Proyecto Final: Minijuego de ping-pong

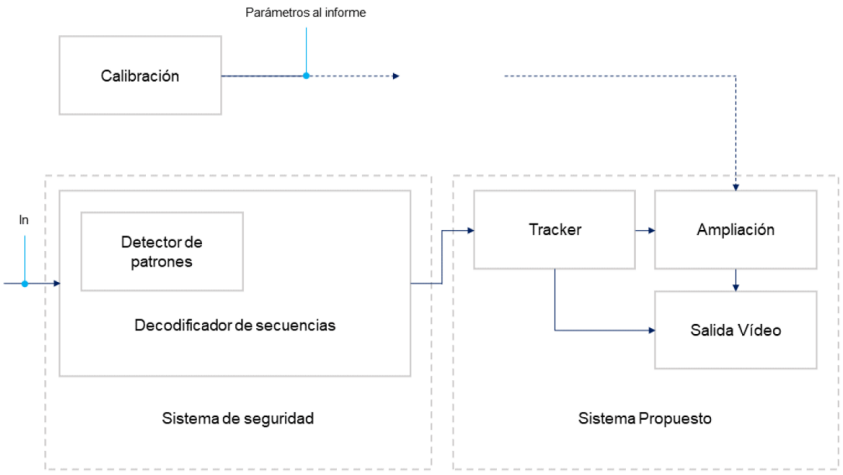
Proyecto desarrollado por Isabel V. Morell Maudes y Sofía Negueruela Avellaneda

Enlace al repositorio de github: https://github.com/isabelMorell/VOI_ProyectoFinal

1. Introducción

El proyecto consiste principalmente en el seguimiento de una partida de ping-pong donde, tras desbloquear un sistema de seguridad, el programa llevará y mostrará la puntuación durante la partida y el ganador junto a la puntuación final al concluir. Además del *tracker*, el proyecto incluye la calibración de la cámara y la detección de patrones con el decodificador de secuencias que servirá como el sistema de seguridad del minijuego.

El diagrama de bloques del sistema completo es el siguiente:



Para el desarrollo del proyecto hemos utilizado una Raspberry Pi y una cámara que hemos colocado con LEGOS como se puede observar en la demo demo.mp4.

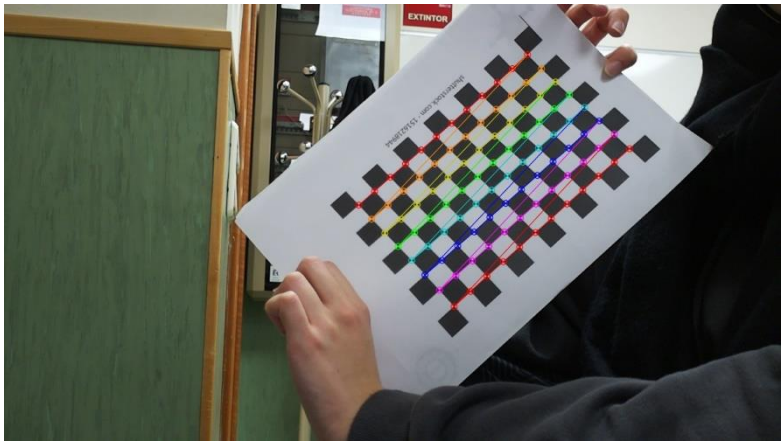
2. Metodología

2.1. Calibración de la cámara

Para calibrar la cámara, hemos utilizado como base el código de la práctica de calibración, en la que usábamos las funciones de cv2 que calibran una cámara usando un tablero de ajedrez.

Con una cuadrícula en blanco y negro impresa simulando dicho tablero, hacemos aproximadamente 30 fotos con la cámara de la Raspberry-Pi: en un intervalo de 30

segundos, guardamos un *frame* por segundo. Después, se hizo una selección previa de aquellas imágenes que usamos para la calibración, buscando cambios en la iluminación y orientación del tablero mientras descartamos las que no mostraban el tablero entero o quedaron borrosas. Finalmente, calibramos la cámara y guardamos las imágenes con los bordes señalados como en la siguiente imagen.



2.2. Sistema de seguridad

Los patrones que va a identificar nuestro sistema de seguridad son unas tarjetas cuadradas de diferentes colores ordenados que se muestran uno a uno. La secuencia a introducir, es decir, la contraseña, está escrito en un archivo secreto (*password.txt*). Si el propietario del sistema de seguridad quisiera modificar la contraseña, podría hacerlo sin problemas: puede variar su longitud y repetir los colores tanto como quiera. En los archivos actuales, la contraseña está compuesta por cuatro colores distintos.

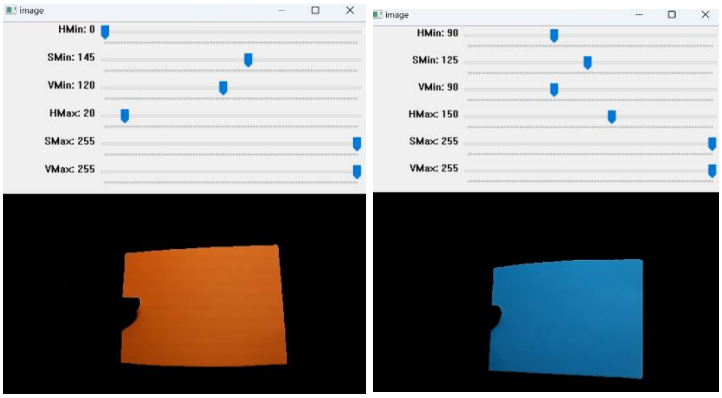
2.3. Secuencia de transformación de la imagen

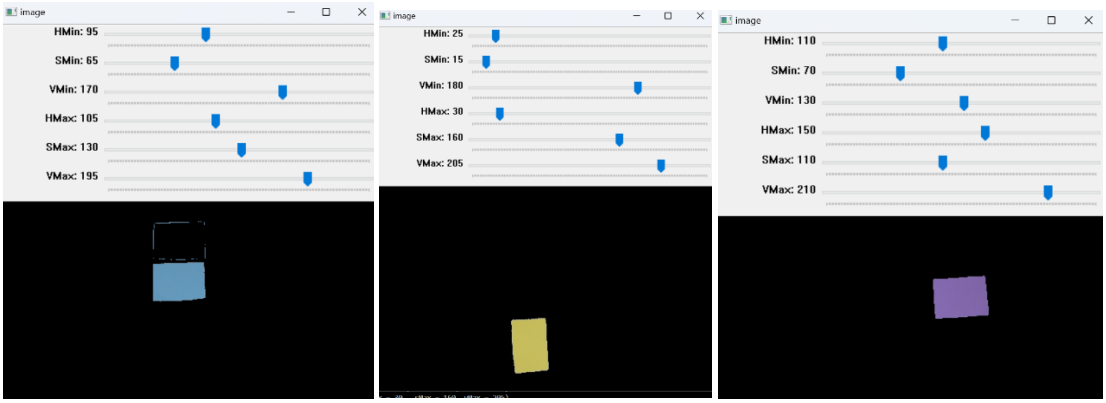
Para analizar el input del usuario, utilizaremos las técnicas de segmentación de colores vistas a lo largo del curso. Las instrucciones para mostrar los patrones son claras: mostrar uno a uno cada patrón que compone la contraseña. Si ha pasado demasiado tiempo (más de 90 segundos) y el sistema no ha detectado un color válido, el sistema se bloquea y el usuario debe volver a empezar el proceso.

Para comprobar que se está mostrando la tarjeta del color correcto en cada caso, el programa captura un *frame* y evalúa cuántos píxeles hay de ese color. Si hay suficientes (más de 15000 aproximadamente), considera que se ha mostrado la tarjeta correcta y espera a recibir el próximo color.

La cámara recibe el frame en el espacio de colores BGR (Blue, Green, Red), pero el proceso de segmentación de color trabaja con imágenes en HSV (Hue, Saturation, Value), por lo que el programa gestiona también el cambio del espacio de color. Por otro lado, el proceso de segmentación requiere cierto trabajo previo. Las tarjetas de

colores que identifica ahora mismo el programa no son reemplazables por otras de colores similares, pues se ha calibrado la imagen y definido los valores mínimos y máximos del espacio HSV que permiten pasar cada color. Si los colores del input cambian, es necesario modificar también estos valores. Para ello está el fichero *color_code.py*, el cual lee la imagen que le indiques y te ayuda a calibrar a mano dichos valores. Aquí se muestran algunos ejemplos. Al principio hicimos el proceso con fotos de cada color (es decir, una foto por tarjeta). Rápidamente nos dimos cuenta de que los valores de HSV mínimos y máximos no eran correctos, pues detectaba el rosa cuando se mostraba el morado, y lo mismo ocurría entre los verdes y los amarillos. Las imágenes de después, sin embargo, filtran una sola imagen que contiene las nueve tarjetas, para asegurarnos de que el problema se soluciona.



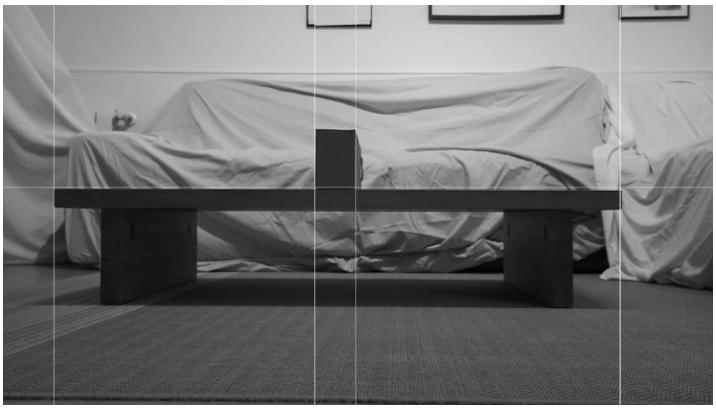


Con estos valores mínimos y máximos del color deseado se crea una máscara de la imagen, es decir, una matriz de 1 y 0 del mismo tamaño que la imagen. Cada píxel vale 1 si se ha detectado el color, y 0 en caso contrario. Por ello, sumando el valor de los píxeles de la máscara obtenemos la cantidad de color detectada en la imagen. Esto se repite para cada color hasta haber introducido la contraseña completa, desactivando el sistema de seguridad. Una vez hecho esto, el usuario podrá observar la funcionalidad del sistema que se propone a continuación.

2.4. Sistema propuesto

El sistema propuesto ha consistido en el seguimiento de una partida de ping-pong en tiempo real. Para ello, el programa principal llamado *tracker.py* comienza calculando el ratio de frames por segundo de la cámara (fps) y creando la carpeta y el video donde se va a guardar la demostración. A continuación, se ejecuta el sistema de seguridad mencionado anteriormente. Si se introduce la contraseña correcta en el tiempo establecido, hay un pequeño tiempo de margen para recolocar la camara y que se posicione el segundo jugador antes de que comience el juego; si no, el programa finaliza y se muestra en el vídeo un mensaje de que la contraseña introducida no es correcta.

Una vez desbloqueado el sistema, comienza la ejecución principal del programa. Primero se calculan los campos de los jugadores mediante la identificación de los laterales de la mesa, su superficie y los laterales de la red. Esto lo hemos conseguido aplicando una segmentación a color y el algoritmo de detección de bordes de Canny a cada componente. A este último algoritmo le hemos añadido una umbralización para asegurarnos de que nos quedábamos con los bordes que buscábamos. Una vez están los campos bien definidos como se muestra en la figura y el jugador 1, que es el de la izquierda por conveniencia, muestra que está preparado para sacar comienza el juego. Esto lo hace mostrando que la pelota está más allá de su campo.



El programa sigue la pelota mediante la aplicación de una segmentación a color y un detector de movimiento de mezcla de gaussianas (Gaussian Mixture). Para localizar los botes, el programa almacena el estado de la pelota en el frame anterior y lo compara en el actual de forma que si estaba bajando la pelota y ahora está subiendo, significa que la pelota acaba de botar.

Cuando se detecta un bote, se chequea en que campo ha sido y si implica que se ha marcado un punto. Si se ha marcado un punto, se actualizarán las diferentes variables del programa y se mostrará en el vídeo la puntuación actualizada. De nuevo, hasta que el jugador al que le toque sacar no esté preparado con la pelota más allá de su campo, la partida no continuará.

Una vez haya ganado un jugador, se mostrará un mensaje en el vídeo indicando el ganador y la puntuación final.

Tanto los puntos necesarios para ganar como todos los límites de los colores necesarios para la segmentación a color se encuentran en el fichero constantes.py.

1. Resultados

En el repositorio de GitHub se encuentra un vídeo que muestra y explica todo el sistema del proyecto que hemos realizado. Este vídeo se llama demo.mp4 y el vídeo en tiempo real que se graba a la vez con la cámara de la Raspberry Pi es el que se encuentra en la carpeta output con el nombre output_tiempo_real_2jugadores.avi. En esta carpeta también se encuentra el vídeo output_tiempo_real_1jugador.avi que muestra más casos a parte de los que se dan en la partida de 2 jugadores. Como se puede observar en ambos vídeos, el sistema de seguridad se ejecuta correcta y eficientemente; sin embargo, el sistema no es completamente capaz de detectar todos los puntos y hay algunos que se asignan al jugador incorrecto. Creemos que se debe a que la cámara no es capaz de seguir a la pelota cuando avanza muy rápido.

En la demo se menciona que íbamos a intentar arreglar el problema mencionado utilizando un sistema de seguimiento que no fuera en tiempo real; sin embargo, esto al final no lo hemos podido realizar.

2. Futuros desarrollos

Para completar el proyecto, se podría hacer que el tracker detectará bien todos los puntos y que se añadieran diferentes efectos al marcar un punto, como confetti o unos aplausos.

En el sistema de seguridad se podrían complementar los diferentes colores con diferentes formas como diferentes polígonos o figuras como corazones o estrellas.