Control Visual del Brazo de un Manipulador Aéreo

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO DE FIN DE GRADO

Autor:

Miguel de Médicis Barrionuevo

Tutor:

Jesús Manuel Gómez de Gabriel

Co-tutor:

Juan Manuel Gandarias Palacios

UNIVERSIDAD DE MÁLAGA

Departamento de Ingeniería de Sistemas y Automática
ESCUELA DE INGENIERÍAS INDUSTRIALES

ENERO 2019

Resumen

En este trabajo se realiza el control visual, mediante cámara RGB-D, de un manipulador delta. El manipulador delta que utiliza en este proyecto ha sido previamente diseñado y construido por el Departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga. Este manipulador puede montarse en un dron para ser usado en operaciones de búsqueda y rescate (SAR). El problema que se presenta es el seguimiento de un objeto de interés por parte del manipulador. Se proponen soluciones que se han implementado en distintos nodos de ROS, que son ejecutados por un sistema embebido Up Board. Se han creado dos nodos distintos calcular la posición real de un objeto a partir de la imagen captada en tiempo real por la cámara RGB-D, a partir del algoritmo de Suzuki-Abe disponible en la librería OpenCV. Uno de estos nodos resuelve la proyección inversa a partir de una nube de puntos, mientras que el otro utiliza la información presente en una imagen de profundidad. También se han creado varios nodos para la planificación de las trayectorias del manipulador. Finalmente se han realizado experimentos que demuestran el comportamiento del sistema. Los resultados sugieren que el brazo se mueve a la posición deseada si la cámara consigue calcular la distancia con éxito.

Palabras clave: Cámara RGB-D, Imagen de profundidad, Nube de puntos, OpenCV, Python, Robot delta, ROS (Robotic Operating System), Sistema de referencia, Visión por computador

Abstract

Visual servoing of a delta robot with an RGB-D camera is carried out in this project. The delta manipulator used in this project has been previously designed and built by the System Engineering and Automation Department at the University of Malaga. This robot can be mounted on a UAV in order to be used in Search And Rescue operations (SAR). The problem that is presented is the tracking of an object of interest by the robot. Proposed solutions to this problem have been implemented in various ROS nodes, which are executed by an Up Board. Two different nodes have been created to calculate the real position of an object based on the real-time image provided by the RGB-D camera, based on the Suzuki-Abe algorithm available in the OpenCV library. One of these nodes solves the inverse projection problem based on a pointcloud, while the other uses data from a depth image. Several nodes have also been created for the planning of the manipulator trajectories. Finally, experiments have been carried out to show the functioning of the system. The results suggest that the arm moves to the desired position if the camera calculates the depth successfully.

Keywords: RGB-D camera, Depth image, Pointcloud, OpenCV, Python, Delta robot, ROS (Robotic Operating System), Reference system, Computer vision

Declaración de Autoría

Yo, Miguel de Médicis Barrionuevo, estudiante del Grado en Ingeniería en Tecnologías Industriales en la Escuela de Ingenierías Industriales de la Universidad de Málaga, declaro que este Trabajo de Fin de Grado, titulado "Control Visual del Brazo de un Manipulador Aéreo", es de mi autoría, y que las fuentes utilizadas para la realización del mismo se encuentran debidamente referenciadas en la bibliografía.

FIRMADO: ..... FECHA: .....

Índice General

Índice de Figuras

ÍNDICE DE TABLAS

TABLA **Página**

1

INTRODUCCIÓN

En este proyecto se pretende realizar el seguimiento visual de un objeto con un manipulador delta. Se han desarrollado algoritmos y esquemas de control que permiten localizar el objetivo mediante una cámara RGB-D y mover el manipulador a la posición en la que se encuentre dicho objeto.

1.1 Visión general

1.1.1 Motivación y justificación

La visión por computador permite a los computadores deducir información de la realidad a partir de imágenes. Aunque esta es una tarea que las personas somos capaces de realizar de forma fluida y consistente, la visión por computador todavía no es capaz de dar soluciones eficaces a algunos problemas que pueden parecer triviales. Uno de los factores que constituyen la dificultad del problema de la visión por computador es que se trata de un problema inverso, lo cual implica que hay información de la realidad que no se puede obtener a partir de una imagen [1].

La visión por computador facilita la automatización en aplicaciones como el control de manipuladores industriales [2]; en cirugía mínimamente invasiva [3] o, como en el caso de este proyecto, operaciones de búsqueda y rescate (en inglés Search And Rescue o SAR), entre otras.

En casos de catástrofe, la localización de las víctimas y la monitorización de sus constantes vitales es esencial para una respuesta efectiva. Esta tarea puede llevarse a cabo con ayuda de robots que, gracias a la visión por computador, pueden encontrar a dichas víctimas con cierta autonomía [4].

En comparación con los robots terrestres, los drones o UAV (Unmanned Aerial Vehicle)

presentan como ventajas una mayor movilidad y la posibilidad de llegar a sitios inalcanzables por robots terrestres, de ahí que el uso de drones se empiece a popularizar en operaciones de búsqueda y rescate. Sin embargo, la capacidad de carga de los drones hace que históricamente se hayan utilizado en tareas que no les exijan entrar en contacto con elementos de su entorno.

Al dron que se utiliza en este proyecto se le ha dotado de un manipulador ligero tipo delta para colocar a las víctimas una pulsera que monitorice sus constantes vitales. La estabilidad del dron puede verse comprometida si el movimiento del brazo introduce un momento significativo en el sistema, así que las trayectorias se han implementado con la idea de minimizar este efecto.

1.1.2 Antecedentes

Este apartado recopila trabajos que exploran temáticas relacionadas con las de este proyecto, como son el seguimiento visual de objetos o el control de manipuladores aéreos.

Actualmente se está desarrollando un nuevo concepto de dron dotado de una carcasa protectora formada por dos hemisferios que pueden rotar. En [5] se propone un brazo retráctil dotado de pinza para este tipo de drones.

En [6] se propone un sistema para organizar paquetes con manipuladores aéreos. Los paquetes se identifican con ayuda de un símbolo, y se transportan con un brazo con dos grados de libertad y un enganche magnético.

La odometría visual consiste en la estimación de la posición de un robot a partir de imágenes tomadas por una o varias cámaras montadas en el mismo. En [7] se presenta un sistema de odometría mediante el seguimiento y mapeado de puntos clave.

El agarre robótico (robotic grasping) es un problema que consiste en que un robot sea capaz de identificar el punto en el que puede agarrar un objeto de la forma más eficaz. Para esta tarea se puede utilizar una red neuronal diseñada para el reconocimiento de objetos y adaptada para calcular agarres. En [8] se le da un enfoque distinto este problema, con una red neuronal específica para esta tarea (Generative Grasping Convolutional Neural Network o GG-CNN). Esta red neuronal es capaz de calcular el agarre óptimo de un objeto de forma rápida y sin necesidad de estar familiarizada con el objeto en concreto.

En [9] se realiza el control visual mediante cámara RGB-D, de un brazo robótico en configuración SCARA para agarrar objetos. El efector final se localiza en la imagen gracias a un símbolo colocado sobre el mismo. Con la cámara se detecta el plano sobre el que está apoyado el brazo, así como el desplazamiento que se percibe en el efector final al moverlo una determinada distancia. Esta información es suficiente para relacionar los sistemas de referencia de la cámara y del brazo. Para identificar los objetos que se quieren agarrar se utiliza la imagen de profundidad.

En [4] se propone un sistema de detección de personas que puede ser utilizado por un robot autónomo en operaciones de búsqueda y rescate. Se utiliza un sensor infrarrojo pasivo que indica cuándo debe tomarse una instantánea. Cuando se toma una imagen, se buscan personas en la misma con una red neuronal.

En [10] se realiza el seguimiento y agarre de un objeto en movimiento. Se utilizan la imágenes de dos cámaras fijas para calcular el flujo óptico estereoscópico y calcular la posición del objeto en tiempo real. Esta posición se envía al sistema del control del brazo para realizar el seguimiento. Cuando el seguimiento se estabiliza, se procede a interceptar y agarrar el objeto.

En [11] se introduce la imagen RGB en una red neuronal R-CNN, tras lo cual se obtienen las coordenadas de la muñeca en píxeles. Las coodenadas en píxeles se convierten a coordenadas reales gracias a una nube de puntos, al igual que en uno de los algoritmos que se proponen en el presente proyecto.

En [12] se utiliza una cámara fija para el seguimiento y la estimación de la posición del efector final de un brazo robótico. Al igual que en el presente trabajo, en la localización se realiza con un algoritmo basado en la segmentación por color de la imagen capturada, pero se utiliza el espacio de color HSV (Hue, Saturation, Value - Matiz, Saturación, Valor) en vez de RGB. Se extraen puntos de interés en el efector final para calcular la homografía entre el sistema de referencia de la pinza y el del mundo.

1.1.3 Objetivos

El problema a resolver consiste en calcular la posición real de un objeto a partir de la cámara para mover el brazo a dicha posición, como se ilustra en la Figura 1.1.

El problema se ha dividido en los siguientes objetivos:

- Implementar en nodos de ROS algoritmos para calcular la posición real de un objeto a partir de la información captada por la cámara RGB-D.

- Implementar las trayectorias para el manipulador delta para dirigirlo a la posición deseada teniendo en cuenta el movimiento del dron y la pinza pasiva del efector final.

- Posicionar la cámara RGB-D en el sistema para que tenga un campo de visión adecuado y para que pueda ser utilizada mientras el dron esté en el aire.

- Relacionar los sistemas de referencia de la cámara y del manipulador para lograr que el efector final se mueva a la posición que determine el nodo de localización.

- Evaluar experimentalmente el funcionamiento de los nodos en los que se implementan los algoritmos de localización y las trayectorias, y comparar los resultados de para identificar cuales de ellos son los más adecuados para el funcionamiento del sistema completo.

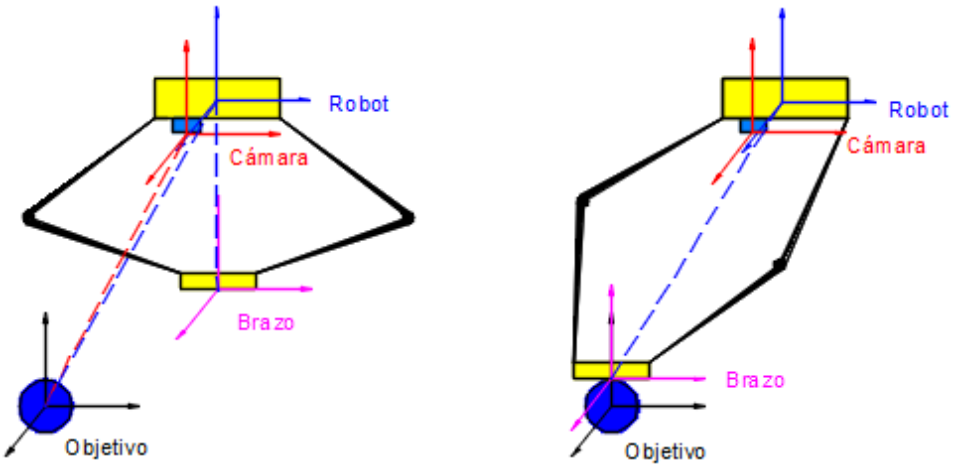

Figura 1.1: Esquema del problema a resolver. Primero se calcula la posición del objetivo en los sistemas de referencia de la cámara y del robot (izquierda). Después se mueve el brazo hacia la posición objetivo (derecha)

1.2 Estructura de la memoria

En el Capítulo 2 de esta memoria se explican algunos conceptos teóricos relacionados con este trabajo. En el Capítulo 3 se describe el material y el software de partida. En los siguientes capítulos se describe la realización del proyecto, correspondiendo el Capítulo 4 al diseño e implementación de los nodos encargados de la localización mediante cámara RGB-D; el Capítulo 5 a la colocación de la cámara y a la relación entre los sistemas de referencia de la cámara y del brazo; el Capítulo 6 a los nodos que permiten el movimiento del brazo, y el Capítulo 7 a la descripción de los experimentos que se han realizado. En el Capítulo 8 se discuten las conclusiones y posibles líneas de trabajo futuras.

2

FUNDAMENTOS TEÓRICOS

Debido a la variedad de conocimientos necesarios para el desarrollo de este proyecto, se considera oportuno tratar conceptos básicos relacionados con la temática del mismo. En este capítulo se explican el modelo de cámara pinhole (Sección 2.1), la relación entre marcos de referencia (Sección 2.2) y diferentes estrategias para localizar y seguir objetos de interés en una imagen (Secciones 2.3 y 2.4).

2.1 El modelo pinhole

La Figura 2.1 muestra un esquema del modelo pinhole. Una cámara pinhole es una cámara ideal consistente en un plano sobre el que se proyectan los rayos de luz, el plano de la imagen, y el punto que determina la dirección que siguen dichos puntos al proyectarse, el centro óptico (C). La recta que pasa por el centro óptico y que es perpendicular al plano de la cámara se denomina eje óptico. La distancia entre el centro óptico y el plano de la imagen es f, la distancia focal. Se consideran los siguientes sistemas de referencia:

- El sistema de referencia $C-(X,Y,Z)$ es un sistema de referencia tridimensional con origen en el centro óptico (C) y cuyo eje Z coincide con el eje óptico. En este sistema de referencia se mide la posición de puntos reales en magnitud real. Este sistema de referencia está representado en la Figura 2.1.

- El sistema de referencia $c-(x,y)$ es un sistema de referencia bidimensional que coincide con el plano de la imagen y con origen en el punto por el que pasa el eje óptico (c). En este sistema de referencia se mide la posición de los puntos proyectados sobre el plano de la imagen en magnitud real. Este sistema de referencia está representado en la Figura 2.1.

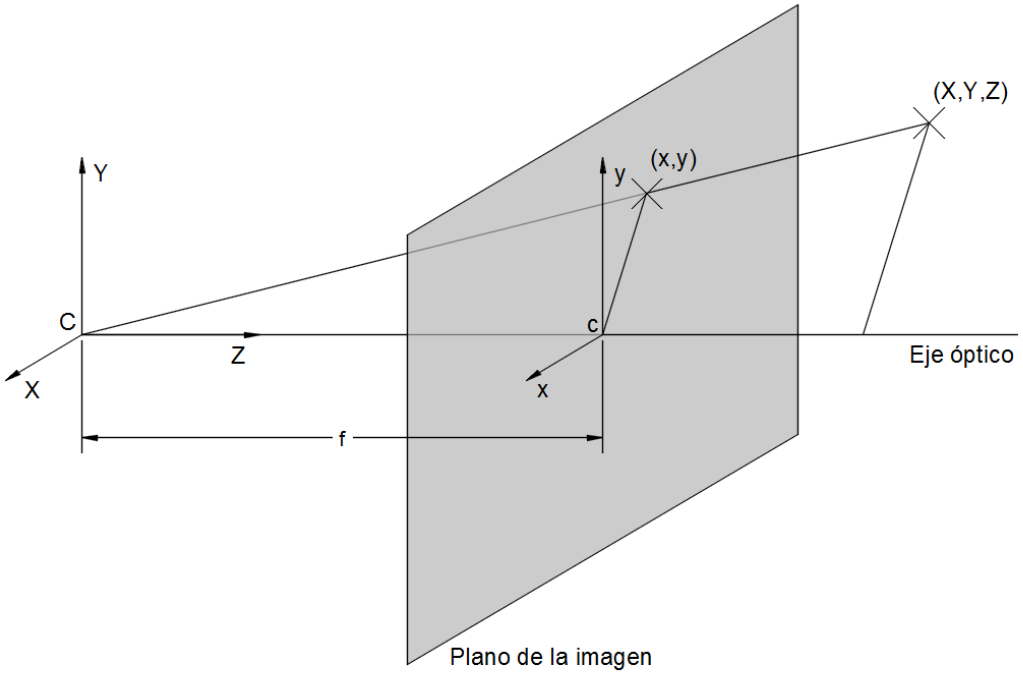

Figura 2.1: Esquema de una cámara pinhole

- El sistema de referencia $o-(x_p, y_p)$ es un sistema de referencia bidimensional que coincide con el plano de la imagen y con origen en una de las esquinas de la imagen. En este sistema de referencia se mide la posición de los puntos proyectados en píxeles. Este sistema de referencia no está representado en la Figura 2.1.

A continuación se justifican las ecuaciones que relacionan los distintos sistemas de referencia que se consideran en la formación de la imagen [1]:

Dado un punto real en el sistema de referencia $C-(X,Y,Z)$ se relaciona con su proyección en el sistema de referencia $c-(x,y)$ conocida la distancia focal (f):

$$\begin{pmatrix} x \\ y \end{pmatrix} = f \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}$$

Se puede obtiene una expresión lineal equivalente a la anterior si se utilizan coordenadas homogéneas ($x = u/w, y = v/w$):

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

Sean s_x y s_y la distancia real entre los centros de dos píxeles contiguos en los ejes x e y, respectivamente. Siendo (c_x, c_y) la posición del punto c en el sistema de referencia $o-(x_p, y_p)$, los sistemas de referencia $c-(x,y)$ y $o-(x_p, y_p)$ se relacionan con un un escalado (dos primeras columnas de la matriz) y un desplazamiento (tercera columna de la matriz):

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} 1/s_x & 0 & c_x \\ 0 & 1/s_y & c_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

La expresión anterior se puede representar en coordenadas homogéneas:

$$\begin{pmatrix} u_p \\ v_p \\ w_p \end{pmatrix} = \begin{pmatrix} 1/s_x & 0 & c_x \\ 0 & 1/s_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Una vez conocidas la relación entre los sistemas de referencia $C-(X,Y,Z)$ y $c-(x,y)$, y $c-(x,y)$ y $o-(x_p, y_p)$, se deduce relación entre los sistemas de referencia $C-(X,Y,Z)$ y $o-(x_p, y_p)$:

$$\begin{pmatrix} u_p \\ v_p \\ w_p \end{pmatrix} = \begin{pmatrix} 1/s_x & 0 & c_x \\ 0 & 1/s_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

A la matriz K se la denomina matriz intrínseca o matriz de calibración. La distancia focal suele representarse en función de las distancias entre píxeles ($f_x = f/s_x$ y $f_y = f/s_y$) para simplificar la matriz.

