COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

IMPLANTACIÓN DE UNA SOLUCIÓN DE PICK AND PLACE EN UN ROBOT INDUSTRIAL UTILIZANDO UN SISTEMA DE VISIÓN ARTIFICIAL BASADO EN REDES CONVOLUCIONALES

Autor: Daniel Sánchez de Pedro Rada

Director: Jaime Boal Martín-Larrauri

Madrid

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

IMPLANTACIÓN DE UNA SOLUCIÓN DE PICK AND PLACE EN UN ROBOT INDUSTRIAL UTILIZANDO UN SISTEMA DE VISIÓN ARTIFICIAL BASADO EN REDES CONVOLUCIONALES

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/21 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.

SPR

Fdo.: Daniel Sánchez de Pedro Rada Fecha: 17 / 06 / 21

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Firmado digitalmente por BOAL MARTIN LARRAURI JAIME - 05304600H
Fecha: 2021.06.17 18:57:07 +02'00'

Fdo.: Jaime Boal Martín-Larrauri Fecha: 17 / 06 / 21

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

IMPLANTACIÓN DE UNA SOLUCIÓN DE PICK AND PLACE EN UN ROBOT INDUSTRIAL UTILIZANDO UN SISTEMA DE VISIÓN ARTIFICIAL BASADO EN REDES CONVOLUCIONALES

Autor: Daniel Sánchez de Pedro Rada

Director: Jaime Boal Martín-Larrauri

Madrid

IMPLANTACIÓN DE UNA SOLUCIÓN DE PICK AND PLACE EN UN ROBOT INDUSTRIAL UTILIZANDO UN SISTEMA DE VISIÓN ARTIFICIAL BASADO EN REDES CONVOLUCIONALES

Autor: Sánchez de Pedro Rada, Daniel.
Director: Boal Martín-Larrauri, Jaime.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este proyecto se ha desarrollado un sistema que permite a un brazo robótico ordenar conjuntos de piezas de LEGO de distintos colores, dispuestas aleatoriamente en el tapete, detectándolas por visión artificial con CNNs. El sistema soporta piezas apiladas y piezas rotadas e implementa un método para optimizar la distancia recorrida por el robot basado en PSO.. El sistema implementa numerosas mejoras respecto a sistemas anteriores, notablemente la integración de los proyectos previos, la finalización del desarrollo de la interfaz, la comunicación bidireccional entre robot y PC, la corrección de la detección mediante varias imágenes y Template Matching, el trabajo por alturas para evitar colisiones y la eliminación de la discretización de la posición de las piezas en el tapete.

Palabras clave: Visión Artificial, Robótica, PSO

1. Introducción

En los últimos años, el interés de la industria en sistemas que combinen robótica con visión artificial para desempeñar tareas complejas ha crecido enormemente. Múltiples empresas e instituciones invierten cada vez más en este sector, organizan competiciones e investigan cuál es el límite al que se puede llevar la automatización de la robótica actualmente. Este proyecto supone una aportación a este esfuerzo colectivo.

2. Definición del proyecto

El objetivo del proyecto es desarrollar un sistema capaz de detectar piezas de LEGO dispuestas arbitrariamente en un plano de trabajo, pudiendo estar apiladas unas encima de otras o rotadas, y lograr que un brazo robótico equipado con una cámara logre ordenarlas todas según su color. Para ello, se partirá del sistema de visión artificial desarrollado en [1], y de la programación de los movimientos del robot introducida en [2], integrando ambos sistemas e implementando las mejoras pertinentes en cada caso.

3. Estado del arte

La conjunción de visión artificial y robótica es un tema que está causando un gran interés a nivel mundial. Muchas instituciones y empresas investigan en este campo, como es el caso de Amazon [3] u Ocado [4], llegando incluso a organizar competiciones en torno a este tema, como es el caso del *Amazon Robotics Challenge*.

De esta competición han surgido numerosos proyectos orientados al pick-and-place con visión artificial y objetos con agarres no triviales. Algunos ejemplos son el *Cartman*, desarrollado en [5], que destaca por su configuración cartesiana y su énfasis en el bajo coste; o el NimbRo de [6], caracterizado por su capacidad para determinar el punto óptimo para agarre por succión en cualquier objeto. Fuera del contexto de estas competiciones, otros proyectos como [7] son notables por su diseño de sistemas

robóticos combinados con visión artificial que colocan el énfasis en la planificación de trayectorias y evitar colisiones de objetos.

En este proyecto se empleará el algoritmo PSO para determinar la ruta óptima a seguir a la hora de recoger todas las piezas. Este problema es muy similar al problema del vendedor ambulante (o *Travelling Salesman Problem*, *TSP)*. PSO está concebido para resolver problemas de optimización continuos, pero el *TSP* es discreto. Por lo tanto hay que discretizar de alguna manera el algoritmo, lo cual ha sido investigado de muchas maneras distintas en la literatura disponible. Desde variantes de PSO como el *Bat Algorithm* [8], *Bird Swarm Algorithm* [9], hibridación con lógica difusa y *Simulated Annealing* [10], o redefinición de los operadores usados por el algoritmo [11]. Se han intentado muchas maneras distintas de discretizar este algoritmo usando el *TSP* como ejemplo.

4. Descripción del sistema empleado

El sistema se compone de tres elementos interconectados, la cámara Intel D-435, el brazo robótico ABB IRB 120 y un PC con MATLAB. La cámara se conecta al ordenador por USB y el robot y el PC están conectados por un socket TCP/IP. La siguiente figura resume la arquitectura del sistema y el rol de cada elemento.

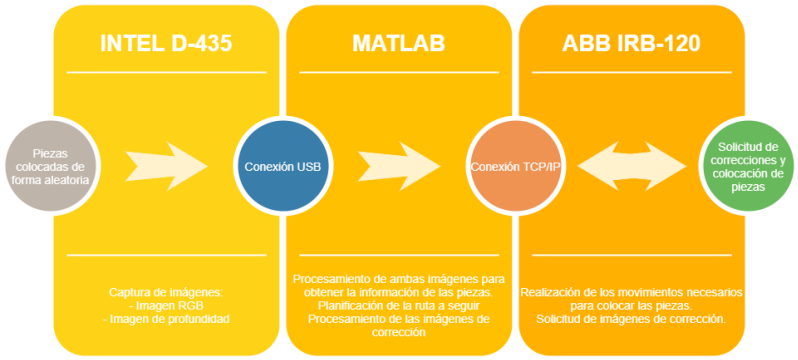


Figura 1. Arquitectura del sistema

El sistema funciona tomando una imagen de la mesa desde una posición predeterminada (también llamada inicial) determinando a partir de ella las ubicaciones aproximadas de las piezas. Detectadas todas las piezas, se eliminan aquellas que se encuentren por debajo de la altura de la pieza más alta, quedándonos así con el plano de trabajo más alto, a fin de evitar posibles colisiones del robot con pilas de piezas durante sus maniobras. A partir de las piezas más altas, se ejecuta una versión modificada del algoritmo PSO (*Particle Swarm Optimization*) para determinar el orden óptimo para recoger las piezas. Se eligió PSO por su rapidez a la hora de converger, como se evidencia en [12] y dado que se busca que este sistema eventualmente funcione en tiempo real, y para estudiar la viabilidad de una nueva manera de discretizar este algoritmo.

La implementación concreta de PSO para este caso ha combinado elementos tales como la reformulación de la distancia y la velocidad en el espacio de las soluciones posibles, la sustitución de la componente inercial de la velocidad de las partículas por una componente aleatoria, la modificación del valor de los pesos de las velocidades para acelerar la convergencia y el ajuste de la función de coste a nuestro caso, que no es un TSP al uso si no que requiere lidiar con que ciertas piezas se dejan siempre en ciertos puntos de entrega.

Una vez determinado el orden óptimo, el robot va a la ubicación aproximada de las piezas que se detectó inicialmente, tomando una nueva imagen para corregir las coordenadas de las piezas, que pueden haber sido sometidas a errores por diversos motivos. Con la segunda imagen se corrige la localización de la pieza y se instruye al robot que la tome y la deje en su punto de entrega correspondiente, según su color. Después se manda a la siguiente pieza en el orden y se repite el proceso hasta despejar el plano actual, volviendo a tomarse una imagen desde la posición inicial y continuando hasta ordenar todas las piezas. En la siguiente figura (derecha) se muestra un diagrama de flujo del proceso.

5. Resultados

El sistema se ha probado en diversas situaciones, combinando circunstancias adversas tales como apilamiento de piezas, rotación, condiciones de luz cambiantes, contigüidad de las piezas, etc. Las siguientes figuras (3,4 y 5) muestran 3 casos representativos en los que el sistema ha realizado un buen desempeño, junto con la detección realizada en cada caso. El sistema logra ordenar todas las piezas satisfactoriamente. Durante el desarrollo del proyecto, se han tenido que salvar dificultades heredadas de los sistemas anteriores tales como: la distorsión radial de la cámara, resuelta mediante la toma de imágenes adicionales cerca de la pieza; la discretización de las posiciones de las piezas en el tapete, resuelta mediante modelos lineales que relacionan las coordenadas en la imagen con posición relativa al robot; los fallos de agarre al lidiar con piezas giradas, resueltas al incorporar una corrección en función del ángulo girado y del radio de la pinza a la muñeca del robot; la colisión de mensajes en el socket TCP y la falta de comunicación bidireccional.

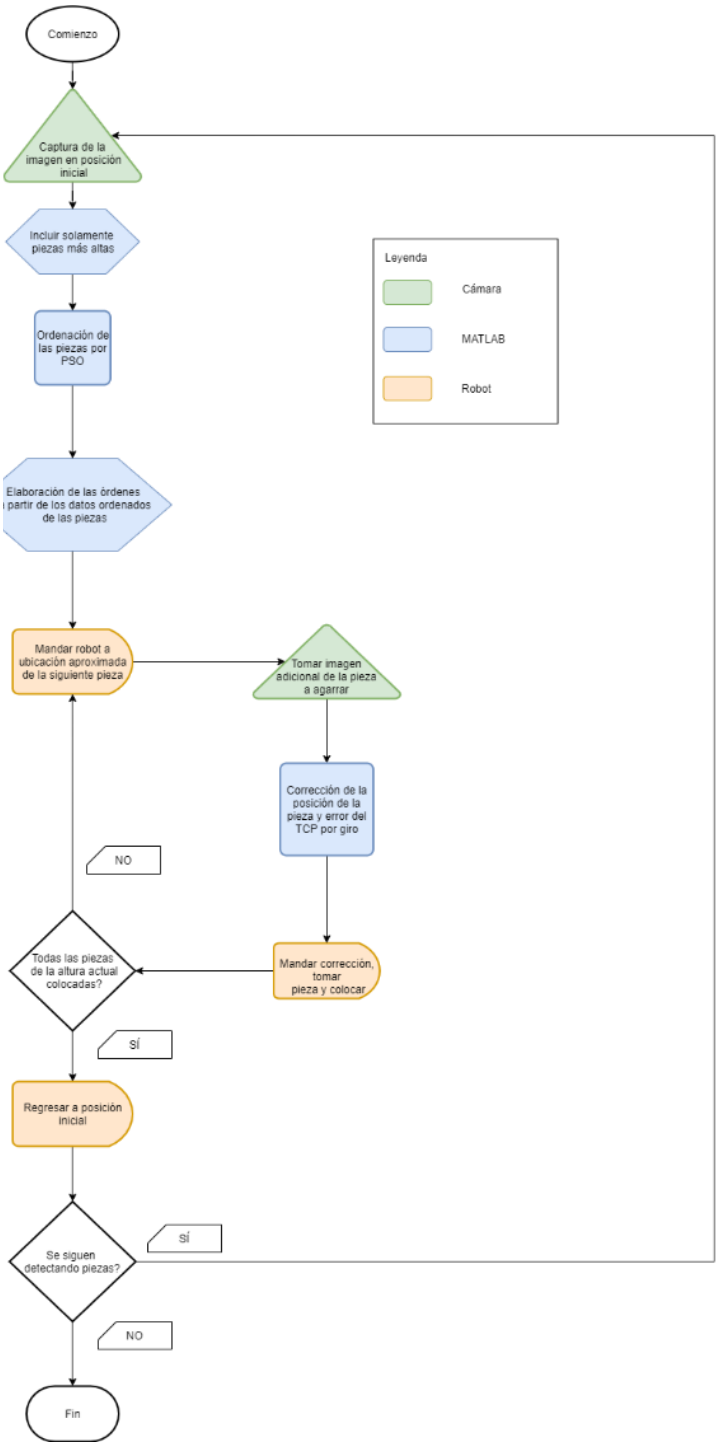
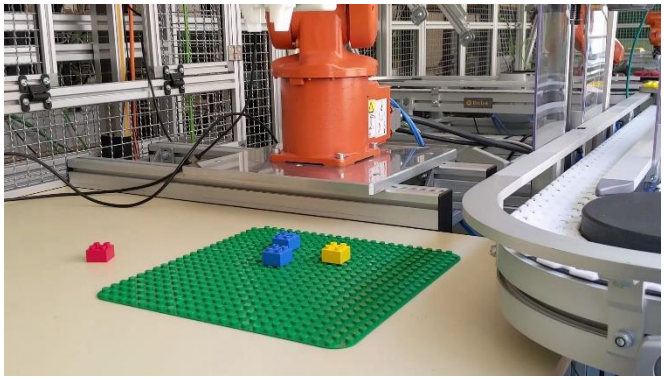


Figura 2. Diagrama de flujo del proceso



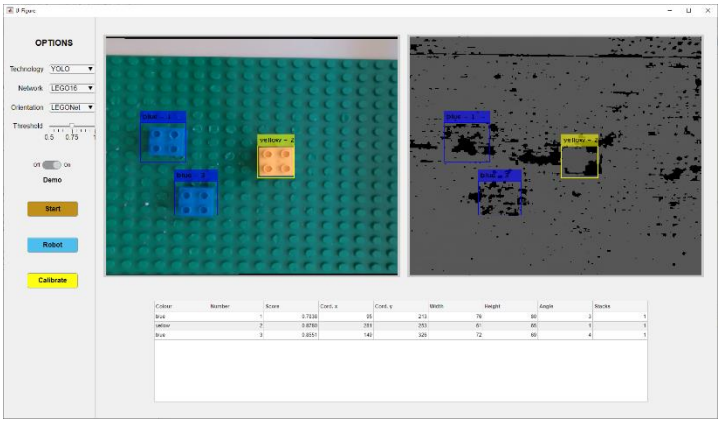
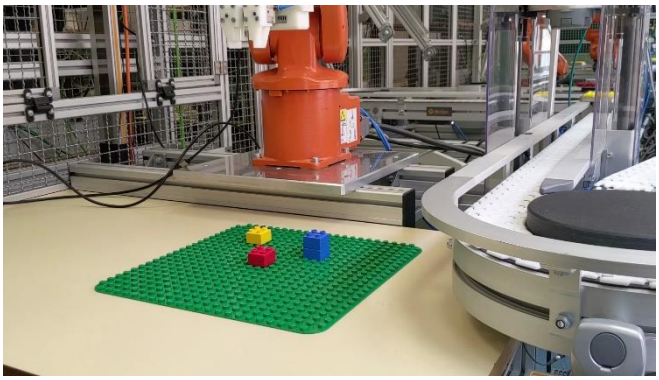


Figura 3. Condiciones del caso 1



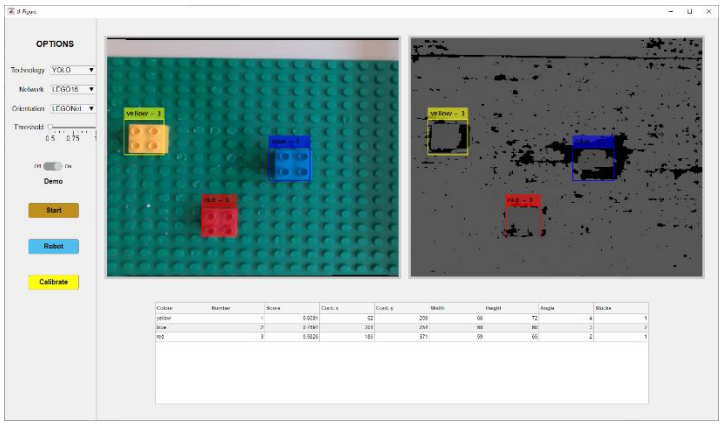
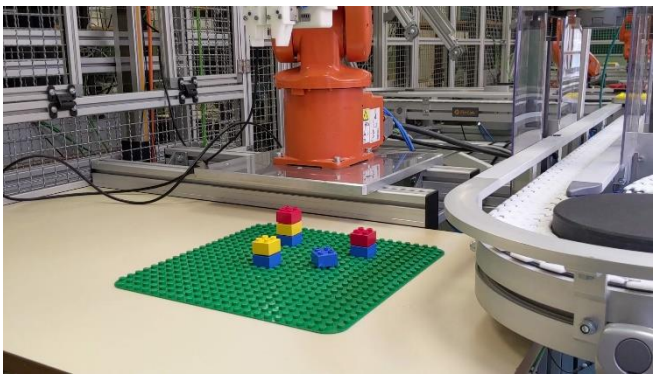


Figura 3. Condiciones del caso 2



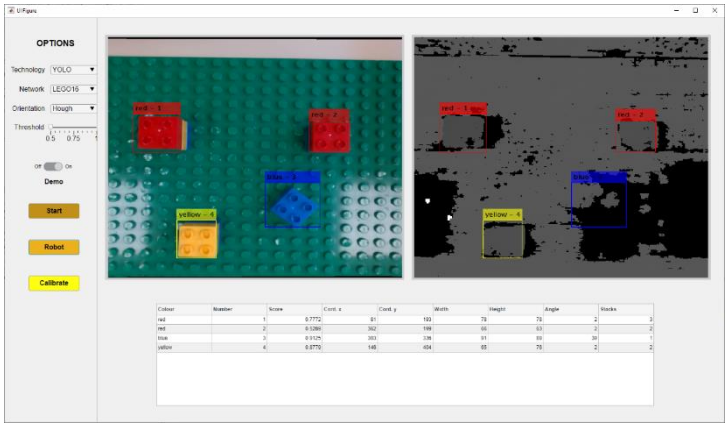


Figura 5. Condiciones del caso 3

6. Conclusiones

El sistema funciona satisfactoriamente en cuanto a la integración de los sistemas anteriores, la introducción de la etapa de corrección y la planificación del orden en el que se deben recoger las piezas. La siguiente tabla resume las diferencias entre las iteraciones anteriores del sistema y la actual.

Sistemas anteriores	Sistema actual
No tiene manera de evitar colisiones	Trabaja por alturas para evitar colisiones con pilas de piezas
Sólo el análisis de máscaras de color es compatible con el movimiento del robot	Tanto análisis de máscaras como detección por CNN son compatibles
Necesita discretizar las posiciones de las piezas en el tapete	Trabaja sin discretizar posiciones
Toma las piezas en orden aleatorio	Determina una ruta óptima para tomar las piezas
La comunicación con el robot es unidireccional	La comunicación es bidireccional
No se consideran posibles colisiones de mensajes	Se garantiza que el socket siempre está limpio para que no haya colisiones
Se usa una sola imagen en todo el proceso	Se usan varias imágenes en el proceso para corregir la detección
No hay posibilidad de recentrar las *bounding boxes*	Uitiliza *Template Matching* para recentrar las *bounding boxes* (en condiciones adecuadas)

No obstante, todavía se ve sometido a algunas limitaciones relacionadas con la etapa de detección de las piezas, mayormente relacionadas con la diferencia entre las condiciones de entrenamiento de la CNN y las condiciones reales de los experimentos, y con la detección de ángulos que es indiferente a si las piezas están rotadas en sentido horario o antihorario. Estos problemas podrían ser solucionados en proyectos futuros.

