Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

Diseño de un sistema de visión para la manipulación de objetos con el IRB120

Autor: David Ibáñez Díaz
Tutor/es: Elena López Guillén

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

“Diseño de un sistema de visión para la manipulación de objetos con el IRB120”

David Ibáñez Díaz

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

“Diseño de un sistema de visión para la manipulación de objetos con el IRB120”

Autor: David Ibáñez Díaz

Tutor: Dª Elena López Guillén

TRIBUNAL:

Presidente: D. Jesús Ureña Ureña

Vocal 1: Dª Ana Isabel De Andrés Rubio

Vocal 2: Dª Elena López Guillén

FECHA:.....

Agradecimientos

Me gustaría agradecer a mi familia, por el esfuerzo que ha realizado todos estos años, aguantándome y apoyándome a través de mis errores y logros, al igual que por haberme dado la oportunidad de estudiar para labrarme un futuro.

También a mis amigos, por hacer el viaje más ameno, compartiendo los momentos difíciles y alegres por igual, mostrándome que hasta en los peores momentos uno puede reír.

"¿Pero quién te dijo que la vida fuera justa?"

David Fuentes

Índice

Lista de figuras

Lista de tablas

Resumen

Este proyecto surge con la intención de mostrar el comportamiento real de un sistema de paletizado mediante un brazo robótico, centrándose en la importancia de reconocer la orientación y posición de las cajas dentro del entorno de trabajo como puede ser una cinta transportadora.

El objetivo principal del proyecto es el desarrollo de una aplicación de visión artificial mediante Matlab. Dicho sistema detectará cuándo un nuevo objeto ha entrado en su área de trabajo y analizará la imagen obteniendo los parámetros que permiten manipular los objetos predispuestos. Posteriormente el sistema de visión se comunicará con el brazo robot indicando cómo debe manipular el objeto.

Palabras clave

Visión artificial, IRB120, IRC5, Manipulación de objetos

Abstract

This Project tryes to show how a real palletized system Works, handlings objects bringed by a conveyort belt with the IRB-120 Robot. With this on mind this priorize the importance of the recognition of position and orientation of the boxes inside the work space.

Telling that we can affirm that the main objective of this projec is the development o fan artificial visión system with Malab. This system Will detect the entry of new objects on the working space, then it Will stop the conveyort belt and analize the image obteining the neccessary parameters to handle the object, sending this data to the IRB-120 Robot.

Key Words

Artificial visión, IRB120, IRC5, Object manipuation

Resumen Extendido

La función principal del proyecto es la de crear un sistema de visión artificial para manipular una serie de objetos de distintos colores en función de sus propiedades y de la herramienta incorporada en el brazo robot.

Existen dos tipos de piezas que podrán ser manipuladas por una herramienta cada una: piezas cilíndricas, que serán manipuladas por medio de una ventosa siendo agarradas por succión, y piezas prismáticas rectangulares, que serán manipuladas por medio de un gripper que las agarrará entre sus pinzas atendiendo a su orientación.

Con anterioridad a cualquier operación se llevará a cabo una calibración del sistema de visión, considerándose casi la parte más importante del sistema. Este proceso permitirá al sistema adquirir los datos básicos del entorno de trabajo de forma que pueda analizar correctamente las dimensiones y corregir las distorsiones de la imagen que deberán ser traspasadas a coordenadas del mundo real. Del mismo modo también captará las condiciones de luminosidad.

Dicha calibración, al igual que el sistema en general, está apoyado por una pequeña interfaz gráfica desarrollada en Matlab, la cual permitirá a los usuarios no especializados utilizar el sistema sin necesidad de conocimientos específicos sobre el funcionamiento interno de los algoritmos desarrollados.

Mediante la interfaz también es posible decidir sobre varias opciones sobre el funcionamiento del sistema, tales como marcar con qué herramienta se va a trabajar, y por tanto qué tipo de objetos se va a manipular, decirle al sistema cuándo debe establecer la conexión con el brazo robot o cuándo debe comenzar la operación.

El ordenador donde se ejecute el sistema de visión se comunicará con el armario de control IRC5. De esta forma con el sistema en marcha, toda la toma de decisiones acerca de los movimientos del brazo robot se llevarán a cabo mediante los algoritmos desarrollados en Matlab.

Las piezas discurrirán por una maqueta de una cinta transportadora hasta que la cámara situada sobre el entorno de trabajo detecte una imagen que el sistema de visión interprete como “detección de un nuevo objeto”. En ese momento la cinta transportadora se parará y la cámara captará una nueva imagen en parado, la cual será analizada en busca del objeto de trabajo, el cual será detectado y analizado para extraer sus parámetros de manipulación. Dichos parámetros serán los que se utilicen para decidir cómo manipular el objeto mediante el brazo robot, agarrándolo y situándolo donde corresponda.

1. Introducción

Tecnología y sociedad

Podemos definir la tecnología como el conjunto de conocimientos y habilidades técnicas que, utilizados de una forma científicamente ordenada, permiten la resolución de determinados objetivos mediante el diseño y creación de diversos bienes y servicios, enfocados principalmente en facilitar los deseos y necesidades de la sociedad. Existen muchas tecnologías y muy diferentes entre ellas, pero todas ellas han tenido un impacto importante dentro de nuestra sociedad.

Entre todas ellas, la robótica se ha convertido en uno de los campos de estudio y aplicación más importantes de los últimos años, siendo utilizada en una gran diversidad de campos de la sociedad, destacando sobre todo en el área de procesos industriales automatizados. No obstante, la robótica industrial presenta un amplio abanico de posibilidades de cara al futuro, haciendo posible la idea de que, en un futuro no muy lejano, se podrán utilizar robots para aplicaciones que en la actualidad se creen imposibles.

La Federación Internacional de la Robótica(IFR) realiza estudios anuales sobre el impacto de la robótica en nuestra sociedad y, según los datos obtenidos, el crecimiento y demanda de los robots industriales en los distintos campos de aplicación no sólo ha aumentado a lo largo de los últimos años, si no que se pronostica que continuará en ascenso.

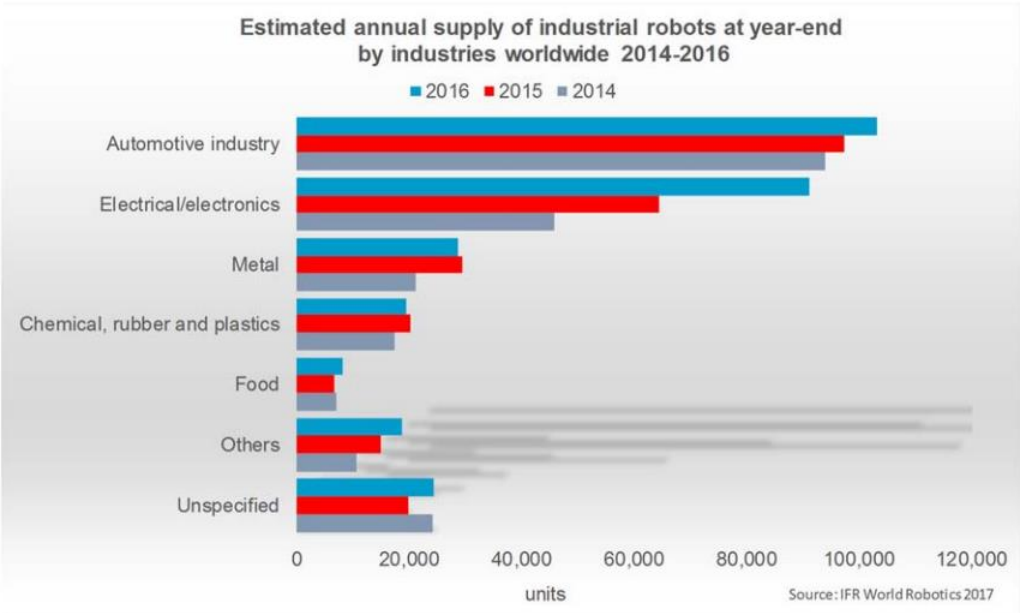

Figura 1.1 Estimación de suministro de robots-industriales por campos

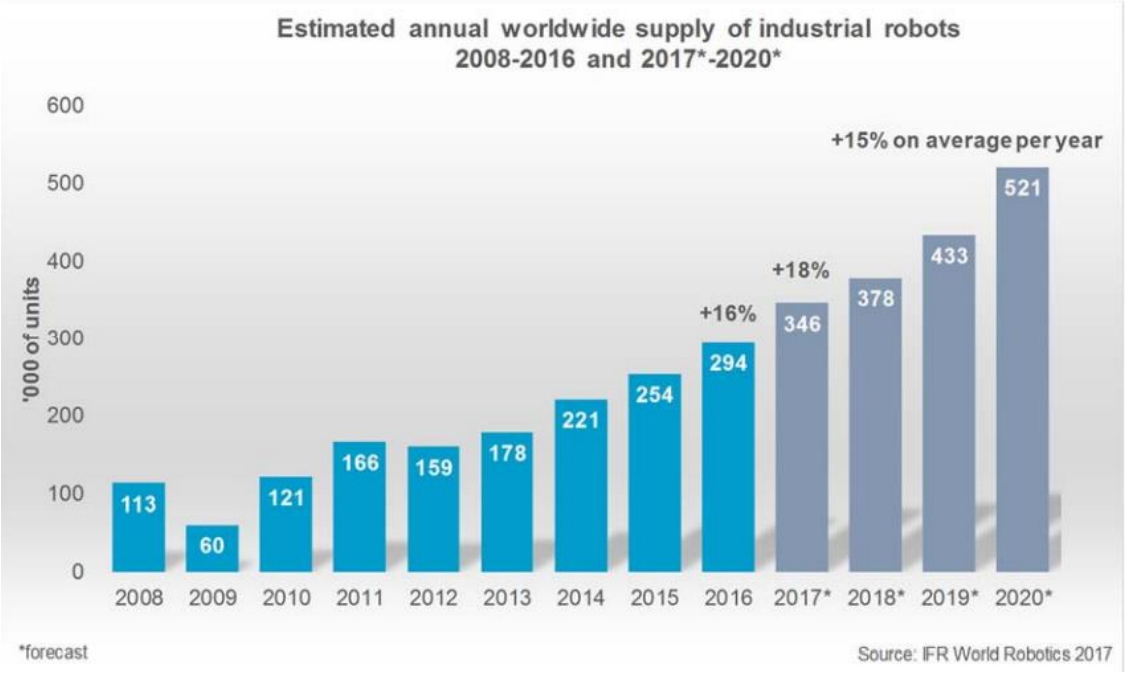

Figura 1.2 Estimación de suministro de robots-industriales para años futuros

La Federación Internacional de la Robótica(IFR) clasifica, siguiendo la UNE-EN ISO 8373, un robot industrial como: un robot de más de tres ejes, reprogramable, multiplicación, ya sea móvil o no, y destinado a aplicaciones de automatización dentro del ámbito industrial.

La gran importancia que ha cobrado dentro de los procesos industriales es debida a las facilidades que aporta a la hora de reproducir sistemas automatizados, en los cuales, el esfuerzo de los seres humanos se ve reducido hasta no más de las tareas de supervisión y control del sistema.

Por otro lado, la visión artificial, es otro de los campos que, con un ritmo irregular, está creciendo poco a poco mientras es utilizado en diversos campos en compañía de otras tecnologías. Por ejemplo, al utilizar un sistema de visión artificial junto con un robot industrial, somos capaces de dotar de cierta inteligencia al robot, haciendo que este sea más autónomo y no precise de tanta supervisión y control por parte del ser humano.

Los sistemas de visión artificial actuales son capaces de reconocer, no sólo formas y colores sencillos, si no también comportamientos o figuras complejas y variadas como puede ser la presencia de un ser humano en la imagen y la acción que está realizando. Esto nos permite, no sólo indicarla a un robot industrial con qué tipo de elementos debe trabajar, si no también cuando debe dejar de actuar por motivos de seguridad para los usuarios que trabajen dentro del área de trabajo del robot.

Este trabajo se centra en la colaboración entre estos dos campos para conseguir obtener un sistema de manipulación de objetos lo más automatizado posible sin depender de la actuación humana más allá de lo estrictamente necesario.

Por ello el proyecto contará con la colaboración de dos elementos de ambos campos que se comunicarán y apoyarán entre ellos formando el sistema final: el brazo robot IRB120 de ABB junto con el armario de control IRC5, y un sistema de visión artificial realizado a través de Matlab diseñado por el alumno.

El objetivo final ha sido el de conseguir que el código diseñado pueda procesar las imágenes captadas por la cámara a tiempo real, detectando cuándo nuevos elementos entran en el área de trabajo definida y detectando si dichos elementos corresponden o no a los elementos que se deben manipular, comunicando en caso afirmativo al armario de control, que movimientos debe realizar el brazo robot para manipularlos, deduciendo dichos movimientos de los parámetros característicos obtenidos de la imagen.

Para el desarrollo del proyecto se han seguido las siguientes pautas en el desarrollo del sistema:

-Estudio de toolbox de Matlab para el procesado de imágenes.

-Desarrollo del sistema de calibración y estudio de parámetros necesarios para la posterior manipulación.

-Desarrollo de programa de detección por formas sobre imágenes estáticas: cuadrados y círculos.

-Desarrollo de programa de detección de colores sobre imágenes estáticas: rojo, verde y azul.

-Unificación de código para la detección por color y forma.

-Desarrollo de programa de detección de centros de objetos y ángulos de giro.

-Desarrollo de flujo de control y comunicación entre el robot y Matlab para transmitir órdenes y movimiento.

-Pruebas sobre el sistema unificado con ayuda de un sensor de detección de movimiento.

-Desarrollo de interfaz gráfica de usuario.

-Desarrollo de un sistema de detección de movimiento para sustituir el sensor de movimiento.

A la hora de estructurar esta memoria nos hemos centrado tanto en conocimientos técnicos como en la correcta comprensión a nivel de usuario, por lo tanto, se ha decidido seguir los siguientes puntos para explicar correctamente el proyecto:

1. Introducción
2. Herramientas Utilizadas
3. Desarrollo del sistema
4. Resultados y Conclusiones
5. Manual de Usuario
6. Planos
7. Presupuesto
8. Bibliografía

2. Herramientas Utilizadas

En este apartado se detallan las herramientas, tanto en apartado físico como software, que han sido utilizadas para desarrollar el proyecto, así como en las distintas pruebas realizadas incluso si no están presentes en el sistema final.

2.1 Matlab (Versión R2013b)

MATLAB es una herramienta de software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. Esta herramienta ofrece una gran cantidad de posibilidades de desarrollo y campos de aplicación:

-Calculo Matricial

-Visión Artificial

-Procesamiento de señales

-Desarrollo de entornos Gráficos

-Análisis Numérico

-Simulaciones de electrónica

-Sistemas de Control

-Robótica

Actualmente Matlab se ha convertido en una herramienta muy utilizada en el campo educativo y de investigación, aumentando su influencia en el mundo de la ingeniería.

El gran abanico de posibilidades de Matlab viene, en parte, de una serie de librerías denominadas ToolBox que presentan diversas funciones para dar soluciones a problemas en distintos campos específicos. Además de ello permite desarrollar funciones propias de forma que se pueden generar soluciones personalizadas o algoritmos únicos para aplicaciones en función de las necesidades del usuario.

Entre las herramientas que Matlab nos aporta nos competen especialmente tres, las cuales se exponen a continuación.

2.1.1 GUIDE

Esta herramienta de Matlab nos permite crear interfaces gráficas de usuario (GUI), permitiendo a usuarios no especializados utilizar de forma simple y sin necesidad de profundizar cualquier aplicación desarrollada mediante Matlab.

El entorno de desarrollo de GUIDE es fácil de entender y utilizar, permitiendo seleccionar e implementar los elementos que el diseñador requiera para desarrollar la interfaz gráfica.

Entre los elementos que permite introducir hay botones, sliders, y cajas para marcado de opciones, todas ellas configurables tanto en aspecto como en funcionalidad mediante las opciones de cada objeto creado.

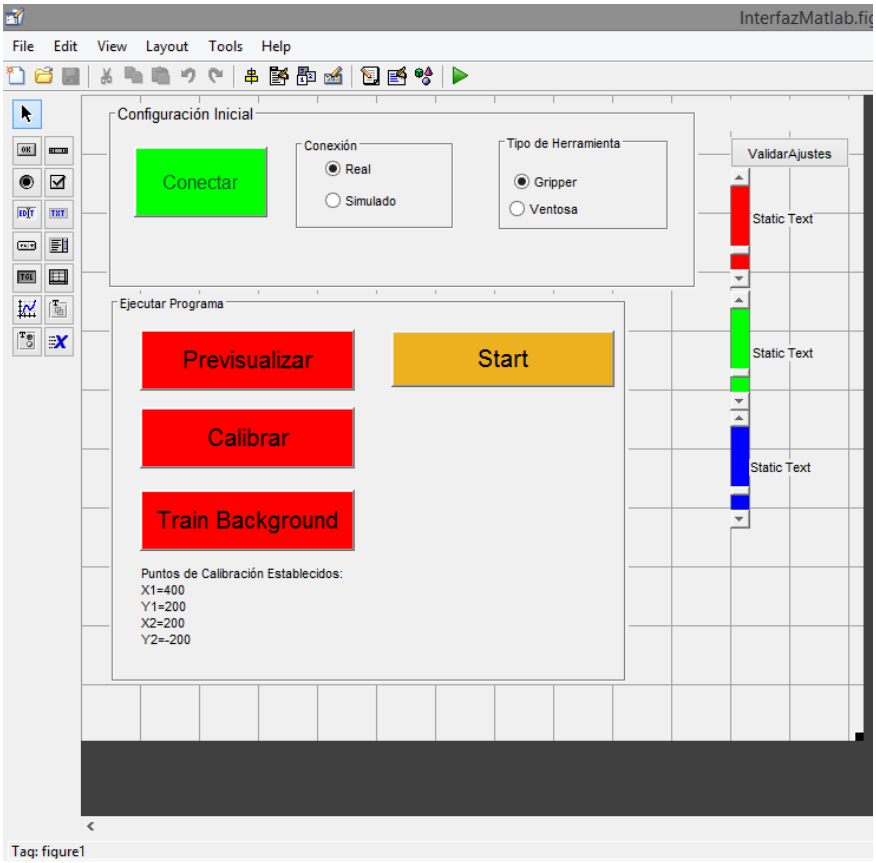

Figura 2.1 Entorno de desarrollo GUIDE

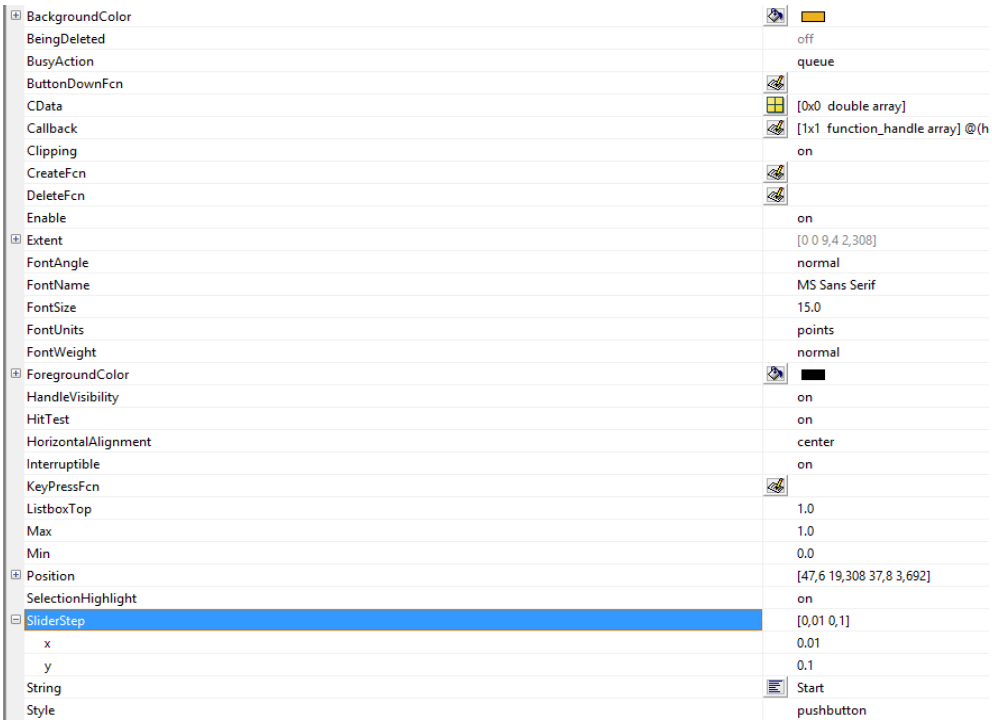

Figura 2.2 Opciones de configuración de un slider de GUIDE

2.1.2 “Image Processing ToolBox” y “Image Acquisition ToolBox”

Se trata de dos librerías de Matlab que nos permiten el acceso a una serie de funciones ya implementadas. Una de ellas nos permite la captación y lectura de imágenes mediante medios externos como cámaras, mientras que la otra, nos da acceso funciones de procesado de imagen simples de forma que podemos utilizarlas como base para generar nuestros algoritmos para sistemas de visión artificial.

La “Image Acquisition ToolBox” aporta funciones de lectura y escritura de imágenes preguardadas, al mismo tiempo que da la posibilidad de leer y previsualizar la imagen captada por una cámara a tiempo real, dando la opción de trabajar con los fotogramas más recientes y en el espacio de color que queramos.

Por otro lado, la “Image Processing ToolBox” aporta funciones de procesado de imágenes, permitiéndonos trabajar sobre las imágenes, aplicando filtros y operaciones morfológicas o modificando el espacio de color a uno que aporte la información deseada. Esta toolbox abarca gran cantidad de funciones, incluyendo funciones de procesado en 3D y de segmentación de imagen.

2.2 Brazo Robótico IRB120

El brazo robótico IRB120 es uno de los muchos que ofrece el catálogo de la compañía ABB. Este brazo presenta una gran capacidad de maniobra y precisión, junto con unas dimensiones más que suficientes para realizar las pruebas deseadas, añadiendo el hecho de que ya se encuentra en las instalaciones de la UAH lo que lo hace la elección óptima para el desarrollo del proyecto.

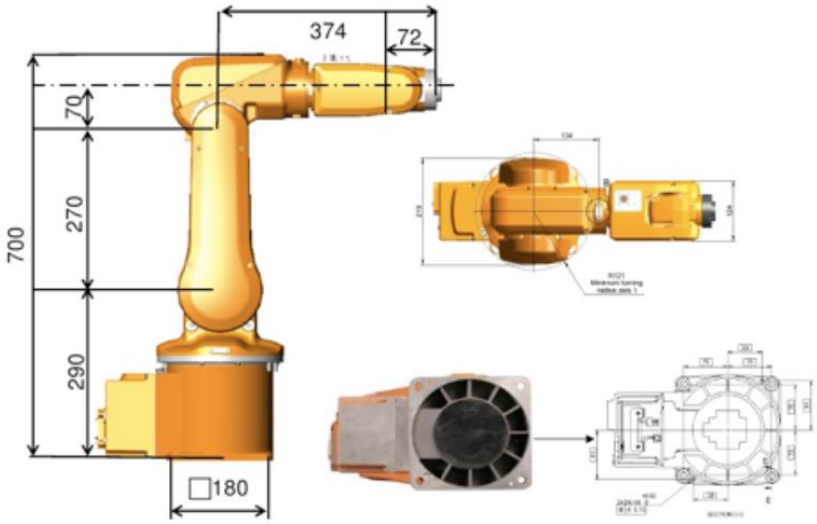

Figura 2.3 Dimensiones brazo robot IRB120

El IRB120 pesa 25 Kg y puede soportar una carga de hasta 3Kg (4Kg si la muñeca se encuentra en posición vertical), con un alcance de 580mm. Se trata del robot industrial multiusos más pequeño que ofrece ABB y alcanza velocidades de hasta 6.2m/s, aunque por motivos de seguridad nuestro proyecto utilizará velocidades más moderadas.

