UNIVERSIDADE DA CORUÑA

Escola Politécnica de Enxeñería de Ferrol

Trabajo Fin de Grado

CURSO 2022/23

IMPLEMENTACIÓN Y PUESTA EN MARCHA DE UN ENTORNO ROBOTIZADO PARA LA REALIZACIÓN DE ENSAYOS DE INTERACCIÓN DINÁMICA

Grado en Ingeniería en Tecnologías Industriales

ALUMNA/O

Alejandro Martínez Chao

TUTORAS/ES

Richard Duro Fernández

Alejandro Romero Montero

FECHA

DICIEMBRE 2022

TÍTULO Y RESUMEN

Implementación y puesta en marcha de un entorno robotizado para la realización de ensayos de interacción dinámica.

Este Trabajo de Fin de Grado tiene como fin la implementación de un entorno robotizado para la realización de ensayos de interacción dinámica. Durante el proceso se analizará tanto el entorno, como las mejores opciones para poner en funcionamiento dicho dispositivo en la industria.

El objeto de estudio de esta investigación es la parada de un objeto en movimiento con un brazo robótico *UR5e de Universal Robots* y cámaras *OptiTrack*. El análisis de riesgos obtenido mediante la normativa de estandarización *ISO* tendrá como fin una puesta en marcha segura y eficaz. Dicho análisis unido a una buena elección de hardware y software tendrá como resultado una serie de pautas que servirán para complementar el estudio.

El trabajo estará dividido en once capítulos, junto con la introducción y las conclusiones.

TÍTULO E RESUMO

Implementación e posta en marcha dun entorno robotizado para a realización de ensaios de interacción dinámica.

Este Traballo de Fin de Grado ten como fin a implemtación dun entorno robotizado para a realización de ensaios de interacción dinámica. Durante o proceso analizarase tanto o entorno, como as mellores opcións para poner en funcionamento dito dispositivo na industria.

O obxetivo do estudo desta investigación é a parar un obxecto en movemento cun brazo robótico *UR5e* de *Universal Robots* e cámaras *OptiTrack*. O análisis de riscos obtido mediante a normativa de estandarización *ISO* terá como fin unha posta en marcha segura e eficaz. Dito análisis unido a unha boa elección de *hardware* y *software* terá como resultado unha serie de pautas que servirán para complementar o estudo.

O traballo estará divido en once capítulos, xunto coa introducción e as conclusións.

TITLE AND SUMMARY.

Implementation and set up of a robotic environment for dynamic interactions tests.

This End of Degree Work aims to implement a robotic environment for conducting dynamic interaction tests. During the process, both the environment and the best options for putting the device into operation in the industry will be analysed.

The object of study of this research is the stopping of a moving object with a *UR5e* robotic arm of Universal Robots and *OptiTrack* cameras. The risk analysis obtained by means of the *ISO* standardization standard shall be aimed at a safe and effective implementation. This analysis together with a good choice of hardware and software will result in a series of guidelines that will serve to complement the study.

The work will be structured into eleven chapter, along with the introduction and conclusions.

UNIVERSIDADE DA CORUÑA

Escola Politécnica de Enxeñería de Ferrol

TRABAJO FIN DE GRADO

CURSO 2022/23

IMPLEMENTACIÓN Y PUESTA EN MARCHA DE UN ENTORNO ROBOTIZADO PARA LA REALIZACIÓN DE ENSAYOS DE INTERACCIÓN DINÁMICA

Grado en Ingeniería en Tecnologías Industriales

Documento

MEMORIA

Índice de contenido

ÍNDICE DE FIGURAS

1 OBJETIVOS

El proyecto tiene como objetivo la implementación y puesta en marcha de un entorno robotizado para la realización de ensayos de interacción dinámica, el cual se centrará en los siguientes puntos para conseguir un análisis óptimo y fiable:

- El estudio de un problema real con interacciones dinámicas en la industria, el cual se solucionará con el diseño de un entorno robotizado.
- Escoger el *hardware* y *software* que mejor se adapten a las necesidades en función de sus características. En su mayor parte se utilizan métodos de *software* libre para poder ponerlos en marcha de forma más sencilla en las futuras instalaciones.
- Montaje y funcionamiento del *hardware* manteniendo la calidad de la normativa de estandarización *ISO*.
- Un análisis de prevención de riesgos del entorno robotizado en función de las normas *ISO* para una puesta en marcha y un funcionamiento seguro.
- Implementación del *software* necesario para la operativa del sistema.
- Realizar una tarea para comprobar la funcionalidad de lo ejecutado, y poder medir de una forma experimental su actividad.
- Un examen de ventajas, desventajas y capacidades. También se analizarán sus limitaciones y potencial futuro sobre la implantación de un entorno robotizado en función de los datos y muestras recopiladas durante la realización de la tarea dinámica.

.

2 ESTUDIO Y ANÁLISIS PREVIO

El presente trabajo de fin de grado busca la implementación de un entorno robotizado para la realización de ensayos de interacción dinámica. Esto se ha convertido en uno de los aspectos básicos dentro de las empresas, ya que robotizar ciertas tareas ayuda a reducir costes y a estandarizar el proceso industrial. Hasta la fecha, la mayoría de las funciones de un robot se han ejecutado en un entorno no variable con movimientos predeterminados o estandarizados. En este proyecto se implementarán estas funciones en situaciones donde los movimientos del entorno sean dinámicos.

Para la puesta en funcionamiento se han contado con diversas herramientas básicas: un brazo robótico, un sistema sensorial y uno o varios ordenadores para la configuración de estos.

En este caso, nuestro principal objetivo es realizar lo mencionado en un contexto educativo y de manera experimental. En un futuro se podría seguir estudiando dicha rama y sacarlo al mercado.

2.1 Robótica

La *RAE* facilita la siguiente definición del término *robótica*: «técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales» [1]. Es decir, esta tecnología engloba diferentes vertientes de la ingeniería, como la electrónica o la mecánica, así como las ciencias de la computación.

Como se ha mencionado antes, actualmente la robótica es una parte fundamental en la mayor parte de los procesos industriales. La robótica va unida a dispositivos capaces de materializar un deseo humano, o desde el punto de vista de una empresa, que sean idóneos para realizar trabajos tediosos o peligrosos para el operario.

En la siguiente gráfica de datos se puede apreciar que en la actualidad, la industria de la robótica está creciendo a un ritmo muy elevado:

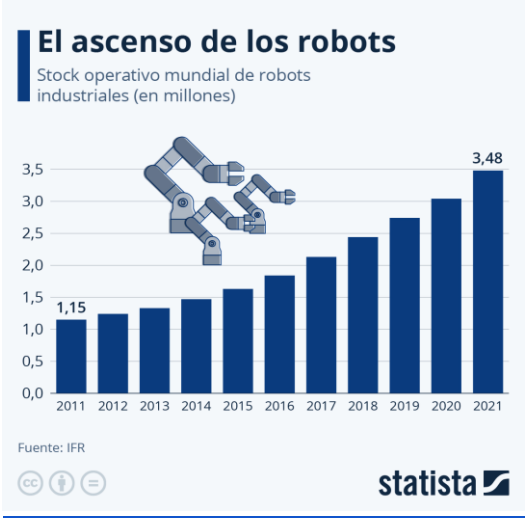

Figura 1. Grafica del crecimiento de la robótica en la industria. Fuente [2]

Este crecimiento se debe a la mejora en dicha rama, haciendo que los robots sean cada vez más fiables, precisos y rápidos. Lo que provoca un aumento en el rendimiento del proceso y en el apartado del riesgo lo disminuye al no tener que realizarlo un operario.

3 NECESIDADES DEL SISTEMA.

El objetivo principal es la realización de interacciones dinámicas. Una iteración dinámica es definida como [3]: la modificación que sufre el movimiento de un cuerpo por la aplicación de una fuerza externa a él. Por lo tanto, para cumplir este objetivo se pensó en un brazo robótico que tuviera la capacidad de parar un objeto en movimiento lanzado por un ser humano. A continuación, se definen los principales actuadores y sensores de nuestra implementación, así como sus características básicas para poder cumplir con el objetivo planteado.

Como sensor se optaron por las cámaras de seguimiento, y para el presente trabajo se escogió la marca *OptiTrack*. La elección de este tipo de cámaras se fundamentó en la necesidad de tener la posición continua del objeto a parar, por lo tanto, se precisaba una cámara que devolviera la posición lo más rápido y preciso posible, ya que esto serviría para poder predecir la posición final del elemento. De este modo se obtuvieron más datos, y se ha conseguido con una mayor exactitud la trayectoria que sigue el cuerpo.

Como actuador se ha empleado un brazo robótico con la finalidad de parar el objeto, para ello se necesitó un brazo más rápido y exacto, pues debe moverse a la posición indicada en el menor tiempo posible. Teniendo en cuenta las necesidades, se optó por la utilización de un brazo que cuente con un sistema de impulsión eléctrico, uno de los más precisos en el mercado actualmente.

4 HARDWARE

Los dispositivos utilizados para el montaje e implementación del sistema robótico han sido un brazo tipo *UR5e* de *Universal Robots* y un conjunto de cámaras *OptiTrack*.

4.1 *UR5e*:

Se trata de un brazo colaborativo fabricado por *Universal Robots*. En esta ocasión se ha empleado dentro del campo de la investigación, pero estos dispositivos tienen diversos usos: entretenimiento, construcción o automatización industrial.

Para este trabajo se ha utilizado un robot de tercera generación, ya que tiene la capacidad de analizar el entorno y de realizar movimientos específicos pudiendo cumplir con el objetivo. En este caso su función es detener un objeto en movimiento.

Este brazo robótico cuenta con un sistema eléctrico para la impulsión del mismo, ya que se le proporciona energía eléctrica para realizar los desplazamientos. Gracias a esto, el *UR5e* tiene una mayor exactitud, algo que se podrá visualizar en las especificaciones. Sin embargo, no se conseguirá la misma velocidad o potencia que facilitan otros tipos, por ejemplo los de impulsión hidráulica.

El modelo que se ha escogido cuenta con las siguientes características [4]:

Carga útil	5kg
Alcance	850mm
Grados de libertad	6 articulaciones giratorias
Repetibilidad	±0.03mm
Velocidad de junta	180º/s

Este modelo se trata de un robot articulado que es empleado para diversos tipos de operaciones, las más frecuentes son: ensamblaje, fundición a presión, máquina de desbardado, soldadura, pintado a *spry* o *pick and place*.

Los componentes del brazo suelen ser los siguientes:

- Actuadores. Se utilizan para trabajar sobre el entorno, es decir, modificar posiciones, realizar movimientos, etc.
- Sensores. En este brazo fueron empleados para conocer la posición del robot.
- Sistema de control. En este caso se cuenta con una caja de control descrita en el montaje. La *tablet* de control llamada *Polyscope* se puede utilizar para movimientos sencillos y controlar el brazo, y por último como implementación *ROS*.

Concluyendo, este dispositivo es el que mejor se adapta al objetivo del trabajo de fin de grado.

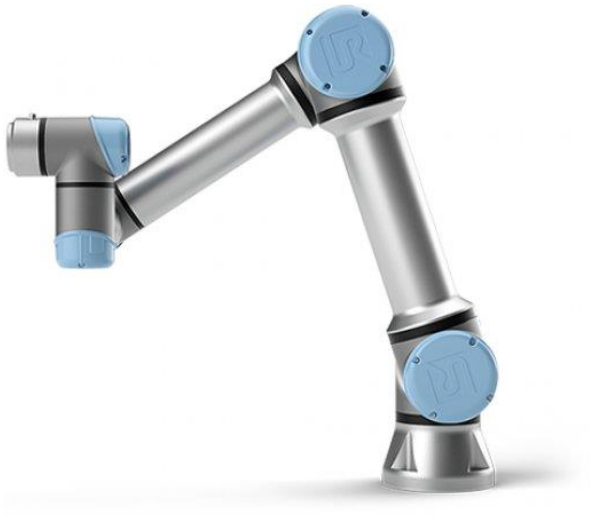

Figura 2. Imagen del brazo robótico UR5e. Fuente [5]

4.2 Cámaras tipo *OptiTrack*:

Se ha incorporado cámaras *OptiTrack*, en este caso *OptiTrack Primex 13W*. Estas tienen [6] una alta velocidad, poca resolución (1.3MP), pero una gran precisión ±0.3 mm en 3D. También cuentan con una alta tasa de velocidad nativa 240FPS, así como con una velocidad máxima de fotogramas de 1000FPS.

Estas cámaras denominadas *cámaras de seguimiento*, son las que más se ajustan al proyecto. Gracias a ellas se tiene un continuo flujo de datos de la posición del objeto, con una devolución alrededor de unos 1000 puntos de posición cada segundo, o lo que es lo mismo, se devolverá un dato cada milisegundo. Esta ventaja es realmente beneficiosa, ya que ayuda a realizar cálculos a gran velocidad, pudiendo pronosticar una futura posición de nuestro objeto y mover el brazo robótico lo antes posible al lugar de su destino. Para llevar a cabo, estes seguimientos se utiliza la detección de marcadores reflectantes mediante una luz infrarroja. Además, las propias cámaras traen un *software* de gran ayuda del que se hablará más adelante.

Por los motivos expuestos, se escogen las *primex 13W* por encima de otras cámaras a causa de que se necesitó una tasa de velocidad muy alta para obtener los valores actuales del objeto, y mover rápidamente el brazo hasta la posición deseada.

Figura 3. Imagen de la *OptiTrack Primex 13W*. Fuente [6]

4.3 Alternativas:

Para el brazo robótico se pueden tener varias alternativas, pero las más interesantes son:

KUKA LBR iiwa7 R800 [7]:

Carga útil	7kg
Alcance	800mm
Grados de libertad	7 articulaciones giratorias
Repetibilidad	±0.1mm
Velocidad de junta	180º/s

Figura 4. Imagen del *KUKA LBR iiwa7 R800*. Fuente [8]

Como se puede observar, este robot es menos adecuado que el utilizado. Tiene un menor alcance y una precisión de 0.08 mm menor al UR5e.

