COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO
EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

ANÁLISIS DE IMÁGENES RGB-D POR VISIÓN ARTIFICIAL PARA EL AGARRE DE PIEZAS INDUSTRIALES

Autor: Daniel Horcajo de la Cruz

Directores:

Ignacio de Rodrigo Tobías

Álvaro Jesús López López

Madrid

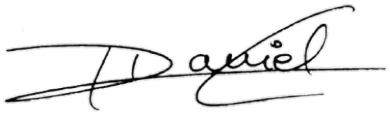
20 de julio de 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Análisis de imágenes RGB-D por Visión Artificial para el agarre de piezas industriales

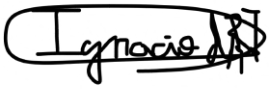
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020-2021 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Daniel Horcajo de la Cruz Fecha: 20/07/2021

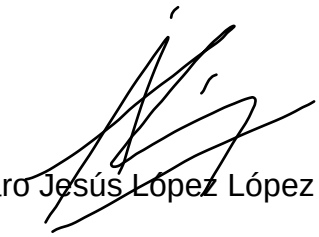


Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: Ignacio de Rodrigo Tobías Fecha: 21 / 07 / 2021



Fdo.: Álvaro Jesús López López Fecha: 20 / 07 / 2021

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO
EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

ANÁLISIS DE IMÁGENES RGB-D POR VISIÓN ARTIFICIAL PARA EL AGARRE DE PIEZAS INDUSTRIALES

Autor: Daniel Horcajo de la Cruz

Directores:

Ignacio de Rodrigo Tobías

Álvaro Jesús López López

Madrid

20 de julio de 2021

ANÁLISIS DE IMÁGENES RGB-D POR VISIÓN ARTIFICIAL PARA EL AGARRE DE PIEZAS INDUSTRIALES

Autor: Horcajo de la Cruz, Daniel.

Directores: Rodrigo Tobías, Ignacio de, y López López, Álvaro Jesús.

Entidad colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

El desarrollo de nuevas tecnologías durante los últimos años ha traído consigo un alto nivel de avance en ámbitos tales como la robótica. Por su parte, el creciente interés en algoritmos cada vez más flexibles y robustos ha propiciado la aparición de nuevos tipos de modelos de inteligencia artificial, incluyendo, entre otras, modelos de detección de objetos. A pesar de estos avances, el uso de brazos robóticos que actualmente se encuentran en centros logísticos y de fabricación sigue viéndose limitado por el tipo de pinza que estos poseen, viéndose así restringida su capacidad de agarre. Este proyecto busca explorar diferentes soluciones a esta problemática, desarrollando un algoritmo que subsane dicha limitación. De esta forma, a partir del uso de cámaras de color y profundidad, en combinación con algoritmos tradicionales y de visión artificial, se pretende dotar a un robot ambidiestro de la capacidad de agarrar cualquiera de los diferentes tipos de piezas existentes en un conjunto de componentes automovilísticos específico. Estos agarres podrán llevarse a cabo a través de una pinza de tipo paralelo o de tipo ventosa, según las características morfológicas y la textura de la pieza que se desee agarrar. De este modo, se consigue ampliar la capacidad de agarre de un brazo robótico, dotándole de flexibilidad a la hora de realizar agarres, sin verse limitado por un tipo particular de pinza.

Introducción

Utilizar un brazo robótico capaz de ofrecer un agarre lo suficientemente robusto para una amplia variedad de objetos es actualmente un gran desafío presente en multitud de ámbitos, desde la logística de comercios online hasta procesos de fabricación. Una de las limitaciones existentes es la dificultad que supone lograr que un único tipo de pinza sea capaz de agarrar

objetos con geometrías y texturas dispares de forma lo suficientemente robusta, lo cual conduce a la necesidad de múltiples tipos de pinzas según su aplicación.

Así pues, este proyecto busca explorar diferentes caminos que permitan solventar esta problemática, lo cual se realizará en colaboración con el Instituto de Investigación Tecnológica de la Universidad Pontificia Comillas (ICAI). Para ello, se busca combinar disciplinas tales como la robótica y la visión artificial de cara a lograr encontrar un algoritmo que permita a un brazo robótico agarrar una pieza cualquiera de un conjunto de componentes automovilísticos, utilizando para ello una única pinza (también llamado *universal picking*). Este proceso será llevado a cabo a través de un conjunto de técnicas que abarcarán desde la visión artificial hasta el uso de imágenes de profundidad.

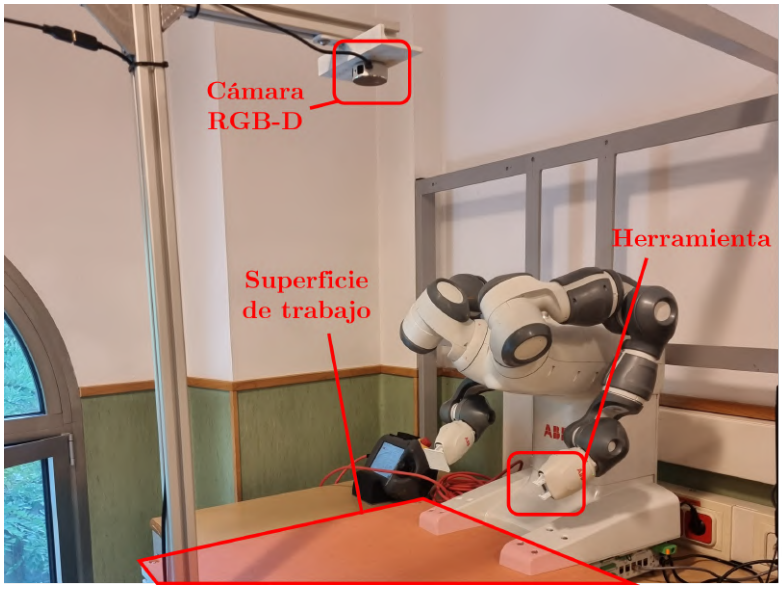


Figura 1. Configuración del robot YuMi del laboratorio en el Instituto de Investigación Tecnológica de ICAI.

Para lograr este objetivo se dispone de la configuración mostrada en la Figura 1. El robot utilizado será un robot YuMi - IRB 14000 de ABB, frente al cual se encuentra una superficie de trabajo en la que se posicionarán los diferentes objetos que se pretenden agarrar. La detección de estos objetos será llevada a cabo a través de una cámara RGB-D capaz de proporcionar tanto una imagen a color (RGB), como un mapa de profundidad (*Depth*) que permitirá medir la distancia desde su sensor hasta cada punto de la escena.

El flujo de información a lo largo del proyecto es el mostrado en la Figura 2. A partir de las imágenes RGB y de profundidad capturadas por la cámara se pretende detectar y extraer la pieza objetivo, sobre la cual se calculará una propuesta de agarre, ya sea mediante una pinza paralela,

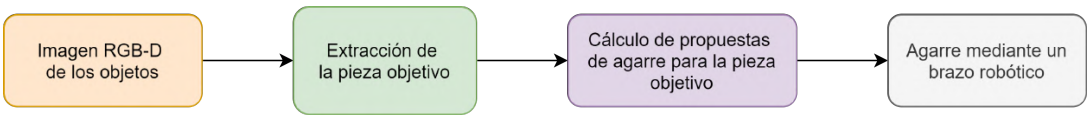


Figura 2. Flujo de información a lo largo del proyecto.

o mediante una ventosa. Esta propuesta de agarre será finalmente enviada al robot YuMi, el cual será responsable de ejecutarla y agarrar la pieza. Asimismo, en la tercera etapa del flujo de información se explorará la posibilidad de utilizar la GQ-CNN de Dex-Net [MMS+19] para calcular un agarre sobre una pieza ya detectada.

El conjunto de piezas que se utilizará en el proyecto es conocido y cerrado, y está conformado por distintos componentes automovilísticos que son normalmente manipulados en un entorno industrial. De esta forma, el contexto de desarrollo del proyecto girará sobre la viabilidad de aplicar la solución encontrada en dicho entorno, lo cual impondrá condiciones de trabajo tales como cambios lumínicos o número limitado de tipos de pieza.

Solución

El flujo de información a lo largo del proyecto mostrado previamente en la Figura 2 posee tres fases diferentes: obtención de la imagen RGB-D a través de una cámara de profundidad, detección y obtención de la pieza objetivo, y cálculo de la propuesta de agarre sobre esta.

Obtención de la imagen RGB-D

Las cámaras RGB-D de las que se dispone son los modelos D435 [Int] y L515 [Intb] de la gama RealSense de Intel. Ambas cámaras deberán pasar por un método de calibración [RM99] que permitirá aproximar sus respectivos parámetros intrínsecos de cara a eliminar los distintos tipos de distorsiones que puedan sufrir sus imágenes. Asimismo, puesto que la primera utiliza visión estereoscópica como método de medición de la profundidad de la escena, frente a la tecnología LiDAR de la segunda, se lleva a cabo un análisis de calidad de la imagen de profundidad ofrecida por cada una de ellas.

Tras dicho análisis, se concluye que la imagen de profundidad proporcionada por el modelo LiDAR L515 es superior a aquella del modelo D435, ofreciendo una precisión real inferior a 3 milímetros frente a los 10 de su competidora. Asimismo, se observa que, a pesar de los filtros de posprocesado utilizados para mejorar la calidad de la imagen de profundidad en ambas cámaras, el modelo LiDAR posee muchas menos fluctuaciones entre tomas que su análogo estereoscópico. Se decide, por tanto, hacer uso del modelo L515.

Extracción de la pieza objetivo

A lo largo del proyecto se exploran diferentes técnicas orientadas a la detección y extracción de objetos en una imagen a través de la librería de Python OpenCV [Ope]: segmentación por profundidad, sustracción de fondo, y detección de objetos mediante redes neuronales convolucionales (YOLOv3) [RDGF16].

El método de segmentación por profundidad es el más sencillo de los tres, y consiste en extraer de la imagen todo aquello cuya distancia a la cámara (situada como se indica en la Figura 1) sea inferior a la distancia al plano de trabajo. La sustracción de fondo, por su parte, compara una imagen donde aparezcan las piezas sobre la mesa de trabajo con una en la que la mesa de trabajo se encuentre vacía. De esta forma, la diferencia entre ambas imágenes proporciona una máscara capaz de localizar lo único cambiante entre estas dos tomas, es decir, las piezas. A pesar de que estos dos métodos ofrecen resultados adecuados en condiciones controladas de laboratorio, ambos poseen grandes limitaciones que hacen que su uso en un entorno industrial no sea posible, como su poca robustez frente a cambios lumínicos o su baja precisión.

Por esta razón, se decide explorar la posibilidad de utilizar algoritmos de detección de objetos como YOLOv3. Para ello, se creará un conjunto de datos propio conformado por las 46 piezas de automóvil disponibles sobre el cual se entrenará un modelo diferente para cada tipo de pieza del conjunto que detecte únicamente este tipo. Dado que el tipo de piezas a detectar será un parámetro conocido en el proceso, este enfoque permite reducir el número de falsos positivos que pueden aparecen, en comparación con utilizar un único modelo capaz de detectar cualquier tipo de pieza.

Las máscaras obtenidas mediante cada uno de estos tres métodos se muestran en la Figura 3. Si bien la calidad de la máscara por sustracción de fondo es altamente detallada en comparación con la segmentación por profundidad, un mínimo cambio del fondo respecto la imagen de referencia original que utiliza el algoritmo es capaz de anular por completo cualquier capacidad de detección. Por otro lado, cabe destacar que YOLO, a pesar de ofrecer una *bounding box* en lugar de una máscara detallada del objeto, posibilita el *detectar* la pieza en sí, y no un cambio en las condiciones de la imagen, como es el caso de los otros dos métodos. Asimismo, de cara a obtener una máscara más detallada a partir de las detecciones de YOLO, se lleva a cabo una segmentación por profundidad sobre el recorte de su *bounding box*.

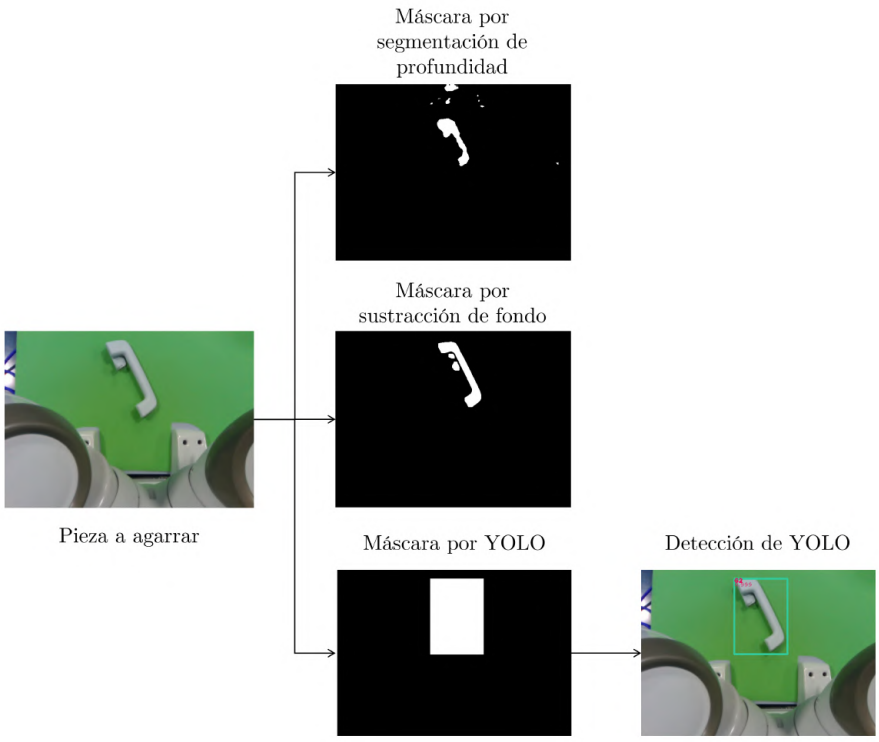


Figura 3. Comparativa de las máscaras resultantes tras la aplicación de los métodos de segmentación por profundidad, sustracción de fondo, y YOLO.

Cálculo de propuestas de agarre

Dado que el robot YuMi posee dos brazos diferentes, se explora la posibilidad de utilizar dos tipos de pinza para agarrar objetos según su morfología y su textura: pinza paralela sobre objetos de tipo paralelepípedo, y ventosa para objetos con superficies planas.

Partiendo de una imagen de profundidad de la escena y una máscara binaria de la pieza objetivo, se explora la posibilidad de implantación de la GQ-CNN de Dex-Net [MMS$^+$19] sobre las piezas del conjunto de datos propio. Sin embargo, como se muestra en la Figura 4, debido a un índice de confianza inferior al 5 % en la mayoría de los agarres propuestos, así como al hecho de que estos agarres son siempre perpendiculares al plano de trabajo y no a la propia pieza, su uso queda descartado y se decide desarrollar un algoritmo propio para cada tipo de herramienta disponible (paralela y ventosa).

El agarre paralelo se realiza de forma perpendicular al plano de trabajo. A partir de un recorte sobre la *bounding box* de YOLO se lleva a cabo una detección de bordes que permite encontrar los contornos de la pieza en cuestión. De esta forma, tomando el centro del recorte y su medida de profundidad como punto de agarre terminal $(x, y, z)_{TP}$, y sabiendo que el objeto es un paralelepípedo, es posible usar los contornos para calcular la orientación θ_{TP} con la cual

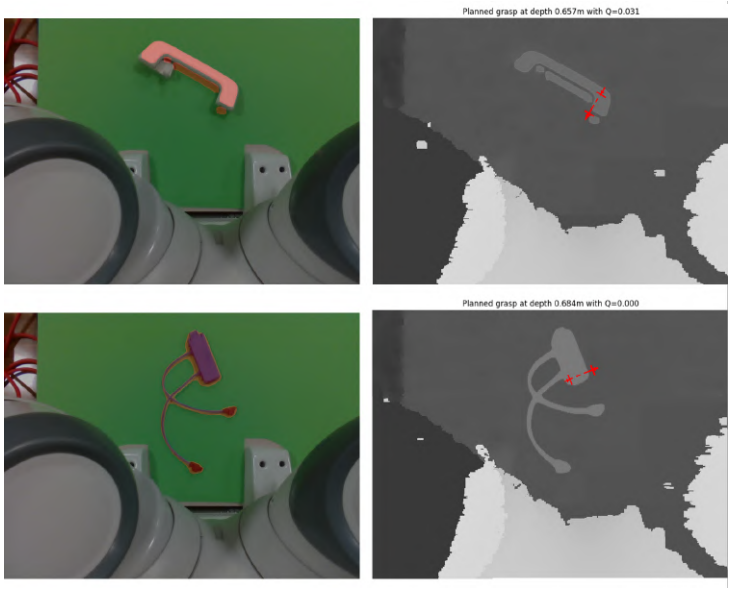


Figura 4. Propuestas de agarre paralelo ofrecidas por Dex-Net (derecha) dadas una máscara del objeto (en rojo, izquierda) y una imagen de profundidad.

deberá cerrarse la pinza. Este proceso queda esquematizado en la Figura 5.

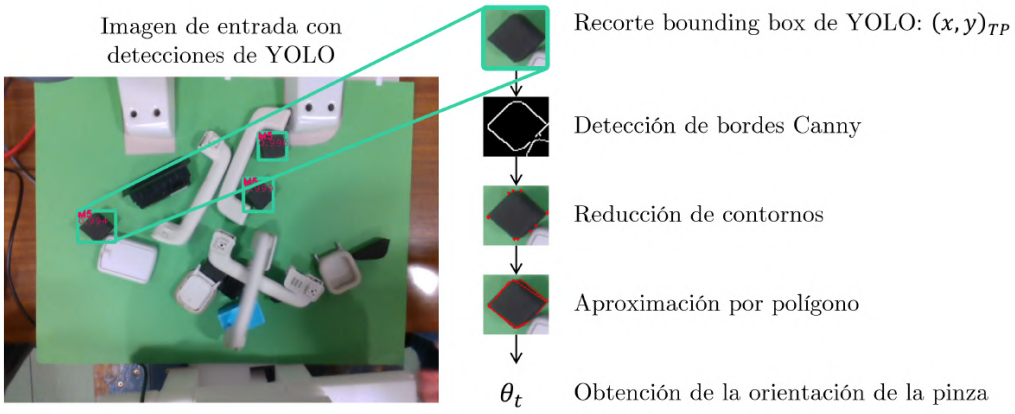


Figura 5. Proceso de cálculo de la orientación del agarre paralelo.

El agarre por ventosa, si bien sigue el mismo principio de funcionamiento que el paralelo, ofrece algunos desafíos añadidos. Al igual que en el agarre paralelo, se parte de un recorte según la *bounding box* de YOLO que, en este caso, se realiza también sobre la imagen de profundidad. Sobre dicho recorte se realiza una segmentación por profundidad que permite obtener una máscara de la pieza objetivo, sobre la cual se puede calcular su centroide, obteniendo así el punto de contacto $(x, y, z)_{TP}$ de la ventosa sobre la superficie superior de la pieza, como se muestra en la Figura 6.

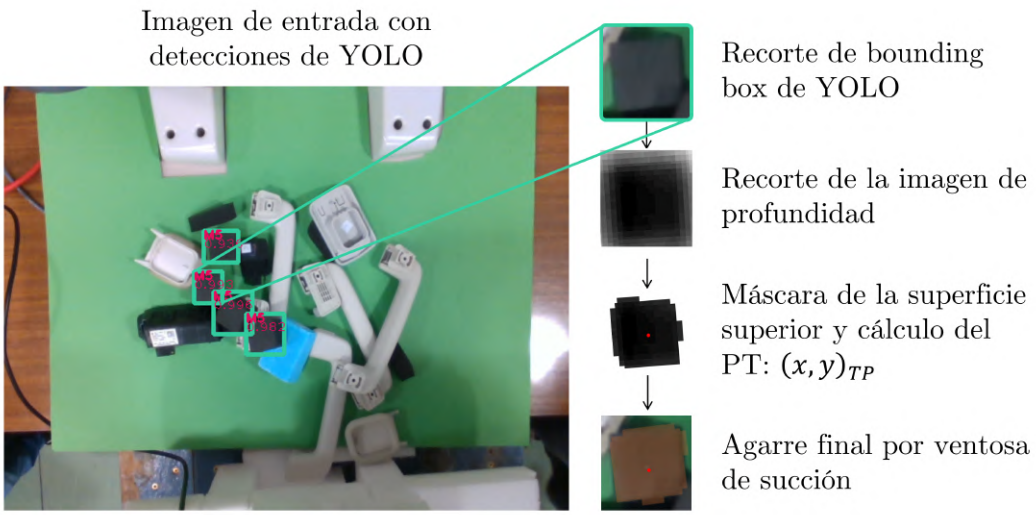


Figura 6. Proceso de cálculo del punto terminal para el agarre con ventosa.

Finalmente, con el objetivo de ofrecer una solución más flexible que permita utilizar la ventosa sobre superficies inclinadas, se lleva a cambo la implantación del algoritmo de RANSAC para la detección de planos inclinados. De esta forma, el cálculo de la orientación del punto terminal del brazo robótico es equivalente al cálculo de la normal al plano de la pieza detectada, como se muestra en la Figura 7.

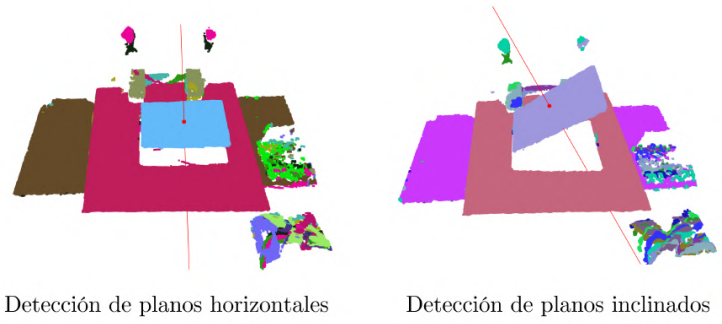


Figura 7. Detección de planos horizontales e inclinados a través del algoritmo RANSAC.

Se dispone, por tanto, de un algoritmo capaz de proponer tanto agarres paralelos, como agarres por ventosa incluso en piezas inclinadas.

Resultados

A lo largo de este proyecto se han analizado diferentes tipos de algoritmos que permiten localizar y extraer una pieza objetivo dadas una imagen RGB y una imagen de profundidad.

Si bien el método de segmentación por profundidad es robusto frente a los en el entorno que pueden tener lugar en un ambiente industrial, la calidad de la máscara proporcionada no era suficiente para facilitar el cálculo de un agarre de calidad. Por otro lado, la sustracción de fondo, a pesar de proporcionar una máscara nítida y de calidad, es una técnica que se ve gravemente afectada por cambios lumínicos y por cambios en la mesa de trabajo, lo cual hace que sufra de un nivel de robustez demasiado bajo en entornos industriales. Asimismo, ambas técnicas tienen su principio de funcionamiento en la detección *fija* de objetos, sin ser capaces de *identificarlos*. Asimismo, se descarta la posibilidad de implantación de la GQ-CNN de Dex-Net en el conjunto de piezas propio que se dispone debido a índices de confianza excesivamente bajos en los agarres calculados, así como a la restricción de los agarres a una orientación normal al plano de trabajo.

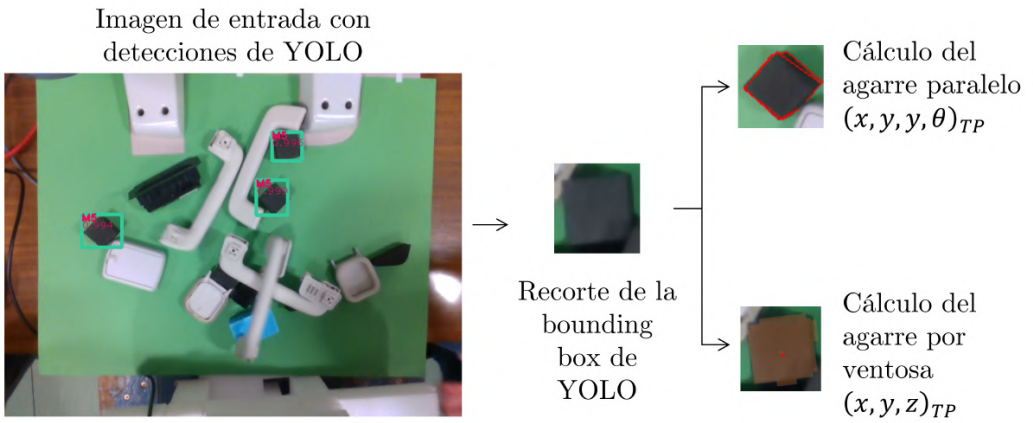


Figura 8. Comparativa de los resultados obtenidos para las diferentes opciones de agarre (paralelo y ventosa).

Este problema es resuelto por YOLO, el cual es capaz de detectar, localizar e identificar la pieza que se desee agarrar, lo cual lo hace mucho más robusto frente a cambios en su entorno en comparación con los dos métodos de cálculo de máscaras anteriores. Así pues, combinando las detecciones de YOLO con los métodos de detección de bordes y de planos (Figura 8), se logra disponer de un algoritmo que posibilita el agarre de cualquier objeto del conjunto de piezas disponibles, ya sea mediante una pinza paralela, o mediante ventosa, incluso en planos no paralelos a la superficie de trabajo.

Conclusiones

Como se mencionó en el apartado anterior, los resultados obtenidos en este proyecto ofrecen la posibilidad de calcular agarres paralelos y por ventosa de forma flexible según la ubicación y el tipo de pieza. Se extraen, por tanto, las siguientes conclusiones:

- Algoritmos tradicionales como la segmentación por profundidad y la sustracción de fondo, aun siendo adecuados en condiciones de laboratorio, no son aptos para su uso en entornos industriales debido a su baja robustez.

- La creación de un conjunto de datos propio *ad-hoc* para un proyecto específico es una labor que requiere gran cantidad de tiempo y recursos, lo cual puede constituir una barrera de entrada para la aplicación de algoritmos de inteligencia artificial.

- El uso de algoritmos de detección de objetos como YOLOv3, una vez superada la barrera del conjunto de datos, ofrece una flexibilidad y robustez que lleva a unos resultados muy satisfactorios, los cuales podrían ser fácilmente extrapolables a un entorno industrial.

- Si bien la detección de planos por RANSAC ofrece buenos resultados en condiciones de laboratorio, su aplicación sobre contenedores de piezas en un entorno industrial, donde estas se encuentran amontonadas y entrelazadas entre sí de forma caótica, puede ser insuficiente.

Referencias

[Inta] Intel, "Intel RealSense D400 Series Product Family Datasheet." [Online]. Available: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet

[Intb] ——, "Intel RealSense LiDAR Camera L515 Datasheet." [Online]. Available: https://dev.intelrealsense.com/docs/lidar-camera-l515-datasheet

[MMS+19] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," Science Robotics, vol. 4, no. 26, 2019, Último acceso: 22/09/2020. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau4984/

[Ope] OpenCV, "OpenCV." [Online]. Available: https://opencv.org/

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” May 2016, arXiv: 1506.02640 Último acceso: 13/11/2020. [Online]. Available: http://arxiv.org/abs/1506.02640

[RM99] A. Romero-Manchado, “"Calibración de cámaras no métricas por el método de las líneas rectas",” Mapping, ISSN 1131-9100, Nº 51, 1999, pags. 74-80, Jan. 1999.

ANALYSIS OF RGB-D IMAGES THROUGH COMPUTER VISION FOR ROBOT GRASPING OF INDUSTRIAL PARTS

Author: Horcajo de la Cruz, Daniel.

Directors: Rodrigo Tobías, Ignacio de, y López López, Álvaro Jesús.

Collaborating entity: ICAI – Universidad Pontificia Comillas.

SUMMARY OF THE PROJECT

The development of new technologies in recent years has brought with it a high level of progress in areas such as robotics. In turn, the growing interest in increasingly flexible and robust algorithms has led to the emergence of new types of artificial intelligence techniques, including, among others, object detection models. Despite these advancements, the use of robotic arms currently found in logistics and manufacturing centers is still limited by the type of gripper they are equipped with, thus restricting their gripping capabilities. This project seeks to explore different solutions to this problem by developing an algorithm to overcome this limitation. Thus, by using color and depth cameras, in combination with traditional and artificial vision algorithms, it is intended to provide an ambidextrous robot with the ability to grasp any of the different types of parts found in a specific set of automotive components. These grips can be carried out by means of a parallel or suction cup type gripper, depending on the morphological characteristics and texture of the part to be gripped. As a result, the gripping capacity of the robotic arm is enhanced, providing it with the flexibility to perform grasping operations without being limited by a particular type of gripper.

Introduction

Using a robotic arm capable of providing a sufficiently robust grip for a wide variety of objects is currently a major challenge present in a multitude of domains, ranging from online retail logistics to manufacturing processes. One of the existing limitations is the difficulty in making a single type of gripper capable of grasping objects with disparate geometries and textures in a sufficiently robust manner, leading to the need for multiple types of grippers

depending on their application.

Thus, this project seeks to explore different ways to solve this problem and will be carried out in collaboration with the Institute of Technological Research of the Universidad Pontificia Comillas (ICAI). To this end, the project seeks to combine disciplines such as robotics and artificial vision to find an algorithm that allows a robotic arm to pick up any part of a set of automotive components, using a single gripper (also known as universal picking). This process will be carried out through a set of techniques ranging from computer vision to the use of depth imaging.

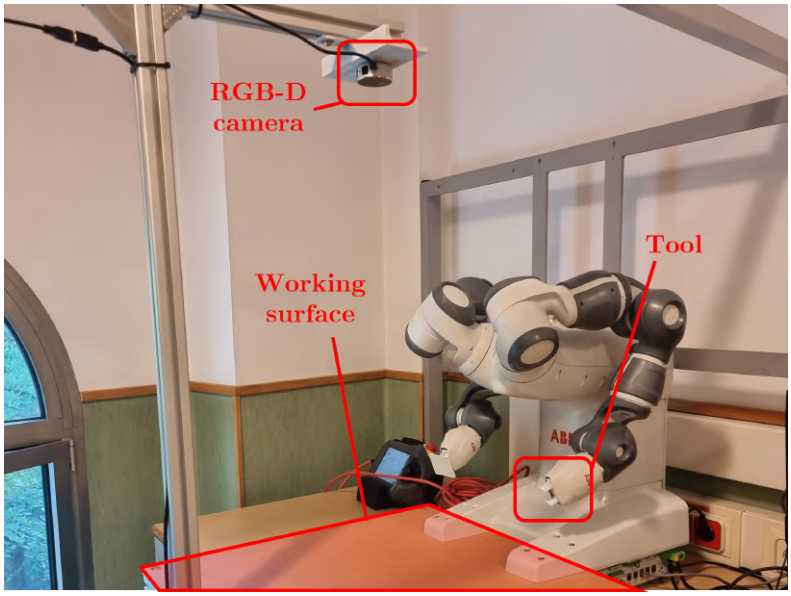


Figure 1. Setup of the YuMi robot in the laboratory at ICAI's Institute for Research in Technology.

To achieve this goal, the configuration shown in Figure 1 is arranged. The robot used will be an ABB YuMi - IRB 14000 robot, in front of which there is a work surface on which the different objects to be grasped will be positioned. The detection of these objects will be carried out through an RGB-D camera capable of providing both a color image (RGB) and a depth map (Depth) that will allow the measurement of the distance from its sensor to each point in the scene.