Cuenta con 6 ejes o articulaciones, consiguiendo 6 grados de libertad: 3 de posicionamiento y 3 de orientación (estos 3 últimos también son denominados “muñeca” del robot).

Con estos 6 grados de libertad podemos situar el efector del robot en un mismo punto utilizando diferentes configuraciones de los ejes, encontrando la que se adapte mejor a nuestro entorno de trabajo.

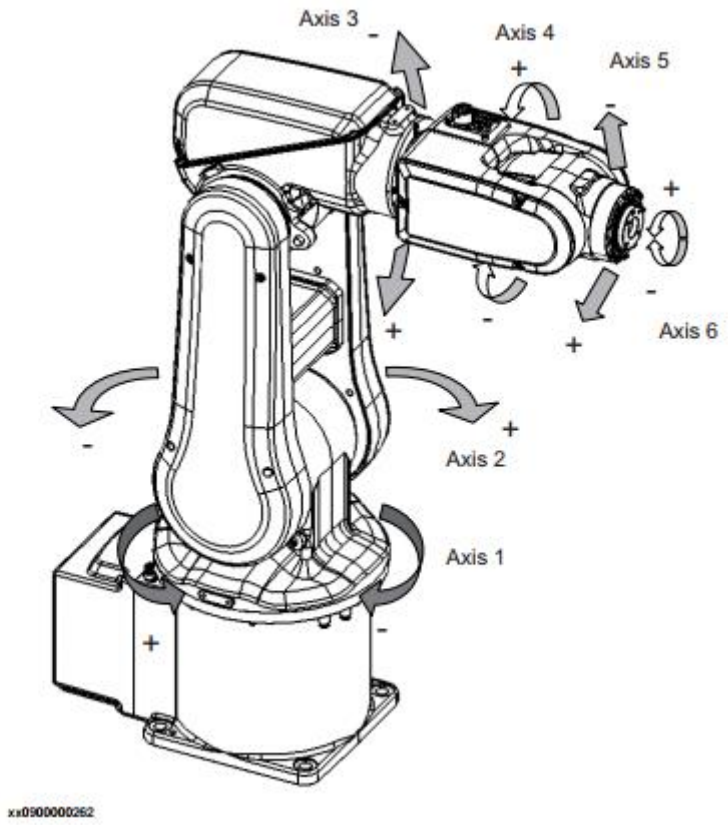

Figura 2.4 Movimientos de los ejes del IRB120 de ABB

Axis	Rango(º)	Velocidad Máxima(º/s)
Axis 1 (Articulación Rotacional)	+165º a -165º	250
Axis 2 (Articulación Rotacional)	+110º a -110º	250
Axis 3 (Articulación Rotacional)	+70º a -110º	250
Axis 4 (Articulación Rotacional)	+160º a -160º	320
Axis 5 (Articulación Rotacional)	+120º a -120º	320
Axis 6 (Articulación Rotacional)	+400º a +400º	420

Tabla 2.1 Rango de giros y velocidades de las articulaciones del IRB120

En la figura se puede observar que, en el extremo del robot, tras el eje número 6, se cuenta con un espacio adaptado para incorporar la herramienta de trabajo. Dicho espacio es de forma circular de 40mm de diámetro, con 4 agujeros separados 31mm entre centros, permitiendo atornillar correctamente la herramienta.

2.3 Herramientas/Efectores del Brazo Robótico

Como hemos mencionado anteriormente el brazo robot cuenta con espacio donde situar diversas herramientas o efectores, en función de las necesidades que presente el trabajo a realizar.

En nuestro proyecto contamos con dos de efectores diferentes en función del tipo de objeto que queramos coger y manipular.

2.3.1 Ventosa

La característica principal de esta herramienta es la de ejercer una succión sobre el objeto a manipular, consiguiendo un agarre firme, preciso y seguro. No obstante, para que esta herramienta sea eficaz, debemos asegurarnos de que los objetos a manipular posean una superficie de contacto lo suficiente grande y lisa.

La ventosa a utilizar será el modelo SMC ZPT32BN-B01, con un diámetro de 35mm y una altura de 165mm. La succión será controlada por la Salida Digital número 9 de armario de control, de forma que cuando este a nivel alto se activará.

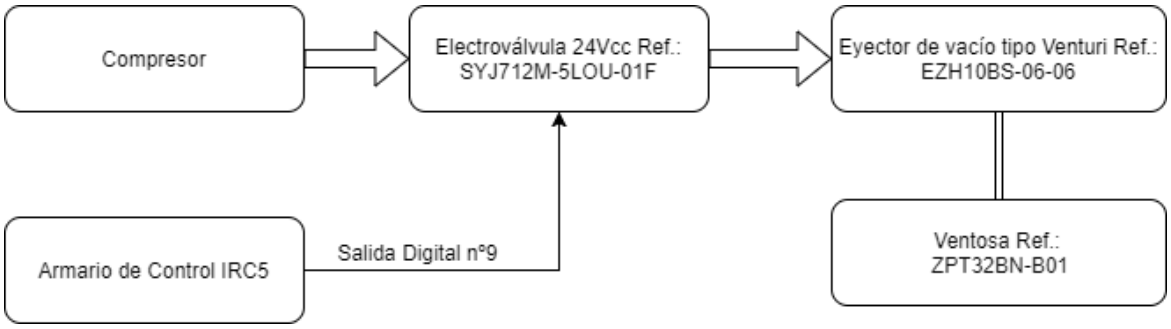

Figura 2.5 Esquema de conexiones ventosa

Figura 2.6 Ventosa modelo SMC ZPT32BN-B01

2.3.2 Gripper

Mediante esta herramienta se agarrarán los objetos a manipular atrapándolos entre las pinzas de la herramienta. Para asegurar el funcionamiento de esta herramienta es necesario que las dimensiones de la pieza a atrapar coincidan con las del Gripper en posición cerrada o sean similares, de otro modo la pieza podría escapar entre las pinzas si el objeto es muy pequeño o ser presionado y deformado si es demasiado grande.

El Gripper a utilizar es el modelo SCHUNK-GBW44, formado por pinzas de forma rectangular de dimensiones 20x15x70 mm, establecidas para que su separación en cerrado sea de unos 30mm. La altura total del Gripper es de 226 mm.

Mediante dos Salidas digitales (E/S 10 y 11) podremos controlar la electroválvula que dará, o no, paso al aire comprimido que realizará el cierre, o apertura, del Gripper.

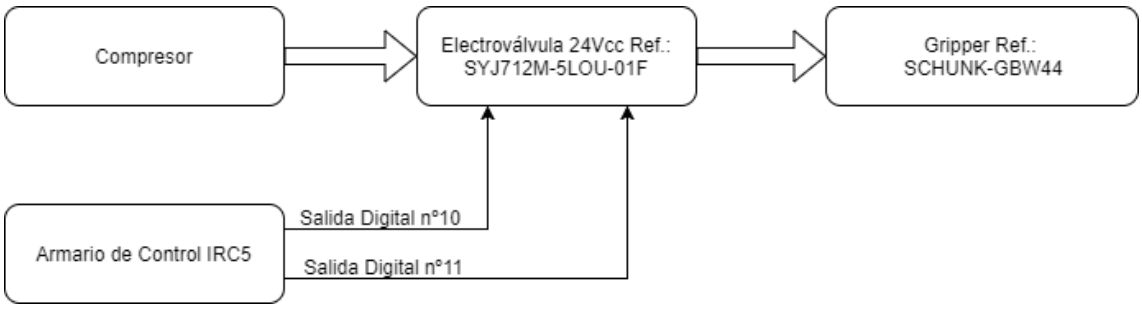

Figura 2.7 Esquema de conexiones gripper

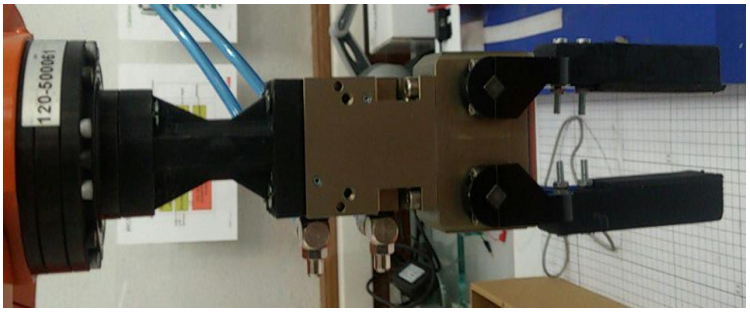

Figura 2.8 Gripper modelo SCHUNK-GBW44

2.4 Armario de control IRC5 y FlexPendant

Se trata de la herramienta principal de control del robot. El IRC5 procesa internamente el programa ejecutable cargado sobre él (el que desarrollaremos sobre RobotStudio), haciéndose cargo de la sincronización con otros elementos, administrar salidas y entradas digitales y controlar las tensiones de alimentación, entre otras.

Características ofrecidas según el catálogo de ABB:

-Puesta en marcha simple
-Portable
-Alimentación Monofásica
-Conectividad Externa
-Posibilidad de controlas 16 Entrada y 16 Salidas de tipo digital
-Una salida analógica de 0 a 10V

La siguiente imagen muestra el esquema de la parte frontal del armario de control:

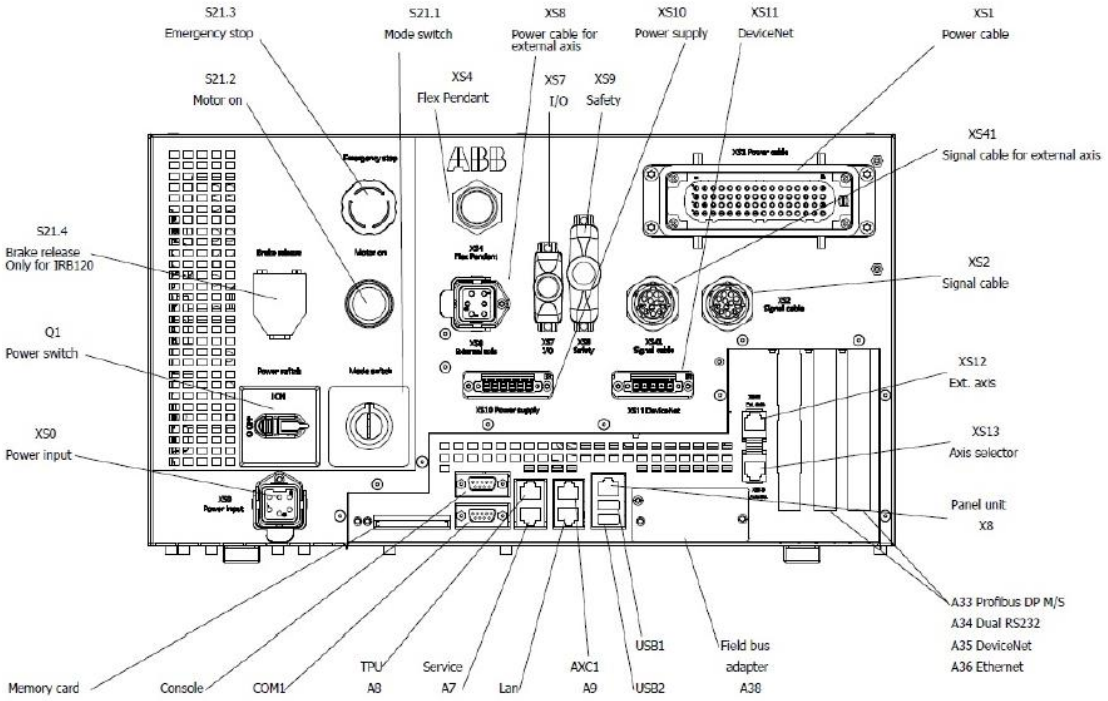

Figura 2.9 Esquema armario de control IRC5

Por otro lado, el IRC5 incluye el denominado FlexPendant, una herramienta qué nos permite comunicarnos de manera intuitiva con el armario de control, así como:

-Crear y cargar nuestros propios programas, así como depurar desde la misma herramienta, observando a su vez los resultados en el sistema real.

-Interrumpir, reanudar y controlar la ejecución según las necesidades del usuario o por motivos de seguridad, pudiendo evitar posibles accidentes o realizar comprobaciones del sistema.

-Controlar de manera manual las Entradas/Salidas del armario, así como manejar de manera individual los ejes del robot utilizando el joystick de la parte frontal.

Figura 2.10 Herramienta FlexPendant

Posición	Descripción
A	Pantalla táctil
B	Teclas programables
C	Botón de paro de emergencia
D	Joystick
E	Teclas de ejecución de programas
F	Conexión de memoria portátil USB
G	Alojamiento de puntero

Tabla 2.2 Tabla de componentes de la herramienta FlexPendant

2.5 Compresor y conexionado de aire comprimido

El compresor del laboratorio suministrará el aire comprimido necesario para efectuar la succión de la ventosa y el agarre del Gripper. El tubo de aire comprimido del compresor desemboca en un cuadro de conexiones desde el cual se gestiona el direccionamiento del aire a las herramientas por medio de las electroválvulas mencionadas en los apartados anteriores.

Figura 2.11 Compresor

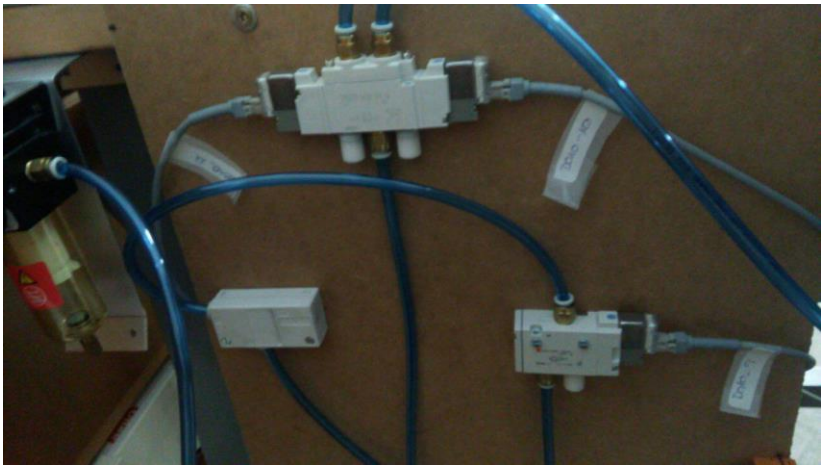

Figura 2.12 Conexionado de aire comprimido y electroválvulas

2.6 Maqueta de cinta transportadora

En el laboratorio podemos encontrar una maqueta de una cinta transportadora con todo el conexionado necesario y elementos adicionales, por lo que formará parte de nuestro entorno de trabajo. Dicha maqueta fue diseñada por el alumno Andrés Senén Estremera en su proyecto "Diseño de una estación de trabajo para el Robot IRB120. Control de cinta transportadora mediante IRC5" [1].

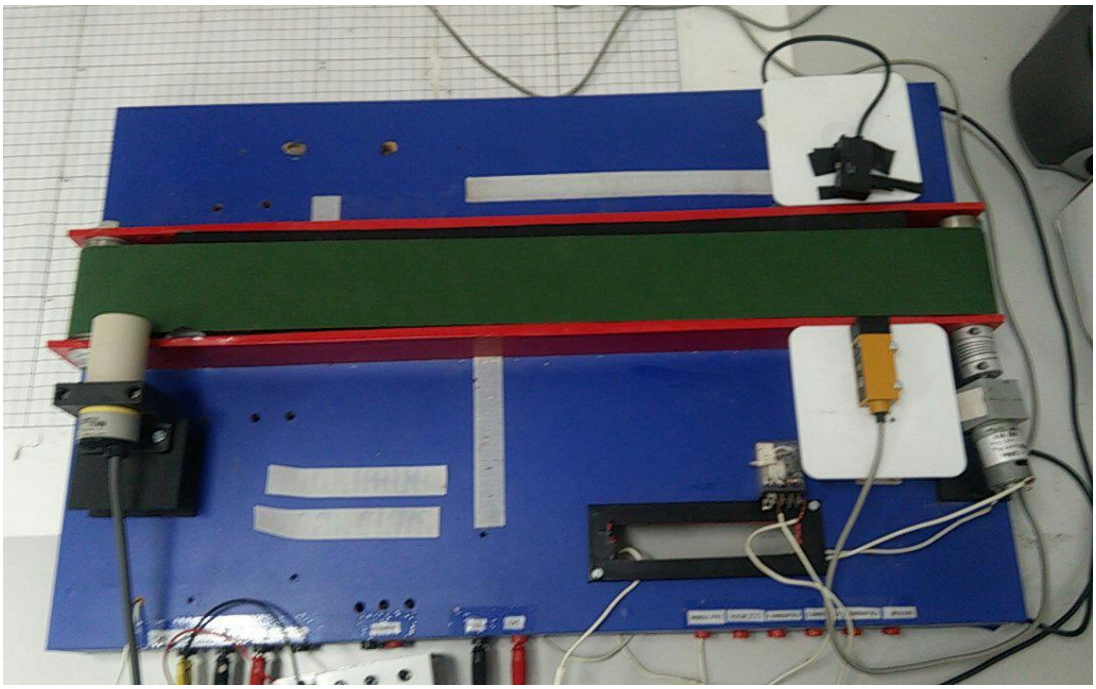

Figura 2.13 Maqueta de cinta transportadora

La maqueta cuenta con los siguientes componentes:

- Tambor que actúa como eje tractor con un diámetro de 35mm y una longitud de 70mm.
- Rodillo que actúa como eje de transmisión del movimiento.
- Rodamientos para unir el tambor a la estructura de 19mm de diámetro y 8mm de ancho.
- Varias piezas de metacrilato unidas entre sí con cola de contacto.
- Un sistema de acoplamiento que permite transmitir la tracción del motor al tambor de la
cinta transportadora.
- Una correa de transmisión de 6.5 cm de ancho y 134 cm de largo.
- Motor K14 2841 con una reductora acoplada que funciona a 24 voltios con una velocidad
fija a esos 24 voltios de 38rpm.

Además, cuenta con dos sensores fotoeléctricos y uno capacitivo, situados en soportes sobre la maqueta, no obstante, para el desarrollo de este proyecto no poseen importancia puesto que no serán utilizados.

Para controlar el movimiento de la cinta se utilizará la salida número 15 del armario de control, provocando que cuando se ponga a '1' aporte la tensión que iniciará su funcionamiento. De manera inversa situar un '0' cesará el funcionamiento de la cinta.

El sensor capacitivo situado al final de la cinta trasportadora se ha utilizado únicamente durante la primera fase del proyecto, no obstante, cabe mencionar su funcionamiento: cuando el sensor detecta la presencia de un objeto en su proximidad envía un nivel alto a través de la entrada numero 14 al armario de control, informando de la presencia de un objeto, y un nivel bajo cuando no hay nada en su proximidad.

2.7 WebCam

Como herramienta de necesidad para construir el sistema de visión artificial que es nuestro proyecto, es obligatorio una forma de captación de imágenes en tiempo real. La encargada de ello será la cámara Web de la marca Logitech C-310, ya presente en el laboratorio y que se situará sobre el espacio de trabajo.

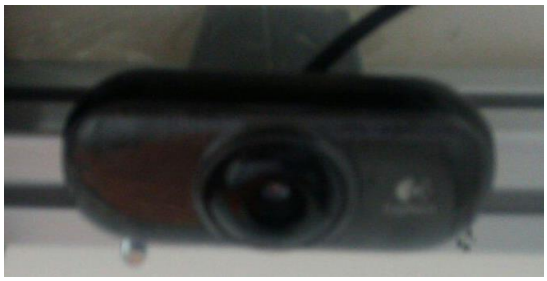

Figura 2.14 WebCam situada en la parte superior del área de trabajo

2.8 Objetos a manipular

El objetivo de este proyecto es el de utilizar un sistema de visión para manipular una serie de piezas mediante las herramientas mencionadas, por ello mencionamos aquí las características de las piezas a manipular:

-Piezas cilíndricas: poseen 60mm de diámetro y 15mm de altura, y en cada una de sus caras se encuentra una pegatina de color, las cuales pueden ser rojas, verdes o azules. Estas piezas serán el objeto de manipulación mediante la ventosa.

Figura 2.15 Piezas cilíndricas

-Piezas prismáticas: poseen una altura de 60mm y una base cuadrada de 30mm de lado, al igual que las cilíndricas en sus bases poseen una pegatina de color roja, verde o azul. Estas piezas serán el objeto de manipulación mediante el Gripper.

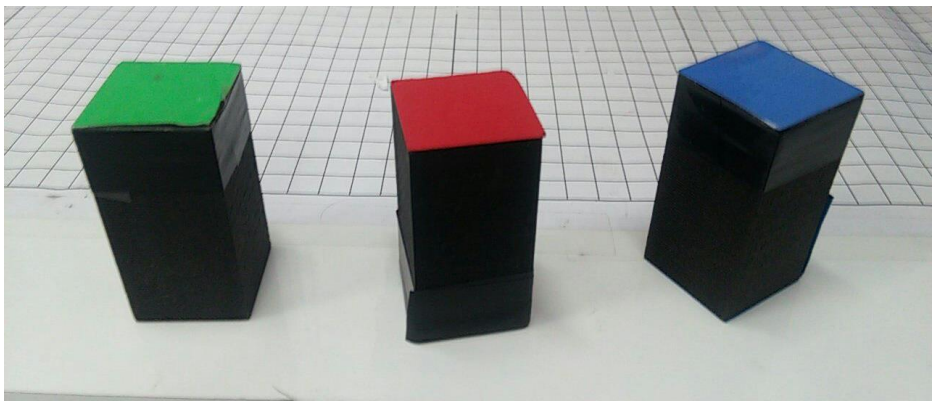

Figura 2.16 Piezas prismáticas

2.9 RobotStudio

RobotStudio es el software de desarrollo de ABB para programar sus sistemas robotizados, así como simular distintas estaciones y entornos de trabajo para un estudio anterior a la prueba en sistemas reales, reduciendo los riesgos por errores de programación. También permite la formación de nuevos desarrolladores sin tener que disponer de un sistema real de pruebas.

Puesto que dicho software se basa en el controlador virtual de la compañía ABB, siendo una copia exacta de los controladores que utilizan sus brazos robóticos, nos asegura simulaciones realistas, con la posibilidad de introducir los parámetros reales del robot sobre el que se va a usar el programa.

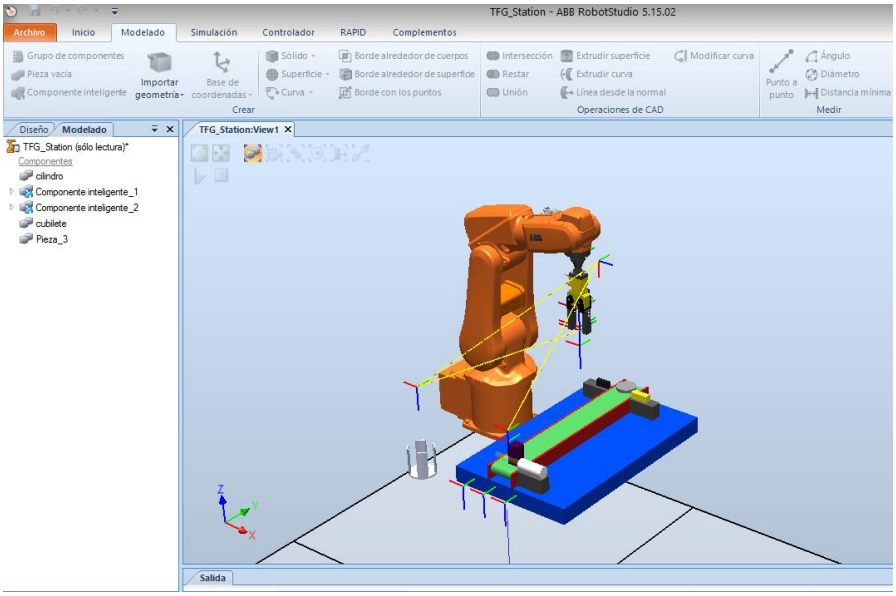

Figura 2.17 Estación de trabajo y entorno de RobotStudio

2.10 Ordenador de trabajo

Por último, es necesario un ordenador de trabajo donde poder ejecutar todas las herramientas software explicada anteriormente, así como poder ejecutar posteriormente el sistema de visión desarrollado con dichas herramientas.

