ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática

Área de Conocimiento: Ingeniería de Sistemas y Automática

TRABAJO FIN DE GRADO

CONTROL DE ROBOT MANIPULADOR MEDIANTE VISIÓN

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Autor: Martín Godoy, David

Tutor: Vázquez Martín, Ricardo

Cotutora: Góngora Rodríguez, Eva María

Málaga, Octubre de 2.024

Resumen

Hoy en día, la robótica está adquiriendo cada vez más importancia ya que permite tanto desarrollar productos y herramientas como controlar procesos, manipular objetos y usar la inteligencia artificial para ayudar en la toma de decisiones. Este aumento en el interés por la robótica ha provocado la necesidad de que los robots sean cada vez más flexibles en las tareas que poco a poco presentan una mayor complejidad. Una posible respuesta ante esta demanda son los sistemas de control visual, comúnmente denominados *Visual Servoing,* ya que permiten corregir la trayectoria del manipulador en movimiento para evitar obstáculos o para realizar un seguimiento a los objetos.

Por esta razón, en el presente proyecto se ha decidido utilizar la técnica de *Visual* Servoing para controlar el manipulador xArm6 mediante un sistema de visión que clasifica y localiza los objetos del entorno. El software, los modelos de impresión 3D, los vídeos y la documentación se encuentran disponible en Github: https://github.com/Deglox/MANIPULATOR-ROBOT-CONTROL-THROUGH-VISION.

Palabras claves

Control visual, manipulación, clasificación, ROS2.

Abstract

Today, robotics is becoming increasingly important because it allows us to develop products and tools, control processes, manipulate objects and use artificial intelligence to help in decision-making. This increase in interest in robotics has caused the need for robots to be increasingly flexible in tasks because more complex processes are required. A possible response to this demand is visual control systems, commonly called Visual Servoing, due to they allow correct the trajectory of the moving manipulator to avoid obstacles or to track objects.

For this reason, to control the xArm6 manipulator, it has been decided to use the Visual Servoing technique with a vision system (ZED Mini) that provides classification and location of objects in the environment. The software and documentation are available on Github: https://github.com/Deglox/MANIPULATOR-ROBOT-CONTROL-THROUGH-VISION

Key words

Visual control, manipulation, classification, ROS2.

Índice:

Índice de Ilustraciones

1. Introducción

1.1. Antecedentes

La comunidad científica internacional desarrolló un importante esfuerzo desde que Shirai e Inoue (1973) describiera como un bucle de realimentación visual puede ser usado para incrementar la precisión de la posición de un robot y Hill et al., (1979) introdujera el término Visual Servoing. Durante la década de los ochenta y principio de la década de los noventa, se realizan algunas investigaciones pioneras en las que la información visual de la escena ayuda a la realización de tareas robotizadas. En 1981, Geschke (1981) describe la tarea de inserción de un tornillo utilizando la información proveniente del sistema de visión estereoscópico con una frecuencia de obtención de las características de la imagen de 10Hz. En 1984, Weiss (1984) propone el uso de un control adaptativo a sistemas de servo control visual basado en las características de la imagen.

En la actualidad, gracias al avance en la capacidad de cómputo y el aprendizaje profundo los sistemas de visión han alcanzado un nivel de robustez en sus resultados que permiten detectar objetos presentes en el entorno y localizarlos con una fiabilidad elevada

Al principio, los sistemas de control visual que integraban visión por computador trabajaban en bucle abierto sin ninguna realimentación visual, lo que provocaba que la precisión de la operación a realizar dependiera directamente de la fiabilidad del sensor de visión y del efector final del robot

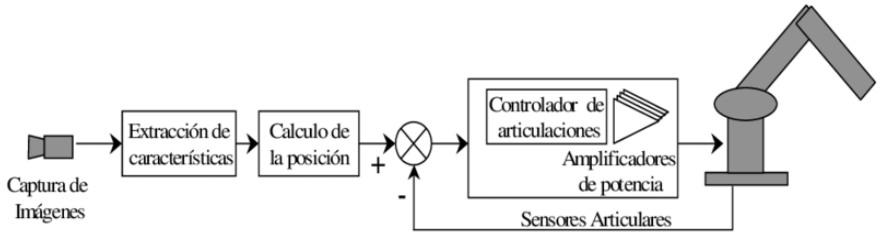

Ilustración 1-1: *Esquema de control visual "Ver y mover" estático. Fuente:* ***[6]***

Durante la década de los 90, surge una mejor alternativa al planteamiento anterior de nominado *Visual Servoing*, el cual consiste en un control en bucle cerrado con realimentación visual. Este nuevo sistema de control visual permite modificar la trayectoria del robot ante posibles movimientos o alteraciones en los objetos del espacio de trabajo.

1.2. Motivación

El control de robots manipuladores mediante técnicas de visual servoing tiene un interés elevado en las actuales aplicaciones industriales debido a que incrementa la flexibilidad de los sistemas robóticos, haciéndolos más versátiles y capaces de interactuar con el entorno sin necesidad de que las piezas de trabajo estén definidas, y a que satisface las necesidades del sector para que los robots se adapten a cambios en la línea de producción. Hoy en día, la industria alimentaria es la que con mayor éxito ha implantado este tipo de sistemas de control visual ya que le permite clasificar los alimentos de una línea de producción.

A causa de este interés en el uso de la técnica de *Visual Servoing* en la industria*, en* el presente proyecto se ha decidido implantar dicho sistema de control visual para detectar, localizar y clasificar los objetos presentes en el entorno.

1.3. Objetivos

En este proyecto de Trabajo de Fin de Grado se realiza un control de los movimientos del manipulador xArm6 a partir de la realimentación visual de un sistema de visión, para así lograr realizar un “Pick and place” (Detectar, localizar y manipular los objetos presentes en el entorno) y una clasificación de objetos.

Para alcanzar dicho objetivo, se ha necesitado un sistema de visión que extraiga las características más importantes de cada uno de los objetos identificados, un software que controle el movimiento del robot a partir de la realimentación visual y usar todas las competencias adquiridas a lo largo de la formación universitaria en el Grado de Ingeniería Electrónica, Robótica y Mecatrónica en materias como Control y programación de robots y Ampliación de robótica.

1.4. Estructura de la memoria

Esta memoria consta de los siguientes capítulos:

1) **Introducción:** Se presenta brevemente la técnica de control visual utilizada, así como se explican los objetivos y motivos del proyecto.

2) **Control de manipuladores mediante visión:** Se plantea la solución al problema del control del manipulador mediante el sistema de visión.

3) **Percepción:** Se describe en profundidad el sistema de visión empleado a lo largo de este trabajo.

4) **Control del manipulador xArm:** Se describe el manipulador xArm6, la aplicación informática utilizada y las herramientas de ROS2 que se han utilizado para controlar el robot.

5) **Integración de la percepción y el control del manipulador:** Se presenta la disposición de cada uno de los elementos del sistema de control visual.

6) **Experimentos:** Se explican en profundidad los dos tipos de experimentos realizados en el trabajo.

7) **Conclusiones y trabajo futuro:** Se muestra lo logrado durante el desarrollo de este trabajo y se plantean las diferentes posibilidades futuras en los que implementar este proyecto.

2. Control de manipuladores mediante visión

En este proyecto se ha seleccionado el sistema de control visual de “Visual Servoing” para controlar el movimiento del efector final del robot ya que ofrece beneficios como una mayor flexibilidad y adaptación en las tareas.

2.1. Visual servoing

La técnica de *Visual Servoing* consiste en una realimentación visual constante que permite corregir posibles errores en la posición del objetivo o modificar la trayectoria del robot ante posibles alteraciones en los objetos del espacio de trabajo.

La técnica conocida como *Visual Servoing* se puede clasificar de diferentes como, por ejemplo, según si se ha aplicado un control directo, llamado “Control visual directo” (Ilustración 2.2), o no, llamado “Ver y mover dinámico (Ilustración 2.1)”. La mayor diferencia entre ambos sistemas es que el “Ver y mover” dinámico no usa el bucle externo para controlar las posiciones de las articulaciones del manipulador al igual que el “Control visual directo”, sino que usa un bucle interno que se encarga exclusivamente de realizar esa tarea de control articular del robot.

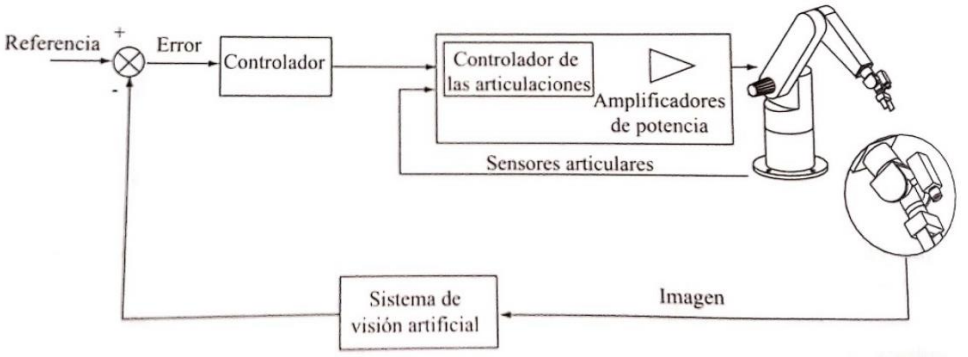

Ilustración 2-1**:” Ver y mover” dinámico Fuente: **[2].

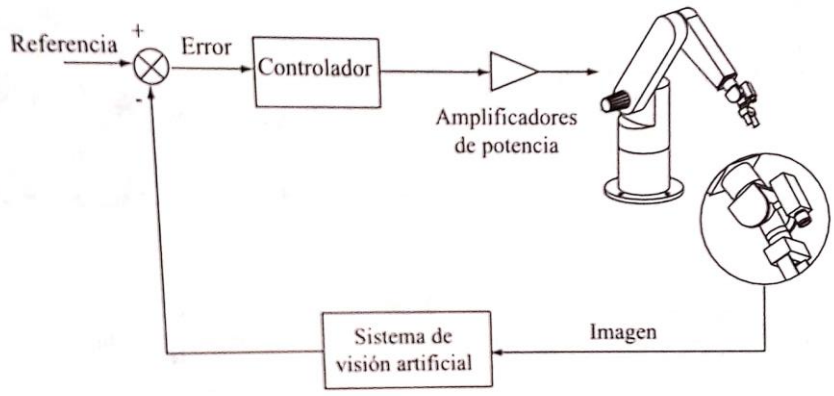

Ilustración 2-2: *Control visual directo. Fuente:* ***[2].***

Otra clasificación atendiendo a la acción de control utilizada en el sistema de control visual está basada en posición (PBVS) o en imagen (IBVS). En el primer caso, donde el control se ejecuta basado en posición, se utiliza las características visuales observadas, una cámara calibrada y un conocido modelo geométrico del objeto para determinar la pose del objetivo con respecto a la cámara (Ilustración 2.3). Sin embargo, a pesar de que este control visual tenga buenos algoritmos para estimar la pose, suele tener mucho gasto computacional y depende mucho de la precisión de la calibración de la cámara y del modelo de la geometría del objeto.

Por otro lado, si el control se ejecuta basado en imágenes la ley de control se expresa en términos de características en imágenes, es decir, se minimiza el error entre las características de las imágenes medidas y las imágenes deseadas, consiguiendo un control robusto ante errores de calibración (Ilustración 2.4). Este tipo de control es más apropiado cuando no se dispone de un modelo geométrico de la tarea a desarrollar, a diferencia del anterior caso donde se necesita gran precisión para obtener buenos resultados en la estimación de la pose.