The flow of information throughout the project is as shown in Figure 2. Using the RGB and depth images captured by the camera, the aim is to detect and extract the target object, on which a gripping proposal will be calculated – either by means of a parallel gripper or by means of a suction cup. This grasping proposal will finally be sent to the YuMi robot, which will be responsible for executing it and grasping the part. It is important to mention that the third stage of the information flow will explore the possibility of using the Dex-Net GQ-CNN [MMS$^+$19]

to compute a grasp on an already detected part.

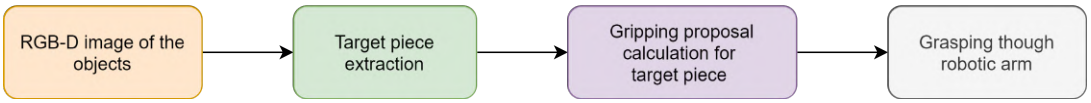


Figure 2. Information flow throughout the project.

The set of parts to be used in the project is known and already fixed, and is made up of different automotive components that are regularly handled in an industrial environment. Hence, the development context of the project will revolve around the feasibility of applying the solution found in such an environment, which will impose working conditions such as light changes or a limited number of part types.

Solution

The flow of information throughout the project previously shown in Figure 2 consists of three different phases: capturing the RGB-D image through a depth camera, detecting and capturing the target object, and calculating the proposed grip on it.

RGB-D image capture

The RGB-D cameras available are the D435 [Int] and L515 [Intb] models from Intel's RealSense line. Both cameras will undergo a calibration method that will make it possible to approximate their respective intrinsic parameters in order to eliminate the different types of distortions that their images may suffer. In addition, given that the former uses stereoscopic vision as a method of measuring the depth of the scene – compared to the LiDAR technology of the latter – a quality analysis of the depth image provided by each of them will be conducted.

Upon completion of this analysis, it is concluded that the depth image provided by the LiDAR L515 model is superior to that of the D435 model, offering an actual accuracy of fewer than 3 millimeters compared to the 10 millimeters of its competitor. It is also observed that, despite the post-processing filters used to improve the quality of the depth image in both cameras, the LiDAR model shows much fewer fluctuations in the depth images produced between shots than its stereoscopic analog. The L515 model is therefore chosen as the main RGB-D camera to be used throughout the project.

Extraction of the target piece

Throughout the project, various techniques oriented to image object detection and extraction are explored using the Python OpenCV library [Ope]: depth segmentation, background subtraction, and object detection using convolutional neural networks (YOLOv3) [RDGF16].

The depth segmentation method is the most straightforward of the three and consists of extracting from the image everything whose distance to the camera (positioned as indicated in Figure 1) is less than the distance to the workbench. Background subtraction, on the other hand, compares an image where the parts appear on the workbench with another image where the workbench is empty. This way, the difference between the two images provides a mask capable of locating the one thing that changes between the two shots – i.e., the workpieces. Although these two methods provide adequate results under controlled laboratory conditions, both have major limitations that make their use in an industrial environment unfeasible – such as low robustness to changes in lighting conditions or low accuracy.

Consequently, it is decided to explore the possibility of using object detection algorithms such as YOLOv3. For this purpose, a custom dataset consisting of the 46 available car components will be created and will then be used to train one specific model for each type of component of the set – thus only being able to detect said type of component. Since the types of components to be detected will be a known parameter beforehand, this approach allows to reduce the number of false positives that may appear, compared to using a single model capable of detecting any type of component.

The masks obtained by each of these three methods are shown in Figure 3. While the quality of the background subtraction mask is highly detailed compared to depth segmentation, the slightest change of the background with respect to the original background image used by the algorithm is capable of completely invalidating any detection capability. On the other hand, it should be noted that YOLO, despite offering a bounding box instead of a detailed mask around the object, makes it possible to detect the part itself, and not a change in the image conditions, as is the case with the other two methods. Additionally, in order to obtain a more detailed mask out of the YOLO detections, depth segmentation is performed on the crop of its bounding box.

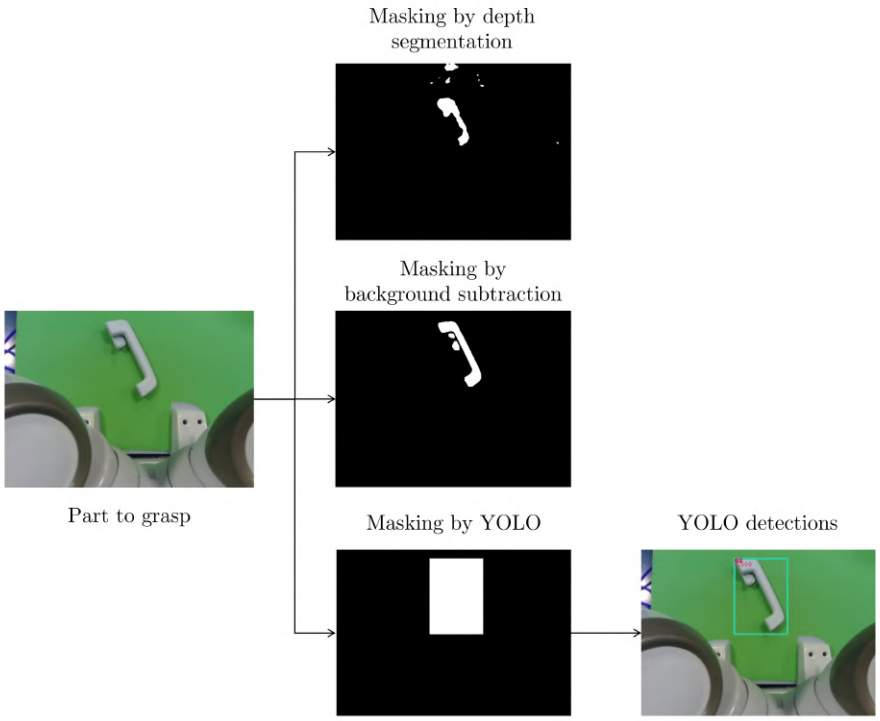


Figure 3. Comparison of the output masks after the implementation of depth segmentation, background subtraction, and YOLO methods.

Calculation of grip proposals

Since the YuMi robot has two different arms, the possibility of using two types of grippers to grasp objects according to their morphology and texture is explored: using a parallel gripper on parallelepiped-type objects, and using a suction cup for objects with flat surfaces.

Given a depth image of the scene and a binary mask of the target part, we explore the possibility of implementing the Dex-Net GQ-CNN [MMS$^+$19] on the parts of the custom dataset. However, as shown in Figure 4, due to an obtained confidence score lower than 5% for most of the proposed grips, as well as the fact that these grips are always perpendicular to the work plane and not to the part itself, its use is discarded and it is decided to develop a custom algorithm for each type of tool available – parallel and suction cup.

Parallel gripping is performed perpendicular to the working surface. Using a cutout from the YOLO bounding box, edge detection is used to find the contours of the relevant workpiece. This way, by taking the center of the cutout and its depth measurement as the terminal grip point $(x, y, z)_{TP}$, and knowing that the object is a parallelepiped, it is possible to use the contours to calculate the orientation θ_{TP} with which the gripper should be closed. This process is

schematized in Figure 5.

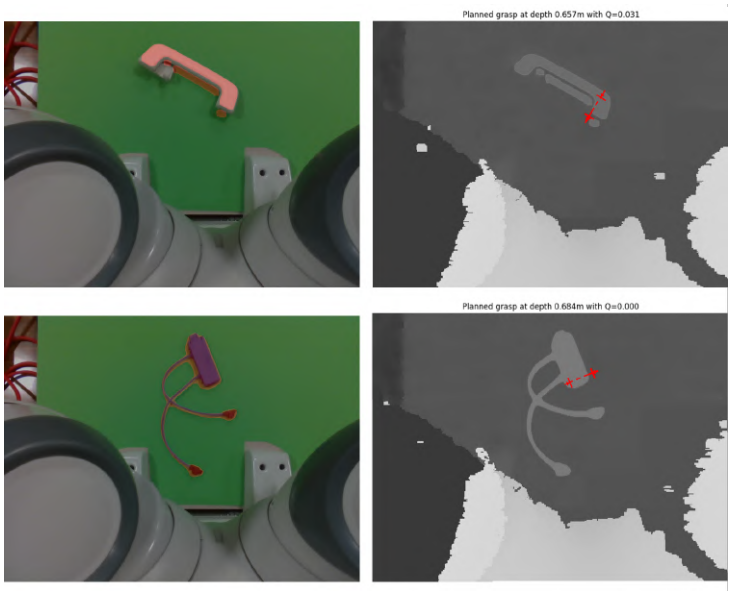


Figure 4. Parallel grip proposals offered by Dex-Net (right) given a mask of the object (in red, left) and a depth image.

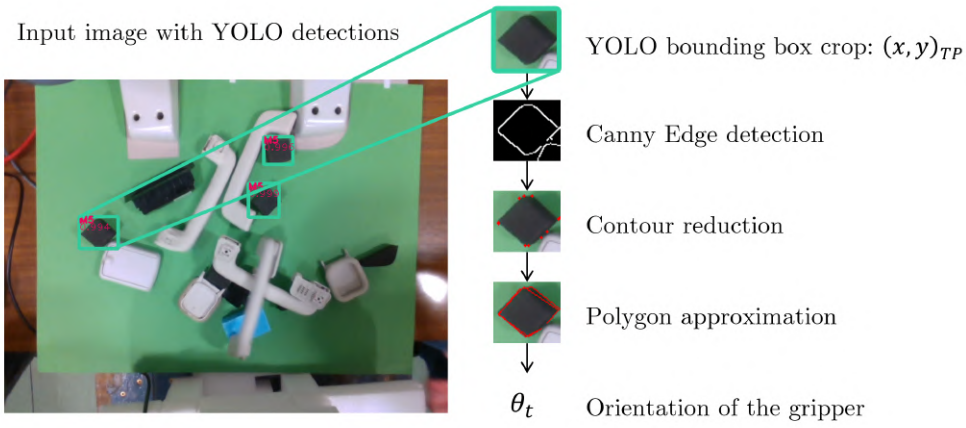


Figure 5. Calculation process of the orientation of the parallel grip.

Suction cup gripping, while following the same operating principle as parallel gripping, offers some additional challenges. As in parallel gripping, the starting point is a cutout according to YOLO's bounding box – however, the cropping is also performed on the depth image in this case. Depth segmentation is then performed on this crop so as to obtain a mask of the target piece, whose centroid can be calculated, thus obtaining the point of contact $(x, y, z)_{TP}$ of the suction cup on the upper surface of the piece, as shown in Figure 6.

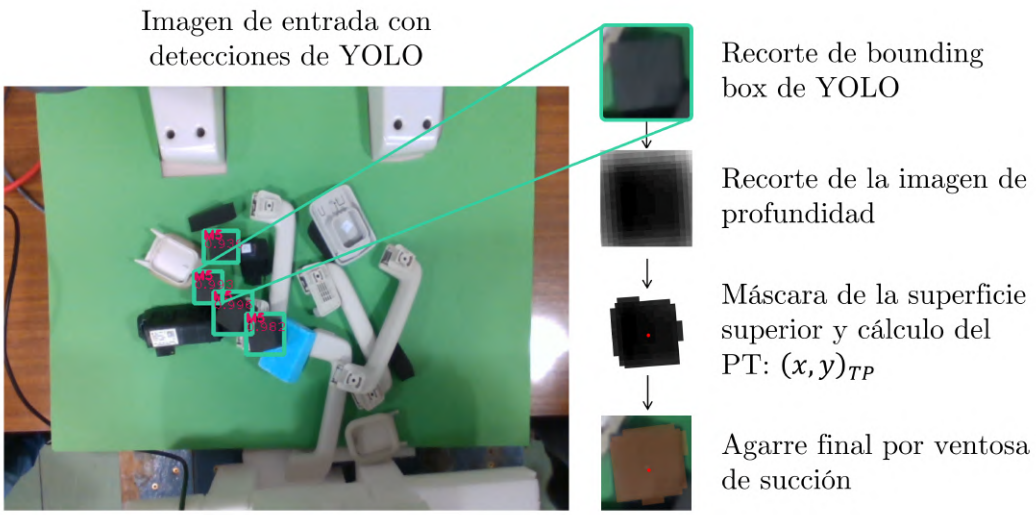


Figure 6. Calculation of the terminal point for the suction cup grip.

Lastly, in order to offer a more flexible solution that allows the use of the suction cup on tilted surfaces, the implementation of the RANSAC algorithm for the detection of oblique planes is carried out. In this way, the calculation of the orientation of the terminal point of the robotic arm is equivalent to that of the normal vector to the plane of the detected part, as shown in Figure 7.

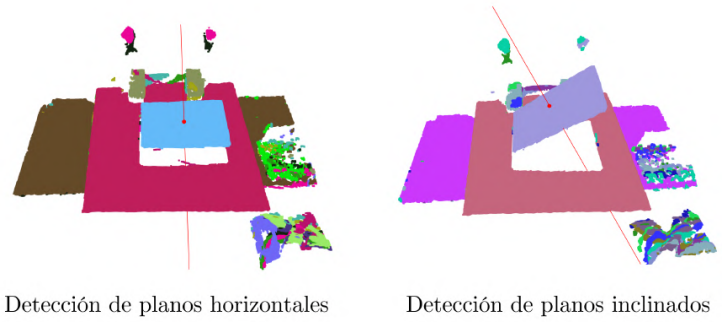


Figure 7. Horizontal and tilted plane detection through the RANSAC algorithm.

As a result, an algorithm capable of proposing both parallel grips and suction cup grips – even on angled pieces – is obtained.

Results

Throughout this project, different types of algorithms that allow to detect and extract a target piece given an RGB image and a depth image have been analyzed.

While the depth segmentation method is robust to the environmental changes that may occur in an industrial environment, the quality of the mask provided was not sufficient to allow for the calculation of a quality grasp. Depth subtraction, while providing a clear and quality mask, is a technique that is severely affected by both light fluctuations and changes in the working table, causing it to suffer from a level of robustness that is far too low for industrial environments. Furthermore, both techniques are based on the *fixed detection* of objects, without being able to actually *identify* them. In addition, the possibility of implementing Dex-Net's GQ-CNN on the custom dataset is ruled out due to excessively low confidence scores on the calculated grips, as well as the restriction of the grips to an orientation normal to the working plane.

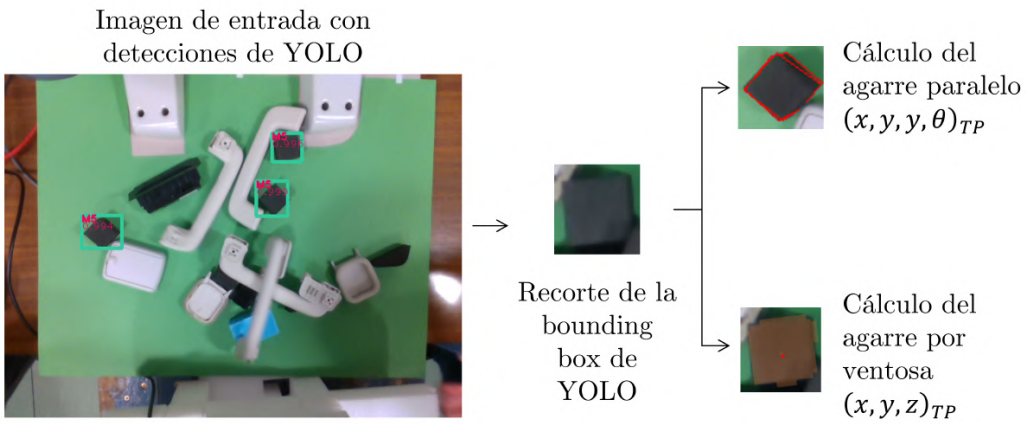


Figure 8. Comparison of the results obtained using the different gripping options (parallel and suction cup).

This problem is overcome by using the YOLO algorithm, which is able to detect, locate and identify the part to be grasped – making it much more robust to changes in its environment compared to the two previous masking methods. Hence, by combining the YOLO detections with edge and plane detection techniques (Figure 8), an algorithm capable of gripping any object from the available set of parts – either by means of a parallel gripper or a suction cup – is achieved, even for planes not parallel to the working surface.

Conclusions

As mentioned in the previous section, the results obtained in this project offer the possibility to calculate parallel and suction cup grips in a flexible way depending on the location and type of the part. Thus, the following conclusions can be drawn:

- Traditional algorithms such as depth segmentation and background subtraction, although adequate in laboratory conditions, are not suitable for use in industrial environments due to their low robustness.
- The creation of a proprietary *ad-hoc* dataset for a specific project is a time-consuming and resource-intensive task, which can present a barrier to entry for the application of artificial intelligence algorithms.
- The use of object detection algorithms such as YOLOv3, once the dataset barrier is overcome, offers a level of flexibility and robustness that leads to very satisfactory results, easily transferable to an industrial environment.
- While plane detection using RANSAC provides good results in laboratory conditions, its application on containers of pieces in an industrial environment, where these are chaotically stacked and interlocked with each other, may not be sufficient.

References

[Inta] Intel, "Intel RealSense D400 Series Product Family Datasheet." [Online]. Available: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet

[Intb] ——, "Intel RealSense LiDAR Camera L515 Datasheet." [Online]. Available: https://dev.intelrealsense.com/docs/lidar-camera-l515-datasheet

[MMS+19] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," Science Robotics, vol. 4, no. 26, 2019, Último acceso: 22/09/2020. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau4984/

[Ope] OpenCV, "OpenCV." [Online]. Available: https://opencv.org/

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 2016, arXiv: 1506.02640 Último acceso: 13/11/2020. [Online]. Available: http://arxiv.org/abs/1506.02640

[RM99] A. Romero-Manchado, ""Calibración de cámaras no métricas por el método de las líneas rectas"," Mapping, ISSN 1131-9100, Nº 51, 1999, pags. 74-80, Jan. 1999.

DOCUMENTO I

MEMORIA

Índice

I. Memoria **9**

1. Introducción **11**

1.1. Contexto y descripción del proyecto 12
1.2. Motivación 13
1.3. Objetivos 13
1.4. Recursos empleados 14

2. Estado del arte **17**

2.1. Metodologías para la percepción de profundidad 18
2.1.1. RADAR 18
2.1.2. LiDAR 19
2.1.3. Luz estructurada 20
2.1.4. Visión estereoscópica 21
2.1.5. *Structure from motion (SFM)* 22
2.1.6. Algoritmos de Inteligencia Artificial 23
2.2. Algoritmos de detección de objetos y segmentación de instancias 25
2.2.1. Redes Neuronales Convolucionales (CNNs) 25
2.2.2. Algoritmos de detección de objetos: YOLO 25
2.2.3. Algoritmos de segmentación de instancias: Mask R-CNN 26
2.3. Algoritmos de robustez de agarre: Dex-Net 27
2.4. Detección de superficies planas: RANSAC 29

3. Diseño de la solución **31**

3.1. Estudio de cámaras RGB-D: Intel D435 y L515 33
3.1.1. Calibración de las imágenes RGB y de profundidad 34
3.1.2. Análisis de consistencia en mediciones consecutivas de profundidad 39
3.1.3. Análisis de deformación 41
3.1.4. Aplicación de filtros y posprocesado de la imagen de profundidad 44
3.1.5. Elección del modelo de cámara a utilizar 46
3.2. Proceso de detección de piezas 47

3.2.1. Máscara por segmentación de profundidad 47
3.2.2. Máscara por sustracción de fondo 50
3.2.3. Máscara por YOLO 55
3.3. Cálculo del agarre 62
3.3.1. Análisis del uso del algoritmo Dex-Net 63
3.3.2. Desarrollo de un algoritmo propio: combinación del algoritmo de YOLO y segmentación por profundidad 65
3.4. Detección de superficies planas (RANSAC) 70

4. Análisis de resultados **73**

5. Conclusiones y futuros desarrollos **79**

6. Alineación con los Objetivos de Desarrollo Sostenible **83**

Bibliografía **85**

Índice de figuras

1. Gráfico comparativo de resultados entre modelos de ML y DL [Sha17] 11
2. Robot YuMi en la plataforma de evaluación de agarre del Laboratorio de la UC Berkeley para Dex-Net 2.0 [MMS+19]. 12
3. Flujo de información a lo largo del proyecto. 12
4. Ejemplo de imagen RADAR [BML20]. 18
5. Ejemplo de imagen LiDAR [BML20]. 19
6. Ejemplo de proyección de un patrón de luz estructurada [str20]. 20
7. Funcionamiento de una cámara estereoscópica [HYK08]. 21
8. Ejemplos de distorsión en una imagen: de barril, de cojín y de bigote [Fri]. 22
9. Funcionamiento del SFM [YK13]. 22
10. Ejemplo de *feature matching* a través del algoritmo SIFT [Low01]. 23
11. Ejemplo de renderización neural volumétrica [MST+20]. 24
12. Ejemplo de robustez en las imágenes de entrada para NeRF-W [MBRS+21]. 24
13. Cálculo del *nerfie* a pesar del movimiento del sujeto [PSB+21]. 24
14. Convolución de una imagen 5x5x1 con un kernel 3x3x1, obteniendo un output de 3x3x1 [Sah18]. 25
15. Ejemplo de aplicación de YOLO [RDGF16]. 26
16. Ejemplo de aplicación de Mask R-CNN [HGDG18]. 27
17. Robot YuMi en la plataforma de evaluación de agarre del Laboratorio de la UC Berkeley para Dex-Net 2.0 [MMS+19]. 28
18. Proceso de cálculo del agarre final a partir de agarres candidatos por la GQ-CNN [MMS+19]. 29
19. Ajuste de planos en una nube de puntos mediante el algoritmo RANSAC [GMR11]. . 30
20. Configuración del robot YuMi del laboratorio. 31
21. Piezas pertenecientes al conjunto de datos propio. 32
22. *Grippers* utilizados sobre el robot YuMi: pinza paralela (izquierda), y ventosa de succión (derecha). 32

23. Ejemplo de imagen RGB-D de la cámara Intel D435: de color o RGB (izquierda), de profundidad valores netos de distancia normalizados (centro), y de profundidad con mapa de color (derecha). 32
24. Cámaras RGB-D utilizadas en el proyecto. A la izquierda, el modelo Intel RealSense D435 (estereoscópica); a la derecha, el modelo L515 (LiDAR). 34
25. Visualización de los parámetros intrínsecos en una cámara con modelo estenopeico [Cor17]. 35
26. Ejemplo de píxel inclinado. 36
27. Ejemplos de distorsión radial en una imagen: de barril, de cojín y de bigote [Fri]. . . 37
28. Detección de los puntos característicos en un damero para la calibración de la cámara Intel RealSense L515 sobre una versión distorsionada de la imagen de entrada durante el proceso de calibración. 37
29. Proyección de unos ejes cartesianos sobre un marcador ArUco previa calibración de la cámara (izquierda), y posterior a ella (derecha), para la cámara Intel RealSense L515. 38
30. Proyección de una máscara de los puntos de distancia mínima a la cámara en la cara superior de un cubo. 39
31. Distribución de medidas consecutivas de un mismo punto para la cámara Intel D435 a una distancia real de 735 mm. 40
32. Distribución de medidas consecutivas de un mismo punto para la cámara Intel L515 a una distancia real de 735 mm. 40
33. Boxplot del error en la toma de medidas consecutivas. 41
34. Sistema de ejes de coordenadas del robot YuMi y de la cámara RGB-D. 42
35. Distribución de los cubos de gomaespuma en la mesa de trabajo durante el análisis de la deformación del eje Y. 43
36. Visualización del error entre los centroides de las caras superiores de los cubos (en rojo) y los reales (en azul) para una coordenada $y = 450\ mm$. 43
37. Error en el cálculo del centroide en diferentes puntos del área de trabajo. A mayor error, mayor tamaño de burbuja. 44
38. Comparación de una imagen de profundidad original sin filtros (izquierda) frente a una imagen de profundidad posprocesada con filtros (derecha), ambas obtenidas con la cámara D435. 45
39. Imagen de profundidad con posprocesado de la cámara estereoscópica D435 (izquierda), y de la cámara LiDAR L515 (derecha). 46
40. Definición de las dimensiones z_{ref}, z_{pieza} y $z_m in$. 48
41. Proceso de obtención de máscara por segmentación de profundidad. 49
42. Ejemplos de máscaras binarias obtenidas por segmentación por profundidad (derecha) para una variedad de piezas (presentadas a través de su imagen RGB, izquierda). . . 50

43. Ejemplo de uso de la técnica de croma en el ámbito cinematográfico [gor20]. 51
44. Primera parte del proceso de sustracción de fondo: imagen base, imagen con el objeto, y diferencia entre ambas imágenes. 51
45. Proceso de obtención de máscara por sustracción de fondo. 52
46. Ejemplos de máscaras binarias obtenidas por sustracción de fondo para una variedad de piezas. 53
47. Ejemplo de máscara defectuosa obtenida por sustracción de fondo. 54
48. Subconjunto de piezas pertenecientes al conjunto de datos propio. 56
49. Disposiciones de las piezas utilizadas para la creación del dataset. De izquierda a derecha, y de arriba a abajo: fondo vacío; únicamente piezas sin interés; únicamente piezas objetivo; combinado de piezas objetivo y piezas sin interés. 57
50. Etiquetado de la pieza M1 a través del software makesense.ai. 58
51. Representación gráfica del concepto de IoU [K.21]. 60
52. Visualización del efecto de aplicar el algoritmo NMS sobre un conjunto de detecciones. Como se puede apreciar, el número de cajas por detección por clase queda reducido a aquellas con mayores índices de confianza; siempre de forma independiente por cada clase de objeto. 61
53. Ejemplos de máscaras obtenidas por YOLO para piezas de diferentes tamaños. . . . 62
54. Flujo de información a lo largo del proyecto. 63
55. Propuestas de agarre paralelo ofrecidas por Dex-Net (derecha) dadas una máscara del objeto obtenida por sustracción de fondo (en rojo, izquierda) y una imagen de profundidad. 64
56. Pieza M5 en posición inclinada. Dado que el agarre óptimo en este caso sería inclinado respecto al plano de trabajo, Dex-Net no sería capaz de calcularlo. 65
57. Pieza M5: cubo negro de gomaespuma. 65
58. Pinza de agarre paralelo original del robot YuMi. 66
59. Flujo de información en la obtención de las coordenadas del agarre final para un agarre paralelo. 67
60. Sistema de coordenadas del agarre paralelo. 67
61. Proceso de cálculo del ángulo θ_{TP} para la orientación del agarre paralelo. 68
62. Proceso de cálculo del punto terminal $(x, y, z)_{TP}$ para el agarre con ventosa. 69
63. Detección de planos horizontales a través del algoritmo RANSAC. 72
64. Detección de planos inclinados a través del algoritmo RANSAC. 72
65. Comparativa de las máscaras resultantes tras la aplicación de los métodos de segmentación por profundidad, sustracción de fondo, y YOLO. 75
66. Comparativa de los resultados obtenidos para las diferentes opciones de agarre (paralelo y ventosa). 76

67. Cálculo del agarre por ventosa mediante detección de planos a través del algoritmo de RANSAC. 76
68. *Gripper* de forma adaptativa desarrollado por la compañía Festo [fUR]. 81
69. Lista de los 17 Objetivos de Desarrollo Sostenible propuestos por la ONU [Uni15]. . 83

PARTE I

MEMORIA

Capítulo 1

Introducción

En los últimos años, el mundo de la inteligencia artificial ha observado un gran avance gracias al crecimiento exponencial en potencia de computación. Esto se puede ver reflejado en muchas de sus áreas de aplicación, que abarcan desde la detección de objetos hasta el procesamiento del lenguaje natural, pasando por la publicidad, así como por los asistentes personales de Google y Amazon.

El último *boom* en el campo de la inteligencia artificial ha sido el denominado Deep Learning. Este modelo de aprendizaje profundo está conformado por una serie de redes neuronales de mucha mayor profundidad que las utilizadas hasta entonces. El Deep Learning, como se ha mencionado anteriormente, requiere una gran capacidad de cálculo y de datos de entrada; a cambio, proporciona unos resultados muy superiores a los modelos de Machine Learning tradicionales [Sha17].

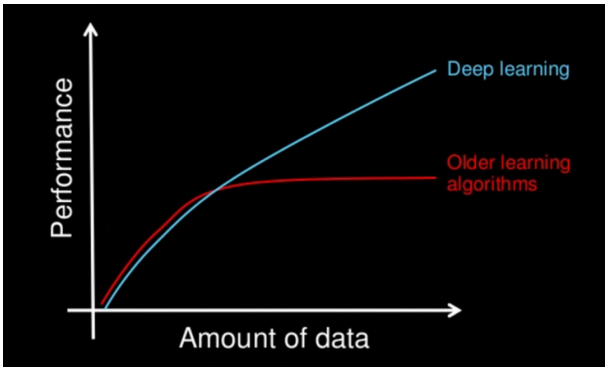


Figura 1. Gráfico comparativo de resultados entre modelos de ML y DL [Sha17]

El avance en este campo ha impulsado el desarrollo de muchas otras áreas, como es el caso del reconocimiento de imágenes en la Visión Artificial a través de redes neuronales convolucionales, o CNNs. Será este campo, precisamente, el que será abordado en el desarrollo de este proyecto.

1.1. Contexto y descripción del proyecto

Utilizar un robot y obtener un agarre suficientemente robusto para una amplia variedad de objetos es actualmente un gran desafío al que se enfrentan los comercios online, los procesos de fabricación y los robots de servicio. Una de las limitaciones existentes es el hecho de que una sola pinza no es capaz de agarrar cualquier tipo de objeto de forma robusta. Este es el caso, por ejemplo, de una ventosa, para la cual es imposible crear vacío en objetos porosos.

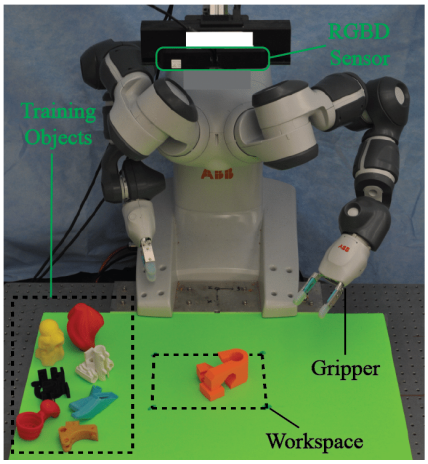


Figura 2. Robot YuMi en la plataforma de evaluación de agarre del Laboratorio de la UC Berkeley para Dex-Net 2.0 [MMS+19].

Para solucionar este problema, este proyecto en cuestión forma parte de un proyecto de mayor envergadura realizado en el Instituto de Investigación Tecnológica (IIT) de la Universidad Pontificia Comillas. Se pretende utilizar un robot en conjunción con una cámara de profundidad RGB-D de cara a que dicho autómata sea capaz de agarrar cualquiera de los objetos comprendidos dentro de un dataset de piezas automovilísticas específico de forma completamente autónoma, únicamente a través de las imágenes de color y profundidad obtenidas a través de una cámara RGB-D según la configuración mostrada en la Figura 2.

