

6_SIFT

November 14, 2024

0.0.1 SIFT

0.0.2

El método SIFT (Scale-Invariant Feature Transform) se utiliza para detectar y describir puntos clave en una imagen que son invariantes a transformaciones como escala, rotación e iluminación. SIFT localiza características distintivas en una imagen mediante la identificación de puntos de interés (keypoints) y el cálculo de descriptores, que representan la estructura local alrededor de cada punto clave. Estos descriptores se pueden comparar entre imágenes para encontrar coincidencias, lo cual permite reconocer objetos o partes de una escena incluso bajo variaciones significativas.

Primero realizamos la transformación de la imagen original en diversas formas, y la guardamos en la misma ubicación que la original. Las modificaciones que se aplican sobre la imagen son:

- 1- Rotación: Aplica rotaciones de 15, 30, 45 y 60 grados.
- 2- Traslación: Aplica desplazamientos de 30 o 50 píxeles en distintas direcciones.
- 3- Escala: Cambia el tamaño de la imagen a escalas de 0.5, 0.75, 1.5 y 2.0.
- 4- Iluminación: Ajusta el brillo de la imagen multiplicando los valores de píxeles.
- 5- Perspectiva: Aplica dos transformaciones de perspectiva que deforman la imagen.

Además, se utilizará una imagen diferente a la elegida para probar si funciona correctamente el algoritmo implementado.

```
[24]: import cv2
import numpy as np

# Se carga la imagen original
imagen = cv2.imread('lanus.png')

# Función para guardar la imagen modificada
def guardar_imagen(imagen, nombre):
    cv2.imwrite(nombre, imagen)
```

```

# 1. Rotación de la imagen
for angulo in [15, 30, 45, 60]:
    altura, ancho = imagen.shape[:2]
    matriz_rotacion = cv2.getRotationMatrix2D((ancho // 2, altura // 2), ↴
    ↵angulo, 1)
    imagen_rotada = cv2.warpAffine(imagen, matriz_rotacion, (ancho, altura))
    guardar_imagen(imagen_rotada, f'Images_SIFT/lanus_rotada_{angulo}.png')

# 2. Traslación de la imagen
for dx, dy in [(30, 50), (-30, -50), (50, -30), (-50, 30)]:
    matriz_traslacion = np.float32([[1, 0, dx], [0, 1, dy]])
    imagen_trasladada = cv2.warpAffine(imagen, matriz_traslacion, (ancho, ↴
    ↵altura))
    guardar_imagen(imagen_trasladada, f'Images_SIFT/lanus_trasladada_{dx}_{dy}.png')

# 3. Cambio de escala
for escala in [0.5, 0.75, 1.5, 2.0]:
    imagen_escalada = cv2.resize(imagen, None, fx=escala, fy=escala, ↴
    ↵interpolation=cv2.INTER_LINEAR)
    guardar_imagen(imagen_escalada, f'Images_SIFT/lanus_escalada_{escala}.png')

# 4. Cambio de iluminación
for alpha in [0.5, 1.5, 2.0]: # Factor de brillo
    imagen_iluminada = cv2.convertScaleAbs(imagen, alpha=alpha, beta=0)
    guardar_imagen(imagen_iluminada, f'Images_SIFT/lanus_iluminada_{alpha}.png')

# 5. Transformación de perspectiva
puntos_originales = np.float32([[0, 0], [ancho - 1, 0], [0, altura - 1], [ancho - 1, altura - 1]])
puntos_destino = [
    np.float32([[0, 0], [ancho - 1, 0], [ancho * 0.1, altura - 1], [ancho * 0.9, altura - 1]]),
    np.float32([[ancho * 0.1, 0], [ancho * 0.9, 0], [0, altura - 1], [ancho - 1, altura - 1]]),
]
for i, destino in enumerate(puntos_destino):
    matriz_perspectiva = cv2.getPerspectiveTransform(puntos_originales, destino)
    imagen_perspectiva = cv2.warpPerspective(imagen, matriz_perspectiva, ↴
    ↵(ancho, altura))
    guardar_imagen(imagen_perspectiva, f'Images_SIFT/lanus_perspectiva_{i}.png')

print("Imágenes modificadas generadas y guardadas.")

```

Imágenes modificadas generadas y guardadas.

Implementación del metodo SIFT

Se aplica el metodo y se compara la imagen original con todas las generadas en el bloque anterior para verificar si las imagenes son las mismas o no, y si puede detectarlo a partir de traslaciones, rotaciones, cambios en iluminacion, cambios de escala y movimientos de perspectiva.

```
[26]: import cv2
import os
import matplotlib.pyplot as plt

directorio_imagenes = "Images_SIFT/"

# Se carga la imagen original
imagen_original = cv2.imread('lanus.png')
imagen_original_gray = cv2.cvtColor(imagen_original, cv2.COLOR_BGR2GRAY)

# Se inicializa el detector SIFT
sift = cv2.SIFT_create()

# Detectar los puntos clave y calcular los descriptores de la imagen original
keypoints1, descriptors1 = sift.detectAndCompute(imagen_original_gray, None)

# Configurar el matcher con FLANN (Fast Library for Approximate NearestNeighbors)
index_params = dict(algorithm=1, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)

# Recorrer cada imagen generada en el directorio
for nombre_archivo in os.listdir(directorio_imagenes):
    if nombre_archivo.startswith("lanus"):
        # Cargar la imagen modificada
        imagen_modificada = cv2.imread(os.path.join(directorio_imagenes, nombre_archivo))
        imagen_modificada_gray = cv2.cvtColor(imagen_modificada, cv2.COLOR_BGR2GRAY)

        # Detectar puntos clave y descriptores de la imagen modificada
        keypoints2, descriptors2 = sift.detectAndCompute(imagen_modificada_gray, None)

        # Realizar la coincidencia de descriptores con KNN
        matches = flann.knnMatch(descriptors1, descriptors2, k=2)

        # Aplicar la razón de Lowe para filtrar las mejores coincidencias con un umbral más bajo
        good_matches = []
```

```

    for m, n in matches:
        if m.distance < 0.5 * n.distance: # Ajuste a 0.5 para aún mayor
            ↵restricción
                good_matches.append(m)

    # Filtrar coincidencias visualizadas si son pocas o inadecuadas
    if len(good_matches) < 10:
        print(f"Fewer than 10 matches found between the original image and the "
            ↵{nombre_archivo}.")
        continue # Salta a la siguiente imagen si hay pocas coincidencias

    # Limitar el número de coincidencias visualizadas
    imagen_coincidencias = cv2.drawMatches(imagen_original, keypoints1,
            ↵imagen_modificada, keypoints2, good_matches[:30], None, flags=cv2.
            ↵DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

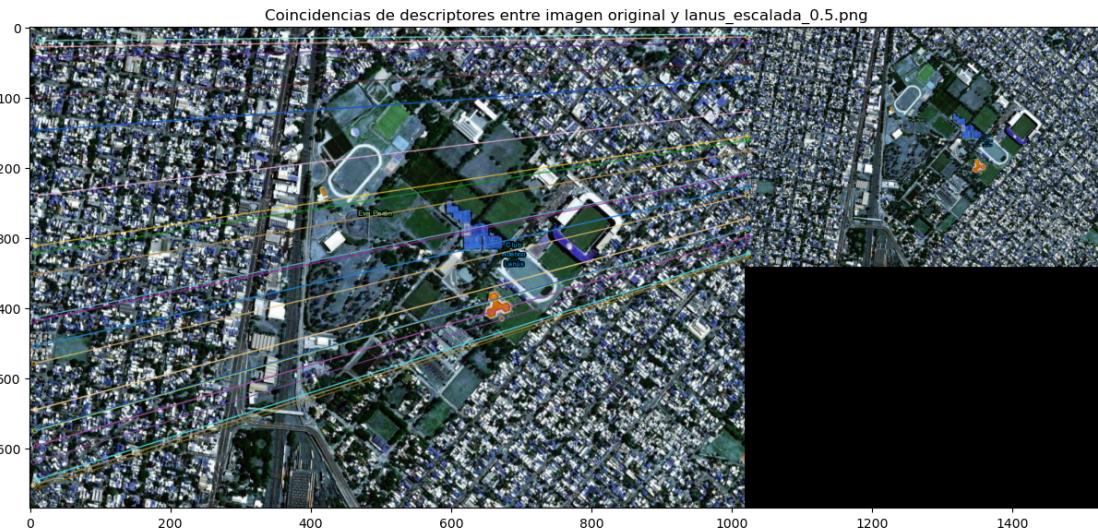
    # Mostrar la imagen con coincidencias
    plt.figure(figsize=(15, 10))
    plt.imshow(imagen_coincidencias)
    plt.title(f"Matches between the original image and the "
            ↵{nombre_archivo}")
    plt.show()

    # Calcular la cantidad de coincidencias
    num_keypoints1 = len(keypoints1)
    num_keypoints2 = len(keypoints2)
    num_matches = len(good_matches)

    print(f"\nComparison with {nombre_archivo}:")
    print(f"Number of key points in the original image: {num_keypoints1}")
    print(f"Number of key points in {nombre_archivo}: {num_keypoints2}")
    print(f"Total matches found: {num_matches}")

    # Determinar similitud en función de las coincidencias
    threshold = 0.1 * min(num_keypoints1, num_keypoints2)
    if num_matches > threshold:
        print("The images are likely the same.")
    else:
        print("The images are different.")