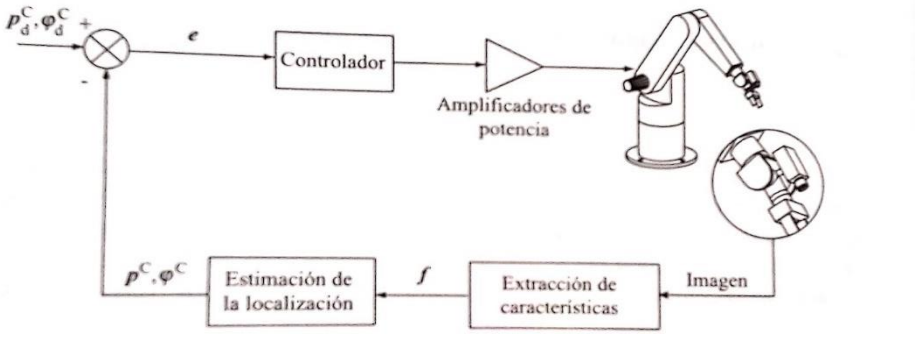

Ilustración 2-3: *Control basado en posición. Fuente:* ***[2]***

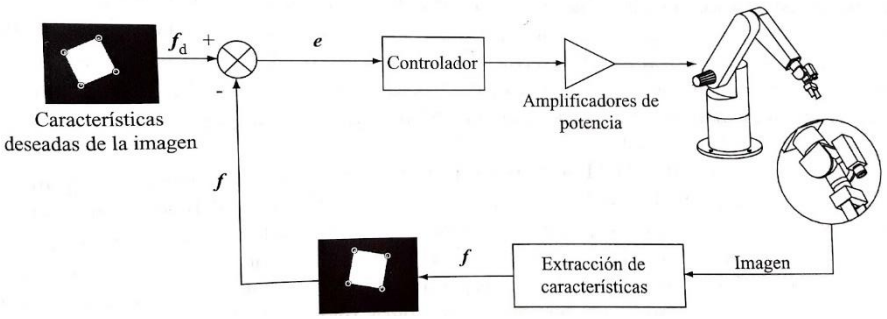

Ilustración 2-4: *Control basado en imagen. Fuente* ***[2].***

En este tipo de técnica también se pueden emplear distintas configuraciones, dependiendo de la ubicación del sistema de visión respecto del robot. La primera configuración ("eye-in-hand") ubica la cámara en el extremo del manipulador (Ilustración 2.5 a), lo que implica que los objetos del espacio de trabajo se encuentran en su campo visual y la relación entre el sistema de coordenadas del extremo del robot y el de la cámara sea conocida y fija durante el movimiento del manipulador.

La otra posible configuración ("eye-to-hand") consiste en fijar la cámara en un punto externo al manipulador (Ilustración 2.5 b), causando que no haya una relación mecánica entre el robot y la cámara, aunque si es conocida la relación entre la cámara y el sistema de coordenadas asociado a la base del robot manipulador.

En el presente trabajo se ha escogido la configuración de la cámara montada en el extremo del robot ("eye-in-hand") ya que la relación entre la posición y la orientación del efector final y de la cámara es fija, facilitando los cálculos de la posición del extremo del robot. Además, esta configuración permite obtener una mejor visión del entorno de trabajo desde el punto de vista de la herramienta, evitando oclusiones y un mayor detalle de los objetos.

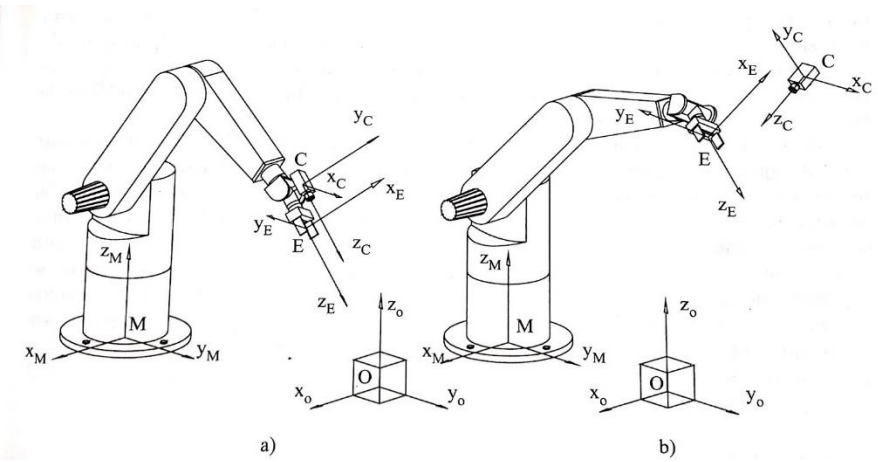

Ilustración 2-5: a) *Configuración de la cámara montada en el extremo del robot.* ***b)*** *Configuración de cámara externa. Fuente:* ***[2].***

2.2. Detección de objetos

La técnica de detección de objetos permite a los sistemas de visión localizar y clasificar los objetos dentro de una imagen gracias a los algoritmos de IA como las redes neuronales (Ilustración 2.6). Esta técnica se puede realizar mediante el método aprendizaje profundo o *Deep Learning* de dos maneras diferentes:

- Utilizando detectores de objetos ya entrenados con grandes conjuntos de datos, para detectar objetos comunes como personas o animales.

- Creando y entrenando un detector de objetos personalizado mediante el aprendizaje por transferencia, para obtener una red de detección de objetos que mejor se ajuste a su aplicación.

Esta técnica funciona mejor cuando se tiene una gran cantidad de imágenes y una GPU para reducir el tiempo necesario para entrenar el modelo.

En la actualidad, la detección de objetos es esencial para muchas aplicaciones prácticas como, por ejemplo, en la conducción autónoma para evitar los obstáculos del entorno, en la robótica para facilitar una mejor comprensión del entorno por parte del robot o más concretamente en el *Bin picking,* ya que se necesita reconocer y seleccionar una pieza en específico dentro de un contenedor.

Ilustración 2-6: *Detección de objetos. Fuente:* ***[8]***

La cámara estéreo ZED Mini es el sistema de visión elegido para este proyecto y se encarga de realizar la detección de objetos mediante el uso de inteligencia artificial y de redes neuronales, pudiendo llegar a identificar ciertas clases de objetos diferentes.

2.3. Localización espacial

La localización espacial se define como la capacidad de ubicar en el espacio los objetos percibidos de acuerdo con los parámetros de dirección y distancia. Esta técnica junto la detección de objetos, son importantes en procesos industriales como tareas de "Bin picking" ya que ubican e identifican una pieza en concreto dentro de un conjunto de objetos.

La localización espacial junto a la visión por computador y a la tecnología SLAM (Simultaneous Localization and Mapping) de la cámara estéreo ZED Mini, permite localizar los objetos del entorno y obtener la pose del sistema de visión.

3. Percepción

Una de las partes fundamentales de la técnica de *Visual Servoing* es el sistema de visión debido a que proporciona una realimentación visual constante necesaria para controlar los movimientos del manipulador. Sabiendo que el sistema de visión debe detectar, localizar y clasificar se ha optado por la cámara estéreo *ZED Mini* ya que localiza con precisión los objetos desde una misma ubicación.

3.1. Cámara estéreo ZED mini

La *ZED Mini* es una cámara estéreo (Ilustración 3.1) que presenta un nuevo modo de detección de profundidad ultra, una tecnología inercial para un mejor seguimiento de movimiento (positional tracking) y un diseño compacto para una integración sencilla.

Algunas de las características importantes de esta cámara son una unidad de medición inercial (IMU) de seis grados de libertad para un seguimiento preciso del movimiento, una detección de profundidad HD con el nuevo modo Ultra, un tamaño pequeño y compacto con separación entre cada cámara de 6.5 cm, una estructura de aluminio para mayor robustez e integración y un USB Type-C para transferencia de datos en alta velocidad (Ilustración 3.2).

Ilustración 3-1: *Vista en perspectiva de la cámara ZED Mini. Fuente:* ***[12]***

Ilustración 3-2: *Puerto USB Type-C de la ZED Mini. Fuente:* ***[12]***

Los sensores que posee la ZED Mini son el giroscopio y el acelerómetro. El acelerómetro detecta la aceleración instantánea de la cámara estéreo, lo que ayuda a determinar si la cámara se está moviendo más rápido o más lenta, en cualquier dirección, con un valor preciso (m/s^2). Además, el acelerómetro permite detectar eventos como caídas libres ya que, cuando este sensor se encuentra estático, mide el valor y la dirección de la aceleración de la gravedad aplicada en la cámara. Por otro lado, el giroscopio mide la velocidad angular de la cámara en grados por segundo (grados/s) y junto al acelerómetro, estima la orientación de la cámara a alta frecuencia (Ilustración 3.3).

DATOS RESULTANTES	DESCRIPCIÓN	UNIDADES
Acelerómetro, giroscopio (IMU)		
aceleración lineal	Fuerza de aceleración aplicada en los tres ejes físicos (x, y, z), incluida la fuerza de gravedad. Los valores se corrigen por sesgo, escala y desalineación.	$m/^{s2}$
aceleración_lineal_sin calibrar	Fuerza de aceleración aplicada en los tres ejes físicos (x, y, z), incluida la fuerza de gravedad. Los valores no están calibrados.	$m/^{s2}$
covarianza_aceleración_lineal	Ruido de medición de la aceleración lineal no calibrada del acelerómetro. Proporcionado como una matriz de covarianza de 3x3.	
velocidad angular	Velocidad de rotación alrededor de cada uno de los tres ejes físicos (x, y y z). Los valores se corrigen por sesgo, escala y desalineación.	grados/s
velocidad_angular_sin calibrar	Velocidad de rotación alrededor de cada uno de los tres ejes físicos (x, y y z). Los valores no están calibrados.	grados/s
covarianza_velocidad_angular	Ruido de medición de la velocidad angular no calibrada del giroscopio. Proporcionado como una matriz de covarianza de 3x3.	
pose	Pose de IMU en el espacio libre, compuesta de posición y orientación. Obtenido de la fusión del sensor de acelerómetro y giroscopio. Proporcionado como una transformación.	
pose_covarianza	Ruido de medición de la orientación de la pose. Proporcionado como una matriz de covarianza de 3x3.	
cámara_imu_transform	Transforme entre cuadros IMU y cámara izquierda.	

Ilustración 3-3: *Tabla de datos que proporcionan los sensores de la ZED Mini. Fuente:* ***[11]***

En este proyecto se ha seleccionado la *Jetson AGX Xavier* por ser la opción perfecta para máquinas autónomas y por su GPU de 32 TeraOPS de cálculo máximo y 750 Gbps de E/S de alta velocidad, esencial para el procesamiento del sistema de visión. El SDK de ZED permite agregar profundidad, detección de movimiento e IA espacial a su aplicación e incluye aplicaciones, herramientas y proyectos de muestra con código fuente (Ilustración 3.4).

A la hora de trabajar con la cámara estéreo en la *Jetson AGX Xavier,* es necesario instalar el *ZED-SDK* de *Stereolabs* para realizar el procesamiento del sistema de visión en el ordenador. La versión del *ZED-SDK* se ha elegido en función del JetPack instalado en la *Jetson AGX Xavier*.

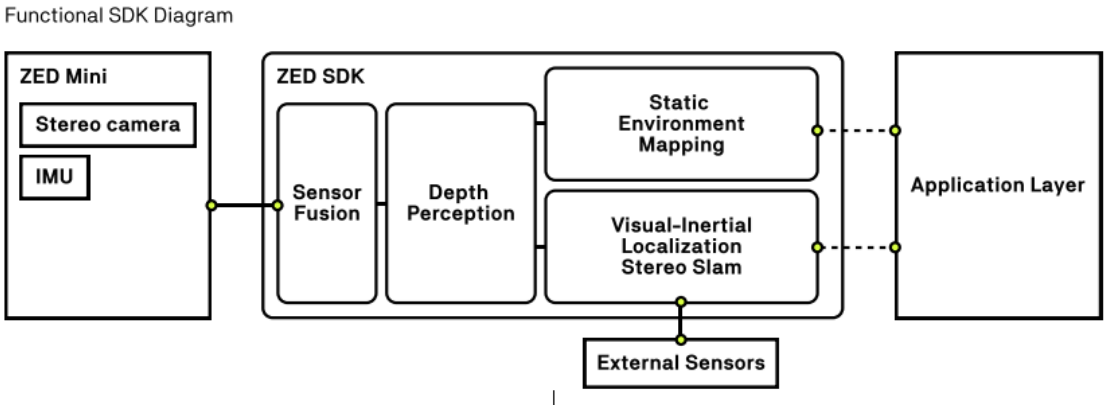

Ilustración 3-4: *Tabla de datos que proporcionan los sensores de la ZED Mini. Fuente:* ***[27]***

Algunas de las herramientas que ofrece el *ZED-SDK* del sistema de visión son:

a) ZED Explorer: Esta aplicación permite cambiar la resolución del video, la relación de aspecto y los parámetros de la cámara, además de capturar instantáneas de alta resolución y videos 3D (Ilustración 3.5).

