COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI | ICADE | CIHS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Trabajo Fin de Grado

INTEGRACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL EN LA MANO DE UN ROBOT INDUSTRIAL

Autor: Ana Berjón Valles

Directores:

Jaime Boal Martín-Larrauri

José Antonio Rodríguez Mondéjar

MADRID

Julio 2019

Este trabajo fue escrito con LaTeX y compilado en TeXmaker usando la distribución TeX-2013. Las familias de fuentes usadas son Bitstream Charter, Utopia, Bookman, and Computer Modern. A menos que se indique lo contrario, todas las figuras fueron creadas por el autor usando Microsoft Visio®, Adobe Illustrator®, y MATLAB®.

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título "Integración de un sistema de visión artificial en la mano de un robot industrial" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2018/2019 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.: Ana Berjón Valles Fecha: 12/07/19

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Jaime Boal Martín-Larrauri Fecha: 12/07/2019

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: José Antonio Rodríguez Mondéjar Fecha: 12/07/2019

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 Dña. Ana Berjón Valles DECLARA ser el titular de los derechos de propiedad intelectual de la obra: "Integración de un sistema de visión artificial en la mano de 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 los 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 12 de Julio de 2019

ACEPTA

Fdo. ANA BERJÓN VALCES

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

COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI | ICADE | CIHS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Trabajo Fin de Grado

INTEGRACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL EN LA MANO DE UN ROBOT INDUSTRIAL

Autor: Ana Berjón Valles

Directores:

Jaime Boal Martín-Larrauri

José Antonio Rodríguez Mondéjar

MADRID

Julio 2019

Este trabajo fue escrito con LaTeX y compilado en TeXmaker usando la distribución TeX-2013. Las familias de fuentes usadas son Bitstream Charter, Utopia, Bookman, and Computer Modern. A menos que se indique lo contrario, todas las figuras fueron creadas por el autor usando Microsoft Visio®, Adobe Illustrator®, y MATLAB®.

A todos aquellos que me han
acompañado estos cuatro años

Agradecimientos

En primer lugar me gustaría agradercerle este proyecto a mis directores de TFG, especialmente a Jaime por haber estado ahí en todo momento, siempre pendiente de lo que necesitara. Estoy infinitamente agradecida por tu paciencia y amabilidad cada vez que picaba a tu puerta, por ayudarme con cada cosa que se me atravesara, por muy simple que fuera.

Estos cuatro años han significado mucho para mi, y he conocido a gente que ha hecho menos duras las tardes de estudio o las épocas de exámenes. A todos vosotros que habéis estado ahí desde el principio y sé que estaréis durante mucho tiempo más: Marina, Jose, Paula, Juan, Conchi, Dani, Sergio, Eagle... Sin vosotros esto habría sido prácticamente imposible. Gracias por pasar de ser mis compañeros de clase a ser mis compañeros de vida.
También he de mencionar a mis descubrimientos de este año: Beltrán y Mena. Habéis sido mis compañeros de laboratorio y de comidas durante todo este proyecto, de penas y alegrías. Sois un cachito más de este enorme proyecto.

Por último, pero no por ello menos importante, quiero agradecérselo a mis padres, por permitirme estudiar desde siempre lo que quiero y ponerle un punto de razón a mis decisiones. Por otra parte, a Bicho, porque por muy insufrible que sea siempre está ahí para apoyarme y ayudarme en cuanto puede.

Resumen

INTEGRACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL EN LA MANO DE UN ROBOT INDUSTRIAL

Autor: Ana Berjón Valles
Directores: Boal Martín-Larrauri, Jaime y Rodríguez Mondéjar, José Antonio
Entidad colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Los robot industriales presentan cada vez un grado mayor de autonomía en infinitos y muy variados ámbitos. En este contexto de automatización, se desarrolla el presente proyecto que busca implementar un sistema de visión artificial que haga posible que un robot industrial sea capaz de localizar, identificar y ordenar piezas de LEGO colocadas aleatoriamente sobre un espacio de trabajo. El sistema propuesto logra su objetivo siempre que las piezas se coloquen respetando una serie de restricciones.

Palabras clave: visión artificial, Robótica

1. **Introducción**
   La Cuarta Revolución Industrial se caracteriza por haber traído un mayor grado de automatización a las empresas, dotando a los robots de un mayor grado de autonomía. Es cada día más común encontrar en las industrias robots capaces de examinar elementos de su entorno y actuar en consecuencia, empleando diversos sensores como cámaras o láseres.

2. **Descripción del proyecto**
   En este contexto se ha elaborado el presente proyecto que busca incorporar un sistema de visión artificial a un robot, de manera que sea capaz de reconocer y ordenar piezas de LEGO DUPLO simples en función de su color (azul, amarillo y rojo), dispuestas de manera aleatoria apiladas o rotadas sobre un espacio de trabajo.

3. **Descripción del sistema**
   El sistema está compuesto por una cámara RGB-D, un ordenador con el programa MATLAB y un robot industrial, tal y como se muestra en la imagen. La cámara tomará una imagen de profundidad y otra de color desde la posición designada para ello y el programa procederá a analizarla.

   El primer paso es analizar la imagen de color, en la cual gracias al uso de máscaras de color, encontrará las piezas, su lugar en la matriz y si su ángulo de rotación (en caso de estar encajadas en la matriz de puntos ese ángulo será cero). En esta parte del proceso se emplean algoritmos como la transformada de Hough o el detector de bordes de Canny. A

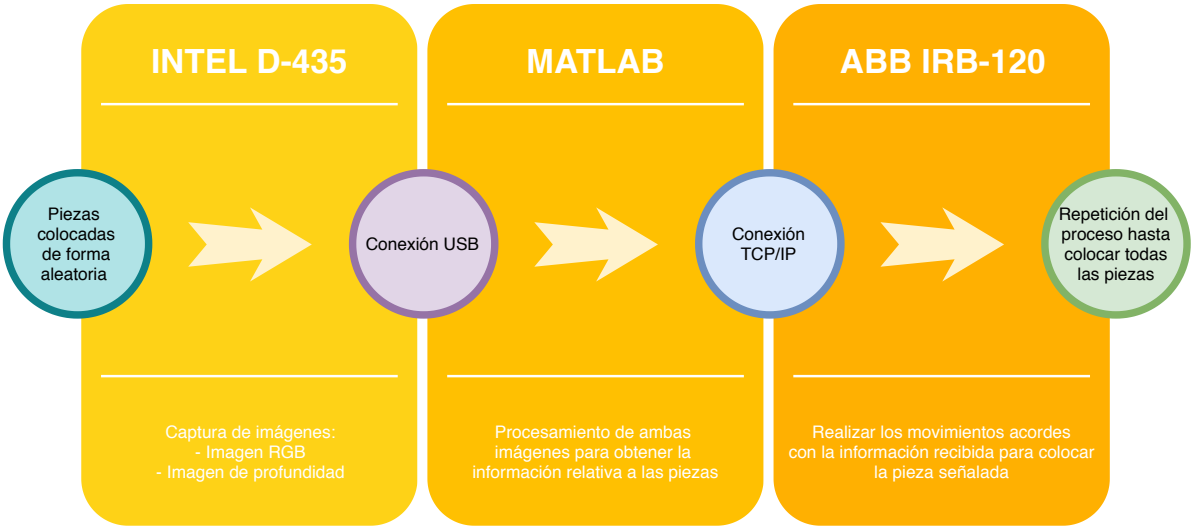


Figura 1. Esquema mostrando la arquitectura hardware del sistema.

continuación se estudia la imagen de profundidad, y se obtiene el valor de la distancia a la que se encuentra dicha pieza de la cámara. Gracias a este dato es fácil obtener el valor de la coordenada Z del robot para lograr cogerla.

Tras analizar ambas imágenes, el sistema ya posee la información relativa a las piezas y deberá transformar las coordenadas en el formato de la imagen a las coordenadas reales en el sistema del robot, datos que se le enviarán mediante una comunicación TCP/IP para que realice los movimientos designados. Finalmente, la programación del robot se ha realizado con el lenguaje RAPID, y se resuelven problemas como la rotación de un único eje de manera independiente o la extracción de piezas apiladas sin poder sujetar las piezas inferiores.

4. Resultados

El sistema acierta en la mayoría de los casos siempre que se respeten las restricciones de ordenación de las piezas. En cuanto a los casos fallidos, se pueden reconocer las situaciones en las que acontecen, principalmente al ubicar piezas bajo las luces del laboratorio o si hay una hilera de tres piezas en los extremos de la zona de trabajo. Las limitaciones existentes a la hora de colocar las piezas son las siguientes:

- Las piezas apiladas no deben superar una altura máxima de tres.
- Se debe verificar que la cámara no impactará contra piezas apiladas a la hora de recoger piezas simples.
- Se debe comprobar que la cámara no impactará contra piezas ya depositadas a la hora de colocar una nueva pieza

5. Conclusiones

Se ha diseñado un sistema capaz de identificar, localizar, recoger y ordenar piezas ubicadas de manera aleatoria sobre un espacio de trabajo, consiguiendo la totalidad de los objetivos propuestos con la limitación de las condiciones que deben cumplir las piezas de los escenarios:

Abstract

INTEGRATION OF AN ARTIFICIAL VISION SYSTEM IN THE HAND OF AN INDUSTRIAL ROBOT

Author: Berjón Valles, Ana
Supervisors: Boal Martín-Larrauri, Jaime and Rodríguez Mondéjar, José Antonio
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Industrial robots have an increasing degree of autonomy in infinite and varied fields. In this context of automation has been developed the following project, which aims to implement an artificial vision system that makes it possible for an industrial robot to be able to locate, identify and order LEGO pieces randomly placed on a workspace. The proposed system achieves its objective provided that the pieces are placed respecting a series of restrictions.

Keywords: Artificial vision, Robotics, Bin picking

1. **Introduction**
   The main carachteristic of the Fourth Industrial Revolution is bringing automation to a higher level in companies, making them more autonomous. Nowadays robots interact with the environment and act consecuently by applying different sensors such as cameras or lasers.

2. **Project definition**
   This is the context in which the present project has been developed, trying to incorporate an artificial vision system into a robot so that it is able to recognize and sort out LEGO DUPLO pieces according to their colour (yellow, red, blue) laying out randomly in the designed working area.

3. **Algorithm description**
   The designed system comprises a RGB-D camera, a computer with a MATLAB license and an industrial robot as shown in the figure. Once the camera is placed in the appointed place, it takes two images: depth and colour which will be sent to the program.

   First of all, RGB image is processed using colour masks in order to learn where are the pieces located, their colour and their rotation angle (if they are fixed in the matrix, this value will be zero). In this phase several artificial vision algorithms are used, as Canny Edge Detector or Hough Transform. Afterwards, the system will evaluate the depth image, from which the distance of the piece to the camera is found. Thanks to this value, the height at which the effector must be placed to grab the piece is known.

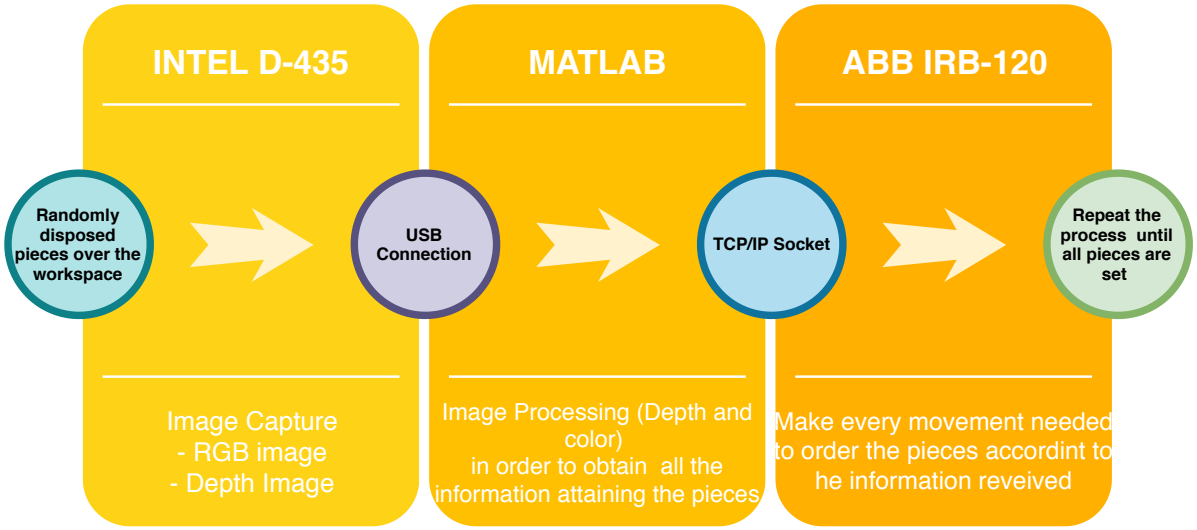


Figure 2. Diagram showing the hardware architecture of the system.

After analyzing both images, the system acknowledges all the information about the pieces and will now transform the coordinates in the image to those in the robot coordinate system. These data will be sent to the robot via TCP/IP socket so that it makes the propper movements. Finally, the robot has been programmed using RAPID language and a considerable amount of problems are solved, such as the rotation of a single axis or the extraction of the piece on top of a pile without taking the pieces immediately below.

4. Results

The system works correctly in most possible scenarios proven that restrictions in the disposal of the pieces are applied. In respect of the failures, there can be recognized a couple concrete situations in which they tend to happen, mainly when pieces are distributed under the reflection of the lights or when piles of pieces are located in the ends of the workspace. The restrictions previously mentioned are:

- Stacked pieces must not overcome a maximum height of three.
- Must be checked that the camera does not impaact piled pieces when trying to reach another one.
- Must be checked that the camera does not impact pieces already set when trying to deliver a new one.

5. Conclusions

It has been developed system able to identify, locate and sort pieces randomly set over a workspace, achieving the totallity of goals that had been set. This is a quite simple project compared to those actually implemented in the industry, however, it is able to grab rotated pieces, stacked pieces (up to three heigths) and simple pieces distributed randomly.

Contents

List of Figures

List of Tables

1

Introducción

Este primer capítulo presenta las razones fundamentales por las que se ha decidido llevar a cabo este proyecto, su principal objetivo y las herramientas empleadas en su ejecución. Además señala la organización del documento para facilitarle su lectura.

Actualmente nos encontramos en los albores de la Cuarta Revolución Industrial (Industria 4.0) [Arr18] o en la Segunda Era de las Máquinas [Bry14]. Toda revolución conlleva repercusiones, avances en la técnica y problemas en la sociedad, pero se espera que esta sea la que más trascendencia tenga. Esta revolución está teniendo lugar de manera muy rápida e integradora, afectando a sectores muy diversos de la sociedad a pesar de estar aun lejos de su culminación.

La Cuarta Revolución Industrial fue una iniciativa tomada por el gobierno alemán en 2011 con el fin de avanzar en la digitalización de la industria, interconectando productos y modelos de negocio. Se persigue lograr la implantación de la Industria 4.0 en un plazo de 10 a 15 años, siendo una medida que pretendía consolidar el liderazgo alemán en la ingeniería [LD17].

En el sector económico, esta revolución desafía la productividad de las personas, ya que su eficiencia es inferior a la de un robot en todas aquellas tareas que no requieran creatividad y originalidad. Los nuevos mercados precisan menos mano de obra ya que las máquinas ahora también son capaces de realizar procesamientos simples. Por otra parte, surgen nuevas necesidades derivadas de los nuevos sistemas que forman parte de nuestro día a día.

En cuanto a la implementación a nivel industrial, muchas fábricas y almacenes se encuentran repletos de robots que realizan tareas pesadas o peligrosas. La automatización se ha ido introduciendo a medida que a las empresas les ha ido suponiendo un aumento de beneficios, lo que explica que haya ocurrido con mayor celeridad en grandes empresas, para quienes supone un menor esfuerzo realizar el desembolso inicial.

En 2017, la producción industrial supuso un 25,1 % del PIB europeo [Age19] y esa tendencia continúa al alza. En ese mismo año, la Unión Europea firmó un compromiso público-privado que tenía como objetivo desarrollar las tecnologías necesarias para mejorar la industria y hacerla más competitiva.

Todas estas máquinas y robots ya implementados tienen una capacidad de procesamiento mínima: realizan tareas básicas previamente programadas y cualquier cambio en ello supone parar y reprogramar. Las tendencias actuales buscan dotar a estos robots de mayor capacidad para tomar decisiones. Esto requiere un mínimo de inteligencia básica y captación de información del exterior.

A la hora de recabar información existen muchos sensores a nuestra disposición: láser [Mar+94], cámaras con distintas características y especificaciones [KPK14] [BWG18], sensores de radiofrecuencias... Sin embargo, las cámaras son elementos principales en este tipo de proyectos, ya que son los sensores que aportan una mayor cantidad de información teniendo un precio razonable. De ahí surge el término "visión artificial", ideado por Larry Roberts en su tesis doctoral [Rob61].

Siguiendo las ideas iniciales de Roberts y empleando los desarrollos posteriores en este tema, el objetivo de este proyecto, que se explicará con más detalle en la sección cuarta de este documento, es la incorporación de una cámara RGB-D a un brazo industrial, de manera que sea capaz de reconocer piezas de lego situadas en un tapiz, cogerlas y ordenarlas por colores.

1.1. Motivación

La sociedad actual se fundamenta en la informatización y automatización de procesos repetitivos. Así es que cada día nos resulta más normal encontrar humanos y robots cooperando en numerosos lugares de trabajo. En estos lugares, los robots se encargan de desempeñar las tareas más aburridas y repetitivas, permitiendo que las personas se puedan centrar en trabajos basados en la creatividad.

Para lograr que esto sea posible, el robot debe tener un mínimo de autonomía, es decir, capacidad para procesar las diversas situaciones que se le puedan plantear de manera que sea capaz de reaccionar correctamente ante ellas. La consecución de esto es el resultado de un largo camino de entrenamiento y programación, que debe comenzar por tareas sencillas.

Este es el caso del presente proyecto que busca incorporar funcionalidades a los robots industriales de la universidad, mejorando el sistema actualmente implementado [Gar18]. Comparado con todos los sistemas que ya hay desarrollados en este sector, el proyecto tiene un alcance limitado, pero es necesario para poder avanzar en el desarrollo de estas tecnologías.

1.2. Objetivos

El objetivo principal del proyecto es conseguir el funcionamiento conjunto y fluido del sistema compuesto por un robot industrial ABB IRB120 y la cámara RGB-D modelo D435 de Intel. El sistema debe ser capaz de identificar, coger y ordenar las piezas de LEGO DUPLO por colores (azul, amarillo, rojo) y tamaño simple (2x2). Estas piezas se encontrarán colocadas de manera aleatoria sobre el área de trabajo en cualquier orientación, apiladas o no pero siempre con los tetones hacia arriba.

1.3. Herramientas

Para llevar a cabo este proyecto ha sido necesaria la utilización de *hardware* y software específico que será listado a continuación:

- Una cámara Intel RGB-D model D435, elegida por su relación calidad-precio y aprovechando que ya fue empleada en un proyecto anterior.
- Un robot de la empresa ABB (ASEA Brown Boveri) modelo IRB120, caracterizado por ser el más pequeño de la compañía. A pesar de su reducido tamaño, es capaz de levantar hasta 3 kg.

(a) **(b)**

Figura 1.1. Herramientas empleadas (a) Cámara Intel D435 y (b) Robot ABB IRB 120.

- Un set de piezas LEGO DUPLO con los colores (azul, amarillo y rojo) de tamaño 2x2. Estas piezas deberán ser recogidas por el robot.
- Para el procesamiento de imágenes se ha decidido emplear MATLAB, debido a la gran variedad de herramientas que ofrece.
- El MATLAB Intel RealSense *wrapper* será el encargado de tomar las imágenes de la cámara y pasarlas a MATLAB.
- ABB Robot Studio, programa desarrollado por la empresa suiza ABB para facilitar la comunicación con el robot.

1.4. Organización del documento

La memoria se distribuye en nueve capítulos, siendo el presente el primero de ellos, que introduce el problema del proyecto. El siguiente contiene la revisión de la literatura existente sobre temas de robótica, visión artificial y la combinación de ambos. El siguiente capítulo describe la arquitectura del sistema a nivel *hardware* y *software*. Los capítulos sucesivos seguirán la estructura del programa, analizando en primer lugar la obtención y el procesamiento de la imagen, la traducción de las coordenadas y la programación del robot. Los tres últimos capítulos recogen los resultados, las conclusiones y los futuros desarrollos del proyecto.

Además, se incluyen dos anexos, uno que describe las características de las piezas que sujetan la cámara al robot y otro que contiene la información relativa a la corrección de las posiciones de las piezas encajadas a diferentes alturas. Finalmente, se puede encontrar la bibliografía con las referencias citadas a lo largo del documento.

2

Estado del arte

Este capítulo encuadra el proyecto en el contexto de las tecnologías ya desarrolladas. Se indican algoritmos empleados en la elaboración de este sistema así como posibles alternativas realizando una revisión de la literatura publicada por diversos investigadores hasta la fecha. La información parte de los puntos más generales hacia la especificación de cada materia.

Desde aquellos primeros esbozos en la tesis de Roberts [Rob61], la técnica ha avanzado paulatinamente a lo largo de la historia. Actualmente los robots son capaces de realizar innumerables tareas sin necesidad de la intervención del ser humano.

Este capítulo tratará la situación en la que se encuentran las diversas tecnologías que participan en este movimiento, tanto la robótica que deberá agarrar el objeto como el tratamiento de imágenes que lo localiza e identifica. Primero de manera individual y finalmente, combinadas.

2.1. Robótica

La Federación Internacional de Robótica (IFR) define un robot industrial siguiendo las especificaciones de la norma ISO 8373 [dEst12] como “un manipulador multifuncional reprogramable controlado automáticamente en tres o más ejes”. Sin embargo, según la definición más empleada, un robot es cualquier pieza de equipamiento que tenga tres o más grados de movimiento.

Los robots industriales se pueden clasificar atendiendo a diversos motivos, como la situación del robot o la forma en la que se controla. Uno de los criterios de clasificación más importantes es el número de grados de libertad que tenga, es decir, los ejes de movimiento que tiene el robot.

Otro criterio muy importante se centra en el efector que emplea el robot. El efector es el elemento final, ubicado en la muñeca del brazo robótico que interactuará con el objeto. Hay infinidad de efectores distintos, diseñados con diversas funcionalidades, desde un recolector de paprika [Cho+18] hasta un efector multifuncional para cirugía [Kim+13].

Algunos de los efectores más comunes son los que se describen a continuación [Shr13]

- Pinza mecánica: dos o más "dedos" que se activan por un controlador abriéndolos y cerrándolos.
- Agarre neumático: ventosas, accionados mediante succión para sujetar objetos planos.
- Elementos magnetizados: empleados para sujetar piezas ferromagnéticas.
- Sistemas adhesivos: emplean sustancias adhesivas para sujetar materiales flexibles.
- Elementos mecánicos simples: como ganchos o palas
- Agarres duales: agarre mecánico con dos elementos de agarre en un mismo efector.

Finalmente, cabe destacar la clasificación en función de la forma de controlar al robot. Aquí encontramos cuatro categorías principales: control manual, a distancia, mediante programación o con una cierta inteligencia.

A pesar de que las tendencias muestran que en el futuro cada vez tendremos más robots ejecutando inteligencia artificial, actualmente la corriente en aumento son los robots colaborativos, también denominados "cobots". En la ISO 10218 [dEst12] se definen cuatro formas de colaboración: parada segura motorizada, guiado manual, limitación de fuerza y energía y control de velocidad y separación [VN16].

Actualmente, todo aquel que visite una fábrica industrial espera encontrarse con cientos de robots protagonizando las cadenas de montaje. No solo se encargan de realizar tareas monótonas o peligrosas de manera automática, sino que incorporan, en función de su desempeño, diversos tipos de procesamiento.

La industria automovilística destaca por el alto grado de implementación de estas nuevas tecnologías. En el caso concreto del grupo BMW, emplean cuatro tipos diferentes de robots, encargados de diversas tareas [Hub18] :

- SplitBots: se encargan de tomar las bolsas de materias primas y colocarlas en el almacén, asegurándose de que quedan perfectamente alineados y colocados en función del tipo de contenedor en el que se encuentran. Pueden identificar hasta 450 tipos diferentes.
- PlaceBot: ubicado en la cadena de montaje, emplea un sistema de reconocimiento visual para identificar el mejor punto de agarre de las cajas cargadas para colocarlo en su estantería correspondiente.
- PickBot: toma pequeñas piezas de bastidores concretos. Reconoce las piezas gracias a una red neuronal convolucional que ha sido sometida a deep learning y a continuación calcula el punto óptimo de agarre. Se calcula, que, a partir de 2019, este tipo de robot será capaz de reconocer y manejar hasta 50000 piezas distintas.
- SortBot: apila los contenedores vacíos antes de que vuelvan a entrar en circulación.

Este es el caso concreto de BMW, pero si examinamos otras empresas del sector, podemos encontrar niveles de automatización muy similares, como es el caso de Audi [Tan17] o Skoda [Bol18].

(a)

(b)

Figura 2.1. (a) Pickbot (Fuente: https://automotivelogistics.media/news/150061 y (b) Placebot (Fuente:https://www.mobilegeeks.de/artikel/industrie-4-0-und-digitalisierung-im-bmw-werk-regensburg-ein-rundgang/

2.2. Visión artificial

Como se ha indicado en el punto anterior, las tendencias actuales en robótica buscan conseguir que el robot tenga la mayor independencia posible. Para ello, es necesario que capten los detalles de su entorno, lo evalúen y en función de ello actúen de una forma u otra. Aquí es donde entra en juego la visión artificial, ya que las cámaras son los sensores que más información aportan.

Según un estudio presentado en la conferencia del IEEE (Institute of Electrical and Electronics Engineers) sobre visión artificial y reconocimiento de patrones, los modelos de reconocimiento de objetos ya superan a la visión humana identificando objetos en imágenes confusas e interpretan mapas de manera más eficiente. Sin embargo, aún están lejos en cuanto al reconocimiento de objetos y de escenas naturales [BI14].

Para lograr entender su entorno, los robots deben tomar una imagen y procesarla. Algunas de las técnicas más empleadas para ello vienen explicadas a en las posteriores secciones. No obstante, antes de comenzar a procesar la imagen, debemos verificar que se esta tomando correctamente, es decir, que la imagen se está tomando correctamente y de acuerdo con la realidad. Para garantizar esto, antes de comenzar a utilizar la cámara se debe calibrar, lo que se explicará en la siguiente sección.

2.2.1. Calibración de la cámara

En los sistemas de visión artificial, la calibración de la cámara es uno de los procesos fundamentales para un correcto funcionamiento del sistema, sin embargo es un tema cuya madurez podría decirse que llegó hace más de 20 años. Se considera que una cámara está calibrada si son conocidos la distancia focal, las coordenadas del punto principal y los parámetros de distorsión de la lente.

En este contexto entra en juego la fotogrametría, que según la definición de ASPRS (American Society for Photogrammetry and Remote Sensing) es el arte, la ciencia y la tecnología de obtener información fiable sobre los objetos físicos y el entorno a través de procesos de registro, medición e interpretación de imágenes, patrones de energía electromagnética radiante y otros fenómenos.

2.2.2. Escalado de grises

El procedimiento de escalado de grises es, generalmente, la primera técnica empleada a la hora de procesar una imagen. Es especialmente importante a la hora de detectar los bordes de los elementos presentes en la imagen.

Hay numerosos algoritmos de transformación de RGB a escala de grises (o *grayscale*) en función del parámetro a emplear. Se pueden tomar los valores (R, G, y/o B) máximos, mínimos, hacer medias entre ellos o ponderarlos [AMS18].

Esta transformación se realiza para facilitar el tratamiento de la propia imagen puesto que reduce tamaño de la imagen a procesar. Esto se debe a que solo precisa de un canal de color, limitando así a ocho bits.

Figura 2.2. Ejemplo de imagen pasada a escala de grises (Fuente: galería de imágenes de MATLAB).

Esta técnica también ayuda en la corrección de errores en la imagen, ya que se simplifica la búsqueda de píxeles corruptos. Cuando se produce la aparición simultánea de píxeles calientes (*hot pixels*), aquellos más brillantes que los de su alrededor, y píxeles muertos (*dead pixels*), que son aquellos permanentemente negros, aparece lo que se denomina “ruido de sal y pimienta” (*salt-and-pepper noise*) [ARM17].

2.2.3. Binarización de imágenes

Partiendo, habitualmente, de una imagen en escala de grises, este proceso limita el valor de los píxeles a cero o uno. En la literatura se han definido múltiples métodos para llevar a cabo esta transformación [Pra+17]. Sin embargo, lo más común es establecer un valor umbral y en función de si el valor del píxel es superior o inferior, se le asignará el valor 0 o 1.

Estos métodos se pueden clasificar típicamente en dos grandes grupos: los de procesamiento global, que establecen un umbral común para toda la imagen, y los de procesamiento local, que dividen la imagen en regiones. Los más empleados son los globales, ya que son mucho más simples y tienen aproximadamente el mismo nivel de eficiencia [SLT11].

2.2.4. Detección automática de bordes

Los bordes definen los límites de separación de las distintas regiones de una imagen y es el primer paso hacia el reconocimiento de objetos. En una imagen, un borde se define como un salto en la intensidad de los píxeles.

Los bordes, idealmente, se componen de un conjunto de píxeles que definen una forma concreta con un cambio drástico. En la realidad, las imágenes no están perfectamente definidas,

Figura 2.3. Ejemplo de binarización de una imagen. (Fuente: galería de imágenes de MATLAB)

con lo que este cambio en intensidad se realiza gradualmente a lo largo de varias hileras de píxeles [JK12].

Para llevar a cabo el reconocimiento de bordes se emplean numerosos métodos que quedan divididos en dos categorías: aquellos que usan operadores basados en el gradiente y los que usan operadores laplacianos. A continuación, se explican algunos de los más comunes.

2.2.5. Operadores basados en gradientes

Son los operadores clásicos para encontrar la dirección e intensidad del borde, fáciles de ejecutar, pero muy susceptibles al ruido. Entre estos se encuentran de Robert, Sobel o Prewitt [Jai+14].

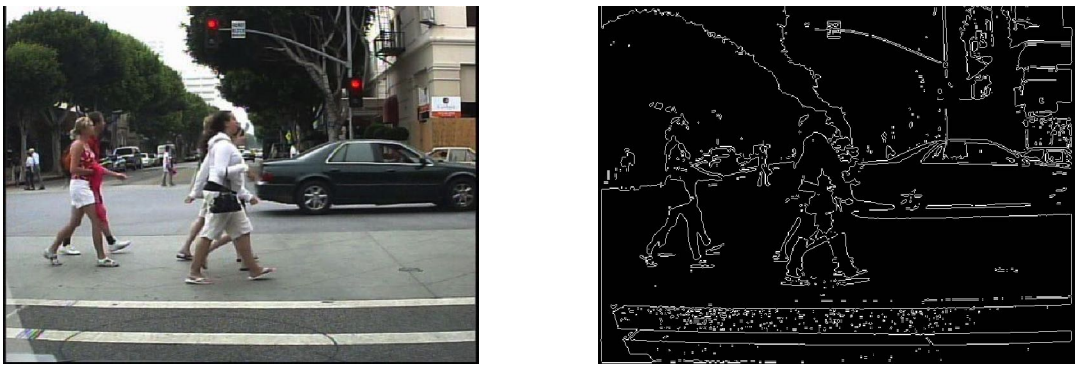


Figura 2.4. Resultado tras aplicar el operador de Prewitt. (Fuente: galería de imágenes de MATLAB)

Estos operadores se consiguen tomando cada imagen como una función bidimensional, donde cada par de coordenadas señala la intensidad de luz en un punto concreto de la imagen. Una vez obtenidos los valores de la dirección y el punto de mayor luminosidad, se emplean máscaras que permitan obtener la imagen con la detección de bordes.

La máscara empleada por el algoritmo de Robert (Robert Edge Detection) es un operador diferencial discreto, una matriz 2x2. Para ejecutar el algoritmo, se convoluciona[1] la máscara con

[1]La convolución es una de las operaciones más empleadas entre los ingenieros de comunicaciones, se define empleando la siguiente fórmula:

$$f[m] * g[m] = \sum_{n} f[n]g[m-n] \tag{2.1}$$

[CC09]

la imagen en horizontal y en vertical. Y se obtienen así los bordes detectados en las direcciones x e y respectivamente.

El algoritmo de Prewitt emplea una máscara que también es un operador diferencial discreto, pero en este caso se trata de dos matrices 3x3 que se convolucionan con la imagen en ambas direcciones. Este método es el más eficiente frente al ruido que sigue la distribución de Poisson (*Poisson noise*).

El tercer algoritmo más importante de esta clase es el de Sobel. La principal diferencia con el de Prewitt son los valores de la máscara, que da más peso a los píxeles del borde [CDG15].

2.2.6. Operadores laplacianos

Emplean la segunda derivada para conseguir una imagen más nítida, pero para reducir el efecto del ruido, hay que introducir métodos estabilizadores.

El algoritmo más importante en este grupo es el de Canny, y de todos los estudiados, es el más eficiente. Precisamente por ello surgen constantes modificaciones y mejoras a este algoritmo [JK12]. Un ejemplo del funcionamiento de este algoritmo se puede observar en la Figura 2.5.

El algoritmo tradicional comienza con un filtro gaussiano para suavizar las imágenes, difuminarlas y reducir el ruido. El siguiente paso es encontrar los gradientes y calcular su dirección y amplitud. Una de las ventajas frente a los anteriores algoritmos, es que solo señala como borde los máximos locales, eliminando así parte del efecto rampa[2]. A continuación, emplea un doble umbral para encontrar los bordes potenciales. Finalmente, encontrará los bordes definitivos, eliminando todos aquellos puntos que no estén conectados con un borde concreto. Esto se realiza mediante un proceso de rastreo por histéresis, buscando aquellos puntos próximos que realmente puedan formar un borde [YX15]. Este algoritmo se puede mejorar empleando filtros adaptativos, lo que puede ayudar a encontrar pseudo-bordes y limitar aun más el efecto del ruido.