7. Referencias

[1] I. Ortiz de Zúñiga Mingot, "Optimización del sistema de visión artificial de un robot industrial para una aplicación de pick and place," *Trabajo Final de Grado, ICAI*, Jul. 2020.

[2] A. Berjón Valles, "Integración de un Sistema de Visión Artificial en la Mano de un Robot Industria," *Trabajo Final de Grado, ICAI*, Jul. 2019.

[3] "Amazon is planning a $40M robotics hub near Boston | TechCrunch." https://techcrunch.com/2019/11/06/amazon-is-planning-a-40m-robotics-hub-near-boston/ (accessed Oct. 06, 2020).

[4] “Ocado overtakes Tesco as most valuable UK retailer,” *BBC News*, Sep. 30, 2020. Accessed: Oct. 06, 2020. [Online]. Available: https://www.bbc.com/news/business-54352540
[5] D. Morrison *et al.*, “Cartman: The Low-Cost Cartesian Manipulator that Won the Amazon Robotics Challenge,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7757–7764.
[6] M. Schwarz *et al.*, “NimbRo picking: Versatile part handling for warehouse automation,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3032–3039.
[7] J. Vachálek, L. Čapucha, P. Krasňanský, and F. Tóth, “Collision-free manipulation of a robotic arm using the MS Windows Kinect 3D optical system,” in *2015 20th International Conference on Process Control (PC)*, 2015, pp. 96–106.
[8] Y. Saji, M. E. Riffi, and B. Ahiod, “Discrete bat-inspired algorithm for travelling salesman problem,” in *2014 Second World Conference on Complex Systems (WCCS)*, Nov. 2014, pp. 28–31. doi: 10.1109/ICoCS.2014.7060983.
[9] M. Lin, Y. Zhong, J. Lin, and X. Lin, “Discrete Bird Swarm Algorithm Based on Information Entropy Matrix for Traveling Salesman Problem,” *Mathematical Problems in Engineering*, Oct. 30, 2018. https://www.hindawi.com/journals/mpe/2018/9461861/ (accessed Dec. 30, 2020).
[10] R. F. Abdel-Kader, “Fuzzy Particle Swarm Optimization with Simulated Annealing and Neighborhood Information Communication for Solving TSP,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 2, no. 5, Art. no. 5, Nov. 2012, doi: 10.14569/IJACSA.2011.020503.
[11] E. F. G. Goldbarg, M. C., and G. R. de Souz, “Particle Swarm Optimization Algorithm for the Traveling Salesman Problem,” in *Traveling Salesman Problem*, F. Greco, Ed. InTech, 2008. doi: 10.5772/5580.
[12] S. Sengupta, S. Basak, and R. II, “Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives,” Apr. 2018, doi: 10.3390/make1010010

IMPLEMENTATION OF A PICK AND PLACE SOLUTION FOR INDUSTRIAL ROBOTS USING CONVOLUTIONAL NEURAL NETWORK BASED COMPUTER VISION

Author: Daniel Sánchez de Pedro Rada.
Supervisor: Jaime Boal Martín-Larrauri.
Collaborating entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

In this project a system has been developed to allow a robotic arm to sort different sets of multicolour LEGO pieces, randomly laid out on the work surface, detecting them through computer vision and CNNs. The system can manage stacked pieces as well as rotated pieces. A method to optimize the distance travelled by the robot arm has been implemented, based on PSO. The system features many improvements over previously developed solutions, notably the integration of all previous work, a completed and fully developed interface, two-way communication between the robot and the commanding PC, correction of the piece detection through taking several images and Template Matching, height discrimination to avoid collision between the robot and stacks of pieces and eliminating the discretization of the possible positions of the pieces on the work Surface.

Key words: Computer vision, Robotics, PSO.

1. Introduction

Over the last few years, the industry's interest in systems that combine robotics and computer vision to carry out complex tasks has grown tremendously. Several companies and institutions invest heavily in this sector, organize competitions and research to find out how far robotics can be pushed. This project is one more contribution to this collective effort.

2. Project definition

The aim of this project is to develop a solution capable of detecting arbitrarily placed LEGO pieces, whether stacked or rotated, and instructing a robotic arm equipped with a camera to sort the pieces by colour. To that end, the computer vision system developed in [1] and the robot movement instructions devised in [2] will be employed, integrating both systems and introducing new features in the process.

3. State of the art

The combination of computer vision and robotics has been raising a great deal of interest world wide. Many companies and institutions research this field, such as Amazon [3] or Ocado [4], even creating challenges around it such as the *Amazon Robotics Challenge.*

From said challenge, several solutions combining computer vision and pick-and-place have been developed. For instance the Cartman, shown in [5], featuring low-cost manufacturing and a cartesian layout or the NimBro [6], designed to find the optimal point to employ suction gripping for any item. Other projects such as [7] are also notable as they design solutions that combine computer vision and robotics while emphasizing the path planning aspect in order to avoid collisions.

In this project, the PSO algorithm will be employed to determine what the optimal routing is when it comes to picking the LEGO pieces. This problem is similar in nature to the Travelling Salesman Problem (TSP). PSO, however, is designed to solve continuous optimization problems, while the TSP is a discrete problem. Therefore, PSO must be adapted somehow. Current literature details many different ways to carry out this adaptation. Variants on PSO such as the Bat Algorithm [8], Bird Swarm Algorithm [9], hybridizing PSO with fuzzy logic and Simulated Annealing [10] or redefining the algorithms operators[11] and many others have been attempted.

4. System description

The system is composed of three interconnected elements: the Intel D-435 camera, the ABB IRB-120 robotic arm and a PC running MATLAB. The following figure summarizes the system architecture.

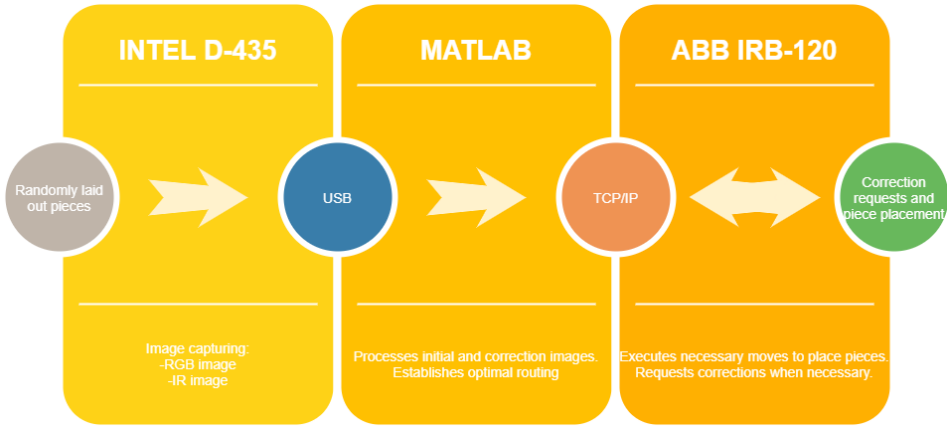


Figure 4. System architecture

The system works by taking an image of the table from a predetermined position (a.k.a initial position) from which the approximate location of the pieces is determined. Once all pieces are detected, those which are lower than the highest detected piece are disregarded, in order to work plane by plane and avoid collisions with high stacks of pieces. The modified version of PSO is then executed to find the order in which to best pick the pieces. PSO was chosen for this task for two reasons: Firstly, it converges quite fast compared to other optimization algorithms, as shown in [12], which will be important since this system is intended to work in real-time; Secondly, to study the viability of a new way to discretize this algorithm.

The implementation of PSO used for this case combines elements such as redefining the concepts of particle distance and speed, replacing the inertial component of the velocity equation with a random component, updating the weights of the velocity equation components with each iteration to facilitate convergence and the modification of the cost function compared to the traditional TSP. Unlike TSP, where the points can be visited in any order, in this case every piece must immediately go to its designated drop-off point, according to its colour, and there is a fixed initial position.

Once the optimal route has been determined, the robot goes to the corresponding piece's approximate location and requests a new image to correct the piece's position. Errors in the piece's initial position can be due to radial distortion, rotated pieces or other reasons. Once the correction has been applied, the robot is instructed to pick up the piece and drop it off at its corresponding drop-off point. The robot is then directed toward the next piece, and continues this process until it clears the current plane. Once a plane is cleared, the robot goes back to the initial position and scans again for pieces and the process is repeated until all pieces have been sorted. The following figure (right) shows the flowchart for this process.

5. Results

The solution has been tested in several different situations, combining adverse circumstances such as stacked pieces, rotated pieces, harsh lighting conditions, etc. The following figures (3, 4 & 5) show such situations along the detection of the pieces for each of them. The solution manages to sort all pieces in a satisfactory manner. During the development of the soultion, many limitations inherited from previous iterations of the project have been dealt with. Some of these limitations include: the camera's radial distortion, which is handled by the correction step and the second image; the discretization of the piece positions implemented previously, which has been replaced by a linear model correlating image coordinates with relative coordinates to the robot; grasping errors due to the tool displacement when grabbing rotated pieces, which are now corrected through geometric considerations of the rotated angle and the distance from the tool to the robot's wrist and the lack of two-way communication and message collisions happening in the TCP socket.

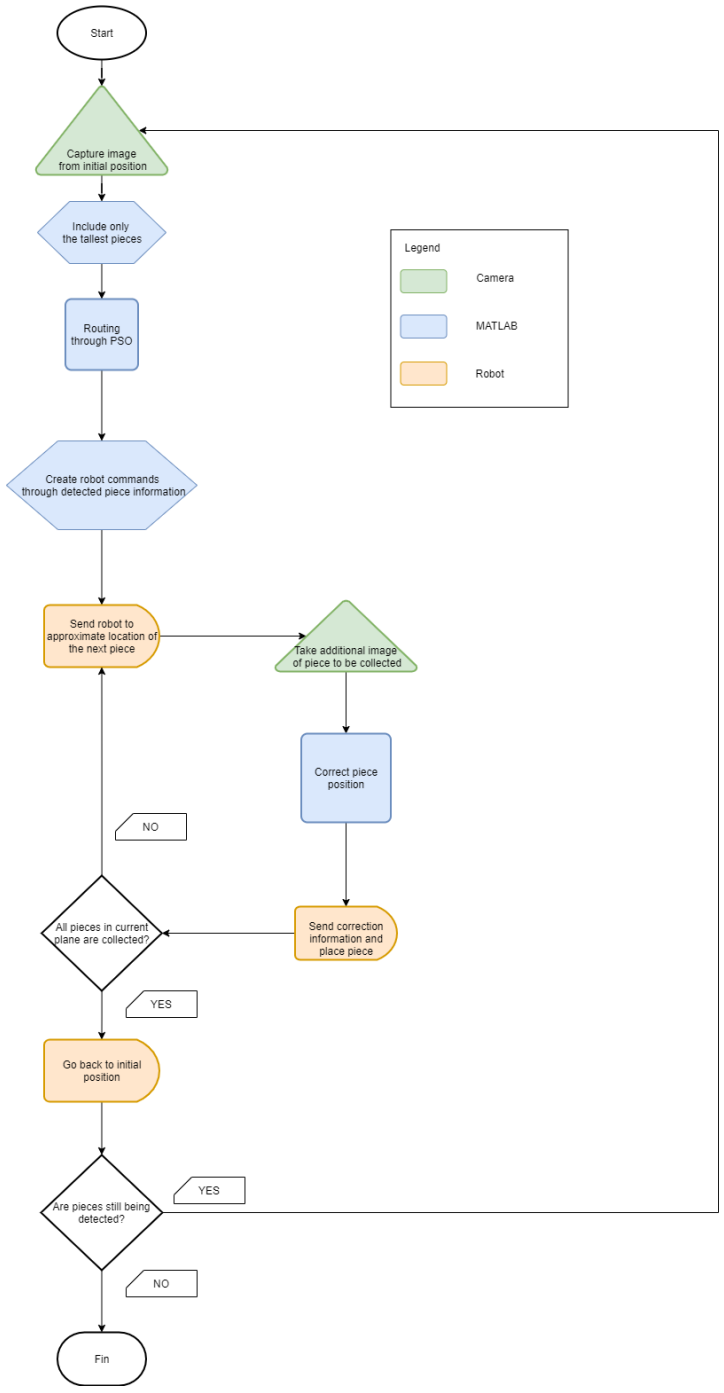


Figure 5. Pick-and-place process flowchart

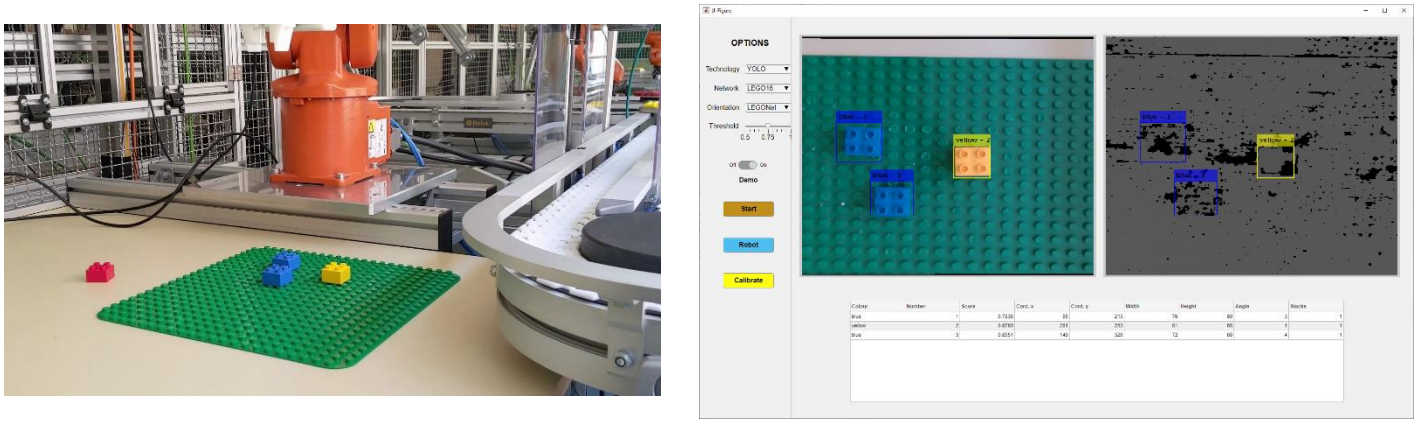


Figure 3. Conditions for case 1

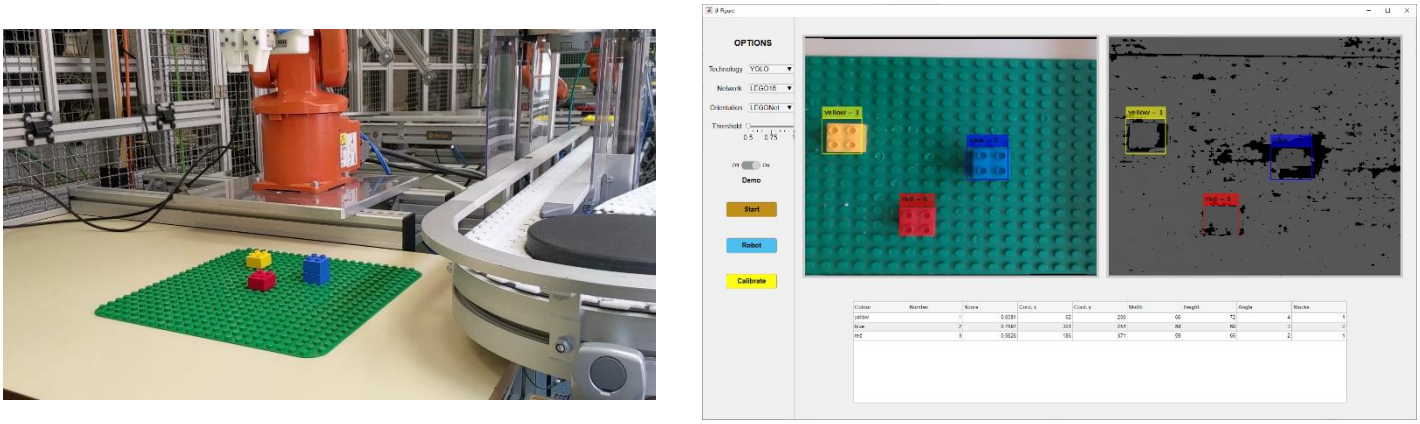


Figure 4. Conditions for case 2

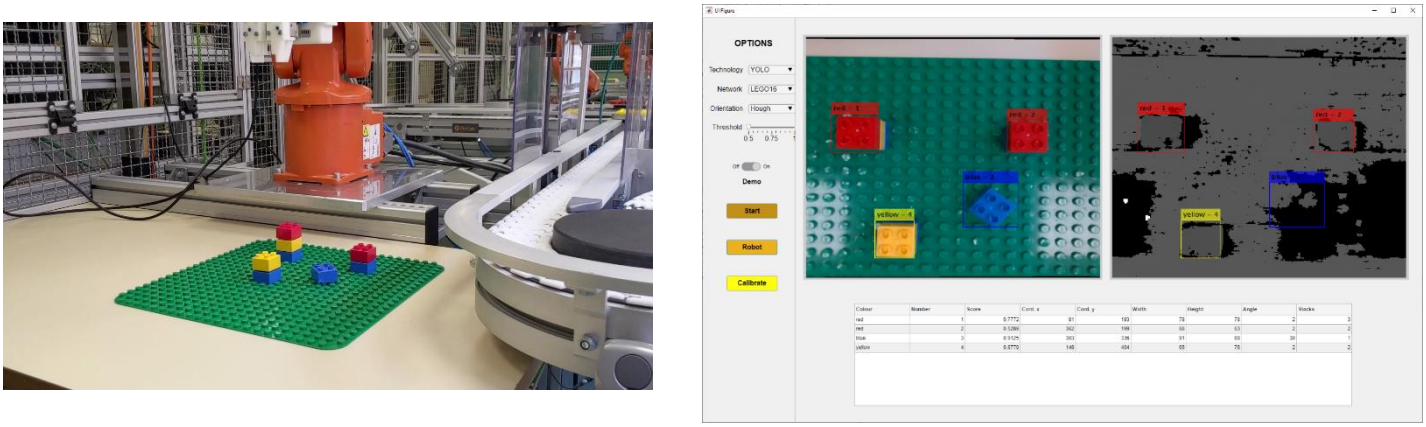


Figure 5. Conditions for case 3

6. Conclusions

The system performs satisfactorily when it comes to integrating previous solutions, adding a correction step and determining the routing to pick up the pieces. The following table summarizes the differences and improvements between this solution and previous iterations.

Previous solutions	Current solution
No way to avoid collisions	Plane-by-plane method avoids collisions with high stacks of pieces
Only colour mask analysis could be effectively used with the robot	Both colour mask and CNNs are compatible with the robot
Needs to discretize the positions of the pieces on the surface	Works without discretization
Takes pieces in a random order	Determines the optimal routing to take the pieces
Communication with the robot is one-way	Communication is now two-way
Message collisions are not considered	The socket is regularly cleared to ensure collision-free communication
One image is used for the whole process	Several images are used to increase accuracy
No way of recentering the bounding boxes	Bounding boxes can be recenterd through Template Matching

Some limitations related to the detection step are still present. These limitations are likely due to the different training conditions for CNN, which are unlike the laboratory conditions, and the lack of differentiation between positive and negative angles during the detection phase. Future work on these issues is advised.

7. References

[1] I. Ortiz de Zúñiga Mingot, "Optimización del sistema de visión artificial de un robot industrial para una aplicación de pick and place," *Trabajo Final de Grado, ICAI*, Jul. 2020.

[2] A. Berjón Valles, "Integración de un Sistema de Visión Artificial en la Mano de un Robot Industria," *Trabajo Final de Grado, ICAI*, Jul. 2019.

[3] "Amazon is planning a $40M robotics hub near Boston | TechCrunch." https://techcrunch.com/2019/11/06/amazon-is-planning-a-40m-robotics-hub-near-boston/ (accessed Oct. 06, 2020).