En los casos en los que hay varias cámaras, se utiliza la matriz de la cámara o P. Esta matriz tiene en cuenta la posición de las cámaras respecto a una cámara principal. Si las cámaras tienen la misma orientación y solo hay desplazamiento en X y en Y:

$$P = K \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & T_x \\ 0 & f_y & c_y & T_y \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Donde (t_x, t_y) es el desplazamiento en (X,Y). El desplazamiento suele expresarse como $T_x = f_x t_x$ y $T_y = f_y t_y$ para simplificar la matriz.

2.2 Relación entre marcos de referencia

Dada la posición de un punto respecto a un marco de referencia fijado a un objeto (x_o, y_o, z_o), la posición de dicho punto con respecto al sistema de referencia base (x,y,z) se puede calcular

conocidos el desplazamiento (X_o, Y_o, Z_o) y la orientación $R(\theta_{Xo}, \theta_{Yo}, \theta_{Zo})$ del marco de referencia del objeto con respecto al marco de referencia base [2]:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = T_o \begin{pmatrix} x_o \\ y_o \\ z_o \\ 1 \end{pmatrix}$$

T_o es la matriz de transformación homogénea del marco de referencia del objeto con respecto al sistema de referencia base:

$$T_o = \begin{pmatrix} r_{11} & r_{12} & r_{13} & X_o \\ r_{21} & r_{22} & r_{23} & Y_o \\ r_{31} & r_{32} & r_{33} & Z_o \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Donde r_{ij} son los componentes de la matriz de rotación $R(\theta_{Xo}, \theta_{Yo}, \theta_{Zo})$.

La matriz de transformación homogénea del marco de referencia del efector final respecto al de la imagen (${}^iT_e$) se puede calcular conocidas las matrices de transformación homogéneas de la base con respecto al efector (${}^eT_b$), la de la cámara con respecto a la base (${}^bT_c$) y la de la cámara con respecto a la imagen (${}^cT_i$):

$${}^iT_e = {}^eT_b\,{}^bT_c\,{}^cT_i$$

2.3 Identificación de puntos óptimos para seguimiento

Para seguir un objeto en imágenes es necesario encontrar características del mismo que a un computador le resulten fáciles de reconocer. A continuación se explican brevemente algunos algoritmos para realizar esta tarea.

2.3.1 Detector de esquinas de Moravec

El detector de esquinas de Moravec sirve para encontrar esquinas en una imagen. Este algoritmo hace pasar una ventana a lo largo de la imagen de interés, y evalúa la el cambio de intensidad en el interior de dicha ventana. Los puntos en los que haya máximos locales en la variación de intensidad, por encima de un umbral, son los que se consideran esquinas. Este algoritmo es menos fiable que los que se comentan más adelante porque no es capaz de diferenciar entre esquinas y bordes, y solo comprueba cambios de intensidad en direcciones separadas 45º entre sí [13].

2.3.2 Detector de esquinas de Harris

El detector de esquinas de Harris, al igual que el de Moravec, se basa en comprobar la variación local de intensidad. Sin embargo, resuelve alguno de los problemas que presenta el algoritmo de Moravec. El detector de esquinas de Harris puede comprobar la variación de intensidad sin importar su dirección, y puede distinguir esquinas y bordes. Además de esto, asigna a esquinas y bordes un valor que indica su calidad [13].

2.3.3 Detector de esquinas de Shi-Tomasi

El detector de esquinas de Shi-Tomasi es igual que el de Harris, pero utiliza un criterio distinto para valorar la calidad de las esquinas y los bordes, de forma que se obtienen mejores resultados [14].

2.3.4 Algoritmo SIFT y similares

Los detectores de esquinas de los que se ha hablado anteriormente son sensibles a la escala, así que no son adecuados para comparar imágenes de distinto tamaño. El algoritmo SIFT (Scale-Invariant Feature Transform) resuelve este problema, y es más robusto en casos de cambio en la iluminación o la perspectiva [15, 16].

El algoritmo SIFT utiliza varias etapas de filtrado para seleccionar los puntos clave de la imagen que menos varían frente a cambios de escala, desplazamientos y rotaciones. La variación local de intensidad en cada punto se caracteriza teniendo en cuenta cambios en la iluminación o la perspectiva [16].

Existen otros algoritmos, como SURF (Speeded Up Robust Features) [17] y FAST (Features from Accelerated Segment Test) [18] que están basados en el algoritmo SIFT pero son más rápidos.

2.4 Flujo óptico

Dada una serie de fotogramas consecutivos, en los cuales se conoce la posición de un conjunto de puntos, el flujo óptico representa el movimiento que experimentan dichos puntos.

2.4.1 Método de Lucas-Kanade

Dadas dos imágenes F y G, el valor de los píxeles de cada una de las imágenes viene dado por $F(x)$ y $G(x)$, respectivamente; siendo x el vector que define la posición de un píxel en la imagen. Se toma una región de interés R, cuyo movimiento se pretende seguir, y de posición conocida en ambas imágenes. Se busca calcular un desplazamiento h que minimice la diferencia entre $F(x+h)$ y $G(x)$ en el interior de R. Todos los cálculos referentes a $F(x)$ y $G(x)$ se realizan únicamente en

el interior de la región R. Se considera que R es suficientemente pequeña para que $F(x)$ y $G(x)$ sean prácticamente lineales, de forma que [19]:

$$F(x+h) \simeq F(x) + hF'(x)$$

El error $E(h)$ que hay entre $F(x+h)$ y $G(x)$ viene dado por la siguiente expresión:

$$E(h) = \sum_x [F(x+h) - G(x)]^2 \simeq \sum_x \left[F(x) + hF'(x) - G(x)\right]^2$$

Para hallar la ecuación que minimiza el error se deriva E respecto a h y se iguala a 0.

$$E'(h) \simeq \sum_x 2F'(x)\left[F(x) + hF'(x) - G(x)\right]^2 = 0$$

De la última expresión se deduce la h que reduce la diferencia entre $F(x+h)$ y $G(x)$ en la región R:

$$h = \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2}$$

3

DESCRIPCIÓN DEL SISTEMA

El sistema que se utiliza en este proyecto está formado por una Up Board (Sección 3.3), un robot paralelo (Sección 3.4), una cámara RGB-D (Sección 3.5) y un dron (Sección 3.6). El software se desarrolla en ROS (Sección 3.1) con el lenguaje de programación Python (Sección 3.2).

3.1 ROS (Robot Operating System)

ROS (Robot Operating System)[1] es un entorno para el desarrollo de software para robots. Consiste en una colección de herramientas, librerías y convenciones destinada a simplificar esta tarea.

Un sistema que funcione en ROS tendrá sus tareas divididas en nodos, de manera que cada nodo realiza una tarea sencilla [20].

Los mensajes permiten el intercambio de información entre los distintos nodos de ROS. Cada mensaje será de un tipo. Los tipos de mensaje que permite utilizar ROS van desde los tipos primitivos como enteros o booleanos, hasta tipos más complejos como puede ser una imagen [20].

Los mensajes son enviados a través de tópicos o topics. Cuando se publique un mensaje en un tópico, este llegará a los nodos que estén suscritos a dicho tópico. Un nodo puede suscribirse o publicar en tantos tópicos como sea necesario, al igual que en un mismo tópico podrán publicar o suscribirse tantos nodos como fuera necesario. La transmisión de mensajes a partir de tópicos facilita el funcionamiento independiente de los nodos, ya que no hace falta que los nodos estén al tanto del origen o el destino de los mensajes [20].

[1] ROS (Acceso online el 31/7/18) `www.ros.org`

En general, en los paquetes de ROS hay archivos con la extensión `.launch`, escritos en XML. Con el comando `roslaunch` seguido del nombre de uno de estos archivos se hace funcionar varios nodos a la vez para realizar una tarea compleja [20].

3.1.1 Control de los motores

El paquete de ROS `dynamixel_controllers` contiene los nodos para el control de los motores Dynamixel MX-64AR[2]. El archivo `.launch` que se utiliza para esta tarea ha sido previamente creado por un alumno de la Universidad de Málaga.

3.1.2 Control de la cámara

El paquete `realsense_camera` de ROS incluye los nodos necesarios para controlar la cámara RealSense R200[3]. A partir de este paquete se han creado los archivos `.launch` que hacen funcionar a estos nodos (Sección 4.1).

3.2 Pyhton

3.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) es una librería de código abierto para visión por computador. Tiene interfaz en C++, Python y Java, y funciona en Windows, Linux, Mac OS, iOS y Android. La librería posee más de 2500 algoritmos[4].

3.2.2 rospy

La librería `rospy` de Pyhton permite utilizar scripts de Pyhton (`.py`) o como nodos de ROS. Incluye las funciones `Subscriber()` y `wait_for_message()` para recibir mensajes; la clase `Publisher()` para publicarlos, y otras funciones esenciales como `init_node()` [5].

3.2.3 Librerías de mensajes de ROS

En la librería `std_msgs.msg` se encuentran definidos los tipos de mensaje básicos y los tipos "MultiArray". El tipo `Float64MultiArray` es el que se ha escogido para las coordenadas[6]. Los tipos de mensaje más comunes en sensores están definidos en la librería `sensor_msgs.msgs`. Son de especial interés los tipos `Image`, imagen; `PointCloud2`, nube de puntos, y `CameraInfo`, que contiene información de la cámara como la resolución o la matriz de proyección[7].

[2] dynamixel_controllers - ROS Wiki (Acceso online el 6/8/18) `wiki.ros.org/dynamixel_controllers`
[3] realsense_camera - ROS Wiki (Acceso online el 10/7/18) `wiki.ros.org/realsense_camera`
[4] OpenCV library (Acceso online el 2/8/18) `https://opencv.org`
[5] rospy - ROS Wiki (Acceso online el 8/8/18) `wiki.ros.org/rospy`
[6] std_msgs - ROS Wiki (Acceso online el 8/8/18) `wiki.ros.org/std_msgs`
[7] sensor_msgs - ROS Wiki (Acceso online el 8/8/18) `wiki.ros.org/sensor_msgs`

3.2.4 cv_bridge

La librería `cv_bridge` contiene el objeto `CvBridge()`, que permite convertir mensajes de ROS de tipo imagen a imágenes de OpenCV con la función `imgmsg_to_cv2`[8].

3.3 Up Board

Toda la programación de este proyecto se realiza en un sistema embebido Up Board[9], de forma que el sistema puede funcionar de forma autónoma. La Up Board se conecta a los motores y a los arduinos a través de sus puertos USB, y a la cámara a través del puerto USB 3.0.

Figura 3.1: Up Board. Imagen extraída de `https://up-board.org/up/specifications/` (Acceso online el 7/1/2019)

3.4 Robot paralelo

Un robot paralelo o manipulador delta consiste en un cuerpo unido a un extremo móvil a través de 3 cadenas cinemáticas independientes [21]. El manipulador que se utilizará ha sido diseñado y construido por el departamento de Ingeniería de Sistemas y Automática de la Universidad de Málaga (Figura 3.2).

[8] cv_bridge - ROS Wiki (Acceso online el 8/8/18) `wiki.ros.org/cv_bridge`
[9] Up Board (Acceso online el 19/8/18) `www.up-board.org/up/`

Figura 3.2: Manipulador delta de 3 grados de libertad. Las piezas amarillas de arriba forman la base, en la que están los motores y la electrónica. La pieza amarilla de abajo es el efector final

3.4.1 Motores

Cada una de las cadenas cinemáticas del brazo es movida por un servomotor Dynamixel MX-64AR. Estos motores tienen control PID ajustable y resolución de 0.088º con encoder absoluto[10].

[10]MX-64 (Acceso online el 25/11/18) `http://emanual.robotis.com/docs/en/dxl/mx/mx-64/`

Figura 3.3: Motor MX-64. Imagen extraída de `http://emanual.robotis.com/docs/en/dxl/mx/mx-64/` (Acceso online el 25/11/18)

3.4.2 Modelos cinemáticos directo e inverso

El modelo cinemático directo permite conocer la posición del extremo del brazo (coordenadas cartesianas) conocidas las posiciones de los actuadores (coordenadas articulares), mientras que el modelo cinemático inverso determina la posición de los actuadores para que el brazo llegue a una posición [21].

Las funciones que utilizan los modelos cinemáticos del robot para realizar la transformación de coordenadas articulares a coordenadas cartesianas y viceversa han sido previamente implementadas por un alumno de la Universidad de Málaga.

3.4.3 Acelerómetros

El robot dispone de dos acelerómetros, uno en la base y otro en el efector final. Estos acelerómetros permitirán comprobar la estabilidad del dron. Esto se implementará en un futuro y no forma parte del este proyecto.

3.5 Cámara RGB-D

La cámara RGB-D que se utiliza es la Intel RealSense R200. Esta cámara dispone de una cámara RGB y un par estereoscópico de cámaras infrarrojas para percibir profundidad, además de un proyector láser infrarrojo para escanear la escena en 3 dimensiones[11].

[11]Introducing the Intel Realsense R200 Camera (Acceso online el 28/7/2018) `https://software.intel.com/articles/realsense-r200-camera`

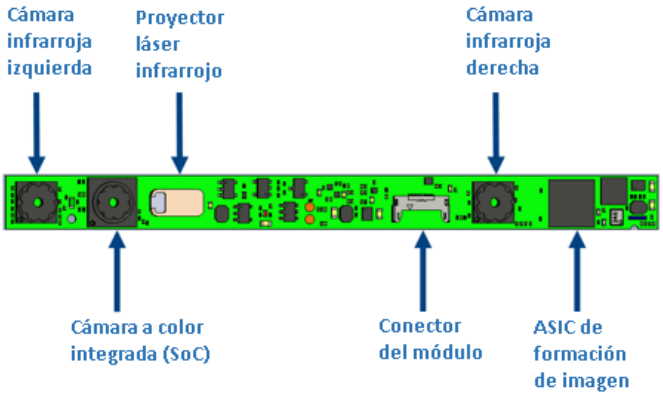

Figura 3.4: Módulo de cámara R200 sin carcasa. Adaptación de la imagen en Intel® RealSense™ Camera R200 Product Datasheet (Acceso online el 7/1/2019) `https://www.intel.es/content/www/es/es/support/articles/000023534/emerging-technologies/intel-realsense-technology.html`

3.6 Dron

En el futuro el brazo manipulador se utilizará montado en un dron FV8 de Atyges[12]. Esto no forma parte del presente proyecto.

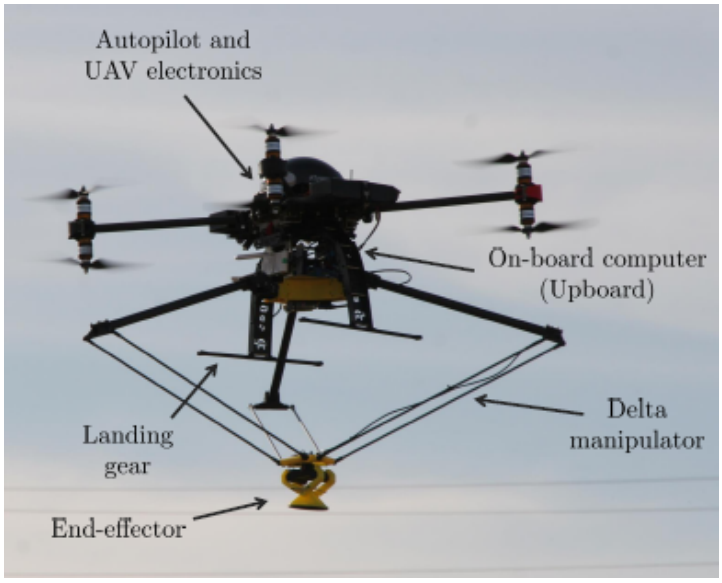

Figura 3.5: Dron FV8 de Atyges en el que están montados tren de aterrizaje y manipulador delta. Imagen extraída de [11]

[12]ATYGES. RPAS FV8 (Acceso online el 7/1/19) `http://www.atyges.es/drones/producto/3/rpas-fv8`

4

IMPLEMENTACIÓN DE LA LOCALIZACIÓN

La primera parte del control visual del brazo, la localización del objetivo mediante cámara RGB-D, se explica en este capítulo. Esta parte requiere el funcionamiento de un `.launch` que controla la cámara (Sección 4.1) y de un nodo de localización (Sección 4.4). También se explica la implementación de otro nodo directamente relacionado con la parte visual, cuya función es la de determinar el rango de color del objeto a localizar antes de iniciar la localización, en la Sección 4.3.