```



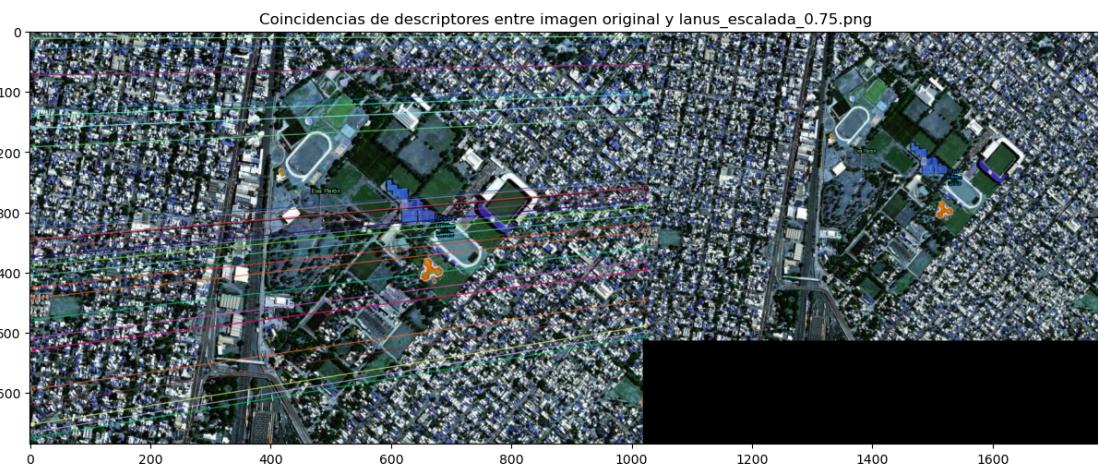
Comparación con lanus_escalada_0.5.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_escalada_0.5.png: 4297

Total de coincidencias encontradas: 3014

Las imágenes son probablemente las mismas.



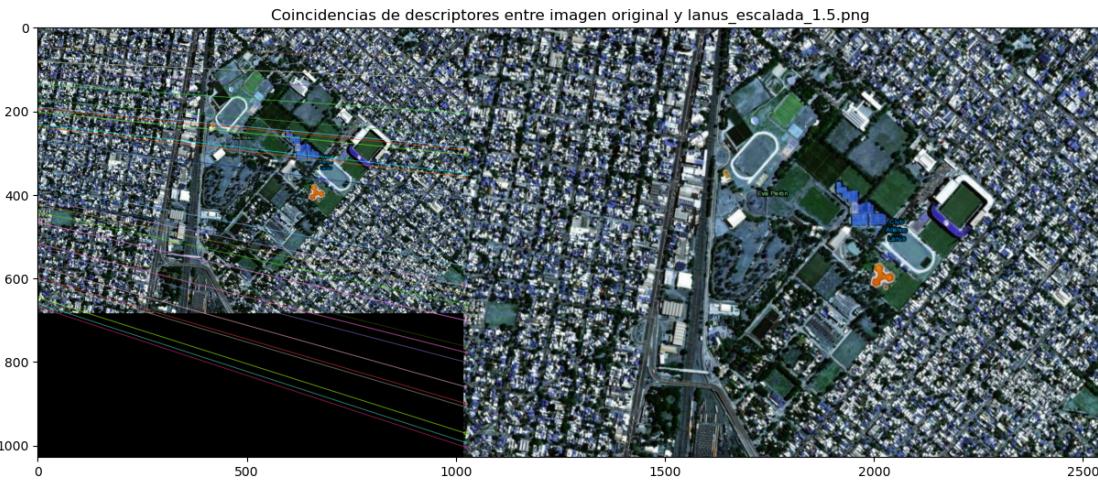
Comparación con lanus_escalada_0.75.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_escalada_0.75.png: 10898

Total de coincidencias encontradas: 7389

Las imágenes son probablemente las mismas.