[4] "Ocado overtakes Tesco as most valuable UK retailer," *BBC News*, Sep. 30, 2020. Accessed: Oct. 06, 2020. [Online]. Available: https://www.bbc.com/news/business-54352540

[5] D. Morrison *et al.*, “Cartman: The Low-Cost Cartesian Manipulator that Won the Amazon Robotics Challenge,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7757–7764.
[6] M. Schwarz *et al.*, “NimbRo picking: Versatile part handling for warehouse automation,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3032–3039.
[7] J. Vachálek, L. Čapucha, P. Krasňanský, and F. Tóth, “Collision-free manipulation of a robotic arm using the MS Windows Kinect 3D optical system,” in *2015 20th International Conference on Process Control (PC)*, 2015, pp. 96–106.
[8] Y. Saji, M. E. Riffi, and B. Ahiod, “Discrete bat-inspired algorithm for travelling salesman problem,” in *2014 Second World Conference on Complex Systems (WCCS)*, Nov. 2014, pp. 28–31. doi: 10.1109/ICoCS.2014.7060983.
[9] M. Lin, Y. Zhong, J. Lin, and X. Lin, “Discrete Bird Swarm Algorithm Based on Information Entropy Matrix for Traveling Salesman Problem,” *Mathematical Problems in Engineering*, Oct. 30, 2018. https://www.hindawi.com/journals/mpe/2018/9461861/ (accessed Dec. 30, 2020).
[10] R. F. Abdel-Kader, “Fuzzy Particle Swarm Optimization with Simulated Annealing and Neighborhood Information Communication for Solving TSP,” *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 2, no. 5, Art. no. 5, Nov. 2012, doi: 10.14569/IJACSA.2011.020503.
[11] E. F. G. Goldbarg, M. C., and G. R. de Souz, “Particle Swarm Optimization Algorithm for the Traveling Salesman Problem,” in *Traveling Salesman Problem*, F. Greco, Ed. InTech, 2008. doi: 10.5772/5580.
[12] S. Sengupta, S. Basak, and R. II, “Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives,” Apr. 2018, doi: 10.3390/make1010010

Índice de la memoria

Índice de figuras

Capítulo 1. INTRODUCCIÓN

1.1 CONTEXTO

La llegada de la Cuarta Revolución Industrial es ya un fenómeno indiscutible. Cada vez más las grandes empresas apuestan por la utilización de sistemas ciberfísicos, caracterizados no sólo por su alto grado de automatización, sino porque dotan a las máquinas de la capacidad de tomar decisiones propias a la hora de resolver sus tareas.

Los beneficios de emplear este tipo de sistemas son múltiples: menor necesidad de atención humana en los procesos automáticos, mayor flexibilidad en la producción, posibilidad de integrar distintas máquinas y sistemas con facilidad mediante comunicaciones M2M (*Machine to Machine*), etc. La ventaja competitiva que supone participar de estas tecnologías es evidente.

Si se quisieran algunos ejemplos reales de lo realmente útil que puede ser la llamada Industria 4.0, no hay más que ver al gigante Amazon. Debido al intensivo uso que hacen de manipuladores robóticos en todas sus actividades logísticas, no es de extrañar que hayan invertido grandes cantidades de dinero en centros de investigación sobre robótica [1]. Su uso cada vez más tecnológicamente innovador de dichos sistemas es lo que le permite enviar sus productos con tanta rapidez y ha contribuido enormemente a que se convierta en la mayor empresa de venta online del mundo.

Otro fantástico ejemplo sería el de la británica Ocado. Esta cadena de supermercados no tiene tiendas físicas. En su lugar, se dedican exclusivamente a envíos de productos alimenticios y similares a domicilio. La piedra angular de su funcionamiento es su avanzado sistema logístico, con un gran grado de digitalización, y que hace un uso extensivo de la robótica. Este aprovechamiento que hace de la Industria 4.0 le ha permitido adelantar a su principal competidor, Tesco, a mediados de este año [2]

En este contexto, se ha estado trabajando en ICAI en montar un sistema robótico de *pick and place* para una fábrica o línea de montaje. Este sistema emplea visión artificial para que el robot sea capaz de identificar los objetos automáticamente y pueda luego reordenarlos según una configuración dada.

1.2 JUSTIFICACIÓN

La motivación detrás de este proyecto, así como de muchos otros relacionados con el área de automatización, es la de mejorar la calidad de vida las personas. En este caso, dotando a los brazos robóticos de una mayor autonomía se logra reducir tareas de desplazamiento de materiales extremadamente repetitivas y en algunos casos peligrosas, permitiendo así que los trabajadores puedan dedicar su tiempo a actividades más edificantes y con mayor valor añadido. Esto a su vez se traducirá en una mayor productividad de los mismos, ya que, al eliminar las tareas más monótonas, podrán recibir una mayor estimulación mental y evitar todos los riesgos a la salud mental asociados a los trabajos repetitivos.

Dada la evidente actualidad de los sistemas robóticos con visión artificial, este proyecto supone una gran oportunidad para ICAI por numerosos motivos: mantenerse al día en las tecnologías más punteras, fortalecer su conocimiento sobre las áreas de visión artificial y robótica, brindar a los alumnos nuevas oportunidades para aprender sobre estos campos, modernizar la minifábrica, mejorar su imagen de cara a las empresas del sector… Este proyecto puede ser una plataforma sobre la que se apoyen numerosos desarrollos futuros en el campo de la robótica industrial.

1.3 OBJETIVOS

- Conseguir que el brazo robótico clasifique piezas de LEGO dispuestas de forma aleatoria sobre el tapete de trabajo, pudiendo estar apiladas o rotadas
- Integrar el reconocimiento con el movimiento del robot (envío de órdenes) e incluir comunicación bidireccional

- Lograr que el sistema funcione en cualquier condición (iluminación, piezas en los bordes, piezas contiguas…)
- Optimizar la ruta del robot a la hora de recoger las piezas, minimizando la distancia recorrida

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 REDES NEURONALES CONVOLUCIONALES Y YOLO

Para detectar las piezas de LEGO, se utilizan las CNN desarrolladas por Ignacio Ortiz de Zúñiga Mingot en [3]. En esta sección se resume brevemente su funcionamiento.

Las CNN son un tipo de red neuronal muy utilizada en el campo de la visión artificial. Se caracterizan por estar dotadas de "capas de convolución". Su comportamiento es el siguiente: En primer lugar, se toma la imagen, en formato RGB que supone una matriz tridimensional de anchura y altura iguales a la resolución de la imagen. Esta matriz de entrada se subdivide en otras sub-matrices que son multiplicadas por una tercera matriz, la matriz de "kernel" o filtro. Posteriormente se suma el resultado de esta multiplicación de las matrices para llegar a la salida de la capa de convolución. Este proceso se repite hasta barrer toda la imagen, dando lugar a una matriz de salida de menor tamaño que la de entrada. A partir de ahí, se pueden aplicar técnicas de *Deep learning* para encontrar los patrones deseados en la imagen a partir de la matriz de salida. Este proceso se ilustra en la Figura 6 [4].

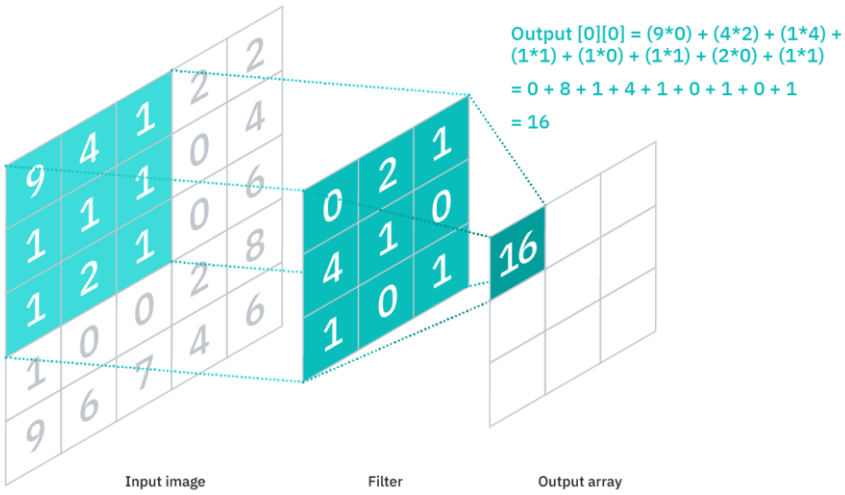


Figura 6. Descripción del funcionamiento de las capas convolucionales

No obstante, esta estrategia de procesado de la imagen, aunque precisa, puede resultar lenta en ocasiones. Por ello, para facilitar el futuro funcionamiento en tiempo real del sistema, se emplea una red convolucional de tipo YOLO (*You Only Look Once*). Este tipo de arquitectura se caracteriza por analizar toda la imagen en su conjunto de golpe, asociando a distintas regiones la probabilidad de pertenecer a una clase u otra. Esto le confiere una mayor velocidad que otras arquitecturas y será la opción elegida en este proyecto.

2.2 PARTICLE SWARM OPTIMIZATION (PSO)

Uno de los elementos sobre los que este proyecto mejora respecto a iteraciones anteriores es en la planificación de la ruta del robot. El algoritmo PSO es un algoritmo de optimización pensado para la resolución de problemas en los que el espacio de soluciones es continuo, inspirado en la manera que muchos animales sociales (pájaros, abejas, etc.) usan para orientarse. Su funcionamiento es el siguiente:

1. Inicializar N partículas (posibles soluciones) aleatorias P_i comprendidas en el espacio de posibles soluciones y sus respectivas velocidades V_i
2. Inicializar los hiperparámetros w (inercia), $c1$ (componente cognitivo) y $c2$ (componente social)
3. Durante cada iteración t:
    a. Evaluar cada partícula de acuerdo a la función de coste $F(P_i)$. Guardar la partícula con mejor evaluación de todas P_b y la mejor evaluación personal de cada partícula a lo largo de todas las iteraciones P_{ip}
    b. Actualizar la velocidad de cada partícula V_i de acuerdo a la **Ecuación 1**

$$V_i(t+1) = w * V_i(t) + c1 * \left(P_{ip} - P_i(t)\right) + c2 * \left(P_b - P_i(t)\right)$$

    c. Actualizar la posición de cada partícula P_i de acuerdo a la **Ecuación 2**

$$P_i(t+1) = P_i(t) + V_i(t+1)$$

    d. (Opcional) Actualizar el valor de los hiperparámetros para forzar la convergencia
4. Finalizadas todas las iteraciones, devolver P_b

La Figura 7 [5], presenta una animación ilustrativa del funcionamiento de PSO. Las "x" representan las partículas, que se mueven por el espacio de soluciones posibles en la dirección de su velocidad, denotada por la flecha. El espacio de soluciones tiene valores de la función de coste asociados que se indican con el gradiente de color. En el caso de la figura y en el del proyecto, se trata de un problema de minimización.

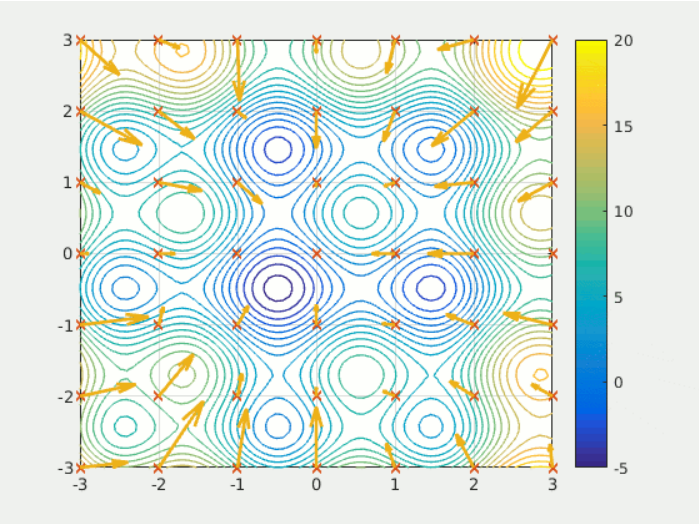


Figura 7. Animación ilustrativa de PSO

Este algoritmo se presta muy fácilmente a la resolución de problemas en los que el espacio de soluciones es continuo (por ejemplo, encontrar la posición en la que un lago tiene mayor profundidad) y en los que los conceptos de "distancia" y "velocidad" son fácilmente aplicables. El caso actual, sin embargo, no es de este tipo. En este proyecto se utiliza PSO para determinar el orden óptimo en el que recoger las piezas detectadas por la cámara. Las posibles soluciones de este problema (los distintos órdenes en los que podemos recoger las piezas) no forman un espacio continuo, si no discreto, y la "distancia" entre dos soluciones no está clara (¿Cuál es la distancia entre recoger las piezas en el orden [A B C] o [C A B]?) Por lo tanto, habrá que hacer ciertas adaptaciones a PSO que se expondrán en el 4.2

2.3 *TEMPLATE MATCHING*

El *Template Matching* es una técnica de visión artificial para detección de objetos. Consiste en tomar una imagen de referencia (*template*) y comparar la imagen objetivo píxel por píxel

con la referencia hasta encontrar la región de mayor coincidencia, donde estaría el objeto. Esta aplicación funciona bien cuando hay pocos objetos que detectar y las imágenes se van a tomar en condiciones muy similares (como es este caso, en el que sólo se detectan piezas de LEGO a partir de imágenes tomadas siempre desde el mismo ángulo a la misma altura). No obstante, dada su exhaustividad es muy lento y computacionalmente caro, por lo que no vale la pena utilizarlo como método principal de detección.

En su lugar, se emplea esta técnica como forma de corrección de la detección lograda por la CNN. En algunas circunstancias, la *bounding box* devuelta por la CNN contiene a la pieza, como cabría esperar, pero se encuentra descentrada, lo que puede causar que el robot no logre agarrarla. Para evitar este tipo de situaciones, se empleará *Template Matching* exclusivamente sobre la región de la *bounding box*, aliviando así las limitaciones de esta técnica.

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 PROYECTOS ANTERIORES

Este proyecto parte de la base de otros proyectos anteriores que han asentado los cimientos del sistema de visión artificial. Nos referimos a los proyectos de Ana Berjón Valles [6] e Ignacio Ortiz de Zúñiga Mingot [3]. Estos proyectos han logrado desarrollar un algoritmo de visión artificial capaz de identificar la posición y orientación de piezas de LEGO Duplo sencillas. También se logró una implementación básica del sistema de pick-and-place con el brazo robótico terminado en pinzas. La primera versión del sistema, desarrollada en [6] tenía algunas dificultades a la hora de lidiar con piezas rotadas, reflejos causados por la iluminación, identificación de piezas a distintas alturas, etc. Estos problemas derivaban, en parte, de los métodos empleados (máscaras de color, algoritmo de Canny…). No obstante, dadas las condiciones oportunas, el sistema lograba agrupar las piezas por colores. Otro de los campos a mejorar en este proyecto es la automatización de la generación de órdenes de movimiento para el robot, al tiempo que se evitan colisiones con objetos que pueda haber en el área de trabajo.

En la segunda versión, llevada a término en [3], se solventan los problemas mencionados anteriormente mediante el desarrollo de algoritmos basados en redes neuronales convolucionales, y se resuelve, manualmente, la comunicación del robot con MATLAB. Pese a su labor, todavía quedaron algunas mejoras pendientes para este sistema que el proyecto aquí expuesto pretende resolver, fundamentalmente: automatización de la comunicación del robot con MATLAB, implementar un sistema anticolisión, robustecer el sistema frente a variaciones en el sistema de coordenadas, trabajar con piezas simultáneamente rotadas y apiladas, etc.

3.2 REDES NEURONALES CONVOLUCIONALES

Por supuesto, las redes neuronales convolucionales jugarán un papel importante no sólo en este proyecto, sino en el futuro del sector industrial. Las CNN (*Convolutional Neural Networks)* son una técnica de *deep learning* extremadamente potente y versátil, y se han encontrado usos para ellas en numerosas aplicaciones de relevancia para la industria. En los siguientes párrafos, se mostrarán algunas de esas aplicaciones y de presentar las investigaciones y proyectos que se están desarrollando en torno a las CNN actualmente.

Una parte importante de las investigaciones sobre CNNs está relacionada con la visión artificial. El artículo contenido en [7] propone una herramienta para diseñar CNNs para facilitar la construcción y testeo de estos algoritmos. La herramienta está pensada para su uso en aplicaciones relacionadas con la detección de defectos en piezas industriales. La herramienta destaca por estar diseñada en MATLAB.

[8] diseña un método basado en un tipo concreto de CNN, la *Broad Convolutional Neural Network*, a fin utilizar visión artificial para detección de piezas defectuosas. Este algoritmo surge como respuesta a dos de los grandes problemas de las CNN: la necesidad de grandes cantidades de datos y la obligación de tener que reentrenar el modelo cuando aparece una nueva clase (en este caso, un nuevo tipo de defecto). El algoritmo propuesto por los autores se caracteriza por dotar a la red neuronal de la capacidad de predecir los nuevos tipos de defecto que puedan ocurrir, generándose así nuevas categorías de salida (posibles defectos) y características que buscar en la entrada (rasgos de la imagen de la pieza)

La publicación de [9] sugiere un método para descubrir defectos en las aspas de turbinas aeronáuticas. La aportación de este artículo consiste en elaborar una FCNN (*Fully Convolutional Neural Network*) multi-escala. Esta red consiste en una serie de redes neuronales entrenadas para la detección de fallos a distintas escalas (a nivel de imagen completa, por sectores, píxel por píxel, etc.) que luego se integran en la red en una arquitectura sin capas totalmente conectadas, de ahí el nombre de *Fully Convolutional*, pues sólo utiliza capas convolucionales. De acuerdo con los autores, este planteamiento pretende

solucionar el problema de que la configuración de las CNN resulta a veces demasiado específica para el problema concreto que se está tratando, utilizando un esquema más genérico.

En industrias en las que la superficie del producto es particularmente importante, como por ejemplo la industria maderera, también son relevantes este tipo de aplicaciones de las CNN. Una solución para este problema es propuesta en [10]. En este caso, la detección de defectos en las superficies resulta difícil para seres humanos, y la CNN diseñada logra un mejor rendimiento que sus contrapartes humanas. Al emplear una combinación de una red poco profunda basada en LeNet y una red más profunda basada en VGG-19, consiguieron salvar la dificultad de requerir grandes cantidades de imágenes para entrenar el modelo. Consiguieron esto empleando transferencia de aprendizaje (*transfer learning*) de la red básica, que había sido entrenada solamente con imágenes de madera, a la más profunda, que venía pre-entrenada.

Detectar fallos en diversos componentes no es lo único de lo que las CNN son capaces. En industrias en las que la inspección de materiales es gran parte del valor añadido, como por ejemplo la producción de perlas, las CNN también pueden suponer una manera de eliminar tareas repetitivas para el ser humano. El artículo de [11] propone un sistema diseñado precisamente para tal fin. Las CNN en este caso particular están especialmente adaptadas al caso. Su capacidad de determinar automáticamente cuáles son los rasgos importantes que determinan el valor de la perla permiten prescindir de diseñarlos manualmente y tener que emplear a expertos para ello. El sistema emplea varias imágenes de la misma perla y una red neuronal basada en *AlexNet*, de forma similar a [3].

Otra aplicación interesante de las CNN es para cuestiones médicas. El algoritmo diseñado en [12] pretende utilizar una CNN para analizar encefalogramas intracraneales y poder predecir cuándo se producirá un ataque epiléptico. Las CNN resultan especialmente interesantes para este caso, ya que a diferencia de otras técnicas de *machine learning*, como por ejemplo *Support Vector Machine* (SVM), logran generalizar mucho mejor y producen modelos válidos para muchos más pacientes.