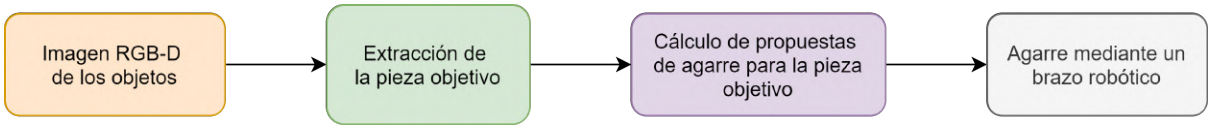


Figura 3. Flujo de información a lo largo del proyecto.

Para ello, como se muestra en la Figura 3, el primer paso es analizar la imagen de una cámara RGB-D y aislar en ella el objeto que se pretende agarrar. Esta información puede posteriormente ser utilizada como input a la red neuronal de calidad de agarre (GQ-CNN) Dex-Net, desarrollada por el Laboratorio de Automatización de Berkeley [MMS+19], la cual se encargará de calcular

un agarre óptimo para la pieza.

Cabe destacar que, aunque originalmente se pretende utilizar la GQ-CNN de Dex-Net para el cálculo del agarre, un análisis de su rendimiento aplicado a las condiciones específicas de este proyecto revela que los resultados que ofrece no son lo suficientemente satisfactorios. Por tanto, en la última versión del proyecto se decide prescindir de este algoritmo y desarrollar en su lugar uno propio.

El área de trabajo abarcada por este proyecto será, por tanto, la parte de visión artificial y cálculo del agarre, cuyo objetivo es determinar una heurística que permita identificar y aislar la pieza más adecuada para ser agarrada en función de su posición, orientación e interacción con otras piezas. Dicha pieza será posteriormente enviada al algoritmo desarrollado para obtener un agarre definitivo.

1.2. Motivación

Como se ha mencionado en los capítulos anteriores, existe actualmente un gran impedimento en los comercios y los procesos de fabricación para poder utilizar un único sistema de agarre en sus robots que les permita manipular una gran variedad de objetos de forma suficientemente robusta.

Para dar solución a este problema, y de acuerdo con lo ya expuesto, la evolución natural de los algoritmos de Machine Learning y del ámbito de la robótica confluye en un punto en el cual existe la posibilidad de dotar de inteligencia a un autómata programable. A partir de ahora, dichos autómatas dejan de ser simples máquinas que realizan una tarea asignada y repetitiva. A partir de ahora, tenemos la posibilidad de *enseñarles* a coger objetos, a manipularlos.

Estas nuevas técnicas de producción abrirían las puertas a la posibilidad de lograr un nuevo grado de flexibilidad en las diferentes industrias. Esto permitiría no solo abaratar costes de logística y de montaje, sino que daría la oportunidad a las empresas de ofrecer una mayor personalización de sus productos e, incluso, de abaratar sus costes hacia sus consumidores finales.

1.3. Objetivos

En consonancia con la motivación del proyecto, los objetivos del mismo se enfocan en el desarrollo de la parte correspondiente a la Visión Artificial que permitirá al algoritmo identificar

el agarre óptimo para un objeto dado.

Así pues, este proyecto se centrará en:

1. Desarrollar un dataset utilizando cada una de las cuarenta y seis piezas disponibles que permita analizar la utilidad de diferentes algoritmos de detección de objetos.
2. Integrar un modelo robusto que permita obtener una detección del objeto óptimo a agarrar por el robot en diversas situaciones. Dicho modelo deberá ser válido tanto para el caso en el que existiera un único objeto, como para aquel en el cual existan múltiples instancias de objetos solapándose entre sí.
3. Desarrollo de una heurística capaz de seleccionar, cuando se disponga de una multitud de objetos solapados entre sí, cuál es aquel que posee más probabilidades de ofrecer un agarre exitoso.
4. Analizar la viabilidad de aplicar algoritmos de IA en entornos industriales, donde las condiciones de trabajo difieren de aquellas de un entorno controlado.

1.4. Recursos empleados

Los recursos necesarios principales para el desarrollo del proyecto son los que se listan a continuación:

- **Cámaras RGB-D Intel RealSense D435 y L515**: de cara a obtener tanto imágenes en color RGB como imágenes de profundidad. Gracias a ellas, le serán enviada al algoritmo Dex-Net la máscara del objeto en cuestión que se pretende agarrar, así como un mapa de profundidad, a través del cual el algoritmo podrá calcular las dimensiones reales del objeto y determinar así el agarre óptimo que realizar.

  Se utilizará, además, el software de visualización oficial, Intel RealSense Viewer, así como la correspondiente librería de Python pyrealsense2 [1].

- **Python**: se escoge como lenguaje de programación debido a su flexibilidad y su extendido uso en la comunidad de inteligencia artificial. La programación se realizará en la aplicación PyCharm, y se hará asimismo uso de librerías como OpenCV u Open3D para el tratamiento y procesamiento de imágenes RGB y de profundidad.

- **GitHub**: como método de control de versiones del código desarrollado, así como plataforma de desarrollo colaborativo con el resto de miembros del equipo.

[1] https://pypi.org/project/pyrealsense2/

- **Google Colab**: debido a las limitaciones de computación a falta de una GPU lo suficientemente potente en los ordenadores disponibles (tanto personal como en el laboratorio), se hará uso de herramientas de computación en la nube. En este caso, se utilizará la ofrecida por Google (https://colab.research.google.com/) en su versión gratuita, lo cual limita la potencia de procesamiento a 15 GB de GPU y 16 GB de memoria RAM.

- **Componentes de automóvil**: las cuarenta y seis piezas utilizadas a lo largo de este proyecto son parte de componentes utilizados en el ensamblaje de un automóvil (guardagafas, faros...) y han sido proporcionadas por la empresa colaboradora. Dichos componentes conformarán los datasets sobre los que se basará la investigación

Cabe destacar que, aunque no se hace uso explícito de él en esta parte específica del proyecto, es parte esencial de este el robot YuMi - IRB 14000 de ABB, sobre el cual se pretende implementar los agarres calculados a través del progreso de este trabajo.

Capítulo 2

Estado del arte

COMO se ha mencionado anteriormente, el Deep Learning, o aprendizaje profundo, es un concepto que se ha puesto muy de moda en los últimos años. La pregunta es, ¿por qué ahora?

La realidad es que esta técnica no solo ha visto un gran empuje en su desarrollo gracias a los avances en potencia de computación. Hay que pensar que este tipo de algoritmos entrenan y aprenden a partir de los datos que les son suministrados, y actualmente nos encontramos inmersos en la era de la información: con la llegada de la digitalización, el abaratamiento de los dispositivos de almacenamiento, y un cambio de mentalidad a la hora de apreciar el valor de la información, hemos entrado en una tendencia de acumular más y más datos, lo que ha dado lugar al nacimiento del Big Data.

Para tratar dicha cantidad de datos, se requiere de potentes y complejas técnicas como, por ejemplo, el Deep Learning. Dicha técnica no es más que una versión vitaminada de las redes neuronales (o *Neural Networks*), las cuales han dado un nuevo resurgir al campo del Machine Learning y, por tanto, al campo de la Inteligencia Artificial. Actualmente, los algoritmos más recientes que hacen uso de estos avances dentro de la Visión Artificial son aquellos dedicados a la detección y segmentación de objetos, así como los dedicados a la detección de poses. Serán estos algoritmos los que permitirán llevar a cabo este proyecto y que supondrán la piedra angular en su desarrollo.

A continuación se analizan más en detalle algunos de estos algoritmos, así como sus principios de funcionamiento.

2.1. Metodologías para la percepción de profundidad

El concepto *range imaging* describe el proceso de producir una imagen 2D capaz de almacenar la profundidad de una escena a partir de un punto origen, normalmente encontrado en el sensor del propio dispositivo de medición. De esta forma, los valores de los píxeles de la imagen de profundidad resultante representan la distancia a cada uno de los puntos de dicha escena.

Existen en la actualidad numerosos procedimientos y tecnologías que permiten la medición de la profundidad. El uso de un procedimiento u otro depende en gran medida de las condiciones en las que se quiere efectuar la medición, así como de la precisión deseada y de los recursos disponibles.

2.1.1. RADAR

El término RADAR proviene de sus siglas en inglés *RAdio Detection And Ranging* (detección y distanciometría de radio), y se define como un "sistema que utiliza radiaciones electromagnéticas reflejadas por un objeto para determinar la localización o velocidad de este" [AR].

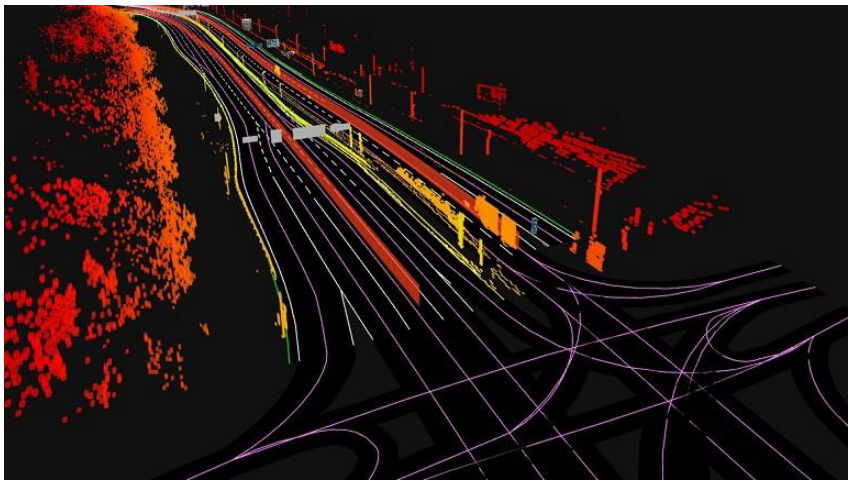


Figura 4. Ejemplo de imagen RADAR [BML20].

En lo que respecta al procesamiento de la señal RADAR, existen dos metodologías diferentes para obtener una medida de distancia. La primera es el uso del tiempo de tránsito (*Time of flight*, o ToF), cuyo principio de funcionamiento consiste en medir el tiempo que tarda en volver el eco de un pequeño pulso transmitido inicialmente [Li24].

La segunda forma de estimar distancias se basa en el principio de modulación de frecuencia [Gib02]. Se emite una señal (normalmente sinusoidal) a una frecuencia cambiante a lo largo del tiempo; cuando el sensor recibe el eco de dicha señal, compara la frecuencia de la señal recibida con la de la señal enviada, pudiendo así inferir cuánto tiempo ha transcurrido entre una y otra y, por lo tanto, qué distancia hay hasta el blanco.

Algunos de los puntos fuertes de la tecnología RADAR son su buen funcionamiento en situaciones de baja luminosidad, su bajo coste y su excelente rendimiento a largas distancias debido a la baja absorción que sufren sus ondas, pudiendo llegar a los 250 metros de rango con frecuencias de alrededor de 76 GHz [BML20]. Por otro lado, una desventaja es la baja resolución que ofrece.

Un ejemplo de imagen RADAR puede observarse en la Figura 4. Entre las numerosas aplicaciones del RADAR se incluyen el control de tráfico aéreo y terrestre, la meteorología y aplicaciones militares.

2.1.2. LiDAR

Similar al RADAR, el término LiDAR proviene del inglés *Light Detection And Ranging* y permite medir la distancia desde un emisor láser a un objeto o superficie, utilizando para ello un haz láser pulsado.

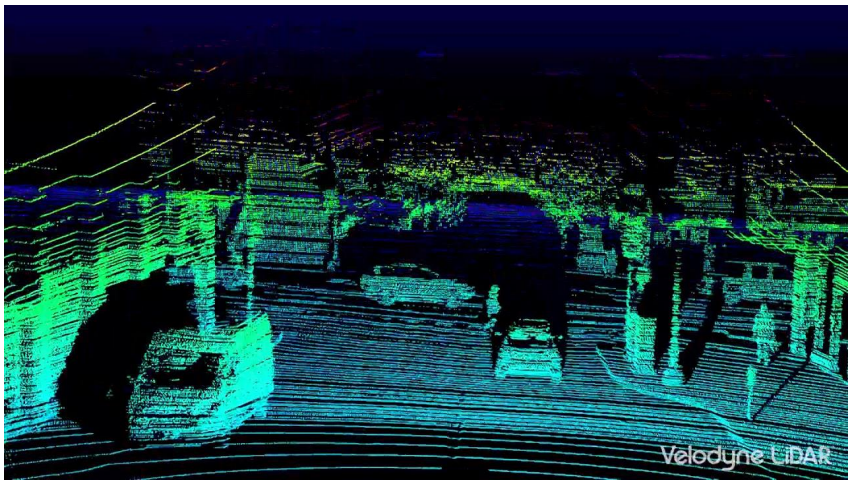


Figura 5. Ejemplo de imagen LiDAR [BML20].

El principio de medida más popular que sigue esta tecnología es el de tiempo de vuelo: conocido el valor de la velocidad de la luz ($3{\cdot}10^{8}$ m/s), se emite un haz láser que rebota contra la superficie del objeto de destino y se mide el tiempo que toma la reflexión en ser recibida por el sensor. Una vez conocidos estos parámetros, se deduce la distancia a dicho objeto de forma

trivial.

La frecuencia de emisión de pulsos láser en los sensores LiDAR es muy alta, pudiendo escanear millones de puntos por segundo [Nat12] y, poseyendo algunos de los modelos un campo de visión horizontal de 360 º, se pueden obtener resultados como los mostrados en la Figura 5.

Aunque el coste de los sensores LiDAR ha sido históricamente muy alto, el gran desarrollo en el campo de la conducción autónoma está favoreciendo en la actualidad la aparición de versiones cada vez más asequibles. Asimismo, estos sensores tienen un rango mínimo inferior a un metro, lo cual hace que sean muy apropiados para aplicaciones interiores, como pueden ser los aspiradores robot.

2.1.3. Luz estructurada

Los escáneres de luz estructurada proyectan un patrón específico (normalmente una rejilla o unas líneas rectas) que se ve deformado al proyectarse sobre la superficie objetivo [Gen11]. La manera en la que este patrón se deforma al golpear las distintas superficies permite a los sistemas de visión inferir la información de profundidad de la escena (ver Figura 6).

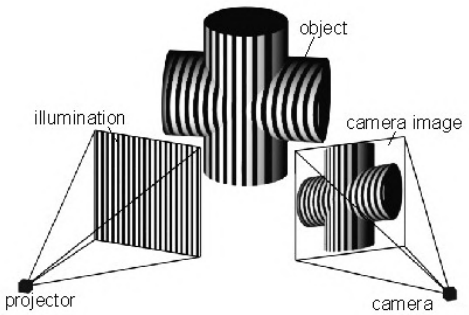


Figura 6. Ejemplo de proyección de un patrón de luz estructurada [str20].

De cara a asegurar la robustez en las medidas, a menudo se utiliza luz estructurada *invisible*, evitando que pueda interferir con otras tareas de visión computacional y confundir así al sistema. Un ejemplo de luz estructurada invisible es la luz infrarroja.

Casos de uso de escáneres de luz estructurada incluyen la fotografía de huellas dactilares en una escena 3D o el reconocimiento facial en muchos de los smartphones de última generación.

2.1.4. Visión estereoscópica

Un proceso tradicional para la medición de la profundidad de una escena es la visión o fotogrametría estereoscópica. En esta técnica, se dispone de dos cámaras diferentes, desplazadas horizontalmente la una de la otra, que permiten capturar dos puntos de vista diferentes de una misma escena, como puede verse en la Figura 7.

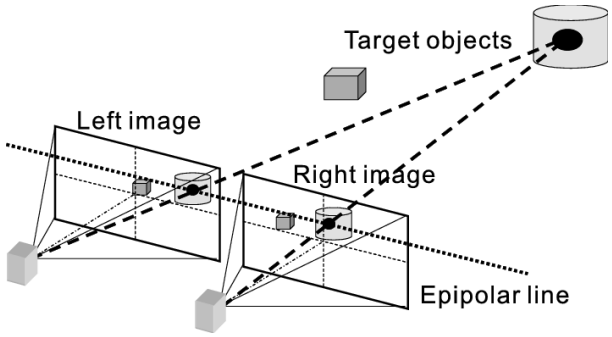


Figura 7. Funcionamiento de una cámara estereoscópica [HYK08].

Al comparar estas dos imágenes, se puede calcular la profundidad de la escena a través de un mapa de disparidad, el cual representa la diferencia entre las imágenes capturadas por ambas cámaras a causa de sus respectivas perspectivas [HYK08]. Este proceso es similar a la visión binocular presente en los seres humanos, la cual funciona de manera análoga.

En visión artificial, es necesario realizar un preprocesado de la información capturada por las cámaras para poder unificarlas:

1. Las imágenes no deben sufrir de ningún tipo de distorsión, como puede ser la distorsión de barril (*barrel distorsion*), de cojín (*pincushion distorsion*) o de bigote (ver Figura 8). El objetivo final es obtener una imagen completamente plana y que coincida en lo máximo posible con la que ofrecería una cámara estenopeica o *pinhole camera*.

2. Se debe llevar a cabo una rectificación de las imágenes; es decir, cambiar su perspectiva para que ambas estén proyectadas sobre un plano equivalente.

3. Se comparan los puntos característicos (*feature points*) de las imágenes de cara a estimar la diferencia en la percepción de ambas a través de algoritmos como SIFT [Low01] y, por ende, inferir la profundidad de cada punto de la imagen.

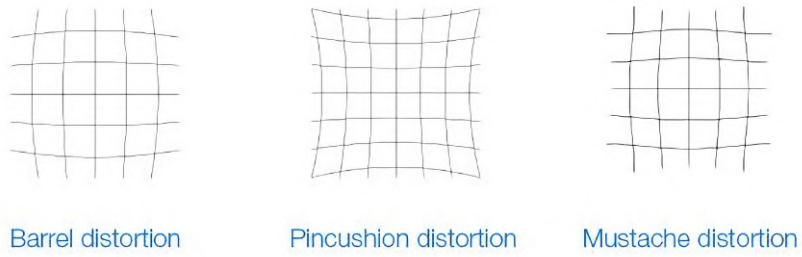


Figura 8. Ejemplos de distorsión en una imagen: de barril, de cojín y de bigote [Fri].

2.1.5. *Structure from motion (SFM)*

Las metodologías hasta ahora presentadas, si bien fiables, poseen un inconveniente común, y es su elevado coste. Por el contrario, el término SFM (*Structure From Motion*) hace referencia a un método fotogramétrico automatizado que ofrece resultados de alta resolución a un coste reducido en comparación con el resto de técnicas analizadas.

Este método sigue el mismo funcionamiento que la fotogrametría estereoscópica, es decir, la información de profundidad se infiere a partir de una superposición de imágenes obtenidas desde diferentes perspectivas; sin embargo, el SFM difiere del método anterior en que la geometría de la escena y las posiciones de la cámara se resuelven computacionalmente y mediante una única cámara que se desplaza por la escena [TRC+16], tal y como se ilustra en la Figura 9.

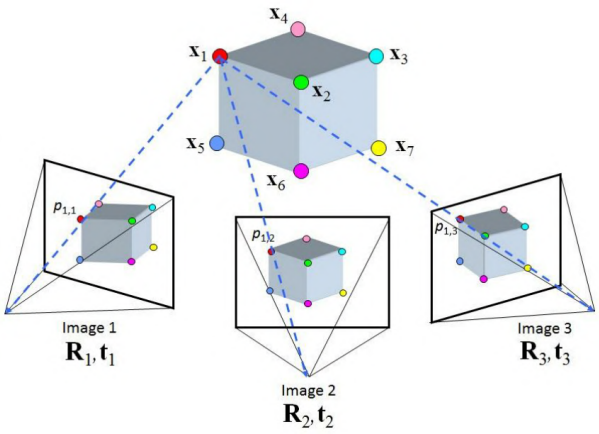


Figura 9. Funcionamiento del SFM [YK13].

Una vez obtenidas las diferentes fotografías, la escena original puede ser reconstruida haciendo uso de algoritmos de detección de características o *feature matching* como SIFT [Low01], mostrado en la Figura 10. Sin embargo, al utilizar una única cámara y usar el tiempo

como segundo parámetro, el uso de esta técnica se ve limitado a escenas estáticas.

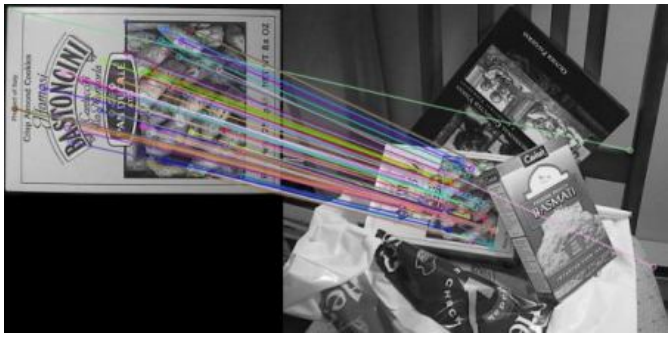


Figura 10. Ejemplo de *feature matching* a través del algoritmo SIFT [Low01].

La técnica SFM encuentra gran aplicación en los campos de la ingeniería y de la reconstrucción tridimensional de obras de arte e hitos arquitectónicos.

2.1.6. Algoritmos de Inteligencia Artificial

El campo de la Inteligencia Artificial ha visto un importante auge en los últimos años en un amplio rango de áreas que abarcan desde la detección de objetos en una imagen hasta el procesamiento del lenguaje natural. Uno de los campos que también está siendo partícipe de este desarrollo es el de la renderización neural volumétrica, con algoritmos como NeRF [MST+20] y que, aun siendo tan recientes, compiten de forma directa con otros más tradicionales, como puede ser la fotogrametría.

NeRF es, por lo tanto, un algoritmo que cae bajo la rama de la renderización neural volumétrica. Esta técnica considera una escena tridimensional perteneciente a un volumen y que está compuesta por *vóxeles*, una versión tridimensional de los píxeles. De esta forma, se puede entrenar un conjunto de redes neuronales que sean capaces de recrear dicha escena tridimensional y renderizarla en una imagen 2D. Una vez obtenida dicha imagen, esta puede ser comparada a la imagen original real; de esta manera, si ambas imágenes son similares, se podría considerar que el algoritmo ha sido capaz de generar con éxito una escena tridimensional correspondiente a la imagen de entrada y que podría ser utilizada a partir de entonces como tal [MST+20].

Desde su publicación original en marzo de 2020, han surgido nuevas versiones posteriores de NeRF que mejoran sustancialmente sus resultados, como NeRF In The Wild [MBRS+21] (presentada en la Figura 11), el cual permite que las imágenes de entrenamiento puedan poseer condiciones lumínicas y ambientales ligeramente diferentes, como puede ocurrir en fotografías

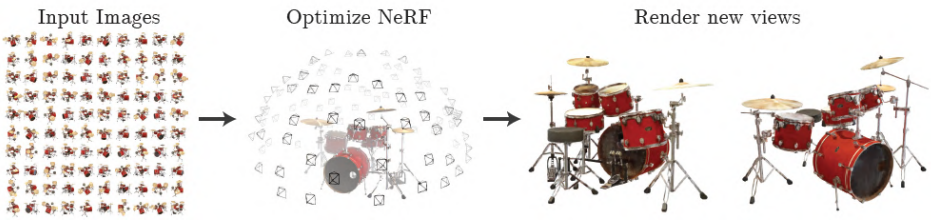


Figura 11. Ejemplo de renderización neural volumétrica [MST+20].

tomadas a diferentes horas del día, como puede verse en la Figura 12.

Figura 12. Ejemplo de robustez en las imágenes de entrada para NeRF-W [MBRS+21].

La evolución más reciente de este algoritmo data de mayo de 2021 y se conoce como *Deformable NeRF* o D-NeRF. D-NeRF abre la puerta a la posibilidad de que existan deformaciones en la propia escena que pueden ser debidas, por ejemplo, a un sujeto en movimiento, y no solo cambios de luminosidad [PSB+21]. Esto permite obtener unos resultados tan impresionantes como los mostrados en la Figura 13, un ejemplo de lo que los autores denominan *nerfies* que permiten crear una representación tridimensional de un sujeto a través de un vídeo que este ha tomado previamente de sí mismo.

Figura 13. Cálculo del *nerfie* a pesar del movimiento del sujeto [PSB+21].

2.2. Algoritmos de detección de objetos y segmentación de instancias

2.2.1. Redes Neuronales Convolucionales (CNNs)

Como se ha mencionado anteriormente en el Capítulo 1, gran parte de la revolución que se ha vivido en los últimos años dentro del Deep Learning se lo debemos en sus inicios a los avances en el campo de la visión por ordenador (o visión artificial); y en concreto, a un tipo de red neuronal muy importante que está especializada para trabajar con imágenes: las redes neuronales convolucionales (o CNNs).

Una Red Neuronal Convolucional es un tipo de red neuronal que se caracteriza por aplicar un tipo de capa donde se realiza una operación matemática conocida como *convolución*. Aplicada sobre una imagen, se trata de utilizar una matriz de pesos iniciales aleatorios (conocida como matriz de convolución) que se desliza a través de la imagen original (ver Figura 14) realizando una operación de convolución sobre los diferentes píxeles, creando así otra imagen resultante en la que, según aplicamos más y más capas, se van detectando patrones más y más complejos de la imagen original.

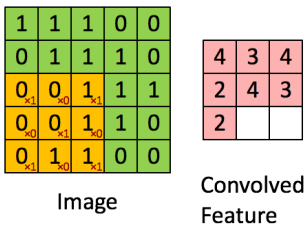


Figura 14. Convolución de una imagen 5x5x1
con un kernel 3x3x1, obteniendo un output de 3x3x1 [Sah18].

De esta forma, se llega a ser capaz de crear un mapa de características que más adelante permite identificar, por ejemplo, un rostro, un vehículo, o incluso un peatón que pudiera cruzarse en el camino de un coche autónomo. Conseguimos así que una máquina sea capaz de percibir el mundo que le rodea tal y como lo haría un humano.

2.2.2. Algoritmos de detección de objetos: YOLO

La detección de objetos ha sido desde siempre uno de los problemas clásicos en la Visión Artificial, y es uno de los pilares de los vehículos de conducción autónoma.

El objetivo es conseguir saber tanto *qué* objetos hay en la imagen, como *dónde* se encuentran. En contraposición a lo que pueda parecer a primera vista, la detección de objetos es mucho más complicada que un simple problema de clasificación. Se ha de tener en cuenta no solo la posición del objeto que se quiere detectar en la imagen, sino también el hecho de que pueden existir múltiples instancias de múltiples objetos.

Uno de los algoritmos de detección en tiempo real más populares en la actualidad es YOLO (You Only Look Once), el cual hace uso de redes neuronales convolucionales. YOLO surgió originalmente en 2015 y fue desarrollado por Joseph Redmon et al. [RDGF16], obteniendo rápidamente una gran repercusión en el mundo de la Visión Artificial; desde entonces, el algoritmo ha visto hasta cinco revisiones. La popularidad de este algoritmo surge de su gran precisión a la hora de detectar y clasificar objetos, pudiendo hacerlo incluso en tiempo real.

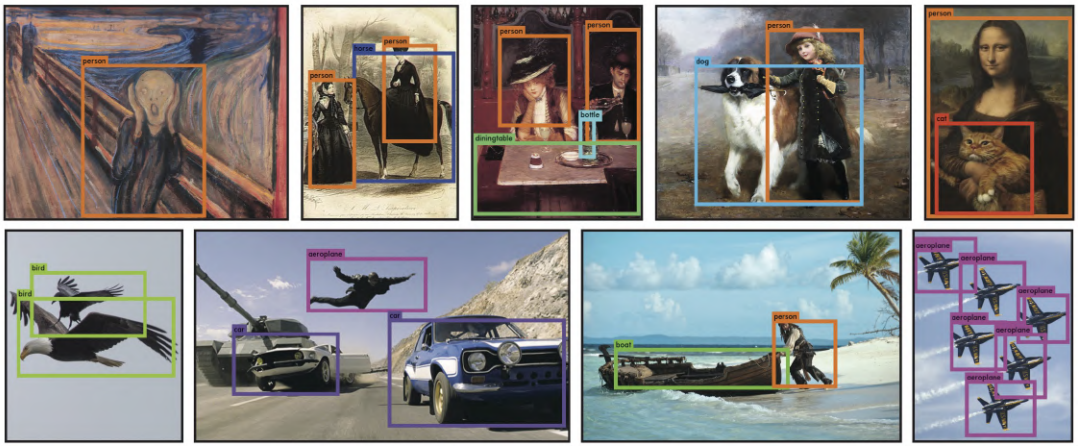


Figura 15. Ejemplo de aplicación de YOLO [RDGF16].

El resultado obtenido es el mostrado en la Figura 15. Obtenemos una serie de *bounding boxes* en las cuales se recuadra el objeto encontrado, así como la categoría en la que ha sido clasificado y la precesión proporcionado por el modelo.

2.2.3. Algoritmos de segmentación de instancias: Mask R-CNN

La siguiente evolución lógica tras la detección de objetos es ir más allá de un simple rectángulo y ser capaz de distinguir con una precisión de píxeles el contorno del objeto. Esto es conocido como segmentación de instancias.

Desde su nacimiento hace un par de años, Mask R-CNN se ha convertido en el estado del arte en el área de la segmentación de instancias. Este algoritmo recibe una imagen, y es capaz de devolver las *bounding boxes*, clases y máscaras correspondientes a cada objeto encontrado

[HGDG18].

Mask R-CNN se conforma de dos etapas. En primer lugar, se generan propuestas sobre las regiones donde *es posible* que haya un objeto en la imagen de entrada; en segundo lugar, se predicen las clases de dicho objeto, se refinan las *bounding boxes* y se genera una máscara a nivel de píxel del objeto, basándose en la propuesta de la primera etapa.

En comparación con otras tareas similares en el campo de la Visión Artificial, la segmentación de instancias es una de las más complicadas. Observemos el ejemplo de la Figura 16.

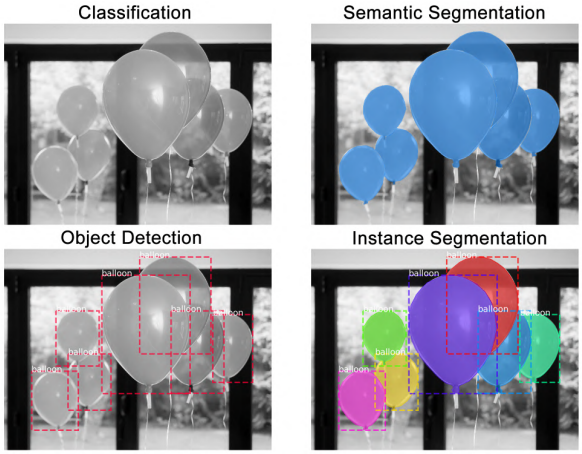


Figura 16. Ejemplo de aplicación de Mask R-CNN [HGDG18].

De esta forma, Mask R-CNN permite realizar las tareas siguientes, en orden ascendente de dificultad:

- **Clasificación**: afirmar que existe un globo en la imagen.
- **Segmentación semántica**: encontrar todos los píxeles en los que hay globos.
- **Detección de objetos**: afirmar que existen siete globos en la imagen, en las posiciones señaladas. A partir de este punto, se comienza a tener en cuenta el hecho de que dos instancias de objetos diferentes pueden solaparse entre sí.
- **Segmentación de instancias**: afirmar que existen siete globos en la imagen, en las posiciones señaladas, y encontrando exactamente qué píxeles pertenecen a cada globo.

