
Universidad
Tecmilenio®

Universidad Tecmilenio

Visión computacional

Características Invariantes de escala (SIFT)

Introducción

La transformación de características invariantes de escala (**SIFT**, por sus siglas en inglés) o transformada SIFT es un algoritmo que obtiene un conjunto de características invariables a la escala y a la rotación de una imagen en particular, utilizado en distintas aplicaciones como reconocimiento de elementos o gestos en una imagen, entre otros. Su objetivo es encontrar las correspondencias de puntos en imágenes de forma automatizada.

En este tema conocerás:

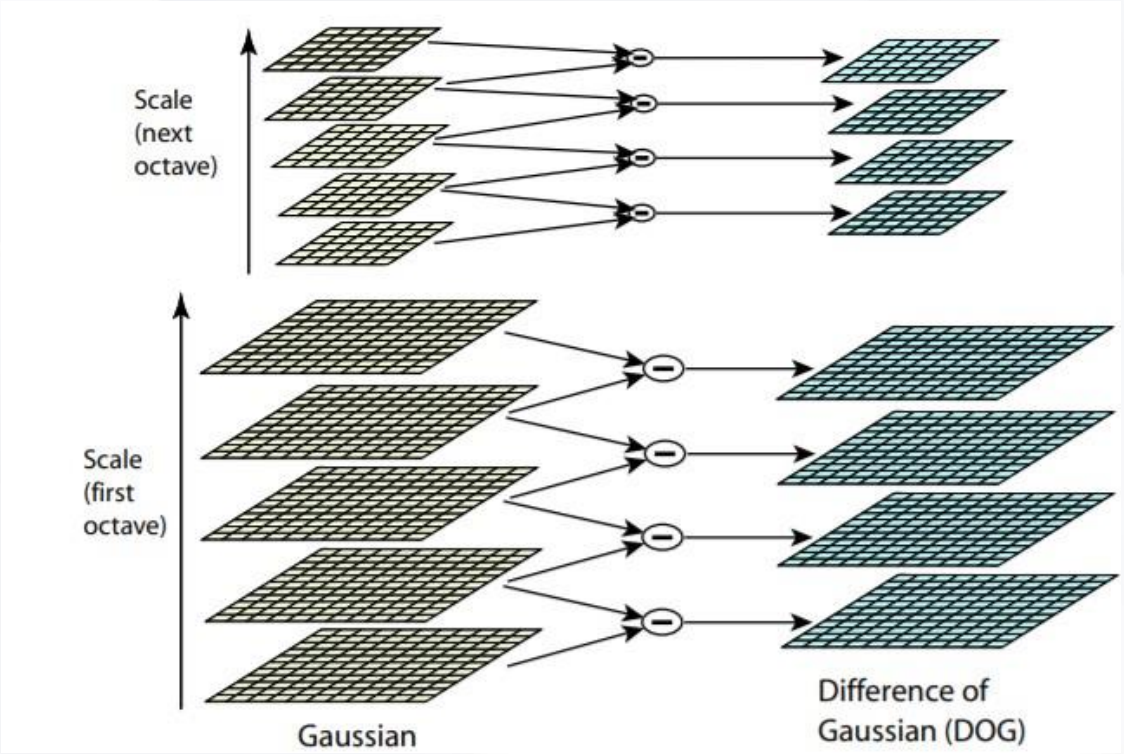
- La transformación SIFT.
- El proceso de la identificación de puntos de interés utilizando SIFT.
- El uso de OpenCV en Python para realizar detección de puntos de interés.

Explicación

El proceso para la identificación de estos puntos de interés se divide en cuatro etapas (Lowe, 2004):
Construcción del espacio de escala. Se identifican la ubicación y escalas de las características que se pueden reconocer de forma repetida desde diferentes perspectivas en el mismo objeto o escena. Se asegura que las características sean independientes a la escala.
Localización de puntos interés. Ajusta un modelo para determinar la ubicación y escalas de las características seleccionando puntos clave en función de una medida de estabilidad.
Asignación de la orientación. Se asegura de que los puntos de interés sean invariantes a la rotación. Se calculan las mejores orientaciones para cada región que contiene puntos de interés.
Descriptor del punto de interés. Asigna una "huella digital" única a cada punto de interés. Se utilizan degradados de imagen en la escala y rotación seleccionada para describir cada región que contiene puntos de interés.