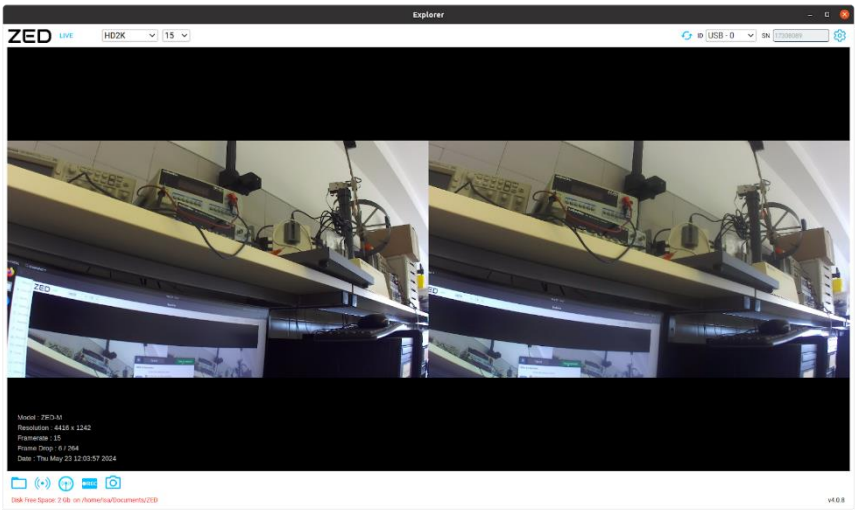

Ilustración 3-5: *Ejemplo de uso de la aplicación ZED Explorer.*

b) ZED Depth Viewer: Esta herramienta utiliza el SDK de ZED para capturar y mostrar el mapa de profundidad y las nubes de puntos 3D (Ilustración 3.6).

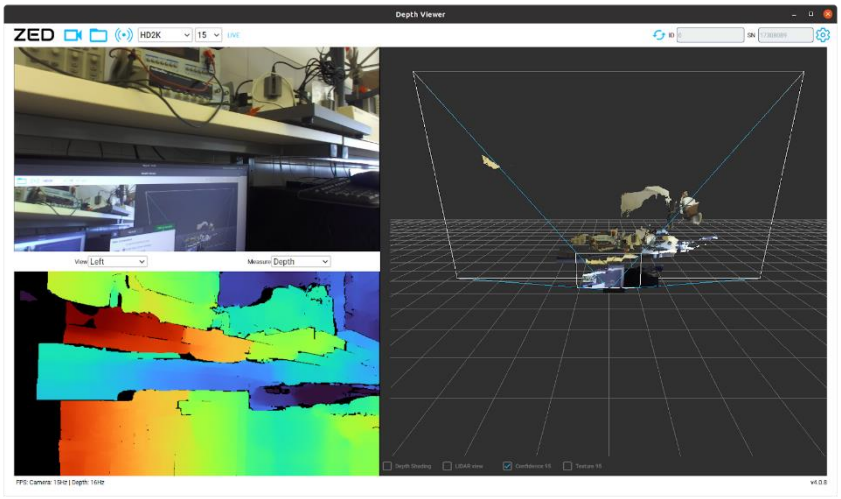

Ilustración 3-6: *Ejemplo de uso de la aplicación ZED Depth Viewer.*

c) ZED Diagnostic: Esta aplicación comprueba si hay algún problema con el *ZED-SDK* o con la cámara estéreo *ZED Mini* (Ilustración 3.7).

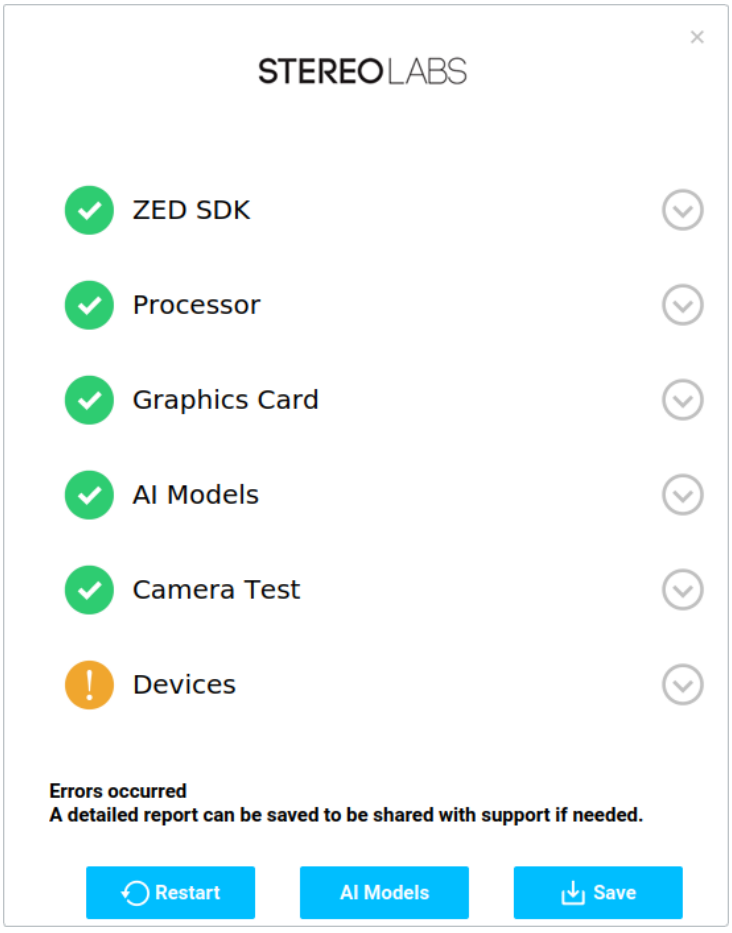

Ilustración 3-7: *Ejemplo de la aplicación ZED Diagnostic.*

d) **ZED Sensor Viewer:** Esta función muestra los valores del giroscopio, acelerómetro y orientación de la cámara, así como una representación gráfica de los valores o una visualización 3D de cómo está orientada la cámara en ese momento (Ilustración 3.8).

Ilustración 3-8: *Ejemplo de la aplicación ZED Sensor Viewer.*

e) **ZED Calibration:** Permite realizar una calibración intrínseca en la cámara manualmente (Ilustración 3.9).

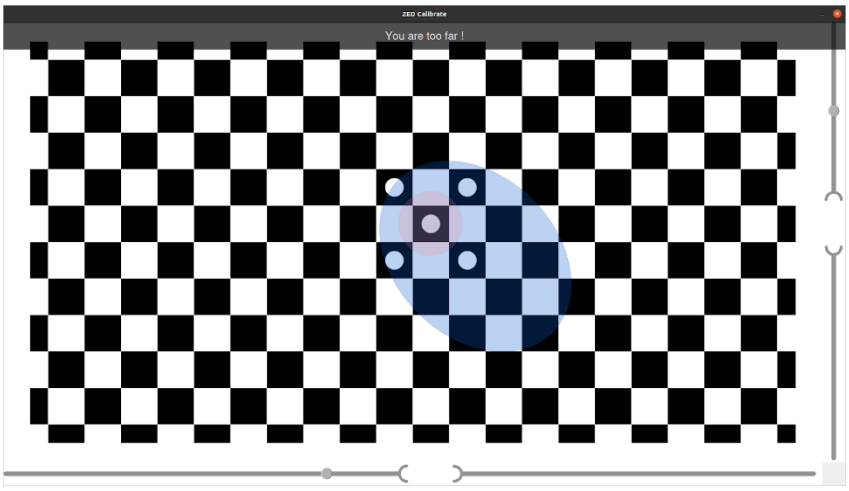

Ilustración 3-9: *Ejemplo de la aplicación ZED Calibration.*

3.2. Detección de objetos

La cámara estéreo ZED Mini, gracias a la inteligencia artificial y a las redes neuronales con aprendizaje profundo o “Deep learning”, detecta y rastrea distintas clases de objetos. Además, las redes neuronales no son solo capaces de clasificar objetos tan distintos como los ordenadores y las frutas, sino que también diferencian productos muy parecidos como las naranjas, las manzanas y los plátanos. La lista de las diferentes clases y subclases que el sistema de visión distingue se muestran en la Ilustración 3.10 y en la Ilustración 3.11 respectivamente. El hecho de que la cámara estéreo ZED Mini diferencie y clasifique tanta variedad de objetos, nos indica que las redes neuronales de la misma han sido entrenadas con una gran cantidad de imágenes de cada objeto mediante el método de aprendizaje profundo *o Deep Learning.*

`enum OBJECT_CLASS` strong

Lists available object classes.

Enumerator	
PERSON	For people detection
VEHICLE	For vehicle detection (cars, trucks, buses, motorcycles, etc.)
BAG	For bag detection (backpack, handbag, suitcase, etc.)
ANIMAL	For animal detection (cow, sheep, horse, dog, cat, bird, etc.)
ELECTRONICS	For electronic device detection (cellphone, laptop, etc.)
FRUIT_VEGETABLE	For fruit and vegetable detection (banana, apple, orange, carrot, etc.)
SPORT	For sport-related object detection (sport ball, etc.)

Ilustración 3-10: *Lista de clases que puede detectar la ZED Mini. Fuente:* ***[16]***

enum OBJECT_SUBCLASS strong

List available object subclasses.
Given as hint, when using object tracking an object can change of **sl::OBJECT_SUBCLASS** while keeping the same **sl::OBJECT_CLASS** (i.e.: frame n: MOTORBIKE, frame n+1: BICYCLE).

Enumerator	
PERSON	PERSON
PERSON_HEAD	PERSON
BICYCLE	VEHICLE
CAR	VEHICLE
MOTORBIKE	VEHICLE
BUS	VEHICLE
TRUCK	VEHICLE
BOAT	VEHICLE
BACKPACK	BAG
HANDBAG	BAG
SUITCASE	BAG
BIRD	ANIMAL
CAT	ANIMAL
DOG	ANIMAL
HORSE	ANIMAL
SHEEP	ANIMAL
COW	ANIMAL
CELLPHONE	ELECTRONICS
LAPTOP	ELECTRONICS
BANANA	FRUIT_VEGETABLE
APPLE	FRUIT_VEGETABLE
ORANGE	FRUIT_VEGETABLE
CARROT	FRUIT_VEGETABLE
SPORTSBALL	SPORT

Ilustración 3-11**: Lista de subclases de objetos detectables por la ZED Mini. Fuente: **[16]

3.3. Dimensiones y localización de objetos

El ZED-SDK, una vez haya detectado el objeto, calcula la caja de contorno 2D o la *Bounding box 2D* (Ilustración 3.12) de la pieza para saber cuáles son los píxeles de la imagen o nube de puntos que pertenecen a dicho objeto. A partir de esta información y con la ayuda del mapa de profundidad, el cual consiste en almacenar un valor (Z) para cada píxel (X, Y) de la imagen capturada por el sistema de visión, la cámara calcula la caja de contorno 3D o la *Bounding box 3D* del objeto (Ilustración 3.13). Con la caja de contorno 3D o la *Bounding box 3D* del objeto, la cámara estéreo *ZED Mini* nos proporciona una estimación de las dimensiones, de la posición y de la velocidad de cada uno de los objetos detectados gracias a la utilización de los datos del *depth sensing*. La detección de profundidad consiste en determinar las distancias entre objetos y ver el mundo en tres dimensiones. Algunas de las características del d*epth sensing son*:

- Una mayor distancia para la medición de la profundidad (hasta 20m).
- Una velocidad de cuadros de captura de profundidad de 100 FPS.
- Un mayor campo de visión (hasta 110° (H) x 70° (V)).
- Un funcionamiento tanto en interiores como en exteriores, a diferencia de sensores activos como *structured-light* o *time of flight (ToF).*

Ilustración 3-12: *Bounding box 2D. Fuente:* ***[14]***

Ilustración 3-13: *Bounding box 3D. Fuente:* ***[14]****.*

3.4. Seguimiento

El seguimiento posicional o *positional tracking* se define como la capacidad de un dispositivo para estimar su posición en relación con el entorno que lo rodea. La ZED Mini es capaz de comprender su posición y su orientación en el espacio y ofrecer un seguimiento posicional completo de seis grados de libertad mediante la utilización de visión por computadora y tecnología SLAM estéreo (Ilustración 3.14).

Esta técnica de seguimiento es muy importante en la robótica ya que los robots necesitan percibir y comprender su entorno, así como operar en entornos previamente desconocidos y no estructurados donde a veces no están disponibles métodos externos confiables de localización como el GPS.

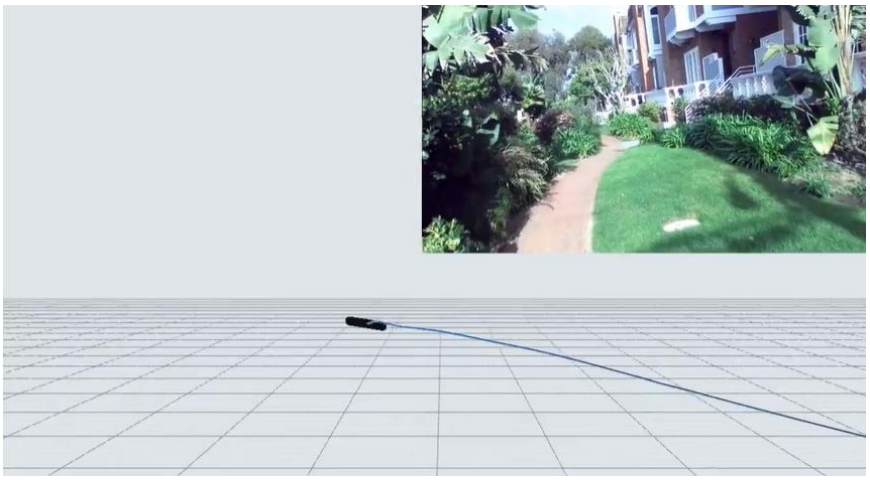

Ilustración 3-14: *Positional tracking de la ZED Mini. Fuente:* ***[18].***

La cámara puede también rastrear los objetos de la imagen, incluso estando en movimiento, con la ayuda de los datos del módulo de seguimiento posicional o *positional tracking.*

En este proyecto, la posición y orientación de la cámara en el entorno se obtiene mediante ROS2 a partir de los datos de los sensores que publica el paquete “zed-ros2-wrapper”, pero la orientación es dada en cuaternios, así que es necesario pasar de cuaternios a grados para un mejor entendimiento de la orientación. Sin embargo, lo que se necesita no es la pose de la cámara sino la pose del efector final así que gracias a que la relación de posición y orientación entre el sistema de coordenadas de la cámara y el efector final es constante en todo momento, se calcula la pose del extremo del robot a partir de la pose de la cámara y de la pose del efector final respecto al sistema de la cámara.

4. Control de manipulador xArm

El manipulador elegido para este proyecto ha sido el xArm 6 de UFactory (Ilustración 4.1) ya que es un robot multieje que equilibra perfectamente potencia y tamaño.

Ilustración 4-1: *Robot colaborativo xArm 6. Fuente:* ***[21]***

Algunas de las características de este robot son:

- Un alcance de 700 mm, una carga útil de 5 kilogramos y tiene seis grados de libertad.

- Un robot duradero para automatización ya que posee accionamiento armónico y servomotores de grado industrial para garantizar un funcionamiento 24 horas al día, 7 días a la semana, sin parar y está fabricado con fibra de carbono con un peso de 15 Kg que le permite un despliegue más sencillo.

- Una implementación flexible con función segura ya que tiene disponible la detección de colisiones para garantizar la seguridad, también tiene enseñanza manual, es liviano, tiene “space-saving” y es fácil de volver a implementar en múltiples aplicaciones sin cambiar el diseño de producción. Por ello, es perfecto para tareas recurrentes.

- Una interfaz gráfica para programación fácil de principiantes que es compatible para varios sistemas operativos, incluido macOS y Windows y tiene una tecnología basada en web compatible con los principales navegadores.

- Un SDK Python/C++ de código abierto completamente funcional para una programación más flexible, también hay paquetes de ROS/ROS2 listos para funcionar, además de algunos códigos de ejemplo para ayudar a desplegar el brazo robótico sin problemas.

4.1. Entorno de programación

En el presente proyecto se ha utilizado como editor de código fuente la aplicación informática llamada *Visual Studio Code*. Se ha optado por esta plataforma ya que, además de depurar, editar y compilar el código, permite instalar extensiones que facilitan al programador desarrollar el software.

Las extensiones usadas durante este trabajo son la extensión de C/C++ que agrega soporte de lenguaje para C/C++ a Visual Studio Code y la extensión de ROS que proporciona soporte para el desarrollo del sistema operativo de robot (ROS) para ROS1 y ROS2 en Windows y Linux. Se han seleccionado estas extensiones porque mejoran la experiencia de programación en el entorno de ROS2 (Robot Operating System) y facilitan la detección de errores y la navegación entre los archivos de la carpeta abierta en dicha aplicación.

Dentro de *Visual Studio Code* ha sido necesario crear una carpeta llamada *vscode* para configurar la aplicación. Dicha carpeta contiene dos archivos:

- **c_cpp_properties.json:** Este archivo se encarga de indicarle al editor de código fuente donde se ubican las librerías incluidas en el código.

- **tasks.json**: Este archivo automatiza la compilación del código mediante un atajo de teclado en vez de ejecutar el código de compilación directamente del terminal.

4.2. Programación en ROS2

Robot Operating System (ROS) es un framework open source que nos permite generar software para robots, nos aporta abstracción del hardware y servicios estándar como una API y una interfaz de comunicación entre los componentes del robot. Por esta razón, se ha utilizado esta plataforma para programar el software encargado de

controlar los movimientos del manipulador. En esta plataforma hay una gran variedad de herramientas, pero en este trabajo se han utilizado los servicios, los nodos, los topics, los archivos launch y los mensajes personalizados:

1. Los nodos son archivos ejecutables dentro de un paquete de ROS2 capaces de publicar o suscribirse a un topic y utilizar o proporcionar un servicio.

2. Los servicios son funciones declaradas en un nodo, que pueden ser llamadas por otros nodos remotos.

3. Los topics se asemejan a un espacio de memoria al que se le asigna un nombre y los nodos llamados publicadores y suscriptores, se conectan para escribir o recibir mensajes.

4. Los archivos *launch* sirven para ejecutar varios nodos en una sola línea de comando.

5. Los mensajes personalizados son una serie de datos organizadas en diferentes campos que los fabricantes o desarrolladores han creado para su software.

El software de este proyecto está formado principalmente por tres paquetes: el *xarm_zed,* el *xarm_ros*" (Propietario: UFactory) y el *zed-ros2-wrapper* (Propietario: Stereolabs). El paquete "xarm_ros2" ofrece servicios que son capaces tanto de activar o desactivar la herramienta del robot como de inicializar y mover el manipulador hacia una posición concreta con una orientación y velocidad determinada. Este paquete también cuenta con archivos *launch* que ejecutan los nodos encargados de la conexión entre la Jetson AGX Xavier y el manipulador xArm6 y con topics donde sus nodos publican datos como el estado del robot.

En el paquete "zed-ros2-wrapper" se han usado especialmente los topics y los archivos *launch.* De entre toda la información recibida por los topics solo se ha utilizado la pose de la ZED Mini y la posición, las dimensiones y la clase del objeto detectado ya que es justo lo necesario para controlar el manipulador adecuadamente. Los archivos *launch se han usado para* establecer la conexión entre la ZED Mini y la computadora.

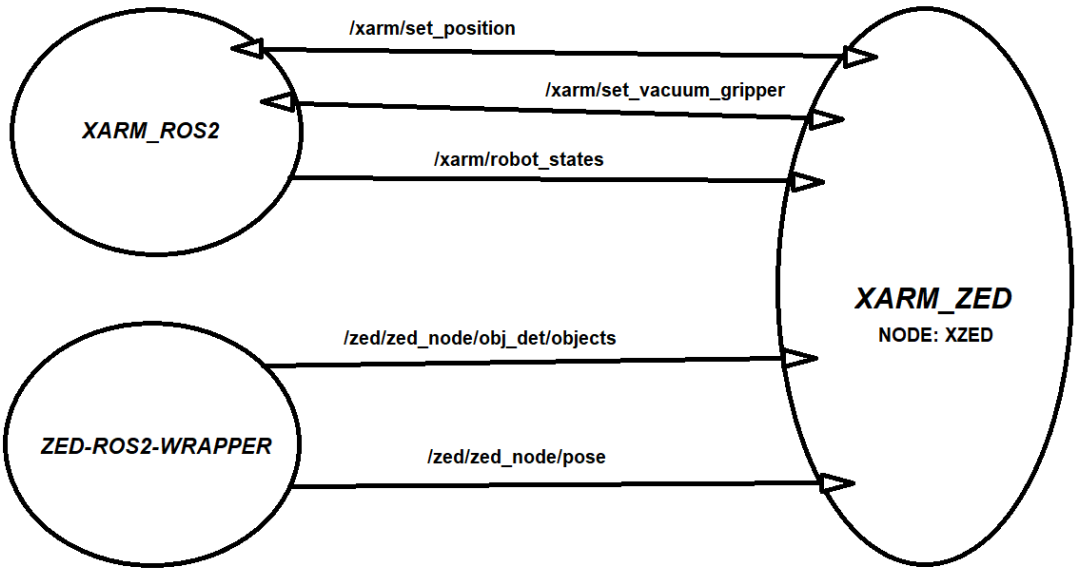

Ilustración 4-2: Esquema de flujo del software en ROS2.

4.3. Nodo de control de movimientos

En el paquete “xarm_zed” se ha creado un único nodo llamado *xzed*. Este nodo se encarga principalmente de controlar los movimientos del manipulador, según las preferencias del usuario, a partir de la coordinación de varias funciones declaradas en el mismo.

La función principal (main) se encarga de configurar la visualización de datos y las funciones según la tarea que desea realizar el usuario y de indicar si el robot ha finalizado la tarea. Las otras funciones que conforman el software de control son:

- *rob_stateCallback* se encarga de recibir e indicar cuál es el estado del manipulador en cada instante.
- *activate vacuum* crea un cliente al servicio de “xarm_ros2” que activa o desactiva la herramienta.
- *keyloop* lee del teclado los caracteres que introduce el usuario.
- *test_robot_movement* es una prueba de movimiento para comprobar el correcto funcionamiento del servicio encargado de mover el robot.
- *poseCallback* recibe y muestra la posición y orientación de la ZED Mini en metros y grados.

- *doCallback* se encarga de clasificar, mostrar y calcular la posición respecto de la base del robot de los objetos deseados por el usuario.

- *move_robot* se encarga de mover el manipulador con una pose, velocidad y aceleración determinada.

5. Integración percepción y control de manipulador

El siguiente paso es integrar todos los componentes del sistema de control visual: El software, el sistema de visión y el manipulador.

5.1. Ubicación de la cámara

La ZED Mini se ubica cerca del efector final más específicamente en la posición X= 7cm, Y=0cm y Z=0.05cm y con una orientación de 180º sobre el eje Y y de 90º sobre el eje Z con respecto al sistema de coordenadas del efector final.

Los diferentes sistemas de coordenadas disponibles en la cámara estéreo ZED Mini son (Ilustración 5.1):

- Right-handed, y-down (image - default).
- Left-handed, y-up (Unity).
- Right-handed, y-up (OpenGL).
- Left-handed, z-up (Unreal Engine).
- Right-handed, z-up.
- Right-handed, z-up, x-forward (ROS).