En nuestro caso el desarrollo y ejecución de las pruebas se han llevado a cabo desde un portátil de la marca ASUS modelo F552C con las siguientes especificaciones:

-Sistema Operativo Windows 8.1

-Procesador Intel(R) Core(TM) i5-3337U CPU 1.8GHz

-Tarjeta Gráfica NVDIA GEFORCE 710M

-Memoria RAM 8 GB

3. Desarrollo del sistema

En este apartado se explicarán con detalle todos los procesos que se llevan a cabo en el funcionamiento del sistema, profundizando en las explicaciones de las funciones y algoritmos realizados, así como las operaciones de análisis de imágenes y comunicación entre dispositivos.

3.1 Esquema general del sistema

Para poder comprender todo lo que abarca nuestro proyecto, es necesario entender cómo se interrelacionan las distintas partes que componen el sistema. Por ello a continuación se muestra en rasgos generales las diversas conexiones que presenta el sistema entre los distintos componentes y herramientas que se utilizan:

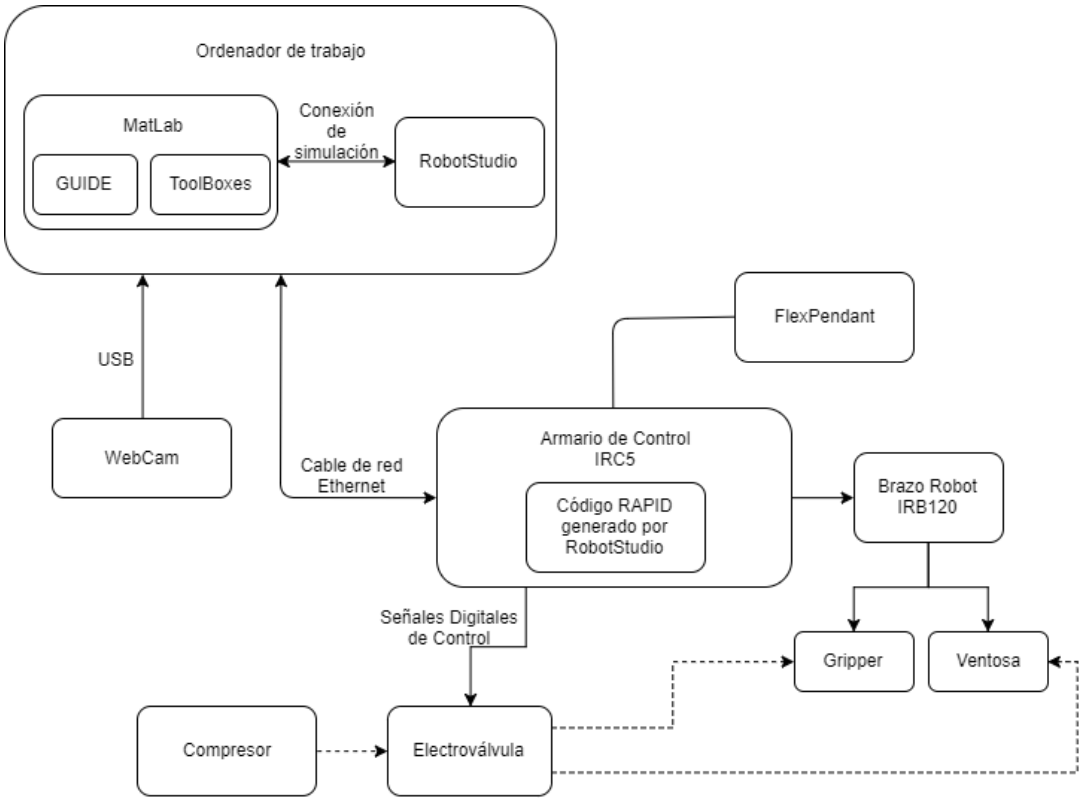

Figura 3.1 Esquema de conexionado del sistema completo

3.2 Comunicación TCP/IP

La comunicación entre el armario de control y el ordenador es una de las piezas esenciales a la hora de poner en marcha el proyecto desarrollado. A pesar de no ser el punto principal del proyecto, es importante que se cree una comunicación entre el IRC5 y el ordenador.

Para comunicar las ordenes de movimiento y manipulación de objetos es necesario una comunicación bidireccional, de forma que se pueda tanto ordenar al robot que realice diversas acciones como informar al ordenador de cuando ha acabado dichas tareas, consiguiendo un flujo de procesos controlados y seguros.

Esta comunicación se lleva a cabo mediante un socket de comunicación, conectando mediante un cable ethernet el ordenador y el armario de control IRC5.

Esta conexión se basa en la previamente desarrollada en el TFG de Javier Ribera Díaz: "Manipulación de objetos en una cinta transportadora mediante un sistema de visión artificial y brazo robot IRB120" [2], el cual a su vez está basado en el proyecto titulado: "Socket based communication in RobotStudio for controlling ABB-IRB120 robot. Design and development of a palletizing station"[3], de Marek Jerzy Frydrysiak .

3.2.1 Comunicación mediante socket

Denominamos socket al concepto abstracto por el cual dos programas, usualmente en sistemas electrónicos distintos, pueden llevar a cabo el intercambio de un flujo de datos bidireccional de manera segura y controlada.

El socket utilizado en este proyecto se centra en la conexión de red, ethernet, utilizando el conjunto de protocolos conocido como TCP/IP:

-TCP: Protocolo de control de transmisión

-IP: Protocolo de internet

Dentro de este conjunto existe una gran cantidad de protocolos diferentes llegando a contar más de 100.

Puesto que el proyecto no tiene como objetivo principal el desarrollo de la comunicación esta no se explicará con excesivo detalle, no obstante, si se abordará una explicación a alto nivel para comprender a grandes rasgos el método seguido a la hora de comunicar la información.

El código desarrollado para utilizar esta comunicación en los proyectos anteriormente mencionados se basa en una comunicación de información mediante datagramas variables. Estos datagramas no son si no arrays de datos numéricos, comprendidos entre 0 y 255 (1 byte) que constan de dos partes diferenciadas: identificados de mensaje y datos.

ID	Dato1	Dato2	Dato3	Dato4	Dato5	Dato6

Figura 3.2 Ejemplo de datagrama genérico

El identificador (ID) hace referencia al tipo de mensaje y la cantidad de datos que se está recibiendo, lo cual permite al programa analizar el número necesario de Datos necesario y saber interpretarlos correctamente.

Con esta estructura en mente hemos desarrollado una serie de datagramas que nos permiten realizar diversas tareas necesarias en nuestro proyecto.

3.1.1.1 Datagrama de Selección de Herramienta.

Este mensaje que deberá ser enviado una única vez al armario de control cuando se inician las operaciones. Como su propio nombre explica indica al robot el tipo de herramienta que estará utilizando para que en función de la orden active las salidas digitales correspondientes al control de dicha herramienta.

7	Herramienta

Figura 3.3 Datagrama de selección de herramienta

El "Dato1" puede variar entre '0' y '1', significando utilización de ventosa y gripper respectivamente.

3.1.1.2 Datagrama de Control de Cinta transportadora

Este mensaje controla el funcionamiento de la cinta transportadora. En función del valor del campo "Dato1", la cinta transportadora inicia su movimiento o lo para ('0' o '1') permitiendo así un control de la cinta desde el ordenador.

8	Encender/Apagar

Figura 3.4 Datagrama de control de cinta transportadora

3.1.1.3 Datagrama de Manipulación de objetos

Este datagrama es el más completo pues es el utilizado para comunicar las órdenes de manipulación de objetos. Dentro del campo de datos existen dos partes diferenciadas: las ordenes de movimiento y orientación que permiten la ejecución de trayectorias desde Matlab, y la orden de agarre o deje de los objetos (I), la cual se ejecutará internamente en función de la herramienta seleccionada.

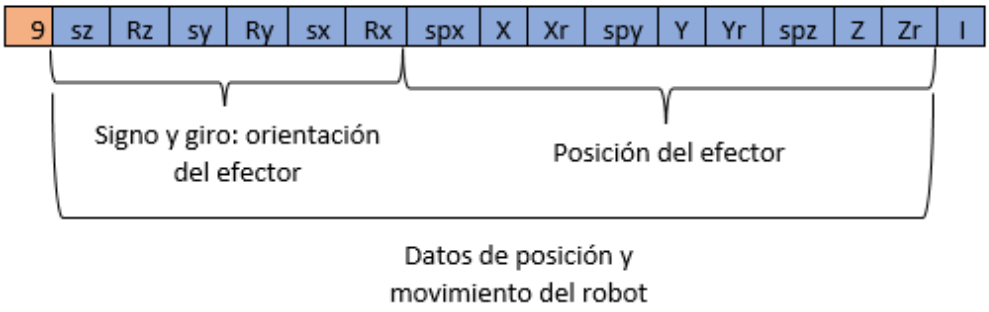

Figura 3.5 Datagrama de manipulación de objetos y movimientos

3.2.2 Comunicación entre Matlab y Armario de Control

Tanto para la creación de la conexión como para definir las funciones de comunicación se hace uso de una clase en Matlab en fichero "irb120.m" que nos permite crear una variable de dicha clase a la que denominamos "Robot". Este fichero fue desarrollado por Azahara Gutiérrez Corbacho en su TFG "Desarrollo de una interfaz para el control del robot IRB120 desde Matlab"[10].

Con dicha variable creada podremos hacer uso de las diferentes funciones de comunicación definidas dentro de la propia clase:

```
%Creación de la variable tipo irb120
Robot=irb120('172.29.28.185',1024);
```

Figura 3.6 Código MATLAB 1. Instanciación de objeto de la clase irb120

```
%Conectar Robot
Robot.connect;
```

Figura 3.7 Código MATLAB 2. Función de conexión incial entre Matlab y el IRB120

```
%Mensaje de selección de herramienta
Robot.TCPSelectTool(Herramienta);
```

Figura 3.8 Código MATLAB 3. Envío de datagrama de selección de herramienta

```
%Mensaje de Control de cinta transportadora
Robot.TCPMovimientoCinta(ActivarDesactivar);
```

Figura 3.9 Código MATLAB 4. Envío de datagrama de control de la cinta

```
%Mensaje de manipulación de objetos y control de movimientos
Message =[RotZ RotY RotX PosX PosY PosZ I];
Robot.TCPOrdenCogerPiezColor(Message);
```

Figura 3.10 Código MATLAB 5. Envío de datagrama de manipulación de objetos

Estos mensajes comunican los datagramas explicados con anterioridad con el armario de control el cual los analiza siguiendo el código RAPID desarrollado mediante RobotStudio, permitiendo interpretar correctamente lo que estos quieren decir

Es conveniente mencionar que el una vez establecida la conexión entre ambos elementos el armario de control queda inmóvil a la escucha de nuevos mensajes.

```
WHILE connectionStatus DO
SocketReceive socketClient \Data:=receivedData \ReadNoOfBytes:=19 \Time:=
WAIT_MAX;
stringHandler(receivedData);
ENDWHILE
```

Figura 3.11 Código RAPID 1. Bucle principal de espera del IRC5

Una vez los mensajes han sido analizados por el IRC5, éste envía un mensaje simple de vuelta hacia Matlab, informando de que el mensaje ha sido recibido y entendido de forma que puede realizarse una nueva operación. Este mensaje consiste en un número “67”.

Este tipo de comunicación puede ralentizar en cierta medida los procesos de movimiento del brazo robot, no obstante, mejoran significativamente el control sobre el sistema y la seguridad del mismo, permitiéndonos saber en cada momento si las órdenes han sido recibidas correctamente o sí han sido ejecutadas.

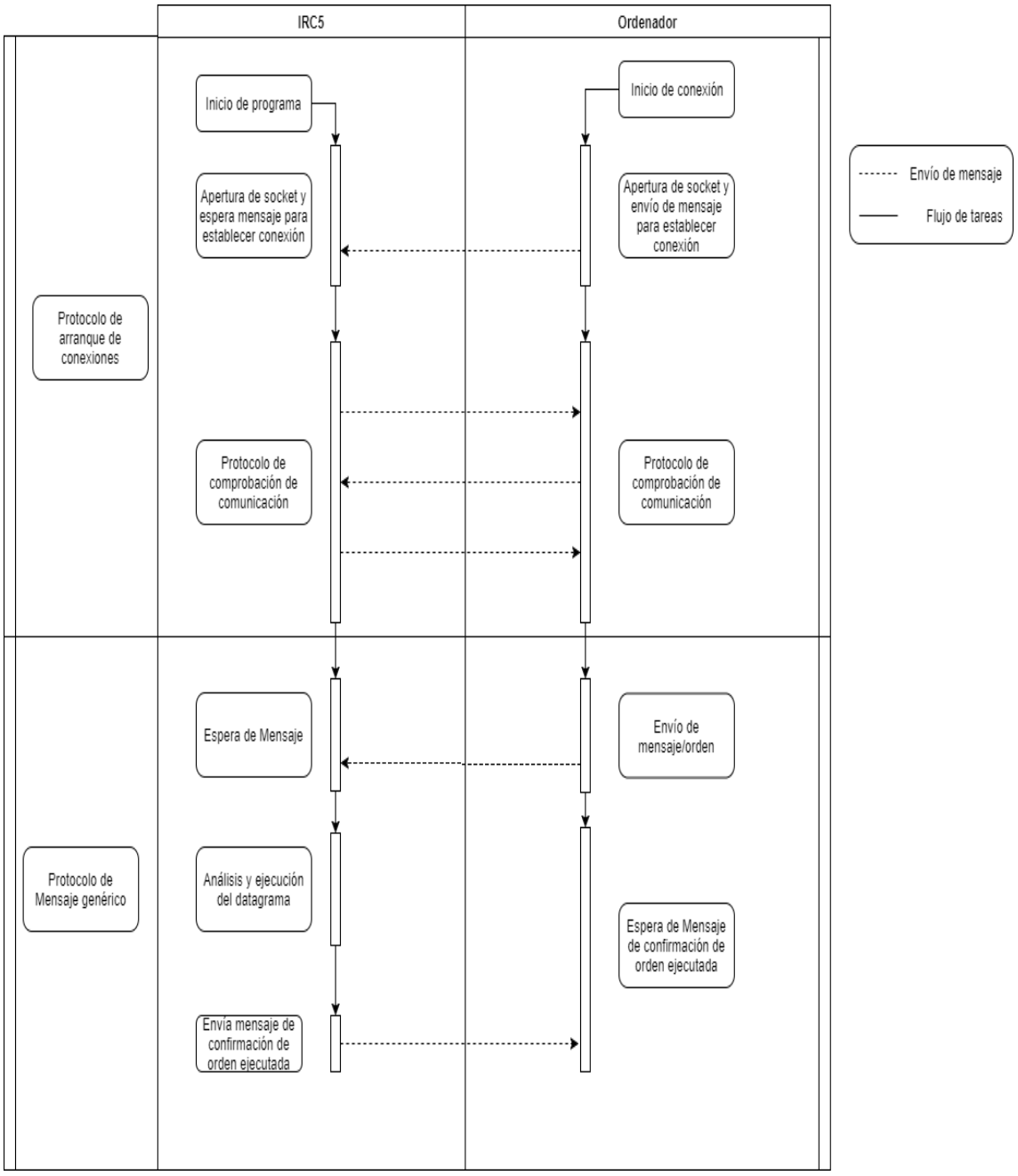

Figura 3.12 Flujo de conexión y comunicaciones entre el ordenador y el IRC5

3.3 Sistema de Visión Artificial

El sistema de Visión Artificial es el principal punto de este proyecto por lo que se explicará a continuación lo más detalladamente posible en función de las diversas tareas que se han abordado:

-Captación de imágenes

-Análisis por color

-Análisis de objetos circulares

-Análisis de objetos cuadrados

-Calibración del sistema

-Conversión de coordenadas pixélicas

-Detección de movimiento

-Alternativas en el sistema de detección

3.3.1 Captación de imágenes

La parte más esencial del sistema es la captación de imágenes en tiempo real del entorno de trabajo. Para ello como hemos explicado anteriormente hemos utilizado una WebCam presente en el laboratorio, situada sobre el espacio de trabajo y según las explicaciones detalladas en el manual de usuario.

Mediante esta cámara conectada al ordenador de trabajo y ayudándonos de la "Image acquisition toolbox" de Matlab se podrá captar la imagen deseada, en el momento indicado.

Cuando el sistema de visión se ponga en marcha encenderá la cámara y la mantendrá encendida hasta que el sistema se apague. Esto permitirá ahorrar un tiempo muy valioso entre capturas de imagen, evitando que la cámara se esté apagando y encendiendo constantemente y por ello, evitando que el número de frames por segundo decaiga a niveles demasiado bajos.

Además, se puede observar en todo momento lo que la cámara está viendo, aunque esto no signifique que el sistema sea capaz de analizar cada instante de forma absoluta como se explicará posteriormente.

```
vid = videoinput('winvideo', 1, 'MJPG_640x480');
src = getselectedsource(vid);
vid.FramesPerTrigger= 1;
vid.ReturnedColorspace= 'rgb';
preview(vid)
```

Figura 3.13 Código MATLAB 6. Encendido de la cámara y previsualización

Figura 3.14 Área de trabajo captado por la cámara

Como se puede apreciar a la hora de encender el sistema de captura se pueden seleccionar varias opciones, entre ellas el espacio de color en el que se captarán las imágenes o la resolución de captación de la imagen.

En el caso de este proyecto se ha decido hacer uso de la máxima resolución posible para una mayor precisión, esta resolución es de 640x480.

Respecto al sistema de color en el que se obtienen las imágenes la elección ha sido captarlas en sistema RGB (Red-Green-Blue). Este sistema de representación cromático permite representar los distintos colores existentes como una mezcla de los 3 colores primarios que dan nombre al modelo: rojo, verde y azul.

Con esto en mente las imágenes captadas en este sistema de representación estarán conformadas por tres matrices superpuestas, cada una representando un canal correspondiente a su color e indicando en cada posición la cantidad del color primario que posea cada pixel de la imagen.

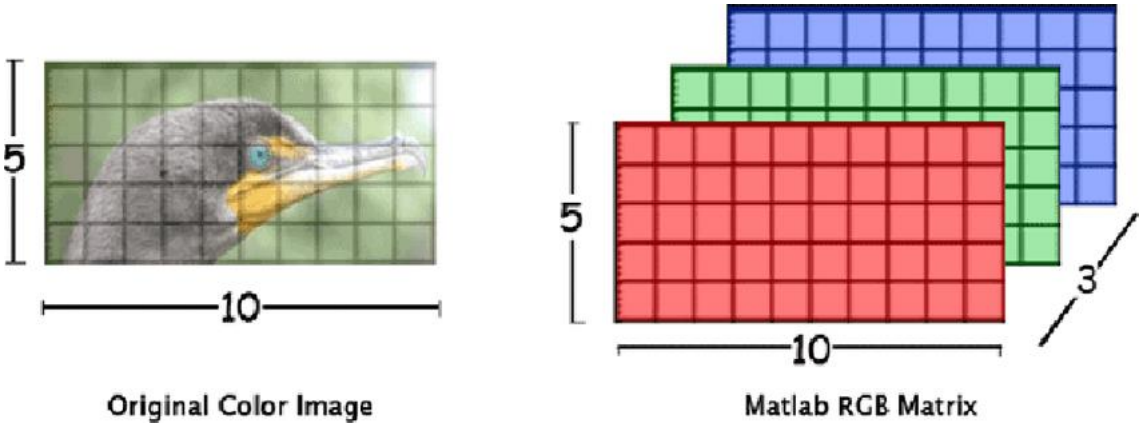

Figura 3.15 Esquema de representación de una imagen RGB en matrices

Puesto que las piezas a analizar presentan los mismos colores que los primarios del modelo podremos hacer un análisis más preciso de los mismos como explicaremos posteriormente.

Cabe mencionar que, aunque gran parte de los análisis se basen en el sistema RGB también se hará uso de otros modelos a lo largo del proyecto y que también se han planteado soluciones alternativas a algunos problemas utilizando varios de ellos.

3.3.2 Análisis de imágenes por color

Como se ha mencionado en el apartado anterior, las fichas poseen diversos colores y uno de nuestros objetivos es la separación de los objetos en función de esta característica, por este motivo analizaremos las imágenes en busca de objetos de los diferentes colores primarios.

Para ello se aprovecharán las características del sistema de representación RGB para una mayor facilidad de análisis: haciendo uso de la matriz correspondiente a cada color base se puede disminuir la carga de procesamiento y obtener resultados basándonos únicamente en la cantidad de color principal que consta en una única matriz.

Figura 3.16 Sección de imagen captada por la cámara en RGB

Primero se crea una nueva imagen en escala de grises a partir de la imagen original captada. Esta imagen contendrá una única matriz evaluando los niveles de gris atendiendo a los colores presentes en los diferentes píxeles.

```
%Pasamos a escala de grises
ImagenGris = rgb2gray(ROI);%Frame or Region
```

Figura 3.17 Código MATLAB 7. Transformación de imagen RGB a escalas de gris

Tras ello se extrae la matriz correspondiente al color que nos interesa detectar y la guardaremos con un nuevo nombre en función del color extraído. Esto generará una nueva imagen que, en una escala de "gris" representará el valor de la componente extraída dentro de la imagen.

```
%Obtenemos valores de un canal
Azul = ROI(:,:,3);
```

Figura 3.18 Código MATLAB 8. Obtención del tercer canal ("Blue") de la imagen

Una vez extraído el canal a analizar, se realizará la resta entre éste y la anterior imagen en escala de grises, consiguiendo así eliminar los objetos que no presentan un color primario lo suficientemente puro, y reduciendo así las listas de posibles objetos a analizar.

```
%Incrementamos Intensidad en un factor K
Azul=Azul-ImagenGris;
FactorK=2;
Azul= Azul*FactorK;
```

Figura 3.19 Código MATLAB 9. Tratamiento del canal "B" para facilitar análisis

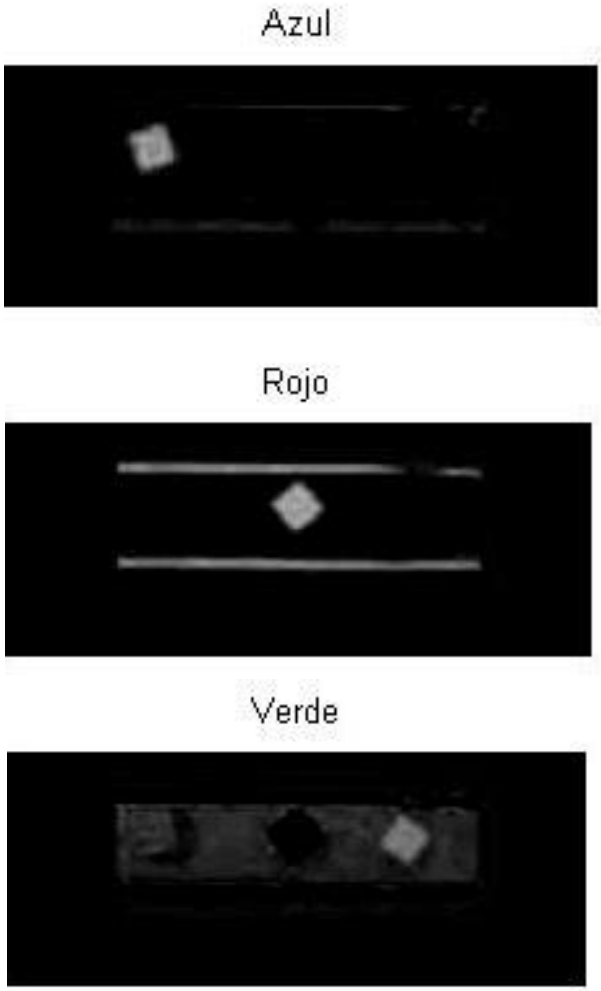

Figura 3.20 Imagen de los canales tras el primer tratamiento

El "FactorK" multiplica la diferencia entre imágenes aumentando la intensidad y haciendo que los objetos de color se tornen más fáciles de detectar. No obstante, este factor también amplía la intensidad del ruido en la imagen, por lo que no es conveniente que este factor se eleve en exceso o impedirá el correcto análisis de la imagen.

