



COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI **ICADE** **CIHS**

Práctica final: Detección y Seguimiento

Ana Rodríguez Jaraba y Laura Gómez Peña
Grupo B

Visión por ordenador
3º Grado en Ingeniería Matemática e Inteligencia Artificial

Contenido

INTRODUCCIÓN	3
METODOLOGÍA	3
CALIBRACIÓN DE LA CÁMARA.....	3
Diagrama de bloques del sistema.....	4
Secuencia de transformación de la imagen.	4
Sistema de seguridad: Detección de patrones y Extracción de información.	5
Sistema propuesto: tracker, ampliaciones y salida de vídeo.	6
RESULTADOS	7
FUTUROS DESARROLLOS.....	7

Link al repositorio: https://github.com/lauragomezpena/ProyectoFinal_Vision.git

Introducción

Este proyecto se enfoca en implementar un sistema capaz de identificar combinaciones de gestos de la mano, específicamente puño cerrado y puño abierto, y utilizar esta información para controlar la aparición y el movimiento de un emoji en la pantalla. Estas combinaciones han sido determinadas previamente.

El objetivo principal es desarrollar una herramienta que no solo detecte de manera precisa las combinaciones de gestos, sino que también permita mover un emoji dinámicamente según el desplazamiento de la mano. Este sistema combina técnicas de detección de patrones, seguimiento de objetos y programación interactiva, lo que lo convierte en una solución innovadora y educativa en el campo de la visión por ordenador.

La motivación para este proyecto radica en la creciente demanda de sistemas interactivos que utilicen gestos naturales para la interacción. Este enfoque no solo mejora la experiencia del usuario, sino que también promueve una interfaz más intuitiva y accesible. A través de este proyecto, se busca explorar las capacidades de las herramientas modernas de visión por ordenador, como bibliotecas de código abierto, y demostrar su potencial para aplicaciones prácticas y recreativas.

Metodología

Calibración de la cámara.

El primer paso en la realización de nuestro proyecto fue calibrar la cámara de la Raspberry Pi proporcionada, para esto hicimos una serie de fotos hechas con la cámara de la Raspberry con las cuales realizaremos la calibración. Estas fotografías son imágenes de un tablero de ajedrez mostrado en un iPad, el cual actuó como patrón de calibración debido a su disposición de cuadros uniformes que facilitan la identificación de puntos clave en la imagen. Se eligió utilizar 10 fotos para la calibración, ya que según la primera práctica realizada para esta asignatura es el número óptimo de imágenes para utilizar en la calibración.

Con la calibración se obtienen los siguientes resultados:

Intrinsics:

```
[[1093.81  0.00 207.91]
 [ 0.00 1096.71 248.58]
 [ 0.00  0.00  1.00]]
```

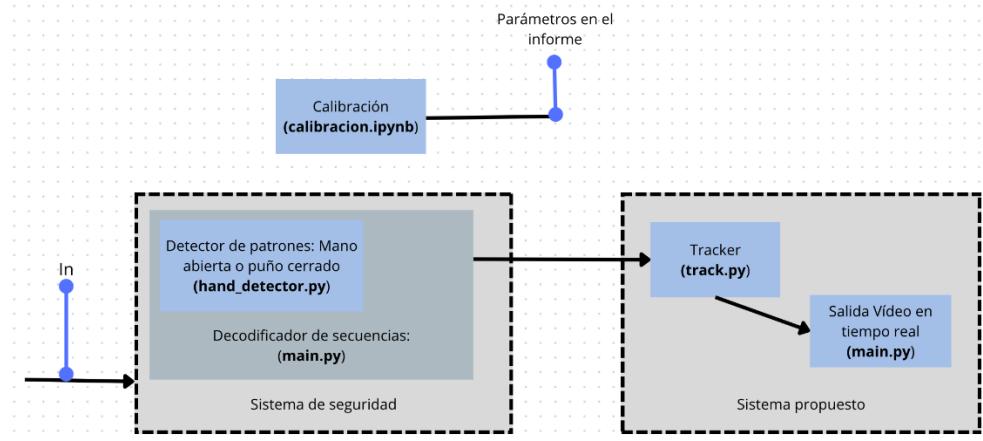
Distortion coefficients:

```
[-0.0290 1.4915 -0.0018 0.0010 -16.3624]]
```

Root mean squared reprojection error: 0.1777

Diagrama de bloques del sistema.

El diagrama de bloques del sistema, teniendo en cuenta nuestros módulos desarrollados es el siguiente:



Secuencia de transformación de la imagen.

Antes de la detección de patrones posterior (detectar puño cerrado o mano abierta), hemos realizado la siguiente secuencia de transformaciones a cada frame capturado, esto nos facilita la posterior detección de patrones.

- 1- Detección de ubicación de la mano (roi) con librería MediaPipe y reducción de espacio de interés
 - Utilizamos la librería MediaPipe para detectar la mano en cada frame. MediaPipe proporciona un modelo robusto de detección de manos, que identifica la ubicación y los puntos clave de la mano. Esto nos permite delimitar la región de interés (ROI) que corresponde a la mano, eliminando el resto de la imagen y reduciendo el espacio de procesamiento a la zona relevante. Este paso es esencial para mejorar la precisión de la detección de patrones al concentrarse solo en la mano, además tras obtener esta región, la reducimos aún más ya que MediaPipe por defecto incluye la muñeca en la región de interés.
- 2- Escala de grises
 - Después de identificar la mano en el frame, convertimos la imagen a escala de grises. Esta transformación simplifica la imagen, eliminando la información de color, lo cual es útil para la posterior detección de patrones que no necesita la información del color.
- 3- Filtro gaussiano
 - Por último, se aplica un filtro gaussiano al frame, que ayuda a suavizar la imagen y facilita la extracción de los patrones.

Sistema de seguridad: Detección de patrones y Extracción de información.

Tras llevar a cabo la transformación de cada frame, proseguimos con la detección de patrones para, más tarde, extraer la información de interés.

1- Transformada de Hough

Para determinar si la mano se encontraba abierta o cerrada, empleamos la transformada de Hough, la cual se encarga de buscar formas circulares en el área de interés (ROI).

El algoritmo busca círculos aplicando una serie de cálculos matemáticos sobre los bordes detectados en la imagen. En este caso, la presencia de círculos significará que el puño está cerrado y la ausencia de estos, lo contrario, que la mano está abierta.

Definimos los círculos con la siguiente función y los siguientes parámetros:

```
cv2.HoughCircles(blurred_roi, cv2.HOUGH_GRADIENT, dp=1.26, minDist=20, param1=50,
param2=30, minRadius=60, maxRadius=66). A estos valores llegamos tras varias pruebas
en la que obteníamos falsos negativos y falsos positivos. Aunque con estos parámetros la
función es muy precisa, en ocasiones siguen ocurriendo pequeños errores de detección,
detectando círculos cuando la mano está abierta. Sin embargo, estos errores son mínimos
y hemos logrado aumentar la precisión con respecto a las pruebas anteriores.
```

2- Canny

El algoritmo de Canny se emplea para detectar los bordes de la mano y extraer sus contornos. Esto nos permite llevar un análisis más preciso de la forma de la mano. Para obtener una mejor precisión, tratando de eliminar las líneas que aparecían entre los dedos al cerrar el puño, decidimos aplicar primero un filtro Gaussiano, empleando la siguiente función y parámetros: cv2.GaussianBlur(gray_frame, (11, 11), 3). Después, aplicamos la función Canny con los siguientes parámetros: cv2.Canny(blurred_frame, 20, 175)

3- Formación de combinaciones

Para determinar una función, creamos un diccionario en el que establecimos algunas condiciones predeterminadas, de modo que, si al mostrar esa combinación con la mano en la cámara, el resultado sería el emotícono asignado a dicha combinación. Las combinaciones son las siguientes:

- “mano abierta”, “puño cerrado”, “mano abierta”, “puño cerrado”: Corazón ❤
- “puño cerrado”, “puño cerrado”, “mano abierta”, “mano abierta”: Perro 🐶
- “mano abierta”, “mano abierta”, “mano abierta”, “mano abierta”): Fiesta 🎉
- “puño cerrado”, “mano abierta”, “mano abierta”, “puño cerrado”: Sonrisa 😊
- “puño cerrado”, “mano abierta”, “puño cerrado”, “mano abierta”: Fuego 🔥

Para gestionar las combinaciones, inicializamos una lista vacía, hand_states, en la que iremos almacenando el estado de la mano en un periodo de 50 frames para cada uno de estos. La lista almacenará 4 estados y una vez esté completa, se compara con las

combinaciones predefinidas. Si la combinación coincide con alguna de estas, se asigna el emotícono que le corresponda. Si no es así, se comenzará de nuevo el proceso.

Sistema propuesto: tracker, ampliaciones y salida de vídeo.

Una vez detectado el patrón introducido por el usuario, si forma parte de nuestros patrones preestablecidos, aparecerá un emoji en la mano del usuario. Este emoji se mueva de manera coherente con la mano del usuario, ya que se pone sobre la mano detectada por MediaPipe.

Resultados

Durante el desarrollo y pruebas de nuestro sistema, hemos identificado ciertas limitaciones y dificultades en el proceso de detección y seguimiento de la mano:

- Identificación en fondos claros: Uno de los principales retos ha sido la identificación de la mano en fondos con colores claros o uniformes. En estos casos, el contraste entre la mano y el fondo no es suficiente, lo que dificulta la detección precisa de la región de interés (ROI).
- Detecciones erróneas: Ocasionalmente, el sistema realiza detecciones incorrectas debido a problemas momentáneos, como movimientos bruscos, o cambios rápidos de iluminación. Estas situaciones afectan la estabilidad del seguimiento y la precisión del reconocimiento de patrones.

A pesar de estas dificultades, el sistema ha mostrado un buen desempeño, logrando identificar y seguir gestos de la mano de manera efectiva en la mayoría de las pruebas realizadas.

Futuros desarrollos

A partir de este proyecto hay una serie de futuros desarrollos y aplicaciones que serían muy fáciles de implementar con el código ya existente.

Nuevas formas con la mano: incluir combinaciones con más formas que las dos existentes. Esto se podría hacer fácilmente analizando las formas que hacen las manos haciendo ciertos gestos, como levantando dos dedos, forma de corazón...

Aplicación a video llamadas: nuestro proyecto podrá ser un complemento divertido en plataformas de video llamada como Skype y FaceTime. Detectando gestos en tiempo real, permite añadir emojis o efectos visuales que siguen los movimientos de las manos, haciendo las conversaciones más expresivas y dinámicas. Esta funcionalidad fomenta la creatividad y ofrece una nueva forma de interacción no verbal, mejorando la experiencia de los usuarios de manera sencilla y entretenida.

Aplicación creativa como contraseña: también podría tener una aplicación como método de autenticación, utilizando este detector para meter la contraseña de tu ordenador o dispositivo electrónico.