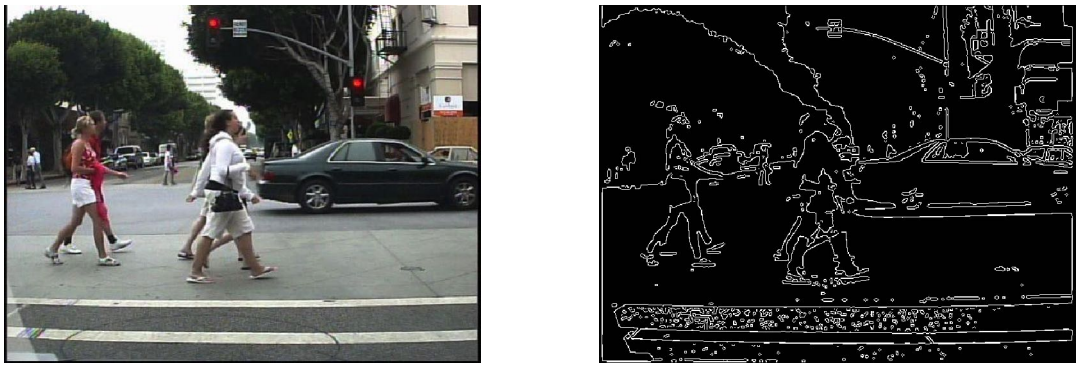


Figura 2.5. Resultado tras aplicar el operador de Canny. (Fuente: galería de imágenes de MATLAB)

2.2.7. Operadores basados en color

A pesar de que el 90 % de los bordes son iguales en escala de grises y en color [NS87], estos métodos son mucho menos empleados debido a las dificultades para identificar los colores. Se

[2]El efecto rampa se debe a que, en las imágenes habitualmente los bordes de los objetos no vienen delimitados por una única línea de píxeles, sino por varias que además van formando un degradado dificultando el reconocimiento del propio borde.

Figura 2.6. Comparativa de resultados al aplicar el algoritmo de Prewitt (verde) y Canny (morado). Las líneas blancas son aquellas en las que coinciden las detecciones de ambos algoritmos (Fuente: galería de imágenes de MATLAB).

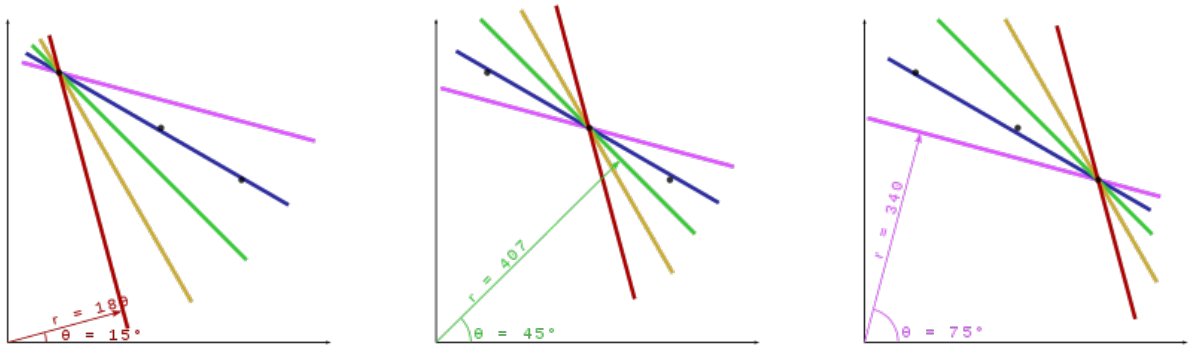
han propuesto detectores de bordes basados en color que buscan diferencias en vectores, en las diferencias de color propiamente dichas, en el valor umbral medio... Algunos llegan a rivalizar en eficiencia con el de Canny, pero su uso no está muy extendido [CC10].

2.2.8. Reconocimiento de líneas

En procesados posteriores de la imagen, no es suficiente con conocer los bordes en sí, y se buscan técnicas para conseguir más información acerca de los bordes detectados. Aquí podemos encontrar numerosos algoritmos diferentes entre los que destacan las distintas modalidades de transformada de Hough o RANSAC (*Random Sample Consensus*). Estos algoritmos surgieron con el fin de solventar el problema de localización y determinación (LDP) a partir de imágenes y se postulan como técnicas robustas a la hora de determinar modelos eliminando el ruido presente en las imágenes.

2.2.8.1. RANSAC

Este es un método iterativo que estima parámetros con un modelo matemático a partir de un conjunto de datos observados [FB81]. Como su propio nombre indica, este modelo toma muestras aleatorias de una imagen y busca un patrón que encaje con los datos seleccionados. Tras ello, toma los datos completos y verifica su modelo. En ocasiones, a la hora de analizar un conjunto de datos, encontramos que en la recopilación de algunos de ellos se han producido grandes errores. Estos pueden provocar una enorme desviación en los resultados, aun cuando el resto de la información esté perfectamente calculada y recopilada. De ahí surgió la necesidad de este algoritmo, que, a diferencia de otros como MCO (Mínimos Cuadrados Ordinarios), consigue eliminar un porcentaje significativo de estos errores.



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3

θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

Figura 2.7. Muestra de funcionamiento del algoritmo en tres puntos alineados. Se plantean los tres puntos negros, para cada uno de los cuales se dibujan líneas con distintas inclinaciones almacenando el valor de r (ρ) y θ. Al finalizar las iteraciones por todos los puntos verificará en función de la longitud de r si pertenecen a una misma recta (Fuente: https://en.wikipedia.org/wiki/Hough_transform)

Un problema básico en el procesamiento de imágenes es conseguir establecer una correspondencia entre dos o más elementos de una escena y en estos casos en los que las imágenes no son especialmente precisas, es necesario poder discernir aquellos puntos que solo aportan confusión. Se ha de determinar un umbral que señalará si un punto determinado pertenece al modelo o no y el número de datos necesarios para considerar que los datos se corresponden con un modelo concreto.

Entre las ventajas que presenta este modelo está su habilidad para determinar un patrón concreto a pesar de la existencia de un alto número de *outliers*. Sin embargo, cuenta con una gran desventaja frente al siguiente algoritmo: solo es capaz de determinar un modelo dentro de una misma imagen. Este problema se ha ido resolviendo a medida que se han ido presentando diversas alternativas y mejoras de este algoritmo, como es el caso del R-RANSAC (Recursive RANSAC) [NB16]

2.2.8.2. Transformada de Hough

La transformada de Hough es un método muy popular y robusto a la hora de detectar las características principales de las distintas líneas de una imagen. Su principal problema es su complejidad computacional y los recursos de memoria que precisa. Por ello se han creado múltiples variaciones a partir del método inicial.

El algoritmo original, conocido como Standard Hough Transform (SHT) analiza cada punto de la imagen haciendo pasar por él las líneas correspondientes a todos los patrones posibles. El sistema almacena los ρ y θ correspondientes a cada posibilidad, siendo ρ la longitud del segmento perpendicular a la posible línea estudiada partiendo desde el origen y θ el ángulo de dicho segmento con el eje X. El funcionamiento de este algoritmo se muestra en la Figura 2.7. Tras analizar todos los puntos, se observan aquellas posibilidades que concuerdan en un mayor número de puntos y se toman aquellas como resultado. Aparte de las modificaciones que se han ido realizando al algoritmo, y que se explican más adelante, también se ha mejorado este algoritmo [NPJ08].

Esto sirve para reconocer patrones simples, como líneas rectas, circunferencias o elipses, cada una identificada con una ecuación distinta. Como se ha indicado anteriormente, esto emplea muchos recursos del ordenador y es, por ello muy lento.

Por este motivo surgieron alternativas como las que se enumeran a continuación que funcionan de una manera muy similar optimizando recursos.

- Random Hough Transform (RHT: como se ha indicado anteriormente, la SHT se caracteriza por analizar cada píxel de la imagen, por ello su complejidad es directamente proporcional al tamaño de la imagen. Esta versión del algoritmo asume que determinados patrones se pueden reconocer a partir de un número n de puntos, con lo que se ahorra gran cantidad de procesamiento innecesario.
- Probabilistic Hough Transform (PHT: mantiene el esquema de acumulación del SHT, pero emplea un muestreado aleatorio, tomando un pequeño subconjunto de los puntos pertenecientes al borde que previamente ha sido identificado. Esto, una vez más, reduce considerablemente los tiempos de ejecución del programa mientras que el rendimiento apenas se ve afectado.
- Progressive Probabilistic Hough Transform (PPHT): mencionado por primera vez por [HYT98], esta modificación selecciona y analiza puntos aleatorios de la imagen añadiendo los resultados obtenidos a los acumuladores totales de la imagen. Se debe fijar un valor umbral que permita diferenciar entre el ruido de la imagen y los datos realmente relevantes. Tras agregar los resultados del punto analizado, se plantea si el valor de alguno de los acumuladores está por encima del umbral de ruido y de ser así, toma esa posibilidad como resultado. El problema de este algoritmo reside en los falsos positivos.
- Modified Hough Transform (MHT): este algoritmo presenta muchas ventajas frente a sus competidores y tiene posibilidades de aplicación tan variadas que, en la literatura, se pueden encontrar infinidad de variaciones diseñadas expresamente para cada objetivo. Entre estas, podemos encontrar opciones tan variadas como localización de iris [LZ07] o detección por radar [Jia+08].

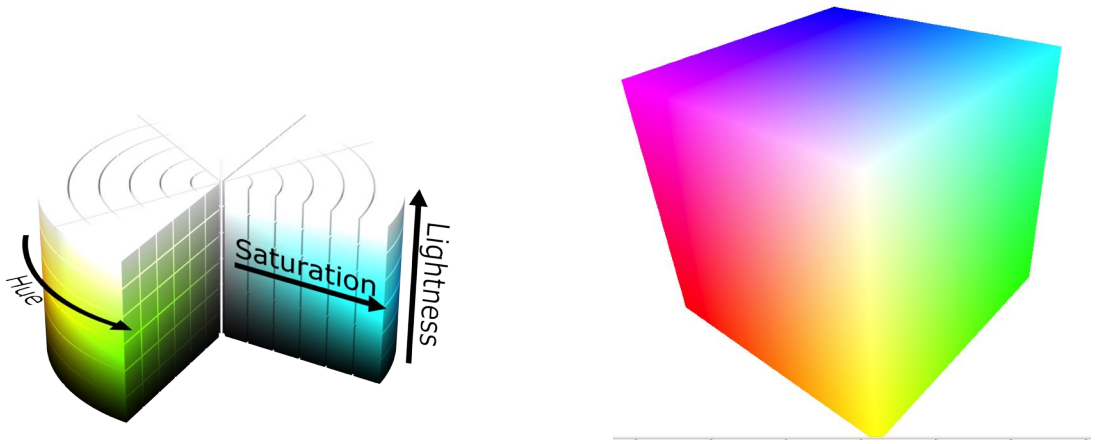
2.2.9. Detección de color

Hay muchos códigos de identificación de colores y sobre ellos destacan el RGB, el HLS y el YIQ. Se elaboran diversas máscaras o filtros que evalúan los valores de los diversos puntos de la imagen, clasificándolos y dividiéndolos. Es habitual que esto se realice en un proceso posterior a la detección de bordes, limitándose a buscar un color concreto en cada región delimitada.

2.3. Combinación

El hecho de conocer y agarrar objetos situados en cajas recibe el nombre de *bin picking* y es uno de los mayores desafíos en la actualidad. Para conseguir que el robot sea capaz de desempeñar estas tareas, el procesamiento debe seguir varias etapas. Hay dos opciones: primero reconocer y luego recoger o viceversa [Zen+18].

Para el reconocimiento de objetos más complejos se emplean las redes neuronales convolucionales (CNN), basadas en la percepción multicapa asemejándose así al funcionamiento



(a) **(b)**

Figura 2.8. (a) Diagrama de color HLS (Fuente: https://commons.wikimedia.org/wiki) (b) Diagrama de color RGB (Fuente: https://forum.luminous-landscape.com/index.php?topic=37695).

biológico de las neuronas. Cada capa procesa un aspecto de la imagen insertada obteniendo características cada vez más complejas.

Si primero reconoce y luego recoge, el sistema puede, mediante redes neuronales convolucionales reconocer el objeto a agarrar y en base a ello proceder de una forma u otra: succionando, empleando modo pinza, con un efector magnetizado... Esta implementación puede resultar muy útil en industrias en las que los elementos a agarrar son sensibles y frágiles. Al tener un número reducido de objetos a recoger, el reconocimiento se hace más sencillo, ya que hay menos posibilidades y por ello el porcentaje de acierto será elevado.

Sin embargo, con la otra aproximación, se emplea una red neuronal completamente conectada (FCN por sus siglas en inglés) para evaluar las posibilidades de éxito empleando cuatro formas de agarre diferentes y procede con aquella que más posibilidades de éxito presente.

Este segundo flujo de procesamiento permite ir aprendiendo cómo coger objetos desconocidos en base a la imagen obtenida independientemente de si reconoce el objeto o no, de forma que el sistema irá aprendiendo paulatinamente.

Para ser capaz de coger el objeto, el robot debe ser capaz de posicionarse correctamente. El término KGF (*Key Grasp Frame* o marco principal de agarre) determina una posición inicial para el efector y que deberá variar en función del tipo de objeto a recoger o el efector a emplear [Buc+13]. Para ello se emplea el algoritmo RANSAM (*Random Sample Matching*), que se basa en el ataque criptográfico *birthday attack*. Este algoritmo, mediante el estudio de un número limitado de puntos, es capaz de evitar las colisiones del efector con los límites del espacio en el que se encuentra. Otra ventaja de este algoritmo es que es capaz de reconstruir la superficie del objeto a agarrar empleando la misma técnica y, si fuera preciso, consultar las referencias obtenidas con una base de datos para reconocer el objeto [IKW09].

3

Arquitectura del sistema

Este capítulo define la estructura del sistema, desde la primera toma de datos mediante el uso de la cámara hasta la reacción por parte del robot procediendo a colocar las piezas.

El sistema se compone de tres elementos clave: la cámara, que toma las imágenes de color y profundidad, el ordenador, que las procesa y extrae toda la información relevante y, finalmente, el robot, que es quien ejecuta los movimientos de la pieza. Para que todo esto sea posible, debe establecerse correctamente la comunicación entre los componentes del sistema. La cámara se conectará al ordenador empleando un cable USB y el ordenador y el robot se conectarán mediante un socket TCP/IP.

La cámara se encuentra sujeta al robot mediante una pieza diseñada en Solid Edge y que la fija como se muestra en la Figura 3.1, protegiéndola ante posibles choques y permitiendo la ventilación. Al comenzar el proceso, el robot se colocará en una posición determinada, con la cámara paralela a la superficie de la mesa para tomar la foto correctamente. Ambas imágenes, la de color y la de profundidad, serán recogidas por MATLAB en el ordenador.

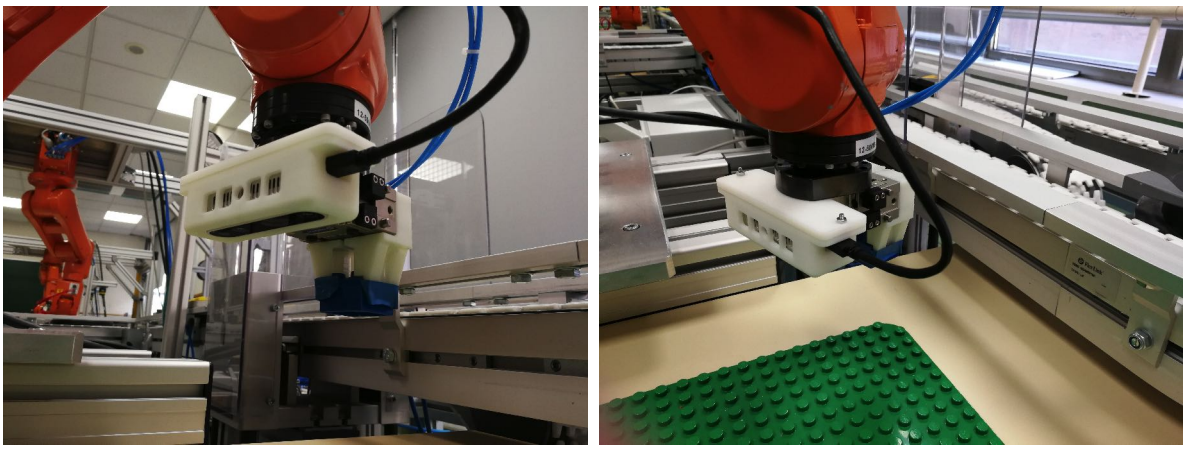


Figura 3.1. Pieza diseñada para sujetar la cámara al robot colocada en su posición

En este programa las dos imágenes serán procesadas y analizadas de manera independiente: primero se analizará la imagen de color, de la que se obtendrá toda la información relativa al color y la posición de las piezas en el plano XY: tanto su localización en la foto como el ángulo de rotación de la pieza. A continuación se analizará la imagen de profundidad, obteniendo, en milímetros, la distancia desde la cámara hasta la pieza.

Una vez almacenados todos estos datos en un vector, se deben transformar las coordenadas calculadas al sistema de referencia del robot, y enviárselas junto a la información sobre el color de la pieza, el ángulo de rotación o la altura a la que se debe depositar (la posición ya la tiene el robot almacenada). Antes de enviar esta información al robot, se ha tenido que iniciar la comunicación. Para ello, se debe poner el robot, que hace las veces de servidor, a la escucha en el puerto designado para que cuando MATLAB, que funciona como cliente, solicite conexión, pueda aceptarlo.

Tras ordenar todas las piezas visibles en la primera imagen, es decir, las de la parte superior de cada hilera, el robot volverá a su posición inicial para tomar otra imagen y volverá a comenzar el proceso. Esto se realizará hasta que no queden piezas por colocar en la zona de trabajo.

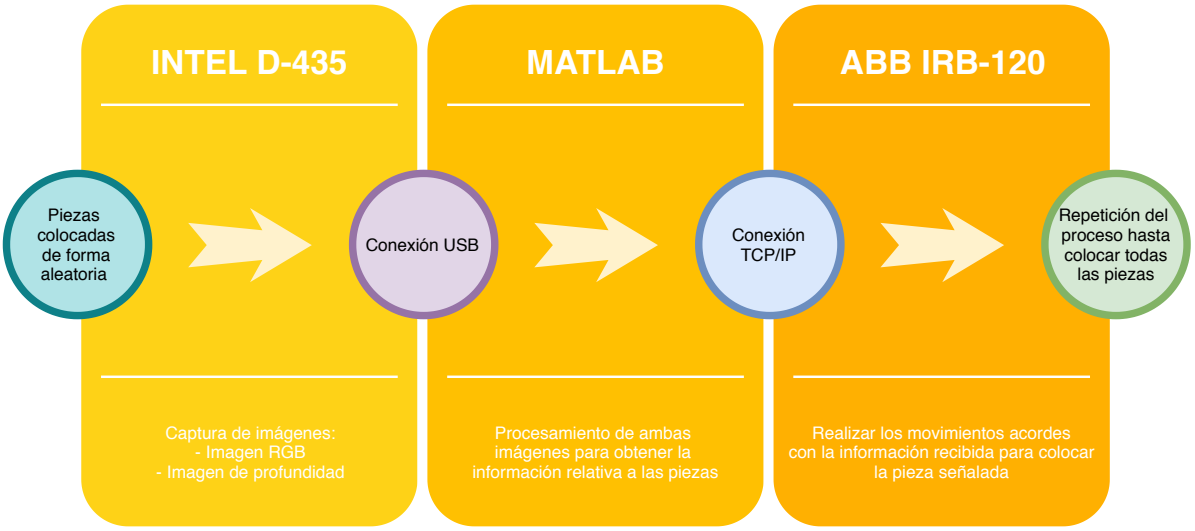


Figura 3.2. Esquema mostrando la arquitectura hardware del sistema

El esquema de la lógica interna del programa se muestra en la Figura 3.3. El primer paso es localizar los objetos presentes en la imagen para poder proceder a su identificación: posición, color, ángulo de rotación y altura. A continuación se determina el punto desde el que se agarrará la pieza y finalmente se enviará toda esa información al robot, que es quien realizará los movimientos pertinentes.

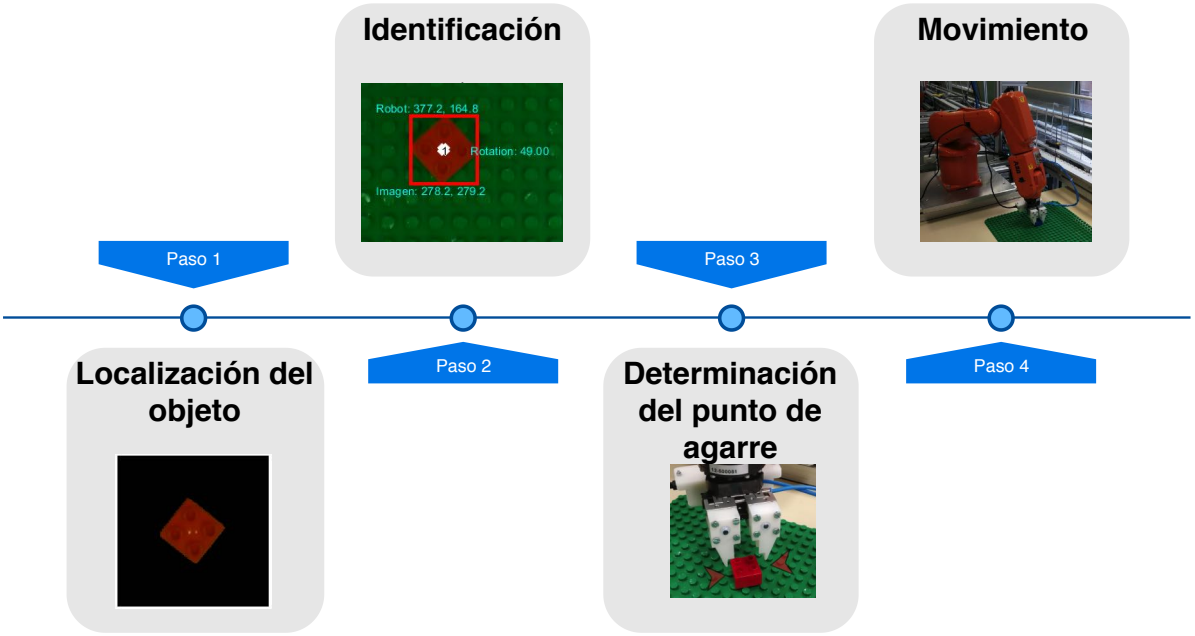


Figura 3.3. Esquema mostrando la arquitectura lógica del sistema.

4

Procesado de imágenes

El presente capítulo describe de forma detallada el trabajo realizado en cuanto al procesamiento de imágenes, desde la captación hasta la obtención del último dato relevante. Para ello se repasan los procedimientos empleados en las imágenes de color y de profundidad.

La primera pieza fundamental de este sistema es la cámara, elemento encargado de obtener las imágenes necesarias para procesar el entorno y obtener así los detalles acerca de las piezas: dónde están, o de qué color son.

4.1. Captación de imágenes

Para llevar a cabo la adquisición de imágenes, se ha empleado el SDK (*Software Development Kit*) de Intel[1]. Se ha aprovechado la reciente creación del *wrapper* para Matlab[2], que permite tomar las imágenes directamente en este programa sin la necesidad de intermediarios como Python o Node.js.

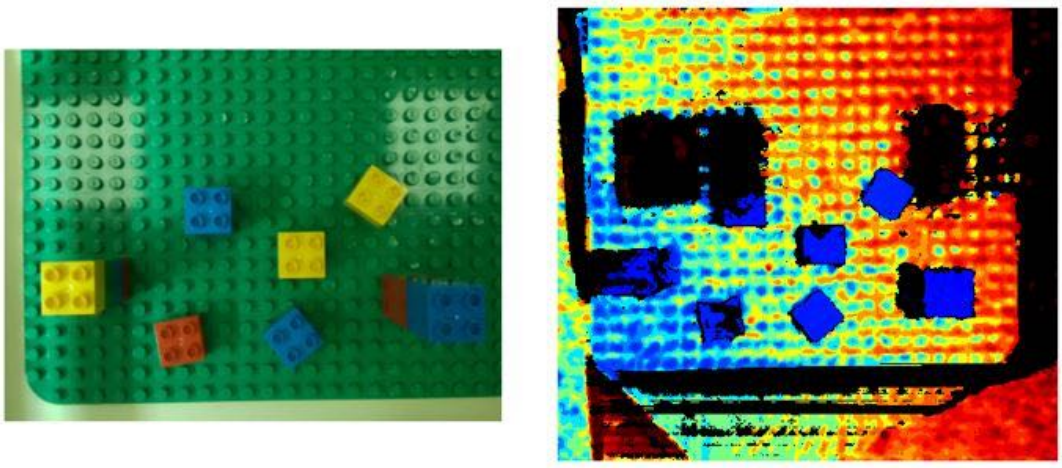
Empleando la librería, se procede a abrir un canal de comunicación con la cámara a través del puerto USB, se toman dos *frames*: uno de color y otro de profundidad y se transforman empleando las distintas funciones diseñadas por Intel para lograr el resultado esperado.

Los resultados obtenidos al terminar la adquisición se muestran en la siguiente figura, con una imagen de color de una resolución de 640x480x3 en formato RGB y una de profundidad de tamaño 720x1280x3, también en formato RGB. La imagen de profundidad ha obtenido los colores a partir del objeto colorizer, quien da un color negro o azul a objetos más próximos y un color amarillo a los más alejados.

En la Figura 4.1 se puede observar un elemento que después será clave en los fallos del sistema: los reflejos de la luz sobre la matriz. En el caso de la imagen de color suponen un problema menor que en la de profundidad. En el primer caso provocan que las piezas presenten

[1] https://software.intel.com/en-us/realsense/sdk
[2] https://github.com/IntelRealSense/librealsense/tree/master/wrappers

una tonalidad distinta a la habitual, lo que muchas veces provoca que el sistema no llegue a reconocerlas. Sin embargo, en el caso de la imagen de profundidad este fallo puede llegar a producir que el sistema entre en un bucle infinito, como se explica en Sección 4.3.

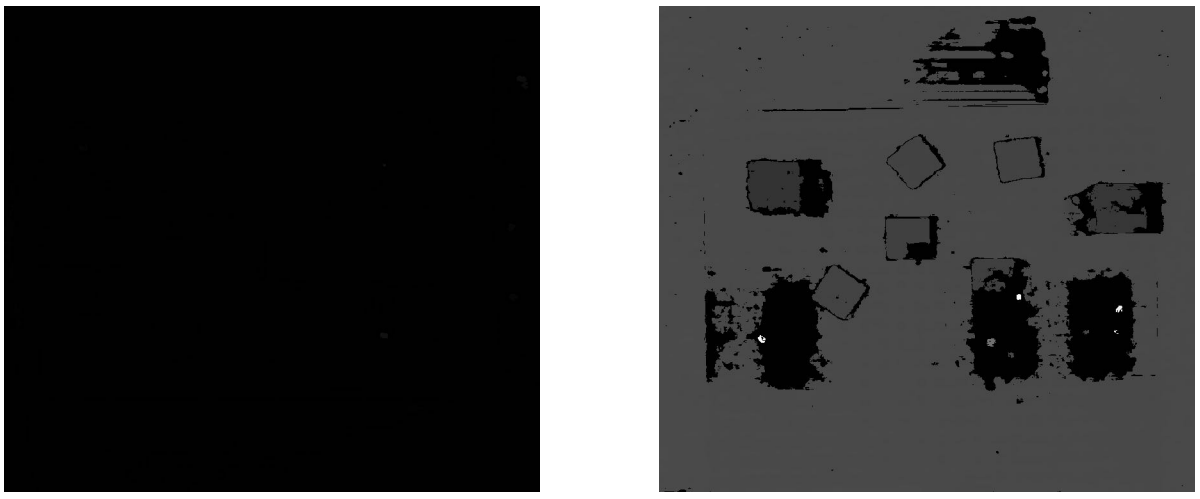


(a) (b)

Figura 4.1. (a) Imagen de color RGB tomada desde el punto de imágenes. (b) Imagen de profundidad coloreada.

En cuanto a la imagen de profundidad y de cara a facilitar la obtención de la información necesaria (la altura a la que se encuentra la pieza), se evita llamar al colorizer, obteniendo así información en formato uint16. De esta manera, la imagen de profundidad pasará a ser una única matriz de 720x1280 en cuyas celdas se almacena lo que simultáneamente es la información de los píxeles de la imagen y la distancia desde la cámara hasta ese punto concreto.

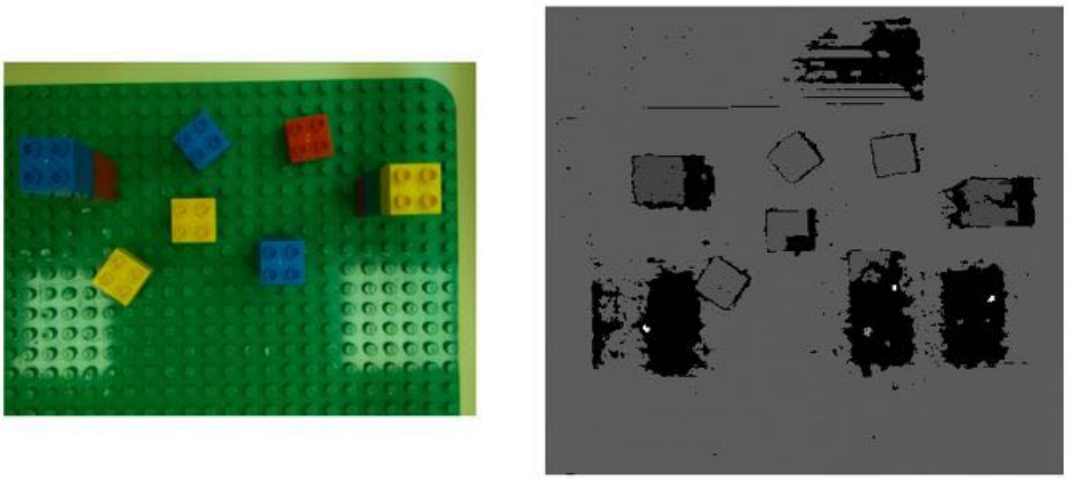
Para mostrar la imagen correctamente, debemos restringir los valores en ella, limitando el valor superior a 1024. En la Figura 4.2 podemos observar la representación de la imagen de profundidad sin limitar el valor máximo y limitándolo.



(a) (b)

Figura 4.2. (a) Imagen de profundidad sin limitación de valores (b) Imagen con limitación máxima de valores a 1024.

Antes de devolver las imágenes tomadas (RGB y profundidad sin colorear) al programa principal, las rotaremos 180 grados, de manera que la imagen que procesa el programa tenga la misma orientación que el usuario colocado en la zona de trabajo.

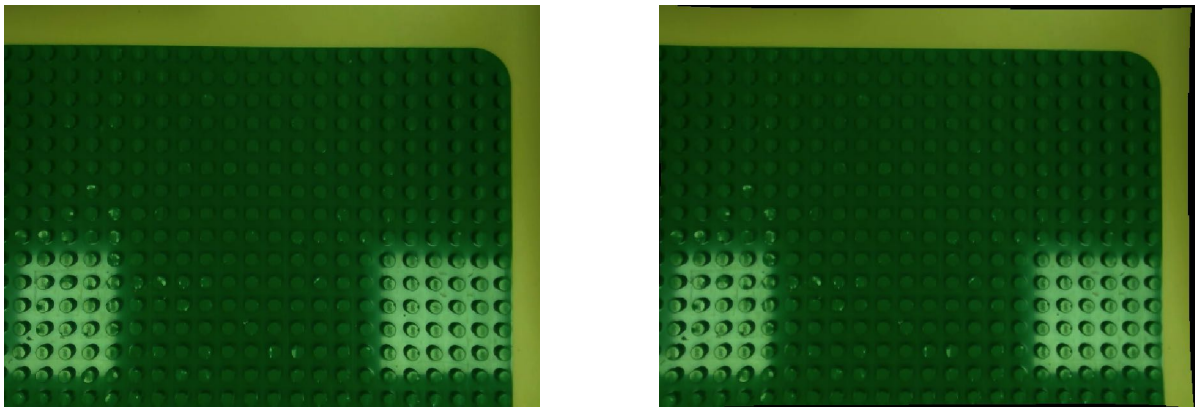


(a) (b)

Figura 4.3. (a) Imagen de color RGB tomada desde el punto de imágenes rotada 180º y (b) Imagen de profundidad sin colorear rotada 180º.

4.2. Procesado de la imagen de color

Como se indicó en el Capítulo 3 el primer paso es procesar la imagen de color para obtener los datos relativos al posicionamiento de la pieza en el plano XY. La cámara empleada presenta cierto grado de distorsión de la imagen en los extremos de la misma, el efecto conocido como "ojo de pez". Para corregirlo, se ha utilizado la función de calibración de cámara de MATLAB. Fue necesario emplear un tablero de ajedrez impreso, en este caso con una cuadrícula de 23 mm y tomar un mínimo de diez fotos del mismo tablero colocado en distintos ángulos (con la cámara siempre fija). A continuación, se suben estas imágenes a la aplicación, que generará automáticamente una serie de coeficientes y funciones destinadas, entre otros, a corregir la distorsión.



(a) **(b)**

Figura 4.4. (a) Imagen antes de aplicar los parámetros de calibración y (b) Imagen resultante tras corregir la distorsión.

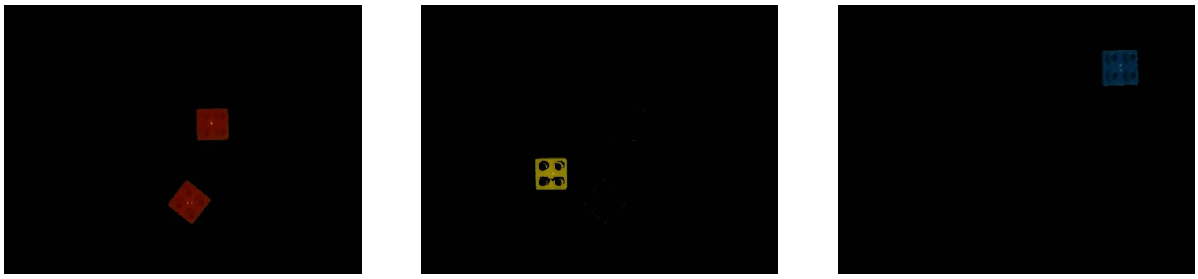
Posteriormente, se tapará la zona de la imagen en la que se depositan las piezas (zona de la izquierda) para evitar que el sistema las tome por piezas a colocar y entre en un bucle infinito. En la Sección 6.3.1 se hace referencia a las posiciones designadas para depositarlas indicando por qué se definieron esos lugares.

Figura 4.5. Imagen procesada, tapando la zona en la que se depositarán las piezas.

