

PROYECTO FINAL

Visión por Computador

Grupo 2

Gonzalo Maldonado Arana

Vanessa Elizabeth Mejía Fajardo

Itsasne Presumido Martínez-Conde

NOTA

Debido a limitaciones de tamaño en Alud, no ha sido posible subir todo el entorno de trabajo. Sin embargo, en el documento se incluye el enlace de acceso al repositorio en Git donde se encuentra todo el contenido.

› Índice

| | |
|---|-----------|
| Contexto..... | 3 |
| Objetivos..... | 4 |
| Estructuración del Espacio de Trabajo..... | 5 |
| Desarrollo..... | 6 |
| Segmentación de los Cubos..... | 6 |
| Obtención de la Vista Alzada y Perfil..... | 7 |
| Primeras Aproximaciones..... | 7 |
| Aproximación Final..... | 8 |
| Obtención de la Vista Superior..... | 13 |
| Primeras Aproximaciones..... | 13 |
| Aproximación Final..... | 15 |
| Obtención de la Localización y Orientación..... | 19 |
| Controlador de las Cámaras..... | 23 |
| Generación de Matriz 3D..... | 23 |
| Programa Principal..... | 25 |
| Interfaz..... | 26 |
| Funcionalidades Principales..... | 26 |
| Repository..... | 28 |

› Contexto

El proyecto tiene como objetivo integrar un sistema de visión artificial en un robot colaborativo para la **detección y reconstrucción de figuras hechas con cubos de colores**.

El flujo de trabajo sería el siguiente: un usuario construye una figura utilizando cubos, y el sistema captura tres imágenes desde diferentes ángulos: alzado, perfil y planta, al igual que una representación diédrica. Posteriormente, se procesan las imágenes para extraer características de la figura, como la cantidad de cubos, los colores y su disposición. [Ver Imagen 1]

En el espacio de trabajo del robot, los cubos estarán distribuidos sobre una mesa. El sistema de visión artificial será responsable de identificar la posición y el color de los cubos, proporcionando las coordenadas necesarias para que el robot pueda reconstruir la figura del usuario. [Ver Imagen 2]

La calibración de las cámaras, junto con la captura y el procesamiento de las imágenes, son aspectos clave del desarrollo de este proyecto en el marco de la asignatura.

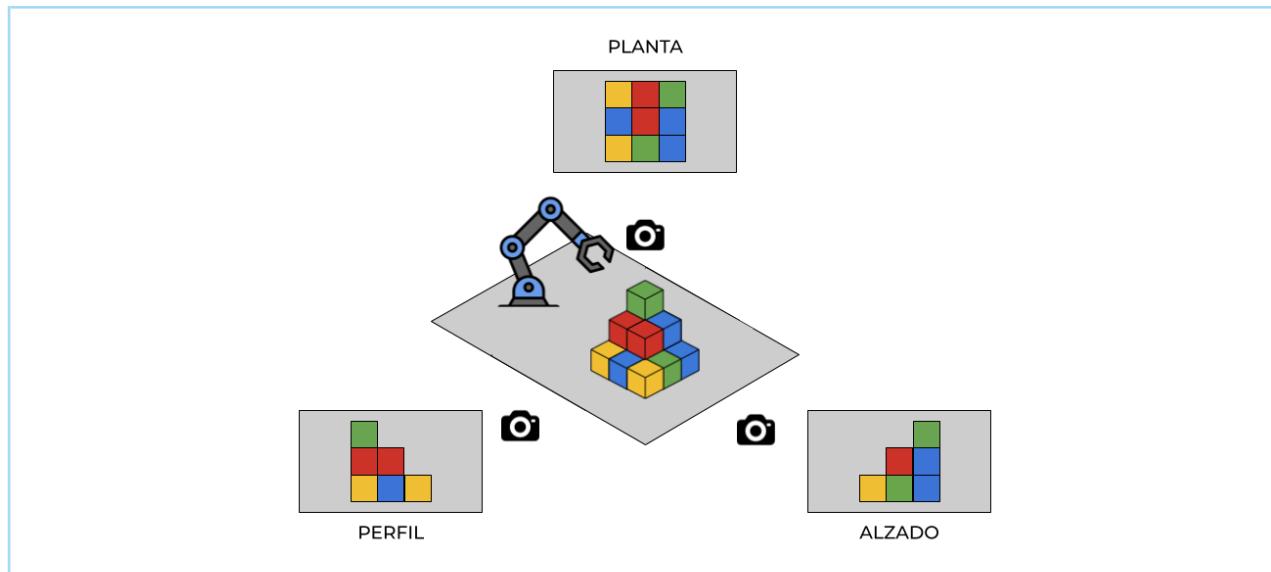


Imagen 1. Representación de las imágenes capturadas por las cámaras

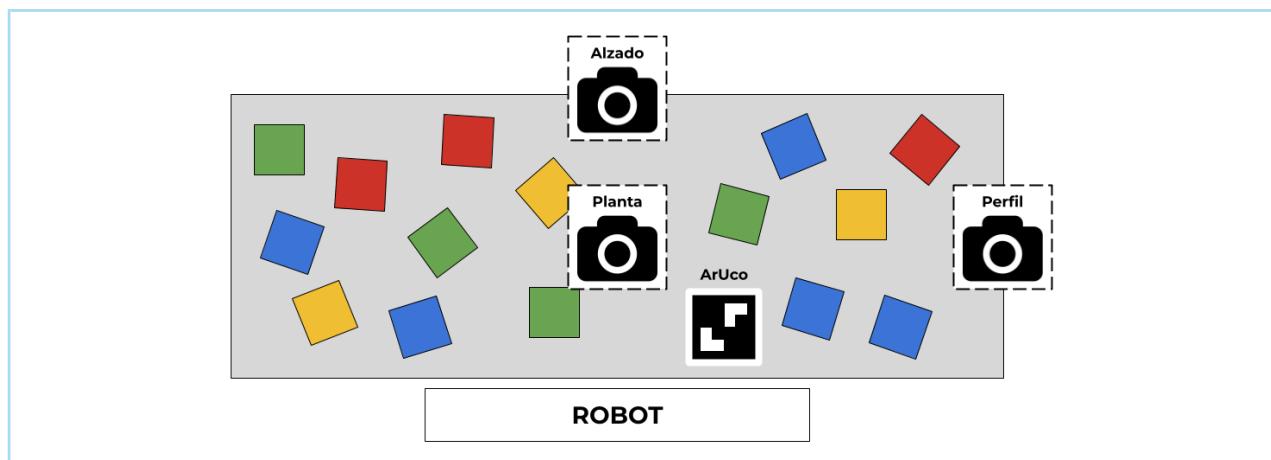


Imagen 2. Representación del espacio de trabajo del robot junto con las cámaras y cubos.

› Objetivos

Para lograr desarrollar el proyecto se han fijado los siguientes objetivos:

- **Objetivo General**

Desarrollar un sistema de visión artificial que permita detectar cubos de colores, extraer sus características y proporcionar al robot colaborativo las coordenadas necesarias para que éste pueda construir de manera parecida figuras creadas por un usuario.

- **Objetivos Secundarios**

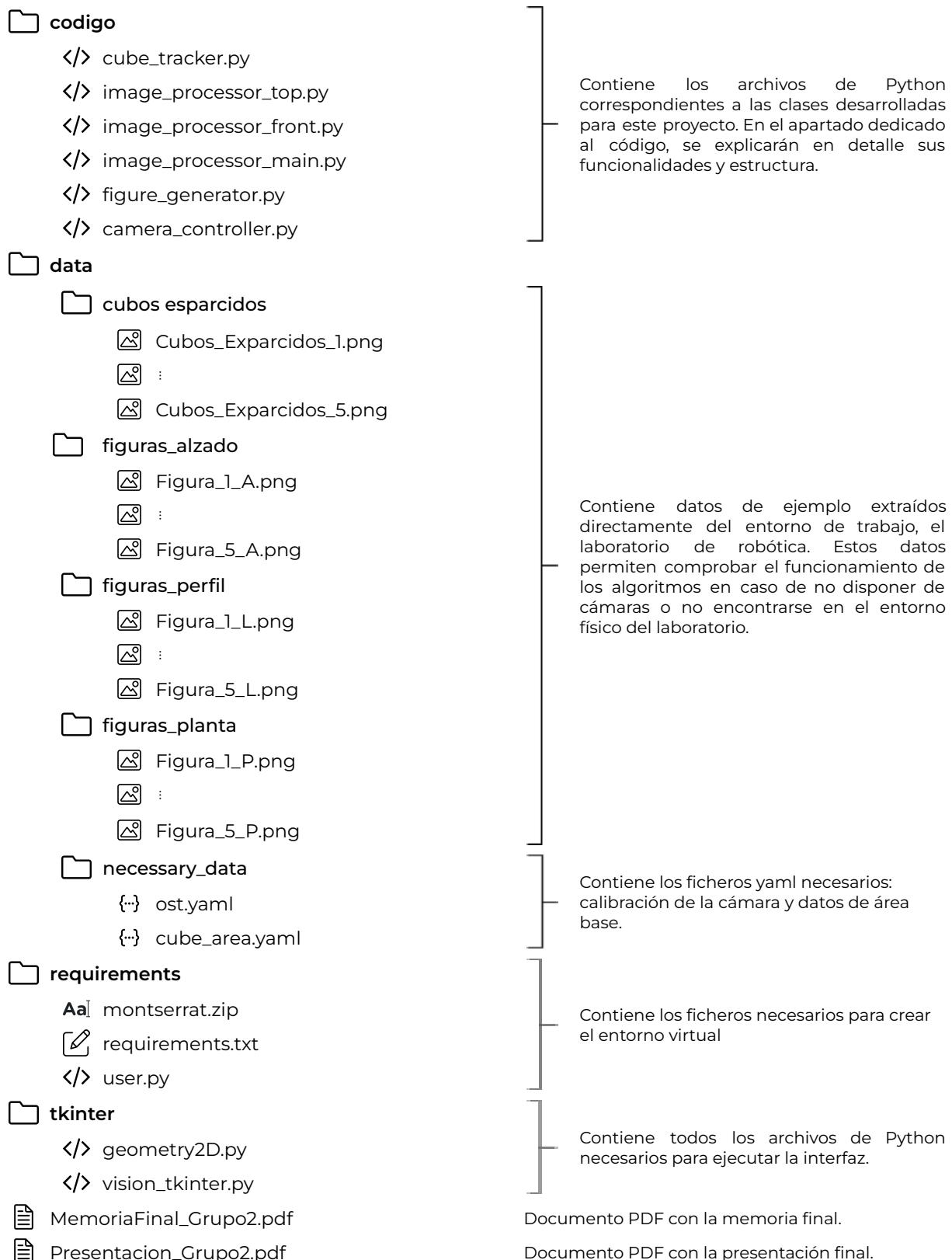
- Planificar y diseñar la disposición de las cámaras en el área de trabajo del robot.
- Crear un algoritmo de calibración que establezca una relación precisa entre los píxeles de la imagen y las unidades métricas reales.
- Desarrollar uno o varios algoritmos de procesamiento de imágenes que permita obtener las características de las figuras construidas, como la segmentación por colores, la detección de bordes, el posicionamiento y el conteo de cubos.
- Localizar y segmentar los cubos según su color, considerando su orientación y posición en el espacio tridimensional.