Uno de los usos más innovadores que se le está dando a las CNN es para identificar anomalías y posibles ataques cibernéticos a sistemas de control de plantas de producción. El riesgo de ataque cibernético es una de las novedades que introduce la industria 4.0, pero afortunadamente también puede paliarlo. Los artículos [11] y [12] utilizan las CNN para evaluar el estado del tráfico de datos en dichos sistemas, sabiendo que existen correlaciones fuertes entre estos y los KPIs de las plantas, para detectar cuándo se está dando una situación extraña o se está produciendo un ciberataque

Finalmente, destacaremos que las CNN también pueden ser parte del propio sistema de control industrial. Esta aplicación ha despertado interés especialmente en el sector petroquímico, dando lugar a proyectos como el recogido en [15]. El elemento novedoso de este proyecto es la inclusión de *cross-features*, características que surgen a partir de la correlación entro otras características detectadas por la CNN y que dan lugar a una mayor capacidad de generalización.

3.3 BRAZOS ROBÓTICOS AUTÓNOMOS

Este proyecto, por supuesto, no sería el único de su especie. Numerosas universidades e instituciones han desarrollado proyectos similares, sea por iniciativa propia o con el fin de ganar alguna competición, como puede ser el Amazon Robotics Challenge [16].

En [17], se desarrolla un sistema basado en redes neuronales convolucionales para labores de pick-and-place en entornos atestados. Este sistema está diseñado para poder agarrar objetos tanto conocidos como nuevos, determinando qué herramienta (succionador o pinza) utilizar en cada caso y de dónde agarrarlos.

[18] fue la solución ganadora del Amazon Robotics Challenge 2018. Empleó un manipulador cartesiano multi-herramienta. De nuevo, se centraron en poder identificar y agarrar objetos novedosos para el sistema, empleando redes convolucionales basadas en RefineNet a tal efecto.

El sistema propuesto en [19] utiliza una técnica conocida como "retroproyección de histogramas" (*histogram backprojection*) consistente en asignar a cada punto de la nube de puntos captada por la cámara RGB-D un vector de 6 características, tales como color, borde/no borde, altura respecto al estante, etc. Tras construir esta nube de vectores, emplea datos conocidos acerca de los objetos que está buscando para segmentar dichas nubes y asignar a cada segmento una probabilidad de tratarse del objeto en cuestión. Cabe destacar que emplearon algunas de las técnicas que ya se han utilizado en los proyectos previos a éste [6] , tales como utilizar el código de color HSV para poder lidiar con las diversas condiciones de iluminación de la sala.

Apoyado en [19], el proyecto de [20] destaca por dos motivos fundamentales:

1. Su algoritmo de detección de objetos basado en *deep learning* y el método SVM
2. El generador de movimientos paramétrico, que permite prescindir de la etapa de planificación de movimiento del robot.

Si bien el elemento característico de la mayoría de los sistemas que se han descrito hasta ahora es su método de reconocimiento de objetos, la planificación de la trayectoria del robot no es menos importante, como refleja el último sistema mencionado. Por ello, se expondrán también algunos sistemas más centrados en dicho aspecto.

El método sugerido en [21] plantea el problema de planificación de los movimientos del robot como un problema de satisfacción de límites (*Constraint Satisfaction Problem, CSP*), para los cuáles ya existen técnicas de resolución. Parte del ingenio de este método consiste en formular las condiciones del problema en dos tipos de términos: simbólicos (se trata de una operación de recogida, movimiento, colocación…) y geométricos (mantenerse en el área de trabajo, evitar colisiones, etc.). A partir de ahí, resuelve el problema de *CSP* por métodos existentes.

En base al método descrito en [21], la tesis de [22] propone combinar los planes generados por la resolución de los *CSP* con técnicas de *deep learning* y de aprendizaje por refuerzo,

para lograr adaptarse a entornos con muchos obstáculos y evitar desplazar otros objetos mientras se manipula el objeto de interés.

El trabajo realizado en [23] busca determinar cuál es la posición óptima para colocar los objetos de interés. Este método está pensado para objetos de superficies irregulares y difíciles de equilibrar. Si bien es un método potente, requiere mucha información, notablemente modelos 3D de tanto los objetos a manipular como del entorno, por lo que no es viable para aplicaciones en las que no se tiene tiempo o recursos para conseguir estos datos, como sería el caso de los almacenes.

La publicación de [24] plantea un sistema para evitar colisiones durante la manipulación de objetos en brazos robóticos dotados de visión artificial. Este sistema define matrices de puntos para cada objeto detectado por la cámara y para el propio robot. Utilizando esas matrices y la cinemática del robot, define un algoritmo que permite al sistema calcular trayectorias libres de colisiones

En definitiva, el estado del arte actual parece apuntar en la misma dirección que nuestro proyecto. El uso de redes neuronales convolucionales parece estar bastante extendido en lo que a reconocimiento de objetos para pick-and-place se refiere, empleándose numerosos planteamientos distintos. Del punto de vista de la planificación de movimientos del robot, hay menos bibliografía, pero existen investigaciones y artículos sobre los que se podría basar un sistema para automatizar la generación de movimientos del brazo robótico del proyecto.

3.4 PSO Y EL PROBLEMA DEL VENDEDOR AMBULANTE

El problema del vendedor ambulante (*Travelling Salesman Problem* o TSP) es un problema clásico de optimización que consiste en determinar cuál es la ruta óptima para pasar por un conjunto de puntos, pasando por todos ellos una sola vez y recorriendo la menor distancia posible. La Figura 8 [25] muestra un ejemplo de este problema, mostrando una solución

subóptima y la solución óptima. Existen numerosas técnicas para resolver este problema, como por ejemplo el método Simplex, pero PSO es una de las más llamativas.

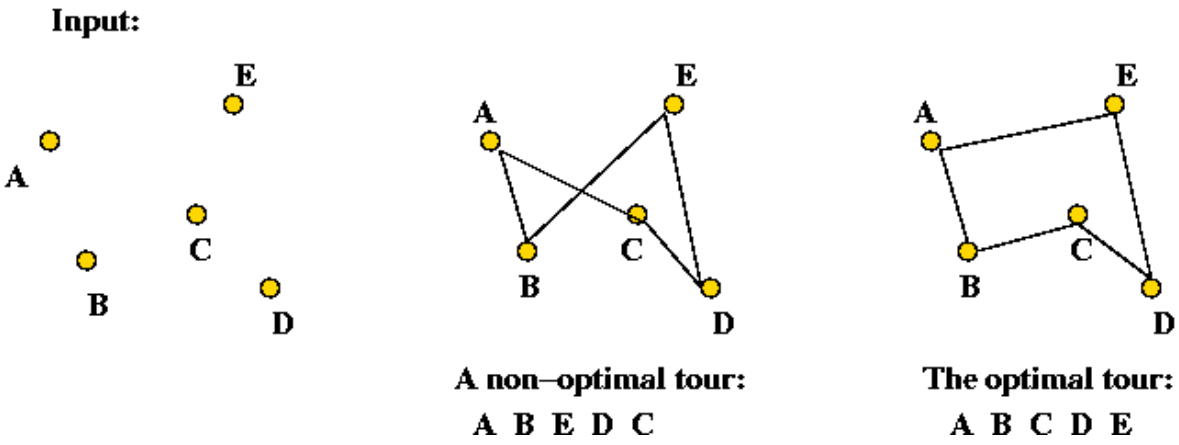


Figura 8. Ejemplo del problema del vendedor ambulante

Como se describió en el Capítulo 2. , PSO es un método diseñado para la resolución de problemas en los que las soluciones pertenecen a un espacio continuo. No obstante, en el problema del vendedor ambulante, las soluciones son listas en las que se detalla el orden en el que se deben visitar los puntos. Por lo tanto, son permutaciones del conjunto de todos los puntos y como tales, no pertenecen a un espacio continuo. Esta peculiaridad hace que haya que adaptar el algoritmo. Por este motivo, existe una amplia literatura respecto a las distintas adaptaciones que se pueden hacer a PSO para resolver este problema.

En [26] se introduce una variante de PSO llamada *Bird Swarm Algorithm* (BSA) para resolver el problema de TSP. Para poder discretizar el espacio en el que se mueven las partículas, introducen el concepto de matriz de entropía de información, una matriz que contiene la posibilidad de que la partícula elija ir desde la ciudad *i* a la ciudad *j*. Además, las partículas se representan con matrices booleanas, conteniendo el valor 1 en las posiciones que representan la ruta que se toma, así la ruta [2 1 3] se representaría de la forma $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ (2 a 1, 1 a 3, 3 a 2). Al operar con estas matrices, es fácil determinar la diferencia entre

partículas y aplicar PSO. Además, al multiplicarlas por la matriz de entropía, se introduce un movimiento aleatorio al sistema que facilita la búsqueda de la solución óptima.

El artículo de [27] introduce el uso del operador de reemplazo como sustituto a las sumas que se utilizan en un PSO tradicional con partículas pertenecientes a un espacio continuo. Además elimina el hiperparámetro inercial y sustituye los hiperparámetros de los componentes social y cognitivo por valores aleatorios entre 0 y 1 que cambian con cada iteración. La formulación de las partículas es como un vector que contiene los elementos de la ruta ordenados, de manera similar a como se hace en este proyecto.

[28] presenta otra variante de TSP, conocida como *Bat Algorithm* (BA). En este algoritmo, las partículas, llamadas murciélagos, tienen una propiedad adicional a su posición y su velocidad: la frecuencia. La frecuencia actúa como un modulador de la velocidad de la partícula. En este algoritmo además, no se produce distinción entre componente social y cognitivo. Las partículas también son discretas y suponen un vector que contiene los puntos de la ruta en orden. Se definen nuevos operadores relacionados con las permutaciones de las partículas

La investigación de [29] propone un método que combina PSO con lógica difusa (*Fuzzy logic*). Este método también incorpora algo similar a la entropía introducida en [26], si bien en este caso afecta también a la posición de la partícula, además de a su velocidad. Esta versión de PSO se utiliza en combinación con otras técnicas como *Simulated Annealing* (SA) y *Neighboring Information Communication* (NIC) para formar un algoritmo híbrido bastante eficaz.

Capítulo 4. ARQUITECTURA DEL SISTEMA

4.1 HARDWARE

A nivel de hardware, el sistema consta de la cámara Intel D-435 (ver figura), conectada por USB al PC que a su vez está conectado por TCP/IP a la red del laboratorio y por ende al robot ABB IRB-120 (ver figura).

Figura 9. Cámara Intel D-435 (izquierda) y robot ABB IRB-120 (derecha)

4.2 SOFTWARE

A nivel de software, cabe distinguir dos bloques principales: En primer lugar, el código RAPID del robot, encargado de ejecutar las órdenes de movimiento y de enviar las solicitudes de imágenes. En segundo lugar, una app de MATLAB se encarga de la inicialización del sistema, la captura de imágenes, la detección de las piezas, la determinación del orden óptimo de recogida, el envío de órdenes al robot y la impresión del

estado actual del robot por consola. La Figura 10 muestra un esquema resumido de la arquitectura HW/SW del sistema.

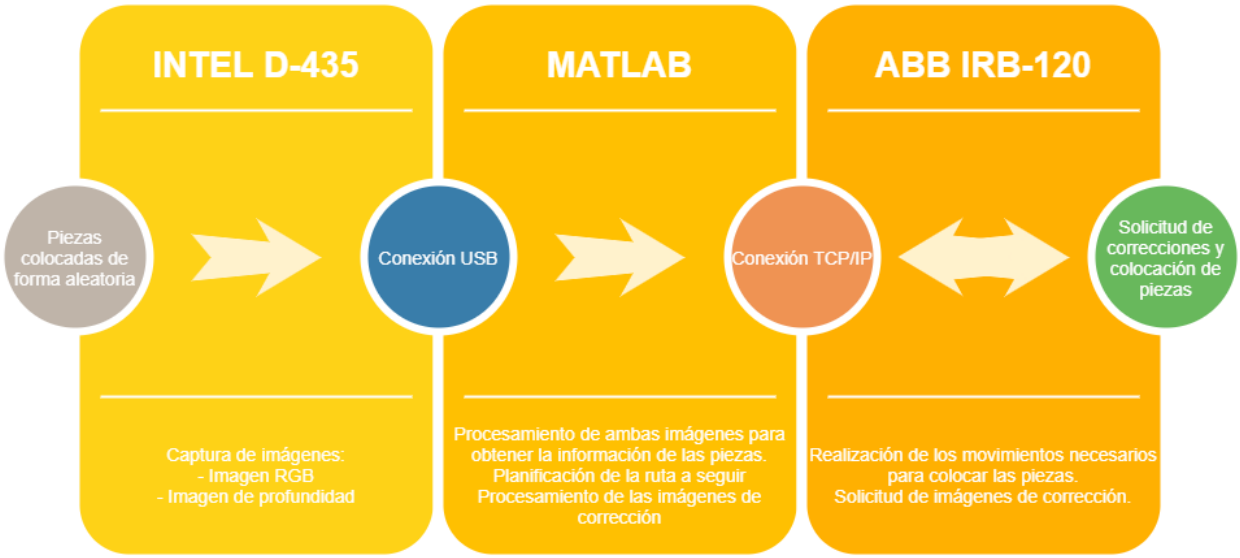


Figura 10. Arquitectura HW/SW del sistema

Entrando más en el detalle de la arquitectura de software, la app de MATLAB depende al mismo tiempo de varios submódulos, cuya jerarquía se resume en la Figura 11. Algunos de los módulos más notables son:

- Paquete *realsense*: Contiene los *drivers* y el resto de funciones y archivos necesarios para interactuar con la cámara a través de MATLAB.
- *app_picture.m*: Detecta las piezas de LEGO contenidas en la imagen, devolviendo sus coordenadas en el sistema de referencia de la imagen, su ángulo y la altura de la pila de piezas (si procede).
- *fdepth.m*: Procesa la imagen infrarroja para detectar la altura de las pilas de piezas
- *orientation.m*: Detecta el ángulo de las piezas mediante la transformada de Hough.
- *R_orientation.m*: Detecta el ángulo de las piezas mediante CNNs.
- *Pic2Robot.m*: Traduce las coordenadas de las piezas en el sistema de referencia de la imagen inicial a coordenadas aproximadas en el sistema de referencia del robot. También elabora las órdenes que se enviarán al robot por TCP/IP.

- *PicCorrection.m*: Corrige la posición de las piezas detectadas en la imagen inicial e introduce la corrección al movimiento necesaria para agarrar piezas giradas debido al desplazamiento del TCP. Especialmente útil para corregir la distorsión radial de la cámara en el borde de la imagen.
- *pso.m*: Determina el orden óptimo para recoger las piezas detectadas previamente recorriendo la menor distancia posible.
- *permute_prod.m* y *permute_distance.m*: Contienen las operaciones necesarias para determinar los conceptos de "distancia" y "velocidad" en el espacio de soluciones discreto del problema del vendedor ambulante. Los detalles de estas operaciones se explicarán más adelante en el punto 265.2.2.
- *distance_matrix.m*: Construye la matriz de distancia entre las piezas y los puntos de entrega necesaria para la resolución del problema del vendedor ambulante.

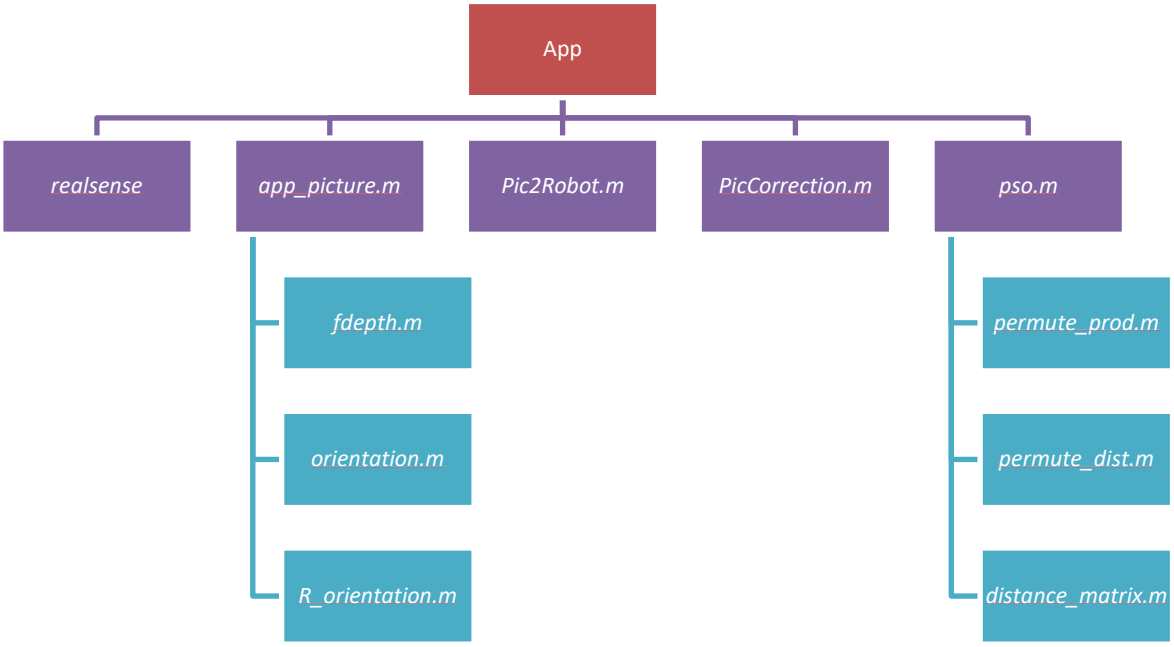


Figura 11. Arquitectura de software MATLAB detallada

Capítulo 5. DISEÑO Y DESARROLLO DEL SISTEMA FINAL

5.1 INTEGRACIÓN DE LOS SISTEMAS ANTERIORES

En [3] se comenzó el desarrollo de la aplicación de MATLAB responsable de integrar los métodos de detección de piezas desarrollados y de mandar las órdenes pertinentes al robot. Una de las tareas iniciales de este proyecto fue completar el desarrollo de la misma, en particular la parte referente a la comunicación con el robot, que era la que menos completada estaba. Completada esta primera versión de la app, resultó evidente que simplemente juntar los sistemas desarrollados hasta el momento no produciría una solución funcional.

El módulo *Pic2Robot.m* descrito en el Capítulo 4. fue elaborado en [6] y solamente estaba pensado para ser utilizado con la detección por máscaras de color elaborada en el mismo proyecto. Una de las particularidades de este proyecto es que se discretizaban las posiciones posibles de las piezas, asumiendo que se encontraban encajadas en el tapete. Esta discretización ocurría siempre durante la elaboración de los mensajes a enviarse al robot, por lo que se descubrió durante las pruebas que, si las piezas no se encontraban encajadas en el tapete, el robot tendría problemas para agarrarlas. Para resolver este problema se optó por modificar *Pic2Robot.m* para que no discretizara la posición de las piezas de ninguna manera, siendo así compatible con la detección por máscaras de color de [6] y por CNN de [3]. Para ello, resultó necesario establecer una correlación entre las coordenadas de las piezas en el sistema de referencia de la imagen y en el sistema de referencia del robot.

En una primera instancia, se decidió establecer una simple correlación lineal entre dichos sistemas de referencia, a través de la medición de la posición de distintas piezas en ambos sistemas. Este método tenía una gran limitación. Funcionaba satisfactoriamente cuando las piezas se encontraban cerca del centro de la imagen, pero si se disponían en los extremos, el

robot fallaba al agarrarlas. Esta limitación procede de la distorsión radial propia de la cámara, como se ilustra en la Figura 12. Como se puede apreciar en la imagen, emplear un modelo lineal para establecer la correlación entre estos dos sistemas en toda la región de trabajo no es la opción adecuada.

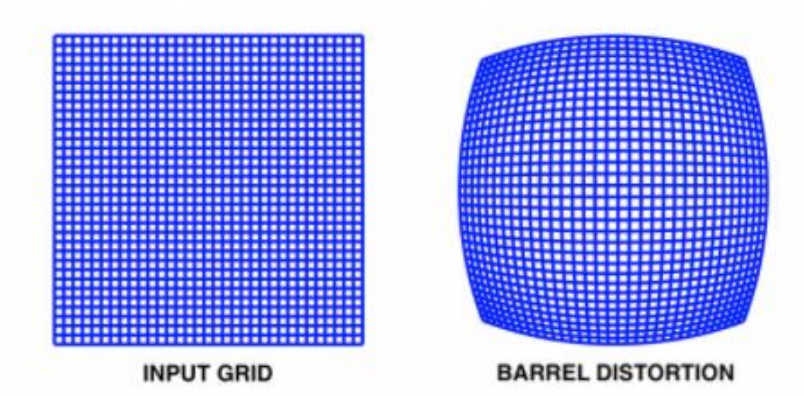


Figura 12. Ilustración de la distorsión radial.

Para solventar este problema se decidió implementar comunicación bidireccional entre el robot y el PC, a fin de que pudiera solicitar la corrección de las coordenadas de la pieza en los casos problemáticos. Hasta el momento, el PC mandaba las órdenes al robot de forma unilateral, y el robot no se comunicaba de vuelta. El funcionamiento de esta comunicación y de la etapa de corrección se explica en el punto 5.2.3.

Otro de los problemas derivados de la integración de los dos sistemas resultó ser el manejo de las piezas rotadas. Al sustituir el método para correlacionar los sistemas de coordenadas empleado en [6], el robot se desviaba al intentar agarrar este tipo de piezas. Esto se debe a que el sistema enviaba las coordenadas del centro de la *bounding box* de la pieza detectada, las cuales coincidían con la posición a la que se tenía que desplazar el robot para agarrarla. Al introducir el giro en el movimiento del robot para tomar piezas rotadas, el TCP se desplaza, como se ve en la Figura 13, causando que el robot falle si no se introduce algún tipo de corrección. La pieza de color sólido representa la posición real de la pieza, girada un ángulo θ, mientras que la pieza a rayas representa la posición a la que el robot acaba desplazándose al no introducir corrección. Dado que se conoce la distancia entre la muñeca del robot y el TCP, llamada *r*, este error se puede corregir, como en efecto se hizo en la etapa de corrección.

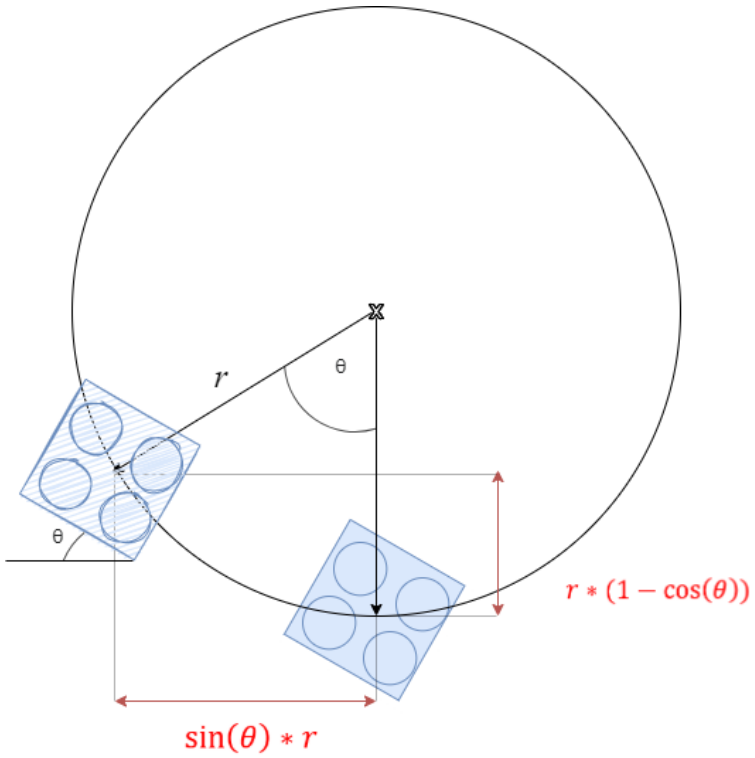


Figura 13. Ilustración del error debido a la rotación

Una de las limitaciones que se encontró en la detección de ángulos desarrollada anteriormente es su incapacidad para distinguir entre ángulos positivos y negativos. Mirando de nuevo la Figura 13, se puede apreciar que, si la pieza estuviera rotada en sentido antihorario, dando lugar a un ángulo negativo, la corrección del movimiento y el lugar por el que debería agarrar el robot la pieza serían completamente distintos. Esta limitación quedó como desarrollo futuro, y sólo se han realizado ensayos con piezas rotadas en sentido horario.

En [6] se desarrolló una maniobra de extracción de piezas muy encajadas para el robot. No obstante, no se encontraba integrada en la programación parcial de la aplicación que se había iniciado en [3]. Por ello, se modificó *Pic2Robot.m* para que mandara la orden de emplear la maniobra de extracción en el caso de que las piezas estuvieran apiladas unas encima de otras.

El último problema a salvar para lograr tener un sistema con un funcionamiento básico fue el envío de mensajes al robot. Tal y como estaba planteado inicialmente, el sistema tenía

problemas a la hora de manejar varias piezas. La primera pieza del conjunto se agarraba y depositaba satisfactoriamente, pero el comportamiento del robot para piezas sucesivas era muy extraño, dirigiéndose a lugares en los que no había nada y girando cuando no debía. Este comportamiento apuntaba a un problema de colisión de mensajes en el socket TCP/IP. Por lo tanto, se decidió implementar una limpieza del socket cada vez que se enviaba un mensaje para evitar este problema. Con todo ello, la implementación de la comunicación bidireccional sigue siendo rudimentaria y el cuello de botella de velocidad del sistema, ya que no hay un hilo dedicado a la comunicación.

5.2 FUNCIONAMIENTO DEL SISTEMA FINAL

5.2.1 PROCESAMIENTO DE LA IMAGEN INICIAL

Una vez iniciada la app y el programa del robot, el robot se colocará automáticamente en la posición inicial, y la app mostrará en la pantalla la imagen que está viendo la cámara en ese momento, junto con la información (posición, ángulo, color, altura) de las piezas detectadas. como se ve en la Figura 14.

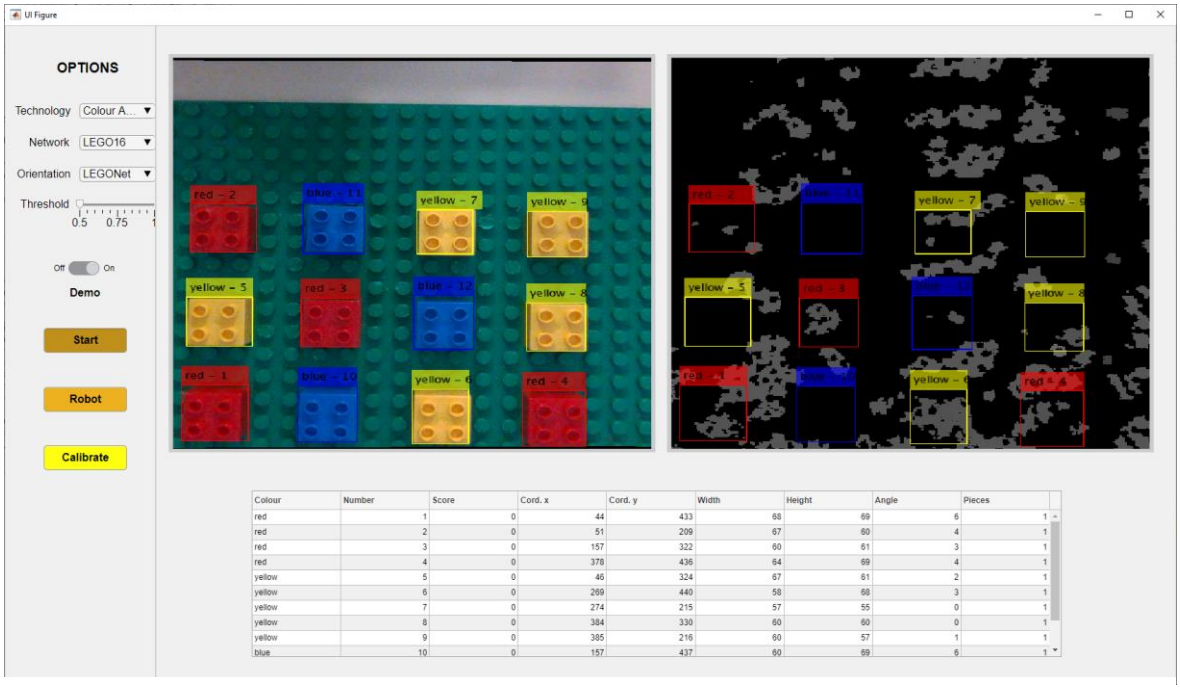


Figura 14. Ejemplo de la interfaz de la app

Con la app también se puede elegir cómo se realizará la detección de la pieza y de su ángulo. A lo largo de varias pruebas se ha comprobado que utilizar una red tipo YOLO basada en LEGO16 y la transformada de Hough para la detección de ángulos proporciona los resultados más precisos. Al emplear otro tipo de redes, como las basadas en LEGONet por ejemplo, las *bounding boxes* aparecían en ocasiones descentradas respecto a la pieza, como se observa en la Figura 15.

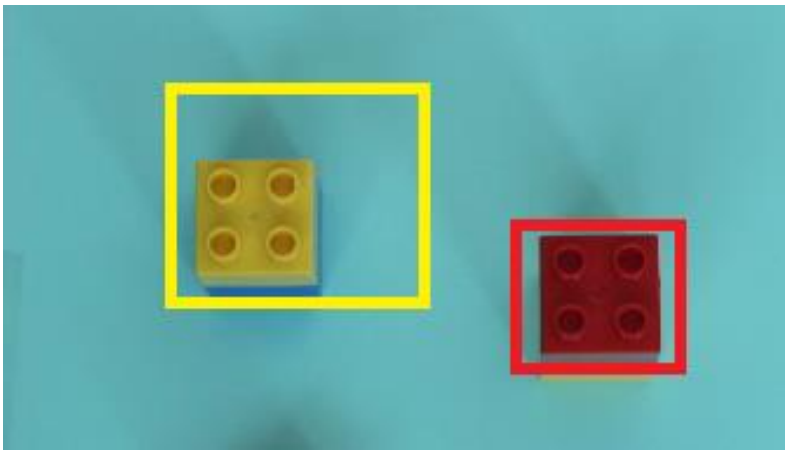


Figura 15. Bounding boxes descentradas

Se intentó emplear *template matching* para recentrar las *bounding boxes*, pero esta técnica añadía un componente adicional de sensibilidad a las condiciones de luz, ya que deben ser similares a las condiciones en las que se tomaron las imágenes de referencia, por lo que finalmente se descartó.

Al presionar el botón “Start” se guarda la información de las piezas detectadas y se procede a determinar la ruta a seguir para recogerlas

5.2.2 PLANIFICACIÓN DE LA RUTA

En primer lugar, se consideran sólo las piezas que se encuentren en las pilas de mayor altura. Esto se hace para evitar posibles colisiones del brazo con pilas de piezas muy altas al intentar extraer una pieza de una pila más pequeña. En la Figura 16 se muestra un caso en el que fácilmente podría darse colisión si no se trabajara por alturas.

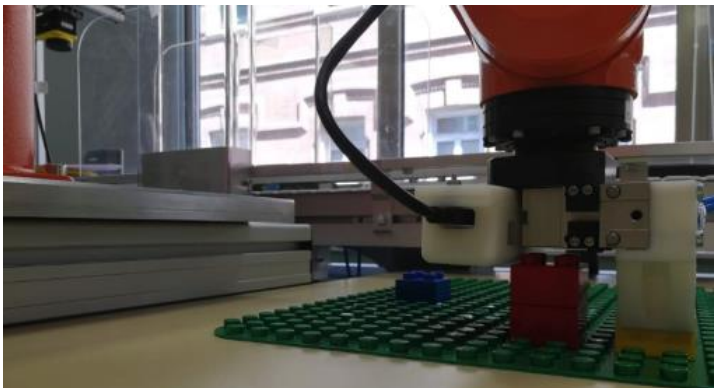


Figura 16. Situación problemática en la que trabajar por alturas es favorable

Con las posiciones de estas piezas, se ejecuta PSO. PSO fue elegido por dos motivos: En primer lugar, es un algoritmo que destaca por la velocidad con la que converge, como se explica en [30]; En segundo lugar, existe un gran interés en la comunidad científica por métodos para discretizar este algoritmo, como se evidencia en el Capítulo 3. y la implementación realizada en este proyecto puede encontrar usos en otros contextos.

Como se introdujo en el Capítulo 2. , se han realizado una serie de adaptaciones a PSO para poder utilizarlo en este caso. Nuestras partículas o posibles soluciones serán listas que denotan los posibles órdenes en que se pueden recoger las piezas. Si tuviéramos 5 piezas, llamadas del 1 al 5, las partículas podrían tomar valores como [1 2 3 4 5] o [1 3 4 5 2]. Estas partículas serán evaluadas por una función de coste que toma en cuenta la longitud del trayecto total que ha de hacer el robot. Este trayecto incluye el desplazamiento del punto inicial a la pieza, el desplazamiento desde la posición de la pieza al correspondiente punto de entrega según su color y el desplazamiento desde dicho punto de entrega a la pieza siguiente. Dada una partícula P_i con un número de elementos N, en la que $x_{Pi(k)}$ representa la

coordenada X de la pieza que ocupa la posición *k* en la partícula y de igual modo con $y_{Pi(k)}$ y la coordenada Y de la pieza correspondiente. Denotando las coordenadas del punto de entrega correspondiente a la pieza según su color como $x_{E(k)}$ e $y_{E(k)}$, así como las coordenadas del punto inicial como x_0 e y_0. La evaluación de la función de coste sería de la siguiente forma:

$$F(P_i) = \sum_{k=0}^{N} M(k, k+1) + \sqrt{\left(x_{P_i(1)} - x_0\right)^2 + \left(y_{P_i(1)} - y_0\right)^2}$$

$$M(i,j) = \sqrt{\left(x_{P_i(i)} - x_{E(i)}\right)^2 + \left(y_{P_i(i)} - y_{E(i)}\right)^2} + \sqrt{\left(x_{E(i)} - x_{P_i(j)}\right)^2 + \left(y_{E(i)} - y_{P_i(j)}\right)^2}$$

Una vez evaluada la partícula, nos encontramos con otro problema. En el algoritmo PSO original, para determinar la velocidad de una partícula es necesario conocer la diferencia de posición entre partículas. Como nuestras partículas son permutaciones de una lista de elementos, la clásica distancia euclídea no funciona, por lo que es necesario redefinir el concepto de distancia entre partículas. Para determinar la distancia entre dos partículas, llamadas P_i y P_j se utiliza el siguiente algoritmo:

1. Recorrer el vector P_i hasta encontrar una posición *k* en la que P_i y P_j difieran.
2. Determinar en qué posición *n* de P_i se encuentra el valor que ocupa la posición *k* en P_j.
3. Permutar el elemento *k* de P_i por el elemento *n* de P_i.
4. Almacenar los valores *k* y *n* en una dupla (*k*, *n*).
5. Continuar el algoritmo hasta recorrer todo el vector P_i.
6. Devolver todas las duplas almacenadas.

Por ejemplo, si quisiéramos determinar la distancia entre las partículas [A B C D E] y [E C B D A], la distancia sería [(1, 5), (2, 3)]

Determinada la distancia, existe un problema adicional en la ecuación de la velocidad. El algoritmo PSO original asume distancias y velocidades que pertenecen a un espacio continuo

convencional y, por tanto, se puede modificar sus módulos al multiplicarlos por hiperparámetros, como los mencionados *w*, c_1 y c_2.

En nuestro caso, el equivalente al módulo de las distancias y de las velocidades sería el número de duplas que contienen. Si modificáramos ese módulo, nos quedaríamos con una fracción de las duplas, en orden. Para mantener un criterio constante, consideraremos que al reducir el módulo de estas magnitudes, se eliminarán primero las últimas duplas. Dada una distancia *d* con *N* duplas, al reducir su módulo en un factor *x*, tomaremos las *N*x* primeras duplas, redondeadas hacia abajo. Por ejemplo, dada la distancia *d*=[(1, 5),(2, 3),(4, 1)] con 3 duplas entonces 0.7**d*=[(1, 5),(2, 3)].

Para aumentar la robustez del algoritmo, se ha decidido reemplazar la componente inercial $w * V_i(t)$ de la **Ecuación 1** por una componente de movimiento aleatorio de la partícula: $w * R_i(t)$ donde R_i representa la distancia entre la partícula P_i y un elemento aleatorio perteneciente al espacio de posibles soluciones.

Además, para garantizar la convergencia de la solución, se ha decidido ir actualizando los valores de los hiperparámetros en cada iteración de acuerdo con:

$$w(t+1) = w(t) * .95$$
$$c_1(t+1) = c_1(t) * 1.01$$
$$c_2 = 1 - (w + c_1)$$

De este modo el componente aleatorio cada vez pierde más peso, mientras que los componentes cognitivo (que tiende hacia la mejor marca personal de la partícula) y social (que tiende hacia la mejor marca del conjunto de partículas) van ganando importancia. La suma de estos componentes en nuestro caso se hace encadenando sus duplas.

Finalmente, nuestra implementación de PSO se hace con 10 partículas y 50 iteraciones.

5.2.3 ENVÍO DE ÓRDENES Y CORRECCIÓN DE LA DETECCIÓN

Una vez determinado el orden óptimo en el que recoger las piezas, se pasa la información de las mismas siguiendo el orden establecido a la función *Pic2Robot.m* que traduce las coordenadas de las piezas en el sistema de referencia de la imagen (píxeles) a coordenadas en el sistema de referencia del robot (milímetros) y elabora los mensajes que han de enviarse al robot por TCP/IP para que recoja las piezas. Una vez los mensajes han sido elaborados, el comportamiento del sistema es el siguiente:

1. El robot se desplaza hacia la ubicación de la siguiente pieza a agarrar.
2. El robot se detiene sobre ella y manda un mensaje al ordenador, solicitando la captura de una nueva imagen para corregir la posición detectada en la imagen inicial.
3. Se limpia el socket TCP/IP para evitar colisiones entre mensajes.
4. El ordenador manda un nuevo mensaje con la corrección y limpia el socket de nuevo.
5. El robot aplica la corrección, recoge la pieza y la deja en su punto de entrega correspondiente.

Al tomar una nueva imagen en la etapa de corrección, existe la posibilidad de que se detecten más piezas aparte de aquella que se desea agarrar. Por lo tanto, se corrigen las coordenadas únicamente de aquella pieza que esté más cerca del punto correspondiente al TCP.

La corrección es necesaria ya que, debido a la distorsión radial de la cámara, es cada vez más difícil encontrar una correlación entre los píxeles de la imagen y las coordenadas del sistema de referencia del robot a medida que nos alejamos del centro de la imagen. Por lo tanto, se emplean dos correlaciones entre píxeles y coordenadas del robot en este proyecto.

Una primera estimación, poco precisa, que se utiliza para determinar la posición aproximada de las piezas y una segunda estimación, muy afinada, que se utiliza en la etapa de corrección cuando las piezas están más cerca del centro de la imagen. Ambas son modelos lineales que relacionan coordenadas X e Y en la imagen con coordenadas X e Y en la realidad, con la diferencia de que el primer modelo fue elaborado con datos pertenecientes a la mayor parte de las regiones de la imagen y el segundo modelo con datos ubicados mayormente en torno

al centro de la imagen. Además, el primer modelo intenta determinar la posición absoluta de la pieza en el sistema de coordenadas del robot, mientras que el segundo modelo determina la posición de la pieza relativa a la posición actual del robot.

A la hora de agarrar piezas giradas, el TCP del robot se desplaza, lo cual no es tenido en cuenta por el código del robot. Durante la etapa de corrección también se introduce un término adicional que asegura que el robot pueda agarrar correctamente las piezas giradas, como se mencionó en la sección 5.1. Para garantizar que no haya confusión con otras piezas durante la etapa de corrección, se analiza solamente aquella pieza que esté más cerca del punto correspondiente a la pinza del robot.

Una vez se ha agarrado la pieza actual, se mandan las órdenes para ir a la ubicación aproximada de la siguiente pieza, se ejecuta de nuevo la etapa de corrección y así sucesivamente hasta que se hayan colocado todas las piezas de la altura actual. Tras ello se vuelve a la posición inicial, se toma otra imagen y se repite el proceso desde el principio.

En la Figura 17, se puede apreciar el diagrama de flujo de todo el proceso:

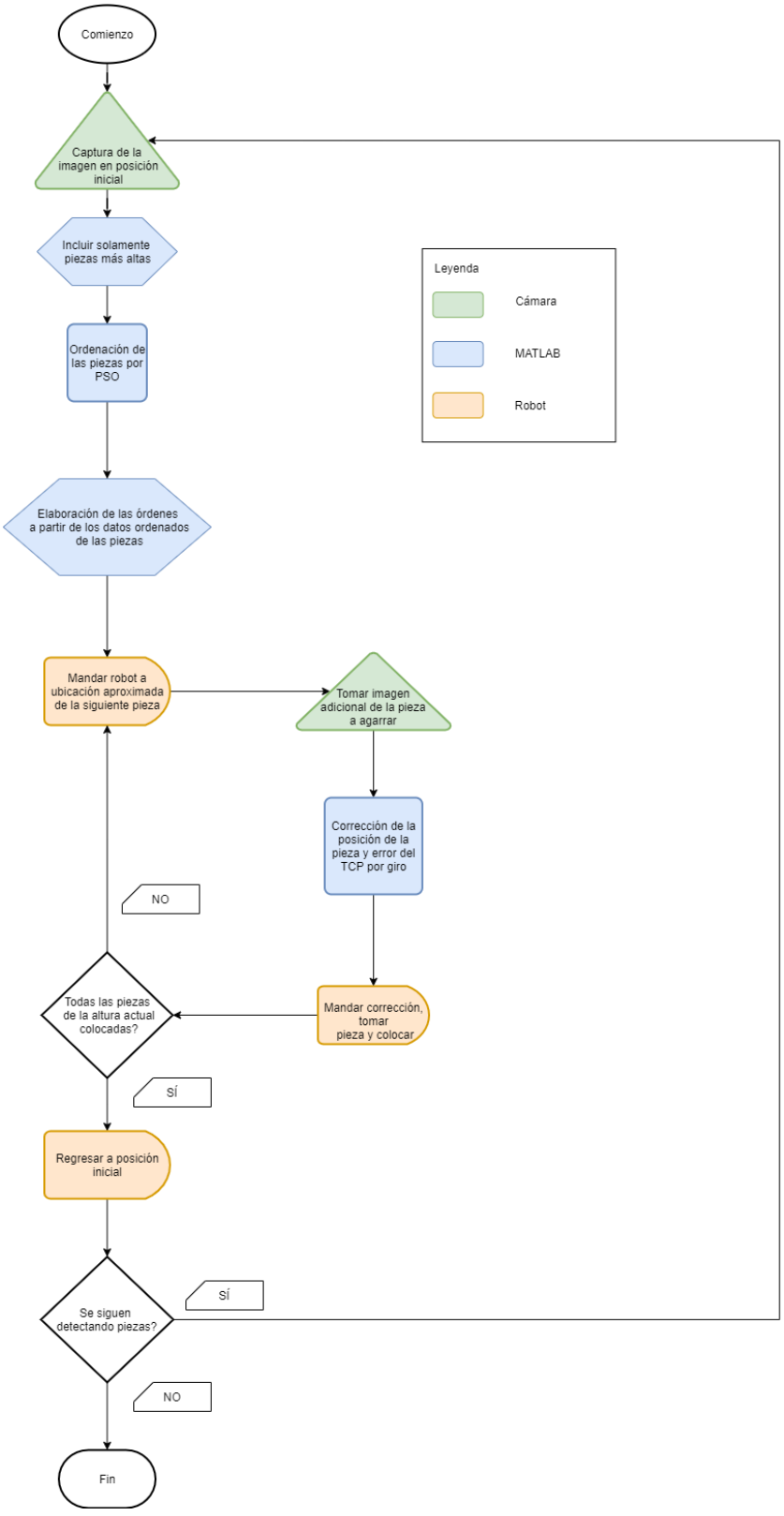


Figura 17. Diagrama de flujo del funcionamiento del sistema

Capítulo 6. ANÁLISIS DE RESULTADOS

En la siguiente sección se presentan vistas de la interfaz de la app de MATLAB y de la ventana de comandos en casos prácticos. También se adjuntarán vídeos que muestren el desempeño del robot en los correspondientes casos.

6.1 CASO 1

Este es un caso sencillo, en el que solamente hay tres piezas a ras de suelo, ninguna está apilada y tan sólo hay dos colores. La situación inicial se muestra en la Figura 18

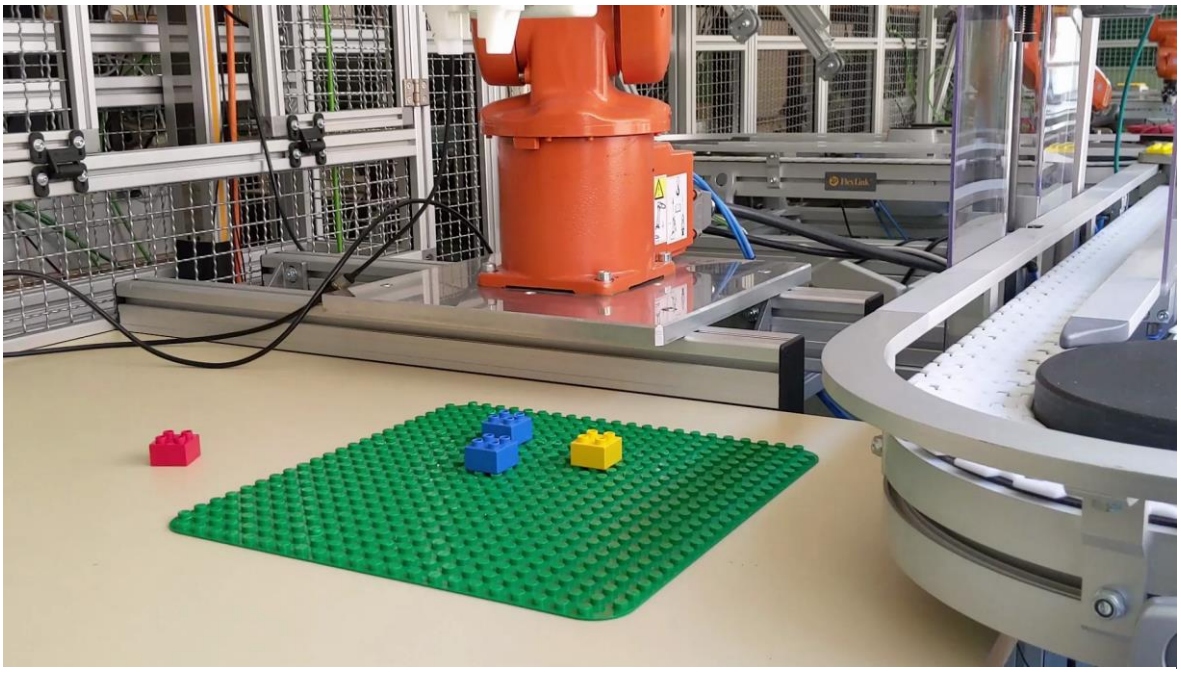


Figura 18. Situación inicial del caso 1

En la Figura 19 se puede ver una captura de la imagen inicial tomada por la cámara en la interfaz. La interfaz muestra las coordenadas de las piezas en el sistema de la imagen, las dimensiones de las *bounding boxes*, la altura de las pilas (Stacks) y el ángulo de las piezas.

En la consola, la app devuelve las coordenadas en el sistema de referencia del robot de las distintas piezas. Cuando aparece la línea “Current tallest pieces coordinates”, la tabla

sucesiva muestra la información de las piezas contenidas en la imagen inicial. Posteriormente, muestra la información de la pieza de interés en la imagen de corrección. El único apartado de la tabla que necesita explicación es "Dropout" que representa si la pieza debe colocarse en la zona de entrega a ras de suelo (Dropout 0) o si ha de apilarse (Dropout 20, 40, 60…). El sistema funciona bien, y en este caso se puede apreciar incluso cómo el robot deposita las piezas en varias alturas según es necesario. La Figura 20 muestra unas capturas ilustrativas del desempeño para este caso.

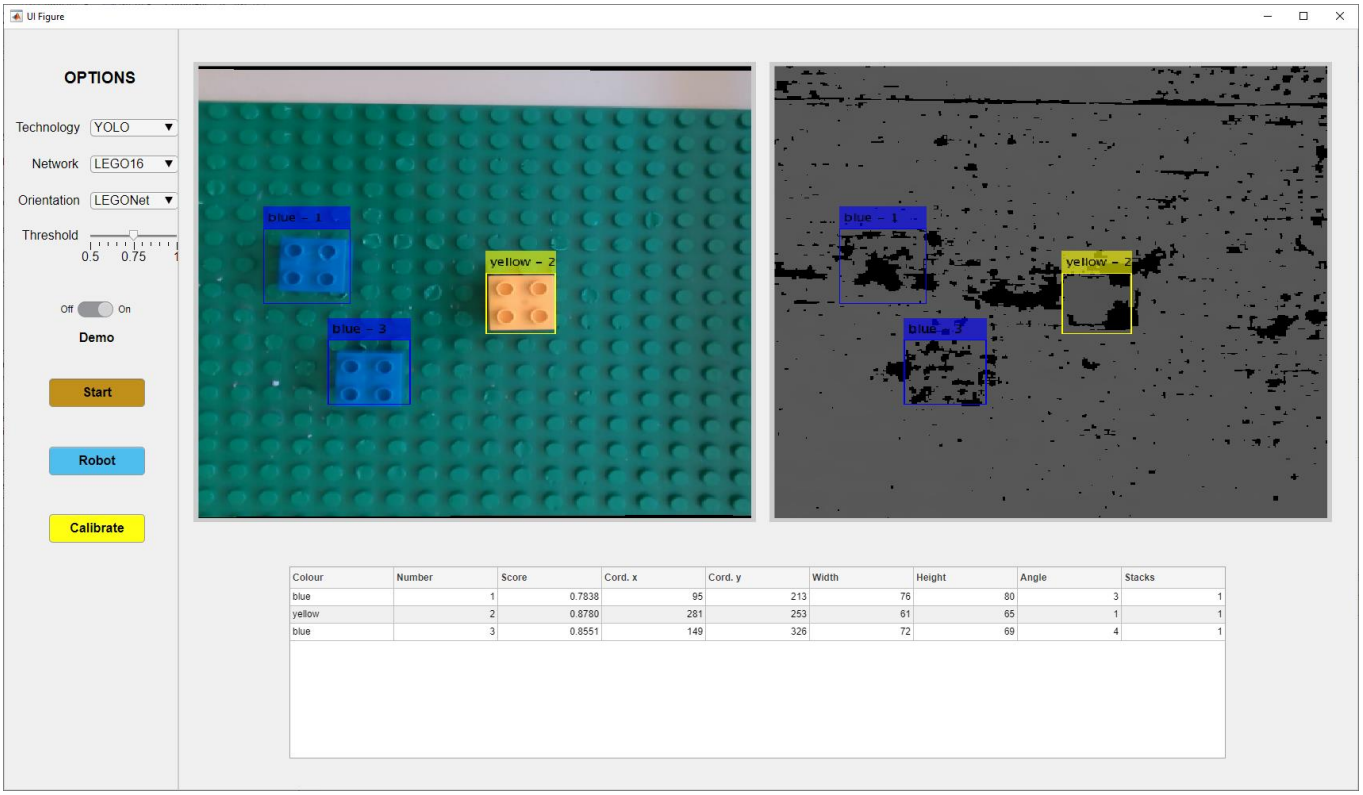


Figura 19. Interfaz durante el caso 1

Análisis de la imagen inicial (piezas a altura 1):

```
Current tallest pieces coordinates
    Colour     Cord.X     Cord.Y     Cord.Z     Angle     Dropout     Last piece
    ______     ______     ______     ______     _____     _______     __________

      3        -36.592    -17.529    -318.98      4          0           0
      2         57.971    -122.58    -318.98      1          0           0
      3          -26.8     77.384    -318.98      3         20           1
```

Corrección aplicada a cada pieza de altura 1:

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	-2.2187	3.2897	-318.98	4	0	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
2	-0.41011	-1.5172	-318.98	1	0	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	-0.16184	7.2192	-318.98	3	20	1

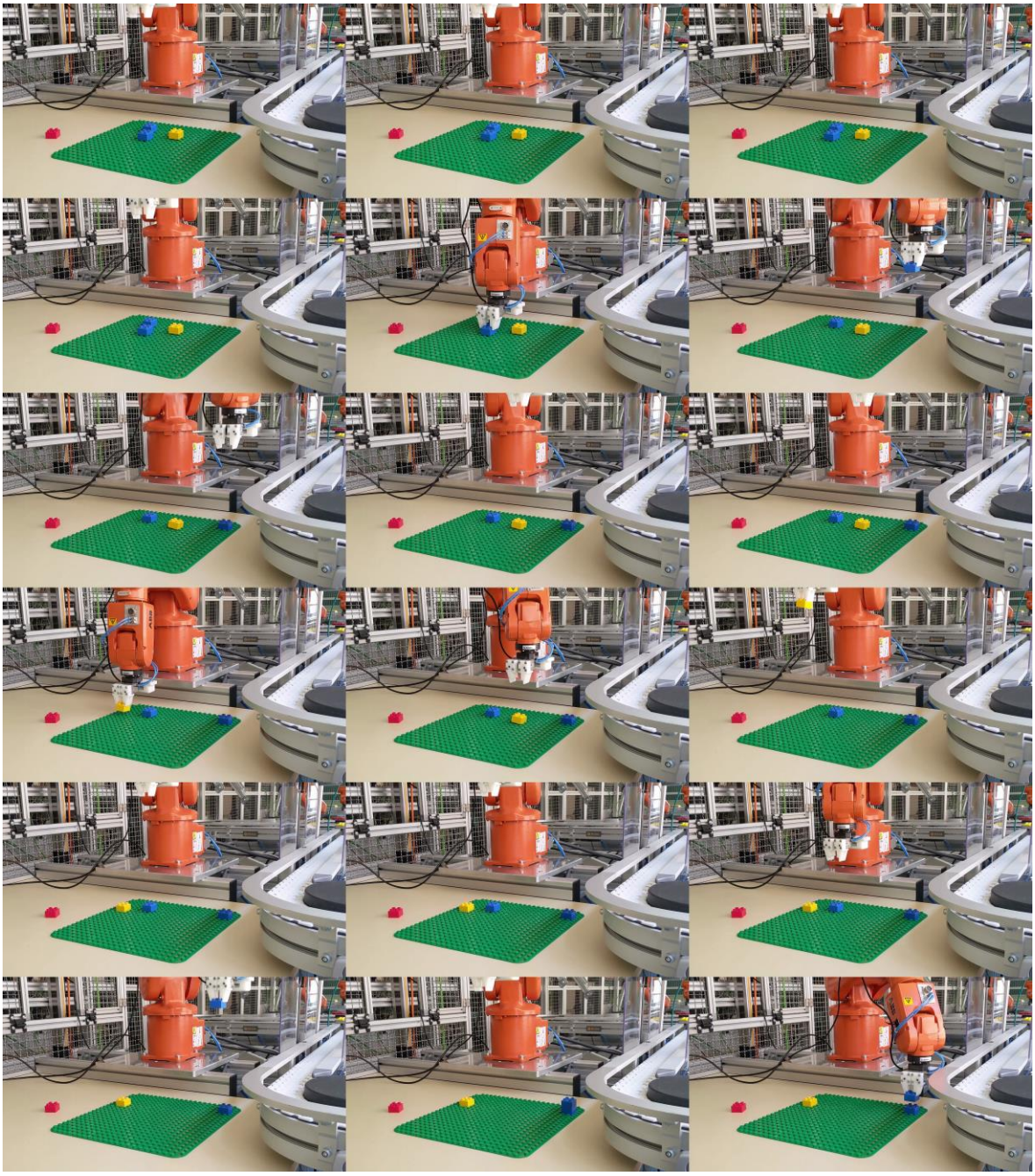
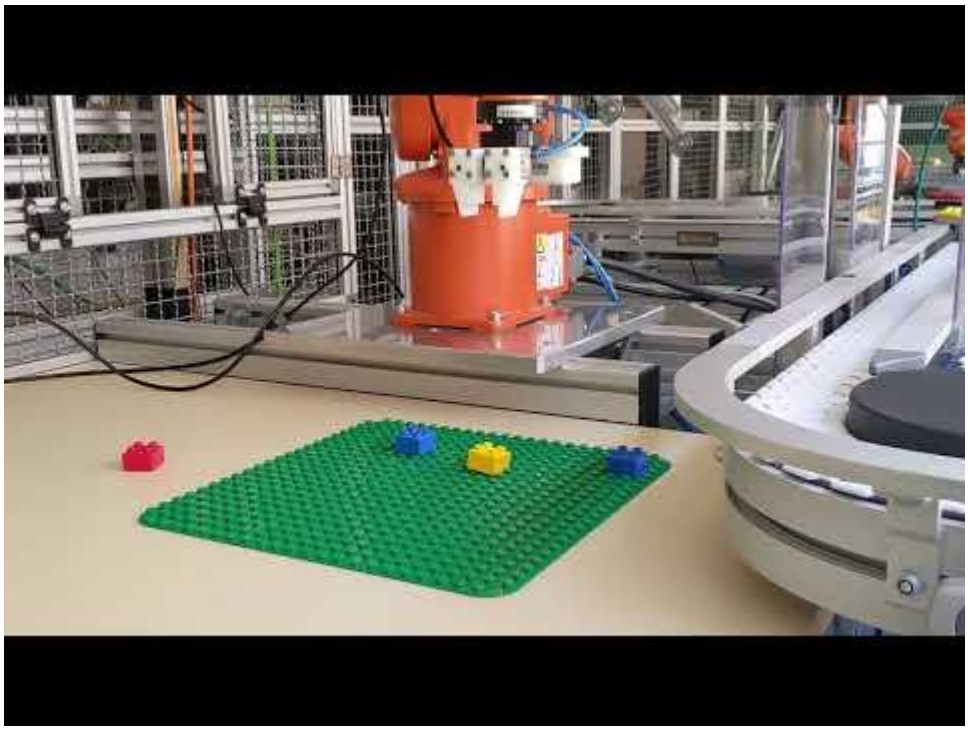


Figura 20. Capturas ilustrativas del caso 1



Vídeo 1. Caso 1

6.2 CASO 2

Este caso introduce algo más de complejidad, al introducir una pila de dos piezas en el tapete, lo cual requerirá emplear la maniobra de extracción apropiada, así como todos los colores disponibles. La Figura 21 resume la situación inicial de este caso.

