Desarrollo de una interfaz natural de usuario para el control de movimiento de un brazo robótico industrial usando visión por computadora

Erick Leandro Linares Rodriguez

Trabajo de grado para optar el título de ingeniero mecatrónico

Directora

Deisy Carolina Páez Casas

Magíster en Ingeniería Electrónica

Codirector

Luis Fernando Perico Remolina

Magíster en Controles industriales

Universidad Santo Tomás, Bucaramanga

División de Ingenierías y Arquitectura

Ingeniería mecatrónica

2023

Dedicatoria

Quiero dedicar este logro a mi madre, y agradecerle por apoyarme en todo mi proceso de formación profesional y personal, por ser mi fortaleza en los momentos de adversidad e inspirarme a buscar constantemente la mejor versión de mí mismo. Gracias por todo tu esfuerzo en sacarme adelante, por tus consejos, por tu paciencia, por todo tu infinito amor, sin ti nada de esto sería posible.

Contenido

Lista de tablas

Lista de figuras

Resumen

En el proyecto actual, se buscó el diseño y la construcción de una interfaz de usuario natural para controlar un brazo robótico. Con este propósito, y debido a la naturaleza del proyecto, se propuso una metodología secuencial, que se enfoca primero en el desarrollo de las etapas o aspectos más importantes del sistema, para luego condensar todo el trabajo en un resultado final. En la etapa inicial, se utilizó una cámara 3D comercial y se desarrolló un sistema de visión por computadora que reconoce los gestos de una persona a través de técnicas de seguimiento del esqueleto. Posteriormente, la información se codificó en instrucciones numéricas y se estableció una comunicación a través de sockets con un sistema de control robótico que interpreta los datos y mueve el efector final de un brazo industrial. En la etapa final, se realizaron pruebas de simulación y experimentos físicos, lo que resultó en la implementación de mejoras en el sistema, como el filtrado de señales o la implementación de una interfaz gráfica.

Abstract

In the present project, the design and construction of a natural user interface were sought to control a robotic arm. For this purpose, and due to the nature of the project, a sequential methodology was proposed, which focuses firstly on the development of the most important stages or aspects of the system, and then condenses all the work into a final result. In the initial stage, a commercial 3D camera was used, and a computer vision system was developed that recognizes gestures of a person through skeleton tracking techniques. Subsequently, the information was encoded into numerical instructions, and communication was established through sockets with a robotic control system that interprets the data and moves the end effector of an industrial arm. In the final stage, simulation tests and physical experiments were conducted, resulting in the implementation of system improvements, such as signal filtering or the implementation of a graphical interface.

Introducción

El controlar las máquinas o la tecnología que inventan los ingenieros para facilitar nuestras vidas es una ardua labor que requiere de mucho tiempo y esfuerzo. A lo largo de los años se ha podido mejorar la manera en la que lo hacemos, hemos pasado de controles netamente manuales, como botones y palancas directamente en las máquinas a controles desde la distancia por medio de conexiones alámbricas y paneles de control que permitían mayor seguridad y facilidad de uso, al tratar de mejorar esta relación de control y comodidad han surgido tecnologías de control inalámbricas como el bluetooth o el wifi, y los avances tecnológicos en sensores nos permiten controlar nuestros sistemas por medio de voz, de sonido, de infrarrojos e incluso por medio de un simple gesto con las manos. Es por esto que surge la idea de desarrollar una interfaz natural de usuario para el control de un brazo robótico por medio de gestos con nuestra mano, y para lograrlo se pretende establecer una conexión entre un dispositivo de reconocimiento visual comercial usado para juegos de la empresa Microsoft denominado Kinect, y un brazo robótico industrial de la empresa ABB con el que se cuenta en el campus universitario denominado IRB120, haciendo análisis cinemáticos y relacionando de manera correcta el posicionamiento del robot con las dimensiones y posición de nuestras manos.

1. Formulación del proyecto

1.1 Planteamiento del problema

A lo largo de los años y a través de muchos avances tecnológicos fue posible innovar en la manera como controlamos los sistemas tecnológicos que creamos, generando diversas técnicas de control que dependen del tipo de adquisición de datos y su procesamiento pero que también se mezclan con otras ramas de la ciencia como la visión por computadora, o la inteligencia artificial, haciendo que surjan interrogantes como ¿Cuál es el mejor método de control? ¿Es mejor un método u otro dependiendo el tipo de sistema que queremos controlar? Lo que nos lleva a nuestra pregunta de investigación: *¿Es el control visual un método eficiente de control para un robot?* Pero además también surgen preguntas derivadas de esta como ¿Es posible mejorar el control visual por medio de inteligencia artificial? ¿Qué tan lejos puede llevarse el control visual en el control de sistemas mecatrónicos complejos? Sin importar el tipo de control que apliquemos al brazo robótico, el poder controlar y programar una máquina industrial de este tipo no es una tarea sencilla y si además el control quiere realizarse a partir de una conexión a un sistema que no es del área de la robótica es un desafío bastante complejo, por lo que debemos abordar el problema por partes, y trabajar para optimizar cada sistema para la posterior conexión entre ellos, realizando un procesamiento de datos adecuado y acorde con un tipo de formato que podamos traducir fácilmente a un sistema de dimensiones y posiciones que el robot pueda comprender en su análisis cinemático.

Para nuestro proyecto debemos analizar los datos tomados por un Kinect de Microsoft que es usado normalmente para videojuegos, pero esta vez lo usaremos para conocer la posición tridimensional de nuestro cuerpo a partir de sus sensores ópticos que nos brindan información de profundidad, procesaremos los datos por medio de librerías de visión artificial que nos permiten

no solo obtener valores numéricos a partir de las imágenes captadas sino también analizar estas para adquirir datos específicos que de otra forma sería imposible.

El campo de la inteligencia artificial y sus métodos como el aprendizaje automático o las redes neuronales han logrado avances increíbles en la sociedad actual, y la visión artificial ha sido ampliamente implementada en el campo de la robótica como, por ejemplo, haciendo que los robots móviles reconozcan su entorno y puedan moverse de forma inteligente.

El brazo robótico tiene integrado un sistema para realizar la programación que puede hacerse de diversas formas, incluyendo la lectura de archivos planos y ayudas de software para establecer las rutas de movimiento sin necesidad de realizar un análisis físico profundo, sin embargo, es necesario que hagamos un análisis cinemático del robot ya que facilitará la correcta interpretación de los datos obtenidos por el dispositivo de adquisición de datos visuales.

1.2 Justificación

Este proyecto trabaja el área de la robótica y la teoría de control, combinado con comunicaciones y programación orientada a objetos, por tanto, este proyecto requiere de elementos de diversas áreas del conocimiento, todas relacionadas con avances tecnológicos actuales y con la llamada cuarta revolución industrial, en la que diariamente muchos países incursionan, pero nuestro país está en un evidente atraso, lo que es causante de la dependencia tecnológica, y por tanto de muchos problemas de nuestra sociedad, como el exceso de importaciones o la pobreza. Este proyecto apunta hacia la innovación y el desarrollo tecnológico, aportando así en la disminución de la inmensa brecha tecnológica de nuestro país.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar y construir una interfaz natural de usuario usando un sensor de profundidad tridimensional para controlar un brazo robótico destinado a la manipulación y control de objetos en un área determinada.

1.3.2 Objetivos específicos

Desarrollar un algoritmo de detección de las características visuales necesarias para la interacción del robot con el humano usando software especializado gratuito.

Diseñar y construir un sistema servo-visual 3D de cámara fija para la manipulación de objetos.

Diseñar un entorno de interacción virtual que permita configurar y controlar el modo de operación del robot por medio de gestos.

Establecer una comunicación entre el sistema de control del robot (IRC5) en lenguaje Rapid y la interfaz natural de usuario desarrollada en lenguaje de código abierto.

Diseñar una capa de aplicación para el protocolo de comunicación que enlaza el IRC5 con la interfaz natural para la codificación y decodificación de datos.

2. Marco referencial

2.1 Marco teórico

2.1.1 Interfaz de usuario

Cualquier tipo de dispositivo o programa cumple una función para el ser humano, pero en la mayoría de los casos es necesario interactuar con estos para usarlos de forma adecuada.

La interfaz de usuario o UI por sus siglas en inglés para “User Interface” nos permite comunicarnos con los sistemas tecnológicos y darles instrucciones determinadas para que puedan funcionar como nosotros queramos.

Las interfaces nos permiten visualizar o enviar información importante para el funcionamiento del dispositivo o sistema, por lo tanto, sin una interfaz de usuario adecuada es muy difícil lograr usarlo de manera eficaz.

Por esta razón existen varios tipos de interfaces y estas se mantienen en constante mejoría, cambiando a medida que surgen avances tecnológicos que permitan generar mayor comodidad al usuario, haciéndose cada vez más intuitivas y amigables.

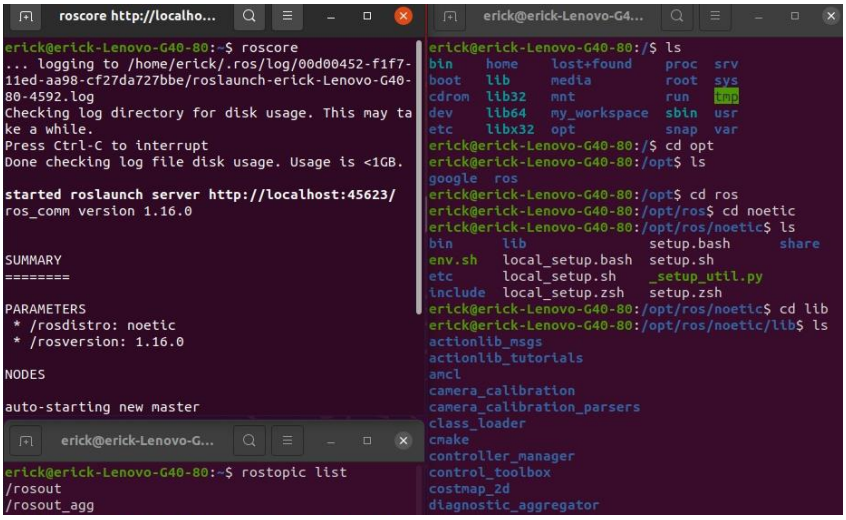
2.1.1.1 Interfaz de linea de comandos (CLI). Las interfaces de líneas de comandos o CLI por sus siglas en inglés para “Command Line Interface” hacen uso de comandos de texto simples para enviar instrucciones a través de un terminal.

Las CLI son las primeras interfaces que surgieron para hacer uso de las computadoras, sin embargo, son ampliamente usadas en la actualidad por la gran variedad de ventajas que pueden ofrecer.

A pesar de ser poco intuitivas, las CLI pueden ser mucho más rápidas y eficientes que otros tipos de interfaces de usuario, debido a que estas no requieren de navegación a través de menús o ventanas, y permiten acceder a opciones avanzadas con una sola linea de texto.

Las CLI son usadas en su mayoría por usuarios avanzados que tienen buen manejo de programación y que pueden usar las ventajas de esta interfaz a su favor optimizando su tiempo de trabajo y automatizando tareas, como es el caso de los usuarios de sistemas operativos basados en UNIX.

Figura 1. *Terminal de comandos en Ubuntu*



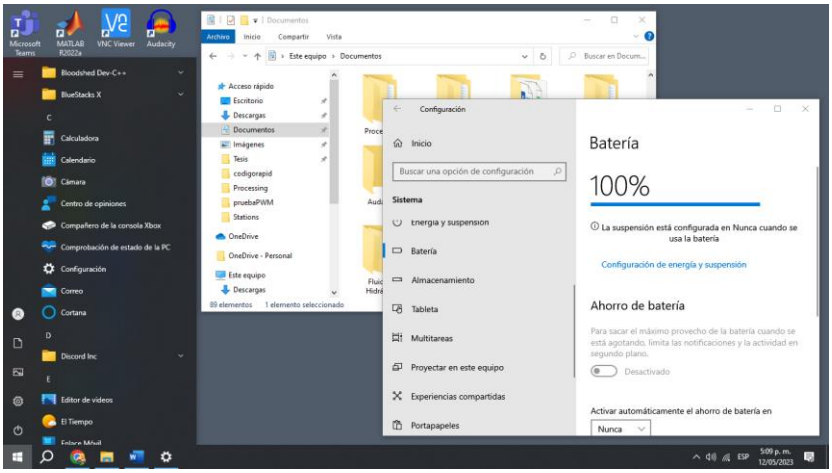
2.1.1.2 Interfaz gráfica de usuario (GUI). Las interfaces graficas de usuario o GUI (Graphical User Interface) son el tipo de interfaz más común en la actualidad debido a que son mucho más intuitivas que las CLI, permitiendo a usuarios menos avanzados introducirse al mundo de la computación y usarla como herramienta de trabajo e incluso personal.

Las GUI constan de elementos visuales que facilitan todos los procesos en los cuales interactúa el usuario y el sistema, como la navegación entre ficheros o el uso de herramientas para

fines específicos, permitiendo visualizar información importante a través de ventanas o alertas y enviar información usando botones, iconos, o controles deslizantes.

Esta accesibilidad de las GUI hace que los usuarios se adapten rápidamente a la interacción con el sistema, y su amplio uso hace que los usuarios puedan pasar de una interfaz aprendida a una nueva en muy poco tiempo, aprendiendo patrones de interacción como dar clic en una equis roja en la parte superior para cerrar el programa o el uso de pestañas en alguna barra lateral para navegar entre las opciones.

Figura 2. *Interfaz de usuario de Windows 10*



2.1.1.3 Interfaz natural de usuario (NUI). Las interfaces naturales de usuario o NUI (Natural User Interface) pretenden ser una evolución de las GUI en donde el usuario pueda interactuar con el sistema usando lenguaje natural, y de esta forma ser todavía más intuitivas y amigables con el usuario.

El lenguaje natural hace referencia a los gestos o acciones que usamos para comunicarnos con el mundo físico, como por ejemplo extender la palma de la mano para decir que queremos detener algo o subir el pulgar para decir que algo está bien.

Las interfaces naturales quieren facilitar el uso de la tecnología cambiando las líneas de comandos de las CLI o los menús despegables de las GUI por simples gestos o movimientos, eliminando los intermediarios como el ratón o el teclado.

Las NUI hacen uso de diferentes tecnologías para generar la interfaz, como el reconocimiento de voz, los sensores de movimiento o las pantallas táctiles, por lo que existen muchos ejemplos de NUI actualmente, y con los constantes avances tecnológicos el uso de estas se mantiene en constante crecimiento.

Figura 3. *Tecnología de control por movimiento*

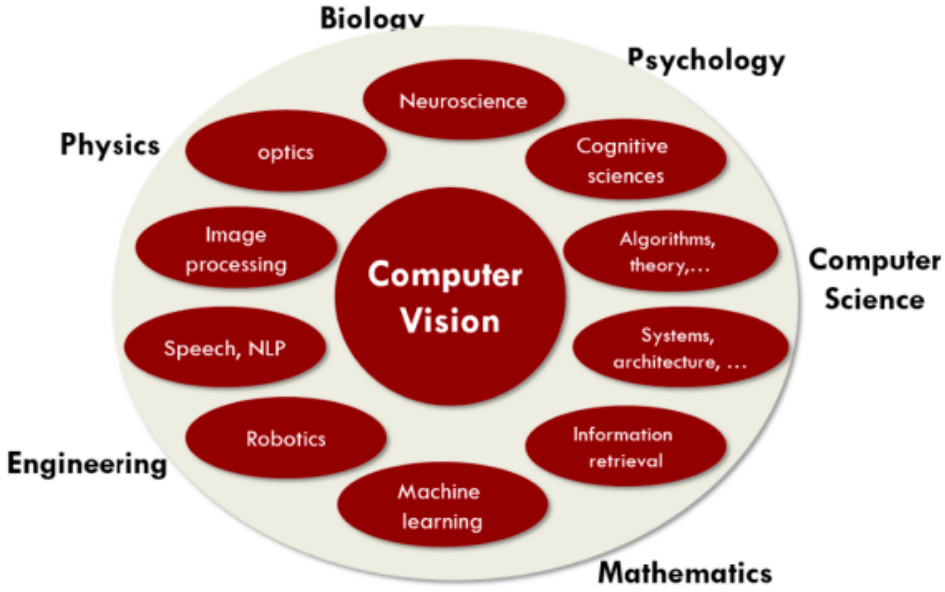
2.1.2 Visión artificial

La visión artificial o visión por computadora es una rama de la inteligencia artificial que se encarga de analizar y procesar imágenes digitales para adquirir datos e información relevante usando un computador.

La visión por computadora se basa en diversas técnicas, como el aprendizaje automático o la segmentación y manipulación de las propiedades de una imagen para reconocer objetos, seguirlos en un video o clasificarlos.

En la visión artificial se construyen algoritmos que permiten interpretar el contenido de las imágenes para usarlos posteriormente en diversas aplicaciones que se usan diariamente, desde cosas cotidianas como la búsqueda por imágenes de Google o la lectura de un código QR hasta usos más avanzados como la robótica, la medicina o la conducción autónoma.

Figura 4. *Visión artificial en múltiples campos*



Tomado de [1]

2.1.2.1 Procesamiento de imágenes digitales. Una imagen digital es un arreglo de pixeles que contienen diversa información, principalmente intensidad de color o luz. Para poder codificar el color se hace uso de modelos o esquemas de color, como por ejemplo RGB, CMYK o HSV, para este proyecto se usará el modelo RGB.

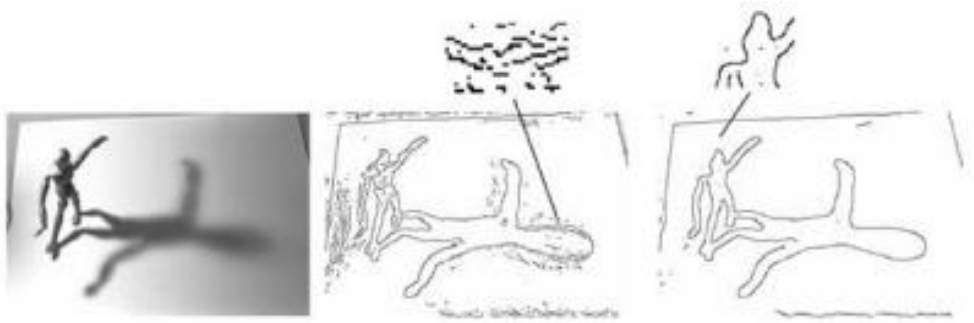
El modelo RGB (rojo, verde y azul) describe más de 16 millones de colores como la suma de rojo, verde y azul en diferentes intensidades, en un rango de valores de 0 a 255. Así pues, una imagen es un arreglo de pixeles, y cada píxel es un arreglo de 3 valores que van de 0 a 255.

Al recorrer uno a uno los pixeles de una imagen y extraer la información de estos, es posible aplicar diversas técnicas de procesamiento y filtros que nos permiten detectar elementos específicos de la imagen para su posterior uso en visión por computadora.

Algunas de las técnicas de procesamiento más comunes son:

2.1.2.2 Detección de bordes. En la detección de bordes se detectan las áreas de la imagen en donde hay un cambio brusco en la intensidad de los pixeles, es una de las técnicas de procesamiento más utilizadas debido a que gracias a ella es posible aplicar muchas otras más avanzadas como la detección y seguimiento de objetos.

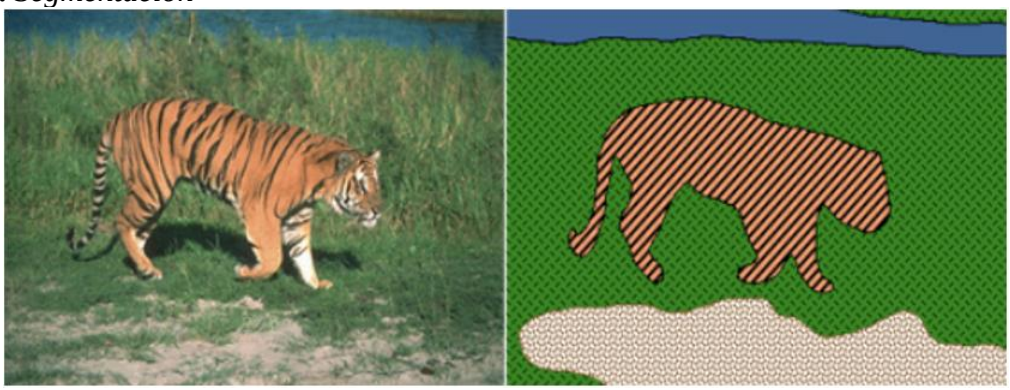
Figura 5. *Detección de bordes*



Tomado de [2]

2.1.2.3 Segmentación. La segmentación es el proceso por el cual se divide una imagen en regiones específicas para posteriormente ser clasificadas e interpretadas. Las regiones que son significativas dependen del problema que se esté intentando resolver, pero por lo general corresponden a objetos físicos de la imagen, por lo que se vuelve un problema de reconocimiento de patrones en los pixeles de la imagen, clasificándolos uno por uno y decidiendo si pertenecen a una región u otra.

Figura 6. *Segmentación*



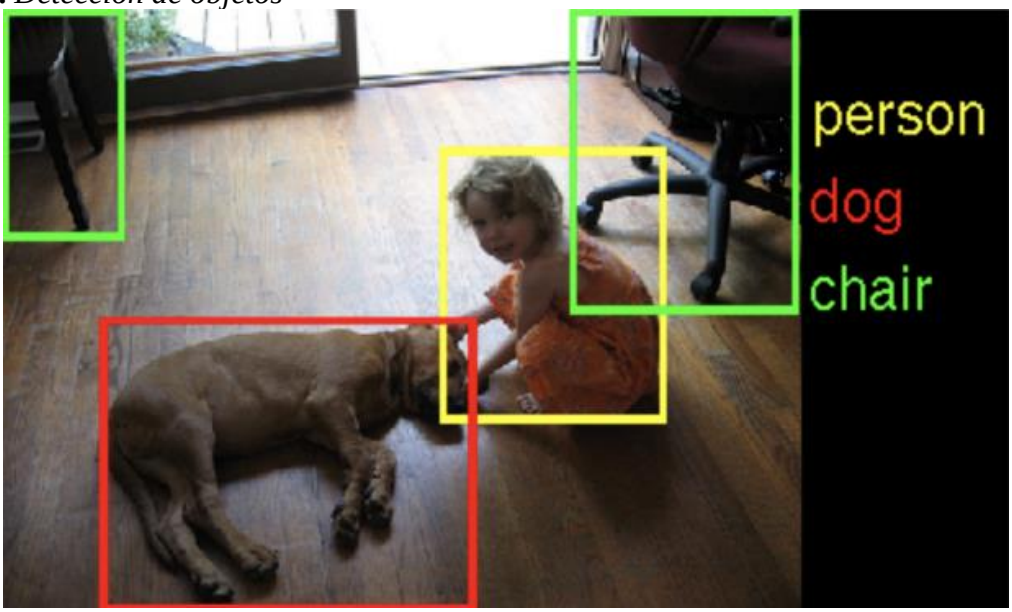
Tomado de [1]

2.1.2.4 Reconocimiento de objetos. El reconocimiento de objetos es un proceso avanzado de la visión artificial que une varias técnicas de procesamiento con el fin de detectar la presencia no sólo de objetos físicos, si no también animales, personas o cualquier cosa que pueda ser diferenciada de otra e identificada.

Al identificar el objeto, es posible obtener su ubicación y orientación, etiquetarlo, hacerle seguimiento, y muchas otras cosas, por lo que existen gran variedad de enfoques y aplicaciones para este proceso, como en seguridad, para detectar placas de autos y rostros de personas, o en fábricas como parte fundamental de un proceso.

Una de las formas más comunes de hacer detección de objetos es usando las técnicas de procesamiento mencionadas anteriormente, como la detección de bordes y esquinas o la segmentación, pero también existen técnicas de detección mucho más avanzadas que surgieron con los avances de la inteligencia artificial, como las redes neuronales y el deep learning.

Figura 7. *Detección de objetos*



Tomado de [1]

2.1.3 Fundamentos de robótica

La robótica es la rama de la ciencia e ingeniería que se centra en el desarrollo de los robots y sus muchas aplicaciones. La robótica es una ciencia muy amplia y se mantiene en constante evolución, por lo que para hablar robótica es conveniente referirse a algunos de sus tantos campos de acción, como la robótica industrial y los robots manipuladores.

Un robot puede definirse mecánicamente como una serie de eslabones unidos entre sí por articulaciones que permiten movimiento. Hay gran variedad de eslabones y articulaciones y de ellas dependen las acciones que pueden realizar los robots.

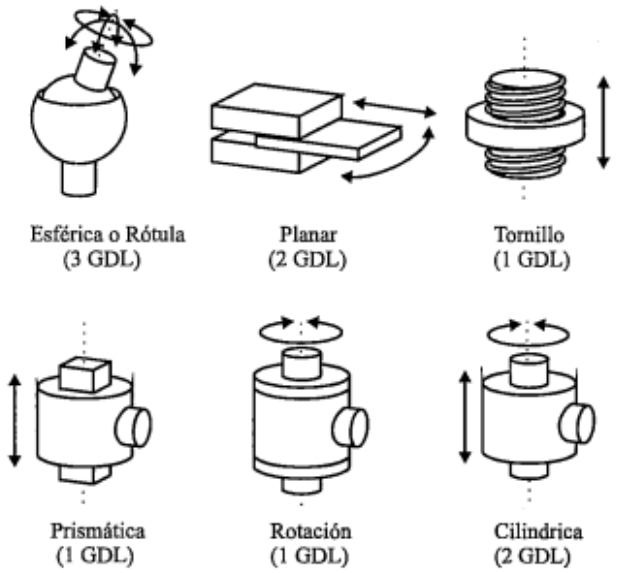
La mayoría de los robots manipuladores tienen similitud con la anatomía del brazo humano, por lo que es posible llamar a los robots manipuladores como brazos robóticos y referirse a sus partes como partes del brazo, como el codo o muñeca.

Para definir el movimiento del robot se habla de grados de libertad, que hacen referencia a la cantidad de movimientos independientes que puede realizar un robot. Por ejemplo, si un robot

puede moverse de un punto a otro en línea recta tiene un grado de libertad o eje (x), pero si puede moverse en un plano, quiere decir que puede moverse a lo largo de dos grados de libertad o dos ejes (x, y).

Los grados de libertad se definen según la cantidad y el tipo de articulaciones que tengan, estas pueden ser de desplazamiento lineal, rotacional o de una combinación de ambas, dando como resultado seis tipos de articulación: Esférica, planar, tornillo, prismática, rotación y cilíndrica, sin embargo, en los brazos robóticos se suelen usar sólo dos tipos de articulación, la prismática, y la rotacional, que sirven para el desplazamiento lineal y la rotación, respectivamente.

Figura 8. *Tipos de articulaciones*



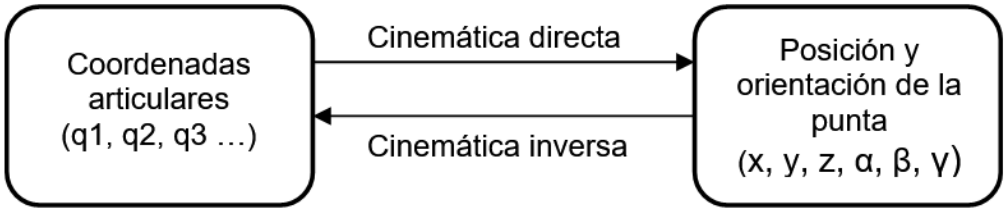
Tomado de [3]

Debido a que sólo se suelen usar las articulaciones prismáticas y rotacionales y estas sólo cuentan con un grado de libertad, la cantidad de grados de libertad de un robot suele coincidir con el número de articulaciones que tiene.

2.1.3.1 Cinemática de robots manipuladores. La cinemática estudia el movimiento del robot con respecto a su base o sistema de referencia, más específicamente se encarga de encontrar la relación entre la posición y orientación del efector final del robot (x, y, z, α, β, γ) y sus valores articulares (q1, q2, q3 …).

Existen dos problemas fundamentales que resolver a la hora de describir la cinemática de un robot, el problema cinemático directo y el problema cinemático inverso. El primero se trata de encontrar la posición y orientación del efector final en función de sus valores articulares y el segundo de encontrar los valores articulares en función de la posición y orientación efector final.

Figura 9. *Esquema de cinemática inversa y directa*



Adaptado de [3]

2.1.3.2 Localización espacial. Para poder manipular la posición y orientación de un objeto en un espacio de tres dimensiones es necesario hacer uso de matrices de transformación homogéneas (MTH), las cuales son matrices de 4x4 que describen las transformaciones que sufre un sistema de coordenadas para convertirse en otro.

En la robótica se usan las MTH debido a que permiten representar de manera conjunta las matrices que describen sus articulaciones, ya sean prismáticas o rotacionales, en una única matriz que relaciona el sistema de coordenadas de la punta y el de la base, facilitando el proceso de resolver el problema cinemático de un robot.

Figura 10. *Matriz de transformación homogénea*

$$\mathbf{T}=\begin{bmatrix}\mathbf{R}_{3x3} & \mathbf{p}_{3x1}\\ 0 & 1\end{bmatrix}=\begin{bmatrix}\text{Rotación} & \text{Traslación}\\ 0 & 1\end{bmatrix} \qquad \mathbf{T}=\begin{bmatrix}n_x & o_x & a_x & p_x\\ n_y & o_y & a_y & p_y\\ n_z & o_z & a_z & p_z\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Adaptado de [3]

2.1.3.3 Algoritmo Denavit – Hartenberg. Para poder definir la cinemática directa de un robot existen diversos métodos que facilitan el proceso, este es el caso del algoritmo Denavit-Hartenberg (D-H), el cual es un método matricial que permite establecer una MTH para cada articulación y posteriormente calcular una MTH que represente el sistema en su totalidad, desde el efector final hasta la base del robot, resolviendo así la cinemática directa.

El algoritmo D-H se basa en cuatro trasformaciones básicas que dependen de la geometría del eslabón, y que ayudarán a definir todo el robot si se escogen adecuadamente los marcos de referencia, con ayuda del algoritmo.

El algoritmo D-H consta de 16 pasos que describen el proceso siguiente:

1. Escoger adecuadamente los marcos de referencia de cada eslabón.
2. Calcular las 4 transformaciones básicas.
3. Obtener las matrices de transformación para cada articulación.
4. Obtener la matriz total del sistema.

Las cuatro transformaciones básicas permiten relacionar el marco de referencia de un elemento i con el marco de un elemento i-1, estas son:

1. Rotación alrededor del eje Z_{i-1} un ángulo θ_i.
2. Traslación a lo largo del eje Z_{i-1} una distancia d_i.
3. Traslación a lo largo de X_i una distancia a_i.

4. Rotación a lo largo del eje X_i un ángulo α_i .

Estas transformaciones deben realizarse en este orden específico debido a que el producto entre matrices no es conmutativo. Al realizar la multiplicación de las matrices se puede obtener una matriz formula que defina el marco de referencia A_i de cada articulación como se muestra en la figura 11.

Figura 11. *MTH Denavit-Hartenberg*

$$
{}^{i-1}\mathbf{A}_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =
$$

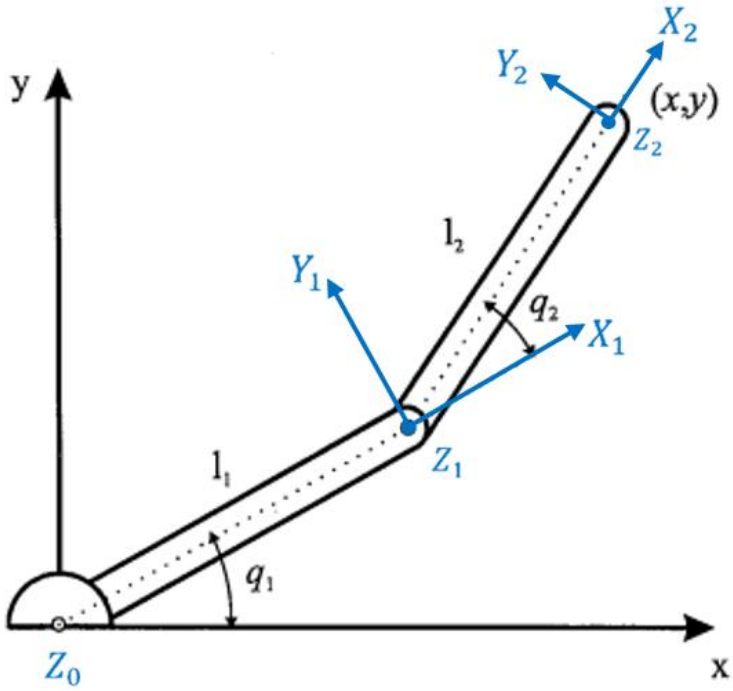
$$
= \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Tomado de [3]

2.1.3.4 Cinemática directa. Se propone entonces resolver el problema de la cinemática directa para un robot de 2 grados de libertad de tipo planar RR, aplicando el algoritmo **Denavit-**Hartenberg y obteniendo la matriz de transformación homogénea total que relaciona el efector final (x, y) y los valores de sus articulaciones (q1, q2).

Primeramente, se definen los marcos de referencia de cada articulación de acuerdo con el algoritmo, dando como resultado lo que se muestra en la figura 12.

Figura 12. *D-H para robot planar 2GDL RR*



Adaptado de [3]

Habiendo ubicado los marcos de referencia de las articulaciones, se procede a calcular para cada marco los 4 parámetros básicos en los que se basa el algoritmo D-H y se registran en la tabla 1.

Tabla 1. *Parámetros D-H*

Marco	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
A_1^0	q_1	0	l_1	0
A_2^1	q_2	0	l_2	0

Teniendo los parámetros D-H se procede a reemplazarlos en la matriz fórmula para cada uno de los marcos y al multiplicar las dos matrices obtenemos la matriz total del sistema.

Figura 13. *MTH D-H para robot planar*

```
A_0_1 =                                  A_1_2 =

[cos(q1), -sin(q1), 0, l1*cos(q1)]   [cos(q2), -sin(q2), 0, l2*cos(q2)]
[sin(q1),  cos(q1), 0, l1*sin(q1)]   [sin(q2),  cos(q2), 0, l2*sin(q2)]
[      0,        0, 1,          0]   [      0,        0, 1,          0]
[      0,        0, 0,          1]   [      0,        0, 0,          1]

     A_0_2 =

     [cos(q1 + q2), -sin(q1 + q2), 0, l2*cos(q1 + q2) + l1*cos(q1)]
     [sin(q1 + q2),  cos(q1 + q2), 0, l2*sin(q1 + q2) + l1*sin(q1)]
     [           0,             0, 1,                           0]
     [           0,             0, 0,                           1]
```

Si recordamos que la cuarta columna de una MTH representa la posición x e y, en sus filas 1, 2, respectivamente tenemos que:

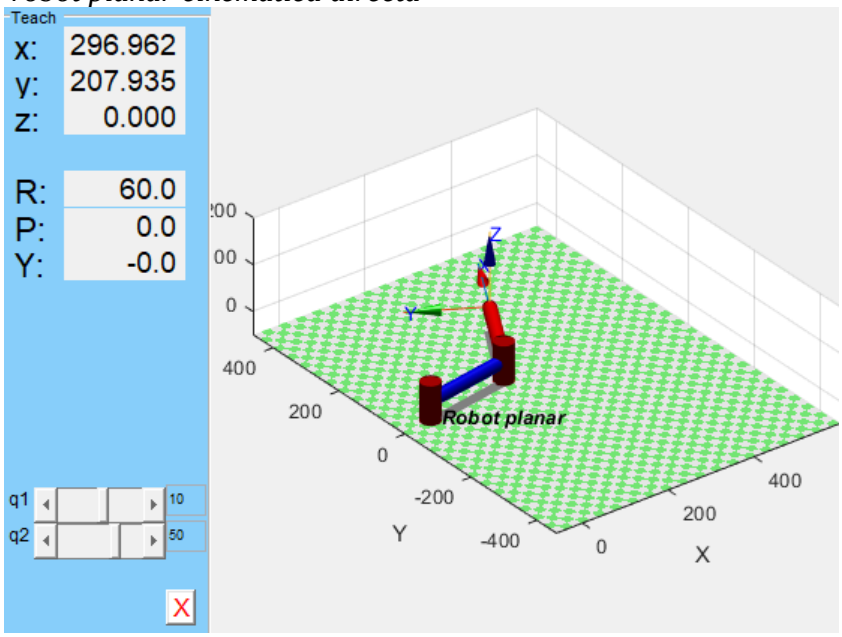
$$X = l_2 * Cos(q_1 + q_2) + l_1 * Cos(q_1) \quad (1)$$

$$Y = l_2 * Sin(q_1 + q_2) + l_1 * Sin(q_1) \quad (2)$$

Teniendo las ecuaciones (1) y (2) que relacionan la posición X e Y de la punta del robot con sus valores articulares q1 y q2 se resuelve el problema cinemático directo para este robot planar.

Podemos corroborar esto haciendo uso de la librería de MATLAB llamada Robotic Toolbox de Peter Corke, que permite realizar simulación de robots manipuladores ingresando parámetros los parámetros de D-H que fueron calculados en la tabla 1.

Figura 14. *RBT robot planar cinemática directa*



Para la simulación se usó una distancia para ambos eslabones de 200 [mm] y valores de 10 y 50 grados para q1 y q2 respectivamente, lo que nos da una posición en el efector final de 296.962, 207.935 y 0 mm para X, Y y Z. Si usamos las ecuaciones obtenidas de la MTH total tenemos que:

Figura 15. *Comprobación de simulación RBT*

```
>> l1 = 200; q1 = 10*pi/180;
   l2 = 200; q2 = 50*pi/180;
   x = l2*cos(q1 + q2) + l1*cos(q1)
   y = l2*sin(q1 + q2) + l1*sin(q1)

x =

  296.9616


y =

  207.9347
```

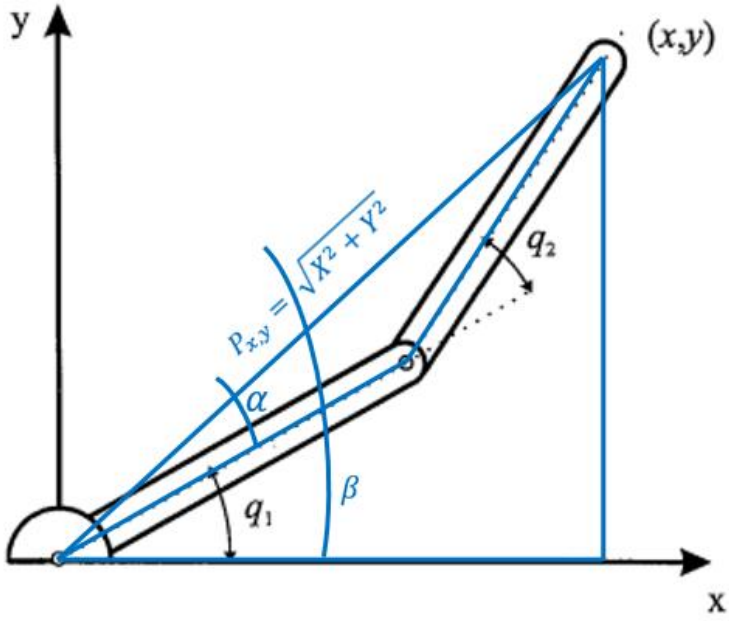
Corroborando así la veracidad de las ecuaciones cinemáticas obtenidas.

2.1.3.5 Cinemática inversa. La cinemática inversa de un robot se centra en encontrar el valor de sus articulaciones con respecto a la posición y orientación del efector final, por lo tanto, con la resolución de este problema será posible llevar la punta del robot a cualquier posición y orientación deseada.

El problema cinemático inverso depende de la configuración del robot, haciéndolo más complicado que el problema directo, sin embargo, existen diversos métodos que facilitan este proceso. Uno de los métodos más comunes para desarrollar la cinemática inversa de un robot manipulador es el método geométrico, en el cual se aplica trigonometría y álgebra para llegar a una ecuación para cada variable articular. Es necesario mencionar que en el caso de la cinemática inversa pueden existir múltiples resoluciones o configuraciones articulares para una posición y orientación específica.

Se propone entonces usar el método geométrico para resolver la cinemática inversa del robot planar que se planteó para problema directo en la figura 12.

Figura 16. Método geométrico robot planar



Adaptado de [3]

Primeramente, se dibujan dos triángulos que permitan relacionar las variables articulares con la punta del robot, los cuales se muestran en la figura 16. Se tiene entonces un triángulo recto en el que su hipotenusa coincide con la distancia desde la base a la punta del robot, y esta misma hipotenusa es uno de los lados del triángulo central cuyos catetos se forman con los eslabones del robot.

Debido a que se conocen los tres lados del triángulo central se puede hallar una expresión para el ángulo de q2 (4) aplicando la ley de cosenos (3).

$$P_{xy}{}^2 = l_1{}^2 + l_2{}^2 + 2l_1 l_2 Cos(q_2) \tag{3}$$

$$Cos(q_2) = \frac{X^2 + Y^2 - l_1{}^2 - l_2{}^2}{2l_1 l_2} \tag{4}$$

Para poder resolver esta ecuación es necesario aplicar arcoseno a ambos lados de la ecuación, sin embargo, al hacer esto podemos tener inconvenientes con la interpretación de los cuadrantes de la función, para esto existe una función matemática denominada “atan2” que resuelve los problemas de interpretación de cuadrantes. Por tanto, procedemos a simplificar la función para que quede en términos de tangente (5), y simplificamos q2 (6).

$$\mathrm{Tan}(q_2) = \frac{Sen(q_2)}{Cos(q_2)} \tag{5}$$

$$q_2 = Tan^{-1}\left(\frac{\pm\sqrt{1 - Cos^2(q_2)}}{Cos(q_2)}\right) \tag{6}$$

Para el ángulo de la primera articulación se debe calcular la diferencia entre los ángulos α y β, en ambos casos es posible aplicar tangente debido a que se conocen las distancias de sus catetos opuestos y adyacentes.

$$\beta = Tan^{-1}\left(\frac{Y}{X}\right) \tag{7}$$

$$\alpha = Tan^{-1}\left(\frac{l_2 Sen(q_2)}{l_1 + l_1 Cos(q_2)}\right) \tag{8}$$

$$q_1 = \beta - \alpha = Tan^{-1}\left(\frac{Y}{X}\right) - Tan^{-1}\left(\frac{l_2 Sen(q_2)}{l_1 + l_1 Cos(q_2)}\right) \tag{9}$$

Teniendo ecuaciones para ambas articulaciones con respecto a la posición final, se resuelve el problema cinemático inverso.

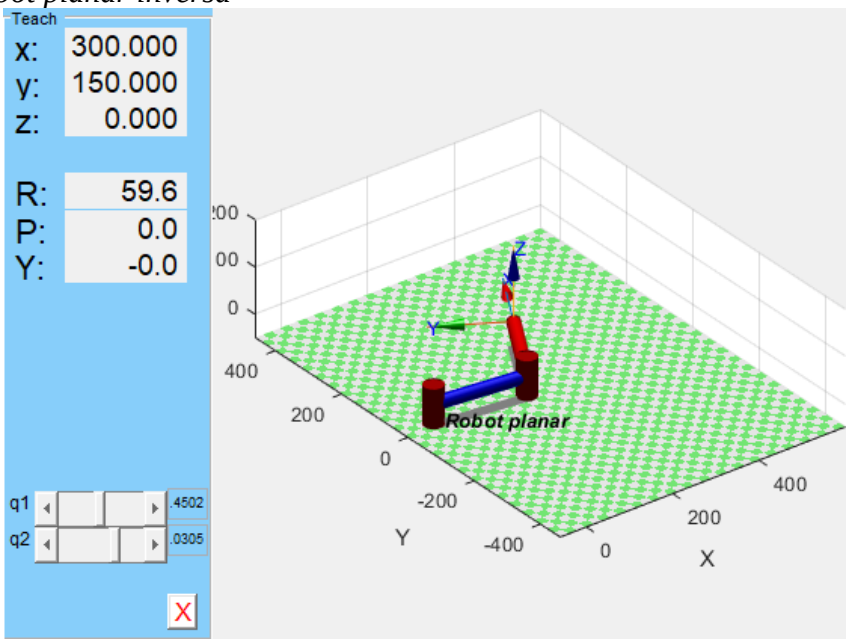
Procedemos a corroborar los resultados en Matlab para un valor X e Y de 300 y 150 mm.

Figura 17. *Cinemática inversa robot planar*

```
>> l1 = 200; l2 = 200;
   X = 300;   Y = 150;
   cosq2 = (X^2+Y^2-l1^2-l2^2)/(2*l1*l2);
   q2 = atan2(sqrt(1-cosq2^2),cosq2);
   q1 = atan2(Y,X)-atan2(l2*sin(q2),l1+l2*cos(q2));
   q1g=q1*180/pi
   q2g=q2*180/pi

q1g =

   -6.4502


q2g =

   66.0305
```

Si ingresamos esos valores de ángulos en las articulaciones usando el simulador que ofrece el Robotic Toolbox de Peter Corke, podemos comprobar la posición del extremo del robot.

Figura 18. *RBT robot planar inversa*



Corroborando así la veracidad de las ecuaciones cinemáticas obtenidas.

2.2 Marco tecnológico y científico

2.2.1 Visual Servoing

Una de las aplicaciones más importantes del procesamiento de imágenes y la visión artificial es el visual based robot control o visual servoing que como indica su nombre, se refiere al control de robots por medio de información visual.

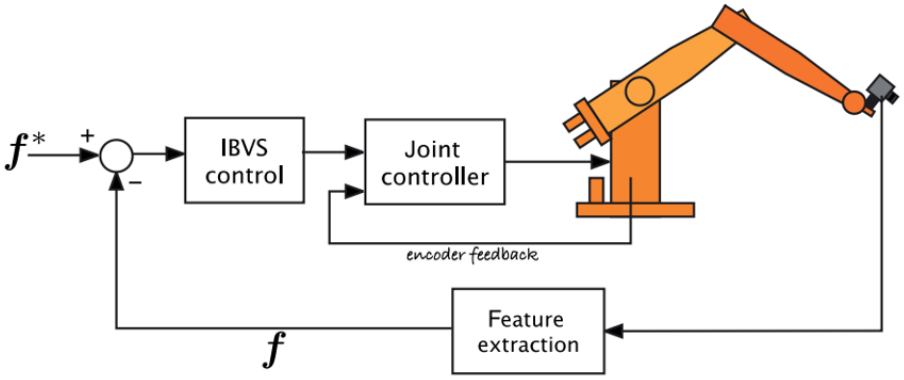
La información visual es muy importante para muchas de las tareas que realizamos los humanos, como la interacción con elementos del entorno, el control de máquinas, o el propio

desplazamiento, por lo que brindar esta información a los robots hace posible que realicen muchas de estas tareas sin necesidad de instrumentación avanzada o entornos prediseñados.

El visual servoing surgió desde los años setenta y desde entonces ha ido creciendo tanto como los avances tecnológicos se lo han permitido, por lo que tan pronto evolucionaron las técnicas de visión artificial y se pudo aplicar a la industria, surgieron muchos campos y enfoques para el visual servoing. El control visual para robots es entonces un campo muy grande de la robótica, pudiendo aplicarse no sólo a robots manipuladores, sino también a robots móviles, de servicio, drones y muchos otros, siempre y cuando se aplique la idea de controlar sus funciones por medio de información visual.

Existen principalmente dos tipos de visual servoing, el control visual basado en posición y basado en imagen y se diferencian principalmente por su costo computacional, debido a que en el primer método se determina la posición y orientación del objeto a controlar por medio de puntos de referencia o características concretas de la imagen y en el segundo método se analiza la imagen completa.

Figura 19. *Visual Servoing*



Tomado de [4]

2.2.2 Proyectos relacionados

En este proyecto se propone controlar el movimiento de un robot industrial usando visión por computadora por lo que se define como un proyecto de visual servoing. Teniendo claro el campo donde se encuentra el proyecto es posible compararlo con otros similares y revisar sus aportes.

Tanto en el marco nacional como global podemos encontrarnos con muchos proyectos y artículos que relacionan las áreas de visión por computadora y la robótica. A continuación, se hablará de algunos de estos proyectos y artículos que aportan a la construcción del marco tecnológico del actual proyecto.

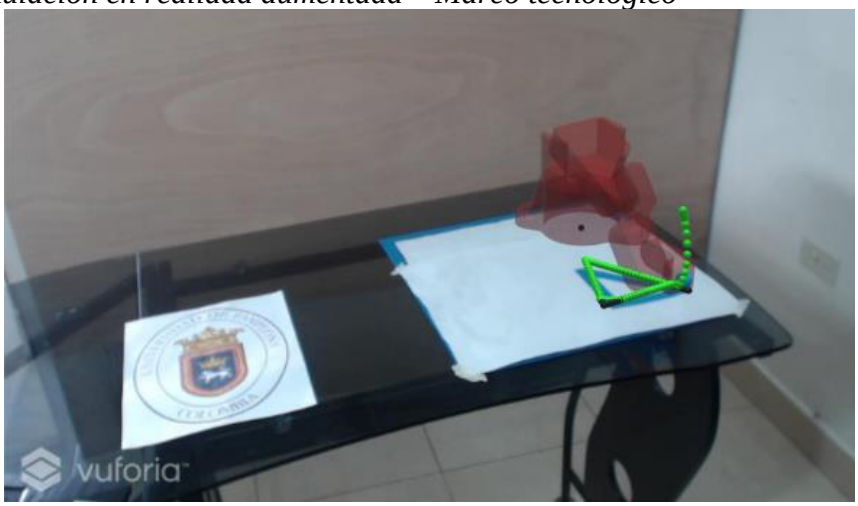
2.2.2.1 Marco nacional.

Tesis de maestría - Control por guiado gestual para un robot antropomórfico basado en técnicas de visión artificial

En este proyecto se desarrolló un control por gestos para un robot antropomórfico de la Universidad de Pamplona. Se usó un sensor Kinect para guardar comandos gestuales con las manos y establecer trayectorias en un área de trabajo específica que se enviaron a un robot simulado con realidad virtual.

Se realizaron varias pruebas aplicando diferentes tipos de filtros con el fin de mejorar el comportamiento del robot y se adjuntó evidencia de mejoría en su control de posición y velocidad, además se desarrolló una interfaz gráfica que permitía al usuario hacer la simulación de las trayectorias.

Figura 20. *Simulación en realidad aumentada – Marco tecnológico*



Tomado de [5]

A partir de este proyecto se lograron dos artículos que se publicaron en revistas categoría B y una ponencia en el congreso internacional de electrónica y tecnología de avanzada del año 2020.

Tesis de maestría - Arquitectura de telecontrol de un robot mediante el uso de interfaces gestuales

En este proyecto se diseñó un sistema de visión artificial para el control presencial y remoto mediante gestos de un robot humanoide. Se desarrolló una interfaz para tele operación del robot, de forma que se pudiera controlar desde cualquier parte del mundo usando gestos corporales.

Se usó un sensor Kinect para capturar los gestos y fueron enviados a través de socket-internet para controlar remotamente las articulaciones de un robot LEGO Mindstorms con forma humanoide al cual se le integró un celular para recibir los comandos a través del socket y enviarlos por bluetooth al controlador LEGO.

Figura 21. *Partes de la implementación del modelo - Marco tecnológico*

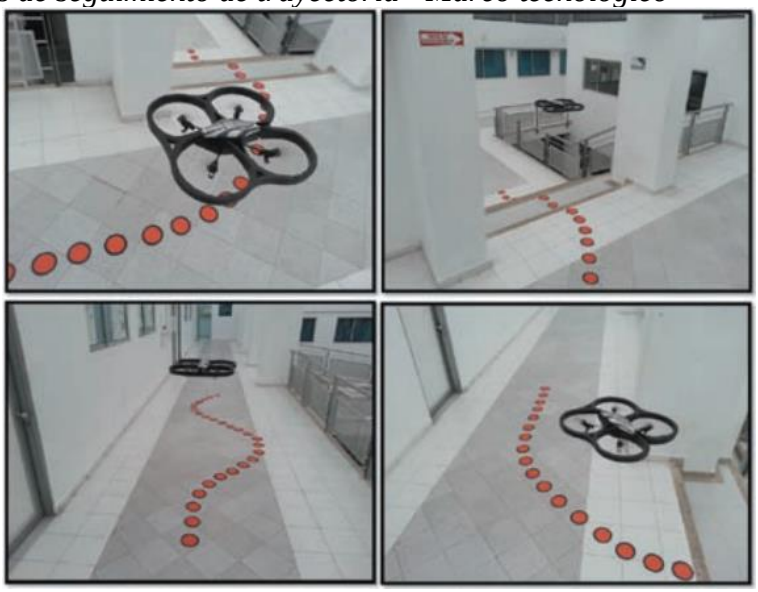
Tomado de [6]

Artículo - Sistema de inspección y vigilancia utilizando un robot aéreo guiado mediante visión artificial

En este artículo se desarrolló un sistema de inspección y vigilancia para la navegación autónoma de un robot aéreo integrado con dos cámaras. Se implementaron varias estrategias de visual servoing para controlar la posición y orientación y se realizó satisfactoriamente la inspección remota de un espacio.

Se implementaron varios tipos de controladores y se hicieron varias pruebas para desarrollar un algoritmo control de posición y orientación por medio de visión artificial, se desarrolló una interfaz de usuario para establecer el modo de navegación, la visualización de datos, la inicialización, entre otras cosas.

Figura 22. *Pruebas de seguimiento de trayectoria - Marco tecnológico*



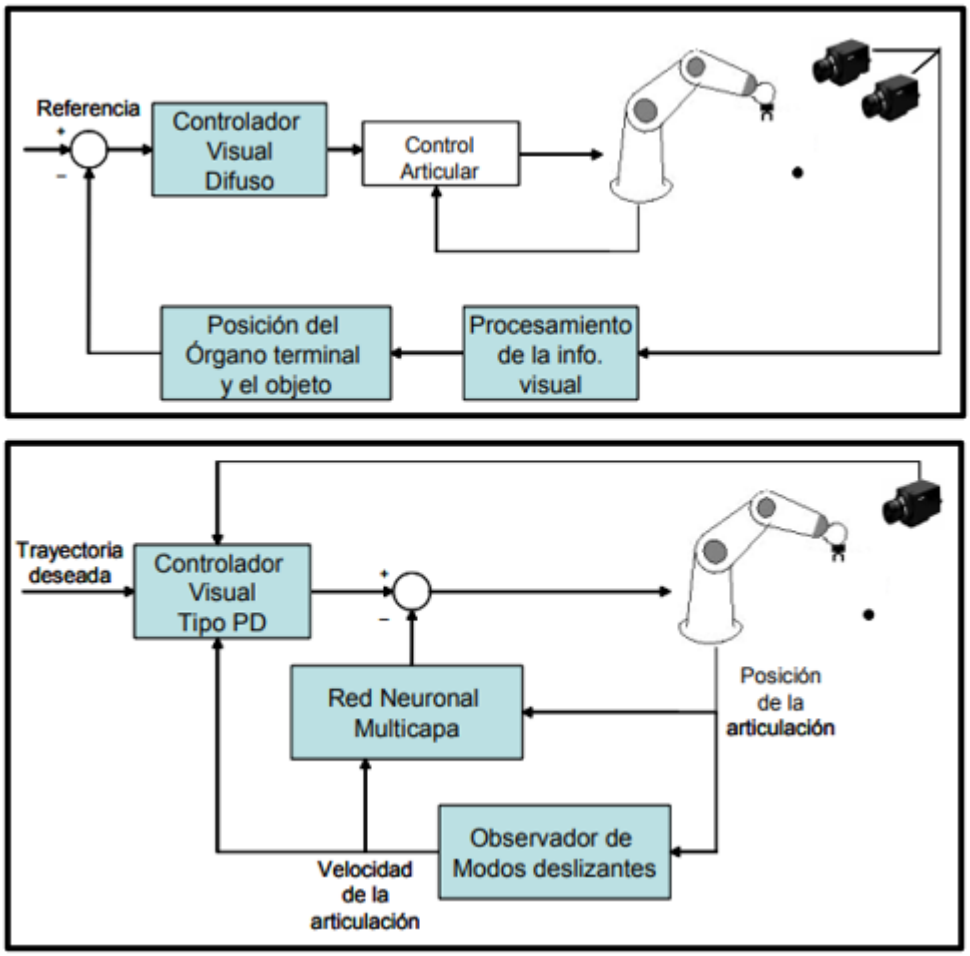
Tomado de [7]

2.2.2.2 Marco global

Tesis doctoral - Visión artificial estero con aplicación al control de un brazo de robot

En este proyecto de 3 años se desarrolló una plataforma de visión artificial para dos cámaras analógicas que trabajaron en conjunto en un sistema de estéreo visión para el control de un robot de 6 grados de libertad. Se desarrollaron varios algoritmos de visión artificial para la adquisición de datos que se usó en el modelo de estéreo visión. Se diseñó e implementó un modelo para calcular la posición tridimensional de un punto en el espacio. Se desarrollaron dos controladores, un controlador visual difuso y un controlador visual proporcional derivativo que implementa redes neuronales para el control en lazo cerrado de un robot de hasta 3 grados de libertad.

Figura 23. *Controladores desarrollados en el proyecto - Marco tecnológico*



Tomado de [8]

Articulo - Una plataforma de control basado en visión para la rehabilitación de robots manipuladores de tipo industrial

En este artículo se desarrolló una plataforma de control basado en visión por computadora para robots manipuladores industriales. Se propuso reemplazar los controladores de robots industriales usados para la realización de tareas complejas por medio de un control por visión artificial, alargando su vida útil y reduciendo costos.

Se usaron dos cámaras Sony para el control de un brazo robótico industrial de 6 grados de libertad en un área determinada, al robot se le sustituyó el sistema controlador original por una tarjeta controladora comercial que se programó a partir de algoritmos de visión artificial solucionando así problemas de control presentes debido al desgaste mecánico de sus piezas.

3. Metodología

En la figura 24 se presenta el diagrama metodológico del proyecto, el cual se dividió en varias etapas principales que se integran de forma secuencial debido a que todas las tareas fueron realizadas por una única persona y autor del proyecto.

En la primera etapa del proyecto se desarrolló el sistema de visión artificial, en el que se definió el modelo de adquisición y procesamiento de datos para el posterior envió de la información.

Posteriormente se configuró el entorno de programación y simulación del robot para poder realizar las pruebas específicas. Se estableció el sistema de comunicación que enlazaba el entorno de control del robot y el sistema de visión artificial, en el que se hizo una codificación de los datos a enviar para su posterior interpretación en el entorno de programación del robot.

Se realizaron pruebas físicas y en simulación que permitieron desarrollar mejoras en la comunicación como la aplicación de filtros o la delimitación del área de trabajo con el fin de evitar errores.

Se diseñó una interfaz gráfica y una interfaz natural de comandos gestuales para enviar datos concretos de posición y se desarrollaron modos de trabajo para el usuario.

Figura 24. *Diagrama metodológico del proyecto*

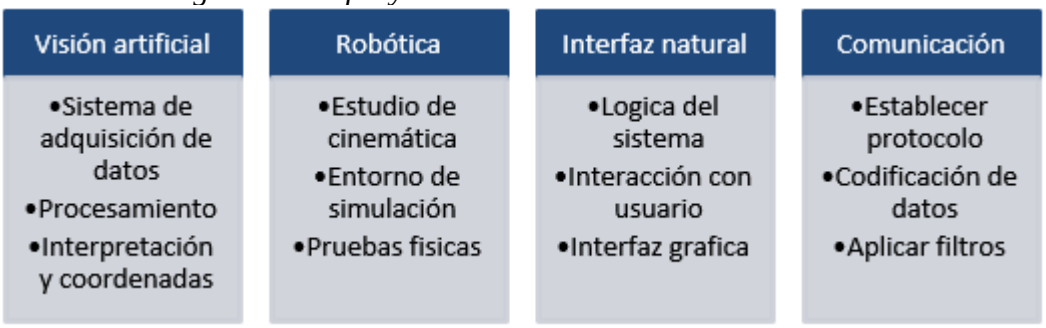
4. Desarrollo

4.1 Estructura del proyecto

Se estableció comunicación local entre un sistema de control robótico industrial y un sistema de visión por computadora para enviar datos de coordenadas 3D a partir de gestos y movimientos con los brazos, lo que permitió desarrollar una interfaz natural de usuario para el control servo visual de un brazo robótico industrial.

Este proyecto de visual servoing como otros del mismo campo de aplicación integra las ramas de visión artificial y la robótica, por lo que para hablar de su desarrollo es conveniente estructurar el proyecto en estas dos ramas, pero además de estas es necesario hablar de dos ramas más, el desarrollo de la interfaz de usuario (uno de los objetivos del proyecto) y la implementación de un protocolo para su comunicación.

Figura 25. *Estructura general del proyecto*



Tal como se indicó en la metodología y estructura, estas cuatro etapas en las que se dividió el proyecto se trabajaron de forma independiente y secuencial, conectando los conocimientos adquiridos para realizar los ajustes correspondientes paso tras paso.

Dicho lo anterior, se dividirá el desarrollo del proyecto en cuatro secciones independientes, en las que se explicarán todas las decisiones que se tomaron, los pasos que se siguieron y los problemas que se corrigieron para cumplir con los objetivos establecidos. Toda la programación del proyecto, tanto el sistema de visión por computadora como la del sistema de control del robot se encuentran resumidas en diagramas de flujo que explican el proceso a grandes rasgos, sin embargo, todos los códigos usados podrán encontrarse en los anexos del documento.

4.2 Visión por computadora

4.2.1 Kinect como dispositivo de adquisición de datos

El punto de partida del proyecto fue la captura de los movimientos o gestos con los manos, específicamente fue necesario obtener la posición tridimensional de la mano derecha para poder controlar la posición del efector final del brazo robótico industrial.

Para poder desarrollar un sistema de visión artificial tridimensional es necesario adquirir datos de profundidad en la imagen, existen técnicas como la estéreo visión en la que se usan dos cámaras con el fin de determinar el tercer eje coordenado en la posición de los objetos, sin embargo y debido a la complejidad que puede representar un sistema de este tipo, se optó por usar un dispositivo que permitiera captar profundidad a nivel de hardware, como es el caso de las cámaras de profundidad o de luz estructurada. Una cámara de profundidad no recoge información de luz y proporciona una imagen bidimensional de los objetos (como lo hacen las cámaras convencionales), si no que registra la distancia de los objetos respecto a su origen, por lo que en vez de proporcionar una imagen de cómo se ven los objetos, proporciona una de donde están ubicados, de su posición en el espacio. Esta imagen de profundidad está compuesta por un arreglo de pixeles que, en vez de

contener información de su intensidad y color, tiene un valor de distancia respecto a la cámara, permitiendo reconocer donde empieza y termina un objeto, que tan lejos están, como es su forma tridimensional y muchas otras cosas. Por lo tanto, en una imagen de profundidad es mucho más fácil trabajar el procesamiento de imágenes para la detección de objetos debido a que técnicas como la detección de bordes o la segmentación no se ven afectados por aspectos como la iluminación o la perspectiva.

El Kinect es una cámara de profundidad comercial desarrollada por Microsoft y fue diseñada específicamente como un complemento de la consola de videojuegos XBOX360 para jugar algunos títulos por medio de movimientos o gestos con el cuerpo. El Kinect está compuesto por tres elementos principales, una cámara RGB, una cámara infrarroja y un proyector que genera una cuadrícula de puntos infrarrojos sobre todo lo que se encuentra frente a él, por lo que podemos capturar ambos tipos de imágenes, tanto imágenes con información de color e intensidad de luz, como imágenes de profundidad.

Figura 26. *Partes del Kinect*

Adaptado de [10]

Cuando el Kinect se lanzó en 2010 fue muy popular y todo un éxito comercial, haciendo que muchos desarrolladores de todo el mundo empezaran a trabajar en controladores y librerías de código abierto para utilizar Kinect en multitud de aplicaciones que entran en el campo de la visión artificial.

A partir de estos esfuerzos surgió la comunidad de Openkinect y su librería Libfreenect, que permite acceder a muchas de las funciones del Kinect incluyendo sus datos de profundidad, por lo que surgieron cientos de proyectos en la comunidad académica y en el mundo comercial.

Es entonces cuando PrimeSense, la empresa que ayudó a Microsoft a construir el Kinect, lanzó al público sus controladores para trabajar con Kinect, haciendo posible acceder a toda la información en profundidad, además lanzó OPENNI (Open natural interaction) un sistema que permite a los desarrolladores acceder a muchos datos y procesar "usuarios" permitiendo así ubicar la posición tridimensional de muchas de las articulaciones del cuerpo.

Max Rheiner, profesor de la universidad de Zúrich creo la librería SimpleOpenNI para trabajar con OPENNI en Processing, el cual es un entorno de desarrollo creado por el MIT enfocado en el desarrollo de aplicaciones visuales y proyectos interactivos, por lo que el uso de SimpleOpenNI en Processing representa un amplio abanico de posibilidades y ayudas para la creación de interfaces naturales con Kinect.

Figura 27. *Desarrollo de Kinect para CV*

a. Logo de OPENNI b. Logo de Processing

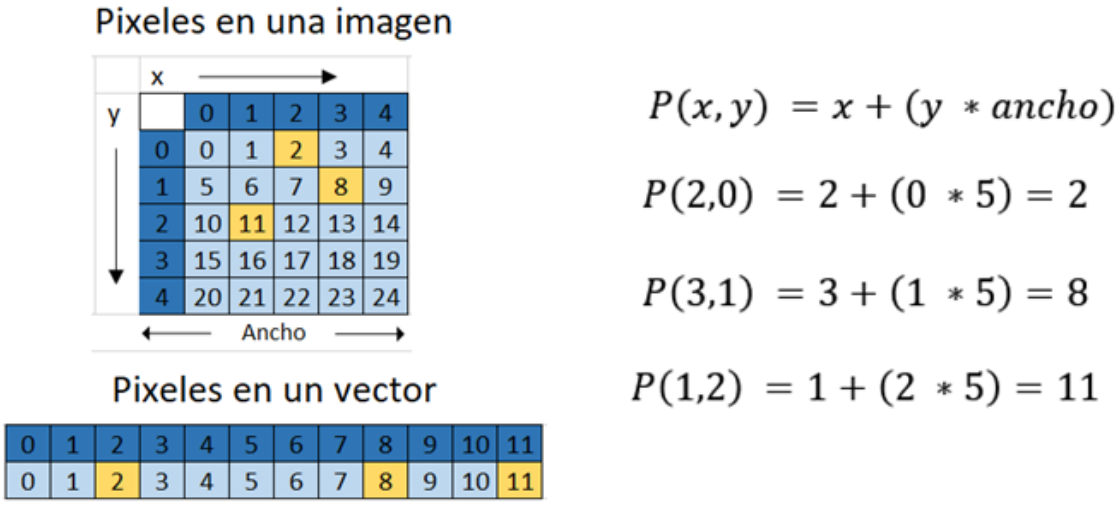
Tomado de [11]

4.2.2 Tratamiento de imágenes de profundidad

Como se mencionó anteriormente, Kinect tiene un proyector infrarrojo que trabaja en conjunto con una cámara infrarroja para generar una imagen de profundidad, el proyector genera una malla de puntos sobre todo lo que se encuentre frente a él, y son captados por la cámara IR, generando un arreglo de pixeles con información de la distancia a la que están cada uno de estos.

Podemos recorrer la información de cada uno de estos pixeles y darles un valor de 0 a 255 de acuerdo con su distancia lo que proporcionará una imagen de intensidad de luz (escala de grises) que representa la profundidad de toda la escena, para lograr esto es necesario entender la forma como se codifican y procesan los pixeles de las imágenes.

Figura 28. *Imágenes digitales*

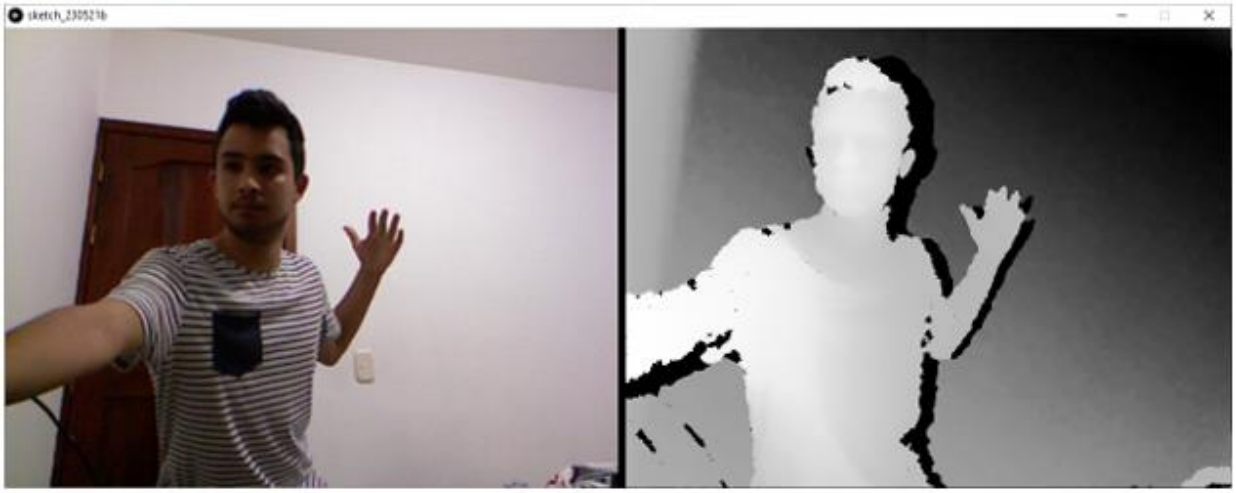


En la figura 28 se puede ver la relación que existe entre una imagen (arreglo de pixeles bidimensional) y un vector de una dimensión, de acuerdo con la relación si se tiene un píxel en la posición X = 3 e Y = 1 en una imagen con 5 pixeles de ancho, se está haciendo referencia a la posición 16 del arreglo de pixeles, de esta forma se puede desarrollar un algoritmo que permita recorrer uno a uno los pixeles de una imagen y generar un mapeado bidimensional con los valores

de profundidad en una escala de 0 a 255, haciendo que las cosas cuanto más lejos estén más oscuras se vean y cuanto más cerca, más blancas (figura 29-a).

Además de la cámara de profundidad, Kinect tiene integrada una cámara RGB, lo que nos proporciona imagen a todo color de la escena (Figura 29-b), esto es muy útil porque podemos visualizar ambas imágenes (color y profundidad) para comparar las salidas y entender mejor el procesamiento que se esté desarrollando.

Figura 29. *Mapeado de profundidad e imagen RGB*

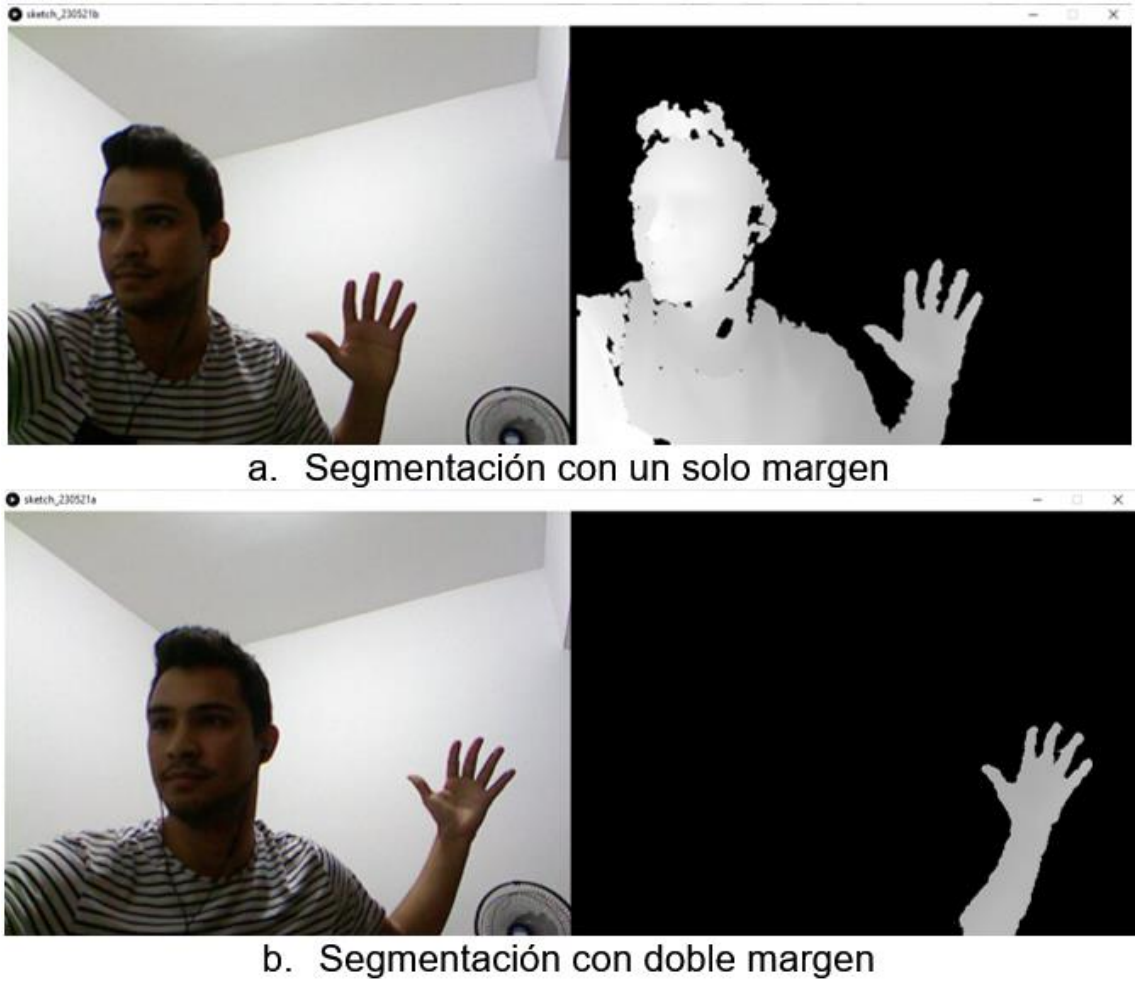


Teniendo el mapeado de profundidad podemos segmentar la imagen de acuerdo con la profundidad de cada píxel, por ejemplo, si le damos un valor de 0 a los pixeles que tengan una profundidad superior a un metro nuestra imagen de profundidad solamente mostrará lo que esté a menos de un metro de distancia, borrando el fondo de la imagen (figura 30-a).

Si establecemos dos márgenes podemos mostrar sólo una porción de la imagen, por ejemplo, si sólo mostramos los pixeles que estén por entre 0.6 y 0.8 metros sólo se mostrará lo que se encuentre en ese margen de 20 centímetros, y si nuestras manos se encuentran en esa posición

lograremos segmentar nuestra imagen para únicamente las manos, tal como se indica en la figura 30-b.

Figura 30. *Segmentación por profundidad*



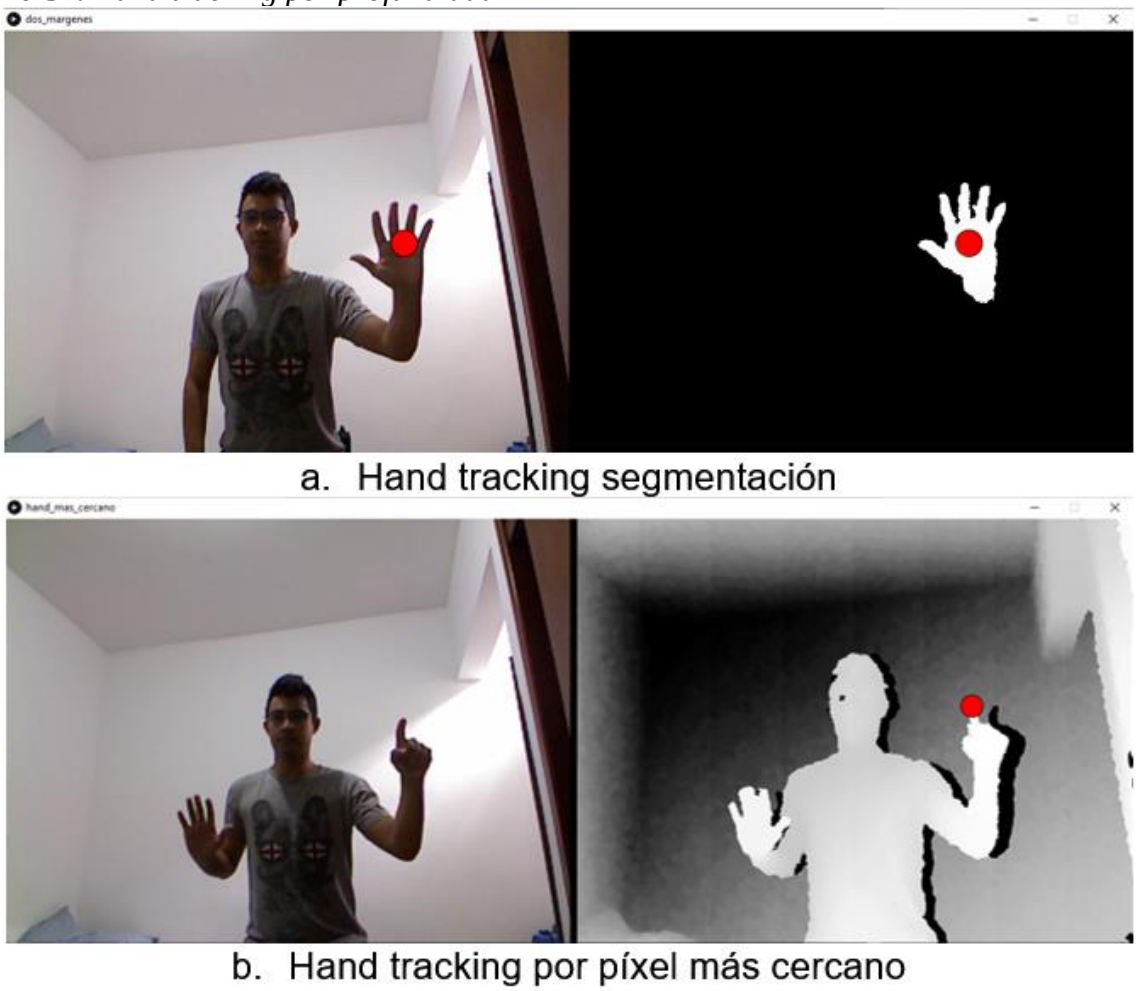
4.2.3 Técnicas de seguimiento

4.2.3.1 Hand tracking. Pudiendo procesar las imágenes con ayuda del mapeado de profundidad es posible usar la segmentación para separar partes de nuestro cuerpo como nuestra mano, si hacemos seguimiento de este conjunto de píxeles y calculamos el centro del área que se

forma, podremos obtener la posición de nuestra mano como las coordenadas de su punto central, resolviendo el primer problema del proyecto (figura 31-a).

Esta propuesta de seguimiento trae consigo una limitación de profundidad ya que, si nos salimos del margen planteado para la segmentación, no se podrá hacer seguimiento de la mano. Una mejoría para esta limitación resulta cuando trabajamos con el píxel que tenga el valor mínimo de profundidad, de esta forma podremos hacer seguimiento siempre y cuando se cumpla que nuestra mano sea la articulación más cercana al Kinect (figura 31-b).

Figura 31. *Hand tracking por profundidad*



4.2.3.2 Skeleton tracking. Para eliminar las limitaciones al hacer Hand tracking con los datos de profundidad es posible aplicar técnicas de seguimiento más avanzadas, como es el caso del Skeleton tracking, el cual como su nombre lo indica, se refiere al seguimiento del esqueleto del ser humano.

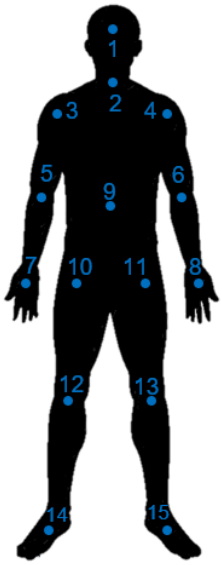
El Skeleton tracking es ampliamente usado en diversos campos, uno de los usos más conocidos es en el campo de la animación 3D, en donde permite seguir correctamente los movimientos del cuerpo y usar estos datos para que un personaje animado se mueva de forma más realista.

Cuando se hace seguimiento del esqueleto en realidad se calcula de forma constante la posición y orientación de un cuerpo o usuario y partir de este se obtienen las posiciones tridimensionales de cada uno de sus puntos articulares, como las manos, los codos, el cuello o las rodillas, haciendo posible procesar gestos de acuerdo con la posición corporal y solucionando las limitaciones anteriormente mencionadas.

Está es la razón por la cual se usará el hardware de Kinect y el procesamiento de OpenNI, ya que brindan un potente sistema de reconocimiento de usuarios y 15 puntos articulares que se pueden usar para codificar gestos o hacer seguimientos de puntos específicos a través de la técnica del Skeleton tracking.

Figura 32. *Puntos articulares que reconoce el Kinect usando SimpleOpenNI*

Numero	Articulación
1	Cabeza
2	Cuello
3	Hombro derecho
4	Hombro izquierdo
5	Codo derecho
6	Codo izquierdo
7	Mano derecha
8	Mano izquierda
9	Torso
10	Cadera derecha
11	Cadera izquierda
12	Rodilla derecha
13	Rodilla izquierda
14	Pie derecho
15	Pie izquierdo



Usando SimpleOpenNI podemos acceder a los 15 puntos mencionados y trazar líneas entre ellos para dibujar el skeleton tracking de nuestra figura completa (figura 33-a). Teniendo en cuenta que estos puntos se calculan a partir de la posición y orientación de nuestro cuerpo, no dependen de la profundidad ni tienen limitaciones de segmentación, por lo que se pueden captar movimientos específicos de forma precisa.

Para hacer skeleton tracking con SimpleOpenNI se habla de “users” que hacen referencia a los usuarios que se captan por el Kinect al iniciar su procesamiento, pudiendo captar varios usuarios (cuerpos humanos) y calculando para cada uno de ellos todas sus posiciones articulares. SimpleOpenNI nos proporciona una “imagen de usuario”, que es una imagen de mapeado de profundidad (escala de grises), pero resaltando en color la silueta del cuerpo de la persona, facilitando mucho la visualización y seguimiento del cuerpo y sus articulaciones (figura 33-b).

Si usamos los puntos articulares correspondientes para la mano derecha e izquierda podemos rastrear la posición de ambas manos y usar estas coordenadas para procesar gestos en nuestra interfaz, de la cual se hablará en la sección 4.5.

Figura 33. *Seguimiento*

a. Skeleton tracking

b. Imagen de usuario con seguimiento de manos

4.3 Robótica

4.3.1 ABB IRB120

El IRB120 es un brazo robótico industrial de seis grados de libertad con capacidad de carga para 3 kilogramos de la empresa ABB Robotics. Este robot manipulador es el propuesto para el desarrollo del proyecto debido a que el campus universitario cuenta con uno de estos brazos en sus laboratorios, por lo que será posible realizar pruebas del funcionamiento de la interfaz natural en un entorno físico.

El robot cuenta con una controladora de la misma empresa llamada IRC5 y el software de control RobotWare, el cual permite acceder a todos los apartados de robot como el control de movimiento de sus articulaciones o efector final, además de una amplia biblioteca de funciones que permiten ejecutar rutinas de movimiento, controlar entradas y salidas o establecer comunicación con otros sistemas, lo que será de vital importancia en el desarrollo del proyecto.

Antes de trabajar con el robot físico se realizarán pruebas en simulación, para esto se usará el software de simulación RobotStudio, el cual, al ser desarrollado por la misma empresa fabricadora del robot, se encuentra perfectamente integrado con el sistema de control RobotWare, facilitando en extrema medida el pasar de simulación a pruebas físicas.

Figura 34. *Sistema robótico del proyecto*

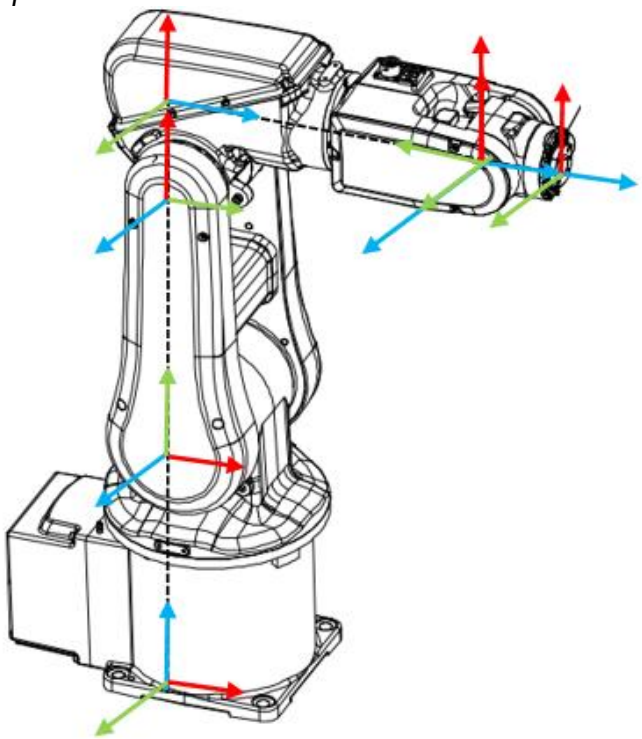
Adaptado de [12]

4.3.2 Cinemática directa

El IRB120 cuenta con 6 articulaciones de tipo rotacional logrando controlar completamente su posición y orientación en los 3 ejes (x, y, z, α, β, γ), y para realizar su cinemática directa se realizará el mismo proceso que se desarrolló como ejemplo en el marco teórico del proyecto para un robot de 2 grados de libertad.

Primeramente, se establecerán los marcos de referencia para las seis articulaciones del robot de acuerdo con los pasos planteados por el algoritmo de Denavit-Hartenberg dando como resultado lo que se muestra en la figura 35.

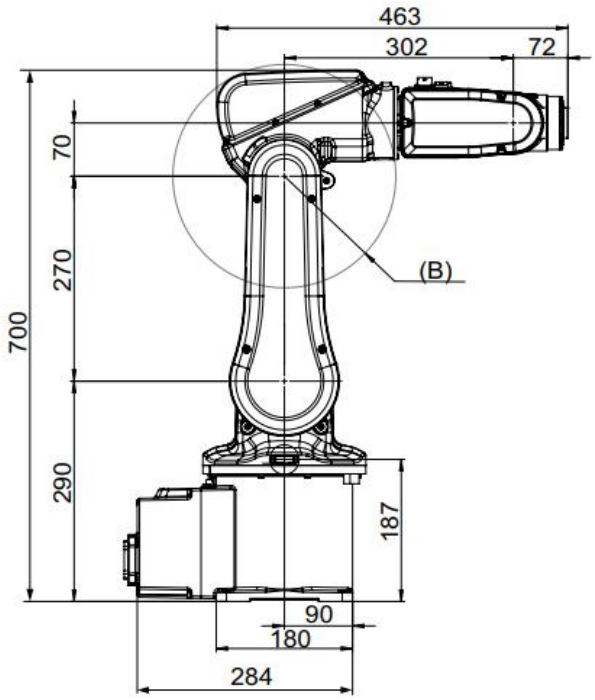
Figura 35. *Marcos de referencia D-H IRB120*



Adaptado de [12]

Habiendo ubicado los marcos de referencia para cada articulación y haciendo uso de todas las medidas del robot (figura 36) se procede a calcular cada uno de los 4 parámetros D-H para cada marco.