Una vez llevadas a cabo todas estas modificaciones, se procede a buscar las piezas de cada color. Para ello se emplearán las máscaras de color: funciones generadas por MATLAB que identifican las características del color deseado y lo aíslan del resto de la imagen. Esto se puede realizar en cualquier código de identificación de color, véase Sección 2.2.9, pero en este caso el código óptimo es HSV (Hue, Saturation, Value) ya que capta con mayor acierto las distintas tonalidades de las piezas en función de la luz de la sala.

Las imágenes con las que se trabajará ahora son las devueltas por la función de la máscara de cada color, y se tratarán de manera independiente, de forma que primero se obtendrá la información relativa a las piezas azules, luego a las amarillas y por ultimo a las rojas. Esta función devolverá dos imágenes: la máscara aplicada, mostrada en la Figura 4.7 y una imagen binaria (Figura 4.6), con unos en la posición de las piezas del color analizado y ceros en el fondo.

A continuación se realiza un primer estudio de las piezas de la figura. La única modificación que se hace antes de este análisis es eliminar aquellos píxeles agrupados inferiores a un determinado tamaño que, sin ser realmente piezas, pudieran llevar al programa a confusión.
La primera información que obtenemos es un número de piezas provisional, ya que si hubiera



(a) (b) (c)

Figura 4.6. Para cada posible color, imagen devuelta tras la aplicación de la máscara mostrada en la Figura 4.7 con la distribución mostrada en la Figura 4.5.

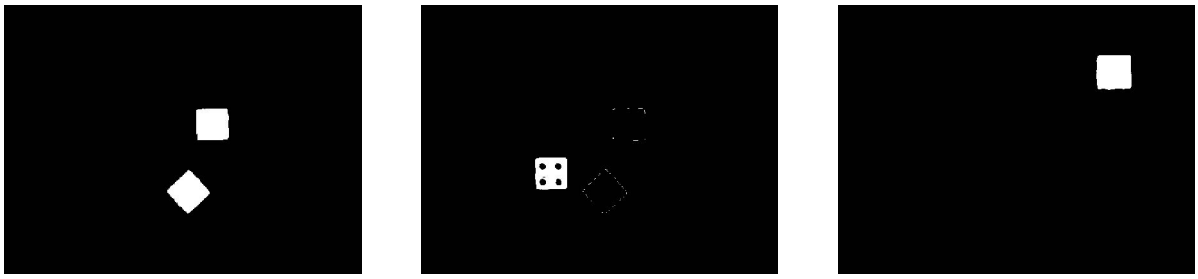
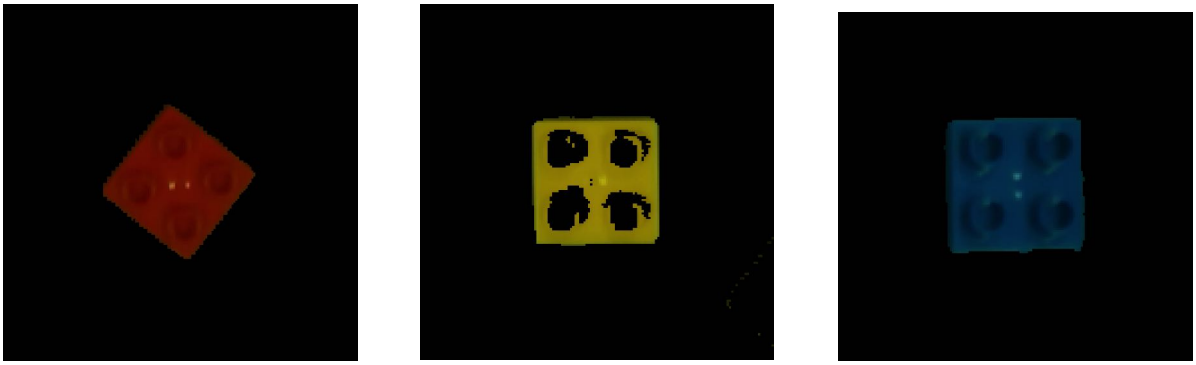


Figura 4.7. Máscara aplicada a la imagen principal por cada color.

dos piezas al lado, es muy probable que el programa aun no llegara a diferenciarlas. De estas piezas provisionales obtenemos información como el tamaño o la posición del centro.

A la hora de realizar un examen más exhaustivo, no se tomará toda la imagen, sino cada pieza previamente encontrada. Puesto que ya se ha obtenido la información relativa al color de

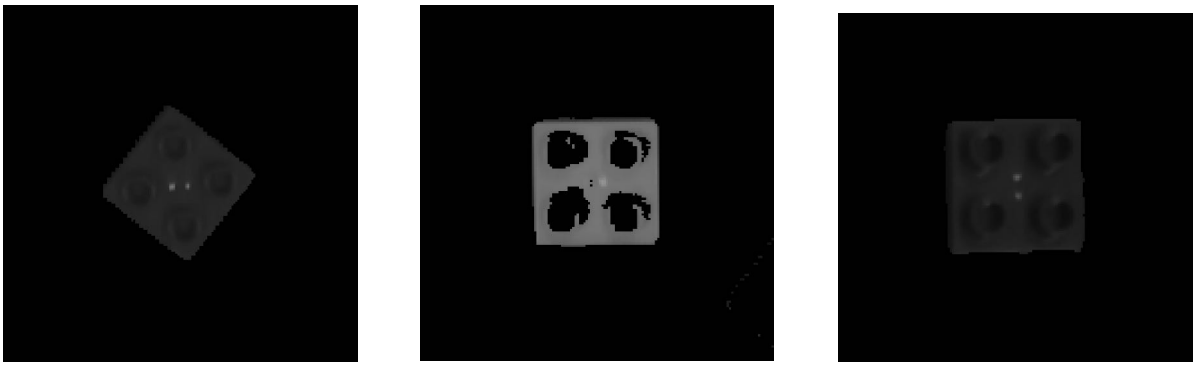


(a) (b) (c)

Figura 4.8. Primer paso en el procesamiento de imágenes: selección de la pieza a analizar.

la pieza, esto se convierte en información sobrante, que dificulta el posterior procesado. Por ello se pasa la imagen a escala de grises y se ajusta la intensidad de los valores, como se puede apreciar en la Figura 4.9.

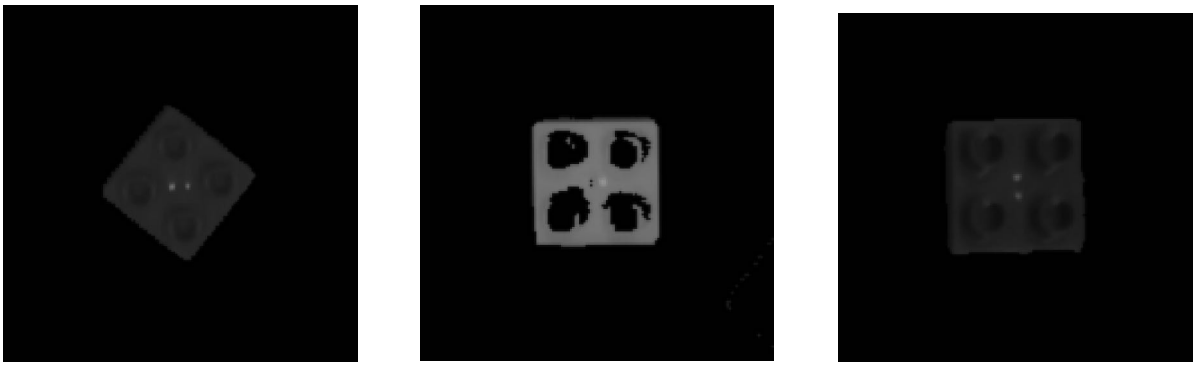
Ahora llega el momento de resaltar los detalles de la imagen. Esto se obtendrá mediante la siguiente secuencia: primero la se difuminará empleando un filtro gaussiano, como se puede observar en la Figura 4.10. Si ahora se resta la imagen difuminada de la imagen original, se



(a) **(b)** **(c)**

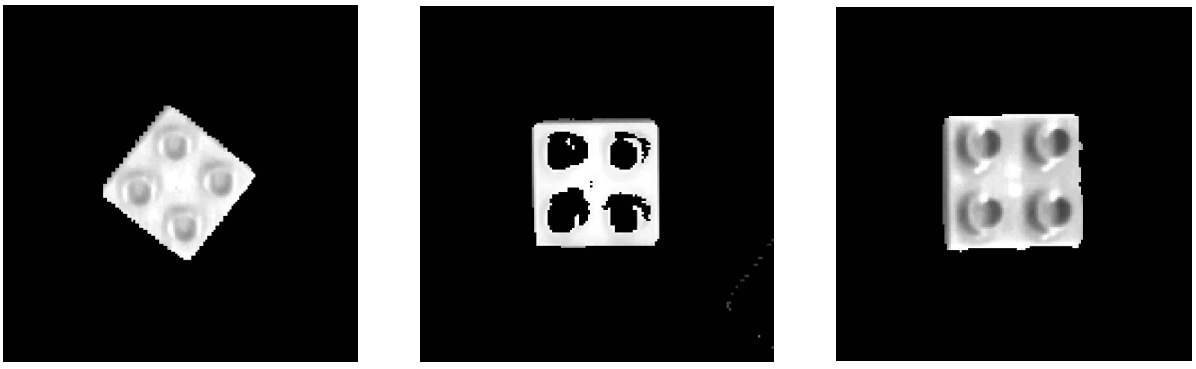
Figura 4.9. Segundo paso en el procesamiento de imágenes: escalado de grises.

obtendrán los detalles de la imagen como tal y, al sumárselos a imagen original, estarán siendo resaltados, Figura 4.11.



(a) **(b)** **(c)**

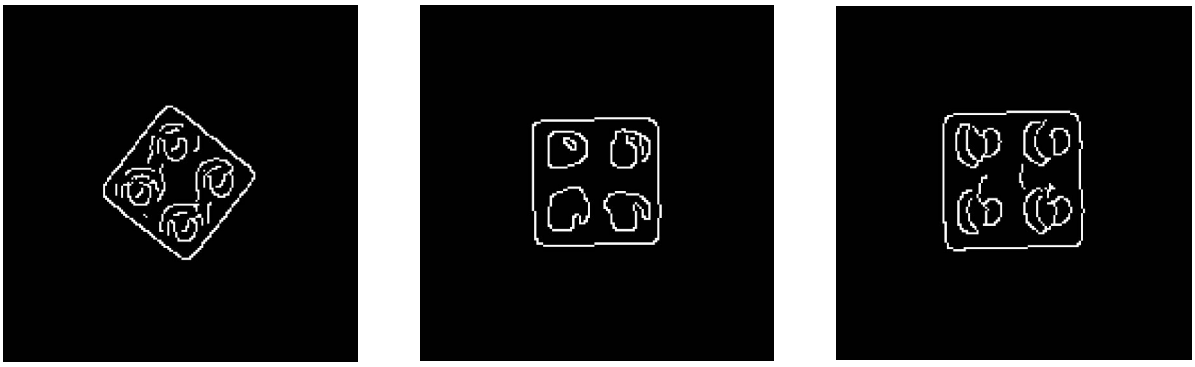
Figura 4.10. Tercer paso: imágenes difuminadas mediante la aplicación de un filtro Gaussiano.



(a) **(b)** **(c)**

Figura 4.11. Cuarto paso: imagen con los detalles resaltados.

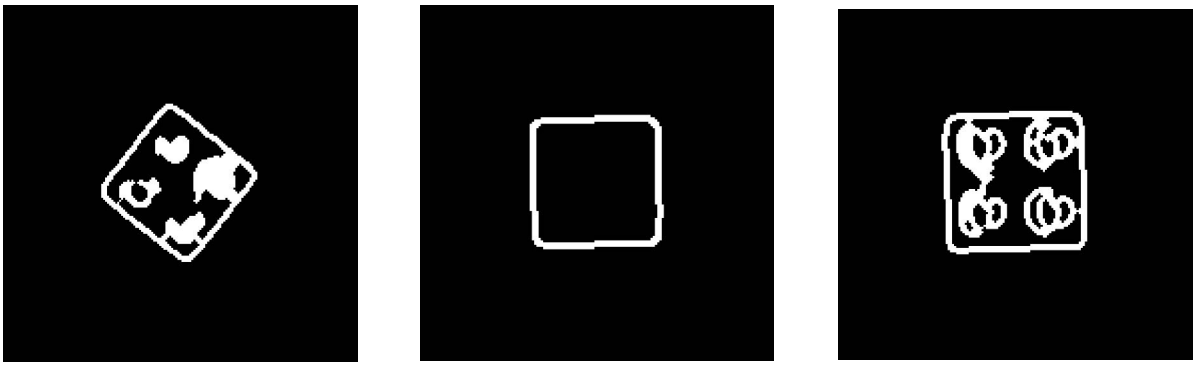
Una vez preparada la imagen, es el momento de aplicar el algoritmo de detección de bordes. En este caso, se usará el algoritmo de Canny que, como ya se explicó en Sección 2.2.6, es el algoritmo más eficiente entre los de su clase. Tras aplicarlo, se obtendrá una imagen como las de la Figura 4.12.



(a) **(b)** **(c)**

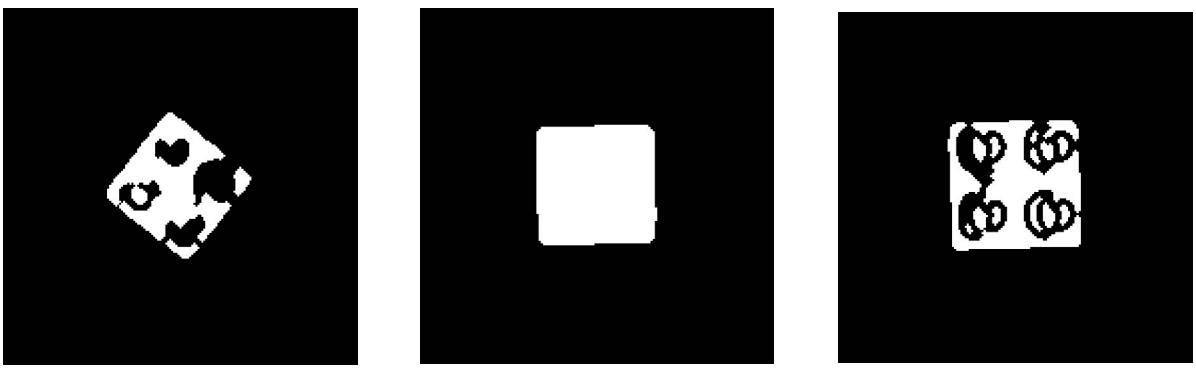
Figura 4.12. Quinto paso: aplicación del algoritmo de detección de bordes. Algoritmo empleado: Canny.

Se dilata y se contrae la imagen de manera que se vuelvan a eliminar los elementos pequeños que pudieran llevar a confusión y ya solo queda invertir los colores, de manera que la imagen analizada no sea un simple borde, sino un bloque. Tras esta inversión de colores, está la imagen lista para obtener toda la información acerca de la ubicación de las piezas: la posición de su centro o el tamaño de la pieza.



(a) **(b)** **(c)**

Figura 4.13. Séptimo paso: dilatación y contracción de la imagen con el fin de eliminar pequeños detalles que pudieran llevar a equivocación a la hora de procesar la imagen

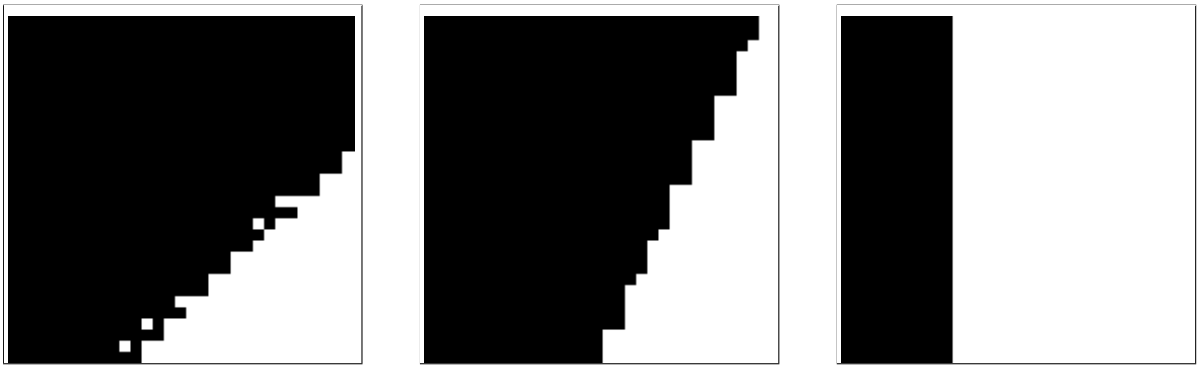


(a) **(b)** **(c)**

Figura 4.14. Octavo paso: inversión en los colores de la pieza

4.2.1. Piezas rotadas

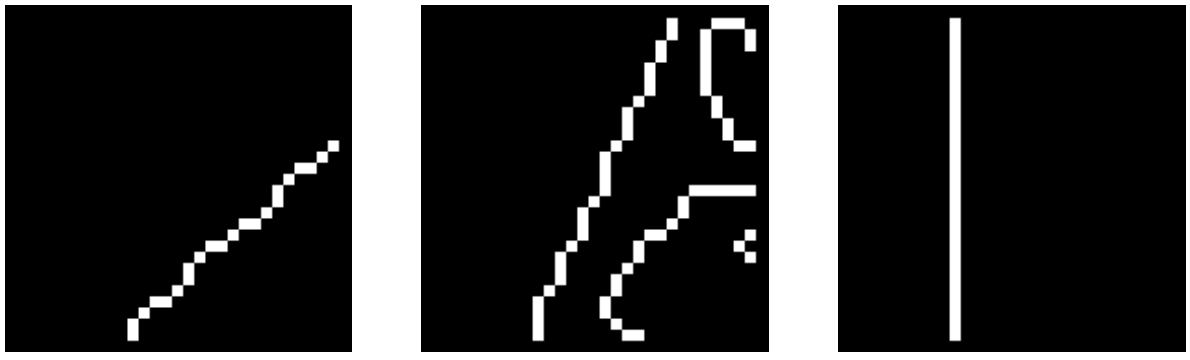
A pesar de todo lo procesado hasta el momento, todavía quedaría información por extraer de la imagen de color, en este caso, el grado de rotación de la pieza. Partiendo una vez más de la imagen de color original, se recorta una porción de la imagen, lo necesario para que contenga un único borde de la pieza. Se pasa a escala de grises, ya que la información sobre el color es, una vez más, irrelevante y dificulta el procesamiento de la imagen.



(a) **(b)** **(c)**

Figura 4.15. Recorte previo a analizar la orientación de la imagen

Esta operación terminaba derivando en errores producidos al interpretar los tetones de la matriz base como líneas de la imagen. En estas circunstancias se decidió operar con la imagen binaria devuelta por la máscara de color, que como se observa en la Figura 4.6 solo contiene las formas de las piezas sin otros elementos que pudieran provocar una equivocación. Se aplica el detector de bordes de Sobel, ya que en esta ocasión no se necesita tanta precisión como en el caso anterior y se obtiene una imagen como la de la Figura 4.16.



(a) **(b)** **(c)**

Figura 4.16. Bordes tras aplicar el algoritmo de Sobel

Para saber el ángulo de giro, se aplicará la transformada de Hough, explicada en la Sección 2.2.8.2. MATLAB devuelve el valor del ángulo θ, cuyo significado se muestra en la Figura 4.17 Puesto que la imagen no es perfecta y tiene ruido, el algoritmo detectará varias lineas distintas, que tendrán ángulos similares (si bien no iguales). Estas lineas diferentes y los resultados devueltos se pueden apreciar en la Figura 4.18.

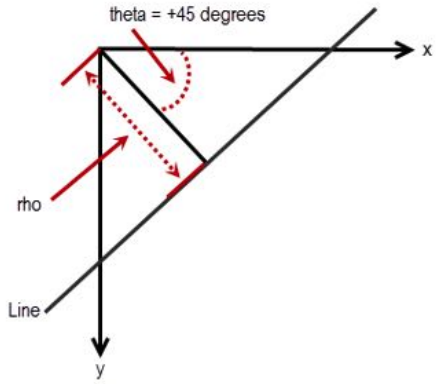
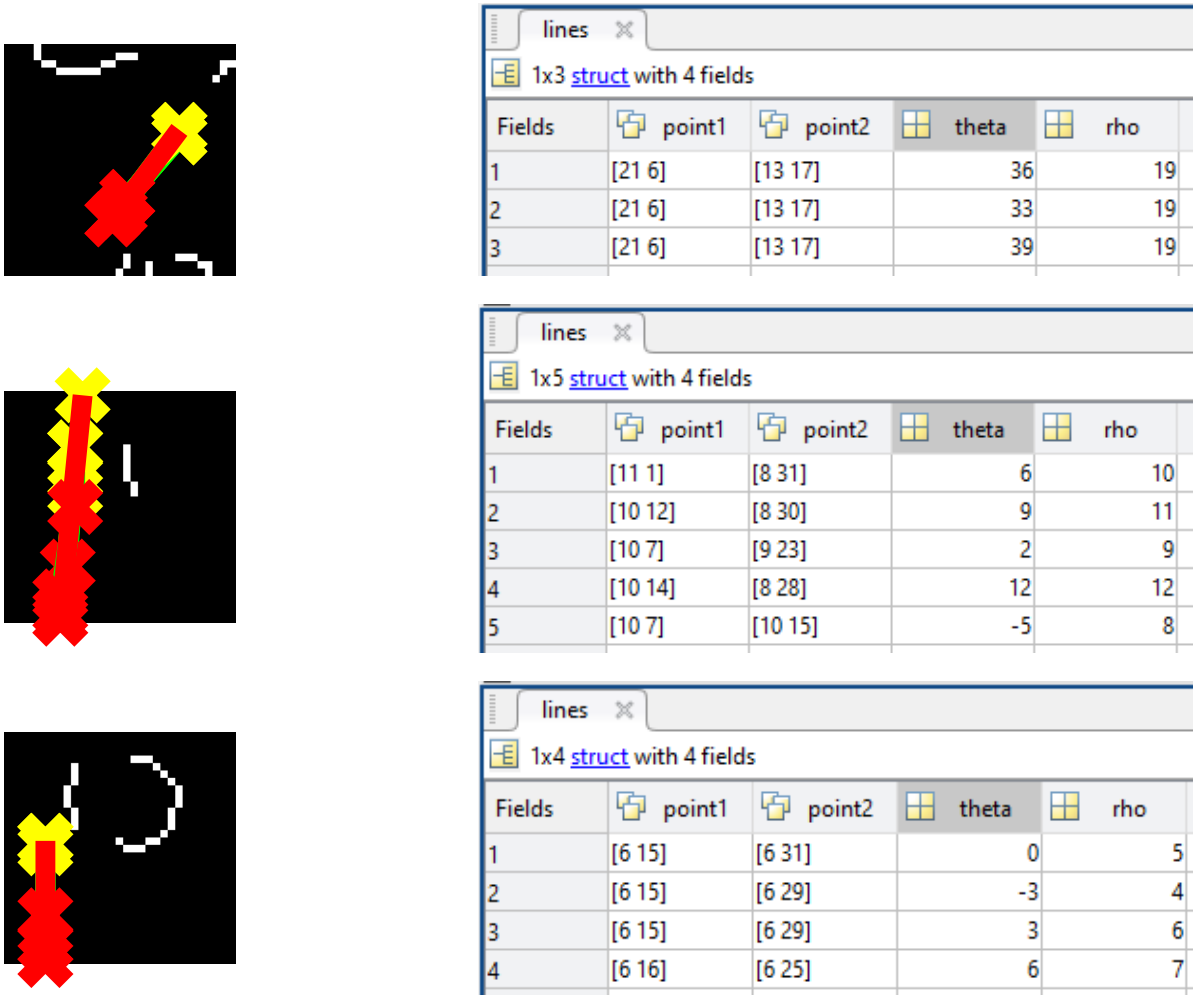


Figura 4.17. Significado de la información obtenida mediante el uso de la transformada de Hough. Fuente: galería de imágenes de MATLAB.



lines

1x3 struct with 4 fields

Fields	point1	point2	theta	rho
1	[21 6]	[13 17]	36	19
2	[21 6]	[13 17]	33	19
3	[21 6]	[13 17]	39	19

lines

1x5 struct with 4 fields

Fields	point1	point2	theta	rho
1	[11 1]	[8 31]	6	10
2	[10 12]	[8 30]	9	11
3	[10 7]	[9 23]	2	9
4	[10 14]	[8 28]	12	12
5	[10 7]	[10 15]	-5	8

lines

1x4 struct with 4 fields

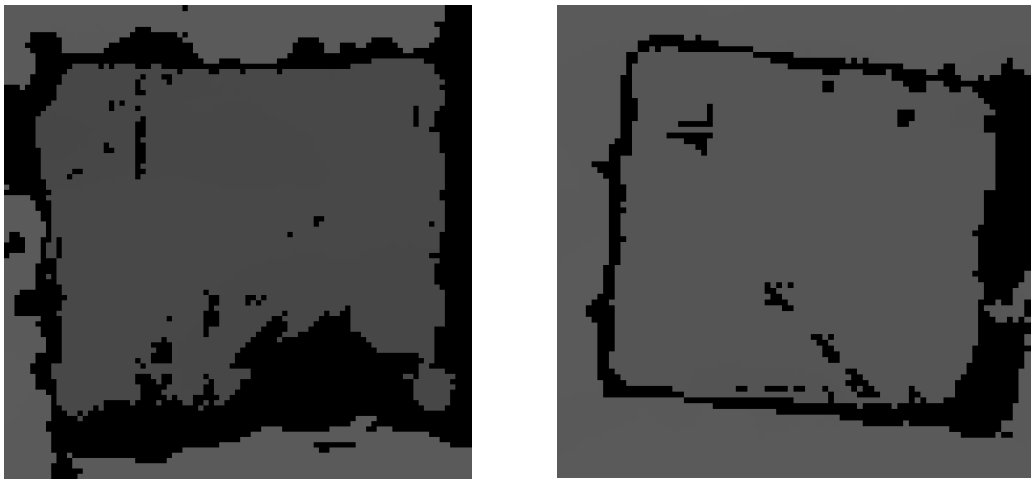
Fields	point1	point2	theta	rho
1	[6 15]	[6 31]	0	5
2	[6 15]	[6 29]	-3	4
3	[6 15]	[6 29]	3	6
4	[6 16]	[6 25]	6	7

Figura 4.18. Resultado de aplicar la transformada de Hough al borde encontrado en Figura 4.16

En esta posición se tomará la mediana de esos valores como valor definitivo ya que, la mediana es una medida más robusta a fallos que otros estadísticos como puede ser el caso de la media.

4.3. Procesado de la imagen de profundidad

Tras obtener toda la información pertinente de la imagen de color, debemos averiguar a qué altura se encuentra la pieza, es decir, cuantas piezas tiene debajo. Sin embargo, y debido al modo de funcionamiento del robot que se explicará más adelante (en el Capítulo 6) lo que nos interesa obtener no es la altura de la pieza en sí, sino la distancia que hay desde el bloque hasta el robot. El primer paso es recortar la parte de la imagen en la que se encuentra la pieza aprovechando la información de la imagen de color como se muestra en la Figura 4.19 y desde ahí, se han barajado varias opciones.



(a) **(b)**

Figura 4.19. (a) Recorte de la imagen de profundidad de una pieza de altura 3. (b) Recorte de la imagen de profundidad de la imagen rotada.

La primera opción considerada fue aplicar una máscara azul a la imagen devuelta por el colorizer, no obstante, esto resultó totalmente infructuoso ya que la máscara devuelve por principio una imagen binaria. Se podía observar donde estaban las piezas, pero la diferencia entre los azules no era significativa como para aplicar una máscara por cada altura y al devolverse una imagen binaria, no se podían realizar operaciones con ella.

Dado que al tener información RGB, las operaciones no se realizan con la misma facilidad que con una única matriz y visto que el color no aportaba ningún dato adicional, se llevó a cabo una modificación en la adquisición de imágenes de profundidad como se indicó en la Sección 4.1. Afortunadamente, trabajando así la información devuelta por la imagen es la distancia a cada punto, por lo que solamente se deben tomar los puntos de la imagen y a partir de ahí calcular la altura.

En este momento se presentaron dos alternativas: trabajar con la media de los puntos de la imagen recortada o con la mediana. Debido a que algunos puntos contienen información nula: valores desproporcionadamente altos o bajos, se optó por emplear el valor de la mediana, que es más robusto frente a este tipo de variaciones.

La cámara de profundidad es muy sensible a cambios de iluminación y las piezas, debido a su textura pulida y a sus brillantes colores reflejan con mucha facilidad la luz artificial de la sala devolviendo, en estos casos concretos, valores nulos. Antes de dar por buena la medida tomada, el sistema validará que la altura calculada se encuentra dentro del rango posible. De no ser así,

asumirá que la imagen anterior era defectuosa y tomará una nueva imagen y repetirá todo el proceso.

No obstante, este sistema continúa presentando pequeñas variaciones de apenas dos o tres milímetros en función de las condiciones de luz y la posición. Estas diferencias son suficientes para provocar que el robot impacte con la pieza[3] o no llegue a aferrarla de manera contundente, lo que provoca que la pieza se escurra.
Para solventar estos inconvenientes, se decidió fijar unos rangos de altura, si la pieza se encuentra entre ellos, se asumirá que hay un número determinado de piezas apiladas y se introducirá el valor de la coordenada que debe tomar el robot. Estos valores se detallan en la Tabla 4.1. En esta tabla se detallan los datos necesarios para tomar una pieza situada en la parte superior de una hilera de hasta 5 alturas aunque, como se explicará en el Capítulo 5, finalmente la altura máxima se limitará a tres piezas apiladas.

Número de piezas apiladas	Rango de valores válidos	Coordenada Z del robot
Cinco piezas	Menor de 275	85
Cuatro piezas	Entre 275 y 295	66
Tres piezas	Entre 295 y 315	47
Dos piezas	Entre 315 y 335	28
Una pieza sin rotar	Valor superior a 335. Rotación entre $\pm 5^o$	10
Una pieza rotada	Valor superior a 335. Rotación fuera del intervalo anterior	12

Tabla 4.1. Tabla de las alturas fijas en función de los valores devueltos por la imagen de profundidad para distinto número de piezas apiladas. En el caso de que haya una única altura, se tendrá en cuenta si está encajada en la matriz de puntos.

[3]Esto genera un error 10025 de colisión, lo que provoca la detención inmediata del robot y obligaría a reiniciar todo el sistema.

5

Sistema de coordenadas

El quinto capítulo engloba todo lo relativo al cambio de coordenadas, explica el motivo por el que se debe realizar, cómo se ha hecho y los problemas que surgen en los distintos casos.

Como se comentó en la Sección 4.2, uno de los datos calculados es la posición de la pieza en el sistema de coordenadas de la foto. Sin embargo, esta información no es suficiente para lograr el objetivo final: colocar las piezas en el lugar designado para cada color. De cara a colocarlas, el robot debe inicialmente lograr cogerlas, y para ello tiene que saber exactamente en qué punto están en la realidad.

El primer paso fue comparar los sistemas de coordenadas, el de la imagen y el del robot. La diferencia fundamental es la colocación de los ejes: el eje de abscisas de la imagen se corresponde con el de ordenadas del robot y viceversa, aunque ambos coinciden en la dirección de crecimiento de los ejes, tal y como se indica en la Figura 5.1.

A continuación, se intentó establecer una relación lineal entre las coordenadas de la imagen y las del robot[1], pero debido a las imprecisiones en los datos obtenidos, tanto en un caso como en el otro, no se pudo fijar una correspondencia directa.

Se distinguen dos casos claros: aquellos en los que la pieza se encuentra encajada en la matriz y cuando la pieza está girada, lo que implica que se encuentra sobre los tetones. En ambos casos se empleará un algoritmo de funcionamiento distinto, siendo el elemento discriminatorio el ángulo de giro de la pieza. En base a este elemento, si el ángulo obtenido se encuentra próximo a cero, se asumirá que la pieza está encajada y en caso contrario, se calcularán las coordenadas en función del valor de este ángulo.

En ambas situaciones se obtendrán primero las coordenadas absolutas de la pieza en términos del sistema del robot y a continuación los offsets, que, como se explicará en el Capítulo 6, serán los datos enviados al robot para su correcto funcionamiento.

[1] A partir de este momento, cada vez que se haga referencia a ambos sistemas de coordenadas, se habrán invertido las coordenadas obtenidas a partir de la imagen, de manera que coincidan ambos ejes

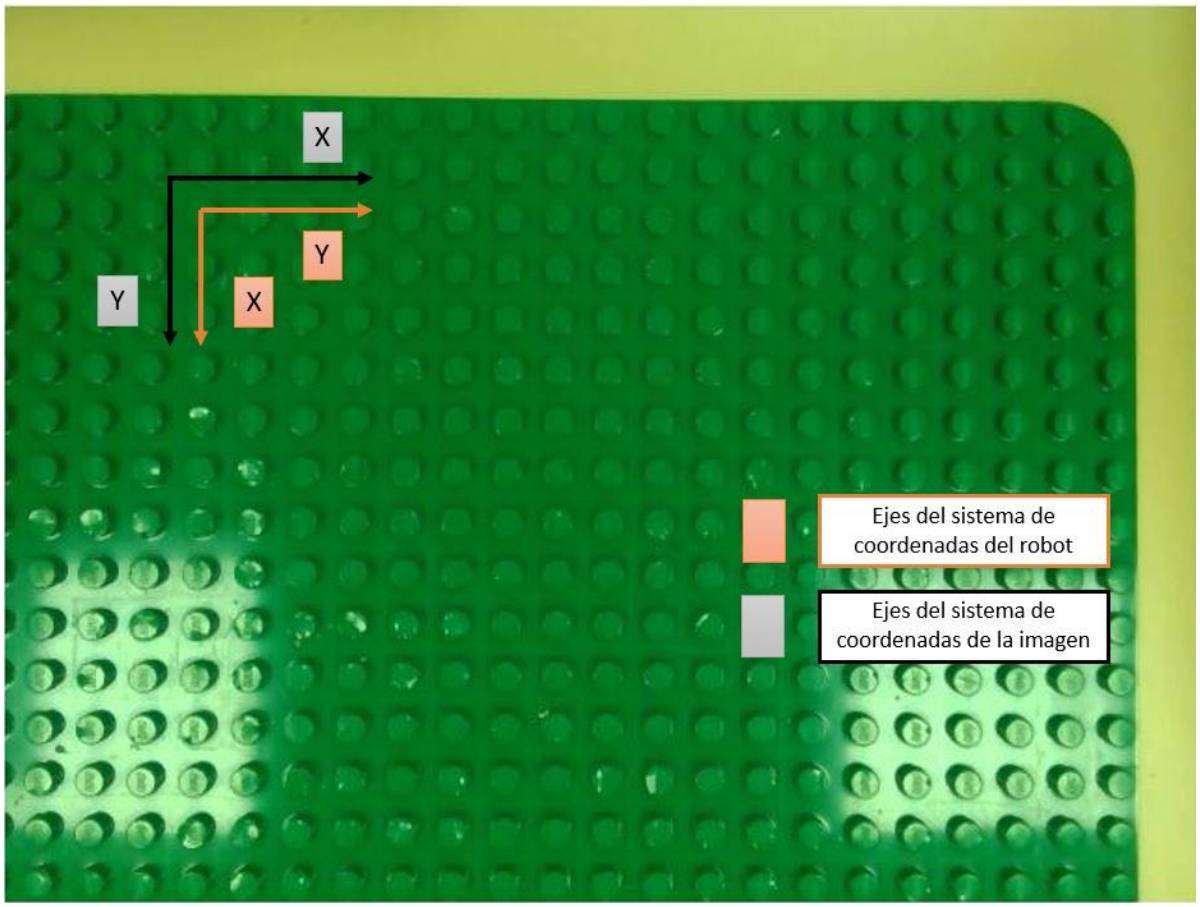


Figura 5.1. Ejes de coordenadas en ambos sistemas. En naranja el sistema del robot. En negro el de la imagen.