4.1 Configuración de la cámara

El paquete RealSense contiene archivos `.launch` para hacer funcionar la cámara con distintas configuraciones. En un primer momento se ha utilizado uno de los incluidos en este paquete. Al hacer funcionar la cámara con la configuración definida en este archivo, se publicaba la información necesaria para el control visual, pero también información adicional, como la nube de puntos registrada, que ralentizaba la publicación de los mensajes de interés. Al final se opta por utilizar un archivo `.launch` personalizado en el que solo se publica la información imprescindible [22].

Se han habilitado la imagen a color en 640×480 y la imagen de profundidad en 320×240, a 30 fps. Se han creado dos archivos distintos: en uno de ellos se habilita la nube de puntos y en el otro no. Las imágenes, la nube de puntos y las matrices de proyección se publican en sus respectivos tópicos durante el funcionamiento del `.launch`. En las Figuras 4.1, 4.2, 4.3 y 4.4, se muestran como ejemplo una imagen RGB, una imagen de profundidad y una nube de puntos. La nube de puntos se ha visualizado con ayuda de VTK[1].

[1]VTK - The Visualization Toolkit (Acceso online el 10/1/19) `https://vtk.org/`

Figura 4.1: Ejemplo de imagen RGB captada por la cámara

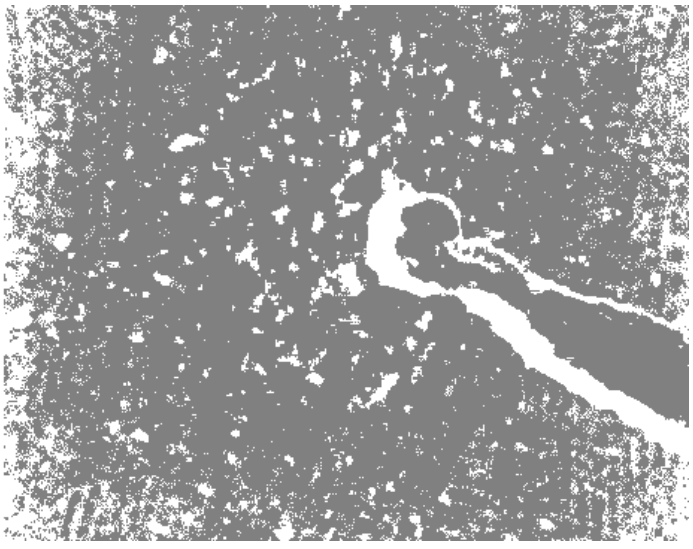

Figura 4.2: Ejemplo de imagen de profundidad captada por la cámara

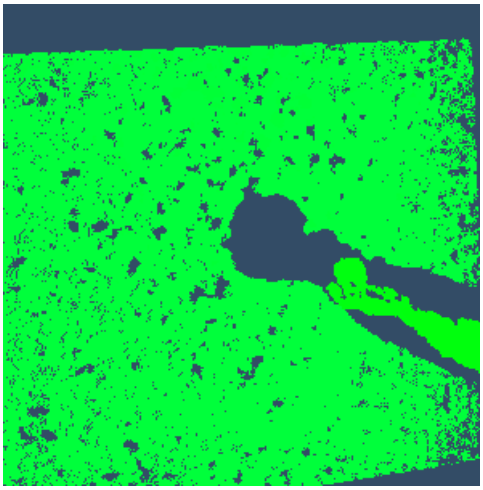

Figura 4.3: Visualización de la nube de puntos con ayuda de VTK

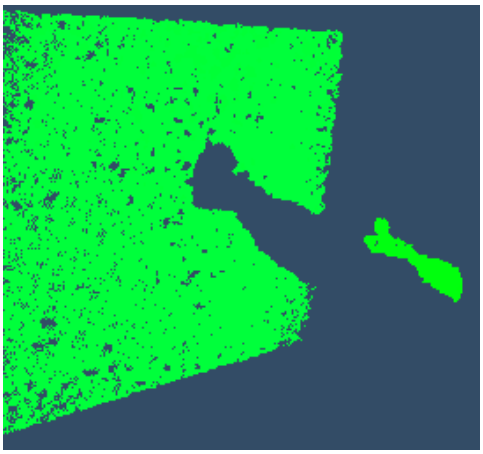

Figura 4.4: Visualización de la nube de puntos con ayuda de VTK (desde otro ángulo)

4.2 Calibración de la cámara RGB

Los parámetros intrínsecos de la cámara se calculan mediante la calibración de la cámara RGB. Para calibrar la cámara se ha utilizado el paquete `camera_calibration` de ROS[2]. El nodo `camera_calibrator.py` permite calibrar una cámara a partir de las imágenes que se publican. Durante la calibración, se mueve un tablero rectangular por distintas posiciones y con distintas inclinaciones para determinar los parámetros intrísecos (Figura 4.5.

[2]camera_calibration - ROS Wiki (Acceso online el 19/12/18) `wiki.ros.org/camera_calibration`

Figura 4.5: Una de las imagenes captadas por la cámara durante la calibración

La matriz que se obtiene de esta manera es la que se utiliza para convertir las posiciones en la imagen RGB en puntos reales en 3 dimensiones. A continuación se muestran los parámetros de la cámara que se obtienen tras la calibración: la matriz de la cámara (K), los parámetros de distorsión (D), la matriz de rectificación (R) y la matriz de proyección (P).

$$K = \begin{pmatrix} 611.455856 & 0.000000 & 336.466258 \\ 0.000000 & 616.925065 & 225.791276 \\ 0.000000 & 0.000000 & 1.000000 \end{pmatrix}$$

$$D = \begin{pmatrix} 0.055727 & -0.332696 & 0.000570 & -0.001576 & 0.000000 \end{pmatrix}$$

$$R = \begin{pmatrix} 1.000000 & 0.000000 & 0.000000 \\ 0.000000 & 1.000000 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 \end{pmatrix}$$

$$P = \begin{pmatrix} 604.822388 & 0.000000 & 336.913277 & 0.000000 \\ 0.000000 & 617.403381 & 225.754649 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 & 0.000000 \end{pmatrix}$$

4.3 Determinación del rango de color

La imagen RGB que captura la cámara debe convertirse en una imagen binaria en la que se localizará el objeto de interés. Para hacer esto se utiliza la función `inRange()` de OpenCv [23]:

```
cv2.inRange(src, lowerb, upperb[, dst]) → dst
```

$$dst(I) = lowerb(I)_0 \leq src(I)_0 \leq upperb(I)_0$$

Cuando a la función `inRange()` se le introducen como argumentos una imagen RGB, unos valores RGB mínimos y unos valores RGB máximos; devuelve una imagen binaria. En esta imagen binaria, los píxeles blancos son los píxeles cuyo color en la imagen original estaba comprendido entre los valores RGB, y el resto de píxeles son negros. Hay dos nodos que hacen uso de esta función: uno para obtener el rango de color de un objeto de interés (a continuación), y otro que hace uso de un rango de color introducido para localizar objetos de dicho color (Sección 4.4).

El nodo con el que deducir el rango de color tiene que utilizarse mientras funcionen los nodos de cámara, con la UpBoard conectada a cámara, monitor teclado y ratón. Dispone de 6 barras en las que se pueden ajustar los valores RGB mínimos y máximos (Figura 4.6). Estas barras se se han creado con las funciones `createTrackbar()` y `getTrackbarPos()` de OpenCV [23]. La imagen RGB a la que está suscrito el nodo (Figura 4.7) y la imagen binaria (Figura 4.8) se muestran simultáneamente, y la imagen binaria se actualiza en tiempo real al modificar el rango de color. En el ejemplo de las figuras se ha ajustado el rango de color para que se pueda localizar la pelota azul en la imagen binaria.

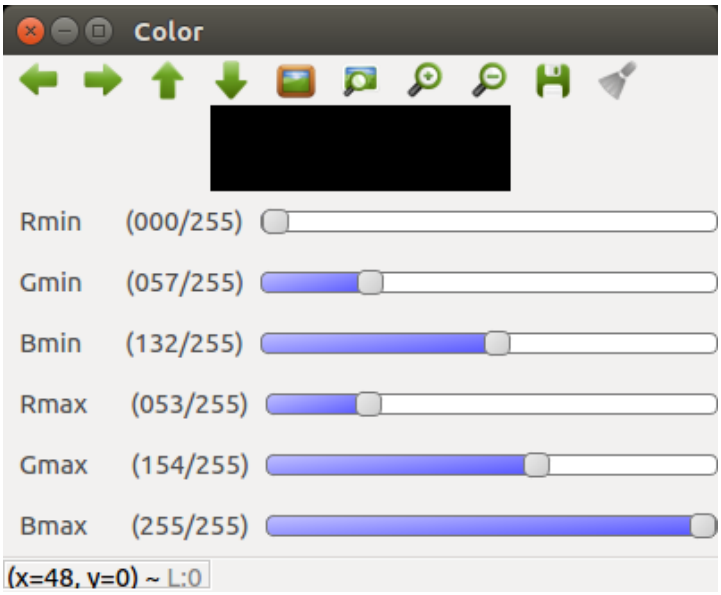

Figura 4.6: Barras para ajustar los valores RGB mínimos y máximos

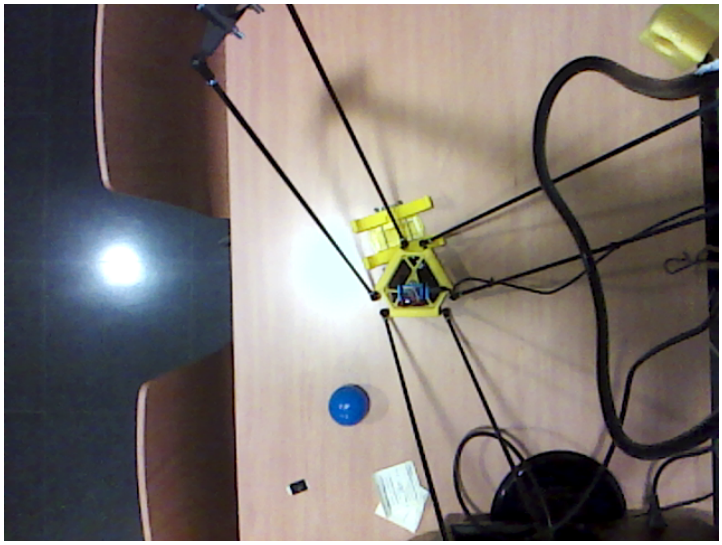

Figura 4.7: Imagen RGB original captada por la cámara

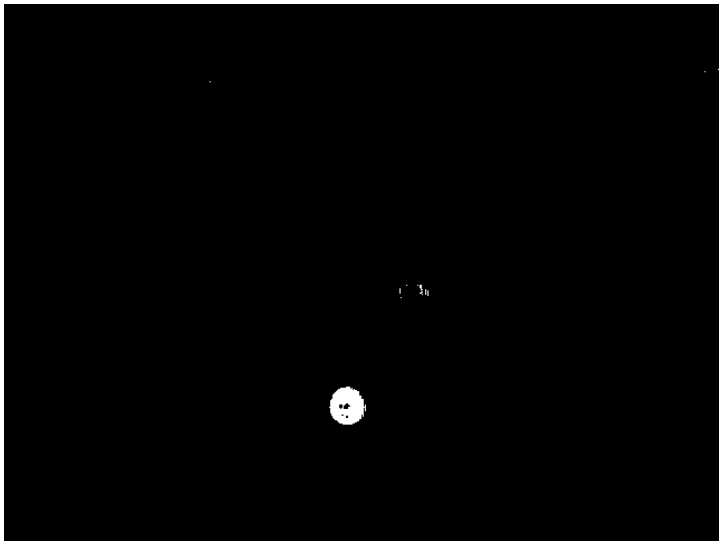

Figura 4.8: Imagen binaria enmascarada según los valores RGB elegidos en las barras

Después de haberse determinado el rango de color, este se ha introducido en el nodo encargado de la localización (Sección 4.4).

4.4 Cálculo de la posición objetivo

A continuación se explica paso a paso cómo funciona el nodo que calcula la posición del objetivo a partir de la información de la cámara.

4.4.1 Obtención de la imagen binaria

La obtención de la imagen binaria se realiza con la función `inRange()` de OpenCV, de forma similar a como se hizo en la Sección 4.3, pero los valores RGB se han introducido en el código para que no sea necesario el ajuste de los mismos cada vez que se utiliza este nodo.

La búsqueda del objeto en la imagen binaria se puede hacer más sencilla si se elimina el ruido. Para hacer esto se ha realizado sobre la imagen binaria una erosión seguida de una dilatación.

La erosión y la dilatación son dos operaciones morfológicas que consisten en obtener una modificación de una imagen a partir de un kernel. El kernel es una figura que normalmente tiene forma de cuadrado o círculo. El punto central del kernel se sitúa sobre cada uno de los píxeles de la imagen original y, según la operación que se esté realizando, se le asigna un valor al píxel que le corresponda en la imagen resultante [23].

En la erosión, el píxel de la imagen resultante tienen el valor mínimo de los píxeles contenidos en el kernel, con el resultado de una imagen con menos ruido y con los objetos blancos más pequeños. Con la dilatación se toma el máximo de los valores de los píxeles contenidos en el kernel, aumentándose así el tamaño de los objetos blancos. En la imagen resultante de la realización de las dos operaciones hay menos ruido y los objetos blancos se muestran en su tamaño original. La erosión y la dilatación se realizan con las funciones de OpenCV `erode()` y `dilate()`, respectivamente [23].

Partiendo de la imagen de la Figura 4.9 se ha obtenido la imagen binaria de la Figura 4.10. A esta imagen binaria se le ha aplicado filtro de ruido con kernel cuadrado de 3×3 píxeles (Figura 4.11) y de 5×5 píxeles (Figura 4.12).

Figura 4.9: Imagen a color de la que se obtiene la imagen binaria

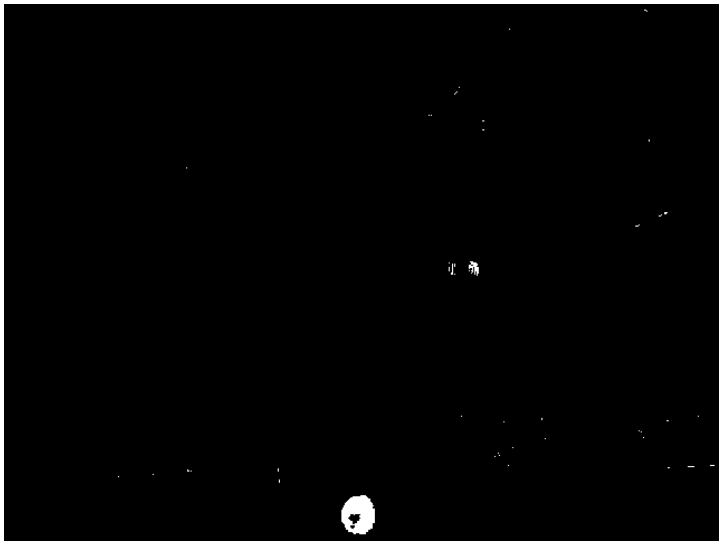

Figura 4.10: Imagen binaria previa al filtro de ruido

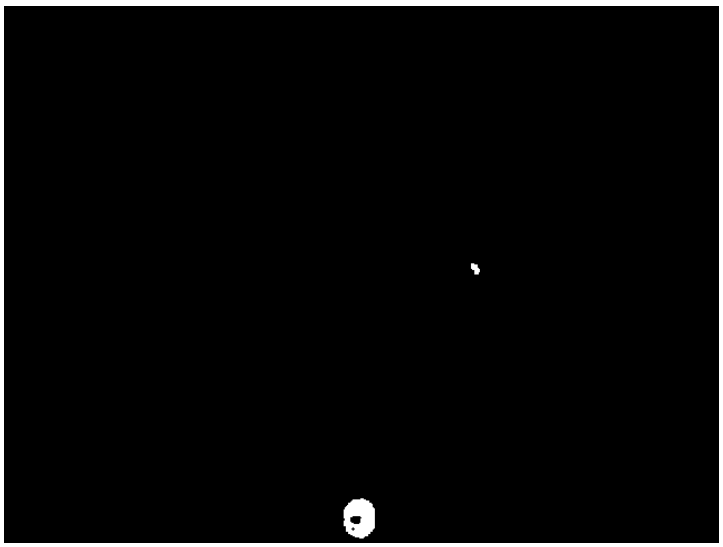

Figura 4.11: Imagen binaria posterior al filtro de ruido con kernel de 3×3 píxeles

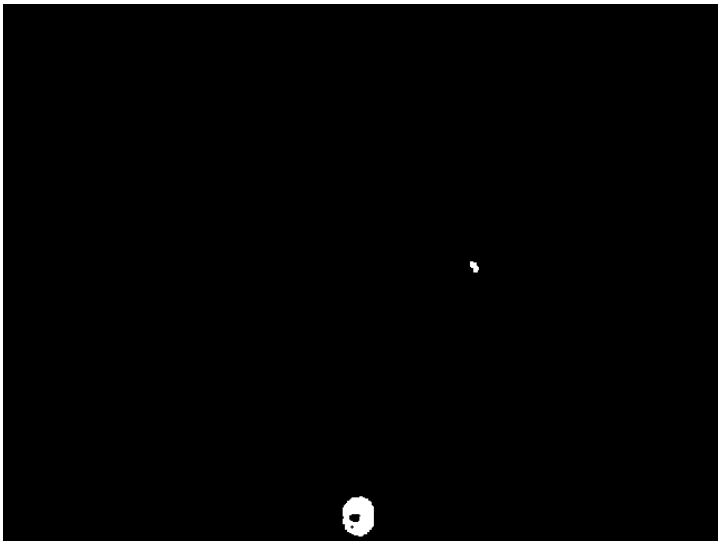

Figura 4.12: Imagen binaria posterior al filtro de ruido con kernel de 5×5 píxeles

Las Figuras 4.10, 4.11 y 4.12 se observa que un kernel de 3×3 píxeles es suficiente para eliminar una gran cantidad de ruido, y el sensor de color azul que está sobre el manipulador (Subsección 3.4.3) deja de verse si el kernel es de 5×5 píxeles. Utilizar el kernel de 5×5 píxeles no supone una mejora significativa en la eliminación de ruido así que se opta por usar el de 3×3 píxeles. El hecho de que el sensor sea del mismo color que la pelota es un asunto que se resuelve en selección de contornos, a continuación.

