ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO ELECTROMECÁNICO

INTEGRACIÓN DE VISIÓN ARTIFICIAL EN UN ROBOT INDUSTRIAL

Autor: Lucía Díaz García
Director: José Antonio Rodríguez-Mondéjar
Jaime Boal Martín-Larrauri

Madrid

Julio 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. LUCÍA DÍAZ GARCÍA

DECLARA ser el titular de los derechos de propiedad intelectual de la obra: "INTEGRACIÓN DE VISIÓN ARTIFICIAL EN UN ROBOT INDUSTRIAL", que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra lòs derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

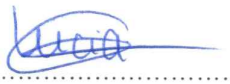
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ...24... de .....JUNIO..... de .2018

ACEPTA



Fdo.....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título: "Integración de visión artificial en un robot industrial" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2017-2018 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Lucía Díaz García Fecha: 23 / 07 / 2018

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fdo.: José Antonio Rodríguez Mondéjar Fecha: 23 / 07 / 2018

Fdo.: Jaime Boal Martín-Larrauri Fecha: 23 / 07 / 2018

Agradecimientos:

A mis directores de proyecto José Antonio y Jaime por su ayuda y dedicación.

A mis padres por la paciencia y la confianza todos estos meses.

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
INGENIERÍA ELECTROMECÁNICA

INTEGRACIÓN DE VISIÓN ARTIFICIAL EN UN ROBOT INDUSTRIAL

Autor: Díaz García, Lucía

Directores: Rodríguez Mondejar, José Antonio

Boal Marín-Larrauri, Jaime

Entidad colaboradora: ICAI –Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto persigue conseguir el control de un robot industrial IRB 120 de la firma ABB mediante un sistema de visión artificial. El fin es la clasificación de piezas LEGO y su distinción por tamaño y color, y la actuación automática del brazo robótico para desmontar las estructuras iniciales en las que se encontraban las piezas y ordenarlas.

Figura 1. Arquitectura del sistema

En la figura 1 se muestran las conexiones del sistema. La cámara encargada de la toma de imágenes se encuentra conectada a un ordenador donde se procesan las imágenes mediante técnicas de visión artificial, se preparan los datos y se envían al robot. El envío de datos entre Matlab y el robot se consigue gracias a sockets de internet.

RESUMEN

Se puede dividir el proyecto en tres áreas diferentes de trabajo; la primera consiste en la captación y procesado de imágenes para la obtención de información del entorno. En segundo lugar, la creación de una comunicación entre Matlab y el controlador del robot para el envío de datos. Por último, tiene lugar la programación del movimiento del robot en función de los datos recibidos.

La captación de imágenes se realiza mediante la cámara Intel RealSense D435. Esta es una cámara del tipo RGB-D, es decir, es capaz de capturar imágenes a color e imágenes de profundidad. Ambas imágenes son necesarias para la clasificación de las piezas; la primera, para separarlas por color y tamaño y conocer sus coordenadas, y la segunda, para conocer la altura a la que se encuentra cada pieza clasificada. El procesado de imágenes se lleva a cabo en Matlab, y una vez terminado el proceso se conoce la posición de todas las piezas presentes en coordenadas reales y sus características.

En segundo lugar, es necesario crear una comunicación entre Matlab y el robot para el envío de información de modo que el controlador del robot pueda interpretarla y actuar conforme a ella; esto se realizará mediante el envío de un vector de información mediante sockets de internet. El primer dato que se envía es un identificador que indica qué acción realizar, si hay que mover una pieza de una posición a otra o si hay que desplazarse a la posición donde se realizan las fotos. Se envían 6 datos diferentes para la acción de desplazar piezas, estos datos son: posición X y posición Y en el espacio de trabajo dónde se encuentra la pieza en su posición inicial, orientación que debe tener la pinza para cogerla, la altura a la que se encuentra, el color de la pieza que determinará su posición final y el número de piezas que se encuentran en esta última posición. Son las posibles configuraciones de estos datos son las que definen el movimiento del robot

Por último, tiene lugar la programación del robot. La programación se hace en lenguaje RAPID, el propio de la compañía ABB desarrollado para sus robots. Se definen todos los movimientos necesarios para la correcta manipulación de las piezas LEGO; estos movimientos estarán determinados por los datos que se han obtenido mediante los sockets e interpretado con RAPID. Los movimientos son los siguientes: desplazamiento desde la

posición actual hasta la posición X,Y recibida con la orientación de la pinza especificada, acercamiento de la pinza a la altura recibida, agarre de la pieza al cerrar la pinza, desplazamiento de la pieza a su posición final y apertura de la pinza para depositarla. Una vez el robot ha depositado la pieza, se sitúa en esa posición a una altura de seguridad elegida previamente y espera al siguiente envío de datos para llevar a cabo la siguiente acción.

Finalmente, se consigue la integración de un sistema de visión artificial, que consigue clasificar todas las piezas presentes en el espacio de trabajo en un momento determinado, y un robot industrial que, con la información recibida, manipula las piezas una a una depositándolas en su posición final ordenándolas según su color, rojo, amarillo y azul, y su tamaño, pieza simple o doble.

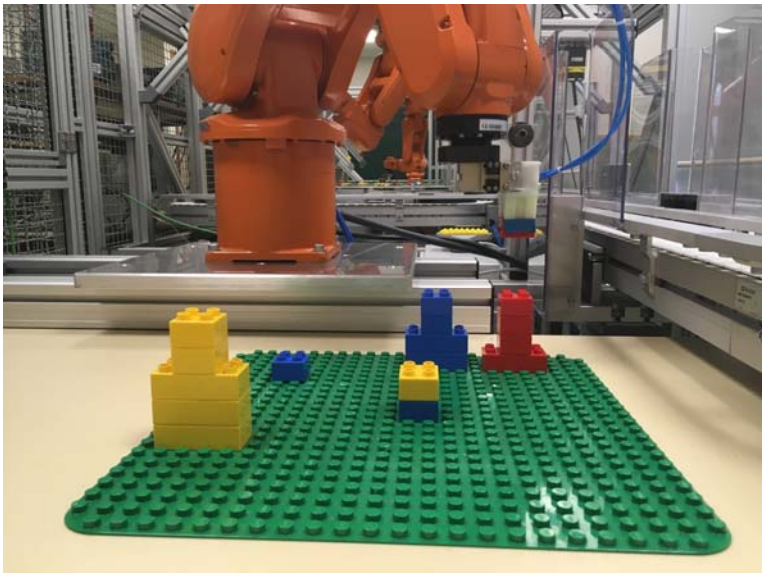


Figura 2. Sistema en funcionamiento

RESUMEN

INTEGRATION OF COMPUTER VISION ON AN IDUSTRIAL ROBOT

Author: Díaz García, Lucía

Directors: Rodríguez Mondejar, José Antonio

Boal Marín-Larrauri, Jaime

Collaborating entity: ICAI –Universidad Pontificia Comillas

ABSTRACT

This project aims to achieve the control of an industrial robot IRB 120 from the Company ABB with a computer vision system. The goal is the classification of LEGO pieces and their distinction of size and colour, and the automatic movement of the robotic arm to disassemble the initial structures the pieces were found in and their organization.

Figure 1. System architecture

The system's connections are shown in figure 1. The camera in charge of taking the images is connected to a computer where the images are processed with computer vision techniques, and data is prepared and set to the robot. The data exchange between the computer and the robot is achieved using internet sockets.

ABSTRACT

The project can be divided in three different areas of work; the first one is the capture and processing of the images to obtain the information of the environment. Secondly, the creation of a communication between Matlab and the robot controller for the data exchange. Lastly, the programming of the robot movements defined by the information received takes place.

The camera used for capturing the images is the Intel RealSense D435. It's a RGB-D camera, meaning that it can take colour images as well as depth images. Both images are necessary for the piece's classification; the first one, to separate them by size and colour and obtaining their coordinates, and the second one, to obtain the height that each classified piece can be found at. The image processing is done in Matlab, and once it's finished, the position in real coordinates of each piece present is known as well as their characteristics.

Secondly, it's necessary to create a communication between Matlab and the robot to send the information to the robot's controller so it can be processed and the robot acts according to the data received; this will be done by sending a vector with all the information using internet sockets. The first data sent is an identifier that indicates which action to perform, whether to move a piece from one position to another or moving to the position where pictures are taken from. For the moving of pieces 6 different data are sent: position X and Y on the working space where the piece is situated, the orientation that the gripper must have to work with it, the height it's at, the colour of the piece that will determine its final position and the number of pieces present in this last position. It's the different possible configurations of this data what will define the robot's movement.

Lastly, the robot's programming takes place. This is done in the language RAPID, developed by ABB for its robots. All movements necessary for the correct manipulation of the LEGO pieces are defined; these movements will be determined by the data that has been received and processed with RAPID. The moves defined are the following: movement from the current position to the X,Y coordinated received with the correct orientation of the gripper, positioning of the gripper at the specified height where the piece is situated at, grabbing of the piece when closing the gripper, displacement of the

piece to its final position and opening of the gripper to leave the piece. Once the piece has been moved, the robot is situated in that position on a secure height that has been previously defined and waits to receive the next set of data to perform the next action.

Finally, the result is the integration of a computer vision system, which achieves the classification of all the pieces present in the working space on a given moment, and an industrial robot that, with the information received, manipulates the pieces one by one placing then in their final position organizing them according to their colour, red, yellow and blue, and their size, double or simple piece.

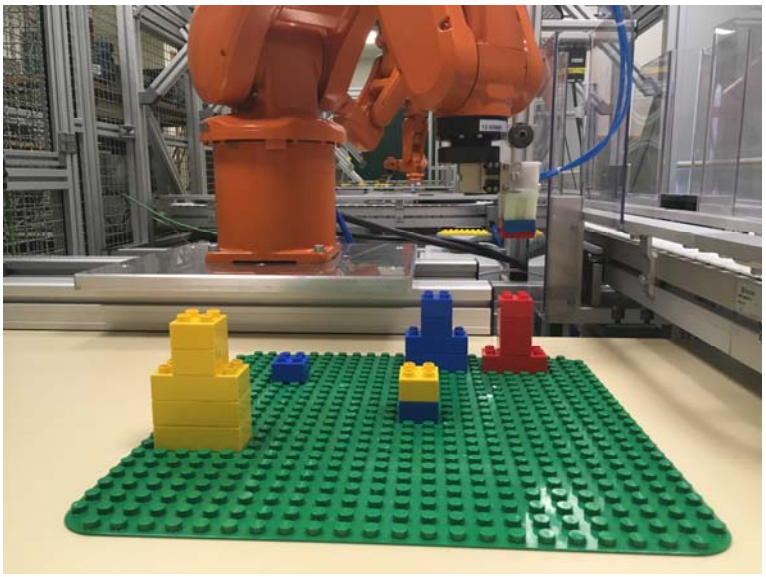


Figure 2.. Functioning system

ABSTRACT

Índice de la memoria

Índice de figuras

Índice de tablas

Índice de códigos

Parte I MEMORIA

Capítulo 1 INTRODUCCIÓN

En la actualidad es ampliamente común el uso de robots en procesos industriales. Estos han mejorado el tiempo y la calidad del trabajo en las grandes fábricas, así como han sustituido al hombre en algunas tareas que son de alto peligro, dejando el trabajo a un brazo robótico. La principal ventaja del uso de la robótica industrial es la automatización de procesos, permitiendo la ejecución de rutinas que no necesitan de la actuación de personas.

Por otro lado, la visión artificial cuenta con métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por un ordenador. En el mundo de la robótica aparece como una nueva forma de captar información del entorno consiguiendo así un control fluido y eficaz de los robots, aportando la capacidad de reaccionar a los cambios en el entorno y actuar conforme a ellos según sea necesario.

Este trabajo busca integrar un sistema de visión artificial que consiste en una cámara RGB-D, con un robot industrial IRB 120. Con esto se pretende conseguir el correcto funcionamiento del robot según la información obtenida y procesada mediante la visión artificial. Para conseguir esto, se estudiarán tanto los métodos de procesado de imágenes necesarios para obtener la información deseada, como la programación del robot en función de estos datos, incluyendo también la comunicación robot-cámara. El sistema será diseñado para clasificar y ordenar piezas lego según su color y tamaño.

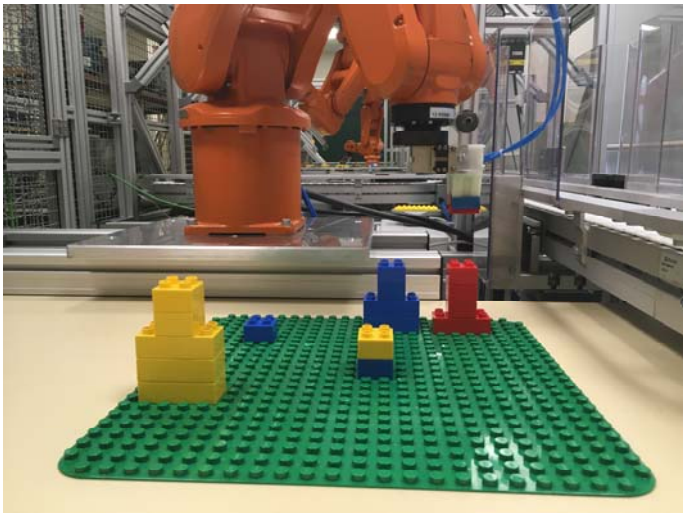


Figura 1. Sistema en funcionamiento

1.1 ESTUDIO DE LOS TRABAJOS EXISTENTES / TECNOLOGÍAS EXISTENTES

1.1.1 ROBÓTICA

Por definición de la Robotics Industries Association (RIA), "un robot industrial es un manipulador multifuncional reprogramable diseñado para desplazar materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variables programados para la ejecución de una diversidad de tareas".

Las tres principales ventajas del uso de robots frente a seres humanos son la mejora en la productividad, velocidad y seguridad. Gracias a estas tres características, es posible conseguir una notable mejora en la calidad de los procesos, así como una reducción en los gastos propios de la producción. Debido a esto, los robots son comúnmente usados en la industria, habiéndose convertido en los últimos años en un elemento indispensable en los procesos de manufactura. Las aplicaciones son

numerosas, desde trabajos de fundición o soldadura, a las cadenas de montaje, paletización o controles de calidad.

Por estas razones, la mayor parte de las industrias incluyen robots en sus procesos para conseguir su automatización. En muchos casos los robots usados son brazos robóticos como el que se usa en este proyecto.

Son numerosos los ejemplos de aplicación de estos robots industriales. Los robots ABB, misma compañía que produce el robot con el que se va a trabajar, levanta, envasan y ubican en pallets la comida y bebidas que para empresas como Nestlé y Cardbury. También tallan, pulen, barnizan, pintan, empaquetan y ubican en pallets los muebles para grandes marcas como IKEA y Tarkett.

1.1.2 VISIÓN ARTIFICIAL

En los últimos años, al ser la visión artificial una tecnología relativamente nueva, esta ha evolucionado a grandes pasos. Esta herramienta se ha convertido en una de las principales en el avance de la Industria 4.0, la nueva revolución industrial. La visión artificial se ha convertido en imprescindible en el control de procesos, como es el control de calidad o los procesos de seguridad industrial.

En la actualidad, estos sistemas de visión artificial están presentes en cada uno de los procesos de producción. Como los sistemas de visión trabajan en tiempo real, permiten disponer de datos precisos en cada momento y generar estadísticas detalladas de todos los componentes intermedios y productos finales, en cualquier instante de la producción. Todas estas acciones que llevan a cabo los sistemas de visión justifican que se hayan convertido en elementos fundamentales en los procesos de automatización industrial.

1.1.3 ROBÓTICA Y VISIÓN ARTIFICIAL

Tanto el mundo de la robótica como la visión artificial se encuentran en constante

desarrollo, y la unión de estos dos ámbitos en dirección a la inteligencia artificial abre puertas a infinitas posibilidades.

Los sistemas que integran estos dos mundos se denominan VGR, Vision Guided Robotics. Estos sistemas proporcionan mayor libertad y versatilidad al mundo de la robótica. Esto es gracias a que el uso de sistemas de visión artificial, como son las cámaras, permite a los robots conocer la posición de cualquier objeto en el espacio, dirigiendo así al robot al punto exacto donde este debe ir. Esto supone una evolución respecto a los casos en los que no se implementan estos sistemas, donde el área de trabajo debe ser constante y predecible, obligando al robot a seguir siempre una rutina predeterminada.

Un ejemplo de este tipo de robots que incluyen un sistema de visión artificial, es el robot colaborativo YuMi de la marca ABB. Los robots YuMi permiten la interactuación con un operario, maximizando así el potencial de ambos y aumentando la productividad. Entre otras, una de las aplicaciones que tienen estos robots es el de manufactura de enchufes en una planta en la República Checa. En el proceso de producción tanto robot como humano desempeñan diferentes acciones en conjunto, adaptándose el robot a las acciones realizadas por el humano.

Figura 2. Producción de enchufes con un robot YuMi – Fuente: https://i.ytimg.com/vi/crNUW1ma94M/maxresdefault.jpg

1.2 MOTIVACIÓN DEL PROYECTO

Como se ha descrito previamente, los sistemas robóticos integrados con visión artificial presentan numerosas ventajas y posibilidades. Con este proyecto se pretende introducir este tipo de sistemas en la minifábrica de ICAI para su futuro uso en otros proyectos y aplicaciones.

Gracias a los procesos diseñados en este proyecto, se podrá acceder a la posición exacta de las piezas que deben ser desplazadas por el robot, que podrán ser distintas cada vez que se ejecute la aplicación, y depositarlas en su posición final conociendo el estado de ambas, para así ajustar a las condiciones del entorno los movimientos del robot. De esta forma, se consigue un control fluido del robot sin necesidad de la intervención humana, ya que éste actuará en función de la configuración de piezas determinada en cada momento gracias a la información obtenida mediante la visión artificial.

1.3 OBJETIVOS

El objetivo principal de este proyecto consiste en el funcionamiento conjunto y fluido de un sistema de visión artificial, en este caso formado por una cámara RGB-D, y el robot industrial ABB IRB120. El sistema deberá capturar imágenes del espacio de trabajo y reconocer su estado para el posterior movimiento del robot acorde a la información que se ha obtenido.

El fin de esta aplicación es la añadir la posibilidad de poder trabajar con objetos que no se encuentren en posiciones predeterminadas. Este objetivo es muy importante en producción ya que si se da el caso de que se produzcan variaciones en las posiciones de los objetos, el robot tiene la capacidad de adaptarse sin necesidad de cambiar la programación. En consecuente, esto permite aplicar el robot en escenarios donde siempre vienen las piezas en desorden.

Un objetivo secundario que se pretende conseguir con este proyecto es de establecer una comunicación entre una plataforma externa al robot, en este caso Matlab, para el envío de los datos que se han captado mediante la visión artificial y así reaccionar a ellos.

1.4 METODOLOGÍA / SOLUCIÓN DESARROLLADA

Para conseguir los objetivos, el desarrollo de este proyecto se puede dividir en cuatro partes diferentes:

Captación y procesado de imágenes: En primer lugar, se indicará a la cámara que tome una foto. Esta función es escrita en lenguaje Python y ejecutada en Matlab mediante NumPy. Una vez las dos fotos han sido tomadas, RGB y profundidad, Matlab las procesa para obtener toda la información necesaria. Esto se consigue gracias a los comandos de Matlab de los paquetes de procesado de imágenes como

la Image Processing Toolbox.

Configuración y simulación del espacio de trabajo: Una vez la foto ha sido procesada, es necesario agrupar la información obtenida de forma que describa el espacio de trabajo efectivamente para conocer su estado en todo momento reaccionando a los cambios que se van produciendo según se mueven las piezas de una posición a otra. Con esto se busca elegir la información adecuada que se enviará posteriormente al robot para que lleve a cabo las acciones deseadas. Las piezas son clasificadas y su información almacenada en un vector que indica las características de cada una: coordenadas de la pieza con respecto a la esquina superior izquierda del área de trabajo, color, tipo, orientación y la altura.

Comunicación con el robot: La información se envía al robot para la posterior ejecución de las instrucciones necesarias para su correcto movimiento. Esta comunicación se consigue a través de sockets de internet, que envían la información desde Matlab, que actúa como cliente, al controlador del robot, que será el servidor. La información que se envía es un vector que contiene un identificador que indica qué acción realizar, si coger o dejar una pieza, la posición en la que llevar a cabo esta acción, la orientación que debe tener la pinza, y la altura a la que trabajar.

Movimiento del robot: Una vez la información ha sido recibida, es procesada mediante el lenguaje propio de ABB, RAPID, que contiene las instrucciones necesarias para que el robot ejecute los movimientos deseados para interactuar con las piezas Lego. La forma de trabajo elegida es la división del movimiento total del brazo en pequeños movimientos lineales con velocidad ajustada según la distancia a la que se encuentre de las piezas, para evitar la colisión con estas y que el proceso sea más fluido. Los movimientos lineales se programan mediante offsets respecto a la posición inicial elegida para el robot. Una vez terminado el proceso coger/dejar pieza, el robot espera al siguiente envío de datos.

1.5 RECURSOS / HERRAMIENTAS EMPLEADAS

1.5.1 ROBOT INDUSTRIAL ABB

1.5.1.1 ABB

ABB (Asea Brown Boveri) es una compañía líder mundial especializada en tecnologías de generación, transporte y distribución de energía eléctrica y tecnologías de automatización industrial que ayudan a incrementar la productividad de las industrias. La compañía tiene su sede en Zúrich y está presente en más de 100 países.

Las operaciones de ABB están compuestas por cuatro divisiones: división Electrification Products (electrificación de todos los puntos de consumo), división Robotics and Motion (soluciones de robótica y de movimiento inteligente), Industrial Automation (automatización industrial) y Power Grids (redes eléctricas).

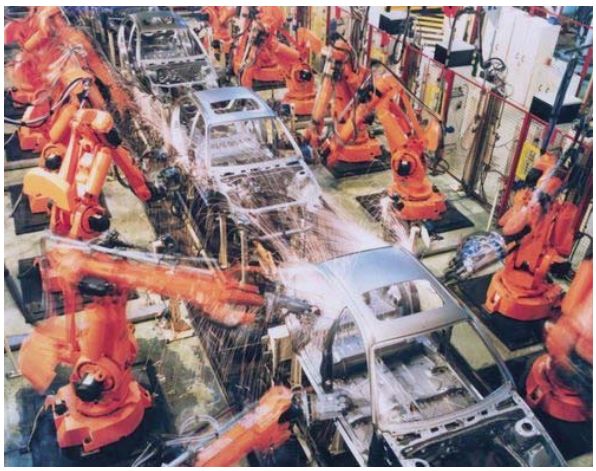


Figura 3. División de Robotics and Motion – Fuente:
http://www.roboticsupdate.com/2013/02/bmw-buys-2400-abb-robots-for-gluing-and-spot-welding/

La división de Robotics and Motion proporciona soluciones y servicios relacionados, que aumentan la productividad industrial y la eficiencia energética. ABB fue pionera en desarrollar el primer robot industrial del mundo alimentado con energía eléctrica, y actualmente cuenta con más de 160.000 robots operativos en todo el mundo. Ésta es la base instalada más grande de cualquier fabricante de robótica. Este suministro de robots se incrementa cada año debido a la gran necesidad de las empresas de evolucionar hacia una eficiencia energética y tecnológica, que les mantenga dentro del mercado.

ABB ofrece una oferta completa en soluciones robóticas, incluyendo robots y controladores, equipos y software para aplicaciones, celdas fabricadas a medida, y un servicio y soporte técnico global.

1.5.1.2 IRB 120

El robot utilizado para este proyecto es el IRB 120, de ABB. Se encuentra en uno de los laboratorios de electrónica de la segunda planta de ICAI. El robot industrial IRB 120 es el más pequeño producido por la compañía ABB. Pesa tan sólo 25 kg y puede soportar una carga de hasta 3kg (4kg en posición vertical de la muñeca), y tiene un alcance es de 580 mm. Este robot es una elección fiable y rentable para generar una gran capacidad de producción a cambio de una baja inversión.

Al ser pequeño, compacto y ligero, permite ser montado virtualmente en cualquier lugar, lo que resulta muy conveniente para la instalación en el laboratorio. Siendo el IRB 120 el robot más económico de la compañía, y contando con una capacidad de carga más que suficiente para esta aplicación, vamos a trabajar con piezas LEGO, de peso mucho menor que la carga máxima, resulta el robot más indicado para este proyecto.

Figura 4. Robot IRB120 en el laboratorio de sistemas digitales

En las figuras 5 y 6 se puede observar el área de trabajo de la muñeca (eje 5) con sus correspondientes dimensiones:

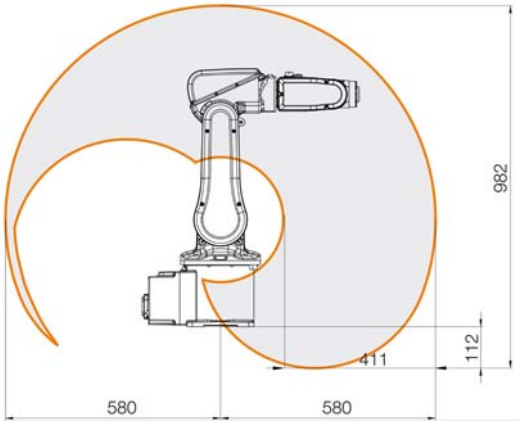


Figura 5. Área de trabajo eje 5

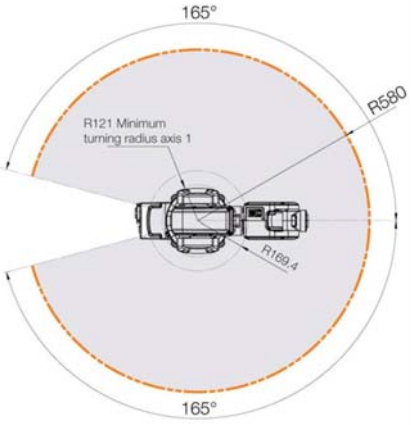


Figura 6. Área de trabajo eje 5 (2)

El robot está formado por seis ejes, es decir, cuenta con seis grados de libertad. Los tres primeros ejes están destinados a la posición del brazo, mientras que los tres últimos corresponden a la orientación de la muñeca. Las características de estos ejes son las siguientes:

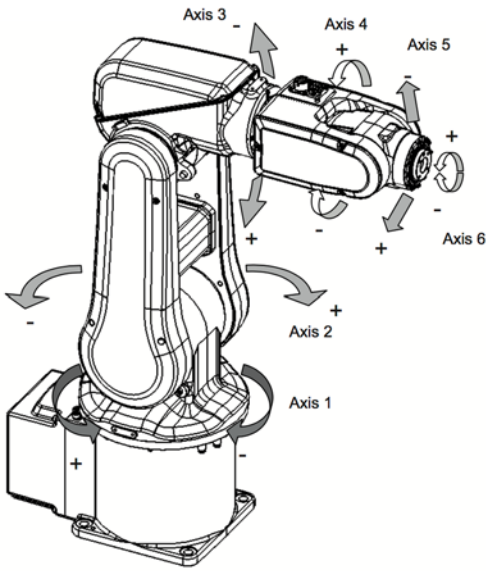


Figura 7. Ejes del robot

Eje	Movimiento	Área de movimiento
Eje 1	Rotación	+165º / -165º
Eje 2	Brazo	+110º / -110º
Eje 3	Brazo	+70º / -110º
Eje 4	Muñeca	+160º / -160º
Eje 5	Doblado	+120º / -120º
Eje 6	Giro	+400º / -400º

Tabla 1. Descripción de ejes

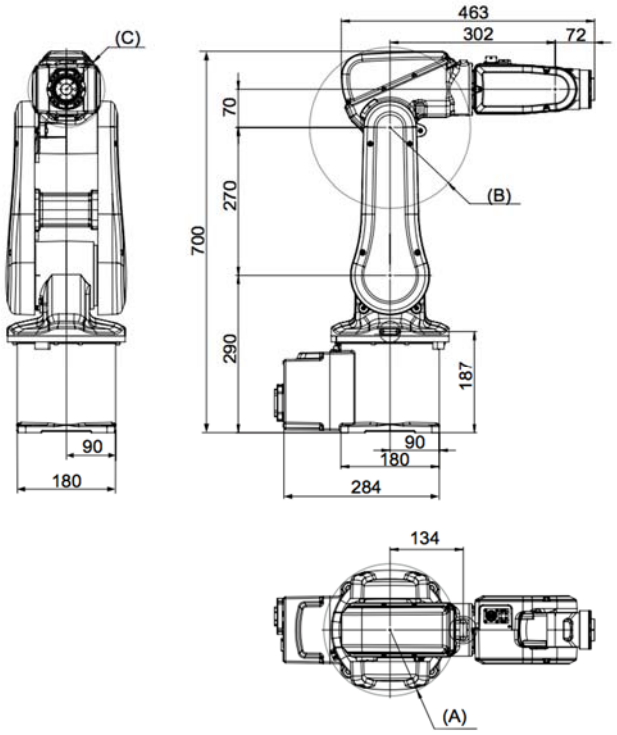


Figura 8. Dimensión del robot

1.5.1.3 Controlador IRC5 Y FlexPendant

Dentro de la familia de controladores que posee la empresa ABB, encontramos el controlador IRC5. ABB cuenta con distintos controladores para sus robots, el utilizado en este proyecto es el armario de control IRC5. Es en este controlador donde tiene lugar el procesado de información que ejecuta el código del programa. Además, sirve de alimentación e intercambio bidireccional de información con el robot, incorporando 16 entradas y 16 salidas digitales y una analógica en el rango de 0-10V. Este armario de control es de tipo portable y puede ser utilizado con brazos robóticos de hasta 8kg. Consiste en un único aparato de medidas 258x450x565mm y 27.5 kg de peso.

Figura 9. Armario de Control IRC5

El IRC5 se compone de los módulos siguientes:

- Módulo de accionamiento, que contiene el sistema de accionamiento
- Control Module, que contiene el ordenador principal, el panel del operador, el interruptor principal, las interfaces de comunicación y la conexión para FlexPendant; descrito a continuación. El controlador también contiene el software de sistema, que incluye todas las funciones básicas de manejo y programación.

El FlexPendant, una herramienta del armario IRC5, es el elemento encargado de comunicar al hombre con la máquina y viceversa. Está compuesto de elementos de hardware (botonera, pantalla táctil y joystick) y de software. El FlexPendant se conecta al controlador mediante un cable con conector integrado. Maneja múltiples funciones relacionadas con el uso del sistema de robot:

- Ejecutar programas
- Monitorizar el estado del manipulador
- Crear y editar programas de aplicación
- Mover cada uno de los ejes del robot de forma manual utilizando el joystick

También puede usarse la memoria de almacenamiento para guardar programas. A partir de ese momento, es posible descargarlos automáticamente. Es posible enviar

o recibir el programa completo o partes del programa a través de la red o con una memoria flash portátil conectada al puerto USB.

Figura 10. Flex Pendant

Pos	Descripción
A	Pantalla
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. Descripción Flex Pendant

1.5.1.4 RobotStudio

RobotStudio es el software de simulación y programación offline de la compañía ABB, que permite crear y simular estaciones robóticas industriales en 3D en un ordenador con robots idénticos a los empleados en la instalación. Este software permite trabajar eficientemente con datos del IRC5. RobotStudio puede considerarse como el compañero ideal del FlexPendant, usándolos de forma que se complementan y cada uno está optimizado para sus tareas concretas.

RobotStudio resulta ideal para el manejo de datos de configuración, gestión de programas, documentación en línea y acceso remoto. RobotStudio actúa directamente sobre los datos activos del controlador. La conexión al controlador puede hacerse localmente a través de la conexión para PC de servicio y, si el

controlador cuenta con la opción de RobotWare PC Interface, a través de una conexión de red. El programa sólo pueda tomar el control de un robot si tal operación se autoriza desde el FlexPendant.

1.5.1.5 RAPID

RAPID es el lenguaje de alto nivel propio de la compañía ABB para programar a sus robots. El lenguaje RAPID constituye una combinación equilibrada de simplicidad, flexibilidad y potencia. Contiene los conceptos siguientes:

- Estructura de programa jerárquica y modular, para admitir la programación estructurada y la reutilización de códigos
- Las rutinas pueden ser funciones o procedimientos
- Datos y rutinas locales o globales
- Expresiones aritméticas y lógicas
- Gestión de interrupciones

Los programas consisten en una secuencia de instrucciones que controlan el robot y cuenta con tres partes diferentes:

- Rutina principal (main): se inicia la ejecución.
- Conjunto de sub-rutinas: el programa se divide en partes más pequeñas formando sub-rutinas que pueden ser llamadas en cualquier momento desde la rutina principal.
- Datos de programa: definen posiciones, datos numéricos, sistemas de coordenada, etc.

El principal uso de RAPID en este proyecto es programar el movimiento del robot. Existen varias instrucciones de movimiento: MoveL, para movimientos lineales en los ejes XYZ (esta será la que se ejecute en esta aplicación), MoveJ para

movimiento de los ejes del robot (sigue la trayectoria más rápida), y MoveC para movimientos circulares. Las instrucciones de movimiento toman como argumento, entre otros, el valor de la posición a la que se desea llevar el robot.

1.5.2 PINZA UNIVERSAL PGM 60

Para poder manipular las piezas LEGO con las que se va a trabajar con el robot contamos con una pinza compuesta por dos partes.

La primera es la pinza PGM 60 de la marca Schunk. Es una pinza hidráulica con un tamaño de 56x70x28 cm, un peso de 0,21 kg y capaz de hacer una fuerza de agarre de 190 N, sus dedos tienen una carrera máxima de 5mm. Esta pinza se encuentra en el extremo del brazo.

Acoplada a esta pinza, se encuentra una pinza compuesta por dos "dedos" que serán los encargados de agarrar las piezas. Esta pinza ha sido diseñada en años anteriores por alumnos de ICAI y ha sido impresa en las impresoras 3D de la universidad.

El control de la pinza se consigue mediante el uso de señales digitales. Por seguridad, son necesarias dos señales, para cambiar la posición de la pinza entre las dos posibles, abierta o cerrada, ambas señales tienen que cambiar su valor. Las señales digitales a utilizar son DO10_7 y D010_8.

Figura 11. Pinza PGM 60

1.5.3 CÁMARA

La cámara elegida para este proyecto es el modelo RealSense Depth Camera D435. Esta cámara cuenta utiliza visión estéreo para calcular profundidad. Cuenta con dos sensores de profundidad, un sensor RGB y un proyector de infrarrojos.

Figura 12. Cámara RealSense Depth Camera D435

Sus características son las siguientes:

Use Environment	Indoor/Outdoor
Depth Technology	Active IR Stereo (Global Shutter)
Main Intel® RealSense™ component	Intel® RealSense™ Vision Processor D4 Intel® RealSense™ module D430
Depth Field of View (FOV)—(Horizontal × Vertical × Diagonal)	86 x 57 x 94 (+/- 3°)
Depth Stream Output Resolution	Up to 1280 x 720
Depth Stream Output Frame Rate	Up to 90 fps
Minimum Depth Distance (Min-Z)	0.2m

Sensor Shutter Type	Global shutter
Maximum Range	Approx.10 meters; Varies depending on calibration, scene, and lighting condition
RGB Sensor Resolution and Frame Rate	1920 x 1080 at 30 fps
RGB Sensor FOV (Horizontal x Vertical x Diagonal)	69.4° x 42.5° x 77° (+/- 3°)
Camera Dimension (Length x Depth x Height)	90 mm x 25 mm x 25 mm
Connectors	USB 3.0 Type - C
Mounting Mechanism	One 1/4-20 UNC thread mounting point Two M3 thread mounting points

Tabla 3. Características de la cámara

La cámara cuenta con Intel® RealSense™ SDK 2.0, una plataforma open-source que incluye herramientas, wrappers para diversos lenguajes y ejemplos de código para obtener datos mediante su programación. Esta librería, conocida como librealsense puede encontrarse en:

https://github.com/IntelRealSense/librealsense

Entre las herramientas que este kit incluye se encuentran:

- Intel® RealSenseTM Viewer: aplicación para una rápida evaluación de los datos que se están recibiendo.
- Depth Quality Test tool for Intel® RealSenseTM Camera – Interfaz para comprobar la calidad de la cámara de profundidad.

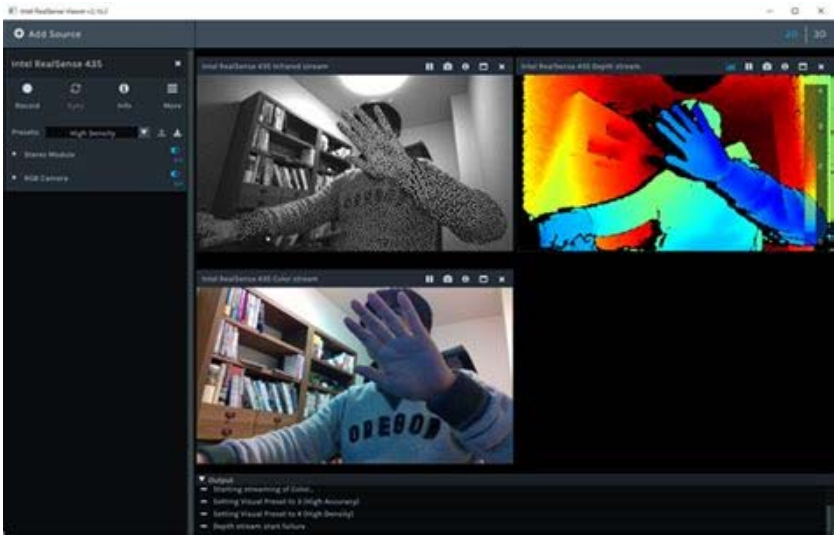


Figura 13. Ejemplo de aplicación cámara - Fuente: https://imagingsolution.net/wordpress/wp-content/uploads/2018/04/Buy_Intel_RealSense_D435_13.png

1.5.4 MATLAB

Matlab es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes), estas incluyen un conjunto de funciones ya implementadas para una tarea específica.

Las fotos tomadas con la cámara descrita anteriormente serán procesadas en Matlab para obtener toda la información necesaria. Para acceder a las fotos desde el programa es necesario el uso de NumPy. Esto es porque la programación de la cámara para la toma de fotos no soporta el lenguaje de Matlab, por lo que las funciones para tratar con ella serán escritas en Python y posteriormente ejecutadas

en Matlab mediante NumPy para poder procesarlas posteriormente.

NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. NumPy es open source.

Una vez obtenidas las imágenes, serán procesadas mediante las funciones de la Image Processing Toolbox. Esta caja de herramientas proporciona un conjunto de algoritmos, funciones y aplicaciones que permiten analizar imágenes, realizar transformaciones geométricas sobre ellas, examinar regiones de píxeles o ajustar el color y el contraste.

1.5.5 OTROS

Por último, para llevar a cabo el proceso que conforma este proyecto, contamos con piezas LEGO DUPLO de diferentes colores: azul, amarillo, rojo y verde. Las piezas presentan dos tamaños distintos, piezas de 2x2 (32x32x19mm) y de 4x2 (32x64x19mm).

Para trabajar con estas piezas y formar estructuras, se utiliza una placa con enganches que coinciden con el tamaño de las piezas para que están puedan quedar encajadas. El tamaño de la placa es de 24x24 puntos o enganches. Esta placa se encuentra en una mesa directamente debajo del robot, como se puede observar en la figura 11. Para evitar que se mueva a causa de los movimientos del robot al levantar y depositar piezas, se encuentra adherida a la mesa con pegamento.

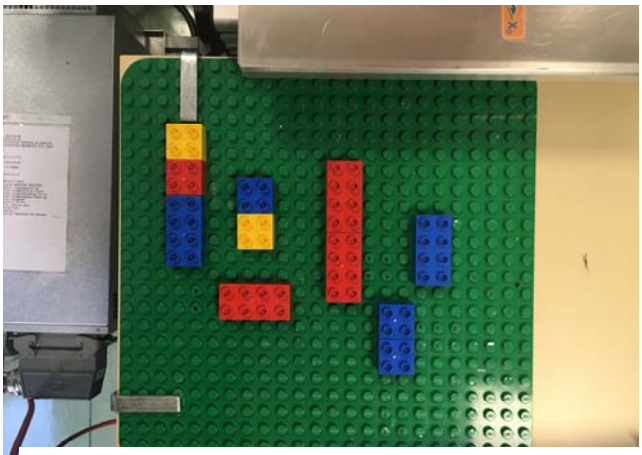


Figura 14. Placa con piezas LEGO

Capítulo 2 ARQUITECTURA DEL SISTEMA

Los elementos principales del sistema son la cámara y el robot. La cámara se conecta mediante un USB 3.0 a un ordenador en el que se obtendrán las fotos captadas por la cámara y se procesarán. El envío de datos se realiza desde este ordenador al robot mediante Matlab, y, por último, tendrán lugar las acciones de montaje y desmontaje de las piezas LEGO presentes en la mesa.

Figura 15. Conexión entre los sistemas

El proceso es el siguiente: en primer lugar, se abre la conexión entre Matlab y el robot, que permanecerá abierta durante todo el proceso; para hacer esto basta con ejecutar primero el programa que se encuentra en el controlador del robot y después el de Matlab. El robot en este momento está a la espera de recibir datos. Mientras tanto, una vez se ejecuta el programa de Matlab, inmediatamente después de aceptar la comunicación, se toman dos fotos distintas, una a color y otra de profundidad. Trabajando con ambas fotos se obtiene la información necesaria de cada pieza que se encuentra en la plataforma verde situada sobre la mesa. Esta información incluye: posición (x,y), color de la pieza, tamaño, orientación y altura. Todos estos datos serán posteriormente usados con diferentes fines.

El siguiente paso consiste en rellenar la matriz con todas estas piezas, de esta forma se puede conocer qué posiciones están ocupadas y qué contienen, y cuales están libres. Cuando se encuentra una pieza que está situada a una altura superior a uno, se procesa la siguiente imagen, que corresponde a la situación en que la pieza ha sido retirada, para conocer qué se encuentra debajo y de esta forma obtener la información de todas y cada una de las piezas con las que se pretende trabajar.

A continuación, se agrupan todas las piezas que se encuentran en el punto más alto de su posición, es decir, si en una posición concreta hay una estructura formada por cinco piezas colocadas una encima de otra, sólo se guarda la más alta, ya que será la única a la que se pueda acceder en ese momento. Con las piezas presentes agrupadas se procede a su clasificación diferenciando entre piezas dobles y simples. En primer lugar, se trabajará con las piezas dobles, llevándolas una a una a tres posiciones predeterminadas distintas, una para cada color. Cada vez que se retira una pieza y se lleva a su posición final, tanto la matriz como el vector que contiene a las piezas se actualizan, eliminando la pieza retirada y añadiendo la que se encontraba debajo. Una vez todas las piezas dobles han sido ordenadas, se repite el mismo proceso para las piezas simples.

La información que se envía al robot incluye un identificador que indica si se debe llevar a cabo la acción de coger una pieza o dejarla, la posición en la que realizar la acción, su altura, y la orientación que debe tener la pinza. El programa escrito en RAPID interpreta estos datos y los utiliza de manera adecuada para llevar a cabo las instrucciones que definirán el movimiento del robot para acudir a la posición adecuada y realizar la acción correspondiente. Una vez la pieza ha sido colocada en la nueva posición, se espera al siguiente envío de datos hasta que todas las piezas hayan sido ordenadas.

Capítulo 3 PROCESADO DE IMÁGENES

3.1 CAPTACIÓN DE IMÁGENES

La captación de imágenes se realiza mediante la integración de Python y Matlab. La cámara es programada en lenguaje Python, al no soportar el lenguaje M propio de Matlab. El procesamiento de estas imágenes, y el posterior envío de datos al robot, se lleva a cabo en Matlab, que actúa como plataforma intermedia entre cámara y robot. Por tanto, hay que buscar una forma de obtener las imágenes en Matlab a partir de una función escrita en Python.

En la función de Pyhton, hay que importar las librerías pyrealsense, para interactuar con la cámara, y numpy, para guardar las fotos en un formato que luego Matlab pueda procesar. Se definen tres funciones: la primera para iniciar la captación de imágenes, la segunda para terminar, y la última que será la que se encargue de obtener las fotos RGB y de profundidad.

La función final para la captación de imágenes se ha obtenido partiendo del código 'Open CV and Numpy integration', que se puede descargar de la carpeta de ejemplos de Wrappers para Python que se encuentra en la siguiente página:

https://github.com/IntelRealSense/librealsense/tree/development/wrappers/python

Se han realizado los siguientes cambios: se han eliminado todas las referencias a Open CV, ya que no solo no son necesarias, si no que pueden llegar a causar problemas en Matlab. En la función start(), se ha cambiado la resolución de la foto RGB a una mayor, y se ha cambiado el espacio de color, der GBR original a RGB.

En la función shot(), las imágenes obtenidas se convierten en datos del tipo numpy array, y se guarda cada una en un archivo diferente que se serán abiertos posteriormente en Matlab.

```
import pyrealsense2 as rs
import numpy as np

pipeline = rs.pipeline()

def start():
        # Configure depth and color streams
        config = rs.config()
        config.enable_stream(rs.stream.depth,  640,  480,  rs.format.z16,
30)
        config.enable_stream(rs.stream.color, 1280, 720, rs.format.rgb8,
6)

        # Start streaming
        pipeline.start(config)
        return 1

def stop():
        pipeline.stop()
        return 1

def shot():
        # Wait for a coherent pair of frames: depth and color
        frames = pipeline.wait_for_frames()
        depth_frame = frames.get_depth_frame()
        color_frame = frames.get_color_frame()



        # Convert images to numpy arrays
```

```
        depth_image = np.asanyarray(depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())


        np.save('imagen_color',color_image)
        np.save('imagen_depth',depth_image)


        return depth_image, color_image
```

Código 1. Captura y guardado de imágenes

La ejecución de esta función se hace a través de Matlab. Para ello hay que cargar Python, eligiendo la versión adecuada con la que se está trabajando. Después hay que acceder al directorio donde se encuentra la función e importarla como el módulo con el que se va a trabajar. Una vez hecho esto, se pueden ejecutar las funciones que se encuentran en ese módulo desde el código en Matlab sin la necesidad de tener la función de Python abierta, simplemente situando ambos códigos en la misma carpeta. Una vez ejecutadas estas funciones, para obtener las fotos se leen, mediante la función *"readNPY"* que lee datos creados en NumPy en Matlab, los dos archivos que se han guardado que contienen las fotos.

```
[~, ~, isLoaded] = pyversion;

if ~isLoaded
    pyversion('/anaconda3/envs/tfg/bin/python3.6')
end

p = py.sys.path;

module_path = '/Users/lucia/Documents/camera';

if count(p, module_path) == 0
    insert(p, int32(0), module_path);
end

camera = py.importlib.import_module('capture');
py.importlib.reload(camera);

camera.start();
a = camera.shot();
```

```
camera.stop();


Icolor = readNPY('imagen_color.npy');
Idepth = double(readNPY('imagen_depth.npy'));
Iprofundidad = Idepth / max(max(Idepth));
figure(); imshow(Icolor);
figure(); imshow(Iprofundidad);
```

Código 2. Lectura de las imágenes

Las fotos que se obtienen son las siguientes:

Figura 16. Imagen RGB obtenida

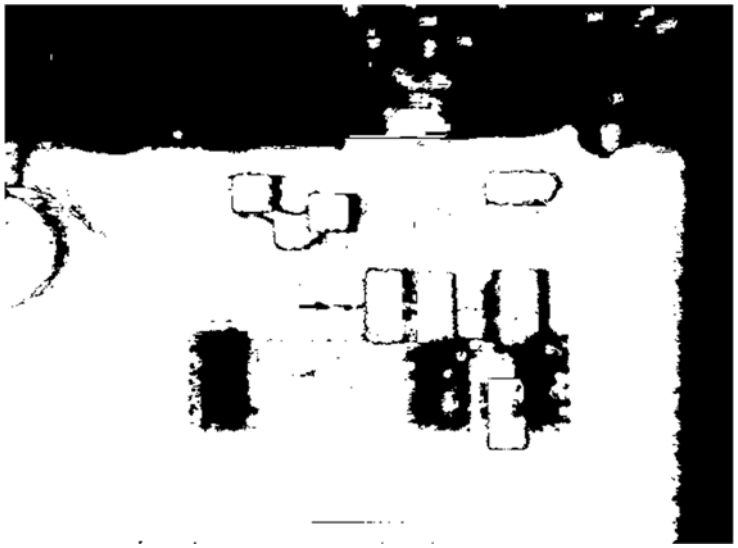


Figura 17. Imagen de profundidad obtenida

Para corregir la foto de profundidad, dividimos todos los elementos de la matriz que la conforman entre su máximo, resultando en valores entre 0 y 1 de sus píxeles para obtener así una imagen en escala de grises. Las distintas tonalidades de grises serán las que determinarán la altura más adelante cuando la imagen sea procesada.

Figura 18. Imagen de profundidad corregida

Una vez se han obtenido las fotos, profundidad y GRB se procede a procesarlas por separado para obtener toda la información necesaria sobre el estado del espacio de trabajo. Cada foto tiene una función diferente. Por un lado, la foto RGB permite conocer la posición de todas las piezas, así como diferenciarlas por color y tamaño. Con la foto de profundidad, se determina la altura a la que se encuentran las piezas detectadas en la foto RGB. La combinación de los datos obtenidos en ambas fotos es toda la información necesaria para el posterior envío al robot.

El primer paso antes de procesar las fotos será ajustarlas para poder trabajar con ellas a la vez, ya que como se puede apreciar en las figuras 16 y 17, tanto el tamaño como las posiciones en cada foto varían notablemente de una a otra. Para solucionar esto, se eligen en cada foto cuatro puntos fijos, correspondientes a las cuatro esquinas de lo que se ha definido como el espacio donde se va a trabajar. De esta forma se puede asegurar que ambas fotos contienen el mismo espacio. Como la foto de profundidad sigue siendo considerablemente más pequeña, se escala con un factor 2.5 para acercarse lo máximo posible al tamaño de la foto a color. La diferencia resultante después de realizar esto es tan pequeña que puede considerarse despreciable.

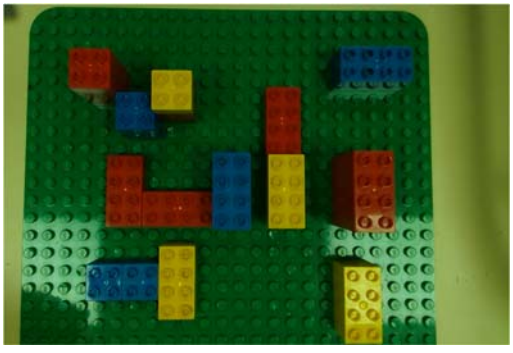


Figura 19. Imagen RGB escalada

Figura 20. Imagen de profundidad escalada

El esquema que se va a seguir para obtener la información necesaria de las imágenes se muestra a continuación. En primer lugar, se procesa la imagen RGB y más adelante la imagen de profundidad.

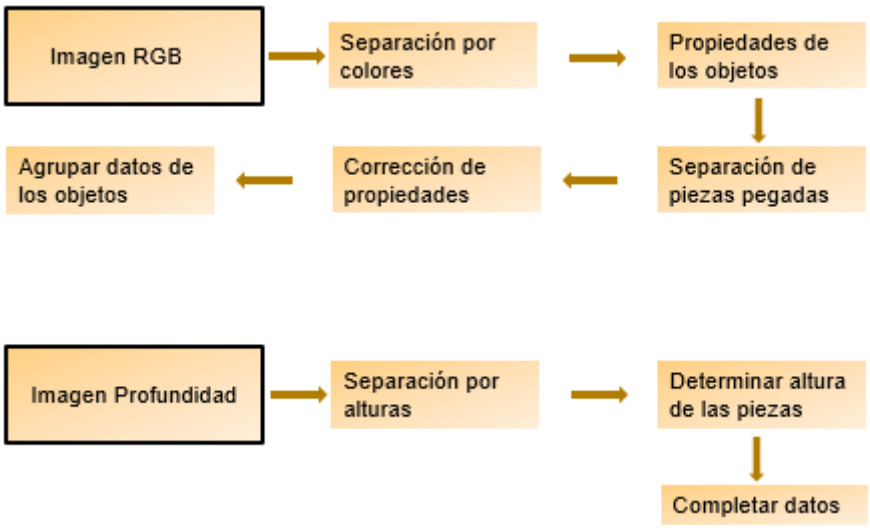


Figura 21. Esquma de procesado de imágenes

3.2 PROCESADO DE LA IMAGEN DE COLOR

La identificación de estas piezas en la foto RGB se hace por separado para cada color. Para conseguir hacer esto, primero se aplican tres máscaras diferentes a la foto original para conseguir dos fotos distintas para cada color. Las máscaras se crean mediante la herramienta de Matlab incluida en la Image Processing Toolbox, colorThresholder. Esta herramienta permite escoger una foto y un espacio de color entre RGB, HSV, YCbCr y L*a*b. El espacio elegido para trabajar con estas fotos es HSV, al ser el más intuitivo de usar.

Las propiedades de HSV son: Hue, Stauration and Value, que se traducen en matiz, saturación y valor. Cada posible combinación de estos tres componentes representa un color diferente.

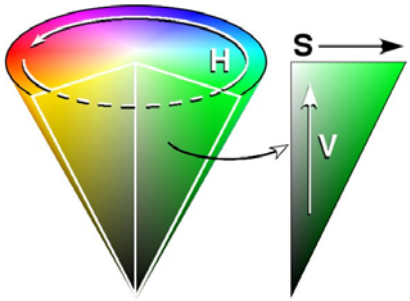


Figura 22. Representación del espacio de color HSV

Para separar las piezas por color, como no sólo es necesario definir tres valores con matices bastante diferenciados, basta con elegir un intervalo en la rueda de colores que representa elmatiz para separar cada color, no es necesario definir los otros dos componentes. En la siguiente figura se puede observar como las piezas rojo en la imagen han sido seleccionadas ajustando el intervalo que representa ese color.

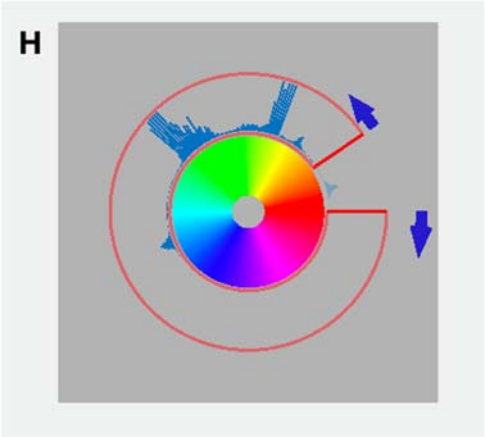


Figura 23. Rueda de HUE

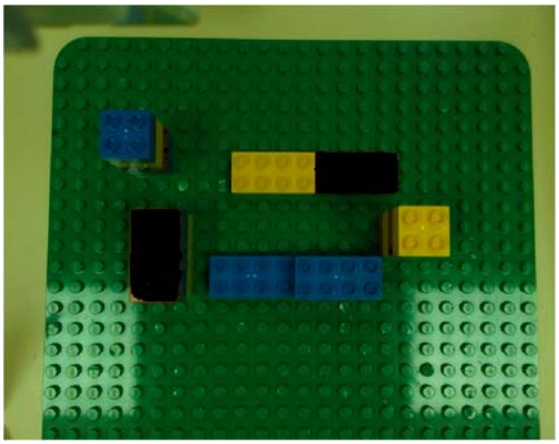


Figura 24. Selección de legos rojos

Una vez elegida la máscara, esta herramienta crea una función que al aplicarla a cualquier imagen devuelve dos fotos, una en binario y otra RGB que sólo incluye los píxeles del color que deseamos separar, con el resto de píxeles de la foto convertidos a negro. El resultado es el siguiente:

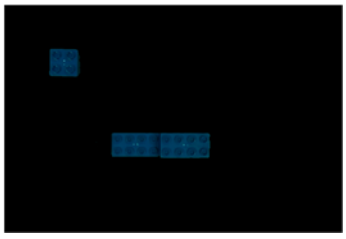


Figura 29. Máscara azul

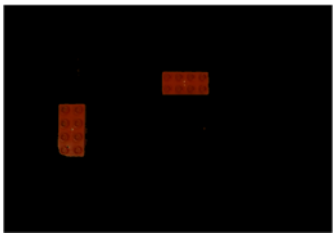


Figura 25. Máscara rojo

Figura 26. Máscara amarillo

Figura 30. Imagen binario azul

Figura 27. Imagen binario rojo

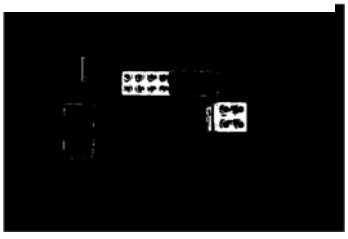


Figura 28. Imagen binario amarillo

El siguiente paso consiste en identificar los objetos que hay en la imagen en binario y sus propiedades; esto se consigue con las funciones de Matlab ***bwboundaries*** *y* ***regionprops****,* la primera traza las fronteras de cada región encontrada y devuelve una matriz donde se 'etiquetan' todos los objetos presentes en la imagen, y la segunda devuelve las propiedades de cada uno de esos objetos. Las propiedades necesarias para esta aplicación son: centroide, área, y bounding box ('caja' que indica las esquinas de la región de la imagen que contiene a ese objeto).

