Rodrigo Hernández Zavala

7 de junio de 2025

Ing. Mecatrónica

22310215

Template

Visión Artificial

Este código realiza una **detección de un objeto dentro de una imagen más grande** utilizando la técnica de **ORB (Oriented FAST and Rotated BRIEF)**. En este caso, busca el **símbolo de Corazón (template)** dentro de una imagen que contiene varios **Ases de cartas (escena)**.

🧠 ¿Qué hace paso a paso?

1. 🖼️ Carga las imágenes

```
python
CopiarEditar
img_scene = cv2.imread("Ases.jpg")
img_template = cv2.imread("Corazon.jpg")
```

- `Ases.jpg`: imagen grande que contiene varias cartas.
- `Corazon.jpg`: imagen recortada de una parte de una carta (símbolo o figura).
- Se valida que ambas se hayan cargado correctamente.

---

2. 🌑 Convierte ambas imágenes a escala de grises

```
python
CopiarEditar
gray_scene = cv2.cvtColor(img_scene, cv2.COLOR_BGR2GRAY)
gray_template = cv2.cvtColor(img_template, cv2.COLOR_BGR2GRAY)
```

Esto es necesario para detectar características visuales sin importar el color.

---

3. 🧠 Detecta características con ORB

```
python
CopiarEditar
orb = cv2.ORB_create(nfeatures=1000)
kp1, des1 = orb.detectAndCompute(gray_template, None)
kp2, des2 = orb.detectAndCompute(gray_scene, None)
```

- ORB encuentra **keypoints** (puntos clave únicos) y **descriptores** (vectores que describen su entorno).
- Se obtienen para ambas imágenes.

---

4. 🔍 Emparejamiento de descriptores con Brute Force

```
python
CopiarEditar
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=False)
matches = bf.knnMatch(des1, des2, k=2)
```

- Usa `BFMatcher` para comparar descriptores y buscar los más similares.
- `k=2` devuelve los dos matches más cercanos para cada punto.

---

5. ✅ Filtrado de buenas coincidencias (Lowe's Ratio Test)

```
python
CopiarEditar
buenas = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        buenas.append(m)
```

- Compara la distancia entre los dos matches más cercanos.
- Si el más cercano es significativamente mejor, se considera confiable.

---

6. 📐 Calcula homografía si hay suficientes coincidencias

```
python
CopiarEditar
if len(buenas) >= 10:
    ...
```

- Si se encuentran al menos 10 coincidencias:
  - Se calcula una **matriz de homografía** para alinear el template con la escena.
  - Se **transforma el contorno** del template a la posición donde se encontró.
  - Se dibuja un **polígono verde** donde el corazón fue localizado.

---

7. 🖼️ Muestra el resultado

```
python
CopiarEditar
cv2.putText(img_resultado, mensaje, ...)
cv2.imshow("Detección de Corazon con ORB", img_resultado)
```

- Se agrega un mensaje con la cantidad de coincidencias encontradas.

- Se muestra la imagen final con la detección resaltada.

✅ Resultado esperado:

- Si se encuentra el **símbolo de corazón** en alguna carta:
  - Se dibuja un **polígono verde** sobre él.
  - Se muestra: `"Coincidencias: X"`.
- Si no se encuentra:
  - Muestra la escena original con el texto `"No se encontró el Corazon"`.

📌 Recomendaciones para buen resultado:

- Que el template esté **completo y claro**.
- Que el símbolo esté **visible y sin rotación o deformaciones extremas**.
- Que el fondo no tenga **muchos patrones similares** (para evitar confusiones).

Código:

```
import cv2

import numpy as np

import os



# ---------- CARGA DE IMÁGENES ----------

# Usamos nombres relativos (mismo directorio)

img_scene = cv2.imread("Ases.jpg")

img_template = cv2.imread("Corazon.jpg")



if img_scene is None or img_template is None:

    print("❌ Error al cargar las imágenes. Asegúrate de que 'Ases.jpg' y 'Corazon.jpg' estén en la misma carpeta.")
```

```
    exit()

# Convertir a escala de grises
gray_scene = cv2.cvtColor(img_scene, cv2.COLOR_BGR2GRAY)
gray_template = cv2.cvtColor(img_template, cv2.COLOR_BGR2GRAY)

# ---------- DETECCIÓN ORB ----------
orb = cv2.ORB_create(nfeatures=1000)
kp1, des1 = orb.detectAndCompute(gray_template, None)
kp2, des2 = orb.detectAndCompute(gray_scene, None)

bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=False)
matches = bf.knnMatch(des1, des2, k=2)

# ---------- FILTRAR BUENAS COINCIDENCIAS ----------
buenas = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        buenas.append(m)

# ---------- HOMOGRAFÍA Y DIBUJO ----------
if len(buenas) >= 10:
    src_pts = np.float32([kp1[m.queryIdx].pt for m in buenas]).reshape(-1, 1, 2)
```

```
    dst_pts = np.float32([kp2[m.trainIdx].pt for m in buenas]).reshape(-1, 1, 2)

    M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)

    h, w = gray_template.shape
    pts = np.float32([[0, 0], [w, 0], [w, h], [0, h]]).reshape(-1, 1, 2)
    destino = cv2.perspectiveTransform(pts, M)

    img_resultado = cv2.polylines(img_scene.copy(), [np.int32(destino)], True, (0, 255, 0),
3)
    mensaje = f"Coincidencias: {len(buenas)}"
else:
    img_resultado = img_scene.copy()
    mensaje = "No se encontro el Corazon"

# ---------- MOSTRAR RESULTADO ----------
cv2.putText(img_resultado, mensaje, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
cv2.imshow("Detección de Corazon con ORB", img_resultado)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Funcionamiento: