COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

Desbloqueo de Seguridad y Automatización de la Activación de Semáforos en Pasos para Peatones

Ulises Díez, Ignacio Felices
Grupo A

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

Contenido

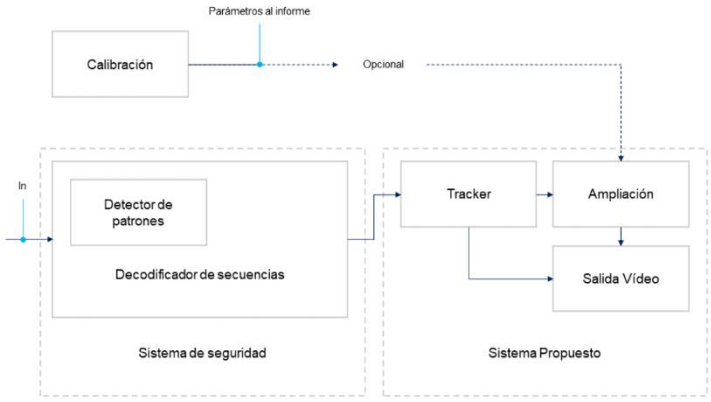
1. Introducción

Este proyecto se enfocará en el desarrollo de un sistema de seguridad para el desbloqueo y activación del programa principal. El sistema incluirá una cámara de videovigilancia que supervisará un paso de peatones. Cuando un peatón entre en la escena, el sistema realizará un seguimiento en tiempo real mediante un **tracker**. Esto permitirá activar automáticamente el semáforo en rojo para detener el tráfico vehicular mientras el peatón cruza. Una vez que el peatón abandone el área supervisada, se reanudará la circulación normal de los vehículos.

La idea surge como respuesta a una problemática común en la vida cotidiana: una sociedad inmersa en el estrés, las prisas y las distracciones de un mundo saturado de estímulos, donde con frecuencia los peatones cruzan carreteras sin mirar o sin respetar los semáforos.

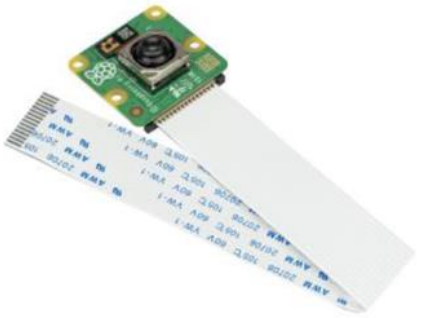
Este sistema busca automatizar la activación de los semáforos, ofreciendo una solución eficiente que podría salvar miles de vidas diariamente.

Para garantizar la seguridad y control del sistema, se ha implementado un mecanismo de desbloqueo que requiere un patrón específico. En este caso, el desbloqueo se llevará a cabo mediante la identificación de un cubo de Rubik y la configuración precisa de colores en sus caras.



2. Material Utilizado

Para la realización de este proyecto, se emplearon diversos recursos de hardware y software. La ejecución del programa se llevó a cabo utilizando una **Raspberry Pi**, equipada con el módulo de cámara **"Camera Module 3 WIDE"** para la captura de video en tiempo real. El desarrollo del código se realizó en el lenguaje **Python**, utilizando el entorno de desarrollo integrado **Visual Studio Code**. El código fue transferido a la Raspberry Pi mediante el software **MobaXterm**, utilizando una conexión SSH y Escritorio Remoto para facilitar su implementación y prueba. Además, se utilizó un repositorio remoto en **GitHub** como herramienta de colaboración para compartir el código entre los integrantes del equipo y para la entrega del proyecto. Este enfoque aseguró un flujo de trabajo eficiente y coordinado.



Durante el desarrollo en Python, se emplearon las siguientes librerías y módulos, seleccionados por sus funcionalidades específicas: typing, numpy, imageio, cv2, copy, glob, os, matplotlib.pyplot, skimage, time.

Estas librerías proporcionaron soporte para el procesamiento de imágenes, análisis de datos, manejo de archivos, y otras operaciones esenciales en la implementación del sistema.