KUKA LBR iiwa7 R820 [7]:

Carga útil	14kg
Alcance	820mm
Grados de libertad	7 articulaciones giratorias

Repetibilidad	±0.15mm
Velocidad de junta	135º/s

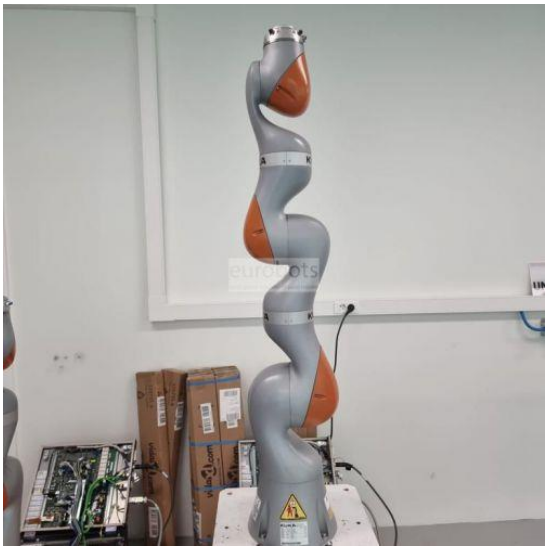

Figura 5. Imagen del KUKA LBR iiwa7 R820. Fuente [7]

En la tabla se puede apreciar que las características son inferiores al escogido para la tarea. Esta inferioridad en la velocidad y en la precisión, hace que resulte una peor opción para nuestro objetivo.

UR3 [9]:

Carga útil	3kg
Alcance	500mm
Grados de libertad	6 articulaciones giratorias
Repetibilidad	±0.1mm
Velocidad de junta	180º/s

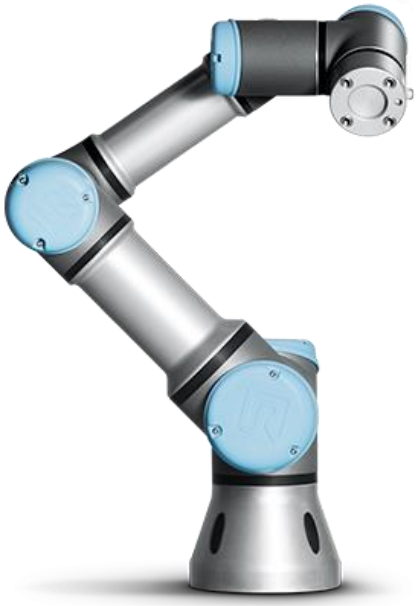

Figura 6. Imagen del robot *UR3*. Fuente [10]

Tiene un menor alcance y repetibilidad, lo que hace que sea un robot menos adecuado que el UR5e en cuanto a características.

Por los motivos expuestos con anterioridad, la mejor opción entre las que se han analizado es el robot UR5e.

En las cámaras no se tienen alternativas fuera de *OptiTrack*, por lo que se puede hacer una elección dentro de las propias opciones de la marca:

Primex 41 [6]:

Resolución	4.1MP
Precisión	±0.1mm
Tasa de velocidad nativa	180 FPS
Velocidad máxima de fotograma	250 FPS

Figura 7. Imagen de la *OptiTrack Primex 41*. Fuente [6]

Es una muy buena cámara de alta precisión, pero no presenta una buena velocidad de fotograma como *Primex 13W*. El fabricante también expone que está diseñada para trabajar a gran escala, sin embargo, en este proyecto se han realizado los experimentos en un laboratorio.

Primex 22 [6]:

Resolución	2.2MP
Precisión	±0.15mm
Tasa de velocidad nativa	360 FPS
Velocidad máxima de fotograma	500 FPS

Figura 8. Imagen de la *OptiTrack Primex 22*. Fuente [6]

Para esta cámara los motivos se repiten con respecto a la *Primex 41*: tiene la mitad de tasa de fotogramas que la empleada, lo que hace que la frecuencia con la que se reciben los datos también se reduzca a la mitad.

Primex 13 [6]:

Resolución	1.3MP
Precisión	±0.20mm
Tasa de velocidad nativa	240 FPS
Velocidad máxima de fotograma	1000 FPS

Figura 9. Imagen de la *OptiTrack Primex 13*. Fuente [6]

Por características parece una cámara mejor que la seleccionada, no obstante, el fabricante expone que se utilizará para áreas medianas.

Slimx 13 [6]:

Resolución	1.3MP
Precisión	±0.3mm
Tasa de velocidad nativa	240 FPS
Velocidad máxima de fotograma	1000 FPS

Figura 10.Imagen de la *OptiTrack Slimx 13*. Fuente [6]

La *Slimx 13,* como se puede observar, tiene las mismas características que la *Primex 13W*, pero solo recoge datos de marcadores activos, y en el proyecto solo se utilizan marcadores pasivos, por lo que es necesario descartar esta cámara.

Debido a todos estos motivos se ha escogido la cámara Primex 13W por encima de las demás opciones que nos proporciona la marca.

5 INSTALACIÓN

5.1 *UR5e*:

En el caso del brazo robótico se siguieron las instrucciones que contiene el manual de usuario que viene con el propio *UR5e* [4], concretamente las instrucciones de interfaz mecánica descritas en el punto 4.

Además, se añadieron planos de restricciones para evitar futuras colisiones. En el proyecto, fue necesario añadir un plano de la mesa para evitar choques con la misma, y otro plano en la parte trasera del robot para evitar golpes contra la jaula o algún operario. La captura del objeto debe realizarse como máximo en la superficie del escritorio, coincidiendo con el máximo rango que presenta este brazo.

El brazo robótico adquirió la siguiente forma una vez montado:

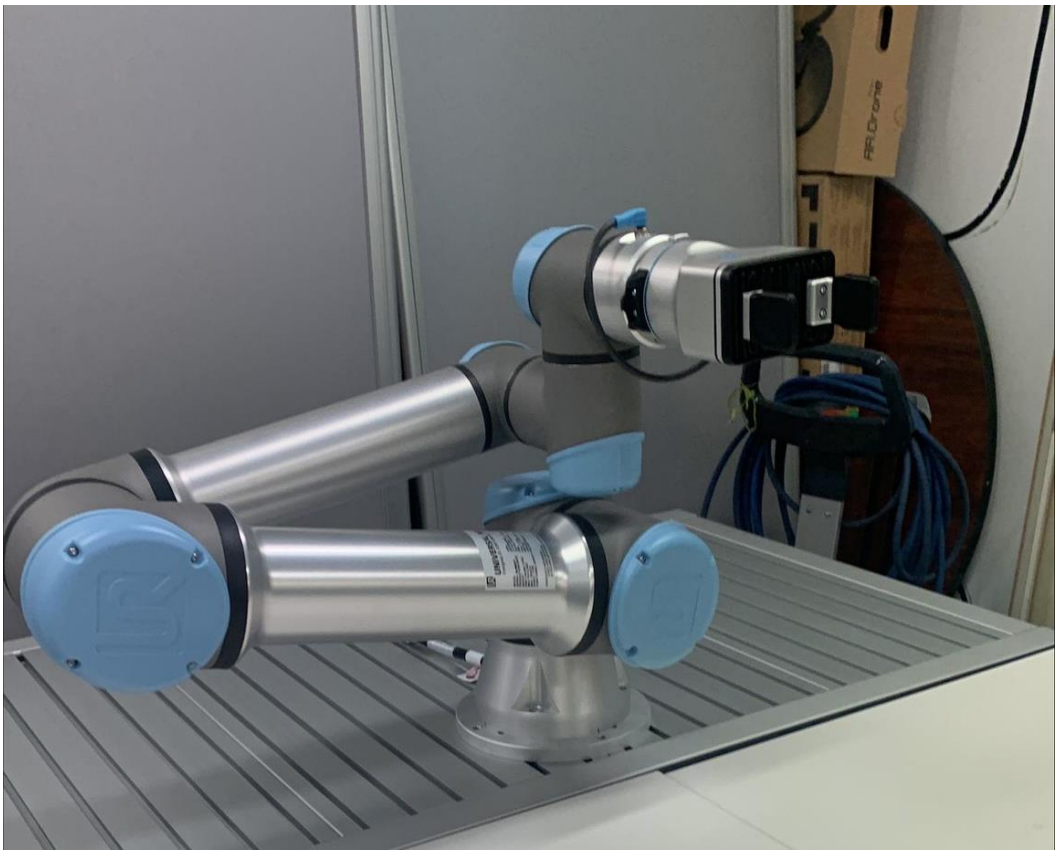

Figura 11.Brazo robótico UR5e recién montado. Fuente [Elaboración propia]

Tras el montaje, se conectó el robot a la caja de control, y esta a la red. Finalmente, se procedió a la conexión a internet mediante la unión de la caja de control y un cable de *Ethernet* al *router*.

5.2 Optitrack

Lo primero a realizar para el montaje de las cámaras fue la preparación del área. Deben evitarse las superficies reflectantes, ya que las luces IR de las cámaras podrían reflejarse en ellas e interferir en el seguimiento. En segundo lugar, no son aptas las superficies flexibles o deformables, pues estas pueden afectar a la calibración de las mismas. Por último, tampoco es admisible cualquier obstáculo que bloquee la visión de las cámaras y evite el seguimiento del objeto en consecuencia.

Otro aspecto importante en el montaje es la colocación de las cámaras [11]:

- Para la captura óptima del movimiento, la compañía recomienda situar las cámaras a gran altura, ya que de esta forma se maximiza la cobertura de imagen de volumen y se minimizan las posibilidades de que se produzcan choques contra las cámaras.
- Se aconseja una distancia constante entre las cámaras, pues cuando dos de ellas se encuentran muy cerca, una no será funcional. Aquí estaríamos ante un problema de superposición, por lo que se produciría una carga computacional innecesaria.
- La distancia con el objetivo influye en dos parámetros: cuanto más alejadas de él, se contará con una mayor cobertura de cámara y una mayor configuración de volumen, pero una menor precisión de datos. Mientras que cuanto más cerca se encuentren las cámaras del objeto, se tendrá el efecto contrario: una mayor precisión, pero una menor cobertura de estas.

Para proceder a la conexión de las cámaras con el ordenador en el que se ha instalado previamente el programa *Motive,* se empleó un cable de *ethernet* de categoría 6 que, además, no debe exceder los 100 metros. El cable escogido no solo ofrece velocidad de transferencia de datos, sino que también es una fuente de alimentación para las propias cámaras. En este caso, todas las *OptiTrack* fueron vinculadas a un *switch POE*, que está conectado a su vez al ordenador.

Por consiguiente, tanto la configuración de las cámaras, como la disposición de cable *Ethernet,* quedó diseñada de la siguiente manera:

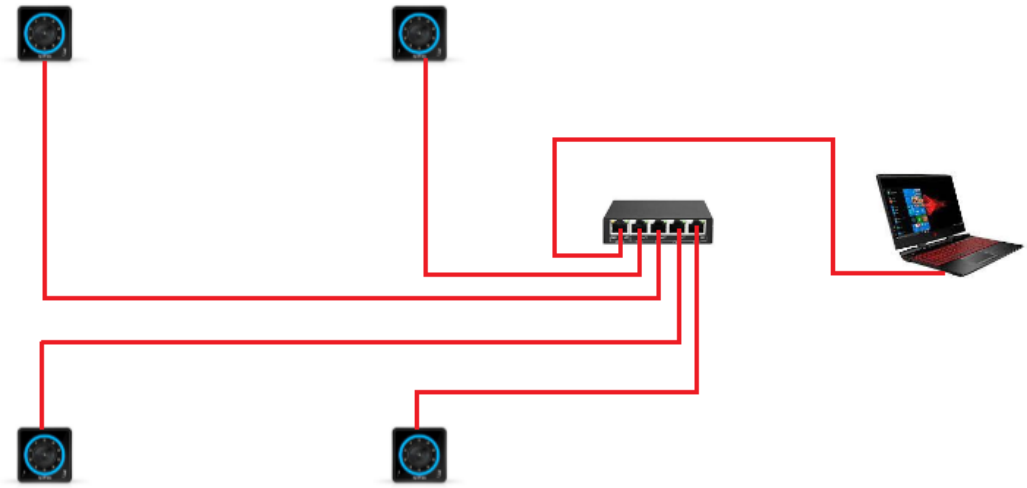

Figura 12. Configuración de las cámaras y distribución de las conexiones. Fuente [Elaboración propia]

Posteriormente, las características de nuestro ordenador deben ser [12]:

- *Windows 10*
- *CPU*: *Intel i7* o mejor.
- *RAM*: 16GB o mejor.
- *GPU*: *GTX 1050* o superior con controladores más recientes.

La puesta en marcha del *software* se ha realizado mediante los pasos de la página oficial de *OptiTrack*: https://docs.optitrack.com/motive/installation-and-activation .

Son necesarios un mínimo de 3 marcadores en el objeto a rastrear para que las cámaras puedan realizar un buen seguimiento del objeto.

Figura 13. Marcador para realizar el seguimiento de objeto. Fuente [13]

En el presente trabajo se han colocado un total de 6 marcadores de la siguiente manera:

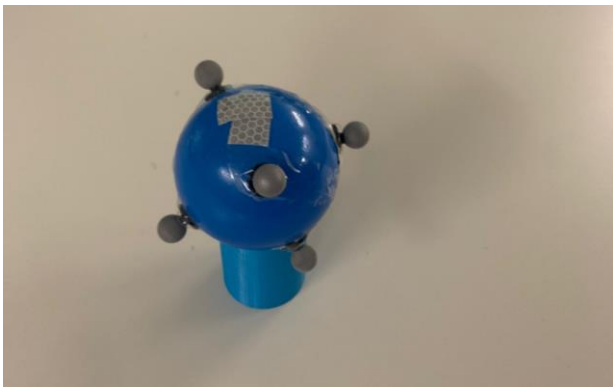

Figura 14. Objeto con sus marcadores. Fuente [Elaboración propia]

En este caso, la cinta reflectante no es una decisión apropiada, ya que con el roce de la mesa y el paso del tiempo esta se acabaría desgastando y las cámaras dejarían de detectarla.

Se ha descartado el uso de otro modelo de cámara debido a que durante su funcionamiento no ha presentado ningún error. Añadir otro tipo de cámara simplemente produciría un aumento de coste sin beneficio, ya que no aporta nada a la funcionalidad del sistema.

5.3 Zona de trabajo

La zona de trabajo tiene la siguiente estructura:

Figura 15. Zona de trabajo. Fuente [Elaboración propia]

La zona principal donde se realizan los movimientos son las dos mesas.

