Sistema de Contraseña con Números

Guzmán Ignacio Pérez Ibarz
Miguel Ángel Vallejo
Grupo B

Visión por Ordenador I
3º Grado en Ingeniería Matemática e Inteligencia Artificial

Índice

INTRODUCCIÓN **3**

METODOLOGÍA **4**

RESULTADOS **8**

FUTUROS DESARROLLOS **9**

CONCLUSION **10**

Introducción

Nuestro proyecto, titulado "Sistema de Contraseña con Números", tiene como objetivo principal implementar un sistema de autenticación basado en visión por computador utilizando una Raspberry Pi y una cámara. Este sistema nos permite detectar números en imágenes capturadas, rastrearlos en tiempo real y gestionar una contraseña de manera dinámica.

En este proyecto abordamos diversas técnicas fundamentales de visión por computador, como la calibración de cámaras, la detección de patrones y el procesamiento de imágenes en tiempo real. Asimismo, empleamos librerías avanzadas como OpenCV y picamera2 para lograr un sistema robusto y escalable.

El desarrollo del sistema lo estructuramos en diferentes módulos de código, incluyendo funciones específicas para la segmentación de color, la extracción de contornos y la comparación de patrones. En este informe detallamos los procedimientos que seguimos para cada etapa del proyecto, los resultados obtenidos y las posibles mejoras futuras.

Metodología

- **Calibración de Cámara:**
  La calibración de la cámara fue un paso crucial para nosotros, ya que nos permitió corregir las distorsiones inherentes a la lente de la cámara. Realizamos este proceso utilizando un patrón de tablero de ajedrez y seguimos estos pasos:

  1. Capturamos imágenes del tablero de ajedrez desde diferentes ángulos.
  2. Detectamos las esquinas en las imágenes utilizando la función cv2.findChessboardCorners().
  3. Refinamos las esquinas detectadas mediante la función cv2.cornerSubPix().
  4. Calculamos los parámetros intrínsecos y extrínsecos de la cámara mediante cv2.calibrateCamera().

  El código que implementamos para la calibración incluye funciones para cargar las imágenes, detectar y refinar las esquinas del tablero, y calcular los parámetros de calibración. Esta calibración nos asegura que las imágenes capturadas por la cámara tengan la menor distorsión posible.
  Resultados Visuales de la correcta calibración:

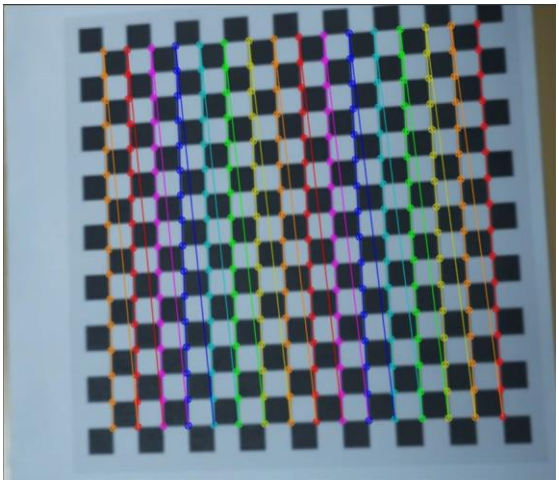


Figura 1: Calibración de la Cámara desde una Orientación.