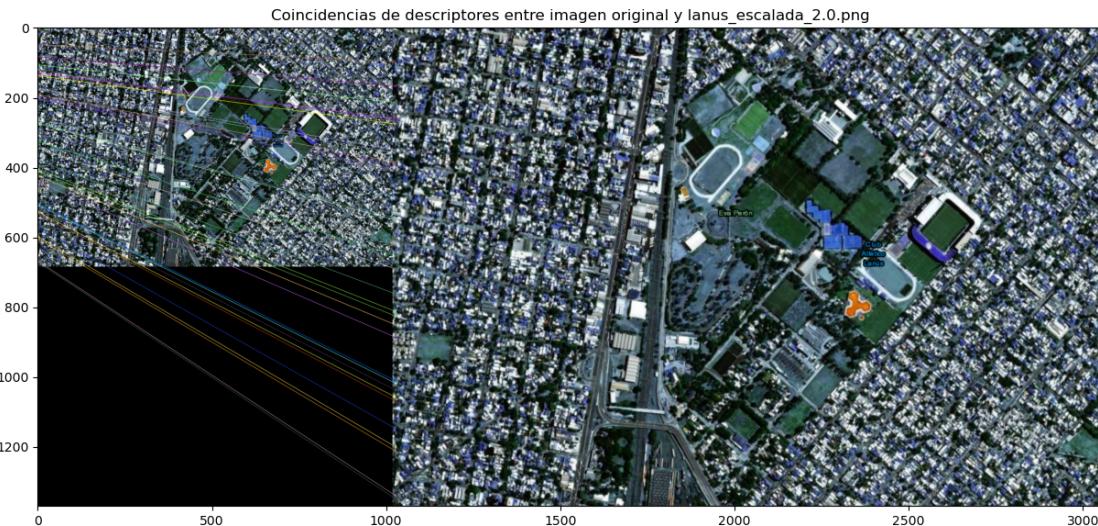
Comparación con lanus_escalada_1.5.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_escalada_1.5.png: 36929

Total de coincidencias encontradas: 11651

Las imágenes son probablemente las mismas.



Comparación con lanus_escalada_2.0.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_escalada_2.0.png: 44852

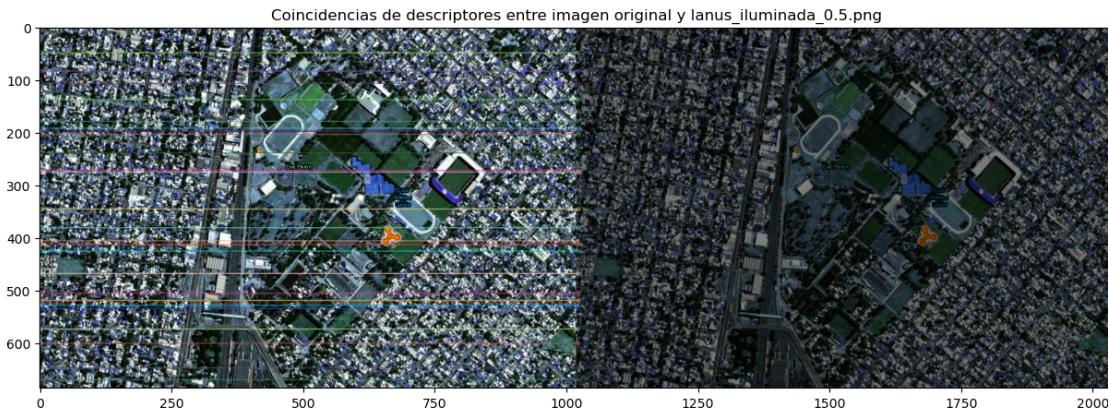
Total de coincidencias encontradas: 12196

Las imágenes son probablemente las mismas.

Pocas coincidencias encontradas entre la imagen original y lanus_falsa.jpg.

Pocas coincidencias encontradas entre la imagen original y lanus_falsa2.png.

Pocas coincidencias encontradas entre la imagen original y lanus_falsa3.png.



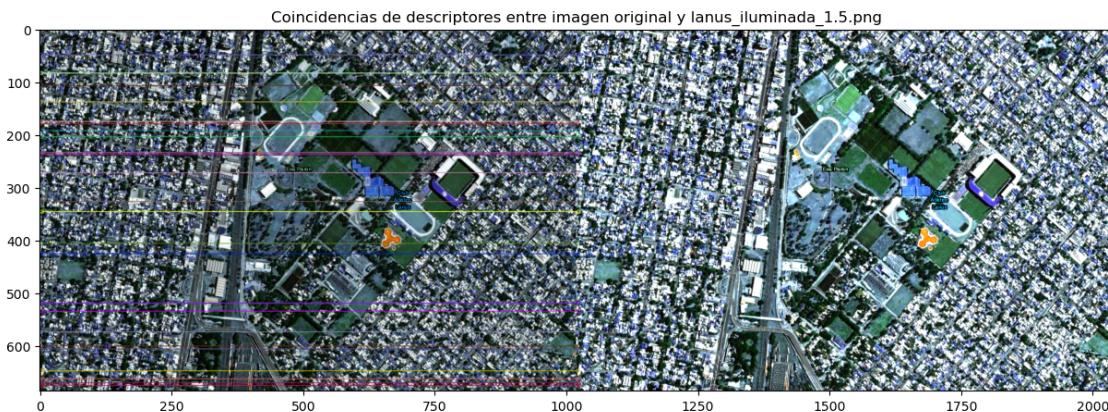
Comparación con lanus_iluminada_0.5.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_iluminada_0.5.png: 15751

Total de coincidencias encontradas: 15014

Las imágenes son probablemente las mismas.



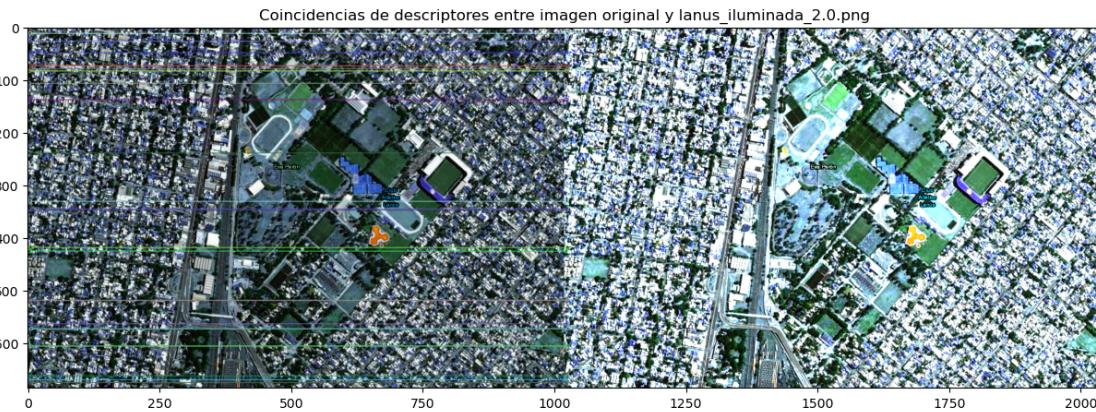
Comparación con lanus_iluminada_1.5.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_iluminada_1.5.png: 18504

Total de coincidencias encontradas: 10134

Las imágenes son probablemente las mismas.



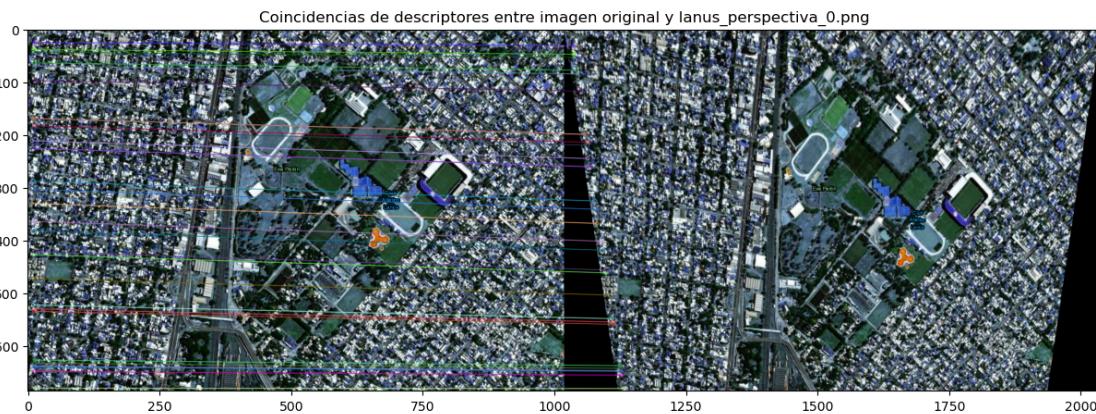
Comparación con lanus_iluminada_2.0.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_iluminada_2.0.png: 18509

Total de coincidencias encontradas: 4245

Las imágenes son probablemente las mismas.



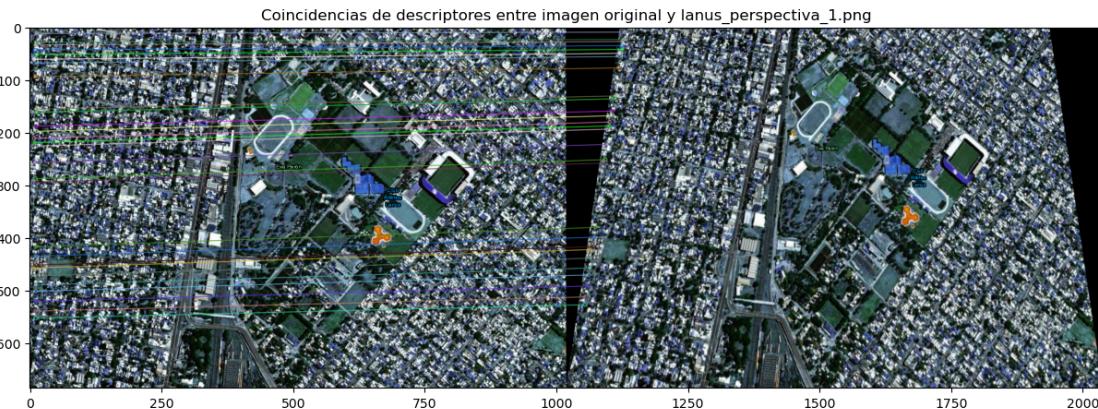
Comparación con lanus_perspectiva_0.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_perspectiva_0.png: 17562

Total de coincidencias encontradas: 8267

Las imágenes son probablemente las mismas.



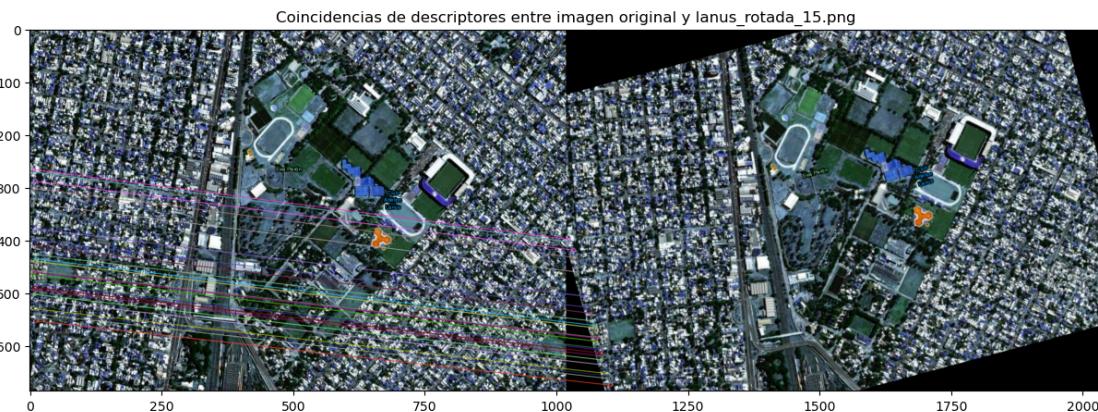
Comparación con lanus_perspectiva_1.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_perspectiva_1.png: 17577

Total de coincidencias encontradas: 8334

Las imágenes son probablemente las mismas.



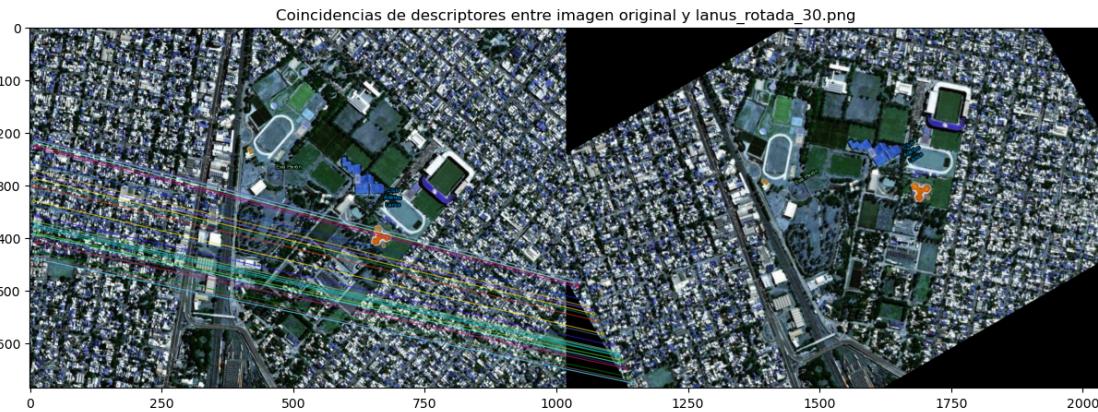
Comparación con lanus_rotada_15.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_rotada_15.png: 17218

Total de coincidencias encontradas: 10363

Las imágenes son probablemente las mismas.



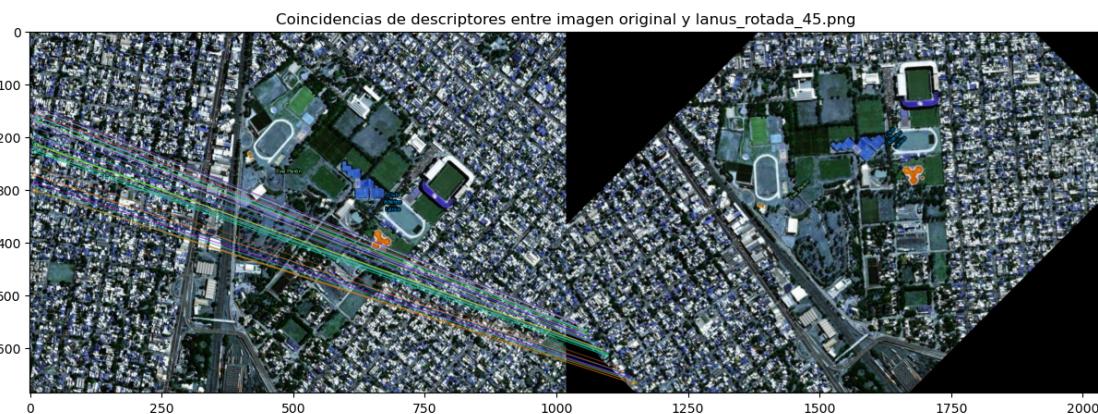
Comparación con lanus_rotada_30.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_rotada_30.png: 15778

Total de coincidencias encontradas: 9603

Las imágenes son probablemente las mismas.



