionally, integrating virtual and real robots, environments, sensors and objects improves the development capacities.

As mentioned earlier ROS framework establishes a communication layer between different process and devices, providing broadcast topics with periodic messages and on demand services. Furthermore, ROS focuses on the message content, letting the sender and receiver transparent to each other. In virtual and real environment integration scenarios, this transparency permits to change between resources without code reimplementation. To increase the portability of applications, standard messages to common required resources, such as laser range finders and robot's joint state, are available.

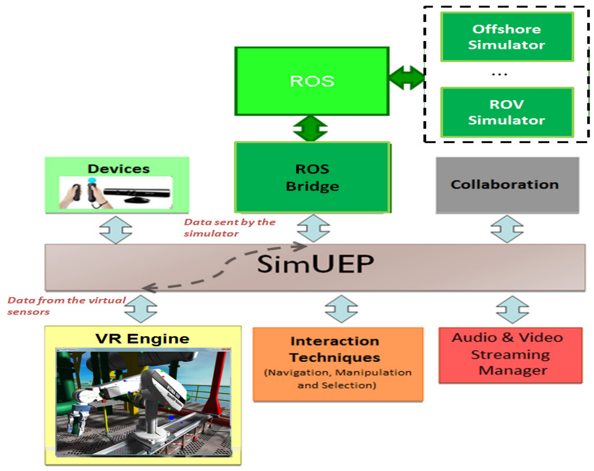
Offshore Simulator
...
ROV Simulator
ROS
Devices
ROS Bridge
Collaboration
Data sent by the simulator
SimUEP
Data from the virtual sensors
VR Engine
Interaction Techniques (Navigation, Manipulation and Selection)
Audio & Video Streaming Manager

Fig. 1. Diagram representing the SimUEP's component based architecture.

Figure 1 shows the SIMUEP-Robotics architecture, which is component-based having, among other components, the VR-

Engine, which is responsible for the visualization of the simulation of actual operations performed with robots in offshore scenarios. Other components of the framework provide resources for manipulating different 3D input devices, 3D interaction techniques, audio and video streaming, collaboration and a communication interface bridge to the ROS system, named ROS-Bridge.

In the following sections, we present further information about ROS Bridge and the Robotics-Simulator Modules currently available: Topside Offshore Simulator and Underwater ROV Simulator.

A. *Ros Bridge*

The ROS Bridge is responsible for exchanging data between ROS compatible robotics simulators and the virtual robots created by the VR-Engine according to the target virtual scenario where the robotic task is being executed. The main goal of development of ROS Bridge is to make the data from the VR-Engine available to ROS compatible simulators. Data communication between the VR-Engine and ROS applications are controlled by ROS Nodes, which are processing units capable of establishing communication with each other through topics or services:

- *Topics*: structures that store messages in an asynchronous communication model (public/subscribe). The nodes publish and receive information in topics that are identified by names.
- *Services*: asynchronous communication with client/server architecture (one provider, multiple users).

The integration of ROS in SimUEP-Robotics (Figure 2) allows the creation of a communication layer between the visualization of a virtual scene and the simulation of real robot's tasks. The exchanged messages are similar to those used to control real robots.

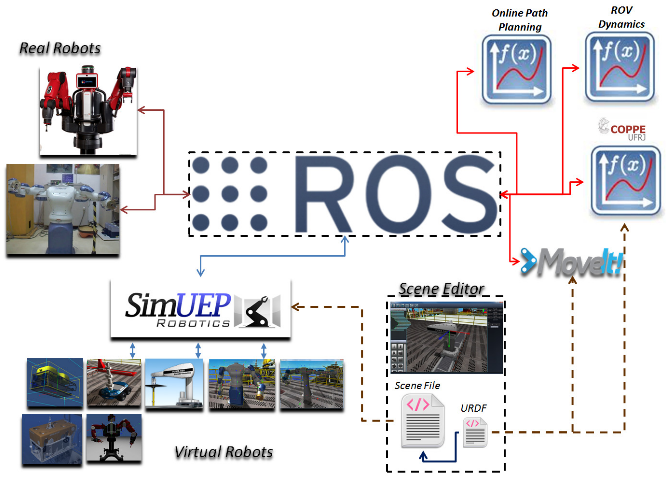

Fig. 2. ROS integration in SimUEP

The SIMUEP component based approach enables the creation of new "bridges" to other robot frameworks, like Rock (ROCK 2012) that is in our near future development plans.