Uno de los problemas que se encuentran a la hora de identificar las piezas surge cuando dos o más piezas de un mismo color están colocadas una junto a la otra, y por tanto se identifican como una sola, es por eso que la función ***separar_piezas*** es necesaria para clasificarlas correctamente.

Esta función recorta la foto RGB obtenida al aplicar la máscara alrededor de la bounding box de cada objeto para poder trabajar con ellos con mayor detalle y poder determinar así cuantas piezas se encuentran en esa región, cuál es su posición exacta y de qué tipo son.

Para conseguir esto, primero hay que convertir la imagen a escala de grises para poder trabajar con ella, para obtener más detalle, se ajusta por un lado el contraste de la foto y por otro se aplica un filtro para 'difuminar' la imagen. Al restar a la imagen ajustada la imagen difuminada, el resultado es una nueva imagen con los detalles importantes y prominentes como las líneas más marcados, lo que facilitará trabajar con ella. A continuación, se buscan las líneas presentes en la imagen mediante el método 'canny'. Este método encuentra esquinas buscando los máximos locales del gradiente de la imagen usando la derivada de un filtro Gaussiano. Una vez obtenidas las líneas, estas se dilatan y se realiza una operación de 'cerrado', que conecta los píxeles que se encuentren lo suficientemente cerca unos de otros, el siguiente paso consiste en 'rellenar' esta foto. Al restar a la última

imagen obtenida la anterior, se obtiene finalmente un espacio lo suficientemente grande entre piezas, si es que lo hay, para poder separarlas.

```
Iaux = rgb2gray(I);
Iad=imadjust(Iaux);
Iblur=imfilter(Iaux,fspecial('gaussian'));
Idet=uint8(Iaux)-uint8(Iblur);
Ifin2=Idet+uint8(Iad);
Ilines = edge(Ifin2,'canny',0.05);
figure();    imshow(Ilines);
If = bwareaopen(Ilines, 50);
figure(); imshow(If);
Id = imdilate(If,strel('disk',1));
figure(); imshow(Id);
Ic=imclose(Id,strel('disk',2));
figure(); imshow(Ic);
Im=imfill(Ic,'holes');
Ib=Im-Ic;
```

Código 3. Separación de piezas

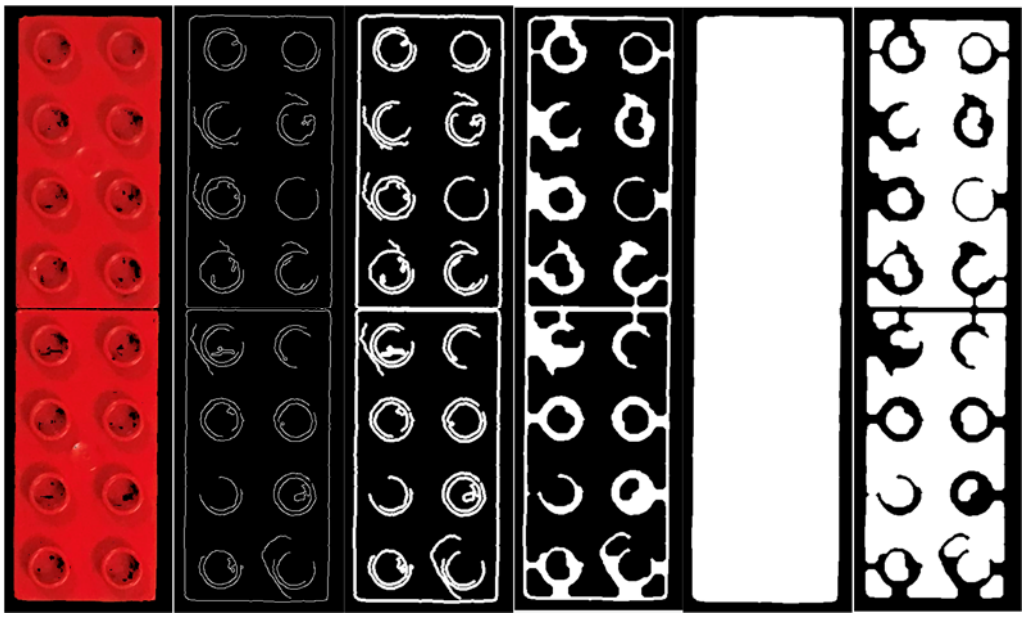


Figura 31. Progresión gráfica de la separación de piezas

El resultado es el siguiente: se pasa de identificar las dos piezas juntas como una sola a separarlas e identificar dos piezas diferentes.

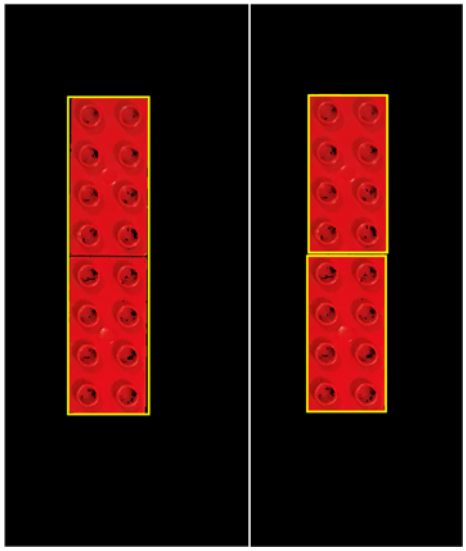


Figura 32. Resultado de la separación de piezas

Una vez se han realizado estas operaciones y una vez obtenida la última imagen de la figura 31, se vuelven a buscar los objetos y sus propiedades en esta nueva imagen, y se añaden al vector que devuelve la función. Este vector incluye la siguiente información: centroide de la pieza, las esquinas de su bounding box, su tamaño, y su orientación.

Para determinar el tamaño, simple o doble, se comprueba que el área del objeto sea mayor o menor que un número determinado, que actúa como límite entre un tamaño y otro. Las piezas simples se identifican con un 1 y las dobles, con un 2. La orientación de las piezas simples siempre será 0, ya que se podrán coger con ambas orientaciones de la pinza al ser cuadradas y ser todos sus lados iguales. Las piezas dobles, sin embargo, sólo podrán ser agarradas de una de las dos formas, por lo que es necesario conocer como están colocadas; basta con comparar los lados de su bounding box y conocer cuál es mayor para saber si está en posición vertical, identificada con un 1, o si es horizontal, con un 2. Una vez obtenida toda esta

información de cada pieza por cada color, se agrupan en un vector que las contenga a todas y se pasa a determinar la altura a la que se encuentra cada una.

3.3 PROCESADO DE LA IMAGEN DE PROFUNDIDAD

Para obtener la altura de cada pieza, hay que establecer en primer lugar el intervalo de valores de los píxeles que indican cada nivel. Para trabajar de forma más cómoda, primero se guarda la foto de profundidad en un archivo JPG, convirtiendo la imagen anterior con valores de 0 a 1, a valores de 0 a 255. Al ser las distancias relativas a la cámara siempre las mismas para cada nivel, estos valores serán fijos. Así, por ejemplo, los valores correspondientes a una pieza que se encuentra a una altura 4, se encuentran siempre entre 65 y 67. Todos los valores se encuentran entre 0 y 255, correspondiendo el 0 al color negro y 255 al blanco. Así, cuanto más oscura es una zona, y por tanto más pequeño el valor de sus píxeles, más cerca se encontrará de la lente.

El método elegido para determinar la altura consiste en obtener de la imagen de profundidad una foto binaria para cada altura, y posteriormente, buscar cada una de las piezas que se han encontrado en la foto RGB previamente en cada una de esas fotos. La foto donde se encuentre la pieza determinará su altura.

Para conseguir esto se siguen los siguientes pasos. Para obtener las fotos para cada altura basta con buscar en la foto original de profundidad los píxeles que tengan uno de los valores correspondientes y crear una foto que contenga únicamente esos píxeles. Para conseguir la foto que contiene las áreas correspondientes a una altura 5, la función hace lo siguiente:

```
for i = 61:64
    v5 = [v5; i];
end
    for j = 1:length(v5)
        h5 = iv  == v5(j);
        i5 = i5 + h5;
    end
```

Código 4. Función para conseguir áreas a la altura 5

Figura 33. Imagen de profundidad

Así obtenemos una nueva imagen que solo contenga los píxeles de la foto que correspondan a una altura 5.

Figura 34. Imagen de píxeles correspondientes a altura 5

El resto de las imágenes para las alturas 1-4 que resultan de la función son las siguientes:

Figura 35. Altura 1

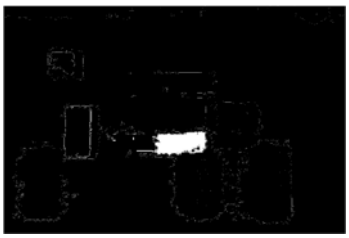


Figura 36. Altura 2

Figura 38. Altura 3

Figura 37. Altura 4

A continuación, pieza por pieza se recorta el área correspondiente a su *bounding box* en todas las fotos y se comparan para encontrar cuál de ellas es la que la contiene. La función que lleva a cabo este proceso es la siguiente:

```
function altura = determinar_altura(bbox, i1, i2, i3, i4, i5)
    k1 = 0; k2 = 0; k3 = 0; k4 = 0; k5 = 0;

Ih1 = imcrop(i1,bbox(1,:));
Ih2 = imcrop(i2,bbox(1,:));
Ih3 = imcrop(i3,bbox(1,:));
Ih4 = imcrop(i4,bbox(1,:));
Ih5 = imcrop(i5,bbox(1,:));

% figure(); imshow(Ih1);
% figure(); imshow(Ih2);
% figure(); imshow(Ih3);
% figure(); imshow(Ih4);
% figure(); imshow(Ih5);

for i=1:bbox(1,3)
    for j=1:bbox(1,4)
        if Ih1(j,i) == 1
            k1 = k1 + 1;
        end
        if Ih2(j,i) == 1
            k2 = k2 + 1;
        end
        if Ih3(j,i) == 1
            k3 = k3 + 1;
        end
        if Ih4(j,i) == 1
            k4 = k4 + 1;
        end
        if Ih5(j,i) == 1
            k5 = k5 + 1;
        end
    end
end

f = [k1;k2;k3;k4;k5];
[~,altura] = max(f);
end
```

Código 5. Recorte de la pieza en su bounding box

A continuación, se muestra el resultado que se obtendría para la pieza roja situada en posición vertical a la izquierda de la imagen:

Figura 39. Resultado tras pasar por código 5

Al observar las imágenes es fácil encontrar en cuál de ellas se encuentra la pieza, en este caso corresponde a una altura de nivel 3. La función cuenta el número de píxeles que hay encada foto que se han encontrado en esa área y elige la que tenga un número más elevado.

```
f =
        98
       429
      9231
       113
        80
altura =
     3
```

Código 6. Cuenta de píxeles y resultado de altura

3.4 SIMULACIÓN DEL ESPACIO DE TRABAJO

La simulación del espacio de trabajo se consigue a través de una matriz de celdas 24x24 que incluirá la información de lo que se encuentra en cada posición. Esta matriz representa el tapiz para piezas lego situado en la mesa del robot, que cuenta justo con 24 filas y 24 columnas de puntos de encaje para las piezas. Cada punto en la matriz representa un punto en el tapiz real. Se ha elegido trabajar con celdas ya

que cada posición contendrá informaciones de distintos tamaños y tipos. Esto se explicará con mayor detalle más adelante.

En un principio todas las celdas se encuentran vacías, lo que indica que están libres, y se rellenan con el valor -1 todas las posiciones a las cual el robot no puede acceder, para indicar que están fuera del rango de trabajo del robot. En la figura 39 podemos ver la zona azul que representa las posiciones inaccesibles por el robot y en blanco las posiciones que se encuentran dentro del área de trabajo del robot.