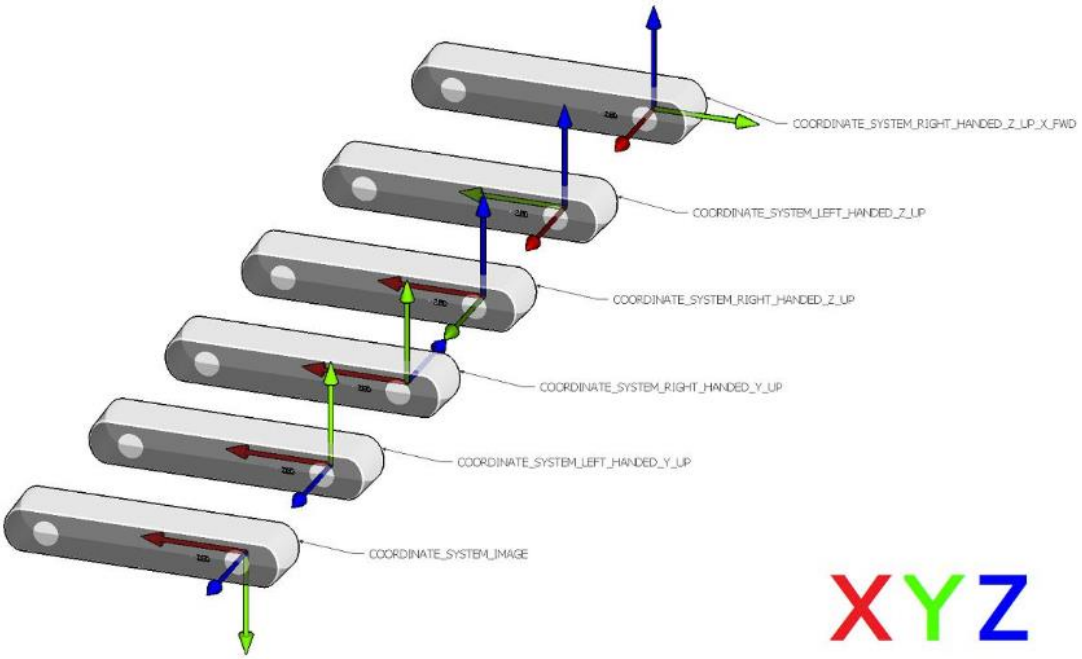

Ilustración 5-1: *Sistemas de coordenadas disponibles en la cámara estéreo ZED Mini. Fuente:* ***[20]***

El sistema de coordenadas de la cámara se ha elegido en función de cuál facilita más los cálculos de las posiciones de los objetos y del efector final y posee una transformación entre la cámara y la base del robot más sencilla. El nombre del sistema de coordenadas seleccionado para este trabajo es el "COORDINATE _SYSTEM_ RIGHT_HANDED_Y_UP" (Ilustración 5.2) ya que es el único que posee el eje Z en el mismo sentido que en el sistema de coordenadas de la base del robot, lo que facilita los cálculos de la localización del objeto y el efector final y la transformación entre ambos sistemas.

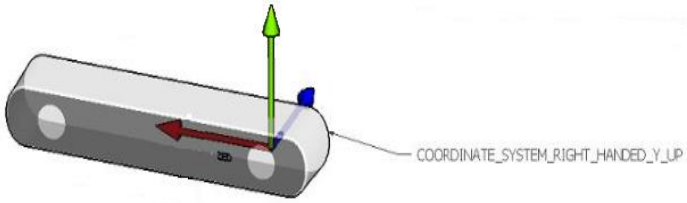

Ilustración 5-2: *Sistemas de coordenadas de la de la ZED Mini. Fuente:* ***[20]***

La integración de la cámara en el efector final del manipulador se ha realizado mediante la impresión 3D de un soporte. Este soporte para la cámara ha tenido varias versiones:

- **V1.0:** En un principio se optó por un sistema de agarre a presión en forma de pinza para fijar el soporte a la parte superior del efector final. Para que la cámara no se moviera o se cayera, se creó un compartimento donde se inserta la cámara y una cubierta para sujetar firmemente el sistema de visión (Ilustración 5.3 y 5.4).

- **V2.0:** Debido a que el sistema de fijación del modelo anterior no realizaba una buena sujeción en el efector final del manipulador, se diseñó una segunda versión donde el sistema de sujeción utilizaba tornillos para fijar el soporte en el extremo del xArm6 y las dimensiones del soporte se ajustaron para ubicar la cámara en el soporte sin dificultad (Ilustración 5.5 y 5.6).

- **V3.0:** En esta versión se modificó el soporte para que la cubierta tuviera una mejor extracción y firmeza. Por último, se modificó la posición de los tornillos ya que no fijaban el soporte en la dirección correcta (Ilustración 5.7 y 5.8).

- **V4.0:** En la última versión se corrigieron varias partes del soporte como la altura de la plataforma que se atornilla al efector final, la ubicación de las cámaras de la ZED Mini para orientarla correctamente, el sistema de extracción de la cubierta para proporcionarle mayor robustez (Ilustración 5.9 y 5.10).

En la última versión del soporte de la cámara se hicieron dos modelos 3D diferentes debido a que se utilizaron dos herramientas en el proyecto (Ilustración 5.11 y 5.12).

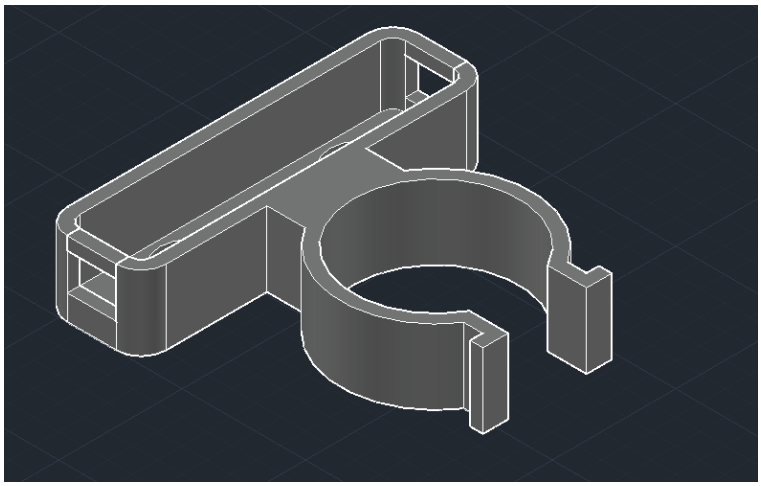

Ilustración 5-3: *Modelo 3D de la primera versión del soporte de la cámara.*

Ilustración 5-4: *Primera versión del soporte fijada al robot.*

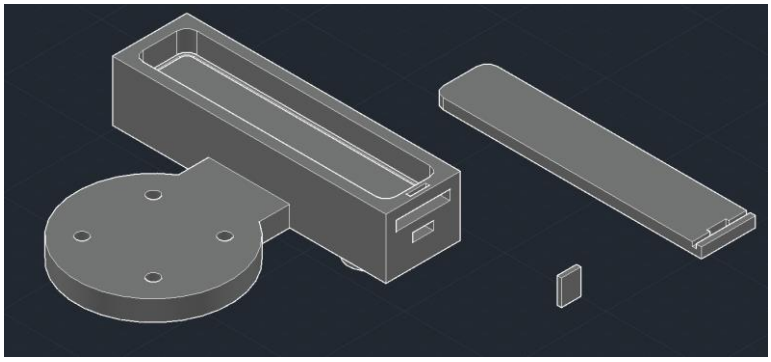

Ilustración 5-5: *Modelo 3D de la segunda versión del soporte de la cámara.*

Ilustración 5-6: *Segunda versión del soporte fijada al robot.*

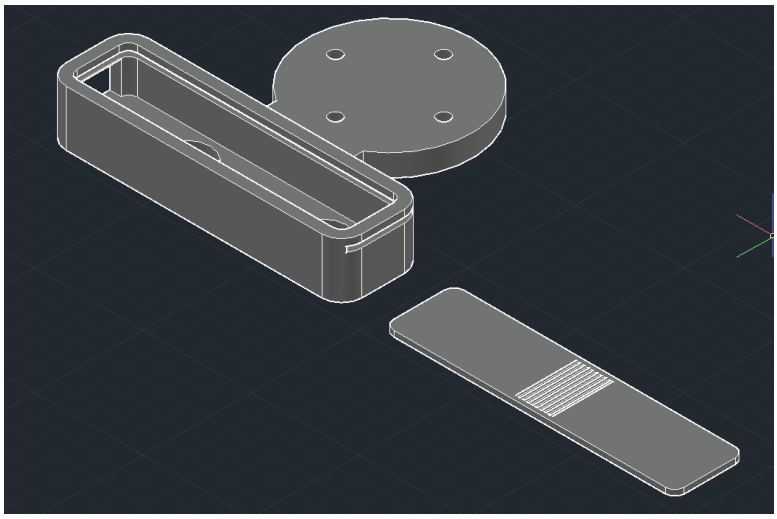

Ilustración 5-7: *Modelo 3D de la tercera versión del soporte de la cámara.*

Ilustración 5-8: *Tercera versión del soporte fijada al robot.*

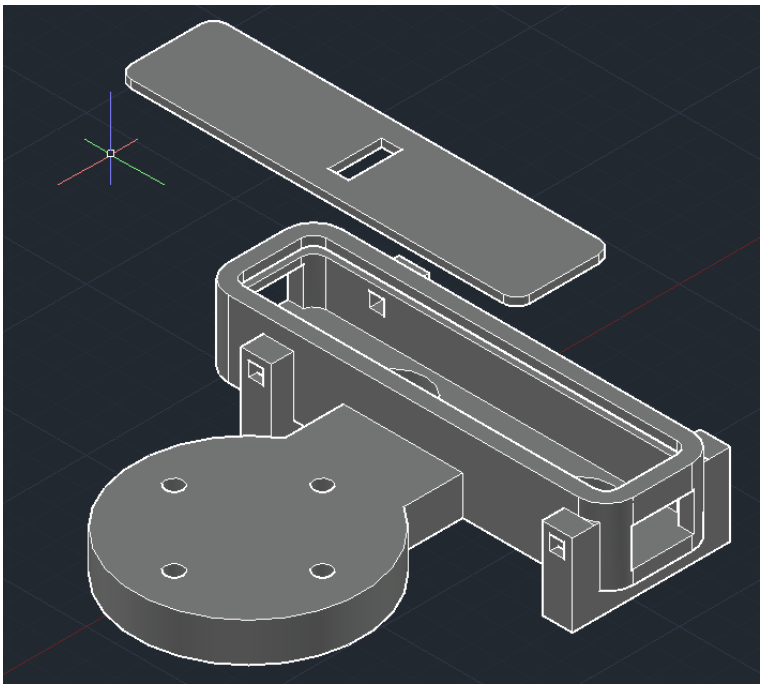

Ilustración 5-9: *Modelo 3D de la cuarta versión del soporte de la cámara para la herramienta con acople rápido.*

Ilustración 5-10: *Cuarta versión del soporte para la herramienta de acople rápido fijada al robot.*

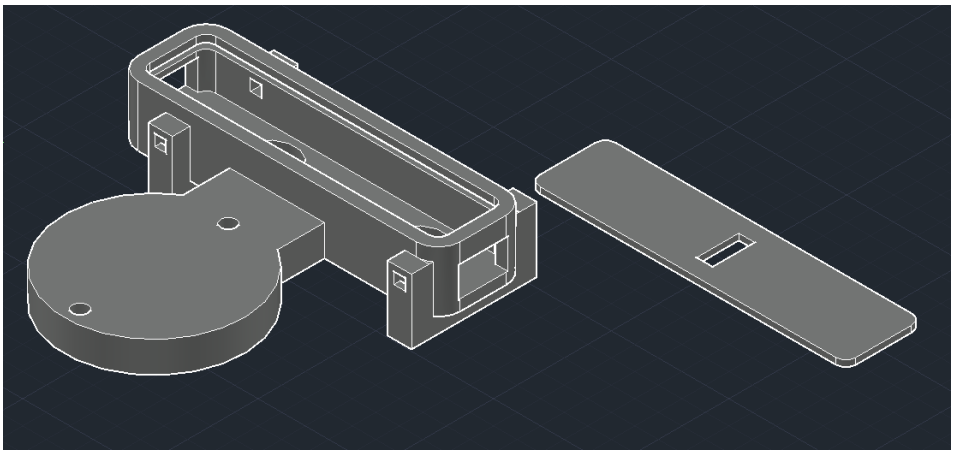

Ilustración 5-11: *Modelo 3D de la cuarta versión del soporte de la cámara para la herramienta UFactory.*

Ilustración 5-12: *Cuarta versión del soporte para la herramienta UFactory fijada al robot.*

5.2. Calibración extrínseca

Los principales sistemas de coordenadas de este trabajo se encuentran situados en la base, en el efector final del manipulador y en la cámara estéreo ZED Mini (Ilustración 5.13). El sistema de coordenadas de la base es el más importante ya que la posición del efector final y del sistema de visión se calculan en función del sistema de coordenadas de la base del robot. La relación de la posición y de la orientación entre los sistemas de coordenadas de la cámara y del efector final es siempre la misma porque se ha utilizado la cámara está fijada en el extremo del robot (eye-in-hand).

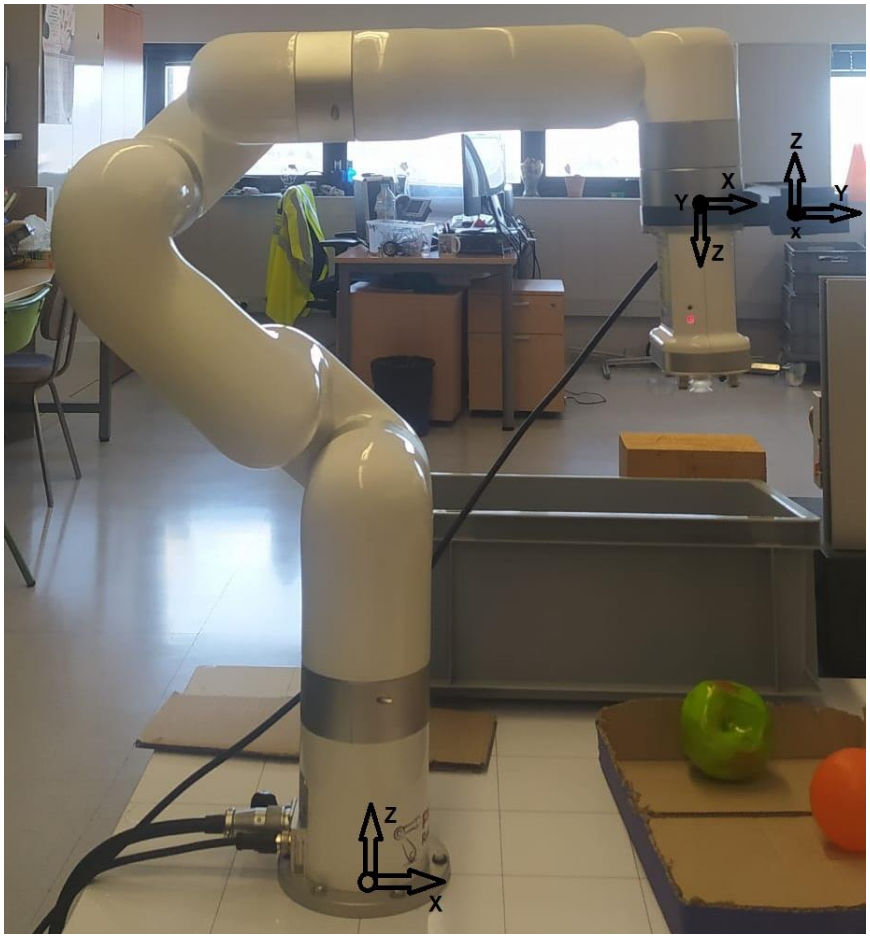

Ilustración 5-13: *Sistemas de coordenadas de la base y el efector final del xArm6. Fuente:* ***[23].***

5.3. Tareas de manipulación

El conjunto de tareas que se llevan a cabo desde que un objeto se detecta hasta que se suelta en la ubicación del contenedor son:

1. Inicialización y conexión de la Jetson AGX Xavier con la ZED Mini y el xArm6.

2. Inicialización y configuración del software según el producto a seleccionar

3. Activación de la detección de objetos de la cámara estéreo.

4. Al detectar un objeto, el software calcula su posición respecto a la base del robot.

5. El software manda la posición calculada al manipulador e indica cuando debe activar o desactivar la herramienta.

Las herramientas utilizadas en este proyecto son: Una herramienta de vacío diseñada por Francisco Muñoz Robles **[27]** y por Pablo Javier Hidalgo Criado **[26]** y una pinza de vacío o *Vacuum Gripper* fabricada por UFactory.

La primera herramienta mencionada (Ilustración 5.14) no necesita ninguna línea de aire comprimido ya que posee una bomba de vacío en el interior que solo requiere de una fuente de electricidad para aspirar el aire y crear un vacío en la cámara donde se ubican las ventosas.

Ilustración 5-14: *Herramienta de vacío con la "Tool side" del sistema de acople rápido.*

Para montar esta herramienta en el manipulador xArm6, se ha optado por usar un sistema de acople rápido diseñado por Pablo Javier Hidalgo Criado **[26]**. Ese sistema consta de dos partes principales:

- ***Tool Side.*** Es una carcasa impresa en 3D que contiene una PCB hembra conectada a la placa de control de la pinza de vacío y fijada en la campana de esta.

- ***Robot Side.*** Es una pieza de impresión 3D que se fija al efector final con tornillos y alberga una pieza central con una PCB macho con varios conectores cuya función es conectar ambas partes del sistema de acople, para así alimentar la herramienta. La unión entre la *Tool Side* y la *Robot Side* se realiza mediante unos resbalones, ubicados en la *Robot Side,* que se encajan en la *Tool side* con la ayuda de unos muelles (Ilustración 5.15).

Ilustración 5-15: *"Robot side" del sistema de acople.*

Para terminar de montar el sistema de acople rápido, solo se necesita ensamblar y conectar todos los componentes de la *Robot Side.* Sin embargo, debido a determinadas variaciones en la conexión eléctrica de la "*Tool Side*, ha sido necesario remodelar algunas partes de la *Robot Side* para poder alimentar correctamente la herramienta.

Como la "Tool side" dispone de tres pletinas metálicas en sustitución a la PCB hembra, se ha decidido realizar un conexionado eléctrico similar utilizando también pletinas, pero con un poco de relieve para hacer contacto con la otra parte. Estas tres pletinas se han fijado a la pieza central modificada (Ilustración 5.16) mediante tornillos y tuercas para evitar posibles contactos entre ellos y los cables que se conectan al robot xArm6.

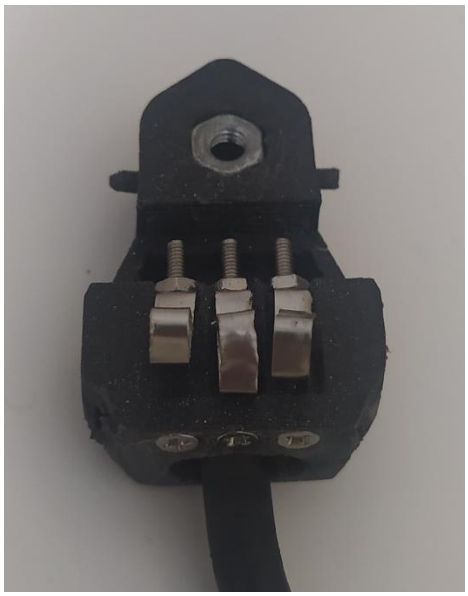

Ilustración 5-16: *Pieza central modificada de la "Robot side".*

La pinza de vacío o *Vacuum Gripper* fabricada por UFactory (Ilustración 5.17) es ideal para manipular piezas de menos de 5 Kg. Esta herramienta está equipada con 5 ventosas que se pueden quitar o poner de acuerdo con el tamaño de la superficie del objeto para que la succión funcione. A la hora de usar esta pinza de vacío hay que tener en cuenta si la superficie del objeto es lisa o no ya que el objeto podría no ser recogido con firmeza debido a las fugas de aire en las ventosas.

Esta herramienta de UFactory está alimentada y controlada directamente por una única conexión que lleva un suministro de 24V CD y un control E/S. La pinza de vacío posee indicadores de estado para saber si está alimentada (Luz roja) o encendida (Luz verde).

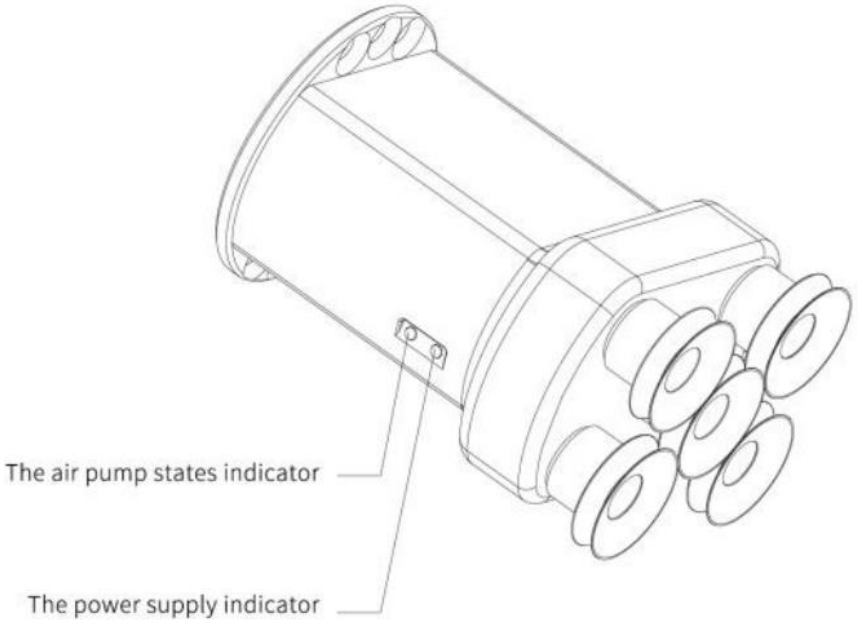

Ilustración 5-17: *Pinza de vacío o Vacuum Gripper de UFactory* ***[24]****.*

6. Experimentos

En este trabajo se han realizado dos tipos de experimentos: La manipulación y la clasificación de los objetos presentes en el entorno. En estos experimentos se han utilizado tres objetos diferentes: Una manzana, un plátano y una naranja (Ilustración 6.1). Para localizar los objetos se utiliza la información recibida del sistema de visión (Todos los cálculos se hacen respecto al sistema de coordenadas de la base ya que las posiciones que se envían al manipulador, mediante servicios de ROS2, deben ser respecto a dicho sistema):

- La coordenada X donde se ubica el objeto se calcula con la suma de la posición en el eje X de la cámara relativa a la base y la posición del objeto en el eje Y de la cámara. La posición en el eje X de los objetos respecto a la base del robot, se calculan de esta manera ya que el eje X de la base coincide con el eje Y de la cámara.

- La posición en el eje Y del objeto se computa como la resta de la posición en el eje Y de la cámara respecto a la base y la posición en el eje X del objeto relativo al sistema de la cámara. En este caso se restan las posiciones ya que los ejes tienen la misma dirección, pero sentido contrario.

- La posición en el eje Z del objeto se calcula con la suma de la altura estimada del objeto a partir del *Bounding Box* 3D y la longitud de la herramienta a usar en el experimento. En este caso, hay que tener en cuenta la herramienta ya que las coordenadas que se envían al robot no son la localización de la herramienta, sino que son la posición donde se va a localizar el efector final.

Este algoritmo, para computar la posición de cada objeto perteneciente a una misma clase, funciona bastante bien en objetos que no tengan formas cóncavas o convexas ya que el centroide de la *Bounding box 3D* puede localizarse en un punto externo a la pieza. Por esta misma razón, el plátano fue descartado de los experimentos de manipulación y clasificación de objetos. Sin embargo, aunque este algoritmo calcule correctamente el punto de agarre, no siempre se podrá manipular el objeto adecuadamente, es decir, al no utilizar objetos con superficies planas en los experimentos con este tipo de herramienta, no se puede asegurar que no haya fugas de aires en la superficie de contacto.