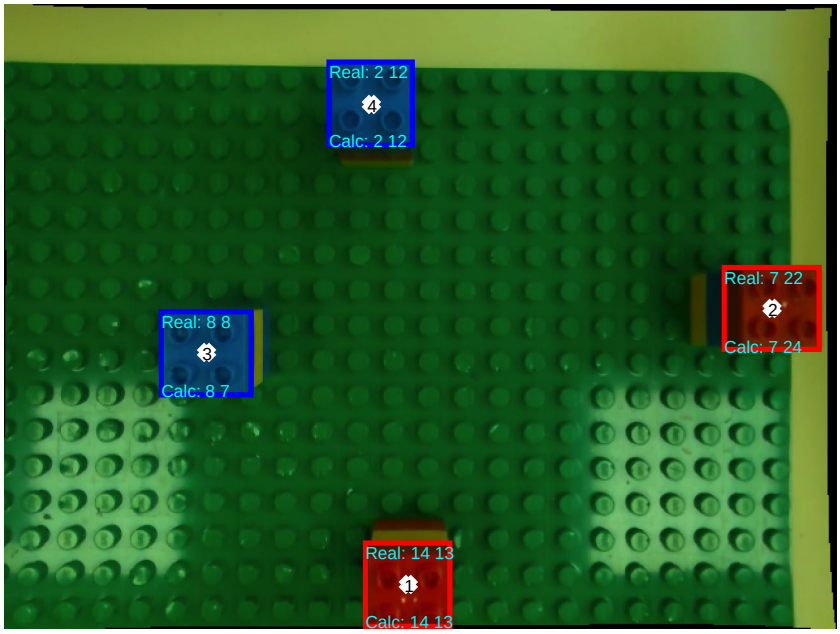
5.1. Pieza encajada

Al suponer que la pieza está encajada en la matriz de 24x24 puntos[2], se asumirá que la pieza tiene que adoptar una de las 529 posiciones posibles, aunque realmente son menos dado que el robot no accede a todas las ubicaciones de la matriz.

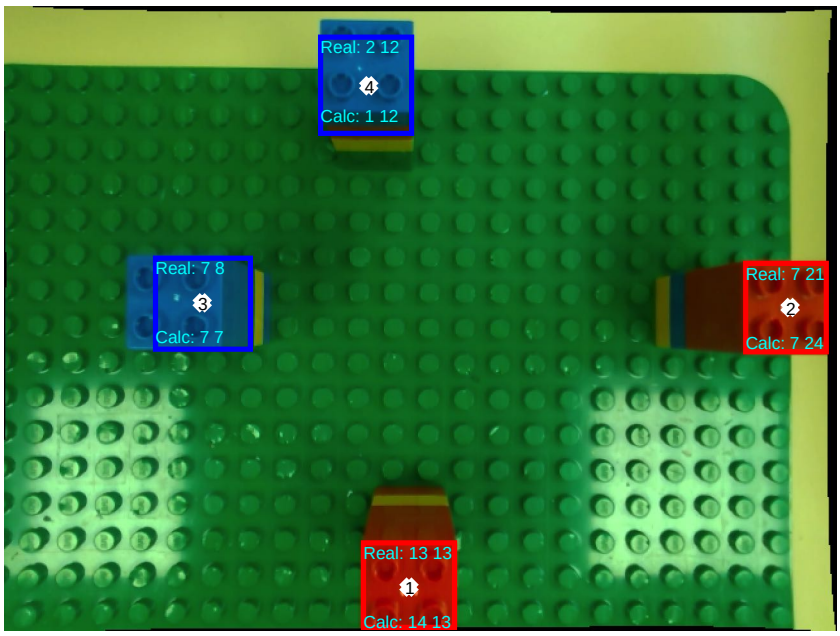
Se calculará la posición en base al número de hueco que ocupa en ambos ejes y a partir de ahí, y teniendo en cuenta que cada posición son 16 mm se calculará la distancia en milímetros desde el punto de inicio del tapiz hasta el punto central de la pieza. A esta información se le sumará la coordenada del robot en el punto inicial del tapiz en ambos ejes y con ello se obtendrá la coordenada final de la pieza.

Sin embargo, cuanto más alejada se encuentre la pieza del centro de la imagen, más fácil es que este cálculo falle puesto que, al ver la pieza desde una determinada altura y apoyado por la distorsión radial insertada por la cámara, la pieza se ve desplazada hacia el lado en el que se encuentre. Como se puede ver en la Figura 5.2, las coordenadas indicadas en la parte superior de la pieza hacen referencia a la posición real de la pieza, mientras que las de la parte inferior señalan lo calculado si no se aplica ninguna corrección. Queda patente en la comparativa entre ambas, que el error de cálculo aumenta de manera proporcional a la cantidad de piezas apiladas.

[2]Es decir, de 23x23 huecos, que son las posiciones válidas



(a)



(b)

Figura 5.2. (a) Comparación entre los valores reales y los calculados respecto a la posición de las piezas con 3 piezas apiladas (b) Comparación entre los valores reales y los calculados respecto a la posición de las piezas con 5 piezas apiladas.

Como también se puede apreciar, otro factor de riesgo es que la imagen considere parte de la pieza el lateral de la misma, en cuyo caso la medida también se desvirtúa. Esto se puede observar en la Figura 5.3. Este problema crece si se encuentran varias piezas del mismo color apiladas sucesivamente, aunque es dependiente de las condiciones de luz en la sala a la hora de ejecutar el programa. Debido a estos dos contratiempos, se decidió limitar la altura de la hilera de piezas a un máximo de tres, de manera que la corrección a realizar fuera viable y el error debido al lateral de la pieza sea menor.

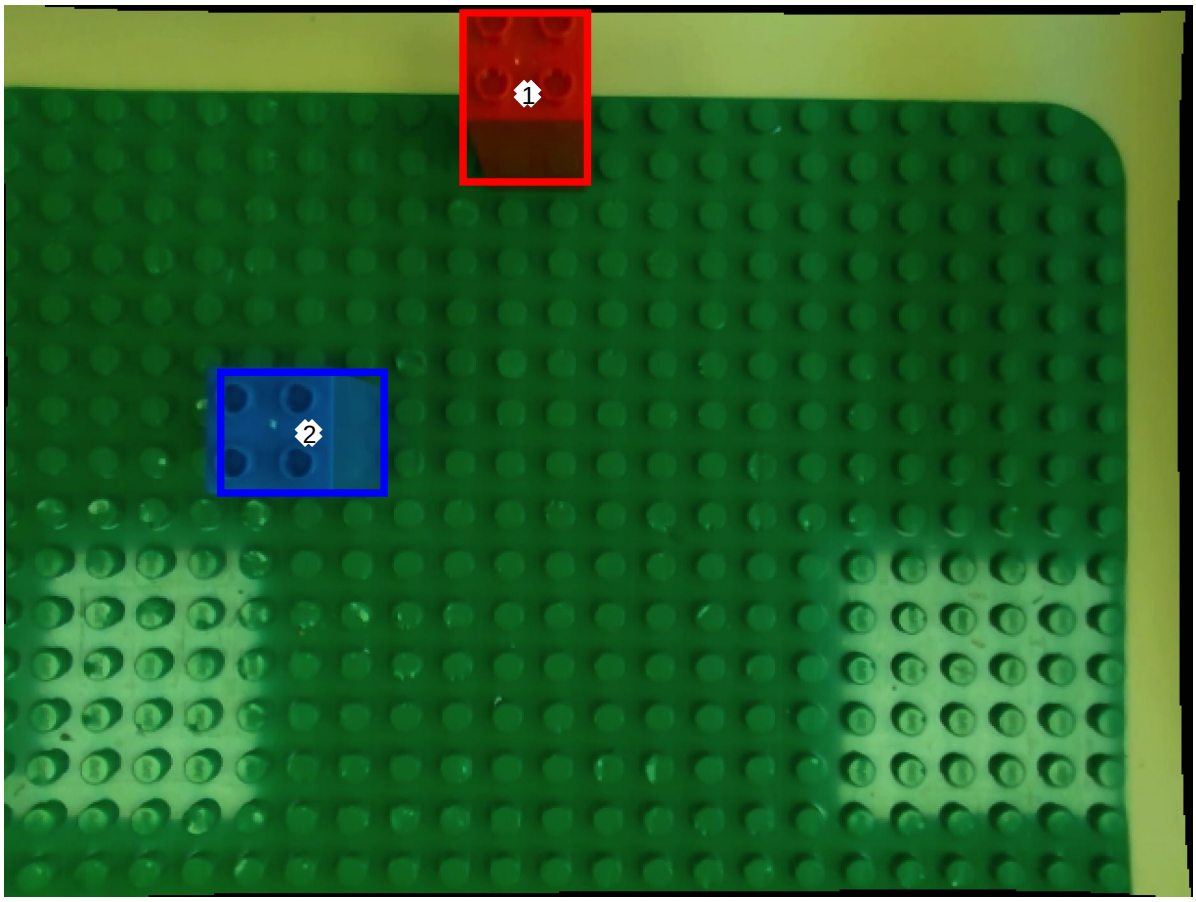


Figura 5.3. Muestra del error ocurrido al apilar varias piezas del mismo color. El sistema considera el lateral de la pieza como si formara parte de la misma, lo que distorsiona el posicionamiento.

5.2. Pieza rotada

En el caso en el que la pieza se encuentra rotada, se decidió tomar un punto de referencia: la esquina superior derecha de la matriz de LEGO, que tiene las coordenadas que se muestran en la figura. Tras muchas pruebas, se comprobó que era mucho más eficiente trabajar con vectores con origen en el punto de referencia y extremo en la posición de la pieza a localizar. Empleando este sistema, se verificó que la relación entre ambos sistemas de coordenadas era del orden de 0.555 en el eje X y 0.585 en el eje Y. En la Figura 5.4 se observa la diferencia en las coordenadas que debe adoptar el robot para coger la pieza si se encuentra girada o fija, suponiendo que se encuentren en la misma posición.

A la hora de calcular las coordenadas finales en las que se debe posicionar el robot, se deben tener en cuenta diversos conceptos geométricos. Estos cambios a realizar en los cálculos se deben a que el manipulador no realiza el giro sobre sí mismo, sino sobre el centro del eje 6 del robot que, simultáneamente, es el que señala las coordenadas del robot. Esto provoca que, si se desea que el efector se encuentre en una posición concreta con un ángulo de giro determinado, realmente las coordenadas absolutas del robot sean diferentes que si se encontrara con la pinza en posición vertical. En la Figura 5.5 se observa cómo, una pieza situada en el mismo lugar, con iguales coordenadas en la imagen, tiene que pasarle al robot valores muy diferentes para lograr que la coja correctamente. En la imagen situada debajo, se puede comprobar cómo se debe posicionar el robot en cada caso concreto.

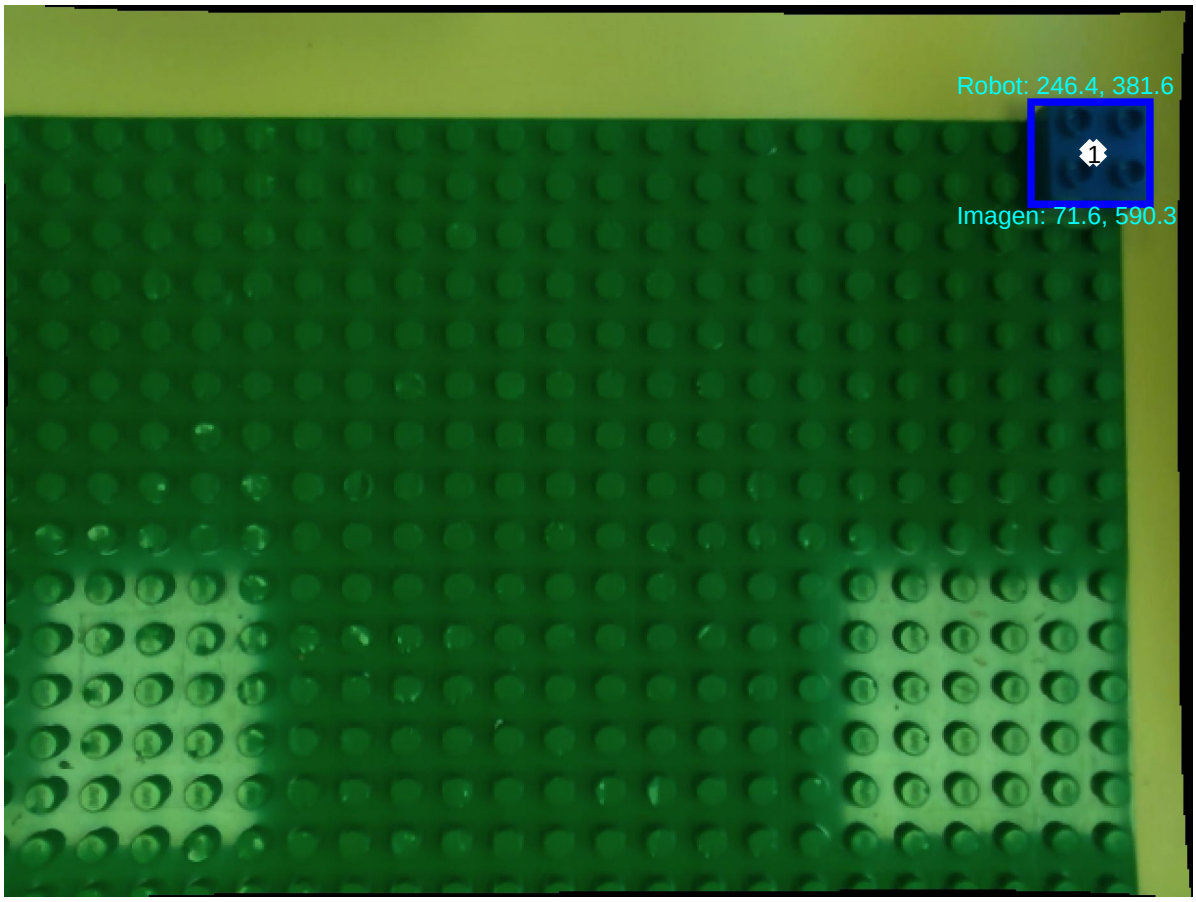
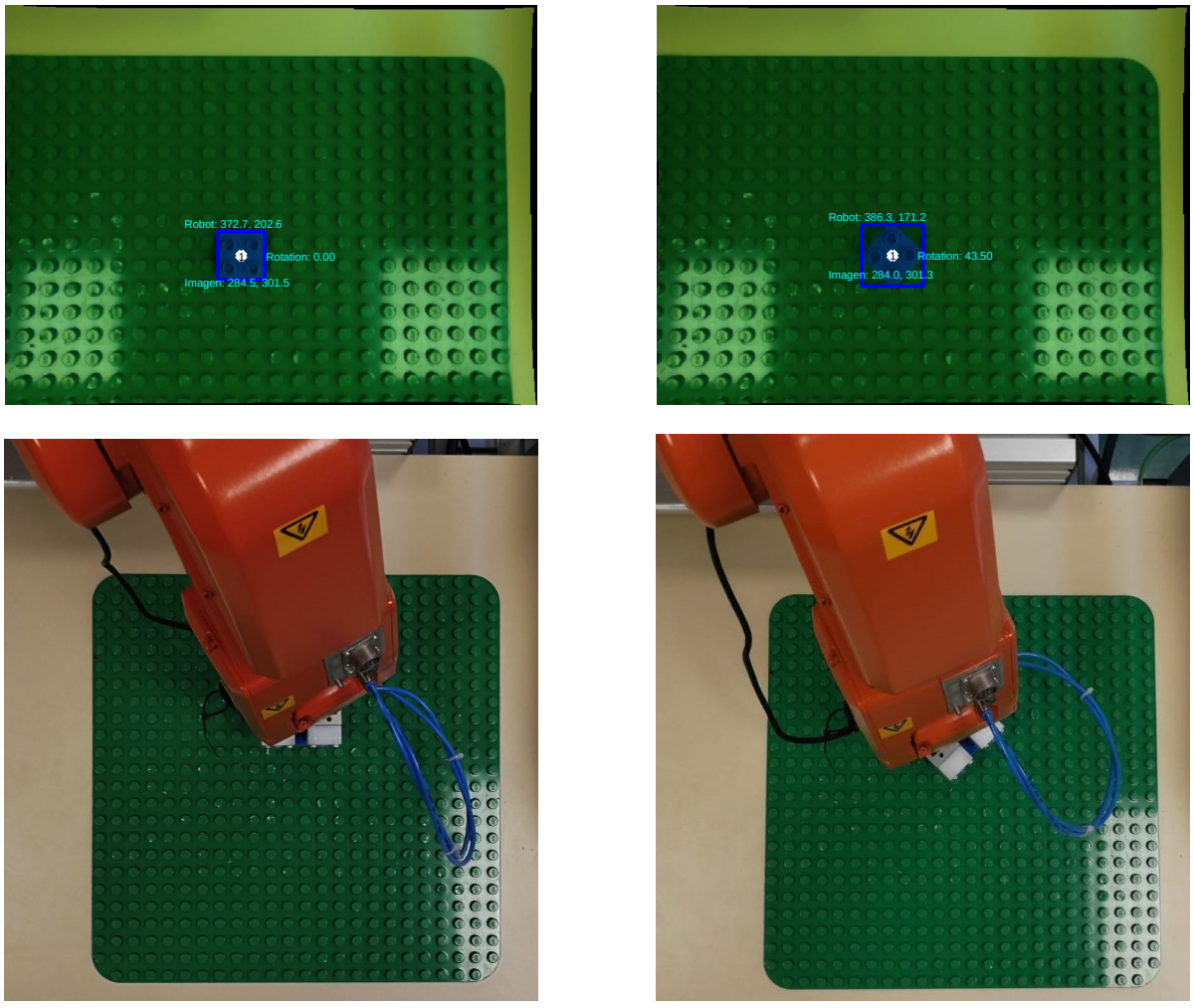


Figura 5.4. Comparativa de las coordenadas de una pieza colocada en la posición de referencia. Coordenadas según el sistema de coordenadas del robot y según lo calculado en la imagen (en este caso invertidas para que coincidan los ejes).

Tomando el valor del radio de giro del efector y el ángulo girado, se aplicarán principios de geometría básica para calcular el desplazamiento en ambos ejes y esta variación se sumará a los valores previamente calculados para obtener las coordenadas finales.

Una vez obtenida la información de posicionamiento en el plano XY, se debe adecuar el dato obtenido a partir de la imagen de profundidad para verificar el correcto movimiento del autómata. Como se indicó en la Sección 4.3, la función de procesamiento de la imagen de profundidad devuelve el valor que debe tomar la coordenada Z del robot para lograr agarrar la pieza correctamente, evitando la posibilidad de colisiones.

Llegados a este punto, se posee la información que identifica la posición de la pieza de manera inequívoca y en coordenadas del robot. No obstante, al autómata se le deben enviar los datos referidos a la diferencia de coordenadas que tiene que llevar a cabo desde la posición en la que se encuentra el robot hasta la que se desea alcanzar. Estos cálculos se realizan en la función de MATLAB Pic2Robot.m, en la que se tiene en cuenta el color de la pieza anterior para saber desde qué punto partirá el robot.



(c) **(d)**

Figura 5.5. Diferencia en la posición de un robot a la hora de coger una pieza colocada en el mismo sitio pero rotada.

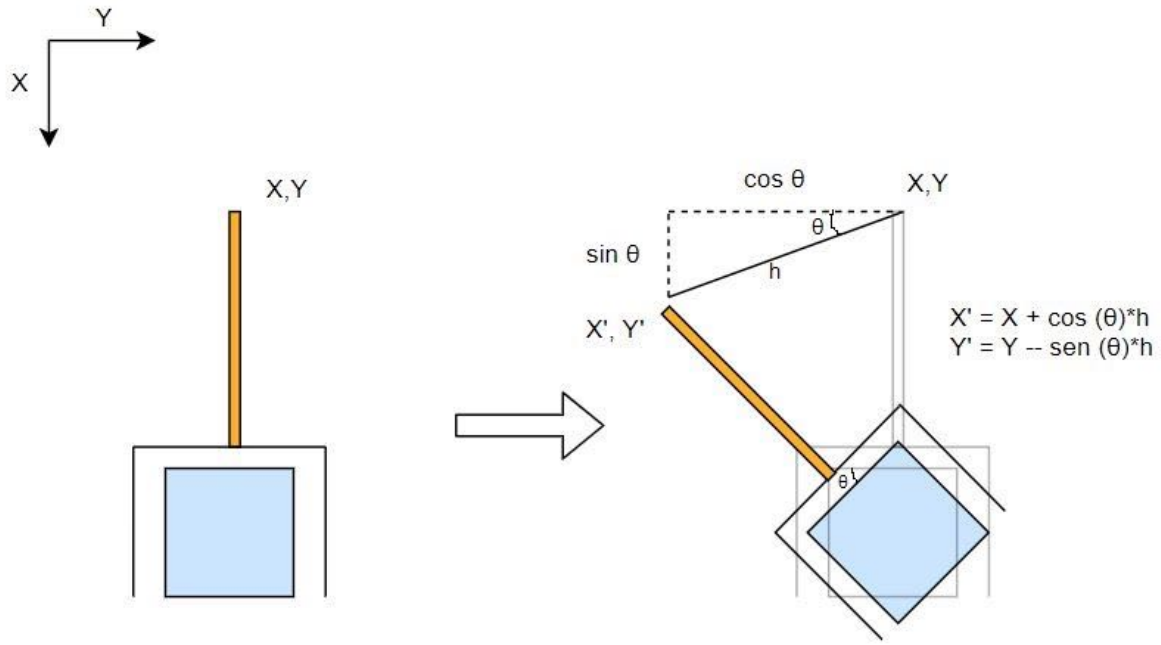


Figura 5.6. Diagrama de los cambios en coordenadas a la hora de coger una pieza rotada.
En la primera parte del diagrama se representa el robot (parte naranja y la "pinza" sujetando una pieza azul). En la segunda, se superpone al dibujo inicial la posición que debe adoptar para agarrar una pieza rotada un ángulo θ que se encontrara en la misma posición. La transformación realizada a las coordenadas iniciales (X e Y) se muestra en el lateral del diagrama, para obtener las finales (X' e Y')

6

Programación del robot

El sexto capítulo engloba todo lo relativo a la programación del robot, la esquematización del programa empleado y algunas de las funciones principales. También se explica el tipo de datos necesario y se justifica la utilización de determinadas funciones frente a otras.

La segunda pieza elemental de este sistema es el robot, que será el encargado de recoger las piezas de LEGO de su posición inicial y colocarlas en su colocación correspondiente en función de su color. Como se indicó en la Sección 1.3, el robot empleado es de la marca ABB, modelo IRB120 y tiene seis grados de libertad, mostrados en la Figura 6.1.

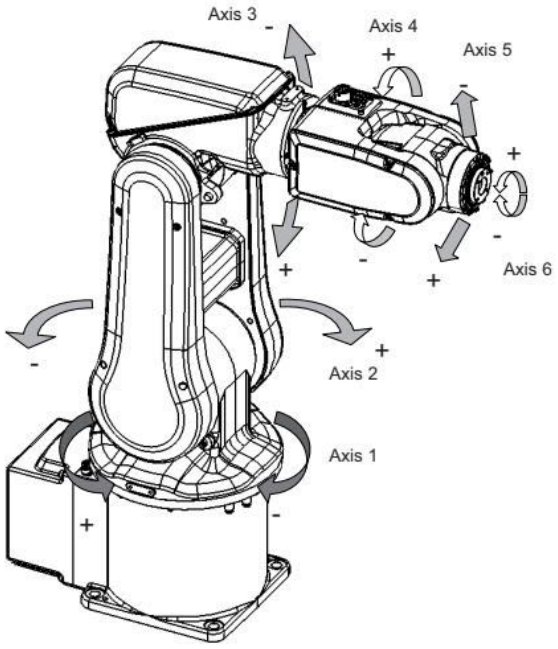


Figura 6.1. Ejes de giro del robot ABB IRB120

6.1. Comunicación ordenador-robot

El ordenador debe enviar todos los datos relativos a las piezas al robot para que este sea capaz de ejecutar su cometido. La comunicación entre ellos se establecerá mediante un socket TCP/IP (Transport Control Protocol/Internet Protocol)[1], en el que el robot hará las veces de servidor y el ordenador de cliente. Por ello, se debe ejecutar en primer lugar el programa del robot, para poder ponerlo a la escucha en el puerto designado (el 1024) y a continuación el programa de MATLAB, que solicitará conexión.

Esta conexión se realiza a través de una red interna cableada (192.168.56.0/24) que conecta los elementos del laboratorio y al menos una tarjeta de red de cada ordenador. Por ello, para que el sistema funcione, se deberá trabajar desde un ordenador del laboratorio. Esto también resulta una ventaja a la hora de trabajar con el FlexPendant[2] ya que facilita el traspaso de archivos desde el ordenador al mismo. De esta manera, se pueden redactar los programas en el ordenador y pasarlos allí automáticamente y viceversa.

A la hora de enviar información, MATLAB pasará una cadena de caracteres que será recibida en el *buffer* del robot y que recogerá y analizará cuando se le dé la orden. MATLAB deberá tener un temporizador que retarde el envío de dicha información, de manera que se evite que se sobrescriba y, por ende, se pierda.

6.2. Recogida de la información

La información recogida por el robot es un *string* que contiene los siguientes datos:

El programa toma los datos, los separa y los convierte a su formato correspondiente, en este caso, todos a formato *num* y los almacenará en variables independientes. Una vez haya procesado y utilizado toda esta información, solicitará datos nuevos al Socket donde, en caso de que haya más piezas por ordenar, MATLAB habrá depositado la información.

6.3. Movimiento del robot

Para programar el robot, se ha empleado RAPID, un lenguaje de alto nivel diseñado por ABB para controlar sus robots industriales. Debido a las características del lenguaje que se explicarán en la Sección 6.3.1, no se puede trabajar directamente con las coordenadas de la pieza en el sistema del robot que como se indicó en Capítulo 5 tampoco coinciden con las identificadas en la imagen.

6.3.1. Selección de posiciones

Las posiciones del robot, se deben definir en formato *robtarget* que se compone de cuatro tipos de valores diferentes:

- Trans: dato del tipo *pos* contiene las coordenadas X, Y, Z de la herramienta expresadas en milímetros. Esta posición se señala en función del sistema de coordenadas especificado, por defecto, el sistema de coordenadas del propio robot con centro en la base del mismo.

[1] Estos protocolos se definieron inicialmente en [Pos81b] y en [Pos81a] respectivamente. A partir de ahí han sufrido actualizaciones a las que se puede acceder desde dichas RFCs (Request For Comments).

[2] El FlexPendant es una pantalla asociada al robot desde la cual se pueden realizar múltiples funciones, entre ellas, manejar manualmente el robot o ver y depurar los programas en ejecución.

Nombre de la variable	Tipo de dato	Misión del dato
Color	*num*	Señala el color de la pieza a recoger: 1 para rojo, 2 amarillo y 3 azul. En función del código, el programa sabrá en qué posición debe depositar la pieza.
Coord_x	*num*	Distancia en el eje X en mm desde la coordenada actual del robot hasta el punto de destino (generalmente el punto donde se encuentra la siguiente pieza a recoger).
Coord_y	*num*	Distancia en el eje Y en mm desde la coordenada actual del robot hasta el punto de destino (generalmente el punto donde se encuentra la siguiente pieza a recoger).
height_get	*num*	Distancia en el eje Z en mm desde la coordenada actual del robot hasta la altura a la que se encuentre la pieza a recoger.
orientation	*num*	Posición de giro en grados del eje 6 necesaria para recoger la pieza correctamente.
height_release	*num*	Modificación que se le debe realizar a la posición donde se debe depositar la pieza en el eje Z. Varía en función del número de piezas anteriormente depositadas.
ID	*num*	Señala si la pieza indicada es la última de esta tanda y debe regresar a la posición de la fotografía para volver a comenzar el proceso o verificar la finalización del mismo.
point_extraction	*num*	Señala si la pieza se encuentra girada o no tiene piezas apiladas debajo de ella. En estos casos, realizará una extracción simple (casos en los que toma el valor de 1). En caso contrario, deberá aplicar la extracción por puntos para evitar levantar las piezas colocadas bajo ella.

Tabla 6.1. Variables enviadas a través del Socket desde MATLAB hasta el robot, tipo de dato y significado del mismo en el programa

- Rot: dato del tipo *orient* especificado en función del sistema de coordenadas, señala la orientación del objeto de trabajo. Este dato se expresa en la forma de cuaterniones.
- Robconf: dato del tipo *confdata* se compone de cuatro datos que indican la posición de los ejes 1, 4 y 6 y agrega un último dato que varía en función del tipo de robot.
- Extax: este dato hace referencia a los ejes externos que, debido a que en este caso concreto no están conectados, se fijaran los seis valores a 9E+09.

Debido a la gran dificultad que conlleva el cálculo de toda la información contenida en este tipo de dato, la única manera factible de generarlos es acudir al robot, colocarlo manualmente en la posición deseada y almacenar los valores que devuelve el Flexpendant para cada campo. Este es el proceso que se ha llevado a cabo para determinar la posición desde la que el robot ha de tomar las imágenes y los puntos en los que se deben depositar las piezas de cada color.

A la hora de elegir la posición para hacer la foto, es especialmente importante para la adquisición de medidas correctas que la cámara se encuentre paralela a la superficie de trabajo.

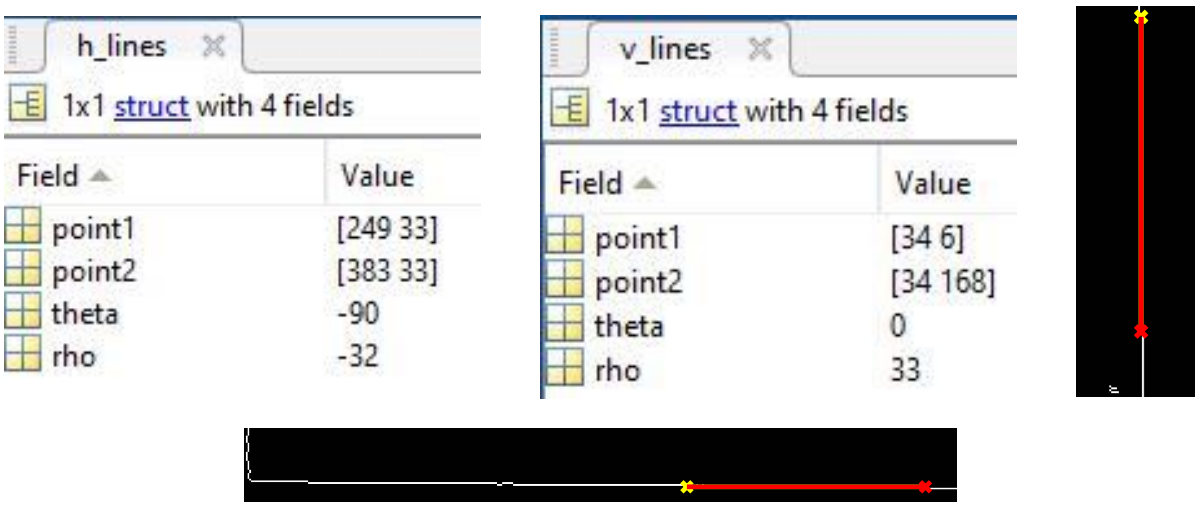


Figura 6.2. Verificación de la posición para tomar la foto. Análisis del extremo vertical derecho y del extremo inferior con los valores obtenidos al aplicar el algoritmo Sobel y la transfromada de Hough al recorte de la imagen

De esta forma, la medida de profundidad será igual para la misma altura independientemente del punto de la imagen en el que se encuentre. En cuanto a las coordenadas XY, es fundamental que la cámara se encuentre alineada con la matriz de puntos, de manera que los ejes de la foto y del robot coincidan en posición. Para verificar todas estas condiciones, se diseñó un programa en MATLAB que, mediante el uso de la transformada de Hough, explicada en la Sección 2.2.8.2 comprobara el ángulo de los bordes de la matriz de trabajo con respecto al punto de origen.

Debido a la complejidad de los datos de posición, explicado anteriormente, es preciso trabajar con distancias relativas entre la ubicación del robot en cada momento y la posición a alcanzar. Esto se traduce en la necesidad de llevar un registro constante de la localización del robot en todo momento. Por ello, todos los datos relativos a las coordenadas se calculan en MATLAB, que tiene mayor capacidad de procesamiento de datos, y permite calcular todos los valores a la vez (no como en el caso de RAPID, donde la información de las piezas se envía una a una).

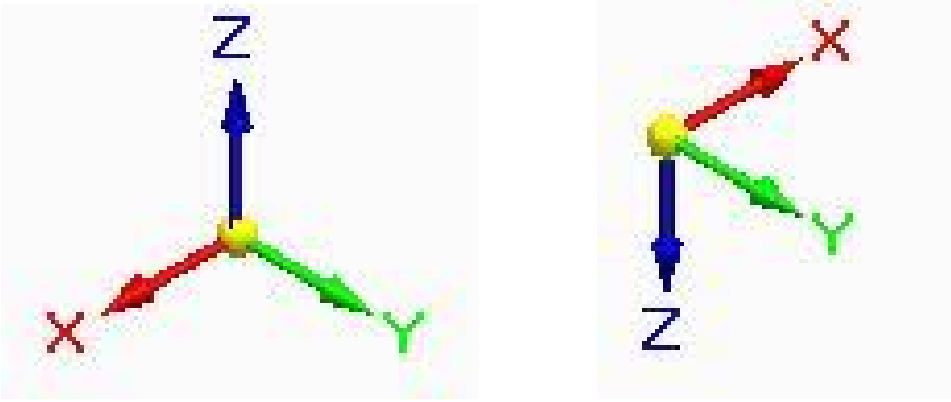
Al calcularlos simultáneamente, se puede almacenar la información relativa a la pieza anterior, lo que permite conocer la posición de la que partirá el robot sin necesidad de fijar un "punto de partida" al que deba retornar en cada iteración. Para provocar un desplazamiento, a la hora de recoger una pieza encuadrada dentro de la matriz de puntos, se emplea la instrucción *Offs*, a la que se le deben pasar como argumentos la posición de partida y las diferencias en las coordenadas X, Y, Z, lo que en el este programa se ha denominado *coord_x, coord_y, height_get*.

En cuanto a las piezas rotadas, el procedimiento es distinto. Las coordenadas propias de este tipo de bloques ya se obtienen calculadas del MATLAB, pero para poder emplearlas, se debe poder rotar el eje 6 del robot los grados que sea necesario.
El primer intento consistió en rotar de manera independiente este eje, para lo cual era preciso cambiar un parámetro del sistema. Por diversos motivos, no se puede acceder a dicho parámetro, con lo que hubo que buscar una solución alternativa.

Esta solución consistió en la utilización de una función distinta: *relTool*, cuyo funcionamiento es muy parecido al de *Offs*. Se pasan como argumentos la posición que se debe modificar, los

parámetros *coord_x, coord_y, height_get* y, en este caso, se puede añadir un argumento opcional que señale el ángulo de giro del sistema completo respecto del eje deseado. Por ello solo habría que incluir en la función el valor de giro, definido por la variable *orientation*.

El principal problema que aporta esta instrucción, es la modificación del sistema de coordenadas: los ejes X y Z se encuentran invertidos, mientras que el eje Y se mantiene, como se muestra en la Figura 6.3. Sin embargo, en el eje X el centro se encuentra desplazado unos milímetros. Esto conllevó que el desplazamiento absoluto se realizara con la instrucción *Offs* y desde ahí tomar la posición del robot empleando *CRobT* y comenzar a utilizar ya ya mencionada instrucción*RelTool*



(a) **(b)**

Figura 6.3. Diferencia en los posicionamientos de los ejes para las diversas instrucciones: (a) Instrucción *Offs* y (b) en la instrucción *RelTool*

Respecto al tema de la extracción, se presenta otro problema a la hora de coger las piezas apiladas: la fuerza ejercida entre los bloques es mayor que la realizada por la matriz sobre la pieza directamente enganchada a ella. Por ello, es muy probable que a la hora de intentar levantar la pieza superior se vayan con ella todas las piezas de esa torre, luego fue preciso seleccionar manualmente una serie de puntos que permitieran una extracción suave de la pieza superior. Tras tomar esos puntos, se buscó la diferencia de posiciones entre ellos y se insertó manualmente en el programa, solo para comprobar que eso no funcionaba correctamente.

Al indicarle los movimientos a realizar, por defecto el robot tiende a adoptar esa posición empleando el menor número de movimientos posibles. Esto produce que los ejes no giren tanto como es preciso para realizar el movimiento de extracción y por ello fue preciso considerar otras alternativas.

Entre las alternativas se barajó emplear la extracción *RelTool* buscando un desenganche sutil de la pieza. Por la posición en la que se coloca la pinza, se deberá realizar una rotación del robot sobre el eje X, de forma que sea imposible que se escurra la pieza una vez agarrada. Esto produce un error en el robot puesto que, se le solicita a los ejes del robot un movimiento que no pueden realizar debido a la oposición de la pieza, que se encuentra enganchada. Esto deriva en un fallo general del robot que detiene el sistema, por lo que la rotación final se realiza sobre el eje Y.

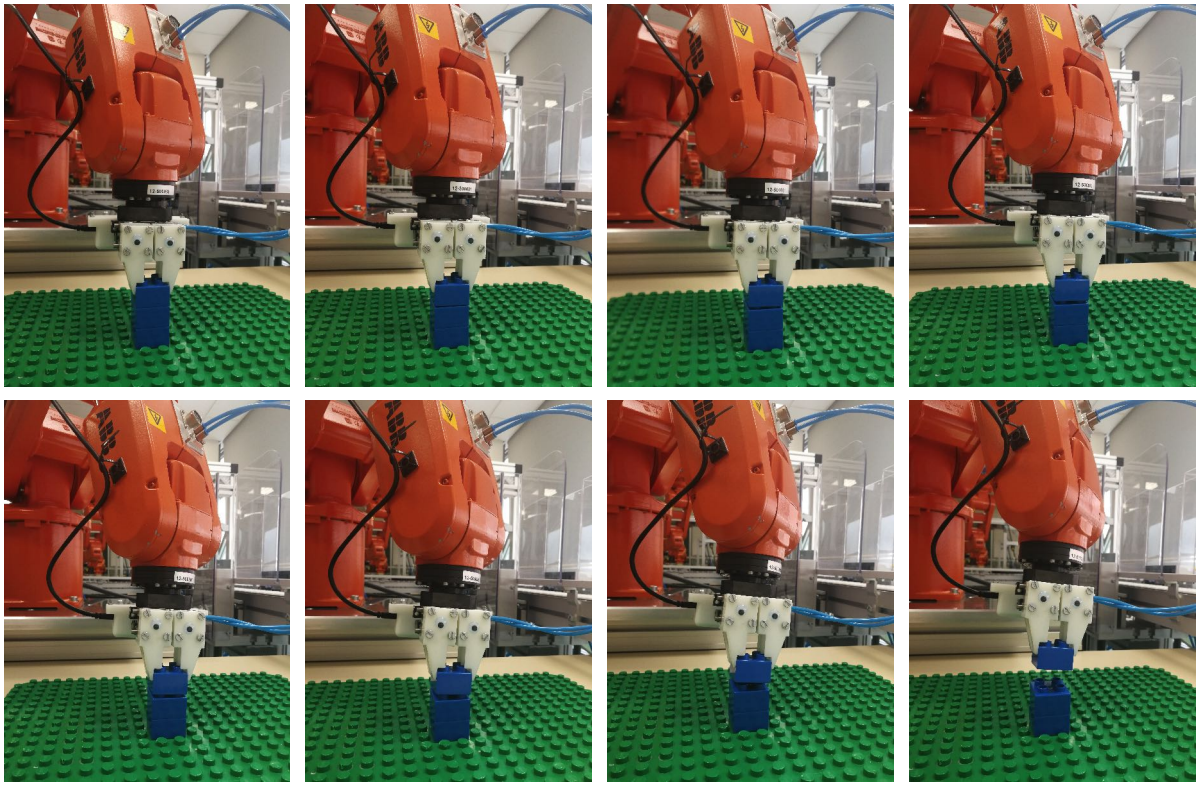


Figura 6.4. Fases de la extracción por puntos

6.3.2. Estructura del programa

El programa ejecutado por el robot se compone de varios procesos englobados en un único módulo. Entre estos procesos, se encuentra uno encargado de administrar la información recibida, otro para establecer la comunicación con el ordenador y los encargados de abrir y cerrar la pinza del efector. Al comienzo del programa se deben definir todas las variables globales: las que hacen referencia a la conexión, a la recepción de datos y, por supuesto, a las posiciones predeterminadas.

Al comenzar la ejecución, se envía al robot a la posición desde la que debe tomar la foto, para que en el momento de ejecutar el programa del cliente no haya fallos en la captura de imágenes (y por ende en la obtención de información). Se establece la conexión con el cliente y se espera hasta que se reciban los datos.

Una vez recibidos, se llama a la función *GetData*, que se encarga de separar el *string* recibido en los valores indicados en la Tabla 6.1 transformándolos, desde el formato *String* en el que son recibidos a formato *Num*. Es cierto, que los datos: *ID* y *point_extraction* funcionarían mejor como *boolean*, ya que solo pueden tomar dos valores (cero o uno en ambos casos) y se emplean para decidir la acción que se debe llevar a cabo.
Con todo, en este lenguaje no existe una instrucción que permita transformar los datos de *string* a *boolean*, por lo que se optó por emplear valores numéricos.

Tras adecuar los datos para su uso, se desplaza el robot hasta la posicionarse sobre la pieza que se va a extraer. Una vez colocado, abrirá el efector, descenderá lentamente hasta la pieza y cerrará la pinza. A continuación se verifica el valor de *point_extraction*, que señala si la pieza está apilada, con lo que se debería emplear la "extracción por puntos" o si está girada o es la última, y se debe emplear la instrucción *RelTool*.

Una vez se ha procedido a la extracción de la pieza, el robot ascenderá con ella hasta una “altura de seguridad”, que se debe tomar para asegurar que el efector no impactará contra ninguna posible torre de piezas que se encuentre en el camino. Por defecto, el robot se desplazará a una altura a 12 cm sobre la superficie de trabajo, 2 cm por encima de la altura máxima.

Tras extraer la pieza se comprueba su color y se determina la posición a la que se debe enviar, almacenando esta posición en una variable. Se envía el robot a la posición en la que se debe depositar, pero aún a una altura superior desde la que descenderá. Por último, abrirá la pinza, regresa a la altura de seguridad y cerrará la pinza.

Finalmente, evaluará si la pieza recogida es la ultima que debe ordenar de esa imagen empleando la variable *ID*. Si no fuera la última, el programa procedería desde ese punto empleando los datos que lleguen mediante el *socket* y si lo fuera, volvería a la posición de tomar una foto y modificaría el valor de la variable que almacena la posición actual.

7
Resultados

En este capítulo se detalla la eficiencia del sistema, a nivel de velocidad de procesamiento y de eficiencia. Se hace referencia a los fallos que provocan detenciones de sistema repasando, una a una, las fases del procesamiento del conjunto hasta la colocación de la pieza en su lugar designado.

7.1. Resultados de eficacia

Antes de comenzar a analizar los resultados, se debe comentar que las piezas se distribuyen en la zona de trabajo de manera aleatoria cumpliendo una serie de restricciones, que se también se pueden encontrar descritas en el Capítulo 9, ya que su corrección queda pendiente para ampliaciones venideras. Estas consideraciones a la hora de colocar las piezas son las siguientes:

- Las piezas apiladas no deben superar una altura máxima de tres.
- A la hora de colocar hileras y piezas únicas, encajadas o rotadas en un mismo modelo, se debe comprobar que si se recogen antes las piezas simples, la cámara no impacte contra las apiladas. Para garantizar esto se debe dejar un espacio de unas cuatro o cinco posiciones vacías tras la pieza a recoger.
- Las piezas se deben ubicar dentro de la zona de trabajo del robot, ya que en caso de situarse fuera de su alcance, el robot provocará un fallo que detendrá el sistema.
- Se debe garantizar que no impactará la cámara contra las piezas ya depositadas a la hora de ir a dejar una nueva pieza. Para esto, se debe comprobar que no haya más de dos piezas rojas en la primera iteración si hay alguna amarilla y que en sucesivas fases nunca superará en más de dos unidades el número de piezas azules al número de piezas rojas. Se recomienda colocar la mayoría de piezas amarillas en las primeras instancias.
- Una última recomendación sería colocar las piezas fuera de la zona de reflejo de los focos, ya que esto puede reducir la eficiencia del sistema.

El sistema ha sido probado en gran cantidad de casos, siempre respetando las restricciones previamente mencionadas. El conjunto presenta un alto porcentaje de acierto y fallos esporádicos debidos a los motivos que se expondrán a continuación. Este proyecto requiere un alto grado de precisión si se desea que opere correctamente y hay gran cantidad de elementos que ponen en peligro esta exigencia. Algunos de estos factores de riesgo son las tuercas que sujetan las dos partes de la pieza (descrita en el Apéndice A) o las condiciones de luz de la sala en el momento de ejecutar el programa.

Para estudiar el rendimiento, se han analizado los aciertos y equivocaciones en cada uno de los pasos que sigue el sistema: procesamiento de la imagen de color, de profundidad y la recogida de las piezas, ya que se considera que, de haber llevado a cabo correctamente estos pasos, el depositado de la pieza se llevará a cabo correctamente[1]. Los procesos se estudian en orden de ejecución, por lo que si fallara el caso de reconocimiento en la imagen de color, ni siquiera procedería a analizar la imagen de profundidad y así sucesivamente.

7.1.1. Reconocimiento en la imagen de color

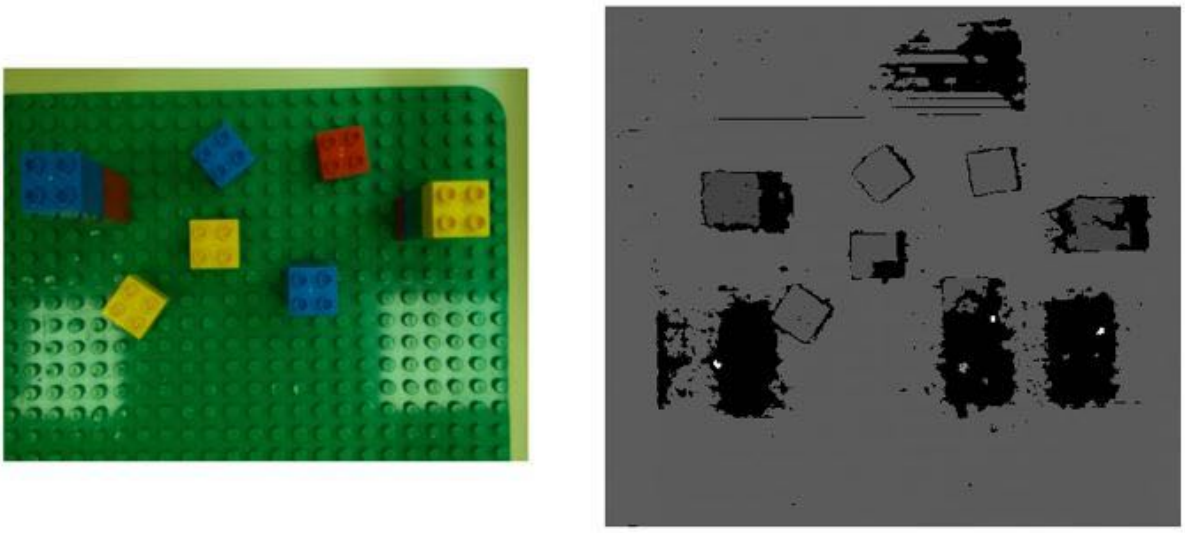
Como se explicó en la Sección 4.2, el primer paso de cara a reconocer las piezas se basa en la máscara de color, que no solo encuentra dónde está sino de qué color es. Por ello, si hubiera algún error en esta fase, el conjunto podría llegar a operar con las demás piezas de la situación omitiendo por completo la existencia del bloque en cuestión.

La máscara se diseñó a partir de una imagen tomada por la cámara en la posición de captura de imágenes y desde ahí se seleccionaron los colores deseados. Debido a estas condiciones de diseño, el porcentaje de acierto de la máscara es muy alto, pero se ve reducido en situaciones concretas, provocadas, en la mayoría de los casos, a las condiciones de iluminación.

La claridad en la calle es tal, que aun bajando los estores del laboratorio se perciben diferencias en los distintos momentos del día. Mientras que por la tarde el funcionamiento es mejor con el estor bajado, por la mañana es recomendable subirlo hasta media altura, de manera que el sol no incida directamente en ninguna de las piezas. En el laboratorio es común trabajar con la luz encendida, pero esto devuelve unos reflejos sobre la matriz señalados en la Figura 7.1 que modifican de manera considerable el color de la pieza. Este efecto es más acusado en las piezas azules que en las amarillas, como se puede apreciar en la Figura 7.2.

Otro fallo destacado se produce al apilar varias piezas del mismo color, como ya se definió en la Sección 5.1 ya que considera el lateral como si fuera parte de la propia pieza, lo que se traduce en un fallo en la obtención de las coordenadas. Este error también suele generar una equivocación en el cálculo del ángulo de rotación de la pieza en caso de que la hilera se ubique en la parte derecha del tapete. Al seleccionar el lateral que se debe analizar, toma la arista señalada en la imagen, que en lugar encontrarse alineada con la matriz se encuentra desviada alrededor de veinte grados. No obstante, este error no provoca ningún fallo general, dado que al ser piezas apiladas, este dato no se tiene en cuenta. Sin embargo, este error también es muy dependiente de la iluminación y, en caso de contar únicamente con la luz artificial de la sala se reduce considerablemente este tipo de equivocación.

[1]El único caso en el que puede fallar al depositar la pieza se da si, aunque consiga agarrarla, la pieza está mal cogida. Si la ha cogido por un extremo, luego no coincidirán los huecos de la pieza con los tetones de la matriz y si no está paralela a la superficie de la mesa, al soltarla producirá un error de colisión en el eje



(a) (b)

Figura 7.1. Muestra del tapiz con el efecto de los reflejos en (a) la imagen RGB y (b) la imagen de profundidad.

7.1.2. Reconocimiento en la imagen de profundidad

Los fallos producidos en esta fase del sistema se deben, en su totalidad, a dos factores: la iluminación de la sala y bloqueos de la cámara de profundidad.

Los reflejos de las luces del laboratorio afectan más a la imagen de profundidad que a la de color, ya que aunque en el primer caso varíen las tonalidades, en este generan una zona de reflejo en la que ni siquiera se aprecian las piezas, como se observa en la Figura 7.1, provocando así una zona ciega. Las piezas amarillas son las que más reflejan la luz, lo que se traduce en que el porcentaje de equivocación en esta fase sea mayor en los bloques de dicho color. Si hay suficiente luz alrededor, sencillamente es preciso apagar la luz artificial unos instantes, lo suficiente para que se pueda tomar una nueva instantánea.

Ya se indicó en Sección 4.3 que fue necesario incluir un bucle que repitiese la imagen de profundidad en caso de obtener valores erróneos, y esto se debe al segundo motivo de fallo principal: en ocasiones y sin previo aviso las imágenes de profundidad tomadas por la cámara presentan un patrón erróneo, con fondo completamente negro y líneas diagonales blancas, como se puede observar en la Figura 7.3. Esto se prolonga durante un número indeterminado de capturas llegando a ser necesario, en ocasiones, desenchufar la cámara, lo que implica reiniciar todo el sistema.

7.1.3. Recogida de piezas

El error en esta etapa realmente se produce en la traducción de las coordenadas del sistema de la imagen al del robot. Se estudian los dos casos posibles de manera independiente debido a que la forma de calcular ambas posiciones difiere en cada situación. Para facilitar la comprensión de los fallos, al comenzar cada apartado se hará un breve resumen del funcionamiento de cada método.

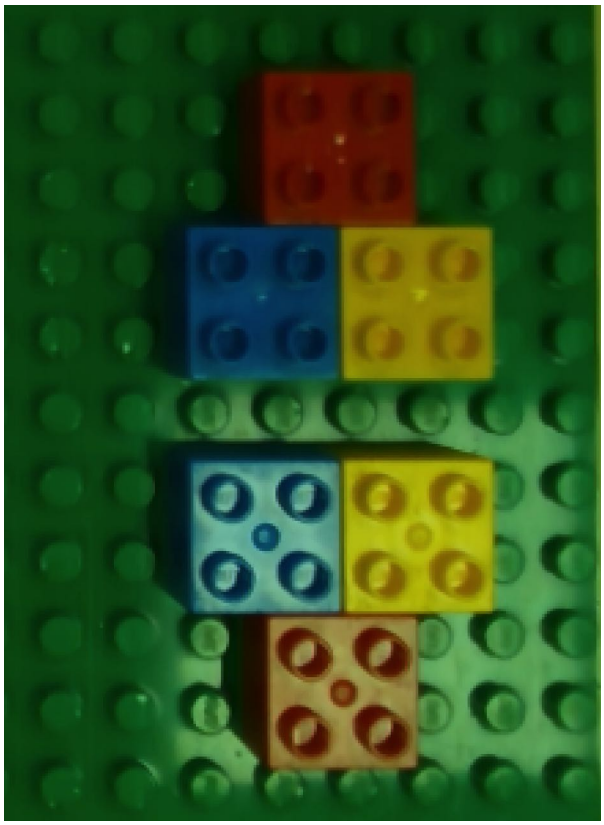


Figura 7.2. Muestra de los cambios de color de las piezas bajo el foco.

El único fallo común se produce cuando el usuario coloca una pieza en un lugar fuera del área de trabajo. El conjunto no identifica los lugares no accesibles por el robot, por lo que si identifica la pieza intentará cogerla, devolviendo el robot un fallo de sistema.

7.1.3.1. Piezas rotadas

Resumiendo lo indicado en capítulos anteriores, a la hora de calcular la posición de las piezas rotadas se toman las coordenadas de una pieza ubicada en el extremo superior derecho de la matriz de puntos. Con esos datos, se calcula la distancia entre ambas piezas en la imagen, se transforma esa distancia a las unidades del robot y se opera con las coordenadas del punto de referencia.

El principal motivo de fallo en este algoritmo reside en la variabilidad del punto de referencia en el sistema de coordenadas de la imagen. Dado que el conjunto es muy sensible a pequeñas variaciones, cualquier cambio mínimo en la posición afecta al sistema. En este caso, las tuercas que sujetan ambas partes de la pieza se sueltan paulatinamente con el funcionamiento del robot, llegando a provocar variaciones de hasta diez píxeles. El error definitivo en cuanto al movimiento del robot es de apenas unos milímetros, suficiente para que el efector impacte contra la pieza.

Una forma de solventar este problema es habilitar también la comunicación robot-MATLAB (es decir, en sentido inverso del que actualmente se emplea) de manera que el robot pueda devolver las coordenadas de su posición actual (obtenida con la instrucción *CRobT*). Con esta información y las coordenadas de la imagen de los puntos donde se depositan las piezas (que se calcularían al comenzar cada ejecución del programa) se evitaría la inserción manual de puntos y coordenadas concretas.

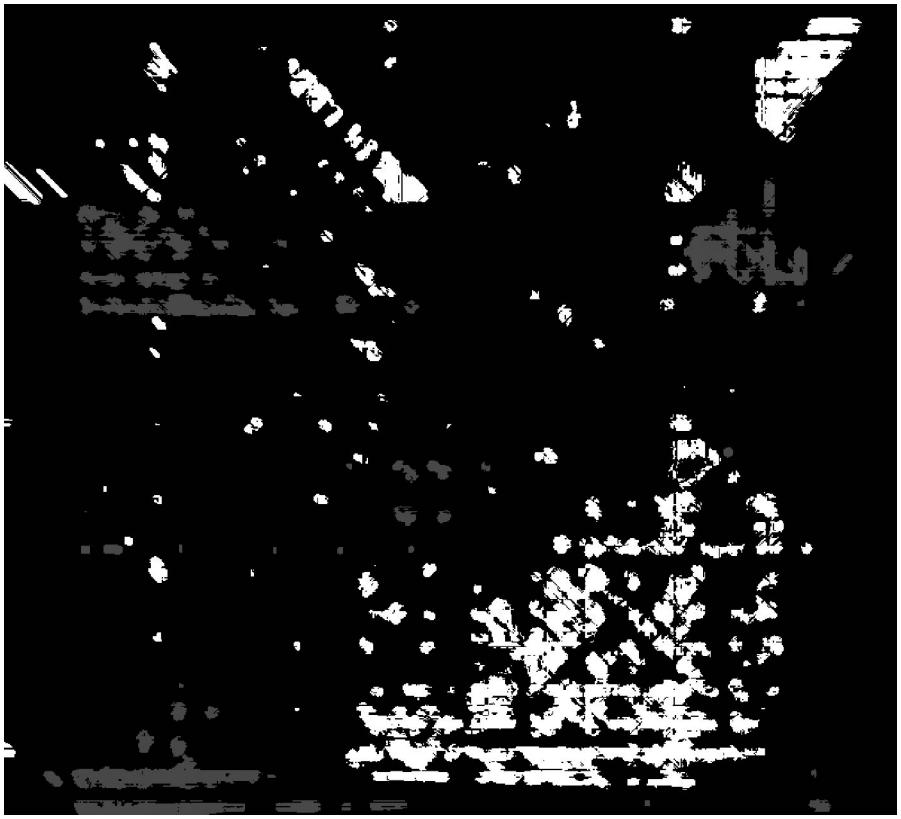


Figura 7.3. Patrón erróneo devuelto en ocasiones por la cámara de profundidad

A la hora de colocar las piezas rotadas se debe tener en cuenta tanto la posición en la que se colocan como el lugar. Por una parte es de vital importancia que la pieza se encuentre correctamente ubicada sobre los tetones de la matriz. De otra forma, al intentar coger la pieza esta podría ladearse cayendo hacia el hueco de la matriz y esto provocaría un error, bien de colisión del sistema o a la hora de depositar la pieza, ya que al estar ladeada no encajaría correctamente.

La otra consideración al preparar el escenario es que las piezas que no se encuentren encajadas presenten un giro superior a ocho grados, ya que este parámetro es el empleado para discernir si una pieza se encuentra encajada o no y por ende la altura a la que debe aproximarse el autómata.

7.1.3.2. Piezas apiladas

En capítulos anteriores se señaló que para calcular las coordenadas de este tipo de piezas se localizaría el hueco de la matriz de puntos en el que se encuentren encajadas. Una vez obtenida esta información (basándose en el número de píxeles por cada posición), lo convierte a las coordenadas del robot multiplicando por el número de milímetros que supone cada ubicación.

Este procedimiento tiene un mayor porcentaje de fallo en la parte inferior de la imagen, donde la coordenada Y del robot se distorsiona en mayor medida. Esto se intentó corregir empleando toda la información señalada en el Anexo B, pero en determinadas ocasiones el ruido de la imagen hace que esa corrección no sea suficiente.

El otro problema fundamental está en la extracción por puntos que, a pesar de ser un sistema muy refinado, en determinadas ubicaciones de la matriz termina por levantar la pieza de debajo.

Esta situación, de ocurrir, sucede al extraer la segunda pieza de una hilera de tres y el bloque desenganchado sería el último. En la mayoría de ocasiones no supone ningún contratiempo, ya que aunque desencaja la pieza inferior, no se la lleva consigo y queda depositada sobre la matriz.

7.2. Resultados de tiempo

En este tipo de ejecuciones también es un parámetro importante el tiempo de ejecución del programa. Para obtener estos datos, se ha ejecutado varias veces el programa y se han ido almacenando los datos relativos a los diversos tiempos de ejecución.

Dado que la ejecución del programa es dependiente del número de piezas que se distribuyan sobre la matriz, se han considerado los tiempos de procesamiento de las distintas fases del programa, recogidos en la Tabla 7.1[2]. Los tiempos se muestran en segundos.

Porceso realizado	Tiempo (s.)
Captura de imágenes	2.425
Procesamiento imagen de color	1.54
Procesamiento imagen de profundidad	0.017
Transformación de las coordenadas	0.021

Tabla 7.1. Tabla de tiempos de ejecución de los distintos procesos del sistema.

7.3. Ejemplos

En esta sección se muestra el funcionamiento del sistema en diversas situaciones. A continuación se mostrarán cuatro situaciones: la primera, el sistema más básico, con piezas simples encajadas en la matriz de puntos. La segunda situación presenta únicamente piezas rotadas, que como se ha indicado previamente deben encontrarse perfectamente apoyadas encima de los tetones, la tercera muestra hileras de tres piezas apiladas.. La combinación de estos tres posibles tipos de piezas no cambiaría nada en la ejecución de un único escenario, por ello no se ha considerado relevante poner un ejemplo de este tipo.
En cada subsección se presenta una tabla de tiempos, los resultados obtenidos por el programa tras procesar la imagen y los pasos realizados por el robot.

7.3.1. Caso 1

En este caso se presenta una situación en la que todas las piezas se encuentran encajadas en la matriz de puntos, es decir, sin rotar. En este primer escenario además no se apilarán las piezas.

El sistema toma una imagen, con el robot como se muestra en la Figura 7.4. A continuación, el programa de MATLAB procesa las imágenes de color y profundidad, y obtiene los resultados mostrados en la Figura 7.5, donde se puede comprobar que el sistema ha reconocido todas las piezas identificando su color, la rotación de las mismas (cero o muy cercano a ese valor en este caso) y la altura a la que se encuentran (altura uno).

[2]El tiempo de procesamiento de la imagen de profundidad indica el retardo si la primera imagen es válida para todas las piezas, es decir, si no tiene que tomar ninguna imagen adicional para obtener la iformación de todas las piezas. En caso contrario el tiempo de procesamiento dependerá del número de imágenes que deba obtener para calcular la información. En el caso más desfavorable llegó a tardar 19 segundos

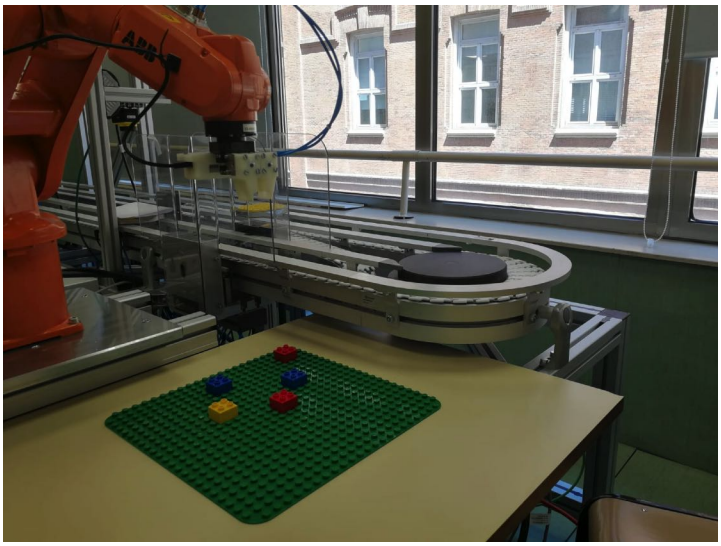


Figura 7.4. Ejemplo 1. Robot tomando la imagen con la primera disposición de piezas

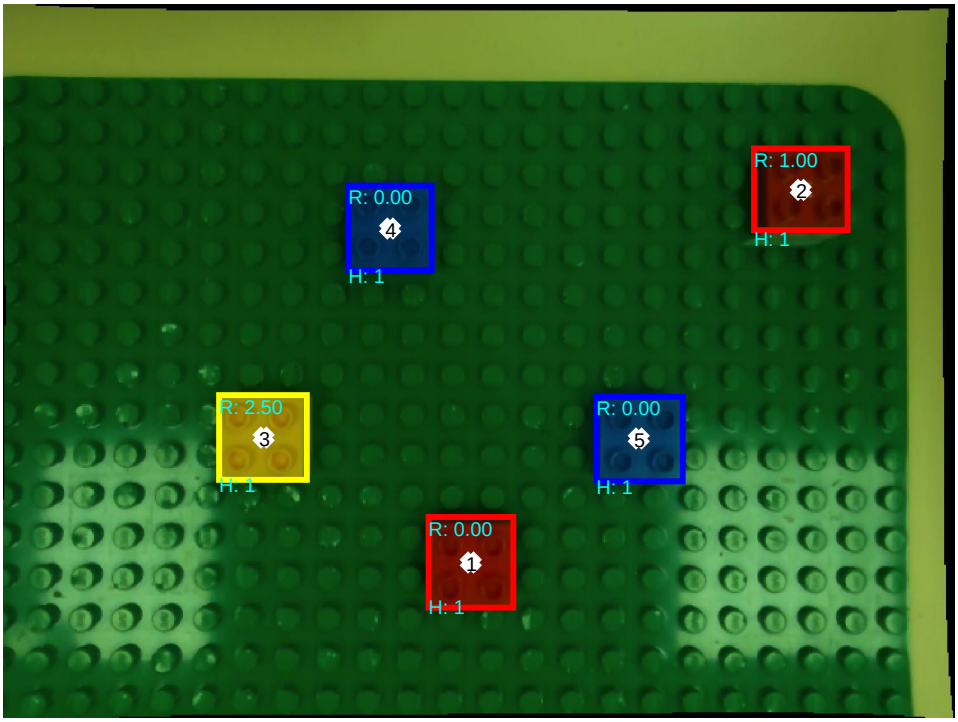


Figura 7.5. Resultados del MATLAB en la primera iteración

En la siguiente secuencia de imágenes se muestra todo el proceso de movimientos que sigue el robot: se coloca justo encima de la pieza a agarrar, abre la pinza, desciende, cierra la pinza y se eleva unos centímetros. A continuación sube a la “altura de seguridad”, ubicada unos centímetros por encima de la altura máxima que puede llegar a alcanzar el sistema y se desplaza al lugar donde debe depositar la pieza y una vez más descenderá lentamente hasta depositarla en el lugar designado Esos son los nueve pasos que se muestran en la Figura 7.7.

Una vez ordenadas todas las piezas visibles en la primera iteración, regresa a la posición desde la que toma las imágenes. Desde allí toma otra imagen con el fin de comprobar si queda alguna pieza por ordenar. En este caso concreto, al haber colocado piezas a una única altura no quedará ninguna pieza pendiente y en ese caso el programa de MATLAB cerrará la conexión con el servidor y finalizará la ejecución del programa.

Tras colocar la primera pieza, cuyo movimiento se ha mostrado con mayor detalle, el sistema fue a por la segunda pieza roja y después a por la amarilla. Esto se observa en la Figura 7.8. A la hora de depositar la pieza amarilla, se puede comprobar la importancia del orden de las piezas. La cámara no llega a impactar contra la hilera de piezas rojas, pero si hubieran sido azules (que se colocan una posición más atrás) o si hubiera habido una pieza roja más sí habría impactado. En la Figura 7.6 se observa esto con mayor detalle.