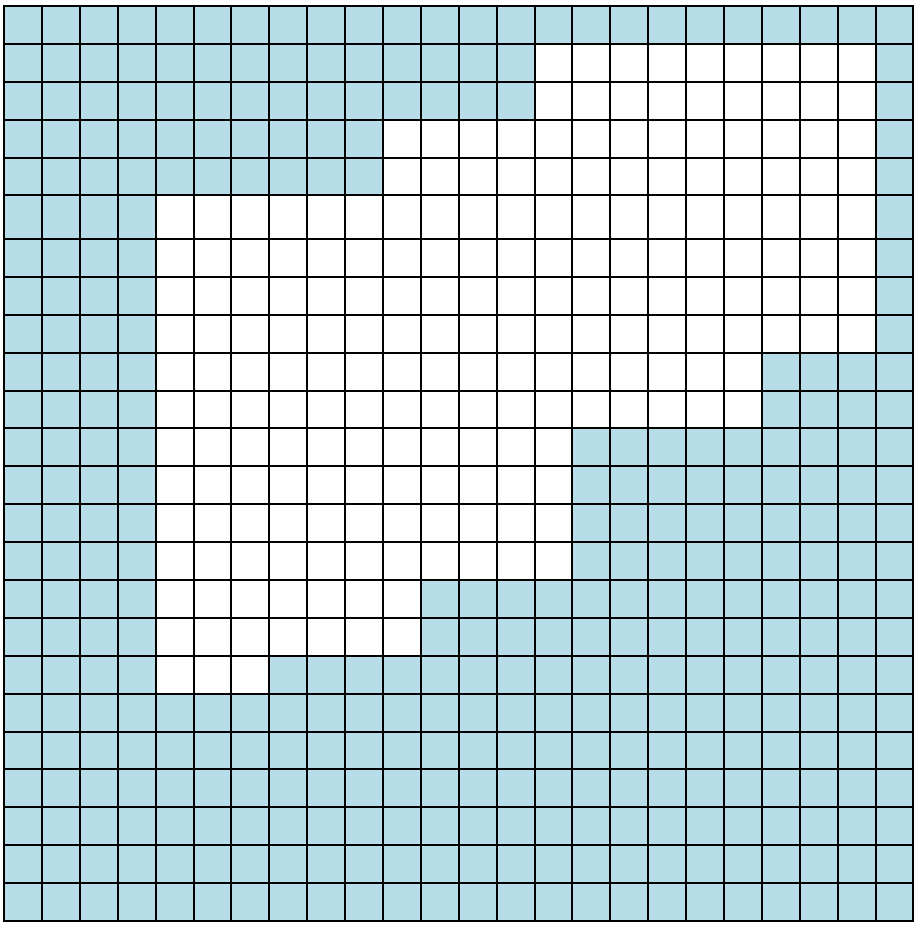


Figura 40. Distinción entre posiciones inaccesibles y área de trabajo

A continuación, se rellena la matriz con la información obtenida previamente de las piezas. Para ello, hay que conocer la posición exacta de cada pieza en la matriz, asignando a cada una un punto concreto. Primero hay que convertir las posiciones de cada una de píxeles a milímetros. Para ello se toma como referencia un punto en la matriz, en este caso el (5,6). Una vez conocido el valor en píxeles de esta posición, para cada pieza se calcula la distancia a este punto, y se divide esta distancia entre el número de píxeles (aproximado) que hay entre una posición y la contigua, obteniendo así la distancia al punto. Para convertir esta distancia relativa, en la posición actual en la matriz, se suma el punto inicial. Se substituyen en el vector de piezas los valores del centroide por estos nuevos valores de posición. El nuevo vector con el que se va a trabajar contiene los siguientes datos: posición en la matriz, color, tipo, orientación y altura. El código de colores es el siguiente: 1 para las piezas rojas, 2 para las amarillas y 3 para las azules.

Para comprobar que los datos que se han obtenido son correctos, se crea la siguiente imagen. Tras clasificar las piezas, se dibuja el perímetro de cada una de su color correspondiente, con un marco blanco si la pieza es simple, y negro si es doble. Se dibujan también los centroides con la altura obtenida de cada pieza a su derecha, y la posición que ocupa en la matriz debajo. Tras comparar los datos de la foto con el espacio de trabajo real donde se encuentran colocadas las piezas, se corrobora que efectivamente las imágenes se han procesado correctamente y que los datos se corresponden con la realidad.

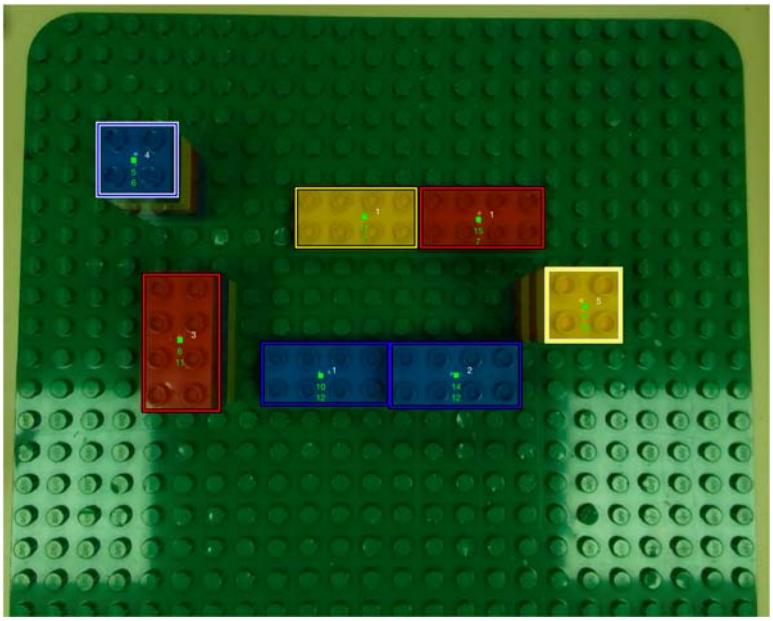


Figura 41. Clasificación de piezas

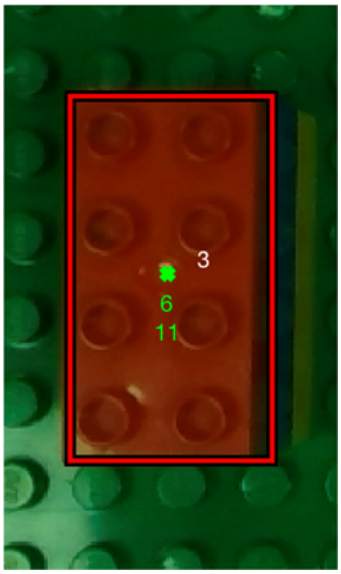


Figura 44. Datos de pieza 1

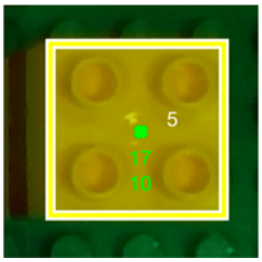


Figura 42. Datos de pieza 2

Figura 43. Datos de pieza 3

En las figuras 41, 42 y 43 se muestran los datos obtenidos de tres piezas diferentes.

En la posición obtenida de cada pieza se asigna un vector en la matriz que indica sus propiedades: color, tipo, orientación, altura. Así, por ejemplo, si en una posición determinada se encontrase una pieza roja, doble, con orientación vertical y a una altura 3, el dato guardado sería un vector con la siguiente información: *[1 2 1 3]*. A continuación, por cada pieza, se incluye también los cuatro valores que representan las esquinas de su bounding box, que será necesario más adelante cuando se quiera averiguar qué se encuentra a niveles inferiores de los capturados en la foto inicial.

Para cada pieza, el punto elegido para contener la información varía según el tipo y la orientación. Para las piezas simples corresponde a la esquina superior izquierda, en las dobles horizontales será fila superior segunda columna y en las verticales, segunda fila primera columna.

Para las piezas con una altura mayor que uno, es necesario obtener más imágenes qué se encuentra en los niveles inferiores hasta llegar a una altura uno. Para obtener todos estos datos adicionales, cada vez que se retira una pieza, se captura una nueva foto. Para averiguar de qué tipo de pieza se trata, se recorta el área de la pieza inicial a través de su bounding box en cada nueva foto y se aplican las máscaras de color ya utilizadas anteriormente. El color de la pieza es el único dato que falta por conocer, ya que la posición la conocemos de antemano, la altura también, ya que solo hay que restar uno a la inicial por cada pieza que se retire, y el tipo y orientación se asume que será el mismo. Al retirar una pieza y obtener la información de la que se encuentra debajo, se substituye en la matriz el valor de la nueva pieza por la anterior.

3.5 ORGANIZACIÓN DE PIEZAS

Con las piezas clasificadas y toda la información necesaria en la matriz, todo está listo para empezar el proceso de organización. Se definen tres posiciones finales, una para cada color. Al comienzo del proceso se inicializan estas posiciones con valor 0, lo que indica que todavía no se encuentra ninguna pieza en la posición. Primero se buscan todas las piezas dobles, y evaluando su color se desplazan a su posición correspondiente. Cuando todas las piezas dobles han sido ordenadas, se procede a hacer lo mismo con las simples hasta que todas hayan sido colocadas en su posición final.

El vector que contiene las piezas cambia constantemente al eliminar y añadir piezas cada vez que una es retirada. Como el tamaño es desconocido, se trabaja siempre con la pieza que se encuentra primero en el vector hasta que este se vacíe. Se evalúan las características de esta pieza y dependiendo de su altura simplemente se elimina una vez que se ha realizado el envío de datos al robot para cogerla y dejarla, o se elimina esa pieza y se añade la siguiente obtenida mediante la nueva foto al final del vector para ser evaluada cuando las demás piezas que ya se encontraban accesibles hayan sido movidas.

Cada vez que se desplaza una pieza, es importante actualizar también el estado de las posiciones finales para conocer hasta que altura habrá que bajar para depositar la siguiente pieza que se lleve a esa posición. Cómo estas posiciones han sido inicializadas a cero, simplemente hay que sumar uno a su valor actual para indicar que la altura ha aumentado un nivel.

```
vector =
  11   7   2   2   2   1
```

```
   6   11    1    2    1    3

vector =

   6   11    1    2    1    3

   15    7    1    2    2    1

vector =

   6   11    3    2    1    2
```

Código 7. Ejemplo de actualización de vector de posiciones

En el ejemplo anterior se aprecia como el vector de piezas cambia en función de la pieza que se retire. En un primer instante hay dos piezas dobles. Como la primera pieza que se encuentra en el vector tiene altura uno, simplemente se elimina, por lo que el vector pasa a contar con una pieza. La siguiente pieza tiene una altura de valor tres; en este caso se elimina esa pieza y se añade al final del vector la que se encuentra debajo, en este caso sigue habiendo una pieza accesible. Como es de esperar, esta última pieza añadida se encuentra en la misma posición que la eliminada, cambiando el color por el de la nueva pieza, manteniendo el tipo y la orientación y reduciendo la altura en uno

.

Capítulo 4 COMUNICACIÓN ENTRE MATLAB Y EL ROBOT

Este proyecto plantea, además del uso de visión artificial y la programación de un robot, la comunicación entre dos sistemas remotos para una aplicación más avanzada que presenta numerosas posibilidades. Esta comunicación permite el control externo del robot desde un sistema remoto. El uso de la cámara y Matlab requiere del uso de sockets de internet para comunicarnos con el controlador del robot.

El término socket es usado para denominar a una interfaz de programación de aplicaciones para protocolos de internet TCP/IP (Transmission Control Protocol/Internet Protocol). Los sockets de internet permiten la comunicación entre dos computadores distintas, o dos programas distintos dentro de una misma, para el envío de datos entre ellas. Éstos constituyen el mecanismo para la entrega de datos entre programas. El identificador de socket está compuesto por una dirección IP y un puerto. El TCP garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes de datos sean entregados en el mismo orden en el que fueron entregados.

En la aplicación de este proyecto, el robot actúa como servidor, y Matlab como cliente. Al ser RAPID el servidor, debe ser el robot quién abra la comunicación, mientras que el papel de Matlab es aceptarla.

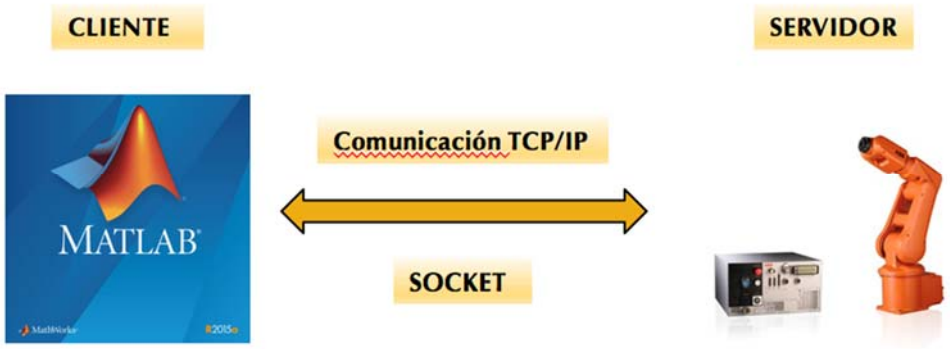


Figura 45. Esquema de comunicación mediante sockets

En primer lugar, se declaran las variables en RAPID y a continuación crear la comunicación con el robot como servidor y esperar que ésta sea abierta por Matlab. Esto se lleva a cabo mediante el proceso ***conexion***.

```
VAR socketdev server;
VAR socketdev client;
VAR bool connected := FALSE;
VAR string receivedData;
```

Código 8.Variables de sockets en RAPID

```
PROC conexion()
   SocketCreate server;
   SocketBind server, "192.168.56.91", 1024;
   SocketListen server;
   SocketAccept server, client;
   connected := TRUE;
ENDPROC
```

Código 9. Proceso Conexión

Para que los sockets funcionen correctamente, es necesario ejecutar primero el programa del robot al ser este el servidor, y en el tiempo que éste espera la repuesta del cliente, se ejecuta el programa de Matlab para aceptar la conexión. Como se puede ver en los códigos 9 y 10, tanto la dirección IP y el puerto son los mismos en ambos programas.

```
%Establecer conexion
   tc = tcpip('192.168.56.91',1024);
   fopen(tc);
```

Código 10. Establecimiento de conexión en Matlab

El envío de datos se realiza desde Matlab a RAPID. El método usado es una adaptación del que describe en el trabajo de fin de grado de la Universidad de Alcalá "Diseño de una estación de trabajo para el Robot IRB120. Control de cinta transportadora mediante IRC5" de Andrés Senén Estremera. Se han modificado las funciones y el tipo de datos que se envían para ajustarse a los fines que se pretenden conseguir en este proyecto.

La información enviada se empaqueta en un vector mediante la función ***enviar_coger***. La información enviada es la siguiente:

- ID: Identificador del vector, indica que acción llevar a cabo
- Pfinal: contiene la información de la pieza
    - Coordenada X de la pieza que se desea coger
    - Coordenada Y
    - Orientación de la pinza
    - Altura a la que se encuentra la pieza
    - Color de la pieza (esto determinará la posición final)
    - Altura de la posición final correspondiente

Para la interpretación en RAPID, se convierte el vector a una cadena de caracteres y se procede al envío con la función ***fwrite***.

```
function enviar_coger(tc,x1,y1,pinza1,altura1,color,altura2)
    % Vector de datos:
    % ID: 1 para coger
    % x coordenada
    % y coordenada
    % orientacion pinza
    % altura pieza
    % color pieza
    % numero de piezas en la posicion final

    ID = 1;
    datos = [ID;x1;y1;pinza1;altura1;color;altura2];
    datos = mat2str(datos);
    fwrite(tc,datos);
end
```

Código 11. Función enviar_coger

La otra función que envía datos al robot, es la función ***enviar_foto***. El identificador que se envía, indica al robot que debe situarse en la posición que se ha elegido para tomar las fotos. Este dato se envía cada vez que se retira una pieza y se quiere conocer qué se encuentra debajo.