Para eliminar el posible ruido existente en la imagen y centrarnos en los objetos de posible interés, debemos llevar a cabo una limpieza mediante una serie de operaciones morfológicas sobre la imagen.

Primero se binariza la imagen, haciendo que la escala de colores pase a ser binaria, es decir, los colores observados serán "Blanco", perteneciente al color puro del objeto, o "Negro", fondo y partes de la imagen sin valor para nuestro análisis.

```
%Binarizamos
ImagenBin = im2bw(Azul, AjusteAzul);
```

Figura 3.21 Código MATLAB 10. Binarización de la imágen

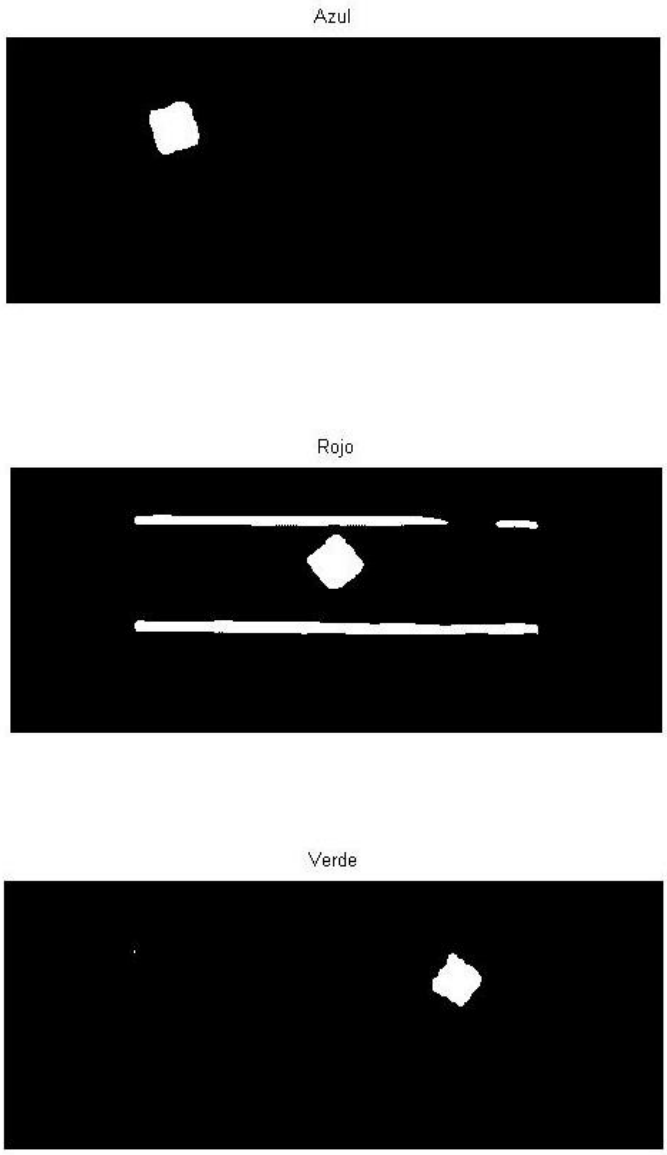

Figura 3.22 Imagen de los canales binarizados

La variable mostrada como “AjusteAzul” es un parámetro comprendido entre ‘0’ y ‘1’, qué nos sirve para determinar por encima de qué umbral debe estar el valor de un píxel para poder considerarse blanco o negro. Este umbral puede ser variable en función de las condiciones de luminosidad en la que nos encontremos por lo que será configurable para cada color desde la Interfaz Gráfica mediante diferentes sliders y un botón de validación. Si no se manipulan lo sliders se les otorgará a dichos umbrales un valor por defecto.

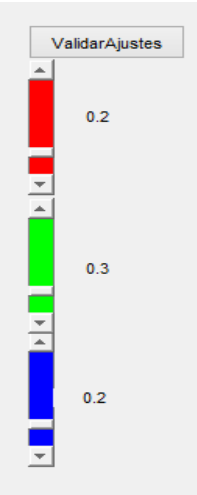

Figura 3.23 Sliders de ajuste de umbrales de binarización

Una vez se posee la imagen binarizada se realizarán dos operaciones que permitirán la eliminación de la mayoría del ruido presente en la imagen: “Erode” y “Dilate”.

La operación “Erode” evaluará cada pixel de la imagen, si el pixel objetivo de análisis o sus vecinos siguiendo la conectividad indicada no valen ‘1’ (Blanco) el pixel tomará valor ‘0’(Negro). Esto eliminará aquellos conjuntos de píxeles que no posean el tamaño necesario para ser considerados objetos y por tanto sean considerados ruido.

Aplicar este filtro reduce el tamaño de los conjuntos de píxeles que sí pueden ser considerados objetos impidiendo un análisis preciso de los mismos. Por ello sobre la imagen que ha eliminado aplicaremos el filtro “Dilate”.

La operación “Dilate” funciona de forma inversa a “Erode”, haciendo qué si el píxel analizado o cualquiera de sus vecinos siguiendo la correlación necesaria poseen un valor ‘1’ el píxel tomará dicho valor. De esta forma los conjuntos de píxeles que no habían sido eliminados por la operación “Erode” restaurarán su tamaño original.

Finalmente, las imágenes suelen presentar un reborde blanco que puede interferir con el análisis por lo que lo eliminaremos también en caso de que exista.

```
ImagenBin = imclearborder(ImagenBin);
```

Figura 3.24 Código MATLAB 11. Binarización de la imagen

Con esto se tiene una imagen que posee remarcados todos los posibles objetos del color que buscamos, permitiendo pasar a un nuevo nivel de búsqueda en función de otros parámetros diferentes al color.

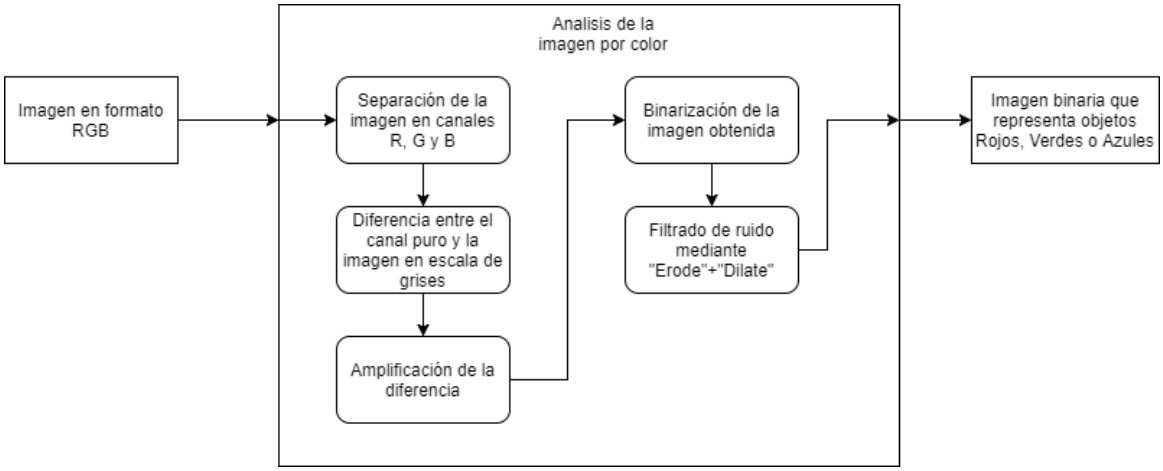

Figura 3.25 Flujograma del análisis de la imagen por color

3.3.3 Análisis de objetos circulares

Siguiendo el curso del análisis en función del color, poseemos una imagen que presenta en ella todas las posibilidades de objetos del color que buscamos. Por ello, partiendo de esta base podemos aplicar un nuevo proceso de búsqueda basándonos en la forma.

Para evaluar los objetos circulares primero debemos hallar y reconocer los contornos de todos los objetos para poder analizarlos uno a uno. Para dicho propósito aplicaremos un filtro tipo “Sobel” a nuestra imagen.

Aplicando el operador Sobel, calcularemos el gradiente de la intensidad de la imagen en cada pixel, de forma que en cada pixel nos aporta la magnitud del cambio, dirección y sentido, es decir, nos muestra cómo cambia la imagen a cada punto de tal forma que si el cambio es muy grande aumenta la probabilidad de que el pixel forme parte de un borde o contorno.

```
Sobel=edge(ImagenBin,'sobel');
```

Figura 3.26 Código MATLAB 12. Filtro Sobel

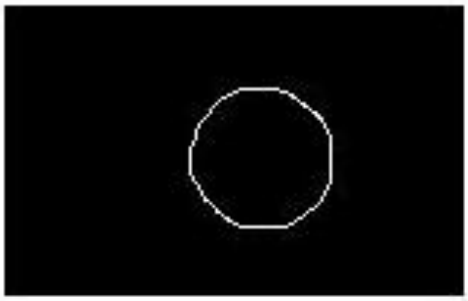

Figura 3.27 Imagen de un disco tras filtro Sobel

Con esto aplicado poseemos ahora una imagen que, no solo posee todos los posibles objetos del color deseado, sino que además sus bordes están resaltados de forma que nos sea más fácil analizarlos.

Haciendo uso de la función "bwboundaries" proporcionada dentro de la toolbox "Image processing" sobre la imagen "Sobel", podremos analizar y generar una serie de matrices que contengan los contornos de los diferentes objetos.

```
%Obtención de Limites-Bordes-Contornos
%Boundary=Matriz de matrices con contornos
%Label=Matriz de objetos y agujeros etiquetados
[Boundary, Label] = bwboundaries(Sobel, 'noholes');
```

Figura 3.28 Código MATLAB 13. Obtención de todos los contornos de la imagen

Con todos los contornos localizados se realizará un análisis individual de cada uno, para ello se genera una imagen cuyo valor de todos los píxeles sea '0', es decir una imagen puramente de color negro.

Luego sobre dicha imagen se dibujará el contorno del borde a analizar y rellenaremos su interior, consiguiendo que en la imagen tengamos el objeto en blanco y sobre fondo negro.

```
%Imagen de Zeros
ProcessImage=zeros(size(ImagenBin));
%Borde a analizar
Borde=Boundary{j};
%Pintamos borde
for i=1:length(Borde(:,1))
   ProcessImage(Borde(i,1),Borde(i,2))=1;
```

```
end
%Rellenar Contornos interiores
ProcessImage=imfill(ProcessImage, 'holes');
```

Figura 3.29 Código MATLAB 14. Creación de imagen con un solo objeto

Con esta nueva imagen que solo contiene la presencia de un único objeto se puede utilizar utilizar la función "regionprops" que aportará una estructura con diversa información de la todas las zonas blancas de la imagen como si fuera un objeto. De toda la información disponible captaremos aquellas variables que nos permita comprobar si el objeto corresponde o no a uno de los discos objetivos o no:

-Área de la región
-Eje menor de la elipse de contención
-Eje mayor de la elipse de contención

```
Propiedades=regionprops(ProcessImage,'all');
Area=Propiedades.Area
MajorAxis=Propiedades.MajorAxisLength;
MinorAxis=Propiedades.MinorAxisLength;
```

Figura 3.30 Código MATLAB 15. Obtención de propiedades del objeto en la imagen

Mediante el uso de ambos ejes de la elipse que contiene la región podremos hacer un cálculo aproximado del radio del posible disco y, con este radio, podremos calcular el área que debería tener si fuera un disco.

```
Radio=((MajorAxis+MinorAxis)/2)/2
AreaCalcu=round(pi*Radio^2);
```

Figura 3.31 Código MATLAB 16. Cálculo del radio y área del posible circulo

Con estos dos elementos podemos utilizar el concepto de radio y circularidad para hacer restricciones. De forma que si el radio hallado del objeto no pertenece al rango establecido para ser uno de nuestros discos objetivo quedará descartado.

Si la región supera la restricción de radio se le someterá a la restricción de circularidad, utilizando las dos áreas calculadas obtendremos un factor de circularidad, el cual terminará de demostrar que el objeto sometido a análisis se trata de un disco objetivo. Si

es así, se tomará el centro del objetivo, que será el parámetro principal que nos permita manipular el disco mediante el uso de la herramienta ventosa.

```
if (Radio>17 & Radio<28)%Contornos con Radios que se adapten
  Circularidad=(Area/AreaCalcu);
  ErrorCircularidad=abs(1-Circularidad)
   if ErrorCircularidad<0.2 %Si se asemeja un 80%de Circularidad
            Centroide=Propiedades.Centroid;
   end
end
```

Figura 3.32 Código MATLAB 17. Comprobación de circularidad del objeto detectado

Se someterán a este procedimiento todos los contornos encontrados y dispuestos en la matriz "Boundary", de esta forma se determinará también si existieran varios discos objetivo.

Figura 3.33 Resultados, centro del disco detectado tras el análisis

Es importante mencionar que el centro del objetivo es tomado en coordenadas de imagen, es decir en píxeles por lo que posteriormente tendremos que transformar este dato a coordenadas respecto a ejes del robot.

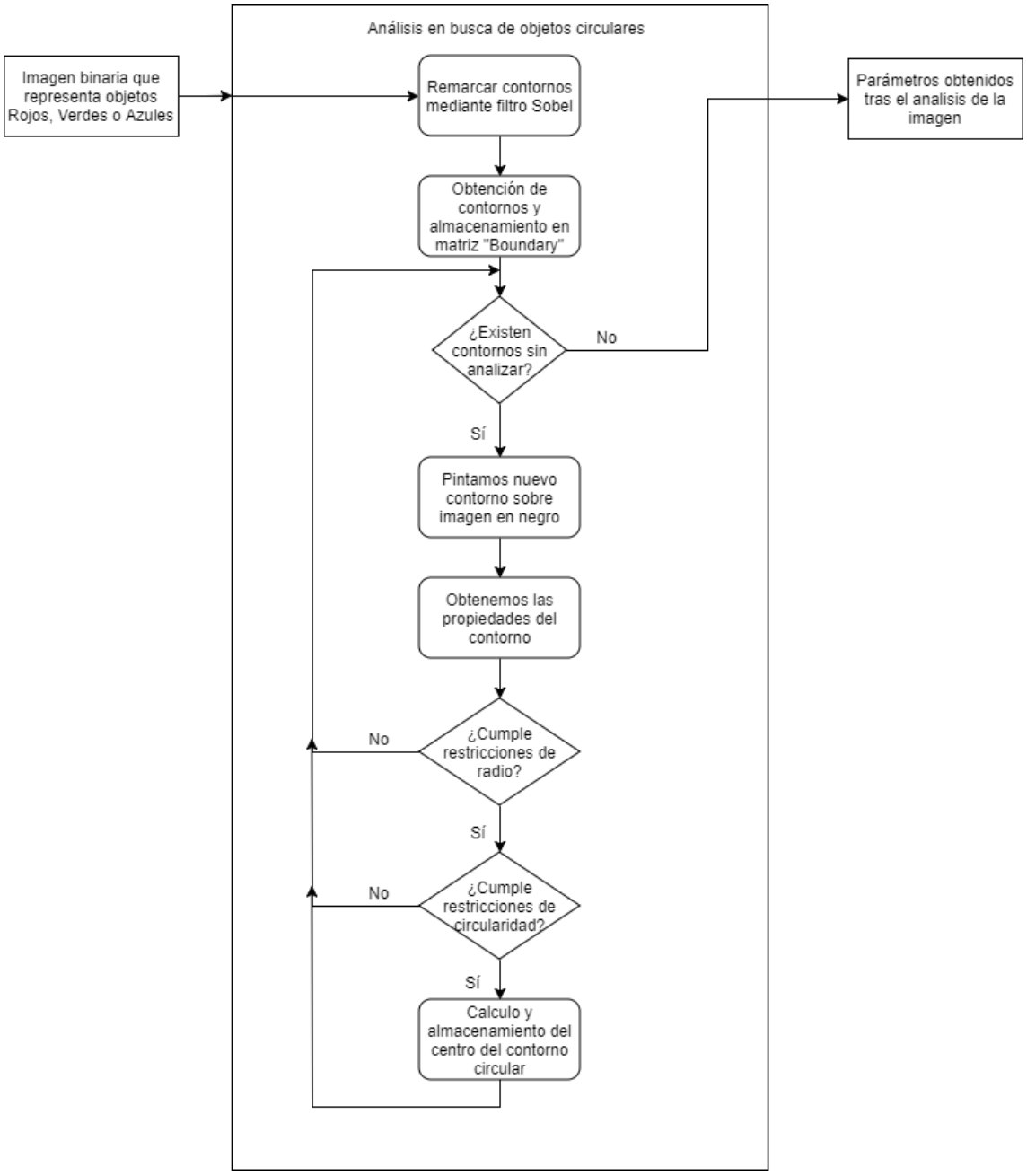

Figura 3.34 Flujograma de análisis de objetos circulares

3.3.4 Análisis de objetos cuadrados

El análisis de objetos cuadrados, prismas rectangulares de base cuadrada en nuestro caso, se lleva a cabo de manera muy similar al análisis de objetos circulares. Partimos de la misma base una vez obtenido el canal limpio de ruido tras el "análisis de la imagen color".

Tras ello ejecutamos el filtro "Sobel" de nuevo para definir correctamente los posibles bordes existentes en la imagen y, una vez más utilizamos la función "bwboundaries" para conseguir una matriz de matrices que contenga todos los contornos que deberemos analizar uno a uno en busca de nuestro prisma.

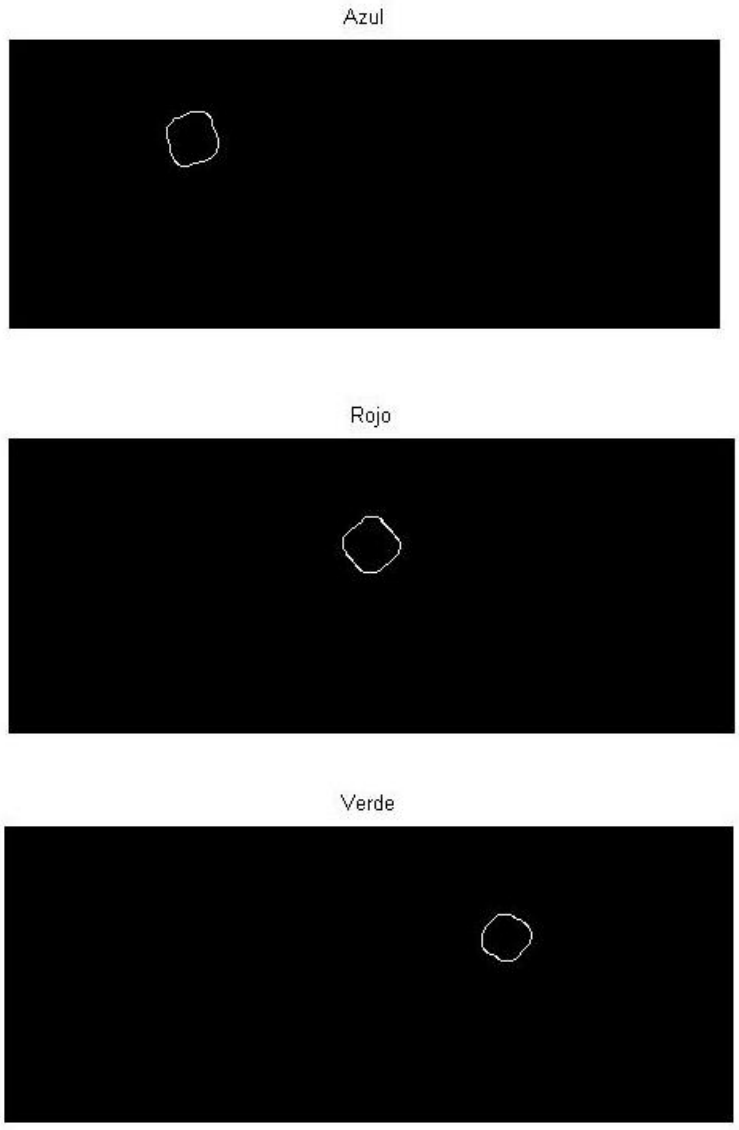

Figura 3.35 Figura 3.22 tras aplicar el filtro Sobel

La forma de análisis es idéntica al caso anterior, pintaremos los objetos uno a uno sobre un fondo negro, obtendremos de ellos las propiedades de la región que ocupan y las someteremos a unas restricciones para determinar si corresponden o no a los objetos buscados.

En el caso de la circularidad los parámetros restrictivos eran la circularidad y el radio, en este caso aplicaremos directamente una restricción de área y posteriormente de cuadratura.

Es importante mencionar que los datos que nos interesan a la hora de obtener una coincidencia de objeto cuadrado son su centro y el ángulo de giro respecto a los ejes de la imagen. Para averiguar estos datos, al igual que para determinar la cuadratura del objeto, nos hemos apoyado en la función: "minBoundingBox", diseñada por el usuario "Julien Diener" y aportada en la comunidad de usuarios de MathWorks[4].

Esta función nos permite, dado un contorno conocido dado en forma de array de puntos, obtener el mínimo cuadrado que lo contiene por completo, devolviéndonos las cuatro esquinas qué permiten delimitar el resultado.

```
minbox=minBoundingBox(Borde');
```

Figura 3.36 Código MATLAB 18. Función para hallarla "Bounding Box"

Mediante la obtención de las cuatro esquinas nos centraremos principalmente en las esquinas derecha-superior e inferior-derecha, utilizando en ese orden la prioridad de búsqueda:

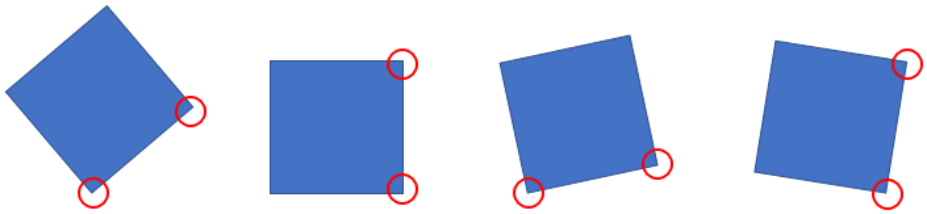

Figura 3.37 Ejemplo de esquinas a detectar

Encontrando dichos puntos se puede obtener una enorme cantidad de información que permitirá obtener todos los parámetros necesarios para la posterior manipulación del objeto.

Se comienza determinando si los dos puntos se encuentran alineados dentro del mismo eje principal, es decir, si ambos puntos poseen la misma coordenada X dentro de la imagen o no, de esta manera se pueden obtener los datos de diferentes maneras según el resultado.

Si ambos puntos se encuentran en la misma coordenada X, podemos relacionar directamente el hecho de que el ángulo de inclinación respecto al eje de coordenadas 'Y' es de 90º.

