Họ tên : Nguyễn Quang Anh

Mã sinh viên : B21DCDT036

CHƯƠNG 7 : TRÍCH XUẤT CÁC ĐẶC TRƯNG VÀ MÔ TẢ TỪ ẢNH

Trong chương này, chúng ta sẽ thảo luận về các bộ phát hiện đặc trưng (feature detectors) và mô tả đặc trưng (feature descriptors), cùng với các ứng dụng khác nhau của chúng trong xử lý ảnh. Chúng ta sẽ bắt đầu bằng cách định nghĩa các bộ phát hiện đặc trưng và mô tả đặc trưng. Sau đó, chúng ta sẽ tiếp tục với các bộ phát hiện đặc trưng phổ biến như Harris Corner, SIFT và HOG, cũng như các ứng dụng của chúng trong các bài toán quan trọng như ghép ảnh (image matching) và nhận diện đối tượng (object detection) bằng cách sử dụng thư viện **scikit-image** và **python-opencv (cv2)**.

THUẬT TOÁN 1 : Phát hiện Blob với LoG, DoG và DoH

1. Giới thiệu qua về thuật toán :

Trong một ảnh, một **blob** được định nghĩa là một vùng sáng trên nền tối, hoặc một vùng tối trên nền sáng. Trong phần này, chúng ta sẽ thảo luận về cách triển khai phát hiện đặc trưng blob trong ảnh sử dụng ba thuật toán sau đây. Ảnh đầu vào là một ảnh bướm màu (RGB).

**Laplacian of Gaussian (LoG)**

Trong Chương 3, **Tích chập và Miền Tần Số**, chúng ta đã thấy rằng việc đối chiếu ảnh với bộ lọc có thể được xem là một dạng nhận diện mẫu; tức là, so sánh một ảnh mẫu (nhỏ) (mà chúng ta muốn tìm) với tất cả các vùng nhỏ trong ảnh. Ý tưởng chính trong phát hiện blob xuất phát từ sự thật này. Chúng ta đã thấy cách bộ lọc **LoG** với điểm giao cắt bằng không có thể được sử dụng để phát hiện biên trong chương trước. LoG cũng có thể được sử dụng để tìm các vùng bất biến với tỷ lệ (scale invariant) bằng cách tìm kiếm các cực trị 3D (vị trí + tỷ lệ) của LoG với khái niệm **Scale Space**. Nếu tỷ lệ của Laplacian (σ của bộ lọc LoG) khớp với tỷ lệ của blob, độ lớn của phản ứng Laplacian đạt cực đại tại trung tâm của blob. Với phương pháp này, các ảnh đã được lọc LoG được tính toán với σ tăng dần và được xếp chồng lên nhau trong một khối. Các blob tương ứng với các cực đại cục bộ trong khối này. Phương pháp này chỉ phát hiện các blob sáng trên nền tối. Nó chính xác nhưng chậm (đặc biệt đối với việc phát hiện các blob lớn).

**Difference of Gaussian (DoG)**

Phương pháp LoG được xấp xỉ bởi phương pháp DoG, và vì vậy nó nhanh hơn. Ảnh được làm mờ (sử dụng Gaussian) với các giá trị σ tăng dần, và sự chênh lệch giữa hai ảnh đã được làm mờ liên tiếp được xếp chồng lên nhau trong một khối. Phương pháp này lại phát hiện các blob sáng trên nền tối. Nó nhanh hơn LoG nhưng kém chính xác hơn, mặc dù việc phát hiện các blob lớn vẫn còn tốn kém.

**Determinant of Hessian (DoH)**

Phương pháp **DoH** là phương pháp nhanh nhất trong số tất cả các phương pháp này. Nó phát hiện các blob bằng cách tính toán cực đại trong ma trận của **Determinant of Hessian** của ảnh. Kích thước của các blob không ảnh hưởng đến tốc độ phát hiện. Cả các blob sáng trên nền tối và các blob tối trên nền sáng đều được phát hiện bởi phương pháp này, nhưng các blob nhỏ không được phát hiện chính xác.

2,Các bước thực hiện thuật toán :

 B1: **Đọc ảnh và tiền xử lý**:

* Đọc ảnh màu **butterfly.png**.
* Nếu ảnh có 4 kênh (RGBA), loại bỏ kênh alpha để chỉ sử dụng 3 kênh màu RGB.
* Chuyển ảnh sang ảnh xám (grayscale) để thuận tiện cho việc phát hiện blob.