2.3. Algoritmos de robustez de agarre: Dex-Net

El proyecto Dex-Net (*Dexterity Network*) [MMS+19] es un proyecto de investigación del Laboratorio de la Universidad de California en Berkeley (EEUU). Su objetivo principal es

lograr implementar sobre un brazo robótico lo que se denomina un "agarre universal" (*universal picking*) capaz de abarcar un amplio rango de objetos de diferentes formas, tamaños y texturas, como los mostrados en la Figura 17.

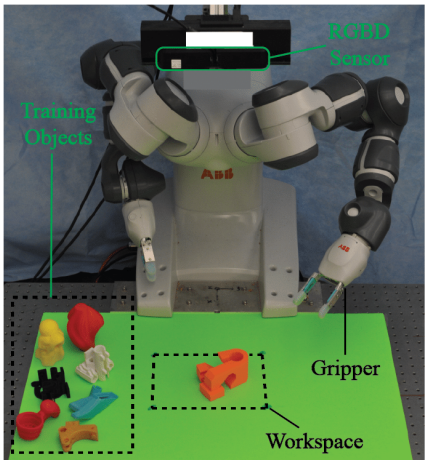


Figura 17. Robot YuMi en la plataforma de evaluación de agarre del Laboratorio de la UC Berkeley para Dex-Net 2.0 [MMS+19].

La versión más reciente de este algoritmo, Dex-Net 4.0, explora la posibilidad de realizar agarres de forma ambidiestra, es decir, utilizando dos brazos diferentes con grippers heterogéneos. El primero de los brazos estaría equipado con una pinza paralela capaz de agarrar objetos de pequeño tamaño o con forma paralelepípeda; el segundo brazo, con una ventosa, la permitiría el agarre de objetos que poseyeran superficies lisas, imposibles de agarrar a través de una pinza paralela.

Para lograr su objetivo, Dex-Net hace uso de modelos de redes neuronales convolucionales (analizadas en la sección 2.2.1). En este caso, las CNNs utilizadas están especializadas en asociar nubes de puntos tridimensionales con una robustez de agarre y se conocen como GQ-CNN (*Grasp Quality Convolutional Neural Networks*).

El primer paso del proceso es analizar la imagen de profundidad enviada por la cámara, y usarla para proponer múltiples candidatos de agarre aleatorios. En el agarre paralelo, cada uno de estos candidatos está compuesto por dos puntos de contacto, a partir de los cuales la GQ-CNN puede inferir la orientación de la pinza y comprobar que no existan colisiones en el recorrido. En el agarre por ventosa, simplemente se busca una superficie plana lo suficientemente grande que esté cercana al centro de gravedad del objeto. Cabe destacar que, en ambos casos, los agarres propuestos son siempre perpendiculares a la superficie sobre la que se encuentra la pieza.

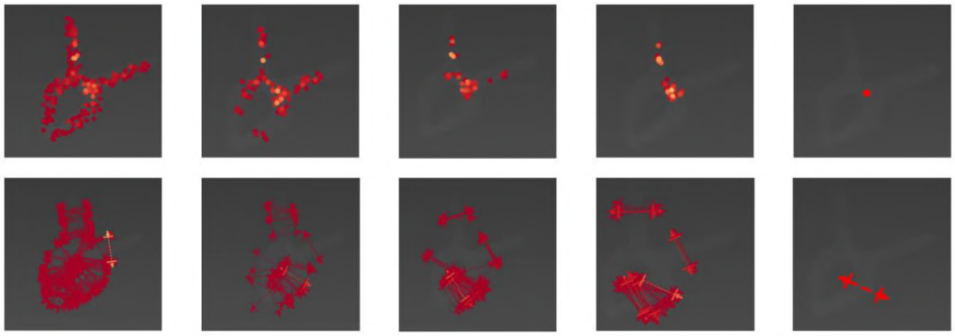


Figura 18. Proceso de cálculo del agarre final a partir de agarres candidatos por la GQ-CNN [MMS+19].

Como se muestra en la Figura 18, una vez que se tienen los candidatos, estos son ordenados en función de la robustez que ofrece su agarre, la cual es calculada por la GQ-CNN. A continuación, el algoritmo ajusta un modelo de gausianas sobre los mejores candidatos, tras lo cual se vuelven a calcular unos nuevos en función de los anteriores de forma iterativa, convergiendo así hacia una propuesta de agarre final.

Ejemplos de una implementación exitosa de este algoritmo incluyen la automatización de un amplio abanico de aplicaciones, como pueden ser en el procesamiento de pedidos online, los procesos de fabricación e inspección, o incluso el desarrollo de robots para el hogar.

2.4. Detección de superficies planas: RANSAC

Random Sample Consensus (RANSAC) es un algoritmo de ajuste de modelos cuyos resultados son comparables a los de una regresión lineal [FB81]. Dicho algoritmo permite modelizar un conjunto de datos observados entre los que se encuentran valores atípicos o *outliers*, es decir, que no encajan en el modelo. Los datos que sí encajan, o *inliers*, serán los que definan los hiperparámetros de dicho modelo.

Aplicando RANSAC a una nube de puntos tridimensionales, es posible segmentar dicha nube en grupos de puntos que compartan unas ciertas características como, por ejemplo, que pertenezcan a un mismo plano (ver Figura 19).

El algoritmo en sí mismo es relativamente simple y sigue los siguientes pasos:

1. Seleccionar tres puntos aleatorios de la nube. Dichos puntos conformarán un plano.
2. Calcular los parámetros de la ecuación que caracterizan a dicho plano.
3. Calcular la desviación del resto de puntos de la nube respecto al plano según su distancia.

4. Si la distancia de un punto está dentro de un determinado límite, añadir dicho punto como *inlier*.

5. Almacenar los puntos pertenecientes a los planos con mayor número de *inliers* y descartar el resto.

6. Repetir el proceso hasta alcanzar el número de iteraciones máximo.

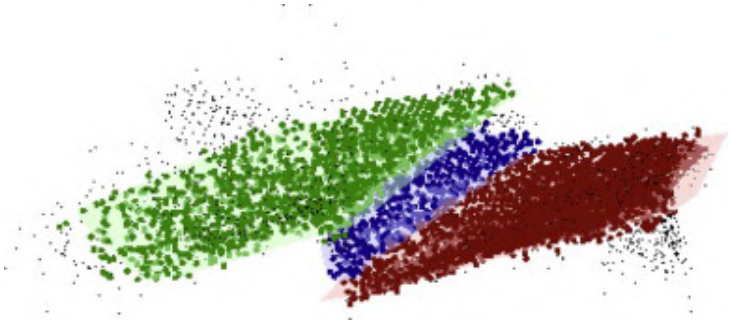


Figura 19. Ajuste de planos en una nube de puntos mediante el algoritmo RANSAC [GMR11].

Capítulo 3

Diseño de la solución

Este capítulo pretende desarrollar y analizar las diferentes vías que se han seguido a lo largo del avance del proyecto para lograr los objetivos necesarios en cada uno de los pasos en el análisis de los datos de profundidad, detección de la pieza a agarrar, y propuesta del agarre.

Como puede verse en la Figura 20, el punto de partida del proyecto es un robot IRB 14000 (YuMi) de ABB frente al cual se dispone una superficie plana en la que se volcarán los diferentes objetos a agarrar. Los objetos que serán utilizados en el proyecto se presentan en la Figura 21 forman parte de un conjunto de piezas automovilísticas y será detallado en la Sección 3.2.3. Cabe destacar que el robot utilizado en este proyecto es "ambidiestro", es decir, posee dos brazos independientes. De esta forma, se pretende poder agarrar objetos bien con una pinza paralela, bien con una ventosa de succión (ver Figura 22), en función de las características del objeto en cuestión.

Figura 20. Configuración del robot YuMi del laboratorio.

Figura 21. Piezas pertenecientes al conjunto de datos propio.

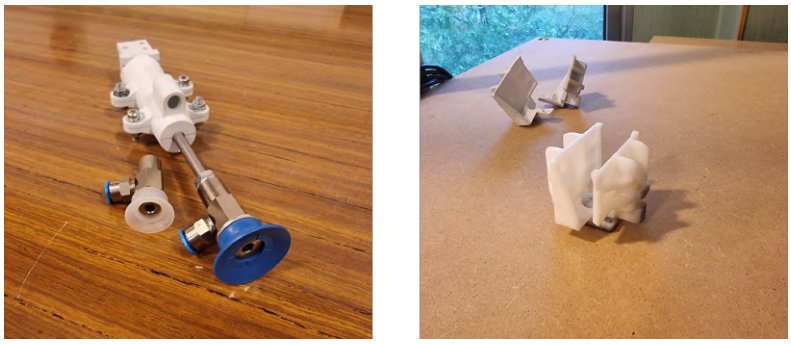


Figura 22. *Grippers* utilizados sobre el robot YuMi: pinza paralela (izquierda), y ventosa de succión (derecha).

De entre todas las tecnologías para la percepción de la profundidad presentadas hasta el momento en el Estado del arte, las utilizadas en este proyecto serán las cámaras de visión estereoscópica y LiDAR, debido principalmente a su facilidad de uso así como a su gran precisión a corto alcance, ofreciendo un margen de error del orden de milímetros a distancias inferiores a un metro. Un ejemplo de imagen tomada con estas cámaras se muestra en la Figura 23.

Figura 23. Ejemplo de imagen RGB-D de la cámara Intel D435: de color o RGB (izquierda), de profundidad valores netos de distancia normalizados (centro), y de profundidad con mapa de color (derecha).

Una vez se es capaz de medir de forma adecuada la profundidad de una escena, el siguiente paso es analizar la información captada para, posteriormente, llevar a cabo la detección de los objetos que el robot pueda encontrarse. Para ello, se exploran: (1) algoritmos tradicionales, tales como la sustracción de fondo, detallada en la Sección 3.2.2; y (2) algoritmos más modernos que hagan uso de la inteligencia artificial, tales como YOLO, detallado en la Sección 3.2.3.

Así pues, con el objeto que se desea coger localizado, se decide qué tipo de agarre se desea realizar:

- A través de una pinza paralela, adecuada para objetos con forma paralelepípeda y de tamaño inferior a la máxima apertura de esta.
- A través de una ventosa de succión que obtiene el caudal de aspiración gracias a un tubo de Venturi, alimentado a su vez por un compresor.

Dado que el conjunto de piezas objetivo es cerrado y conocido, el tipo de agarre que se implantará en cada una (agarre paralelo o ventosa) será definido de antemano en función de su geometría y regiones óptimas de agarre. En caso de decidir utilizar la ventosa, se lleva a cabo una detección de superficies planas sobre el objeto mediante el algoritmo de RANSAC (desarrollado en la Sección 3.4) para determinar las posibles zonas donde esta podría adherirse; por otro lado, en aquellas piezas paralelepípedas en las que se precise un agarre paralelo se utilizará una detección de contornos clásica.

3.1. Estudio de cámaras RGB-D: Intel D435 y L515

El término cámara RGB-D representa una cámara capaz de captar tanto información de color de la escena, como de profundidad. De esta forma, esta doble cámara ofrece dos puntos de vista diferentes de una misma escena de forma simultánea: una imagen a color (RGB), y una imagen de profundidad (*Depth*). Cabe destacar que esta imagen de profundidad no es más que una matriz bidimensional en la cual los valores de cada uno de sus elementos representan la distancia de cada punto de la escena al dispositivo.

El proyecto comenzará haciendo uso de la cámara de profundidad estereoscópica Intel RealSense D435, la cual ofrece un rango de medición de profundidad de entre 0,2 y 10 metros, una resolución máxima de 1920 x 1080 píxeles en la imagen RGB y de 1280 x 720 en la de profundidad, y una precisión de entre 2,5 y 5 milímetros a 1 metro de distancia [Int]. Sin embargo, conforme el proyecto avance, se observa que los resultados en las medidas de profundidad de la D435 son muy ruidosos y poco precisos. Por esta razón, se transicionará hacia el modelo Intel L515, más moderno, y que incorpora un módulo LiDAR en lugar de uno de cámaras

estereoscópicas, ofreciendo un rango de medición de profundidad de entre 0,25 y 9 metros, una precisión en la medida de profundidad de 5 milímetros a 1 metro de distancia, y una resolución máxima de 1920 x 1080 y de 1024 x 768 píxeles para las imágenes RGB y de profundidad, respectivamente. Ambas cámaras se muestran en la Figura 24.

Figura 24. Cámaras RGB-D utilizadas en el proyecto. A la izquierda, el modelo Intel RealSense D435 (estereoscópica); a la derecha, el modelo L515 (LiDAR).

3.1.1. Calibración de las imágenes RGB y de profundidad

La fotogrametría se basa en la perspectiva cónica a la hora de realizar medidas sobre las imágenes. Sin embargo, es habitual observar que la posición de los puntos en la imagen difiere ligeramente de su posición en el mundo real, debido a una serie de factores [RM99]:

1. Distorsiones procedentes del objetivo de la cámara.
2. Falta de planeidad de la película.
3. Condiciones de toma.

El origen de estas distorsiones se encuentra en las mínimas aberraciones que sufren las lentes de las cámaras que, como consecuencia, hacen que la perspectiva de la imagen no sea perfectamente cónica.

Para subsanar dichas distorsiones, se hace uso de diferentes parámetros que permiten corregir matemáticamente la percepción de la escena por parte de la cámara. Se considera que una cámara es métrica si tanto su geometría interna como las características de distorsión del objetivo son repetibles, estables y susceptibles de ser modeladas mediante parámetros que puedan estimarse con precisión a través de un proceso de calibración fotogramétrico [LBJB12]. Estos parámetros, sin embargo, no siempre se encuentran disponibles, en especial en cámaras no métricas, es decir, aquellas que poseen una geometría inestable y que, por lo tanto, tienen unos parámetros de orientación interna desconocidos. De esta forma, si se quiere utilizar una cámara no métrica

para medir profundidades, es necesario determinar previamente sus características internas a través de un proceso de calibración que permita estimar los parámetros de un modelo de cámara estenopeica (*pinhole camera*) que capturase una imagen equivalente a la imagen real. A partir de entonces, dicha cámara puede considerarse como *semimétrica*: de parámetros teóricos desconocidos, pero aproximados experimentalmente [RM99].

Parámetros intrínsecos de la cámara RGB (*intrinsics*)

Para a comprender los ajustes que se llevarán a cabo más adelante en el proceso de calibración, es necesario entender los de dónde provienen los parámetros intrínsecos de una cámara.

Sea P un punto cualquiera de la escena, f la distancia focal de la lente de la cámara y C su origen, una escena puede ser convertida en una imagen, tal y como se representa en la Figura 25, donde el plano amarillo representa la imagen resultante, cuyo tamaño es HxW y cuyo sistema de coordenadas es (u, v). De esta forma, un punto P de la escena de coordenadas (X, Y, Z) que halla su homólogo $p = (x, y, z)$ en la imagen cumple que:

$$\frac{x}{f} = \frac{X}{Z} \qquad \frac{y}{f} = \frac{Y}{Z} \tag{1}$$

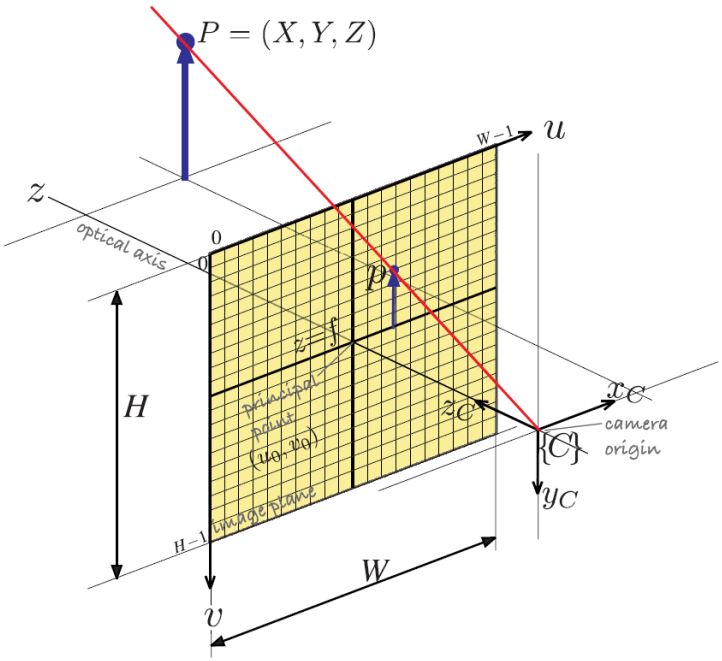


Figura 25. Visualización de los parámetros intrínsecos en una cámara con modelo estenopeico [Cor17].

Así pues, sean ρ_ω y ρ_h la anchura y altura en píxeles de la imagen, obtenemos que las coordenadas (u, v) del punto p se expresan como:

$$u = u_0 + \frac{x}{\rho_\omega} = u_0 + \frac{f}{\rho_\omega} \cdot \frac{X}{Z} \qquad\qquad v = v_0 + \frac{y}{\rho_h} = v_0 + \frac{f}{\rho_h} \cdot \frac{Y}{Z} \tag{2}$$

donde f/ρ_ω y f/ρ_h representan la distancia focal expresada en píxeles. Estas ecuaciones pueden reescribirse en forma matricial, obteniendo:

$$\begin{pmatrix} \lambda_u \\ \lambda_v \\ \lambda \end{pmatrix} = \begin{pmatrix} f/\rho_\omega & 0 & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} = K \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} \qquad \lambda = Z_C \tag{3}$$

donde la matriz K es conocida como la matriz de parámetros intrínsecos.

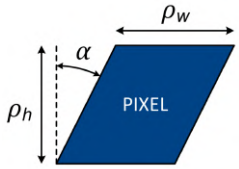


Figura 26. Ejemplo de píxel inclinado.

En caso de que los píxeles estuvieran inclinados, como muestra la Figura 26, la matriz K resultaría más compleja:

$$K = \begin{pmatrix} f/\rho_\omega & s & u_0 \\ 0 & f/\rho_h & v_0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} f_u & f_v \tan\alpha & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

En caso de tener píxeles cuadrados y no inclinados, la matriz se simplifica:

$$K = \begin{pmatrix} f/\rho & 0 & u_0 \\ 0 & f/\rho & v_0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} f_p & 0 & u_0 \\ 0 & f_p & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

La obtención de estos parámetros permitirá corregir las diferentes deformaciones y distorsiones que pueda tener la imagen de salida, como podrían ser la tangencial o la radial (ver Figura 27), al poseer la cámara una lente gran angular. Este aspecto será de vital importancia en el proyecto, ya que será precisamente esta la imagen sobre la que se realizará el procesado para

determinar cómo llevar a cabo un agarre sobre la pieza objetivo.

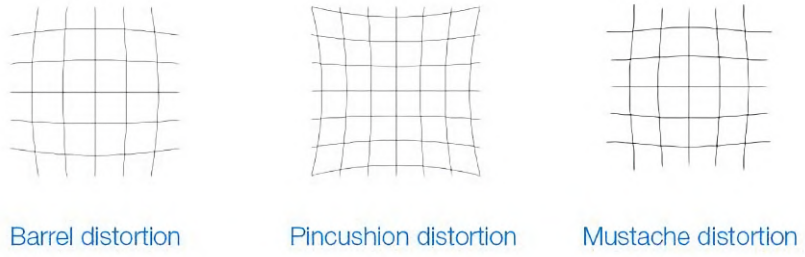


Figura 27. Ejemplos de distorsión radial en una imagen: de barril, de cojín y de bigote [Fri].

Proceso de calibración

Para paliar los efectos de la distorsión en la imagen a color, se procede a estimar los parámetros intrínsecos y de distorsión de las cámaras RGB a través del método del damero, descrito en la documentación oficial de la librería de Python OpenCV [Ope].

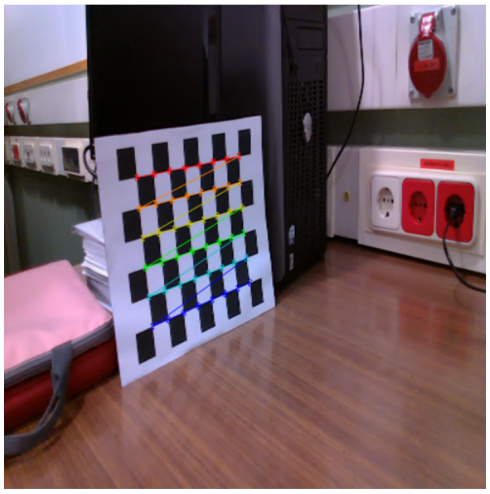


Figura 28. Detección de los puntos característicos en un damero para la calibración de la cámara Intel RealSense L515 sobre una versión distorsionada de la imagen de entrada durante el proceso de calibración.

Este procedimiento se basa en el algoritmo de Zhang [Bur16], el cual busca correspondencias entre las coordenadas de puntos característicos de un objeto (normalmente las esquinas de un damero) en la imagen y en la escena de forma iterativa (ver Figura 28). El procedimiento, aunque lento, es realizado *offline* (es decir, no en tiempo real) una única vez, y tiene los siguientes pasos:

1. Imprimir un damero sobre una superficie completamente plana.

2. Fotografiar el damero al menos entre diez y veinte veces desde ángulos, orientaciones y perspectivas diferentes, moviendo bien el damero, bien la cámara.

3. Detectar los puntos característicos (esquinas del damero) en la imagen, conocidas las dimensiones de este (número de filas, número de columnas, y ancho de cuadrado), sobre las diferentes distorsiones que se le aplican a las imágenes de entrada durante el proceso de calibración.

4. Estimar los parámetros intrínsecos y extrínsecos de la cámara asumiendo coeficientes de distorsión nulos.

5. Estimar el resto de parámetros simultáneamente, incluyendo los de distorsión, a través de una minimización por mínimos cuadrados no lineales (algoritmo Levenberg-Marquardt) [Wyn59].

Para visualizar los efectos de la calibración de la cámara, se proyecta un sistema de ejes cartesianos en realidad aumentada sobre un marcador ArUco. La Figura 29 muestra la proyección de la terna antes y después de la calibración de la cámara. Como se puede observar, el sistema de referencia obtenido tras estimar los *intrinsics* a través de la calibración ofrece mayor ortogonalidad que en el caso original.

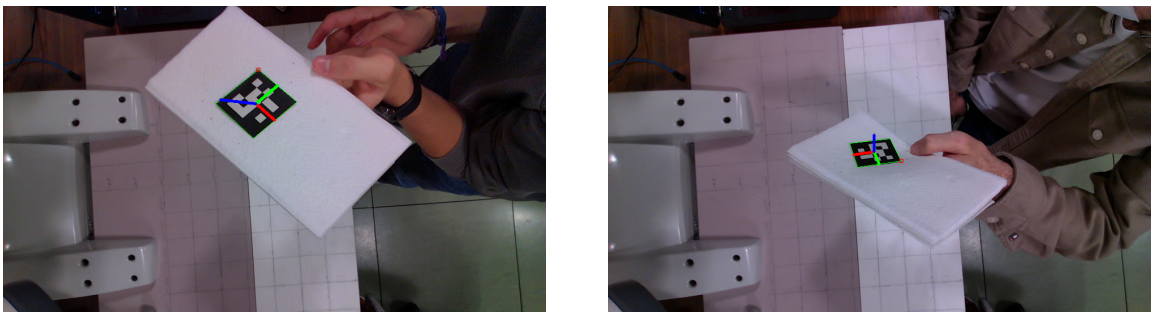


Figura 29. Proyección de unos ejes cartesianos sobre un marcador ArUco previa calibración de la cámara (izquierda), y posterior a ella (derecha), para la cámara Intel RealSense L515.

Cabe destacar que aunque la librería oficial de Intel para Python, pyrealsense2, ofrece la posibilidad de obtener los parámetros intrínsecos teóricos de las cámaras, en este caso fue necesario utilizar el método del damero, ya que los resultados obtenidos con los parámetros teóricos no eran satisfactorios.

En lo que respecta a la calibración de la imagen de profundidad, se ha utilizado el proceso de calibrado automático proporcionado por Intel (*on-chip Self-Calibration*), y que puede realizarse fácilmente a través de su programa Intel RealSense Viewer [GJSK+20].

3.1.2. Análisis de consistencia en mediciones consecutivas de profundidad

Durante el desarrollo de las pruebas de medición de profundidad se observa que las medidas proporcionadas por la cámara Intel D435 parecen no ser consistentes a lo largo del tiempo.

Para verificar esta problemática, se lleva a cabo un experimento en el que, partiendo de un cubo de goma, se superpone en tiempo real sobre la imagen a color una máscara roja que representa los puntos de mínima distancia a la cámara z_{min} (o máxima altura del cubo) dentro de una cierta tolerancia ε (Figura 30). La máscara recoge, por tanto, todos los puntos cuya distancia z cumpla que:

$$z \in [z_{min} - \varepsilon, \quad z_{min} + \varepsilon] \tag{6}$$

Dado que las superficies del cubo son completamente planas, se espera que dicha máscara coincida de forma cuasi exacta con la superficie superior del cubo; en cambio, los resultados son los mostrados en la Figura 30. Se puede observar que la máscara, en lugar de cubrir la totalidad de la superficie superior de los cubos, lo hace de forma parcial.

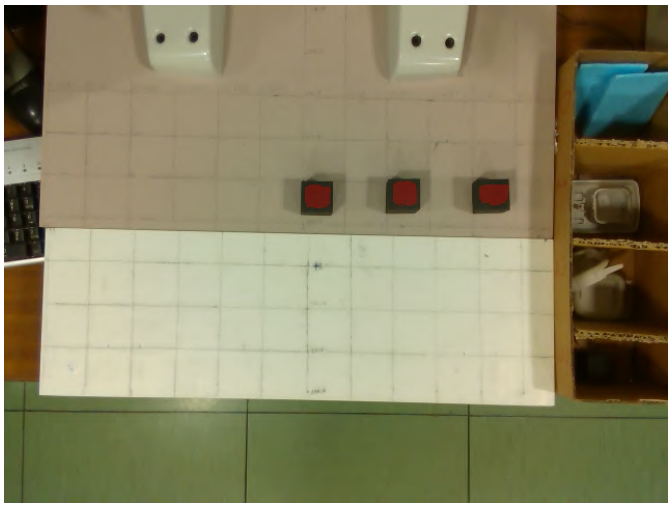


Figura 30. Proyección de una máscara de los puntos de distancia mínima a la cámara en la cara superior de un cubo.

De cara a cuantizar la necesidad de una cámara de mayor calidad que la actual, se procede a realizar un análisis de robustez en medidas consecutivas para un mismo punto. Este análisis compara, por un lado, la cámara original estereoscópica Intel D435; y por otro, una cámara más moderna y que hace uso de tecnología LiDAR, la Intel L515. Para ello, se toman 100 muestras del mismo punto central de la imagen de profundidad frente a una superficie plana a 735 milímetros de distancia, activando y desactivando el pipeline de la cámara entre tomas.

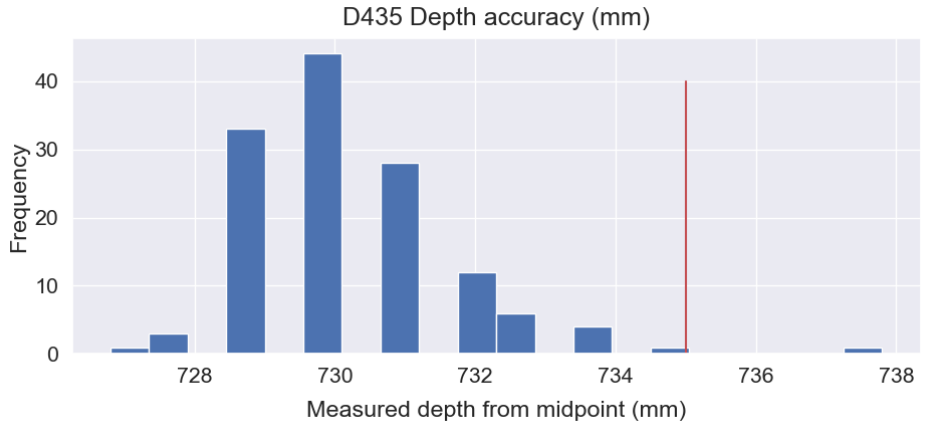


Figura 31. Distribución de medidas consecutivas de un mismo punto para la cámara Intel D435 a una distancia real de 735 mm.

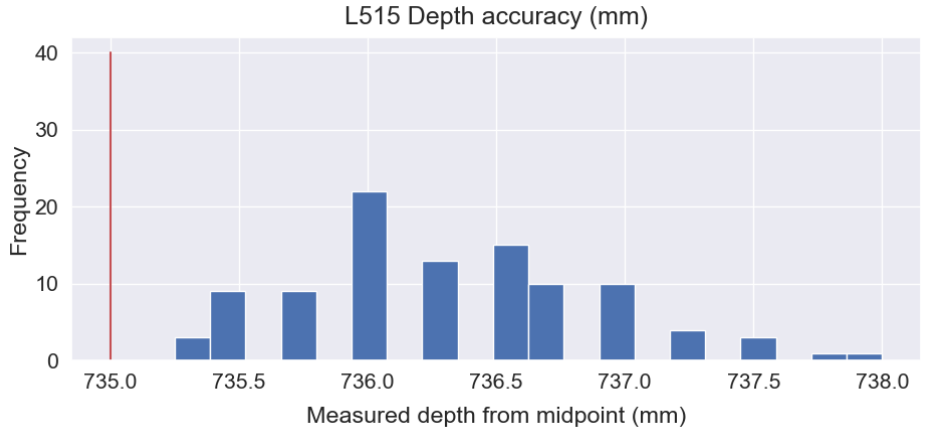


Figura 32. Distribución de medidas consecutivas de un mismo punto para la cámara Intel L515 a una distancia real de 735 mm.

Las figuras 31 y 32 muestran a través de un histograma la distribución de las 100 tomas obtenidas con las cámaras D435 y L515, respectivamente, donde la línea vertical roja representa la distancia real de 735 mm que deberían medir las cámaras; el eje x, los resultados de las mediciones realizadas durante las 100 tomas; y el eje y, la frecuencia de aparición de cada medida. Como se puede observar, la distribución de las mediciones del modelo D435 tiene una mayor variabilidad, estando comprendidas entre los 727 y los 735 mm. Las medidas del modelo L515, por su parte, tienen una variabilidad menor, y están comprendidas entre 735 y 738 mm, por lo que sus resultados son mucho más consistentes que los ofrecidos por la D435.

Un análisis en mayor profundidad puede encontrarse en la Figura 33 y en la Tabla 1. Como se puede ver, no solo es la desviación estándar del modelo L515 la mitad de la del D435, sino que, además, su error medio es casi cuatro veces más pequeño. Asimismo, como se puede observar en la figura, el modelo D435 posee mayor número de mediciones inconsistentes (mostradas

como *outliers*) que el L515, ofreciendo por tanto este último un rango de precisión en la medida inferior a los 3 milímetros frente a los 10 del modelo D435.

	Error medio	*Desv. estándar*	*Mediana*	*Mín*	*Máx*	*Rango de precisión*
Intel D435	-4.66	1.60	-5.20	-7.20	2.80	10.00
Intel L515	1.33	0.59	1.25	0.25	3.00	2.75

Cuadro 1. Estadísticas de error (en mm) respecto a una distancia fija conocida para cada conjunto de cien muestras por cámara.