La forma de calcular el centro de la región cuadrada y el lado también se simplifica significativamente de esta forma:

```
CentroY1=round(minbox(1,1)+minbox(1,2)+minbox(1,3)+minbox(1,4))/4;
CentroX1=round(minbox(2,1)+minbox(2,2)+minbox(2,3)+minbox(2,4))/4;
Centro1=[CentroX1,CentroY1]
Centro=Centro1;
L=abs(EsquinaDerecha(1,1)-EsquinaInferior(1,1));
```

Figura 3.38 Código MATLAB 19. Cálculo de centro y lado del cuadrado (Método1)

Por otro lado, si los puntos no están dispuestos en la misma coordenada 'X' el cálculo se complica ligeramente, teniendo presente que ahora los lados y el cuadrado mismo presentan una inclinación.

En este caso primero hallaremos el ángulo de inclinación pertinente mediante trigonometría.

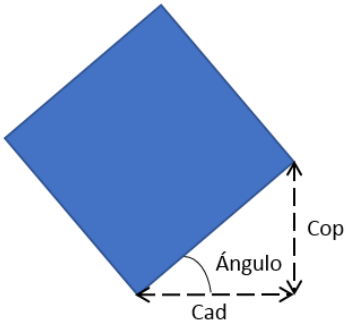

Figura 3.39 Datos del cuadrado para hallar el ángulo de giro

```
%Cateto Ayacente X
Cad=EsquinaDerecha(2,1)-EsquinaInferior(2,1);
%Cateto Opuesto Y
Cop=EsquinaInferior(1,1)-EsquinaDerecha(1,1);
tangente=Cop/Cad;
Angulo=atand(tangente);
```

Figura 3.40 Código MATLAB 20. Obtención del ángulo de giro

Tras ello, teniendo en cuenta la geometría del cuadrado y mediante la obtención del lado del mismo podremos hallar el centro, siguiendo los cálculos explicados a continuación.

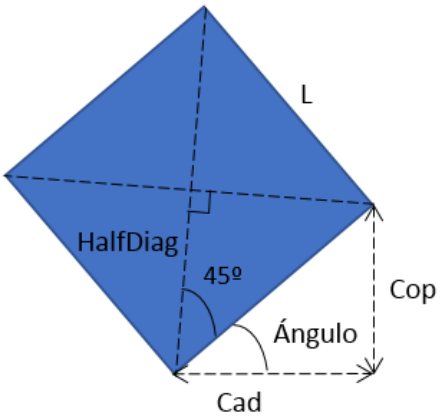

Figura 3.41 Datos del cuadrado para hallar el centro del cuadrado

```
L=sqrt((Cad^2)+(Cop^2));%Modulo del lado
Alfa=45+Angulo;
HalfDiag=L*cosd(45); %Modulo de la diagonal
CentroY2=EsquinaInferior(1,1)-HalfDiag*sind(Alfa);
CentroX2=EsquinaInferior(2,1)+HalfDiag*cosd(Alfa);
Centro2=[CentroX2,CentroY2];
Centro=Centro2;
```

Figura 3.42 Código MATLAB 21. Cálculo de centro y lado del cuadrado (Método2)

Con esto obtenemos los parámetros necesarios para la manipulación del objeto en caso de que la región haya sido identificada como el mismo, en caso contrario la región será descartada.

Figura 3.43 Resultados, puntos de interés detectados tras el análisis

Al igual que la búsqueda de objetos circulares, el análisis será llevado a cabo sobre todas las regiones de contornos que hayan sido determinadas y almacenados en la matriz "Boundary" y, de igual forma, los datos del centro del objeto también son tomados en coordenadas en pixélicas.

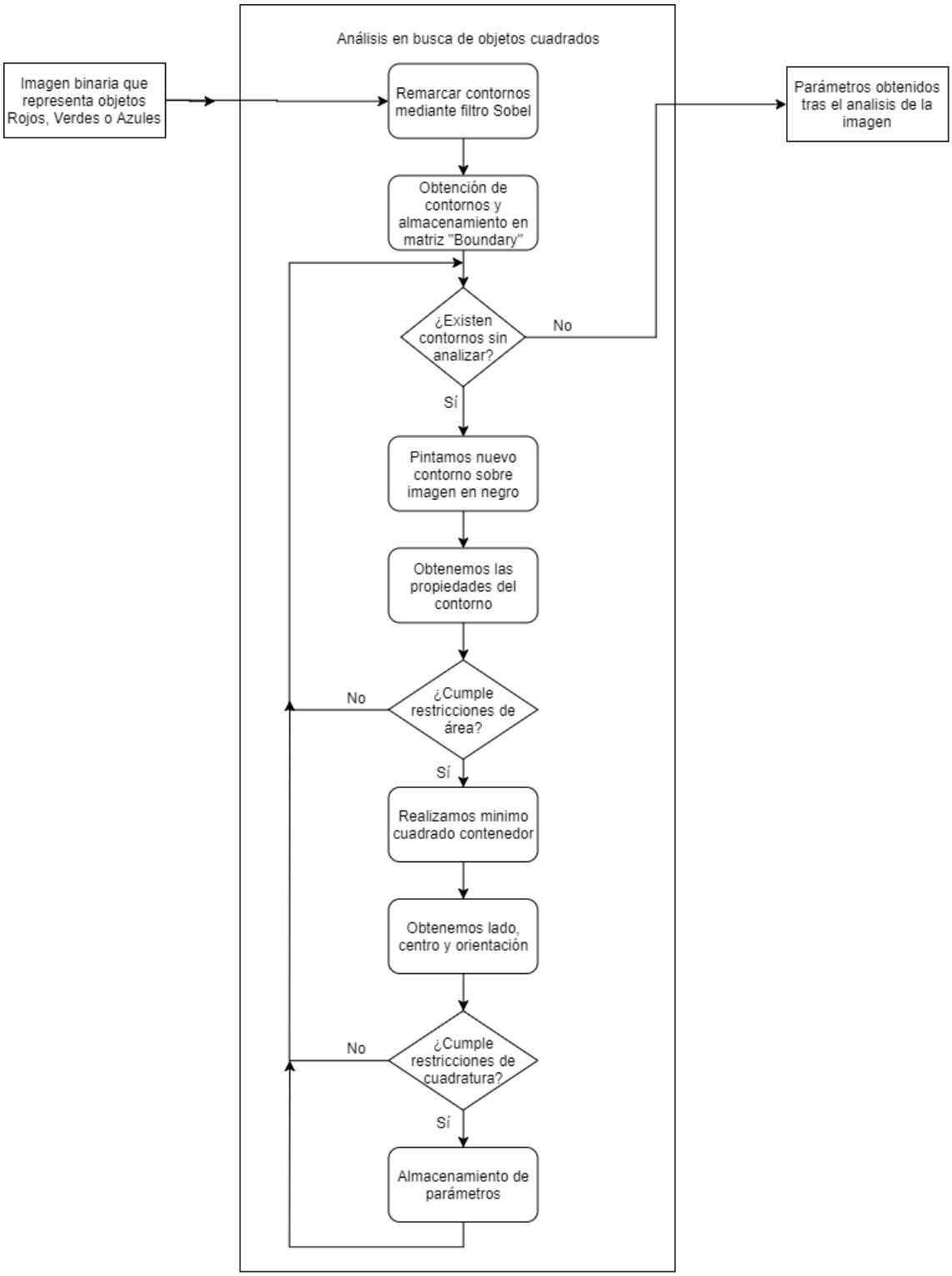

Figura 3.44 Flujograma de análisis de objetos cuadrados

3.3.5 Calibración del sistema

Sabiendo analizar y localizar los dos tipos de objetivos de manipulación que presentes en nuestra maqueta, debemos pasar al siguiente paso de nuestro sistema, la manipulación de los mismos.

Como se ha explicado anteriormente los centros de los objetos están tomados en coordenadas pixélicas, pero para poder manipular correctamente los objetos mediante el brazo robótico debemos entregarle al armario de control órdenes de movimiento y posición basadas en coordenadas en los ejes del robot, es decir en mm.

Para poder realizar esta transformación debemos obtener un plano de referencia a partir del cual hacer transformaciones relativas, este apartado explicará brevemente el funcionamiento de la función "calibración" que se ocupa de obtener esos parámetros.

En primer lugar se fijaran dentro del programa dos posiciones de referencia, las cuales han sido designadas como P1=[400,200] y P2=[200,-200], en coordenadas del robot. En dichas posiciones situaremos dos discos azules, asegurándonos de que los centros de los mismo estén situados lo más aproximadamente a los puntos de referencia.

Tras ello la función se ocupará de captar una imagen y analizarla basándose en un análisis de color y circularidad, obteniendo así el centro de los objetos en base pixélica.

Con esto tenemos dos puntos de referencia con los que generar un plan tanto en coordenadas de imagen como en coordenadas reales, pudiendo así generar cálculos relativos de la posición de otros puntos.

3.3.6 Transformación entre sistemas

Este es uno de los puntos principales que determina la precisión del sistema, con los datos obtenidos a partir de la calibración y los datos aportado en coordenadas pixélicas debemos ser capaces de obtener las coordenadas de posición del objeto dentro del espacio de trabajo para poder guiar el brazo robot a su objetivo.

Esta transformación se lleva a cabo mediante un cálculo relativo basándonos en los puntos de referencia tal como se muestra a continuación.

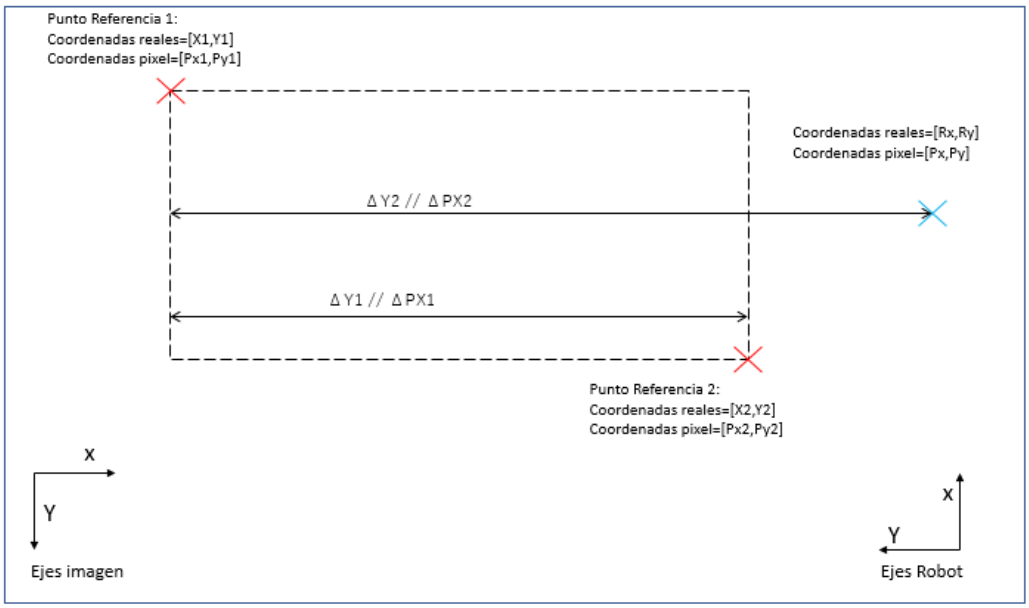

Figura 3.45 Plano de referencia y ejes de referencia de sistemas de representación

```
DistanciaY=((Px-gReferencia1X)*gRefDifY)/(gReferencia2X-
gReferencia1X)%mm
DistanciaX=((Py-gReferencia1Y)*gRefDifX)/(gReferencia2Y-
gReferencia1Y)%mm
PosicionX=gRef1Xmm+DistanciaX
PosicionY=gRef1Ymm+DistanciaY
```

Figura 3.46 Código MATLAB 22. Localización ideal del objeto en ejes del robot

No obstante, existe un detalle importante a tener en cuenta en la transformación, la lente de la cámara presenta un ángulo de visión que hace que al alejarse del centro de la cámara se perciba la distancia erróneamente sobre el plano si el objeto tiene una determinada altura, es decir se produce un error debido a que la cámara no es capaz de interpretar la profundidad del objeto.

En la imagen se puede observar que el error que se produce entre el punto calculado por la cámara y el puto real sobre el plano puede llegar a ser bastante significativo si el objeto presenta una altura suficiente:

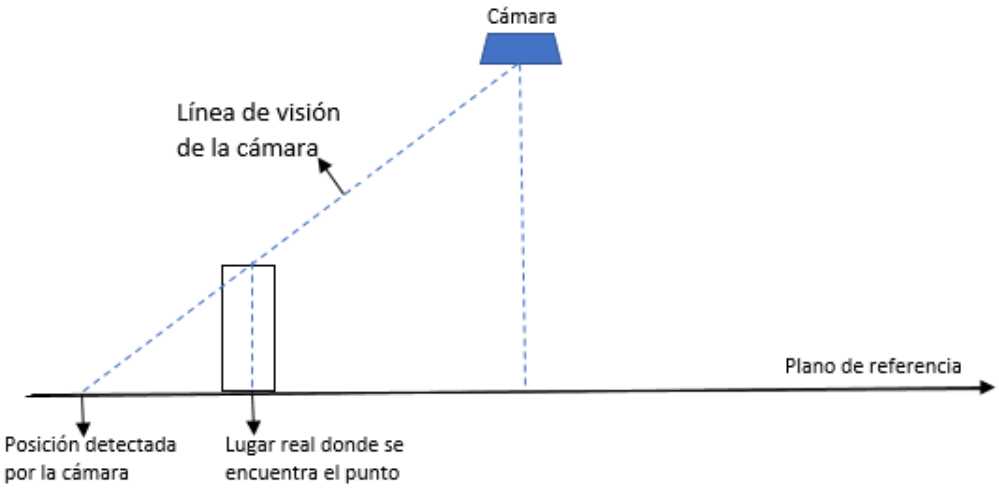

Figura 3.47 Diagrama de línea de visión de la cámara

Este error es sistemático y se produce en cualquier dirección al alejarnos del dentro de la cámara. Por ello teniendo en cuenta la posición del centro de la cámara y la altura que presentan los objetos de trabajo podemos eliminar el error.

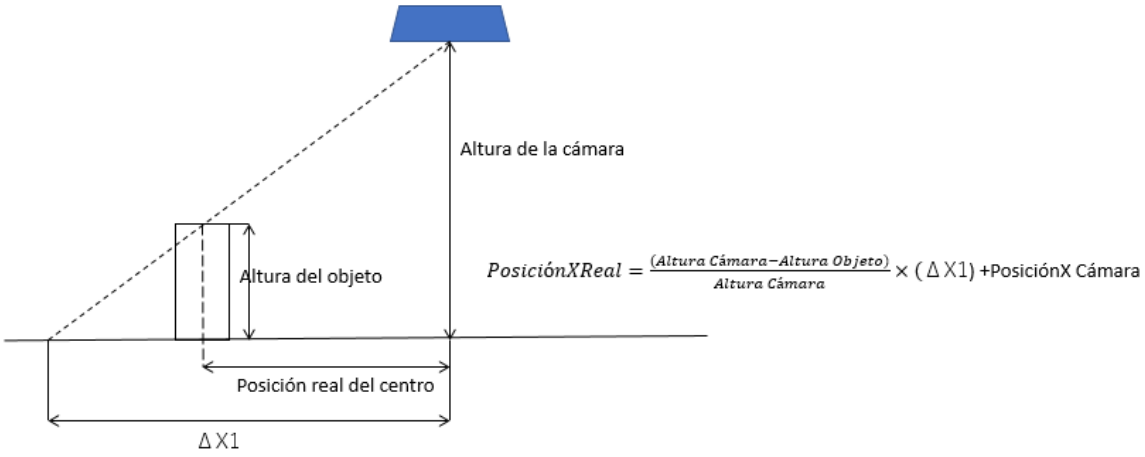

Figura 3.48 Diagrama de corrección de posición por altura

```
%Ajuste de Errores por Altura de objetos
PosX=((AlturaCamara-AlturaObjeto)/AlturaCamara)*(PosicionX-
PosicionXCamara)+PosicionXCamara%-5
PosY=((AlturaCamara-AlturaObjeto)/AlturaCamara)*(PosicionY-
PosicionYCamara)+PosicionYCamara%+5
```

Figura 3.49 Código MATLAB 23. Corrección de posición del objeto por altura

Todos los cálculos y correlaciones entre las distintas medidas están, en su origen, en base de píxeles, por ello el tamaño del píxel es un dato importante a la hora de definir la presencia del sistema, cuanto mayor sea la resolución de la cámara menor será el tamaño de los píxeles y por tanto se podrá obtener una mayor precisión de cálculo. Si por el contrario la resolución de la cámara es menor el sistema perderá precisión.

3.3.7 Detección de nuevos objetos en la escena

Mediante los apartados anteriores el sistema es capaz de analizar una imagen y localizar los objetos, obteniendo los datos necesarios para manipularlos cuando tenemos la cinta transportadora parada.

Para que se dé dicha manipulación se debe entender que la cinta transportadora llevará los objetos a través de ella hasta que se detecte la entrada del objeto en el rango de trabajo o el lugar donde se quiera manipular la pieza, en cuyo momento la cinta transportadora cesará su actividad hasta que se lleve a cabo el análisis de la imagen con la cinta parada.

Para llevar a cabo esta detección del objeto se ha iniciado el proyecto utilizando el sensor capacitivo presente en la maqueta, el cuál conectado al armario de control enviaba una señal que permitía la parada de la cinta.

No obstante, este método conlleva dos puntos no aptos: en primer lugar, la señal de parada es enviada hacia el armario de control, y por tanto es este el que toma el control del flujo de los procesos del sistema por unos instantes, aumentando las posibilidades de desincronización entre el IRC5y el ordenador. En segunda instancia la tarea puede ser llevada a cabo mediante visión artificial, abaratando el coste del proyecto al permitirnos eliminar de la lista de materiales sensores de este tipo.

Teniendo en cuenta estos puntos se ha desarrollado un sistema de detección de movimiento mediante visión artificial atendiendo a un método denominado como “Background substraction” o “substracción de fondo”.

Como su propio nombre indica, este método consiste en la eliminación de todos los elementos denominados como "fondo" de la imagen, captando únicamente los elementos no considerados parte del mismo y, por tanto, determinados como posibles nuevos objetos entrantes en el área de trabajo.

Este método funciona en dos partes diferenciadas: captación/entrenamiento del fondo y comparación de imagen a tiempo real.

La captación o entrenamiento del fondo se lleva a cabo antes de comenzar a utilizar el sistema mediante el botón "Train Background" como se explica en el funcionamiento del sistema a nivel de usuario, tomando una serie de imágenes de las cuales se obtendrá el "fondo":

La tarea llevada a cabo por la función consta de varios procesos menores, los cuales se mencionan a continuación junto con una explicación de su función:

- Primero se sitúa el brazo robot en la posición intermedia o, posición en la que se situará el brazo durante los momentos de reposo, es decir, mientras espera a que el sistema le envíe instrucciones de movimiento que coincide con el momento en que la cinta está en funcionamiento.

  Este movimiento puede parecer innecesario a simple vista, no obstante, es bastante importante debido a los cambios que produce en la captación de la imagen la posición del brazo robot. En función de la posición en la que se encuentre el mismo no solo generara un fondo diferente debido a su propia presencia en la imagen, sino que también se presentarán distintas sombras y variaciones en la iluminación de la maqueta, produciendo que, en el momento de comparar la imagen de fondo y el fotograma en el momento real, una diferencia entre luminosidad reflejada pueda provocar una captación errónea de diferencias entre imágenes.

  Esta diferencia en la luminosidad no es claramente visible para el ojo humano, debido a que la diferencia de luminosidad solo provoca una pequeña variación en el valor de los píxeles, no obstante, es la variación del valor de todos los píxeles y la media que se realiza posteriormente en toda la imagen la que provocan el error de falso positivo al superar la media umbral que determina la presencia de un nuevo objeto.

- Segundo pondrá en funcionamiento la cinta transportadora, la cual se deberá mover sin llevar ningún objeto sobre ella para evitar errores en el funcionamiento de la función.

  La función de poner la cinta en marcha es debida a que la superficie de la misma presenta irregularidades a lo largo de su superficie, provocando ligeras diferencias en imágenes captadas en diferentes transcursos de la cinta, pudiendo generar al igual que las sombras un error por captación de objetos en la cinta que no existen.

  Al ponerla en movimiento y tomar varias imágenes a lo largo de su recorrido podemos realizar una media eliminando ese factor de error.

- Tercero y último, la cámara captará 20 imágenes del entorno de trabajo en estas condiciones en una escala de grises. Esta adquisición durará aproximadamente 20 segundos, permitiendo que las imágenes tomadas abarquen la totalidad de la superficie de la cinta.

  Tras obtener las imágenes se realizará la media de todas ellas píxel a píxel, consiguiendo una única imagen de fondo con la que podremos comparar los fotogramas en tiempo real en busca de diferencias en la imagen, es decir, detectando la presencia de objetos nuevos.

La comparación con imágenes a tiempo real es la parte esencial de la captación de movimiento. Cuando se inician las operaciones y la cinta transportadora comienza a funcionar, pudiendo llevar consigo los objetos la cámara capta la imagen en tiempo real.

La captación de diferencias se realiza utilizando varias operaciones tal como se explica a continuación:

- La imagen captada en formato RGB es transformada a escala de grises, manteniendo toda la información sin extraer ningún canal en concreto como se hacía en los análisis por color.
- Sobre dicha imagen captada en tiempo real se restará la imagen de "fondo" obtenida mediante el procedimiento explicado anteriormente.
- Una vez restadas en la imagen resultante permanecerán aquellas regiones que no poseyeran el mismo valor de píxel, es decir, se mantendrán en la imagen aquellas regiones en las que se presente un posible nuevo objeto.
- Para simplificar las operaciones siguientes se convertirá la imagen en una imagen binaria, posicionando las regiones de posibles objetos con el valor '1' (Blanco) y el resto de píxeles tomará valor '0' (Negro) si no lo poseía ya.

- Las regiones que mantienen un valor distinto de ‘0’ pueden haber perdido cierta intensidad tras la operación por lo que es conveniente multiplicar la imagen por un factor que intente compensar la posible pérdida de valor.
- Así mismo, la imagen resultante puede presentar ruido o regiones diferentes que no puedan ser consideras un objeto si no un simple error en la captación de la imagen por lo que deberemos aplicar un filtro que lo elimine. Por ello utilizaremos las mismas operaciones morfológicas de apartados anteriores: “Erode” y “Dilate”.
- Sobre el resultado final se realizará una media del valor de los píxeles de la imagen. Dicha media representará numéricamente el valor de diferencia presente en la imagen con respecto a una imagen que no presente ningún cambio, es decir, cuanto mayor sea el contenido de píxeles blancos en la imagen, mayor será el número obtenido y por tanto mayor diferencia existirá entre la imagen tomada del fondo y la imagen captada en tiempo real.

Con esto, si el número obtenido es mayor al umbral de diferencia mínimo fijado se tomará como que un objeto nuevo está presente en el sistema y por tanto se mandará desde el ordenador la orden de parar la cámara y analizar la imagen, buscando en ella los posibles objetos que puedan ser manipulados por el brazo robot.