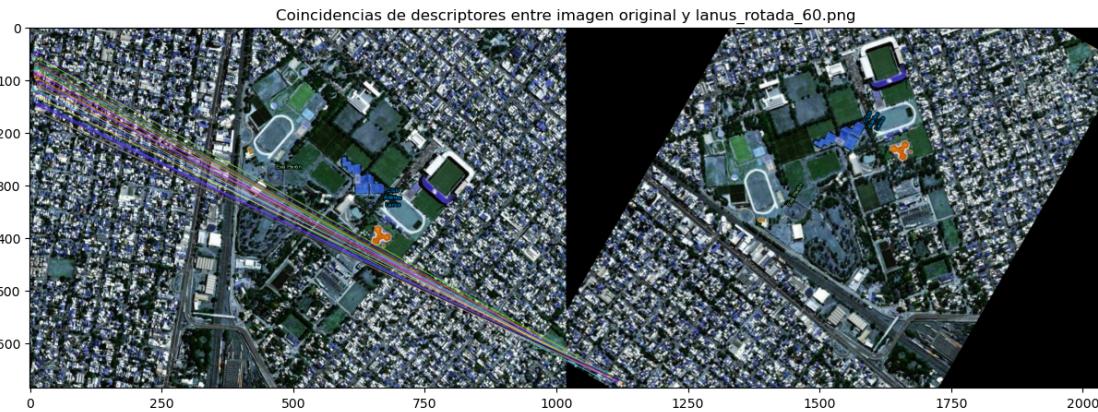
Comparación con lanus_rotada_45.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_rotada_45.png: 14971

Total de coincidencias encontradas: 9090

Las imágenes son probablemente las mismas.



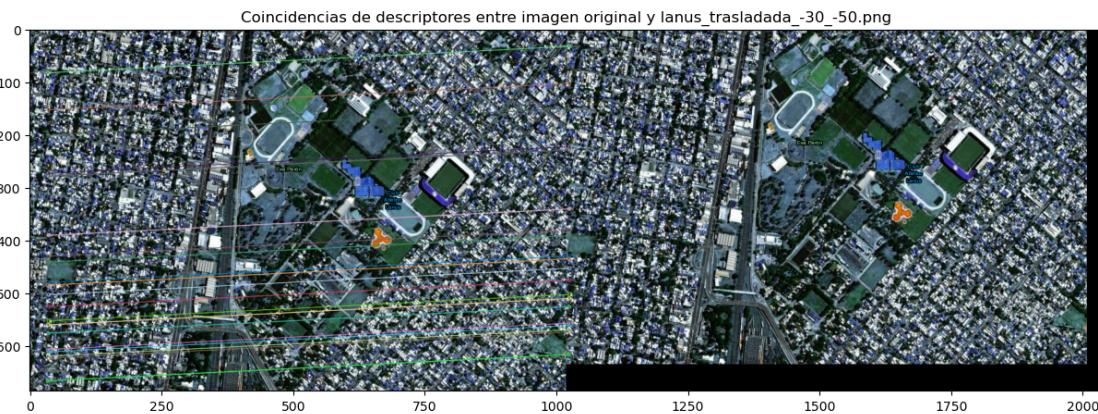
Comparación con lanus_rotada_60.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_rotada_60.png: 14360

Total de coincidencias encontradas: 8699

Las imágenes son probablemente las mismas.



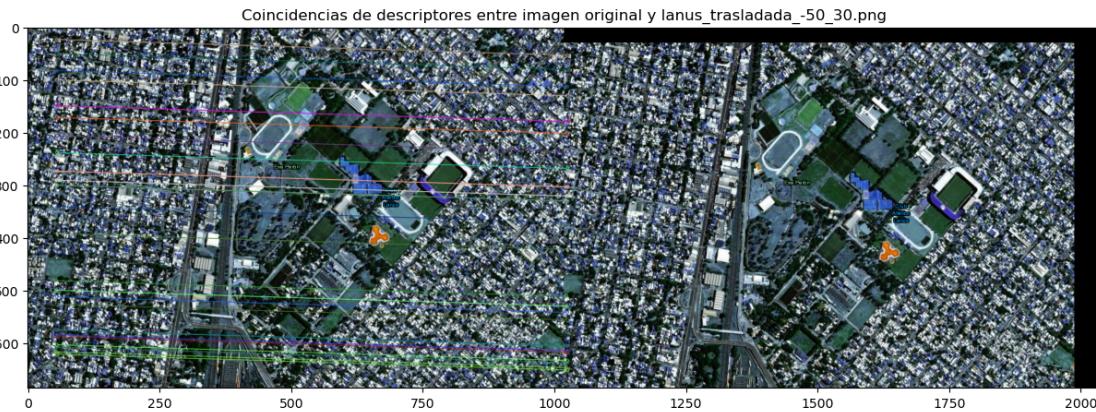
Comparación con lanus_trasladada_-30_-50.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_trasladada_-30_-50.png: 16170

Total de coincidencias encontradas: 15433

Las imágenes son probablemente las mismas.