5.4 Zona de Control

La zona de control se mostrará en la siguiente imagen:

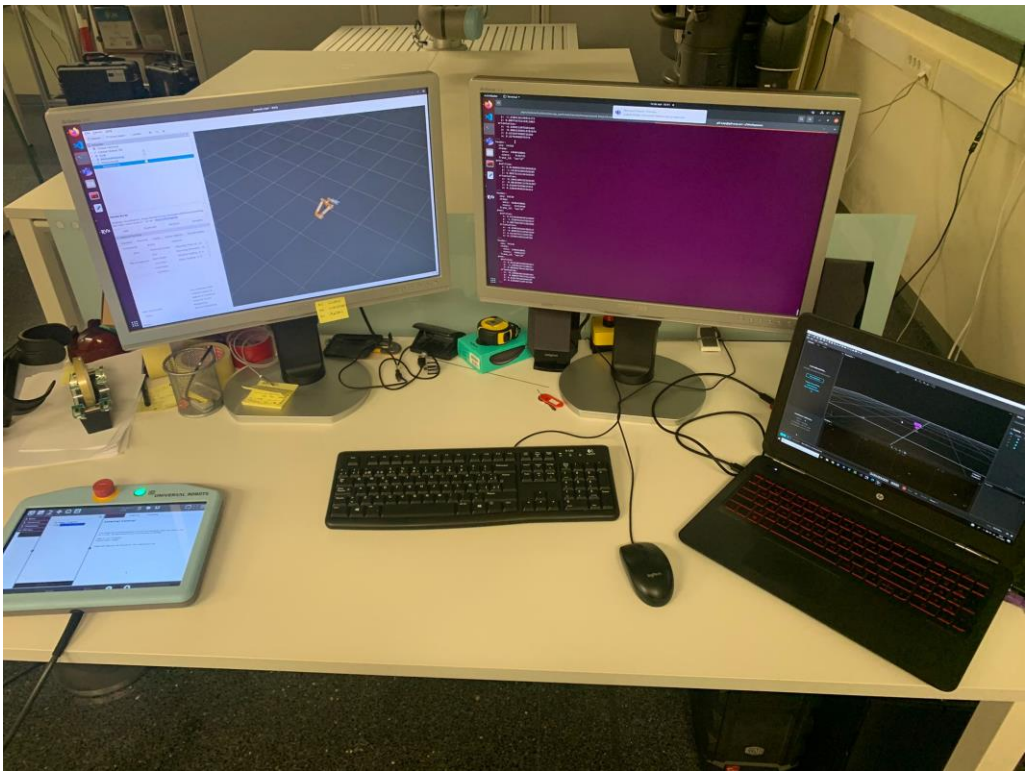

Figura 16. Zona de control. [Elaboración propia]

Está constituida por:

- Un ordenador con el sistema operativo *Linux* y dos monitores en los que se puede visualizar tanto la actual posición del robot, como la posición del objeto mediante datos.
- Otro ordenador con el sistema operativo *Windows,* donde se puede observar la posición del objeto visualmente, mediante el programa *Motive*.
- Una tableta de control del *UR5e*.

6 ANÁLISIS DE PREVENCIÓN DE RIESGOS

Con el fin de prevenir riesgos en el uso del entorno robotizado y del *UR5e*, primero se ha realizado un análisis de los peligros que tiene el robot respetando las normativas de estandarización y de prevención: *ISO 12100, ISO 10218-2*.

Teniendo en cuenta la normativa ISO10218-2 [14]: «es necesario identificar los peligros y evaluar los riesgos asociados con el robot y sus aplicaciones antes de seleccionar y diseñar las correspondientes medidas de seguridad para reducir adecuadamente los riesgos». Se procederá a la explicación de la normativa, así como la descripción sobre cuál es su función para evaluar posteriormente sus posibles riesgos.

La finalidad del robot es la detención de un objeto en movimiento, por lo que deberá desplazarse a alta velocidad.

Los principales riesgos que presenta el entorno, siguiendo el 4º punto de la normativa *ISO10218-2* [15], son los siguientes:

- Movimientos con alto par, velocidad y aceleración realizados por el brazo. Esto pueden provocar colisiones contra operarios u otros elementos que estén a su alcance.
- Altas temperaturas tanto del brazo robótico como de la caja de control.
- Tensiones y corrientes elevadas en la caja de control.
- Fallos o errores en el robot.
- Modificación del sistema provocando desajustes en el mismo
- Falla en los elementos de seguridad.
- Un mal mantenimiento del *UR5e*.

Para el diseño de la instalación de los sistemas de seguridad ha de seguirse el punto 5 de la normativa *ISO10218-2* [15], donde se muestran los diferentes elementos a tener en cuenta. En consecuencia, se dispondrán las siguientes prevenciones:

- Una función de parada de emergencia que será dispuesta mediante una seta disponible en la tableta de control.
- Un apagado de equipamiento asociado y procedimiento de recuperación de emergencias, los cuales trae de serie el *UR5e* en su *software* de control.
- Señales de aviso que ya vienen integradas en *PolyScope*.
- Una protección perimetral a la zona de trabajo del brazo según la normativa *ISO 13857*.
- Se procederá a la instalación de una celda para el brazo, por lo que se podrán controlar los peligros desarrollados por el movimiento con alta velocidad y par del mismo, siguiendo la normativa *ISO 12100:2010* punto 6.3.3.1.
- El robot deberá estar completamente apagado para proceder a la entrada en la celda de operaciones, debido a prácticas de mantenimiento o cualquier otro procedimiento, pues de esta manera se evitarán posibles daños o colisiones contra el operario.
- Señalización para evitar quemaduras, descargas o zonas peligrosas.

Figura 17. Señal de peligro por temperatura. Fuente [16]

Figura 18. Señal de peligro por corriente eléctrica. Fuente [17]

Figura 19. Señal de peligro. Fuente [18]

- Durante su funcionamiento, no se debe abrir ni tocar la caja eléctrica, ya que al calentarse puede provocar quemaduras o descargas eléctricas.
- No tocar el brazo robótico inmediatamente después de su uso debido a que puede causar quemaduras por un calentamiento de este durante su funcionamiento.
- Uso obligatorio de casco dentro de la celda o cerca de la zona de operación del brazo para minimizar los daños en caso de ser golpeado por el robot.

Figura 20.Señal de uso obligatorio de casco. Fuente [19]

- Uso obligatorio de guantes para realizar el trabajo en la caja de control, pues funcionan como un aislante para trabajos con electricidad.

Figura 21.Señal de uso obligatorio de guantes. Fuente [20]

7 SOFTWARE

7.1 *Motive*:

Figura 22. Logo de *Motive*. Fuente [12]

Este programa se utiliza para las cámaras *OptiTrack* con la finalidad de facilitar las operaciones. Su función es procesar los datos recibidos por las mismas y ofrecer tanto las posiciones, como la orientación de los objetos en el espacio, gracias a la identificación de los marcadores implementados en el objeto.

Es necesaria una primera calibración [12] empleando una escuadra y una vara, el proceso de calibración es explicado en el anexo del trabajo. A continuación, el programa *Motive* se calibra automáticamente con los datos recopilados durante su uso habitual, no se degrada con el tiempo, exceptuando un cambio brusco del entorno en el que se trabaja.

Motive ofrece la posibilidad de customizar las salidas para el trabajo, en este caso, se transmitirán en tiempo real a *ROS* para poder utilizarlas en conjunto con el brazo.

7.2 *PolyScope*:

PolyScope es un entorno de desarrollo gratuito basado en *Linux* y se utiliza como la interfaz de usuario que trae el *UR5e* en una consola portátil.

Figura 23. Captura de pantalla del inicio de *PolyScope*. Fuente [21]

En ella se pueden realizar varias tareas como [4]:

- Manejar el brazo robótico y la caja de control.
- Ejecutar y crear programas del robot.
- Mostrar fallos o posibles errores.
- Introducir funciones de restricción de movimientos y propiedades del *UR5e*.

- Establecer conexión con *ROS* para lanzar programas.
- Ejecutar un movimiento libre.
- Conocer el posicionamiento del brazo en el espacio en todo momento.
- Realizar una parada de emergencia mediante una seta.

7.3 *ROS*:

Figura 24. Logotipo del sistema operativo de *ROS*. Fuente [22]

El *software* principal del proyecto es *Robot Operating System (ROS)* [23], un sistema desarrollado en 2007 por el Laboratorio de Inteligencia Artificial de Stanford. *ROS* es un *software* libre con una licencia que permite libertad de uso comercial e investigadora.

A pesar de no ser un sistema operativo como tal, sino un *framework*, tiene algunas funciones similares como: el control de dispositivos a bajo nivel, la implementación de funciones de uso común, el paso de mensaje entre procesos y el mantenimiento de paquetes. Su uso está enfocado para *UNIX* (*Ubuntu-Linux*), aunque actualmente está en proceso de adaptación para otros sistemas operativos como: *Fedora, MAC OS, Arch, Gentoo, OpenSUSE, Slackware, Debin y Windows*.

Este *software* recibe actualizaciones continúas debido a que está hecho por la comunidad, lo que permite que se tengan muchos paquetes que se integran fácilmente gracias al sistema.

ROS ayuda a diseñar un *software* complejo sin saber realmente cómo funciona cierta parte del *hardware* del sistema. Principalmente, proporciona una vía para conectar una red de procesos mediante nodos con un único eje central. Está basado en una arquitectura de grafos donde el procesamiento se basa en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores. Además, tiene las opciones de control, estados, planificaciones y actuadores. Añade la facilidad de permitir apoyo en cualquier lenguaje de programación al determinar las clases de comunicación en *C++*, o desarrollarlas manualmente.

Los principales conceptos de *ROS* que se utilizan en este proyecto son los siguientes [24]:

- Repositorios. Ficheros compuestos por uno o varios paquetes, estos nos permiten descargar los archivos necesarios de forma más cómoda.
- *Stack* o pila. Conjunto de paquetes que comparten una misma funcionalidad.
- Paquetes. La unidad básica de organización del *software* del sistema. Contiene todo lo necesario para ser funcional.
- Nodo. Procesos ejecutables que están incluidos dentro del paquete.

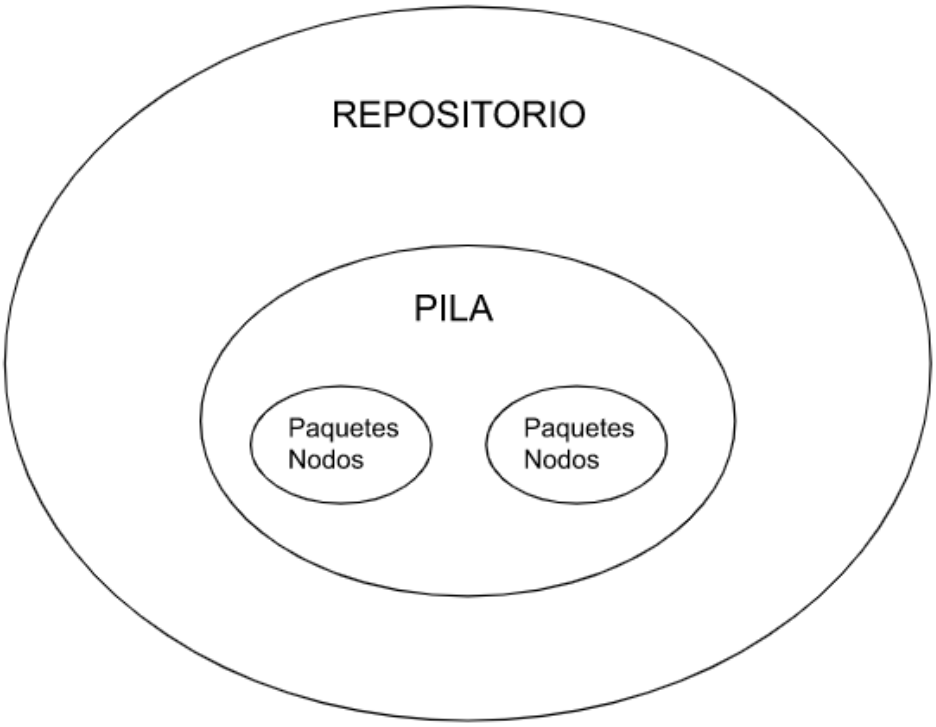

Figura 25. Esquema para comprender el funcionamiento de *ROS*. Fuente [Elaboración propia]

Para entender la computación de *ROS* es necesario familiarizarse con las siguientes definiciones [24]:

- Servicios. Se encargan de realizar la comunicación entre nodos mediante un sistema de petición y respuesta.
- Tópicos. Son los temas que identifican el contenido de un mensaje, y estos se enrutan de dos maneras. Un nodo se suscribe para recibir un tipo de dato; mientras otro es el publicador, es decir, el que transmite un dato.
- Mensajes. Una estructura de datos.
- Maestro. Proporciona un registro de nombres y la búsqueda para el resto de los nodos.
- *Bags*. Formato para guardar y reproducir datos de mensajes de *ROS*.

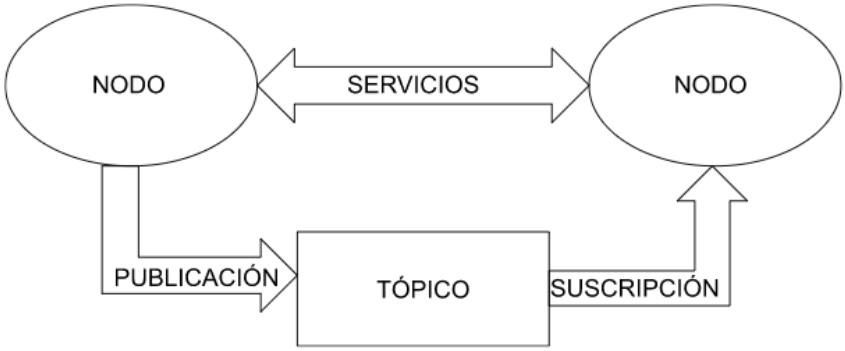

Figura 26. Esquema para comprender el funcionamiento de *ROS*. Fuente [Elaboración propia]

Comandos principales [24]:

- *Rosrun*. Permite lanzar programas utilizando el nombre del paquete y del programa.
- *Roslaunch*. Se empleó para ejecutar múltiples nodos que se tienen en *ROS* tanto de forma local como remota. Los archivos de configuración pueden automatizar de forma sencilla diferentes procesos del robot con un único comando.

Dentro del sistema se han utilizado mayoritariamente estas herramientas:

- *Rviz*. Herramienta de simulación y visualización 3D para robots. En nuestro caso se usó para el *UR5e* y *OptiTrack*. Gracias a este programa se ha podido visualizar la información de los sensores de manera más visual, así como la posición tanto del brazo como del objeto a capturar en cada caso.

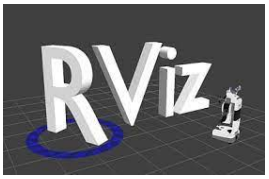

Figura 27. Logotipo del programa *Rviz*. Fuente [25]

- *MoveIT*. [26] Es un *software* de código abierto para *ROS* y se utiliza para la manipulación móvil. Empieza a considerarse un programa estándar ya que actualmente es utilizado en más de 65 robots. Resulta sencillo el trabajo con brazos o manipuladores móviles, ya que, los paquetes de *software* ayudan a ahorrar la realización de cálculos complicados que se hicieron con anterioridad. En este proyecto, dentro de sus funciones, se han utilizado el constructor de tareas y el visualizador interactivo 3D con el *Rviz,* mencionado anteriormente.