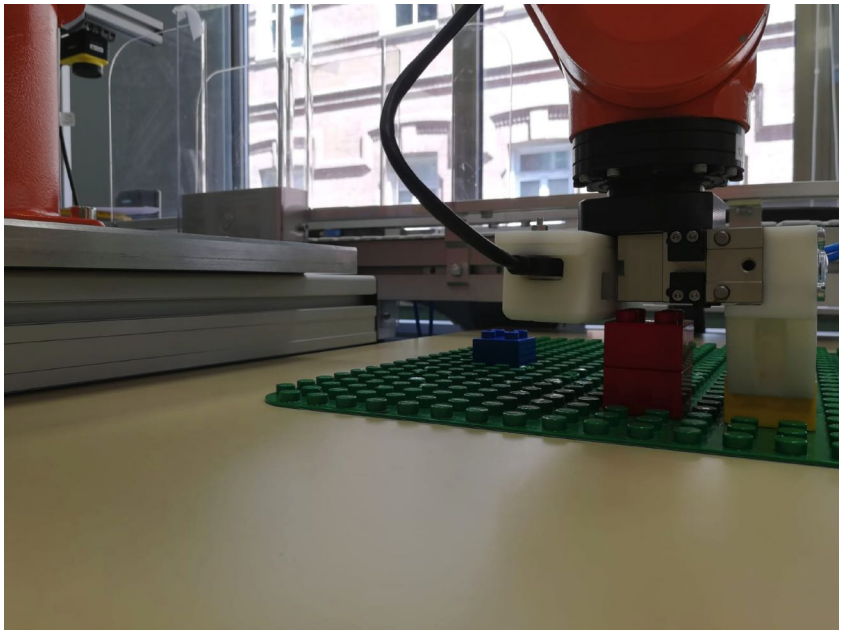


Figura 7.6. Ejemplo 1. Detalle de la colocación de la posición de la cámara en el momento de depositar la pieza amarilla. Se puede observar cómo, en caso de que hubiera habido una pieza roja más o dos o más piezas azules colocadas de una iteración anterior, la cámara habría impactado.

Finalmente, quedan por ordenar las piezas azules, movimientos que se pueden observar en la Figura 7.9. Tras colocarlas en su sitio, volverá a la posición desde la que toma las imágenes para, como se ha explicado anteriormente, verificar que no queden piezas por colocar.

En el procesamiento de la información de este ejemplo, los tiempos registrados han sido los mostrados en la Tabla 7.2.

Porceso realizado	Tiempo (s.)
Captura de imágenes	2.529
Procesamiento imagen de color	2.4
Procesamiento imagen de profundidad	2.23
Transformación de las coordenadas	0.026

Tabla 7.2. Tabla de tiempos de ejecución de los distintos procesos del sistema en el primer caso.

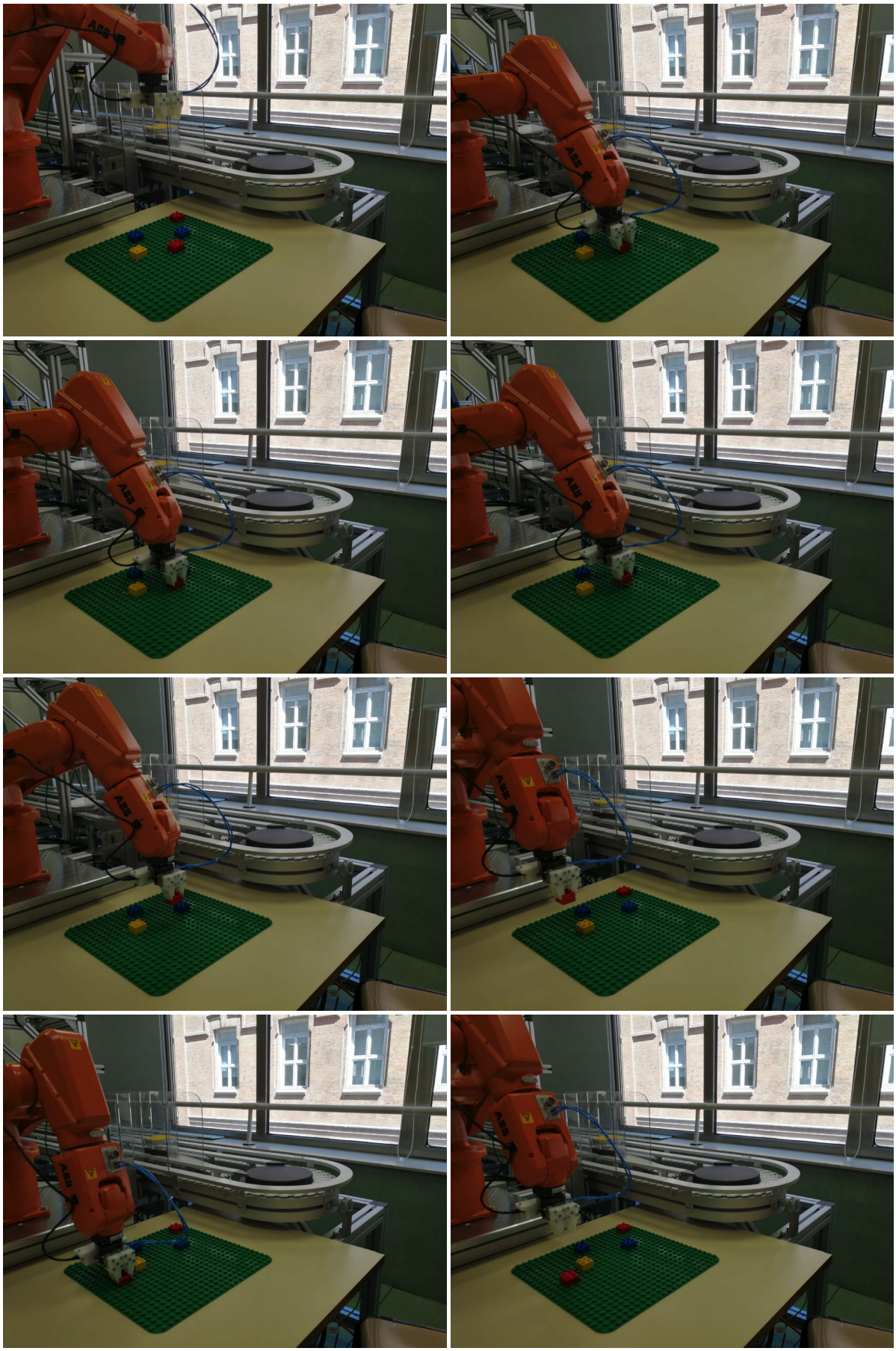


Figura 7.7. Ejemplo 1. Colocación de la primera pieza paso por paso, desde la toma de imágenes hasta el punto en el que se deposita la pieza y espera a futuras instrucciones.

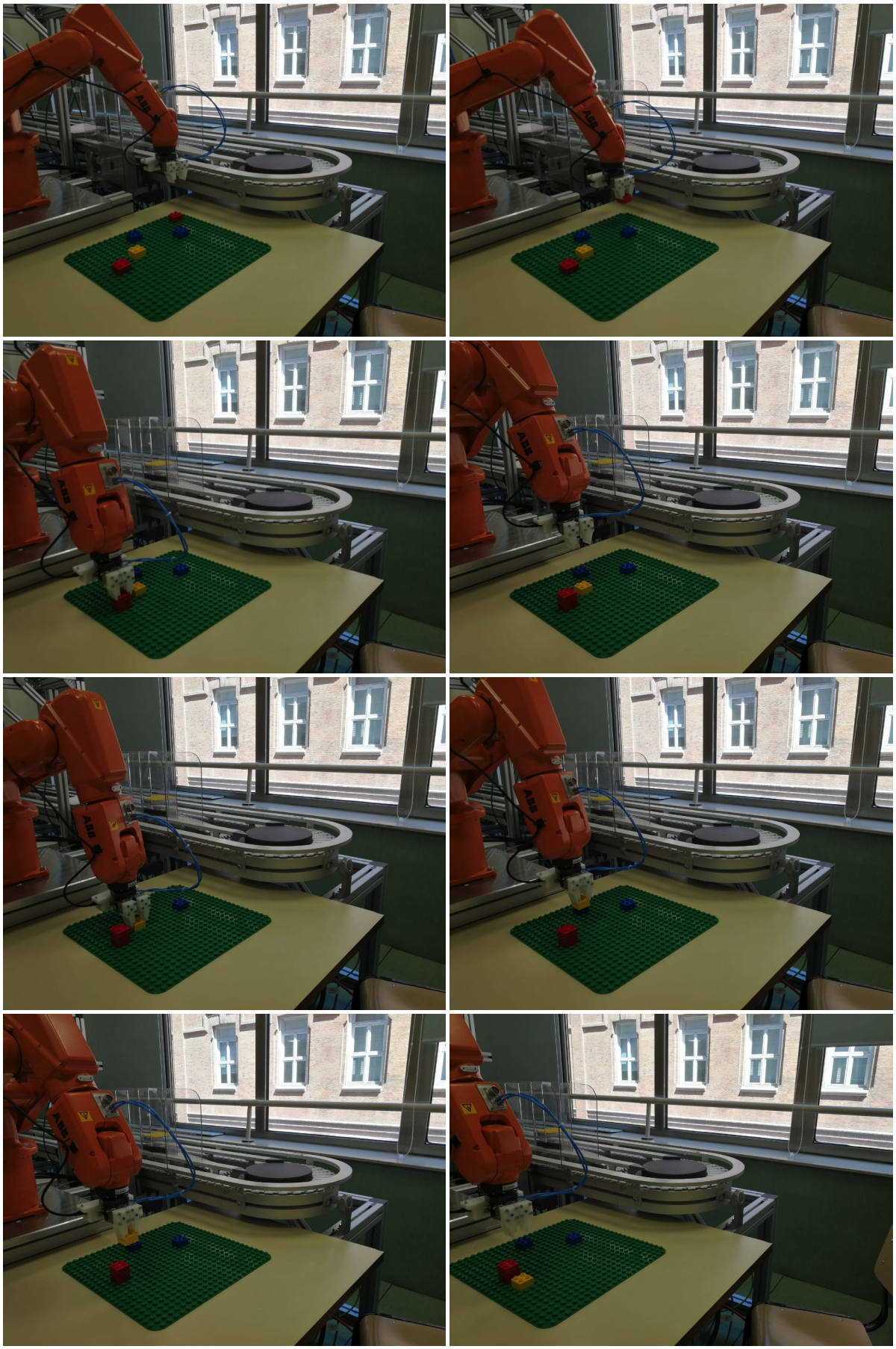


Figura 7.8. Ejemplo 1. Recogida de la segunda pieza roja y la amarilla.

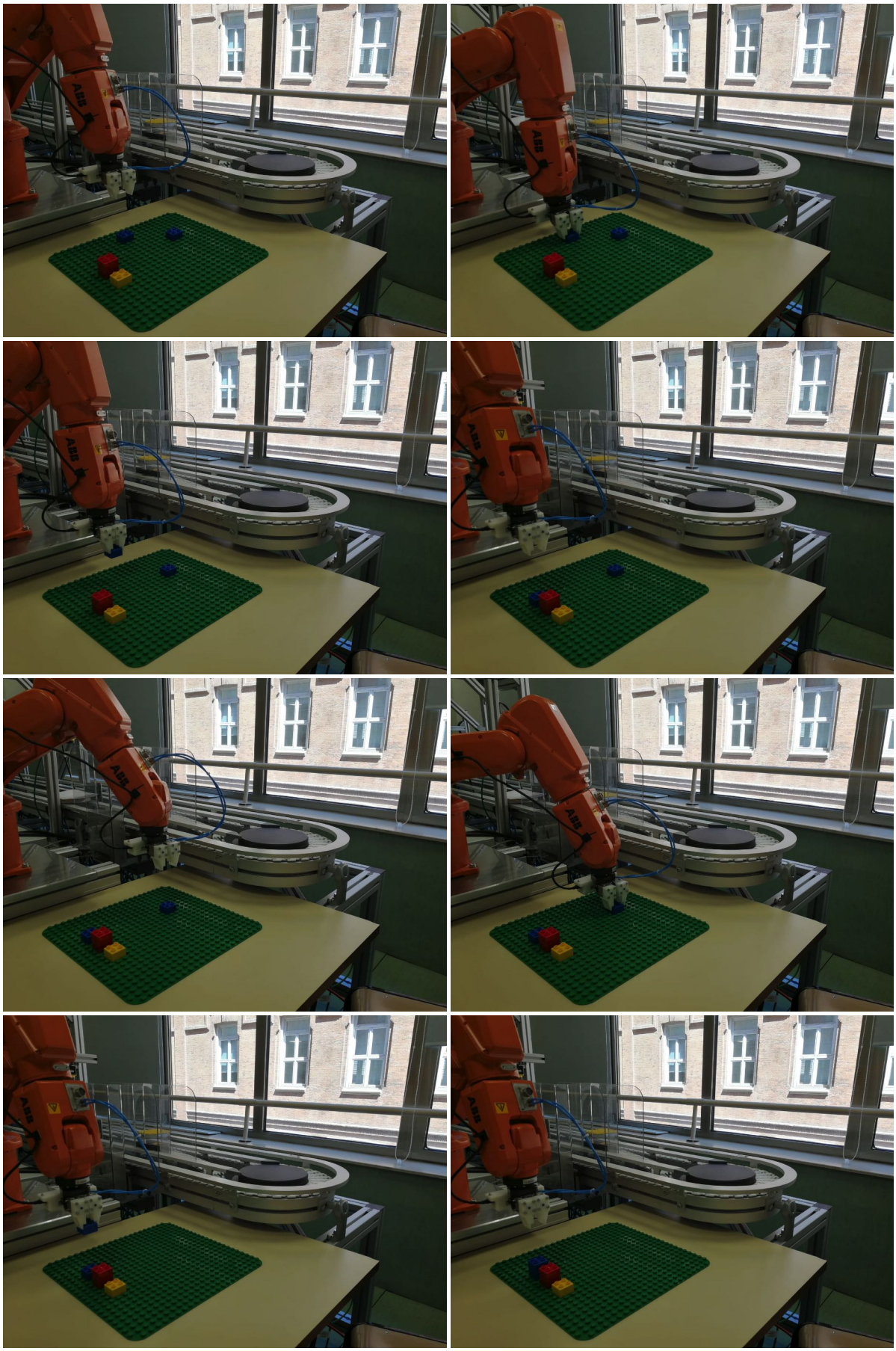


Figura 7.9. Ejemplo 1. Secuencia de ordenación de las piezas azules.

7.3.2. Caso 2

En este segundo caso se ha preparado un escenario con piezas rotadas y ubicadas sobre los tetones de la matriz de lego. En la Figura 7.10 se puede observar al robot tomando la imagen con la nueva situación y en la Figura 7.11 los resultados de procesamiento obtenidos por MATLAB. Se puede observar cómo la altura de todas las piezas continúa siendo uno mientras que la rotación en este caso varía considerablemente de unas piezas a otras.

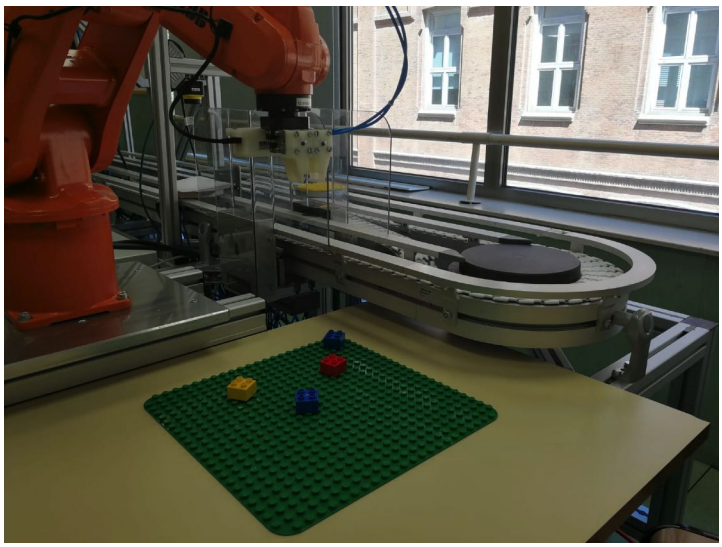


Figura 7.10. Ejemplo 2. Robot tomando la imagen con la segunda disposición de piezas

La secuencia de movimientos al encontrar una pieza rotada es ligeramente diferente a la realizada en el caso anterior. En esta situación, el robot se desplazará a la altura de seguridad a la posición de la pieza a agarrar, abrirá la pinza y solo en este momento, girará el efector para poder tomar la pieza. Tras realizar el giro, descenderá y cuando se encuentre la pieza entre las partes de la pinza la cerrará. Una vez agarrada la pieza, el efector recuperará su orientación original mientras asciende para desplazarse hasta la ubicación donde depositará la pieza. Esta secuencia se puede observar en la Figura 7.12.

Los movimientos necesarios para ordenar las de piezas restantes de este escenario se muestran en la Figura 7.13, donde se puede apreciar el giro del efector para cada uno de los casos presentados.

Los tiempos de ejecución en este escenario son los mostrados en la Tabla 7.3.

Porceso realizado	Tiempo (s.)
Captura de imágenes	2.707
Procesamiento imagen de color	2.2
Procesamiento imagen de profundidad	0.022
Transformación de las coordenadas	0.023

Tabla 7.3. Ejemplo 2. Tabla de tiempos de ejecución de los distintos procesos del sistema en el primer caso.

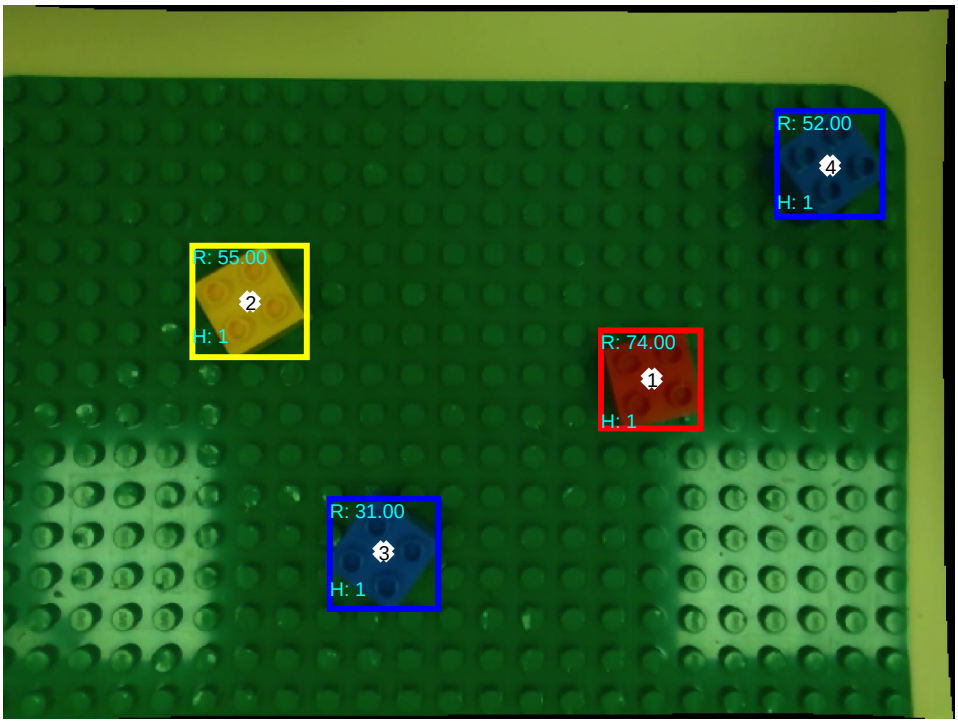


Figura 7.11. Ejemplo 2. Resultados del MATLAB en la primera iteración

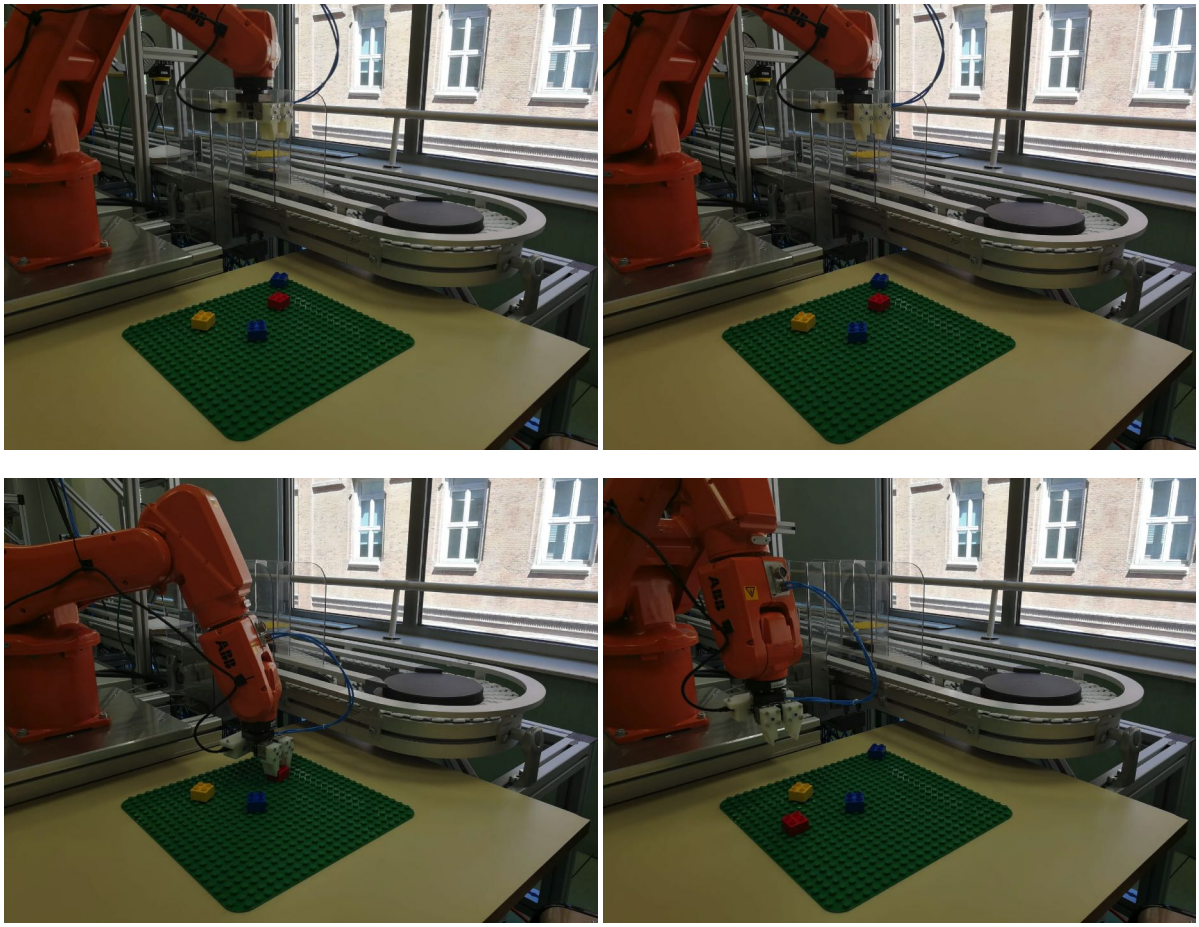


Figura 7.12. Ejemplo 2. Colocación de la primera pieza paso por paso.

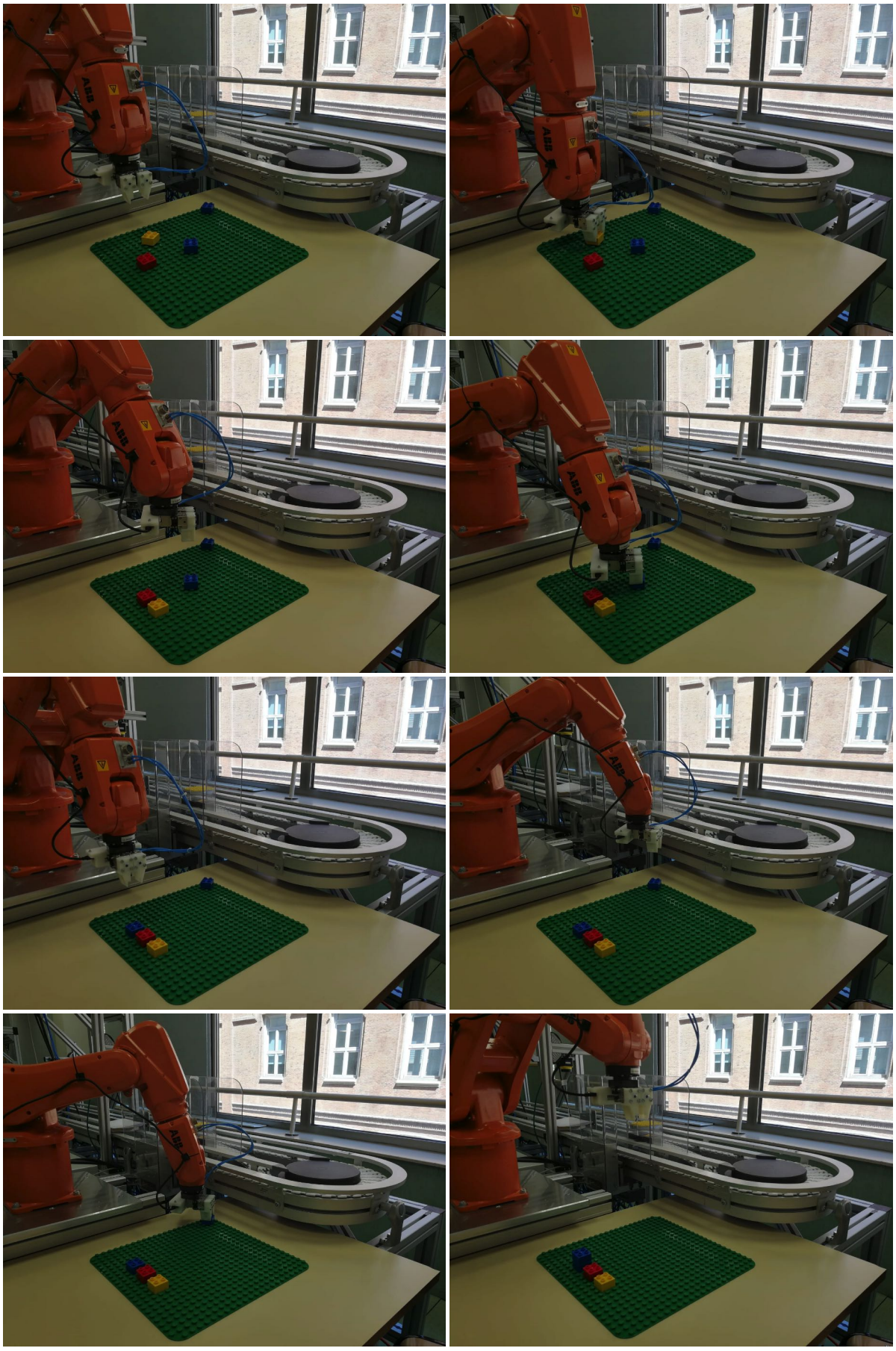


Figura 7.13. Ejemplo 2. Colocación de las sucesivas piezas de este escenario: una pieza amarilla y dos azules

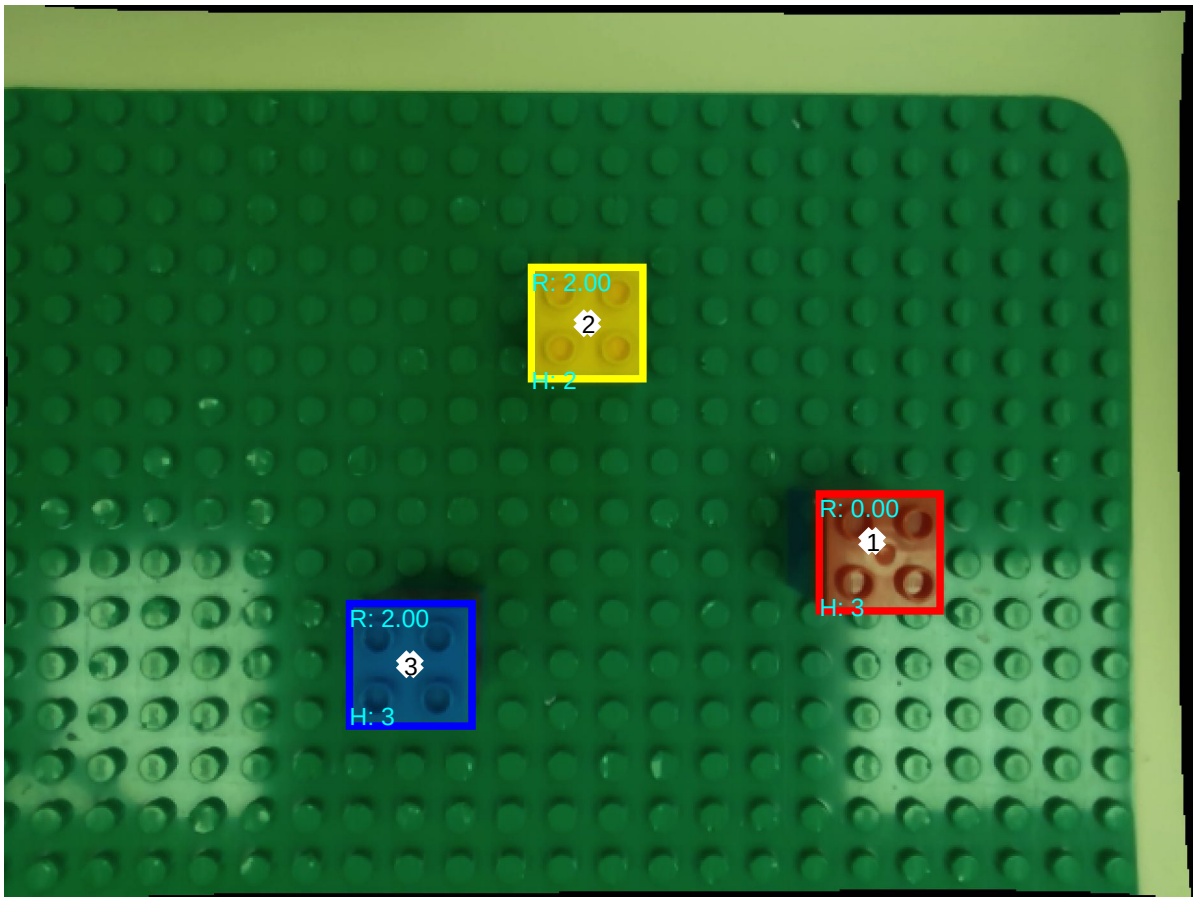


Figura 7.14. Piezas detectadas en la primera iteración del tercer ejemplo.

7.3.3. Caso 3

En este escenario se colocan únicamente piezas apiladas, en hileras de dos y tres alturas. En la (imagen, referencia) se observa el robot tomando la imagen que luego procesará el programa y cuyos resultados se observan en la (otra imagen).

La secuencia de movimientos llevada a cabo en estos casos es igual que en la Sección 7.3.1, pero en este caso, tras agarrar la pieza, no se limita a ascender, sino que se realiza una serie de movimientos, explicados en la Sección 6.3, que provocan la extracción de la pieza superior de la hilera sin mover las inferiores.

7.3.3.1. Iteración 1

En esta ocasión, al haber hileras de hasta tres piezas será necesario tomar tres imágenes, ya que en cada iteración ordenará las piezas inicialmente visibles. En la primera iteración, las piezas se muestran en la imagen Figura 7.14 y los tiempos de procesamiento en la Tabla 7.4.

Porceso realizado	Tiempo (s.)
Captura de imágenes	2.719
Procesamiento imagen de color	3.29
Procesamiento imagen de profundidad	0.019
Transformación de las coordenadas	0.023

Tabla 7.4. Tabla de tiempos de ejecución de los distintos procesos del sistema en la primera iteración del tercer caso.