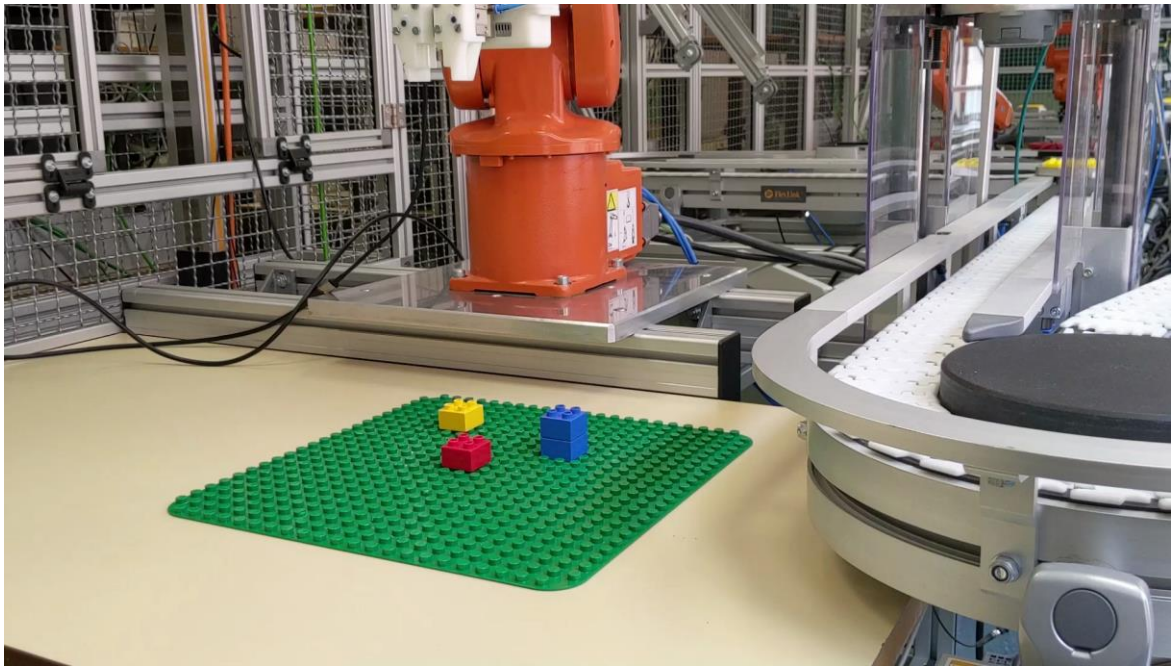


Figura 21. Situación inicial del caso 2

En la Figura 22 se aprecia la interfaz para este caso. El robot trabaja en este caso sin problemas, ejecutando correctamente la extracción de las piezas que se encuentran apiladas y apilándolas correctamente en las zonas de entrega. La Figura 23 muestra unas capturas que resumen el comportamiento del robot.

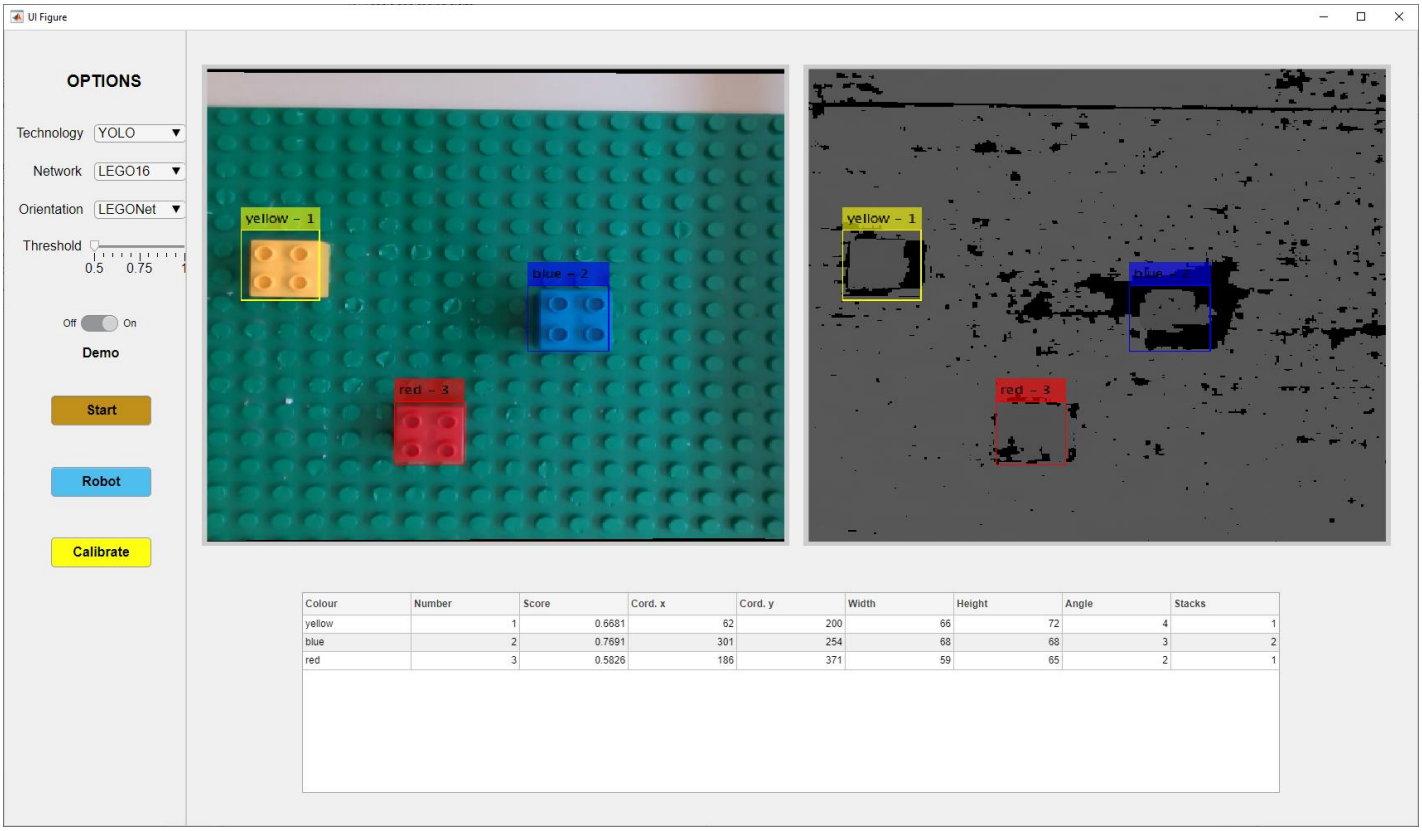


Figura 22. Interfaz durante el caso 2

Análisis de la imagen inicial (piezas a altura 2):

```
Current tallest pieces coordinates
    Colour    Cord.X    Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    ______    ______    ______    _____    _______    __________

      3       -115.16    193.16    -310.98     4         0          1
```

Corrección de las piezas a altura 2:

```
    Colour    Cord.X     Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    _______    ______    ______    _____    _______    __________

      3       -0.23765    5.2893    -310.98     4         0          1
```

Análisis de la imagen inicial (piezas a altura 1):

Current tallest pieces coordinates

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
2	-111.7	-64.938	-318.98	2	0	0
1	66.574	128.57	-318.98	2	0	0
3	-115.16	193.16	-318.98	4	20	1

Corrección de las piezas a altura 2:

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
2	3.8418	0.70206	-318.98	2	0	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
1	-2.046	-0.28791	-318.98	2	0	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	-0.23765	5.2893	-318.98	4	20	1

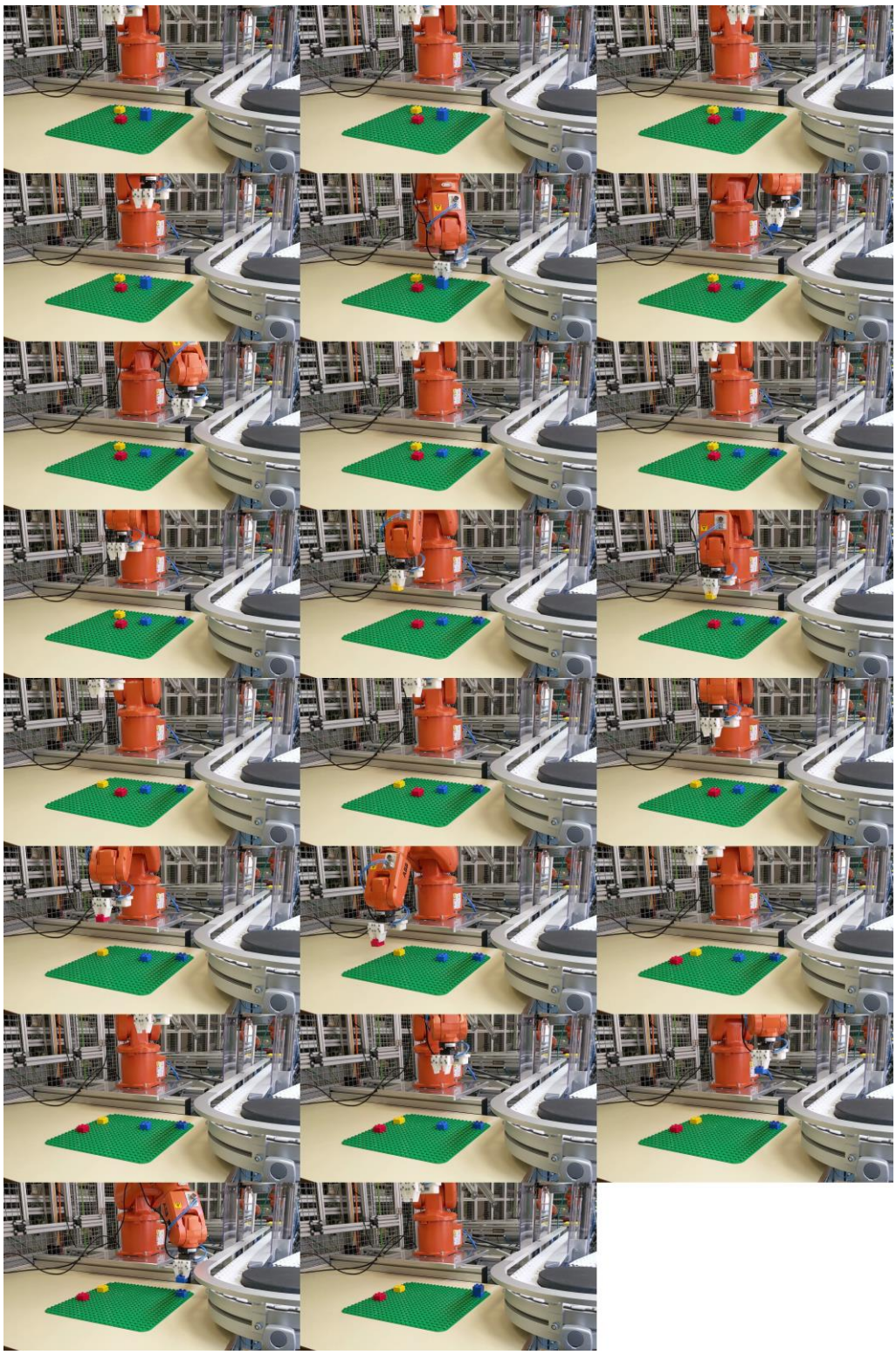
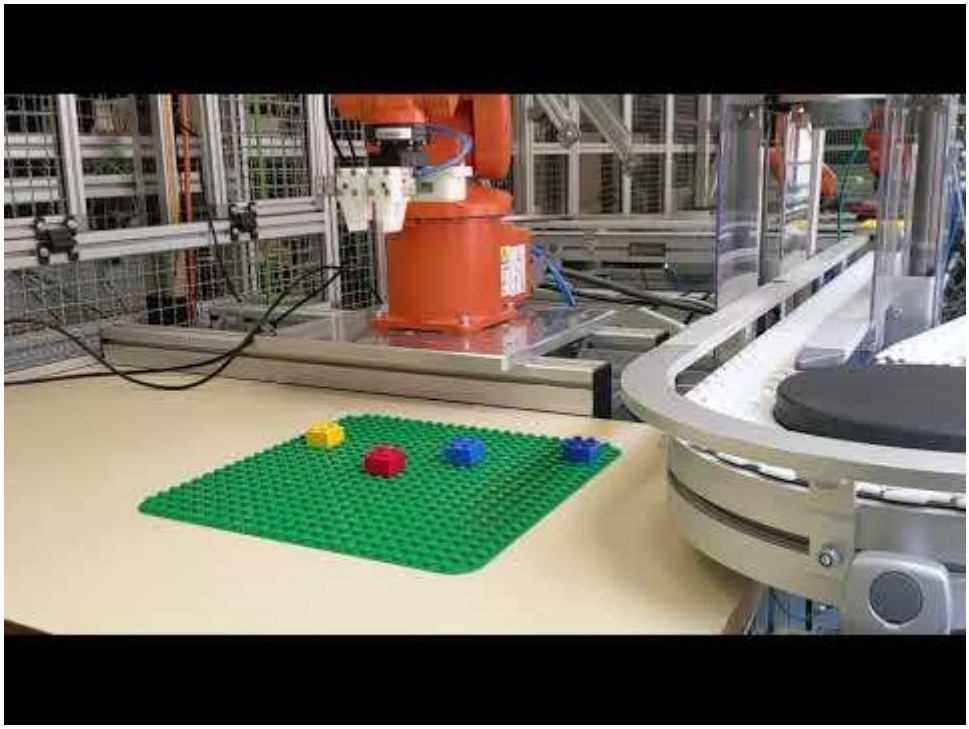


Figura 23. Resumen del comportamiento del robot en el caso 2



Vídeo 2. Caso 2

6.3 CASO 3

Este es el caso más complejo. Aparecen todos los colores, pilas de tres piezas y piezas rotadas, las cuales requerirán emplear la corrección del TCP por rotación. Además, se modifican las condiciones de iluminación, introduciendo luz artificial. La FIGURA muestra la situación inicial para este caso.

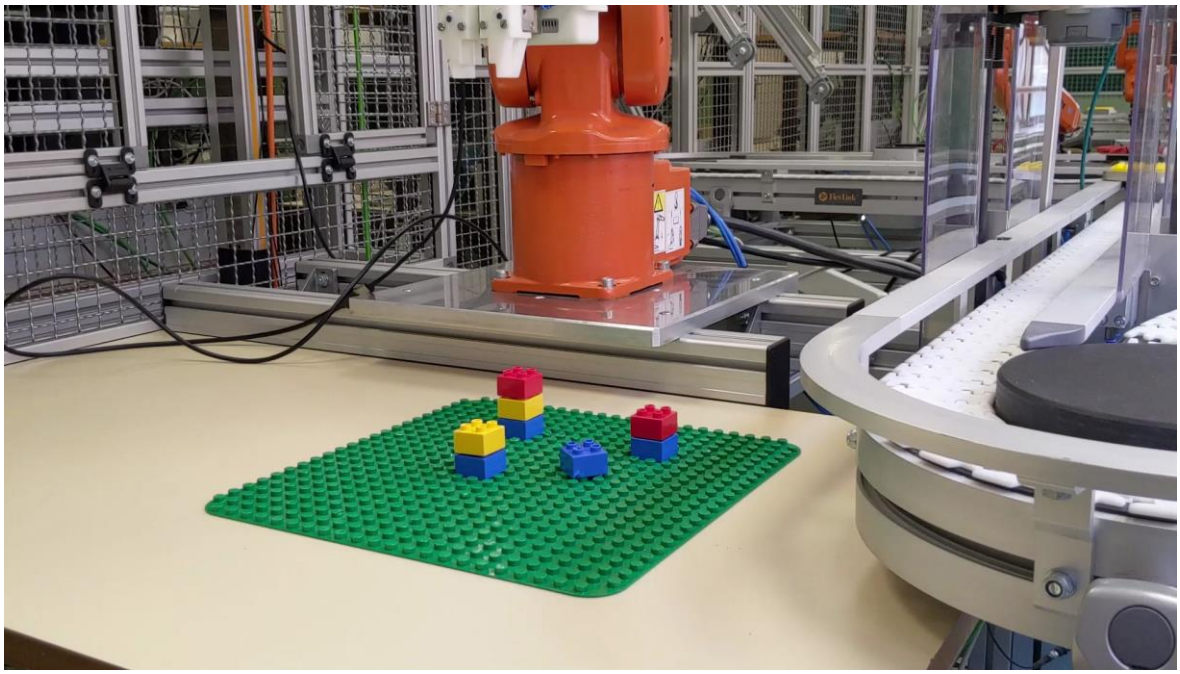


Figura 24. Situación inicial del caso 3

La Figura 25 presenta la interfaz para este caso. El robot desarrolla este caso satisfactoriamente, incluyendo la corrección por giro mencionada a la perfección, además de todo lo indicado en los casos anteriores.

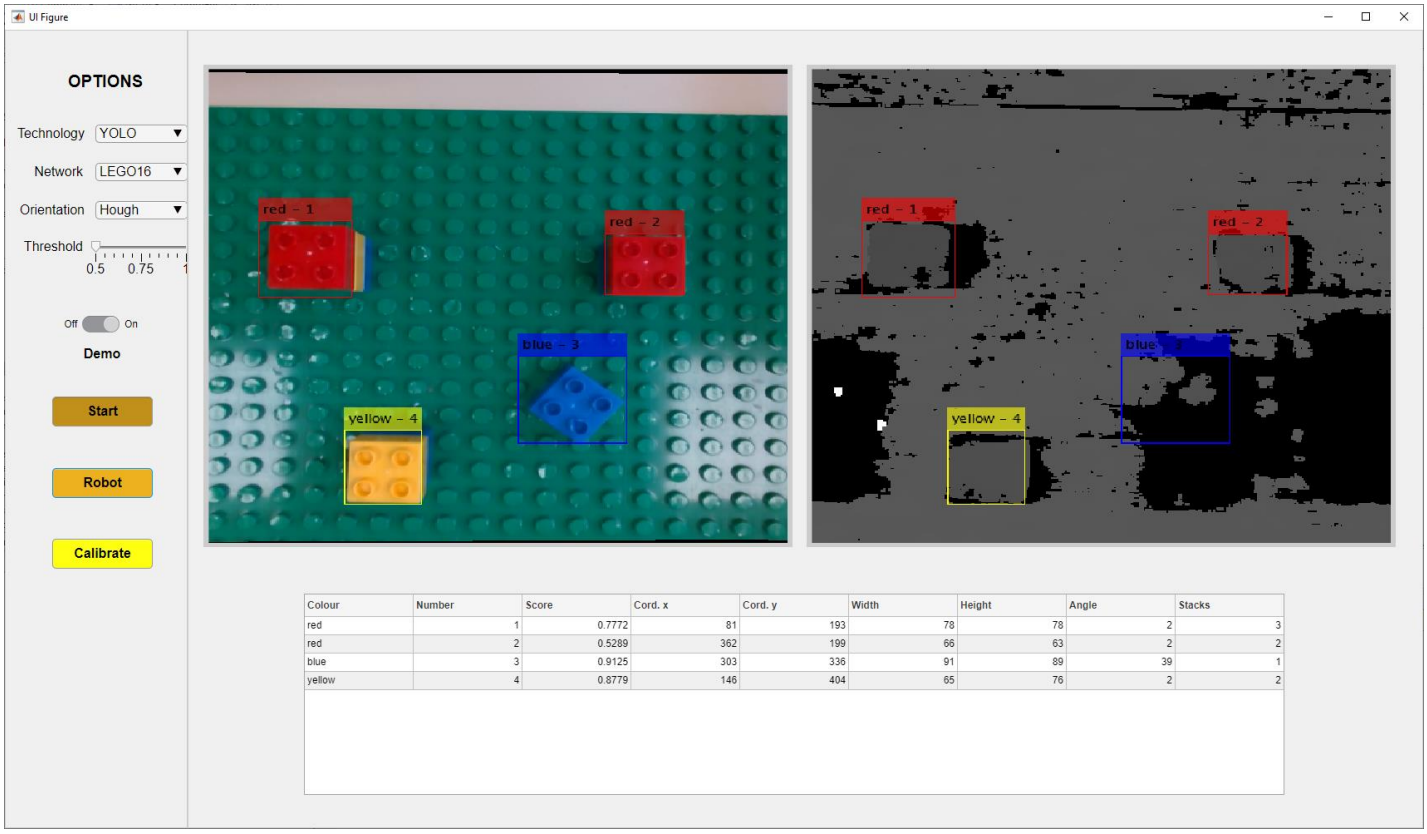


Figura 25. Interfaz para el caso 3

Análisis de la imagen inicial (piezas a altura 3):

```
Current tallest pieces coordinates
    Colour    Cord.X    Cord.Y    Cord.Z     Angle    Dropout    Last piece
    ______    ______    ______    _______    _____    _______    __________

      1       -3.194    -18.59    -281.98      2         0           1
```