4.4.2 Búsqueda de contornos

El objeto se encuentra en la imagen buscando los contornos con la función de OpenCv `findContours()`, que utiliza los algoritmos de Suzuki-Abe. El algoritmo de Suzuki-Abe que se ha elegido es el que únicamente tiene en cuenta los contornos de mayor jerarquía, es decir, aquellos que no están rodeados por otro contorno [24]:

1. Se escanea la imagen recorriendo cada fila de píxeles de izquierda a derecha. El comienzo de un nuevo contorno se identifica como un píxel de valor 1 precedido por un píxel de valor 0. Los píxeles de valor 1 seguidos por un píxel de valor 0 son finales de contorno. Al nuevo contorno se le asigna un número de identificación NBD.

2. Se recorre todo el contorno y se cambia el valor de sus píxeles a NBD, excepto a aquellos píxeles que son final de contorno, a los que se les cambia el valor a -NBD. Se reanuda el escaneo de la imagen.

3. Se utiliza una variable LNBD que indica el valor del último píxel distinto de 0 por el que se ha pasado. Si se encuentra un nuevo contorno cuando LNBD es mayor que 0, el nuevo contorno se ignora por encontrarse dentro de otro contorno.

4. LNBD se pone a 0 al final de cada fila.

Una vez encontrados todos los contornos de mayor jerarquía en la imagen binaria, se calcula el área de los mismos con la función de OpenCv `contourArea()`. Se considera que el contorno de mayor área es el correspondiente al objeto de interés que se quiere localizar y, por tanto, el que se tiene en cuenta en los pasos siguientes. A dicho contorno se le exige que envuelva un área mínima de 450 píxeles, que se ha comprobado que es suficiente para que la pelota pueda ser detectada hasta a 1200mm de distancia a la cámara y el sensor azul sea ignorado por el algoritmo en cualquier punto del espacio de trabajo del brazo (el espacio de trabajo se trata en la Sección 6.1).

4.4.3 Cálculo del centroide

El centroide de un contorno es la media aritmética de las posiciones de los puntos en su interior. El centroide se calcula para obtener la posición del píxel en el que se considera que se encuentra el centro del objeto cuyo contorno se ha identificado.

Los momentos del contorno se con la función `moments()` de OpenCv. En la ecuación siguiente se define el momento m_{ji} de un contorno $C(x,y)$ [23]:

$$m_{ji} = \sum_{x,y} \left(C(x,y) x^j y^i \right)$$

El centroide $(\overline{x}, \overline{y})$ del contorno se calcula a partir de los momentos como se muestra a continuación:

$$\overline{x} = \frac{m_{10}}{m_{00}}, \overline{y} = \frac{m_{01}}{m_{00}}$$

4.4.4 Proyección inversa

El paso siguiente a localizar el objeto en la imagen es calcular sus coordenadas reales. Para hacer esta operación se considerará el modelo pinhole, que a cada punto real (X,Y,Z) le asigna un píxel (x,y) según la matriz de perspectiva P:

$$P = \begin{pmatrix} f_x & 0 & c_x & T_x \\ 0 & f_y & c_y & T_y \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$x = \frac{u}{w}, y = \frac{v}{w}$$

Cada uno de los términos que forman matriz P se explicó en la Sección 2.1. La matriz de perspectiva de la imagen RGB se obtiene a partir de la calibración de la cámara en el caso de la imagen RGB (Sección 4.2), mientras que en el caso de la imagen de profundidad se utiliza la que se publica en el tópico `/camera/depth/camera_info`. Al despejar las coordenadas reales (X,Y) se llega a las siguientes expresiones generales:

$$X = \frac{(x - c_x)Z - T_x}{f_x}, Y = \frac{(y - c_y)Z - T_y}{f_y} \tag{4.1}$$

Que en este caso concreto de la imagen binaria obtenida a partir de la cámara RGB, se pueden simplificar ya que la cámara RGB es la cámara principal de la RealSense R200 y su origen de coordenadas coincide con el centro óptico ($T_{xb} = T_{yb} = 0$):

$$X = \frac{(x_b - c_{xb})Z}{f_{xb}}, Y = \frac{(y_b - c_{yb})Z}{f_{yb}} \tag{4.2}$$

La coordenada Z, que se corresponde con la distancia perpendicular al plano de la cámara, no se puede deducir de la ecuación de perspectiva con los datos que se extraen de la imagen bidimensional, y es necesario conocerla para calcular las otras dos coordenadas. A continuación se proponen dos formas de obtener la coordenada Z.

4.4.4.1 Proyección inversa con nube de puntos

Una de las soluciones por las que se opta para calcular la posición real del punto de interés a partir de la posición en la imagen, hace uso del tópico `camera/depth/points`, en el que se publica una nube de puntos del tipo `sensor_msgs/PointCloud2`. En esta nube de puntos se guardan las coordenadas (X,Y,Z) de los puntos que la cámara registra.

A partir de la Ecuación 4.2 se pretende calcular cual de los puntos de la nube de puntos pasa por la línea que se proyecta en el píxel (x_b, y_b). Como se trata de un proceso iterativo, se definen las constantes K_x y K_y para relacionar las coordenadas (X,Y) con Z sin necesidad de repetir los cálculos:

$$K_x = \frac{x_b - c_{xb}}{f_{xb}}, K_y = \frac{y_b - c_{yb}}{f_{yb}}$$

$$X = K_x Z, Y = K_y Z \tag{4.3}$$

Se recorre la nube de puntos en busca del punto más cercano a la recta que se proyecta en el píxel (x_b, y_b), que es la que se corresponde con la Ecuación 4.3. Se calcula la distancia euclídea entre las coordenadas (X,Y) de cada punto de la nube de puntos con las coordenadas (X,Y) que tendría dicho punto si pasara por la línea que se proyecta en el píxel.

$$\min\left[(X - K_x Z)^2 + (Y - K_y Z)^2\right]$$

De esta forma se obtienen las coordenadas reales (X, Y, Z) con respecto a la cámara.

4.4.4.2 Proyección inversa con imagen de profundidad

La imagen de profundidad se publica en tiempo real en el tópico `camera/depth/image_raw`. Esta es una imagen en la que el valor de cada píxel es la profundidad en milímetros, en formato de enteros sin signo de 16 bits.

Se aplica la Ecuación 4.1 a la imagen de profundidad:

$$X = \frac{(x_d - c_{xd})Z - T_{xd}}{f_{xd}}, Y = \frac{(y_d - c_{yd})Z - T_{yd}}{f_{yd}} \tag{4.4}$$

Si se igualan las expresiones de (X, Y) según la imagen binaria (Ecuación 4.2) con las de la imagen de profundidad (Ecuación 4.4), se obtienen ecuaciones con las que se puede pasar de coordenadas en la imagen binaria (x_b, y_b) a coordenadas en la imagen de profundidad (x_d, y_d):

$$x_d = c_{xd} + \frac{f_{xd}}{f_{xb}}(x_b - c_{xb}) - \frac{T_{xd}}{Z}, y_d = c_{yd} + \frac{f_{yd}}{f_{yb}}\left(y_b - c_{yb}\right) - \frac{T_{yd}}{Z}$$

En estas ecuaciones hay dos términos que dependen de la coordenada Z, que se pueden ignorar si esta es mucho mayor que la distancia entre las cámaras:

$$x_d = c_{xd} + \frac{f_{xd}}{f_{xb}}(x_b - c_{xb}), y_d = c_{yd} + \frac{f_{yd}}{f_{yb}}\left(y_b - c_{yb}\right)$$

Cuando se hayan calculado las coordenadas en la imagen de profundidad (x_d, y_d), se consulta la distancia Z en dicha imagen y se calculan las coordenadas (X, Y) con las Ecuaciones 4.2.

En la imagen de profundidad que captura la cámara, la profundidad no está definida para todos los puntos de dicha imagen. Por ello, se ha visto necesario un método que contemple la posibilidad de que la profundidad no esté definida en el punto que quisiera consultarse. El método que se ha implementado está basado en el que se propone en [25].

Cuando se da el caso de que la cámara no ha sido capaz de medir la distancia en un píxel de la imagen de profundidad, el valor que presenta dicho píxel es 0. Si después de consultarse el valor de un píxel se obtiene un 0, se busca un valor distinto en los píxeles cercanos al mismo. Se ha tomado una región cuadrada centrada en el píxel de interés. Esta región se recorre y se toma el mayor valor de distancia en vez de el 0 que se obtuviera inicialmente. De esta forma se obtiene un valor de distancia con una precisión razonable y con gran velocidad.

4.5 Esquema de la etapa de localización

En la Figura 4.13 se presenta un esquema en el que se resume la tarea de localización.

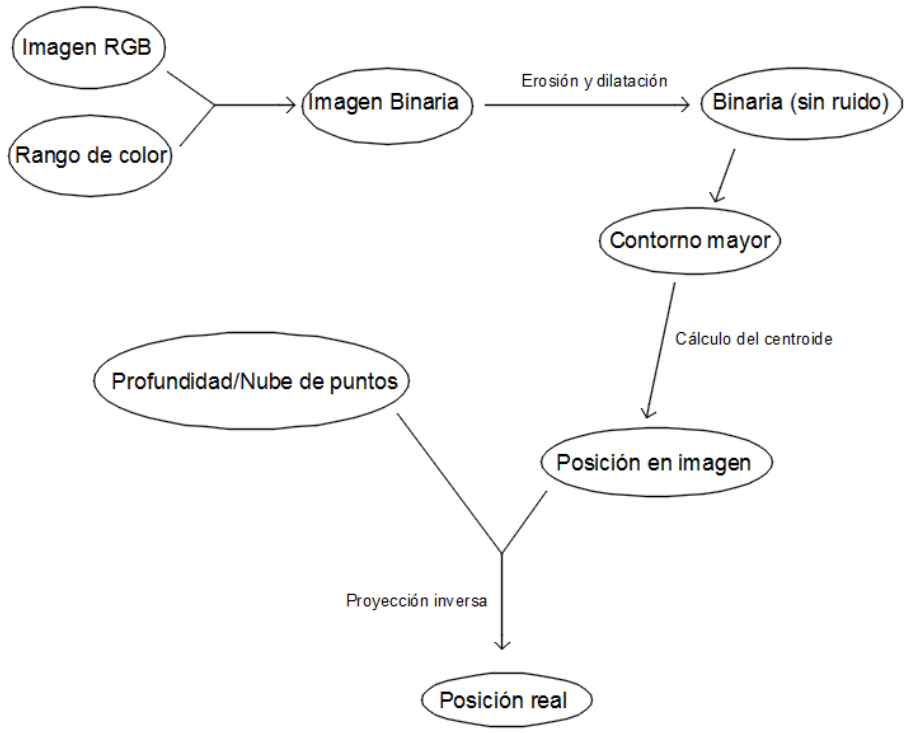

Figura 4.13: Esquema del proceso de localización. Se calcula la posición real del objeto partiendo de la imagen RGB y el rango de color, y el mapa de profundidad o la nube de puntos

5

INTEGRACIÓN DE LA CÁMARA EN EL SISTEMA

La integración física de la cámara en el sistema es esencial para realizar el control visual del brazo. Se ha decidido situar la cámara en la base del robot, que es una posición en la que la cámara no está sometida al movimiento del brazo ni a posibles contactos con el entorno. Se ha procurado que la posición de la cámara le permita abarcar la mayor parte posible del espacio de trabajo del brazo, así que se ha situado mirando hacia abajo, de la forma más centrada posible. Una vez seleccionada la posición que ocupa la cámara, se han diseñado y fabricado soportes para mantenerla en esta posición (Sección 5.1). Después de fijar la cámara, se procede a hallar la relación entre el sistema de referencia de la misma y el del manipulador (Sección 5.2).

5.1 Colocación de la cámara

Para fijar la cámara bajo la base del robot se han fabricado mediante impresión en 3D dos piezas para mantener cámara en la posición deseada. Las piezas se han diseñado con el software SolidWorks[1].

[1]SolidWorks (Acceso online el 10/8/18) `https://www.solidworks.com`

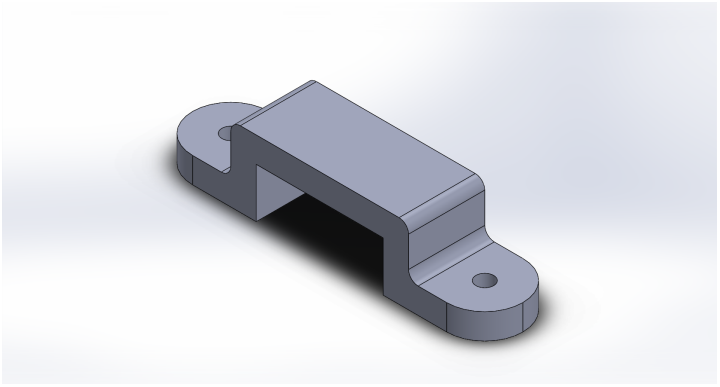

Figura 5.1: Vista de la pieza en SolidWorks

La sección de la cámara tiene dimensiones 8mm × 20mm, mientras que el hueco de las piezas es de 7.8mm × 20mm. Las piezas se han atornillado a la base del robot, de forma que envuelven la cámara y la presionen contra la base del robot sin holgura. Después de colocar la cámara se ha comprobado que los soportes no obstruyen el campo de visión de ninguna de las cámaras (RGB ni infrarrojos).

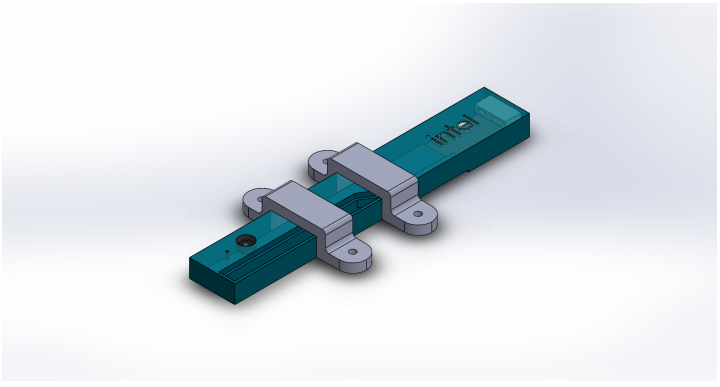

Figura 5.2: Vista de las piezas sobre la cámara en SolidWorks

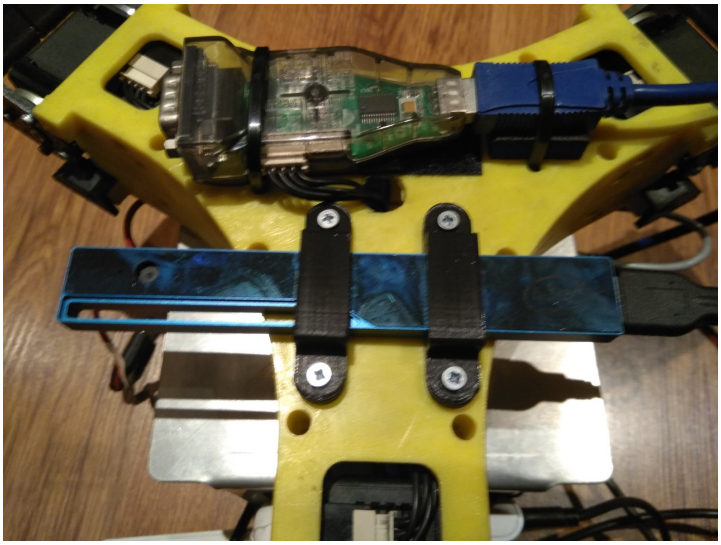

Figura 5.3: Fotografía de la cámara unida a la base del robot con las piezas

5.2 Relación entre el sistema de referencia de la cámara y el del brazo

Los algoritmos de localización, cuya implementación se explica en el Capítulo 4, permiten obtener la posición real del objeto de interés con respecto al sistema de referencia de la cámara. Para mover el brazo a dicha posición, es necesario transformar las coordenadas del sistema de referencia de la cámara en coordenadas en el sistema de referencia del brazo. Estos sistemas de referencia están relacionados por una matriz de transformación como la que se muestra en la siguiente ecuación:

$$\begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & x_{b0} \\ \sin\theta & \cos\theta & 0 & y_{b0} \\ 0 & 0 & -1 & z_{b0} \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} \tag{5.1}$$

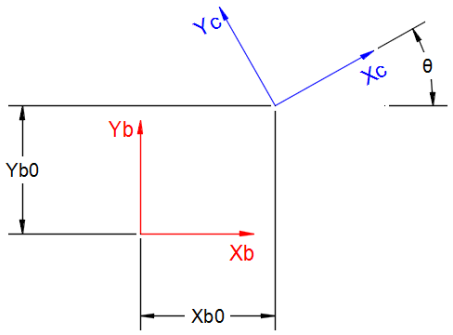

Figura 5.4: Relación entre el sistema de referencia de la cámara y el del brazo en los ejes X e Y

Donde (x_c, y_c, z_c) son las coordenadas en el sistema de referencia de la cámara, (x_b, y_b, z_b) son las coordenadas en el sistema de referencia del brazo, θ es el ángulo de giro entre ambos sistemas de referencia con respecto al eje Z, (x_{b0}, y_{b0}, z_{b0}) son las coordenadas del origen del sistema de referencia de la cámara en el sistema de referencia del brazo. Por la forma en la que se ha fijado la cámara, se considera que la rotación con respecto a los ejes X e Y es despreciable. El término de la tercera fila y la tercera columna en la matriz es negativo debido a que se ha invertido uno de los ejes porque la disposición de los mismos impide que una rotación haga coincidir completamente a ambos sistemas de referencia (Figura 5.5).

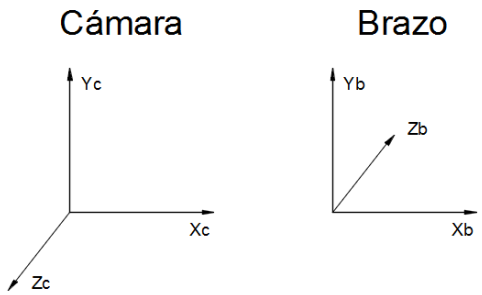

Figura 5.5: Comparación de la disposición de los ejes de la cámara y del brazo

Se procede a determinar los parámetros que conforman la matriz de transformación. Para ello se han tomado las coordenadas de varios puntos en los sistemas de referencia de la cámara y del brazo (Tablas 5.1 y 5.2), siguiendo el procedimiento que se explica en la Sección 7.3.

Tabla 5.1: Datos de las coordenadas X e Y (experimento de la Sección 7.3)

	Cámara (mm)		Brazo (mm)	
Punto	x	y	x	y
P1	125,02	124,34	237,56	-95,34
P2	-226,27	206,97	138,6	230,76
P3	-205,68	-125,78	-131,5	43,6
P4	95,1	-219,36	-75,45	-223,65
P5	3,55	44,79	91,48	5
P6	267,29	235,46	273,18	-134,5
P7	-54,74	-231,41	-168,32	-118,32
P8	-401,34	135,06	2,88	288,06

Tabla 5.2: Datos de la coordenada Z (experimento de la Sección 7.3)

	Cámara (mm)	Brazo (mm)
Z1	1034	-923,67
Z2	898	-776,88
Z3	769	-645,09
Z4	908	-784,82
Z5	967	-873,93
Z6	827	-719,9

5.2.1 Coordenadas X e Y

A partir de los puntos obtenidos se calcula θ, el ángulo de giro entre los sistemas de referencia. Este ángulo entre los sistemas de referencia es igual al giro que experimenta cada pareja de puntos con el cambio de sistema de referencia.

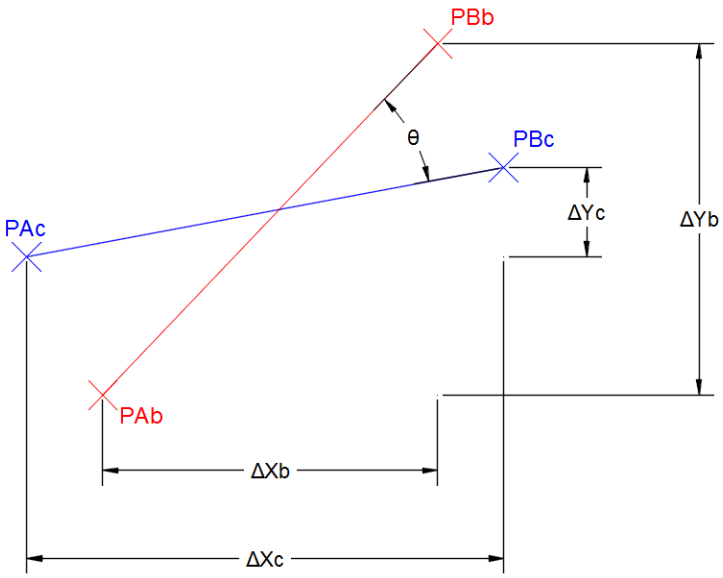

Figura 5.6: Sea una pareja de puntos PAc-PBc en el sistema de referencia de la cámara. Si se cambia de sistema de referencia, la pareja de puntos aparenta convertirse en PAb-PBb. El ángulo de giro θ es el mismo que el que hay entre los sistemas de referencia. También se representa la distancia entre los puntos paralela a los ejes del sistema de referencia

En la Figura 5.6 se han superpuesto las posiciones de una pareja de puntos genérica PA-PB respecto a los sistemas de referencia de la cámara y del brazo, para visualizar el giro que experimenta. Δx_c es la distancia entre los puntos en el eje X de la cámara, Δy_c es la distancia entre los puntos en el eje Y de la cámara, Δx_b es la distancia entre los puntos en el eje X del brazo y Δy_b es la distancia entre los puntos en el eje Y del brazo. El ángulo de giro se puede obtener a partir del coseno, que se deduce de la expresión siguiente:

$$\cos\theta = \frac{\Delta x_c \cdot \Delta x_b + \Delta y_c \cdot \Delta y_b}{\sqrt{\Delta x_c^2 + y_c^2} \cdot \sqrt{\Delta x_b^2 + \Delta y_b^2}}$$

En la Tabla 5.3 se muestra el ángulo que se ha calculado para cada pareja de puntos, así como la media de todos ellos y el error al tomar la media en cada caso.

Tabla 5.3: Ángulo de giro que experimenta cada pareja de puntos

Pareja de puntos		$\cos\theta$	θ (º)	Error (º)
P1	P2	0,501777	59,88235	-1,77999
P1	P3	0,53389	57,73134	-3,931
P1	P4	0,458108	62,73494	1,0726
P1	P5	0,379374	67,7051	6,042763
P1	P6	0,074945	85,70191	24,03957
P1	P7	0,500724	59,95209	-1,71025
P1	P8	0,539324	57,36236	-4,29997
P2	P3	0,517704	58,82166	-2,84068
P2	P4	0,465893	62,23199	0,569655
P2	P5	0,397478	66,57941	4,91707
P2	P6	0,291081	73,07733	11,41499
P2	P7	0,45877	62,69224	1,029904
P2	P8	0,704392	45,21953	-16,4428
P3	P4	0,486748	60,87292	-0,78942
P3	P5	0,655941	49,00898	-12,6534
P3	P6	0,482888	61,12582	-0,53652
P3	P7	0,377419	67,82611	6,163769
P3	P8	0,411963	65,67177	4,009431
P4	P5	0,570033	55,24744	-6,4149
P4	P6	0,574722	54,91985	-6,74249
P4	P7	0,599094	53,19499	-8,46735
P4	P8	0,451203	63,17907	1,516735
P5	P6	0,28602	73,38015	11,71781
P5	P7	0,606119	52,69059	-8,97175
P5	P8	0,499229	60,05096	-1,61138
P6	P7	0,537265	57,50232	-4,16002
P6	P8	0,407794	65,93368	4,271339
P7	P8	0,402769	66,24859	4,586249
Media:			61,66234	

Si se toma θ = 61.66234º, la pareja que forman los puntos P1 y P6 es la que mayor error presenta, hasta el punto de ser inadmisible. Debido a esto, se ha considerado que este error es debido a que el punto P6 no se ha medido de forma correcta. Por tanto, se repiten los cálculos ignorando el punto P6 en la Tabla 5.4.

Tabla 5.4: Ángulo de giro que experimenta cada pareja de puntos, excluyendo P6

Pareja de puntos		$\cos\theta$	θ (º)	Error (º)
P1	P2	0,501777	59,88235	0,124998
P1	P3	0,53389	57,73134	-2,02601
P1	P4	0,458108	62,73494	2,977585
P1	P5	0,379374	67,7051	7,947748
P1	P7	0,500724	59,95209	0,194739
P1	P8	0,539324	57,36236	-2,39499
P2	P3	0,517704	58,82166	-0,9357
P2	P4	0,465893	62,23199	2,474639
P2	P5	0,397478	66,57941	6,822055
P2	P7	0,45877	62,69224	2,934889
P2	P8	0,704392	45,21953	-14,5378
P3	P4	0,486748	60,87292	1,11556
P3	P5	0,655941	49,00898	-10,7484
P3	P7	0,377419	67,82611	8,068754
P3	P8	0,411963	65,67177	5,914416
P4	P5	0,570033	55,24744	-4,50991
P4	P7	0,599094	53,19499	-6,56236
P4	P8	0,451203	63,17907	3,42172
P5	P7	0,606119	52,69059	-7,06676
P5	P8	0,499229	60,05096	0,293608
P7	P8	0,402769	66,24859	6,491233
Media:			59,75736	

El ángulo que se obtiene es $\theta = 59,75736^{\circ}$. Sin embargo, al haberse deducido este resultado a partir del coseno, no se puede descartar la solución $\theta = -59.75736^{\circ}$ hasta que se demuestre cuál de estos ángulos es el correcto. Quedan por calcular (x_{b0}, y_{b0}), las coordenadas de en X y en Y del sistema de referencia de la cámara según el brazo, que se deducen a partir de los puntos medidos, los puntos calculados y el ángulo de rotación. Se utiliza la Ecuación 5.1 para predecir (x_{b0}, y_{b0}) según cada punto que se ha considerado, tomando $\theta = 59.75736^{\circ}$ en la Tabla 5.5 y $\theta = -59.75736^{\circ}$ en la Tabla 5.6.

Tabla 5.5: Desplazamiento en (X,Y) según cada punto tomando $\theta = 59.75736^{o}$

	xb0	yb0
P1	282,0094	-322,995
P2	431,3653	356,427
P3	-136,568	287,5187
P4	-312,854	-267,806
P5	128,3861	-44,1419
P7	-340,665	13,74646
P8	321,6985	633,3272
Media:	53,33902	93,72526

Tabla 5.6: Desplazamiento en (X,Y) según cada punto tomando $\theta = -59.75736^{o}$

	xb0	yb0
P1	67,1747	-106,985
P2	73,7624	-34,5224
P3	80,75478	-67,8553
P4	66,1568	-103,492
P5	50,99791	-38,0082
P7	59,16566	-80,8333
P8	88,34175	-60,1083
Media:	69,47914	-70,2578

A la vista de los resultados obtenidos en las Tablas 5.5 y 5.6, se decide tomar $\theta = -59.75736^{o}$, $(x_{b0}, y_{b0}) = (69.47914, -70.2578)$mm.

5.2.2 Coordenada Z

Como se pudo ver en la Ecuación 5.1, la coordenada Z es independiente de las otras dos a la hora de realizar la transformación entre el sistema de referencia de la cámara y el del brazo. El único dato que hay que determinar es z_{b0}, que se calculará de forma empírica a partir de los datos de la Tabla 5.2. Se ha calculado z_{b0} según cada uno de los puntos que se ha tomado, y se ha tomado la media como una primera aproximación al valor real de z_{b0}, como se muestra en la Tabla 5.7.

Tabla 5.7: Desplazamiento en Z según cada punto

	Cámara (mm)	Brazo (mm)	zb0 (mm)	Error (mm)
Z1	1034	-923,67	110,33	-2,78833
Z2	898	-776,88	121,12	8,001667
Z3	769	-645,09	123,91	10,79167
Z4	908	-784,82	123,18	10,06167
Z5	967	-873,93	93,07	-20,0483
Z6	827	-719,9	107,1	-6,01833
Media:			113,1183	

Se observa que el error que se comete tomando $z_{b0} = 113,1183$mm es del orden de 10mm en todos los puntos medidos excepto en el punto Z5. Posiblemente esto se deba a que la medida de este punto se haya realizado de forma errónea, así que se opta por repetir los cálculos ignorando el punto Z5 (Tabla 5.8).

Tabla 5.8: Desplazamiento en Z según cada punto, excluyendo Z5

	Cámara (mm)	Brazo (mm)	zb0 (mm)	Error (mm)
Z1	1034	-923,67	110,33	-6,798
Z2	898	-776,88	121,12	3,992
Z3	769	-645,09	123,91	6,782
Z4	908	-784,82	123,18	6,052
Z6	827	-719,9	107,1	-10,028
Media:			117,128	

Se toma $z_{b0} = 117.128$mm.

5.2.3 Implementación de la transformación

La transformación (Ecuación 5.1, con $\theta = -59.75736^{\circ}$, $(x_{b0}, y_{b0}) = (69.47914, -70.2578)$mm, $z_{b0} = 117.128$mm) se aplica a la posición real del objeto según el sistema de referencia de la cámara para obtener la posición respecto del sistema de referencia del brazo, como se muestra en el esquema de la Figura 5.7.

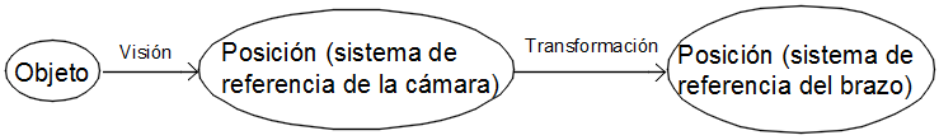

Figura 5.7: Esquema en el que se muestra la obtención de las posición en el sistema de referencia del brazo. La etapa de visión corresponde al Capítulo 4 y la transformación a la Ecuación 5.1 con los parámetros calculados

6

Implementación del movimiento del brazo

La segunda parte del control visual del brazo, la referente al movimiento del brazo, se trata en este capítulo. Para el movimiento del brazo son necesarios un `.launch` que controla los motores, un `.launch` para el movimiento rectilíneo (Sección 6.2) y uno de los nodos para la trayectoria desde el plano de reposo (Sección 6.3).

6.1 Espacio de trabajo

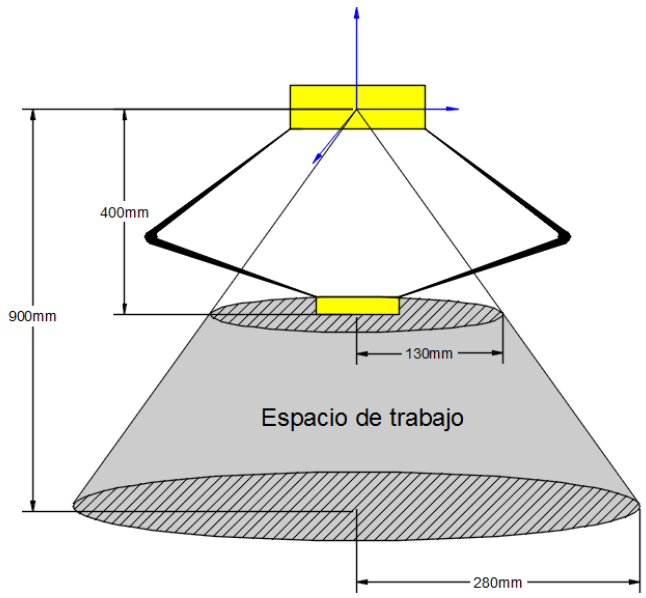

Figura 6.1: Esquema del espacio de trabajo utilizado

Se van a limitar las posiciones a las que los nodos puedan exigir al manipulador que se mueva para evitar que se someta a una una carga excesiva por intentar moverse a posiciones fuera de su alcance. Se considera un espacio de trabajo simplificado contenido en el espacio de trabajo total, con la forma de un cono truncado cuyo eje coincide con el eje Z del manipulador. Las bases del cono truncado tienen radios de 130mm y 280mm, y están situadas 400mm y 900mm por debajo del cuerpo del manipulador, respectivamente. Este espacio de trabajo simplificado se tiene en cuenta en cada una de las trayectorias que se proponen. En la Figura 6.1 se muestra un esquema del espacio de trabajo que se utiliza.

6.2 Movimiento rectilíneo entre dos puntos

Este nodo es el encargado de mover el manipulador en línea recta. Los nodos que determinen la posición a la que deba moverse el brazo publican las coordenadas cartesianas de dicha posición si quieren que el movimiento se realice en línea recta y a velocidad constante. Con ayuda de otros nodos pueden implementarse trayectorias formadas por la sucesión de movimientos en línea recta, lo que hace que este nodo sea útil en varios escenarios.

El nodo en cuestión está suscrito a la posición actual del manipulador y a la posición objetivo, ambas en coordenadas cartesianas. Si se recibe una posición objetivo fuera del espacio de trabajo que se ha delimitado, esta se ignora.