- **Diagrama de Bloques:**

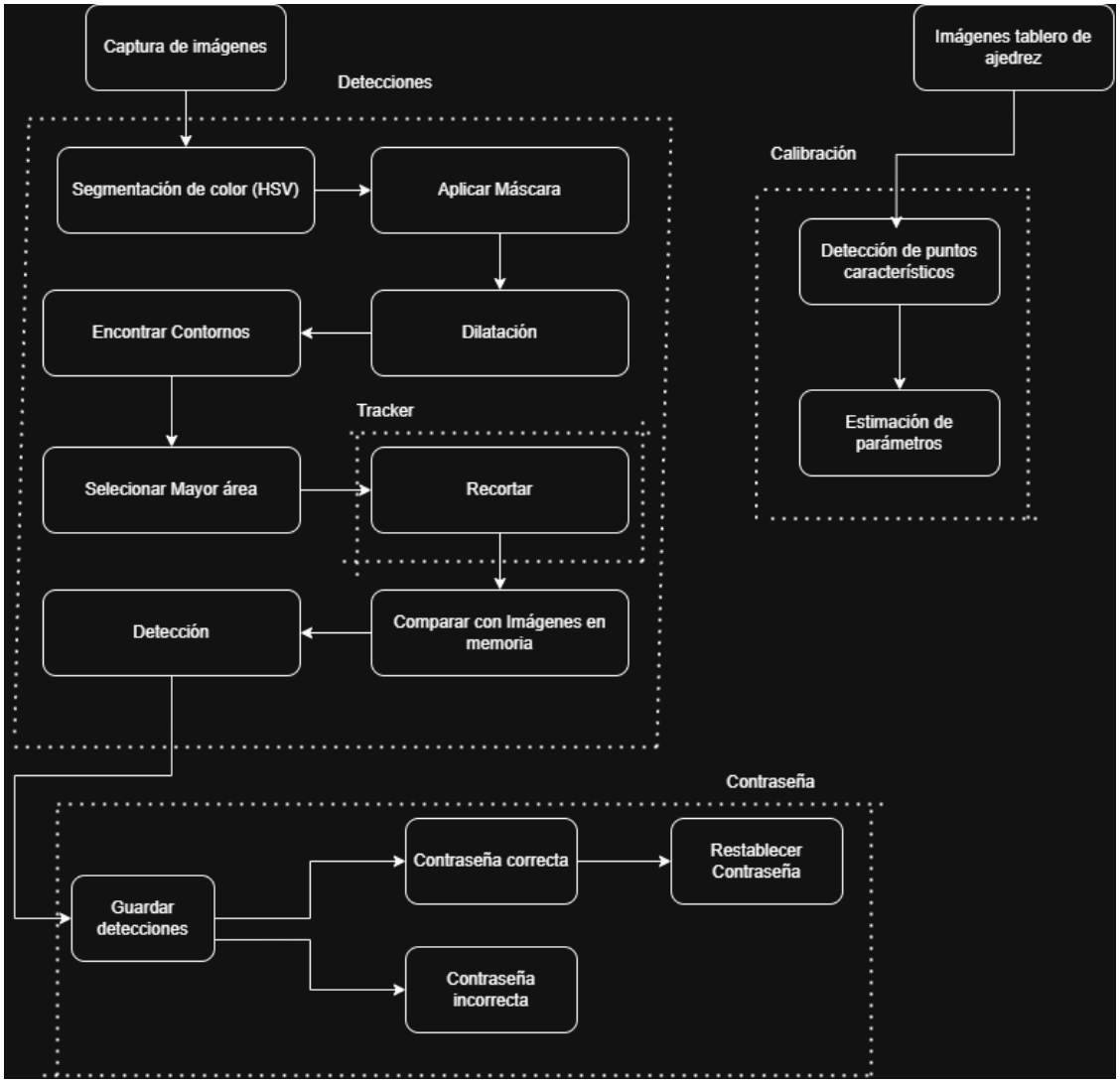
1. **Captura de imágenes**: Utilizamos la librería picamera2 para que la Raspberry Pi capture imágenes en tiempo real.
2. **Segmentación de color (HSV)**: Aplicamos una segmentación basada en el espacio de color HSV para aislar los números verdes del fondo.
   - **Aplicar Máscara**
   - **Dilatación**
3. **Extracción de contornos**: Identificamos los contornos en la imagen segmentada.
   - **Encontrar Contornos**
   - **Seleccionar Mayor área**: Seleccionamos el contorno de mayor área, que corresponde al número.