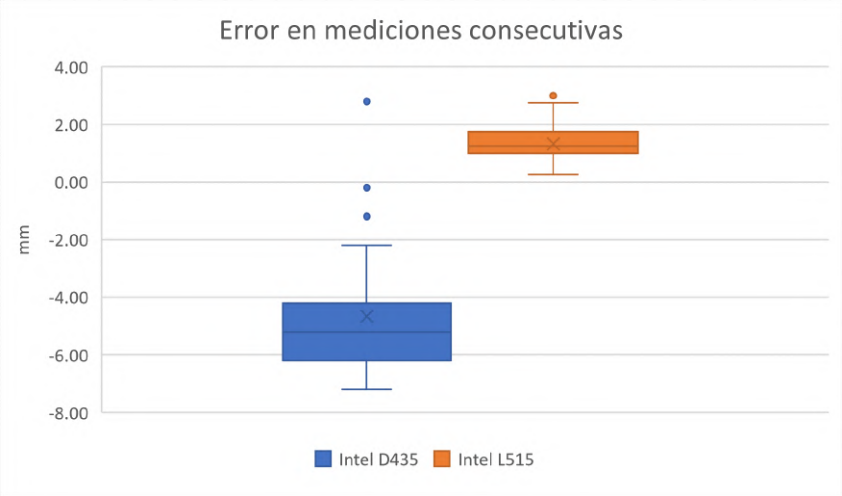


Figura 33. Boxplot del error en la toma de medidas consecutivas.

3.1.3. Análisis de deformación

Como se verá más adelante, una de las formas utilizadas para encontrar el punto terminal (en inglés, *Terminal Point* o TP) objetivo al que debe llegar el brazo robótico consiste en detectar la superficie superior del cubo y calcular las coordenadas (x, y) de su centroide. Estas coordenadas, junto con su medición de profundidad z correspondiente, proporcionan un punto situado en el espacio y definido en las tres dimensiones $(x, y, z)_{TP}$ sobre el que posteriormente se puede especificar la orientación con la que se desea que llegue el gripper (que, en el caso de cubo, será la pinza paralela).

Durante el análisis de consistencia en medidas consecutivas de la sección anterior se observa también que con la cámara D435, cuanto más alejado se encuentra el cubo del centro de la imagen, mayor es error cometido por el robot YuMi al intentar agarrar el cubo, chocando por tanto contra él en lugar de agarrarlo con éxito.

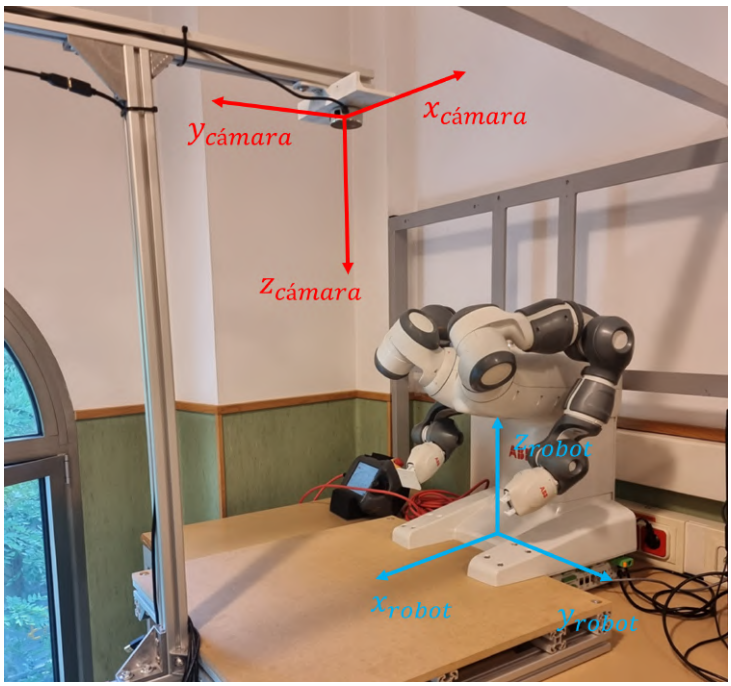


Figura 34. Sistema de ejes de coordenadas del robot YuMi y de la cámara RGB-D.

Con el objetivo de verificar si este error es debido a mediciones por parte del módulo de la cámara RGB-D o a otro módulo externo (por ejemplo, falta de precisión del soporte de la cámara, algoritmo de cinemática del robot, o error en la conversión entre el sistema de coordenadas de la cámara y el del robot), se realiza un estudio sobre la deformación de la imagen en el eje y_{camara} (ver Figura 34). Para ello, se dispone sobre la mesa de trabajo un conjunto de cubos equidistantes los unos de los otros horizontalmente, en los cuales se habrá marcado previamente el centro de su plano superior, y que serán desplazados progresivamente a lo largo del eje X de cara a caracterizar la totalidad de la superficie de trabajo, como puede verse en la Figura 35.

Con esta configuración, se desarrolla un script en Python que permite tomar fotografías RGB-D en planta de la escena, detectar las superficies superiores de los cubos, y comparar sus respectivos centroides con sus centros reales, señalados en el entorno físico de antemano. Un ejemplo del resultado de este proceso puede verse en la Figura 36. El método utilizado para detectar la superficie de cada cubo es el mencionado anteriormente en el Análisis de consistencia en mediciones consecutivas de profundidad, donde se obtiene los puntos cercanos a aquel de mínima distancia z_{min} (correspondiente al plano superior) con una cierta tolerancia ε.

Con los centroides calculados de forma experimental, y conociendo los centros reales para cada uno de los cubos, es posible calcular la diferencia entre estos puntos en cada una de las distintas posiciones del plano de trabajo. Dicho error se muestra en la Figura 37, siendo

directamente proporcional al tamaño de la burbuja.

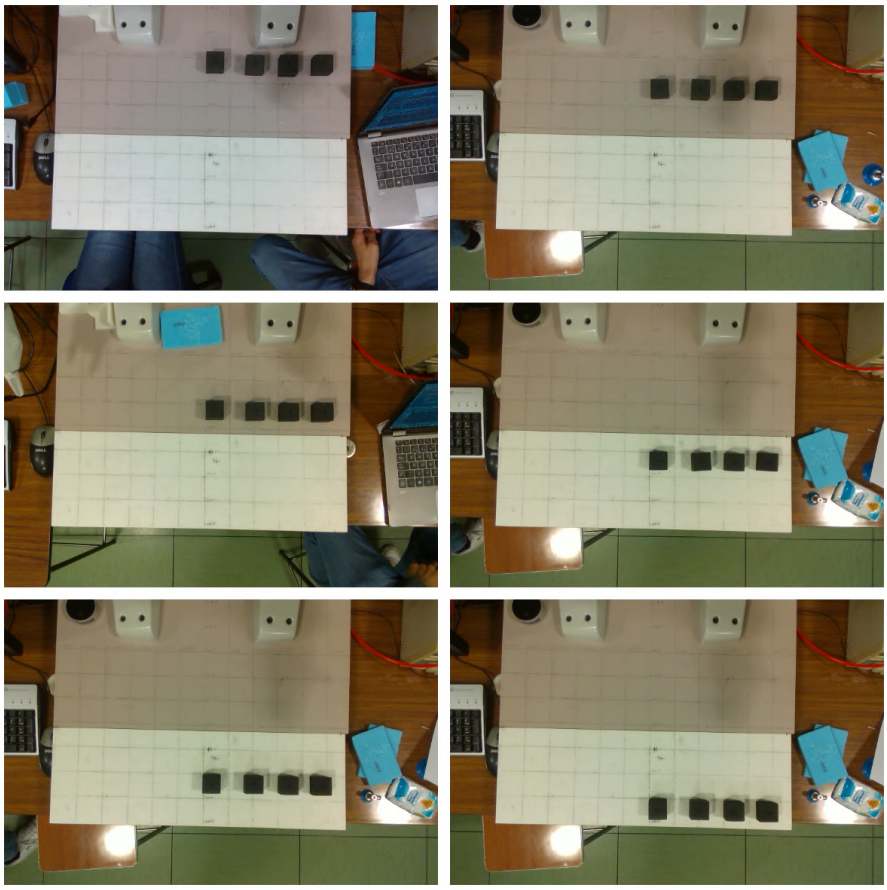


Figura 35. Distribución de los cubos de gomaespuma en la mesa de trabajo durante el análisis de la deformación del eje Y.

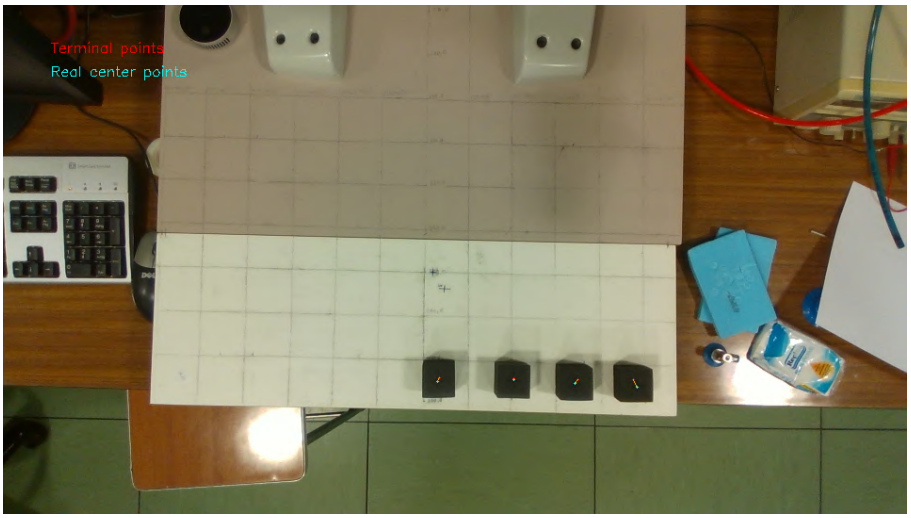


Figura 36. Visualización del error entre los centroides de las caras superiores de los cubos (en rojo) y los reales (en azul) para una coordenada $y = 450\ mm$.

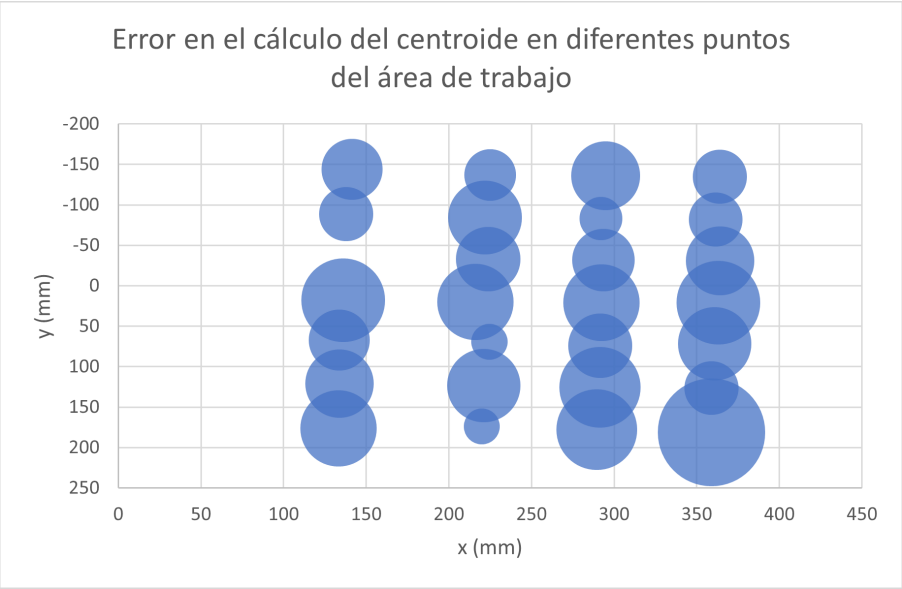


Figura 37. Error en el cálculo del centroide en diferentes puntos del área de trabajo. A mayor error, mayor tamaño de burbuja.

Como se puede observar en la figura, el error en el cálculo de los centroides no sigue ningún patrón aparente, sino que es uniforme por toda la superficie. Por esta razón, se descarta el módulo de la cámara RGB-D como causante de la imprecisión del brazo robótico.

3.1.4. Aplicación de filtros y posprocesado de la imagen de profundidad

Como ya se ha comentado anteriormente, la imagen de profundidad ofrecida por la cámara RGB-D es una imagen en escala de grises en la que cada elemento, cada píxel, representa la distancia a cada punto de la escena. No obstante, un problema que puede surgir es la aparición de imperfecciones en la medición de distancia y, por lo tanto, en la imagen resultante. Para compensar estas imperfecciones, la librería oficial de Intel RealSense, *pyrealsense2* [SDK21], ofrece una serie de filtros que permiten reducir los niveles de ruido en la imagen, mejorando así la calidad de la información de profundidad.

A alto nivel, algunos de estos filtros son:

- **Filtro de diezmado**: disminuye la complejidad de la escena al calcular la mediana de los valores de profundidad de cada grupo de píxeles. Como consecuencia, sus dimensiones se reducen y, por lo tanto, la imagen es comprimida. Asimismo, este filtro incluye un cierto grado de rellenado de huecos, ya que la compresión no hace uso de los valores nulos de la matriz.

- **Filtro espacial de conservación de bordes**: realiza un suavizado general de la imagen, pero sin perder detalles en los bordes. Basado en el artículo publicado por Eduardo S. L. Gastal y Manuel M. Oliveira [GO11].
- **Filtro temporal**: mejora la consistencia de las medidas entre fotogramas. Para ello, utiliza el historial de tomas anteriores, logrando reducir el parpadeo.
- **Rellenado de huecos**: rectifica aquellos datos de la imagen de profundidad que sean inexistentes. Según el método elegido, estos pueden rellenarse según el valor del píxel de su izquierda, según el mayor valor de profundidad de sus píxeles vecinos, o según el menor.

Aplicando todos estos filtros de forma simultánea obtenemos una imagen como la mostrada en la Figura 38. Como se puede observar, se obtiene una imagen que, aun estando más suavizada, ofrece una representación más consistente y con menor ruido y huecos (píxeles negros) que la original. Cabe destacar que, en el caso específico del modelo D435, existe una banda en el lateral izquierdo de la imagen de profundidad que, debido a la naturaleza estereoscópica de la cámara, da lugar a una franja ciega en la cual la cámara es incapaz de medir. Una posible solución a este problema podría ser rellenar estos valores según los valores de los píxeles cercanos, o incluso utilizar una segunda cámara de profundidad situada en una perspectiva diferente, complementaria a la primera.

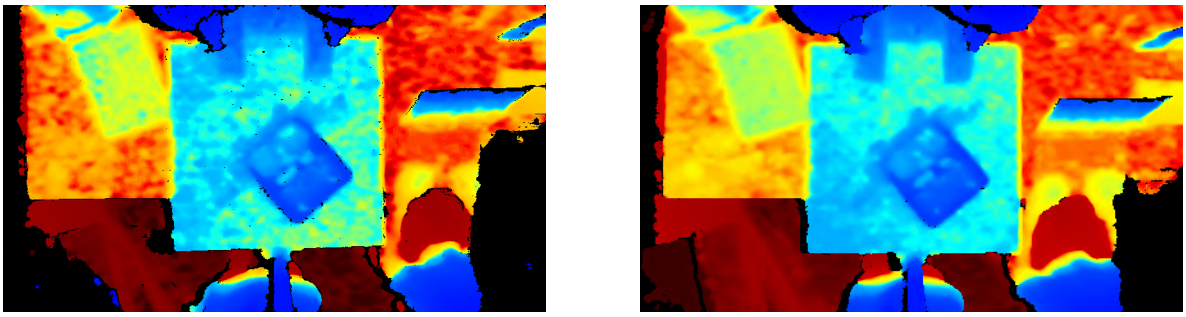


Figura 38. Comparación de una imagen de profundidad original sin filtros (izquierda) frente a una imagen de profundidad posprocesada con filtros (derecha), ambas obtenidas con la cámara D435.

Un punto importante a considerar cuando se utiliza una cámara RGB-D es el hecho de que las perspectivas desde las cuales se toman las imágenes RGB y de profundidad no coinciden físicamente, lo que resulta en que un píxel de la imagen RGB no encuentra su valor correspondiente en la imagen de profundidad. Por esta razón, es necesario realizar una alineación de ambas imágenes antes de comenzar a utilizarlas, a través del siguiente extracto de código:

```
import numpy as np
import pyrealsense2 as rs
import matplotlib.pyplot as plt

# Create alignment primitive with color as its target stream:
align = rs.align(rs.stream.color)
frameset = align.process(frameset)

# Update color and depth frames:
aligned_depth_frame = frameset.get_depth_frame()
colorized_depth = np.asanyarray(colorizer.colorize(aligned_depth_frame).
    ↪ get_data())

# Show the two frames together:
images = np.hstack((color, colorized_depth))
plt.imshow(images)
```

Fragmento de código 3.1. Alineación de imagen RGB y de profundidad.

De esta forma, se obtienen dos imágenes alineadas, lo cual permite utilizar la información de profundidad como un canal más de la imagen, facilitando su manipulación a partir de la imagen a color.

3.1.5. Elección del modelo de cámara a utilizar

Con lo analizado en apartados anteriores, se valora la necesidad de transicionar de un modelo de cámara RGB-D estereoscópico, como la D435, a un modelo LiDAR, como la L515, cuyas muestras de imágenes de profundidad se presentan en la Figura 39.

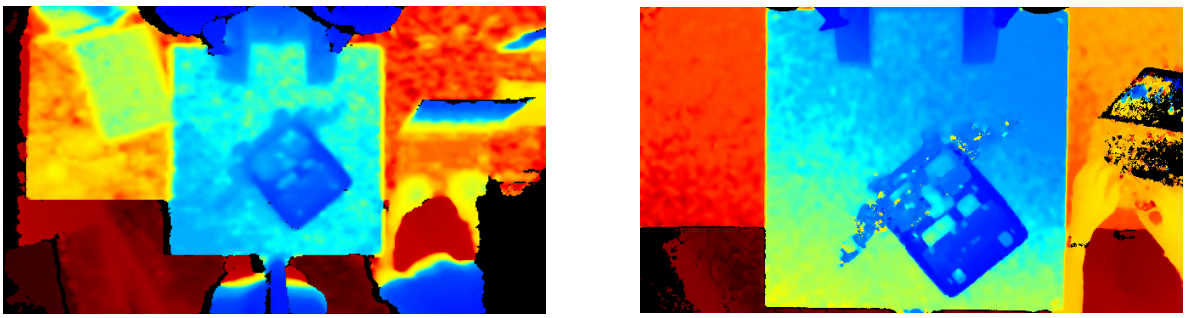


Figura 39. Imagen de profundidad con posprocesado de la cámara estereoscópica D435 (izquierda), y de la cámara LiDAR L515 (derecha).

Si bien la D435 ofrece resultados satisfactorios, su naturaleza estereoscópica produce ciertas zonas ciegas en la imagen donde no se dispone de información de profundidad, en especial en los bordes de los objetos. Asimismo, la variabilidad de 10 milímetros en la consistencia de las medidas de profundidad así como la existencia de *outliers* hallados en la Sección 3.1.2 terminan por ser demasiado grandes para la escala del proyecto, debido en especial a la existencia de piezas pequeñas en el conjunto de datos.

El modelo LiDAR L515, por su parte, ofrece una precisión experimental de unos 3 milímetros, muy superior a la del modelo estereoscópico. Asimismo, como puede verse en la Figura 32, la imagen obtenida por el modelo L515 ofrece unas medidas mucho más consistentes para puntos de distancia similar, con menor nivel de ruido, y sin zonas ciegas.

A la luz de estos resultados, se decide proseguir el trabajo utilizando el modelo de cámara LiDAR L515.

3.2. Proceso de detección de piezas

El objetivo del proyecto es hallar un agarre que permita a un brazo robótico coger un objeto que se encuentra delante suyo. Por lo tanto, una vez capaces de medir distancias a través de una cámara RGB-D, el siguiente paso lógico es lograr detectar el objeto en cuestión, del cual se obtendrán posteriormente las coordenadas que serán enviadas al robot YuMi.

Como se explica a continuación, a lo largo del desarrollo del proyecto se han explorado diferentes alternativas para lograr detectar el objeto que se desea agarrar: unas más tradicionales (máscara por segmentación de profundidad y por sustracción de fondo), y otra que utiliza algoritmos de inteligencia artificial (YOLO). Se pretende comparar las fortalezas y debilidades de cada una de ellas y elegir aquella que proporcione mejores resultados, siempre teniendo en cuenta el contexto en el cual se implantaría el proyecto, es decir, en un entorno industrial. El hecho de poseer una potencial aplicación en un entorno industrial tiene como consecuencia implicaciones tales como condiciones de iluminación variables, geometría de piezas conocidas, o alto nivel de cambio en el entorno.

3.2.1. Máscara por segmentación de profundidad

Una de las formas más sencillas y directas de aislar la pieza objetivo de la mesa de trabajo consiste en obtener la distancia a la que se encuentra la superficie de trabajo y considerar como pieza todo aquello que está por encima suyo. Dado que la cámara RGB-D se encuentra

fijada sobre el robot YuMi y es paralela al plano de trabajo, esta distancia puede considerarse constante e invariante entre tomas. Esto difiere de la configuración utilizada en los experimentos de Berkeley, en la cual la cámara poseía una inclinación de 30º respecto del plano horizontal [MMS+19], lo que imposibilitaba la exploración de esta técnica.

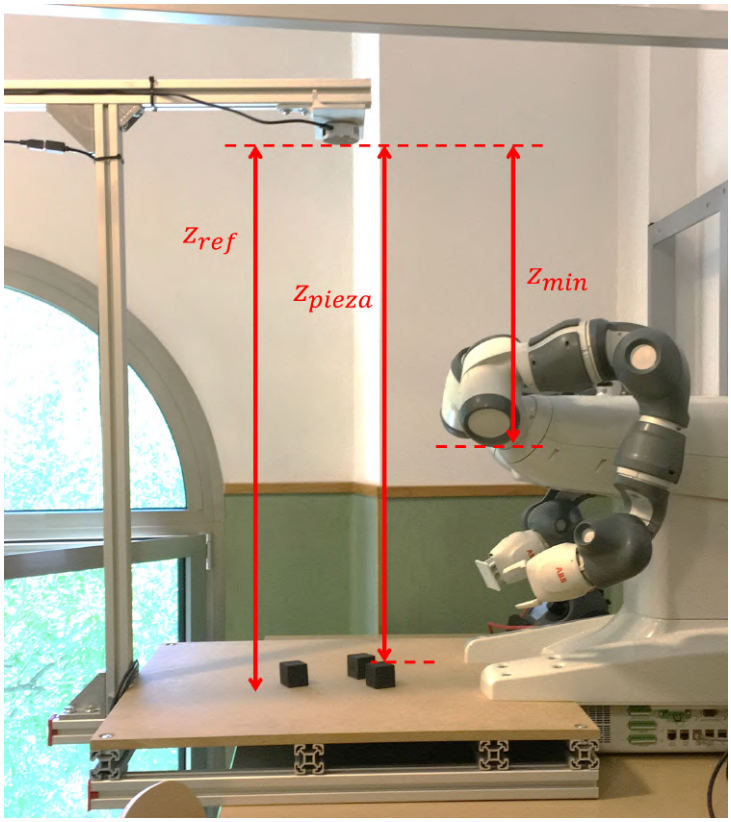


Figura 40. Definición de las dimensiones z_{ref}, z_{pieza} y z_min.

Así pues, partiendo de una configuración como la mostrada en la Figura 40, donde z_{ref} es la distancia de la cámara RGB-D al plano de trabajo y z_{pieza} es la distancia perteneciente cualquier punto de la pieza objetivo, es posible filtrar la imagen de profundidad de la escena capturada en un momento dado y crear una máscara binaria en la que el valor de cada punto queda determinado por la cota z_{ref}: los puntos de cota $z < z_{ref}$ serán de color blanco, y los puntos de cota $z > z_{ref}$, de color negro (ver Figura 41). Puesto que el valor de los elementos en la imagen de profundidad aumenta con la distancia al sensor de la cámara, se toma como válida para la máscara la nube de puntos cuya distancia sea inferior a la del plano de trabajo, es decir, los puntos de color blanco.

Además de la aplicación de la máscara de profundidad, se deben aplicar otros procesos a la imagen para obtener la región de interés, o ROI, que en este caso es la mesa de trabajo sobre la que se posa la pieza. De esta forma, se debe eliminar la zona asociada a los hombros del robot a través de una máscara binaria auxiliar. Así pues, sabiendo que una máscara binaria no es más

que una matriz de booleanos en la cual la zona blanca es considerada *verdadera*, y la negra *falsa*, es posible realizar una operación *and* lógica de la máscara de profundidad sobre la binaria, obteniendo la máscara final deseada, tal y como se muestra en la Figura 41.

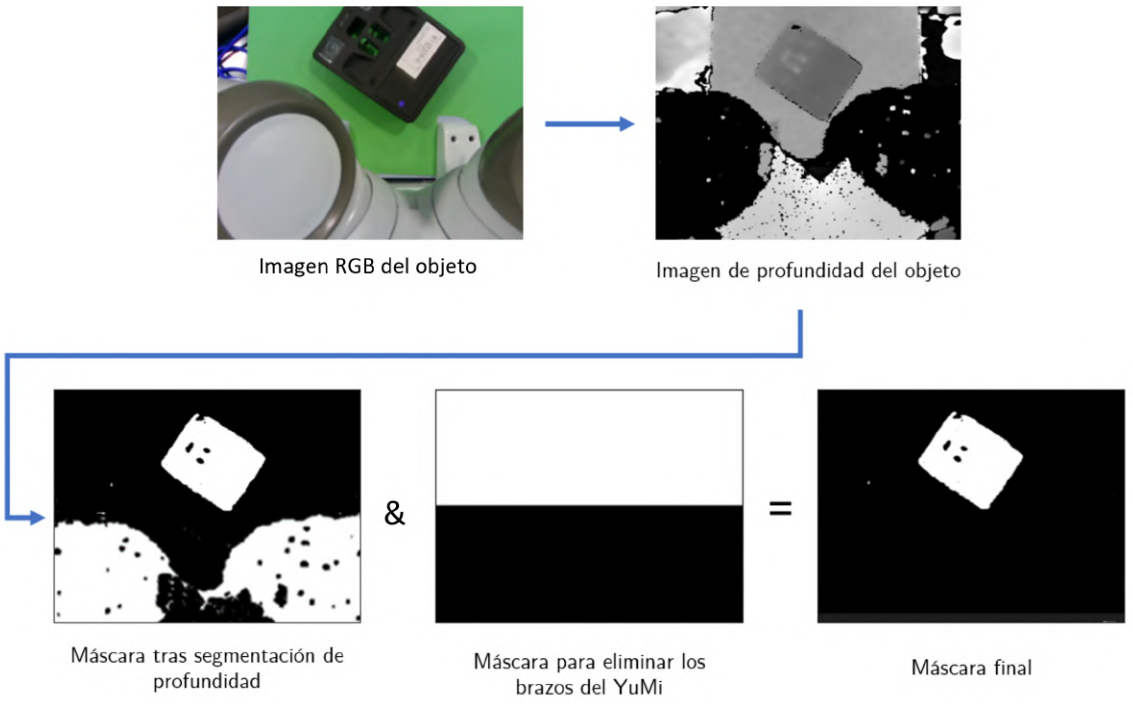


Figura 41. Proceso de obtención de máscara por segmentación de profundidad.

Cabe destacar que, aunque la máscara auxiliar es adecuada para eliminar los hombros del robot YuMi en ciertas ocasiones, resulta mucho más cómo y efectivo modificar el filtro de profundidad de forma que, en vez descartar únicamente los puntos cuya distancia z sea superior a z_{ref}, se imponga además una segunda restricción que también ignore aquellos puntos de distancia z menor a una distancia z_{min}, la cual será ligeramente superior a la distancia a la que se encuentran los hombros del robot. Al ser esta distancia constante, asegura que los hombros del robot nunca aparezcan en la máscara. De esta forma, se crea una franja de trabajo $z_{min} < z < z_{ref}$ (ver Figura 41) que será la que conforme la máscara final, equivalente en este caso a la obtenida anteriormente, pero mucho más flexible en su uso.

Los resultados obtenidos se muestran en la Figura 42. Debido a los 10 o 3 mm de precisión de las cámaras utilizadas calculados en la Sección 3.1.2, el error de medida es, en ocasiones, demasiado alto como para aislar con una precisión adecuada la pieza del plano de trabajo, lo cual conlleva en muchos casos altos niveles de ruido. Este efecto se acentúa especialmente en las piezas más planas o de menor tamaño debido a que la propia variabilidad en la medida puede

llegar a superar la altura total de la pieza.

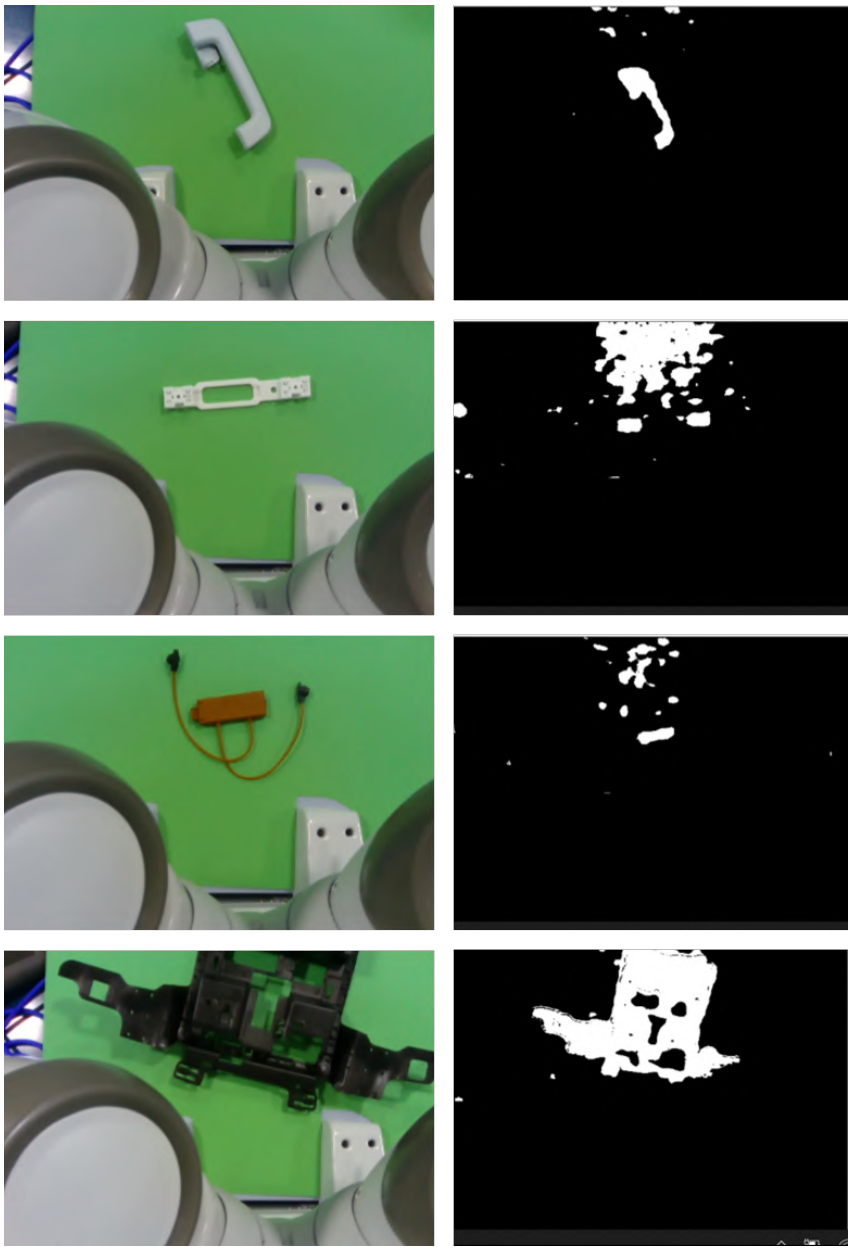


Figura 42. Ejemplos de máscaras binarias obtenidas por segmentación por profundidad (derecha) para una variedad de piezas (presentadas a través de su imagen RGB, izquierda).

