

VisionCalc

Proyecto Final

Visión por Ordenador I

Gloria Vidal-Quadras Fuertes
Fernanda Marcos Gámez



Gloria Vidal-Quadras Fuertes
Fernanda Marcos Gámez

Curso 2024/25

VisionCalc

Proyecto Final

Visión por Ordenador I

Profesor: Email
Erik Velasco evelasco@icai.comillas.edu



Contents

1	Introducción	1
2	Metodología	2
	Calibración de cámara	2
	Diagrama de bloques	3
	Secuencia de transformación de la imagen	3
	Seguimiento de trayectorias	4
	Detección de patrones y extracción de información	4
3	Resultados	6
4	Futuros desarrollos	8

1

Introducción

A lo largo de este cuatrimestre, se han realizado cuatro sesiones prácticas (calibración, procesamiento de imágenes, extracción de características y detección de objetos), a partir de las cuales se pueden realizar una variedad de proyectos de Visión por Ordenador. En este proyecto, se van a aplicar los conocimientos para desarrollar una calculadora con un sistema de seguridad.

En primer lugar, se cuenta con un sistema de seguridad basado en la detección de patrones. El usuario deberá desbloquearlo dibujando una secuencia de polígonos con un bolígrafo de color en el aire. El programa detectará el recorrido seguido y el dibujo obtenido. Si se ha dibujado el patrón adecuado, el sistema se desbloqueará permitiendo el acceso a las funcionalidades del programa.

El segundo bloque del proyecto consiste en el reconocimiento y cálculo de operaciones matemáticas. Se dispone de dos modos: juego y cálculo. En el modo juego, el sistema propone una operación, cuyo resultado tendrá que ser indicado por el usuario con el bolígrafo de color, al igual que en la primera parte. El sistema reconoce el número y comprueba si la respuesta del usuario es correcta. En el modo cálculo, el usuario es el que indica la operación, que seguidamente el sistema reconoce y resuelve.

2

Metodología

Para realizar el proyecto, se cuenta con una RaspberryPi 4 y una cámara. Todo el código ha sido desarrollado en python, utilizando las librerías cv2, utils (proporcionada en las prácticas) y pytesseract. Se debe instalar Tesseract OCR para poder ejecutar el proyecto.

Calibración de cámara

La cámara de la RaspberryPi tiene una lente plana, por lo que no se espera distorsión en la imagen. En cualquier caso, se realiza la calibración de la cámara. Para ello, se ha utilizado la detección de esquinas de un tablero de ajedrez con los métodos de cv2, como se estudio en la primera práctica de laboratorio. En el proceso, se han empleado 15 fotos, tomadas con la RaspBerry y accediendo a la cámara mediante PiCamera20). A continuación se adjuntan un par de ejemplos de los resultados.

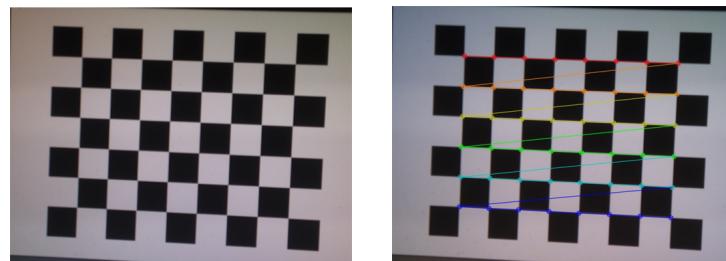


Figure 2.1: Calibración: Detección de esquinas. Ejemplo 1.

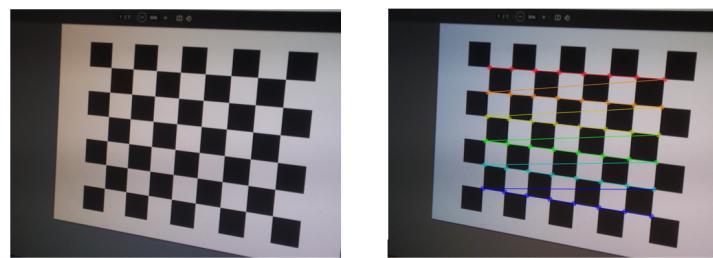


Figure 2.2: Calibración: Detección de esquinas. Ejemplo 2.

La **matriz de intrínsecos** obtenida es la siguiente:

$$\begin{bmatrix} 1.91018691 \times 10^3 & 0.0 & 3.82604611 \times 10^2 \\ 0.0 & 1.9110511 \times 10^3 & 2.96148067 \times 10^2 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Como se esperaba, los **coeficientes de distorsión** no son significativos:

$$[-2.02069785 \times 10^{-1} \quad 3.93681354 \quad -1.13638832 \times 10^{-3} \quad -4.62949156 \times 10^{-3} \quad -2.09041141]$$

Por lo tanto, es posible desarrollar el sistema sin necesidad de calibrar la imagen. Además, esto supondría aplicar los parámetros a cada frame, lo que ralentizaría el sistema, por lo que hemos optado por realizar la calibración offline y no aplicarla a los siguientes pasos del proyecto.

Por último, el **error de reproyección (RMSE)** es: 0.2148575

Diagrama de bloques

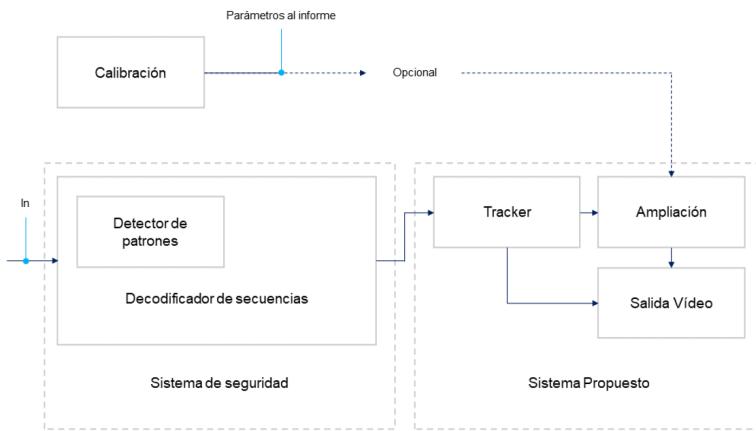


Figure 2.3: Diagrama de bloques del sistema.

El sistema de seguridad y la calculadora utilizan los mismos métodos y herramientas: transformación de la imagen obtenida, seguimiento de la trayectoria del bolígrafo, detección de patrones y extracción de información. A continuación, se detalla cada uno de estos pasos.

Secuencia de transformación de la imagen

Para poder extraer la información del vídeo en tiempo real, se debe transformar cada frame. En el caso de este proyecto, el objetivo es detectar la posición del bolígrafo en cada momento, y con ello, su trayectoria. Las transformaciones aplicadas son las siguientes, en el orden en que se mencionan:

- **Transformación de BGR a HSV:** En el espacio HSV el tono (Hue) y la intensidad (Value) son independientes el uno del otro, lo que permite detectar los colores (en este caso aquel del bolígrafo) bajo diferentes condiciones de iluminación, proporcionando robustez al programa.
- **Máscara binaria:** Para detectar el bolígrafo mediante su color. En este caso, se utiliza un bolígrafo de color azul, por lo que se fijan los extremos inferior y superior de la máscara como (100, 150, 50) y (140, 255, 255) respectivamente.
- **Erosión:** Sobre el resultado obtenido se aplica un filtro de erosión para reducir el ruido que pueda haber sido detectado como parte del bolígrafo.
- **Dilatación:** Se suavizan los bordes de la región detectada y se uniformiza la misma. Además, tiene el objetivo de recuperar áreas que se hayan podido excesivamente en la erosión.

- **Contorno:** Utilizando la función `cv2.findContours` de OpenCV se obtiene el contorno del bolígrafo.
- **Centro:** Finalmente, se descartan los contornos con áreas muy pequeñas para evitar incluir ruido irrelevante y se obtiene el centro de la región que representa el bolígrafo. Este punto se marcará por pantalla con un pequeño círculo e indicará el trazo del marcador.

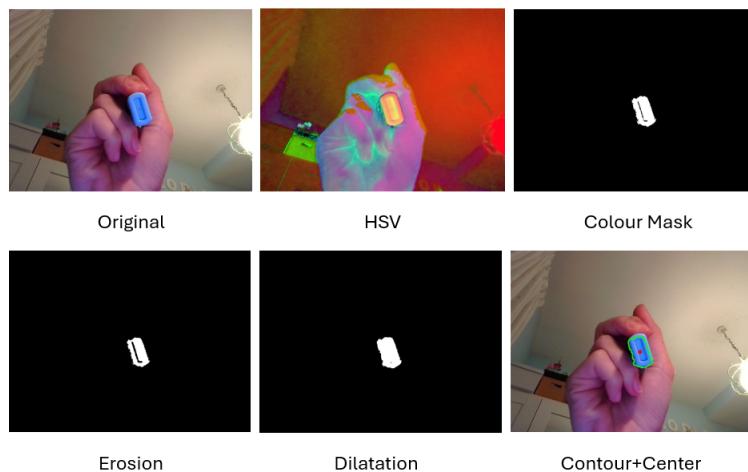


Figure 2.4: Secuencia de transformación de la imagen.

Seguimiento de trayectorias

Como se ha mencionado anteriormente, el centro del bolígrafo se detecta para cada frame en tiempo real. Para dibujar su trayectoria, primero se comprueba si el sistema está en modo "drawing". Este método puede activarse y desactivarse con la tecla "d" del ordenador. En caso de que el modo de dibujo esté activo, se añade el punto actual a la matriz "drawing_canvas", de modo que se actualiza la trayectoria, resultando en un trazo sobre este lienzo. Mediante `cv2.addWeighted()` se combinan los fotogramas y el lienzo permitiendo así mostrar el dibujo sobre la imagen.

Detección de patrones y extracción de información

Sistema de Seguridad

Antes de pasar a la detección de patrones, se debe preparar la imagen. Los pasos del pre-procesamiento son:

- **Transformación a escala de grises:** Para el problema que nos ocupa no son tan relevantes los colores de los píxeles como su intensidad o el contraste entre áreas. Por lo tanto, transformar la imagen a escala de grises nos permite reducir la complejidad computacional sin ninguna pérdida de información.
- **Desenfoque Gaussiano:** Reducción de ruido y suavización de la imagen, lo que facilita la posterior segmentación.
- **Binarización:** Se establece el umbral en 60. De esta forma, los píxeles por debajo del umbral se convierten en blanco, y aquellos de valor superior en negro, resultando en un texto blanco sobre un fondo negro.

De la misma forma que con la detección del bolígrafo, se hallan los contornos de las figuras (`cv2.findContours`). A continuación, se filtran las regiones pequeñas para evitar detecciones de ruido, y se emplea la función `approxPolyDP()` de OpenCV para identificar qué polígono representan los contornos.

Programa Propuesto

De nuevo, comenzamos por pre-procesar la imagen con los siguientes pasos:

- **Transformación a escala de grises**
- **Binarización:** Se fija el umbral a 20.

Una vez se tiene la imagen binaria, se realiza la identificación de la expresión matemática. Para ello se utiliza el motor de reconocimiento óptico de caracteres (OCR) **Tesseract**. En concreto, se utiliza la librería **pytesseract**, diseñada para la aplicación de OCR en Python. Esta librería permite limitar los caracteres reconocidos mediante el parámetro "*tessedit_char_whitelist*". En este caso, se ha especificado este parámetro para que el reconocimiento se centre en números y operadores matemáticos.

Cabe destacar que esta librería está principalmente orientada al reconocimiento de textos, por lo que su rendimiento en aplicaciones con expresión matemáticas es mejorable. Para compensar esto se probaron algunas transformaciones de reducción de ruido, sin embargo, las imágenes obtenidas tras la binarización ya son bastante claras, por lo que las transformaciones adicionales no demostraron presentar ningún beneficio.

A partir de este punto el procedimiento cambia para los distintos modos del programa. En el modo de juego, tras reconocer la respuesta escrita por el usuario, el texto obtenido se limpia de espacios ya que son innecesarios en el contexto numérico. Luego, este se compara con la solución correcta de la operación. En caso de haber acertado, se genera automáticamente una nueva operación.

Por otro lado, para la opción de calculadora basta con limpiar el texto reconocido por pytesseract y después evaluarlo para obtener el resultado. Tanto la expresión reconocida como su solución se mostrarán por pantalla.

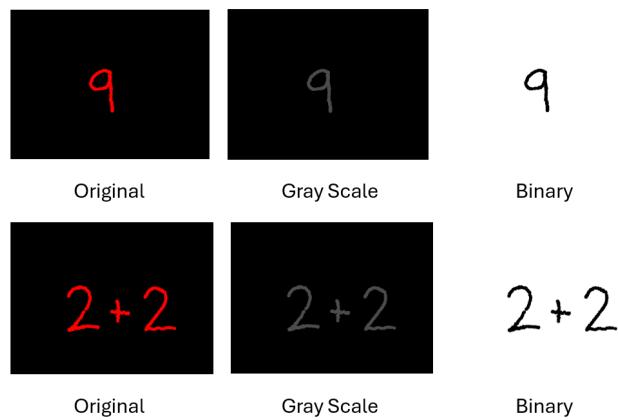


Figure 2.5: Secuencia de transformación de la imagen.

3

Resultados

Una vez completado todo lo anterior, se lleva a cabo la implementación del programa.

Sistema de Seguridad

Como se ha mencionado anteriormente, al iniciar el programa se debe desbloquear un sistema de seguridad. La contraseña del mismo está compuesta de hasta cuatro figuras. El usuario deberá dibujar las figuras correctas en un orden específico. Para ello, puede activar y desactivar el modo dibujo del marcador mediante la tecla "d" (*draw*). Además, si desea limpiar el lienzo, puede hacerlo pulsando "c" (*clear*). Tras dibujar cada figura el usuario lo añadirá a su secuencia con la tecla "a" (*add*). Al hacerlo se añadirá la última figura detectada a la lista "secuencia introducida" que se muestra en la parte superior de la pantalla. Una vez haya finalizado debe introducir y comprobar la secuencia introducida pulsando "r" (*resolve*).

Sistema Propuesto

Al desbloquear el sistema, se accede a un menú donde el usuario puede escoger con el bolígrafo el modo de la aplicación que desea utilizar. En ambos programas, al igual que en la sección anterior, la tecla "d" activa y desactiva el modo dibujo, la "c" limpiará el lienzo de la pantalla y al pulsar "r" se reconocerá el patrón dibujado, aplicando la lógica correspondiente en cada caso.

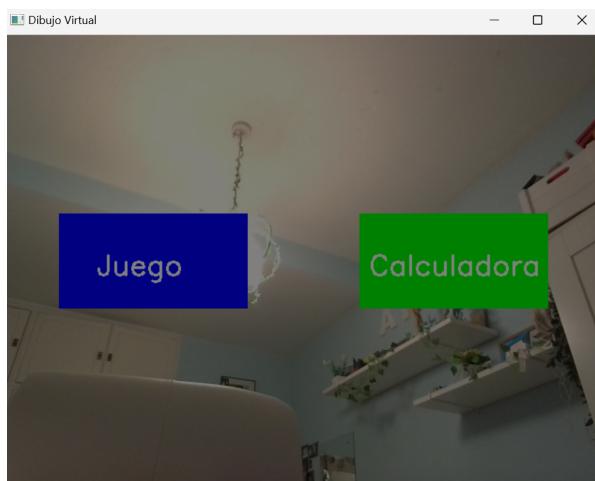


Figure 3.1: Menú del programa propuesto.

En el modo juego, se muestra la operación a resolver en la parte superior de la pantalla. En la esquina superior izquierda se cuenta con un botón "Menú" seleccionable con el marcador, para regresar al menú inicial. Además, en caso de ser necesario, se puede obtener una expresión diferente pulsando la tecla "u" (*update*). Por último, al resolver, si la respuesta es correcta se generará una nueva operación que sustituirá a la anterior.

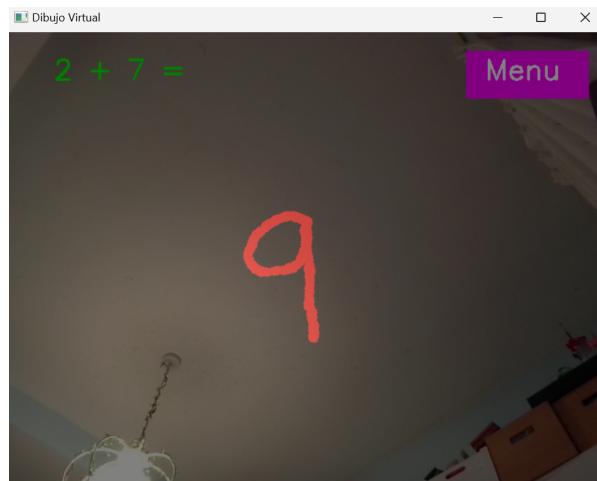


Figure 3.2: Ejemplo interfaz del modo juego.

Igualmente, la calculadora tiene un botón para regresar al menú principal. En esta ocasión, al pulsar "r" se escribirá en la pantalla la expresión reconocida y su solución.

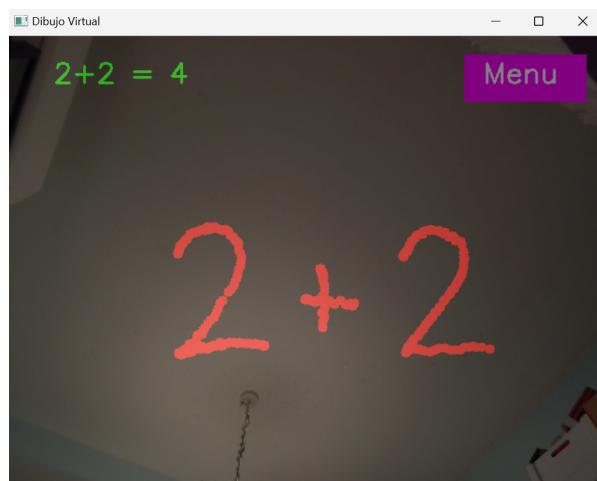


Figure 3.3: Ejemplo interfaz del modo calculadora.

4

Futuros desarrollos

Limitaciones del sistema

A pesar de que se reconoce correctamente la trayectoria del bolígrafo, no siempre se detecta el símbolo correspondiente, lo que lleva a fallos en la resolución de operaciones o comprobación de resultados. Esto ocurre por varios motivos.

En primer lugar, es posible que el color del bolígrafo no sea tan distintivo como debería, lo que lleva a una trayectoria poco limpia, que dificulta el posterior reconocimiento del patrón. Se ha tomado la decisión de implementar el color azul por ser más accesible, y así facilitar la corrección del proyecto. En segundo lugar, el software utilizado no es el más preciso, se ha utilizado por su sencillez y facilidad de implementación. Estos factores provocan que el sistema tenga errores independientemente del usuario.

Propuestas

Para solucionar estos fallos en futuros desarrollos existen varias propuestas. La primera consiste en utilizar un bolígrafo de un color más distintivo. De este modo, se podrá establecer un rango más pequeño de color del elemento que dibuja, evitando confusiones con otros elementos del fondo, y así consiguiendo resultados más limpios. Por último, es necesario mejorar el algoritmo de detección de números y símbolos. Una posible solución consiste en entrenar una red neuronal u otro algoritmo de Machine Learning para la detección de los símbolos deseados. De nuevo, no ha sido implementado en este proyecto por exceder los límites de la asignatura, y ser un mecanismo considerablemente más complejo.