En función de la distancia entre las posiciones inicial y final se calcula el número de puntos en los que se divide la trayectoria, con un mínimo de 2 puntos y un punto adicional cada 50mm. Mediante interpolación se calculan las coordenadas cartesianas de cada punto intermedio. Los tiempos en que se alcanza cada punto se calculan a partir de la velocidad lineal deseada, que se ha fijado en 200mm/s. Las coordenadas articulares de cada punto y la velocidad articular en cada tramo se calculan con ayuda de los modelos cinemáticos del manipulador.

6.3 Trayectoria hacia el objetivo

En esta sección se explica el funcionamiento del nodo que planifica las trayectorias. Este nodo está suscrito a la posicion objetivo, publicada por el nodo de localización (Sección 4.4), y publica la posición a la que debe moverse el brazo, a la cual está suscrito el nodo de la Sección 6.2.

La trayectoria del manipulador tiene como origen el punto $(0, 0, -400)$mm. El movimiento está formado por una fase de reposo y otra fase en la que el brazo se mueve hacia la posición deseada de forma que se accione el mecanismo de la pinza al tocar el objetivo. El paso de la primera fase a la segunda se realiza tras comprobarse que la posición del dron sea estable (como ya se dijo en la Subsección 3.4.3, no forma parte de este proyecto). La trayectoria se da por finalizada cuando se acciona la pinza o cuando transcurren 5 segundos desde que se saliera del estado de reposo, en cuyo caso se regresa al punto de origen.

Se han implementado en ROS varios diseños para el nodo que planifica las trayectorias, con la intención de estudiar cuál de estas es la más óptima en esta aplicación. En algunas de estas trayectorias se permite el movimiento del brazo durante la etapa de reposo, aunque limitado a un plano de reposo ($Z = -400$mm).

En todos los tipos de trayectoria propuestos, se contempla la posibilidad de que el manipulador impida que el objeto a seguir aparezca en la imagen. La solución por la que se opta es utilizar la última posición publicada, ya que en caso de no localizarse el objeto en la imagen no se publica ningún mensaje en el tópico correspondiente. Sin embargo, en los nodos en los que se permite el movimiento en el plano de reposo, existen posiciones en las que el manipulador no obstruye la vista del objeto en la imagen RGB, pero sí impide que la distancia se calcule de forma correcta, con la consecuencia de que se publica una medida errónea de la posición. Se ha procurado evitar esto almacenando la última posición antes de iniciarse el movimiento.

6.3.1 Movimiento rectilíneo desde el origen

Durante la fase de reposo, el manipulador se encuentra quieto en el punto de origen de trayectoria. Cuando se cumpla la condición de estabilidad, el brazo se mueve directamente al punto de destino en línea recta.

Al realizarse la trayectoria de esta forma no se permite el movimiento del brazo durante la etapa de reposo y se aprovecha por completo el espacio de trabajo inicial, pero no se tiene control en el ángulo de incidencia del manipulador sobre el objetivo.

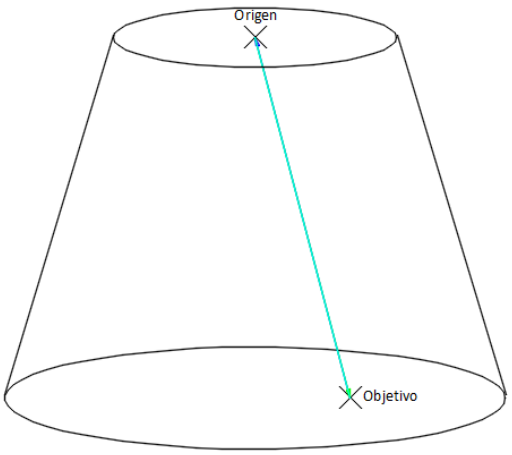

Figura 6.2: Movimiento desde el origen del plano de reposo

En la Figura 6.2 se representa este tipo de trayectoria. Las flechas representan la trayectoria

hacia el objetivo (verde) y la de regreso al origen (azul).

6.3.2 Movimiento desde el plano de reposo con trayectoria vertical

Mientras el manipulador se encuentre en la fase de reposo, este buscará moverse al punto del plano de reposo que se encuentre encima de la posición objetivo. Cuando se cumpla la condición para que el brazo se mueva hacia la posición objetivo, éste se realiza siguiendo una trayectoria vertical.

Esta trayectoria facilita el accionamiento de la pinza gracias a que el movimiento es completamente vertical. Sin embargo, se limita el espacio de trabajo de partida ya que hay puntos que, aunque estén al alcance del manipulador, los puntos del plano de reposo por encima del mismo no lo están. Es el caso de los puntos fuera del cilindro cuya base superior coincide con la base superior del cono truncado.

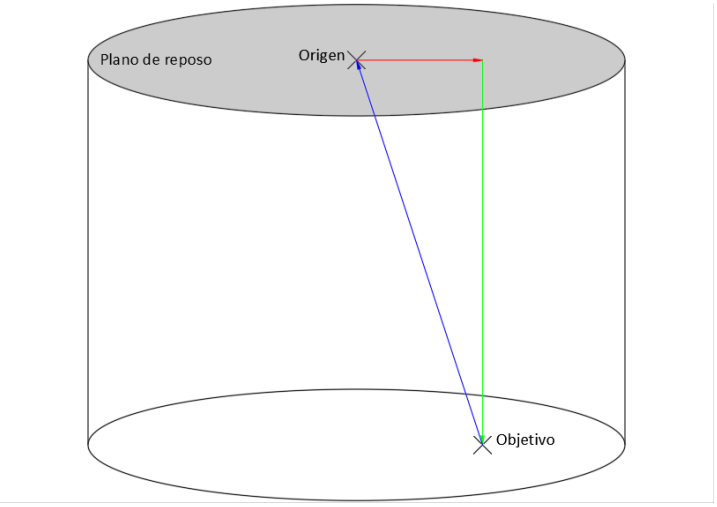

Figura 6.3: Movimiento desde el plano de reposo con trayectoria vertical

En la Figura 6.3 se puede ver el espacio de trabajo que se aprovecha al realizar la trayectoria de esta forma. Se ha dibujado un ejemplo con trayectoria en el plano de reposo (rojo), trayectoria hacia el objetivo (verde) y regreso al origen del plano de reposo (azul).

6.3.3 Movimiento desde el plano de reposo con trayectoria recta

Cuando el manipulador se encuentra en la fase de reposo, este se mueve al punto del plano de reposo que se encuentre encima de la posición del objetivo. En los casos en los que el objetivo está dentro del espacio de trabajo del brazo pero el punto del plano de reposo superior al mismo no lo está, el manipulador se mueve a la posición del espacio de trabajo del plano más cercano. El movimiento desde el plano de reposo hasta la posición objetivo se realizará en línea recta.

Con esta trayectoria se aprovecha completamente el espacio de trabajo que se consideró en la Sección 6.1, y se obtienen trayectorias completamente verticales cuando el objetivo está dentro del cilindro cuya base coincide con la base superior del cono truncado, lo cual hace que este tipo de trayectoria sea estrictamente mejor que el de la Subsección 6.3.2. El ángulo respecto a la vertical aumentará conforme aumente la distancia al cilindro.

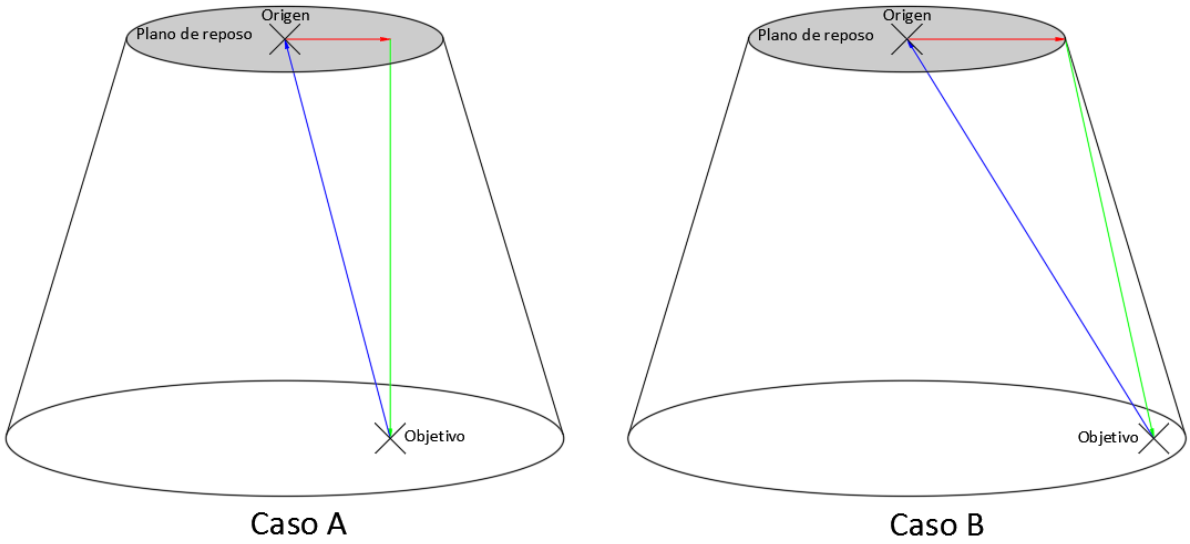

Figura 6.4: Movimiento desde el plano de reposo con trayectoria recta

En la Figura 6.4 se representan dos casos genéricos con este tipo de trayectoria. En el Caso A, la trayectoria hacia el objetivo es completamente vertical, y en el Caso B, el ángulo con el que el manipulador se acerca al objetivo depende de la posición objetivo. Las flechas representan la trayectoria en el plano de reposo (rojo), trayectoria hacia el objetivo (verde) y regreso al origen del plano de reposo (azul).

6.3.4 Movimiento desde el plano de reposo con trayectoria recta seguida de vertical

Esta trayectoria se ha implementado con la intención de forzar que el manipulador llegue al objetivo realizando un movimiento vertical para facilitar el accionamiento de la pinza. Se ha determinado una distancia mínima durante la cual el movimiento es completamente vertical.

Durante la fase de reposo, el manipulador se mueve a la posición del plano de reposo más cercana a la posición objetivo. Cuando se compruebe la estabilidad, se realiza un movimiento en línea recta hacia el punto intermedio que se encuentre a la altura que se haya determinado por encima del punto objetivo, y luego verticalmente desde este punto intermedio hasta el punto final.

Antes de iniciar una trayectoria se comprueba que todos los puntos dentro de la misma se encuentren dentro del espacio de trabajo inicial. Debido al último tramo vertical, los puntos finales a los que se puede acceder por esta trayectoria son los contenidos en la intersección entre el cono truncado que conforma el espacio de trabajo inicial y un cono truncado idéntico desplazado hacia abajo según la distancia vertical mínima que se tome. Cuanto menor sea la distancia vertical mínima, mejor se aprovecha el espacio de trabajo.

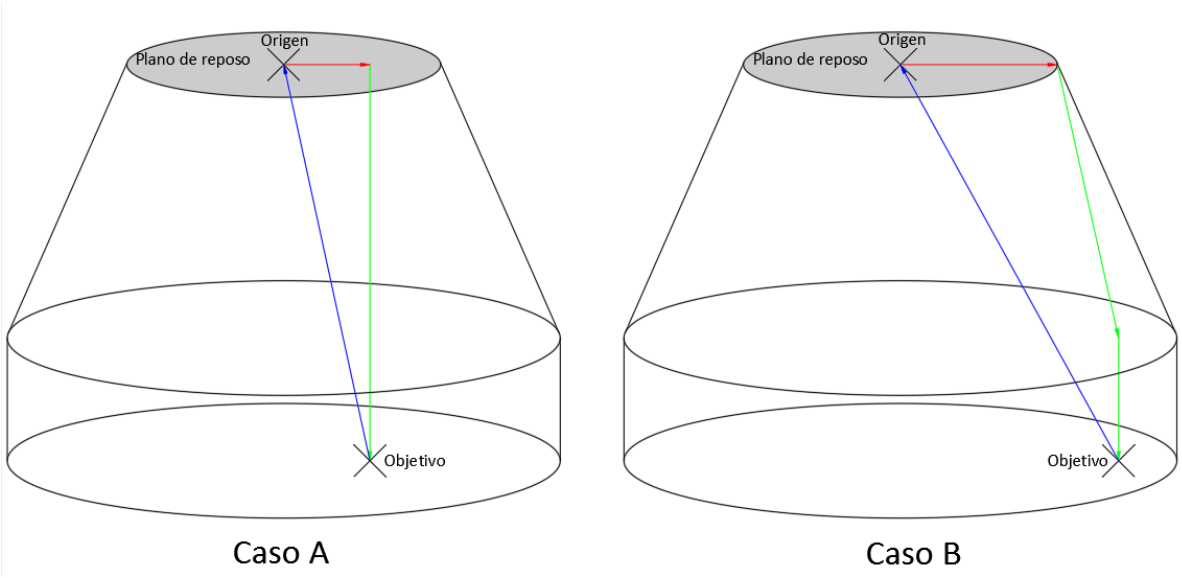

Figura 6.5: Movimiento desde el plano de reposo con trayectoria recta seguida de vertical

En la Figura 6.5 se representan dos casos que se pueden con este tipo de trayectoria. En el Caso A, la trayectoria hacia el objetivo es completamente vertical, y en el Caso B, la trayectoria de bajada tiene un punto intermedio situado a una altura determinada sobre el objetivo. Se han dibujado flechas que representan la trayectoria en el plano de reposo (rojo), trayectoria hacia el objetivo (verde) y regreso al origen del plano de reposo (azul).

6.3.5 Movimiento con ajuste manual de offset

Este nodo es similar al nodo del que se habla en la Subsección 6.3.3, pero muestra en pantalla 3 barras (Figura 6.6), añadidas con las funciones `createTrackbar()` y `getTrackbarPos()` de OpenCV [23], con las que se puede ajustar el offset (x_0, y_0, z_0) de la posición en la que se encuentra el objetivo. El espacio de trabajo se comprueba después de aplicarse el offset para evitar que se fuerce al brazo a moverse a posiciones no válidas.

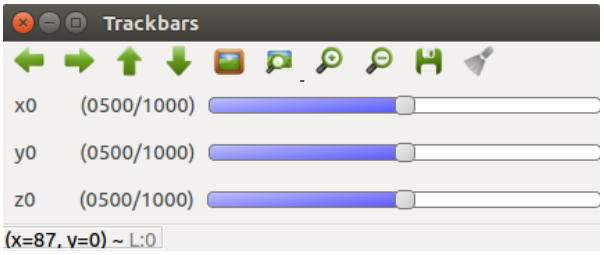

Figura 6.6: Barras para ajustar el offset. Con las barras en la posición que se muestra, se applica como offset $(0,0,0)$

7

EXPERIMENTOS Y RESULTADOS

7.1 Prueba de detección con nube de puntos

Esta prueba se ha realizado para comprobar el funcionamiento del nodo que calcula la posición de un objeto de un determinado color a partir de una imagen RGB y nube de puntos (Subsecciones 4.4 y 4.4.4.1).

Para realizar esta prueba, se ha colocado una pelota de color azul encima de una mesa hacia la que está mirando la cámara. Para distintas posiciones sobre la mesa, se han anotado las coordenadas que calcule el programa y se han comparado con las que se han medido midan físicamente con una cinta métrica. Las medidas con cinta métrica se han realizado respecto a la primera posición en la que se coloque la pelota, en la que se considera que la medida que da el programa y la real son la misma.

La altura de la cámara sobre la mesa es de 117cm. A la hora de interpretar los resultados ha de tenerse en cuenta que los ejes en los que se ha medido las coordenadas reales no son perfectamente paralelos a los de la cámara.

En la Tabla 7.1 se comparan la medida física y la calculada por el nodo, y se muestra el error en cada uno de los ejes y el error total para cada posición de la pelota.