3.2.2. Máscara por sustracción de fondo

La segunda técnica utilizada para la detección del objeto a agarrar sigue, en cierto modo, el mismo principio de funcionamiento que los llamados cromas (también conocidos como “fondos verdes”) en el ámbito audiovisual, donde se extrae un color de una imagen y se reemplaza el área previamente ocupada por ese color por otra imagen diferente, como se muestra en la Figura 43.

Figura 43. Ejemplo de uso de la técnica de croma en el ámbito cinematográfico [gor20].

En este caso, sin embargo, no será un color determinado lo que se eliminará de la imagen. La idea, tal y como se muestra en la Figura 44, es obtener una única imagen "base" de la mesa de trabajo en la que no aparezca ningún objeto para, posteriormente, sustraer a una nueva imagen capturada la imagen "base", obteniendo así una máscara de aquel objeto que se encuentre encima de la mesa. Esta máscara puede entonces ser posprocesada y limpiada para eliminar el posible ruido que pueda aparecer. Aunque esta metodología puede aplicarse tanto sobre la imagen RGB como sobre la imagen de profundidad, se elige utilizar la imagen a color por ser menos ruidosa y consistente en el tiempo.

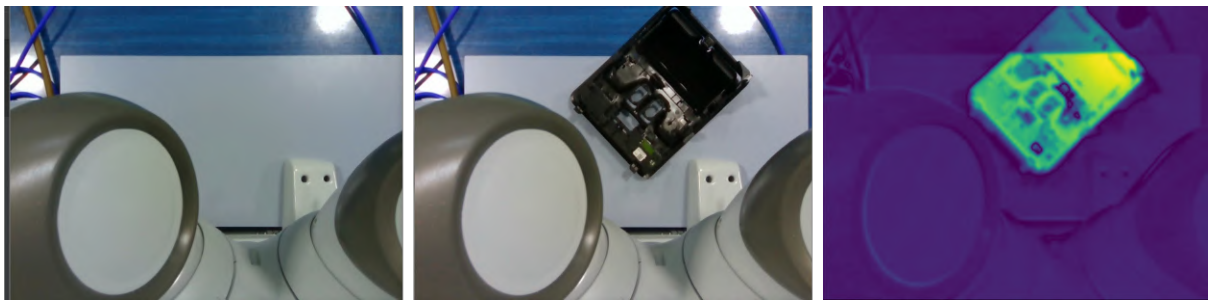


Figura 44. Primera parte del proceso de sustracción de fondo: imagen base, imagen con el objeto, y diferencia entre ambas imágenes.

La toma de la imagen a color base se realiza una única vez y de forma *offline*, es decir, no en tiempo real, ya que se supone una escena estática en la que el único cambio entre tomas será la presencia del objeto a agarrar. Tras guardar esta imagen, se procede a activar la cámara RGB en tiempo real. Dado que una imagen RGB no es más que una matriz tridimensional con valores para cada uno de los canales rojo, verde y azul, es posible restar las matrices que conforman las imágenes del objeto y del fondo y obtener una imagen *diferencia* entre ambas, como se muestra

en la Figura 44.

De cara a reducir el tiempo de computación, y dado que la información contenida en una imagen puede ser simplificada en una única matriz bidimensional y, por tanto, en un solo canal, la diferencia entre las imágenes fondo y con el objeto se realizará en escala de grises.

Dado que lo que se pretende en última instancia es aislar el objeto en su totalidad, la imagen *diferencia* obtenida puede ser convertida en una máscara binaria en blanco y negro según si los valores de cada uno de sus elementos superan o no un cierto umbral especificado de antemano. Es importante, además, realizar una dilatación de esta máscara de cara a reducir el nivel de ruido que pueda tener. Por último, se calculan todos los contornos presentes en la máscara dilatada y se filtra aquel que tenga una mayor superficie, eliminando así las manchas (*blobs*) que puedan aparecer en la máscara debido a fluctuaciones en la percepción del color de la cámara RGB. El proceso completo de obtención de máscara por sustracción de fondo puede verse en la Figura 45.

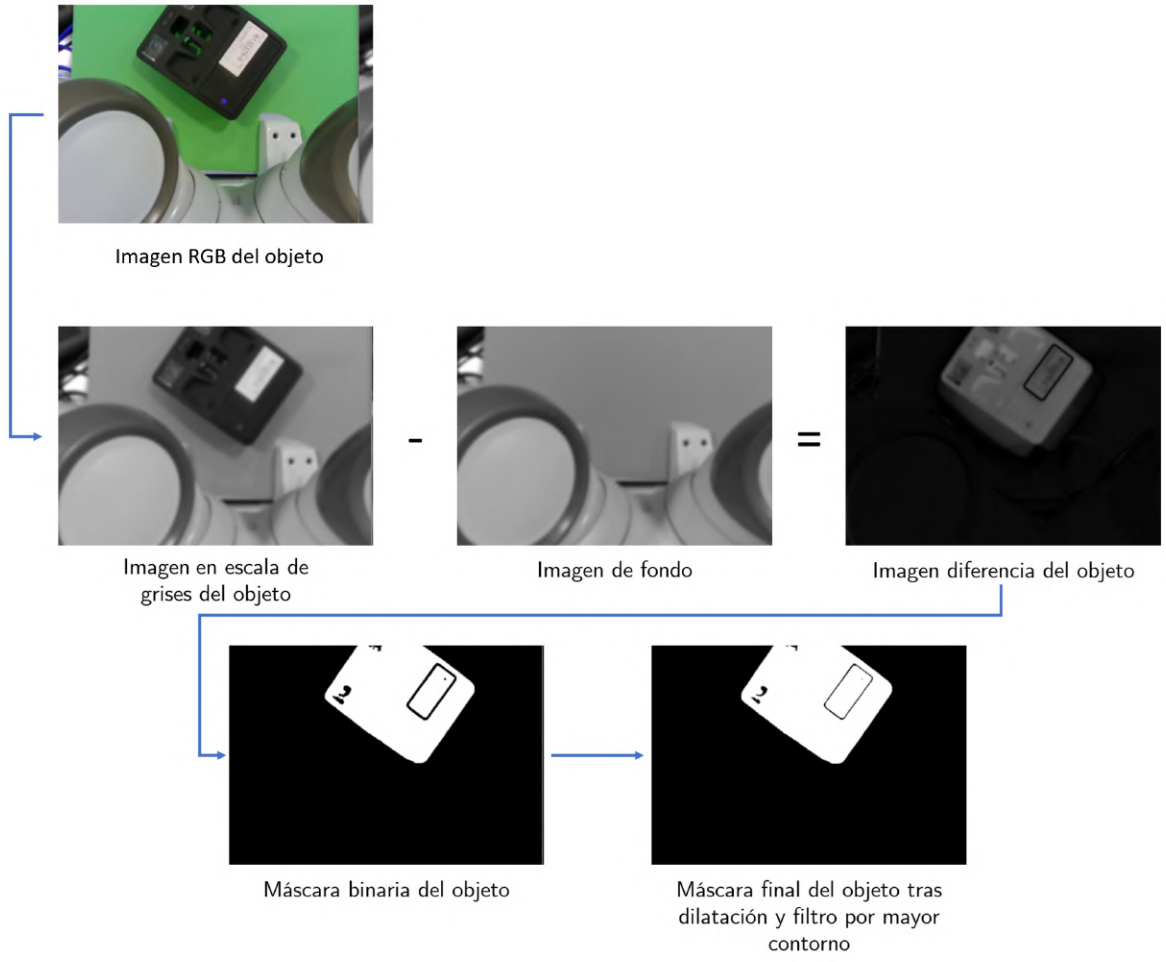


Figura 45. Proceso de obtención de máscara por sustracción de fondo.

Los resultados obtenidos son muy satisfactorios y se muestran en la Figura 46. Una vez ajustado el umbral, la máscara proporcionada por el algoritmo es casi impecable para una gran variedad de objetos, indistintamente de su forma, tamaño o complejidad, y funcionando de forma muy adecuada incluso en casos extremadamente complicados para el procedimiento de enmascaramiento por segmentación de profundidad, tal y como se mostró previamente en la Figura 42, como es el caso de los cables finos.

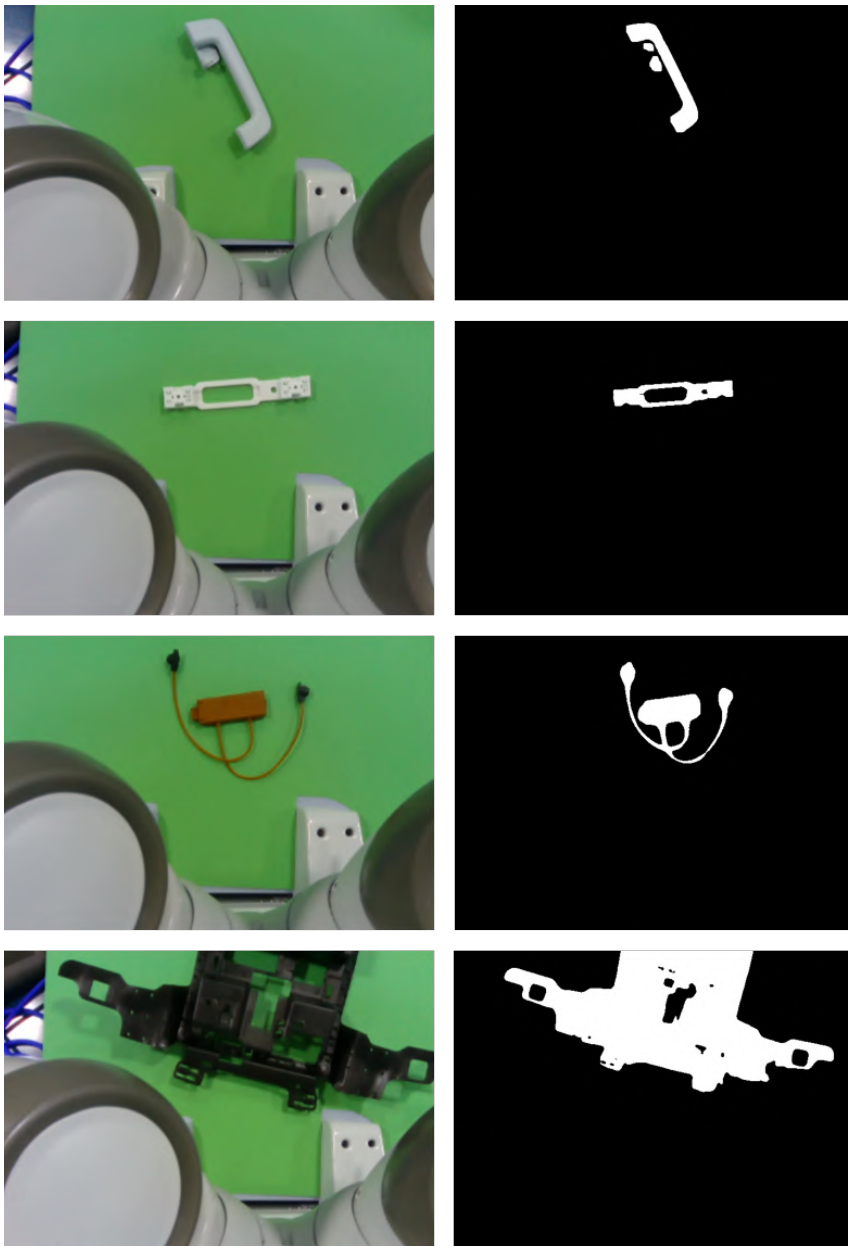


Figura 46. Ejemplos de máscaras binarias obtenidas por sustracción de fondo para una variedad de piezas.

A la vista de los resultados obtenidos, se pueden deducir tres conclusiones importantes. La primera es que, debido a que la dilatación de la máscara reduce los niveles de ruido de esta,

también hace desaparecer los orificios de pequeño tamaño que pueda tener el objeto, como es el segundo caso en la Figura 46. Aunque en esta ocasión es irrelevante dado que la herramienta del brazo robótico que agarre el objeto sería demasiado grande para estos orificios, es un aspecto a tener en cuenta si este algoritmo se aplicara en otros ámbitos.

La segunda conclusión, de mayor importancia, es la hipótesis de partida sobre la que se sustenta todo este algoritmo: la estaticidad de la escena. Se da por hecho que, exceptuando el objeto que se quiere agarrar, el resto del entorno no cambia desde el momento en que se capturó la imagen de fondo. Esta hipótesis, aunque aceptable en condiciones de laboratorio, no es lo suficientemente robusta para mantenerse a flote en entornos industriales, donde tanto las condiciones de luminosidad como la posición relativa del resto de la escena varían constantemente debido al movimiento de materiales y a los procesos de fabricación asociados.

El último aspecto importante a tener en cuenta es que este método funciona única y exclusivamente si el color del objeto a agarrar, al convertirlo en blanco y negro, es diferente al del fondo. Por tanto, si los colores del objeto son similares a aquellos del fondo, o incluso si existen cambios bruscos de iluminación en la escena, la máscara obtenida no será adecuada. Este es el caso de las superficies reflectantes que, como puede verse en la Figura 47, son invisibles a la máscara.

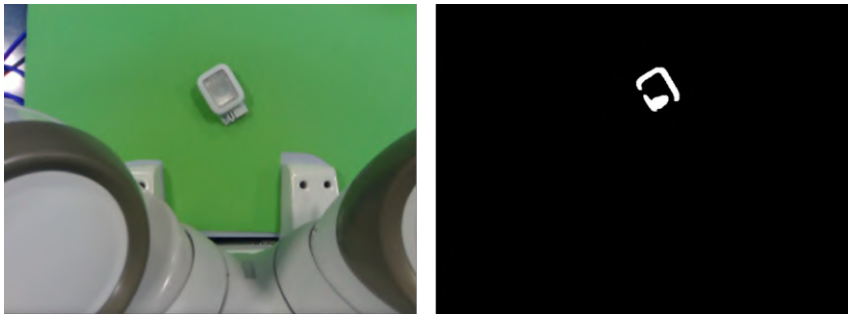


Figura 47. Ejemplo de máscara defectuosa obtenida por sustracción de fondo.

Debido en especial al segundo y tercer punto, y a pesar de ofrecer excelentes resultados en condiciones de laboratorio, se decide prescindir de este método en pro de una solución más flexible y robusta en entornos industriales. Aun así, las máscaras obtenidas sí serán utilizadas para explorar la calidad de los resultados ofrecidos por el algoritmo Dex-Net en la Sección 3.3: Cálculo del agarre.

3.2.3. Máscara por YOLO

Ambos métodos presentados anteriormente, más tradicionales, tienen un inconveniente en común: aunque poseen un tiempo de computación muy bajo, no ofrecen una flexibilidad suficiente para ser implementados fuera de un entorno controlado, como es el laboratorio. Por esta razón, se propone una forma más robusta para afrontar el problema de la detección de las piezas: usar algoritmos de inteligencia artificial.

Como ya se ha mencionado en la Sección 2.2.2, YOLO es un sistema de detección de objetos basado en redes neuronales convolucionales que permite la localización de entidades en imágenes ofreciendo tiempos de detección mucho menores que la competencia, pero sin perder en fiabilidad [RDGF16]. De esta forma, los objetos serán ubicados a partir de sus *bounding boxes*, las cuales vendrán acompañadas de un índice de confianza para la detección. YOLOv3 será, por tanto, el modelo elegido para la continuación de este proyecto.

Por defecto, YOLO viene pre-entrenado sobre el conjunto de datos COCO, lo cual puede resultar realmente útil en aquellos proyectos en los que es necesario reconocer objetos comunes como personas, vehículos o animales. Otra opción útil y que permite ahorrar un gran esfuerzo en su implementación es el uso de conjuntos de imágenes ya existentes en los cuales los objetos ya han sido localizados. Esto ofrece una gran ventaja frente a la creación de un conjunto de datos propio, ya que es precisamente esta tarea una de las más mecánicas y costosas del proceso de implantación del modelo.

Las piezas que se pretenden detectar en este proyecto, sin embargo, son componentes muy específicos de automóviles que no se encuentran en ningún otro conjunto de datos. Por esta razón, será necesario crear un conjunto de datos propio, desde cero, con el cual entrenar al algoritmo. Los pasos seguidos para la implantación del algoritmo YOLO serán, por tanto, los siguientes:

1. Creación del conjunto de datos de entrenamiento.
2. Entrenamiento del modelo en Google Colab.
3. Implantación del modelo.

Creación del conjunto de datos de entrenamiento

El conjunto de datos que se pretende utilizar para entrenar al modelo proviene de diferentes componentes automovilísticos y se compone de un total de 50 piezas, parte de las cuales se presentan en la Figura 48. Como se puede observar, existe una gran variedad en el tamaño, forma

e incluso tipo de superficie de estas, por lo que serán catalogadas según estas características para realizar diferentes ensayos sobre ellas y verificar así la robustez del algoritmo.

Figura 48. Subconjunto de piezas pertenecientes al conjunto de datos propio.

Debido a la diversidad en el tamaño de las piezas, estás serán clasificadas en grandes (G), medianas (M) y pequeñas (P). Además, se tratarán los cables (C) de forma separada. Se destaca el hecho de que la gran mayoría de las piezas son de tamaño pequeño (48 %) y mediano (24 %), lo cual podría suponer una complicación en su detección al implantar el algoritmo. Como se muestra en la imagen, las piezas quedan por tanto identificadas con una letra, representando su tamaño, seguida del número de pieza correspondiente. También se declarará en esta etapa la amplitud de pinza necesaria para aquellas piezas que serán agarradas con la pinza paralela, oscilando entre 10 y 30 milímetros.

Una vez clasificadas las piezas, se procede a la obtención de las imágenes que serán etiquetadas más adelante. Para ello, se tomarán al menos 100 imágenes diferentes por cada una de las 46 piezas del conjunto de datos según lo indicado en el tutorial del canal de YouTube Pysource [Pys20], obteniendo un número total de imágenes superior a 4600. Cabe destacar que, aunque es posible entrenar un único modelo capaz de detectar cualquier pieza del conjunto total, se decide seguir un enfoque más modular y entrenar un modelo diferente por cada tipo de pieza. De esta manera, si se necesitara añadir una pieza nueva al conjunto, simplemente sería necesario entrenar un modelo nuevo específico para esa pieza, evitando tener que reentrenar sobre las piezas que ya formaban parte del conjunto de datos anterior. Asimismo, puesto que las piezas estarán separadas en cajas diferentes según su tipo y se conoce de antemano qué caja es la que se va a procesar, esta solución permite eliminar errores de falsos positivos, ya que únicamente se cargaría el modelo de la pieza que se desee (el de la caja), y no un modelo general con múltiples

clases. Si, de lo contrario, se utilizara un único modelo general capaz de detectar cualquier tipo de pieza y se analizara una caja de piezas M5, este podría detectar falsos positivos de piezas, por ejemplo, M6, algo que no sería posible y que ya se sabía de antemano, puesto que las piezas contenidas en una misma caja son de un solo tipo.

De cara a obtener una mayor diversidad, y como se muestra en la Figura 49, de cada conjunto de 100 imágenes por pieza:

- 5 serán un fondo limpio de la mesa de trabajo, sin ninguna pieza, pero en condiciones variables (fondo de color verde, rosa, o marrón).
- 5 en las que aparecerán únicamente otras piezas que no sean de interés.
- 10 en las que aparecerán múltiples instancias de la pieza objetivo, en posiciones y perspectivas aleatorias, pero sin ninguna otra pieza que pudiera introducir ruido.
- 80 en las que aparecerán combinadas múltiples instancias de la pieza objetivo y otras piezas sin interés, favoreciendo en ocasiones la oclusión de las piezas objetivo, de cara a introducir ruido en las imágenes.

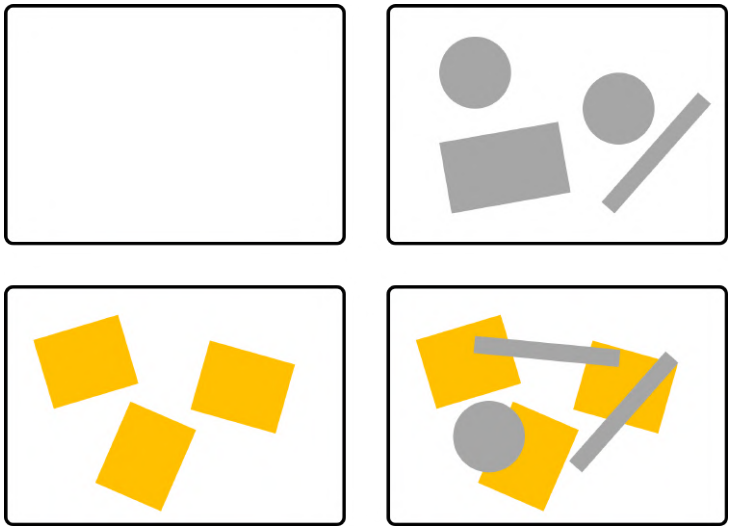


Figura 49. Disposiciones de las piezas utilizadas para la creación del dataset. De izquierda a derecha, y de arriba a abajo: fondo vacío; únicamente piezas sin interés; únicamente piezas objetivo; combinado de piezas objetivo y piezas sin interés.

El proceso de etiquetado de las imágenes se realiza inicialmente con el software de código abierto *labelImg* [dar21]. Sin embargo, este software local acabará siendo sustituido por la página web makesense.ai debido a su facilidad de uso gracias a su amigable interfaz web. De esta forma, se obtiene un archivo de texto (‘.txt’) para cada una de las imágenes en el cual se guarda la información de las *bounding boxes* que localizan las piezas en cada imagen, como se

puede ver en la Figura 50. Dichas *bounding boxes* son vectores de coordenadas $[C, x, y, w, h]$, donde:

- C es la clase del objeto etiquetado.
- x e y son las coordenadas normalizadas de la esquina superior izquierda de la *bounding box* con respecto a las dimensiones de la imagen original.
- w y h son las dimensiones de anchura y altura normalizadas de la *bounding box*.

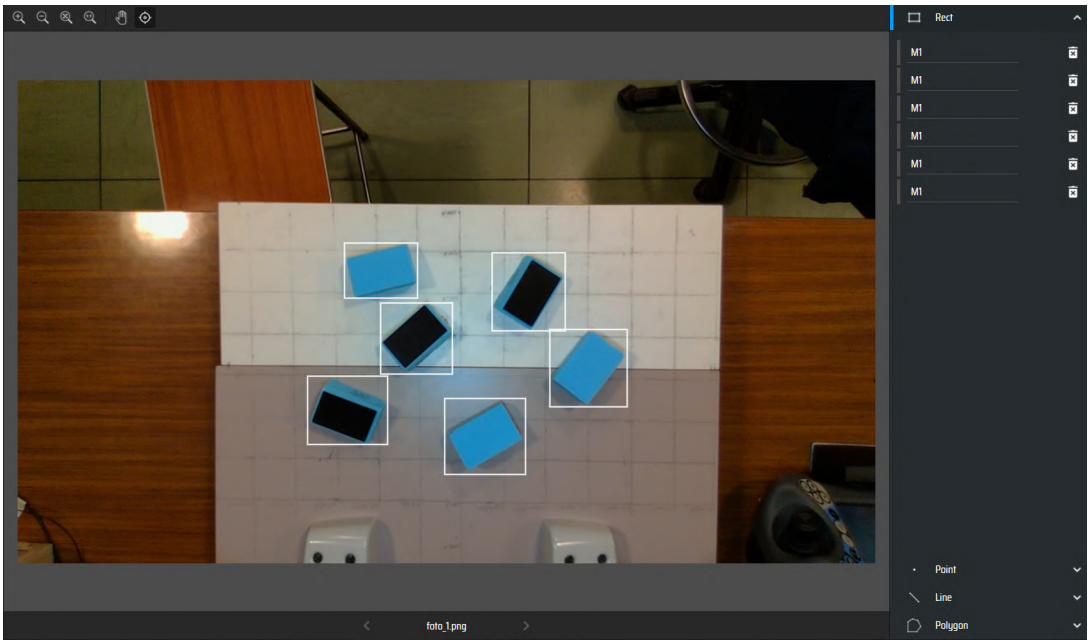


Figura 50. Etiquetado de la pieza M1 a través del software makesense.ai.

Entrenamiento del algoritmo en Google Colab

Una vez elaborado el conjunto de imágenes de entrenamiento junto a sus etiquetas, es hora de entrenar el modelo. Se decide utilizar Google Colab gracias a la posibilidad que este ofrece de utilizar un entorno con GPU de forma gratuita, permitiendo entrenar los modelos de forma mucho más rápida que si se hiciera un una CPU. El único inconveniente que presenta Colab es que, precisamente por ser un entorno gratuito, impone un límite de tiempo de uso por cada usuario y, por tanto, limita el tiempo de entrenamiento que es posible dedicarle a cada modelo.

Para realizar el entrenamiento en Colab se seguirán las instrucciones indicadas en el videotutorial de *The AI Guy* [The]. Dado que Colab es un servicio de Google, ofrece una integración muy estrecha con Google Drive, por lo que será esta la plataforma a la cual deberán subirse los archivos necesarios para el entrenamiento del modelo de YOLOv3. Los archivos principales son:

- **Archivo 'obj.zip'**. Se trata da una carpeta comprimida en la cual se encuentran las imágenes y los archivos de texto con las bounding boxes de las piezas generados previamente en makesense.ai, y que permitirán a YOLO ubicar las piezas en cada una de las imágenes.

- **Archivo 'obj.names'**. Es un archivo de texto en donde se especifica el nombre de cada objeto que se desea localizar. En este caso particular, debido a que cada modelo será encargado de detectar un único tipo de pieza, este archivo contendrá únicamente el nombre de la pieza a detectar.

- **Archivo 'obj.data'**. Incluye información relacionada con las rutas de las carpetas y archivos relevantes para el entrenamiento, como son el número de clases a entrenar, las carpetas de *train* y *valid*, o el archivo *obj.names*.

- **Archivo 'yolov3_custom.cfg'**. En este archivo se establecen las especificaciones técnicas de cada una de las capas del *backbone* que se quieren utilizar para entrenar un modelo personalizado, como son el número de clases a detectar, el número de filtros o el umbral de confianza del modelo.

- **Archivo 'generate_train.py'**. Se trata de un script en Python que genera un archivo de texto en el cual se recogen las rutas relativas de las imágenes subidas a Google Drive y que son cargadas a Google Colab.

De forma iterativa, se concluye que es necesario realizar alrededor de 2000 iteraciones por modelo para obtener unos resultados adecuados. Tras terminar el entrenamiento, se obtiene un archivo *yolov3_custom.weights* en el cual se encuentran los diferentes pesos de la red neuronal del modelo entrenado.

Implantación del algoritmo

Una vez entrenado el modelo, se puede dar paso a su implantación en el proyecto. Para poder comenzar a utilizarlo, son necesarios dos archivos:

- **Archivo 'yolov3_custom.weights'** obtenido como resultado del entrenamiento, que incluye los diferentes pesos calculados de la red neuronal.

- **Archivo 'yolov3_custom.cfg'** (previamente cargado en la fase de entrenamiento), que incluye parámetros de configuración y especificaciones técnicas del backbone de YOLO.

Disponiendo de estos dos archivos, se procede a cargar el modelo a través de la función *readNet* de OpenCV, como se muestra en el siguiente extracto de código:

```
weights = 'path_to_file_yolov3_final.weights'
custom_cfg = 'path_to_file_yolov3_custom.cfg'

model = cv2.dnn.readNet(weights, custom_cfg)
```

Fragmento de código 3.2. Comando de carga de los archivos de pesos y configuración del modelo entrenado.

En este momento, el modelo ya es capaz de utilizar una imagen como entrada y realizar detecciones de las pieza con la que ha sido entrenado, ofreciendo un nivel de confianza para cada una de ellas; sin embargo, los resultados que ofrece poseen una gran cantidad de ruido e incluso de falsos positivos debido a que estos resultados aún no han sido filtrados. Por esta razón, se llevan a cabo dos posprocesados diferentes de los resultados ofrecidos por YOLO.

El primero de ellos es la imposición de un umbral de confianza del 50 %, el cual asegura que todas las detecciones de piezas encontradas por el modelo sean fiables. El segundo filtro es una técnica conocida como NMS (*Non-maximum Suppresion*), la cual asegura que no existan múltiples detecciones similares de un mismo objeto y mantiene aquella cuyo índice de confianza es mayor. El NMS se basa en el concepto de IoU (*Intersection over Union*), que representa el solape existente entre dos *bounding boxes* diferentes y cuya definición queda ejemplificada en la Figura 51. De esta forma, dos *bounding boxes* serán consideradas como la misma detección si su solape (su IoU) es superior al umbral establecido, y se mantendrá únicamente aquella cuyo nivel de confianza sea mayor.

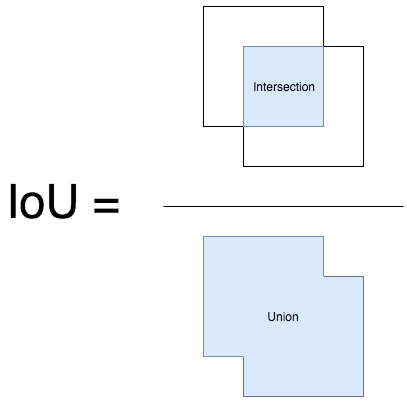
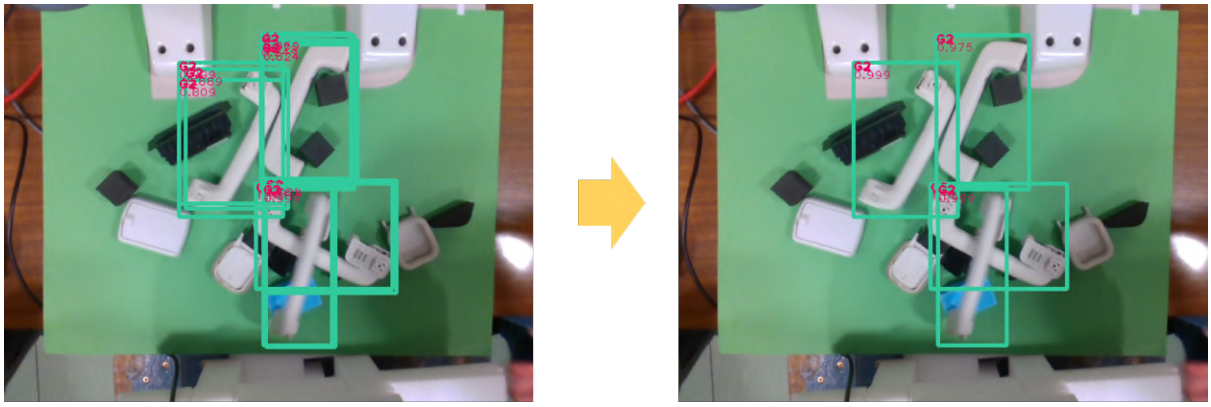


Figura 51. Representación gráfica del concepto de IoU [K.21].

