CENTRO DE ENSEÑANZA
TÉCNICA INDUSTRIAL

Maestro: Mauricio Alejandro Cabrera Arellano

Alumno: Alejandro Retana Rubio 22110315

Materia: Visión Artificial

Practica 10

Fecha: 09-06-2025

Detección de Esquinas en Imágenes con el Detector de Harris

1. Procesamiento de Imágenes Digitales

El procesamiento de imágenes digitales consiste en aplicar técnicas computacionales para transformar, analizar y extraer información relevante de imágenes. Uno de los propósitos más comunes es identificar características visuales significativas, como bordes, esquinas, contornos, patrones o formas.

Este tipo de procesamiento es ampliamente utilizado en áreas como:

- Visión artificial
- Robótica
- Sistemas biométricos
- Inspección industrial
- Reconocimiento facial y de objetos

2. Escala de Grises y ROI (Región de Interés)

Para facilitar el análisis computacional, muchas veces las imágenes a color se convierten a escala de grises. Esta conversión reduce la complejidad, ya que trabaja con una sola capa de intensidad en lugar de tres (RGB).

La **Región de Interés (ROI)** es un fragmento específico de la imagen que se selecciona para enfocarse en una zona de análisis, optimizando el rendimiento y evitando procesar zonas irrelevantes.

3. Detección de Características: Esquinas

Una de las características más importantes en una imagen son las **esquinas**. Estas son puntos donde hay un cambio brusco en dos direcciones, es decir, donde convergen dos bordes. Las esquinas son útiles porque:

- Son invariantes ante rotaciones y cambios de iluminación
- Pueden ser usadas como puntos de referencia
- Permiten emparejar imágenes, reconocer objetos o seguir movimiento

4. Detector de Harris (Corner Detection)

El detector de esquinas de Harris es un algoritmo clásico propuesto por Chris Harris y Mike Stephens en 1988. Se basa en calcular el cambio de intensidad en una ventana cuando se desplaza en diferentes direcciones. Si el cambio es alto en múltiples direcciones, se considera que hay una esquina.

El método de Harris calcula una matriz de auto-correlación que representa cómo varía la intensidad en una vecindad de píxeles. A partir de esto, se deriva una función de respuesta RRR, y los valores altos de RRR indican la presencia de una esquina.

Parámetros importantes del detector:

- `blockSize`: tamaño del vecindario para el cálculo de derivadas.
- `ksize`: tamaño del filtro de Sobel que estima el gradiente.
- k: constante de sensibilidad (usualmente entre 0.04 y 0.06).

5. Visualización de Resultados

Para visualizar los puntos detectados como esquinas, se marcan sobre la imagen original en color (usualmente con un color llamativo como rojo). Esto permite identificar visualmente las zonas clave de la imagen.

Aplicaciones del Detector de Esquinas de Harris

- Seguimiento de objetos en video
- Reconocimiento de patrones
- Emparejamiento de imágenes (matching)
- Creación de mapas tridimensionales (en visión estéreo)
- Sistemas de navegación autónoma

```
import cv2  # Librería OpenCV para procesamiento de imágenes
import numpy as np  # Librería NumPy para operaciones numéricas con matrices

# --------- Cargar imagen y convertir a escala de grises ---------
img_color = cv2.imread('luffy5.png')  # Cargar imagen en color desde archivo
img_gray = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)  # Convertir imagen a
escala de grises para análisis más sencillo
```

```python
# --------- Definir y extraer la Región de Interés (ROI) ---------
x, y, w, h = 450, 330, 100, 100  # Coordenadas iniciales (x, y) y dimensiones (w: ancho, h: alto) del recorte
roi_color = img_color[y:y+h, x:x+w]  # Extraer ROI en color
roi_gray = img_gray[y:y+h, x:x+w]    # Extraer ROI en escala de grises

# --------- Detección de esquinas usando el algoritmo de Harris ---------
roi_gray = np.float32(roi_gray)  # Convertir la imagen a tipo float32, requerido por cornerHarris
dst = cv2.cornerHarris(roi_gray, blockSize=2, ksize=3, k=0.04)
# Parámetros:
# - blockSize: Tamaño del vecindario considerado para cada píxel
# - ksize: Tamaño del kernel de Sobel usado para calcular derivadas
# - k: Parámetro de sensibilidad entre 0.04 y 0.06

dst = cv2.dilate(dst, None)  # Dilatar la imagen para resaltar mejor las esquinas detectadas

# --------- Marcar las esquinas detectadas en la imagen ---------
roi_color[dst > 0.01 * dst.max()] = [0, 0, 255]  # Pintar de rojo (BGR) los píxeles que superan el umbral

# --------- Mostrar el resultado en una ventana ---------
cv2.imshow('ROI con esquinas detectadas', roi_color)  # Mostrar solo el ROI con las esquinas marcadas
cv2.waitKey(0)  # Esperar a que el usuario presione una tecla
cv2.destroyAllWindows()  # Cerrar la ventana de la imagen
```