Metodología

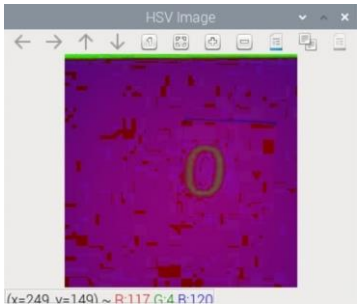
4. **Comparación de patrones**: El número recortado lo comparamos con plantillas almacenadas utilizando la función cv2.matchTemplate(), devolviendo el número detectado.
    - **Recortar**
    - **Comparar con Imágenes en memoria**
5. **Detección**: Devolvemos el número detectado.
6. **Guardar detecciones**
7. **Contraseña**:
    - **Contraseña correcta**: Restablecer Contraseña
    - **Contraseña incorrecta**

    Además, el diagrama incluye una sección de **Calibración** con los siguientes pasos:

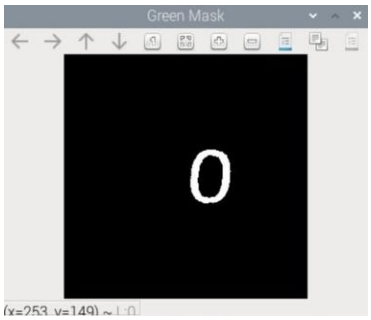
- **Detección de puntos característicos**
- **Estimación de parámetros intrínsecos y extríns**



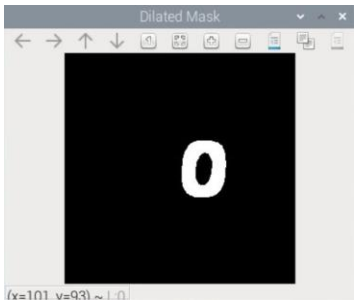
- **Secuencia de Transformación de la Imagen:**
  Para transformar las imágenes capturadas seguimos esta secuencia:
  - Convertimos la imagen capturada al espacio de color HSV para facilitar la segmentación.



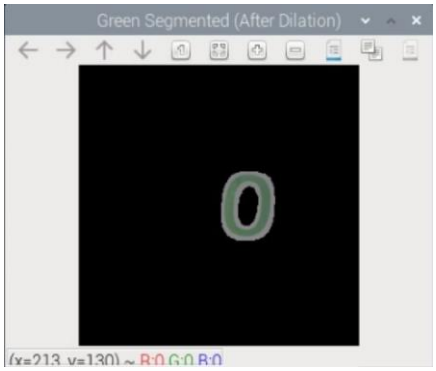
  - Creamos una máscara para aislar los píxeles que se encuentran dentro del rango de color verde definido.



  - Aplicamos una operación de dilatación para unir los píxeles verdes cercanos y mejorar la calidad de la segmentación.

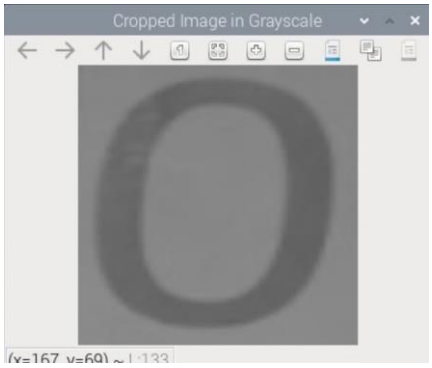


- Identificamos los contornos en la imagen segmentada y seleccionamos el de mayor área.



- Recortamos la imagen utilizando la caja delimitadora del contorno seleccionado.

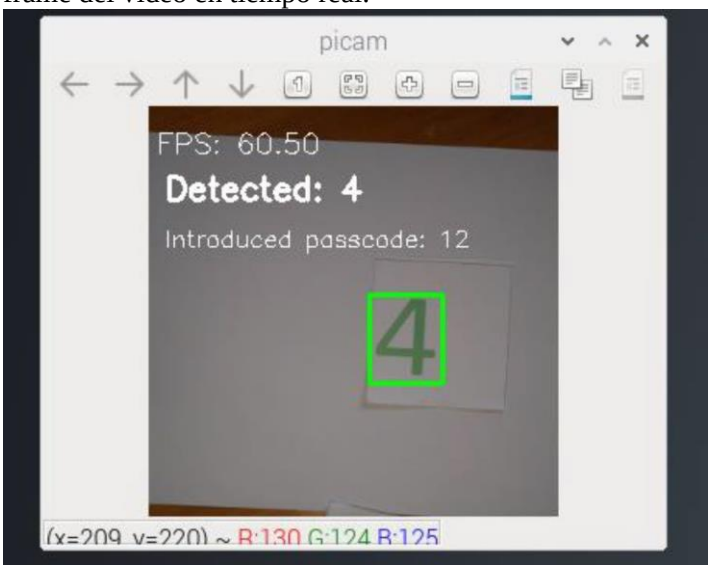
- Convertimos el recorte obtenido a escala de grises para facilitar la comparación con las plantillas.



