CC3182 – Visión por Computadora

Laboratorio 3

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.

Consideraciones Generales

- El lenguaje sugerido para este laboratorio es Python, con las librerías de OpenCV y Numpy
- Se debe entregar tanto Jupyter Notebook (.ipynb) como la versión PDF del mismo, en este debe contener el código, las visualizaciones y principalmente el análisis escrito justificando cada decisión
- Usted es responsible de importar las librerías, leer las imagenes y estructurar su solución

Task 1

Usted deberá demostrar comprensión teórica del criterio de Harris sin usar librerías. Para ello considere lo siguiente.

Se te da la siguiente Matriz del Segundo Momento (Tensor de Estructura) M calculada en un píxel específico (u,v) de una imagen:

$$M = \begin{bmatrix} 120 & 5 \\ 5 & 115 \end{bmatrix}$$

Y una segunda matriz M' para otro píxel diferente:

$$M' = \begin{bmatrix} 200 & 10 \\ 10 & 1 \end{bmatrix}$$

Con esta información haga lo siguiente:

1. Calcule manualmente (muestra tu procedimiento) los **Eigenvalores** ($\lambda 1$, $\lambda 2$) para ambas matrices.
2. Calcule la **Respuesta de Harris** (R) para ambas matrices usando la fórmula vista en clase. Asuma k = 0.04
3. Basado en tus resultados numéricos, clasifica qué representa cada píxel geométricamente: **¿Es una Esquina, un Borde o una Región Plana**? Justifica tu respuesta usando las definiciones de eigenvalores vistas en clase.

Task 2

En sta parte se busca que usted implemente un sistema de correspondencia completo. Para esto debe escribir un script en Python usando OpenCV. No se provee código base, debe estructurarlo usted mismx. Para esta parte debe crear su propia imagen a usar para esto:

- Tome dos fotografías propias de un objeto con textura (i.e. una caja de cereral, una portada de libro, un edificio)
- Foto 1: Vista frontal
- Foto 2: Vista rotada (aproximadamente 45 grados) y con cambio de escala (aléjese o haga zoom). El cambio debe ser evidente

Con esto haga lo siguiente en su código

1. Cargue ambas imágenes en escala de grises
2. Implemente la detección y descripción usando SIFT.
3. Implemente la detección y descripción usando ORB.
4. Realice la parte de Matching:
    a. Para SIFT: Utilice BFMatcher con normal L2 (Euclidiana)
    b. Para ORB: Utilice BFMatches con norma de Hamming
5. Implemente Lowe's Ratio Test para ambos algoritmos
    a. Debe filtrar los matches donde la distancia del mejor vecino sea mayor a 0.75 x la distancia del segundo mejor vecino
6. Genere una imagen final donde se dibujen las líneas de correspondencia solamente de los "buenos matches" (inliers) tras el filtro.

Task 3

En esta parte lo que se busca es evaluar trade-offs y tomar decisiones técnicas basadas en datos. Por ello es importante considerar que la implementación por si sola no nos sirve para saber cuánto "cuesta" ejecutarla. Debe modificar un poco su código del task 2 (en su entrega deben mostrar el código de la parte 2 solamente y luego copie y pegue en otra sección el codigo y en este haga los cambios necesarios) para medir el tiempo.

Imagine que ha sido contratadx como Ingenierx de Visión Computacional en una startup de drones autónomos. La empresa tiene dos productos en desarrollo y usted debe decidir qué algoritmo implementar en cada uno:

1. Producto A (Drone de Carreras): Vuela a 80 km/h en interiores. Necesita estimar su posición (Odometría Visual) a 60 FPS mínimo. La calidad de la imagen es baja y borrosa.
2. Producto B (Drone de Inspección de Grietas): Vuela estático frente a presas hidroeléctricas. Toma fotos de altísima resolución (4K) para crear un mapa detallado (Stitching). El tiempo de procesamiento no es crítico, pero la precisión del emparejamiento debe ser milimétrica.

Su misión es medir y justificar qué algoritmo usará para cada caso basándote en datos empíricos.

Con esto en mente, realice lo siguiente:

1. Usando librerías como time o o timeit para medir tiempo por separado, mida:
    a. Tiempo promedio de Detección + Descripción (en milisegundos)
    b. Tiempo promedio de Matching (en milisegundos)
    c. Asegurese de realizar esto apra ambos algoritmos de la parte 2 (SIFT y ORB)

2. Cree una tabla en su reporte con las siguientes columnas:
    a. Algoritmo
    b. Tiempo total (en milisegundos)
    c. Número de Keypoints detectados (Imagen A / Imagen B)
    d. Número de Matches “Buenos” (post-Ratio Test)
3. Finalmente, realice un análisis crítico, para ello responda
    a. ¿Cuál algoritmo elegiría para el Producto A (Drone de Carreras) y por qué? Base su respuesta en los milisegundos que mediste y la tasa de refresco requerida (60 FPS ≈ 16ms de presupuesto total).
    b. ¿Cuál algoritmo elegirías para el Producto B (Inspección) y por qué? Analice la calidad visual de los matches en los cambios de escala y rotación que probaste. ¿Falló ORB en algún caso donde SIFT tuvo éxito?
    c. ¿Las conclusiones que estamos alcanzando son justas y generalizables? ¿Por qué? ¿Qué deberíamos considerar en futuras iteraciones?

Entregas en Canvas

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

Evaluación

1. [1 0 pt] Task 1
2. [2.5 pt] Task 2
3. [1.5 pt] Task 3

Total 5 pts