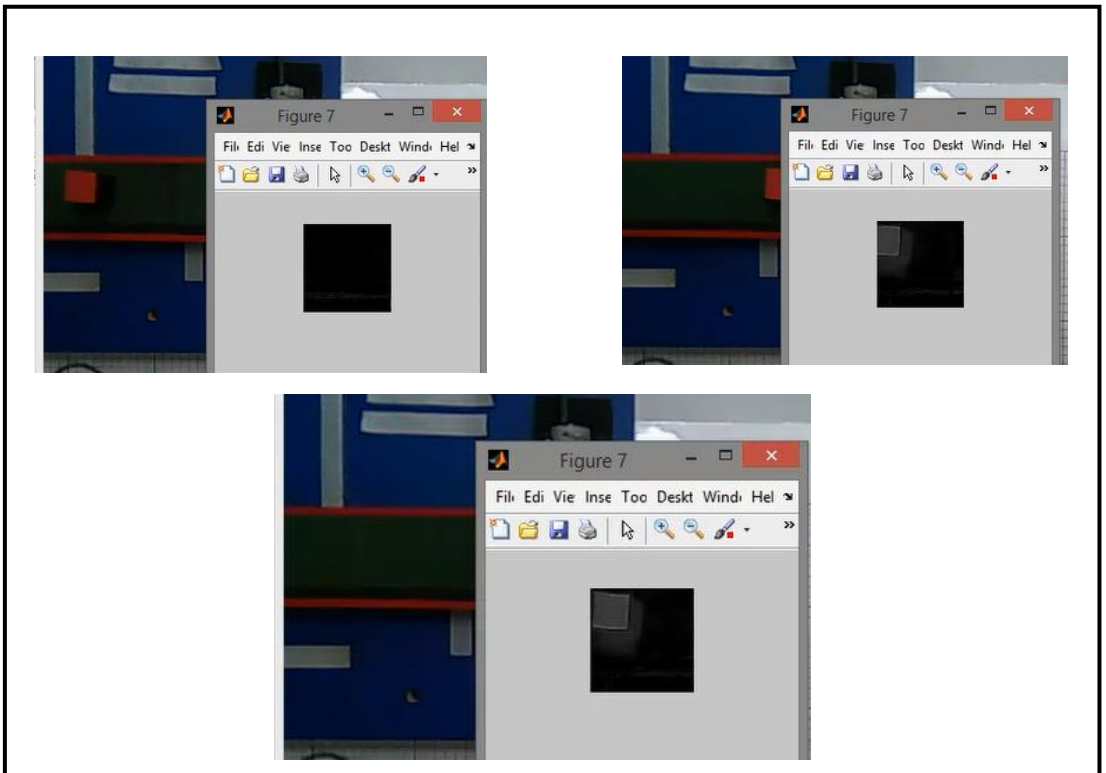

Figura 3.50 Detección de objetos entrando en el ROI

3.3.8 Región de interés (ROI)

Hasta el momento hemos hablado del "espacio de trabajo" refiriéndonos a toda la zona que se encuentra dentro del rango de movimiento del brazo robot y además es captado por la cámara de forma que podamos obtener información del mismo, pero, si bien es cierto que analizando una imagen más grande podemos tener el control de lo que ocurre sobre un espacio mayor, esto ralentiza las operaciones al tener que realizar cálculos sobre matrices más grandes, es decir aumenta el tiempo de procesado.

En sistemas en los que se necesita analizar áreas de trabajo muy grandes por motivos de mayor peso, este aumento en el tiempo de procesado puede ser pasable en vistas al funcionamiento del sistema global, no obstante, nuestro sistema presenta unas cualidades que nos permiten reducir el campo en el que sabemos que queremos operar sin necesidad de observar todo el espacio de trabajo.

El lugar donde realmente deben centrarse las operaciones de visión es sobre la cinta transportadora, más concretamente sobre el área de la misma en la que queremos que se detecten los objetos y sean capturados por el brazo robot.

Esto nos da la posibilidad de reducir el espacio de trabajo analizado a solo la zona final de la cinta transportadora, reduciendo la zona de análisis y con ello pudiendo reducir significativamente el tiempo de procesado pues los cálculos se realizan sobre matrices con un tamaño significativamente menor.

Denominamos a esta zona del espacio de trabajo ROI o región de interés, y será sobre la cual se realizarán todos los análisis mencionados en apartados anteriores.

En nuestro caso presentamos dos regiones de interés, centradas en el mismo punto, pero con tamaño diferente:

- La primera región de interés, centrada en el tramo final de la cinta presenta un tamaño reducido y será la zona sobre la que se realizará el análisis de detección de movimiento o nuevos objetos.
- La segunda región de interés estará centrada en el mismo tramo final de la cinta transportadora, pero en este caso el área de la región se expandirá más hacia el inicio y final de la cinta, permitiendo un mayor rango de captación. Sobre este ROI se analizará en busca de objetos concretos y la obtención de los datos necesario para su manipulación.

El motivo por el que existen dos regiones de interés en vez de uno es debido a la presencia de un error durante el periodo de pruebas del sistema.

Cuando se realiza la detección de movimientos existe la posibilidad de que, debido al retardo entre la detección, el envío del mensaje al armario de control y la ejecución de la parada de la cinta el objeto pueda desplazarse ligeramente fuera de la región de interés.

Si utilizásemos la misma región para analizar las propiedades del objeto y este se encontrase en parte fuera de la región, al realizar los análisis de forma los resultados serían negativos, dando como resultado que se descartase el objeto por no cumplir los requerimientos durante el análisis.

Aumentando ligeramente el rango de la región hacia los extremos de la cinta, nos aseguramos de que, aunque parte del objeto no se encuentre en la región de la detección de movimiento, sí esté presente en análisis de obtención de parámetros de manipulación.

Figura 3.51 Regiones de interés de detección y análisis

Como se aprecia en las imágenes la diferencia entre el ancho de ambas regiones de interés permite utilizar un área inferior en la región de detección, disminuyendo la carga y tiempo de procesado en esta operación, y aunque no esté presente la pieza al completo, la imagen de procesado de formas la contiene en su totalidad, ofreciéndonos un análisis más fiable sobre la pieza.

3.3.9 Otras opciones evaluadas para la detección de movimiento

A la hora de desarrollar el sistema de detección de movimiento se han evaluado varias opciones sobre los tratamientos y filtros a aplicar sobre la imagen. Principalmente las opciones se han centrado en eliminar el efecto de la luz y las sombras provocadas por el brazo robot.

La idea es transformar las imágenes tanto de fondo como las tomadas en tiempo real a un sistema de representación que nos permita eliminar la luminosidad. Los sistemas ideales para este propósito son HSV (Hue-Saturation-Value) o YUE.

Estos sistemas poseen varios canales, pero, al contrario que en el caso del sistema de representación RGB, los canales representantes de la crominancia están separados de los que indican la luminosidad e intensidad del sistema.

De esta manera es posible modificar directamente la intensidad de la luz sobre la imagen, tras lo cual, devolviendo la imagen al sistema RGB y realizando la diferencia en la detección de diferencias se habrían eliminado los efectos de la intensidad de color.

Estos métodos presentaban un error fundamental y es que, siendo el color de la propia cinta transportadora de color verde, al eliminar todos los efectos de la luz y las intensidades dentro de la gama de color, la cinta presenta la misma tonalidad que la de las figuras de color verde, de forma que estas últimas quedan confundidas como parte de la propia cinta y se vuelven indetectables.

Aunque este error solo se produce en las figuras de color verde y en las demás el resultado es el óptimo, hemos preferido no hacer uso de ellas puesto que eliminaríamos de la lista de objetos detectables los de color verde.

En circunstancias en la que el color de la cinta transportadora fuera diferente al color de los objetos a detectar estos métodos ofrecerían un mejor resultado menos dependiente de las condiciones de iluminación.

3.4 Esquema de ejecución del programa

Atendiendo a las funciones y tareas que se han explicado hasta ahora, el funcionamiento y flujo del sistema una vez realizada la calibración inicial corresponde a la siguiente figura:

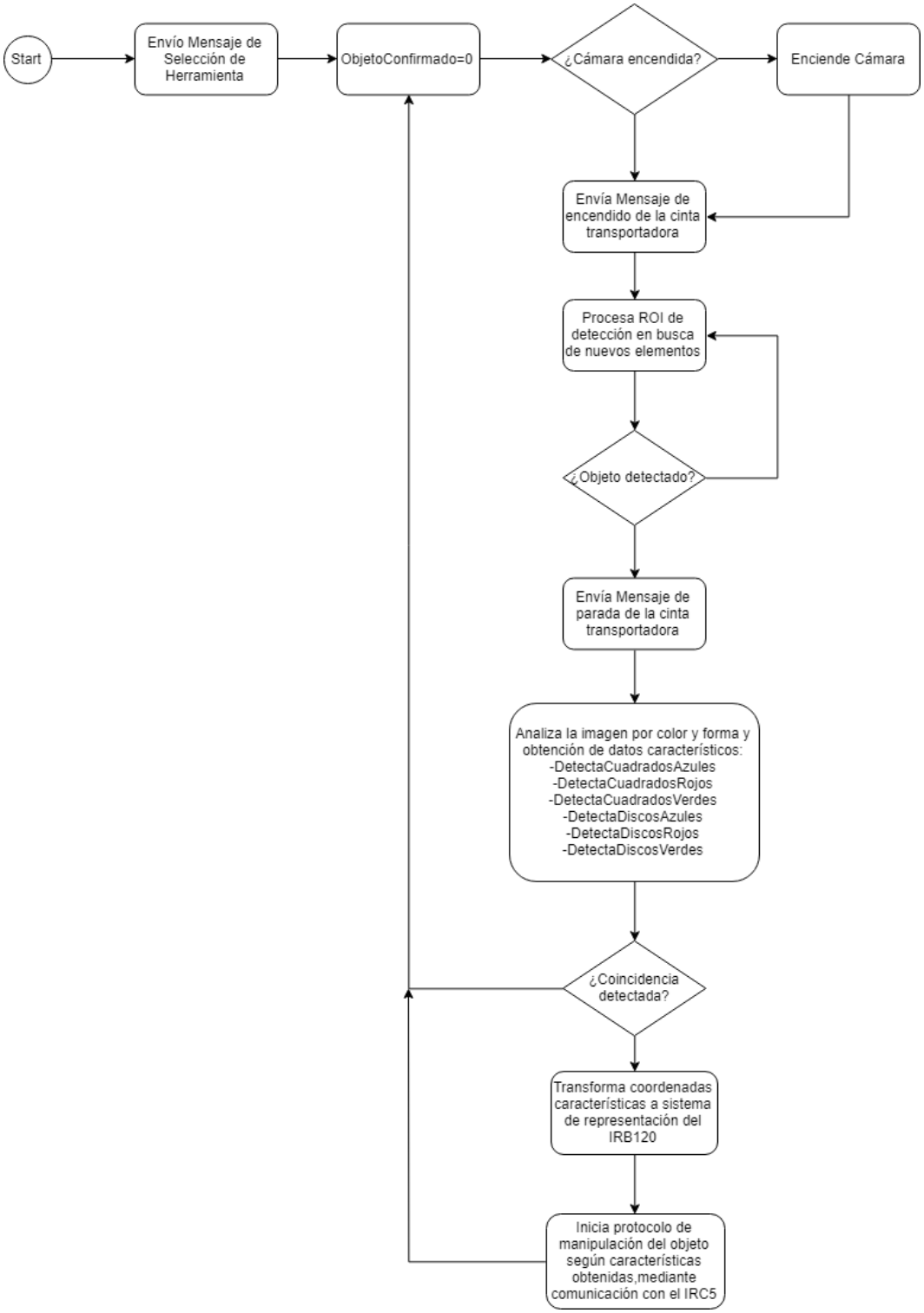

Figura 3.53 Flujograma de funcionamiento global del programa

4. Resultados y Conclusiones

Como conclusión podemos decir que el sistema desarrollado cumple las funciones encomendadas de forma efectiva atendiendo a la captación y procesado de imágenes a tiempo real, reduciendo costes mediante la eliminación del uso de sensores y otros elementos de detección y utilizando el material disponible.

Dentro de las funcionalidades y especificaciones que posee el sistema, incluyendo las ya mentadas en apartados anteriores, podemos destacar las siguientes como aquellas que hemos visto realizadas a lo largo del proyecto.

- Comunicación controlada entre el sistema de visión y el armario de control IRC5 para la ejecución de órdenes y movimientos.

-Detección de nuevos elementos de un tamaño reducido dentro de la región de interés (ROI).

-Control sobre errores en la captación de imágenes mediante bucle de seguridad.

-Análisis de imágenes para la detección de formas y colores específicos de las piezas con las que se trabaja dentro del área de trabajo.

-Análisis de las figuras detectadas en busca de parámetros de interés para su manipulación: centro de la pieza y ángulo de giro.

-Obtención de coordenadas en base a ejes del IRB120 a través de coordenadas pixélicas.

-Manipulación de piezas de trabajo con margen de error despreciable.

-Interfaz de usuario de uso fácil e intuitivo para el control del sistema.

Para cerciorar estos hechos se han realizado videos del funcionamiento del sistema, no obstante, a continuación, se añaden diversas imágenes de dichos videos en referencia al flujo de funcionamiento.

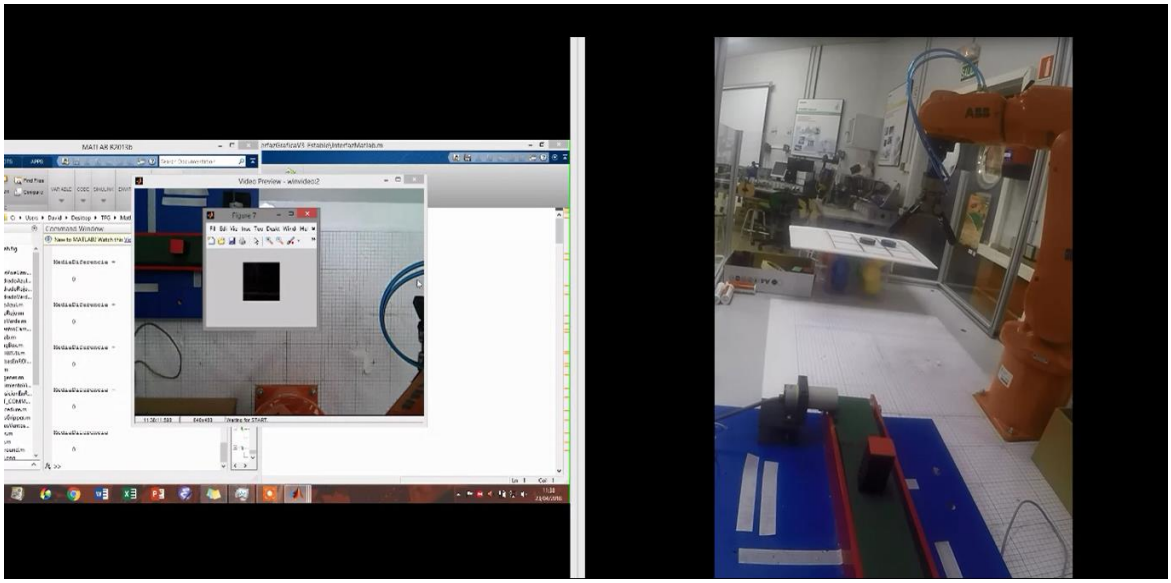

Figura 4.1 Visionado del sistema en funcionamiento 1.

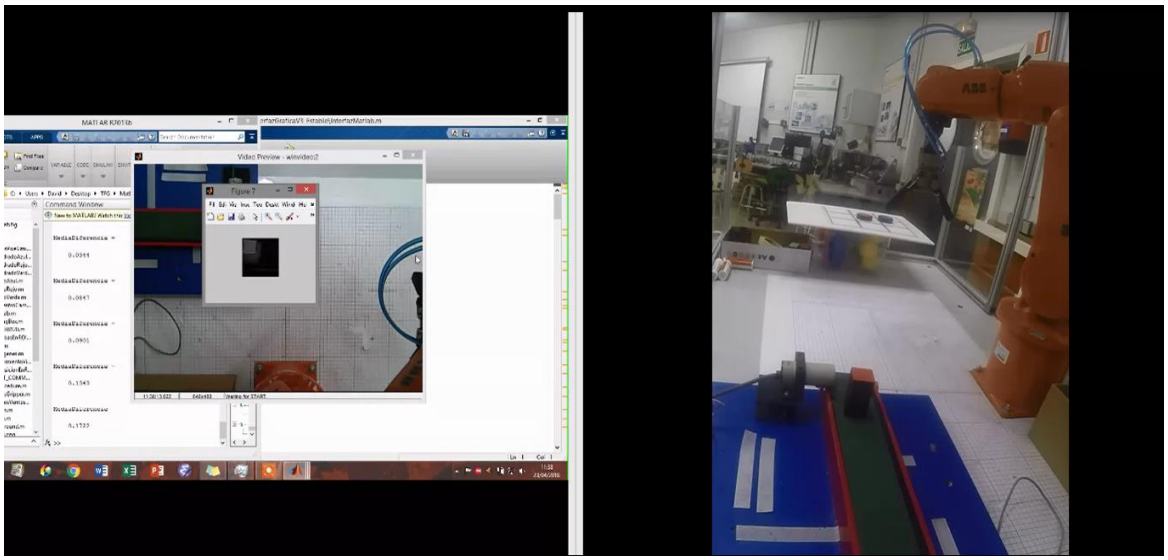

Figura 4.2 Visionado del sistema en funcionamiento 2.

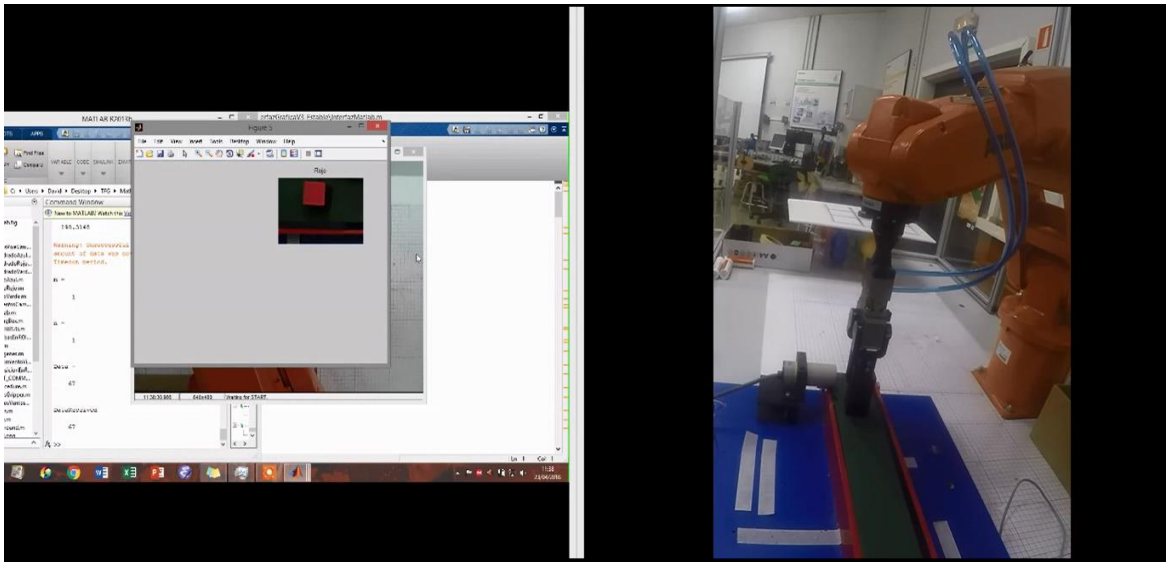

Figura 4.3 Visionado del sistema en funcionamiento 3.

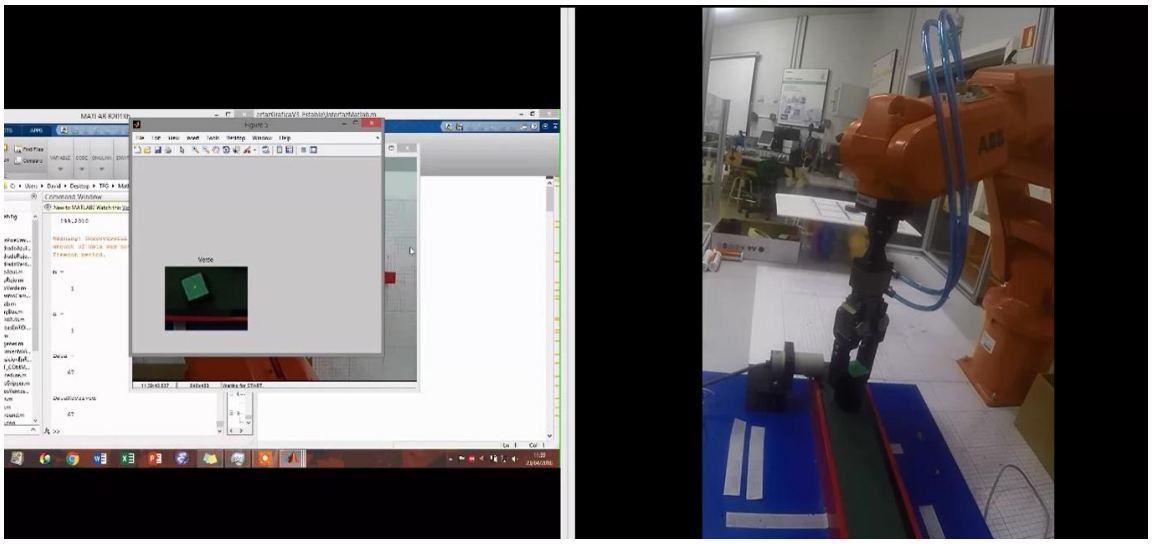

Figura 4.4 Visionado del sistema en funcionamiento 4.

Se puede mencionar también que, si bien la comunicación entre el armario de control y el ordenador, junto a los movimientos del robot pueden ralentizar ligeramente la ejecución de algunas tareas se debe tener en cuenta que el desarrollo del sistema ha priorizado la seguridad, tanto de ejecución de tareas como la del usuario, y el desarrollo del sistema de visión artificial frente al desarrollo de las comunicaciones.

Así mismo mencionar que el proyecto tiene como base directa la captación y procesamiento de imágenes, por lo que, estableciendo una relación directa entre ambas, podemos afirmar que cuanta mayor calidad posea nuestro sistema de captación de imágenes, mayor será la facilidad de procesamiento y análisis de imágenes, y con ello se obtendrían mejores resultados de precisión y eficiencia del sistema. Este hecho es destacable tras observar que la cámara utilizada captaba imágenes con desperfectos como píxeles desplazados que entorpecían o engañaban el análisis del sistema de visión.

De cara a futuros proyecto cabe mencionar que aunque el sistema de visión artificial desarrollado se centra en la manipulación de unos objetos determinados como son las cajas y los cilindros mostrados en el apartado de material, el mismo sistema puede ampliarse o modificarse para otro tipo de aplicaciones tales como, eliminar objetos no deseados o irreconocibles del espacio de la cinta transportadora, clasificar en función de formas y/o colores apilando objetos unos sobre otros o mejorar la seguridad del sistema reconociendo cuando un usuario se encuentra en la zona de trabajo para impedir que el robot pueda herirlo.

5. Manual de usuario

5.1 Introducción

A lo largo de este apartado se describirá el funcionamiento del sistema, explicando dentro de subapartados concorde a las diversas partes que lo componen, sin entrar en detalles específicos del sistema solo se aportarán los conocimientos necesarios para poder operar el sistema a nivel de usuario.

Dichos subapartados han sido divididos concorde a la línea de puesta a punto y funcionamiento del sistema y por tanto explican el orden de ejecución de las tareas llevadas a cabo tanto por el sistema como por el usuario.

Procesos como la comunicación entre el brazo robot y el sistema de visión, que se llevan a cabo de forma automática en la mayoría de los casos y ocurren en varias tareas del sistema no se han explicado aquí y se les ha dedicado una explicación más extensa en otros apartados.