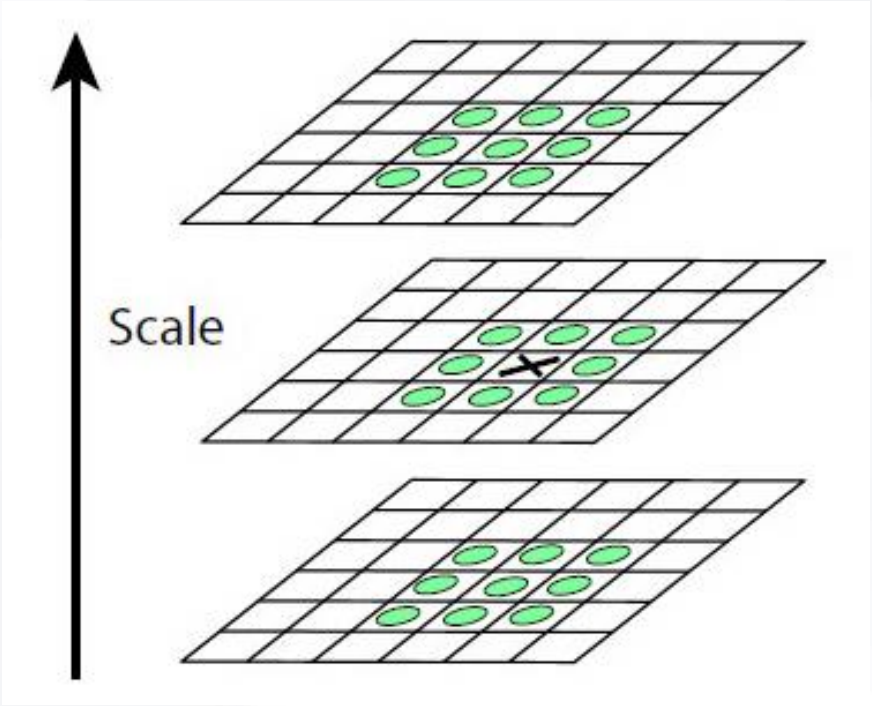
Explicación

La **construcción del espacio de escala** se hace a partir de la identificación de las características potenciales dentro de una imagen, sin tener en cuenta el ruido; es importante asegurarse que tales puntos de interés no dependan de la escala.
Una vez que se han obtenido las imágenes a escalas múltiples (distintos valores de σ) y aplicado desenfoque Gaussiano con el objetivo de reducir el ruido en la imagen, se utiliza la diferencia de Gauss (**DoG**) para mejorar la identificación de características.



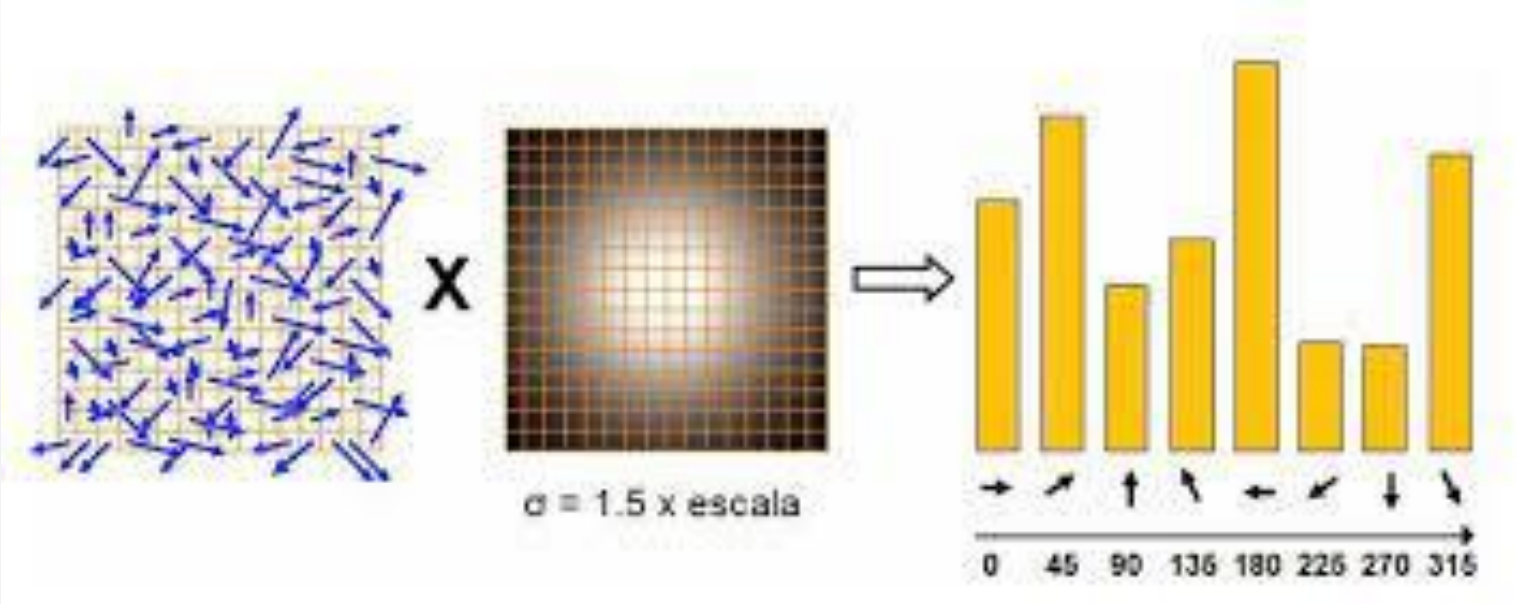
Explicación

El paso siguiente es **localizar los puntos de interés** relevantes en la imagen que puedan ser utilizados para hacer coincidir características similares. La idea es encontrar máximos y mínimos locales dentro de la imagen.
Para determinar los máximos y mínimos locales, se recorre cada píxel en la imagen y se compara con los pixeles vecinos. El vecindario utilizado no solo abarca aquellos pixeles que rodean al analizado, sino además, los 9 pixeles de las imágenes previas y siguientes en la octava, dando un total de 26 pixeles con los que se compara el píxel original con la finalidad de encontrar un mínimo y un máximo local.



Explicación

El proceso de **asignación de la orientación** se divide en dos etapas: el cálculo de la magnitud y la orientación, y la creación de un histograma para la magnitud y orientación. A partir de un vecindario alrededor del punto de interés a una escala específica, se calcula la magnitud de la gradiente (intensidad del píxel) y su dirección (orientación del píxel) en esa región. El histograma de orientación se construye con 36 barras cuyos ángulos de cobertura van de 10° en 10° cubriendo 360° y permite identificar cuántos pixeles tienen un cierto ángulo.

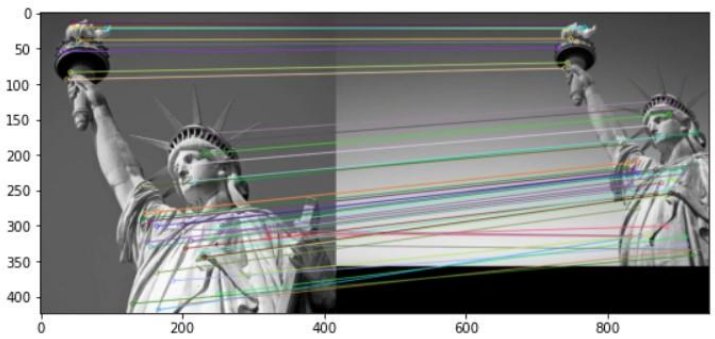


Explicación

Se utilizan los píxeles vecinos, sus orientaciones y magnitud, para generar una huella digital única para cada punto de interés llamado "descriptor". Se considera un vecindario de 16x16 pixeles alrededor del punto de interés que se subdivide en 16 subbloques de 4x4 pixeles. Se calcula un histograma de 8 barras utilizando magnitud y orientación para cada subbloque, obteniendo un total de 128 barras disponibles que se representan como un vector para formar el descriptor de puntos de interés en la imagen.

[18]:
```
# SIFT
sift = cv.xfeatures2d.SIFT_create()

kp1, descriptor1 = sift.detectAndCompute(img1, None)
kp2, descriptor2 = sift.detectAndCompute(img2, None)

# Coincidencias de caracteristicas
bf = cv.BFMatcher(cv.NORM_L1, crossCheck=True)

match1 = bf.match(descriptor1, descriptor2)
match1 = sorted(match1, key=lambda x: x.distance)

img3 = cv.drawMatches(img1, kp1, img2, kp2, match1[:50], img2, flags=2)
plt.figure(figsize=(9, 9))
plt.imshow(img3)
```

[18]:
```
<matplotlib.image.AxesImage at 0x7fe930256f28>
```



[ ]:

Actividad

Enumera por lo menos dos situaciones reales en donde creas que aplicar la transformada SIFT resolvería algún problema.

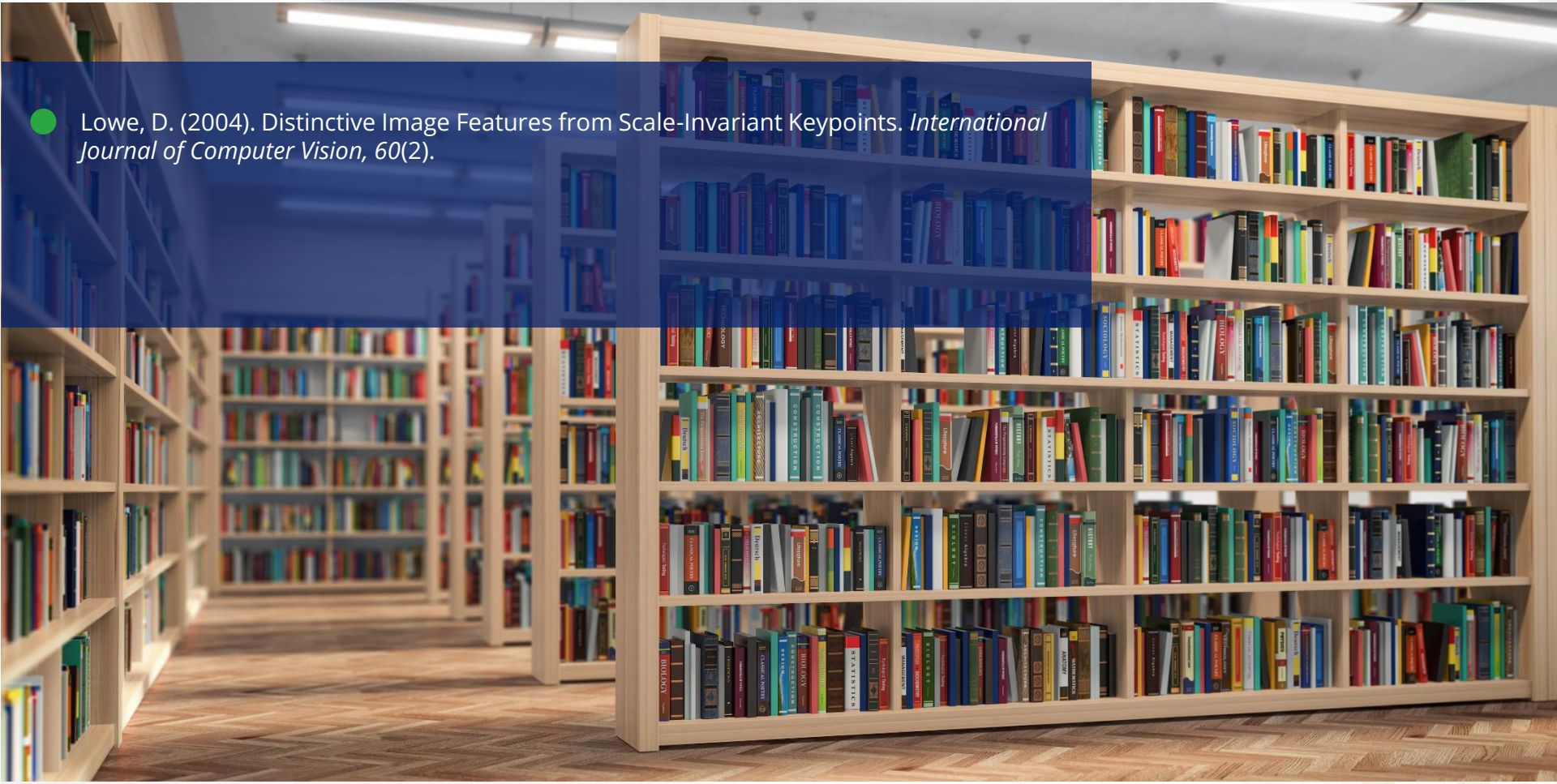
De lo anterior, investiga cómo se está resolviendo este problema actual.

Cierre

Este algoritmo permite identificar y describir características locales de imágenes digitales; localiza ciertos puntos de interés y los convierte en información cualitativa (descriptores), que son invariantes a la escala y rotación, así como a la traslación e iluminación.

Bibliografía

Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision, 60*(2).



Universidad Tecmilenio

Visión computacional

Consenso de muestra aleatoria (RANSAC)

Introducción

Es probable que algunos de los puntos encontrados por SIFT tengan una correspondencia errónea en el proceso de identificación de las coincidencias y para mejorar la identificación de estas es posible utilizar consenso de muestra aleatoria (RANSAC, por sus siglas en inglés).

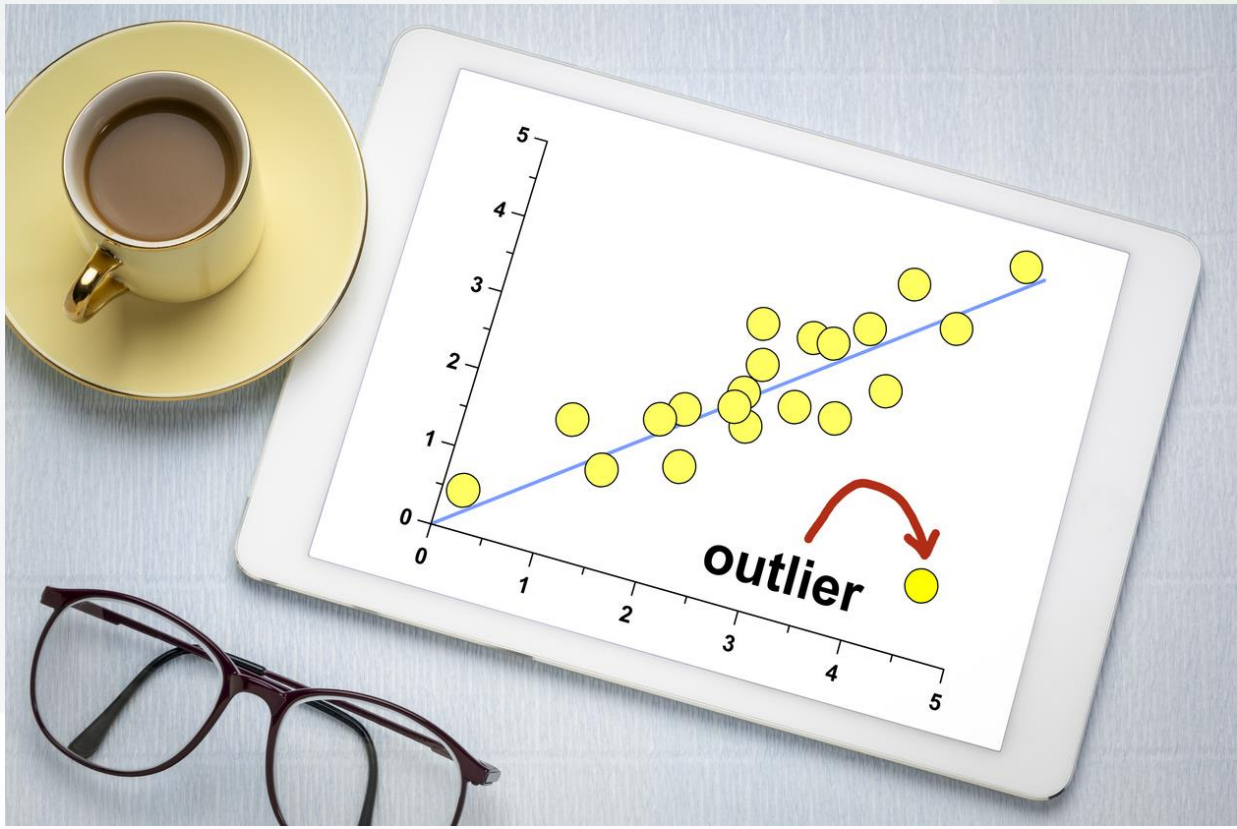
En este tema conocerás:
- El algoritmo de consenso de muestras aleatorias (RANSAC).
- La aplicación de RANSAC en una tarea de visión computacional.

Explicación

RANSAC fue propuesto en 1981 por Fischler y Bolles como un algoritmo de ajuste de modelos. Es un método iterativo para estimar parámetros de modelos, cuya distribución explica un conjunto de datos que presentan valores atípicos (*outliers*).

Una tarea común en visión computacional es intentar ajustar un modelo parametrizado a las características de la imagen, independientemente si se tienen datos que surgen de falsos positivos de un detector de características (como un borde o una esquina) o de la presencia de varios objetos en la imagen. Para resolver la tarea, el algoritmo RANSAC sigue este proceso:

- Selecciona de forma aleatoria una cierta cantidad de muestras para una estimación inicial del modelo (*inliers*, datos que se explican por un conjunto de parámetros del modelo).
- Se verifica el modelo con el resto de los datos.
- Se selecciona el mejor modelo en función de la verificación anterior.
- Finaliza según si se cumple algún criterio de iteraciones o métrico específico.

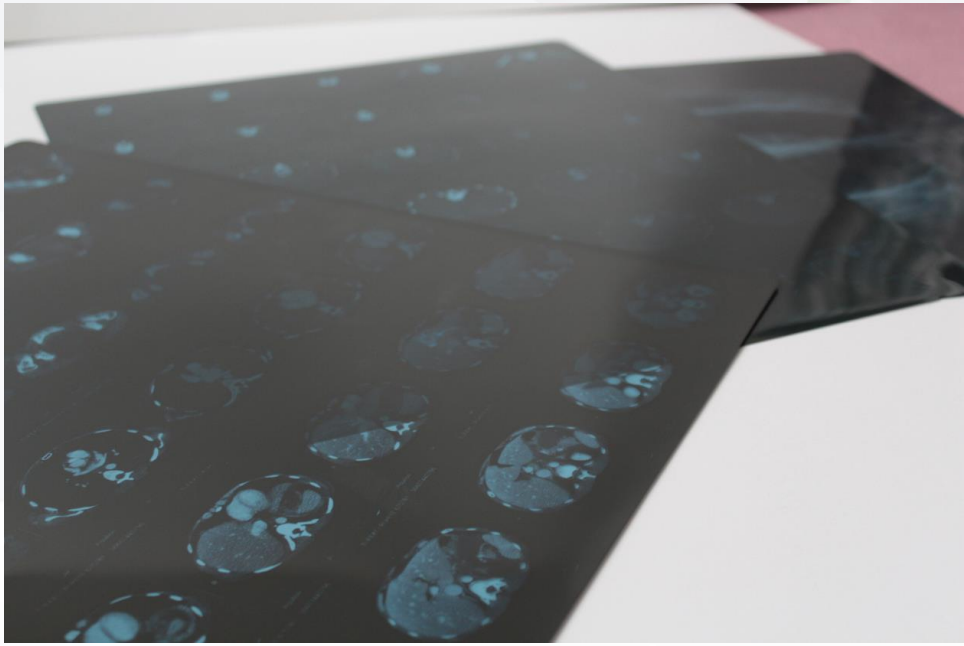


Explicación

El **registro de imágenes** es un proceso que implica sobreponer y alinear dos imágenes del mismo objeto, que pudieron tomarse en momentos distintos o desde diferentes perspectivas, y que requiere combinar la información de estas para obtener una mayor y mejor cantidad de información sobre las representaciones mostradas (Zitova, 2019). Este proceso toma cuatro pasos:

1. Detección y extracción de las características de las imágenes.
2. Establecer las coincidencias entre las características obtenidas.
3. Ajustar la función de transformación.
4. Aplicar la transformación y obtener un nuevo muestreo de la imagen.

RANSAC se utiliza en el paso 3 para encontrar la función de transformación; a partir de dos imágenes se busca la homografía (similitud) entre ambas, identificando puntos característicos comunes de forma aleatoria y con ello se construye una matriz de homografía, cuya representación final deberá tener la mayor cantidad de *inliers*.

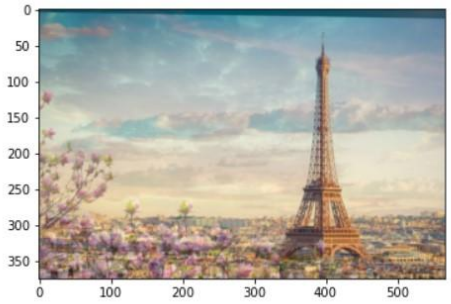