Figura 28. Logotipo del programa MoveIt. Fuente [27]

8 IMPLEMENTACIÓN DEL *SOFTWARE*

En este punto se explicará cómo funcionó la implementación tanto del brazo robótico de *Universal Robots* tipo *UR5e* como el conjunto de cámaras tipo *OptiTracks* con *ROS*. Para su puesta en marcha se diferenciaron dos puntos: por un parte, la conexión del brazo con *ROS* y su movimiento mediante *MoveIt*, y por la otra, la conexión de las cámaras *OptiTrack* con *ROS,* así como la devolución de la posición actual del objeto.

8.1 Implementación del brazo *UR5e* con *ROS*

Para su instalación se utilizó el repositorio oficial de Universal Robots de la página de GitHub. Para ello se siguieron los siguientes pasos:

- La creación de un *Workspace* llamado «*ur_ws*» con una carpeta denominada «*src*».
- Dentro de la misma se clonó el repositorio de la página *GitHub* del siguiente enlace: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver.

Una vez realizados estos pasos, se tuvo que preparar el robot para ser controlado con *ROS*. Por lo que a continuación explicaré los puntos que se siguieron:

- Se descargó el archivo «*externalcontrol-1.0.5.urcap*» del siguiente enlace: https://github.com/UniversalRobots/Universal_Robots_ExternalControl_URCap/releases.
- Para su instalación del archivo en *PolyScope*, se entró en el desplegable de la esquina superior derecha:

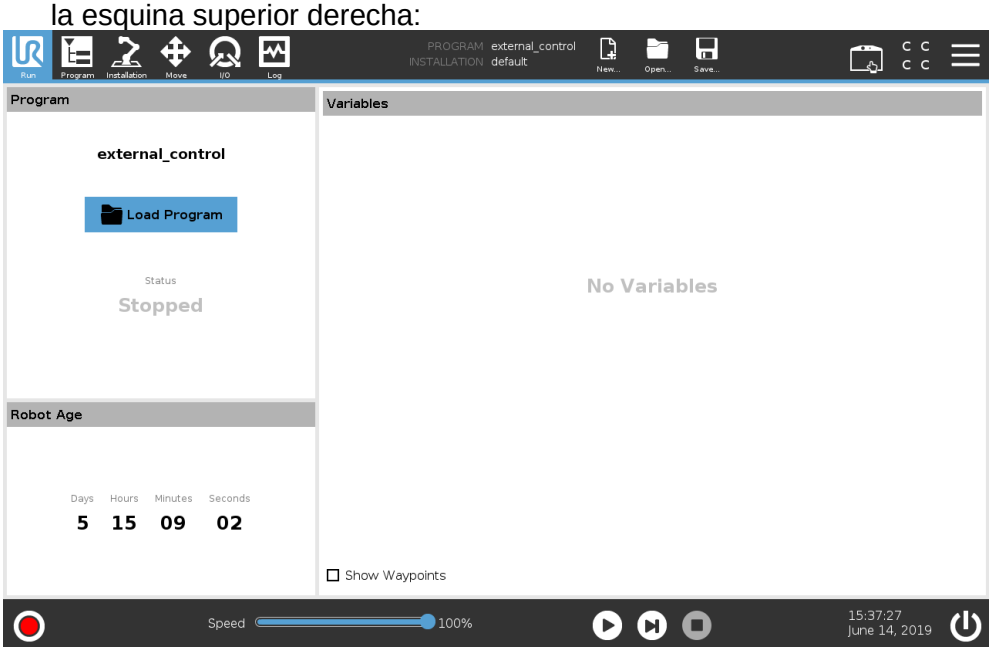

Figura 29. Captura Polyscope. Fuente [28]

- Posteriormente, se accedió en la zona de «*system*» a «*URCaps*». A continuación, se procedió a clicar en el signo «+» que se encuentra en la zona inferior de la pantalla, se seleccionó el archivo «*externalcontrol-1.0.5.urcap*» y se abrió. En este momento, *PolyScope* muestra una pantalla emergente donde te recomienda reiniciar el sistema, por lo tanto, se aceptó y se reinició.

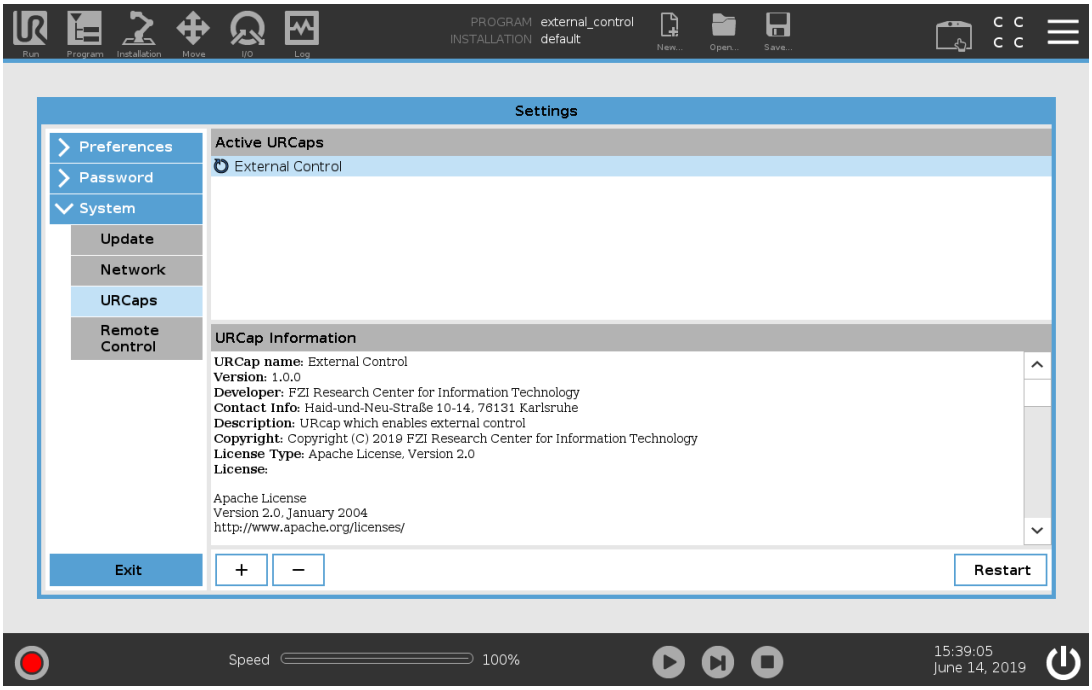

Figura 30. Captura Polyscope. Fuente [28]

- Tras el reinicio del sistema se procedió a ir a la pestaña de «*installation*» y se seleccionó «*URCaps*». Al pulsar en el «*external control*» se mostró lo siguiente:

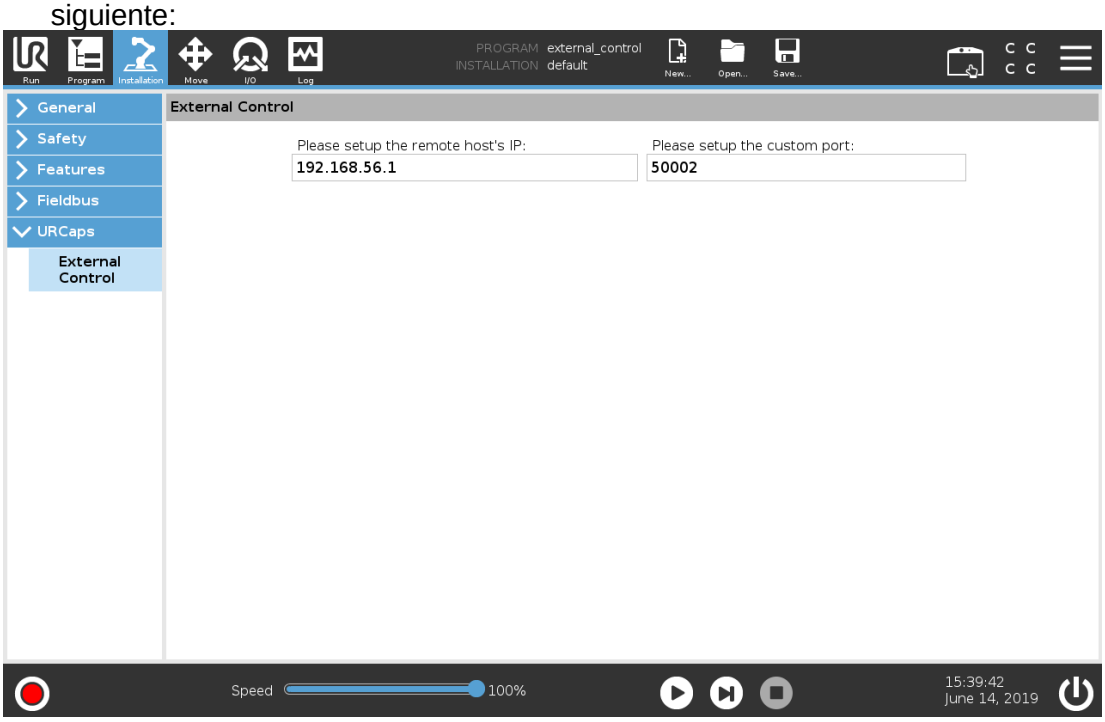

Figura 31. Captura Polyscope. Fuente [28]

- Se cambió la dirección IP a la utilizada en el proyecto: 10.113.36.96.
- Para enlazar el robot con ROS es necesario seguir el siguiente paso: pulsar en la pestaña «*Program*», y a continuación, en «*URCaps*» y «*External Control*».

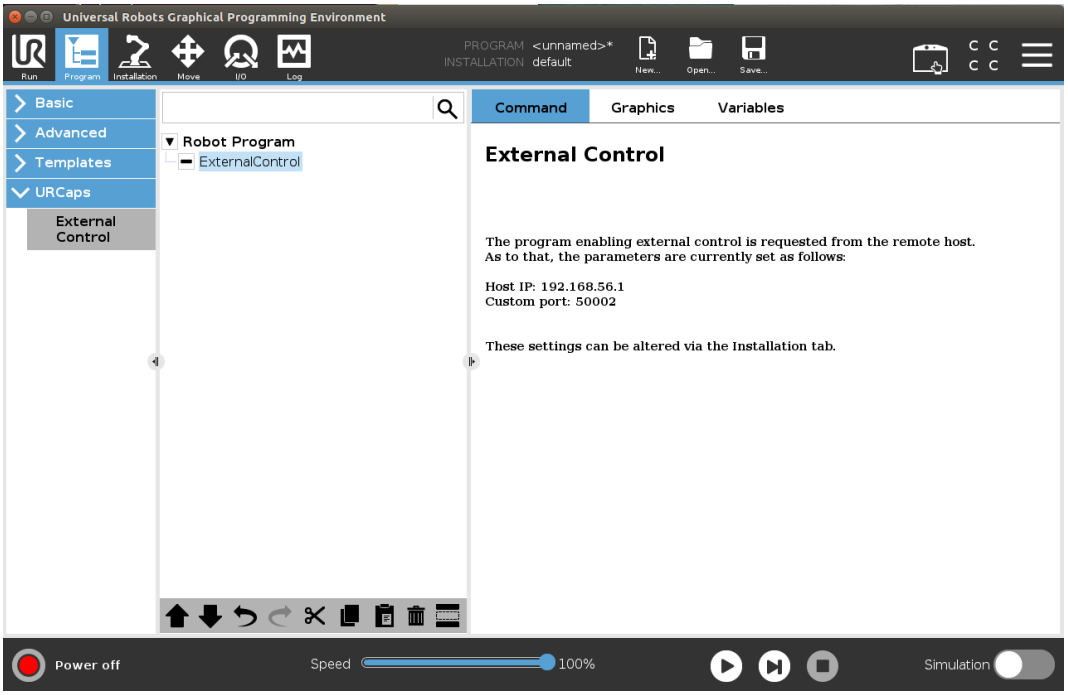

Figura 32. Captura Polyscope. Fuente [28]

Siguiendo con el proceso, se realizó una calibración del brazo lanzando el siguiente comando en la terminal:

- *roslaunch ur_calibration calibration_correction.launch \robot_ip:= 10.113.36.96 target_filename:="/ur_ws/src/Universal_Robots_ROS_Driver/ur_calibration/etc/my_robot_calibration.yaml"*

Para finalizer la tarea de enlazar el UR5e con ROS, se introdujo el *roslaunch* necesario: «*roslaunch my_ur5e myur5e_bringup.launch use_tool_communication:=true tool_voltage:=24 tool_parity:=2 tool_baud_rate:=1000000 tool_stop_bits:=1 tool_rx_idle_chars:=1.5 tool_tx_idle_chars:=3.5 tool_device_name:=/tmp/ttyUR robot_ip:=10.113.36.96 kinematics_config:=$(rospack find ur_calibration)/etc/ur5e_calibration.yaml*. Como último paso en la pantalla del *PolyScope* se presionó el botón de *play*».

Como en este proyecto se requirió de la utilización de *MoveIt*, dentro de la carpeta «*ur_ws/src*» se procedió a la clonación del repositorio del siguiente enlace: https://github.com/ros-industrial/universal_robot.git. MoveIt ayuda a la utilización del software por lo tanto para el presente. Para poder enlazar nuestro programa en *Python* con *MoveIt* se emplearon los comandos facilitados:

```
moveit_commander.roscpp_initialize(sys.argv)
robot = moveit_commander.RobotCommander()
scene = moveit_commander.PlanningSceneInterface()
arm_group = moveit_commander.MoveGroupCommander("arm")
```

Figura 33.Captura de pantalla del Código. Fuente [Elaboración propia]

Con el fin de mover el brazo y realizar una planificación del movimiento, se utilizaron los siguientes comandos de *MoveIt*:

```
wpose = arm_group.get_current_pose().pose
wpose.position.x += scale*posexf
wpose.position.y += scale*posey
wpose.position.z += scale*posezf
waypoints.append(copy.deepcopy(wpose))

(plan, fraction) = arm_group.compute_cartesian_path(
                                   waypoints,   # waypoints to follow
                                   0.01,        # eef_step
                                   0.0)         # jump_threshold

arm_group.execute(plan, wait=True)#mueve el brazo
waypoints.clear()#limpia la lista waypoints
```

Figura 34.Captura de pantalla del Código. Fuente [Elaboración propia]

Finalmente, para poder acceder a posición de la mano actual respecto a la base, se ejecutó lo siguiente:

```
corx_mano= (arm_group.get_current_pose().pose.position.x)
cory_mano= (arm_group.get_current_pose().pose.position.y)
corz_mano= (arm_group.get_current_pose().pose.position.z)
```

Figura 35.Captura de pantalla del Código. Fuente [Elaboración propia]

8.2 Implementación de las cámaras *OptiTrack* con *ROS*

Se creó un *Workspace* para la implementación de las *OptiTrack*, dentro de él se instaló el paquete proporcionado por *Ubuntu* de las cámaras. Después, se realizaron los pertinentes cambios en *Motive* para poder enlazarlo con *ROS*, los cuales se explicarán a continuación.