El resultado de aplicar estas dos técnicas sobre los datos brutos obtenidos de YOLO puede visualizarse en la Figura 52.



Bounding boxes antes de aplicar NMS

Bounding boxes tras aplicar NMS

Figura 52. Visualización del efecto de aplicar el algoritmo NMS sobre un conjunto de detecciones. Como se puede apreciar, el número de cajas por detección por clase queda reducido a aquellas con mayores índices de confianza; siempre de forma independiente por cada clase de objeto.

Tras llevar a cabo este filtrado de resultados, el modelo está listo para detectar las diferentes piezas del conjunto de datos dada una imagen de entrada. En la Figura 53 se muestran las detecciones que el modelo es capaz de realizar para piezas de tamaño grande, mediano y pequeño, así como las máscaras asociadas a cada una de ellas, incluso estando las piezas parcialmente ocluidas. Es de notar que, para cada una de las detecciones que realiza el algoritmo, este devuelve también el nivel de confianza correspondiente de la detección, el cual se utilizará para establecer el orden de agarre de las piezas de tal forma que detecciones con mayor índice de confiaza serán agarradas primero.

Es importante destacar que, puesto que YOLO es un algoritmo de *detección* de objetos y no de *segmentación* de instancias tal y como se indica en la Sección 2.2.2, el resultado que ofrece como salida es un conjunto de bounding boxes rectangulares alrededor de las piezas encontradas, y no una máscara detallada de estas. Este comportamiento es más crítico cuanto más grande sea una pieza, o más irregularidades incluya su geometría. Por ejemplo, en la Figura 53 se aprecia que en el caso de la primera pieza (de mayor tamaño e irregular), el porcentaje de área de la *bounding box* ocupada por la pieza es mucho menor que en el caso de la segunda pieza (más pequeña y rectangular). Consecuentemente, y en contraposición a los algoritmos de segmentación por profundidad y de sustracción de fondo vistos anteriormente, en este caso será necesario realizar un paso suplementario sobre los resultados ofrecidos por YOLO que permita definir de forma más exacta la silueta de las piezas encontradas, y que será desarrollado en

mayor profundidad en la Sección 3.3.2. A pesar de esto, se decide tomar la detección de piezas con YOLO como punto de partida para continuar el proyecto gracias a la flexibilidad y robustez en la detección que ofrece el algoritmo.

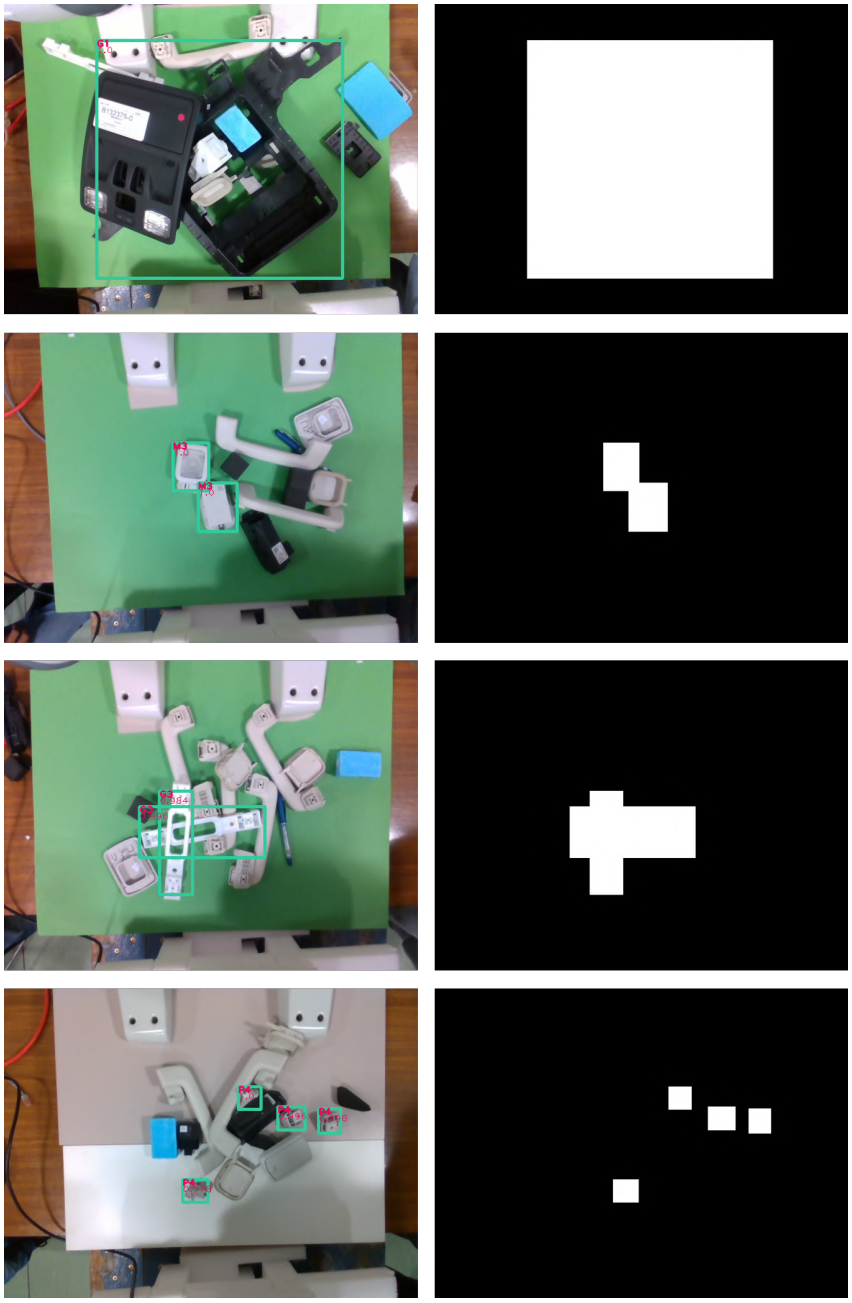


Figura 53. Ejemplos de máscaras obtenidas por YOLO para piezas de diferentes tamaños.

3.3. Cálculo del agarre

Tras la exploración de métodos de detección de piezas realizado en la sección anterior, en este punto del proyecto se dispone de una imagen de entrada RGB y de profundidad de las

piezas a agarrar, así como de una máscara que permite localizar las piezas encontradas, y una serie de confianzas para cada una de ellas.

Como ya se anticipó anteriormente y se muestra una vez más en la Figura 54, el objetivo inicial del proyecto es poder utilizar las máscaras binarias de las piezas encontradas, las imágenes RGB, y las de profundidad, e introducirlo como entrada al algoritmo Dex-Net (presentado en la Sección 2.3) de cara a calcular un agarre adecuado para cada una de las piezas. Este proceso, desarrollado a continuación en la Sección 3.3.1, acabará siendo descartado y sustituido por un algoritmo propio (desarrollado en la Sección 3.3.2) que combina la detección de piezas por YOLO y el uso de la información de profundidad capturada a través de la cámara Intel RealSense L515.

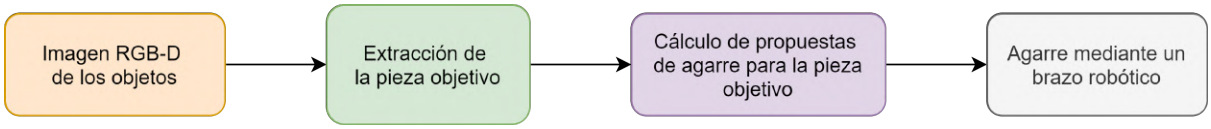


Figura 54. Flujo de información a lo largo del proyecto.

3.3.1. Análisis del uso del algoritmo Dex-Net

Como ya se introdujo en el Capítulo 2: Estado del arte, Dex-Net 4.0 es un algoritmo que utiliza redes neuronales convolucionales de robustez de agarre (GQ-CNN) para calcular el agarre óptimo de una pieza dadas informaciones sobre su profundidad y su ubicación a través de una máscara binaria. Es importante destacar que, tal y como se indica en el tutorial de instalación, este algoritmo únicamente ha sido probado en distribuciones Ubuntu [Mah17], por lo que será necesario disponer de un ordenador con sistema operativo Ubuntu o, en su defecto, configurar una máquina virtual con este sistema operativo donde poder ejecutar el algoritmo a través del hipervisor Oracle VM VirtualBox.

Debido a que las pruebas iniciales realizadas con Dex-Net se realizan de forma simultánea al análisis exploratorio de técnicas de detección de objetos (Sección 2.2), se utilizarán como imágenes de profundidad las producidas por la cámara RGB-D tras aplicar el posprocesado de filtros explicado en la Sección 3.1.4, y como máscaras binarias de entrada a Dex-Net, las obtenidas mediante el método de sustracción de fondo, presentado anteriormente en la Sección 3.2.2. Como ya se ha indicado, a pesar de no acabar siendo este método el elegido para continuar en el proyecto, las máscaras que ofrece en condiciones controladas de laboratorio son muy precisas. Por esta razón, son ideales para ser utilizadas como entrada para el algoritmo de Dex-Net y poder así analizar la viabilidad de su uso en el agarre paralelo.

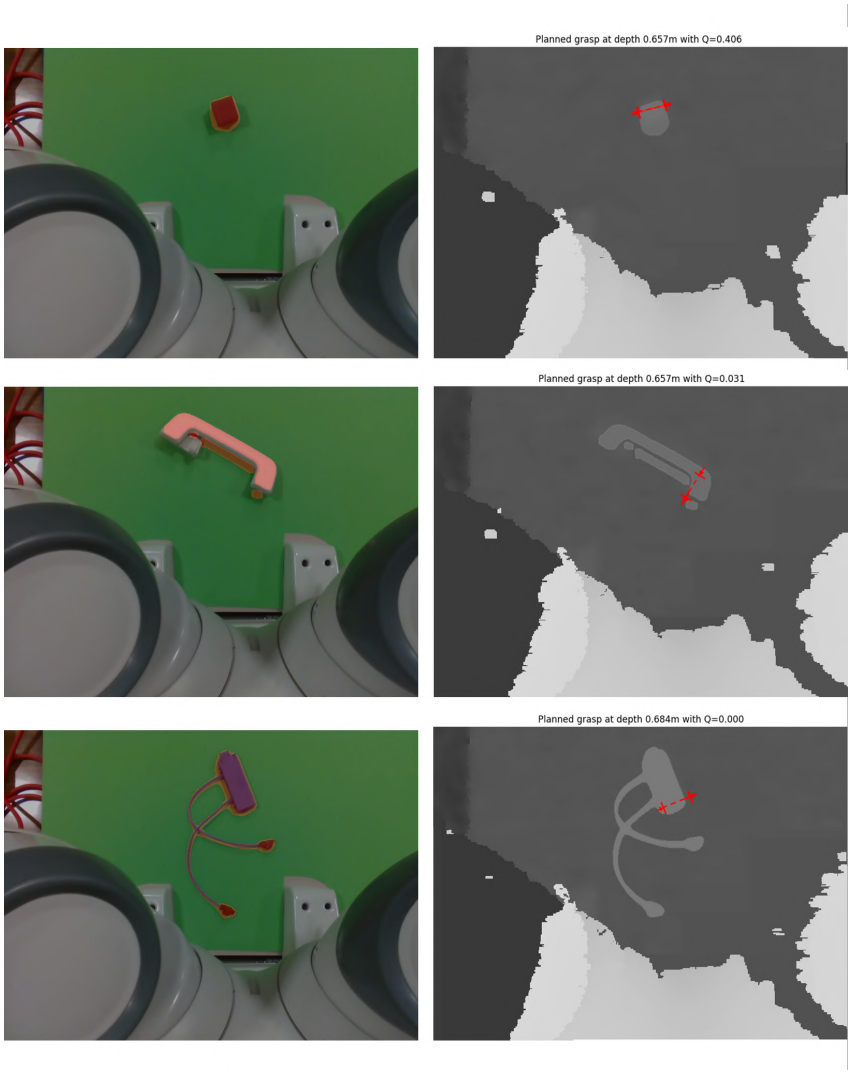


Figura 55. Propuestas de agarre paralelo ofrecidas por Dex-Net (derecha) dadas una máscara del objeto obtenida por sustracción de fondo (en rojo, izquierda) y una imagen de profundidad.

Los resultados obtenidos se muestran en la Figura 55. Como se puede apreciar, aun con una máscara precisa de las piezas (marcada en rojo), la GQ-CNN no es capaz de devolver agarres paralelos correctos con unos índices de confianza aceptables. En muchos de los casos, incluso, el algoritmo es incapaz de encontrar un agarre factible en el mundo físico. Si bien es verdad que la confianza obtenida varía según la pieza, aun en aquellos agarres que sí podrían llegar a ser realizables físicamente, en la mayoría de los casos los índices de confianza son inferiores al 10 %, e incluso al 5 %. Asimismo, los agarres propuestos por la GQ-CNN de Dex-Net son exclusivamente perpendiculares al plano de trabajo (no a la pieza), lo que hace cuasi-imposible agarrar piezas en posiciones transversales como la mostrada en la Figura 56, donde los dados de la pinza paralela deberían entrar paralelos a las caras del cubo para realizar el agarre, y no paralelos al vector normal del plano de trabajo.

Figura 56. Pieza M5 en posición inclinada. Dado que el agarre óptimo en este caso sería inclinado respecto al plano de trabajo, Dex-Net no sería capaz de calcularlo.

Se decide, por tanto, explorar la posibilidad de desarrollar un algoritmo de cálculo de agarre propio.

3.3.2. Desarrollo de un algoritmo propio: combinación del algoritmo de YOLO y segmentación por profundidad

A la luz de los resultados ofrecidos por la GQ-CNN de Dex-Net sobre nuestro conjunto de piezas propio, surge la necesidad de desarrollar un algoritmo propio que permita calcular un agarre apropiado para una pieza ya localizada. Cabe recordar que, según la forma, tamaño y superficie de la pieza, se dispone de dos agarres diferentes: pinzas paralelas y ventosa de succión.

Vista la complejidad que esta problemática supone, se decide comenzar a desarrollar el algoritmo sobre una única pieza. La pieza elegida es la etiquetada como M5, y se trata de un cubo negro de gomaespuma, mostrado en la Figura 57. Se elije esta pieza en especial debido, por un lado, a su geometría paralelepípeda, ideal para un agarre paralelo; y por otro, por poseer superficies planas sobre las que poder crear vacío con la ventosa de succión. De esta forma, se consigue poder usar una única pieza para practicar los dos tipos de agarres de los que se dispone.

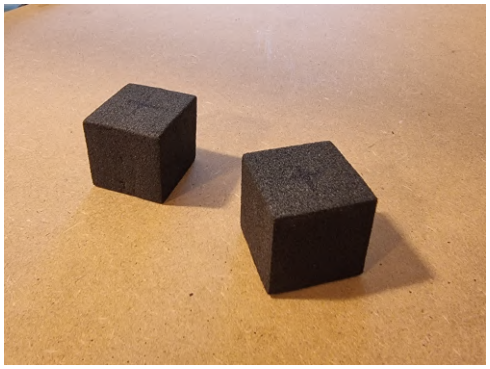


Figura 57. Pieza M5: cubo negro de gomaespuma.

Esta sección cubre, por tanto, los pasos tomados para obtener las coordenadas necesarias para realizar un agarre exitoso con ambas herramientas.

Agarre con pinza paralela

Las piezas que serán agarradas a través de un agarre paralelo serán aquellas que tengan formas paralelepípedas, como es el caso del cubo de gomaespuma. También es necesario tener en cuenta el tamaño de la pieza, pues la pinza original utilizada en el robot YuMi (mostrada en la Figura 58) tiene una apertura máxima de 50 milímetros. Aunque esto puede ser un limitante en cuanto a la variedad de piezas que el robot será capaz de agarrar, sí es suficiente para agarrar el cubo de gomaespuma, quedando pendiente como punto futuro de mejora del proyecto el diseño de una pinza paralela mejorada con mayor amplitud máxima.

Figura 58. Pinza de agarre paralelo original del robot YuMi.

Un esquema para el cálculo del agarre por pinza paralela puede encontrarse en la Figura 59. Como se puede observar, el agarre de la pieza viene definido por la ubicación tridimensional del punto terminal TP) de la pinza a través de las coordenadas $(x, y, z)_{TP}$, más los ángulos de Euler $(\theta, \beta, \gamma)_{TP}$ que definen la orientación de esta.

En una primera aproximación y con el objetivo de simplificar inicialmente la complejidad del problema, se asume que el cubo apoya completamente la cara inferior sobre el plano de trabajo. Esto se traduce en que la herramienta debe aproximarse de forma perpendicular al suelo (y en consecuencia perpendicular a la cara superior del cubo). Esta simplificación se verá resuelta en la Sección 3.4: Detección de superficies planas (RANSAC), en la cual se usa la ventosa y la aproximación de la herramienta hacia la pieza objetivo no es perpendicular al suelo, sino al plano de interés de la pieza. Así pues, siguiendo esta simplificación, los ángulos que definen la orientación de la pinza se ven reducidos a un único ángulo θ_{TP} alrededor de la normal al plano de trabajo. Las coordenadas finales necesarias para el agarre son, por tanto, $(x, y, z, \theta)_{TP}$, representadas en la Figura 60.

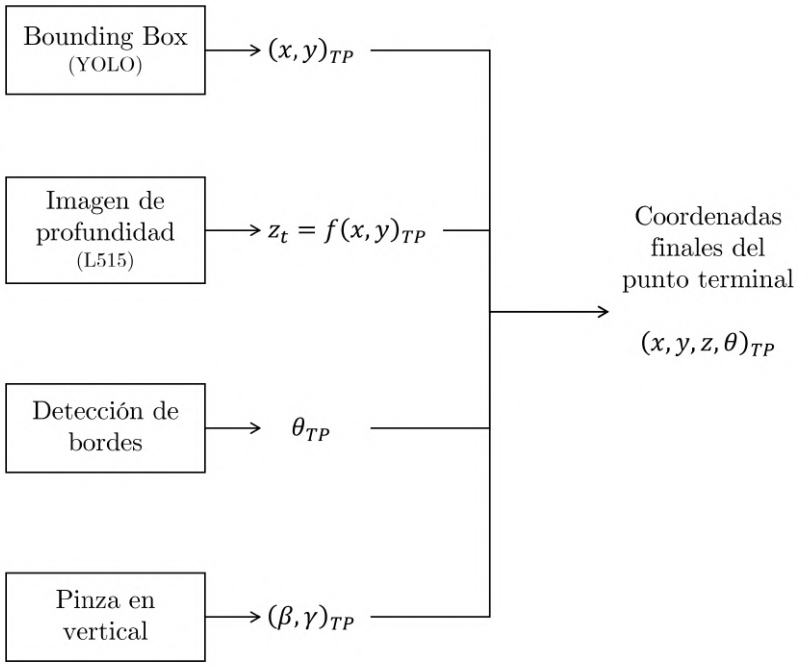


Figura 59. Flujo de información en la obtención de las coordenadas del agarre final para un agarre paralelo.

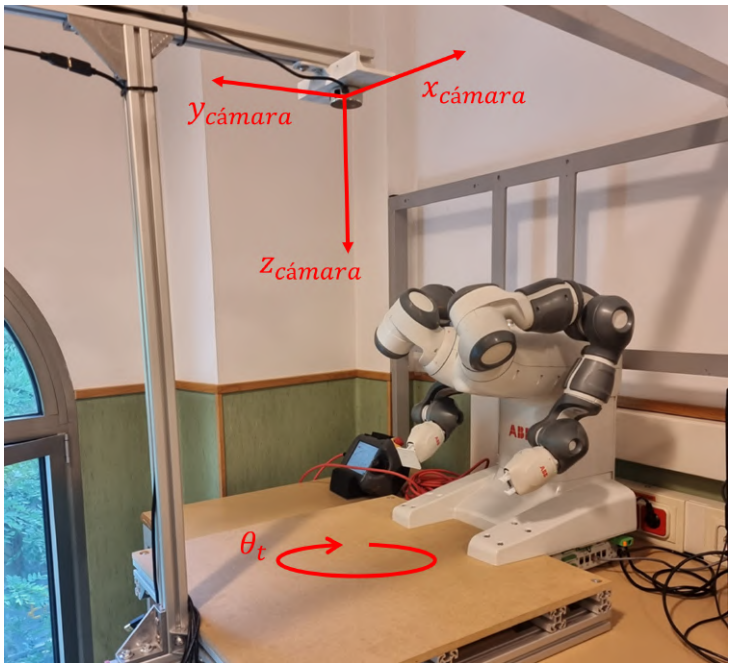


Figura 60. Sistema de coordenadas del agarre paralelo.

Una visión general del proceso de cálculo del agarre paralelo se muestra en la Figura 61. El primer paso es calcular la posición $(x, y, z)_{TP}$ en el espacio en la que se debe situar el punto terminal de la pinza. Las coordenadas $(x, y)_{TP}$ en el plano de trabajo pueden ser calculadas como el punto medio de la bounding box detectada por YOLO en la cual se encuentra la pieza. La coordenada z_{TP} del eje perpendicular a la superficie de trabajo puede obtenerse de forma directa a través de la distancia medida por la cámara RGB-D para el píxel correspondiente a las coordenadas $(x, y)_{TP}$.

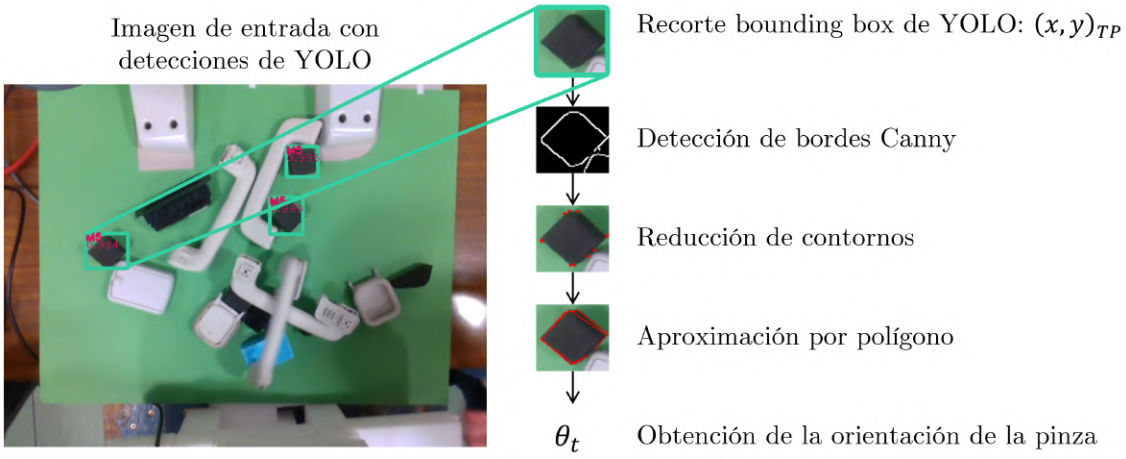


Figura 61. Proceso de cálculo del ángulo θ_{TP} para la orientación del agarre paralelo.

El segundo paso es el cálculo de la orientación con la cual la pinza paralela deberá agarrar el objeto. Dado que se pretende que la pinza agarre el objeto de forma vertical, es decir, perpendicular al plano de trabajo, el problema se ve reducido al cálculo de un único ángulo θ_{TP} alrededor del vector normal a dicho plano. Para ello, se recorta la región de interés (ROI) de la imagen RGB de la pieza utilizando la *bounding box* producida por YOLO, convirtiendo posteriormente la imagen a escala de grises y llevando a cabo finalmente una detección de bordes a través de la función *Canny()* de la librería de Python OpenCV. Seguidamente, la función *findContours()* unifica y cierra los bordes detectados para convertirlos en contornos.

Dado que es posible que la función *Canny()* haya detectado algunos bordes que no pertenecen a la pieza (debido fundamentalmente a sombras o elementos externos a la pieza de interés que quedan recogidos dentro del *bounding box* de YOLO), resultando en falsos positivos, se lleva a cabo un filtrado de estos contornos según su área para obtener aquel que sea de mayor tamaño. De esta forma, el contorno calculado puede ser recreado a través de la función *approxPolyDP()*, obteniendo así un polígono que coincide con el perímetro de la cara superior del cubo y que puede ser posteriormente filtrado en función de la longitud de sus lados para eliminar detecciones

errónas debido a sombras. Dado que este polígono tendrá forma rectangular y la perspectiva de la pieza por parte de la cámara es en planta, el cálculo del ángulo θ_{TP} de la pinza alrededor del eje z de la muñeca vendrá dado por la orientación que tengan, precisamente, los lados de este rectángulo, que serán vistos por la cámara en su verdadera magnitud.

Agarre con ventosa de succión

Como ya se ha mencionado en secciones anteriores, es posible utilizar una ventosa de succión como punto terminal de uno de los brazos del robot YuMi. Esta ventosa obtendrá su caudal de aspiración a través de un tubo de Venturi, el cual es, a su vez, alimentado por un compresor externo.

El agarre con ventosa se practicará sobre aquellas piezas que posean superficies planas. En su mayoría, estas estarán fabricadas en plástico o gomaespuma, como es el caso del cubo analizado en la sección anterior. Por tanto, siempre que dicha superficie plana posea un tamaño lo suficientemente grande como para posar la ventosa, y siempre que su peso no sea superior a la capacidad de succión máxima del compresor, la ventosa será el método preferido de agarre.

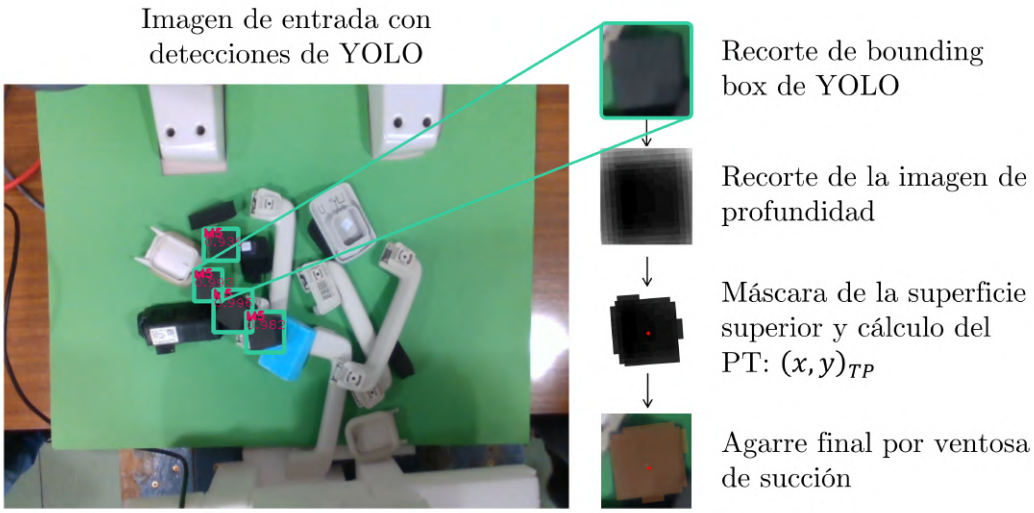


Figura 62. Proceso de cálculo del punto terminal $(x, y, z)_{TP}$ para el agarre con ventosa.

El proceso de cálculo para el agarre con ventosa difiere ligeramente del mencionado en la sección anterior para el agarre paralelo y se muestra en la Figura 62. Tras realizar la detección de piezas a través de YOLO, se busca detectar la cara superior el cubo utilizando la cámara de profundidad. Para ello, se utiliza la *bounding box* proporcionada por YOLO para crear una máscara que se aplicará posteriormente sobre la imagen de profundidad de la escena, "recortándola" y manteniendo únicamente la información de profundidad de la región de interés

(ROI) de la pieza. De este "recorte", y de forma similar al procedimiento utilizado en la Sección 3.1.2 (Análisis de consistencia en mediciones consecutivas de profundidad), se filtran aquellos puntos p_i (donde i representa las coordenadas (x, y) de cada punto) cuya distancia a la cámara de profundidad sea mínima, con una cierta tolerancia ε, que corresponderán a los puntos de la cara superior S del cubo y que conformarán, por tanto, la superficie sobre la cual se posará la ventosa para crear el vacío:

$$p_i \; = \; p \in S \; \mid \; z_{min} - \varepsilon < z_i < z_{min} + \varepsilon \tag{7}$$

Una vez obtenida esta superficie, es posible calcular el centro de masas de esta a través de la media de las coordenadas x e y de todos los puntos que la conforman. La coordenada z_{TP} del eje vertical puede en este momento ser inferida como función de las coordenadas $(x, y)_{TP}$ del punto terminal haciendo uso de la imagen de profundidad obtenida a través de la cámara RGB-D.

Dado que esta detección de planos se realiza única y exclusivamente para el agarre con ventosa, no es necesario realizar un paso adicional para calcular el ángulo θ_{TP} de la muñeca del brazo robótico. Gracias a la simetría axial de la ventosa, así como al hecho de que el agarre se realiza en todo momento de forma vertical, y bajo la hipótesis de que el cubo está completamente apoyado sobre el plano de trabajo, el valor de θ_{TP} sobre el eje perpendicular al plano de trabajo será irrelevante en el agarre.

Este proceso queda esquematizado en la Figura 62 y tiene como resultado las coordenadas finales del punto terminal, que serán de la forma $(x, y, z)_{TP}$.

3.4. Detección de superficies planas (RANSAC)

Si bien la solución presentada en la sección anterior ofrece la posibilidad de detectar una superficie plana sobre la cual llevar a cabo un agarre con ventosa de succión, el método utilizado parte de seleccionar la nube de puntos de altura z similar. El inconveniente que esta metodología supone es el hecho de que este tipo de detección de planos únicamente funcionará si la orientación de la cámara de profundidad sigue la normal de la superficie a detectar, es decir, si la superficie plana es perpendicular al eje z de la cámara y, por lo tanto, paralela a la superficie de trabajo. Aunque este método es suficiente para la detección de las caras de un cubo completamente apoyado sobre el plano de trabajo, en cualquier otro caso, el plano detectado no sería más que la vista en planta de la parte más alta de la pieza.

De cara a mejorar la robustez de la solución, se busca un método que permita detectar superficies planas que puedan estar inclinadas. Es por esta razón por la que se propone hacer uso del algoritmo de RANSAC, mencionado anteriormente en el Capítulo 2: Estado del arte.

A modo de recordatorio, el algoritmo RANSAC selecciona puntos de forma aleatoria, los cuales conforman un plano. Se puede entonces calcular la desviación del resto de puntos respecto a este plano, de forma que aquellos puntos cuyas desviaciones sean inferiores a un determinado umbral son considerados parte del plano seleccionado. Este proceso se realiza de forma iterativa, manteniendo aquellos planos con mayor número de *inliners*, y descartando el resto.

