CC3182 – Visión por Computadora

Laboratorio 1

Instrucciones

- Esta es una actividad en grupos de no más de 3 integrantes.
  - Recuerden **unirse al grupo de canvas**
- No se permitirá ni se aceptará cualquier indicio de copia. De presentarse, se procederá según el reglamento correspondiente.
- Tendrán hasta el día indicado en Canvas.
  - No se confíen, aprovechen el tiempo en clase para entender todos los ejercicios y avanzar lo más posible.
- **NOTA**: Limiten el uso de IA generativa. Intenten primero buscar en fuentes de internet y si en verdad necesitan usarla, asegúrense de colcar el prompt que utilizar para cada task donde corresponda, así como una explicación de por qué ese prompt funcionó.

Usted ha sido contratado como Ingeniero de Visión Computacional Junior en "AutonoVision", una startup que desarrolla sistemas de navegación para robots de almacén.

El equipo de hardware ha instalado nuevas cámaras, pero las imágenes llegan con mucho ruido térmico debido a las condiciones de luz del almacén. El módulo actual de detección de obstáculos está fallando: detecta la textura del suelo de concreto como si fueran "bordes" de obstáculos, frenando el robot innecesariamente.

Su Director de Proyecto le ha asignado la tarea de construir un pipeline de pre-procesamiento robusto desde cero para entender el problema a nivel matemático y ajustar los parámetros óptimos para el despliegue.

Task 1 - Análisis Teórico y Analítico

Considerando el escenario previamente planteado, conteste:

1. Su jefe sugiere usar un filtro de media (Box Filter) de 7x7 para eliminar el ruido rápido. Usted cree que es una mala idea. Explique matemáticamente y con un diagrama visual (dibujado) por qué un Box Filter de ese tamaño es perjudicial para la detección precisa de la posición de un obstáculo comparado con un filtro Gaussiano del mismo tamaño.
2. Al realizar la convolución en los bordes de la imagen (por ejemplo, en el píxel 0,0), el kernel "se sale" de la imagen.
   a. Si el robot navega por pasillos oscuros con luces brillantes al final, ¿por qué el *Zero-Padding* podría generar falsos positivos de bordes en la periferia de la imagen?
   b. ¿Qué estrategia de padding (Reflect, Replicate, Wrap) recomendaría para evitar esto y por qué?
3. Dada la siguiente sub-imagen I de 3x3 y el kernel K:
   a. Calcule el valor del píxel central resultante de la convolución
   b. ¿Qué tipo de estructura detecta este filtro K (conocido como Laplaciano)?

$$I = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 0 & 10 \\ 10 & 10 & 10 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Task 2 – Práctica

Se permite el uso de OpenCV unicamente para lectura y/o escritura de imágenes y visualizaciónd de las mismas, **no** para algoritmos de procesamiento. Es decir, está prohibido usar cv2.filter2D, cv2.GaussianBlur, cv2.Sobel, o cv2.Canny. Debe usar NumPy y operaciones matriciales

Ejercicio 1: Convolución 2D Genérica

Escriba una función `mi_convolucion(imagen, kernel, padding_type='reflect')`, Considerando lo siguiente:

- **Restricción 1**: La función debe manejar imágenes en escala de grises.
- **Restricción 2**: Debe implementar el padding manualmente antes de operar.
- **Reto de optimización**: Intente no usar 4 bucles for anidados. Investigue cómo usar slicing de NumPy o np.einsum para hacerlo vectorizado, o al menos reduzca a 2 bucles.
- **Nota**: Recuerde que matemáticamente la convolución invierte el kernel. Implemente el "flip" del kernel dentro de la función.

Ejercicio 2: Generador de Gaussianos

Escriba una función generar_gaussiano(tamano, sigma). Para ello considere:

- La función debe devolver una matriz cuadrada de tamano x tamano con los coeficientes de una distribución Gaussiana 2D centrada.
- **Importante**: Asegúrese de que la suma de todos los elementos de la matriz sea igual a 1.0 (Normalización).

Ejercicio 3: Pipeline de Detección de Bordes (Sobel)

Cree una función detectar_bordes_sobel(imagen).. Para ello considere:

- Aplique los kernels de Sobel G_x y G_y usando su función mi_convolucion
- Calcule y retorne dos matrices:
  - **Magnitud del gradiente**: $G = \sqrt{G_x^2 + G_y^2}$ (Normalizada a 0-255 para visualizar)
  - **Dirección del gradiente**: θ=arctan2(*Gy*,*Gx*) (En radianes o grados)

Task 3 – Evaluación de Ingenería y Criterio

En esta parte se evaluará la aplicación de sus algoritmos en situaciones reales. Use imágenes propias o descargue un datase de “suelos de almacén” o “carreteras con textura”.

Experimento A: El efecto de Sigm (σ)

Cargue una imagen con ruido (agregue ruido "Sal y Pimienta" o Gaussiano artificialmente a una foto limpia si es necesario).

1. Genere 3 versiones de detección de bordes (Magnitud Sobel) variando el pre-procesamiento Gaussiano:
   a. Sin suavizado.
   b. Gaussiano $\sigma = 1$ (kernel sugerido 5x5).
   c. Gaussiano $\sigma = 5$ (kernel sugerido 31x31).

2. **Análisis**: Muestre las tres imágenes de bordes resultantes. ¿Qué pasa con los bordes finos cuando σ es muy alto? ¿Qué pasa con la textura del suelo cuando no hay suavizado? Como ingeniero, ¿cuál elegiría para detectar pallets grandes ignorando grietas pequeñas en el suelo?

Experimento B: Histéresis Manual (Simulación de Canny)

Usted ha calculado la Magnitud del Gradiente en el paso 3.3. Ahora implemente una función simple de umbralización umbral_simple(magnitud, T) y compare visualmente con cv2.Canny.

1. Intente encontrar un valor T único que limpie el ruido pero mantenga los bordes.
2. Observe el resultado: ¿Se rompen las líneas de los bordes?
3. **Pregunta Crítica**: Explique por qué un simple umbral de corte (Thresholding) nunca será tan efectivo como el método de Histéresis usado en Canny. ¿Qué problema específico resuelve la conectividad de la histéresis en el contexto de un robot moviéndose y vibrando (lo que causa cambios leves de iluminación en los bordes)?

Entregas en Canvas

1. Documento PDF con las respuestas a cada task
2. Archivo .py, o link a repositorio de GitHub (No se acepta entregas en otros medios)

Evaluación

1. [1 pt] Task 1
2. [2.5 pt] Task 2
3. [1.5 pt] Task 3

Total 5 pts