Tabla 7.1: Medidas de la prueba de detección con nube de puntos

Posición	Medida cámara (mm)			Medida física (mm)		
	x	y	z	x	y	z
1	113,623	-10,179	-1166	113,623	-10,179	-1166
2	5,311	-6,487	-1167	3,623	-10,179	-1166
3	213,696	188,62	-1172	223,623	189,821	-1166
4	-132,717	152,986	-1021	-116,377	136,821	-1021
5	216,082	-229,375	-1021	260,623	-240,179	-1021
6	-93,898	-18,08	-991	-76,377	-10,179	-926

Tabla 7.2: Errores calculados de los puntos de la Tabla 7.1

Posición	Error (mm)			
	x	y	z	Total
1	0	0	0	0
2	1,688	3,692	-1	4,180934
3	-9,927	-1,201	-6	11,66138
4	-16,34	16,165	0	22,98484
5	-44,541	10,804	0	45,8326
6	-17,521	-7,901	-65	67,78209

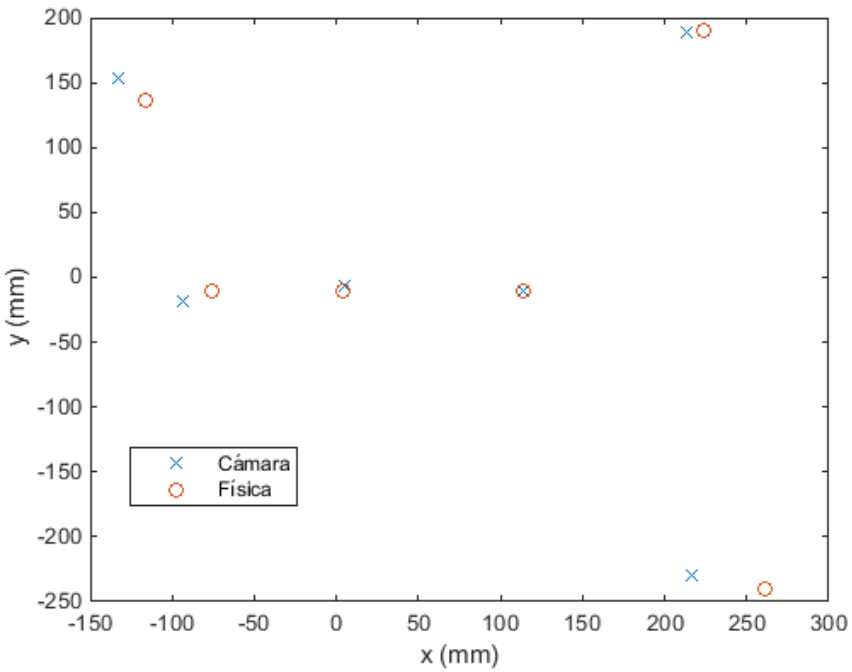

Figura 7.1: Representación en el plano XY de los datos de la Tabla 7.1

7.2 Comparación entre el uso de nube de puntos e imagen de profundidad

En este experimento se toman puntos arbitrarios, en los que se coloca la pelota azul que se ha estado utilizando, para medir la posición de dichos puntos según cada uno de los dos nodos que se han diseñado para determinar la misma (Sección 4.4.4). El cada una de las medidas obtenidas de los nodos se compara con la posición que se ha medido físicamente para evaluar el error que se comete.

Para la medida física se tomando como referencia la primera posición que se ha medido (Posición A), a la cual se le ha asignado el valor medio de las medidas que realiza cada uno de los nodos en dicha posición. el resto de medidas físicas se han calculado a partir de la Posición A procurando seguir los ejes de coordenadas de la cámara.

En la Tabla 7.3 se presentan las medidas realizadas en este experimento. El error cometido al utilizar la nube de puntos respecto a la medida física se muestra en la Tabla 7.4, y el error cometido al utilizar la imagen de profundidad respecto a la medida física, en la Tabla 7.5.

Tabla 7.3: Coordenadas de varios puntos según la medida física y la realizada con el nodo de localización (Capítulo 4) con la proyección inversa a partir de la nube de puntos y del mapa de profundidad

Posición	Medida física			Nube de puntos			Profundidad		
	x	y	z	x	y	z	x	y	z
A	6,5	-15,75	1039,5	3,3	-12,6	1039	9,7	-18,9	1040
B	-125,75	-45,75	1039,5	-85,5	-41,8	1019	-101,4	-43,5	1037
C	334,25	-295,75	1039,5	353,6	-308,2	1064	368,1	-305	1060
D	-245,75	-130,75	904,5	-242,6	-135	895	-224,1	-158	902
E	164,25	-215,75	777,5	297,8	-317,6	1061	278	-315,9	1074
F	-265,75	176,75	909,5	-287,8	143,1	901	-267,4	165,7	897

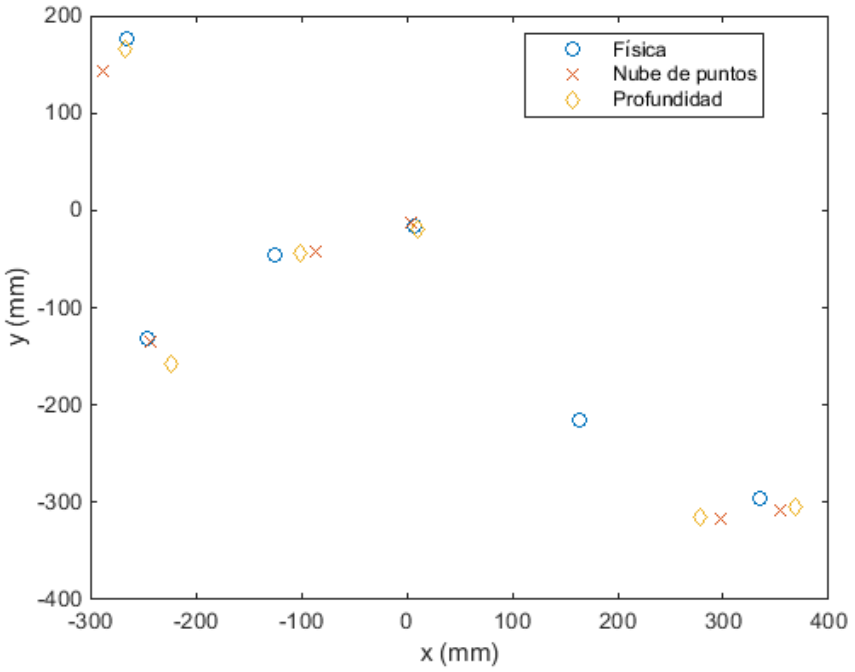

Figura 7.2: Representación en el plano XY de los puntos de la Tabla 7.3

Tabla 7.4: Error nube de puntos. *Ignorando la posición E

Posición	Error nube de puntos (mm)				
	x	y	z	Horizontal	Total
A	-3,2	3,15	-0,5	4,490267	4,518019
B	40,25	3,95	-20,5	40,44336	45,3422
C	19,35	-12,45	24,5	23,00924	33,61064
D	3,15	-4,25	-9,5	5,290085	10,87359
E	133,55	-101,85	283,5	167,9554	329,5167
F	-22,05	-33,65	-8,5	40,2309	41,11903
Media:	28,50833	-24,1833	44,83333	46,90321	77,4967
Media*:	7,5	-8,65	-2,9	22,69277	27,0927

Tabla 7.5: Error imagen de profundidad. *Ignorando la posición E

Posición	Error profundidad (mm)				
	x	y	z	Horizontal	Total
A	3,2	-3,15	0,5	4,490267	4,518019
B	24,35	2,25	-2,5	24,45373	24,58119
C	33,85	-9,25	20,5	35,0911	40,64031
D	21,65	-27,25	-2,5	34,80352	34,89319
E	113,75	-100,15	296,5	151,5556	332,9885
F	-1,65	-11,05	-12,5	11,17251	16,76529
Media:	32,525	-24,7667	50	43,59445	75,73108
Media*:	16,28	-9,69	0,7	22,00223	24,2796

En las Tablas 7.4 y 7.5 se observa que en el punto E el error es significativamente mayor que en el resto de puntos, probablemente porque la pelota se encontrara en una posición en la que la distancia a la cámara no se pudiera determinar correctamente. Por este motivo se ha decidido repetir el cálculo del error medio sin tener en cuenta este punto. La magnitud del error que se comete con los dos nodos es similar. Se señalan como principales causas de este error a las aproximaciones realizadas al realizar la proyección inversa (Subsección 4.4.4) y a limitaciones del hardware de medida de distancia.

7.3 Relación entre el sistema de referencia de la cámara y el del brazo

En esta sección se explica el procedimiento con el que se han tomado los puntos que se utilizan para relacionar los sistemas de referencia de la cámara y del brazo (Sección 5.2). Las posiciones según el sistema de referencia de la cámara se han medido con el nodo de localización que utiliza la imagen de profundidad (Sección 4.4) y una pelota azul.

7.3.1 Coordenadas X e Y

Se ha situado la pelota en distintas posiciones a la misma altura (misma coordenada Z), dentro del espacio de trabajo del brazo y del campo de visión de la cámara. Las posiciones han sido escogidas arbitrariamente y relativamente alejadas entre sí. Para cada posición de la pelota, se toman las coordenadas X e Y que publica el nodo de localización, para posteriormente mover el brazo de forma manual a la misma posición física en la que se encuentra la pelota y tomar las coordenadas X e Y que se publica en el tópico correspondiente a la posición actual del manipulador en coordenadas cartesianas. En la Tabla 7.6 se muestran los datos tomados.

Tabla 7.6: Datos de las coordenadas X e Y según la cámara y el brazo, después de haberse realizado el cambio de sistema de referencia

	Cámara (mm)		Brazo (mm)	
Punto	x	y	x	y
P1	125,02	124,34	237,56	-95,34
P2	-226,27	206,97	138,6	230,76
P3	-205,68	-125,78	-131,5	43,6
P4	95,1	-219,36	-75,45	-223,65
P5	3,55	44,79	91,48	5
P6	267,29	235,46	273,18	-134,5
P7	-54,74	-231,41	-168,32	-118,32
P8	-401,34	135,06	2,88	288,06

7.3.2 Coordenada Z

Se sigue el mismo método que para las coordenadas X e Y, pero esta vez se sitúa la pelota a diferentes alturas y se toma la coordenada Z de cada punto. En la Tabla 7.7 se muestran los datos tomados.

Tabla 7.7: Datos de la coordenada Z

	Cámara (mm)	Brazo (mm)
Z1	1034	-923,67
Z2	898	-776,88
Z3	769	-645,09
Z4	908	-784,82
Z5	967	-873,93
Z6	827	-719,9

7.4 Error cometido con el cambio de sistema de referencia

Después de calcular la relación entre las coordenadas de la cámara y las del brazo (Sección 5.2) se procede a comparar las coordenadas que se obtienen con ayuda de la cámara después de realizar la transformación y las coordenadas que se publican en el tópico correspondiente a la posición del manipulador: se sigue el mismo método que en la Sección 7.3, pero a la posición que facilita la cámara está expresada en el sistema de referencia del brazo.

7.4.1 Error a lo largo del eje X

Se han tomado varias medidas a lo largo del eje X con la intención de encontrar la relación entre la coordenada X y el error que se comete en esta misma coordenada. En la Tabla 7.8 se presentan las medidas realizadas.

Tabla 7.8: Errores en la coordenada X

Cámara (mm)		Brazo (mm)		Error (mm)
x	y	x	y	x
-179,05	-4,1	-179,79	13,82	-0,74
-150,11	4,98	-141,6	4,75	8,51
-127,79	2,35	-121,27	25,3	6,52
-99,44	-1,68	-99,74	1,43	-0,3
-53,38	2,74	-63,58	20,7	-10,2
-36,73	-0,39	-40,84	7,89	-4,11
3,8	-7,99	-5,08	10,6	-8,88
30,78	0,63	28,56	0,13	-2,22
64,08	-2,7	52,75	1,53	-11,33
93,93	3,22	82,46	8,14	-11,47
124,02	3,31	115,4	15,94	-8,62
150,82	-4,44	144,37	13,42	-6,45
184	-0,2	181,46	5,42	-2,54
215,16	0,73	205,79	12,4	-9,37
247,47	-1,4	227,45	6,31	-20,02
280,34	2,24	256,37	18,97	-23,97
306,01	-4,19	271,69	12,36	-34,32

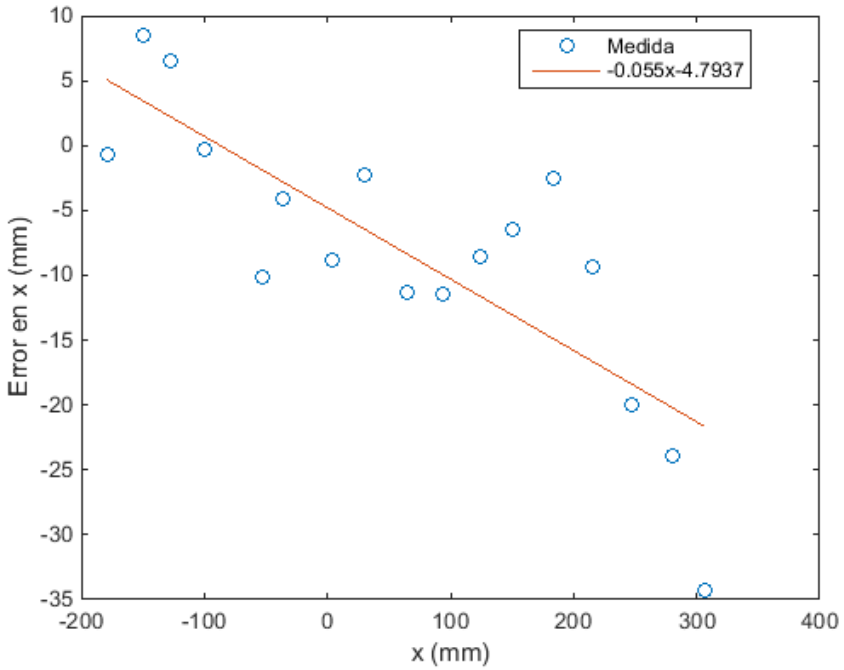

Figura 7.3: Errores en la coordenada X y recta de regresión correspondiente

En la Figura 7.3 se representa el error cometido en la medida de X en función de esta misma

medida, así como la recta de regresión lineal que se ha calculado: $\text{Err}(x) = -0.055x - 4.7937$.

7.4.2 Error a lo largo del eje Y

Al igual que en el eje X, se han tomado varias medidas a lo largo del eje Y para hallar la relación entre la coordenada Y y el error que se comete en esta coordenada. Las medidas realizadas se muestran la Tabla 7.9.

Tabla 7.9: Errores en la coordenada Y

Cámara (mm)		Brazo (mm)		Error (mm)
x	y	x	y	y
1,71	-292,62	-21,49	-279,08	13,54
4,11	-269,87	2,81	-256,11	13,76
5,45	-245,91	17,59	-253,32	-7,41
-6,6	-221,99	-8,84	-210,93	11,06
3,9	-182,67	-11,56	-154,75	27,92
0,67	-136,88	-14,13	-96,27	40,61
-0,75	-135,4	2,42	-127,21	8,19
-0,64	-88,37	-6,16	-71,62	16,75
5,05	-60,92	0,75	-37,89	23,03
-3,92	-20,36	-16,87	-2,21	18,15
-4,35	-20,06	-8,07	-2,98	17,08
3,8	-7,99	-5,08	10,6	18,59
-7,06	59,23	-8,35	76,83	17,6
1,89	85,9	4,71	125,47	39,57
-1,39	130,76	10,68	138,89	8,13
-1,28	150,17	-5,93	156,94	6,77
-3,61	188	5,75	196	8
3,02	219,81	6,83	228,119	8,309
-1,5	238,97	1,82	250,12	11,15
-3,84	285,44	26,93	254,84	-30,6

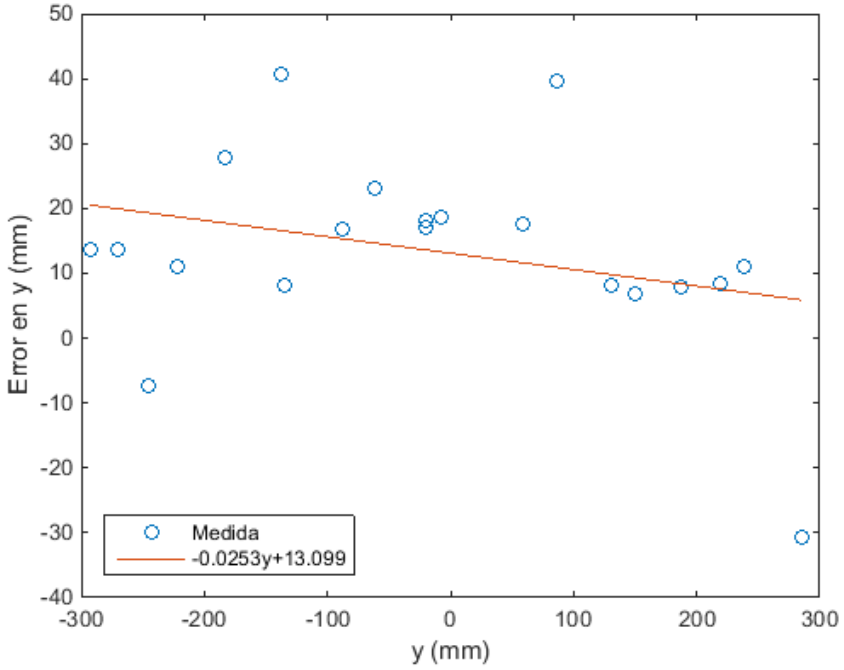

Figura 7.4: Errores en la coordenada Y y recta de regresión correspondiente

En la Figura 7.4 está representado el error cometido en la medida de Y en función de esta misma medida, así como la recta de regresión lineal que se ha calculado: $\text{Err}(y) = -0.0253y + 13.099$.

7.4.3 Rectificación del error

Las líneas de tendencia de las Figuras 7.3 y 7.4, con $R^2 = 0.6492$ y $R^2 = 0.0909$, respectivamente; no son una aproximación precisa del error. Sin embargo, se pueden utilizar para reducir en alguna medida el error que se comete al cambiar de sistema de referencia. Si se diera el caso de que los parámetros de la matriz de transformación (Ecuación 5.1) no se hubieran calculado de forma correcta, las líneas de tendencia permiten corregir los parámetros θ, x_{b0} y y_{b0}.

Se parte de la premisa de que $\theta = -59.75736^{\text{o}}$ es erróneo, en cuyo caso el término lineal cada una de las líneas de tendencia supone la corrección de $\cos\theta$ y del $\sin\theta$, respectivamente. Se obtienen los siguientes resultados:

$$\cos\theta = \cos(-59.75736) + (-0.055) = 0.448663 \rightarrow \theta = -63.3421$$

$$\sin\theta = \sin(-59.75736) + (-0.0253) = -0.8892 \rightarrow \theta = -62.7729$$

El resultado obtenido a partir de ambas correcciones es similar, así que se considera válido. Se toma como nuevo valor la media de ambos: $\theta = -63.0575^{\text{o}}$.

