CENTRO DE ENSEÑANZA
TÉCNICA INDUSTRIAL

Maestro: Mauricio Alejandro Cabrera Arellano

Alumno: Alejandro Retana Rubio 22110315

Materia: Visión Artificial

Practica 9

Fecha: 09-06-2025

Práctica: Detección de Regiones por Coincidencia de Plantilla

En esta práctica se utilizó la técnica de **template matching** para localizar una región específica (plantilla) dentro de una imagen más grande. Se cargaron ambas imágenes en escala de grises para facilitar el procesamiento. Luego, se aplicó el método de **correlación cruzada normalizada** (`cv2.TM_CCOEFF_NORMED`), que compara la plantilla con cada posible posición en la imagen principal y genera un mapa de similitud con valores entre -1 y 1.

Para identificar coincidencias válidas, se estableció un **umbral de confianza de 0.85**. Todas las regiones con una similitud mayor o igual a este valor fueron consideradas detecciones. Finalmente, se dibujaron **rectángulos** alrededor de las coincidencias encontradas y se mostró el total de detecciones en consola.

Esta técnica es útil en tareas como reconocimiento de objetos, control de calidad en visión industrial o detección automática de símbolos o logotipos.

Codigo

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

# Cargar la imagen principal y la plantilla en escala de grises
img = cv2.imread('luffy5.png', cv2.IMREAD_GRAYSCALE)
template = cv2.imread('template.png', cv2.IMREAD_GRAYSCALE)
# 'img' es la imagen donde se buscará el patrón; 'template' es el patrón a localizar

# Obtener dimensiones del template
h, w = template.shape
```

```
    # Altura (h) y anchura (w) necesarias para dibujar los rectángulos de
                                coincidencia

       # Aplicar template matching con correlación cruzada normalizada
       result = cv2.matchTemplate(img, template, cv2.TM_CCOEFF_NORMED)
# Devuelve una matriz de similitud con valores entre -1 (mala coincidencia) y 1
                           (coincidencia perfecta)

         # Definir un umbral para considerar una coincidencia válida
                               threshold = 0.85
                    loc = np.where(result >= threshold)
# 'loc' contiene coordenadas donde el valor de coincidencia es mayor o igual al
                                    umbral

   # Convertir la imagen a color para poder dibujar los resultados en color
              img_color = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

                  # Contador de coincidencias encontradas
                               detect_count = 0
 for pt in zip(*loc[::-1]):  # Coordenadas (x, y) intercambiadas para dibujar
                                correctamente
                                detect_count += 1
       cv2.rectangle(img_color, pt, (pt[0] + w, pt[1] + h), (0, 0, 55), 5)
    # Dibujar un rectángulo azul oscuro de grosor 5 px alrededor de la región
                                 detectada

    # Imprimir cuántas coincidencias se encontraron por encima del umbral
  print(f"Regiones detectadas con confianza >= {threshold}: {detect_count}")

           # Mostrar la imagen con las coincidencias detectadas
                    cv2.imshow('Detecciones', img_color)
      cv2.waitKey(0)  # Espera hasta que el usuario presione una tecla
   cv2.destroyAllWindows()  # Cierra todas las ventanas abiertas de OpenCV
```

Codigo 9.0

```
                               import cv2

              # Cargar la imagen original en escala de grises
            img = cv2.imread('luffy5.png', cv2.IMREAD_GRAYSCALE)
```

```python
# Coordenadas del ROI que quieras extraer (ajústalas tú)
x, y, w, h = 450, 330, 100, 100
template = img[y:y+h, x:x+w]

# Guardar la plantilla
cv2.imwrite('template.png', template)
```