Corrección de las piezas a altura 3:

```
    Colour    Cord.X     Cord.Y    Cord.Z     Angle    Dropout    Last piece
    ______    _______    ______    _______    _____    _______    __________

      1       -2.9295    1.4928    -281.98      2         0           1
```

Análisis de la imagen inicial (piezas a altura 2):

```
Current tallest pieces coordinates
    Colour    Cord.X    Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    ______    ______    ______    _____    _______    __________

      2        -114.1    -47.59    -300.98     1         0           0
      2        87.007    108.19    -300.98     2        20           0
      1       -34.869     230.1    -300.98     2        20           1
```

Corrección de las piezas a altura 2:

```
    Colour    Cord.X    Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    ______    ______    ______    _____    _______    __________

      2       9.8055    -1.0781   -300.98     1         0           0

    Colour    Cord.X     Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    ______    ________   ______    _____    _______    __________

      2       -4.0822   -0.72622   -300.98     2        20           0

    Colour    Cord.X    Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    ______    ______    ______    _____    _______    __________

      1       5.6433    -4.9259   -300.98     2        20           1
```

Análisis de la imagen inicial (piezas a altura 1):

```
Current tallest pieces coordinates
    Colour    Cord.X    Cord.Y    Cord.Z    Angle    Dropout    Last piece
    ______    ______    ______    ______    _____    _______    __________

      3       18.184    -12.289   -318.98     2         0           0
      3       124.58    -144.62   -318.98    39        20           0
      3       26.559    -81.704   -318.98     2        40           0
      3       24.462    -226.19   -318.98     1        60           1
```

Corrección de las piezas a altura 1:

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	-3.4955	-0.50682	-318.98	2	0	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	-9.5544	58.997	-318.98	39	20	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	1.3155	2.921	-318.98	2	40	0

Colour	Cord.X	Cord.Y	Cord.Z	Angle	Dropout	Last piece
3	3.0135	2.0641	-318.98	1	60	1

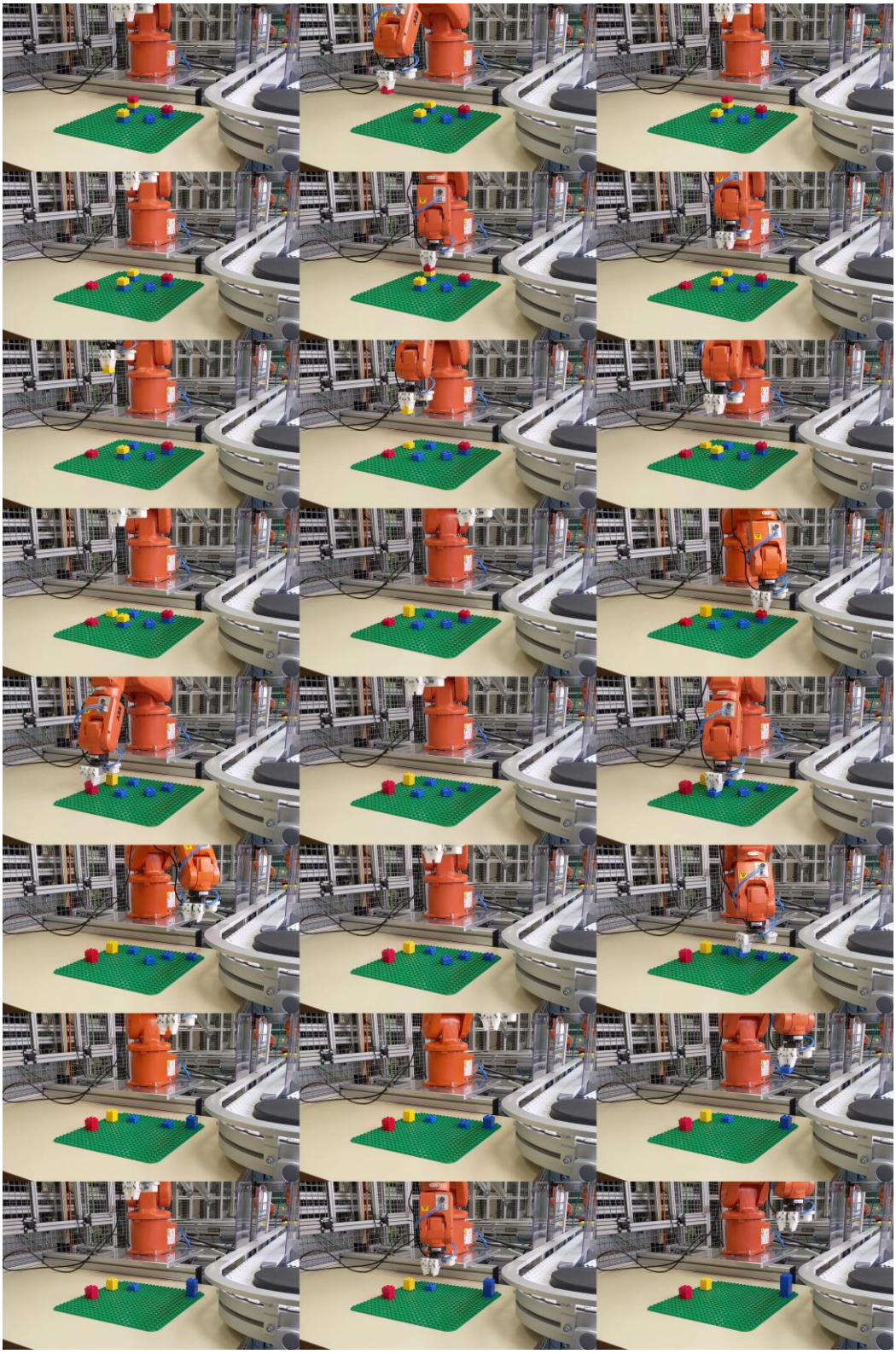
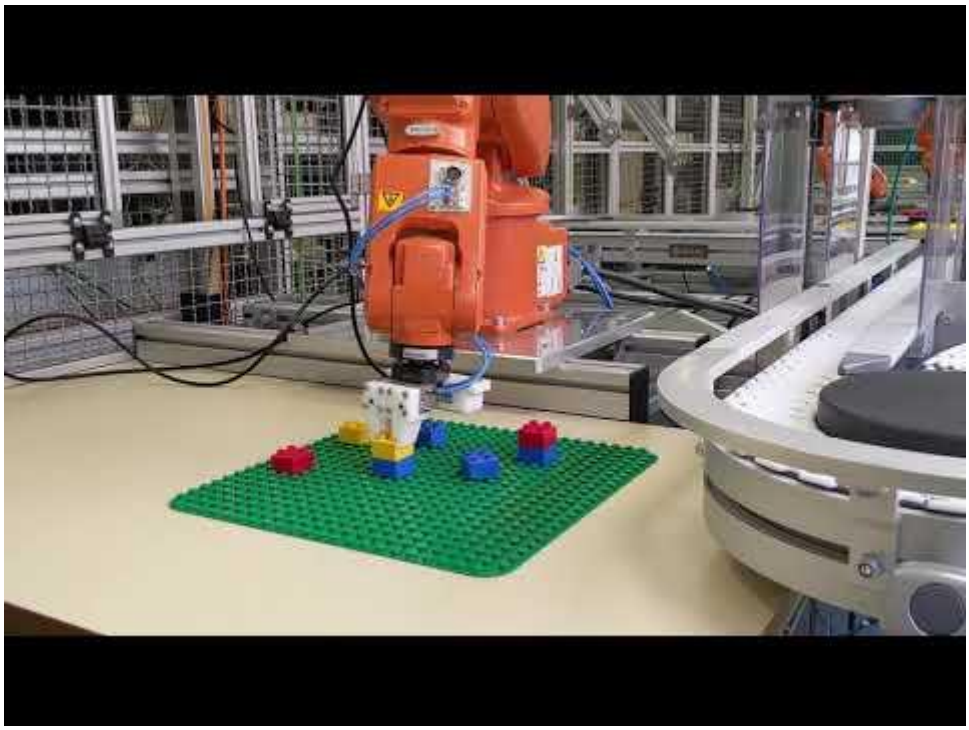


Figura 26. Resumen del desempeño en el caso 3

Respecto al vídeo para este caso, desgraciadamente, el estado de los tapetes del laboratorio no es el mejor, por lo que en ocasiones las piezas no se agarran suficientemente bien al mismo, dificultando la maniobra de extracción de pilas y arrastrando en ocasiones el resto de piezas de la pila.



Vídeo 3. Caso 3

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

A lo largo del desarrollo de este proyecto, se han logrado los siguientes hitos:

- Finalizado el desarrollo de la app comenzada en [3], permitiendo la operación de todos los módulos desde una sola interfaz.
- Modificado el código del robot elaborado en [6], añadiendo comunicación bidireccional entre el robot y el PC, para permitir la solicitud de imágenes correctivas.
- Desarrollada una variante de PSO para optimizar la distancia recorrida por el robot a fin de planificar en qué orden se toman las piezas.
- Implementado un método para evitar colisiones del robot con pilas de piezas, haciendo que vaya trabajando por alturas.
- Introducido la etapa de corrección al funcionamiento del sistema, compensando la distorsión radial de la cámara y el desplazamiento del TCP del robot al agarrar piezas giradas con ángulos grandes.
- Perfeccionado el funcionamiento del robot, siendo ahora capaz de interactuar con piezas apiladas o rotadas sin problemas.
- Planteado una primera implementación de *Template Matching* para corregir el desempeño de las CNN.

La siguiente tabla resume las mejoras implementadas en el sistema actual respecto a versiones anteriores:

Sistemas anteriores	Sistema actual
No tiene manera de evitar colisiones	Trabaja por alturas para evitar colisiones con pilas de piezas
Sólo el análisis de máscaras de color es compatible con el movimiento del robot	Tanto análisis de máscaras como detección por CNN son compatibles
Necesita discretizar las posiciones de las piezas en el tapete	Trabaja sin discretizar posiciones
Toma las piezas en orden aleatorio	Determina una ruta óptima para tomar las piezas
La comunicación con el robot es unidireccional	La comunicación es bidireccional
No se consideran posibles colisiones de mensajes	Se garantiza que el socket siempre está limpio para que no haya colisiones
Se usa una sola imagen en todo el proceso	Se usan varias imágenes en el proceso para corregir la detección
No hay posibilidad de recentrar las *bounding boxes*	Uitiliza *Template Matching* para recentrar las *bounding boxes* (en condiciones adecuadas)

No obstante, el sistema todavía tiene algunas limitaciones sobre las que se podrían trabajar en proyectos futuros:

- El sistema por el momento sólo trabaja con piezas de LEGO cuadradas. Emplear piezas rectangulares u otros objetos que requieran de agarres no triviales puede ser una futura vía de desarrollo.
- Las CNN entrenadas en [3] tienen problemas para detectar piezas de color rojo en el laboratorio. En ocasiones no se detectan o se confunden por piezas de color amarillo.
- En ocasiones el sistema tiene problemas para determinar la profundidad de las piezas de color azul, lo cual seguramente esté relacionado por la proximidad del color azul de la pieza y el verde del tapete en la imagen en escala de grises con la que trabaja el

sensor de profundidad infrarrojo. Otras vías para estimar la profundidad de las piezas podrían investigarse.

- Las CNN encargadas de la detección del ángulo solamente devuelven ángulos positivos. Esto resulta intuitivo para un humano, pero para el robot es importante la distinción entre ángulos positivos y negativos para lograr un agarre correcto. Tal y como está ahora, solo puede agarrar piezas que hayan sido giradas hasta 45º en sentido horario.
- Las etapas de comunicación son lentas debido a la ausencia de un hilo dedicado a las mismas.

-

Capítulo 8. BIBLIOGRAFÍA

[1] "Amazon is planning a $40M robotics hub near Boston | TechCrunch." https://techcrunch.com/2019/11/06/amazon-is-planning-a-40m-robotics-hub-near-boston/ (accessed Oct. 06, 2020).
[2] "Ocado overtakes Tesco as most valuable UK retailer - BBC News." https://www.bbc.co.uk/news/business-54352540 (accessed Oct. 06, 2020).
[3] I. Ortiz de Zúñiga Mingot, "Optimización del sistema de visión artificial de un robot industrial para una aplicación de pick and place," *Trabajo Final de Grado, ICAI*, Jul. 2020.
[4] "What are Convolutional Neural Networks?," Jan. 06, 2021. https://www.ibm.com/cloud/learn/convolutional-neural-networks (accessed May 31, 2021).
[5] "Particle swarm optimization," *Wikipedia*. May 18, 2021. Accessed: Jun. 16, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Particle_swarm_optimization&oldid=1023895887
[6] A. Berjón Valles, "Integración de un Sistema de Visión Artificial en la Mano de un Robot Industria," *Trabajo Final de Grado, ICAI*, Jul. 2019.
[7] F. Nagata *et al.*, "Design Tool of Deep Convolutional Neural Network for Intelligent Visual Inspection," *IOP Conference Series: Materials Science and Engineering*, vol. 423, p. 012073, Nov. 2018, doi: 10.1088/1757-899X/423/1/012073.
[8] W. Yu and C. Zhao, "Broad Convolutional Neural Network Based Industrial Process Fault Diagnosis With Incremental Learning Capability," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 6, pp. 5081–5091, Jun. 2020, doi: 10.1109/TIE.2019.2931255.
[9] X. Bian, S. N. Lim, and N. Zhou, "Multiscale fully convolutional network with application to industrial inspection," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2016, pp. 1–8. doi: 10.1109/WACV.2016.7477595.
[10] S. Y. Jung, Y. H. Tsai, W. Y. Chiu, J. S. Hu, and C. T. Sun, "Defect Detection on Randomly Textured Surfaces by Convolutional Neural Networks," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2018, pp. 1456–1461. doi: 10.1109/AIM.2018.8452361.
[11] Q. Xuan *et al.*, "Automatic Pearl Classification Machine Based on a Multistream Convolutional Neural Network," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6538–6547, 2018.
[12] N. D. Truong, A. D. Nguyen, L. Kuhlmann, M. R. Bonyadi, J. Yang, and O. Kavehei, "A Generalised Seizure Prediction with Convolutional Neural Networks for Intracranial and Scalp Electroencephalogram Data Analysis," *arXiv:1707.01976 [cs]*, Dec. 2017, Accessed: Oct. 15, 2020. [Online]. Available: http://arxiv.org/abs/1707.01976
[13] M. Kravchik and A. Shabtai, "Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, New York, NY, USA, Jan. 2018, pp. 72–83. doi: 10.1145/3264888.3264896.
[14] Y. Lai, J. Zhang, and Z. Liu, "Industrial Anomaly Detection and Attack Classification Method Based on Convolutional Neural Network," *Security and Communication Networks*, vol. 2019, p. 8124254, Sep. 2019, doi: 10.1155/2019/8124254.
[15] Z. Geng, Y. Zhang, C. Li, Y. Han, Y. Cui, and B. Yu, "Energy optimization and prediction modeling of petrochemical industries: An improved convolutional neural network based on cross-feature," *Energy*, vol. 194, p. 116851, Mar. 2020, doi: 10.1016/j.energy.2019.116851.
[16] "Amazon Robotics," *Amazon Robotics*. https://www.amazonrobotics.com/#/ (accessed Oct. 06, 2020).
[17] A. Zeng *et al.*, "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3750–3757.
[18] D. Morrison *et al.*, "Cartman: The Low-Cost Cartesian Manipulator that Won the Amazon Robotics Challenge," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7757–7764.

[19] R. Jonschkowski, C. Eppner, S. Höfer, R. Martín-Martín, and O. Brock, "Probabilistic multi-class segmentation for the Amazon Picking Challenge," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 1–7. doi: 10.1109/IROS.2016.7758087.

[20] M. Schwarz *et al.*, "NimbRo picking: Versatile part handling for warehouse automation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3032–3039.

[21] T. Lozano-Perez and L. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3684–3691, 2014, doi: 10.1109/IROS.2014.6943079.

[22] F. Alet Puig, "Machine Learnig for Robotic Manipulation in cluttered environments," PhD Thesis, UPC, Facultat de Matemàtiques i Estadística, Departament de Matemàtiques, 2016. [Online]. Available: http://hdl.handle.net/2117/87153

[23] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, and H. Onda, "Validating an object placement planner for robotic pick-and-place tasks," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1463–1477, 2014, doi: https://doi.org/10.1016/j.robot.2014.05.014.

[24] J. Vachálek, L. Čapucha, P. Krasňanský, and F. Tóth, "Collision-free manipulation of a robotic arm using the MS Windows Kinect 3D optical system," in *2015 20th International Conference on Process Control (PC)*, 2015, pp. 96–106.

[25] "Travelling salesman problem branch and bound example ppt. The Travelling Salesman Problem. 2019-01-25." http://keplarllp.com/travelling-salesman-problem-branch-and-bound-example-ppt.html (accessed Jun. 16, 2021).

[26] M. Lin, Y. Zhong, J. Lin, and X. Lin, "Discrete Bird Swarm Algorithm Based on Information Entropy Matrix for Traveling Salesman Problem," *Mathematical Problems in Engineering*, Oct. 30, 2018. https://www.hindawi.com/journals/mpe/2018/9461861/ (accessed Dec. 30, 2020).

[27] E. F. G. Goldbarg, M. C., and G. R. de Souz, "Particle Swarm Optimization Algorithm for the Traveling Salesman Problem," in *Traveling Salesman Problem*, F. Greco, Ed. InTech, 2008. doi: 10.5772/5580.

[28] Y. Saji, M. E. Riffi, and B. Ahiod, "Discrete bat-inspired algorithm for travelling salesman problem," in *2014 Second World Conference on Complex Systems (WCCS)*, Nov. 2014, pp. 28–31. doi: 10.1109/ICoCS.2014.7060983.

[29] R. F. Abdel-Kader, "Fuzzy Particle Swarm Optimization with Simulated Annealing and Neighborhood Information Communication for Solving TSP," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 2, no. 5, Art. no. 5, Nov. 2012, doi: 10.14569/IJACSA.2011.020503.

[30] S. Sengupta, S. Basak, and R. II, "Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives," Apr. 2018, doi: 10.3390/make1010010.

ANEXO – ALINEACIÓN CON LOS ODS

Este proyecto se alinearía particularmente bien con los siguientes ODS:

- 8.2: Lograr niveles más elevados de productividad económica mediante la diversificación, la modernización tecnológica y la innovación, entre otras cosas centrándose en los sectores con gran valor añadido y un uso intensivo de la mano de obra. El uso de sistemas robóticos autónomos dotados de visión artificial es un elemento altamente innovador, con mucho potencial en industrias con gran uso de mano de obra, como pueden ser la del almacenamiento y logística (Amazon, DHL, etc.)
- 9.5: Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países, en particular los países en desarrollo, entre otras cosas fomentando la innovación y aumentando considerablemente, de aquí a 2030, el número de personas que trabajan en investigación y desarrollo por millón de habitantes y los gastos de los sectores público y privado en investigación y desarrollo. Los sistemas robóticos autónomos son indudablemente innovadores y están estrechamente ligados al sector industrial, de ahí la contribución de este proyecto al objetivo 9.5