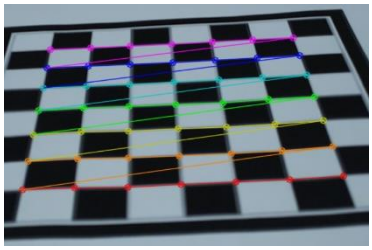
3. Calibración Cámara

Antes de comenzar con el desarrollo del programa, se necesitan unos pasos previos. Entre ellos, el más importante a destacar es la calibración de la cámara a usar, ya que es esencial corregir distorsiones ópticas y obtener parámetros intrínsecos clave, como la matriz de intrínsecos y los coeficientes de distorsión. Estos parámetros nos permiten transformar las imágenes capturadas en representaciones libres de distorsión y más precisas. Además, el error cuadrático medio reproyectado (1.176) refleja la calidad de la calibración.

Este proceso es crucial para garantizar mediciones consistentes y un seguimiento confiable, optimizando el rendimiento de los algoritmos de visión artificial en tu proyecto.

```
Intrinsics:
 [[2.17089681e+03 0.00000000e+00 1.52082974e+02]
 [0.00000000e+00 4.21235834e+03 1.46422261e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Distortion coefficients:
 [[-0.5317102   3.03395891 -0.03161913 -0.03142607 -6.53863547]]
Root mean squared reprojection error:
 1.1762713604858739
```

La calibración de la cámara se llevó a cabo con un total de 16 imágenes de tablero de ajedrez, identificando los puntos de cuadrícula del tablero, para posteriormente obtener los parámetros de calibración.

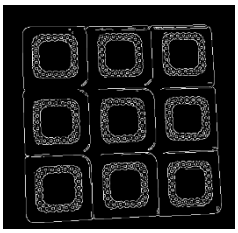


4. Sistema de Seguridad

El sistema de seguridad y desbloqueo del programa se basa en la identificación de un cubo de Rubik y la verificación de un patrón específico de colores en sus caras. Este cubo no es convencional, ya que su disposición de colores es distinta, como se muestra en la imagen adjunta. El proceso para implementar este sistema consta de los siguientes pasos:

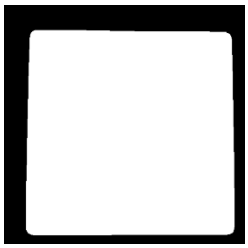
1. Identificación de Contorno y Bordes

Para determinar si el objeto mostrado es un cubo de Rubik, se transforma la imagen a escala de grises y se aplica un filtro de Canny, seleccionado por su robustez frente a otros métodos probados en análisis previos.



2. Operadores Morfológicos

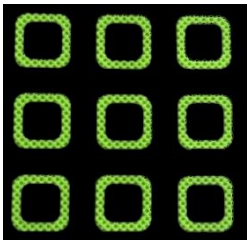
Como se ha podido apreciar, el cubo de Rubik utilizado no es un cubo convencional, por lo que para que este pudiese ser identificado como uno, se aplicó un operador morfológico de dilatación para eliminar los huecos dentro de los componentes del cubo de Rubik.



3. Identificación de Forma

Una vez procesada la imagen, se implementó una función para calcular el contorno y el área del objeto detectado. Para evitar falsos positivos, se estableció un tamaño mínimo de área, además de analizar la relación entre las dimensiones del objeto. Si el ancho y el alto presentan una relación cercana a 1:1 (con un rango de tolerancia de 0.9 a 1.1), se considera que el objeto es un cubo.

```
-----------------------------------
Imagen 6: Rubik's Face Detected: False
-----------------------------------
Imagen 7: Rubik's Face Detected: False
-----------------------------------
Imagen 8: Rubik's Face Detected: True
```



4. Máscara de colores

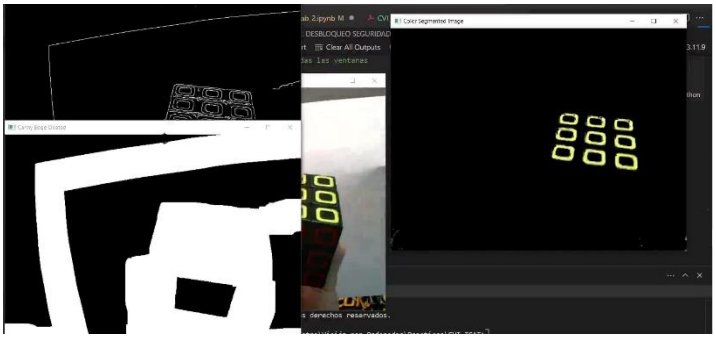
Tras confirmar que el objeto es un cubo de Rubik, se aplican máscaras de color a la imagen para aislar colores específicos (azul, amarillo y verde) en sus rangos desde tonalidades claras hasta oscuras.

5. Verificación de Colores

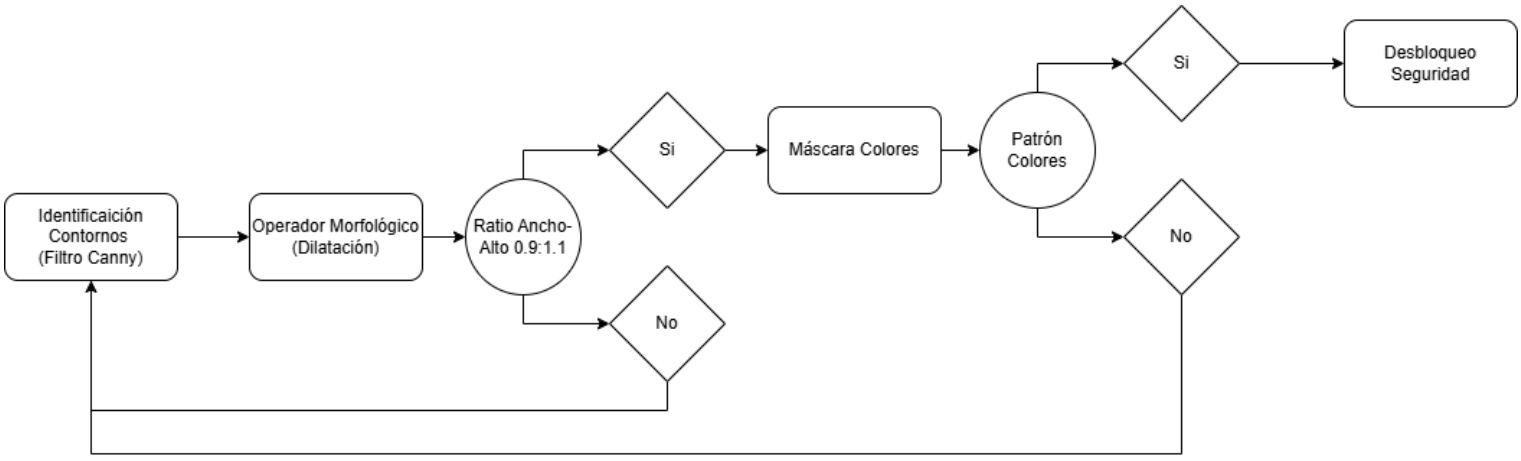
Finalmente, se desarrolló una función para verificar la presencia de colores detectados mediante las máscaras. Se cuentan los píxeles no nulos de la imagen y, para evitar errores causados por condiciones de iluminación o similitudes entre colores (como confundir naranja con rojo), se aplica un umbral que permite cierta tolerancia sin comprometer la precisión de la detección.

```
----------------------------------
Imagen 14: Rubik's Face Detected: True
Imagen 14: Comprobando colores...
Color rojo detectado: False
Color amarillo detectado: True
Color verde detectado: False
Color detectado: True
----------------------------------
Imagen 15: Rubik's Face Detected: True
Imagen 15: Comprobando colores...
Color rojo detectado: False
Color amarillo detectado: False
Color verde detectado: False
Color detectado: False
```

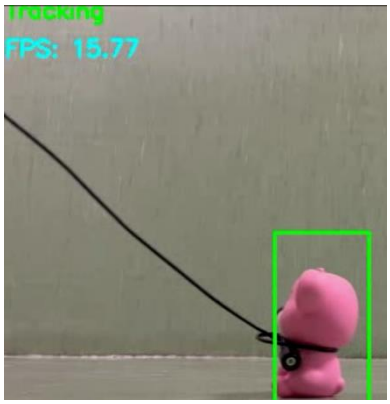
Este proceso garantiza la identificación precisa del cubo y la validación de los colores requeridos para desbloquear el sistema.
Para probar el funcionamiento en tiempo real, se dio el paso de probar esto a través de muchas fotografías tomadas de una carpeta del ordenador, mezcladas entre sí, y aplicar todo el proceso para identificar el patrón entre ellas, a comprobar el funcionamiento de todo el sistema con captura de video a tiempo real.