Se inició con una calibración de las cámaras, (se explica el procedimiento en el anexo). Posteriormente, se creó el *rigid body* del objeto que vamos a rastrear (se explica el procedimiento de la creación del rigid body en el anexo).

Para continuar con el enlace de las cámaras se tuvieron que modificar los siguientes parámetros de *Motive*:

- Se introdujo en la pestaña «*Local Interface*» la *IP* del ordenador al que se quiere conectar las *OptiTrack,* en este caso fue: 10.113.36.96.
- Se cambió el tipo de «*Transmission Type*» a «*Multicast*».
- El resto de las características se dejaron como predeterminadas.

Se volvió al ordenador con ROS para realizar modificaciones el fichero «mocap.yaml». El cual tendrá el siguiente aspecto:

#

Definition of all trackable objects

Identifier corresponds to Trackable ID set in Tracking Tools

#

mocap_node:

ros__parameters:

rigid_bodies:

```
        1:
          pose: "Robot_1/pose"
          pose2d: "Robot_1/ground_pose"
          child_frame_id: "Robot_1/base_link"
          parent_frame_id: "world"
        2:
          pose: "Robot_2/pose"
          pose2d: "Robot_2/ground_pose"
          child_frame_id: "Robot_2/base_link"
          parent_frame_id: "world"
      optitrack_config:
          multicast_address: "224.0.0.1"
          command_port: 1510
          data_port: 9000
```

Por lo tanto, para su óptima configuración fue necesario cambiar el siguiente parámetro:

- En nuestro caso se cambiaron todos los «Robot_1» por el nombre de nuestro rigid body creado que fue «pelota».

Lo que hace que el fichero «*mocap.yaml*» se visualice de la siguiente manera tras los cambios:

```
#
# Definition of all trackable objects
# Identifier corresponds to Trackable ID set in Tracking Tools
#
mocap_node:
  ros__parameters:
    rigid_bodies:
      1:
        pose: "pelota/pose"
        pose2d: "pelota/ground_pose"
        child_frame_id: "pelota/base_link"
        parent_frame_id: "world"
    optitrack_config:
        multicast_address: "224.0.0.1"
        command_port: 1510
        data_port: 9000
```

A continuación, se lanzó el *roslaunch* del repositorio. Para finalizar el enlace con *ROS*, en el programa *Motive* en la pestaña de «*Streaming*» se desactivará y se activará la opción «*Eneable*».

Después, se visualizaron los datos que nos devolvían las *optritrack* desde la terminal:

Figura 36.Datos publicados por las OptiTrack. Fuente [Elaboración propia]

Para poder tener acceso a estos datos en el programa que se realizó en *Python,* fue imprescindible hacer una suscripción al «*topic*» donde son publicados. Esta suscripción se realizó de la siguiente manera:

```
sub=rospy.Subscriber("/mocap_node/pelota/pose", PoseStamped, funcion)
```

Figura 37. Código del programa. Fuente [Elaboración propia]

9 RESOLUCIÓN DEL PROBLEMA

9.1 Descripción

Para la ejecución de la tarea se ha intentado parar un objeto sobre un vehículo impulsado por una mano. Se realizó una predicción de la posición del elemento, esperando que el brazo se moviera y lo pudiera detener. Además, con el fin de optimizar los movimientos, el robot partirá siempre desde el centro de la mesa, a la altura del objeto y desde la línea de parada de este.

9.2 Resolución

Primero, se estableció un eje de coordenadas para que su ejecución fuera más cómoda y efectiva.

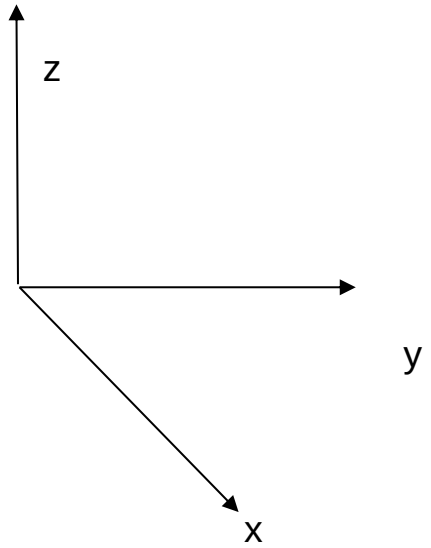

Figura 38. Eje de coordenadas que se utilizara para la resolución del problema. Fuente [Elaboración propia]

Se comenzó por mover el brazo a la posición que se le ordenó. Sin embargo, este se mueve a partir del *gripper,* donde tiene la base (0,0,0). Por lo tanto, las órdenes que se mandan al robot son que se desplace a partir de la posición actual. Para poder mover el brazo desde la base de este, se tuvo como ayuda el conocimiento de la posición del *gripper* en referencia a la base del robot. Así pues, para poder introducir la posición en el espacio a la que se quiere llegar en referencia a la base se tuvo que solucionar de la siguiente manera:

Ejemplo:

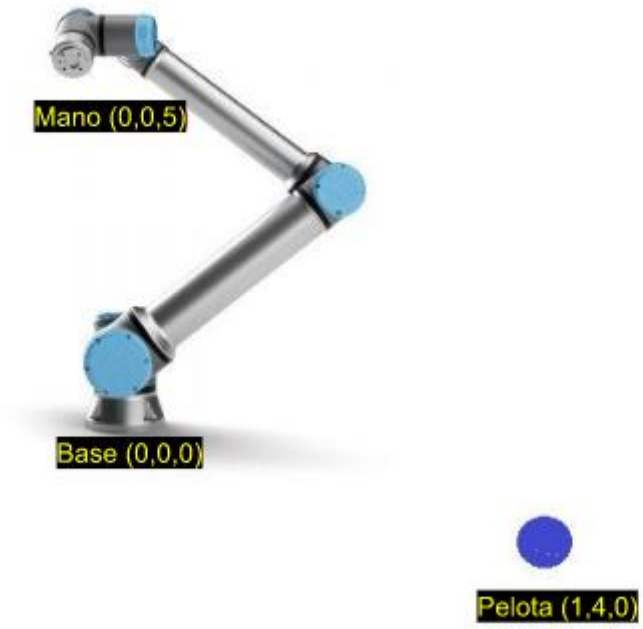

Figura 39. Ejemplo visual para la solución del problema. Fuente [Elaboración propia]

Para que se desplace desde la mano hasta el objeto se resolvió:

- Movimiento=Objeto-Mano= (1, 4, 0) - (0, 0, 5) = (1, 4, -5)

Esta solución serían la coordenada que hay que introducir en la orden *wpose* para que se alcance el objeto.

De manera general:

- Movimiento=Objeto-Mano=(x`,y`,z`) - (x,y,z) = (x`-x, y`-y, z`-z)

Cuando las *OptiTrack* devuelven la posición del elemento se volverá a tener el mismo problema y método de solución, ya que las cámaras tendrán la referencia (0,0,0) desplazada respecto a la base del brazo. No obstante, se necesitó esta referencia en la base del *UR5*e para obtener unos cálculos más sencillos. En este caso el problema resulta más fácil, ya que las referencias son fijas y por lo tanto, se tendrá que sumar siempre el mismo valor.

Ejemplo:

Figura 40.Distancia entre la base del robot y la referencia de coordenadas de las OptiTrack. Fuente [Elaboración propia]

De manera general como la base es fija siempre se realizará el siguiente cálculo:

- Base = Objeto + Referencia = (x, y, z) + (0’14, -0’01, 0) = (x+0’14, y-0’01, z+0)

Cuando se requiera mover el brazo a la posición que se calculó en «x» del objeto, se añadió a la posición calculada 0’14, y cuando se obtuvo la posición «y» a la que se quiso mover el brazo se tuvo que sumar -0’01.

A continuación, se procedió a analizar el método mediante el que se predice la posición del objeto a parar. El cual se resolvió mediante la ecuación de la recta. Gracias a que se tiene normalmente un movimiento lineal, simplemente será necesario saber los dos puntos de la trayectoria del objeto y la solución de la ecuación fue la siguiente: $(x - x_1) = m \cdot (y - y_1), donde\ m = (x_2 - x_1)/(y_2 - y_1)$. Primero, se calculó la pendiente de la recta (m). Debido a que en la primera resolución siempre se capturó el elemento en x=0’4, para saber el punto al que se moverá el robot se resolvió la siguiente ecuación: $y = (0'4 - x_1 + m \cdot y_1)/m$. De esta manera se obtuvo la predicción del punto «y», lo que indicará al brazo robótico que se desplace a la posición (0’4, y). Con la excepción de que si la m=0 entonces, $y = y_1 = y_2$, donde tendremos un movimiento perpendicular a la zona de detención.

Como se puede observar, se encontró el problema de que se necesitaba el punto actual del objeto y del anterior, para resolver este problema se introdujo una variable de iteraciones con el bucle, denominada «*ite*», la cual se inició a 1 y cada vez que entra en el bucle aumenta su valor en 1. Resultando útil dentro del «*while*» donde se ejecutó la predicción del punto al que se tiene que mover el brazo hasta que pare el objeto, por lo tanto, se guardaron dos datos de manera continua: el dato actual y el anterior. Cuando «*ite*» sea número impar guardará la posición del objeto en (x1, y1) y cuando «*ite*» sea par guardará la posición en (x2, y2).

En el programa se realizó de la siguiente manera:

```
if ite % 2 == 0:
    rospy.sleep(0.1)
    x2=-objeto_y
    y2=objeto_x
    objeto_basez=objeto_z
    ite=ite+1


else:
    rospy.sleep(0.1)
    x1=-objeto_y
    y1=objeto_x
    objeto_basez=objeto_z
    ite=ite+1
```

Figura 41.Código del programa. Fuente [Elaboración propia]

A continuación, fue necesario saber cuál es la menor de las dos posiciones. Para conocer la más cercana al robot, y por lo tanto saber la más actual, se adaptó la ecuación de la recta de la siguiente manera, considerando que el objeto siempre se mueve hacia delante:

- Cuando «x1» es la actual y «x2» la anterior, es decir, «x2>x1», y además se debe cumplir que «y2» sea diferente de «y1»: $m = (x_2 - x_1)/(y_2 - y_1)$ e $y = (0'4 - x_1 + m \cdot y_1)/m$.
- Cuando «x2» es la actual y «x1» la anterior, es decir, «x1>x2», y además se debe cumplir que «y2» sea diferente de «y1»: $m = (x_1 - x_2)/(y_1 - y_2)$ e $y = (0'4 - x_2 + m \cdot y_2)/m$.
- Cuando «y2=y1», se tendrá: $y = y_1 = y_2$

En el programa se realizaría de la siguiente manera:

```
if x2>x1:
    if y2!=y1:
        m=(x2-x1)/(y2-y1)
        y=(0.4-x1+m*y1)/m


if x1>x2:
    if y2!=y1:
        m=(x1-x2)/(y1-y2)
        y=(0.4-x2+m*y2)/m


if y2 == y1 :
    y2=y
```

Figura 42. Código del programa. Fuente [Elaboración propia]

Este punto se utilizó para mover el robot a la posición de la mesa, a excepción de cuando el objeto está parado, donde «x1=x2» y el robot se queda quieto.

Quedando la primera parte resuelta, si el cuerpo llega hasta el final de la mesa el robot puede predecir la posición en «y» para detenerlo. En la realización de pruebas se observan dos inconvenientes: no se utiliza el máximo rango del robot, y este no optimiza sus movimientos en función de la velocidad del objeto, haciendo que se mueva de manera continua para desplazarse escasos milímetros de distancia.

Para poder utilizar la máximo capacidad del brazo y que se adelante a parar el objeto antes de que caiga de la mesa, se realizaron los siguientes cálculos: se midió el ancho máximo de la mesa, la cual es de 0’75 metros a cada lado desde el centro de la superficie. Por lo tanto, primero se realizó una predicción de la posición «y» del objeto sabiendo de esta manera si el cuerpo caerá antes de la posición de recogida de la mesa. Esto se sabe cuando la «y» que se calcula es inferior a -0’75 o superior a 0’75. Si sucediera, se procedería a calcular la posición «x» de la misma manera que se predijo

la «y», moviendo el brazo a la posición (x, -0’75) para un lado de la mesa, y para el otro lado de la mesa a (x, 0’75).

Al calcular la posición «x» se siguieron los mismos pasos que se explicaron en el cálculo de «y», es decir, primero se calcula la pendiente de la recta «m» y después se predijo la «x» con las siguientes ecuaciones:

- Cuando «y<-0’75», «x2>x1»: $m = (x_2 - x_1)/(y_2 - y_1)$, $x = m \cdot (-0.75 - y_2) + x_2$
- Cuando «y-0’75», «x1>x2»: $m = (x_1 - x_2)/(y_1 - y_2)$, $x = m \cdot (-0.75 - y_1) + x_1$
- Cuando «y>0’75», «x2>x1»: $m = (x_2 - x_1)/(y_2 - y_1)$, $x = m \cdot (0.75 - y_2) + x_2$
- Cuando «y>0’75», «x1>x2»: $m = (x_1 - x_2)/(y_1 - y_2)$, $x = m \cdot (0.75 - y_1) + x_1$

Obtenida la posición «x» a la que se moverá el brazo para poder coger el objeto antes de que este pueda caer, se tuvo que delimitar la posición máxima de la «x» a la que llega el brazo en la esquina de la mesa. En este caso, solo podrá coger el objeto para valores de «x<0’5», por lo que cuando el valor sea mayor, el brazo se levantará a una posición determinada para avisar de que no llegará antes de que el cuerpo caiga.