Explicación

Una de las grandes ventajas que ofrece RANSAC es la habilidad de hacer una estimación robusta de los parámetros del modelo, puede estimar parámetros con precisión a pesar de la presencia de atípicos en el conjunto de datos. En contraste, el tiempo que puede tomar esta búsqueda paramétrica puede ser muy largo, por lo que se recomienda hacer una evaluación costo-beneficio entre la cantidad de iteraciones y un modelo que razonablemente se ajuste a los datos.

Code | git | Run as Pipeline | Python

```
[7]: img_source = cv2.imread("E1.jpg")
     img_target = cv2.imread("E2.jpg")

[ ]: keypoint_source, descriptor_source = extract_SIFT(img_source)
     keypoint_target, descriptor_target = extract_SIFT(img_target)
     pos = match_SIFT(descriptor_source, descriptor_target)
     H = affine_matrix(keypoint_source, keypoint_target, pos)

[21]: rows, cols, _ = img_target.shape
      warp = cv2.warpAffine(img_source, H, (cols, rows))
      merge = np.uint8(img_target * 0.5 + warp * 0.5)

[26]: import matplotlib.pyplot as plt
      plt.imshow(cv2.cvtColor(merge, cv2.COLOR_BGR2RGB))

[26]: <matplotlib.image.AxesImage at 0x7fd4351cfcf8>
```



Actividad

En Python, carga alguna imagen en donde se identifique claramente un objeto.
Gira la imagen ciertos grados.
Utiliza el algoritmo RANSAC para encontrar la transformación afín de la imagen girada hacia la imagen original.

Cierre

La tarea de hacer coincidir puntos de interés en dos imágenes puede atenderse desde dos perspectivas: la paramétrica y la no paramétrica. Los métodos no paramétricos buscan optimizar algún métrico asociado a los pixeles para establecer la coincidencia.

Los métodos paramétricos asumen que dos imágenes están relacionadas a partir de una transformación paramétrica (transformación afín, homografía, etc.) y utilizan algoritmos para estimar esta transformación, específicamente los parámetros de esta.

RANSAC es un algoritmo que estima los parámetros de un modelo a partir de la selección aleatoria de datos de un conjunto específico.