Figura 36. *Medidas del robot IRB120*



Tomado de [12]

Tabla 2. *Parámetros D-H IRB120*

Marco	$\boldsymbol{\theta_i}$	$\boldsymbol{d_i}$	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$
A_1^0	q_1	290	0	-90
A_2^1	q_2 - 90	0	270	0
A_3^2	q_3	0	70	-90
A_4^3	q_4	302	0	90
A_5^4	q_5	0	0	-90
A_6^5	q_6	72	0	0

Habiendo calculado todos los parámetros D-H se procede a calcular las 6 matrices de los parámetros que se obtuvieron en la tabla 2.

Figura 37. *MTH D-H IRB120*

```
A_0_1 =

[cos(q1),  0, -sin(q1),   0]
[sin(q1),  0,  cos(q1),   0]
[      0, -1,        0, 290]
[      0,  0,        0,   1]

A_1_2 =

[ sin(q2), cos(q2), 0,  270*sin(q2)]
[-cos(q2), sin(q2), 0, -270*cos(q2)]
[       0,       0, 1,            0]
[       0,       0, 0,            1]

A_2_3 =

[cos(q3),  0, -sin(q3), 70*cos(q3)]
[sin(q3),  0,  cos(q3), 70*sin(q3)]
[      0, -1,        0,         0]
[      0,  0,        0,         1]

A_3_4 =

[cos(q4), 0,  sin(q4),   0]
[sin(q4), 0, -cos(q4),   0]
[      0, 1,        0, 302]
[      0, 0,        0,   1]

A_4_5 =

[cos(q5),  0, -sin(q5), 0]
[sin(q5),  0,  cos(q5), 0]
[      0, -1,        0, 0]
[      0,  0,        0, 1]

A_5_6 =

[cos(q6), -sin(q6), 0,  0]
[sin(q6),  cos(q6), 0,  0]
[      0,        0, 1, 72]
[      0,        0, 0,  1]
```

Al realizar el producto de las seis matrices se obtiene la matriz de transformación homogénea total (MTHT) que relaciona la base con el efector final del robot, si observamos las tres primeras posiciones de la última columna obtenemos las ecuaciones que me indican la posición del efector final (x, y, z) en función de los valores articulares (q1, q2, q3, q4, q5), resolviendo así el problema cinemático directo para el robot ABB IRB120 propuesto para el proyecto.

Figura 38. *Ecuaciones de posición IRB120*

```
posy =

270*sin(q1)*sin(q2) - 72*sin(q5)*(cos(q2)*cos(q4)*sin(q1)*sin(q3) - cos(q1)*sin(q4)
+ cos(q3)*cos(q4)*sin(q1)*sin(q2)) - 302*sin(q1)*sin(q2)*sin(q3) + 72*cos(q2 + q3)*cos(q5)*sin(q1)
+ 302*cos(q2)*cos(q3)*sin(q1) + 70*cos(q2)*sin(q1)*sin(q3) + 70*cos(q3)*sin(q1)*sin(q2)

posx =

270*cos(q1)*sin(q2) - 72*sin(q5)*(sin(q1)*sin(q4) + cos(q1)*cos(q2)*cos(q4)*sin(q3)
+ cos(q1)*cos(q3)*cos(q4)*sin(q2)) + 72*cos(q2 + q3)*cos(q1)*cos(q5) + 302*cos(q1)*cos(q2)*cos(q3)
+ 70*cos(q1)*cos(q2)*sin(q3) + 70*cos(q1)*cos(q3)*sin(q2) - 302*cos(q1)*sin(q2)*sin(q3)

posz =

70*cos(q2 + q3) - 302*sin(q2 + q3) + 270*cos(q2) + 36*sin(q4 - q5)*cos(q2 + q3)
- 36*cos(q2 + q3)*sin(q4 + q5) - 72*sin(q2 + q3)*cos(q5) + 290
```

Para corroborar las ecuaciones podemos usar las herramientas de MATLAB. Calculamos la MTHT como la multiplicación de las seis matrices articulares y extraemos las tres primeras filas de la última columna para la posición x, y, z respectivamente.

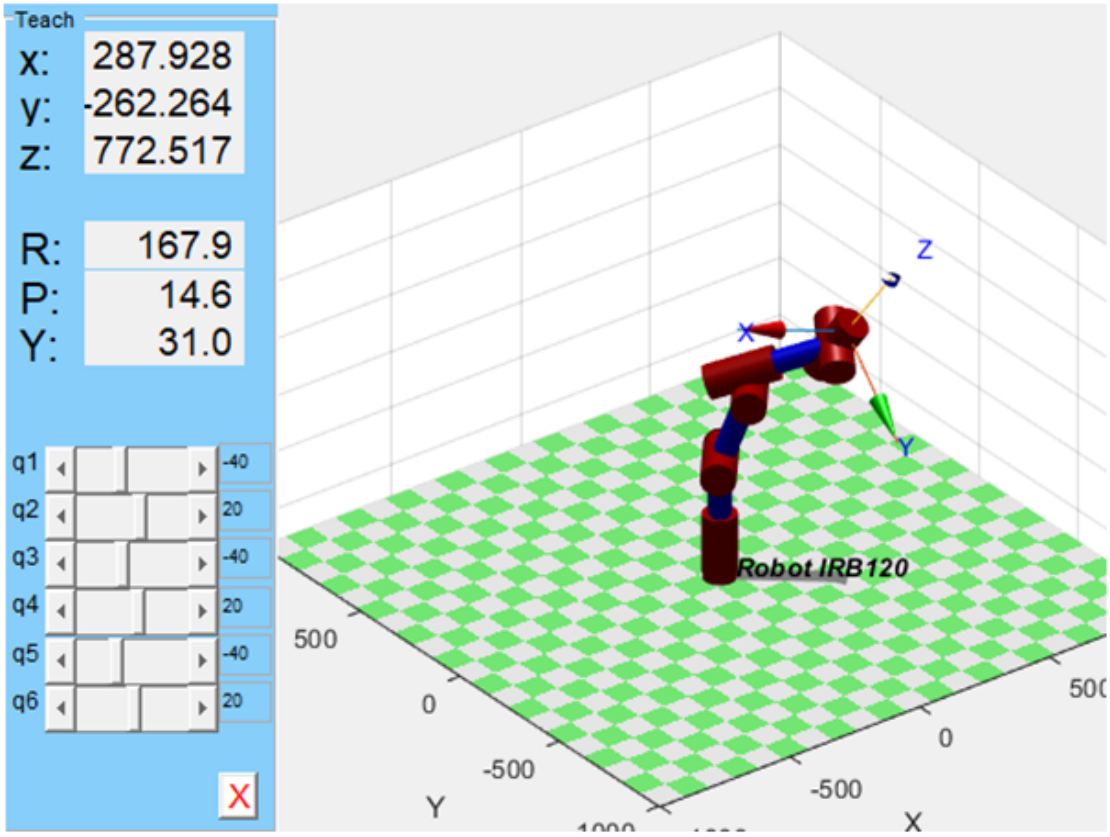
Posteriormente, usamos el método matlabFunction de MATLAB para convertir las ecuaciones de posición en funciones evaluables para los valores articulares deseados.

Finalmente, podemos volver a usar la librería Robotic Toolbox de Peter Corke para graficar el robot a partir de los parámetros D-H y comprobar los resultados obtenidos con las ecuaciones.

Figura 39. *RBT IRB120*

```
>> A_0_6 = simplify(A_0_1*A_1_2*A_2_3*A_3_4*A_4_5*A_5_6);

posx = A_0_6(1,4);
posy = A_0_6(2,4);
posz = A_0_6(3,4);

calcularposx = matlabFunction(posx);
calcularposy = matlabFunction(posy);
calcularposz = matlabFunction(posz);

art1 = -40 *(pi/180);
art2 = 20 *(pi/180);
art3 = -40 *(pi/180);
art4 = 20 *(pi/180);
art5 = -40 *(pi/180);
art6 = 20 *(pi/180);

calcularposx(art1,art2,art3,art4,art5)
calcularposy(art1,art2,art3,art4,art5)
calcularposz(art2,art3,art4,art5)
```

```
ans =

  287.9283

ans =

 -262.2638

ans =

  772.5167
```

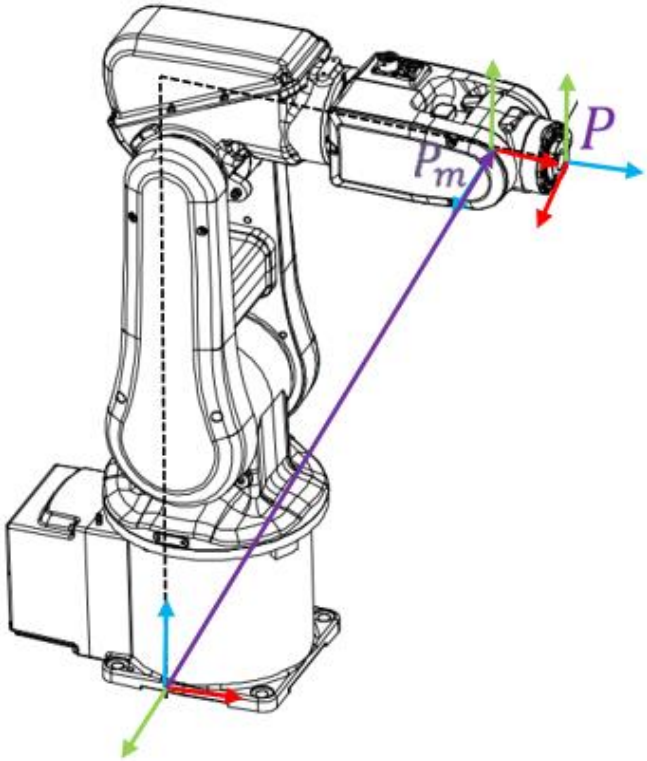


4.3.3 Cinemática inversa

Como se mencionó en el marco teórico del proyecto, la cinemática inversa de un robot puede ser mucho más compleja de determinar que su cinemática directa, por lo que existen varios métodos que permiten facilitar la resolución de este problema.

Para el caso del IRB120 es posible aplicar el método de desacoplo cinemático, el cual permite dividir el desarrollo de la cinemática inversa en dos problemas, el cálculo de la posición y el cálculo de la orientación del efector final, siempre y cuando las últimas tres articulaciones se crucen en un punto, comúnmente denominado “muñeca del robot”. Al analizar la composición articular del IRB120 nos damos cuenta de que la posición del efector final depende de 4 articulaciones, y puede calcularse como la posición que resulta de las tres primeras articulaciones (la muñeca del robot Pm) más la distancia que le aporta la quinta articulación hasta la punta P.

Figura 40. Desacoplo cinemático IRB120



Adaptado de [12]

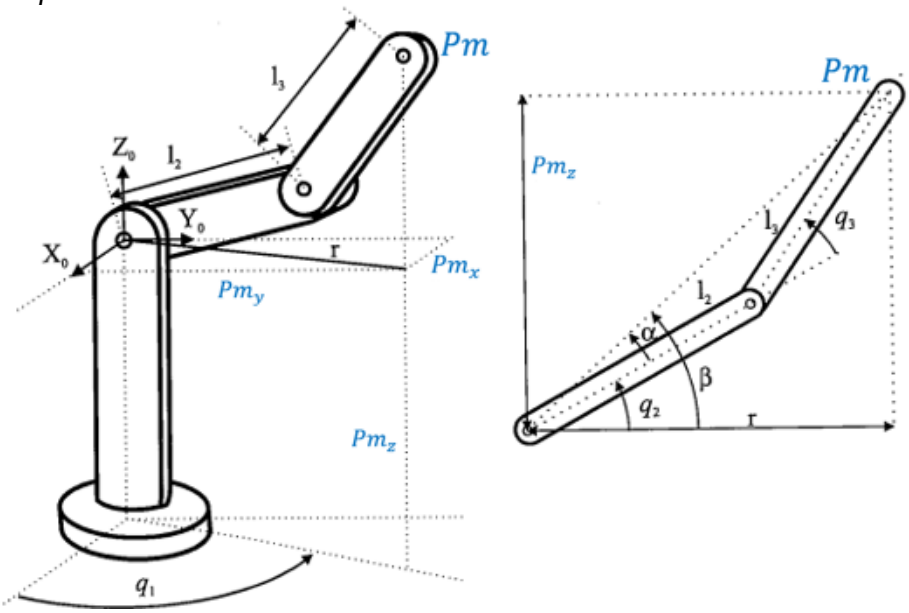
Por tanto, para hallar la posición de Pm solo debemos restarle a la posición de la punta deseada (Px, Py, Pz) la distancia d que aporta la quinta articulación (72 mm) multiplicada por la orientación deseada del efector final.

$$Pm = P - R \cdot d \tag{10}$$

$$\begin{bmatrix} Pmx \\ Pmy \\ Pmz \end{bmatrix} = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix} - \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix} = \begin{bmatrix} P_x - R_{13} \cdot d \\ P_y - R_{23} \cdot d \\ P_z - R_{33} \cdot d \end{bmatrix} \tag{11}$$

Si analizamos las tres articulaciones hasta la muñeca del robot Pm, nos damos cuenta de que tiene una estructura muy simple, debido a que el valor de la primera articulación (q1) se relaciona de forma directa con las coordenadas de Pm y las dos articulaciones siguientes (q2 y q3) representan un robot planar como el desarrollado en el marco teórico.

Figura 41. *Composición articular hasta muñeca del robot*



Adaptado de [3]

Tenemos entonces para la primera articulación el ángulo que se forma entre Pm_x y Pm_y que se obtiene aplicando la función arco tangente.

$$q_1 = tan^{-1}\left(\frac{Pm_y}{Pm_x}\right) \tag{12}$$

Y para las dos articulaciones siguientes el mismo procedimiento y resultados obtenidos en el marco teórico del proyecto.

$$r = \pm\sqrt{Pm_x{}^2 + Pm_y{}^2} \tag{13}$$

$$q_2 = Tan^{-1}\left(\frac{Pm_z}{r}\right) - Tan^{-1}\left(\frac{l_3 Sen(q_3)}{l_2 + l_3 Cos(q_3)}\right) \tag{14}$$

$$Cos(q_3) = \frac{r^2 + Pm_z{}^2 - l_2{}^2 - l_3{}^2}{2l_2 l_3} \tag{15}$$

$$q_3 = Tan^{-1}\left(\frac{\pm\sqrt{1 - Cos^2(q_3)}}{Cos(q_3)}\right) \tag{16}$$

Habiendo obtenido las ecuaciones para las tres primeras articulaciones, se resuelve la primera parte del desarrollo cinemático inverso por desacoplo cinemático correspondiente a la posición del robot.

Para las tres últimas articulaciones que corresponden a la orientación del robot, se puede aplicar un método algebraico que consiste en despejar la orientación de las tres últimas articulaciones considerando que la matriz de orientación total del sistema R_6^0 es igual a la matriz de orientación desde la base hasta la tercera articulación R_3^0 multiplicada por la matriz de orientación desde la tercera hasta ultima articulación R_6^3.

La matriz de orientación desde la base hasta tercera articulación R_3^0 se puede obtener con el método Denavit-Hartenberg (figura 42) y fue calculada en la resolución del problema cinemático directo, por lo que sólo basta con reemplazar los valores articulares (q1, q2, q3) con los obtenidos en la primera parte del desacoplo cinemático al evaluar en la posición de la muñeca (Pmx, Pmy, Pmz).

Figura 42. *Matriz de orientación de 0 a 3*

```
>> A_0_3 = simplify(A_0_1*A_1_2*A_2_3);
   R_0_3 = A_0_3(1:3,1:3)

R_0_3 =

[-sin(q2 + q3)*cos(q1),  sin(q1), cos(q2 + q3)*cos(q1)]
[-sin(q2 + q3)*sin(q1), -cos(q1), cos(q2 + q3)*sin(q1)]
[         cos(q2 + q3),        0,          sin(q2 + q3)]
```

Como la orientación total del sistema R_6^0 y la matriz de orientación desde la base hasta tercera articulación R_3^0 son valores conocidos, se despeja R_6^3 como se muestra a continuación.

$$R_6^0 = R_3^0 \cdot R_6^3 \tag{17}$$

$$R_6^0 = (R_3^0)^T \cdot R_6^0 \tag{18}$$

Habiendo obtenido R_6^3 se despejan los valores articulares para q4, q5 y q6 usando los ángulos de Euler en la configuración ZYZ, como se muestra a continuación.

$$q_4 = tan^{-1}\left(\frac{R_6^3(2,3)}{R_6^3(1,3)}\right) \tag{19}$$

$$q_5 = tan^{-1}\left(\frac{\sqrt{1 - R_6^3(3,3)^2}}{R_6^3(3,3)}\right) \tag{20}$$

$$q_6 = tan^{-1}\left(\frac{R_6^3(3,2)}{-R_6^3(3,1)}\right) \tag{21}$$

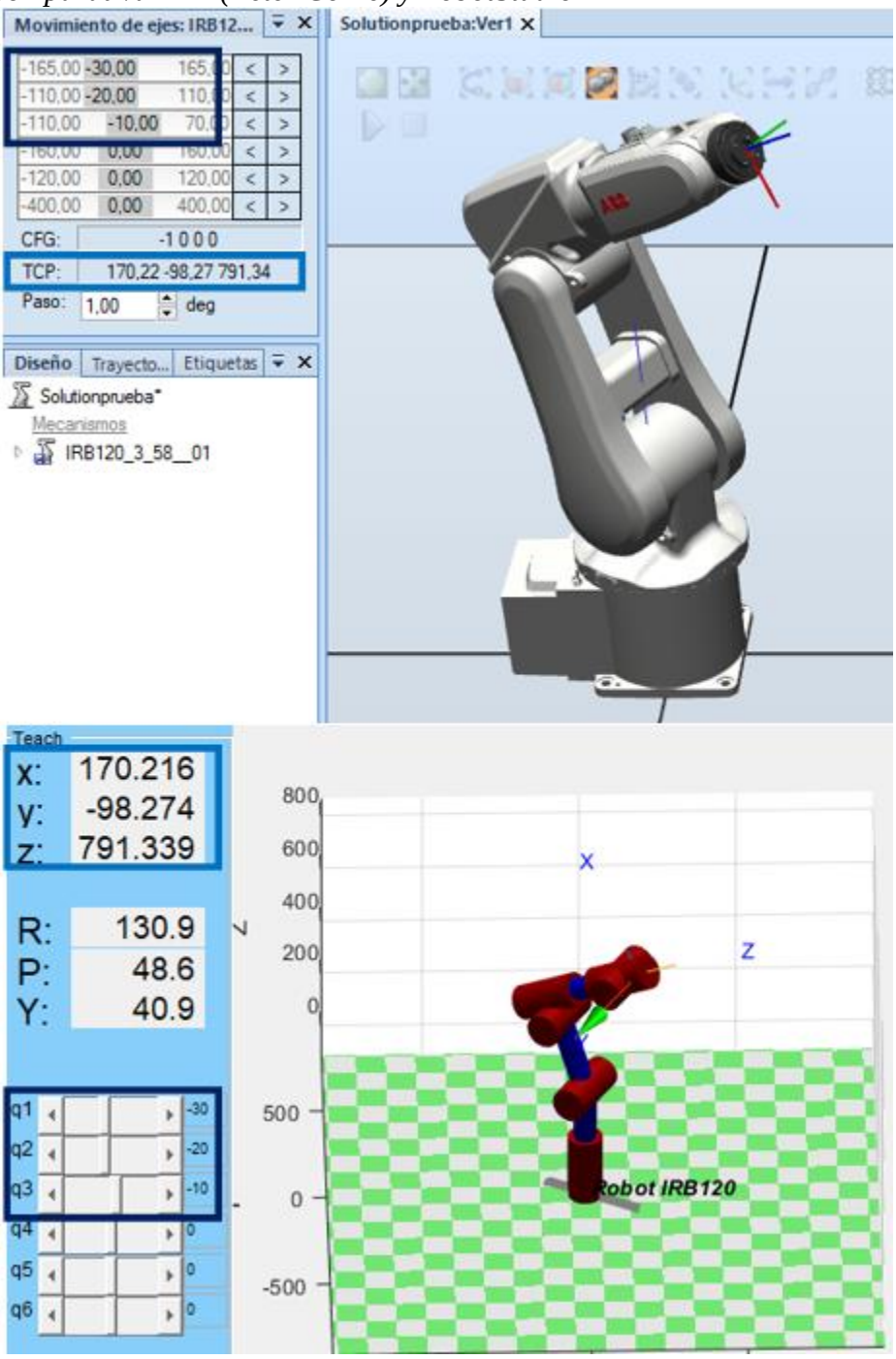
4.3.4 RobotStudio y lenguaje RAPID

RobotStudio es un software de simulación y programación robótica, ampliamente usado por la comunidad debido a su facilidad de uso y a su ambiente intuitivo y de fácil adaptación.

RobotStudio nos permite crear modelos virtuales de los robots de la familia ABB y simular su comportamiento en entornos tridimensionales, permitiendo calcular y optimizar los movimientos del robot de forma simulada antes de la implementación física, gracias a una amplia gama de herramientas y utilidades, reduciendo errores y mejorando la eficiencia de todo el proceso de programación.

RobotStudio tiene muchísimas funcionalidades para la programación del robot, como poder arrastrar la posición y orientación del efector final del robot a donde se requiera o mover de forma individual cada una de sus articulaciones tal como lo hace la librería de Matlab RBT de Peter Corke (figura 43) simplificando muchísimo el proceso de controlar la cinemática del robot.

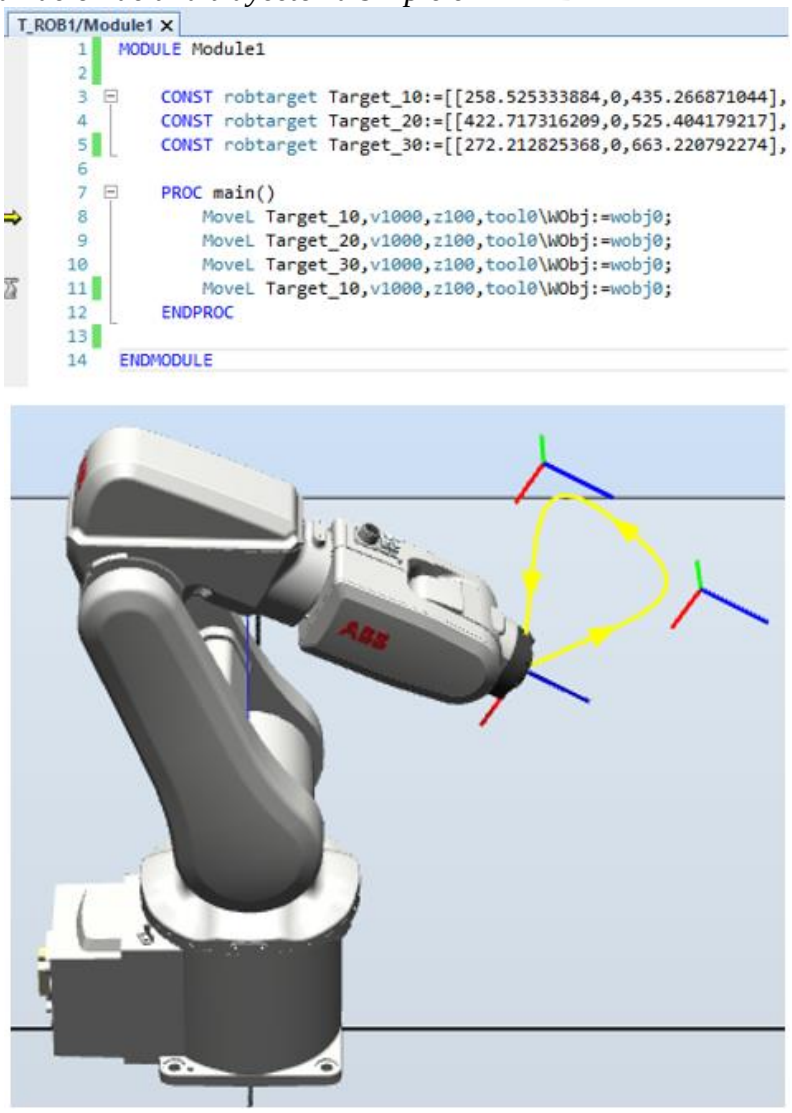
Figura 43. *Comparativa RBT (Peter Corke) y RobotStudio*



Como se mencionó anteriormente RobotStudio fue desarrollado por ABB Robotics, la misma empresa responsable del robot IRB120, por esta razón RobotStudio está perfectamente

integrado con sus funcionalidades, permitiendo incluso poder usar el robot desde la misma interfaz del programa e implementar el proceso simulado en el robot físico con unos cuantos clics. Para facilitar todo este proceso, RobotStudio permite usar una controladora IRC5 virtual y programarla en lenguaje RAPID, el cual permite acceder a todas las funcionalidades del robot mediante comandos y parámetros sencillos, permitiendo sincronizar con el ambiente simulado y viceversa.

Figura 44. *Programación de una trayectoria simple en RAPID*



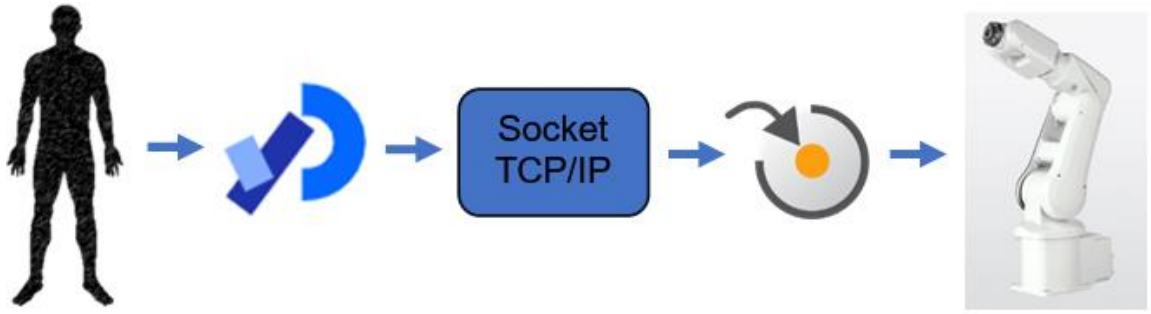
4.4 Comunicación

4.4.1 Comunicación por socket y modelo TCP/IP

Habiendo definido el sistema robótico y el sistema de visión artificial, el paso siguiente es poder establecer comunicación entre estos y enviar los datos codificados de un sistema a otro, para esto es necesario seleccionar una forma de comunicación que sea compatible con ambos sistemas y que cumpla con los requerimientos del proyecto.

Es aquí donde entra la implementación de los sockets y el modelo TCP/IP mediante el cual, es posible enviar datos específicos como la posición de las manos obtenidas por el sistema de visión artificial en Processing y recibirlos en RobotStudio para controlar los movimientos del robot.

Figura 45. *Lazo de comunicación entre sistemas*



El modelo TCP/IP es un modelo de comunicación muy parecido al modelo OSI, con la diferencia de que el primero hace uso de 4 capas en vez de las 7 del modelo OSI. Este modelo está compuesto por dos protocolos principales, el protocolo TCP (Protocolo de control de transmisión) y el protocolo IP (Protocolo de internet).

El protocolo TCP está orientado a la conexión segura, por lo que garantiza que los datos siempre lleguen al destino y se reciban de forma correcta, mientras que el protocolo IP es

responsable de establecer una ruta entre los dispositivos de la red, utilizando direcciones de red únicas también conocidas como direcciones IP.

Los sockets también llamados “conectores” o “enchufes” son un mecanismo de transferencia de datos que permite establecer un lazo de comunicación entre dos procesos o sistemas de una misma red, usando la metodología Cliente-Servidor a través protocolos de red como el TCP o el UDP.

Tabla 3. *Comparativa entre modelos de comunicación*

#	Modelo OSI	#	Modelo TCP/IP
7	Aplicación	4	Aplicación
6	Presentación		
5	Sesión		
4	Transporte	3	Transporte
3	Red	2	Red
2	Enlace	1	Enlace
1	Física		

Usando sockets se simplifica mucho el proceso de transferencia de datos del modelo TCP/IP, al ser una abstracción de los detalles de bajo nivel de los protocolos de red sólo se debe especificar una dirección IP (que hace referencia al dispositivo) y un puerto (que hace referencia al proceso), por lo que permite dejar de lado detalles complejos de la red y centrase en el desarrollo de la lógica de los programas.

Es posible hacer uso de sockets en Java, y Processing permite hacer uso de librerías de comunicación que facilitan mucho el proceso de enviar los datos a procesos externos a la

aplicación, RobotStudio por su parte integra muchas formas de comunicación con software externo, y su implementación en el lenguaje RAPID es muy intuitiva.

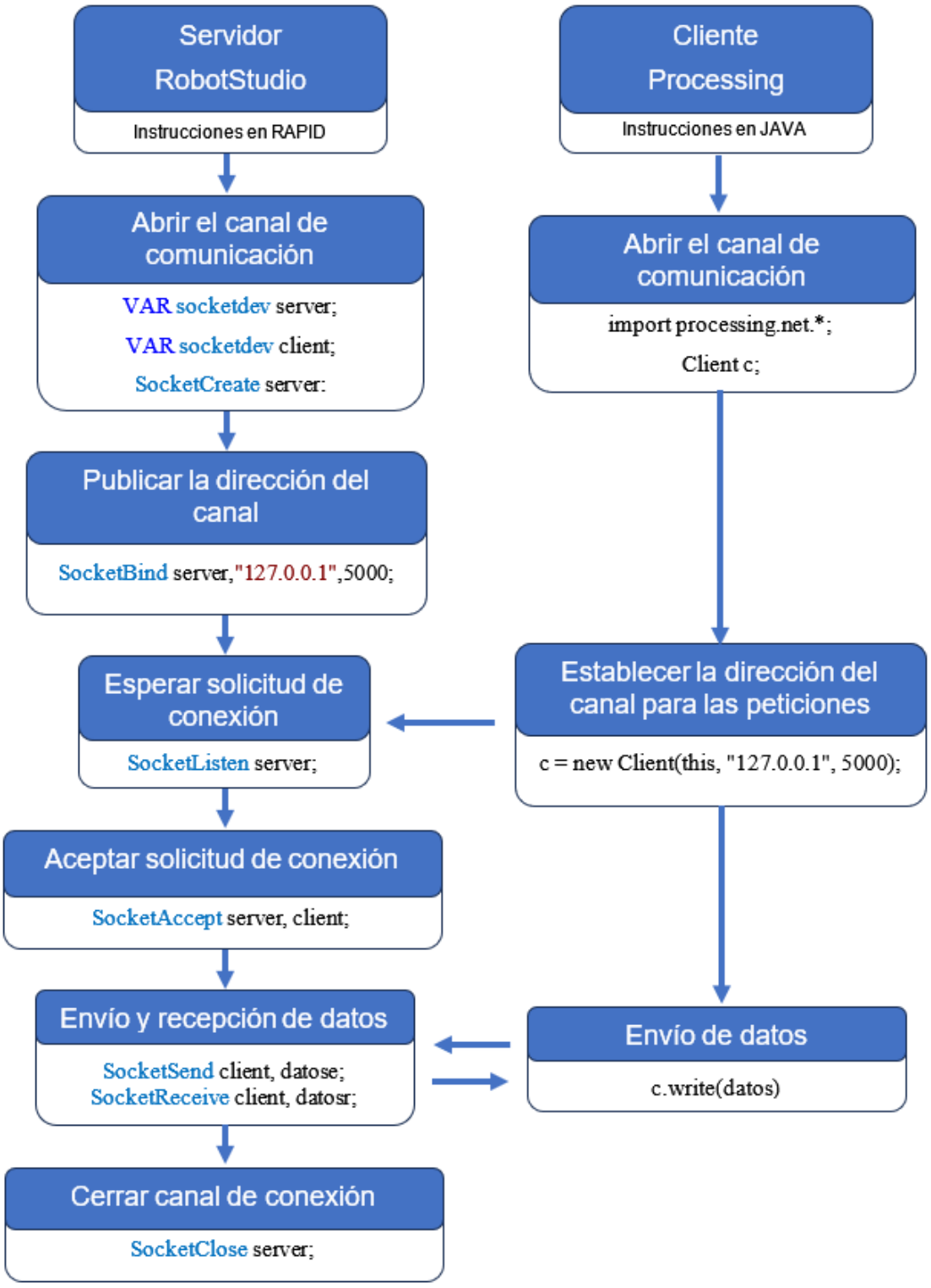
4.4.2 Proceso de comunicación del sistema

Habiendo seleccionado un sistema de comunicación compatible entre el sistema de visión artificial y el de control robótico solo resta establecer los roles y funciones de ambos sistemas dentro del proceso de conexión y transferencia.

Como el modelo de comunicación es del tipo cliente servidor, se establece el sistema de visión artificial como el cliente, que se encargará de recopilar los datos de posición de ambas manos y codificarlos para enviarlos al servidor. El servidor por su parte que es el sistema robótico debe recibir los datos e interpretarlos en acciones y controles específicos del brazo.

Definidos los roles y el proceso general, se establece un diagrama de conexión por socket que se puede observar en la figura 46.

Figura 46. *Diagrama de conexión del sistema*



4.5 Interfaz natural de usuario

Habiendo definido el sistema de visión artificial, el sistema de control robótico y el modo de comunicación entre ambos, es posible desarrollar una interfaz natural para el control del brazo robótico por medio de visión por computadora y cumplir con el objetivo principal del proyecto. A continuación, se presentan los aspectos y funcionalidades que conforman cada una de las partes fundamentales de la interfaz natural diseñada.

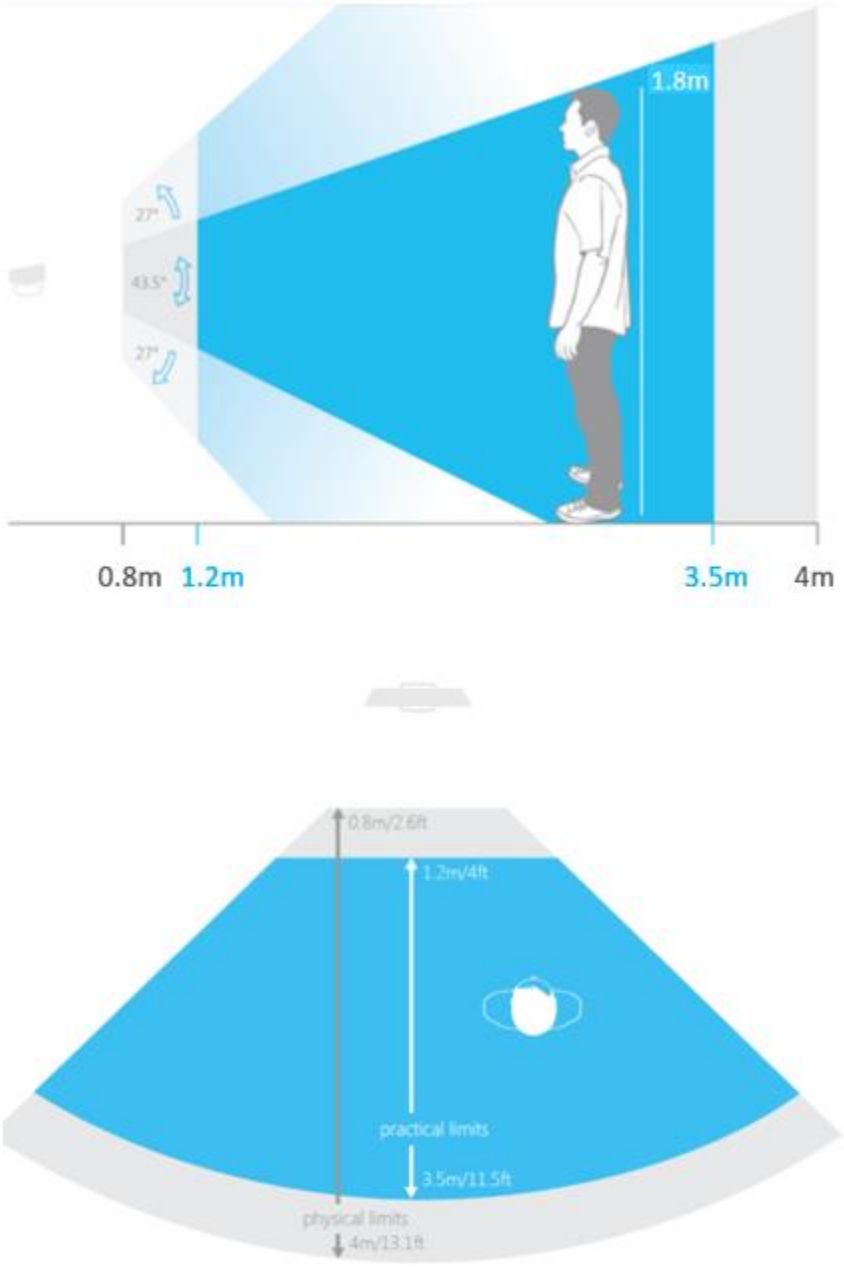
4.5.1 Sistema de visión por computadora (Cliente)

Todo el sistema de visión fue desarrollado usando lenguaje JAVA y el IDE Processing del MIT, aprovechando las ventajas y funcionalidades que este posee para el desarrollo de aplicaciones interactivas.

En primer lugar, el sistema cumple con sus funciones de cliente y declara todas las variables y parámetros necesarios para establecer la conexión por socket con el servidor y establece una ruta que servirá para enviar los datos que se procesen. Seguidamente se encarga de extraer los datos de profundidad proporcionados por el sensor infrarrojo del Kinect, además también adquiere la imagen de la cámara RGB con el fin de hacer el sistema más amigable e intuitivo con el usuario.

Con la imagen RGB se genera una ventana que muestra la escena a todo color del entorno frente al Kinect y con la información de profundidad, el sistema busca usuarios o personas en el espacio de trabajo (figura 47).

Figura 47. *Espacio de trabajo del Kinect*



Tomado de [13]

En caso de encontrar algún usuario, el sistema desarrolla el Skeleton tracking de este y dibuja un punto de color rojo en la posición tridimensional de las articulaciones correspondientes a la mano derecha e izquierda del usuario en la imagen RGB de la escena.

Finalmente, el sistema comprueba que las posiciones obtenidas para la mano derecha no superen el espacio de trabajo del brazo robótico y en caso de no hacerlo se convierten los datos en una cadena de caracteres con el formato de codificación y se envían al servidor.

El formato de codificación es “XXXYYYZZZ” y para poder usar este tipo de formato, el sistema debe enviar los datos de posición únicamente si las posiciones en el eje X y en el eje Y de la mano derecha están a una distancia de entre 99 mm y 999 mm respecto al centro del sistema de coordenadas del Kinect (Figura 48), para el caso de la posición en el eje Z, y debido a que está siempre se tomará a partir de los 1200 mm de acuerdo con el área de trabajo, esta se le resta 1100 mm y se limita a 2099 mm de forma que las medidas tomadas siempre estén en el rango que necesita el formato de codificación.

Figura 48. *Sistema de coordenadas del Kinect*

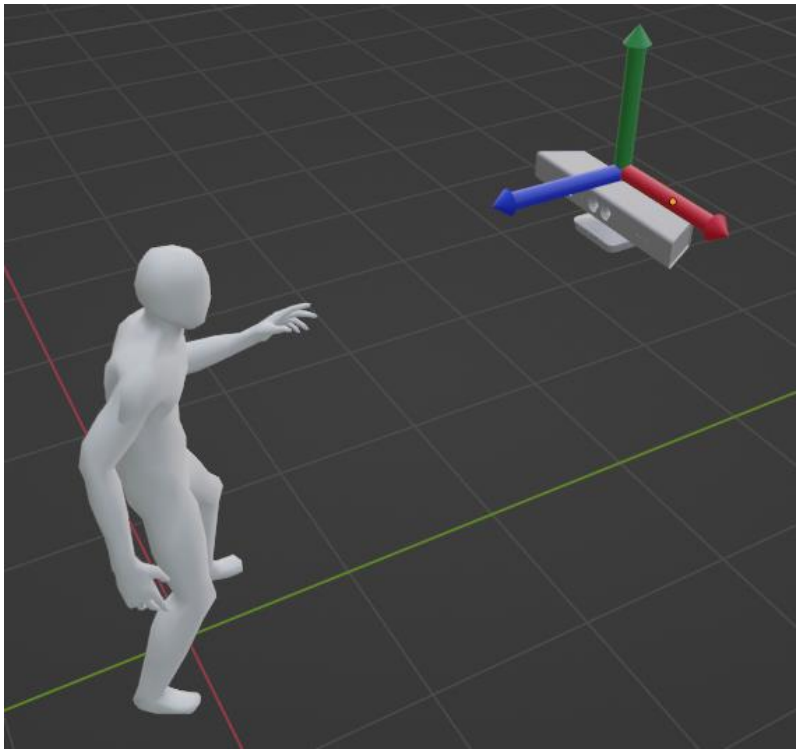
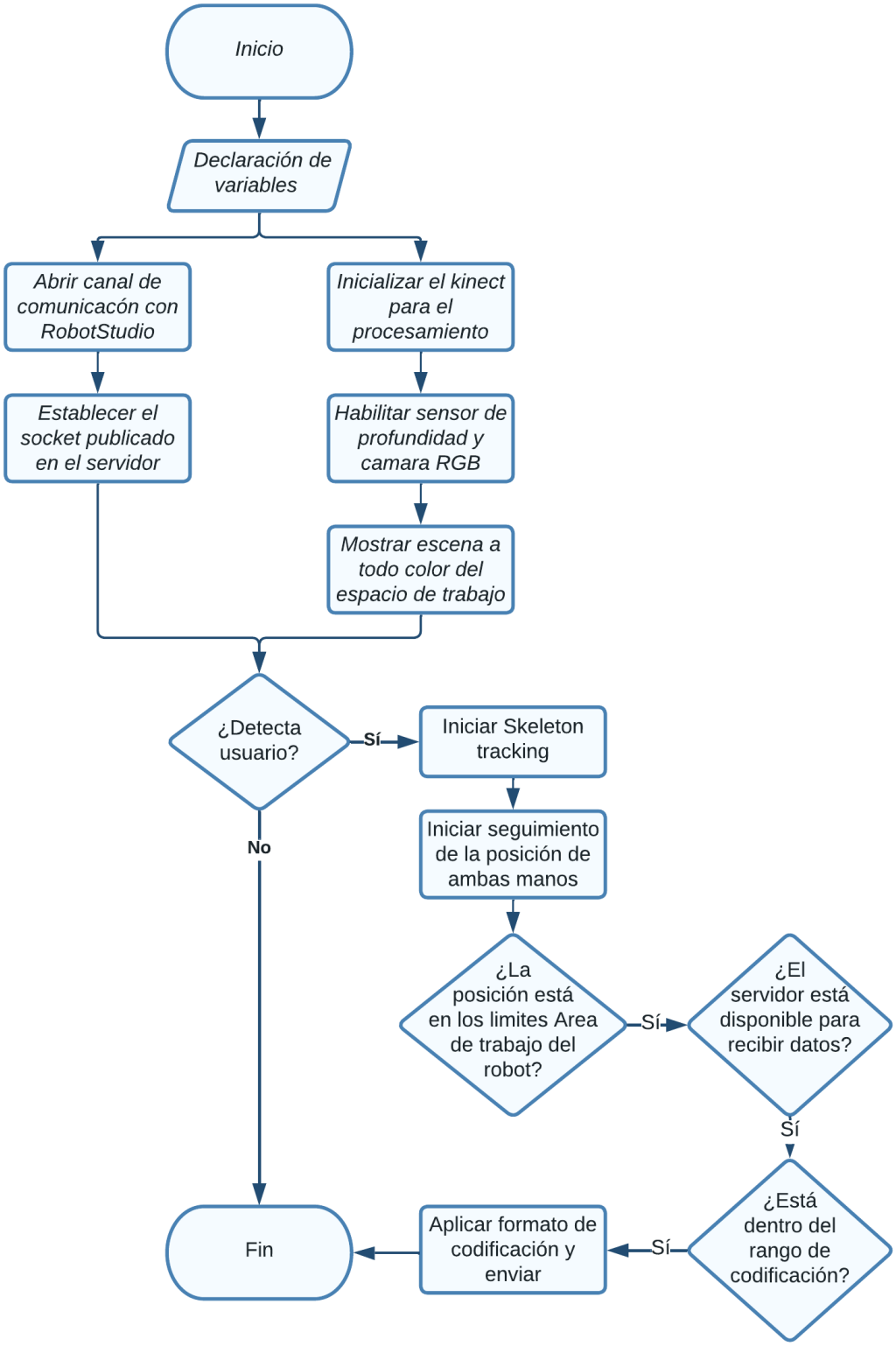


Figura 49. *Diagrama de flujo del proceso general de visión*

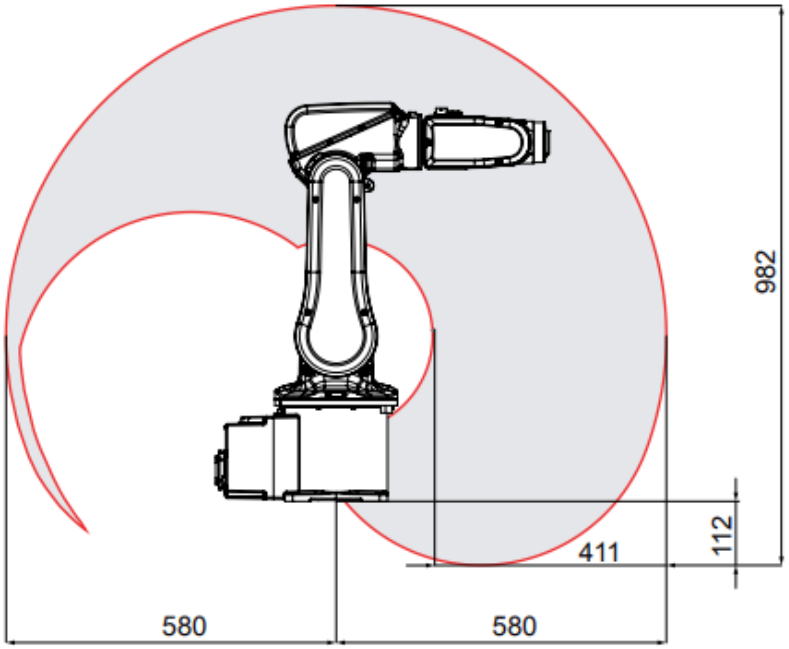


4.5.2 Sistema de control robótico (Servidor)

Todo el sistema de control robótico fue desarrollado usando lenguaje RAPID en el entorno de simulación RobotStudio de ABB, aprovechando las ventajas y funcionalidades que este posee para la simulación y programación de robots industriales.

En primer lugar, el sistema cumple con sus funciones de servidor y abre el canal de comunicación, por lo que establece una dirección IP y un puerto para la conexión por socket, con el fin de establecer una ruta que servirá para recibir los datos procesados por el Kinect. Luego el sistema espera la solicitud de conexión del cliente y tan pronto esta exista inicia el ciclo principal del programa. Tan pronto llega una solicitud de conexión, el sistema la acepta y envía el brazo robótico a una posición inicial para empezar a recibir los datos con el formato de codificación establecido (XXXYYYZZZ) y las verificaciones correspondientes del espacio de trabajo del Kinect (figura 47) y del IRB120 (figura 50).

Figura 50. *Espacio de trabajo del IRB120*



Tomado de [12]

Posteriormente, el sistema decodifica los datos descomponiendo la cadena de caracteres y extrae la información específica para el movimiento, que este caso son valores de desplazamiento para cada eje coordenado del efector final del robot respecto a la posición inicial x, y, z (250, -300, 200), es decir, una instrucción de movimiento de RAPID de tipo offset.

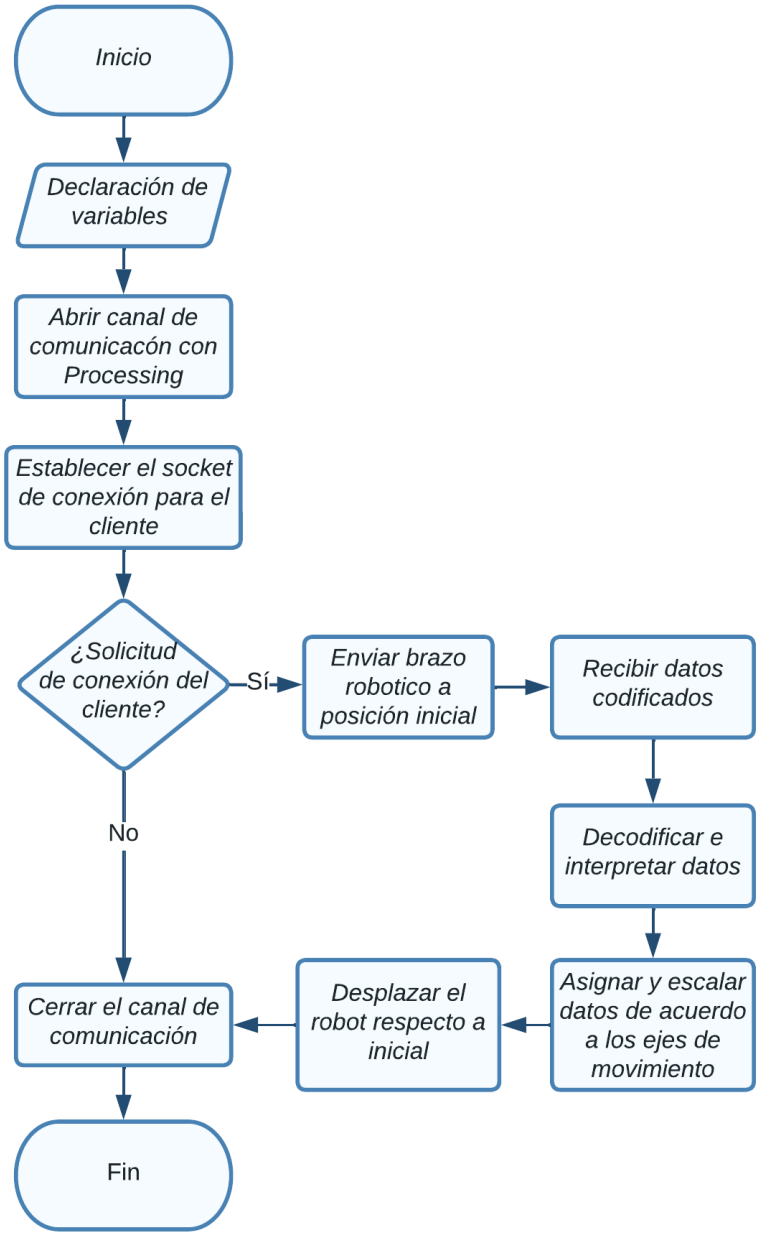
Antes de iniciar el movimiento se asignan y escalan los valores para limitar el desplazamiento de la punta del robot e impedir que los movimientos más bruscos del usuario no tengan grandes afectaciones en el sistema robótico.

Finalmente, el robot se desplaza a las posiciones de cada eje especificadas y cierra el canal de comunicación.

El movimiento del robot se realiza usando la función de “movimiento articular”, la cual nos permite desplazar la punta del robot de un punto a otro usando una trayectoria que no necesariamente es lineal (movimiento articular correspondiente a la función movej de RAPID) y que prioriza que el robot haga movimientos más cómodos y fáciles de computar.

A continuación, se muestra un diagrama de flujo que representa todo el proceso general que realiza el código en RobotStudio (código en RAPID) para la interfaz natural en su sistema de control Robótico.

Figura 51. *Diagrama de flujo del proceso general del control robótico*



4.5.3 Modos de operación

Luego de realizar pruebas y comprobar el funcionamiento general de la interfaz natural para el control de posición del brazo robótico se implementaron nuevas funcionalidades que mejoran el sistema general o que permiten un mejor control y la hacen más amigable con los usuarios.

4.5.3.1 Modo herramienta. La primera de estas funcionalidades es la implementación de una pinza electroneumática para la manipulación de objetos en el área de trabajo del brazo robótico, tanto en el entorno simulado en RobotStudio como en el físico, debido a que en el campus universitario se cuenta con una pinza electroneumática de la marca Festo.

Debido a esto, y de cara a agregar distintos modos de control del brazo robótico al modo general de control se le llamó modo herramienta, el nombre de “modo herramienta” y no “modo pinza” se debe a que hay varias herramientas que se pueden acoplar a la punta del robot y que funcionan de forma similar a la pinza, con una lógica binaria, es decir, de abrir y cerrar, encender y apagar o activar y desactivar.

Para implementar el modo herramienta se hicieron ligeros cambios tanto en el sistema de visión como en el de control robótico, pero ambos se resumen a una nueva variable para el estado de la pinza (abierta o cerrada).

4.5.3.2 Sistema de visión en el modo herramienta. En el caso del sistema de visión, se crea una variable auxiliar que se inicializa en el valor de cero, este valor representa la pinza en estado abierto. Al iniciar el programa y hacer el seguimiento de posición de las manos, se compara la posición de ambas en el eje Z y la diferencia entre estos dos valores marcará si la pinza se abre o cierra, de acuerdo con la siguiente lógica.

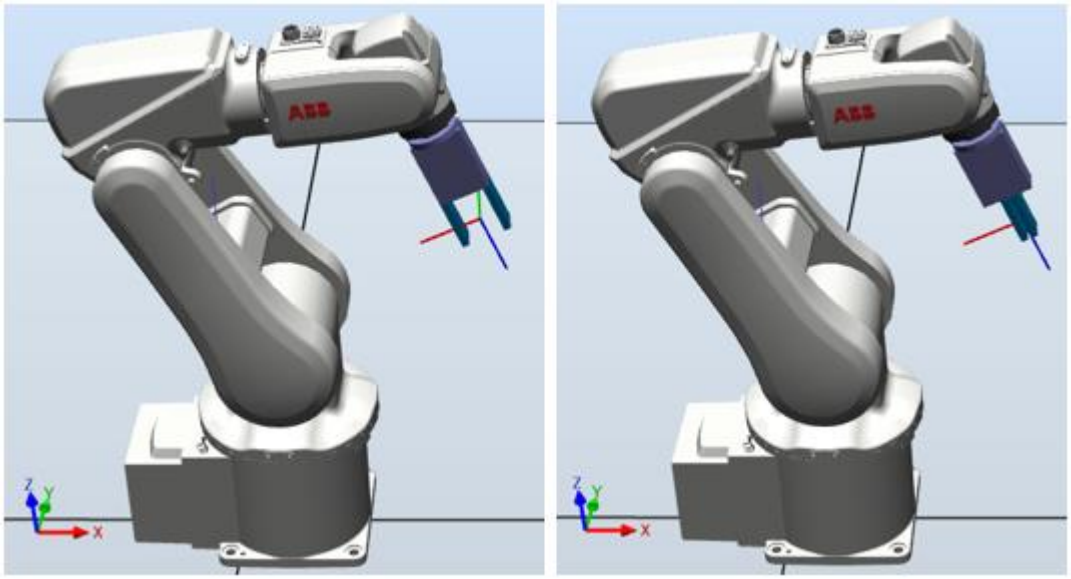
En caso de que la posición en z de la mano derecha sea mayor que la posición en z de la mano izquierda, la pinza se mantendrá abierta (variable auxiliar igual a cero), de lo contrario se cerrará (variable auxiliar igual a uno), por lo que la instrucción de cerrar o abrir la pinza en el sistema de vision se traduce en acercar o alejar la mano izquierda del Kinect.

Una vez asignado el valor de la variable auxiliar se procede a enviar los datos al servidor, pero al incluir esta variable auxiliar en los datos de la interfaz natural, el formato de codificación recibe un ligero cambio, quedando de la forma “XXXYYYZZZA” donde “A” es el valor asignado a la variable auxiliar de acuerdo con la posición de las manos.

4.5.3.3 Sistema robótico en el modo herramienta. El sistema robótico en el modo herramienta prácticamente no tiene novedades, exceptuando el hecho de que ahora recibe un dato más en la cadena de caracteres del cliente, el cual se debe comprobar al final de cada instrucción de movimiento para saber si cambia o mantiene el estado de la pinza.

RobotStudio y su entorno de simulación permite modelar herramientas en 3D y usarlas por medio de salidas y entradas virtuales que funcionan igual que el sistema de entradas y salidas de la controladora del robot IRC5, por lo que resulta muy sencillo modelar la herramienta que se tenga en físico y programarla, para luego usar la herramienta en físico con sólo modificar un parámetro en una instrucción del programa en RAPID.

Figura 52. *Herramienta simulada en RobotStudio*



4.5.3.4 Modo precisión. La segunda de estas funcionalidades nuevas para la interfaz natural es la implementación de otro modo de controlar el desplazamiento del robot, el cual permite que este se haga de forma más precisa y controlada, debido a esto se le llamó modo precisión.

Esta modalidad consiste en aumentar o disminuir el valor del desplazamiento de cada eje coordenado del efector final del robot de forma individual y a una velocidad constante.

Esta forma de mover el robot representa una diferencia muy grande respecto al modo herramienta, debido a que en este se desplaza la punta del robot en los tres ejes coordenados de forma simultánea y la velocidad del desplazamiento depende de la velocidad con la que el usuario mueva la mano.

Para lograr el movimiento en el modo precisión se hacen ajustes en prácticamente todos los componentes de la interfaz natural, tanto la forma como se usa el sistema de visión, como el formato de codificación de los datos que se envían y la interpretación de estos por parte del sistema robótico

4.5.3.5 Sistema de visión en modo precisión. Para el caso del sistema de visión en el modo precisión, el usuario debe hacer uso de ambas manos para controlar el movimiento de la punta del robot. Se debe usar la mano izquierda para especificar el eje coordenado en el cual desea desplazar el efector final, esto se hace moviendo la mano a través de tres espacios bidimensionales (plano XY) en el área de trabajo del Kinect, por lo que el sistema debe comprobar si la posición de la mano izquierda en tanto en X como en Y están dentro del rango que corresponde a mover alguno de los ejes coordenados de la punta del robot.

Al comprobar que la mano izquierda está en alguno de los espacios correspondientes a algún movimiento, el sistema pasa a comprobar la posición de la mano derecha del usuario, la cual

sirve para indicar si se requiere aumentar o disminuir valor del desplazamiento de eje coordenado de la punta del robot que se seleccionó con la mano izquierda, y para hacer esto se hace uso de otro espacio bidimensional (Plano XY) en el área de trabajo del Kinect, si la mano derecha está en cualquier parte de la pantalla que no sea dentro de este espacio, el valor del desplazamiento aumenta, pero si encuentra dentro del espacio, disminuye.

Si la mano izquierda está "seleccionando" alguno de los ejes coordenados y la mano derecha está seleccionando si se quiere aumentar o disminuir el valor del desplazamiento, el sistema le asigna valores a la misma variable auxiliar usada para especificar el estado de la pinza, por lo que se logra que no haya modificaciones en el formato de codificación de los datos.

Teniendo en cuenta que hay tres ejes coordenados en los cuales se puede desplazar la punta del robot y que se puede tanto aumentar como disminuir el valor del desplazamiento, existen un total de seis valores posibles para la variable auxiliar, además de los valores del estado de la pinza, estos valores se pueden ver la siguiente tabla.

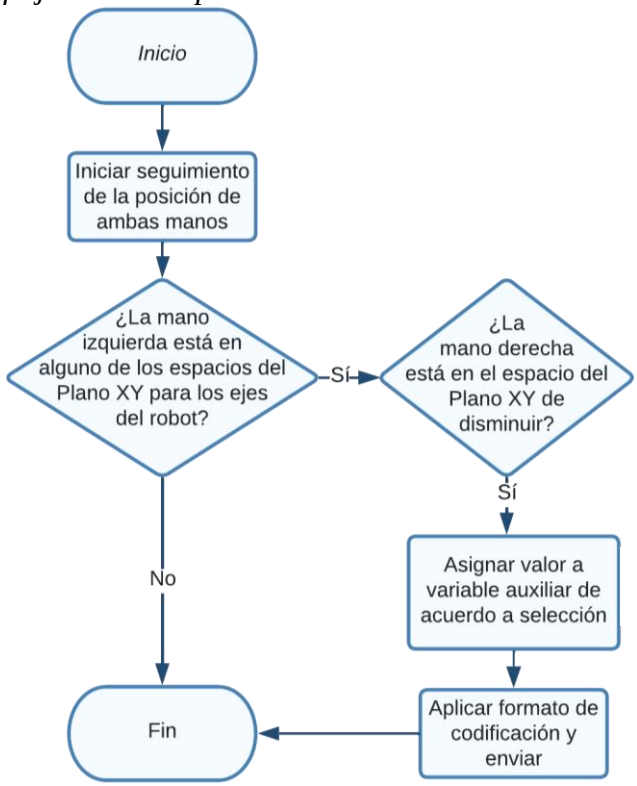
Tabla 4. *Variable auxiliar*

Descripción de selección	Valor
Valor de inicialización - pinza abierta	0
Pinza cerrada	1
Eje coordenado X y Avance	2
Eje coordenado Y y Avance	3
Eje coordenado Z y Avance	4
Eje coordenado X y Retroceso	5
Eje coordenado Y y Retroceso	6
Eje coordenado Z y Retroceso	7

Una vez asignado el valor de la variable se procede a enviar los datos al servidor, pero antes de esto, y siempre y cuando la variable auxiliar tenga un valor mayor a 1, los datos de las

posiciones en X, Y, y Z se establecen en “000”, debido a que estos nos son necesarios para el desplazamiento del efector en modo precisión.

Figura 53. *Diagrama de flujo del modo precisión en el sistema de visión*



4.5.3.6 Sistema robótico en el modo precisión. El sistema robótico para el modo precisión hace uso de tres variables de posición (posx, posy, posz) para desplazar cada uno de los ejes coordenados del efector final, estas se inicializan en cero al comenzar el programa y aumentan o disminuyen de acuerdo con los datos que se reciben del sistema de visión.

Al recibir los datos, se debe comprobar el valor de la variable auxiliar, y de acuerdo con esta, especificar las variables que harán mover el robot.

En el caso de que el valor de la variable auxiliar sea menor a 2, el modo de funcionamiento es el general, por lo que el sistema toma los valores que dependen de la posición de la mano derecha

en el sistema de visión; “XXX” para el desplazamiento a lo largo del eje X, “YYY” para el eje Y, y “ZZZ” para el Z.

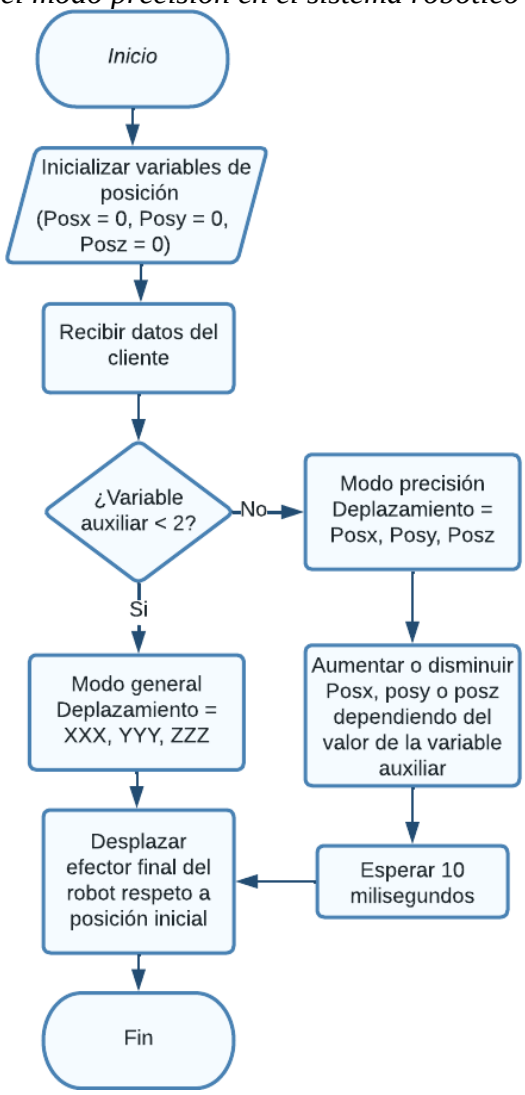
En el caso de que el valor de la variable auxiliar sea mayor a 2, el sistema asigna las tres variables de posición del modo precisión al valor del desplazamiento en milímetros de cada eje respecto al punto inicial. Dependiendo del valor de la variable auxiliar (selección del usuario) se aumenta o disminuye de uno en uno (milímetros) el valor de las variables de posición (tabla 3), y el efector final del robot se moverá de acuerdo con esto a una velocidad específica.

Para el caso de la velocidad a la que se desplaza la punta del robot, esta se define por medio de un retardo a la hora de aumentar o disminuir alguna de las variables de precisión. El retardo es de 10 milisegundos por cada ciclo en el que se le suma o resta un milímetro a alguna de las variables, por lo que la punta del robot recorre los ejes coordenados a una velocidad de 0.1 metros por segundo.

Tabla 5. *Variables de posición según variable auxiliar*

Valor de la variable auxiliar	Acción
2	posx = posx + 1
3	posy = posy + 1
4	posz = posz + 1
5	posx = posx - 1
6	posy = posy - 1
7	posz = posz - 1

Figura 54. *Diagrama de flujo del modo precisión en el sistema robótico*



4.5.4 Interfaz gráfica

Uno de los aspectos más importantes en una aplicación que se hace para interactuar con usuarios es la facilidad con la que estos pueden usarla y la velocidad con la que se adaptan a ella, para lograr esto, es recomendable hacer uso de elementos visuales que faciliten las interacciones.

La implementación de una interfaz gráfica permite visualizar de mejor forma los datos proporcionados por el sistema de visión, además permite entender muy fácilmente al usuario el modo de operación que está usando para el control del robot.

Para el desarrollo de la interfaz gráfica se usaron los principios del diseño de interfaces, como el usar menos elementos para comunicar más cosas al usuario o la implementación de paletas de colores que trabajen bien en conjunto.