B. *Robotics Simulator Modules*

At present, the SimUEP-Robotics has two robotics simulator modules implemented allowing us to instantiate two different applications: one for topside offshore operations and another for sub-sea engineering. For the topside, the robot simulation strategy depends on the used virtual device and the task that should be planned. As a proof-of-concept we have developed a pick-and-place task in a typical offshore scenario. In the second application, the ROV simulator module implements the dynamic motion equations for an underwater ROV vehicle from Liu Hsu et. al. (2000) (Figure 3). SimUEP-Robotics can be extended with other ROS compatible robotic simulators addressing other operation problems.

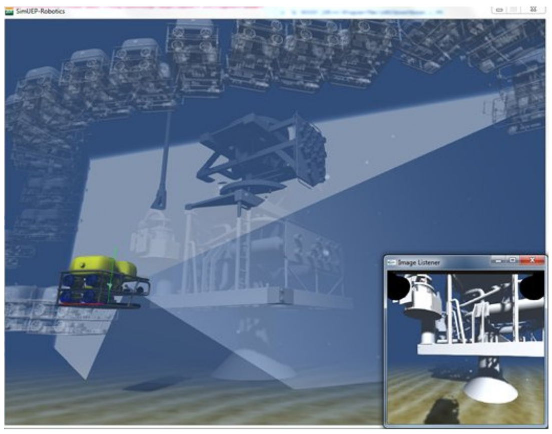

Fig. 3. Visualization of ROV Trajectory using virtual realty techniques.

The robotics-simulator modules implementation seamlessly integrates some open-source community maintained ROS packages according to its needs. This integration is driven by the definition of the robots on the Unified Robot Description File (URDF) associated to its virtual robot used in the target scenario. The URDF is an XML file format that provides information about robots and associated sensors (e.g., laser, camera), trough joints and links definition at different levels such as kinematic, dynamic and visual. Sensors can be incorporated using the Gazebo interface, which will be described later.

Upon reading the robot URDF files in the scene the Robotics-Simulator configures the necessary standard messages to interact with sensors and robots. In addition, it uses different ROS integrated libraries and packages to perform tasks like: navigate with mobile robots (navigation stack), transform data through different coordinate systems (TF package), build 3D maps (OctoMap) and process images (OpenCV).

After selecting one of the SimUEP-Robotics available applications (instantiated by selecting an specific Robotics-Simulator module according to the addressed task), the VR-Engine reads the scene description defined by the user and the related URDF files to create the virtual environment together with the visual representations of the robots (Figure 4, steps 1 to 3). The ROS-Bridge is responsible for creating a ROS's root node and exchanging information through its associated ROS topics and services. Some of those created topics are the sensor's camera and laser information for each virtual sensor instantiated in the virtual scenario according to its URDF definition. After that, the VR-Engine starts the corresponding simulation module that communicates with robots controllers and read sensor data (Figure 4, steps 4 to 7). Besides that, it also starts services to calculate robot's dynamics and kinematic under request of other nodes (see Figure 4, step 8). The robot's behavior in virtual environment (yellow side) is the same in the real environment (red side). In fact, we have conducted some experiments in the university's lab confirming this assumption using the same Robot Simulator Module.

