Drone Factory

Vania Elizondo
Drone Factory
Tecnológico de Monterrey
Monterrey, México
vaniaelizondomartinez@gmail.com

Gabriel Zamora
Drone Factory
Tecnológico de Monterrey
Monterrey, México
gzl.2995@gmail.com

Juan Palacios
Drone Factory
Tecnológico de Monterrey
Monterrey, México
jnpdrpalacios@gmail.com

***Abstract*—This document contains the work done for the Drone Factory activity regarding the kinematic calculus, robot modelling and connection for Unity to build a simulation of the manufacturing laboratory at ITESM in Monterrey, Nuevo León.**

Keywords—virtual reality, Unity, UR5e, Kuka, Networks, kinematics.

I. Introduction

Simulation is always a powerful tool that helps people understand or be trained with laboratory equipment. Nowadays, since virtual reality technology emerged, simulation tools may be upgraded in order to improve in areas such as immersion and precision. In a factory there are too many machines that interact with each other and require to be manipulated by a trained person who understands the operation of each tool in the factory assuring the quality of the manufactured product.

Virtual Reality is an experience which simulates situations from the real world or from the imagination. It requires the use of special glasses which shows the user the developed simulations through its screens. As technology advanced also controllers had been implemented in order to emulate the hands basic movements such as pointing, grabbing objects or clicking a button between other functionalities. Those helpful operations are responsible for a more immersive experience making as well possible for a laboratory or factory being emulated.

Since drones are an important asset due to its many useful applications, precision regarding its manufacture is required. Therefore, the machines implemented in the factory need to be used by capable hands which had previous experience with the manipulation of such technological tools. That experience doesn't necessarily require a person's first contact with the machine since the training to understand the machinery may be implemented via simulation and taking advantage of the new technologies such as virtual reality.

II. Universal Robot 5 e-Series Kinematics

Kinematics is the study of the motion of bodies or systems of bodies without the consideration of forces involved, such as causes and effects of the motions or torques. In robotics, two types of kinematics are used in order to know the end-effector location in the coordinate space (forward kinematics) or the joint angles required to move the end-effector to a specific location (inverse kinematics).

The Universal Robot 5 e-Series (UR5e) is a 6 degrees-of-freedom (DOF) robot which can rotate 360º in all of its joints. In order to simulate the UR5e in a virtual reality environment, both forward and inverse kinematics need to be considered for the correct functionality in the simulation. The kinematics of the robot was calculated using Matlab.

A. Forward Kinematics

Forward kinematics refers to the use of kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters. It is defined as:

$$\xi_n = \kappa(q), \quad q\{q_i, i \in [1,2,\dots,n]\} \tag{1}$$

To find the pose ξ_n given joint position q_i.

The Denavit-Hartenberg (DH) representation specifies 4 link parameters which are the angle (θ) between 2 links, the distance (offset) between links (d), the length of the link along the common normal (l) and the twist angle between axes (α). The DH values for the UR5e is the following.

TABLE I. UR5e Denavit Hartenberg

Link	θ	d [mm]	l [mm]	α
1	θ_1	0	162.411	0
2	θ_2	-422.925	0	-π/2
3	θ_3	-393.575	0	0
4	θ_4	0	133.336	- π/2
5	θ_5	0	98.501	0
6	θ_6	0	99.396	0

The values of the distances and offsets were calculated through the computer-aided design (CAD) model of the robot shown in Fig. 2.1.

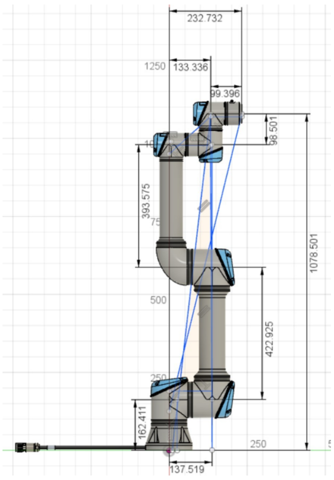

Fig. 2.1. UR5e measurements.

With the DH table, the forward kinematics were calculated using the transformation matrix between link and link as follows.

$$A_n = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\cos\alpha_n & l_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & l_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using Matlab computation, specifically Robotics Toolbox using the commands *teach()* for the graphical user interface (GUI) that shows the position and rotation of the end effector and the angles for each joint and *fkine* to obtain the forward kinematics given an angle-based starting position. The model for the robot is as seen in Figure 2.2.

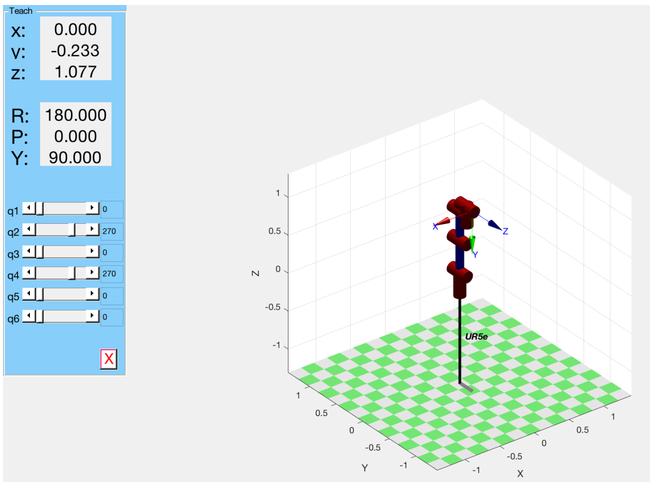

Fig. 2.2. Matlab simulation for UR5e.

In order to be certain that the model was correct, a comparison was made with the actual robot's teach pendant, as shown in Figure 2.3.

Fig. 2.3. UR5e teach pendant.

As it can be seen, the teach pendant shows an offset in the Y axis of 30 mm, which is known by fabrication. There is a slight error in both X and Z positions of 0.69 mm and 3 mm respectively which were taken as software offsets.

B. Inverse Kinematics

Inverse kinematics refers to the use of kinematic equations of a robot to compute the joint angles needed to move the end-effector location to a certain position. It is defined as:

$$q = \kappa^{-1}(\xi_n), \quad q\{q_i, i \in [1,2,\dots,n]\} \tag{2}$$

To find the joints q_i given a pose ξ_n.

In order to know the angles of each joint, the Robotics Toolbox for Matlab was again used, this time applying the command *ikine* to obtain the inverse kinematics given a position or trajectory to follow.

The use of Matlab for the completion of the project is a way to calculate all joint positions and angle values correctly in order to simulate them efficiently in Unity.

III. Machinery Simulations

Regarding the factory's specifications: three types of robots, CNC mills and conveyor models were required to be implemented and also programmed according to their real counterparts movements and limits. First of all, the UR5 robot model and basic mobilities were implemented previously so its kinematics were calculated according to the previous section. Nonetheless, the KUKA and FANUC robots were just models requiring a program and the CNC mills and conveyor models were required to be implemented.

A. KUKA Robot

The KUKA model was separated in order to distinguish its joints which are six corresponding to the six degrees of freedom of the robot. Also the teach pendant was replicated but its functionality was just the buttons of the movement for each joint. The movements were limited as the datasheet of the KUKA robot indicated. Figure 3.1 demonstrates the implemented model with its controller as seen on the Unity software.

All the coding was made using C# language and the libraries given by Unity. As means of design for the simulation the controller required to be suspended on air while it was tossed or released from the grab. The model as well required to have pivots in each joint in order to rotate them without losing the figure.

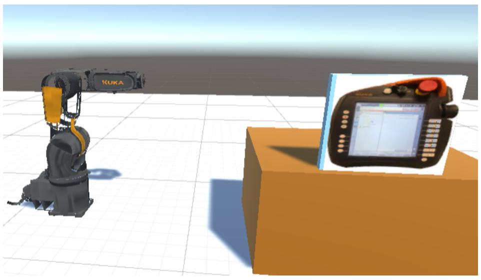

Figure 3.1. KUKA Robot

B. *FANUC Robot*

As the model KUKA, FANUC model had its own model already created. Nevertheless, its controller and motion was yet to be implemented. So the same process was required. Firstly the separation of the model in order to identify each joint separately and place the pivot on its center. Next, the reassemble of the FANUC robot and the construction of its controller. Finally, the program of the joint rotations which used primarily the Transform properties of Unity in order to gain access to the joint's angles. Furthermore, each button in the teach pendant was associated with the robot's six joints being twelve buttons as the movements required left and right or up and down movements. The figure 3.2 illustrates the FANUC model as shown in Unity software.