En el siguiente flujograma se describe a grandes rasgos el funcionamiento del sistema, atendiendo a tareas que serán detalladas a continuación:

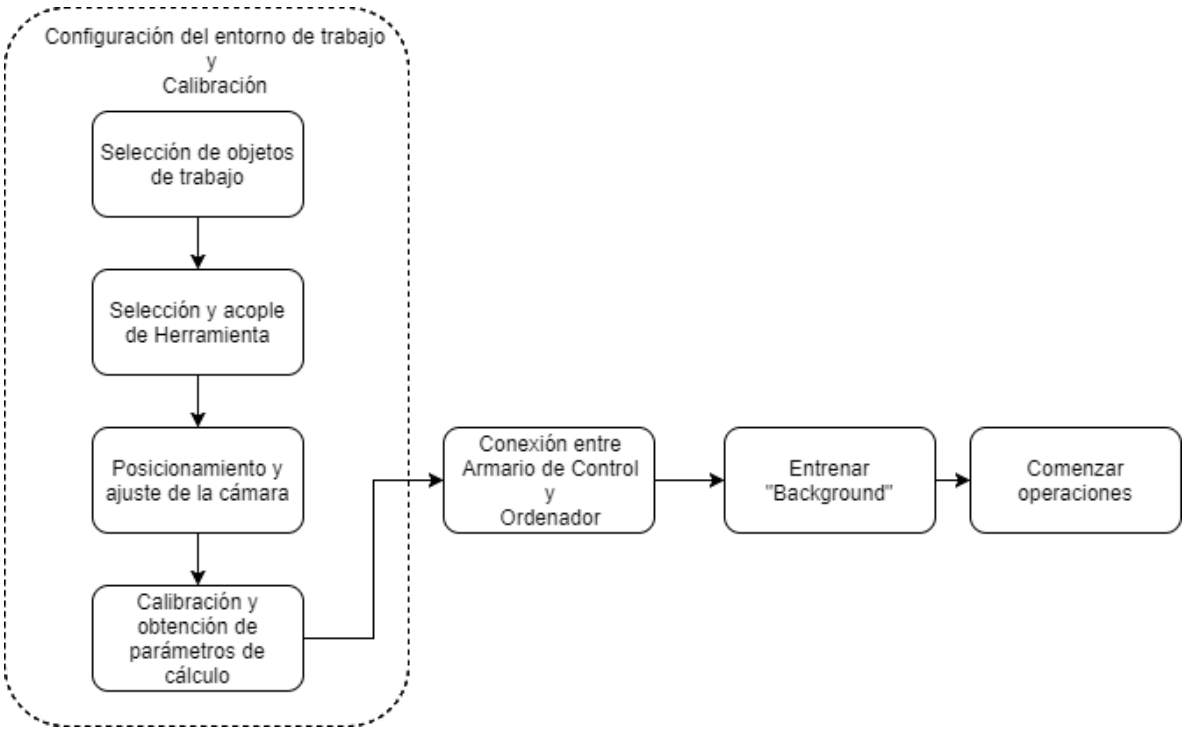

Figura 5.1 Flujo de funcionamiento del sistema

5.2 Configuración del entorno de trabajo y calibración

Antes de que el sistema comience a operar es necesario especificar algunos detalles del entorno de trabajo sobre el que se va a operar, así como preparar espacio impidiendo la presencia de cualquier obstáculo que pueda interrumpir el funcionamiento del sistema en marcha.

5.2.1 Objetos a manipular y herramienta seleccionada

Para comenzar será necesario que se sepa con qué tipo de piezas va a trabajar el sistema, cilíndricas o prismáticas, y en función de dicha elección se deberá instalar en el brazo robot el efector que más convenga a su manipulación, ya sea la ventosa o el gripper.

Para instalar cualquiera de las dos herramientas es necesario atornillarla al extremo del robot de forma que el efector quede ligado como una parte más del brazo.

Una vez situada la herramienta en el extremo del brazo hay que realizar las conexiones neumáticas a través de la electroválvula para el correcto funcionamiento de la herramienta a través de las salidas digitales del IRC5.

Atendiendo a la imagen, el tubo que lleva el aire a presión deberá ir conectado a uno de los puntos señalados, la posición superior hace que el aire circule por los conductos del gripper mientras que la posición inferior permite el funcionamiento de la ventosa.

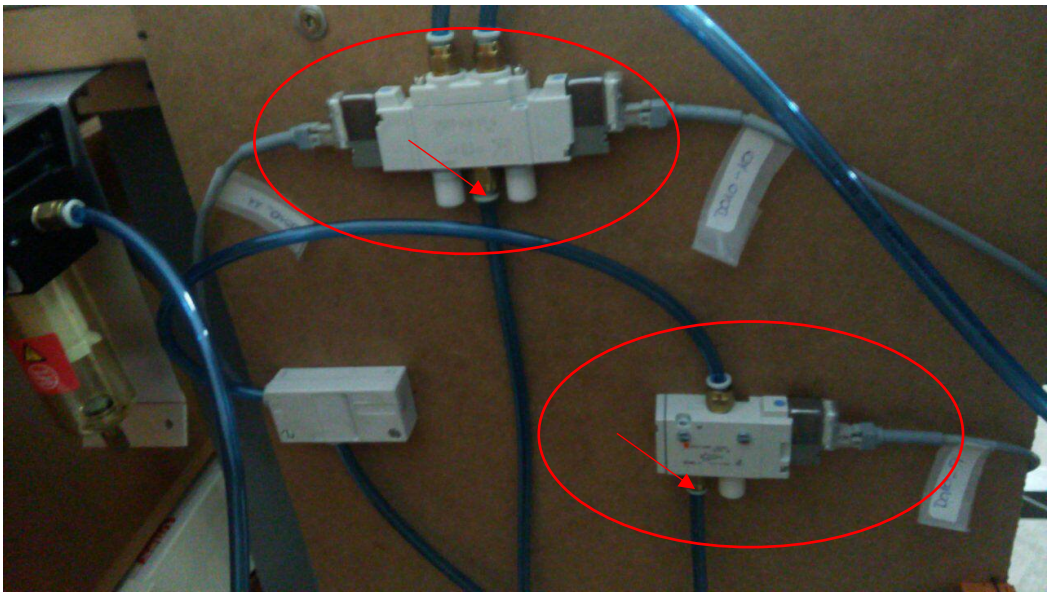

Figura 5.2 Electroválvulas y conexionado neumático

Tras tener conectada la alimentación de aire comprimido es necesario conectar el tubo de la herramienta correspondiente al canal adecuado en la parte de atrás del brazo robot.

Figura 5.3 Conexionado neumático de la parte trasera del IRB120

Los tubos están señalizados y únicamente es necesario intercambiar la primera entrada por el de la herramienta que se vaya a utilizar.

5.2.2 Posicionamiento y ajuste de la cámara

Una vez preparado el conexionado de la herramienta es el momento de calibrar el sistema, para lo cual el usuario podrá ayudarse de la interfaz gráfica creada.

Figura 5.4 Interfaz gráfica de usuario desarrollada con GUIDE

Como se puede apreciar en la imagen la interfaz cuenta con varias opciones simples, por el momento nos centraremos en aquellas dedicadas a la ayuda de la calibración, a las cuales se les ha aplicado un fondo rojo para remarcar su importancia y diferenciarlas de las demás opciones.

Para empezar, hay que comprobar que la cámara este bien orientada, en un sistema de visión artificial es esencial tener controlado aquello que capta la cámara para poder mantener un control sobre los errores que puedan surgir, así mismo también se debe conocer la posición en la que se encuentra la cámara con respecto a las coordenadas del entorno de trabajo para poder corregir errores sistemáticos debido al ángulo de visión de la cámara.

Es importante que la cuadrícula de los ejes del robot se presente en la posición más recta posible, de otra forma en el momento de traspasar los sistemas de representación en píxeles a milímetros se presentarán errores incorregibles. Al mismo tiempo la cámara debe encontrarse mirando hacia el entorno de trabajo lo más planamente posible.

Así ayudándonos del botón “Previsualizar” podremos captar y mostrar una imagen de lo que está viendo la cámara y podremos modificar su orientación conforme a lo que veamos consiguiendo ajustar manualmente la cámara lo mejor que podamos.

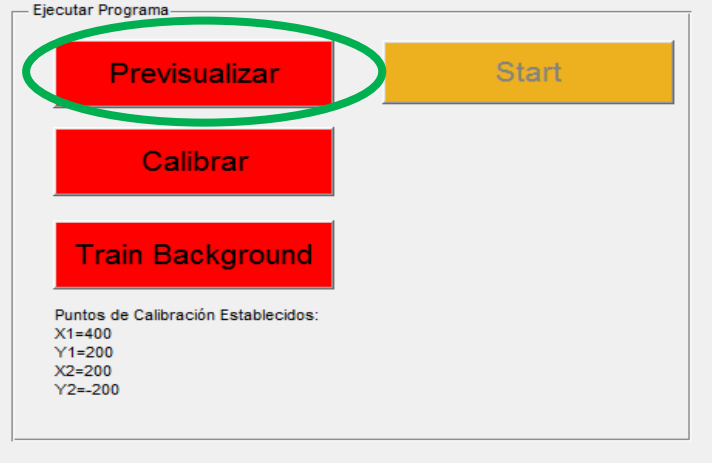

Figura 5.5 Botón “Previsualizar” de la Interfaz gráfica

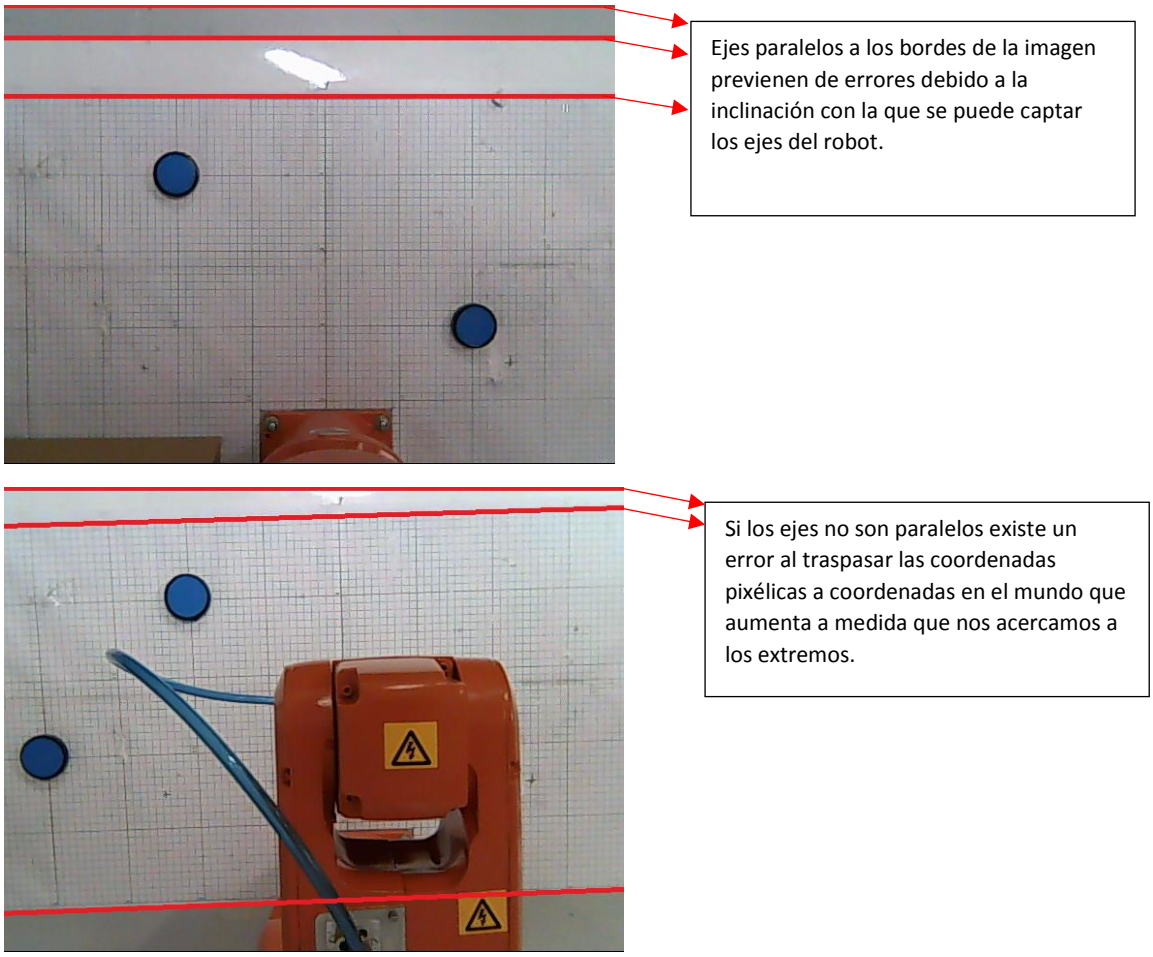

Figura 5.6 Imagen explicativa de errores por posición incorrecta de la cámara

En un sistema de visión real la posición y orientación de la cámara deberán ser fijas una vez se ha establecido una primera vez, de esta forma se podrá seguir utilizando sin realizar este tipo de calibración cada vez que se encienda el sistema, no obstante, en algunos casos no se puede asegurar que o se ha modificado el espacio de trabajo o la cámara por lo que se le ha otorgado cierta importancia a la colocación de la cámara.

Para poder visualizar aquello que capta la imagen es necesario haberla conectado previamente mediante conexión USB al ordenador de trabajo.

5.2.3 Calibración y obtención parámetros de cálculo

Una vez obtenida una buena situación de la cámara pasaremos a aportarle parámetros de cálculo al sistema de visión. Los parámetros aportados serán esencialmente la posición de dos puntos del entorno de trabajo, de esta forma conociendo las coordenadas de 2 puntos tanto en coordenadas pixélicas como en coordenadas en el sistema de representación del mundo (ejes del robot) el sistema será capaz de realizar transformaciones de un tipo de coordenadas a otras.

De igual manera que con la colocación de la cámara, la interfaz gráfica nos facilitará la calibración en gran medida:

-Primero situaremos dos discos de color azul en las coordenadas indicadas por la interfaz: P1=[400,200] y P2=[200,-200]. Es importante que los centros de ambos discos queden lo más centrados posibles sobre las coordenadas indicadas pues será las que el sistema utilizará como referencia más adelante.

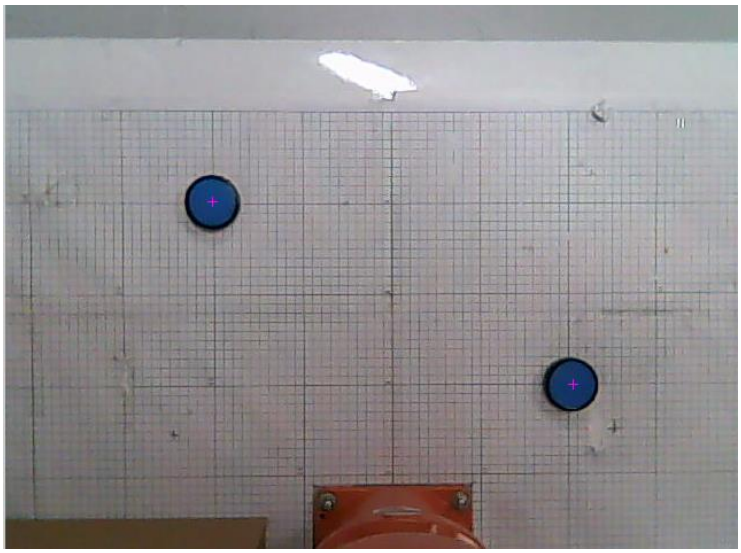

Figura 5.7 Calibrado, detección de puntos de referencia del plano

-Tras ello se utilizará el botón “Calibrar” de la interfaz gráfica y el sistema se ocupará de obtener los datos, ejecutando la función “Calibrando”.

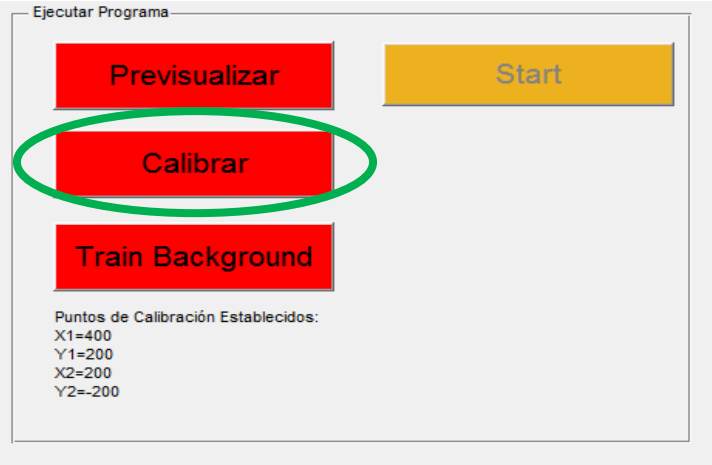

Figura 5.8 Botón “Calibrar” de la Interfaz de usuario

Hecho esto, el sistema posee los parámetros necesarios para comenzar por lo que se puede proceder a establecer la maqueta dentro del área de trabajo. Puesto que se entiende que en un entorno de trabajo industrial la situación de una cinta transportadora permanecerá fija en una posición hemos decidido situar la maqueta con la esquina inferior derecha en una posición x=270 sobre los ejes del robot, e “y” comprendida entre 80 y 120.

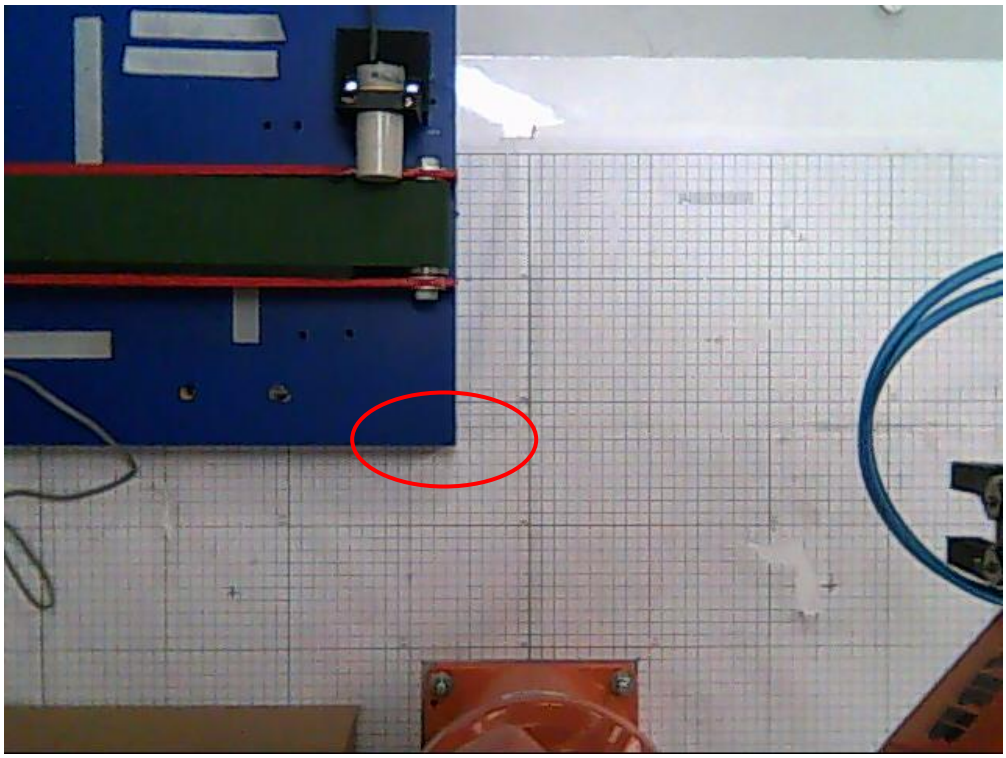

Figura 5.9 Posicionamiento de la cinta transportadora

Es importante que el espacio de trabajo no posea obstáculos para el movimiento del brazo robot más allá de la propia maqueta y los objetos que circulen por ella, de otra forma el brazo podría colisionar con ellos a lo largo de sus trayectorias.

También es destacable el hecho de que, al iniciar el sistema, éste cargará los datos de la última calibración realizada por lo que si se tiene constancia de que nada en el entorno de trabajo ha sido modificado se podrá obviar todo el procedimiento hasta este punto y proceder directamente con la cinta ya situada.

5.3 Conexión Armario de Control con Ordenador

Con todo preparado dentro del entorno de trabajo procederemos a conectar el ordenador de trabajo con el armario de control mediante la comunicación TCP/IP, para ello seguiremos los siguientes pasos:

- Primero nos aseguraremos de dar alimentación al armario de control IRC5, puesto que si el robot no escucha las peticiones de conexión desde Matlab el sistema puede llegar a bloquearse.
- Tras ello ejecutaremos el código desarrollado en RAPID, el cuál automáticamente llevará el robot a una posición inicial y se quedará a esperas de recibir una petición

de conexión por parte del ordenador. Para ello solo será necesario que el programa se halle cargado y se pulse comenzar desde el mando FlexPendant.

- Para llevar a cabo dicha petición solo tendremos que pulsar el botón "Conectar" de la interfaz gráfica. Es necesario que antes de pulsarlo el usuario se asegure de que las opciones de herramienta y tipo de conexión hayan sido marcadas correctamente pues se deshabilitarán una vez se pulse el botón.

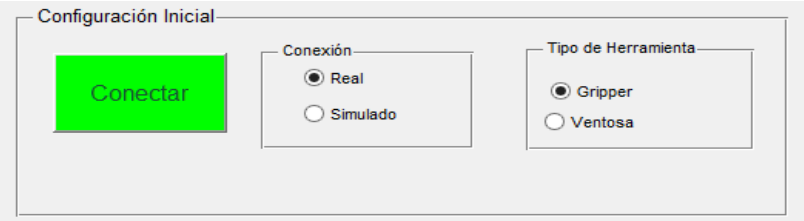

Figura 5.10 Opciones de configuración y botón "Conectar"

- Tras pulsar "Conectar" deberemos esperar a que se complete la creación del socket y se lleve a cabo un protocolo de envío y recepción entre el ordenador y el IRC5. Cuando finalice correctamente se habrá recibido un mensaje de "Conexión establecida" que indicará que el ordenador ya puede comunicarse con el robot, se deshabilitará la elección del tipo de herramienta y conexión y se habilitarán los botones "Train Background" y "Start".

5.4 Entrenar "Background"

Antes de comenzar la operación por completo es necesario una última obtención de datos esencial para la detección de la entrada de objetos en el área de trabajo: la obtención de un "fondo" sobre el que comparar las nuevas imágenes a tiempo real que tome la cámara.

Mediante el uso del botón "Train Background" el ordenador comunicará al armario de control la orden de comenzar el movimiento de la cinta. Tras ello se captarán imágenes del entorno durante 20 segundos y se obtendrá el "fondo" deseado y volverá a parar la cinta.

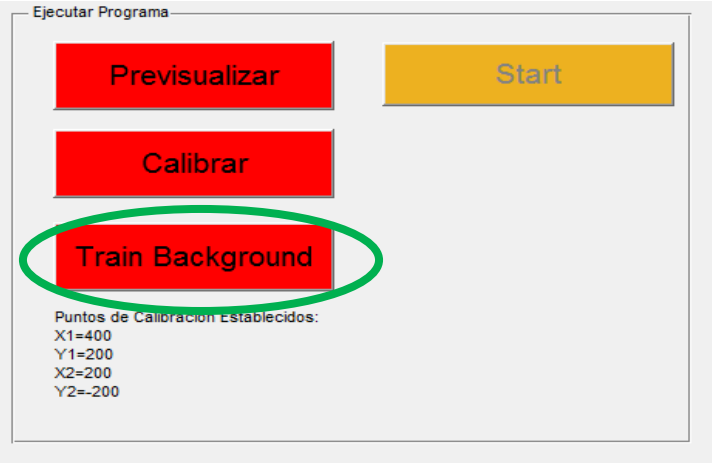

Figura 5.11 Botón "Train Background" de la Interfaz de usuario

5.5 Comenzar operaciones

Con todos los parámetros listos podemos comenzar las operaciones pulsando el botón "Start". Cuando este botón se pulse comenzará la operación ininterrumpida del sistema.