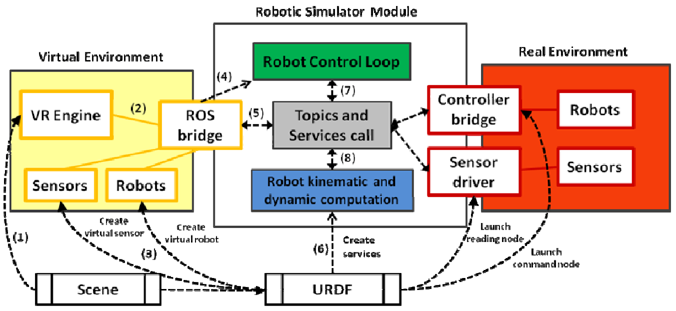

Fig. 4. Diagram representing the Robotics Modules interaction with SimUEP's ROS Bridge

In the following section, we describe the SimUEP-Robotics Virtual Environment main features and VR tools available to Robotics-Simulator modules.

IV. Virtual Environment

In any instantiated SimUEP-Robotics application, the virtual environment is loaded from a scene description file. This file contains a scene description of the target scenario. A scenario is basically composed by an observer, environment objects, lights and robots.

SimUEP-Robotics uses the game engine Unity3D as its graphic engine component. Unity3D is a tool for developing games created by Unity Technologies. This engine was chosen due to resources and graphic quality it provides, enabling the creation of realistic environments.

A. 3D Scene Description

The scene description is an XML file that describes the target scenario. It has a hierarchical structure describing all items in the environment. The first item of this file is displaySettings, which defines the properties of virtual camera (near and far planes) and output settings to enable visualization modes for immersive systems, e.g., CAVEs, PowerWalls and other projection systems. Global settings of the physics engine, such as update frequency and gravity vector are specified in item physicsSettings. The user visualizes the scene by specifying the observer item, which has properties such as position, orientation, height and speed. All the lights are defined inside the item lights. They can be of type point, spot and directional. Common attributes are position, orientation, color, intensity and type of shadow. The scene objects are specified by envObjects items. For each envObject the following attributes are required: position, orientation, bounding box, mesh, and some physical properties, such as mass, inertia matrix and material information (friction, bounciness, etc).

Each robot in the scenario file is defined by a URDF, a position and a set of actuators and sensors. Some of these sensors are also defined in the URDF, for example, cameras and lasers. For each sensor and actuator, it is necessary to define a topic name to identify the associated ROS message.

B. Virtual Robotic System

We have implemented eight different ROS compatible robots, namely, Motoman DIA10, Motoman SDA10, Grant Crane, Seekur, a generic ROV, Baxter and ROV XLS 150. Six of them are in the Figure 5. Any robot can be instantiated in any application and positioned (position, attitude and scale) defined in the 3D scene file of the target scenario.