- **Sistema de Seguridad: Detección de Patrones y Extracción de Información:**

  La detección de números la realizamos mediante la comparación del recorte obtenido con plantillas almacenadas previamente. Este proceso lo llevamos a cabo utilizando la función cv2.matchTemplate(), que calcula el grado de similitud entre la imagen recortada y cada una de las plantillas.

  El número detectado lo almacenamos en una variable global que es utilizada por el sistema de gestión de contraseña. La contraseña introducida se muestra en la pantalla junto con el número detectado y el frame del video en tiempo real.



- **Sistema Propuesto: Tracker, Ampliaciones y Salida de Video:**

  Incluimos un tracker sencillo que sigue la posición del número detectado en la pantalla. Calculamos la posición utilizando la caja delimitadora del contorno de mayor área. Además, mostramos los siguientes elementos en la pantalla:

  - FPS (Frames Per Second): Calculados cada 40 frames para evaluar el rendimiento del sistema.
  - Número detectado: Mostrado en tiempo real.
  - Contraseña introducida: Actualizada cada vez que detectamos un nuevo número.

  El sistema permite ampliar sus funcionalidades mediante:

  1. Autenticación multiusuario: Implementación de contraseñas personalizadas para diferentes usuarios.
  2. Detección de patrones adicionales: Como letras o símbolos.
  3. Integración con sistemas de control de acceso: Como cerraduras electrónicas o sistemas de domótica.

  El resultado del tracker se puede ver perfectamente en el video que hemos llevado a cabo con toda la explicación.

Resultados

Los resultados que obtuvimos en las pruebas realizadas demuestran que el sistema es capaz de detectar números con un alto grado de precisión y gestionar una contraseña de manera eficaz. A continuación, resumimos los resultados clave:

- **Detección de números:** La tasa de detección es superior al 90% en condiciones de iluminación adecuada.
- **Rendimiento:** El sistema mantiene un rendimiento de 50-60 FPS en la Raspberry Pi.
- **Gestión de contraseña:** El sistema permite ingresar y validar contraseñas correctamente.

Realizamos pruebas con diferentes números y plantillas, demostrando la robustez del sistema incluso ante pequeñas variaciones en el tamaño y la posición de los números detectados.

Futuros Desarrollos

Para mejorar y ampliar el sistema propuesto, sugerimos las siguientes líneas de desarrollo:

1. **Mejora del algoritmo de detección:** Utilizar técnicas más avanzadas, como redes neuronales convolucionales (CNN), para mejorar la precisión en la detección de números.

2. **Implementación de un sistema de autenticación multiusuario:** Permitir que diferentes usuarios ingresen contraseñas personalizadas y gestionar un sistema de permisos.

3. **Optimización del rendimiento:** Reducir el tiempo de procesamiento de cada frame para mejorar el rendimiento del sistema en tiempo real.

4. **Integración con sistemas de control de acceso:** Como cerraduras electrónicas o sistemas de domótica, para ofrecer una solución completa de seguridad.

Conclusión

Con este proyecto hemos demostrado que es posible implementar un sistema de autenticación mediante visión por computador utilizando técnicas básicas de procesamiento de imágenes y una plataforma accesible como la Raspberry Pi.

El "Sistema de Contraseña con Números" ha demostrado ser una solución eficaz y escalable que podemos ampliar y adaptar a diversas aplicaciones. Este proyecto representa un paso importante en la exploración de soluciones basadas en visión artificial para la seguridad y la automatización, y abre la puerta a futuras investigaciones en este campo.