Muestra del ejemplo de cómo se movió en este caso:

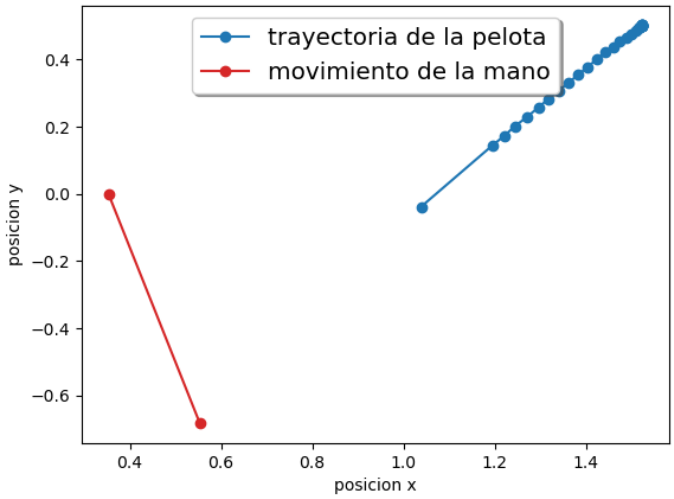

Figura 43. Gráfica de como varia el brazo y del objeto. Fuente [Elaboración propia]

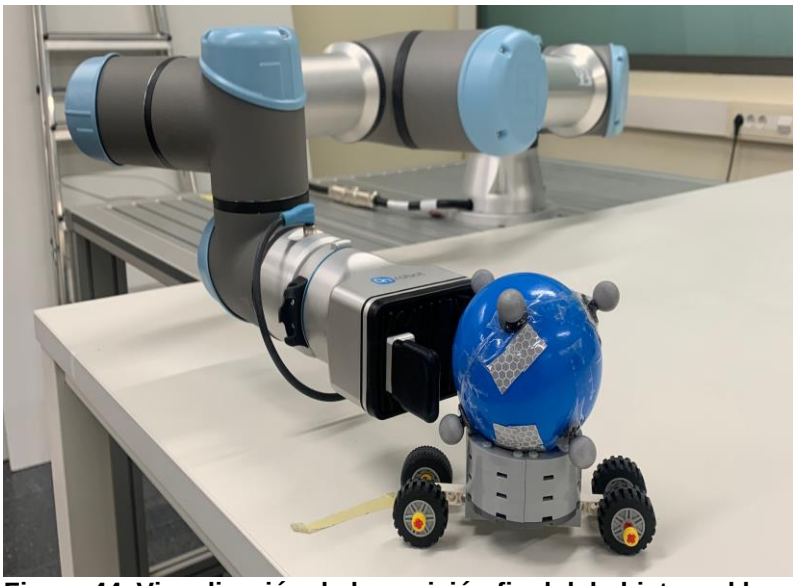

Figura 44. Visualización de la posición final del objeto y el brazo. Fuente [Elaboración propia]

Tanto en la gráfica como en la foto, se puede apreciar cómo se adelanta en la posición «x» para intentar atrapar el objeto antes de que este caiga por el lateral.

Por último, para que el robot sea óptimo en sus movimientos, se hicieron cuatro tramos en función de la velocidad a la que se empieza a mover el brazo para poder detener el objeto. Como el programa tarda el mismo tiempo en recopilar los datos, se hizo una resta de la posición en «x» para saber cuánto avanzó el cuerpo en ese determinado tiempo. Se obtuvo como resultado, tras varias pruebas para conocer su límite, los siguientes tramos:

- Tramo 1 o de baja velocidad. Cuando el objeto se mueve muy despacio y la diferencia de la posición «x» es inferior a 0’02 en cada iteración del bucle. Entonces, el robot se moverá cuando el elemento alcance la posición 0’8 de la mesa. Ejemplo:

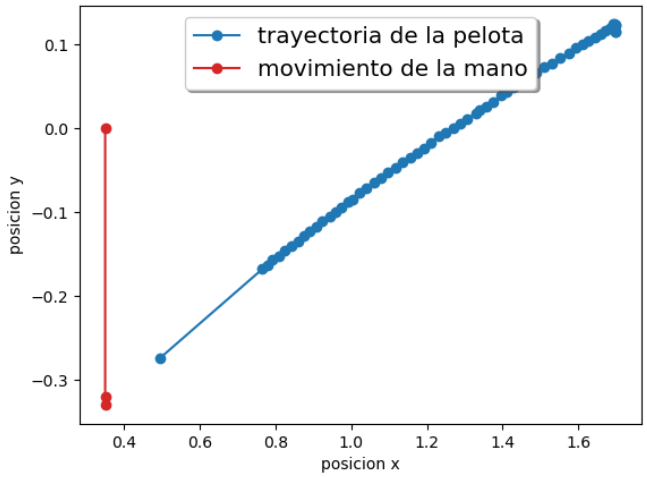

Figura 45. Gráfica de como varia el brazo y del objeto. Fuente [Elaboración propia]

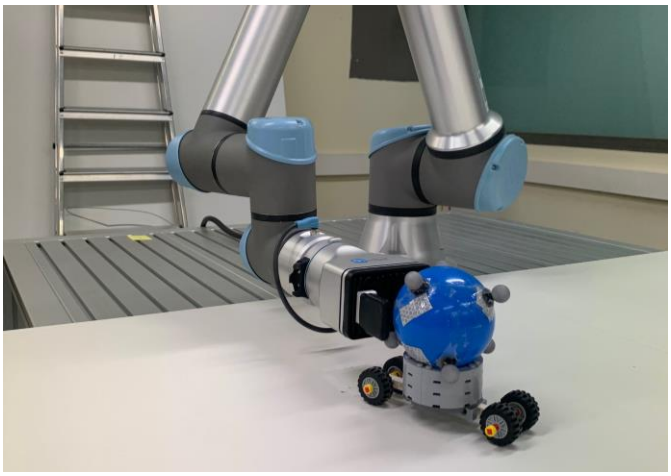

Figura 46.Visualización de la posición final del objeto y el brazo. Fuente [Elaboración propia]

```
if v<0.02:
    if x2<0.8:
```

Figura 47.Condicionales para entrar a velocidad baja. Fuente [Elaboración propia]

- Tramo 2 o velocidad media. Cuando el objeto se mueve a una velocidad media, y la diferencia de la posición «x» está comprendida entre valores de 0’02 y 0’035 entre cada iteración. Entonces, el brazo comenzará a moverse cuando el objeto alcance la posición 1’2 de la mesa. Ejemplo:

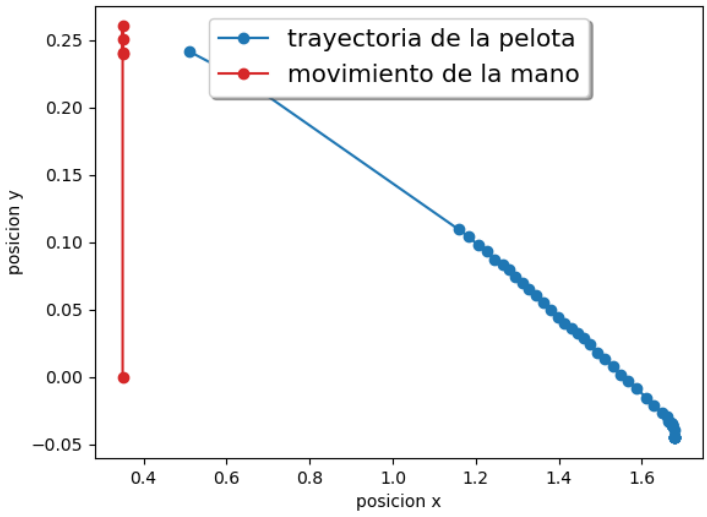

Figura 48.Gráfica de como varia el brazo y el objeto. Fuente [Elaboración propia]

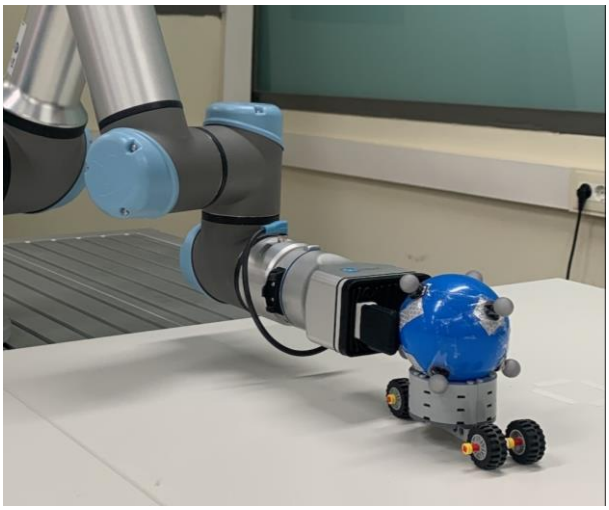

Figura 49.Visualización de la posición final del objeto y el brazo. Fuente [Elaboración propia]

```
if v>0.02:
    if v<0.035:
        if x2<1.2:
```

Figura 50.Condicionales para entrar a velocidad media. Fuente [elaboración propia]

- Tramo 3 o velocidad alta. Cuando el objeto se mueve a una velocidad alta, y la diferencia de la posición «x» está comprendida entre valores de 0’035 y 0’045 entre cada iteración. El brazo comenzará a moverse cuando el objeto alcance la posición 1’6 de la mesa. Ejemplo:

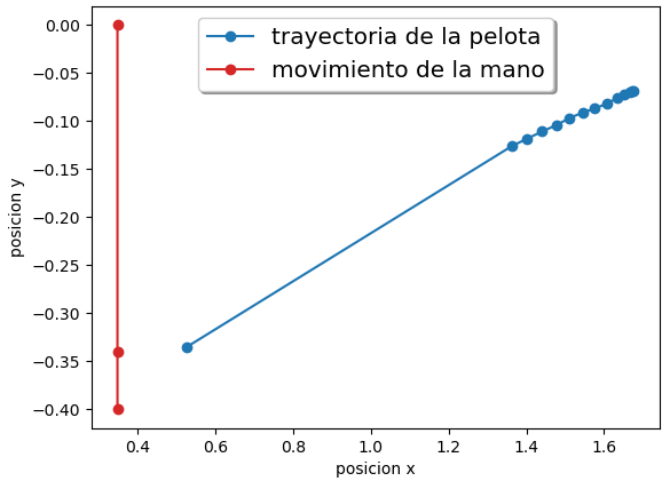

Figura 51. Gráfica de como varia el brazo y el objeto. Fuente [Elaboración propia]

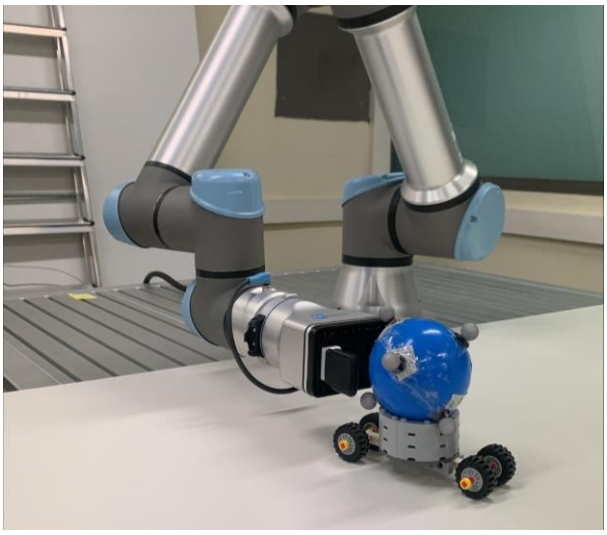

Figura 52. Visualización de la posición final del objeto y el brazo. Fuente [Elaboración propia]

```
if v>0.035:
    if v<0.045:
        if x2<1.6:
```

Figura 53. Condicionales para entrar a velocidad alta. Fuente [Elaboración propia]

- Tramo 4 o velocidad de corte. En este tramo el brazo no llega a coger el objeto por su alta velocidad. Esto sucede cuando la diferencia de la posición «x», entre cada iteración, es superior a 0’045. En consecuencia, el brazo se levantará a una posición determinada para señalar que no la puede alcanzar.

Si el objeto va a caer de la mesa o por la velocidad del mismo, no es capaz de pararlo, el robot tomará la siguiente posición:

Figura 54. Visualización de la posición final del brazo. Fuente [Elaboración propia]

```
if v>0.045:
```

Figura 55. Condicionales para entrar a velocidad de corte. Fuente [Elaboración propia]

Cabe destacar que durante la tarea, el robot configura su altura posicionándose siempre a la misma altura que el objeto.

9.3 Resultados.

Para la comprobación de su funcionamiento, se realizaron una serie de pruebas que constan del lanzamiento del cuerpo con diferentes alturas y ángulos de salida, para saber cómo de sólido es el sistema de detención del elemento.

Comenzaremos alterando la altura, donde para cada altura tendremos 4 casuísticas: rango de velocidades altas, rango de velocidades medias, rango de velocidad bajas y paradas ante caídas. En cada una de ellas, se realizaron 20 lanzamientos y se calculó con ellos el porcentaje de acierto conseguido. Obteniendo de esta manera las siguientes gráficas:

- Para una altura de 9,8 centímetros:

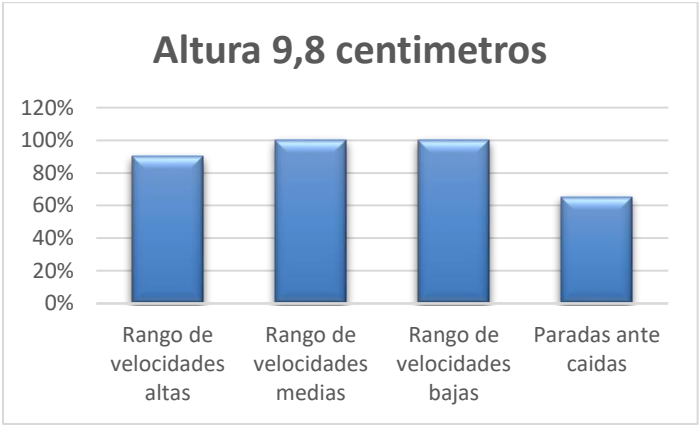

Figura 56. Gráfica de porcentaje de acierto a la altura de 9,8cm. Fuente [Elaboración propia]

- Para una altura de 12 centímetros:

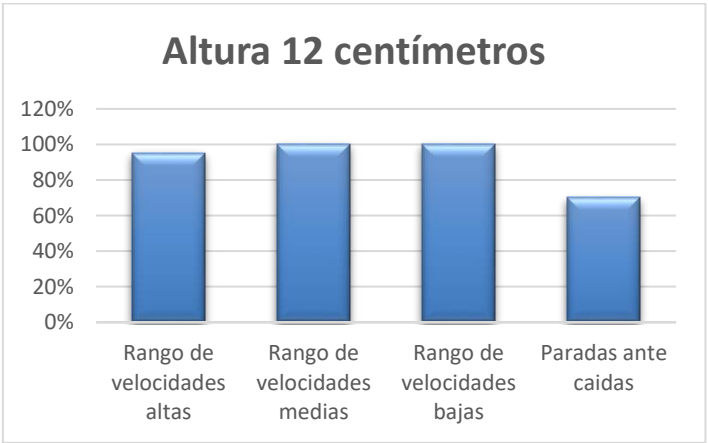

Figura 57. Gráfica de porcentaje de acierto a la altura de 12cm. Fuente [Elaboración propia]

- Para una altura de 18 centímetros:

Figura 58. Gráfica de porcentaje de acierto a la altura de 18cm. Fuente [Elaboración propia]

De estas gráficas se puede deducir que el sistema trabaja igual de bien a diferentes alturas. Además, en el rango de velocidades altas tenemos un pequeño porcentaje de error debido, seguramente, a los tiempos que tarda el brazo en planificar y moverse a una posición alejada. También se puede deducir que el porcentaje de parada ante caída es menor respecto al del resto. Se supone que esto es debido a que el robot no regula la velocidad e intenta pararla siempre que vaya a caer de la mesa

A continuación, se hicieron pruebas alterando el ángulo de salida desde el centro de la mesa teniendo las siguientes trayectorias:

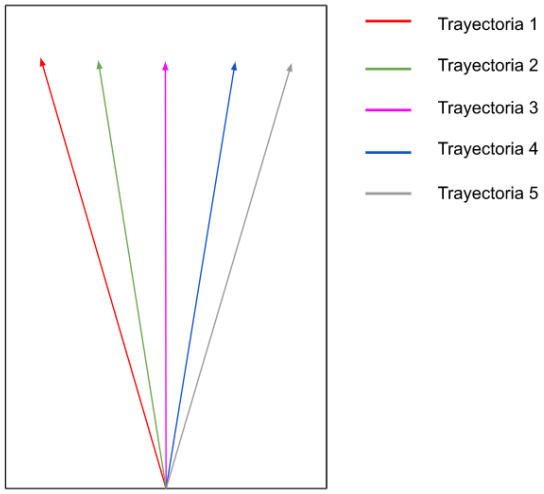

Figura 59. Plano de las trayectorias del objeto en la mesa. Fuente [Elaboración propia]

Como resultados a los diferentes ángulos de salida se obtuvo:

- Para un ángulo de salida de -24º equivalente a la trayectoria 1:

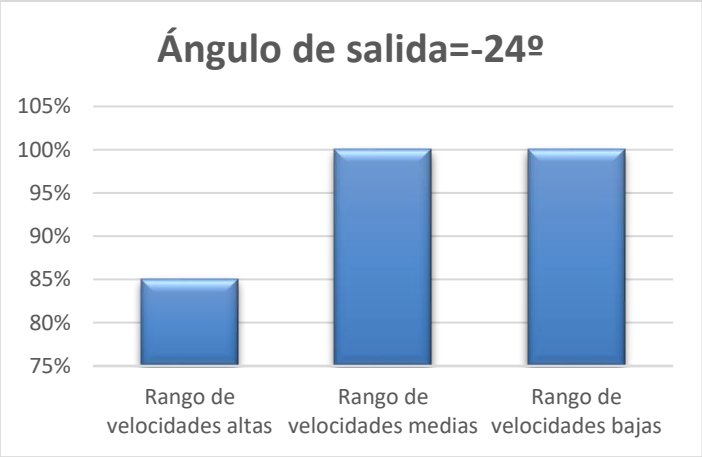

Figura 60. Gráfica de porcentaje de acierto con el ángulo de salida de -24º. Fuente [Elaboración propia]

- Para un ángulo de salida de -11º equivalente a la trayectoria 2:

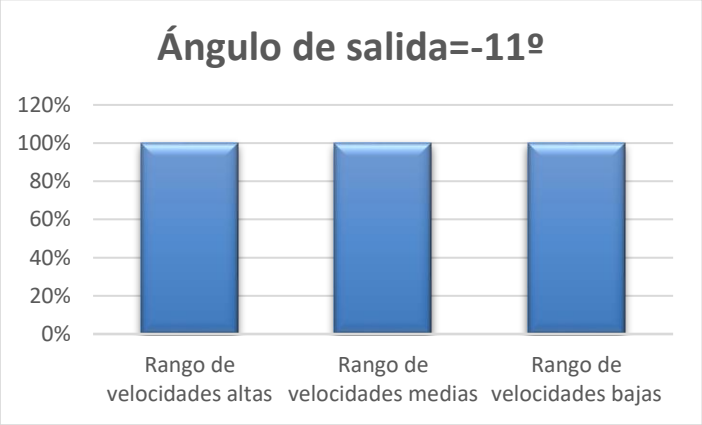

Figura 61. Gráfica de porcentaje de acierto con el ángulo de salida de -11º. Fuente [Elaboración propia]

- Para un ángulo de salida de 0º equivalente a la trayectoria 3:

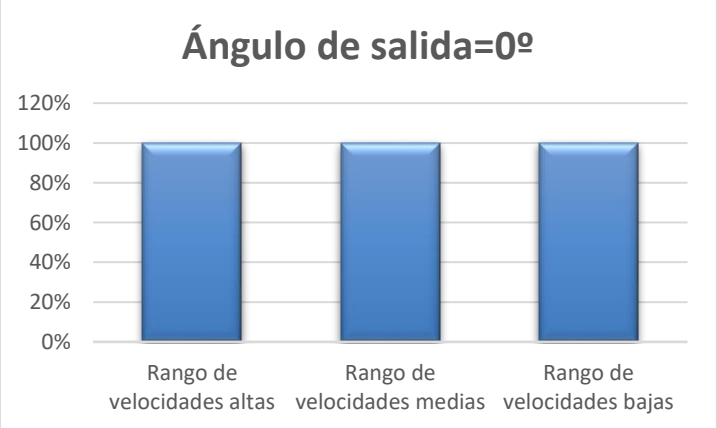

Figura 62. Gráfica de porcentaje de acierto con el ángulo de salida de 0º. Fuente [Elaboración propia]

- Para un ángulo de salida de 14º equivalente a la trayectoria 4:

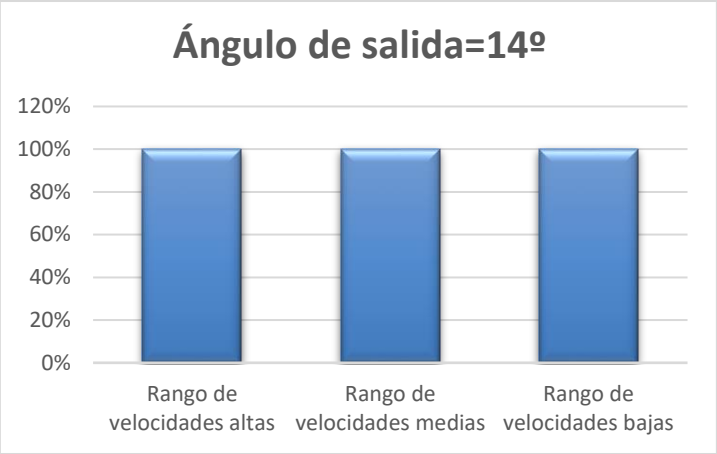

Figura 63. Gráfica de porcentaje de acierto con el ángulo de salida de 14º. Fuente [Elaboración propia]

- Para un ángulo de salida de 25º equivalente a la trayectoria 5:

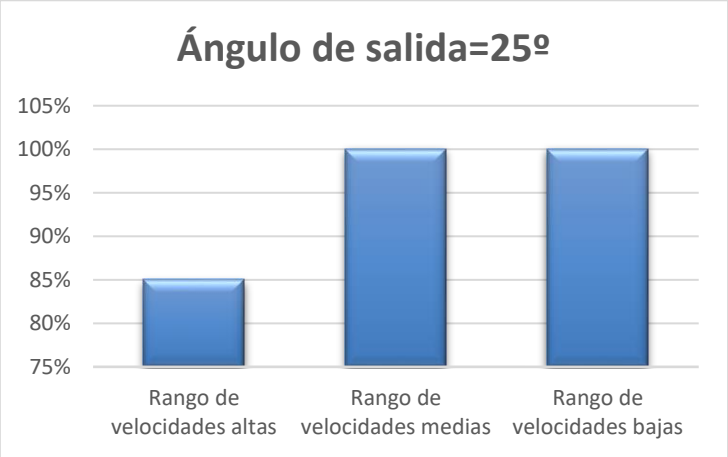

Figura 64. Gráfica de porcentaje de acierto con el ángulo de salida de 25º. Fuente [Elaboración propia]

Se pudo concluir, con las gráficas mostradas, que los fallos son producidos en el rango de velocidades altas y más concretamente cuando el valor absoluto del ángulo de salida crece. Realmente, esto es producido por un aumento en la planificación y movimiento del brazo a esta posición, que al ser la más alejada necesita un mayor tiempo. Además, para agravar el problema se encuentra en el rango de mayor velocidad del elemento haciendo que se tenga un menor tiempo de respuesta.

9.4 Código del programa.

El código del programa se encontrará disponible en el siguiente enlace: https://github.com/GII/student_projects/tree/main/2022_alejandro_martinez/source_code.

10 CONCLUSIONES

10.1 Análisis de los resultados

Una vez obtenidos los resultados del presente trabajo, se realizó un estudio para poder tener unas conclusiones concretas.

Todavía queda un largo camino por recorrer en la robótica en la captura de objetos en movimiento, ya que son necesarios ciertos cambios para conseguir una mejoría debido a la velocidad, a la aceleración y a la trayectoria. La predicción del movimiento de un objeto puede llegar a ser muy tedioso por el método analítico, llevando la investigación al fracaso por el exceso de horas invertidas o el exceso de cálculo computacional, provocando una lentitud extrema en el sistema y por lo tanto incapaz de capturar un objeto a alta velocidad. Sin embargo, mediante un método predictivo o estadístico se pueden llegar a obtener buenos resultado en menor tiempo, aunque siempre tendremos un cierto porcentaje de fallo a la hora de realizar los experimentos.

Como se pudo observar, en los experimentos, no es un sistema veloz y para que pueda parar el cuerpo, tiene que ir a una velocidad baja en comparación al movimiento de una pelota lanzada naturalmente con la mano. No obstante, aunque es posible mejorar los resultados con un ordenador que permita hacer una planificación más rápida, o con la utilización de un brazo con mejores características, esto produciría un aumento del presupuesto notable. Además, es muy difícil implementar estas opciones a nuestra escala y en el laboratorio, ya que el tamaño del mismo y el coste suponen un problema.

10.2 Estudio de aplicabilidad

Durante el proceso me he encontrado con ciertos inconvenientes a la hora de implementar este proyecto en la industria debido al sensor utilizado, ya que, aunque es el mejor del mercado por su alta precisión y constante actualización de la posición, se tendrían que añadir manualmente los marcadores correspondientes a cada objeto dentro de la industria. También sería necesario añadir el denominado *rigid body* tanto en el programa *Motive* como en el código de *ROS*, lo que hace este proceso más complejo para poder ser implementado en una empresa. Uno de los principales defectos del brazo es su corto alcance y una pequeña tasa de carga, dentro de una industria se necesitarían mejores características en estos aspectos y seguramente esto se consigue disminuyendo nuestra buena precisión o aumentando el presupuesto. A mayores, como se pudo observar en la realización de pruebas, con una velocidad baja el robot ya no llega a atrapar el objeto, por lo que necesitaríamos un brazo más veloz y con esto volveríamos a tener un descenso de precisión o un precio mucho más elevado.

Aun con todos los inconvenientes presentes en una futura implementación industrial, para el presente trabajo fin de grado se han cumplido los objetivos de la implementación del robot y los sensores para movimientos dinámicos básicos. Es decir a una escala pequeña y mediante movimientos simples, se pudo detener el objeto impulsado por nuestra propia mano.

10.3 Comparación con sistemas existentes

Actualmente se le ve un gran potencial, ya que ahorraría una gran cantidad de tiempo en situaciones de variación del entorno o de la posición, velocidad y aceleración del objeto. Su principal objetivo se centra en tareas de pick and place, o la recogida de piezas móviles en sitios de difícil acceso de manera automática. Además, si se ejecutara una mejoría de los sensores se podría aplicar a tareas militares o sanitarias de gran riesgo. Sin embargo, la mayoría de estos proyectos no están implementados en la

industria debido a un alto coste, y la mayoría de las industrias utilizan una producción en línea, donde no existen movimientos que no sean predeterminados con anterioridad. En conclusión, este tipo de implementaciones serán utilizadas mayoritariamente en el campo de la investigación.

En la línea de investigación se encontró un brazo robótico de la compañía *KUKA*, programado en el laboratorio de *Aprendizaje de Algoritmos y de Sistemas de la EPFL (LASA)* [29]. Este fue diseñado y creado para atrapar objetos con diferentes formas y trayectorias en menos de cinco centésimas de segundo. El brazo que acabo de mencionar, en comparación con el que se utilizó en el trabajo, tiene mayor rango, más tasa de reacción, una velocidad mayor. En el presente, se utilizó un sistema analítico para poder ejecutar la parada del elemento, mientras que en la investigación se inspiró en el aprendizaje autónomo del robot por imitación, además de prueba y error. También utilizaron cámaras de seguimiento con marcadores pasivos en los objetos, tal y como como se dispuso en el proyecto.

10.4 Trabajo futuro

Como futuro trabajo, se podría estudiar la realización de la predicción en el eje «z» para diferentes cuerpos u objetos de manera analítica. Además, se debe intentar depurar el código al máximo para que el robot tenga la capacidad de realizar movimientos más veloces, ya sea mediante predicciones analíticas, estadísticas o por un sistema de aprendizaje.

A partir de esta base se podría introducir una mano robótica al brazo haciendo que, además de pararla, sea capaz de coger el cuerpo lanzado. De esta manera tendríamos un sistema más completo e inteligente.

11 PRESUPUESTO

HARDWARE			
PRODUCTO	**UNIDAD**	**PRECIO/UNIDAD**	**IMPORTE**
UR5e [30]	1	27.000,00€	27.000,00€
OptiTrack Primex 13W [6]	4	2.409,16€	9.636,64€
Ordenador [31]	2	1.299€	2.598,00€
Cable *ethernet* categoría 6 de 35m [32]	5	22,99€	114,95€
Switch POE [33]	1	277,09€	277,09€
IMPORTE TOTAL:			**39.626,68€**

SOFTWARE			
PRODUCTO	**UNIDAD**	**PRECIO/UNIDAD**	**IMPORTE**
ROS	1	0€	0€
Motive:Tracker [34]	1	963,09€/año	963,09€/año
Windows 10 Pro [35]	1	259,00€	259,00€
IMPORTE TOTAL:			**1.222,90€**

MANO DE OBRA			
PRODUCTO	**HORA**	**PRECIO/HORA**	**IMPORTE**
Desarrollo del proyecto	400	35,00€	14.000,00€
IMPORTE TOTAL:			**14.000,00€**

TOTAL DEL PROYECTO	
PRODUCTO	**PRECIO**
Hardware	39.626,68€
Software	1.222,90€
Mano de obra	14.000,00€
IMPORTE TOTAL:	**54.849,58€**

12 BIBLIOGRAFÍA

[1] «RAE, definición de robótico/a,» [En línea]. Available: https://dle.rae.es/rob%C3%B3tico.