Por ahora, solo se ha calculado el punto de agarre de los objetos y se necesitan varias posiciones más para mover el robot desde el punto de agarre hacia el contenedor:

1. Tras agarrar el objeto, se eleva el robot hasta la altura de reposo (0.45m), pero manteniendo los valores de las coordenadas en los ejes X e Y.

2. En este momento, se le indica al robot que se desplace hacia la posición (X e Y) del contenedor, pero con el mismo valor de la coordenada Z (0.45m).

3. Una vez ubicado el manipulador encima del contenedor, se desplaza hacia la altura del punto de agarre del objeto, pero con un pequeño desfase hacia arriba para evitar posibles colisiones con el contenedor u otros objetos.

4. Ya terminado la manipulación del objeto, se mueve el robot hacia su posición de reposo para seguir detectando y clasificando objetos.

La clasificación de objetos lo realiza en gran medida el sistema de visión a partir de sus redes neuronales entrenadas, pero el software también participa en esta tarea. El software, a partir de los datos de cada objeto que se reciben de la cámara mediante ROS2, selecciona los objetos presentes en el entorno según si pertenecen o no a la clase elegida al principio del programa y una vez clasificados, el software envía las posiciones de cada objeto de dicha clase al manipulador.

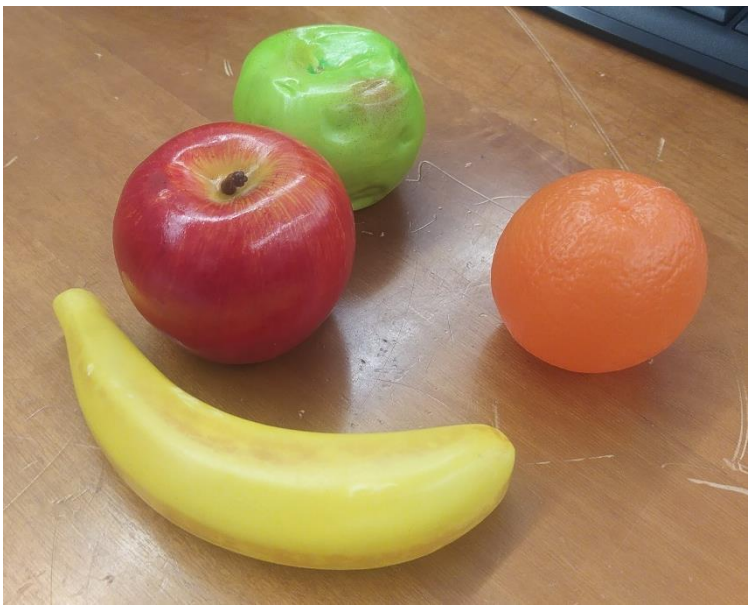

Ilustración 6-1: *Imágenes de los objetos usados en los experimentos.*

6.1. Pick and place

El *pick and place* o *recoger y colocar* es un proceso que consiste en controlar un manipulador para agarrar un producto y ubicarlo en otro lugar. En este trabajo, dicha operación es una parte fundamental en la manipulación de los objetos que se encuentran dentro área de trabajo del manipulador. Este experimento se puede dividir en dos partes: Hardware y software. El hardware lo conforman la Jetson AGX Xavier, procesando el sistema de visión y ejecutando el software, el robot xArm6, manipulando con precisión los objetos, y la cámara estéreo ZED Mini, proporcionando una realimentación visual del entorno. El software controla el manipulador a partir de los datos procedentes del hardware y los servicios que ofrecen los paquetes de ROS2 de Stereolabs y UFactory.

Además, este proceso es muy utilizado en la industria actual ya que es el principal foco de automatización para medianas y grandes empresas por su capacidad de adaptación en el entorno de trabajo.

6.2. Selección de objeto

Este experimento consiste en clasificar y seleccionar los objetos presentes en el campo de visión de la cámara, es decir, esta operación es un *pick and place,* pero con un software capaz de controlar el robot y de clasificar las piezas según el objeto a almacenar en el contenedor. Durante el experimento se han reproducido dos condiciones de trabajo distintas para comprobar los límites de este proyecto:

1. Se ha recreado la tarea de *Bin picking* en este experimento para comprobar si el sistema de visión muestra dificultades en la detección, localización y clasificación de las piezas que se encuentran dentro del contenedor (Ilustración 6.2 y 6.3):

    1.1. Durante el desarrollo de este proceso, se observa que la cámara estéreo ZED Mini no presenta problemas aparentes para localizar el punto de agarre de los objetos a seleccionar (Ilustración 6.4).

    1.2. Tras agarrar correctamente la pieza, se procede a almacenarlo en el contenedor desactivando la herramienta una vez ubicados (Ilustración 6.5 y 6.6).

2. En este caso los objetos no se ven al completo, sino que se encuentran parcialmente cubiertos por diferentes piezas presentes en el entorno (Ilustración 6.7 y 6.8). En este

experimento se va a evaluar la influencia de la presencia de oclusiones en el campo de visión de la cámara estéreo. Tras realizar varias pruebas, se ha llegado a la conclusión de que una obstrucción parcial de la visión de la cámara no afecta en gran medida al funcionamiento del proyecto (Ilustración 6.9), excepto en el caso de que dicha oclusión interfiera en el recorrido del manipulador.

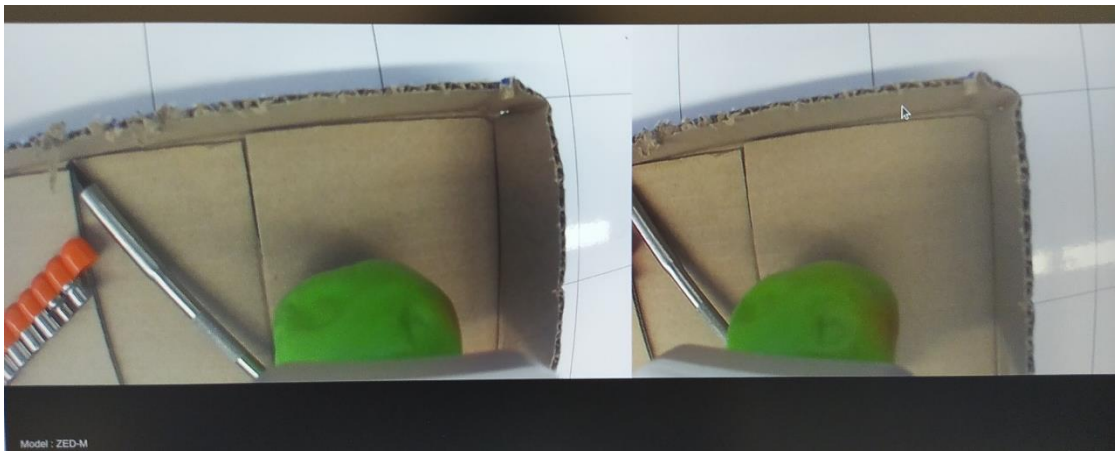

Ilustración 6-2: *perspectiva del sistema de visión en la tarea de* Bin picking*.*

Ilustración 6-3: *Situación inicial de la tarea de Bin picking.*

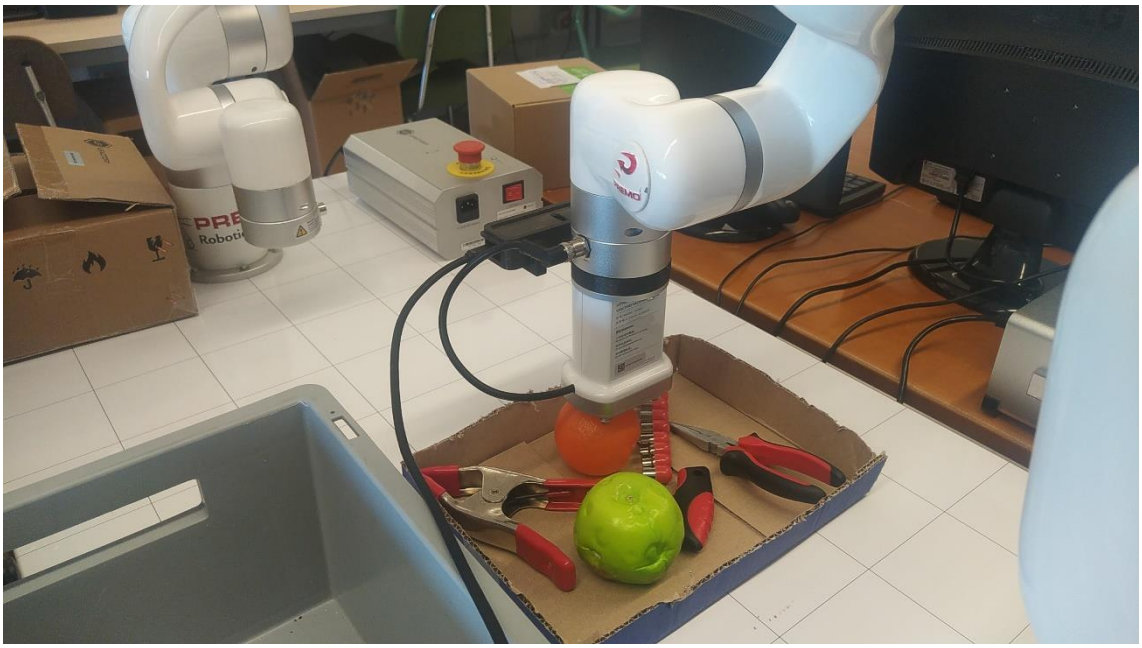

Ilustración 6-4: *Manipulación de objetos en el contenedor.*

Ilustración 6-5: *Elevación hasta la ubicación del contenedor.*

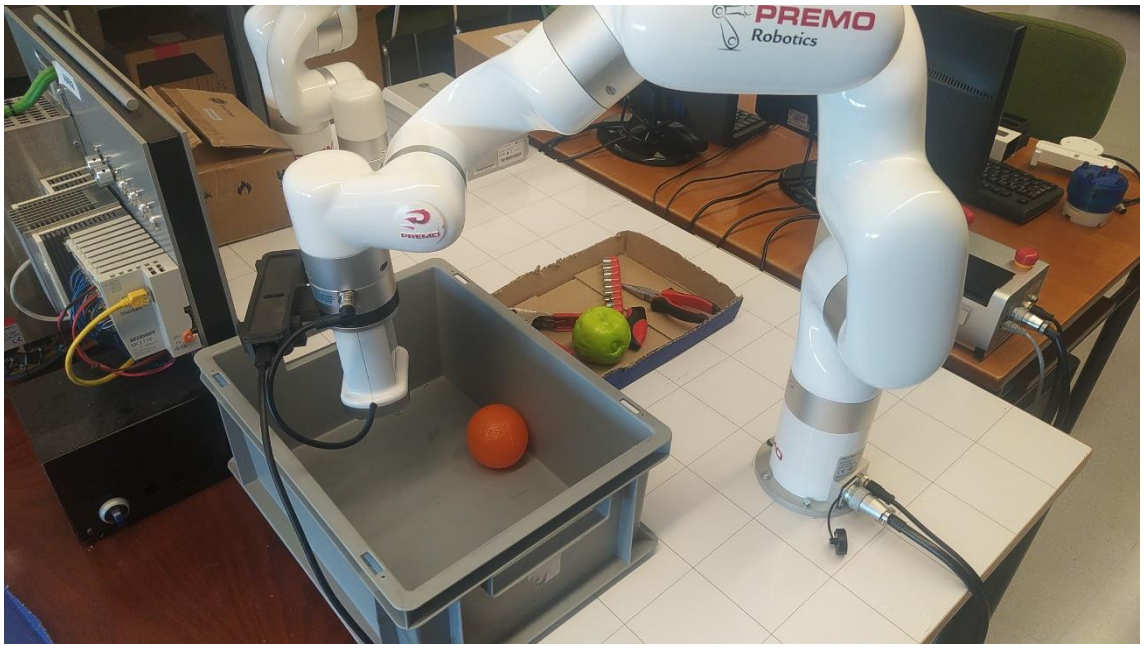

Ilustración 6-6: *Almacenamiento de la pieza en el contenedor.*

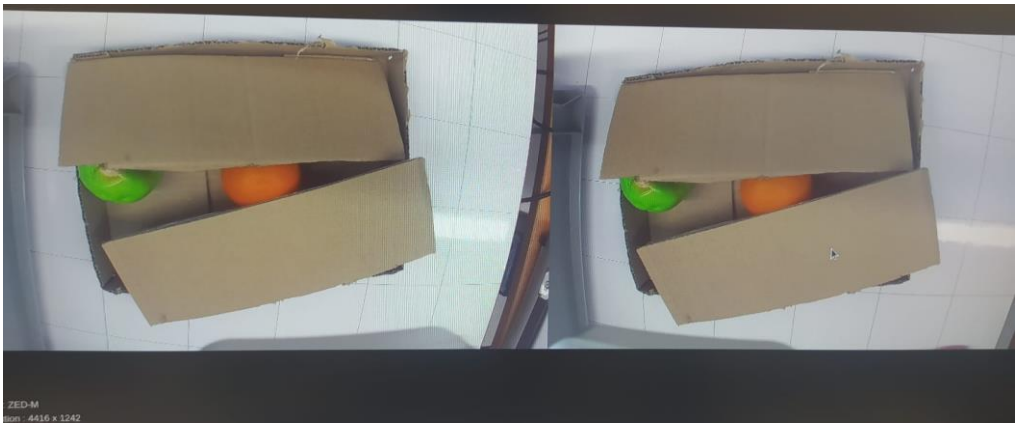

Ilustración 6-7: *Perspectiva del sistema de visión con oclusiones*

.