- **Objetivos Opcionales**

- Desarrollar una aplicación de visualización interactiva del proceso.
- Implementar un sistema de visión que permita desmontar la figura para analizar los cubos que no son visibles inicialmente.

› Estructuración del Espacio de Trabajo

Con el fin de facilitar la comprensión del espacio de trabajo del proyecto, se presenta a continuación un diagrama resumen que detalla las localizaciones de los archivos y su respectiva funcionalidad:



› Desarrollo

En este apartado se describe el desarrollo del proyecto, cuyo objetivo principal es segmentar cada uno de los cubos presentes en las imágenes capturadas. El primer paso consistió en configurar el entorno de trabajo, posicionando las cámaras a la altura y ubicación necesarias para obtener la mejor calidad de imagen posible.

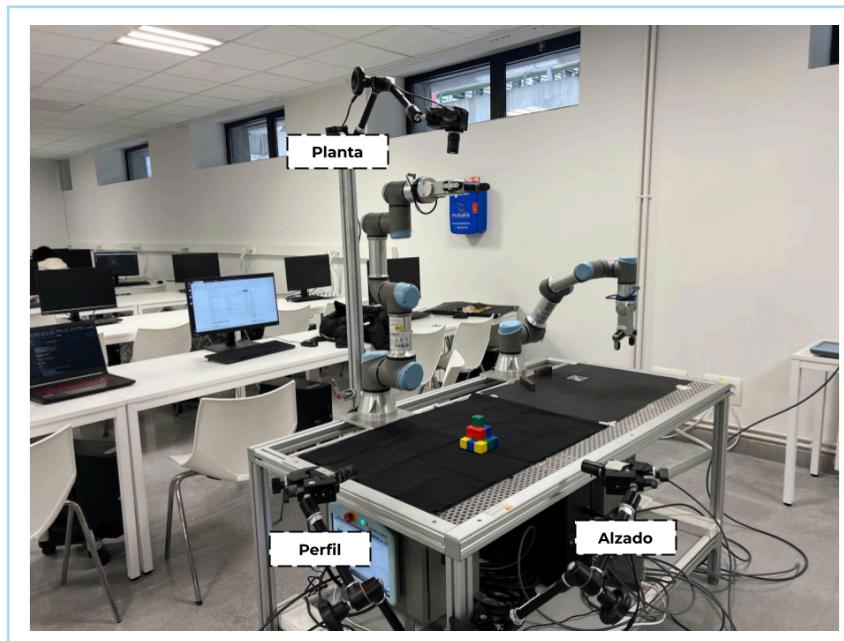


Imagen 3. Posicionamiento de las cámaras en el espacio de trabajo

Después, se desarrollaron varios archivos en Python que facilitaron la gestión de las cámaras, la creación de los algoritmos y el procesamiento final de las imágenes.

› Segmentación de los Cubos

Como se ha mencionado, la segmentación se realiza con fines diferentes según el caso: en tres de ellos, se busca obtener la vista de la figura, mientras que en el otro, se requiere determinar la posición espacial de los cubos.

Aunque el objetivo principal de segmentación es común a todas las situaciones, las condiciones del entorno varían significativamente entre estas. Por esta razón, fue necesario desarrollar tres algoritmos distintos, cada uno diseñado para abordar las especificidades de cada escenario, a excepción de las vistas de alzado y perfil que tienen un escenario similar. Cada uno de estos algoritmos se implementó en una clase separada utilizando Python, lo que permite modularidad, reutilización y adaptabilidad en el desarrollo del sistema. Además, también simplificó su integración en la interfaz.

Obtención de la Vista Alzada y Perfil

La segmentación de los cubos en las cámaras laterales tiene como objetivo obtener la descripción de la figura desde las vistas de alzado y perfil. Para ello, se emplean métodos de visión artificial a través de los cuales se delimita la localización de los cubos en la imagen y se determina el número de cubos visibles.

Las dos cámaras utilizadas adquieren imágenes del alzado y perfil de la figura de la siguiente manera:

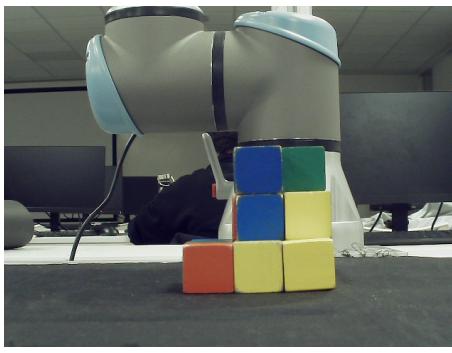


Imagen 4. Perspectiva Alzada Imagen Original

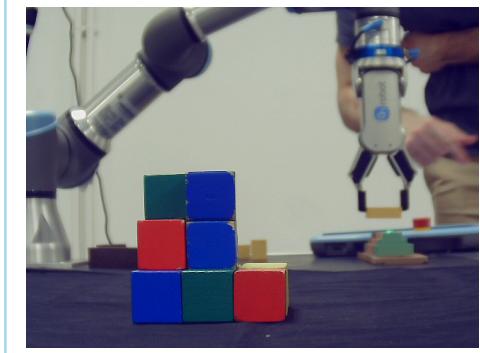


Imagen 5. Perspectiva Perfil Imagen Original

En las imágenes se observan los cubos formando la figura a procesar, además del fondo del laboratorio, que se desea eliminar.

Primeras Aproximaciones

Aproximación 1: Canny, Sobel y Morfología

La aproximación inicial se centró en resaltar los bordes de los cubos mediante técnicas de detección de contornos. Para ello, se optó por aplicar un filtro combinado que se enfocara en localizar las líneas verticales y horizontales que forman los lados rectos de los cubos.

Una vez destacadas las formas cuadradas en la imagen, se utilizó el filtro Canny, que permite reducir el ruido generado por el filtro Sobel. Posteriormente, se aplicó la operación morfológica cierre para cerrar los bordes, seguida de una búsqueda de contornos y un filtrado por tamaño con el fin de identificar los contornos de los cubos.

Sin embargo, este enfoque presenta una limitación significativa, ya que depende en gran medida de la nitidez y la iluminación de la imagen. Es especialmente sensible a sombras y al ruido, lo que puede ocasionar la detección de contornos erróneos o superpuestos.

Aproximación 2: Aproximación 1 + Umbralización y Filtrado de Bordes por jerarquía

Para mejorar la robustez del sistema, se incorporaron dos mejoras principales al método anterior. La primera consistió en la umbralización del filtro Sobel, lo que permitió utilizarlo como una máscara sobre la imagen inicial, resaltando aún más los bordes a localizar con el filtro Canny y eliminando de manera significativa el ruido y parte del fondo de la imagen.

La segunda mejora fue la aplicación de un filtro de contornos basado en la jerarquía, que selecciona únicamente aquellos contornos que tienen relaciones de parente e hijo. Esta técnica eliminó de manera inmediata los contornos más grandes y los más pequeños, facilitando la filtración por tamaño y reduciendo la interferencia del ruido.

Aunque este método disminuye el error causado por el ruido, durante las pruebas en el laboratorio se observó que su robustez frente a objetos en el fondo era limitada. Además, se presentaron errores de detección en los cubos con colores similares, lo que comprometió la precisión del sistema en ciertas condiciones.

Aproximación 3: Aproximación 2 + Aumento de Contrastes y Filtrado por Cuadraticidad

Debido a los problemas de detección derivados de la iluminación, se optó por aplicar un aumento de contraste, lo que favoreció una mayor diferenciación entre la superficie relativamente homogénea de los cubos y sus bordes. Esta estrategia mejoró la robustez frente a sombras y facilitó la detección de los cubos.

Además, para eliminar los contornos generados por ruido o por objetos en el fondo, se implementó un filtro basado en la cuadraticidad, que elimina todos los contornos que no sean cuadrados. Este filtro también es capaz de detectar contornos rectangulares que pueden detectar varios cubos erróneamente unidos como uno solo.

Aunque esta aproximación resulta ser más robusta y eficaz, sigue presentando dificultades con los objetos en el fondo de la imagen y enfrenta problemas al intentar detectar los cubos ubicados a mayor profundidad en la escena.

Aproximación Final

Finalmente, tras las diferentes versiones del programa, se ha decidido juntar el trabajo realizado hasta el momento y aplicarlo con un enfoque diferente. De esta forma el proceso final aplicado a la imagen para obtener el resultado deseado es el siguiente:

Paso 1: Filtrado por Color en HSV

El primer paso del proceso consiste en convertir la imagen original en formato BGR a la escala de colores HSV. Esta transformación se realiza debido a que las características del modelo HSV ofrecen ventajas significativas frente al modelo BGR para este proyecto. En HSV, el componente de tono (H) describe el color, la saturación (S) indica la intensidad del color y el valor (V) representa la iluminación. Esta representación permite identificar los colores de manera más precisa y sencilla, además de ser más robusta ante variaciones en la iluminación.

De este modo, se segmentan los colores presentes en la imagen en los cuatro colores clave que se buscan identificar (rojo, verde, azul y amarillo), registrando su localización en una máscara binarizada. Esta máscara se aplica sobre el fotograma original, lo que permite aislar los cubos del fondo, eliminando en gran medida el ruido y la mayor parte de los elementos no relevantes en la imagen.

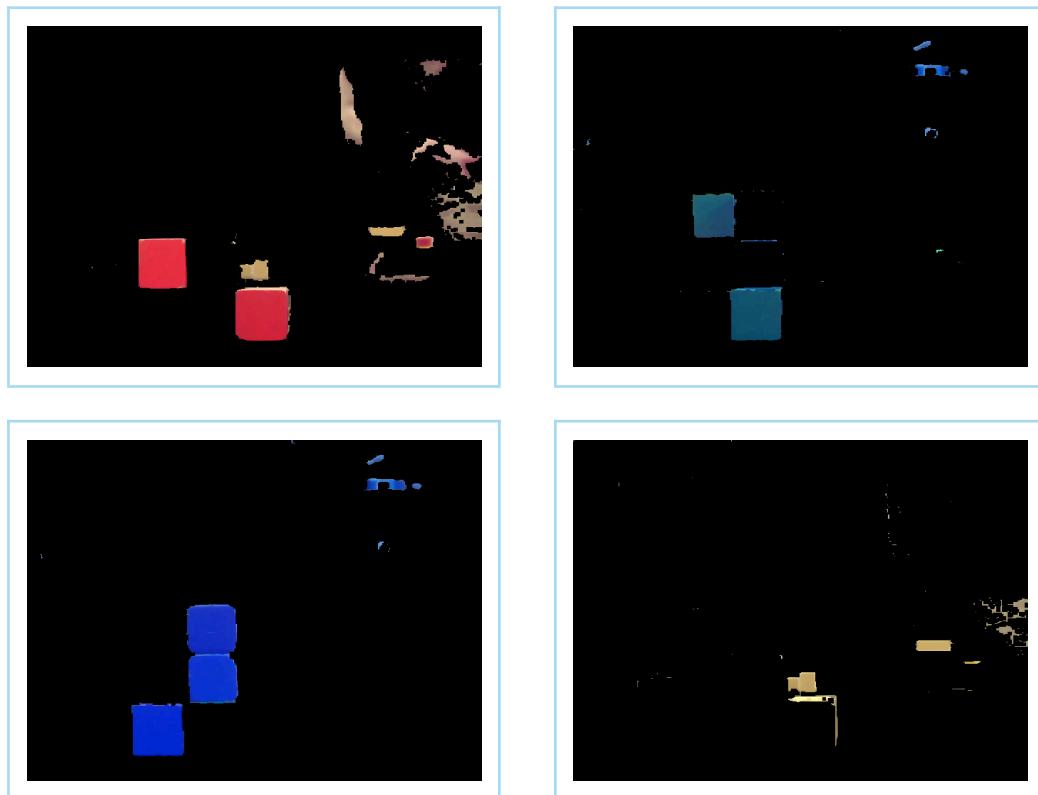


Imagen 6. Perspectiva Perfil Máscaras de Colores Segmentados



Imagen 7. Perspectiva Perfil Imagen tras Segmentación por colores

Paso 2: Gaussian Blur y Filtro Sobel

Una vez obtenida la máscara que localiza los cubos, la imagen se convierte a escala de grises con el fin de facilitar la aplicación de filtros, en este caso el filtro de Sobel.

Con la imagen en escala de grises, el primer paso es aplicar un filtro gaussiano suave que difumina la imagen. Este paso tiene dos objetivos fundamentales antes de aplicar el filtro Sobel: eliminar el ruido y homogeneizar la superficie de los cubos. Esto se debe a que las cámaras suelen generar imágenes de baja resolución con niveles moderados de ruido, y los cubos, ya sea por la iluminación o por imperfecciones en su superficie, pueden generar falsos positivos en la detección de bordes.

Finalmente, sobre la imagen difuminada en escala de grises, se aplican dos filtros Sobel: uno enfocado en las líneas verticales y otro en las horizontales. La combinación de ambos filtros permite identificar la localización relativa de los bordes de los cubos.

Sin embargo, este proceso de filtrado mediante Sobel genera múltiples detecciones de ruido en la superficie de los cubos. Dado que estas detecciones son más superficiales y presentan una menor coincidencia, se representan con valores inferiores en el resultado del filtro. Con una posterior umbralización, es posible obtener una máscara binarizada que resalte los bordes más significativos.

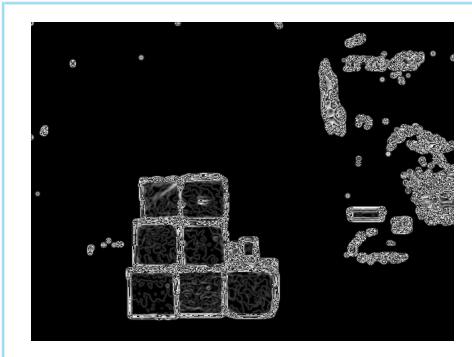


Imagen 8. Perspectiva Perfil filtro Sobel



Imagen 9. Perspectiva Perfil Sobel Umbralizado

Paso 3: Aislamiento de Cubos y Filtro Canny

El último paso del preprocesamiento de la imagen se centra en aislar los cubos de la manera más precisa posible y obtener sus contornos. Para ello, el primer paso consiste en aplicar un filtro Sobel umbralizado como máscara de forma inversa, es decir, en lugar de conservar los bordes de los cubos, se conserva el área central de estos.

Esta máscara genera una imagen en escala de grises con los cubos completamente segmentados, a la cual se le aplica una umbralización con el único propósito de binarizar la imagen y poder extraer los bordes externos de los cubos mediante un filtro Canny.

Previo a la aplicación del filtro Canny, se llevan a cabo tres operaciones morfológicas. La primera consiste en una combinación de dos erosiones con máscaras de tamaño 3, que eliminan tanto el ruido residual como las pequeñas uniones entre cubos, comunes entre aquellos de colores similares. A continuación, se aplica una operación "Close" con una máscara de tamaño 3, que fusiona cualquier separación entre cubos provocada por variaciones en la iluminación. Finalmente, se realiza una última erosión que define con mayor precisión los límites entre los cubos.

Una vez aplicados todos los filtros mencionados, el filtro Canny permite extraer únicamente los bordes externos de los cubos, facilitando la localización de los contornos y reduciendo la posibilidad de errores.

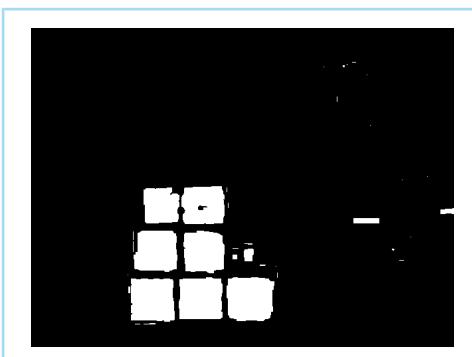


Imagen 10. Perspectiva Perfil Umbralización Sobel Inversa

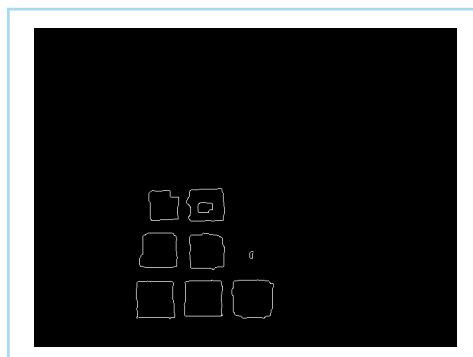


Imagen 11. Perspectiva Perfil Resultado Final

Paso 4: Filtrado de Contornos y Localización del color

Para convertir la información de la imagen en cubos, es necesario localizar todos los cubos detectados durante el preprocesamiento y determinar el color que representan. Este proceso se lleva a cabo inicialmente localizando los contornos mediante la función `findContours` de OpenCV, seguida de un análisis de los contornos y su localización.