B2: **Phát hiện blob với các thuật toán LoG, DoG, DoH**:

* **blob\_log**: Áp dụng bộ lọc Laplacian of Gaussian để phát hiện blob.
* **blob\_dog**: Áp dụng bộ lọc Difference of Gaussian để phát hiện blob.
* **blob\_doh**: Áp dụng bộ lọc Determinant of Hessian để phát hiện blob.

B3: **Hiển thị kết quả**:

* Tạo một cửa sổ hiển thị với 4 ảnh: ảnh gốc và các ảnh phát hiện blob từ các phương pháp LoG, DoG, DoH.
* Mỗi blob được vẽ dưới dạng vòng tròn trên ảnh với màu sắc khác nhau (yellow, lime, red).

3, Giải thích đoạn mã nguồn :

from numpy import sqrt  
from skimage.feature import blob\_dog, blob\_log, blob\_doh  
from skimage.color import rgb2gray  
from skimage.io import imread  
import matplotlib.pylab as pylab  
  
# Đọc ảnh và kiểm tra số lượng kênh  
im = imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\butterfly.png")  
  
# Kiểm tra nếu ảnh có 4 kênh (RGBA) thì chuyển sang RGB  
if im.shape[-1] == 4: # Ảnh có 4 kênh  
 im = im[..., :3] # Loại bỏ kênh alpha  
  
# Chuyển đổi ảnh sang xám  
im\_gray = rgb2gray(im)  
  
# Áp dụng các thuật toán phát hiện blobs  
log\_blobs = blob\_log(im\_gray, max\_sigma=30, num\_sigma=10, threshold=0.1)  
log\_blobs[:, 2] = sqrt(2) \* log\_blobs[:, 2] # Tính bán kính  
  
dog\_blobs = blob\_dog(im\_gray, max\_sigma=30, threshold=0.1)  
dog\_blobs[:, 2] = sqrt(2) \* dog\_blobs[:, 2]  
  
doh\_blobs = blob\_doh(im\_gray, max\_sigma=30, threshold=0.005)  
  
# Danh sách blobs và thông tin hiển thị  
list\_blobs = [log\_blobs, dog\_blobs, doh\_blobs]  
colors = ['yellow', 'lime', 'red']  
titles = ['Laplacian of Gaussian', 'Difference of Gaussian', 'Determinant of Hessian']  
sequence = zip(list\_blobs, colors, titles)  
  
# Tạo các subplots  
fig, axes = pylab.subplots(2, 2, figsize=(20, 20), sharex=True, sharey=True)  
axes = axes.ravel()  
  
# Hiển thị ảnh gốc  
axes[0].imshow(im, interpolation='nearest')  
axes[0].set\_title('Original Image', size=30)  
axes[0].set\_axis\_off()  
  
# Hiển thị blobs với các phương pháp khác nhau  
for idx, (blobs, color, title) in enumerate(sequence):  
 axes[idx + 1].imshow(im, interpolation='nearest')  
 axes[idx + 1].set\_title('Blobs with ' + title, size=30)  
 for blob in blobs:  
 y, x, row = blob  
 col = pylab.Circle((x, y), row, color=color, linewidth=2, fill=False)  
 axes[idx + 1].add\_patch(col)  
 axes[idx + 1].set\_axis\_off()  
  
pylab.tight\_layout()  
pylab.subplots\_adjust(wspace=0.1, hspace=0.1) # Tăng khoảng cách giữa các subplot  
pylab.show()

**Import thư viện**

* sqrt: Được sử dụng để tính căn bậc hai.
* blob\_dog, blob\_log, blob\_doh: Các hàm từ thư viện skimage.feature dùng để phát hiện blob bằng các phương pháp DoG, LoG, và DoH.
* rgb2gray: Hàm chuyển đổi ảnh từ màu RGB sang ảnh xám (grayscale).
* imread: Hàm đọc ảnh từ tệp và trả về ảnh dưới dạng ma trận (array).
* matplotlib.pylab: Thư viện dùng để vẽ và hiển thị đồ họa.