El ordenador comunicará la orden al robot de activar la cinta transportadora e iniciará la cámara para monitorear la actividad sobre el entorno de trabajo. Más concretamente el sistema analizará la entrada de objetos en el tramo final de la cinta, donde esperamos la entrada de objetos, de esta manera el entorno a analizar es menor y supondrá una menor carga de procesamiento, consiguiendo un tiempo de procesado más corto que permitirá alcanzar una mayor tasa de "frames" por segundo analizados.

Cuando el sistema detecta una diferencia de la imagen tomada a tiempo real con la tomada durante el entrenamiento de fondo comunicará la parada de la cinta y tomará una nueva imagen en parado que analizará para confirmar la presencia de un objeto de trabajo.

Si la diferencia es detectada como objeto de trabajo se obtendrán una serie de parámetros para su manipulación y se comunicará al robot todos los movimientos que debe realizar para agarrar el objeto y dejarlo en el lugar que corresponda.

Es conveniente mencionar que todos los movimientos del robot, así como el control de la maqueta, a pesar de ser efectuados por el armario de control son completamente controlados desde el ordenador de trabajo mediante Matlab, haciendo innecesaria la manipulación de código RAPID y por tanto la recarga del programa en el armario de control a no ser qué se quieran implementar nuevas funcionalidades.

Al finalizar la manipulación del objeto se situará el brazo en su zona de reposo y se retomará la marcha de la cinta transportadora repitiendo el proceso hasta que se apague el sistema.

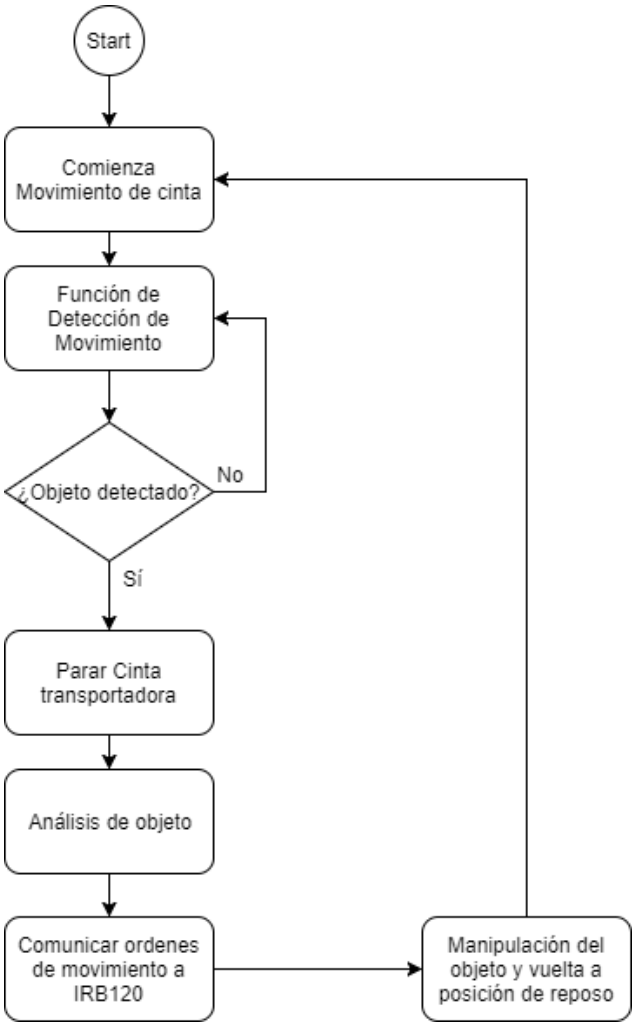

Figura 5.12 Flujograma de funcionamiento del sistema en marcha

6. Descripción de ficheros y funciones

6.1 Ficheros y funciones de Matlab

Conectar:
Función:

Conectar(RealSimulado)

Entradas:

-RealSimulado: '1'= Conexión a simulación; '0'=Conexión real

Salidas:

-Sin salidas

Descripción:

Se llama desde la Interfaz Gráfica pulsando el botón "Conectar", en función de la variable "RealSimulado" utiliza las funciones de la clase "irb120" para conectarse con el robot simulado o real.

Calibrando:
Función:

-

Entradas:

-Sin entradas

Salidas:

-Sin salidas

Descripción:

Se llama desde la Interfaz Gráfica pulsando el botón "Calibrar": esta función detecta los centros de 2 discos azules utilizados para crear un plano de referencia sobre el cual realizar transformaciones del sistema pixélico al sistema de coordenadas del robot.

CoordenadasPixel2milimetros:
Función:

[PosX, PosY]=CoordenadasPixel2milimetros(CentroPixel,AlturaObjeto)

Entradas:

-CentroPixel: Posición en pixeles del punto a localizar: [Px, Py]

-AlturaObjeto: Altura que presenta la cara superior del objeto

Descripción:

Esta función transforma las coordenadas pixélicas de un punto en coordenadas en base al robot, incluyendo la corrección por altura del objeto. Para ejecutar esta función es necesaria la obtención de datos mediante calibración.

DetectaCuadradoAzul

Función:

[Angulo, Coincidencia, Centro] = EncuentraCuadradoAzul(ROI)

Entrada:

-ROI: región de interés donde detectar el cuadrado azul

Salida:

-Angulo: inclinación de la pieza respecto al eje X de la imagen

-Coincidencia: número de cuadrados azules detectados

-Centro: centro en coordenadas pixélicas del cuadrado detectado

Descripción:

Detecta los prismas azules presentes en la región de interés y obtiene los parámetros necesarios para su manipulación.

DetectaCuadradoRojo

Función:

[Angulo, Coincidencia, Centro] = EncuentraCuadradoRojo(ROI)

Entrada:

-ROI: región de interés donde detectar el cuadrado rojo

Salida:

-Angulo: inclinación de la pieza respecto al eje X de la imagen

-Coincidencia: número de cuadrados rojos detectados

-Centro: centro en coordenadas pixélicas del cuadrado detectado

Descripción:

Detecta los prismas rojos presentes en la región de interés y obtiene los parámetros necesarios para su manipulación.

DetectaCuadradoVerde

Función:

[Angulo, Coincidencia, Centro] = EncuentraCuadradoVerde(ROI)

Entrada:

-ROI: región de interés donde detectar el cuadrado rojo

Salida:

-Angulo: inclinación de la pieza respecto al eje X de la imagen

-Coincidencia: número de cuadrados verdes detectados

-Centro: centro en coordenadas pixélicas del cuadrado detectado

Descripción:

Detecta los prismas verdes presentes en la región de interés y obtiene los parámetros necesarios para su manipulación.

DetectaDiscoAzul

Función:

[Coincidencia, Centro]= DetectaDiscoAzul(ROI)

Entrada:

-ROI: región de interés donde detectar el disco

Salida:

-Coincidencia: número de discos azules detectados

-Centro: centro en coordenadas pixélicas del disco detectado

Descripción:

Detecta los discos azules presentes en la región de interés y obtiene los parámetros necesarios para su manipulación.

DetectaDiscoRojo

Función:

[Coincidencia, Centro]= DetectaDiscoRojo(ROI)

Entrada:

-ROI: región de interés donde detectar el disco

Salida:

-Coincidencia: número de discos rojos detectados

-Centro: centro en coordenadas pixélicas del disco detectado

Descripción:

Detecta los discos rojos presentes en la región de interés y obtiene los parámetros necesarios para su manipulación.

DetectaDiscoVerde

Función:

[Coincidencia, Centro]= DetectaDiscoVerde(ROI)

Entrada:

-ROI: región de interés donde detectar el disco

Salida:

-Coincidencia: número de discos verdes detectados

-Centro: centro en coordenadas pixélicas del disco detectado

Descripción:

Detecta los discos verdes presentes en la región de interés y obtiene los parámetros necesarios para su manipulación.

InterfazMatlab

Función:

-

Entradas:

-Sin entradas

Salidas:

-Sin salidas

Descripción:

Interfaz Gráfica de Mataba través de la cual se controlan las opciones del robot, sus funcionalidades se explican a lo largo de la memoria y son utilizadas para apoyar a los usuarios a operar correctamente el sistema sin necesidad de conocimientos específicos.

MinBoundingBox

Función:

bb = minBoundingBox(X)

Entradas:

-X= Matriz de puntos que se desea "rodear"

Salidas:

-bb= Matriz que contiene la situación de las 4 esquinas que conforman el mínimo cuadrado que contiene el contorno introducido

Descripción:

Función que haya las esquinas del mínimo cuadrado que envuelve una serie de puntos (en este caso un Borde-Contorno), aportada por el usuario "Julien Diener" en la comunidad de usuarios de MathWorks[4].

RecibirDatoIRB120

Función:

Data=RecibirDatoIRB120(r)

Entradas:

- r = objeto creado de la clase "irb120" que posea una conexión ya creada

Salidas:

-Data: byte de confirmación recibido desde el IRC5

Descripción:

Función que utiliza la clase IRB120 para recibir bytes de confirmación provenientes del socket del Robot con el que se ha conectado(Simulado o Real).

DetectaMovimientoVisión
Función:
-
Entradas:
-Sin entradas
Salidas:
-Sin salidas
Descripción:
Proceso que desarrolla en análisis en busca de la entrada de nuevos elementos en la región de interés, indica el inicio de la cinta transportadora en su inicio y su parada cuando detecta la entrada de un nuevo elemento.

ProcesaCajasGripper
Función:
-
Entradas
-Sin entradas
Salidas:
-Sin salidas
Descripción:
Función que comienza el funcionamiento del sistema cuando la herramienta seleccionada es el gripper. Manipula las funciones de detección y análisis de objetos controlando el funcionamiento autómata del IRB120 mediante mensajes a través de la comunicación TCP/IP. Establece el flujo de control para la manipulación y detección de prismas.

ProcesaDiscosVentosa
Función:
-
Entradas
-Sin entradas
Salidas:
-Sin salidas
Descripción:
Función que comienza el funcionamiento del sistema cuando la herramienta seleccionada es la ventosa. Manipula las funciones de detección y análisis de objetos controlando el funcionamiento autómata del IRB120 mediante mensajes a través de la comunicación TCP/IP. Establece el flujo de control para la manipulación y detección de discos

.

IB120

Descripción:

Clase que contiene las variables y funciones necesarias para el envío de instrucciones al IRC5 para llevar a cabo la manipulación del IRB120 desde el ordenador.

Código RAPID

Fichero TCPIPConnectionHandler

```
MODULE TCPIPConnectionHandler
VAR socketdev socketServer;
VAR socketdev socketClient;
VAR bool      connectionStatus := FALSE;
VAR string    data2ClientType;
LOCAL VAR num       initialVel{2} := [200, 200];
LOCAL VAR string    ipAddress;
LOCAL VAR num       portNumber := 1024;
LOCAL VAR num       err_counter := 0;
   PROC EstablishConnection()
      !============ Default values ==============
      ToolChoiceProcedure 1;
      VelocityChoiceProcedure initialVel;
      PrecisionChoiceProcedure 1;
      pos_limitation.trans.z := 0;  !Define the limitation for the Z axis
      pos_limitation.trans.x := 0;  !Define the limitation for the X axis
      !=========================================

      ipAddress := "172.29.28.185";  !172.29.28.185 for the real station
                        !127.0.0.1 for simulaiton
      SocketCreate socketServer;
      SocketBind socketServer, ipAddress, portNumber;
      SocketListen socketServer;
      SocketAccept socketServer, socketClient, \ClientAddress:=ipAddress, \Time:=WAIT_MAX;
      connectionStatus := TRUE;
      Target_xyz := CRobT(\Tool:=tool \WObj:=wobj0);
      data2ClientType := "ONLINE";
      SendDataProcedure 0;
      !EDIT SendDataProcedure 1;
    ERROR    !The timeout has been temporary disabled (see \Time:=WAIT_MAX instruction
above)
      IF ERRNO=ERR_SOCK_TIMEOUT THEN
         Incr err_counter;
         IF err_counter = 3 THEN
            ErrWrite "Fatal connection error", "Total waiting time ERROR"
               \RL2:="Connection re-try failure."
               \RL3:="Maxiumum attempts to connect a client = 3";
            EXIT;
         ELSE
            ErrWrite \W, "Connection error", "Waiting time ERROR."
               \RL2:="If the max. waiting time (120s) has been exceeded"
```

```
            \RL3:="the error handler returns the warning."
            \RL4:="SOLUTION: Check if the client program is connected to the socket.";
          RETRY;
        ENDIF
      ENDIF
    ENDPROC
ENDMODULE
```

Fichero SendDataHandler

```
MODULE SendDataHandler

    LOCAL VAR num          j;
    LOCAL VAR jointtarget   currentJoint;
    LOCAL VAR num          currentJointMatrix{6};
    LOCAL VAR string       str_x;
    LOCAL VAR string       str_y;
    LOCAL VAR string       str_z;
    LOCAL VAR string       PosMatrix{3};
    LOCAL VAR num          strLength{3};
    LOCAL VAR byte         byteCPos{3};
    LOCAL VAR byte         byteCPosPart{3};
    LOCAL VAR pos          currentPos;
    LOCAL VAR num          eulerAngles{3};
    VAR jointtarget        check_reachability;
    VAR byte               Data2Client{28};
    PROC SendDataProcedure(byte SendMode)
      TEST SendMode
      CASE 0:
        WaitTime 1;
        Data2Client{1} := 42;
        SocketSend socketClient \Data:=Data2Client, \NoOfBytes:=1;
        SocketReceive socketClient \Data:=receivedData \ReadNoOfBytes:=1 \Time:=
WAIT_MAX;
        IF receivedData{1} = 1 THEN !!IF receivedData{1} = 0 THEN
          SocketSend socketClient \Str:="Connection has been established successfully";
        ENDIF
/***********************************************************/
/***********************************************************/
/***********************************************************/
       CASE 3:
      WaitTime 1;
      !Enviar mensaje de objeto detectado al sistema de vision
      Data2Client{1}:=67;
      SocketSend socketClient \Data:=Data2Client, \NoOfBytes:=1;
      !SocketReceive socketClient \Data:=receivedData \ReadNoOfBytes:=1 \Time:= WAIT_MAX;
      ENDTEST
      ERROR
      IF ERRNO=ERR_SOCK_CLOSED THEN
        connectionStatus := FALSE;
```

```
        SocketClose socketServer;
        WaitTime 2;
        SocketClose socketClient;
        WaitTime 2;
        main;
      ENDIF
   ENDPROC
ENDMODULE
```

Fichero IRB120_Control

```
MODULE IRB120_Control
VAR num      receivedData{19};
VAR jointtarget axis_pos := [ [0, 0, 0, 0, 0, 0], [ 0, 9E9, 9E9, 9E9, 9E9, 9E9] ];
VAR jointtarget jtar;
VAR orient effector_orient;
VAR robtarget Target_xyz;
VAR robjoint ax;
VAR num axMatrix{6};
VAR num rotMatrix{3};
VAR num posMatrix{3};
VAR num limitMatrix{2};
VAR num velocityMatrix{2};
VAR num n;
VAR bool confFirstError;
VAR num cnfNum;
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
VAR num HerramientaSeleccionada:=0; !Gripper=1 //Ventosa=2
VAR num Orden_Coger_Soltar:=1;!Coger/Cerrar/Activar=0 Soltar/Abrir/Desactivar=1

   PROC DataProcessingAndMovement()!FUNCION ORIGINAL, INTENTAR NO USAR
      WHILE connectionStatus DO
         SocketReceive socketClient \Data:=receivedData \ReadNoOfBytes:=19 \Time:=
WAIT_MAX;   !Receive data from the socket
         stringHandler(receivedData);  !Handle the received data string
      ENDWHILE
    ERROR
      IF ERRNO=ERR_SOCK_CLOSED THEN
         connectionStatus := FALSE;
         SocketClose socketServer;
         WaitTime 2;
         SocketClose socketClient;
         WaitTime 2;
         main;
      ENDIF
   ENDPROC

   PROC stringHandler (num Data2Process{*})
      n := 0;
```

```
Target_xyz := CRobT(\Tool:=tool \WObj:=wobj0);

TEST Data2Process{1}
CASE 1: !MOVIMIENTO DE LOS EJES
            /*…*/
CASE 2: !ROTACION
            /*…*/
CASE 3: !POSICION
            /*…*/
 CASE 4:  !MOVIMIENTO A OBJETOS
            /*…*/
 CASE 5: !CONTROL DE VELOCIDAD
            /*…*/

  CASE 6: !ACCIÓN DE VENTOSA
           /*…*/
  CASE 7: !Elegir Herramienta Gripper O Ventosa
     IF Data2Process{2} = 0 THEN !Sin Herramienta Seleccionada

     ELSEIF Data2Process{2} = 1 THEN !GRIPPER SELECCIONADO
        HerramientaSeleccionada :=1;
        SetDO DO10_7,1;
        SetDO DO10_8,0;
     ELSEIF Data2Process{2} = 2 THEN !VENTOSA SELECCIONADO
        HerramientaSeleccionada :=2;
        SetDO DO10_7,0;
        SetDO DO10_8,1;
     ENDIF
     SendDataProcedure (3);
  CASE 8: !MOVER CINTA//PARAR CINTA
  IF Data2Process{2} = 0 THEN!Orden Activar Cinta
   !Activar Cinta transportadora
   SetDO DO10_1,1;!Led de Informa Visual
   SetDO DO10_15,1;
  ELSEIF Data2Process{2} = 1 THEN!Orden Parar Cinta
   !Desactivar Cinta transportadora
   SetDO DO10_1,0;!Led de Informa Visual
   SetDO DO10_15,0;
  ENDIF
   !Send Object Detected
   ! SendDataProcedure (3);
   !////////////////////////CASE 9/////////////////////////////////////////////
   CASE 9: !COGER/DEJAR Objeto en Posicion Indicada
     !Abrir Gripper
     !Proccess Message of Position Move
     FOR i FROM 2 TO 6 STEP 2 DO     !Evalua el datagrama de posicion y rotacion
        Incr n;
        IF Data2Process{i} = 0 THEN !Evalua el signo de la rotacion
           rotMatrix{n} := -Data2Process{i+1};
        ELSEIF Data2Process{i} = 1 THEN
           rotMatrix{n} := Data2Process{i+1};
        ENDIF
     ENDFOR
```

```
            n := 0;
            FOR i FROM 8 TO 14 STEP 3 DO
               Incr n;
               IF Data2Process{i} = 0 THEN  !Evalua el signo de la posicion
                  posMatrix{n} := -(100*Data2Process{i+1} + Data2Process{i+2});
               ELSEIF Data2Process{i} = 1 THEN
                  posMatrix{n} := 100*Data2Process{i+1} + Data2Process{i+2};
               ENDIF
            ENDFOR

            effector_orient := OrientZYX(rotMatrix{1}, rotMatrix{2}, rotMatrix{3});
            Target_xyz.rot := effector_orient;
            Target_xyz.trans.x := posMatrix{1};
            Target_xyz.trans.y := posMatrix{2};
            Target_xyz.trans.z := posMatrix{3};

            !====== Security procedure (reachability) ======
            CheckReachabilityXYZ posMatrix;
            !================================================
            IF statusAVAILABLE THEN
               SingArea \Wrist;
               jtar := CalcJointT (Target_xyz, tool\WObj:=wobj0);
               ConfJ \Off;
               MoveL Target_xyz,vel,fine,tool\WObj:=wobj0; !tcp_prec (future use) !MODIFICADO
!MoveAbsJ jtar, vel, fine, tool;     !tcp_prec (future use)
!               SendDataProcedure 1;
            ELSEIF statusAVAILABLE = FALSE THEN
               SendDataProcedure 2;
            ENDIF
            !Mover a posicion intermdedia
            Orden_Coger_Soltar:=Data2Process{17};
            TEST Orden_Coger_Soltar
               CASE 0:
                  IF HerramientaSeleccionada = 1 THEN
                     !Close Gripper
                     SetDO DO10_2,1;
                      SetDO DO10_11,0;!NO Abrir Gripper
                     SetDO DO10_10,1;!SI Cerrar Gripper
                     WaitTime 2;
                     SetDO DO10_2,0;
                  ELSEIF HerramientaSeleccionada = 2 THEN
                     !Activar Ventosa
                     SetDO DO10_9,1;
                  ENDIF
               CASE 1:
                  IF HerramientaSeleccionada = 1 THEN
                     !Open Gripper
                     SetDO DO10_3,1;
                     SetDO DO10_10,0;!SI Cerrar Gripper
                     SetDO DO10_11,1;!NO Abrir Gripper
                     WaitTime 2;
                     SetDO DO10_3,0;
                  ELSEIF HerramientaSeleccionada = 2 THEN
```

```
                    !Desactivar Ventosa
                    SetDO DO10_9,0;
                ENDIF
            ENDTEST
            SendDataProcedure (3);
        !////////////////////////////////////////////////////////////////
        ENDTEST
         /*Protocolo anti-errores*/
   ENDPROC
   ENDPROC
ENDMODULE
```

7. Pliego de condiciones

A continuación se describen las características de las herramientas utilizadas o necesarias para disponer del correcto funcionamiento del sistema:

Hardware:

1. Ordenador portátil ASUS F552C:
   -Procesador Intel(R) Core(TM) i5-3337U CPU 1.8GHz
   -Tarjeta Gráfica NVDIA GEFORCE 710M
   -Memoria RAM 8 GB
2. Brazo robótico IRB120
   -Controlador IRC5
   -Peso: 25Kg
   -Posibilidad de levantar cargas hasta 3Kg
   -6 grados de libertad
3. Cámara Logitech C310
   -Resolución de 640x480 píxeles
4. Sistema neumático
   -Compresor
   -Electroválvulas
   -Sistema de succión
   -Conexionado
5. Herramientas/Efectores del brazo robótico
   -Gripper
   -Ventosa

Software:

1. Sistema operativo Windows 8.1
2. Matlab R2013b
3. ABB RobotStudio
4. Microsoft Office 2017

8. Presupuesto

A continuación, se exponen los costes teóricos asociados al desarrollo del proyecto, dividiéndolos en costes materiales y profesionales:

Costes materiales

Material	Cantidad	Coste Ud. (€)	Coste TOTAL
Cámara Logitech C310	1	40	40
Robot IRB120	1	10945	10945
Matlab R2013b	1	-	-
RobotStudio ABB (4 meses)	1	-	-
ASUS F552C	1	650	650
Herramienta Gripper	1	-	-
Herramienta Ventosa	1	-	-
TOTAL			11635

Tabla 8.1 Tabla de costes materiales del proyecto

Costes humanos/profesionales:

Trabajo	Tiempo (mes)	Coste Unitario (€/mes)	Coste TOTAL (€)
Ingeniería	4	1400	5600
TOTAL			5600

Tabla 8.2 Tabla de costes profesionales

Costes totales:

Motivo	Coste (€)	IVA (€)	Coste TOTAL (€)
Material	11635	2443.35	14078.35
Tasa profesional	5600	1176	6776
TOTAL			20854.35

Tabla 8.3 Tabla de costes totales

9. Bibliografía

[1] TFG:"Diseño de una estación de trabajo para el Robot IRB120. Control de cinta transportadora mediante IRC5" de Andrés Senén Estremera.

[2]TFG: "Manipulación de objetos en una cinta transportadora mediante un sistema de visión artificial y brazo robot IRB120" de Javier Ribera Díaz

[3] TFG:"Socket based communication in RobotStudio for controlling ABB-IRB120 robot. Design and development of a palletizing station", de Marek Jerzy Frydrysiak

[4] https://es.mathworks.com/matlabcentral/fileexchange/31126-2d-minimal-bounding-box?focused=5188215&tab=function

[5] MathWorks.com

[6] Manual operador RobotStudio de ABB

[7] Manual de operador de RAPID de ABB

[8] Manual de operador de IRC5 de ABB

[9] Diapositivas y documentación educativa aportada por los profesores de la asignatura de Sistemas Robotizados

[10] TFG: "Desarrollo de una interfaz para el control del robot IRB120 desde Matlab". Azahara Gutiérrez Corbacho. Trabajo Fin de Grado.

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR