Pablo Bilbao Renuncio
Álvaro Fernández Gutiérrez

# Proyecto Visión por Ordenador

Link Github: https://github.com/202203829/FinalProjectComputerVision.git

# Introducción

Este proyecto tiene como objetivo implementar un sistema de visión por ordenador utilizando una Raspberry Pi y una cámara. El sistema incluye un bloque de seguridad basado en la identificación de patrones visuales y un segundo bloque enfocado en la persecución de objetos rojos. El trabajo combina conocimientos de calibración, procesamiento de imágenes y detección de objetos, destacando su utilidad en escenarios reales.

# Metodología

Para calibrar la cámara de la Raspberry Pi, se tomaron varias fotos de un tablero de ajedrez desde diferentes ángulos. Utilizando OpenCV, se detectaron y refinaron las esquinas del patrón de ajedrez (7x7 celdas de 30 mm), generando puntos 3D correspondientes. Con estos puntos y las imágenes, se calcularon los parámetros intrínsecos y extrínsecos de la cámara, así como los coeficientes de distorsión, usando la función `cv2.calibrateCamera()`. Finalmente, se evaluó la calibración mediante el error RMS, que mide la discrepancia entre las posiciones proyectadas de los puntos 3D y las esquinas detectadas.

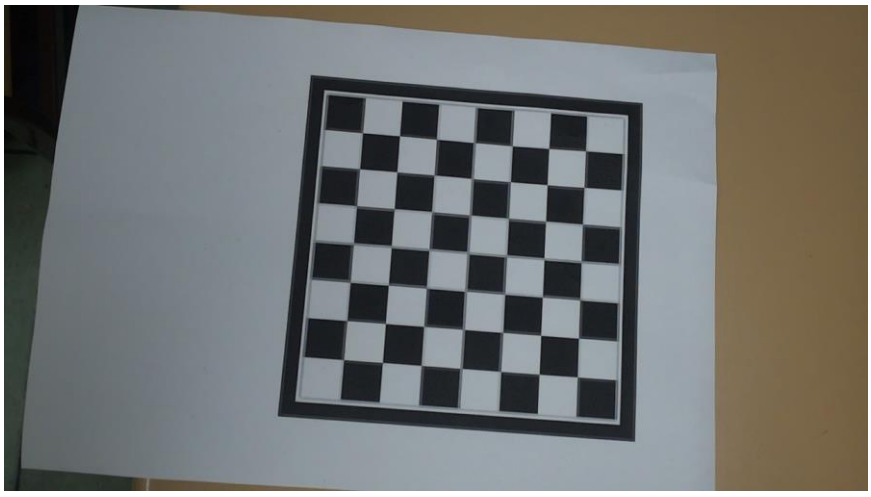

Figura 1. Ejemplo de foto tomada con la raspberry pi

```
Intrinsics:
 [[1.42210784e+03 0.00000000e+00 6.12981844e+02]
 [0.00000000e+00 1.42369571e+03 3.53771116e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Distortion coefficients:
 [[-1.00465843e-01  1.77742800e+00 -5.62540439e-04 -4.44126805e-03
  -8.30940759e+00]]
Root mean squared reprojection error:
 0.9651623152383275
```

Figura 2. Valores Intrínsecos de la cámara

Para la segunda parte se utilizó la cámara de la Raspberry Pi con la biblioteca `picamera2` para capturar imágenes en tiempo real. La cámara fue configurada para tomar fotos con una resolución de 1280x720 en formato RGB. Las imágenes capturadas se procesaron mediante técnicas de visión por computadora para detectar y clasificar figuras geométricas, específicamente un cuadrado, un triángulo y una estrella.

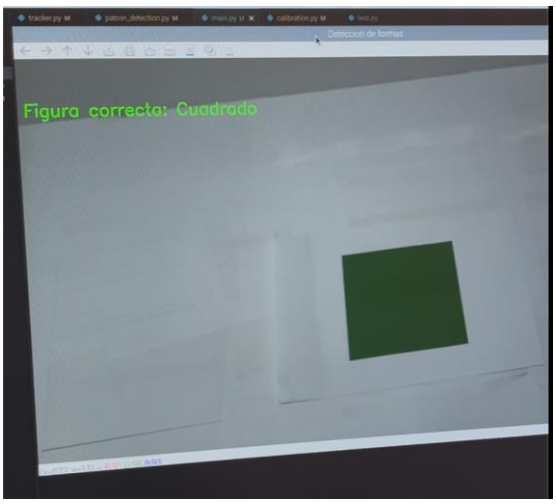


Figura 3. Ejemplo cuadrado

Se implementó el algoritmo de Shi-Tomasi para la detección de esquinas, que permitió identificar los puntos clave de cada figura en la imagen. La imagen original se convertía a escala de grises y luego se utilizaba este algoritmo para localizar las características más prominentes. Una vez detectadas las esquinas, los puntos eran ordenados y se calculaban los ángulos entre ellos, lo que permitía clasificar las figuras según sus características geométricas.

Además, se aplicó un filtro de color en el espacio de color HSV para aislar las áreas rojas de la imagen. Esto fue útil para detectar y procesar figuras que podrían tener ese color específico. El filtro se dividió en dos rangos para abarcar todo el espectro de color rojo, y las máscaras generadas se combinaban para asegurar una cobertura completa.

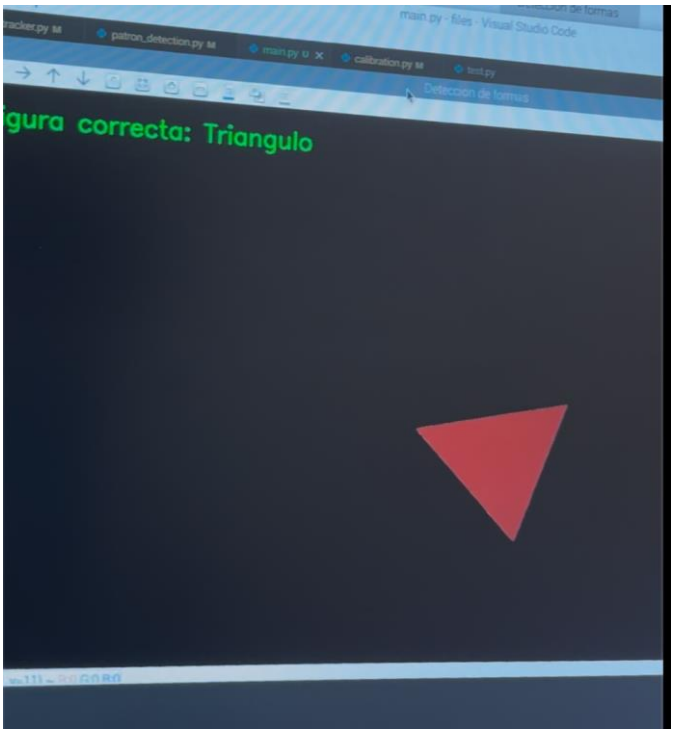


Figura 4. Ejemplo del triángulo con máscara roja

El proceso de verificación consistió en una secuencia de figuras a reconocer, las cuales debían ser confirmadas por el usuario. Al detectar correctamente una figura, el sistema avanzaba a la siguiente; en caso de error, se reiniciaba la secuencia. La interacción con el usuario se realizó mediante el teclado, donde se presionaba la tecla 'f' para confirmar que la figura detectada era la correcta. Si todas las figuras en la secuencia eran verificadas correctamente, el sistema mostraba un mensaje de "Acceso concedido" y terminaba el proceso.


Figura 5 Diagrama de bloques del funcinoamiento

Este enfoque permitió la detección, verificación y clasificación en tiempo real de las figuras geométricas, utilizando una cámara de bajo costo y algoritmos de visión por computadora de código abierto.

Una vez desbloqueado el sistema, se inicia la parte en la que se configura la cámara utilizando `Picamera2`, se ajusta la resolución a 1280x720 y se establece el formato en RGB. La cámara comienza a capturar imágenes para ser procesadas. El usuario puede seleccionar los objetos a seguir utilizando la herramienta de selección de regiones de interés (ROI) de OpenCV. Al seleccionar los objetos, el sistema genera trackers para cada uno, asignando un color basado en el valor del píxel central del objeto, lo que permite distinguir visualmente cada uno.

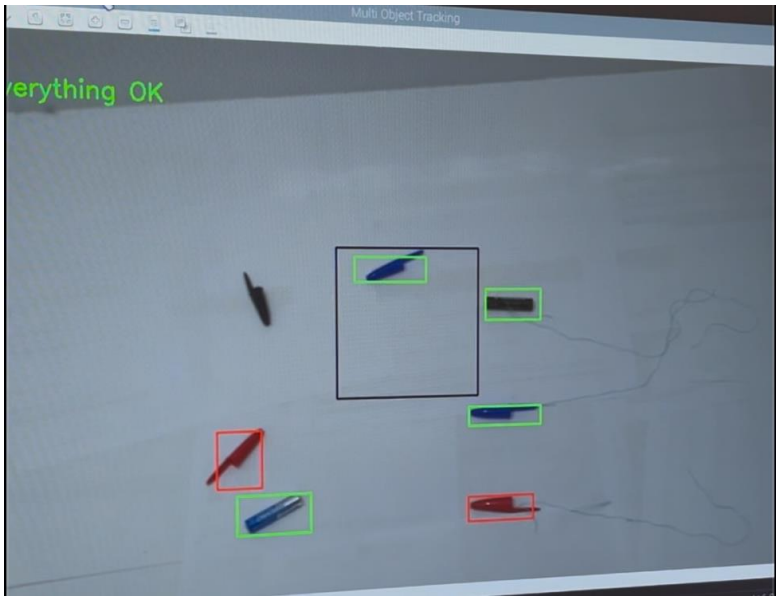


Figura 6. Ejemplo de diferenciación de objetos rojos

A continuación, se dibuja un cuadrado negro en el centro de la imagen para crear un área de referencia. En cada frame capturado, el sistema actualiza el seguimiento de los objetos y verifica si algún objeto rojo entra dentro del cuadrado central. Si un objeto rojo está dentro, se activa una alarma visual mostrando el mensaje "ALARM" en rojo. Si no hay objetos rojos dentro, se muestra el mensaje "Everything OK" en verde. El seguimiento de objetos y la actualización del estado se realizan en tiempo real, y la ventana de la interfaz muestra el resultado mientras el sistema está en funcionamiento.

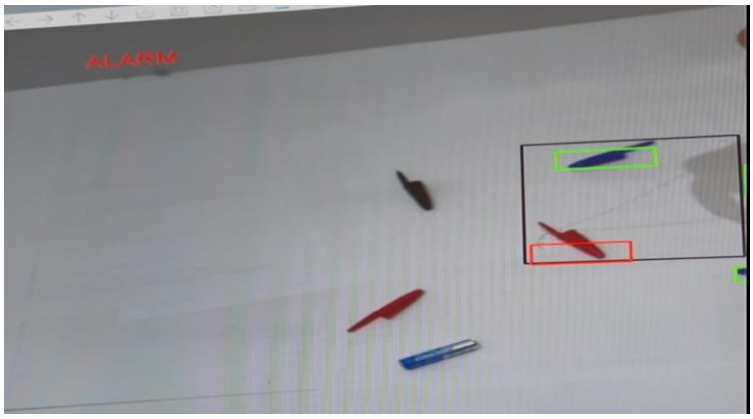


Figura 7. Cuerpo rojo dentro de la zona conflictiva

El proceso continúa hasta que el usuario presiona la tecla 'q' para salir del programa, momento en el cual se liberan los recursos utilizados por la cámara y se cierran las ventanas de OpenCV.

# Resultados

El sistema implementado permite realizar el seguimiento de objetos en tiempo real, diferenciando entre los objetos mediante colores específicos. La funcionalidad principal de la alarma, que se activa cuando un objeto rojo entra dentro de un área delimitada por un cuadrado central, funciona de manera efectiva. Los resultados obtenidos muestran la capacidad del sistema para detectar y seguir múltiples objetos simultáneamente, alertando de manera visual cuando ocurre un evento relevante, lo que demuestra su utilidad en aplicaciones de monitoreo y seguridad.

# Implementaciones futuras

En futuras implementaciones, se podría optimizar el sistema para permitir el seguimiento de objetos en condiciones de luz variable, mejorando el algoritmo de selección de colores. También se podría ampliar la funcionalidad para permitir el seguimiento de más tipos de objetos además de los rojos, utilizando técnicas de aprendizaje automático para la detección avanzada. Otra mejora posible sería integrar un sistema de notificación remota en tiempo real, alertando a los usuarios si la alarma se activa, o incluso integrar el sistema con cámaras de mayor resolución y capacidad de procesamiento.