A continuación se puede observar el diseño de flujo seguido para la elaboración del sistema de seguridad, teniendo en cuenta las diferentes opciones ante las que se puede encontrar la cámara.

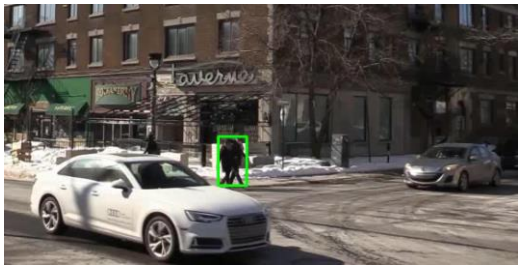


5. Seguimiento de Peatones y Activación Semáforo



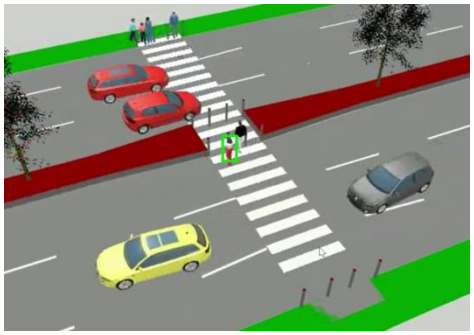
Al introducir la secuencia de patrones correcta, el semáforo activara su sistema de seguimiento al peatón. Para ello comenzara en un estado de rojo para el peatón. Una vez detecte un peatón mediante un bounding box, le va a seguir el movimiento. Una vez salga del área peatonal y se haya dejado de detectar, el semáforo para el peatón se cambiará a rojo y seguidamente los coches podrán circular.

Para este apartado, se ha de destacar que se han utilizado una variedad de trackers de la librería de CV2, y a continuación se va a explicar cómo se ha elegido el más optimo. Para ello, se ha acudido a royalty-free videos, donde se han buscado cinco videos afines a la realización de nuestro proyecto. Hemos descargado los cinco videos, uno virtual y otro con agentes reales, estos se encuentran en la carpeta: *'Videos'*, y se han usado los siguientes trackers:



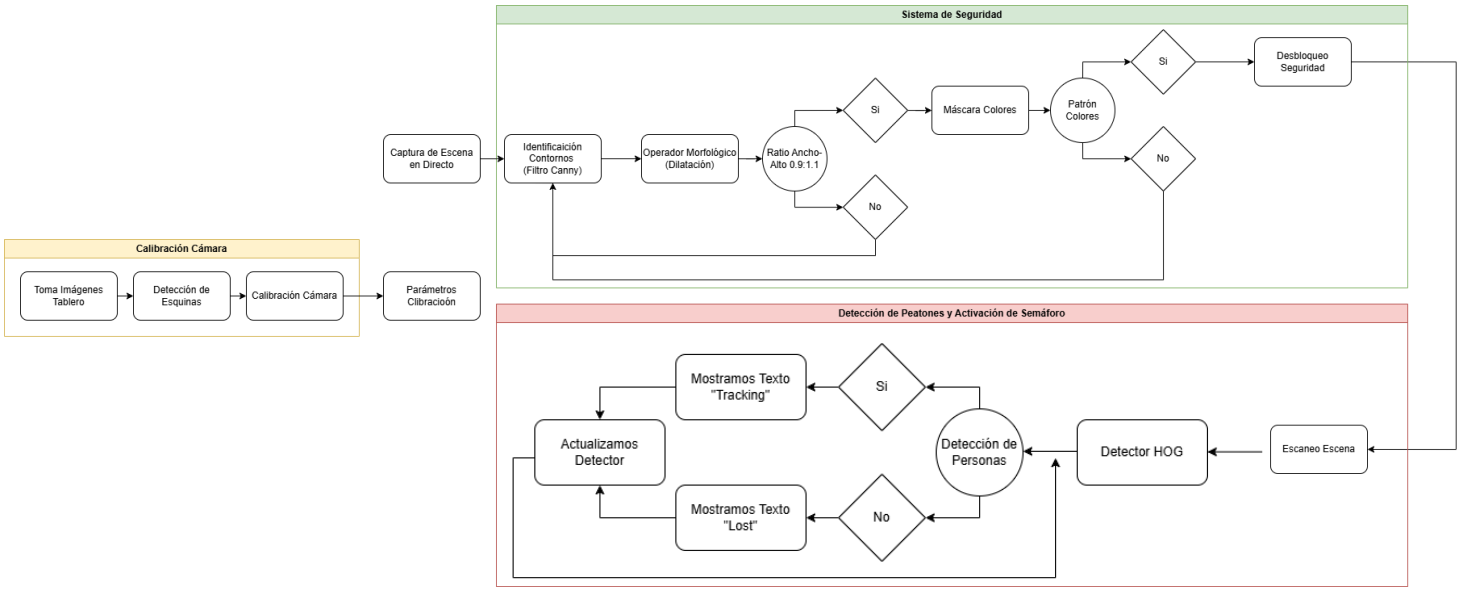
- KCF (Kernelized Correlation Filter)
- CSRT (Channel and Spatial Reliability Tracker)
- MIL (Multiple Instance Learning)
- HOG (Histogram of Oriented Gradients)

Primero se probó con el tracker KCF, donde en primer lugar no funcionaba el tracker. Seguidamente, se probamó con CSRT donde no detectaba de forma correcta al peatón. Por otra parte, MIL no era capaz de monitorear al peatón durante todo el trayecto.



Finalmente se decidió utilizar el tracker HOG (Histogram of Oriented Gradients). Este tracker convierte una imagen en características basadas en los gradientes de intensidad de los píxeles. Es comúnmente utilizada para para detectar y seguir personas en videos. Es el que mejor funcionó por su robustez frente a cambios de iluminación, esto es debido a que se centra en gradientes y no en valores absolutos de color. La monitorización es satisfactoria ya que una vez identificado al peatón en un fotograma, el algoritmo puede seguirlo en los fotogramas posteriores comparando características similares.

A continuación, se puede observar el diagrama de flujo final y completo del proyecto desarrollado:



6. Resultados

Finalmente, tras haber analizado y probado individualmente las diferentes partes del proyecto utilizando la webcam del ordenador y videos descargados de internet o grabados con el móvil, procedimos a adaptar y estructurar el código para emplearlo con la cámara de la Raspberry Pi. Para ello, se creó un archivo en Python que integra dos funciones principales: la primera, correspondiente al sistema de seguridad; y la segunda, que activa el seguimiento de objetos una vez superado el sistema de seguridad. Ambas funciones procesan video y capturan frames en tiempo real utilizando el módulo de cámara de la Raspberry Pi.

Para facilitar la transferencia de archivos y el trabajo remoto con la Raspberry Pi, se utilizó la aplicación MobaXTerm, estableciendo una conexión SSH para mover los archivos necesarios y una conexión RDP para acceder al escritorio remoto del dispositivo. Este flujo de trabajo y la configuración empleada se muestran detalladamente en el video entregado, complementado por una presentación en PowerPoint que explica el desarrollo y las pruebas realizadas. El video evidencia el correcto funcionamiento del sistema propuesto y desarrollado, demostrando su eficacia en tiempo real.

7. Futuros Desarrollos

Para futuros desarrollos, resulta crucial realizar pruebas más robustas para evaluar el sistema en condiciones reales, mejorando la precisión del tracker y su capacidad de adaptarse a entornos dinámicos. Esto incluye pruebas con personas en movimiento en un paso de cebra real, donde las condiciones de iluminación, el ruido visual y las variaciones en la velocidad de los peatones puedan representar desafíos adicionales.

Además, una posible extensión del proyecto sería incorporar técnicas de aprendizaje profundo, como el uso de redes neuronales convolucionales (CNN) para la detección y seguimiento más preciso de peatones. También se podría explorar la integración de sensores adicionales, como LiDAR o cámaras estéreo, para mejorar la detección en 3D y obtener información más precisa sobre la distancia y el movimiento en tiempo real. Estas mejoras no solo aumentarían la eficacia del sistema, sino que también abrirían la puerta a su implementación en sistemas avanzados de asistencia a la conducción (ADAS) o en aplicaciones de seguridad urbana.