

**20 Seguimiento de Puntos con OpenCV y Tkinter**

**Tutorial Procesamiento de Imagen con webcam**

**Estudiantes Practicantes:**

Luis Pereira

**Profesor:**

Luis Vera

**Laboratorio CIMUBB**

2023-2

**Introducción al Programa de Seguimiento de Puntos con OpenCV y Tkinter:**

En la vanguardia de la visión por computadora, el seguimiento de objetos y la detección de características son cruciales. Presentamos un programa que fusiona las potentes bibliotecas OpenCV y Tkinter en Python para crear una aplicación interactiva. Esta aplicación permite el seguimiento en tiempo real de un punto específico en la imagen capturada por la cámara de un dispositivo.

OpenCV, una biblioteca de código abierto, se destaca en el procesamiento de imágenes y visión por computadora. En nuestro programa, aprovechamos sus funciones para la detección de bordes, el seguimiento óptico de puntos mediante el método de Lucas-Kanade, y la detección de líneas utilizando la transformada de Hough.

El método de Lucas-Kanade se basa en el flujo óptico, modelando cómo los píxeles se desplazan entre fotogramas consecutivos. La función cv2.calcOpticalFlowPyrLK implementa este algoritmo, rastreando el movimiento del punto seleccionado.

La transformada de Hough se emplea para la detección de líneas, específicamente a través de cv2.HoughLinesP. Este método transforma información de píxeles en el espacio de parámetros de líneas, donde las líneas se representan como puntos. Al encontrar picos en este espacio de parámetros, identificamos las líneas en la imagen.

Además, el programa busca intersecciones entre las líneas detectadas, resaltándolas visualmente en la imagen. Esta aplicación no solo demuestra la versatilidad de OpenCV y Tkinter, sino que también sirve como base para aplicaciones más avanzadas en el campo de la visión por computadora.

**Objetivo del Tutorial:**

El objetivo principal de este programa es proporcionar una herramienta versátil para el seguimiento de puntos y la detección de líneas e intersecciones en la imagen de la cámara en tiempo real. La interfaz gráfica creada con Tkinter facilita la interacción del usuario, permitiendo la selección manual de un punto de interés.

**Funcionalidades Destacadas:**

1. **Seguimiento Óptico de Puntos:**

Utiliza el método de Lucas-Kanade para rastrear un punto seleccionado en la imagen, proporcionando una experiencia interactiva.

1. **Detección de Líneas:**

Aplica el operador Canny para detectar bordes y utiliza la transformada de Hough para encontrar y visualizar líneas en la imagen.

1. **Detección de Intersecciones:**

Identifica los puntos de intersección entre las líneas detectadas, destacando áreas de convergencia en la escena.

1. **Interfaz Gráfica Amigable:**

Utiliza Tkinter para crear una interfaz gráfica de usuario (GUI) que permite ajustar el umbral de detección de bordes y muestra las coordenadas del punto seleccionado.

**Cómo Utilizar el Programa:**

1. **Inicio**:

Al ejecutar el programa, la interfaz gráfica se abrirá, y la cámara comenzará a capturar imágenes en tiempo real.

1. **Selección de Puntos:**

Haz clic en la imagen para seleccionar manualmente un punto de interés. Este punto será rastreado en fotogramas sucesivos cada 1 segundo.

1. **Ajuste de Parámetros:**

Utiliza la barra deslizante para ajustar el umbral de detección de bordes (Canny), adaptando la sensibilidad a las condiciones específicas de la escena.

1. **Visualización en Tiempo Real:**

Observa cómo el programa realiza el seguimiento del punto seleccionado y detecta líneas e intersecciones en tiempo real.

1. **Coordenadas y Registro:**

Las coordenadas del punto seleccionado se muestran en la interfaz, y se registran automáticamente en un archivo de texto para su posterior análisis.

**Explicación de las funciones:**

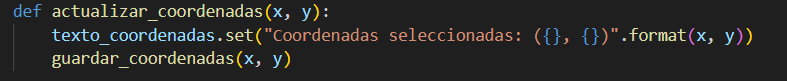
**mostrar\_imagen(imagen):**

**Texto

Descripción generada automáticamente**

1. Convierte la imagen de formato BGR a RGB utilizando OpenCV y PIL.
2. Crea un objeto ImageTk.PhotoImage a partir de la imagen convertida.
3. Configura el widget panel\_imagen para mostrar la imagen actualizada.

**actualizar\_coordenadas(x, y):**

****

1. Actualiza el texto del widget label\_coordenadas con las coordenadas (x, y).
2. Llama a la función guardar\_coordenadas(x, y) para guardar las coordenadas en un archivo.

**guardar\_coordenadas(x, y):**

**Texto

Descripción generada automáticamente**

Abre el archivo "coordenadas.txt" en modo de añadir ("a") y escribe las coordenadas en formato de cadena

**procesar\_imagen():** Se explicara el código de la función en 2 imágenes ya que la función es robusta.

**Texto

Descripción generada automáticamente**

**Hasta aquí la función hace lo siguiente:**

1. **Declaración de Variables Globales:**

* Se declara que las variables gris\_anterior, punto\_seleccionado, y punto\_interseccion son globales. Esto es necesario para poder modificar estas variables dentro de la función.

1. **Obtención del Valor del Slider:**

* Obtiene el valor actual del slider (slidebar\_canny) que se utiliza como umbral para el detector de bordes Canny.

1. **Captura de Fotograma:**

* Lee un fotograma de la cámara utilizando cap.read().
* Verifica si la operación de lectura (ret) fue exitosa. Si no, imprime un mensaje de error y sale de la función.

1. **Preprocesamiento de la Imagen:**

* Convierte el fotograma a escala de grises (gris).
* Aplica un desenfoque gaussiano (desenfoque) para reducir el ruido.
* Utiliza el operador Canny para detectar bordes en la imagen (bordes).

1. **Rastreo del Punto Seleccionado:**

* Comprueba si hay un punto seleccionado (punto\_seleccionado).

Si hay un punto seleccionado, utiliza el método de Lucas-Kanade para rastrear ese punto en el fotograma actual. Actualiza la posición del punto seleccionado si el seguimiento es exitoso.

Texto

Descripción generada automáticamente

**Esta parte del código se encarga de:**

1. **Dibujar Punto Seleccionado:**

* Si hay un punto seleccionado, dibuja un círculo verde en la imagen en esa posición.

1. **Detectar y Dibujar Líneas:**

* Utiliza la transformada de Hough para detectar líneas en la imagen.
* Dibuja las líneas detectadas en rojo.

1. **Buscar e Imprimir Intersecciones:**

* Encuentra y dibuja los puntos de intersección entre las líneas en azul.

1. **Actualizar Imagen y Coordenadas:**

* Muestra la imagen actualizada con el punto, líneas e intersecciones.
* Actualiza la imagen anterior y las coordenadas del punto seleccionado.
* Llamada Recursiva:
* Programa una llamada recursiva para procesar el siguiente fotograma después de 1 segundo.

**encontrar\_puntos\_interseccion(lineas):**

**Pantalla de computadora

Descripción generada automáticamente**

La función encontrar\_puntos\_interseccion calcula los puntos de intersección entre todas las combinaciones de líneas proporcionadas. Para cada par de líneas, determina si son no paralelas y, en caso afirmativo, calcula las coordenadas de intersección. Luego, devuelve la lista de todos los puntos de intersección encontrados.

**ventana:**

**Texto

Descripción generada automáticamente**

**Texto

Descripción generada automáticamente**

La sección 'ventana' del código realiza las siguientes acciones: Crea la ventana principal de la interfaz gráfica de usuario con Tkinter, inicia la captura de video (cámara) utilizando OpenCV, configura widgets como panel\_imagen, slidebar\_canny, y label\_coordenadas, inicializa el punto seleccionado y la imagen anterior, define funciones para cerrar la ventana y manejar clics en la imagen, inicia el procesamiento de imágenes llamando a procesar\_imagen(), y lanza el bucle principal de la interfaz gráfica con ventana.mainloop().