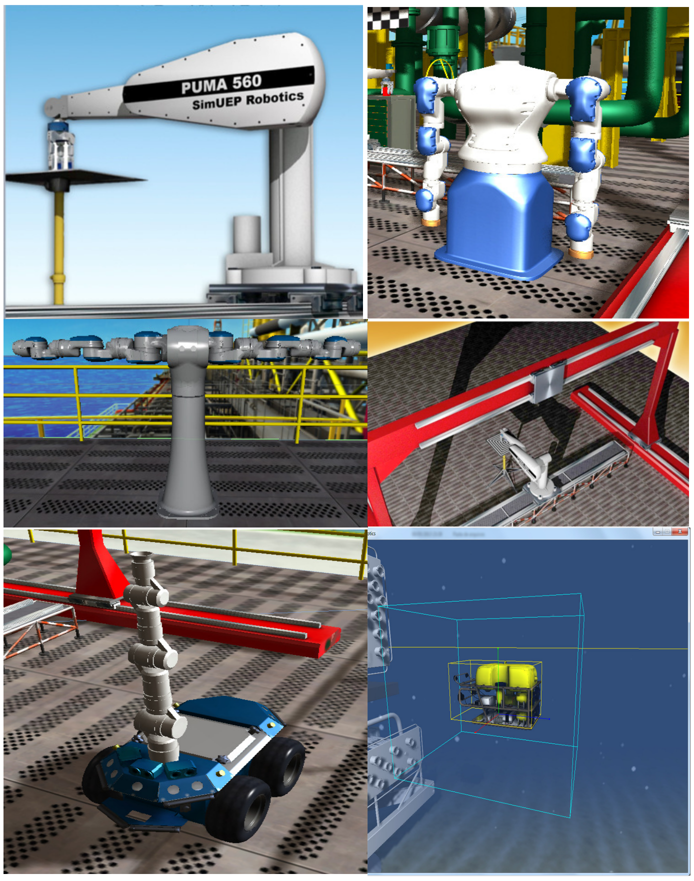

Fig. 5. Six Robots available in SimUEP-Robotics (Puma560, Motoman DIA10, MotoMan SDA10, Grant Crane, SeekUR and ROV).

C. Actuators

The actuators are used to modify some positional aspects of the robots. There are two actuators available: the first one is based on the use of joint values; and the second on the absolute position of the whole robot.

D. Sensors

Sensors are used for a robot to perceive the environment and, through them, being able to react to its surroundings. Each robot may be composed of one or more types of sensors. In

SimUEP Robotics, four different types of sensors are available: lasers, cameras, proximity sensors, and collision sensors.

Camera sensors are able to generate images from the virtual environment. A robot can have multiple camera sensors that are normally positioned along its links. This positioning allows the visualization of different views of the same robot at the same time. Each camera is described (within the URDF file) following the description format SDF (from Gazebo Project) and has attributes such as image size, lens aperture, pixel format, among others (Figure 6). The Virtual Environment camera sensor component is responsible for capturing data from the camera and sending them to the Robotics-Simulator Module through the associated ROS topic defined in the robot URDF file. A feature that generates a visual representation of these cameras is available.

```
<!-- Camera -->
    <gazebo reference="linkCamera1">
        <sensor:camera name="linkCameral">
                <imageSize>512 512</imageSize>
                <imageFormat>R8G8B8</imageFormat>
                <hfov>0.7853982</hfov>
                <nearClip>0.01</nearClip>
                <farClip>4.5</farClip>
                <updateRate>0.25</updateRate>
        </sensor:camera>
    </gazebo>
```

Fig. 6. Camera sensor description in URDF file.

Robots can also use sensors like lasers. These are used to recognize the environment through a one-dimensional scan. As a result of this scan a vector containing values corresponding to the distance between the laser and the obstacle of the corresponding scenario is obtained. Figure 7 shows the visual representation of two lasers in SimUEP, as well as images that represent the distance vector captured by each one. The laser sensor is also defined in the robot URDF file. Its parameters contain information such as the initial position of the sensor, resolution and maximum and minimum angles, topic name of the message, as well as parameters that control their display in virtual scene.

Collision sensors are sensors that allow the robot to identify when an object is close to some of its physical components (links).This allows the robot to be able to avoid obstacles, and enable the measurement of these obstacles to a certain part of the robot. This sensor is implemented by using bounding boxes. When any object penetrates a bounding box, all interception points are informed (Figure 8) using the message sensor_msg/PointCloud2. The description of this sensor is present in the scene description file and it has a parameter for the precision.

Proximity sensors are similar to collision sensors; however, they only inform the name of the link hit. Basically, when something enters a bounding of this sensor, a message is sent to ROS.

E. Visualization Tools

An important aspect of visualization in SimUEP Robotics is to support the evaluation of simulation results as well as the validation of different offshore scenarios composed of robots, valves and other equipment defined in the scene description file. In this scope, the following visualization data can be mentioned: Bounding Boxes and Axis Joints (Figure 9).

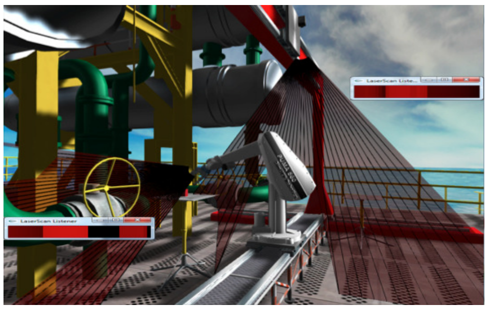

Fig. 7. Laserscan visualization in SimUEP-Robotics.

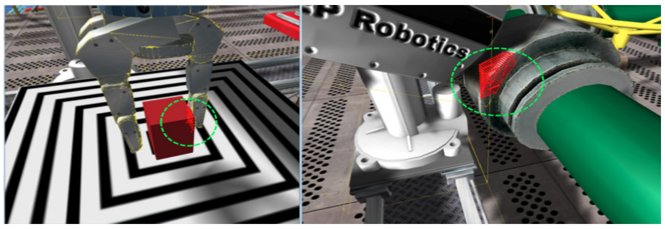

Fig. 8. Collision Sensor, the red dots are the intersections points detected in the link's bounding box.

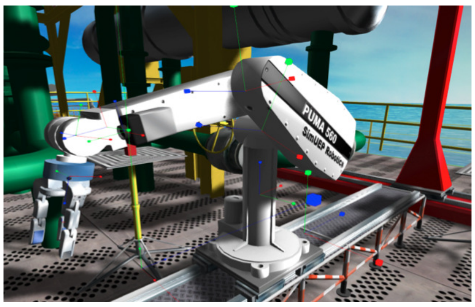

Fig. 9. Axis features of SimUEP Robotics visualizer.

The SimUEP-Robotics has two extra features, Ghostview and 3D Point Trajectory, which allow the visualization of the entire motion path performed by robots. In Ghostview the viewer captures successive snapshots of the robot along the trajectory movement (Figure 10). Thus, it may be used to visualize the robot at different points in the simulation. The later feature, Point Trajectory, allows visualization of the trajectory of a point, such as a joint or a link.

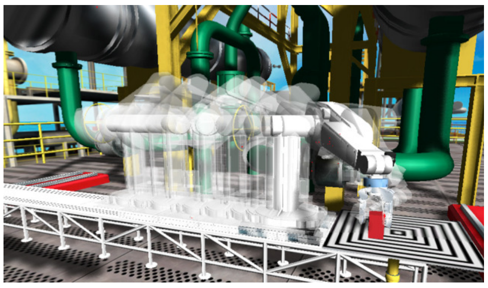

Fig. 10. Robot Trajectory visualization in Ghostview mode.

The SimUEP-Robotics can be used in desktop environments as well as in immersive environments such as CAVE, PowerWalls and other projection systems. One can navigate in the environment by means of specific controls such as 3D trackers.

V. Scene Editor

SimUEP-Robotics has a tree scene editor to compose scenes and robots (Figure 11). The editor provides access to a library of objects to compose common offshore and subsea scenarios and save it as Scene File (Figure 2). Besides this, it also has a collection of complete robots and pieces of robots, such as links, sensors and actuators. The interface uses drag-and-drop interaction to facilitate the composition of the objects in 3D space. The user can construct a robot (URDF file) from scratch by moving links individually from the library to the 3D scenarios, and finally creating joints between then.

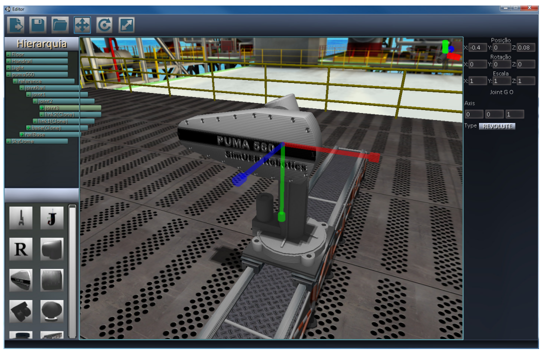

Fig. 11. SimUEP Robotics scene editor.

VI. Conclusion

Hostile environments are seen as one of the greatest challenges for oil & gas companies developing new fields. The use of robots in these situations is foreseen as a great opportunity for the robotics industry. Thus a lot of effort and investments are being pushed on the development of specialized robots for those environments. This development can be extremely improved if proper tools and methodologies are used.

Tools such as simulators and immersive visualization are promising development strategies, providing an environment for testing and visual debugging. The ability to compare different simulations with the support of different visualization tools can help in the interpretation of virtual simulated tasks allowing the development of new robots and, eventually, more appropriate offshore scenarios and tools to accomplish the proposed tasks. Furthermore, development methodologies that focus on the reuse of resources are extremely important to give flexibility in the process of developing new products.

The work presented here is based on the above mentioned development strategies by creating flexible development environment to support the modeling and visualization of the simulation of actual operations performed with robots in Stationary Production Units. ROS as communication middleware provides a powerful framework to create and reuse different robot algorithms. In the same way, the use of SimUEP-Robotics component-based platform resulted in the development of a flexible and useful framework and scene editor, which can be considered as a rich visual debugger to support the development process and to analyze the results of the simulations. Such component approach gave the possibility to develop different offshore robot applications paving the way towards platform robotizing furnishing the robotization and automation of offshore facilities.

Acknowledgments

Thanks to Petrobras for supporting this project. The authors would also like to thank the support provided by the research funding organizations CNPq and FAPERJ.

References

[1] ABB. RobotStudio. 2001. http://new.abb.com/products/robotics/robotstudio.

[2] Bjerkeng, M., A. A. Transeth, K. Y. Pettersen, E. Kyrkjebo, and S. A. Fjerdingen. "Active Camera Control with obstacle avoidance for remote operations with industrial manipulators: Implementation and experimental results." Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. 2011. 247-254.

[3] Bradski, G. "The OpenCV Library." Dr. Dobb's Journal of Software Tools. 2000.

[4] Coppelia Robotics. Virtual Robot Experimentation Platform (V-REP). 2010. http://www.coppeliarobotics.com/.

[5] Cyberbotics. Webots 7. 1996. http://www.cyberbotics.com/.

[6] Flacco, F., T. Kroger, A. De Luca, and O. Khatib. "A depth space approach to human-robot collision avoidance." Robotics and Automation (ICRA), 2012 IEEE International Conference on. 2012. 338-345.

[7] From, P.J. Off-Shore Robotics - Robust and Optimal Solutions for Autonomous Operation. PhD Thesis: Norwegian University of Science and Technology, 2010.

[8] Hornung, A., Wurm, K. M., Bennewitz, M. Stachniss, C. & Burgard, W. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees." Autonomous Robots, 2013.

[9] Johnsen, S. O., Ask, R. & Roisli, R. "Reducing Risk in Oil and Gas Production Operations." IFIP International Federation for Information Processing (Springer New York) 253 (2007): 83–95.

[10] Liu Hsu, Costa, R.R., Lizarralde, F. & Da Cunha, J.P.V.S. "Dynamic positioning of remotely operated underwater vehicles." Robotics & Automation Magazine, IEEE, Sep de 2000: 21-31.

[11] Microsoft Corp. Microsoft Robot Developer Studio (RDS). 2010. http://www.microsoft.com/robotics/.

[12] Moghaddam, A. F. , Lange, M., Mirmotahari, O. & Hovin, M. "Novel Mobile Climbing Robot Agent for Offshore Platforms." World Academy of Science, Engineering and Technology, n. 68 (2012): 29-35.

[13] Quigley, M., Conley, K., Gerkey, B. P.., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. "ROS: an open-source Robot Operating System." ICRA Workshop on Open Source Software. 2009.

[14] ROCK. Rock: The Robot Construction Kit. 2012. http://rock-robotics.org.

[15] ROS-Industrial™ Consortium. ROS-Industrial. 2012. http://www.rosindustrial.org/.

[16] Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo. Robotics: Modeling, Planning and Control. Springer-Verlag London Ltd., 2009.

[17] Technologies, Unity3D. s.d. http://unity3d.com/.