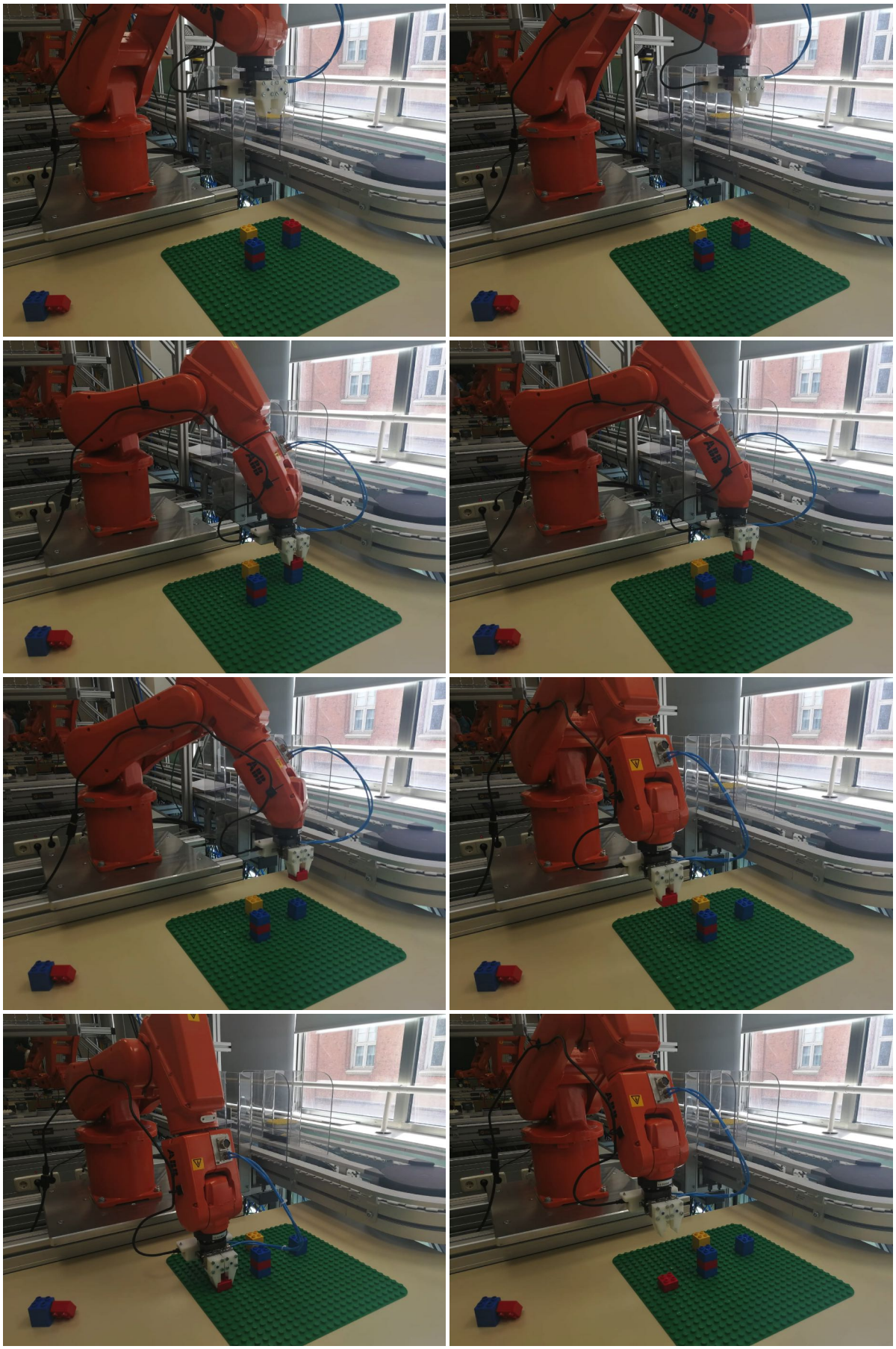


Figura 7.15. Ejemplo 3. Colocación de la primera pieza de este escenario

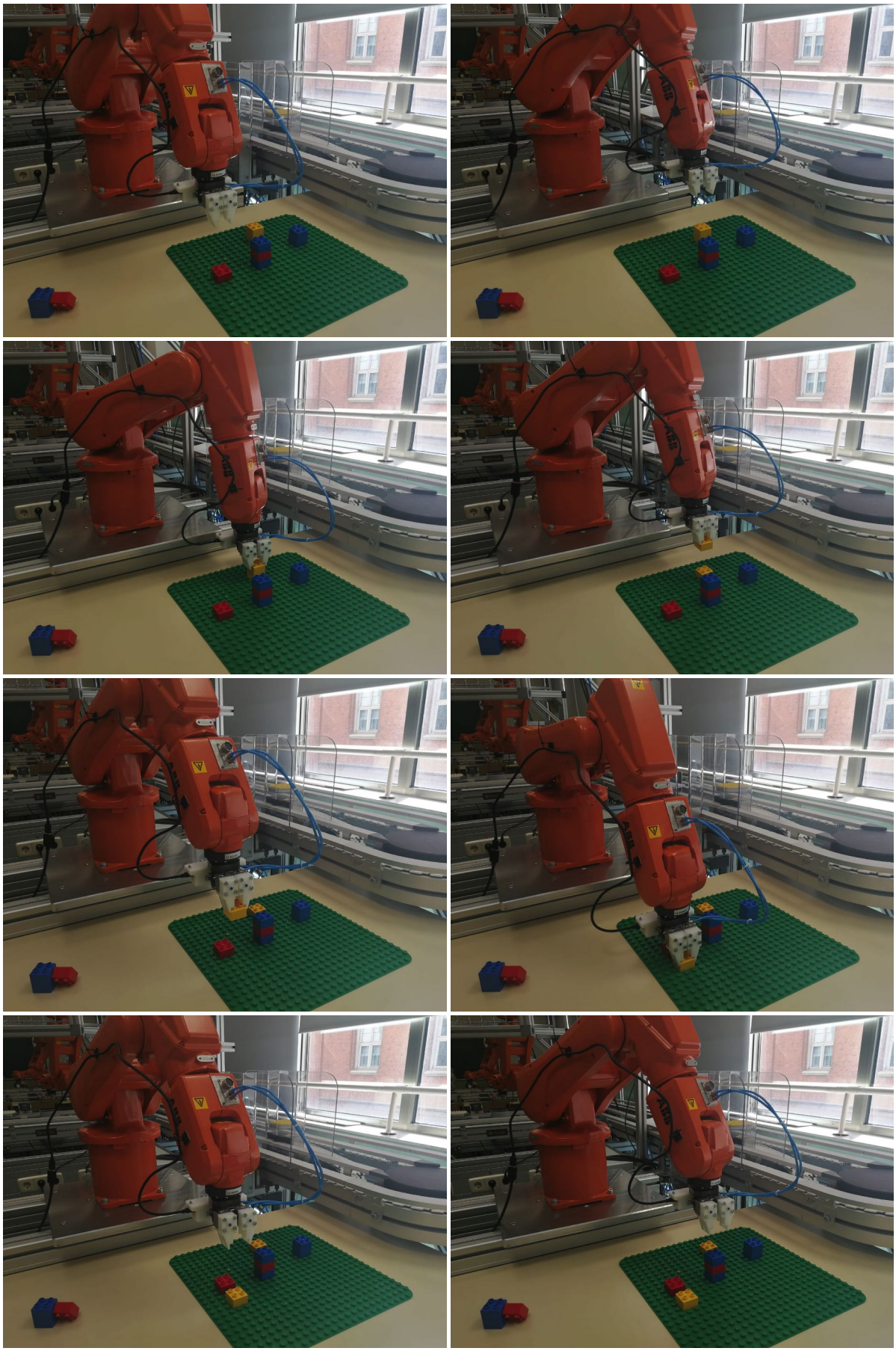


Figura 7.16. Ejemplo 3. Colocación de la segunda pieza de este escenario

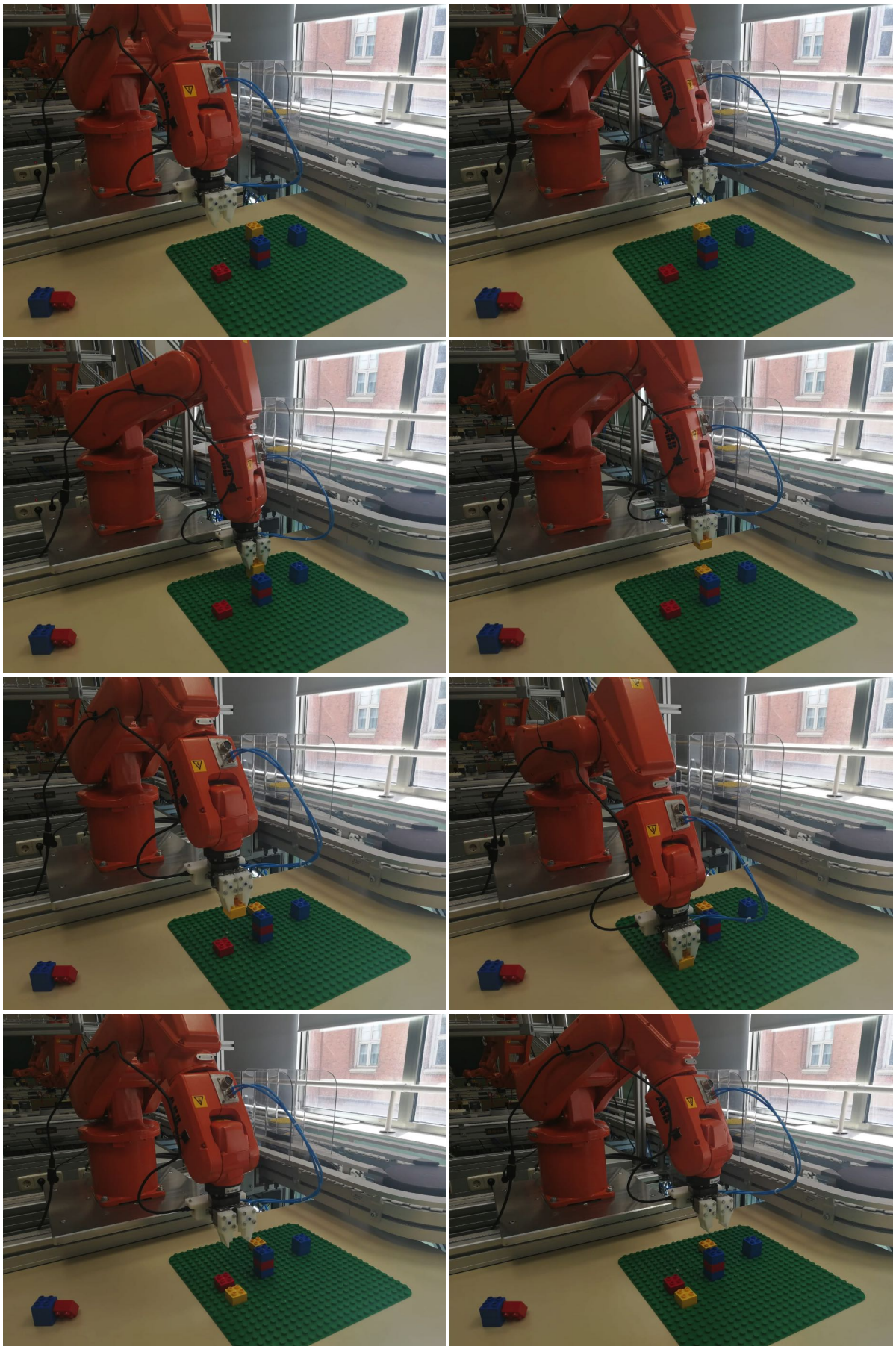


Figura 7.17. Ejemplo 3. Colocación de la segunda pieza de este escenario

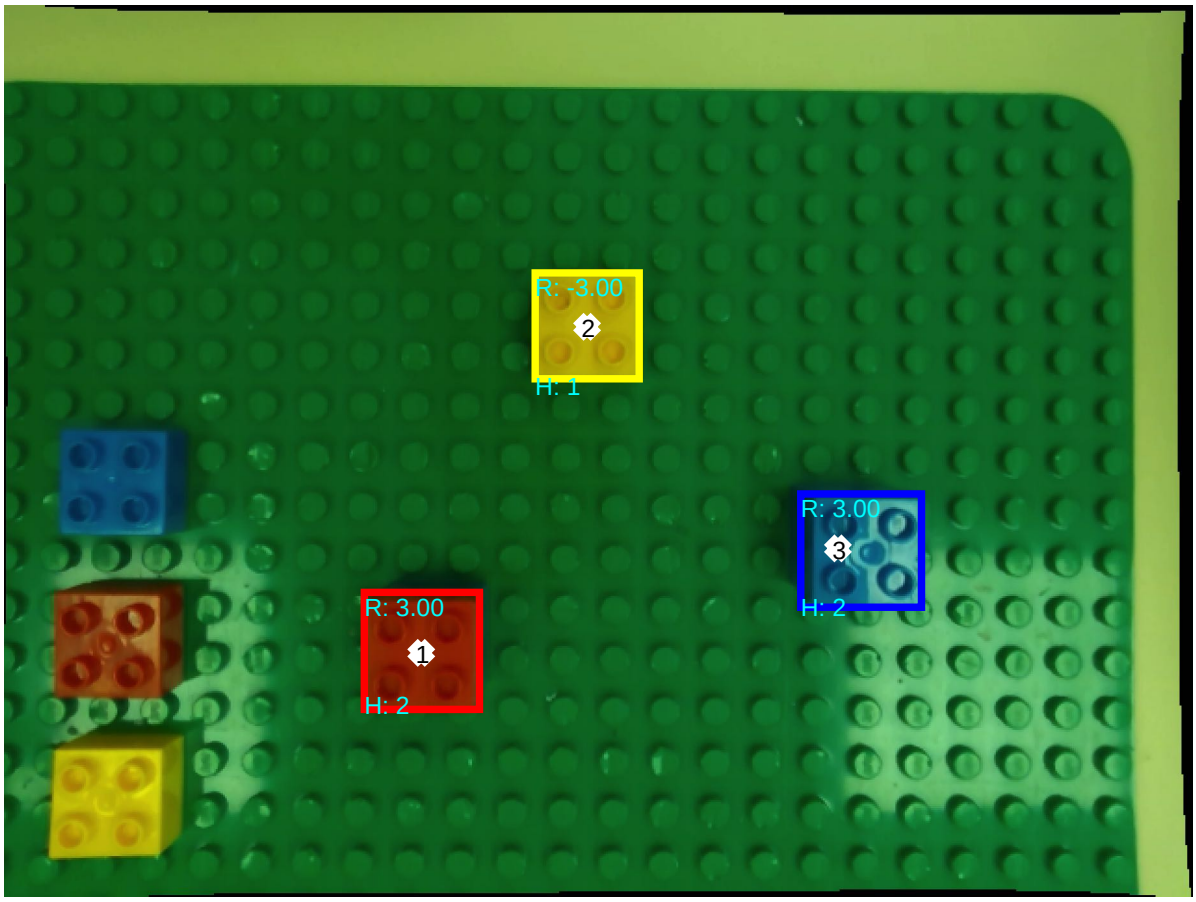


Figura 7.18. Piezas detectadas en la segunda iteración del tercer ejemplo.

7.3.3.2. Iteración 2

En la segunda iteración, el robot toma otra imagen y analiza de nuevo los resultados obtenidos partiendo de cero, es decir, no almacena ninguna información. Una vez más, los resultados obtenidos se observan en la Figura 7.18 y los tiempos de procesamiento asociados a esta iteración se muestran en la Tabla 7.5.

Porceso realizado	Tiempo (s.)
Captura de imágenes	2.283
Procesamiento imagen de color	0.397
Procesamiento imagen de profundidad	2.209
Transformación de las coordenadas	0.015

Tabla 7.5. Tabla de tiempos de ejecución de los distintos procesos del sistema en la segunda iteración del tercer caso.

En las siguientes páginas se muestra la secuencia de movimientos realizados por el robot con el fin de llevar a cabo la ordenación de las piezas analizadas e identificadas en el procesamiento de la imagen de color.

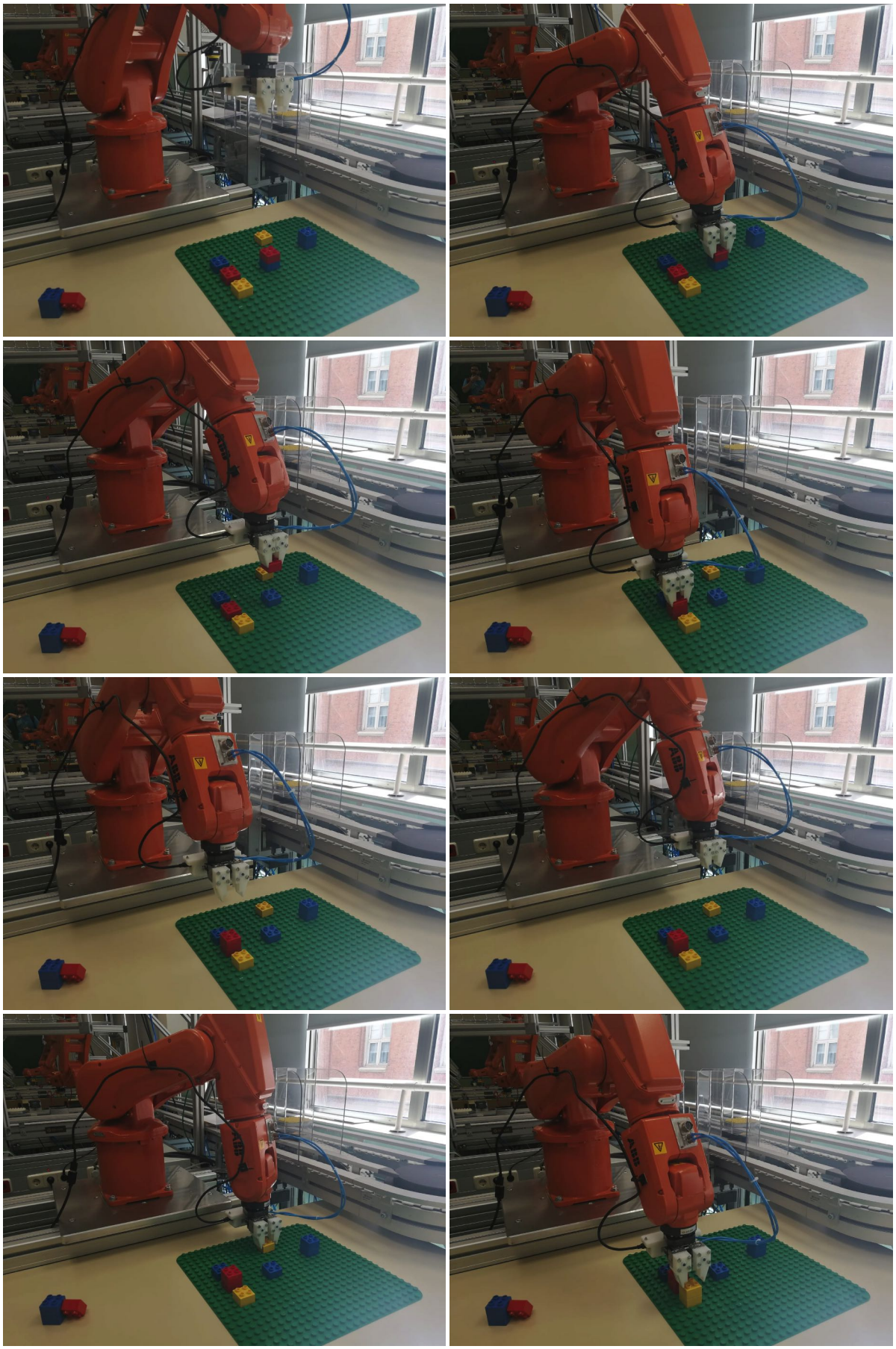


Figura 7.19. Ejemplo 3. Colocación de las piezas en la segunda iteración de este escenario

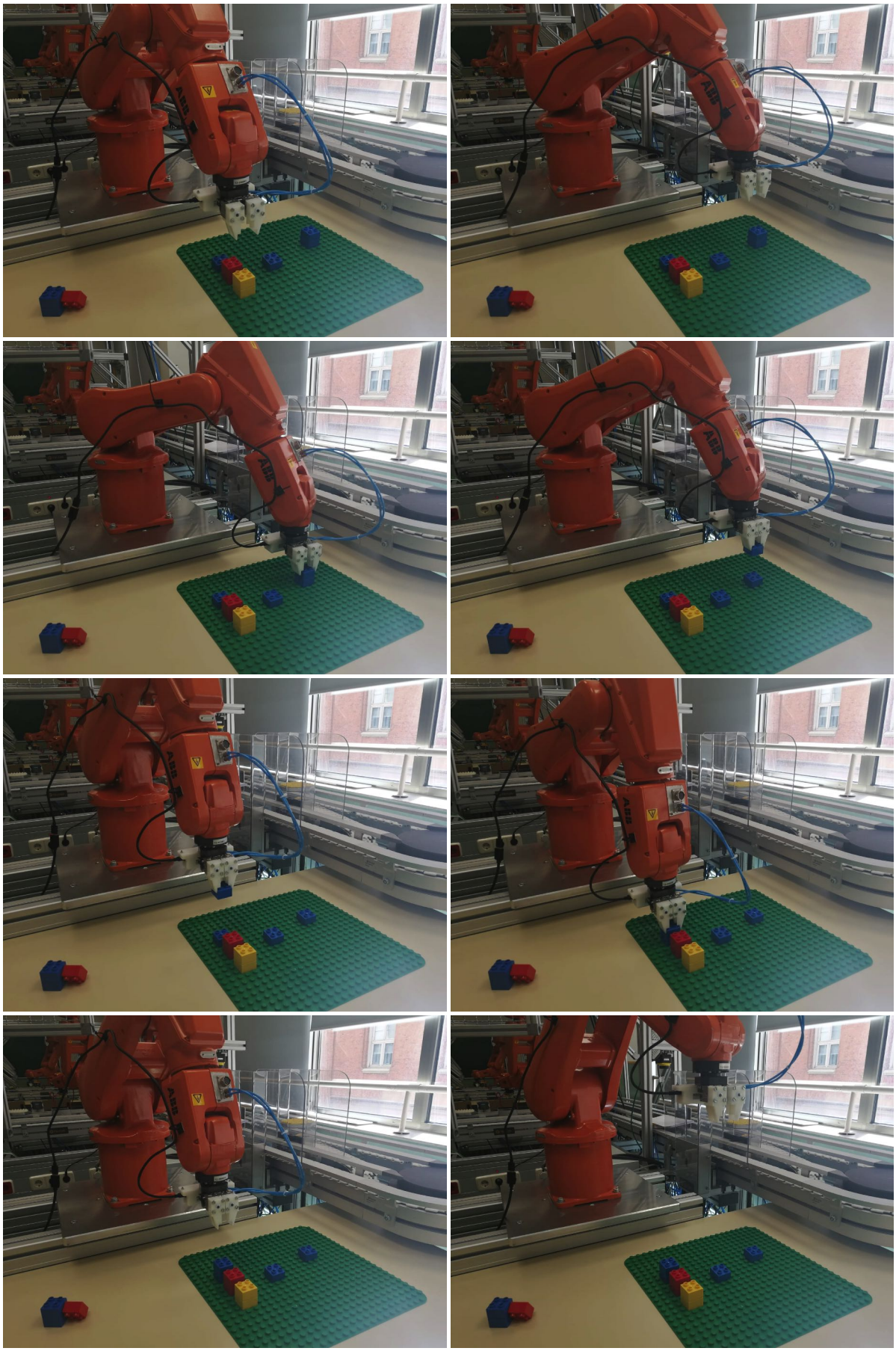


Figura 7.20. Ejemplo 3. Colocación de las piezas en la segunda iteración de este escenario

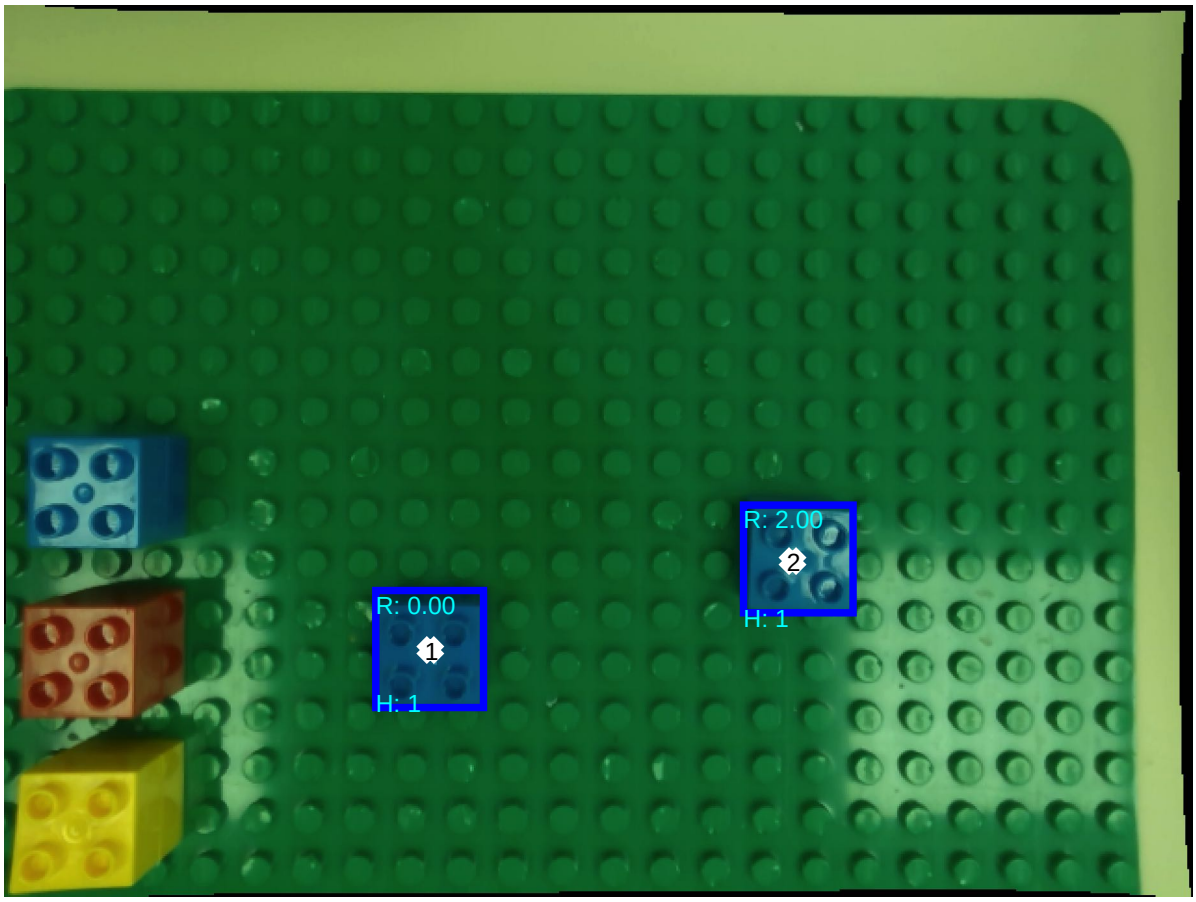


Figura 7.21. Piezas detectadas en la tercera iteración del tercer ejemplo.

7.3.3.3. Iteración 3

Finalmente queda una última iteración por llevar a cabo una última iteración, en la que se recogerán las últimas piezas de las dos hileras de tres piezas, como se puede observar en la Figura 7.21. De nuevo, los resultados de procesamiento de esta iteración se pueden comprobar en la Tabla 7.6 donde se puede comprobar en comparación a resultados de otras iteraciones o ejemplos que el tiempo de procesamiento de las imágenes es muy dependiente de la cantidad de piezas que haya, mientras que en el caso de la transformación de coordenadas, ese dato es irrelevante.

Porceso realizado	Tiempo (s.)
Captura de imágenes	2.252
Procesamiento imagen de color	0.195
Procesamiento imagen de profundidad	0.002
Transformación de las coordenadas	0.003

Tabla 7.6. Tabla de tiempos de ejecución de los distintos procesos del sistema en la tercera iteración del tercer caso.

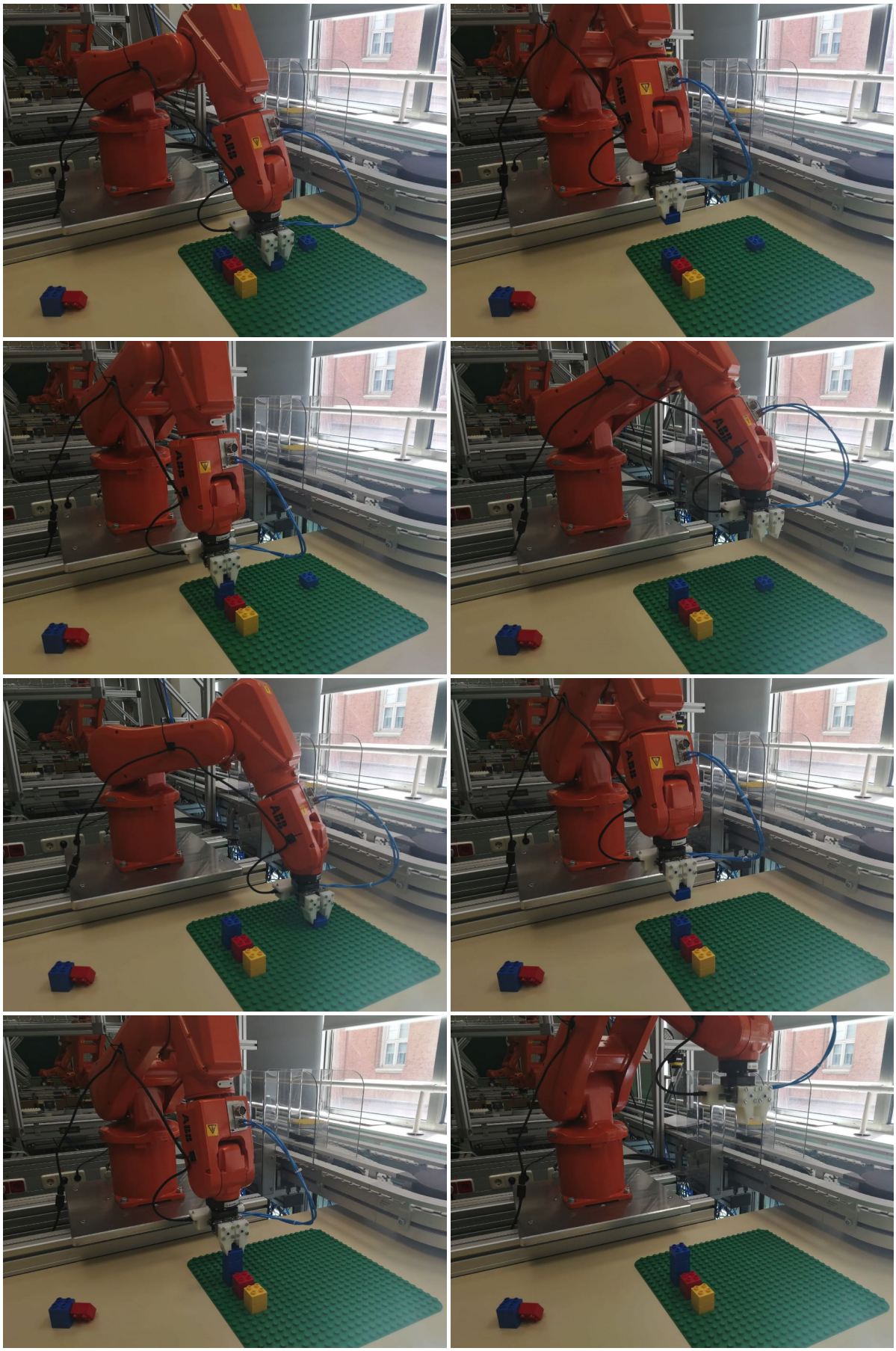


Figura 7.22. Ejemplo 3. Colocación de las piezas en la tercera iteración de este escenario

8

Conclusiones

El presente capítulo expone las conclusiones finales del proyecto extraídas a partir de los resultados obtenidos de las diferentes pruebas realizadas.

Durante la elaboración de este proyecto se ha desarrollado un sistema compuesto por una cámara de color y profundidad, un robot ABB IRB-120 y un ordenador con MATLAB capaz de identificar, localizar, recoger y ordenar piezas dispuestas de manera aleatoria sobre un espacio de trabajo designado. Estas piezas se pueden encontrar encajadas en la matriz de puntos o rotadas y depositadas sobre ella. En el caso de estar encajadas, el conjunto es capaz de colocar hileras de hasta tres piezas apiladas, separándolas gracias a una secuencia de movimientos que garantiza que solo se tomará la pieza deseada.

Respecto a los objetivos del proyecto, señalados en la Sección 1.2, se han conseguido en su totalidad, con la salvedad de que la aleatoriedad de la colocación de las piezas se encuentra limitada. Esta limitación proviene de la necesidad de verificar que al intentar coger una pieza, la cámara no impacte contra una hilera o que al depositar piezas situadas más próximas al usuario no haya colocadas un número superior de piezas cuya ubicación esté por detrás. En conjunto, la nueva implementación presenta muchas más funcionalidades que la anterior [Gar18]. Entre estas nuevas funcionalidades se destaca la posibilidad de poder separar y ordenar piezas apiladas hasta una altura máxima de tres y la recogida y colocación de piezas rotadas y posicionadas sobre la matriz de LEGO.

Tomando un punto de vista más global, como ya se indicó en la Sección 1.1, este proyecto es sencillo en comparación con los sistemas actualmente implementados en industrias y fábricas a pesar de que estos sistemas se encuentran diseñados para funcionar en condiciones ideales y admiten menos variabilidad. No obstante, es un gran avance en la tecnología de los robots industriales de la universidad.

9

Futuros desarrollos

El último capítulo recoge una serie de mejoras que podrían realizarse en el futuro partiendo del elaborado en el presente proyecto. Se relatan las posibles mejoras y se proponen diversas maneras de llevarlas a cabo

---

Este proyecto es solo el principio de un desarrollo mucho más ambicioso, por lo que los puntos de mejora son amplios y variados. Debido a las infinitas direcciones que podría tomar el proyecto respecto a la incorporación de nuevas funcionalidades, en este apartado solo se tratarán las ampliaciones orientadas a mejorar el procesamiento actual: el reconocimiento, agarre y ordenación de piezas colocadas de manera aleatoria sobre el área de trabajo.

Una primera mejora consistiría en hacer que el sistema identifique las posiciones válidas, dentro de la zona de trabajo. De esta forma, en caso de que el usuario colocara por descuido una pieza a la que el robot no pueda acceder, el sistema no produciría un error, sino que informaría al usuario de la condición de la pieza y la omitiría a la hora de ordenarlas.

Actualmente, el sistema requiere que el usuario esté pendiente de la ejecución del programa. MATLAB deposita en el *socket* la información de la pieza que se va a proceder a ordenar y hasta que el usuario no presione una tecla no se enviará la información de la siguiente. Esto se realiza para evitar que la información de las distintas piezas se sobrescriba en el *socket* o que se ejecute la siguiente iteración antes de que el robot esté correctamente posicionado.
Esto se podría conseguir habilitando la comunicación del robot al MATLAB, de forma que este enviara una señal según finalizara todos los movimientos requeridos en ese ciclo de manera que, en ese momento, MATLAB procediera a continuar su ejecución.

Otro de los problemas de este sistema reside en la incapacidad de detectar si la cámara impactará contra otras piezas al descender para agarrar una en concreto. Este problema solo se da cuando en un máximo de cuatro posiciones por detrás de la pieza seleccionada hay alguna hilera de piezas, por lo que el remedio más sencillo sería evaluar si hay alguna torre en la cuadrícula en esas posiciones y en caso afirmativo girar el efector noventa grados y evaluar las nuevas posiciones de riesgo.

A la hora de preparar un escenario de ejecución, una de las consideraciones que se debe tener es la de colocar las piezas de manera que, al depositarlas, no haya más piezas azules que rojas ni que amarillas. Esto se debe a la posición que se ha designado para colocar las piezas, ya que de otra forma la cámara impactaría contra las piezas ya depositadas. La manera más sencilla de solucionar este problema es modificar la posición del efector al colocar las piezas, de manera que la cámara quede hacia el extremo de la matriz de puntos que ya es una zona fuera del área de trabajo. Esto supondría cambiar las coordenadas almacenadas e introducir una posición en la parte superior de la zona de liberación con el efector en la misma posición que toma para la captura de imágenes.

Un obstáculo más al que se enfrenta el sistema es la variabilidad de las coordenadas del punto de referencia en la imagen, problema ya comentado en la Sección 7.1.3.1, que provoca imprecisiones a la hora de ir a por una pieza rotada. Esta contrariedad se podría resolver de varias maneras, una de ellas es modificar el diseño de la pieza de manera que las tuercas quedaran encajadas impidiendo así su rotación. La otra posibilidad planteada, que no es incompatible con la anterior, sería realizar un proceso iterativo, que se podría combinar con el problema de la automatización. En este caso, la señal que debería enviar el robot contendría la posición del mismo en ese preciso instante, información obtenida con la instrucción *CRobT* y que permitiría que MATLAB calculara en ese momento la información de la siguiente pieza. Esto reduciría la incertidumbre del conjunto y lo haría más resistente a posibles imprevistos.

El siguiente desarrollo lógico de este proyecto consistiría en poder coger piezas apiladas que estuvieran rotadas, para lo cual sería preciso refinar el sistema de reconocimiento de alturas, acotar los rangos para las diferentes alturas o diseñar un sistema que evalúe de manera conjunta el ángulo de la pieza y la altura. De forma se añadirían tres milímetros a la coordenada Z designada para dicha altura, si la pieza determinara un ángulo de rotación concreto, aunque esto supondría mejorar el reconocimiento de piezas apiladas del mismo color de forma que no haya confusiones con el lateral. Una vez obtenida la altura a la que se debe aproximar el autómata, debería enganchar la hilera en una posición cercana y desde ahí proceder como si hubiera existido en esas condiciones desde el primer momento.
Para mejorar el reconocimiento de la rotación de la pieza, se podría procesar la imagen de color de forma que el algoritmo de detección de bordes reconozca los enganches de la pieza y trabaje con su posición relativa.