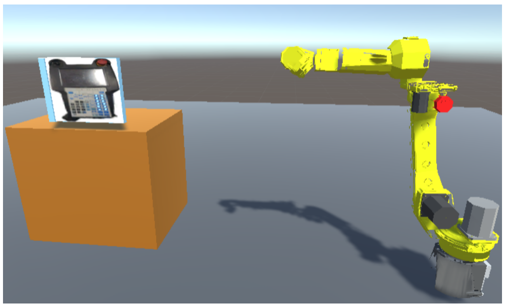

Figure 3.2. FANUC Robot.

C. *CNC mills*

The CNC mill was modeled after Haas' Mini Mill CNC. After retrieving the model for the page a problem was presented because it required to be separated by parts for the desired purposes. The parts separated consisted of the door of the CNC, the X axis part, the Y axis part, the Z axis part, the screen and controller and finally, the enclosure of the Mini Mill model. Once achieved the correct dissection, the controller model was implemented to the screen and controller part of the machinery using buttons as the user interface part in order to control the machine as the manual configuration is done in the real world.

The configuration for manual manipulation is pressing the Handle Jog button in order to unlock the Jog's movement, assigning the speed of movement for the desired edge. Four speeds are eligible for the Mini Mill being the speed one the slowest and the speed four the fastest. Once a speed is selected, it is required to press a button between Z edge, Y edge and X edge in order to manipulate one edge. Once achieved the jog is moved to the right for downward movement or left for upward on the Z edge, right for left movement and left for right movement on the X edge and right to move the axis to the front or left to move back the Y axis.

Also, since the grab was a problem for the door to be opened, a button was created on the controller to open or close the door of the Mini Mill being the fastest solution and the most useful. The Mini Mill model and controller may be found on Figure 3.3.

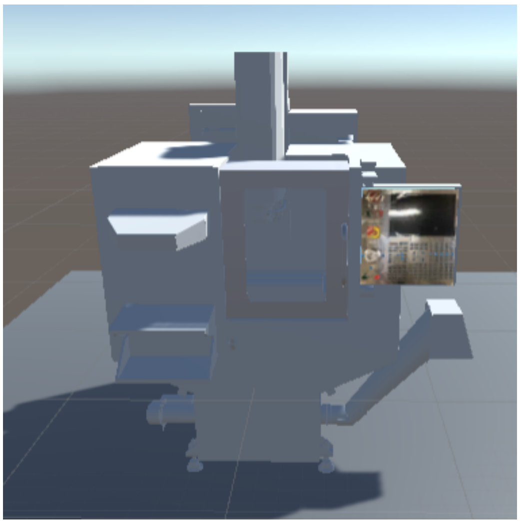

Figure 3.3. CNC Mini Mill Model

D. *Conveyor*

The conveyor model was made from scratch by using the Unity basic tools for object creation. The model consisted of a table hollow from inside and a piece that served as the moving part which traverses from one side to another continuously if an object is placed over the piece. The movement programmed was just a linear move in X then after achieving the goal of getting to the other side of the conveyor, the piece is lifted and placed on the return side of the conveyor emulating an invisible piston that pushes the object. The conveyor model is shown with all its components on figure 3.4 being the part colored in black the moving piece and the white cube the object.

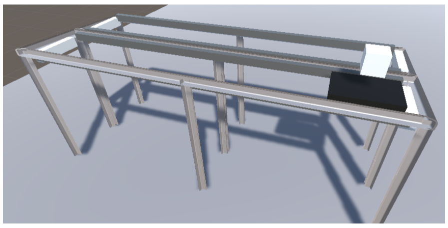

Figure 3.4. Conveyor

IV. Unity Connection with URSIM

URSim is a simulation software intended for offline programming and simulation of both robot programs and manual movement of robots. When Simulation mode is selected, it is possible to simulate digital inputs on the I/O page. The programs created for the movement of the robot are handled by the simulation software.

This simulation software creates a virtual UR Controller. The Real-Time Data Exchange (RTDE) interface provides a way to synchronize external applications with the UR controller over a standard TCP/IP connection, without breaking any real-time properties of the UR controller. The provided data is representing the robot's state, such as positions, temperatures, etc. through a few server sockets in the controller. A custom program was written to read these streams.

The data is interpreted by a socket created in the Unity environment. The socket contained the IP of the VM running the URSim, the RTDE port configuration and the transmission speed. The result was a socket receiving the data provided by the RTDE as shown in Figure 4.1.