Una vez localizados los contornos, se realiza un filtrado por tamaño para eliminar cualquier ruido residual. Este filtrado se basa en el área del contorno, la cual se calcula utilizando la función `contourArea` de OpenCV. Se establece un área mínima de 500, lo que, teniendo en cuenta que el tamaño promedio de los cubos varía entre 1500 y 3500 píxeles, proporciona un rango lo suficientemente amplio para detectar cubos que estén parcialmente visibles.

Tras obtener el contorno, se calcula el *Bounding Box* (rectángulo delimitador) de este y se determina su centro dentro de la imagen. Con las coordenadas del centro, se busca en las máscaras obtenidas en el primer paso, donde se aislaban los diferentes colores de los cubos en la escala HSV. Si el valor del píxel correspondiente al centro del contorno es 255 en la máscara, se asigna el color correspondiente a esa máscara como un valor entero.

Los valores asignados a cada color son los siguientes:

- **Rojo:** 0
- **Verde:** 1
- **Azul:** 2
- **Amarillo:** 3
- **No detectado:** 4 = Error (En este paso del proceso no se admiten otros valores)

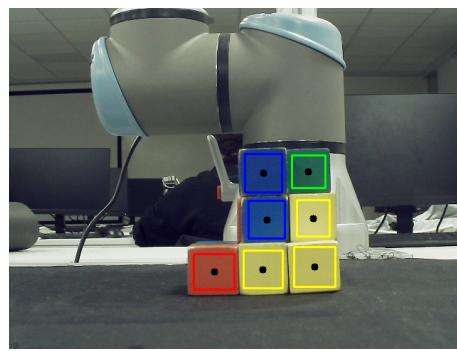


Imagen 12. Perspectiva de alzado con contornos coloreados y centros

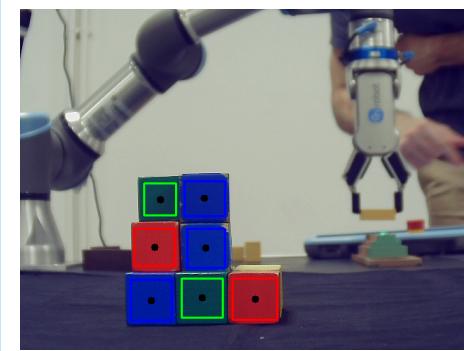


Imagen 13. Perspectiva de perfil con contornos coloreados y centros

Paso 5: Alineamiento de Cubos y Almacenamiento en Matriz

Finalmente, para representar los cubos detectados, su posición y color se visualizan en una matriz 5x5. Para ello, se genera una rejilla sobre la cual se determina el índice de la matriz del cubo en función de su ubicación en relación con cada fila y columna de la rejilla.

Dado que los contornos originales han sido erosionados y los bordes eliminados, se toma el lado más grande del *Bounding Box* del cubo como referencia. Luego, se considera el cubo cuyo centro se encuentra en la posición con la coordenada x más baja (más a la derecha de la imagen) y la coordenada y más alta (más abajo de la imagen) como origen.

A partir de este punto, se crea una rejilla con 5 espacios, cuyos lados están separados por una distancia igual al valor del lado del *BoundingBox* mencionado anteriormente.

Una vez generada la rejilla, se calcula la distancia entre el centro de cada cubo y la fila y columna más cercanas en la rejilla. Posteriormente, se asigna la posición de cada cubo en la matriz 5x5 según su proximidad a la fila y columna correspondientes, y se guarda el valor del color de ese cubo en la posición adecuada de la matriz.

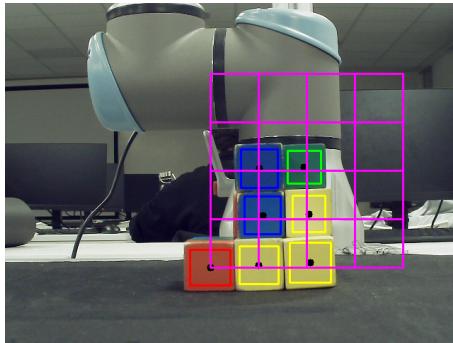


Imagen 14. Perspectiva alzado con la conversión de los contornos a matriz

| | | | | |
|----|----|----|----|----|
| -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 |
| -1 | 2 | 1 | -1 | -1 |
| -1 | 2 | 3 | -1 | -1 |
| 0 | 3 | 3 | -1 | -1 |

Perspectiva alzado matriz resultante del proceso

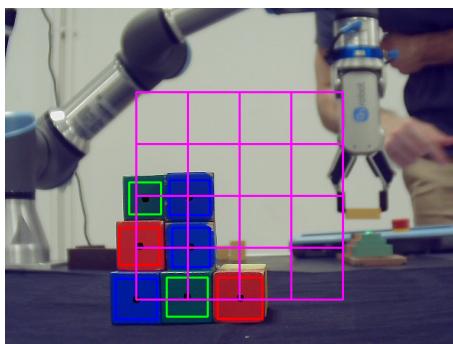


Imagen 15. Perspectiva perfil con la conversión de los contornos a matriz

| | | | | |
|----|----|----|----|----|
| -1 | -1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 | -1 |
| 1 | 2 | -1 | -1 | -1 |
| 0 | 2 | -1 | -1 | -1 |
| 2 | 1 | 0 | -1 | -1 |

Matriz resultante de la perspectiva de perfil.

Obtención de la Vista Superior

La segmentación de los cubos en la cámara superior tiene como objetivo obtener la descripción de la figura desde la vista de planta de esta. Para ello se emplean métodos de visión artificial a través de los cuales se delimita la localización de los cubos en la imagen y se determina el número de cubos visibles.

La cámara utilizada adquiere imágenes de la planta de la figura de la siguiente manera:



Imagen 16. Perspectiva Planta Imagen Original

En la imagen se observan los cubos formando la figura a procesar sobre un fondo negro.

Primeras Aproximaciones

Aproximación 1: Umbralización + Canny + Operaciones Morfológicas

La aproximación inicial consistió en umbralizar la imagen utilizando métodos de umbralización global, como el de Otsu, o un enfoque de umbralización local adaptativa. Este proceso permite separar los cubos del fondo mediante una máscara, reduciendo significativamente el ruido en la imagen.

Con los cubos segmentados de la imagen original, se optó por aplicar un filtro Canny, ya que, a diferencia de la vista frontal, en esta imagen los cubos están más alejados, lo que hace que filtros como Sobel se vean demasiado afectados por el ruido. Para ello, se utilizó un filtro Canny con un valor umbral bajo, que se empleó como máscara para eliminar los bordes de la imagen en escala de grises, lo cual incrementa el contraste en los bordes más definidos. Después de aplicar esta máscara, se utilizó un filtro Canny más agresivo para detectar los bordes de la imagen con mayor precisión.

Finalmente, al resultado del último filtro Canny, se le aplicaron operaciones morfológicas de cierre (Close) para fusionar los bordes que, debido a variaciones en la iluminación o al ruido, no estaban completamente conectados.

El principal inconveniente de este método radica en su dependencia de una imagen clara, sin cambios de iluminación. Aunque inicialmente funcionaba correctamente en condiciones estables, en el laboratorio generaba muchos falsos positivos debido al ruido y las sombras, lo que complicaba la detección precisa de los cubos.

Aproximación 2: Aproximación 1 + Filtrado de Bordes por Jerarquía, Tamaño y Cuadraticidad

Partiendo del procesamiento previo, se decidió aplicar tres tipos de filtros a los contornos obtenidos. El primero de estos filtros se basa en la jerarquía de los contornos, el segundo en su tamaño y el tercero en la cuadraticidad del contorno.

Inicialmente, al igual que en la vista frontal, se verificó que los contornos correspondieran a los bordes externos de los cubos, y no a figuras no deseadas. Para ello, se buscó un contorno hijo de uno mayor y, a su vez, que tuviera contornos dentro de sí mismo, es decir, que fuera un contorno "padre". Este filtro permite eliminar los contornos demasiado pequeños o demasiado grandes, centrando la atención sólo en los cubos.

Una vez aplicado este primer filtro, se introdujo un segundo filtro basado en el área de los contornos. Para ello, se calculó la mediana del área de todos los contornos y, a partir de este valor, se definieron límites inferior y superior. Los contornos válidos se restringieron a aquellos cuyo tamaño estuviera entre el 50% (0.5) y el 150% (1.5) de la mediana, eliminando así los contornos demasiado pequeños o excesivamente grandes.

Finalmente, para eliminar cualquier contorno residual que pudiera ser causado por ruido o imperfecciones en los cubos, se aplicó un filtro basado en la cuadraticidad del contorno. Este filtro compara la altura y la anchura del contorno, verificando que la diferencia entre estas dimensiones no supere un umbral definido de 20 píxeles. Esto elimina los contornos que, en lugar de ser cuadrados, son rectangulares, garantizando que solo se mantengan los contornos correspondientes a los cubos.

Esta aproximación, aunque mejoró los resultados en comparación con el caso anterior, limitaba significativamente los contornos que correspondían a los cubos. Como consecuencia, cualquier error en la detección resultaba en la eliminación del cubo de la lista de detección, lo que convertía el sistema en uno poco robusto. La alta dependencia de umbrales y filtros específicos hacía que, si un contorno no se ajustaba perfectamente a los criterios establecidos, el cubo quedaba fuera de la detección, afectando la fiabilidad del sistema en condiciones menos controladas.

Aproximación 3: Aproximación 2 + Separación de Contornos por Áreas

Partiendo de la aproximación anterior, se observó que los filtros que más impactaron en los contornos correspondientes a los cubos eran el límite superior de las áreas aceptadas y el filtro por cuadraticidad. Esto se debía a que la detección de bordes entre cubos del mismo color era muy poco robusta, lo que resultaba en la aparición de contornos que abarcaban varios cubos a la vez.

Para solucionar este problema, se decidió aplicar una separación de contornos basada en la mediana de las áreas previamente calculadas. De este modo, los contornos muy grandes se procesan de manera más detallada para determinar si abarcaban múltiples cubos. Esta separación se realiza utilizando como referencia el lado de la mediana, calculado a partir de la mediana del área de los contornos. Al dividir los lados del contorno por este valor, se determinaba si el contorno representaba uno o más cubos. Si el resultado de la división supera 1.85 veces el lado promedio, se consideraba que el contorno debía dividirse.

Una vez identificado que un contorno abarcaría varios cubos, se procedía a realizar la separación en el eje vertical u horizontal, utilizando el lado que había superado el umbral

de 1.85 veces el lado promedio para determinar la orientación de la división. Este enfoque mejoraba la robustez del sistema, permitiendo detectar y separar cubos que antes podían haber sido agrupados erróneamente en un solo contorno.

Esta aproximación, aunque relativamente robusta a cambios de iluminación, presenta un error crítico relacionado con su dependencia en el cálculo del área de los contornos. Este enfoque falla cuando varios cubos tienen el mismo color, ya que el sistema no puede diferenciarlos adecuadamente, lo que compromete la precisión de la mediana respecto a un cubo real. Además, otro problema relevante es que, cuando se encuentran contornos con forma de "L" (generados por cubos de un mismo color colocados de esa forma), el sistema interpreta el contorno como un contorno cuadrado el cual contenía 4 cubos en vez de 3, detectando un cubo donde ya se encontraba otro o directamente donde no había nada..

Aproximación Final

Finalmente, tras analizar las diferentes soluciones aplicadas, se decidió prescindir de la mayoría de los métodos utilizados para ofrecer una aproximación menos versátil pero mucho más robusta.

Esta aproximación realiza las siguientes tareas:

Paso 1: Umbralización de la Imagen

El primer paso del proceso consiste en umbralizar la imagen para separar los cubos del resto. En este caso dado que el fondo es estático y negro, esta umbralización se realiza mediante la función *thresholding* de OpenCV, a la cual se le indican los umbrales máximos y mínimos de los píxeles, y se le indica que el resultado debe ser una umbralización binaria a través del método *THRESH_BI_NARY*. Esta umbralización se aplica en forma de máscara a la imagen original para dejar únicamente las zonas más importantes de la imagen.

Durante este proceso, parte del ruido de la imagen no es siempre completamente eliminado del resultado final, pero no supone un problema mayor ya que es descartado posteriormente, a su vez, la aparición de ruido aleatorio tras el umbral es intencionado ya que se prioriza la correcta segmentación de la figura por encima de eliminar todo el fondo, por lo que el umbral tiene un valor más bajo que el necesario.

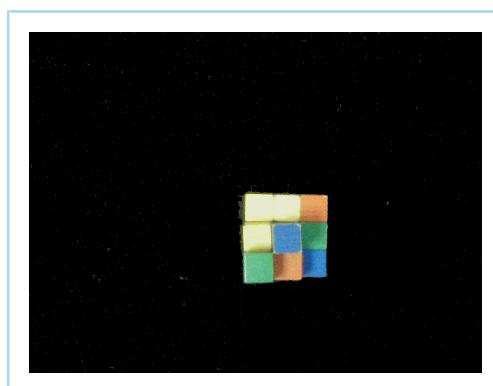


Imagen 17. Perspectiva Planta Imagen tras Umbralización

Paso 2: Segmentación de Colores a través del modelo de color HSV

Con la imagen umbralizada, el método más eficaz para segmentar los colores se basa en su distinción por rangos en el modelo de color HSV. Esto se debe a que, dada la resolución de la cámara, no se han encontrado técnicas más efectivas y robustas capaces de separar los cubos entre sí sin verse afectadas por el ruido.

De esta manera, al igual que en las vistas laterales, se asignan rangos específicos que permiten segmentar los diferentes colores de la figura. La principal diferencia entre esta segmentación y la descrita para las vistas laterales radica en el rango del color verde. Este color, debido a las variaciones en la iluminación del entorno, presenta grandes fluctuaciones, lo que complica la tarea de encontrar rangos de color válidos para localizar el verde en distintas figuras sin incluir otros colores o detectar ruido.

Por ello, la solución implementada consiste en generar una máscara combinando las detecciones de los demás colores y aplicarla a la imagen umbralizada original. Esto permite obtener una imagen que conserva solo los cubos verdes, eliminando los cubos de los demás colores.

Finalmente, se obtienen los contornos derivados de la segmentación de los diferentes colores, ofreciendo el contorno de los cubos que se encuentren separados o en caso de encontrar varios cubos de un mismo color unidos, se obtiene el contorno que los contenga todos.

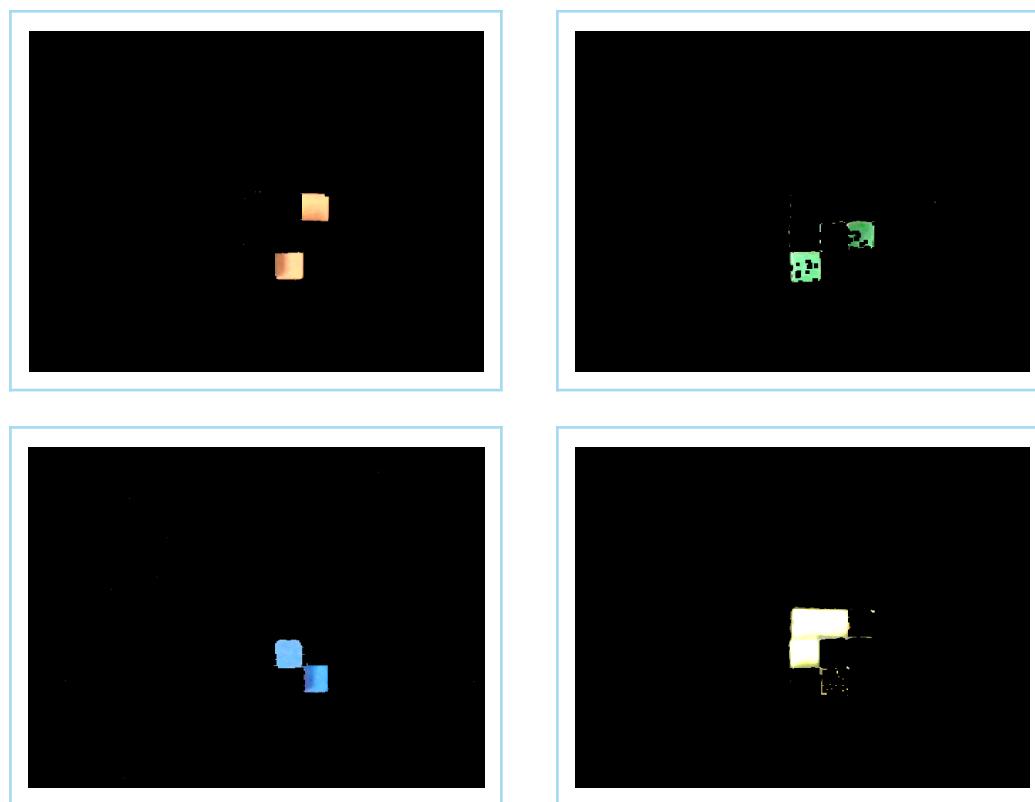


Imagen 18. Perspectiva Planta Máscaras de Colores Segmentados

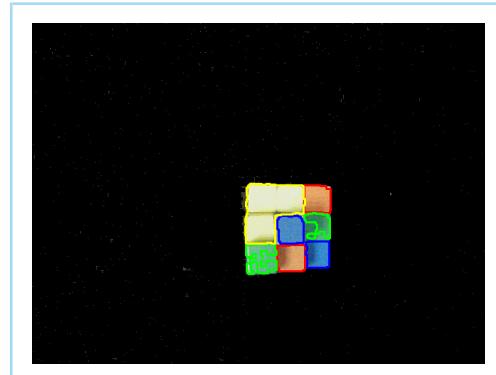


Imagen 19. Perspectiva Planta Contornos según color de cubos

Paso 2.5: Cálculo del Área del Cubo

Para llevar a cabo el paso 3, es necesario calcular el área media de un cubo. Esto se debe a que la calidad de la imagen capturada por la cámara no es lo suficientemente alta como para segmentar los cubos de manera efectiva mediante filtros clásicos.

En este sentido, el método utilizado para segmentar los cubos consiste en contar el número de cubos presentes en un mismo contorno, comparando las dimensiones de dicho contorno con las dimensiones de un cubo promedio. Para ello, se ha desarrollado una función llamada `calibrate_cube_area`, que toma una imagen de la cámara y, en caso de que solo haya un cubo dentro del rango de trabajo de la cámara, almacena su área en un archivo YAML. Este valor se extrae posteriormente de dicho archivo y se utiliza en el proceso de segmentación.



Imagen 20. Perspectiva Planta Imagen resultante de Calibración de Cubo

Paso 3: División de Contornos

Tal como se explicó en el paso 2.5, actualmente se cuentan tanto con contornos de cubos individuales como con contornos compuestos por varios cubos del mismo color. Para resolver este problema, se genera un Bounding Box en relación con el contorno capturado y se compara la diferencia entre el alto y el ancho de la caja con el alto y el lado promedio de un cubo. Si esta diferencia es al menos 1.6 veces el tamaño de un cubo promedio, se considera que hay dos cubos; en caso contrario, se asume que solo hay uno.

Esta comparación se realiza tanto en el rango vertical como horizontal. En el caso de que se detecten cubos en forma de "L", donde el Bounding Box calculado podría incluir un cubo fuera del contorno inicial, se asegura que el centro del cubo detectado tenga un

valor de 255 en la máscara umbralizada correspondiente al color en el que se haya detectado el contorno.

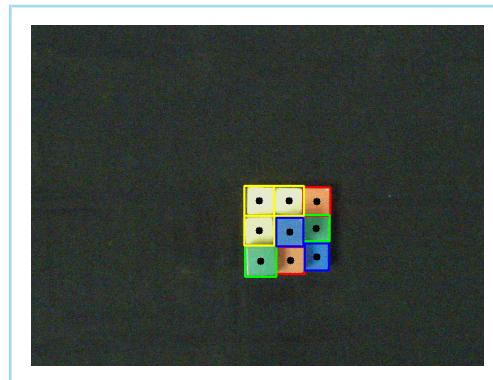


Imagen 21. Perspectiva Planta Imagen tras el procesamiento

Paso 4: Alineamiento de Cubos y Almacenamiento en Matriz

Finalmente, para representar los cubos detectados, su posición y color se visualizan en una matriz 5x5. Para ello, se genera una rejilla sobre la cual se determina el índice de la matriz del cubo en función de su ubicación en relación con cada fila y columna de la rejilla.

Dado que los contornos originales han sido erosionados y los bordes eliminados, se toma el lado más grande del *Bounding Box* del cubo como referencia. Luego, se considera el cubo cuyo centro se encuentra en la posición con la coordenada x más baja (más a la derecha de la imagen) y la coordenada y más alta (más abajo de la imagen) como origen. A partir de este punto, se crea una rejilla con 5 espacios, cuyos lados están separados por una distancia igual al valor del lado del *Bounding Box* mencionado anteriormente.

Una vez generada la rejilla, se calcula la distancia entre el centro de cada cubo y la fila y columna más cercanas en la rejilla. Posteriormente, se asigna la posición de cada cubo en la matriz 5x5 según su proximidad a la fila y columna correspondientes, y se guarda el valor del color de ese cubo en la posición adecuada de la matriz.

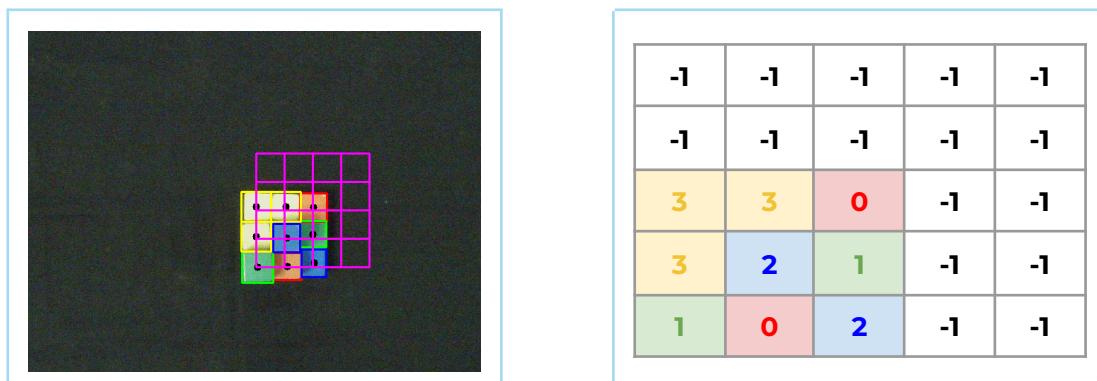


Imagen 22. Perspectiva Planta con conversión de contornos a Matriz

Matriz resultante de la perspectiva de planta.

Obtención de la Localización y Orientación

En esta última etapa de segmentación, el objetivo principal es determinar la localización y orientación de los cubos distribuidos sobre la mesa de trabajo. Para ello, se utiliza un marcador ArUco, que permite establecer una relación precisa entre las coordenadas de la cámara y el espacio real.

El marcador ArUco es un sistema empleado en visión artificial para la detección y seguimiento en tiempo real de objetos en un entorno tridimensional. Estos marcadores consisten en patrones cuadrados que contienen un identificador único en forma de código binario. Este código es interpretado por algoritmos especializados, permitiendo obtener información clave sobre la posición y orientación del marcador en relación con la cámara.

La cámara captura el espacio de trabajo del robot de la siguiente manera:



Imagen 23. Espacio de trabajo capturado desde la cámara superior.

En este entorno, se observa tanto el marcador ArUco como los cubos distribuidos sobre un fondo negro. El entorno controlado, con cubos separados y esparcidos sobre la mesa, facilita la segmentación, y las primeras aproximaciones desarrolladas han mostrado resultados satisfactorios. A continuación, se detallan los pasos que sigue el algoritmo para lograr los objetivos de detección:

Paso 1: Corrección de la distorsión

Antes de calcular las coordenadas, es necesario calibrar la lente de la cámara. Este proceso corrige las distorsiones ópticas y establece la relación entre las coordenadas en píxeles de la imagen capturada y las coordenadas reales del espacio físico.

La calibración de la cámara se realiza una única vez y, en este caso, se utilizó el paquete [camera_calibration](#). Este proceso implica la utilización de patrones conocidos, como tableros de ajedrez o patrones de puntos. Se capturan varias imágenes del patrón desde diferentes ángulos y distancias. En cada imagen, se detectan puntos clave del patrón, como esquinas de los cuadros del tablero o los puntos del patrón, lo que permite relacionar las coordenadas en píxeles con las coordenadas reales.

A partir de este procedimiento, se obtienen dos parámetros fundamentales:

- **Matriz de cámara:** Define las propiedades internas de la cámara, como la distancia focal y el centro óptico.

- **Coeficientes de distorsión:** Corrigen las deformaciones causadas por la lente, tanto radiales como tangenciales.

Con estos parámetros almacenados, el primer paso del algoritmo consiste en realizar una copia del fotograma corrigiendo la distorsión utilizando la matriz de calibración de la cámara y los coeficientes de distorsión. A partir de ahora, se trabajará sobre esta imagen.



Imagen 24. Espacio de trabajo capturado desde la cámara superior.

Paso 2: Detección del marcador ArUco

El segundo paso consiste en localizar el marcador ArUco, que funciona como punto de referencia para calcular con presión la posición y orientación de los cubos en el espacio tridimensional.

Si el marcador no se encuentra en la imagen, la función devuelve un error y detiene el programa. Si el marcador es detectado, se genera una máscara un poco más grande que el marcador para eliminarlo de la imagen. Esto incluye un margen alrededor del marcador para evitar que genere ruido o errores en los pasos posteriores.



Imagen 25. Imagen con la detección del ArUco y su enmascaramiento.

Paso 3: Conversión a escala de grises y espacio HSV

Una vez eliminada la influencia del marcador ArUco, la imagen enmascara de convier a escala de grises y al espacio de color HSV:

- La escala de grises es útil para operaciones de binarización y detección de bordes.
- El espacio HSV permite segmentar los colores de manera robusta, incluso bajo condiciones de iluminación variables.

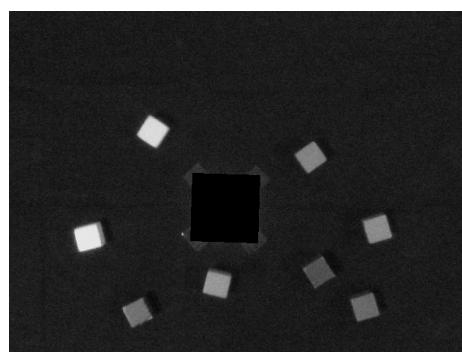


Imagen 26. Imagen en escala de grises.

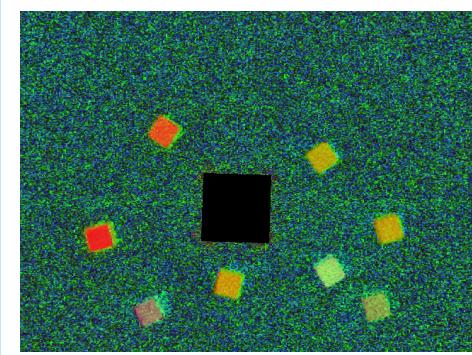


Imagen 27. Imagen en el espacio de HSV..

Paso 4: Binarización utilizando un umbral manual

La imagen en escala de grises se binariza para separar los objetos del fondo. Tras probar diferentes métodos de umbralización, se determinó que el método manual era el más adecuado, dado que el entorno presenta variaciones significativas en la iluminación.

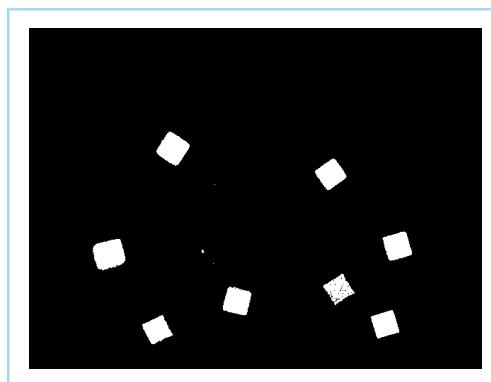


Imagen 28. Imagen con binarizada de forma manual.

Paso 5: Detección de bordes con el filtro de Canny

Con la imagen binarizada, se aplica primero un filtro de Gausiano para desenfocar la imagen y después, el filtro de Canny para detectar bordes. La detección de bordes es un paso crítico, ya que proporciona contornos claros y definidos, fundamentales para localizar y segmentar los cubos.

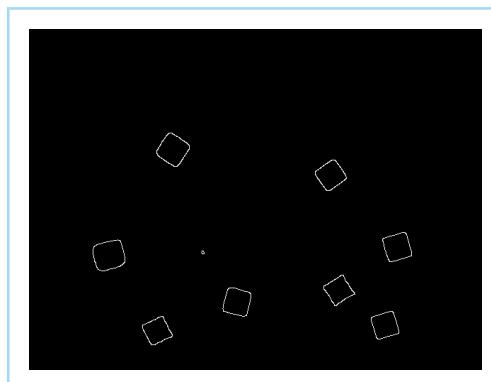


Imagen 29. Detección de bordes con el filtro Canny.

Paso 6: Encontrar contornos

En esta etapa, se identifican los contornos en la imagen procesada. Los contornos resultantes forman la base para determinar si los objetos detectados corresponden a cubos.



Imagen 30. Detección de todos los contornos de la imagen.

Paso 7: Identificación de cubos y asignación de color

Finalmente, cada contorno filtrado se analiza para verificar si corresponde a un cubo. Este análisis incluye un cálculo de la proporción entre el ancho y el alto del contorno, que debe de ser cercana a 1 para asegurar que el objeto es cuadrado.

Cuando un contorno cumple con los criterios, se crea un diccionario que contiene:

- Posición relativa: Calculada utilizando el marcador ArUco como referencia.
- Orientación: Representada en radianes.
- Color: Determinado en el espacio HSV

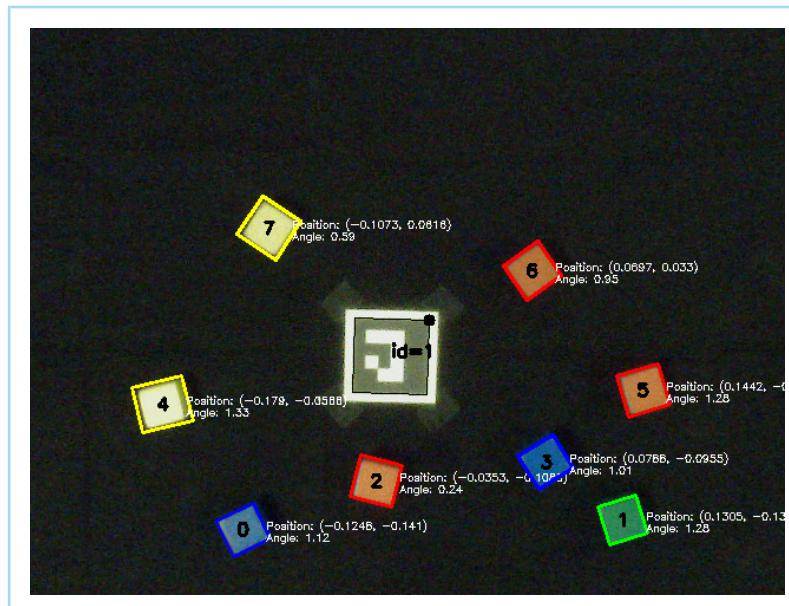


Imagen 31. Resultado final de la detección de cubos.

› Controlador de las Cámaras

Durante el desarrollo de este proyecto, se trabajó con diversas cámaras, lo que hacía necesario identificar el puerto al que cada una estaba conectada y gestionarlas adecuadamente. Este proceso, en ocasiones, requería tiempo adicional. Para simplificar esta tarea, se diseñó una clase llamada `CameraController`, cuyo propósito es gestionar de manera sencilla y estructurada las cámaras conectadas al sistema.

Sus principales funciones son:

- **Detección y gestión de cámaras disponibles:** La clase permite identificar de forma automática qué cámaras están conectadas y listas para ser utilizadas, probando índices consecutivos hasta un número máximo especificado.
- **Captura de fotogramas:** Proporciona una forma directa y eficiente de obtener imágenes (frames) de una cámara específica utilizando su índice.
- **Libera los recursos asociados a las cámaras:** Asegura que las cámaras se cierren correctamente cuando ya no se necesitan.

› Generación de Matriz 3D

Uno de los objetivos finales del proyecto es obtener una representación tridimensional de la figura creada. Para lograrlo, se ha realizado un procesamiento a través de tres cámaras, con el fin de convertir la información contenida en las imágenes en matrices que representan la posición y el color de los cubos detectados en cada una de las vistas. Como resultado, se generan tres matrices: alzado, perfil y planta.

A partir de estas tres matrices, se ha desarrollado un código que recoge la información de las mismas y las correlaciona para definir la posición exacta de cada cubo en el espacio tridimensional. Sin embargo, como se explicará más adelante, aún con tres vistas diferentes, persisten casos de cubos cuya localización sigue siendo desconocida.

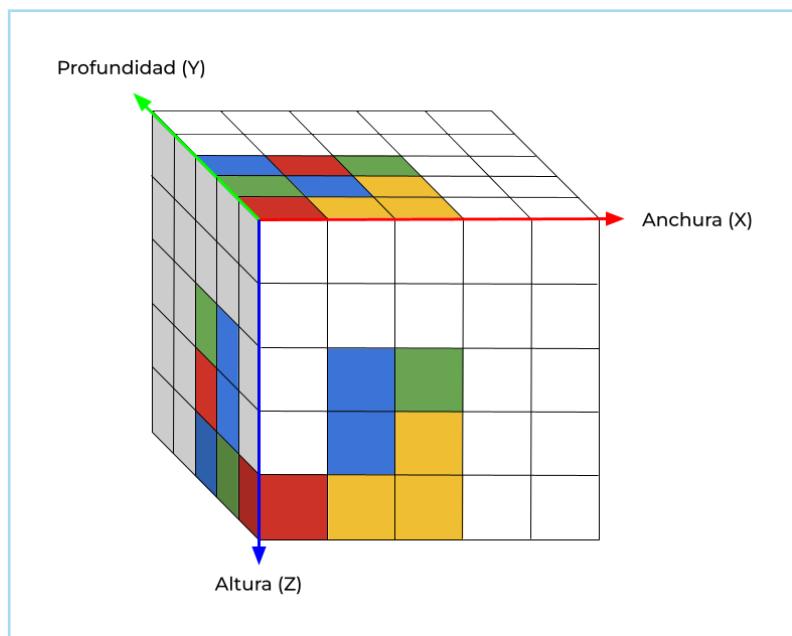


Imagen 32. Representación de las matrices obtenidas sobre Matriz 3D

Antes de comenzar con el procesamiento de la figura, es necesario comprender la relación entre las diferentes matrices obtenidas. Para ello, y teniendo en cuenta la posición de las cámaras, se puede deducir lo siguiente: las dos cámaras laterales comparten el eje Z, que representa la altura de la figura. A su vez, la cámara con perspectiva de alzado comparte el eje X con la planta, siendo este el eje de la anchura de la figura. Finalmente, la cámara con perspectiva de perfil comparte el eje Y con la planta, representando este el eje de la profundidad de la figura.

Esta relación es crucial para entender el procesamiento de la figura final, ya que sirve como guía para definir la posición de cada cubo en estos tres ejes.

Inicialmente, se analizan las matrices para obtener la anchura, profundidad y altura de la figura, y también para verificar que las matrices calculadas son correctas, sin discrepancias en las dimensiones de las matrices de cada perspectiva. Una vez confirmado que las matrices están en buen estado, se recortan para trabajar únicamente con las celdas que corresponden a la figura.

A partir de las matrices recortadas, el primer objetivo es definir la altura de los cubos localizados en la matriz de planta, ya que estos determinan la mayor altura de cada columna de la figura. Para ello, comenzando por el extremo superior derecho de la matriz, se localiza el primer índice donde se encuentra un cubo. Desde ese punto, se procesa toda la columna, comenzando por el cubo superior, teniendo en cuenta el resto de las matrices (alzado y perfil), así como los cubos que ya han sido localizados en las matrices frontal y lateral. Esto permite identificar y definir los cubos ocultos, los cuales se muestran en gris.

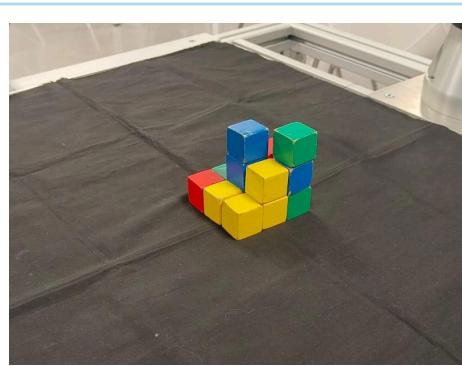


Imagen 33. Imagen vista 1 Figura real

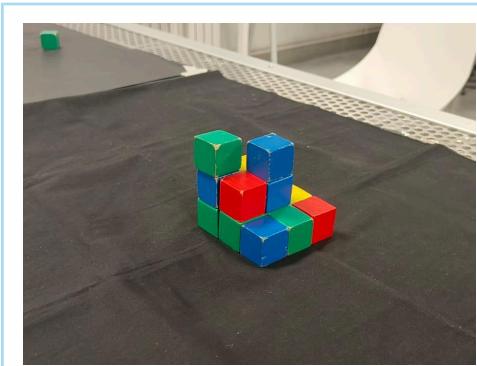


Imagen 34. Imagen vista 2 Figura real

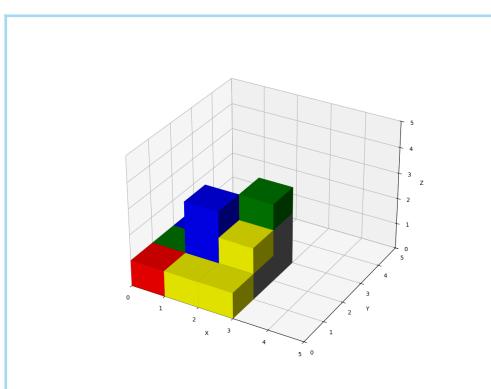


Imagen 35. Imagen vista 1 Figura procesada

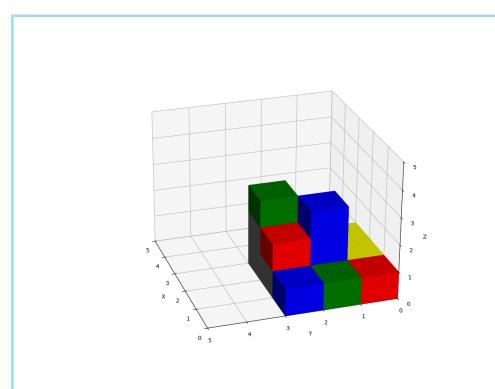


Imagen 36. Imagen vista 2 Figura procesada

6

› Programa Principal

Para la ejecución de todo el proceso desde la captura de las imágenes hasta la visualización de la figura resultante, se ha realizado un programa encargado de llevar a cabo la secuencia de forma automática.

Este programa de python requiere como entrada las 3 imágenes las cuales pueden ser obtenidas a través de las cámaras o en caso de no tener acceso a ellas, se puede hacer uso de las imágenes almacenadas en la carpeta data.

Para facilitar el uso de las diferentes opciones que ofrece el proceso completo, se han definido diferentes variables con las siguientes funcionalidades:

- **use_cam (bool)** - True en caso de querer utilizar las cámaras
- **num_figure (int)** - Número de la figura a procesar (Solo si no hay cámara)
- **mostrar (bool)** - True para mostrar los resultados finales de cada cámara
- **debug (bool)** - True para mostrar los procesos intermedios de cada cámara
- **save_image (bool)** - True en caso de querer guardar las imágenes capturadas

Una vez se ejecuta el código, este se comporta de forma diferente en base a las variables previamente mencionadas, en cualquiera de los casos, al finalizar el proceso, se visualiza la figura tridimensional resultante del proceso para comprobar si el resultado final del proceso es correcto.

› Interfaz

Finalmente, en este apartado presentamos la HMI (Human-Machine Interface) desarrollada para visualizar de manera más precisa, intuitiva y sencilla el funcionamiento del código y los algoritmos descritos en el apartado anterior.

La interfaz gráfica de usuario ha sido desarrollada utilizando **Tkinter**, la biblioteca estándar de Python para la creación de interfaces gráficas. Esta elección nos permitió integrar de forma directa y eficiente las clases previamente desarrolladas en el back-end de la interfaz.

› Funcionalidades Principales

Esta tiene una ventana desde donde se pueden realizar todas las acciones: [Ver Imagen 37]

- **Controles para Alternar entre Captura y Carga de Imágenes**
 - Switch para capturar la figura 3D o cargar fotos: Permite alternar entre capturar la figura en 3D en tiempo real mediante las cámaras o cargar fotos previamente tomadas.
 - Switch para capturar el entorno de trabajo o cargar una foto: Similar al anterior, pero en este caso se refiere a la captura o carga de imágenes del entorno de trabajo.
- **Controles para la Calibración y Actualización de Cámaras**
 - Botón para calibrar el área del cubo: Permite recalibrar el área del cubo en caso de que se haya cambiado la altura de la cámara superior, ajustando la visualización de los cubos.
 - Botón para actualizar los selectores de cámaras: Si se han añadido nuevas cámaras después de ejecutar Tkinter, este botón actualiza los selectores de cámaras disponibles en la interfaz.
- **Área de Visualización del Entorno de Trabajo**
 - Representación 2D del entorno de trabajo: Muestra una vista en 2D del espacio de trabajo, donde se pueden ver las imágenes y alternar entre las cámaras conectadas al ordenador.
 - Botones para procesar o borrar la imagen del workspace: Estos botones permiten procesar la imagen cargada en el entorno de trabajo o borrar la imagen actual, facilitando la actualización del entorno.
- **Representación 3D y Vistas Adicionales**
 - Gráfica de la figura 3D procesada: Muestra la representación 3D de la figura después de ser procesada, permitiendo una visualización más detallada de la figura reconstruida.
 - Imágenes de alzado, perfil y planta: Muestra las vistas de alzado, perfil y planta, permitiendo alternar entre las cámaras conectadas y añadir nuevas imágenes, ofreciendo una visualización más completa del entorno.
- **Mensajes de la terminal**: Muestra los mensajes de la terminal, incluyendo información, advertencias (warnings) y errores, proporcionando retroalimentación sobre el estado del sistema.

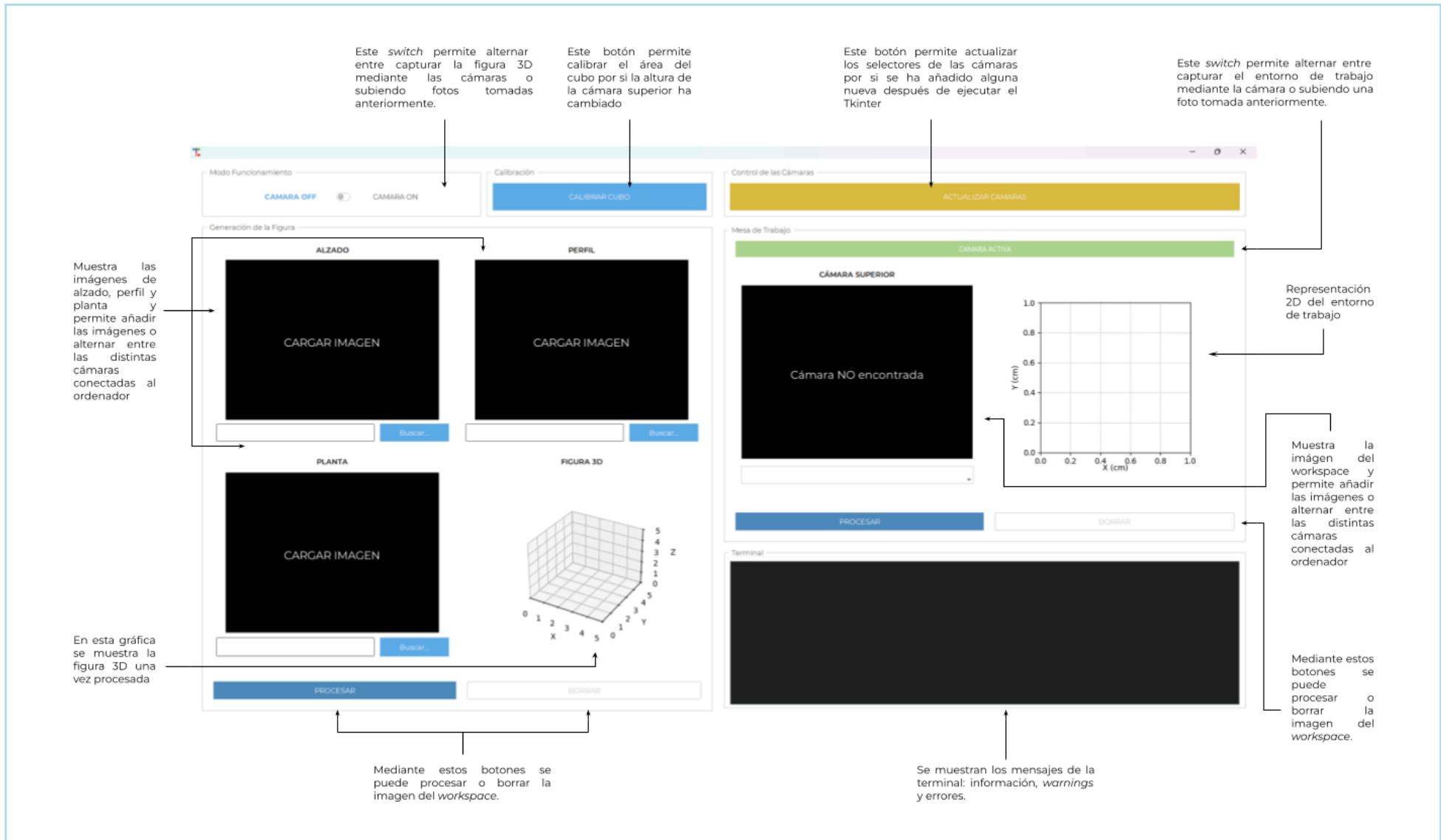


Imagen 37. Imagen HMI y sus componentes

› Repositorio

Todo lo necesario para la ejecución del proyecto está recopilado en un repositorio público de github, donde se incluye el código íntegro, además de las instrucciones para la instalación de las dependencias necesarias.

El enlace a este repositorio es el siguiente:

https://github.com/gonzalo002/Vision_Proyecto_Final/