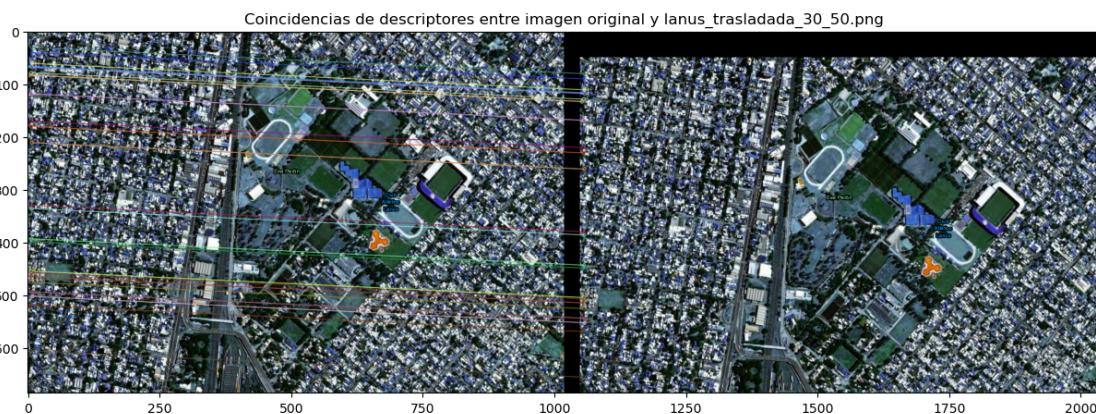
Comparación con lanus_trasladada_-50_30.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_trasladada_-50_30.png: 16398

Total de coincidencias encontradas: 15679

Las imágenes son probablemente las mismas.



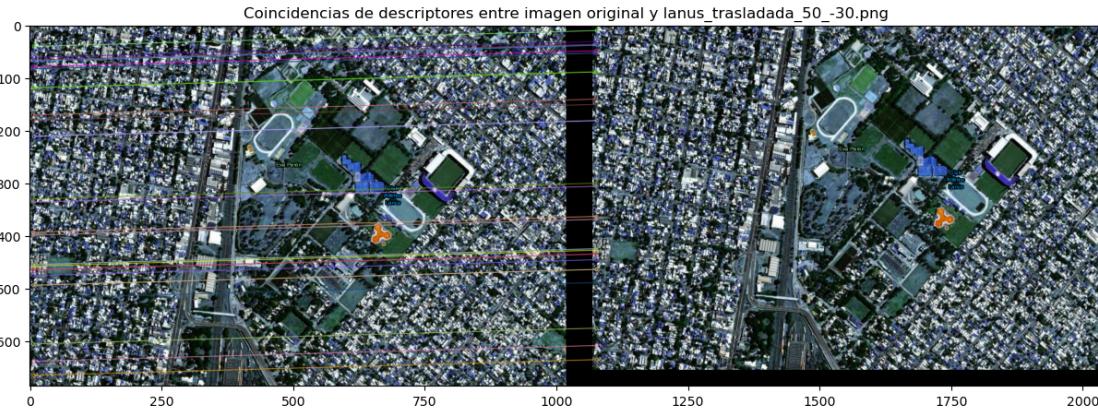
Comparación con lanus_trasladada_30_50.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_trasladada_30_50.png: 16260

Total de coincidencias encontradas: 15495

Las imágenes son probablemente las mismas.



Comparación con lanus_traslada_50_-30.png:

Número de puntos clave en la imagen original: 18025

Número de puntos clave en lanus_traslada_50_-30.png: 16400

Total de coincidencias encontradas: 15581

Las imágenes son probablemente las mismas.

0.0.3 Conclusiones

El método implementado logra detectar los patrones similares en las imágenes transformadas a partir de la original. Sin embargo, en casos de algunas modificaciones cuando la imagen sufre cambios significativos en perspectiva o resolución, la cantidad de coincidencias efectivas puede disminuir. Para los casos donde realizamos la comparación entre imágenes que no se generaron a partir de la original, se puede ver que el algoritmo encuentra pocas coincidencias entre las imágenes, y no termina generando un resultado final, a diferencia de las generadas a partir de la original.