Los términos independientes de las líneas de tendencia pueden ser directamente sumados a 69.47914mm y −70.2578mm, valores anteriores de los parámetros x_{b0} y y_{b0}, para corregirlos:

$$x_{b0} = 69.47914 + (-4.7937) = 64.68544$$

$$y_{b0} = -70.2578 + 13.099 = -57.1588$$

Aunque de esta forma no se reduce significativamente el error, se utilizan los nuevos valores ($\theta = -63.0575^{o}$, $x_{b0} == 64.68544$mm, $y_{b0} = -57.1588$mm) por ser una mejor estimación de los parámetros de la transformación que los valores que se calcularon en la Sección 5.2. Sin embargo, el error viene dado principalmente por los motivos que se discuten en la Sección 7.2.

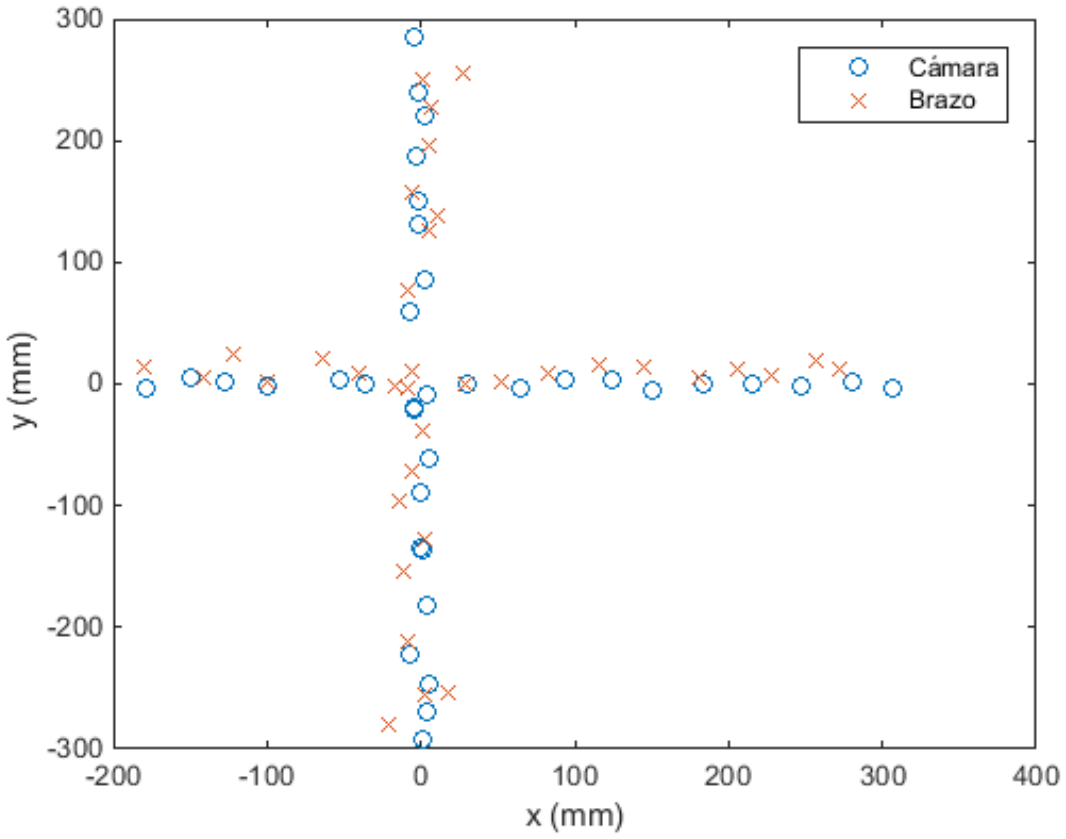

Figura 7.5: Representación en el plano XY de los puntos de la Tabla 7.6 según la cámara y según el brazo, después de haberse realizado el cambio de sistema de referencia

7.5 Estudio del error en el mapa de profundidad

Este experimento se ha realizado para estudiar la precisión de la cámara utilizada a la hora de determinar distancias. Para ello se ha situado la cámara mirando en dirección perpendicular a

una superficie plana (la cámara se ha orientado hacia el techo, sin obstáculos en su campo de visión), a diferentes distancias de dicha superficie en cada medida. Se ha creado un nodo de ROS que calcule el promedio de profundidades en el mapa de profundidad, así como el promedio en las regiones del mapa de profundidad según la distancia en píxeles al centro de la imagen (Figura 7.6). Los resultados de la medida de la distancia a la pared se muestran en la Tabla 7.10, y el error en cada medida en la Tabla 7.11.

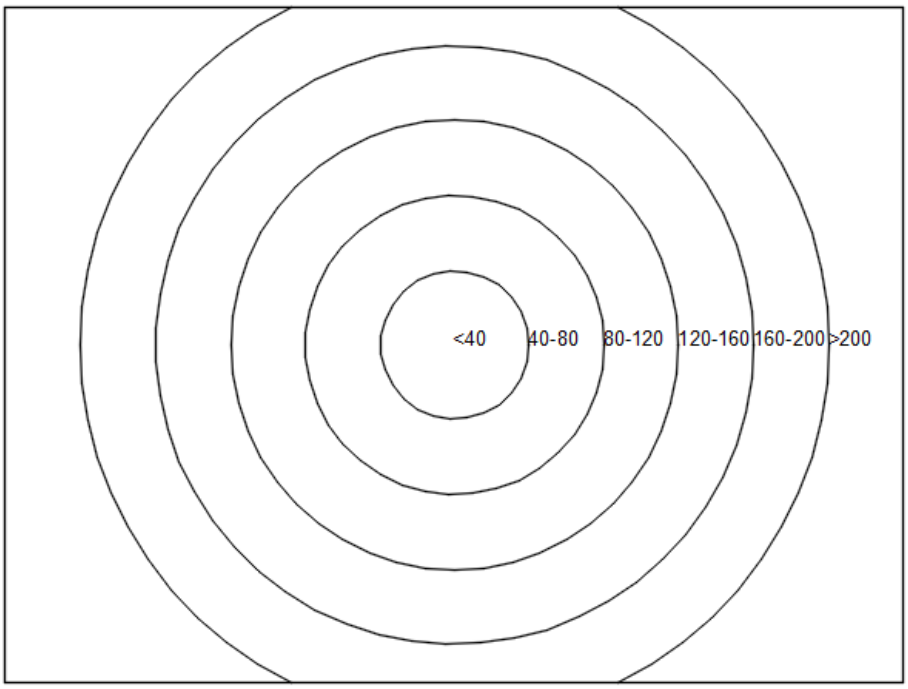

Figura 7.6: División de la imagen de profundidad en regiones según la distancia en píxeles al centro de la imagen

Tabla 7.10: Profundidad promedio total y en cada región de la imagen de profundidad

Real (mm)	Calculada (mm)						
	Total	<40	40-80	80-120	120-160	160-200	>200
1575	1585	1585	1585	1587	1588	1586	1579
1245	1242	1242	1243	1243	1243	1242	1239
1057	1057	1059	1057	1057	1058	1057	1055
840	839	840	839	839	840	839	837
647	647	649	648	648	648	647	646

Tabla 7.11: Error promedio total y en cada región de la imagen de profundidad

Real (mm)	Errores (mm)						
	Total	<40	40-80	80-120	120-160	160-200	>200
1575	10	10	10	12	13	11	4
1245	-3	-3	-2	-2	-2	-3	-6
1057	0	2	0	0	1	0	-2
840	-1	0	-1	-1	0	-1	-3
647	0	2	1	1	1	0	-1

Se observa que el error es despreciable, especialmente en distancias cortas y cerca del centro de la imagen. Debe tenerse en cuenta que, aunque no se cometa error, sí hay puntos en los cuales la distancia no se ha podido determinar (fenómeno que se comenta en la Subsección 4.4.4.2). Se señala la existencia de grandes áreas en las que no se ha podido definir la profundidad como principal causante del error que se comete en el resto de experimentos realizados.

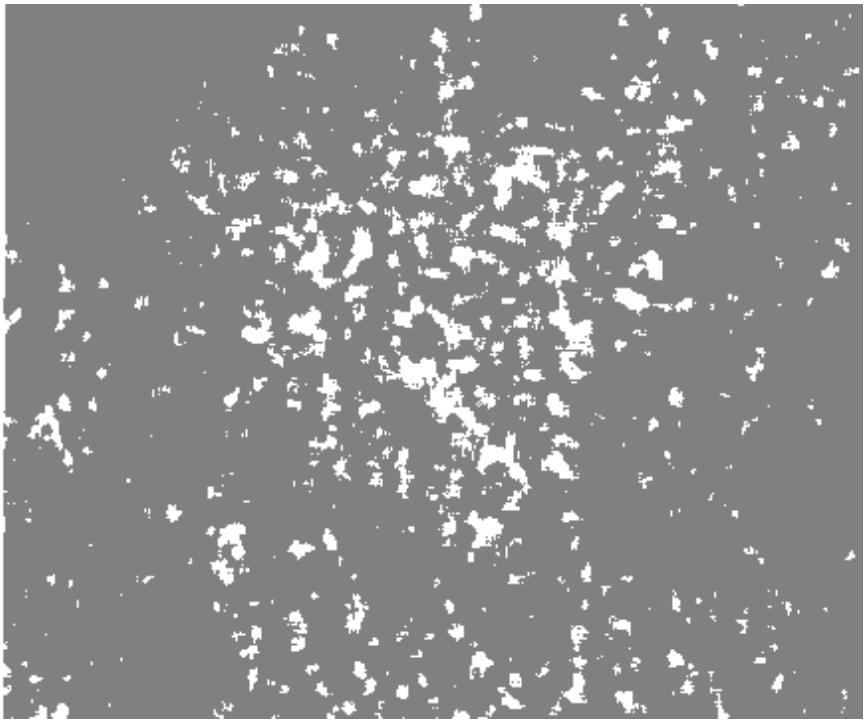

Figura 7.7: Visualización del techo a 840mm en la imagen de profundidad. Los píxeles blancos son aquellos en los que la profundidad no está definida

8

Conclusiones

En este capítulo se discuten las conclusiones que se extraen del trabajo realizado. En la Sección 8.1 se proponen posibles líneas de trabajo a seguir para dar continuidad a este proyecto.

La forma de detectar el objeto de interés se basa en que el color que se utilice se encuentre dentro de un determinado margen (Sección 4.3). Esto limita las condiciones en las que se puede utilizar el nodo de localización de forma efectiva, siendo necesario que el color del objeto a localizar sea distinto a otros objetos en el campo de visión de la cámara, y que la iluminación sea estable.

Para determinar la posición real del objeto de interés, es necesario conocer la distancia del objeto a la cámara. Se han propuesto dos acercamientos para hallar esta distancia (Subsección 4.4.4): uno de ellos selecciona en la nube de puntos el punto que se corresponde con el punto de interés evaluando cada uno de los puntos, y el otro localiza el píxel de la imagen de profundidad correspondiente a partir de una proyección aproximada sobre esta imagen. El algoritmo que utiliza la imagen de profundidad resulta ser significativamente más rápido por no implicar un proceso iterativo.

Como ya se comentó en la Sección 6.3, existen determinadas posiciones en las que el brazo impide que la distancia al objetivo pueda calcularse correctamente. La solución por la que se ha optado, que se explica en esa misma sección, permite que sistema funcione en casos como este, pero no de forma consistente.

De todas las trayectorias que se proponen en la Sección 6.3, las que se comentan en las Subsecciones 6.3.1 y 6.3.3 resultan ser las más interesantes en cuanto al aprovechamiento del espacio de trabajo. La trayectoria de la Subsección 6.3.1 es también la que mejor aprovecha el campo de visión de la cámara ya que no se realiza un seguimiento desde el plano de reposo.

Como se ha visto en la Sección 7.5, el fenómeno más problemático a la hora de calcular la

posición de un objeto a partir de la cámara RGB-D es la existencia de zonas en las que la distancia no puede ser determinada.

En resumen, se ha conseguido hacer que el brazo se mueva hacia el objetivo si las condiciones son las adecuadas, pero existe un retardo sensible entre el movimiento del objetivo y el del brazo.

8.1 Líneas de trabajo futuras

Los parámetros intrínsecos de la cámara RGB se han obtenido mediante la calibración de la misma, que se explica en la Sección 4.2. Sin embargo, sería conveniente calibrar completamente la cámara RGB-D con la intención de reducir el error en la localización.

Aunque el manipulador se mueva hacia el objetivo de forma consistente, el movimiento del brazo puede considerarse brusco. Se considera como futura línea de trabajo implementar trayectorias que hagan que el movimiento del brazo sea más fluido.

El error en régimen permanente del robot delta es distinto de cero. Se ha identificado como posible causa el controlador PID de los motores, el cual habría que calibrar adecuadamente. Ha de tenerse en cuenta que la configuración del control PID de los motores está almacenada en la memoria RAM de los mismos, así que después de apagarlos regresa a los valores por defecto. Se propone como línea de trabajo futura ajustar el PID de los motores, así como la implementación de un script que ajuste los parámetros PID de los motores cuando estos se enciendan.

Después de utilizar el manipulador de forma continuada, los motores tienden a recalentarse, en cuyo caso el fabricante recomienda desconectarlos de la alimentación durante un tiempo para que se enfríen. Durante la realización de este trabajo, dicha desconexión se ha realizado desconectando manualmente la clavija que los une a la batería, lo cual resulta ser poco práctico. Se propone como línea de trabajo el diseño de un sistema que facilite esta tarea, utilizando un relé que permita el encendido y el apagado de forma remota.

REFERENCIAS

[1] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[2] Vicenzo Lippiello, Bruno Siciliano, and Luigi Villani. Position-Based Visual Servoing in Industrial Multirobot Cells Using a Hybrid Camera Configuration. *IEEE Transactions on Robotics*, 23(1):73–86, 2007.

[3] Jiawei Zhou and Shahram Payandeh. Visual Tracking of Laparoscopic Instruments. *Journal of Automation and Control Enginering*, 2(3):234–241, 2014.

[4] Fahed Awad and Rufaida Shamroukh. Human Detection by Robotic Urban Search and Rescue Using Image Processing and Neural Networks. *International Journal of Intelligence Science*, 4:39–53, 2014.

[5] Kenjiro Tadakuma, Carl John Salaan, Eri Takane, Yoshito Okada, Kazunori Ohno, and Satoshi Tadokoro. Design of Aerial Manipulator Suitable for a UAV with Two Passive Rotating Hemispherical Shells. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, 2018.

[6] Gowtham Garimella, Matthew Sheckells, Soowon Kim, and Marin Kobilarov. A Framework for Reliable Aerial Manipulation.

[7] Mohamed Aladem and Samir A. Rawashdeh. Lightweight Visual Odometry for Autonomous Mobile Robots. *Sensors*, 18(9), 2018.

[8] Douglas Morrison, Peter Corke, and Jürgen Leitner. Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach. *Robotics: Science and Systems (RSS)*, 2018.

[9] P. Đurović, R. Grbić, and R. Cupec. Visual servoing for low-cost SCARA robots using an RGB-D camera as the only sensor. *Automatika*, 58(4):495–505, 2017.

[10] Peter K. Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman. Automated Tracking and Grasping of a Moving Object with a Robotic Hand-Eye System. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, 1993.

[11] Jesús M. Gómez de Gabriel, Juan M. Gandarias, Francisco J. Pérez-Maldonado, Francisco J. García-Núñez, Emilio J. Fernández-García, and Alfonso J. García-Cerezo. Methods for Autonomous Wristband Placement with a Search-and-Rescue Aerial Manipulator. 2018.

[12] Jose Ruben Sanchez-Lopez, Antonio Marin-Hernandez, and Elvia R. Palacios-Hernandez. Visual Detection, Tracking and Pose Estimation of a Robotic Arm End Effector. In *Proceedings of the ROSSUM*, pages 41–48, 2011.

[13] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. In *Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[14] Jianbo Shi and Carlo Tomasi. Good Features to Track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[15] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[16] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[17] Hebert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[18] Edward Rosten and Tom Drummond. Machine Learning for High-Speed Corner Detection. In *European Conference on Computer Vision*, pages 430–443, 2006.

[19] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, pages 674–679, 1981.

[20] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[21] Jean-Pierre Merlet. Direct kinematics of parallel manipulators. *IEEE Transactions in Robotics and Automation*, 9(6):842–846, 1993.

[22] Changing Camera Parameters from the Default Presets - ROS Wiki (Acceso online el 2/8/18) `http://wiki.ros.org/realsense_camera/Tutorials/change_camera_parameters`.

[23] Gary Bradski and Adiran Kaehler. *Learning OpenCV*. O'Reilly Media, Inc., Gravenstein Highway North, Sebastopol, 2008.

[24] Satoshi Suzuki and Keiichi Abe. Topological Strutural Analysis of Digitalized Binary Images by Border Following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[25] Suolan Liu, Chen Chen, and Nasser Kehtarnavaz. A Computationally Efficient Denoising and Hole-Filling Method for Depth Image Enhancement. In *Proceedings of the SPIE*, 2016.