La implementación de este algoritmo se realizará a través de un código público alojado en GitHub [Den21] cuyo objetivo es, precisamente, la detección de múltiples planos en una escena. Dada una imagen de profundidad, el primer paso es preprocesarla y reducir el nivel de ruido que posea, eliminando los posibles *outliers* de la nube de puntos a través de la función *statistical_outlier_removal()* incorporada en la librería open-source Open3D [ZPK18], orientada a la manipulación de datos en 3D. Una vez limpiados los datos, la función *segment_planes()* aplica el algoritmo de RANSAC sobre la nube de puntos y detecta los planos encontrados en la escena. Una vez conocidos cada plano, es posible elegir como punto de contacto de la ventosa el centro de masas del plano, y calcular la orientación de la muñeca del brazo robótico como el vector normal a cada plano a través de sus hiperparámetros.

En las figuras 63 y 64 se muestran los resultados que ofrece este algoritmo. Como se puede observar, el algoritmo no solo es capaz de identificar planos horizontales situados a diferentes alturas (cota z en el sistema de referencia del robot), sino que también es capaz de detectar planos inclinados, por ejemplo, a 45º. Así pues, a partir de las coordenadas de los puntos que conforman cada plano es posible calcular tanto la posición a la cual deberá llegar la ventosa de succión, como la orientación con la que deberá hacerlo, obteniendo todas las componentes necesarias para llevar a cabo un agarre adecuado.

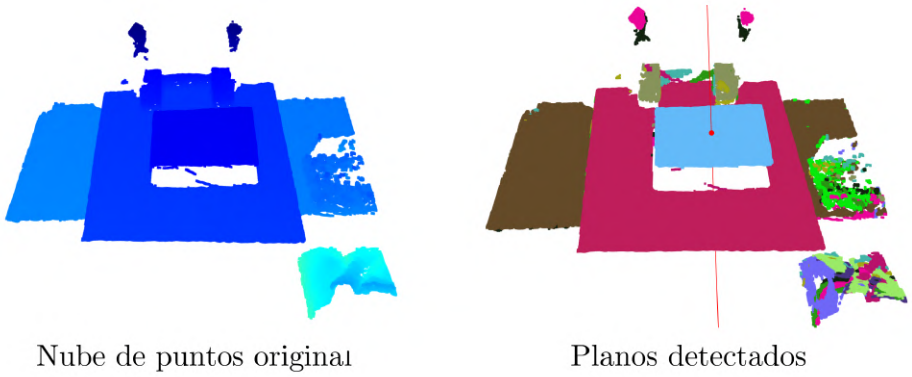


Figura 63. Detección de planos horizontales a través del algoritmo RANSAC.

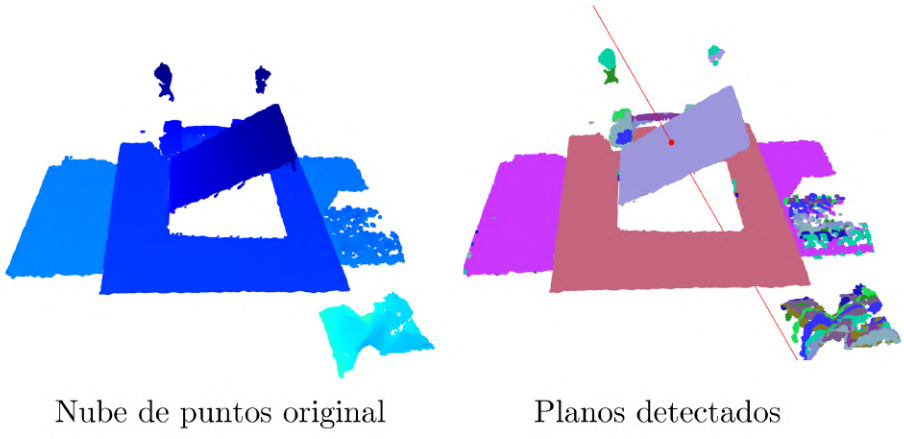


Figura 64. Detección de planos inclinados a través del algoritmo RANSAC.

Capítulo 4

Análisis de resultados

ESTE capítulo busca mostrar los resultados obtenidos a lo largo del desarrollo del proyecto, tanto aquellos que fueron positivos y que ofrecieron un rendimiento adecuado, como aquellos que no ofrecieron los resultados esperados.

En primer lugar, es necesario destacar la superioridad de la tecnología LiDAR respecto a los módulos estereoscópicos en lo que a medición de distancia respecta. Si bien es cierto que el precio de los módulos LiDAR es notablemente superior (alrededor de 350 $ por el modelo L515 frente a unos 250 $ por el modelo D435), la mejora en la calidad de la imagen de profundidad en comparación con la obtenida a través del módulo estereoscópico es notoria (pasando de 10 milímetros de precisión del modelo D435 a menos de 3 milímetros en el L515), ofreciendo mediciones más precisas y consistentes.

Por otro lado, se han explorado las fortalezas y debilidades de diferentes métodos en la búsqueda de un algoritmo que permitiera detectar y aislar las diferentes piezas que se quieren agarrar en cada momento, cuyos resultados se presentan en la Tabla 2. Debido a su propia naturaleza, pese a que el método de sustracción de fondo proporciona excelentes resultados en condiciones constantes y controladas, la calidad de estos disminuye rápidamente frente a cualquier tipo de variación en la luminosidad o distribución de la escena, considerando "pieza" todo aquello que no coincida con el fondo referencia que se tiene de antemano, lo cual provoca la aparición de falsos positivos ante cambios mínimos.

La segmentación por profundidad, a pesar de ser un método robusto ante cambios en las condiciones lumínicas, únicamente realiza un corte horizontal de la representación tridimensional de la escena, lo que lo invalida en situaciones donde la parte superior de las piezas no constituyen una superficie plana horizontal. Asimismo, es importante destacar que tanto la sustracción de fondo, como la segmentación por profundidad, sufren de un importante problema común: si

bien ambos detectan la presencia de *algo* que cumple unas condiciones dadas, ninguno de los dos es capaz de *reconocer* o *identificar* las piezas, es decir, siguen los principios de la visión artificial clásica.

	Segmentación por profundidad	**Sustracción de fondo**	**YOLO + segmentación por profundidad**
Rendimiento en escenas estáticas	***	***	***
Rendimiento en escenas dinámicas	*	*	**
Robustez frente a cambios lumínicos	***	*	**
Robustez frente al color de la pieza y del entorno	***	*	***
Robustez frente a cambios en el entorno	*	*	***
Precisión de la máscara	**	***	**
Reconocimiento individual de piezas	*	*	***

Cuadro 2. Comparación del rendimiento ofrecido por cada uno de los métodos de detección de objetos analizado en el proyecto. Una estrella (*) representa un mal rendimiento; dos estrellas (**), un rendimiento medio; y tres estrellas (***), un rendimiento alto.

Es precisamente este el punto fuerte de YOLO, el cual sí es capaz de diferenciar qué es y qué no es una pieza. De cara a evaluar la fiabilidad de este modelo, se analizan las detecciones que este devuelve para una serie de imágenes conteniendo múltiples piezas diferentes entremezcladas y superpuestas, detectando un tipo de pieza cada vez. Así pues, se obtiene que el modelo es capaz de identificar y localizar de forma correcta la pieza deseada el 84 % de las veces. Una comparación de los resultados obtenidos a través de cada una de estas tres técnicas se muestra en la Figura 65. Si, además, se combinan las detecciones calculadas por YOLO con

una segmentación básica de profundidad, se obtiene un algoritmo inteligente que es capaz de localizar de forma precisa dónde se encuentran las piezas deseadas y, en caso de poseer una superficie superior plana, como los cubos de gomaespuma, calcular un agarre.

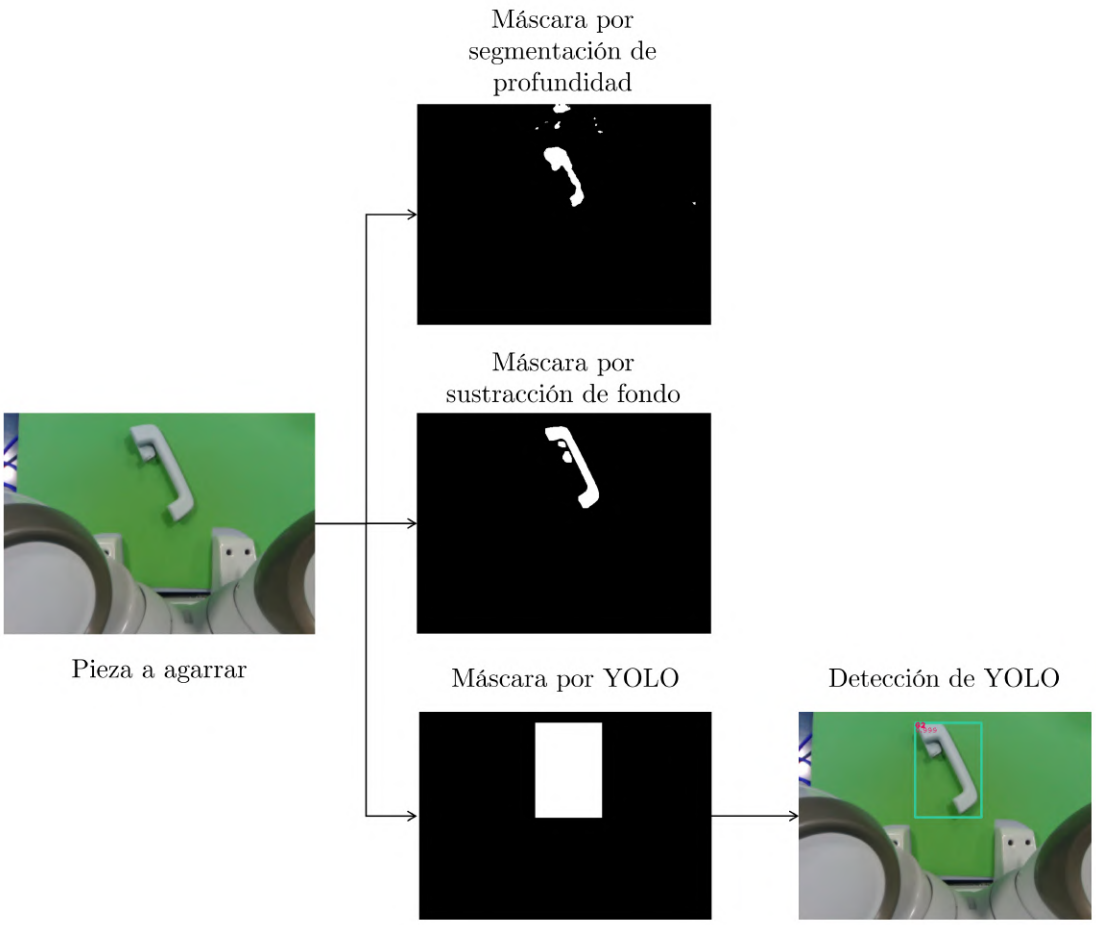


Figura 65. Comparativa de las máscaras resultantes tras la aplicación de los métodos de segmentación por profundidad, sustracción de fondo, y YOLO.

Precisamente gracias a la implantación del algoritmo YOLO en el proyecto, se lleva a cabo la creación de un conjunto de datos propio que incluye las piezas de componentes automovilísticos que se busca poder agarrar, así como un conjunto de archivos que hace uso de dicho conjunto de datos para el entrenamiento del modelo. Así pues, se pone a disposición del resto de participantes del Instituto de Investigación Tecnológica de ICAI los recursos encontrados y desarrollados necesarios para entrenar un modelo propio del algoritmo YOLOv3.

Los agarres contemplados en este proyecto son de tipo paralelo y por ventosa. Dado que el agarre paralelo necesita tanto unas coordenadas tridimensionales $(x, y, z)_{TP}$ de posición del punto terminal del brazo, como la orientación θ_{TP} que debe tener la muñeca, es posible aplicar

una detección de bordes sobre las detecciones de YOLO. Por otro lado, si se pretende realizar un agarre por ventosa de succión, una segmentación por profundidad permite obtener el plano superior de la pieza, sobre el cual se podrá posar la ventosa. Los resultados de ambos agarres se presentan en la Figura 66.

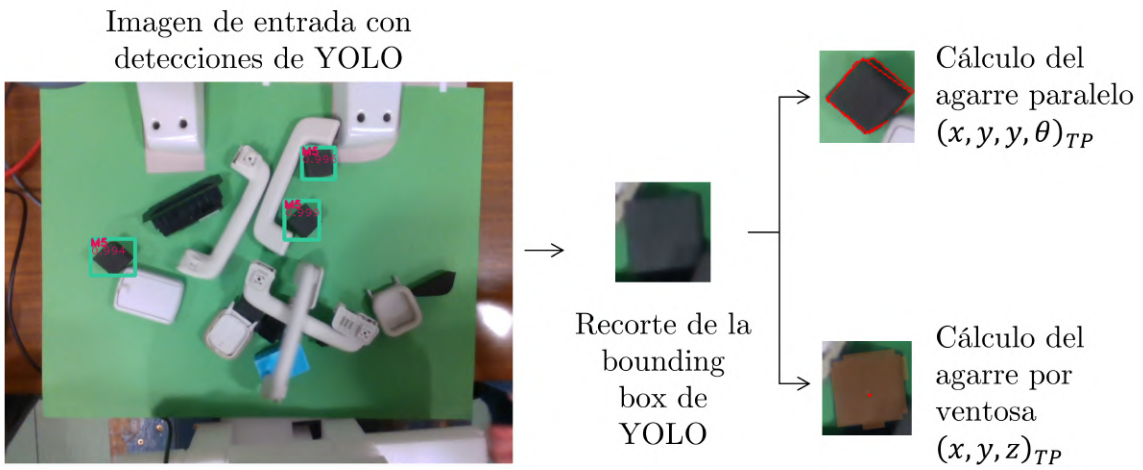


Figura 66. Comparativa de los resultados obtenidos para las diferentes opciones de agarre (paralelo y ventosa).

Dado que no todas las piezas poseerán una superficie superior plana, se lleva a cabo una implantación del algoritmo de RANSAC que permite detectar las superficies planas de la escena. De esta forma, el agarre por ventosa no se ve limitado a realizarse de forma vertical al plano de trabajo, sino que se abre las puertas a la posibilidad de posicionar el punto de agarre en planos inclinados, tal y como se muestra en la Figura 67.

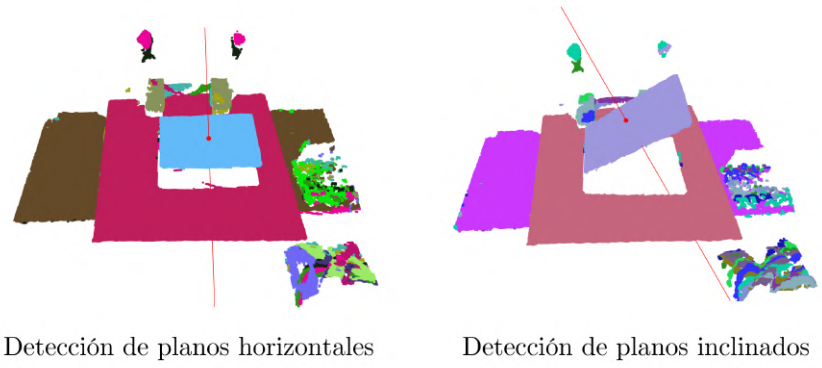


Figura 67. Cálculo del agarre por ventosa mediante detección de planos a través del algoritmo de RANSAC.

Gracias a la combinación del algoritmo de YOLO con la detección de bordes (en caso de agarre paralelo) y la detección de planos por RANSAC (en caso de agarre por ventosa), se logra finalmente tener un algoritmo que es capaz de detectar, identificar, y proponer un agarre fiable, utilizando como entrada una imagen RGB-D que contenga la pieza que se desea agarrar.

Capítulo 5

Conclusiones y futuros desarrollos

COMO se mencionó en el capítulo anterior, los resultados obtenidos en este proyecto ofrecen la posibilidad de calcular agarres paralelos y por ventosa de forma flexible según la ubicación y el tipo de pieza. Se extraen, por tanto, las siguientes conclusiones:

- **Integración de la GQ-CNN de Dex-Net.** Si bien los resultados publicados por el Laboratorio de Automatización de Berkeley [MMS+19] son muy prometedores, los resultados obtenidos en su implantación sobre el conjunto de piezas específicas utilizadas en este proyecto no han sido suficientemente satisfactorios. Este hecho puede deberse, entre otros, a una geometría irregular en muchas de las piezas (lo que provoca la oclusión de partes de la pieza), a un mapa de profundidad inadecuado, o al color y textura de las piezas utilizadas.

- **Inviabilidad del uso de algoritmos tradicionales en entornos industriales.** Si bien los métodos de segmentación por profundidad y segmentación de fondo son capaces de ofrecer buenos resultados, requieren un nivel de control sobre el entorno que no es posible lograr en la mayoría de entornos industriales. Su uso, sin embargo, sí puede ser de gran utilidad en tareas específicas, donde las condiciones de trabajo pueden ser controladas más fácilmente. Algoritmos más flexibles, como YOLO, proporcionan una herramienta mucho más robusta y consistente que puede ofrecer excelentes resultados sin necesidad de realizar grandes inversiones, ya que usan como entrada una simple imagen a color.

- **Dificultad de desarrollo de nuevos conjuntos de datos.** Los datos son la piedra angular en el uso de algoritmos de inteligencia artificial, y su calidad desempeña un papel importante en la calidad final del algoritmo. La creación de un conjunto de datos propio es un proceso largo y que requiere de gran cantidad de recursos, lo cual puede constituir una barrera de entrada en muchos casos, especialmente en aquellos en los cuales sea inviable reutilizar, incluso parcialmente, otro conjunto de datos ya existente. Una posible solución a este problema es la generación de conjuntos de datos sintéticos a partir de

herramientas de modelado 3D como Blender [Fou], aumentando así la variedad de las piezas de entrenamiento de forma asequible.

- **Implantación del algoritmo YOLO.** Una vez superada la barrera de entrada que supone la creación de un conjunto de datos propio, el entrenamiento, y la implantación del algoritmo, YOLOv3 proporciona unos resultados altamente satisfactorios y robustos, los cuales podrían ser fácilmente extrapolados a un entorno industrial real, pudiéndose incluso mejorar los tiempos de procesado si se usase una variación de este conocida como Tiny-YOLO, el cual posee tiempos de procesado hasta 4 veces más rápidos [Ros20].

- **Detección de planos a través del algoritmo de RANSAC.** Si bien es cierto que el algoritmo de RANSAC proporciona buenos resultados en los experimentos realizados, su escalabilidad a entornos industriales más caóticos puede ser insuficiente. Se detecta una alta dependencia de las cualidades de la cámara que se utiliza como entrada de datos, causando inconvenientes en especial en piezas pequeñas (como es el caso del cubo) o zonas "escarpadas".

A la vista de los resultados alcanzados, a continuación se proponen una serie de potenciales mejoras que podrían ayudar a incrementar las funcionalidades del proyecto y facilitar su escalabilidad a entornos diferentes:

- **Enriquecimiento de los datos de entrada.** La configuración actual utiliza una única cámara RGB-D con vista en planta del plano de trabajo, por lo que el éxito del agarre no siempre se consigue debido a la perspectiva y resolución de la cámara. El uso de múltiples cámaras RGB-D podría ayudar enormemente en esta tarea. Una cámara integrada en el punto terminal de la pinza no solo permitiría obtener una imagen más cercana de las piezas, sino que también abriría la puerta a la posibilidad de tener una segunda perspectiva no estática de la escena, pudiendo, por ejemplo, obtener una imagen de profundidad lateral de los objetos, posibilitando realizar agarres laterales u oblicuos, e incluso recrear la totalidad de la escena de forma tridimensional.

- **YOLO por regiones.** Alternativamente a la detección de planos, se puede entrenar un segundo algoritmo YOLO cuya tarea fuera detectar las superficies planas de cada pieza, determinadas de antemano de forma manual o a través de un sistema automatizado.

- **Uso de aprendizaje por refuerzo.** Dado que la cinemática actual del robot no dispone de control de trayectorias ni de un lazo cerrado de realimentación que le permita verificar su posición, la implantación de un algoritmo de aprendizaje por refuerzo podría mejorar sustancialmente el éxito en los agarres de las piezas contempladas.

- **Desarrollo de una pinza universal.** La vía explorada en este trabajo pone su énfasis de la parte de inteligencia del agarre en el software, utilizando *grippers* tradicionales. Otra posibilidad de lograr un agarre universal es que sea el propio *gripper* el que se adapte al objeto, como es el caso de la pinza mostrada en la Figura 68, desarrollada por la compañía Festo [fUR] e inspirada en la mecánica de la lengua del camaleón. Puesto que la pinza se amolda a la forma del objeto, se elimina tanto la necesidad de utilizar dos brazos diferentes para ser capaz de agarrar piezas de diferentes formas, como la necesidad de detectar superficies planas.

Figura 68. *Gripper* de forma adaptativa desarrollado por la compañía Festo [fUR].

Capítulo 6

Alineación con los Objetivos de Desarrollo Sostenible

En el año 2015, la ONU aprobó la Agenda 2030 sobre el Desarrollo Sostenible, una oportunidad para que los países y sus sociedades emprendan un nuevo camino con el objetivo de mejorar la calidad de vida de todos, sin dejar a nadie atrás. Dicha Agenda cuenta con lo que se conoce como los 17 Objetivos de Desarrollo Sostenible (ODS) [Uni15], donde se incluye desde la eliminación de la pobreza hasta el combate al cambio climático, la igualdad de la mujer, el diseño de ciudades o la educación.

Figura 69. Lista de los 17 Objetivos de Desarrollo Sostenible propuestos por la ONU [Uni15].

A continuación se citan aquellos ODS que guardan mayor relación con el trabajo:

- **"Objetivo 8: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos"**. Dotar a unos brazos robóticos de la capacidad de agarrar cualquier tipo de objeto abre las puertas a la posibilidad de automatizar tareas repetitivas, como es el caso de la carga y descarga de piezas, o del transporte de elementos en las líneas de montaje. Este tipo de tareas monótonas puede traer consigo la alienación del trabajador, convirtiéndole en algo distinto al producto de su labor, el cual se convierte en

una mercancía que se enajena. De esta forma, el trabajo de la persona se reduce a realizar una tarea a cambio de un salario, sin llegar del todo a sentirla como propia, y sin sentirse realizado en su desempeño.

Un desarrollo suficientemente avanzado de la versión propuesta en este proyecto podría llegar a lograr la automatización estas tareas, liberando tiempo y recursos que podrían ser utilizados por los empleados en otras áreas de trabajo menos monótonas, más productivas y, en ocasiones, incluso más seguras. Si bien es cierto que esta automatización conlleva la destrucción de ciertos empleos, su implantación crea, en paralelo, puestos de trabajos nuevos, diferentes a los anteriores, como son el cuidado y la revisión de los autómatas.

- **"Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación"**. Este proyecto, como se mencionó anteriormente en la Motivación y en los Objetivos, busca responder al problema actual real que es la limitación a la que se ven sometidos los brazos robóticos en los entornos industriales, y cómo el fusionar la inteligencia de los modelos de Machine Learning con la rapidez de actuación de estos brazos podría catapultar la productividad y la flexibilidad en la industria, abriendo las puertas a nuevas aplicaciones de estos autómatas, por ejemplo, a través de tecnologías IoT.

- **"Objetivo 12: Garantizar modalidades de consumo y producción sostenibles"** . Los avances económicos y sociales logrados durante el último siglo han sido acompañados de una degradación medioambiental que está actualmente poniendo en peligro los mismos sistemas de los que depende nuestro desarrollo futuro. Las diferentes tecnologías analizadas en este proyecto tienen el potencial de abrir vías como un control inteligente de inventarios, que permitiría conocer con exactitud las necesidades reales en los diferentes puntos de fabricación y, por tanto, resultaría en una disminución del stock de productos.

  La cantidad de infraestructuras necesarias destinadas al almacenamiento de estos productos se vería entonces rebajada, logrando así la reducción de los recursos consumidos durante los procesos de fabricación y logística de bienes

Bibliografía

[AR] R. ASALE and RAE, "RADAR." [Online]. Available: https://dle.rae.es/radar

[BML20] J. Boal Martín-Larrauri, "Unit 5: Perception," in Cyber-Physical Systems and Robotics, 2020.

[Bur16] W. Burger, "Zhang's camera calibration algorithm: In-depth tutorial and implementation," May 2016.

[Cor17] P. Corke, Robotics, vision and control. Fundamental algorithms in MATLAB., 2nd ed. Springer, 2017.

[dar21] darrenl, "tzutalin/labelImg," Jul. 2021, original-date: 2015-09-17T01:33:59Z. [Online]. Available: https://github.com/tzutalin/labelImg

[Den21] Y. Deng, "yuecideng/Multiple_planes_detection," Apr. 2021, original-date: 2020-07-05T06:34:24Z. [Online]. Available: https://github.com/yuecideng/Multiple_Planes_Detection

[FB81] M. A. Fischler and 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, no. 6, pp. 381–395, Jun. 1981, number of pages: 15 Publisher: Association for Computing Machinery tex.address: New York, NY, USA tex.issue_date: June 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[Fou] B. Foundation, "blender.org - Home of the Blender project - Free and Open 3D Creation Software." [Online]. Available: https://www.blender.org/

[Fri] A. Frich, "Optical distortions: how to correct them?" [Online]. Available: https://www.panoramic-photo-guide.com/optical-distortions-panoramic-photography.html

[fUR] F. D. for Universal Robots, "FESTO ADAPTIVE SHAPE GRIPPER KIT." [Online]. Available: https://www.universal-robots.com/plus/products/festo/festo-adaptive-shape-gripper-kit/

[Gen11] J. Geng, "Structured-light 3D surface imaging: a tutorial," Adv. Opt. Photon., vol. 3, no. 2, pp. 128–160, Jun. 2011, publisher: OSA. [Online]. Available: http://aop.osa.org/abstract.cfm?URI=aop-3-2-128

[Gib02] S. Gibilisco, Teach yourself electricity and electronics. New York : McGraw-Hill, 2002. [Online]. Available: http://archive.org/details/teachyourselfele00gibi

[GJSK+20] A. Grunnet-Jepsen, J. Sweetser, T. Khuong, S. Dorodnicov, D. Tong, and O. Mulla, "Intel® RealSense™ Self-Calibration for D400 Series Depth Cameras," 2020. [Online]. Available: https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras

[GMR11] O. Gallo, R. Manduchi, and A. Rafii, "CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data," Pattern Recognition Letters, vol. 32, no. 3, pp. 403–410, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865510003557

[GO11] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," in ACM SIGGRAPH 2011 papers on - SIGGRAPH '11. Vancouver, British Columbia, Canada: ACM Press, 2011, p. 1. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1964921.1964964

[gor20] gordonlim, ""The World is Your Green Screen" — what I've learnt from studying the paper," Apr. 2020. [Online]. Available: https://medium.com/swlh/the-world-is-your-green-screen-what-ive-learnt-from-studying-the-paper-115a56ed8350

[HGDG18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Jan. 2018, arXiv: 1703.06870 Último acceso: 20/11/2020. [Online]. Available: http://arxiv.org/abs/1703.06870

[HYK08] M. Hariyama, N. Yokoyama, and M. Kameyama, "Design of a trinocular-stereo-vision VLSI processor based on optimal scheduling," IEICE Transactions on Electronics, vol. 91-C, pp. 479–486, Apr. 2008.

[Int] Intel, "Intel RealSense D400 Series Product Family Datasheet." [Online]. Available: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet

[K.21] S. K., "Non-maximum Suppression (NMS)," Apr. 2021. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

[LBJB12] T. Liu, A. W. Burner, T. W. Jones, and D. A. Barrows, "Photogrammetric techniques for aerospace applications," Progress in Aerospace Sciences, vol. 54, pp. 1–58, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042112000267

[Li24] L. Li, "Time-of-Flight Camera -An Introduction," Jan. 2024. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1048.8288&rep=rep1&type=pdf

[Low01] D. Lowe, "Object recognition from local scale-invariant features," Proceedings of the IEEE International Conference on Computer Vision, vol. 2, Jan. 2001.

[Mah17] J. Mahler, "Installation Instructions — Dex-Net 0.2.0 documentation," 2017. [Online]. Available: https://berkeleyautomation.github.io/dex-net/install/install.html

[MBRS+21] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the wild: Neural radiance fields for unconstrained photo collections," 2021, arXiv: 2008.02268 [cs.CV].

[MMS+19] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," Science Robotics, vol. 4, no. 26, 2019, Último acceso: 22/09/2020. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau4984/

[MST+20] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," 2020, arXiv: 2003.08934 [cs.CV].

[Nat12] National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, "Lidar 101: An Introduction to Lidar Technology, Data, and Applications," Tech. Rep., Nov. 2012. [Online]. Available: https://spationetblog.files.wordpress.com/2016/01/an-introduction-to-lidar-technology.pdf

[Ope] OpenCV, "OpenCV: Camera Calibration." [Online]. Available: https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html

[PSB+21] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," 2021, arXiv: 2011.12948 [cs.CV].

[Pys20] Pysource, "Train YOLO to detect a custom object (online with free GPU)," Apr. 2020. [Online]. Available: https://youtu.be/_FNfRtXEbr4

[RDGF16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 2016, arXiv: 1506.02640 Último acceso: 13/11/2020. [Online]. Available: http://arxiv.org/abs/1506.02640

[RM99] A. Romero-Manchado, ""Calibración de cámaras no métricas por el método de las líneas rectas"," Mapping, ISSN 1131-9100, Nº 51, 1999, pags. 74-80, Jan. 1999.

[Ros20] A. Rosebrock, "YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS," Jan. 2020. [Online]. Available: https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/

[Sah18] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," Medium, Dec. 2018, Último acceso: 21/10/2020. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[SDK21] I. R. SDK, "IntelRealSense/librealsense," Jun. 2021, original-date: 2015-11-17T20:42:18Z. [Online]. Available: https://github.com/IntelRealSense/librealsense/blob/fb7b9de86bb0e621ac5aa6667ffbd421a53d6710/doc/post-processing-filters.md

[Sha17] F. Shaikh, "Difference between Deep Learning & Machine Learning," Apr. 2017, Último acceso: 19/11/2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/

[str20] "3D Scanning With Structured Light," Apr. 2020. [Online]. Available: https://bitfab.io/blog/3d-structured-light-scanning/

[The] The AI Guy, "YOLOv3 in the CLOUD : Install and Train Custom Object Detector (FREE GPU)." [Online]. Available: https://youtu.be/10joRJt39Ns

[TRC+16] R. Tomás, A. Riquelme, M. Cano, A. Abellan, and L. Jorda, "Structure from Motion (SfM): una técnica fotogramétrica de bajo coste para la caracterización y monitoreo de macizos rocosos," Oct. 2016.

[Uni15] N. Unidas, "Objetivos y metas de desarrollo sostenible," Sep. 2015, Último acceso: 04/11/2020. [Online]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[Wyn59] C. G. Wynne, "Lens designing by electronic digital computer: I," Proceedings of the Physical Society, vol. 73, no. 5, pp. 777–787, May 1959, publisher: IOP Publishing. [Online]. Available: https://doi.org/10.1088/0370-1328/73/5/310

[YK13] O. Yilmaz and F. Karakus, "Stereo and kinect fusion for continuous 3D reconstruction and visual odometry," Nov. 2013, pp. 115–118.

[ZPK18] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," arXiv:1801.09847, 2018.