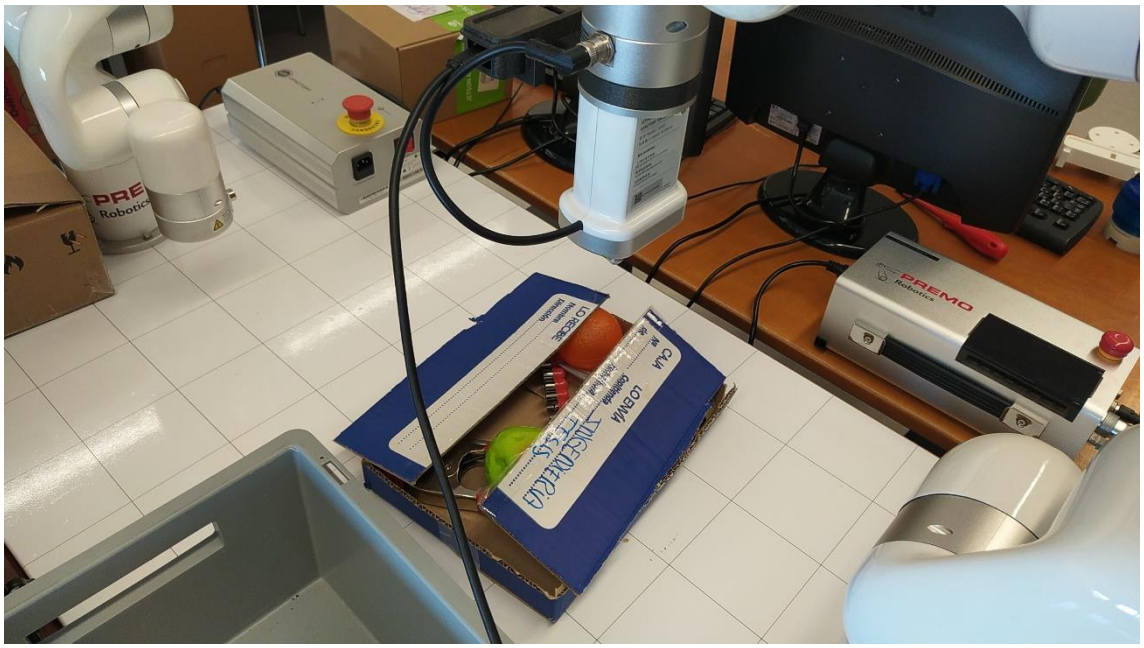

Ilustración 6-8: *Situación inicial del experimento con oclusiones.*

Ilustración 6-9: *Manipulación del objeto con obstrucciones.*

7. Conclusiones y trabajo futuro

7.1. Conclusiones

En este proyecto se ha logrado procesar adecuadamente el sistema de visión mediante el SDK de Stereolabs instalado en la Jetson AGX Xavier. Se ha desarrollado un software en ROS2 que controla el manipulador xArm6 y filtra los objetos detectados a partir de la información recibida del robot y del sistema de visión. Se ha modelado un soporte impreso en 3D que fija la cámara en el efector final del robot. También, se han remodelado los componentes de un sistema de acople rápido, para alimentar y activar la pinza de vacío o *Vacuum Gripper.*

En resumen, se ha alcanzado el objetivo de este proyecto de realizar un sistema de control visual que permita manipular y clasificar los objetos presentes en el entorno. Todo el contenido del software, los modelos de impresión 3D, los vídeos y la documentación de este proyecto se encuentran disponibles en GitHub: https://github.com/Deglox/MANIPULATOR-ROBOT-CONTROL-THROUGH-VISION

.

7.2. Trabajo futuro

En este proyecto se han visto algunas limitaciones que con mayor tiempo y recursos se pueden ir mejorando. Algunas posibles mejoras para este trabajo son:

- Cambiar la herramienta de vacío o *Vacuum Gripper* por una herramienta de pinza *o Gripper* para evitar posibles fugas de aire en el contacto con los objetos.

- Utilizar un modelo de detección con segmentación semántica en vez de una *Bounding Box 3D,* para calcular correctamente el centroide de objetos con formas cóncavas, convexas o similares.

- Utilizar una cámara monocular en vez de la cámara estéreo ZED Mini, para poder localizar objetos cercanos al objetivo de la cámara. Sin embargo, este tipo de sistemas de visión no pueden localizar los objetos con precisión y desde una misma ubicación al igual que las cámaras estéreo, sino que necesitan usar técnicas de *Structure from Motion* para reconstruir la escena en 3D con el movimiento de la cámara, dificultando la realización del software del proyecto y la precisión en el cálculo de la localización.

Este trabajo se podría utilizar perfectamente como base para nuevos proyectos futuros ya que proporciona a los manipuladores mayor flexibilidad en los procesos. Un caso práctico de uso futuro para este proyecto en la industria sería en las tareas de “Bin picking” donde se necesitaría entrenar las redes neuronales de la cámara con los productos a manipular y utilizar otra herramienta más adecuada para el trabajo designado. Otro trabajo futuro podría ser en el cuidado de personas discapacitadas o dependientes que necesiten una atención durante todo el día donde el manipulador junto a un robot móvil podría controlar el suministro de medicamentos diarios del paciente.

8. Referencias y bibliografías

[1] Peter Corke: 2017, Robotics, Vision and Control: Fundamental Algorithms.

[2] Torres, F.; Pomares, J.; Gil, P.; Puente, S. T., y Aracil, R.: 2002, Robots y Sistemas Sensoriales.

[3] GAMCO S.L: 2020, ¿Qué es Detección de objetos? Concepto y definición. Última consulta el 17 de Junio de 2024.

URL:https://gamco.es/glosario/deteccion-de-objetos/#:~:text=En%20otras%20palabras%2C%20la%20detecci%C3%B3n,de%20seguridad%20y%20la%20rob%C3%B3tica.

[4] Juan Ernesto Solanes Galbis: 2010, Visual Servoing Multifrecuencia. Última consulta el 17 de Junio de 2024.

URL:https://riunet.upv.es/bitstream/handle/10251/14220/Memoria%20Tesina.pdf?sequence=1.

[5] IBM: 2024,¿Qué es el Deep Learning?. Última consulta el 17 de Junio de 2024.

URL:https://www.ibm.com/es-es/topics/deep-learning#:~:text=Las%20redes%20neuronales%20de%20deep,objetos%20dentro%20de%20los%20datos.

[6] ResearchGate: 2005, 4 Esquema de control visual “mirar y mover estático”. Última consulta el 17 de Junio de 2024.

URL: https://www.researchgate.net/figure/Ilustración-14-Esquema-de-control-visual-mirar-y-mover-estatico_fig3_39425275

[7] MDPI: 2022, Fuzzy Gain-Scheduling Based Fault Tolerant Visual Servo Control of manipulator. Última consulta el 17 de Junio de 2024.

URL: https://www.mdpi.com/2504-446X/7/2/100

[8] Stereolabs Inc: 2024, Camera Calibration - Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/video/camera-calibration

[9] Stereolabs Inc: 2024, Get Started with ZED - Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/get-started-with-zed.

[10] Stereolabs Inc: 2024, Sensors Overview- Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/sensors#:~:text=The%20ZED%20family%20of%20depth,%2C%20barometer%2C%20magnetometer%20and%20more.

[11] Stereolabs Inc.: 2024, ZED Mini Stereo Camera | Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://store.stereolabs.com/en-es/products/zed-mini.

[12] Stereolabs Inc: 2024, ZED SDK 4.1 - Downloads | Stereolabs. Última consulta el 17 de Junio de 2024.

URL: https://www.stereolabs.com/developers/release.

[13] Stereolabs Inc: 2024, 3D Object Detection Overview - Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/object-detection.

[14] Stereolabs Inc: 2024, Adding Object Detection In ROS 2. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/ros2/object-detection

[15] Stereolabs Inc: 2024, Group Object Class - Stereolabs . Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/api/group__Object__group.html#ga13b0c230bc8fee5bbaaaa57a45fa1177.

[16] Stereolabs Inc: 2024, Depth Sensing Overview - Stereolabs . Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/depth-sensingAprende más.

[17] Stereolabs Inc: 2024, Positional Tracking Overview - Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/positional-tracking.

[18] Stereolabs Inc: 2024, Positional Tracking Use Cases - Stereolabs . Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/positional-tracking/use-cases.

[19] Stereolabs Inc: 2024, Coordinates Frames - Stereolabs. Última consulta el 17 de Junio de 2024.

URL:https://www.stereolabs.com/docs/positional-tracking/coordinate-frames.

[20] UFactory: 2023, UFactory xArm6 [en línea]. Última consulta el 17 de Junio de 2024.

URL:https://www.ufactory.cc/cost-effective-cobot-robots/?utm_source=google+search&utm_medium=cpc&utm_campaign=xArmUfactory&utm_id=branding&gad_source=1&gclid=Cj0KCQjwsPCyBhD4ARIsAPaaRf24Av9BZiwZVIYR4p6KFdo5E9iyRvS9RCCVjX_YWx84JDAdycBR3qUaAmE8EALw_wcB.

[21] UFactory: 2023, xarm_ros2 | UFactory. Última consulta el 17 de Junio de 2024.

URL:https://docs.ufactory.cc/xarm_ros2/readme_en.

[22] UFactory: 2023, xArm User Manual - V2.0.0 - UFACTORY. Última consulta el 17 de Junio de 2024].

URL:https://www.ufactory.cc/wp-content/uploads/2023/05/xArm-User-Manual-V2.0.0.pdf

[23] Danica Kragic y Henrik Christensen: Survey on Visual Servoing for Manipulation

URL:https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=16a71f602a103a7ba29658999ddde75a0f2a3848

[24] Seth Hutchinson, Gregory D. Hager and Peter I, Coke: 1996 A tutorial on Visual servo control

URL:https://faculty.cc.gatech.edu/~seth/ResPages/pdfs/HutHagCor96.pdf

[25] Stereolabs, Inc: ZED Mini Datasheet.

URL:
https://cdn.sanity.io/files/s18ewfw4/staging/ebcd46896092d1ee6212b7f4d81aaa1c479c2440.pdf/ZED%20Mini%20Datasheet%20v1.1.pdf

[26] Pablo Javier Hidalgo Criado: 2022, Diseño de acople universal de herramienta para robot industrial.

[27] Francisco Muñoz Robles: 2023, Diseño de herramienta de vacío para robot manipulador.