2025

Reporte

PRACTICA 9

ALEXIS DANIELCORTES TAVERA

Introducción

En esta práctica se trabajó con la técnica de **template matching**, que permite buscar regiones específicas dentro de una imagen que coincidan con un patrón o plantilla previamente definido. Este tipo de detección es útil cuando se desea localizar objetos repetidos o elementos iguales en distintas posiciones de una imagen.

A diferencia de otras técnicas, el template matching **no se basa en el color**, sino en la **forma, textura y distribución de intensidades** de los píxeles del template respecto a la imagen original. Por eso, es necesario trabajar en escala de grises para lograr una comparación efectiva

Desarrollo Teórico y Practico

En estas prácticas se usaron diversas herramientas para su realización que serán explicadas a continuación:

librerías:

- cv2: Biblioteca de OpenCV, usada para cargar imágenes, convertirlas a escala de grises, comparar plantillas y dibujar resultados.

- numpy: Utilizada para procesar matrices y localizar coordenadas con resultados mayores al umbral de coincidencia.

Practica 8:

Carga una imagen principal y dos templates diferentes, convierte todo a escala de grises para facilitar la comparación.

Aplica cv2.matchTemplate() para comparar cada template con la imagen, busca todas las coincidencias con una similitud mayor o igual a **0.85**.

Por ultimo dibuja rectángulos verdes para las coincidencias del primer template y rectángulos azules para las del segundo.

P9

```
import cv2import numpy as np

# Cargar imagen principal
img = cv2.imread('imagen.jpg')
img_copy = img.copy()
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Cargar template 1template1 = cv2.imread('template1.jpg')
template1_gray = cv2.cvtColor(template1, cv2.COLOR_BGR2GRAY)
w1, h1 = template1.shape[1], template1.shape[0]
# Cargar template 2
template2 = cv2.imread('template2.jpg')
template2_gray = cv2.cvtColor(template2, cv2.COLOR_BGR2GRAY)
w2, h2 = template2.shape[1], template2.shape[0]
# Umbral de coincidencia
umbral = 0.85

# ---------- Coincidencias para Template 1 ----------

res1 = cv2.matchTemplate(gray_img, template1_gray,
cv2.TM_CCOEFF_NORMED)
```

Maestro:
Mauricio Alejandro Cabrera Arellano

```
loc1 = np.where(res1 >= umbral)

for pt in zip(*loc1[::-1]):

    cv2.rectangle(img_copy, pt, (pt[0] + w1, pt[1] + h1), (0, 255, 0), 2)  # Verde

# ---------- Coincidencias para Template 2 ----------

res2 = cv2.matchTemplate(gray_img, template2_gray,
cv2.TM_CCOEFF_NORMED)

loc2 = np.where(res2 >= umbral)

for pt in zip(*loc2[::-1]):

    cv2.rectangle(img_copy, pt, (pt[0] + w2, pt[1] + h2), (255, 0, 0), 2)  # Azul

# Mostrar resultado

cv2.imshow('Coincidencias encontradas', img_copy)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Conclusión

Esta práctica fue muy útil para entender cómo se puede buscar un objeto específico dentro de una imagen de manera sencilla. El uso de dos plantillas permitió ver cómo el algoritmo encuentra diferentes elementos si tienen suficiente similitud.

Aunque es una técnica simple de implementar, también tiene limitaciones si el objeto está rotado, escalado o parcialmente cubierto. Aun así, para casos donde la imagen y los templates son similares en forma y tamaño, el **template matching** funciona bastante bien y de forma rápida.

Referencias:

Ibm. (2025, 6 enero). Computer Vision. *Vision Artificial*. https://www.ibm.com/mx-es/topics/computer-vision

Maestro:
Mauricio Alejandro Cabrera Arellano