El último desarrollo propuesto en este capítulo consistiría en la mejora del método de extracción de piezas apiladas, de forma que las piezas inferiores quedaran perfectamente encajadas en la matriz del fondo.

A

Diseño pieza de impresión

En este anexo se detallan las características de la pieza diseñada para la sujección de la cámara al robot.

El diseño de las piezas para la sujección de la cámara al robot se ha llevado al cabo empleando el *software Solid Edge*. El diseño consta de dos piezas, una de ellas se acoplará al robot en los puntos mostrados en la figura mientras que en la segunda se introducirá la cámara. Este diseño de dos piezas se debe a la imposibilidad de ajustar simultáneamente los tornillos que acoplen la pieza al robot y a la cámara. Para unir las piezas entre sí se emplearán dos tornillos con sus correspondientes tuercas.

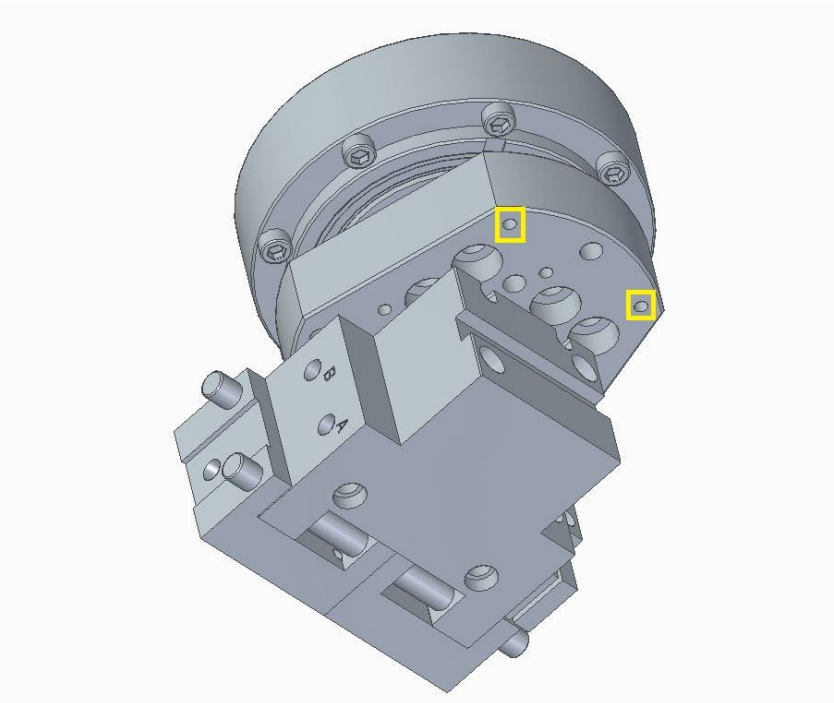


Figura A.1. Puntos donde se acopla la pieza al robot

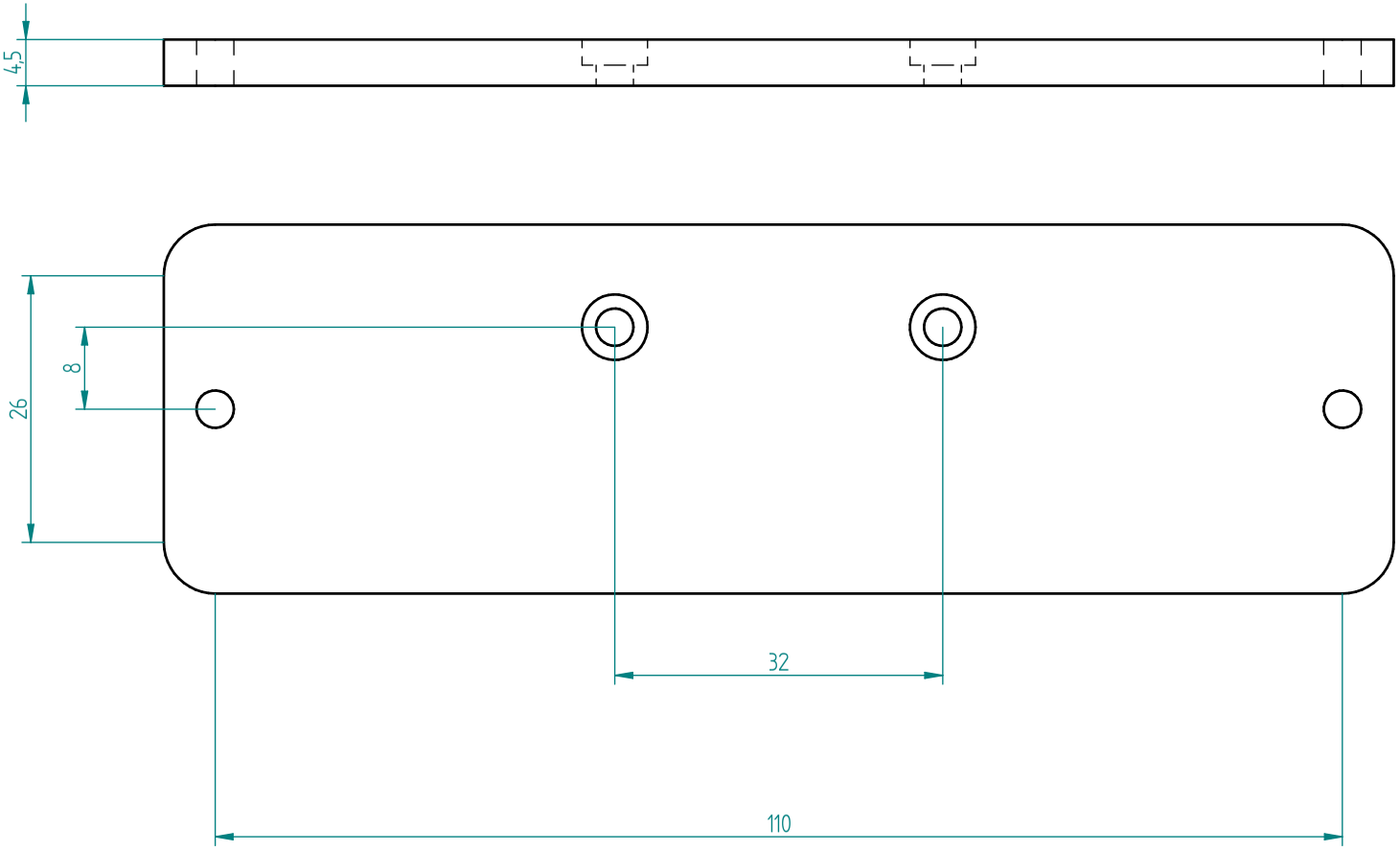
La primera pieza, cuyos planos se observan en la Figura A es una placa de cinco milímetros de grosor con abocardados para los dos tornillos de métrica M3, que sujetarán la pieza al robot y que se encuentran desviados del centro de la pieza, ya que de otro modo esta no se podría colocar al chocar con la parte que sujeta el efector final.

La segunda pieza, mostrada en la Figura A, sujetará la cámara empleando dos tornillos de métrica M3 y un tornillo lateral de métrica imperial. Aunque el objetivo principal de esta pieza es el de sujetar la cámara, también se diseñó con la finalidad de protegerla frente a golpes permitiendo la ventilación. Con estas consideraciones se realizaron los agujeros laterales haciendo que la cámara quedara totalmente cubierta por la pieza. Para permitir la conexión de la cámara con el ordenador, se dejó un hueco en la parte izquierda de la pieza, de forma que el cable USB-C pudiera conectarse.

La unión entre ambas piezas se realiza con dos tornillos de métrica M3 ubicados en los extremos de la pieza, con abocardados en la parte de la segunda pieza para que la cabeza del tornillo no choque con el cable. Las tuercas se colocarán por la parte superior.

Como mejoras a la pieza diseñada, se plantea la posibilidad de hacer también hueco para el tornillo lateral y el cable en los lados opuestos de la pieza, de manera que se pueda colocar la cámara en cualquier orientación dentro de la pieza y, por ende, en el robot. Por otra parte también se debería considerar la realización de un abocardado para las tuercas en la parte superior de la pieza uno, de manera que se asegure que no se van a aflojar con el movimiento del robot.

Revisiones			
Rev	Descripción	Fecha	Aprobado



	Nombre	Fecha
Dibujado	Ana Berjón	5/07/19
Comprobado		
Aprobado 1		
Aprobado 2		

Salvo indicación contraria
cotas en milímetros
ángulos en grados
tolerancias ±0,5 y ±1°

Solid Edge
Siemens PLM

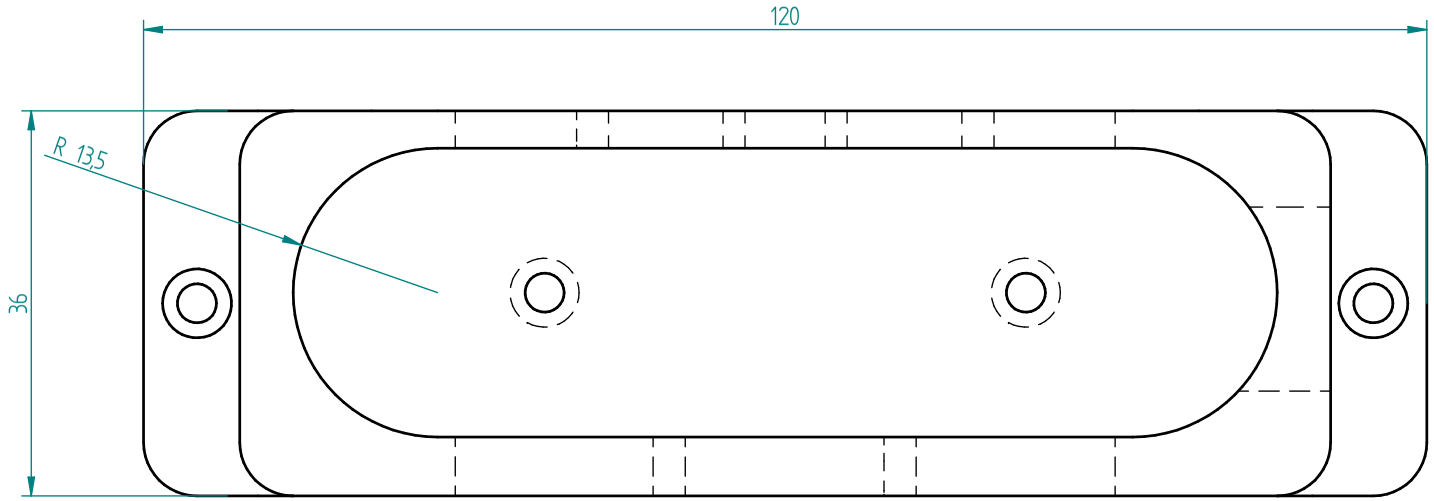
Título: Unión al robot

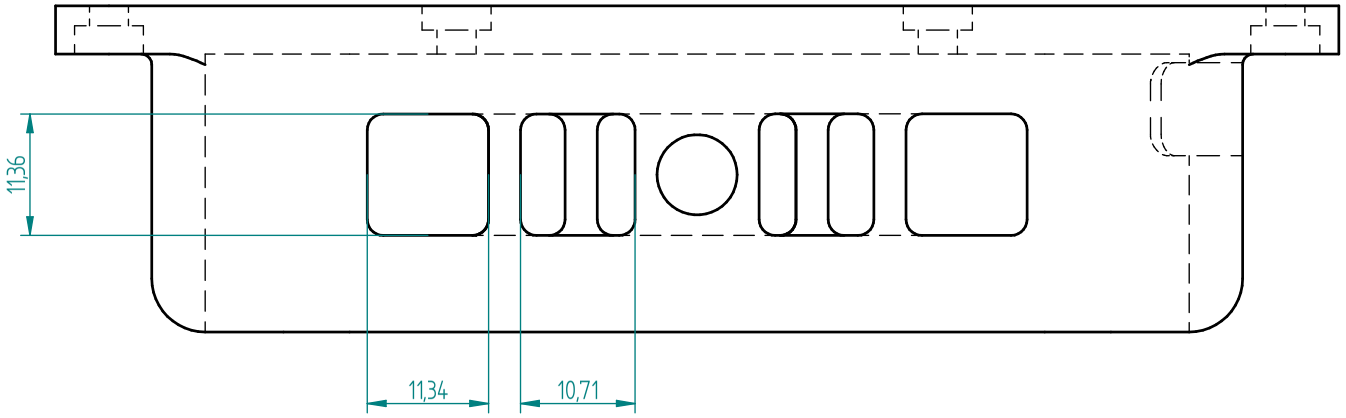
A3	Plano	Rev

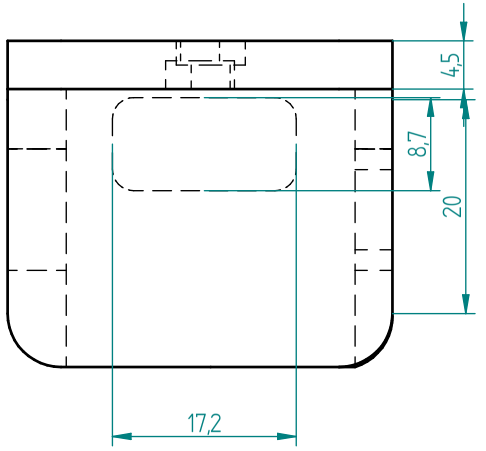
Archivo: pieza1.dft

Escala	Peso	Hoja 1 de 1

Revisiones			
Rev	Descripción	Fecha	Aprobado







	Nombre	Fecha
Dibujado	Ana Berjón	5/07/19
Comprobado		
Aprobado 1		
Aprobado 2		

Salvo indicación contraria
cotas en milímetros
ángulos en grados
tolerancias ±0,5 y ±1°

Solid Edge
Siemens PLM

Título: Soporte cámara

A3	Plano	Rev

Archivo: pieza2.dft

Escala	Peso	Hoja 1 de 1

B

Corrección de coordenadas

En este anexo se presenta la información tomada para poder llevar a cabo la corrección del cálculo de la posición de las piezas encajadas.

A la hora de calcular la posición de las piezas encajadas, se buscó una proporción de píxeles por hueco. Sin embargo, se pudo comprobar que esta relación no era lineal a lo largo de la imagen, debido a la distorsión radial.º Se procedió a calibrar la cámara empleando el calibrador de cámaras de MATLAB y la distorsión disminuyó considerablemente, pero se pudo comprobar que el sistema aun no era perfecto y llegados a este punto se plantearon dos alternativas: establecer un proceso iterativo, más fiable pero costoso ya que implicaba habilitar la comunicación robot-MATLAB o realizar un ajuste manual.

Debido a la complejidad de la primera opción se optó por la segunda, para lo que se diseñó un programa que iteraba todas las posiciones posibles de la matriz, una a una para evitar posibles equivocaciones y mientras se iba desplazando manualmente la pieza. Este proceso se repitió cinco veces, una por cada altura considerada[1] y el programa almacenaba, para cada posición, los siguientes datos:

- Una matriz de celdas donde se almacena la posición correcta que debería calcular, la que calcula y las coordenadas de la pieza en cuestión.
- El programa analiza si la posición es viable, en caso de no ser así, la posición se señala con el código “-100”. A continuación el sistema comprueba si hay alguna pieza en la imagen. En todas las situaciones está colocada, pero si la torre es muy alta, desde la posición de la cámara se ve ladeada y en ese caso se sale de la imagen. Si no encuentra una pieza, inserta el código “100”. La leyenda de los colores se encuentra en la imagen Figura B.1. Finalmente, si encuentra una pieza:
    - Almacena la diferencia entre la posición real y la calculada para la coordenada X
    - Almacena la misma información para la coordenada Y

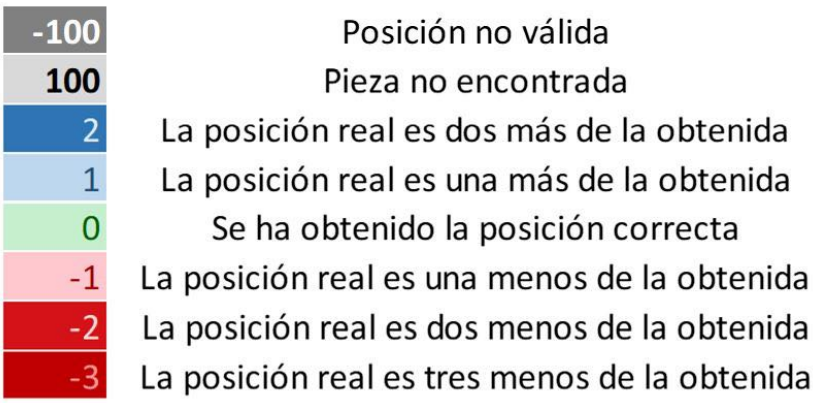
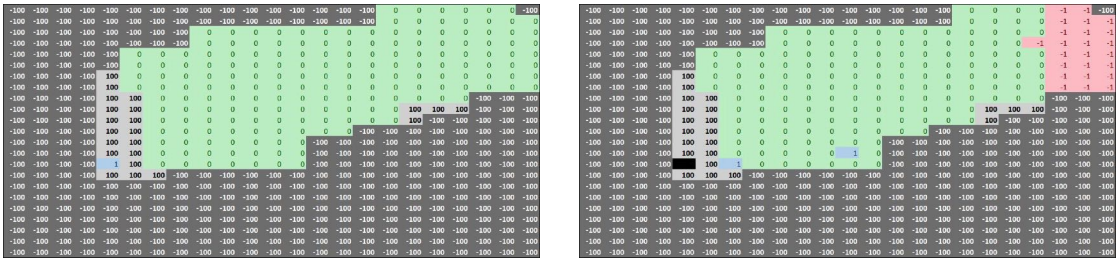


Figura B.1. Leyenda de los diagramas de posición a las distintas alturas.

A continuación, se muestra y analiza la información de las matrices para X e Y a las distintas alturas.

En la Figura B.2 se observa cómo la coordenada X siempre se calcula correctamente, mientras que en la coordenada Y ya empieza a fallar. Este fallo se entiende que es debido a la distorsión radial de la imagen, pero su corrección es sencilla ya que el punto en el que comienza a fallar es a partir de la columna veintiuno y en las coordenadas de las piezas se aprecia fácilmente el corte.



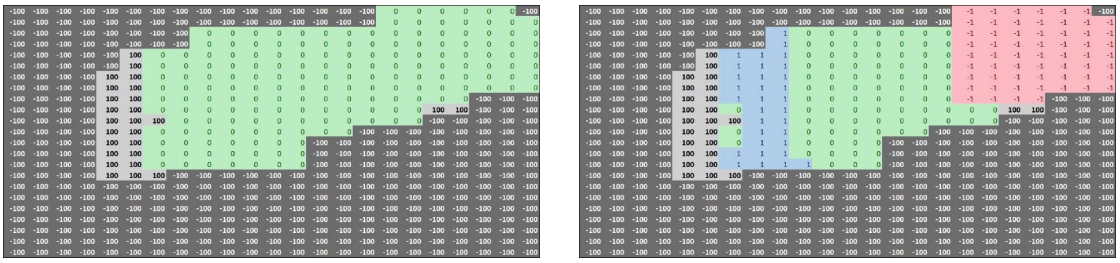
(a) **(b)**

Figura B.2. Información de las piezas encajadas a altura uno.
(a) Información de la coordenada X y (b) de la coordenada Y.

A dos alturas, como se muestra en la Figura B.3, mientras que la coordenada X sigue acertando en todo el espacio de trabajo, en la coordenada Y ya se empiezan a observar errores más significativos. Empieza a fallar por ambos lados, aunque sigue siendo clara la coordenada en la que comienza a fallar y, por consiguiente, corregirlo.

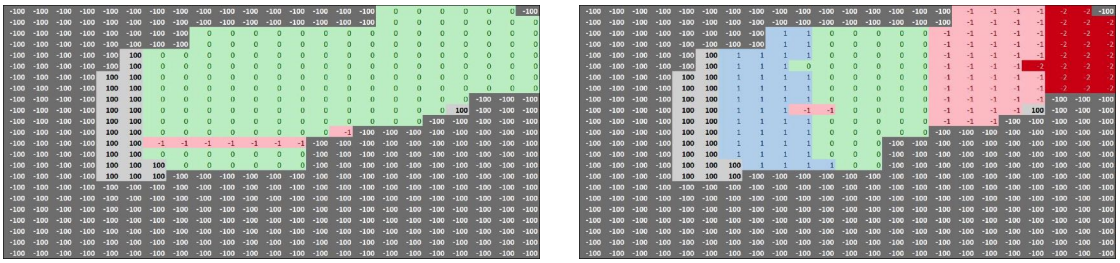
Al introducir una tercera altura, la coordenada Y ya presenta variaciones mayores. Desde el extremo izquierdo, donde se señala una posición menos de la que realmente es, hasta el derecho donde devuelve una posición dos unidades superior. La delimitación de fallo sigue siendo evidente, por lo que su modificación sigue siendo sencilla.
Por otra parte, la coordenada X acierta en todas las posiciones salvo en la fila decimotercera, donde se devuelve una posición menos de la que debería. Sin embargo, dado que las filas sucesivas las devuelve correctamente, se ha interpretado esto como un error humano a la hora de colocar la pieza para su análisis y por ello no se lleva a cabo ninguna corrección asociada.

[1]El sistema inicialmente se había diseñado con intención de poder apilar hasta cinco piezas.



(a) **(b)**

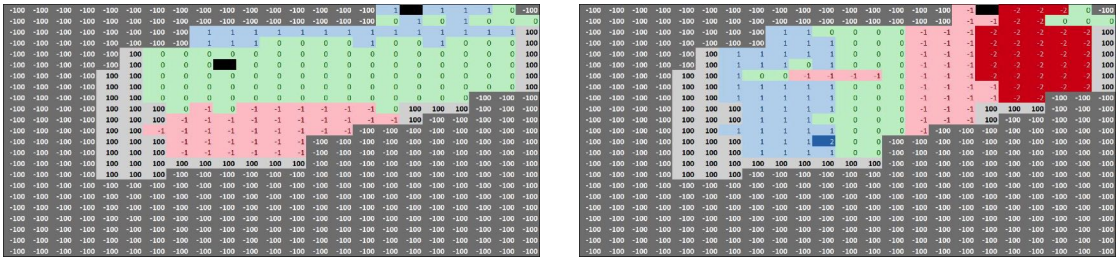
Figura B.3. Información de las piezas encajadas a altura dos.
(a) Información de la coordenada X y (b) de la coordenada Y.



(a) **(b)**

Figura B.4. Información de las piezas encajadas a altura tres.
(a) Información coordenada X y (b) de la coordenada Y.

A partir de la cuarta altura, las líneas de fallo no están tan delimitadas. En esto se considera que influye la distorsión radial de la cámara y el hecho de que, al ver una hilera de una determinada altura a una cierta distancia, su centro parece desplazarse, lo que contribuye a este tipo de fallos. Por otra parte, también se puede observar que, desde esta altura, al colocar las piezas en las posiciones límite se salen del campo de visión de la cámara y dejan de ser vistas (vigésimo tercera columna y décimo séptima fila) a pesar de que esas ubicaciones sí son validas a otras alturas.



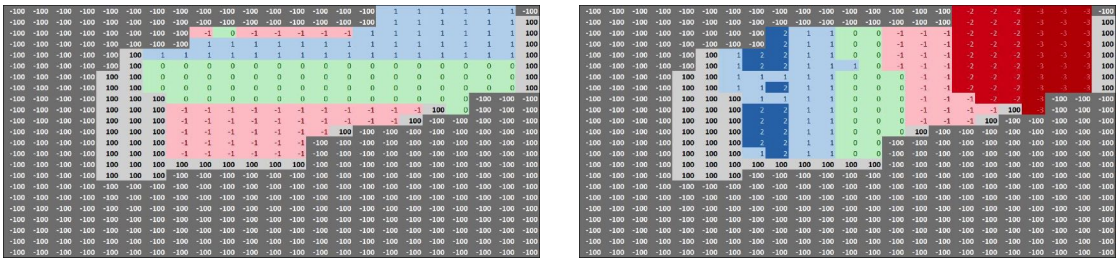
(a) **(b)**

Figura B.5. Información de las piezas encajadas a altura cuatro.
(a) Información de la coordenada X y (b) de la coordenada Y.

En la Figura B.5 la coordenada X presenta fallos similares a los de la coordenada Y de la segunda altura, y su rectificación sería sencilla, pero si se analiza detenidamente la información devuelta en el diagrama de la coordenada Y ya se aprecia que los fallos no siguen una lógica clara: en la séptima fila se rompe el patrón que seguían el resto en cuanto al eje Y, en las posiciones más al extremo acierta posiciones (cuando debería presentar los mayores fallos) y la división entre fallar en dos unidades o solo en una no está claramente definida. Por estos

motivos se decidió limitar la altura a tres, a pesar de que la implementación está realizada para un máximo de cinco alturas. En cualquier caso se podría trabajar con una cuarta altura siempre que esta se ubicara en el centro de la matriz, ya que de esta forma la localización se llevaría a cabo correctamente.

Finalmente se tomó la información de la quinta altura, mostrada en la Figura B.6, donde la información obtenida es totalmente caótica, especialmente en la coordenada Y. En esta coordenada se puede observar cómo pasa de señalar dos posiciones menos (y no en las posiciones del extremo de la imagen) a indicar hasta tres puestos más. Por esta razón, si bien con cuatro alturas todavía podría realizarse una puesta en funcionamiento en un número determinado de posiciones, con cinco este número se vería reducido, mínimo, a la mitad.



(a) **(b)**

Figura B.6. Información de las piezas encajadas a altura cinco.
(a) Información coordenada X y (b) de la coordenada Y.

Bibliografía

[Age19] C. I. A. (I. Agency, *The World Factbook*, feb. de 2019. dirección: https://www.cia.gov/library/publications/the-worldfactbook/geos/ee.html.

[AMS18] I. Ahmad, I. Moon y S. J. Shin, «Color-to-grayscale algorithms effect on edge detection — A comparative study», en *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, ene. de 2018, págs. 1-4.

[ARM17] L. A. Aranda, P. Reviriego y J. A. Maestro, «Error Detection Technique for a Median Filter», *IEEE Transactions on Nuclear Science*, vol. 64, n.º 8, págs. 2219-2226, ago. de 2017.

[Arr18] M. A. Arroyo Lazo, «The Fourth Industrial Revolution.», *Economía*, vol. 41, n.º 81, págs. 194-197, 2018. dirección: http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=133230470&authtype=shib&lang=es&site=ehost-live&scope=site&authtype=ip,shib.

[BI14] A. Borji y L. Itti, «Human vs. Computer in Scene and Object Recognition», en *2014 IEEE Conference on Computer Vision and Pattern Recognition*, jun. de 2014, págs. 113-120.

[Bol18] M. Boleslav, *Skoda Auto uses fully autonomous transport robot ar Vrchlabi plant*, jun. de 2018. dirección: https://www.skoda-storyboard.com/en/press-releases/skoda-auto-uses-fully-autonomous-transport-robot-at-vrchlabi-plant/.

[Bry14] A. Brynjolfsson Erik McAfee, *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*, I. W. W. Norton & Company, ed. W. W. Norton & Company, Inc., 2014.

[Buc+13] D. Buchholz, M. Futterlieb, S. Winkelbach y F. M. Wahl, «Efficient bin-picking and grasp planning based on depth data», en *2013 IEEE International Conference on Robotics and Automation*, mayo de 2013, págs. 3245-3250.

[BWG18] K. Bothe, A. Winkler y L. Goldhahn, «Effective Use of Lightweight Robots in Human-Robot Workstations with Monitoring Via RGBD-Camera», en *2018 23rd International Conference on Methods Models in Automation Robotics (MMAR)*, ago. de 2018, págs. 698-702.

[CC09] A. B. Carlson y P. B. Crilly, *Communication Systems*. McGraw-Hill Education, 2009. dirección: https://www.amazon.com/Communication-Systems-Professor-Electrical-Engineering/dp/0073380407?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0073380407.

[CC10] X. Chen y H. Chen, «A novel color edge detection algorithm in RGB color space», en *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*, oct. de 2010, págs. 793-796.

[CDG15] G. N. Chaple, R. D. Daruwala y M. S. Gofane, «Comparisions of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA», en *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, feb. de 2015, págs. 1-4.

[Cho+18] S. Cho, J. P. Jun, H. Jeong y H. I. Son, «Design of a 4-Finger End-Effector for paprika harvesting», en *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, oct. de 2018, págs. 255-257.

[dEst12] O. I. de Estandarización (ISO), «Robots and Robotic devices (ISO 8373:2012)», Organización Internacional de Estandarización (ISO), Vernier, Suiza, inf. téc., 2012.

[FB81] M. A. Fischler y R. C. Bolles, «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography», *Communications of the ACM*, vol. 24, n.º 6, págs. 381-395, 1981.

[Gar18] L. D. García, «Integración de un sistema de visión artificail en un robot industrial», TFG, Universidad Pontificia Comillas (ICAI-ICADE), jul. de 2018.

[Hub18] H. Huber, *Connected, flexible, autonomous: BMW Group expands use of innovative technologies in production logistics*, nov. de 2018. dirección: https://www.press.bmwgroup.com/global/article/detail/T0287775EN/connected-flexible-autonomous:-bmw-group-expands-use-of-innovative-techno.

[HYT98] Z. Y. Hu, Y. Yang y H. T. Tsui, «In defense of the Hough transform», en *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170)*, vol. 1, ago. de 1998, 24-26 vol.1.

[IKW09] R. Iser, D. Kubus y F. M. Wahl, «An efficient parallel approach to Random Sample Matching (pRANSAM)», en *2009 IEEE International Conference on Robotics and Automation*, mayo de 2009, págs. 1199-1206.

[Jai+14] A. Jain, M. Gupta, S. N. Tazi y Deepika, «Comparison of edge detectors», en *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)*, nov. de 2014, págs. 289-294.

[Jia+08] Z. Jiankui, H. Zishu, M. Sellathurai y L. Hongming, «Modified Hough Transform for Searching Radar Detection», *IEEE Geoscience and Remote Sensing Letters*, vol. 5, n.º 4, págs. 683-686, oct. de 2008.

[JK12] S. R. Joshi y R. Koju, «Study and comparison of edge detection algorithms», en *2012 Third Asian Himalayas International Conference on Internet*, nov. de 2012, págs. 1-5.

[Kim+13] K. R. Kim, H. S. Song, H. S. Yim y S. H. Jung, «Development of the multifunctional end-effector for interventional surgery robot», en *2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)*, oct. de 2013, págs. 86-91.

[KPK14] M. Kim, J. Park e I. S. Kweon, «RGBD sensor and mirrors: A practical setup for 3D reconstruction of dynamic objects», en *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, nov. de 2014, págs. 651-652.

[LD17] O.-K. L. C. M.-D. L̃. N. A. P. Laurent Probst Bertrand Pedersen y M. R. Demetrius Klitou Johannes Conrads, *Digital transformation scoreboard 2017: Evidence of positive oucomes and current opportunities for EU businesses*, European Comission, ene. de 2017.

[LZ07] Y. Li y S. Zhu, «Modified Hough Transforms for Iris Location», en *2007 IEEE International Conference on Control and Automation*, mayo de 2007, págs. 2630-2632.

[Mar+94] L. Marques, F. Moita, U. Nunes y A. de Almeida, «3D Laser-based Sensor For Robotics», en *Proceedings of MELECON '94. Mediterranean Electrotechnical Conference*, vol. 3, IEEE, abr. de 1994, págs. 1328-1348.

[NB16] P. C. Niedfeldt y R. W. Beard, «Convergence and Complexity Analysis of Recursive-RANSAC: A New Multiple Target Tracking Algorithm», *IEEE Transactions on Automatic Control*, vol. 61, n.º 2, págs. 456-461, feb. de 2016.

[NPJ08] T. T. Nguyen, X. D. Pham y J. W. Jeon, «An improvement of the Standard Hough Transform to detect line segments», en *2008 IEEE International Conference on Industrial Technology*, IEEE, abr. de 2008, págs. 1-6.

[NS87] C. L. Novak y S. A. Shafer, «Color edge detection», en *Proc. DARPA Image Understanding Workshop*, vol. 1, 1987, págs. 35-37.

[Pos81a] J. Postel, «Internet Protocol», RFC Editor, STD 5, sep. de 1981, http://www.rfc-editor.org/rfc/rfc791.txt. dirección: http://www.rfc-editor.org/rfc/rfc791.txt.

[Pos81b] ——, «Transmission Control Protocol», RFC Editor, STD 7, sep. de 1981, http://www.rfc-editor.org/rfc/rfc793.txt. dirección: http://www.rfc-editor.org/rfc/rfc793.txt.

[Pra+17] I. Pratikakis, K. Zagoris, G. Barlas y B. Gatos, «ICDAR2017 Competition on Document Image Binarization (DIBCO 2017)», en *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, nov. de 2017, págs. 1395-1403.

[Rob61] L. G. Roberts, «Machine Perception of Three-Dimensional Solids», Tesis doct., Massachusetts Institute of Techcology, 1961.

[Shr13] N. J. Shrikrishna, *Industrial robotics: end effectors*, sep. de 2013. dirección: https://nptel.ac.in/courses/112103174/module7/lec6/2.html.

[SLT11] B. Su, S. Lu y C. L. Tan, «Combination of Document Image Binarization Techniques», en *2011 International Conference on Document Analysis and Recognition*, sep. de 2011, págs. 22-26.

[Tan17] S. Taner, *Audi media center*, mar. de 2017. dirección: https://www.audimediacenter.com/en/press-releases/human-robot-cooperation-klara-facilitates-greaterdiversity-of-versions-in-production-at-audi-9179.

[VN16] A. Vysocky y P. Novak, «Human - Robot collaboration in industry», *MM Science Journal*, vol. 2016, págs. 903-906, jun. de 2016.

[YX15] L. Yuan y X. Xu, «Adaptive Image Edge Detection Algorithm Based on Canny Operator», en *2015 4th International Conference on Advanced Information Technology and Sensor Application (AITS)*, ago. de 2015, págs. 28-31.

[Zen+18] A. Zeng, S. Song, K. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morena, P. Qu Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser y A. Rodriguez, «Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching», en *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, págs. 1-8.