Figura 55. *Diseño de la interfaz gráfica*

a. Paleta de colores b. Diseño preliminar

a. Tomado de [14]

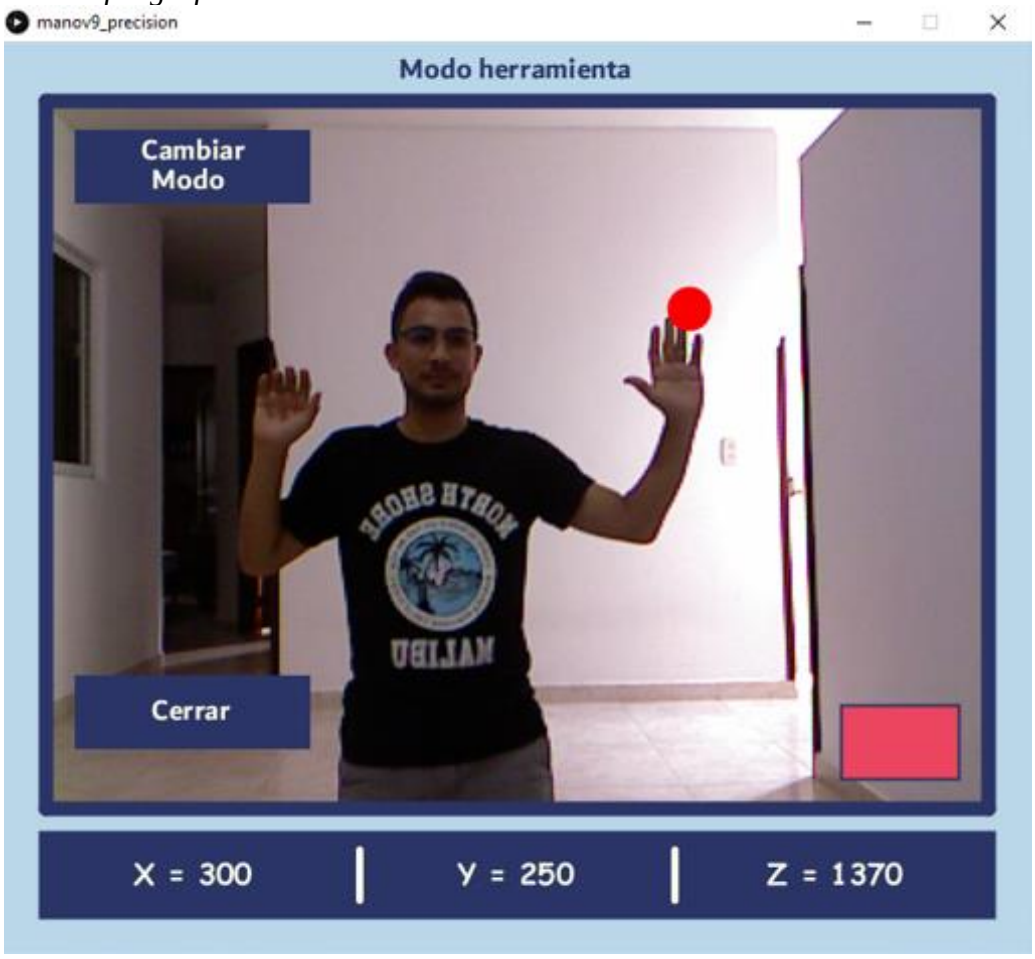
Después de probar con varios diseños preliminares se hicieron varios ajustes enfocados a mejorar la comodidad con la que el usuario usa los botones, hasta que finalmente se obtuvieron interfaces gráficas funcionales para ambos modos de trabajo.

4.5.4.1 Interfaz del modo herramienta. La interfaz gráfica en el modo herramienta mejora la visualización de los datos de posición de la mano derecha usando un diseño minimalista, agregando una barra horizontal que contiene los datos de posición y un indicador que cambia entre rojo y verde para indicar cuando se hace clic en los botones. Además de las mejoras en la

visualización de la información, se incluyen botones grandes y accesibles para cambiar al modo precisión y para cerrar el programa. El botón de cambio de modo hace uso de una variable exclusiva de la interfaz gráfica y que sólo tiene esta función, por lo que no se envía al servidor y no afecta el formato de codificación.

Para “hacer clic” en los botones de la interfaz, se usa el mismo gesto con el que se cierra y se abre la herramienta, por lo que el indicador de la parte inferior también sirve para mostrar el estado de la pinza en el sistema robótico. Toda la interfaz gráfica, incluyendo los botones y sus animaciones de selección, se realizó usando las funcionalidades que Processing trae consigo al ser un IDE centrado en el desarrollo de aplicaciones visuales.

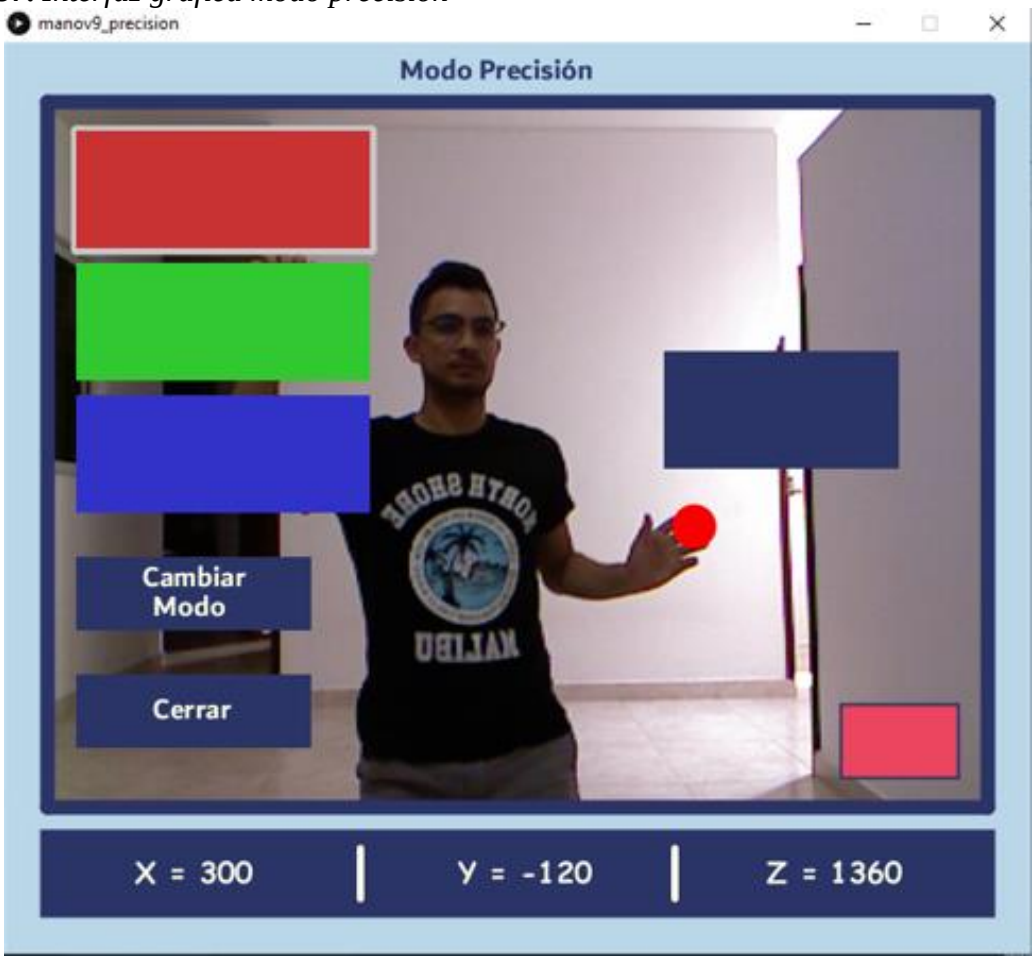
Figura 56. *Interfaz gráfica en el modo herramienta*



4.5.4.2 Interfaz del modo precisión. Para el caso del modo precisión, se usaron recuadros para marcar las áreas de selección de los ejes coordenados, y un recuadro para indicar al sistema robótico que se quiere disminuir el valor de la variable de desplazamiento, además de los mismos botones del modo herramienta, pero en otra distribución.

Para indicar el eje coordenado que se selecciona se prescindió de usar marcas textuales siguiendo el principio de diseño de interfaces que nos dice que menos es más y se usaron los colores convencionales que se usan en los ejes coordenados en cualquier programa con una interfaz tridimensional, es decir, rojo para el eje X, verde para él Y y azul para el Z.

Figura 57. *Interfaz gráfica modo precisión*

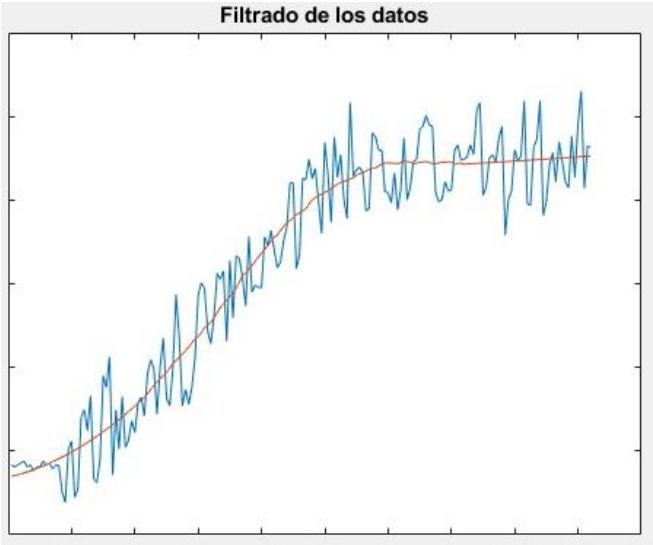


4.5.5 Filtrado de señal

La última mejora implementada en la interfaz natural surge luego de las primeras pruebas físicas, debido a una característica que no es fácil de notar en el entorno simulado. Al implementar la interfaz natural en el robot físico por primera vez se notó que cuando se dejaba la mano en una posición estática, aún seguía registrando pequeños cambios en su posición tridimensional, estos pequeños movimientos no representaban un problema en la funcionalidad del sistema, pero si una mejora muy recomendable debido a que el robot en físico traduce estos pequeños movimientos en constantes esfuerzos para los motores de la articulaciones que nunca se detenían en una posición estática si no que estaban en constante trabajo, aun con la mano del usuario detenida.

Para solucionar este problema se debe implementar un filtro que suavice los datos de posición de las manos que adquiere el Kinect, por lo que se usa un filtro de tipo media móvil, que es un tipo de filtro que suaviza una señal promediando los valores en un intervalo de tiempo o muestras determinado.

Figura 58. *Funcionamiento general del filtro de media móvil*



El filtro de media móvil toma un conjunto de valores de la señal y las promedia para generar un valor de salida, luego el filtro se mueve hacia adelante (toma el siguiente conjunto de valores) para generar un nuevo valor de salida, logrando así un suavizado continuo de la señal. El sistema de visión usa un filtro de media móvil con una ventana de 10 valores (eliminando un decimal de precisión) para cada una de las tres posiciones de la mano derecha antes de ser enviadas al servidor, logrando eliminar el ruido de los datos y solucionando de esta forma el inconveniente planteado.

5. Resultados

A continuación, se presenta un recuento y análisis de los resultados obtenidos en todas las etapas del proyecto, con el fin de determinar y complementar las conclusiones generales que se expondrán en el siguiente capitulo.

Se realizó una amplia investigación y se desarrollaron conceptos de las distintas áreas que en conjunto forman el objetivo general del proyecto, como lo son la visión la visión artificial y la robótica en el control servo visual.

Se investigaron varios proyectos relacionados tanto en el marco nacional como de forma global, logrando establecer una perspectiva general del desarrollo de este tipo de trabajos en el país, y aportando valor a los alcances y la justificación del proyecto.

Se estableció una metodología secuencial, compuesta por varias etapas centradas en los aspectos y componentes principales del proyecto para finalizar con la integración del todo el trabajo en el cumplimiento de los objetivos plateados.

Se desarrolló un algoritmo de visión por computadora que detecta las características visuales de los usuarios usando técnicas de seguimiento avanzadas como el skeleton tracking,

logrando reconocer no solo sus puntos articulares si no también desarrollar el procesamiento de gestos a partir de la interpretación de las posiciones de los puntos.

Se desarrolló un sistema de visión artificial usando software especializado gratuito para reconocer usuarios y usar sus gestos articulares en la codificación de instrucciones visuales que permiten controlar otros sistemas complejos y herramientas de software.

Se implementó un sistema robótico industrial y se usó software especializado para la creación de un algoritmo que permite recibir e interpretar información para el control de movimiento de un brazo robótico.

Se estableció comunicación entre un sistema el sistema de visión artificial y el sistema para el control del brazo robótico industrial usando protocolos de comunicación adecuados y creando en el proceso una capa de aplicación para el envío de datos codificados y la interpretación de datos recibidos.

Se desarrolló una interfaz natural de usuario que permite controlar la posición del efector final del brazo robótico industrial IRB120 usando gestos y movimientos con las manos por medio de un sistema de visión por computadora.

Se desarrolló un control gestual por medio las manos para una pinza mecánica que permita la manipulación de objetos y se implementó en el sistema general de control, además se desarrolló un conjunto de gestos para la implementación de un control especializado que permite al usuario controlar el movimiento del brazo robótico de forma precisa, logrando así agregar un apartado de modos de trabajo a la interfaz natural de usuario que faciliten el proceso de control servo visual.

Se diseñó una interfaz gráfica de usuario que mejora varios aspectos de la interfaz natural como la visualización de los datos y la interpretación de los modos de trabajo, además de agregar

botones y características que permiten navegar y comprender de mejor manera la interfaz, logrando un entorno de trabajo amigable y de fácil adaptación.

Figura 59. *Robot físico*

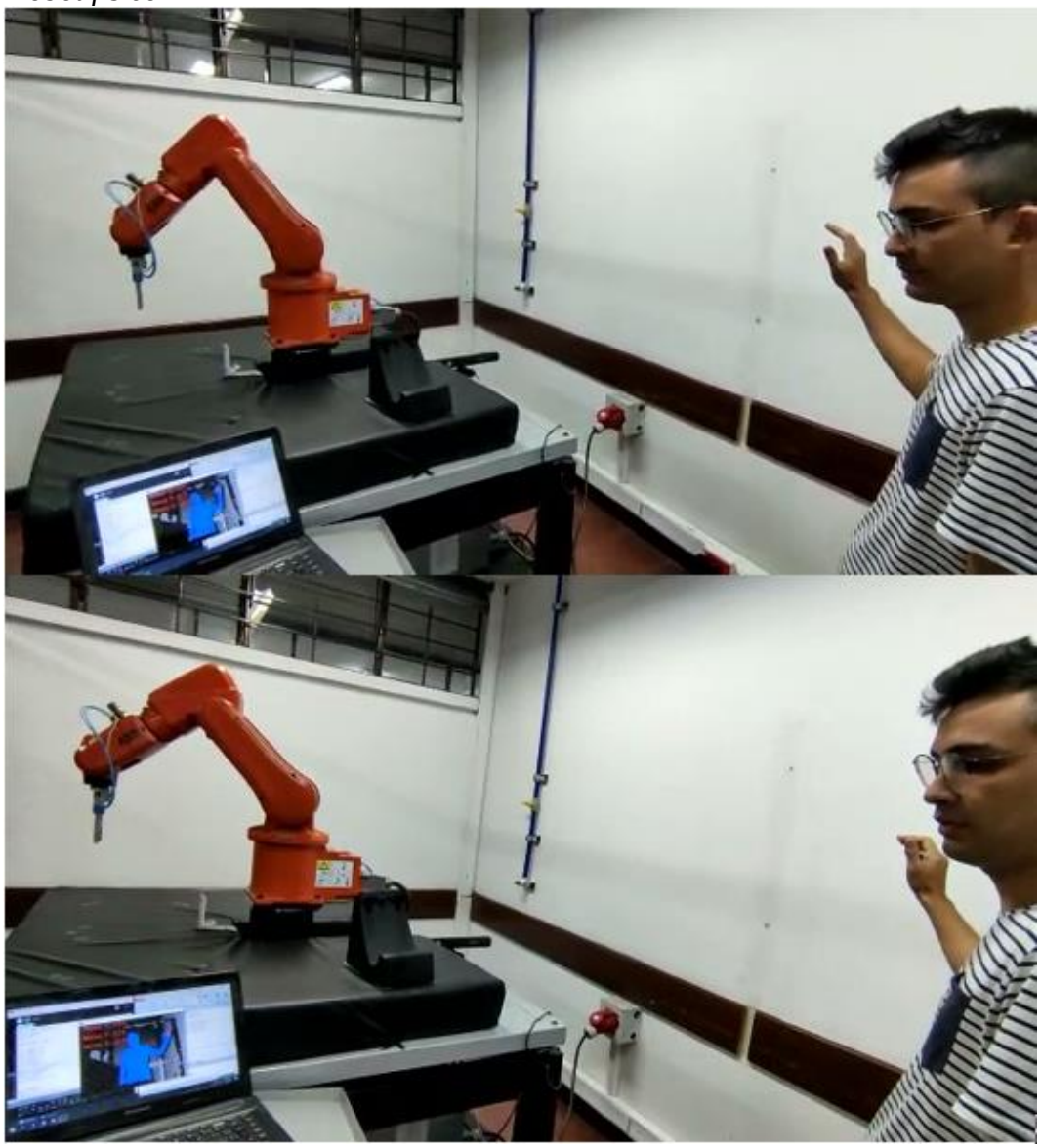


Figura 60. *Robot simulado*

6. Conclusiones

Se diseñó y construyó una interfaz natural de usuario que permite controlar el movimiento de un brazo robótico industrial. Para el desarrollo de esta interfaz natural de usuario se usó la cámara tridimensional comercial Microsoft Kinect y se desarrolló un algoritmo que permite codificar la información visual y los gestos una persona en instrucciones que se codifican en una

capa de aplicación y se envían para el control del brazo robótico industrial ABB IRB120, apto para la integración de herramientas y la manipulación de objetos.

Se realizaron pruebas tanto en entornos simulados como físicos y se implementaron mejoras a partir de observaciones en dichas pruebas, como la implementación de modos de trabajo, filtros para los datos y una interfaz gráfica intuitiva.

De acuerdo con los resultados, es posible responder a la pregunta planteada en la formulación del proyecto, y afirmar que el control visual y el uso de las interfaces naturales es una estrategia de control eficiente en el control de sistemas robóticos, y de acuerdo con las investigaciones realizadas en el marco tecnológico, afirmar que el control servo visual es un gran avance en la rama de la robótica y del control de sistemas en general, ya sean simples o complejos.

Gracias al desarrollo de este proyecto se pudieron aprender y afianzar de forma práctica muchos conceptos directamente relacionados con la carrera de ingeniería mecatrónica y se pudo establecer una perspectiva del avance nacional en el campo de la robótica y la visión por computadora.

Referencias

[1] R. Krishna, *Computer vision: Foundations and applications.* Stanford University, 2010.

[2] R. Szeliski, *Computer Vision: Algorithms and Applications.* Springer, 2020.

[3] A. Barrientos, *Fundamentos de robótica*, 2a ed. Madrid [etc.]: McGraw-Hill, 2007.

[4] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB®*, 2a ed. Springer Science & Business Media, 2017.

[5] A. F. Padilla Mantilla, "Control por guiado gestual para un robot antropomórfico basado en técnicas de visión artificial", Tesis de maestría, Universidad de Pamplona, Pamplona, 2021.

[6] J. C. Estefan Angel, "Arquitectura de telecontrol de un robot mediante el uso de interfaces gestuales", Tesis de maestría, Universidad EAFIT, Medellín, 2013.

[7] A. F. Silva Bohórquez, L. E. Mendoza y C. A. Peña Cortés, "Sistema de inspección y vigilancia utilizando un robot aéreo guiado mediante visión artificial", ITECKNE, vol.10, n.º 2, febrero de 2014. [En línea]. Disponible: https://doi.org/10.15332/iteckne.v10i2.421

[8] M. A. Moreno Armendariz, "Visión artificial estero con aplicación al control de un brazo de robot", Tesis doctoral, Instituto Politécnico Nacional, México, 2003.

[9] M. A. García, A. Cárdenas, J. M. Rendón, M. Maya Méndez. Una Plataforma de Control Basado en Visión para la Rehabilitación de Robots Manipuladores de Tipo Industrial. Computación y Sistemas [en linea]. 2009, 12(4), 409-420. ISSN: 1405-5546. Disponible en: https://www.redalyc.org/articulo.oa?id=61513249004

[10] K. Wiens. "Reparación de Microsoft Kinect". Ifixit. https://es.ifixit.com/Device/Microsoft_Kinect (accedido el 16 de marzo de 2023).

[11] Processing. https://processing.org/.

[12] “Product Manual IRB120”, ABB Robotics., Auburn Hills, Michigan, USA. [En linea]. https://search.abb.com/library/Download.aspx?DocumentID=3HAC035728001&LanguageCode=en&DocumentPartId=&Action=Launch

[13] “Microsoft Kinect”, Microsoft., USA. [En linea]. Disponible en: https://msdn.microsoft.com/en-us/library/hh973074.aspx

[14] Color Hunt, “Paleta de colores”, [En linea] Disponible en: https://colorhunt.co/palette/eb455ffcffe7bad7e92b3467

[15] K. Kozłowski, Robot Motion and Control 2011. London: Springer London, 2012.

[16] S. Hutchinson, G. D. Hager y P. I. Corke, "A tutorial on visual servo control", IEEE Trans. Robot. Automat., vol. 12, n.º 5, p. 651–670, 1996. Accedido el 7 de febrero de 2023. [En línea]. Disponible: https://doi.org/10.1109/70.538972

[17] A. Singhvi y N. Tarkhan-, "Integrated Occupant Motion Sensing with Real-Time CFD Simulation as a Design Assessment Tool", en 2017 Building Simul. Conf. IBPSA, 2017. Accedido el 7 de febrero de 2023. [En línea]. Disponible: https://doi.org/10.26868/25222708.2017.648