[2] M. M. Roa, «Statista: El número de robots industriales se ha triplicado en la última década,» 24 Octubre 2022. [En línea]. Available: https://es.statista.com/grafico/28534/stock-operativo-mundial-de-robots-industriales/.

[3] M. M. S. López, «Tesis: Interacción dinámica suelo,» 2016. [En línea]. Available: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/12135/Tesis.pdf?sequence=3.

[4] U. Robots, «Manual Universal Robots, UR5e,» [En línea]. Available: https://cfzcobots.com/wp-content/uploads/2018/06/UR5e_User_Manual_es_Global-reducido.pdf.

[5] «Direct Industry: Robot articulado UR5e,» [En línea]. Available: https://www.directindustry.es/prod/universal-robots-s/product-206583-2442605.html.

[6] «Cámaras, optitrack.,» [En línea]. Available: https://optitrack.com/cameras/.

[7] «Eurobost: Especificaciones robot KUKA,» [En línea]. Available: https://www.eurobots.net/robot-usado-lbr-iiwa-es.html.

[8] «Robot Worx: KUKA LBR IIWA 7 R800,» [En línea]. Available: https://www.robots.com/robots/lbr-iiwa-7-r800.

[9] «Universal Robots, UR3 Especificaciones técnicas,» [En línea]. Available: https://www.universal-robots.com/media/240781/ur3_sp.pdf.

[10] «Epl-si: Universal Robot UR3,» [En línea]. Available: https://epl-si.com/2015/03/universal-robots-ur3/.

[11] «Optitrack, Hardware,» [En línea]. Available: https://docs.optitrack.com/hardware.

[12] «OptiTrack: Software Motive,» [En línea]. Available: https://optitrack.com/software/motive/.

[13] «OptiTrack, Marker configurator,» [En línea]. Available: https://optitrack.com/accessories/marker-configurator/.

[14] «UNE-EN ISO 12100,» de *Seguridad de las máquinas, principios generales para el diseño, evaluación del riesgo y reducción del riesgo*, AENOR, Mayo 2012, p. 88.

[15] «UNE-EN ISO 10218-2,» de *Robots y dispositivos robóticos, requisitos de seguridad para robots industriales*, AENOR, Septiembre 2016, p. 86.

[16] «Seton: Señal de peligro por quemaduras,» [En línea]. Available: https://www.seton.es/placas-senalizacion-maquinas-riesgo-quemadura-no-tocar.html.

[17] «Pinterest: Señal de reisgo eléctrico,» [En línea]. Available: https://www.pinterest.es/pin/317644579965596517/.

[18] «MundoSeñal: Señal de peligro.,» [En línea]. Available: https://mundosenal.com/comprar/senal-de-peligro-en-general/.

[19] «ATProtección: Seña de uso de casco.,» [En línea]. Available: https://www.atproteccion.com/senales-de-obligacion/1737-es-obligatorio-el-uso-del-casco-a4-y-a3.html.

[20] «ATProtección: Señal de uso de guantes.,» [En línea]. Available: https://www.atproteccion.com/senales-de-obligacion/1740-es-obligatorio-el-uso-de-los-guantes.html.

[21] «Universal Robots: PolyScope Manual,» [En línea]. Available: https://s3-eu-west-1.amazonaws.com/ur-support-site/44018/Software_Manual_en_Global.pdf.

[22] «Wikipedia: Logo de ROS,» [En línea]. Available: https://es.m.wikipedia.org/wiki/Archivo:Ros_logo.svg.

[23] «Wikipedia: ROS,» [En línea]. Available: https://es.wikipedia.org/wiki/Robot_Operating_System.

[24] «Wiki.ROS,» [En línea]. Available: http://wiki.ros.org/.

[25] «GitHub: RVIZ logo,» [En línea]. Available: https://github.com/ros-visualization/rviz.

[26] «MoveIt, documentación,» [En línea]. Available: https://moveit.picknik.ai/humble/index.html.

[27] «MoveIt, logo,» [En línea]. Available: https://moveit.ros.org/about/press_kit/.

[28] «GitHub: Installing a URCap on a e-Series robot,» [En línea]. Available: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/doc/install_urcap_e_series.md.

[29] «RTVE: Un brazo robótico ultrarrápido que atrapa objetos en el aire,» [En línea]. Available: https://www.rtve.es/noticias/20140512/brazo-robotico-ultrarrapido-atrapa-objetos-aire/937861.shtml.

[30] «Unchained robotics: Precio UR5e,» [En línea]. Available: https://unchainedrobotics.de/en/products/robot/cobot/universal-robots-ur5e.

[31] «PCComponentes: MSI Katana GF66 12UD-081XES Intel Core i7-12700H/16GB/512GB SSD/RTX3050Ti/15.6",» [En línea]. Available: https://www.pccomponentes.com/msi-katana-gf66-12ud-081xes-intel-core-i7-12700h-16gb-512gb-ssd-rtx3050ti-156.

[32] «Amazon, Cable de Red Ethernet.,» [En línea]. Available: https://www.amazon.es/Mr-Tronic-Ethernet-Conectores-

Velocidad/dp/B09NQFLPNC/ref=sr_1_2_sspa?adgrpid=61681371248&gclid=Cj0KCQiApb2bBhDYARIsAChHC9trQGa7LsCu_shVJoEX2pR2XvQ3Dpwysw4LJ3urw5g8tX4Kf632dGwaApjSEALw_wcB&hvadid=601260961116&hvdev=c&hvlocphy=20272&h.

[33] «FS, switch PoE,» [En línea]. Available: https://www.fs.com/es/products/90130.html?country=ES&currency=EUR&languages=Espa%C3%B1ol&paid=google_shopping&utm_country=20272&gclid=Cj0KCQiApb2bBhDYARIsAChHC9swomG0t1sZ0mCPrwjNn85EHNDvCPg2VG7InJ6ZINqfE6GFds8glLwaAibsEALw_wcB .

[34] «Motive: precio,» [En línea]. Available: https://optitrack.com/software/motive/pricing.html.

[35] «Microsoft Windows 10 pro,» [En línea]. Available: https://www.microsoft.com/es-es/d/windows-10-pro/df77x4d43rkt/48dn?rtc=1&activetab=pivot:informaci%C3%B3ngeneraltab.

[36] OptiTrack, «Manual del program Motive,» [En línea]. Available: https://v22.wiki.optitrack.com/index.php?title=Motive_Documentation.

[37] «GitHub, repositorio de Universal Robots ROS driver.,» [En línea]. Available: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver.

[38] «GitHub, repositorio ur5e para MoveIt!,» [En línea]. Available: https://github.com/ros-industrial/universal_robot.

[39] «GitHub, repositiorio mocap_optitrack,» [En línea]. Available: https://github.com/ros-drivers/mocap_optitrack.

[40] «Wiki.ROS, repositorio Mocap OptiTrack,» [En línea]. Available: http://wiki.ros.org/mocap_optitrack.

[41] «Estructura de un robot industrial,» [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/sistema/morfologia.htm.

[42] «Wikipedia: Robótica.,» [En línea]. Available: https://es.wikipedia.org/wiki/Rob%C3%B3tica#Seg%C3%BAn_su_estructura.

[43] «Wikipedia: Brazos robóticos.,» [En línea]. Available: https://es.wikipedia.org/wiki/Brazo_rob%C3%B3tico .

[44] «Wiki.ROS: mocap optitrack,» [En línea]. Available: http://wiki.ros.org/mocap_optitrack.

ANEXO

1. Manual de calibración de las *OptiTrack*.

Para la realización de la calibración de las cámaras se ejecutarán los siguientes pasos:

- Se deberá tener libre de marcadores cualquier zona de captura de las cámaras
- Se clica en «*New Calibration*» en la pantalla principal:

Figura 65.Captura de pantalla de Motive. Fuente [Elaboración propia]

- A continuación, se aplicarán la máscara en el programa para evitar zonas reflectantes, para ello se ciclará:

Figura 66. Captura de pantalla de Motive. Fuente [12]

- Se empezará a realizar la calibración, en la siguiente pestaña al darla a «*Start Wanding*».

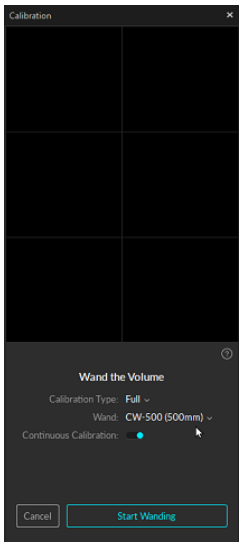

Figura 67. Captura de pantalla de Motive. Fuente [12]

- Se comenzará a mover la vara por toda el área de captación de las cámaras *OptiTrack* hasta que el anillo de las mismas esté completamente iluminado de un único color:

Figura 68. Vara utilizada para la calibración. Fuente [Elaboración propia]

- Una vez realizado el anterior paso se clicará en Start Calculating:

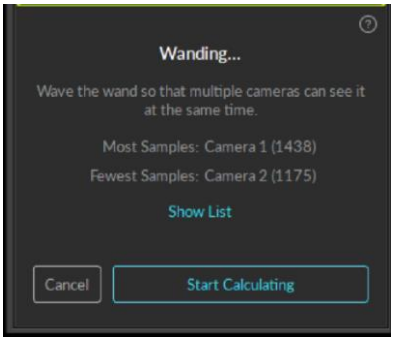

Figura 69. Captura de pantalla de Motive. Fuente [12]

- Se colocará el plano de tierra y el punto de origen a partir del cual se devolverán las coordenadas de los objetos rastreados, una vez colocado se *clicará* en «*Set Ground Plane*». Se realizará de la siguiente manera:

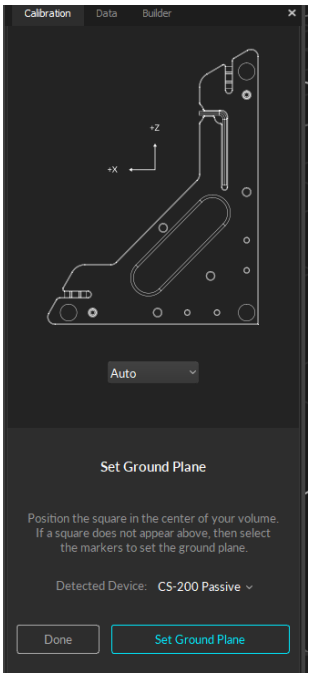

Figura 70.Captura de pantalla de Motive. Fuente [12]

Figura 71. Escuadra real con el plano de tierra. Fuente [Elaboración propia]

- Se podrá realizar un desplazamiento en el plano vertical, por lo tanto, se debe bajar para la escuadra utilizada 45mm.

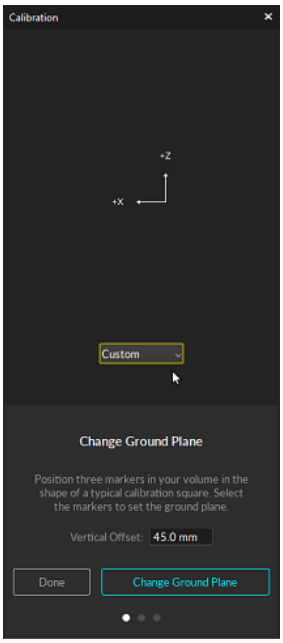

Figura 72. Captura de pantalla de Motive. Fuente [12]

2. Manual de la realización de un *rigid body*.

- Se seleccionaron todos los marcadores que constituyen al *rigid body* en la pantalla:

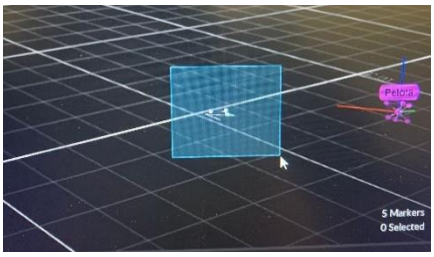

Figura 73.Captura de pantalla de Motive. Fuente [Elaboración propia]

- Desde en el panel siguiente se clicó en «+» y quedó creado el *rigid body*.

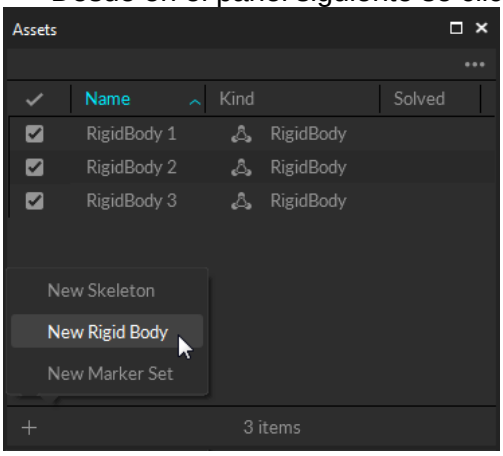

Figura 74. Captura de pantalla de *Motive*. Fuente [12]

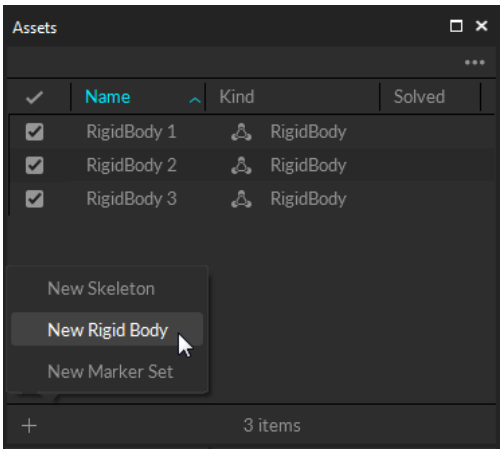

Figura 75. Captura de pantalla de *Motive*. Fuente [12]

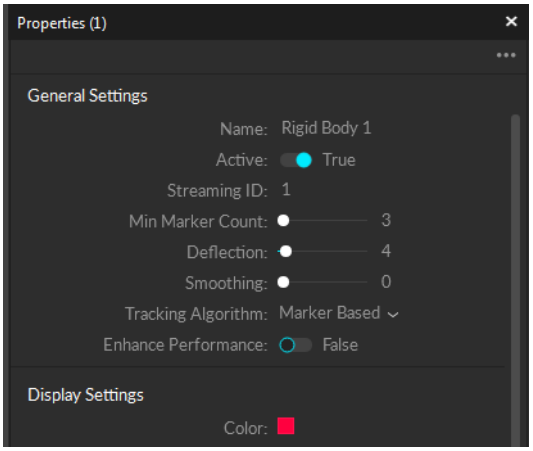

Figura 76. Captura de pantalla de *Motive*. Fuente [12]