Una vez el paquete de datos es recibido por el robot, el siguiente paso consiste en su interpretación en RAPID. Los datos recibidos se guardan en una cadena de caracteres que se ha definido previamente. El código que contiene las funciones que realizan estas acciones son las siguientes:

```
!Create communication
conexion;

WHILE(connected) DO
   !Receive a message from client
   SocketReceive client, \Str:=receivedData\Time := 15;
   Identificador;
   ProcesarDatos;
```

Código 12. Lectura RAPID de datos recibidos

Mediante el proceso ***Identificador*** primero se interpreta únicamente el primer dato del vector, que indica al robot si debe ejecutar la función de coger o dejar, que se describirán más adelante en el capítulo de programación del robot, con los datos que reciba de la pieza.

```
MODULE Ident
  VAR string Idf :="";
  VAR bool p;
    PROC Identificador()

        Idf := StrPart(receivedData,2,1);
        p := StrToVal(Idf,ID);
    ENDPROC
ENDMODULE
```

Código 13. Proceso Identificador

El proceso ***ProcesarDatos*** separa la información de la cadena de caracteres para obtener uno a uno todos los datos recibidos y guardarlos para su uso en otros procesos.

```
MODULE DatosCD
   CONST num ndatos:=6;
   VAR num Vec{ndatos};

   VAR string st :="";
```

```
    VAR string staux := "";
    VAR string ch := ";";
    VAR num pos;
    VAR num res;
    VAR bool ok;


    PROC ProcesarDatos()

        st := StrPart(receivedData,2,Strlen(receivedData)-2);

        !Identificador
        pos  := StrMatch(st, 1, ch);
        st := StrPart(st,pos+1,Strlen(st)-pos);

        !Vector de datos
        FOR j from 1 TO ndatos-1 DO
            pos  := StrMatch(st, 1, ch);
            staux :=  StrPart(st,1,pos-1);
            ok := StrToVal(staux,res);
            Vec{j} := res;
            st := StrPart(st,pos+1,Strlen(st)-pos);
        ENDFOR

        ok := StrToVal(st,res);
        Vec{ndatos} := res;

        X1 := Vec{1};
        Y1 := Vec{2};
        pinza := Vec{3};
        n1 := Vec{4};
        color := Vec{5};
        n2 := Vec{6};
    ENDPROC
ENDMODULE
```

Código 14. Proceso ProcesarDatos

Capítulo 5 PROGRAMACIÓN DEL ROBOT

El robot es controlado mediante instrucciones escritas en lenguaje RAPID a través del controlador FlexPendant.

El programa principal ejecuta los procesos adecuados en cada momento dependiendo de la información que haya recibido a través de los sockets.

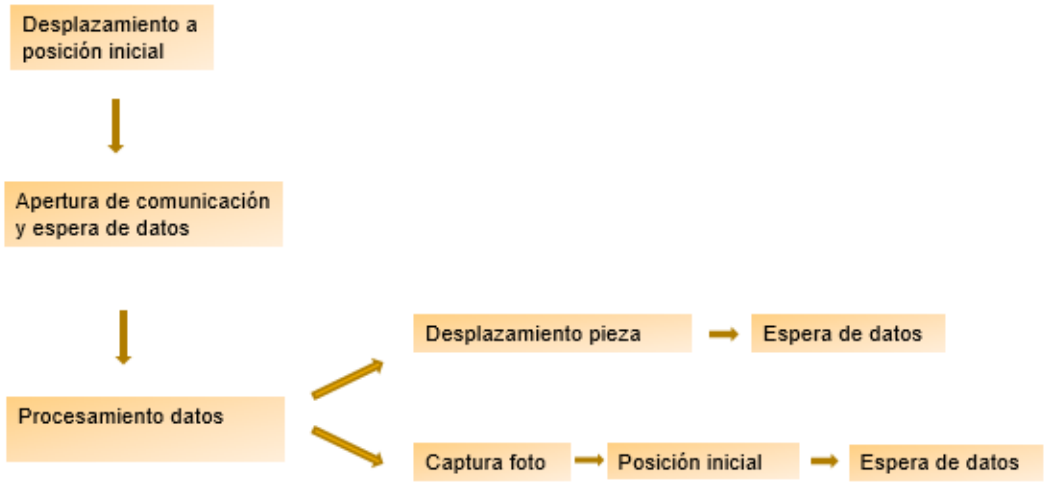


Figura 46. Proceso RAPID

Al comenzar la aplicación, el robot se sitúa en su posición inicial, que será la que sirva de referencia para el resto de movimientos. Esta posición se encuentra en la esquina (5,5), mismo punto de referencia que usamos en Matlab, a una altura de 200 mm de la plataforma donde se colocan las piezas.

Las posiciones se guardan en datos del tipo ***robtarget***. Este tipo de dato no sólo define la posición del robot, sino también la de sus ejes externos, ya que se puede alcanzar la misma posición de diversas maneras, por lo tanto, es necesario

especificar. Se definen dos constantes robtarget distintas para esta posición inicial, una para cada orientación de la pinza, a partir de estas dos posiciones se programarán todos los movimientos del robot. Estos datos indican, por ejemplo en el caso de la pinza en vertical, que su posición en coordenadas del robot correspondiente a la posición (6,5) en la matriz es en (x,y,z), (301.18,-200.0) en mm. El siguiente campo, que cuenta con cuatro datos, está formado por los cuaternios, que indican la orientación del efector final del robot.

```
CONST robtarget pv:=[[301.18,-269.93,200],[0.00357515,-0.485996,0.873934,-
0.00589287],[-1,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Código 15. Definición de posiciones

Estos datos están definidos para el robot ROBOT4_81 del laboratorio de sistemas digitales 1 de ICAI. Para desarrollar esta aplicación en cualquiera de los otros tres robots que se encuentran en la planta, bastaría con cambiar estas dos medidas en un punto nuevo de referencia a elegir y averiguar cuáles son los límites del robot para evitar errores en su movimiento al tratar de llevarlo a posiciones que no puede alcanzar, generando un fallo en todo el sistema.

El método que se ha llevado a cabo para elegir estas dos posiciones es el siguiente: para ambas orientaciones de la pinza, manualmente, utilizando el joystick del controlador FlexPendant, se coloca la pinza en la posición (6,5) y se ajusta mediante pruebas con una pieza hasta que se compruebe que en la posición elegida se puede coger y volver a dejar la pieza en el mismo lugar con precisión. Una vez obtenida esta posición (x,y) se sube el robot hasta la altura deseada, en este caso 200 mm. Una vez en esa posición se guarda manualmente en el programa como una constante. A partir de ahora estas posiciones se convierten en la referencia para alcanzar cualquier punto en la matriz con precisión, ya que la distancia entre punto y punto es conocida.

Con el robot situado en la posición inicial y una vez procesada toda información de Matlab y conocidos los valores del identificador, la posición (x,y), la altura y la orientación de la pinza, se llevan a cabo los procesos que contienen las instrucciones para mover al robot de la forma deseada.

En primer lugar, se comprueba la orientación de la pinza, ya que, dependiendo de esto, el punto de referencia a elegir para el resto de los procesos serán *pv*, si la pinza tiene que colocarse en vertical, o *ph* si es en horizontal.

```
TEST pinza
CASE 1:
   p2 := ph;
CASE 2:
  p2 := ph;
ENDTEST
```

Código 16. Comprobación de orientación de la pinza

El siguiente paso consiste en comprobar el valor del identificador: si es 1, se lleva a cabo el proceso ***Coger***, que retira las piezas de su posición inicial y las lleva hasta la posición final, y si es 2 el robot se mueve a la posición desde la que se toman las fotos mediante ***HacerFoto***.

```
TEST ID
CASE 1:
   Coger;
CASE 2:
   HacerFoto;
ENDTEST
```

Código 17. Comprobar ID

El proceso ***Coger*** comienza abriendo la pinza, preparándola para bajar hasta la pieza y cogerla. Como se ha explicado en el apartado de herramientas al hablar de la pinza, las salidas digitales que controlan la pinza son la D010_7 y la D010_8. Estas dos señales deben tener valores opuestos para abrir o cerrar la pinza, es decir, una de ellas debe ser 0 y la otra 1. Esto es por motivos de seguridad, es necesario que ambas señales cambien su valor para cambiar de una posición a otra. El cambio de

abierta a cerrada y al revés se lleva a cabo mediante los procesos ***AbrirPinza*** y ***CerrarPinza***.

```
PROC AbrirPinza()
   Set DO10_7;
   Reset DO10_8;
ENDPROC

PROC CerrarPinza()
   Reset DO10_7;
   Set DO10_8;
ENDPROC
```

Código 18. Señales para abrir y cerrar la pinza

La forma de movimiento elegida es el movimiento lineal mediante offsets. De esta forma, se trabaja siempre de forma segura, eligiendo trabajar desde una altura elevada primero, evitando así chocar con las estructuras que haya sobre la plataforma. Las velocidades serán distintas dependiendo de la distancia en altura a la que trabajemos de las piezas para evitar movimientos demasiado rápidos o bruscos cuando el robot se encuentra cerca de ellas y aumentando cuando se encuentra lejos para un movimiento más ágil y fluido.

El robot se desplaza primero a la posición de la pieza recibida a una altura de 200mm a velocidad máxima, ya que se considera que a esta altura no se va a encontrar ninguna pieza y se puede acceder a cualquier punto sin problema. Para obtener el offset de la posición deseada relativo al punto que se ha elegido como referencia basta con restar a la posición total el valor de referencia (6,5) y multiplicar este por la distancia conocida entre cada punto de la plataforma, que es de 16mm.

```
! Offset de la posicion respecto a p1
OffX := (X1 - 6) * distancia;
OffY := (Y1 - 5) * distancia;

! Ir a la posicion a una altura de 200mm
MoveL Offs(p1,OffX,OffY,0), v1000, fine, tool0;
```

Código 19. Cálculo de offstes

A continuación, en función de la altura a la que se encuentre la pieza, se calculan do valores distintos para los offsets en el eje Z. Estos valores son correspondientes a un punto intermedio, a 25mm de la pieza y la altura final a la que hay que bajar para cogerla. A este primer punto accedemos a una velocidad intermedia, y al último, a la velocidad mínima para trabajar con la pieza de la forma más segura posible y evitar así cualquier problema.

En el código 21, se encuentra un ejemplo de las instrucciones que se llevarían a cabo para coger una pieza que se encuentre a una altura uno o dos. En la aplicación final, habría que definir distancias intermedias y finales para todos los posibles niveles que pueden alcanzar las estructuras que se pueden construir.

Una vez la pinza se encuentra a la altura de la pieza, se procede a cerrarla para agarrar así la pieza. Antes de pasar a la siguiente instrucción, se esperan 0.5 segundos para asegurarse de que la pinza esté cerrada y ha cogido la pieza antes de intentar levantarla.

```
TEST n
   CASE 1:
      Zmedio := arriba-medio1;
      Zabajo := arriba - h1;
   CASE 2:
      Zmedio := arriba-medio2;
      Zabajo := arriba - h2;
ENDTEST

MoveL Offs(p1,OffX,OffY,-Zmedio), v2000, fine, tool0;

! Bajar hasta la posici?n de la pieza
MoveL Offs(p1,OffX,OffY,-Zabajo), v40, fine, tool0;

CerrarPinza;
WaitTime 0.5;
```

Código 20. Agarre de pieza

El siguiente paso consiste en levantar la pieza hasta una altura de 200 mm y esperar a recibir los datos sobre la posición de destino. El proceso para levantar la pieza

varía según dos casos diferentes. Si la pieza se encuentra a una altura 1, lo que quiere decir que no hay ninguna otra pieza debajo, el proceso se puede llevar a cabo de forma habitual, exactamente al inverso que el que se lleva a cabo para bajar hasta ella para cogerla.

Si la pieza se encuentra a una altura superior, esto quiere decir que está encima de otras, y la forma de levantarla tiene que tener en cuenta que es primero necesario desencajarla para asegurarnos de coger únicamente una pieza y no todas las que se encuentren en esa posición en niveles inferiores.

La solución al problema de desencajar piezas se basa en la realización de numerosas pruebas. En un primer momento se comprobó cómo al levantar la pieza en un solo movimiento vertical el robot levanta todas las piezas que estaban situadas debajo. Para evitar este problema, con la pieza agarrada, se gira la pinza hasta separar la pieza superior ligeramente del resto, y en esta posición, se procede a levantarla.

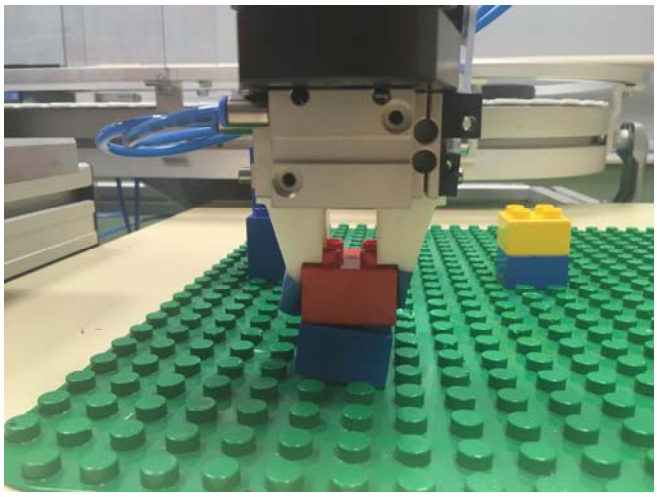


Figura 47. Desencajar piezas

El proceso de dejar la pieza es muy similar al de cogerla, siendo éste bastante más simple de conseguir. El robot se sitúa sobre la posición final para cada pieza según

el dato de color que haya recibido, con la pieza ya agarrada previamente y baja lentamente hasta la altura deseada, la pinza se abre y el robot sube hasta la altura de referencia (200 mm). Una vez en esta posición, espera a la siguiente comunicación para coger la siguiente pieza y repetir todo el proceso indefinidamente hasta que la comunicación se cierre una vez se haya ejecutado todo el código en Matlab.

Capítulo 6 RESULTADOS/EXPERIMENTOS

Para evaluar la efectividad de la aplicación se ha realizado el mismo experimento con varias configuraciones de piezas diferentes. El sistema cumple con su fin de procesar las imágenes, clasificar las piezas presentes y organizarlas correctamente.

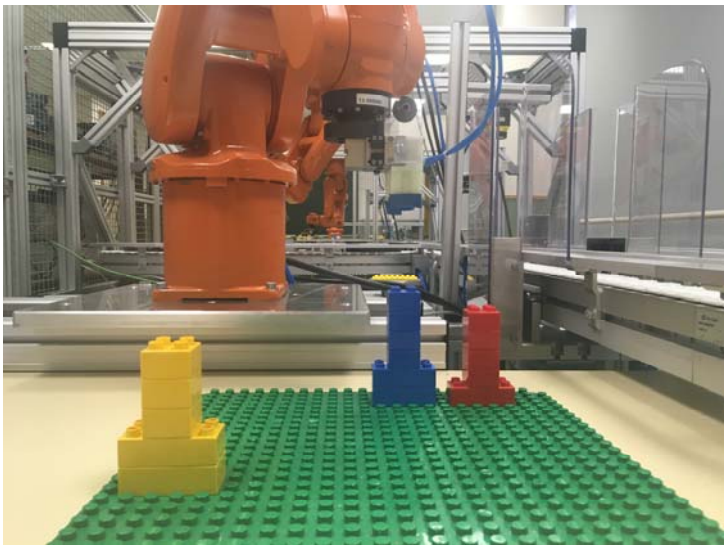


Figura 48. Resultado final

Para llevar a cabo los experimentos, se ha colocado la cámara detrás de la pinza del robot sujetándola con una pieza metal recortada a medida. Colocando la cámara de esta forma se obtienen las imágenes óptimas ya que se puede situar en el centro del espacio de trabajo eligiendo la altura más favorable para captar ambas imágenes, color y profundidad, con la mayor calidad posible. Una vez encontrada esta altura a base de pruebas, se colocan las piezas en el espacio de trabajo teniendo cuidado de no situarlas en posiciones no accesibles.

El procesado de imágenes en su totalidad se ejecuta en menos de un segundo, con una media de aproximadamente 0,63 segundos. La aplicación completa, incluyendo

todos los movimientos del robot, resulta ser de unos 25 segundos en modo automático para 4 piezas.

Para el correcto funcionamiento, se deben cumplir con tres condiciones. Primero, las piezas deben encontrarse en posiciones que se encuentren dentro del rango de trabajo del robot, esto reduce considerablemente las opciones de posicionamiento, pero es necesario evitarlo para que el programa no se pare por fallo de funcionamiento. En segundo lugar, no se debe colocar piezas simples encima de una pieza doble, ya que al encontrarse oculta parte de la pieza, la parte visible podría clasificarse erróneamente como una pieza simple, resultando en un fallo en el programa. Por último, el espacio de trabajo se encuentra situado bajo dos focos de luz, dificultando el procesado de las imágenes en esas áreas, por la tanto es recomendable evitar las posiciones de la matriz que se encuentren en estas áreas. En la siguiente figura se observar cómo, para una configuración concreta, todas las piezas son clasificadas correctamente menos la que se encuentra bajo el foco de luz:

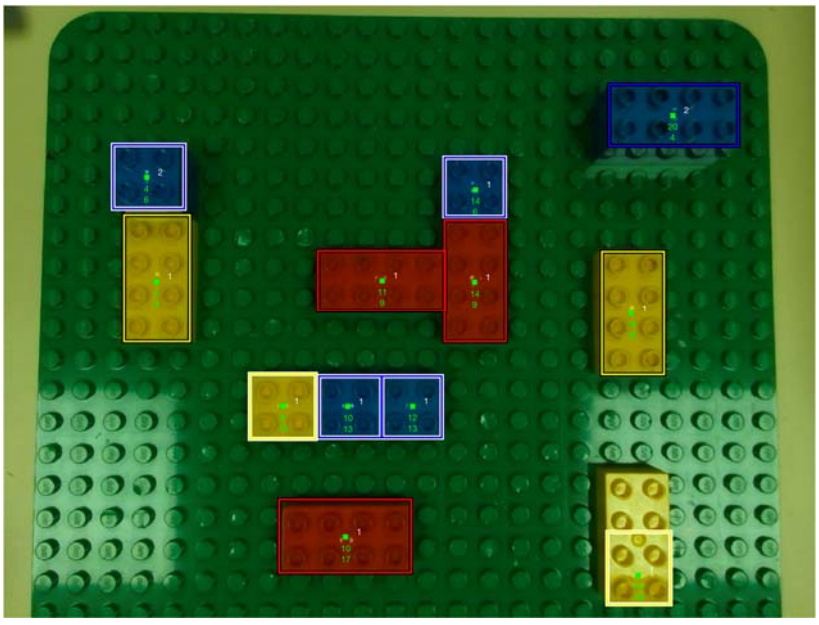


Figura 49. Ejemplo de fallo debido a luminosidad

Para comprobar el funcionamiento del sistema se han llevado a cabo diferentes pruebas. En estas pruebas, para la ejecución en modo manual, el robot en varias ocasiones no era capaz de desencajar unas piezas de otras, provocando un fallo en el sistema. En modo automático no se ha conseguido en ninguna ocasión. Por este motivo, se ha decido trabajar con las piezas colocadas a una única altura para evitar fallos en el funcionamiento, aunque el sistema sea capaz de reconocer correctamente las alturas de las piezas.

Capítulo 7 CONCLUSIONES

La integración de un sistema de visión artificial a uno de los robots de la minifábrica de ICAI constituye un paso más hacia la completa automatización de la planta. Aunque los objetivos de este proyecto son únicamente didácticos por el momento, ya que no hay una utilidad real detrás de esta aplicación, los métodos llevados a cabo podrían integrase, con modificaciones que se adapten a cada situación, en situaciones reales.

Al finalizar el proyecto se ha conseguido el objetivo de envío de datos obtenidos mediante procesado de imágenes, para su posterior movimiento en función de lo que se encuentre en el área de trabajo en cada momento, adaptándose a los cambios y sin necesidad de introducir los datos manualmente gracias a la visión artificial.

Capítulo 8 FUTUROS DESARROLLOS

La principal mejora que necesita este proyecto es el diseño de una pieza para su impresión en 3D con el fin de conseguir una eficaz sujeción de la cámara. Esta pieza debería ser diseñada para su acoplamiento en un punto estratégico del robot, que permita una total movilidad sin riesgos de chocar con otros objetos o entorpecer e incluso impedir el movimiento del robot.

Otro avance que se podría incluir sería la sincronización del robot con la cinta transportadora y el resto de robots que se encuentran en la planta, aumentando así las posibilidades y el alcance de este proyecto. De este modo se podrían realizar operaciones con fines más útiles y parecidos a los de una fábrica real.

En último lugar, una de las partes de este proyecto que no se ha conseguido a falta de tiempo, es el correcto levantamiento de las piezas desencajándolas de las que se encuentran debajo, por lo que se ha limitado la aplicación a configuraciones con piezas a una única altura, aunque el sistema sea capaz de reconocer diferentes niveles. Queda como futuro desarrollo el diseño de los movimientos necesarios del robot para el correcto levantamiento de las piezas y poder así trabajar a diferentes alturas.

Capítulo 9 PUESTA EN MARCHA DEL SISTEMA

Para poder trabajar con la cámara, hay que descargar diferentes archivos y programas en el ordenador con el que se va a trabajar. Esto se explica paso por paso a continuación.

Para interactuar con la cámara, los programas CMake y Anaconda deben estar instalados en el ordenador si se trabaja con Windows, todos los comandos para la instalación se llevan a cabo a través de la terminal. Los archivos necesarios para la instalación se encuentran en la siguiente página:
https://github.com/IntelRealSense/librealsense/releases/tag/v2.13.0

En primer lugar, una vez descargados los programas y archivos mencionados anteriormente, hay que crear un entorno específico en anaconda donde se instalará todo lo necesario, al crear este entorno es importante especificar la versión de Python con la que se va a trabajar, que debe ser la misma en todos los pasos. En este caso es la versión 3.6. Una vez creado el entorno y trabajando dentro de él, hay que crear un nuevo directorio en la carpeta donde se haya descargado la librería realsense y usando CMake se construyen las librerías en este nuevo directorio como se explica en la siguiente página en el apartado de instalación para Windows.
https://github.com/IntelRealSense/librealsense/tree/development/wrappers/python

Una vez hecho esto los únicos dos archivos necesarios que se deben copiar en la carpeta donde se sitúen todas las funciones de Python y Matlab con las que se va a trabajar son: ***realsense2.dll*** y ***pyrealsense2.pyd.*** El último paso es instalar numpy en Anaconda a través de la terminal para así poder trabajar con Python en Matlab.

Una vez realizado lo anterior, se consigue el acceso a la cámara con el ordenador, y se pueden obtener las imágenes deseadas tanto en Python como en Matlab mediante las funciones descritas en el apartado 3.1.

Para el correcto funcionamiento de la comunicación mediante sockets, al haberse definido el robot como servidor y Matlab como cliente, primero hay que ejecutar el código escrito en RAPID en el controlador del robot, y una vez se ha situado en la posición desde la cual se toman las fotos, ejecutar el programa en Matlab desde el ordenador y esperar hasta el fin del programa. La dirección IP que conecta ambos programas es la dirección del controlador del robot con el que se va a trabajar, y el puerto, para el uso de los ordenadores del laboratorio es el 1024 en todos los casos.

Capítulo 10 BIBLIOGRAFÍA

[1] Definición robot industrial - http://historiayusosdelarobotica.blogspot.com/2010/06/la-actualidad-de-los-robots.html [Último acceso: julio 2018]

[2] Funciones robots ABB – https://new.abb.com/central-america-caribbean/sobre-nosotros/tecnologia/robots-industriales [Último acceso: julio 2018]

[3] Página de inicio de ABB - https://new.abb.com/es [Último acceso: julio 2018]

[4] Definición visión artificial - https://es.wikipedia.org/wiki/Visión_artificial [Último acceso: julio 2018]

[5] Visión artificial en la Industria 4.0 – https://nunsys.com/vision-artificial-industria-4-0/ [Último acceso: julio 2018]

[6] Visión artificial en la Industria 4.0 – https://blog.infaimon.com/la-vision-artificial-industria-4-0/ [Último acceso: julio 2018]

[7] Sistemas VGR – https://blog.infaimon.com/robotica-guiada-por-vision-bin-picking-una-solucion-para-los-procesos-productivos/ [Último acceso: julio 2018]

[8] Image Processing Toolbox - https://es.mathworks.com/products/image.html [Último acceso: julio 2018]

[9] ABB - https://es.wikipedia.org/wiki/ABB [Último acceso: julio 2018]

[10] IRB120 - https://new.abb.com/products/robotics/industrial-robots/irb-120 [Último acceso: julio 2018]

[11] IRC5 - https://new.abb.com/products/robotics/es/controladores/irc5 [Último acceso: julio 2018]

[12] RobotStudio - https://new.abb.com/products/robotics/robotstudio [Último acceso: julio 2018]

[13] RAPID - https://en.wikipedia.org/wiki/RAPID [Último acceso: julio 2018]

[14] RAPID Reference Manual – https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf [Último acceso: julio 2018]

[15] RealSense D435 – https://click.intel.com/intelr-realsensetm-depth-camera-d435.html [Último acceso: julio 2018]

[16] Librería realsense - https://github.com/IntelRealSense/librealsense [Último acceso: julio 2018]

[17] Realsense viewer application – https://imagingsolution.net/wordpress/wp-content/uploads/2018/04/Buy_Intel_RealSense_D435_13.png [Último acceso: julio 2018]

[18] NumPy - http://www.numpy.org [Último acceso: julio 2018]

[19] Librealsense Python Wrappers - https://github.com/IntelRealSense/librealsense/tree/development/wrappers/python/examples [Último acceso: julio 2018]

[20] Read NumPy files - https://github.com/kwikteam/npy-matlab [Último acceso: julio 2018]

[21] Sockets de Internet - https://en.wikipedia.org/wiki/Network_socket [Último acceso: julio 2018]

[22] Andrés Senén Estremera, "Diseño de una estación de trabajo para el Robot IRB120. Control de cinta transportadora mediante IRC5" - https://ebuah.uah.es/dspace/bitstream/handle/10017/26937/TFG-Sen%C3%A9n-Estremera.pdf?sequence=1 [Último acceso: julio 2018]

[23] Example socket communication between Matlab and Robotstudio - https://www.youtube.com/watch?v=JGJt17c69Bk [Último acceso: julio 2018]

[24] Cono del espacio de color HSV - https://es.wikipedia.org/wiki/Modelo_de_color_HSV -/media/File:HSV_cone.jpg [Último acceso: julio 2018]

Parte II ESTUDIO

ECONÓMICO

Capítulo 1 ESTUDIO ECONÓMICO

En este apartado se analiza el presupuesto del proyecto. Para obtener el coste total se tienen en cuenta los elementos necesarios para llevar a cabo el proyecto y la mano de obra.

Para la ejecución de este proyecto, se necesita lo siguiente:

Elemento	**€/Unidad**	**Unidades**	**€ Total**
Robot IRB 120	14.000	1	14.000
Cámara Intel RealSense D435	153.89	1	153.89
Cable extensor USB 3.0 1.8m	4.5	1	4.5
Kit piezas LEGO	21.99	1	21.99
		TOTAL	14,180.38

Además del coste de maquinaria, se calcula el coste total de la mano de obra:

Elemento	**Horas**	**€/hora**	**€ Total**
Programación del robot	80	40	3,200
Programación	120	50	6,000
		TOTAL	9,300

El coste total del proyecto es de 23,480.38€. Este presupuesto no incluye las licencias de los programas usados ni la instalación y mantenimiento de la maquinaria.