Web Environment for Robotic Manipulator Simulation

Abstract— **In this study, analysis, modeling, and simulation of three robot manipulators, in a web environment, is presented. Initially, a mathematical representation of manipulator kinematics was obtained using the Devanit Hartenberg (D-H) parameters. Subsequently, the three devices were modeled using 3D software tools, and lastly, a web simulation environment was generated through the use of several 3D frameworks.**

Keywords— robot manipulator, web environment, 3D modeling

I. INTRODUCTION

Process automation has changed substantially since its creation. Initially, the industrial revolution based its development on the exploitation of hard manual labor, which resulted in long execution times, and tasks that posed a serious hazard to workers. The current world outlook, however, has shifted. Nowadays, development of optimal processes involving several production lines, diminishing execution times, and continuous product quality improvement is sought.

Robot manipulators have been incorporated into the dynamic of industrial processes for years. They replace manual labor and use simulation environments in order to predict production errors and improve production line designs. Industrial process simulation presents important opportunities for resource optimization, in terms of algorithm design, testing, routine debugging, and global process views, which consider the behavior of robot manipulators while looking to correct possible flaws that may emerge.

The use of robot manipulators in industrial process automation requires skilled personnel who are knowledgeable about their operation and handling. Nonetheless, given the high costs resulting from robot manipulator development, academia has focused on the market, offering development environments for industrial process simulation that allow companies to create, innovate, and optimize existing developments [1], [2].

However, since most robot manipulator simulators are not freely accessed, as they are generally licensed to the robot manipulator, they are costly. This has generated interest in development of a web interface that provides accurate, effective, and efficient simulations.

The tools selected for the development of this web simulator and the reason for their selection is detailed below. Said tools include several Javascript and WebGL-based video game *frameworks* that were both considered and used, along with visual demonstrations of the development outcome with real robot examples, including KUKA, SCARA, and PUMA.

II. THEORETICAL BASIS

A. Robot manipulators

A conventional robot manipulator can be represented as a succession of links and unions (joints). These joints are commonly of the rotational or prismatic type (interlinear), which enable relative movement between links. Each of the independent motions performable by each joint, relative to the previous joint, defines a degree of freedom (DOF), and is likewise the number of independent parameters that set the situation (position and orientation) of the end element, known as the *gripper*. Robot manipulators are generally articulated arms, which behave similarly to human arms. When studying the behavior of a robot manipulator, the first task to be performed is the description of its motion, using the robot's structural parameters, which are known as kinematics.

B. Robot manipulator kinematics

Kinematic analysis is concerned with the study of robot geometry, in relation to a fixed reference coordinate system as a function of time, without considering those forces that create motion. Kinematics also studies the position, speed, acceleration, and in general, all derivatives of higher-order position variables (in relation to time or any other variable) [3]. From the above, it can be concluded that kinematics is divided into two parts: positional kinematics and differential kinematics. Positional kinematics, or simply kinematics, studies the position and orientation of each one of the parts of the robot arm, while differential kinematics focuses on

speeds and accelerations, the first of which is the principal focus of this project. Kinematics can likewise be analyzed from two perspectives: directly, so as to calculate *gripper* position and orientation, and inversely, which allows for the determination of the position and orientation that joints should have, based on the position of the final element.

In accordance with that set out on [3], in order to solve this problem, the *Denavit – Hartenberg* (D-H) method is used. This method represents components' spatial geometry in a general kinematic chain, and the robot arm in particular, in relation to a fixed reference system. This method employs the homogeneous transformation matrix, which describes the spatial relationship between two adjacent rigid components. The direct kinematics problem then finds a 4x4 homogenous transformation matrix which relates the spatial location of the end of the robot's arm to its base's coordinate system.

C. Manipulator description

The first robot used was the SCARA manipulator (*Selective Compliant Articulated Robot for Assembly*). It is a very common configuration that, as implied by its name, was created for assembly operations. Nonetheless, it has an RRP configuration (Rotational – Rotational – Prismatic), that is different from the spherical configuration both insofar as its appearance and range of applications. It has three parallel angular joints (which enable it to move and align on a plane), along with a fourth prismatic joint to move the end effector normally to the plane. The axes of the first two revolute joints are vertical, causing the links to move on a horizontal plane, while the third element moves in relation to a vertical axis [4]. Once the parameters have been established, the next step involves setting up the transformation matrix in such a way that the relation describing the direct kinematics model is obtained.

The PUMA robot (*Programmable Universal Machine for Assembly*) is a widely-known manipulator. It has six degrees of freedom, three of which serve to position the end effector, and another three for orientation [5]. In order to set forth the homogeneous transformation matrices that describe this manipulator, it is initially necessary to define the D-H parameters. Once obtained, the homogeneous transformation matrix is given for this manipulator.

Finally, the KUKA KR16 robot is widely used for small loads, as its capacity is a mere 16 kg. It also may be installed with different configurations, whether on a roof or a wall. Additionally, it has six spherical joints, and just as with the PUMA robot, three of them are for positioning the end effector, and the other three align it [6].

III. 3D MODELING

This section details various industrial robot models, beginning with their physical composition, given by parameters obtained from specification documents for each robot. The application of specialized 3D-modeling technologies makes robot simulation easier, enabling the use of several platforms. Export to different formats enables the use of various technologies and programming languages.

A. 3D design

Firstly, the SCARA robot was evaluated, using the manufacturer model as a basis. Here, its specifications are set out, as shown in Figure 1.

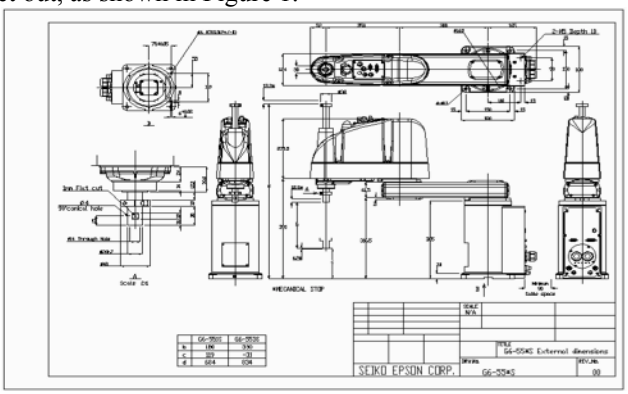

Figure 1. SCARA robot CAD design [7].

The tool used for converting models from 2D to 3D is called ARKITool [8]. This is an application installed on AutoCAD® that enables models to be converted and have surfaces applied to them. Thereafter, 3Ds Max® [9] was used (Figure 2).

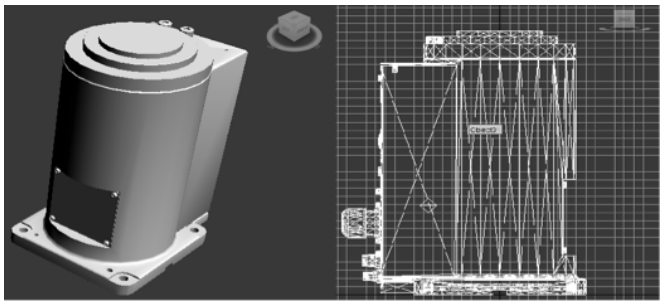

Figure 2. 3D model of a part of the SCARA robot.

This image shows that the model is no more than a set of points that form the figure. The more points present, the more complex the object is. An object can be formed based on others, where separate parts are created and then combined to form a new object. This is defined as a polygon combination, as seen in Figure 3.

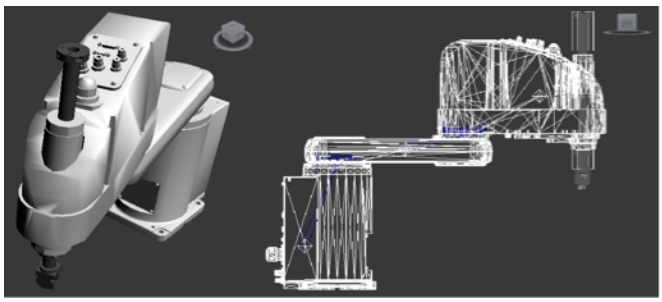

Figure 3. View of the SCARA robot.

In order to add motion to the robot, motion points must be included. To that effect, small cylinders were added and aligned with the center of rotation for each movable part of the robot, the detail of which is shown in Figure 4.

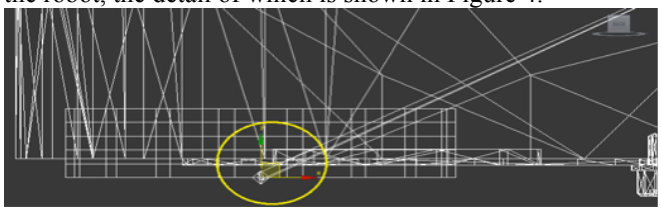

Figure 4. Rotation point.

A similar process was used for the generation of 3D models for the two remaining robot manipulators, as shown in Figure 5.

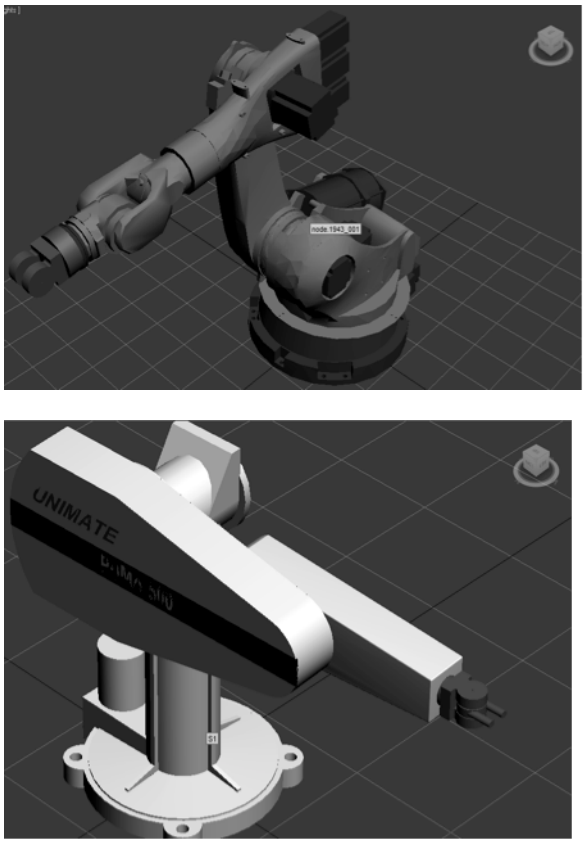

Figure 5. The KUKA and PUMA robots.

B. 3D web environment

Using the 3D models designed, a web environment was implemented for kinematic manipulator simulation, using the CopperLicht [10] and Babylon [11] *frameworks*. At first, the development was carried out with CopperLicht, where several problems arose with the *framework* logic. These prevented the advancement of work on the simulation, and so, development was migrated to Babylon. Although the scene had to be manually created, it involved more accurate logic for simulation requirements. Babylon was much better suited to the methods and image manipulation required. The interfaces for each robot manipulator are shown below in Figure 9.

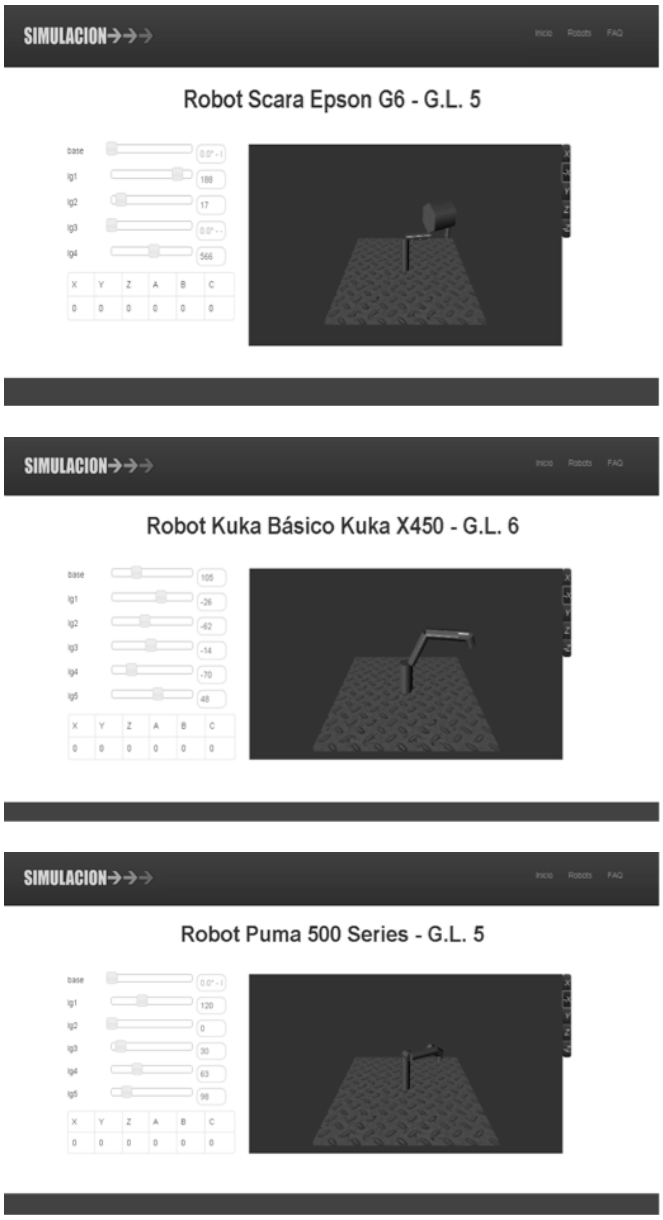

Figure 9. Robot manipulator simulation interface.

IV. Conclusions and Future Work

In this document, the process of developing a web environment for kinematic robotic manipulator simulation is presented in the form of mathematical modeling, 3D model generation of each manipulator, and is completed in the environment constructed.

Other widely known environments, such as Gazebo [12] and V-Rep [13], have numerous features in addition to the direct kinematic simulation presented here. However, they differ in that they are not web-oriented, although they may

use a tool developed by third parties, such as The Construct [14], for said purpose.

This simulator may be useful as a tool for robotics instruction, and subsequent developments in a collaborative environment within the University of Manizales will enhance 3D web development skills. Logical continuing steps include the incorporation of inverse kinematics, path planning, dynamics, and integration with ROS, as well as measurement of environmental performance through comparison of results obtained with the tool and those performed using Matlab®.

V. Acknowledgments

The authors would like to thank those undergraduate Systems and Telecommunications Engineering students at the University of Manizales who were involved in the various stages of this project for their valuable contributions, as well as the Faculty of Sciences and Engineering for the time granted to the researchers for the implementation thereof.

References

[1]. C.A: Jara, F.A: Candelas, F. Torres, "Laboratorios virtuales y remotos basados en EJS para la enseñanza de robótica industrial", *Universidad de Alicante*. 2010.

[2]. R. Pollak, J. Schützner and T. Bräunl, "RoboSim- Robot Manipulator Simulation", Robotics.ee.uwa.edu.au, 2015. [Online]. Available: http://robotics.ee.uwa.edu.au/robosim/. [Accessed: 09- Oct- 2017].

[3]. J. Craig. *Introduction to Robotics*. México: Pearson, 2005.

[4]. O. Vivas, "Predictive Control of a SCARA Robot", *Ingeniare*, vol. 14, no. 2, pp. 135-145, 2006.

[5]. G. Kim, "Programming and Controlling PUMA Robots Arms", *Open-robotics.com*, 2005. [Online]. Available: http://open-robotics.com/wp/wp-content/uploads/presentation/programming-and-controlling-puma-arms.pdf?ckattempt=1. [Accessed: 08- Oct- 2017]

[6]. Kuka Robotics, "KUKA Robots industriales - Pequeños Robots", *Kuka-robotics.com*, 2015. [Online]. Available: http://www.kuka-robotics.com/es/products/industrial_robots/small_robots/. [Accessed: 10- Nov- 2017].

[7]. Epson Robotics. *Epson SCARA G6 Product Detail*, 2015. [Online]. Available: http://robots.epson.com/product-detail/3. [Accessed: 19- Jun- 2017].

[8]. ARKITool "ARKISoft, utilidades CAD.", *Arkisoft.es*, 2017. [Online]. Available: http://www.arkisoft.es/programas/arkitool. [Accessed: 15- Nov- 2017].

[9]. Autodesk. 3ds Max | "3D Modeling, Animation & Rendering Software | Autodesk", *Autodesk.com*, 2017. [Online]. Available: http://www.autodesk.com/products/3ds-max/overview. [Accessed: 19- Jun- 2017].

[10]. "CopperLicht –Open Source JavaScript 3D Engine using WebGL", *Ambiera.com*, 2015. [Online]. Available: http://www.ambiera.com/copperlicht/. [Accessed: 19- Nov- 2017].

[11]. "Babylon.js demos & documentation", *Babylon.js*, 2015. [Online]. Available: http://www.babylonjs.com. [Accessed: 19- Jun- 2017].

[12]. Open Source Robotics Foundation, "Gazebo", *Gazebosim.org*, 2016. [Online]. Available: http://gazebosim.org. [Accessed: 16- Aug- 2017].

[13]. Coppelia Robotics, "Coppelia Robotics v-rep: Create. Compose. Simulate. Any Robot", *Coppeliarobotics.com*, 2016. [Online]. Available: http://www.coppeliarobotics.com/index.html. [Accessed: 12- Aug- 2017].

[14]. The Construct, "The Construct – Just Simulate!", *Theconstructsim.com*, 2016. [Online]. Available: http://www.theconstructsim.com. [Accessed: 17- Aug- 2017