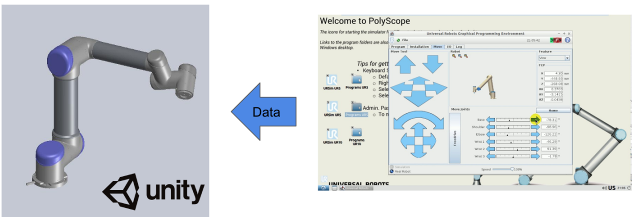

Fig. 4.1. Conceptual communication between URSIM and Unity

With the data in the Unity Environment and a model of the UR5e a simulation of the Drone Factory is possible. Adjustments of the interpretation of the data were necessary by the conversion of units. The model provided was simulated in the Zero State position taking care of the rotation of each joint and if necessary adjusting them in the robot controller script.

The end result is a robot in the Drone Factory that is controlled by the URSIM as shown in Figures 4.2, 4.3, 4.4.

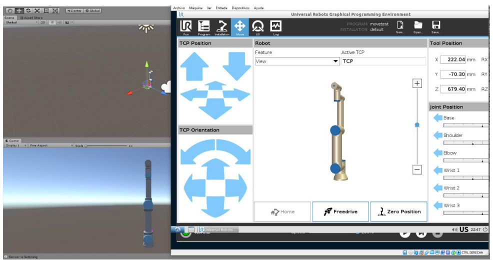

Fig. 4.2. Picture shows the Zero position send by URSIM

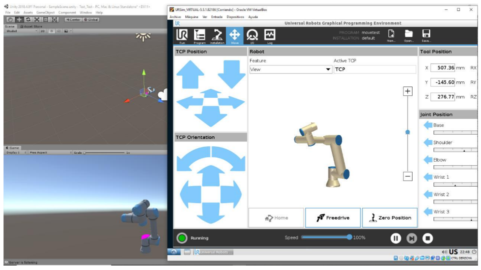

Fig. 4.3. Picture shows the First program position send by URSIM

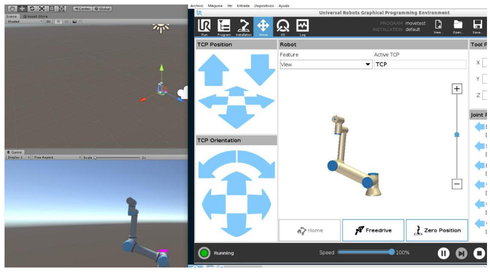

Fig. 4.4. Picture shows the Second program position send by URSIM.

A. Annex 1

The following table shows the activities done during “Semana i” showing the day of termination of each activity.

TABLE I. Table Type Styles

Activities done by the Week					
Activities	***Mon***	***Tue***	***Wed***	***Thu***	***Fri***
UR5e Denavit-Hartenberg table	✓				
UR5e measurements		✓			
UR5e forward kinematics modelation in Matlab			✓		
UR5e inverse kinematics modelation in Matlab				✓	
Kuka Denavit-Hartenberg table					✓
Kuka forward kinematics modelation in Matlab					✓
URSIM connection with Unity			✓		
Unity Prefab for UR5e ZeroPosition				✓	
Unity Script for URSIM data interpretation					✓
Kuka model assemble and simulation	✓				
Fanuc model assemble and simulation		✓			
CNC assemble			✓		
CNC error testing and simulation				✓	
Conveyor assemble and simulation					✓

Conclusion

Virtual Reality is a useful tool to create simulations of the real world in a secure way without the problems of breaking something or creating a harmful code that can break the machines used in the drone Factory.

Including the main tool known as Unity there are lots of knowledge surrounding it to make a big project like this work. Robotics, modeling, programming and networking were some of the topics used during these weeks.

The Drone Factory is an ongoing project that is searching to create a safe place for the students to learn how to use the heavy duty equipment.

References

[1] C. Herman (2019). Kinematics. 20-02-2019, de Tecnológico de Monterrey.

[2] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[3] "KR AGILUS | KUKA AG", *KUKA AG*, 2019. [Online]. Available: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/kr-agilus. [Accessed: 01- Nov- 2019].

[4] "Mini Mill-EDU", Haascnc.com, 2019. [Online]. Available: https://www.haascnc.com/es/machines/vertical-mills/mini-mills/models/minimill-edu.html. [Accessed: 01- Nov- 2019].

[5] Universal Robots Support. (2019). Overview of client interfaces. 10-01-2019, de Universal Robots.

[6] Universal Robots Support. (2019). Remote Control Via TCP/IP. 11-01-2019, de Universal Robots.