**Đọc ảnh và kiểm tra số lượng kênh**

* Lệnh này sẽ đọc ảnh từ đường dẫn đã chỉ định. Tuy nhiên, trong đoạn mã bạn đưa ra, imread() chưa có đối số đường dẫn hợp lệ. Để chạy mã đúng, bạn cần cung cấp đường dẫn đúng vào ảnh bạn muốn xử lý.

Tiếp theo là đoạn kiểm tra xem ảnh có 4 kênh (RGBA) hay không:

* im.shape[-1]: Kiểm tra chiều cuối cùng của ma trận ảnh (tức là số kênh màu). Nếu số kênh là 4 (RGBA, trong đó có một kênh alpha), thì đoạn mã sẽ loại bỏ kênh alpha, chỉ giữ lại các kênh RGB (kênh màu đỏ, xanh lá cây và xanh dương).

**Chuyển đổi ảnh sang ảnh xám (grayscale)**

* rgb2gray(im): Chuyển đổi ảnh màu RGB sang ảnh xám (grayscale) bằng cách tính toán độ sáng trung bình của mỗi pixel.

**. Áp dụng các thuật toán phát hiện blobs**

* **Laplacian of Gaussian (LoG)**:
* blob\_log(im\_gray, ...): Phát hiện blobs trong ảnh xám sử dụng phương pháp Laplacian of Gaussian. Các tham số:
  + max\_sigma: Giá trị lớn nhất của độ rộng (σ) của bộ lọc Gaussian.
  + num\_sigma: Số lượng mức độ mờ Gaussian.
  + threshold: Ngưỡng để xác định các blobs.

Sau khi phát hiện blob, bán kính của mỗi blob được tính lại bằng cách nhân giá trị bán kính hiện tại với sqrt(2) để phù hợp với cách tính của LoG.

* **Difference of Gaussian (DoG)**:
* blob\_dog(im\_gray, ...): Phát hiện blobs bằng phương pháp Difference of Gaussian. Các tham số giống như LoG, nhưng phương pháp này xấp xỉ LoG bằng cách tính sự chênh lệch giữa các ảnh mờ.
* **Determinant of Hessian (DoH)**:
* blob\_doh(im\_gray, ...): Phát hiện blobs bằng phương pháp Determinant of Hessian, một phương pháp nhanh chóng để phát hiện blob sáng và tối trên nền sáng hoặc tối.

**Danh sách các blobs và thông tin hiển thị**

* list\_blobs: Danh sách các blobs được phát hiện từ ba phương pháp LoG, DoG và DoH.
* colors: Màu sắc để vẽ các blob (vòng tròn) cho từng phương pháp.
* titles: Tiêu đề của từng phương pháp phát hiện blob.
* sequence: Tạo một bộ ba (tuple) kết hợp blobs, màu sắc và tiêu đề để dễ dàng lặp qua khi vẽ.

**Tạo các subplots để hiển thị kết quả**

* pylab.subplots(2, 2, ...): Tạo một lưới các subplot 2x2 để hiển thị bốn hình ảnh (ảnh gốc và ba ảnh blobs).
* axes.ravel(): Chuyển đổi mảng 2D của các axes thành mảng 1D để dễ dàng xử lý trong vòng lặp.

**Hiển thị ảnh gốc**

* imshow(im): Hiển thị ảnh gốc.
* set\_title('Original Image'): Thêm tiêu đề "Original Image".
* set\_axis\_off(): Tắt hiển thị các trục toạ độ.

**Hiển thị blobs với các phương pháp khác nhau**

for idx, (blobs, color, title) in enumerate(sequence):

axes[idx + 1].imshow(im, interpolation='nearest')

axes[idx + 1].set\_title('Blobs with ' + title, size=30)

for blob in blobs:

y, x, row = blob

col = pylab.Circle((x, y), row, color=color, linewidth=2, fill=False)

axes[idx + 1].add\_patch(col)

axes[idx + 1].set\_axis\_off()

* Vòng lặp này sẽ hiển thị các blobs được phát hiện từ ba phương pháp. Với mỗi phương pháp:
  + imshow(im): Hiển thị lại ảnh gốc.
  + set\_title('Blobs with ' + title): Thêm tiêu đề cho mỗi phương pháp.
  + Vòng lặp trong mỗi phương pháp: Duyệt qua các blob, vẽ các vòng tròn (đại diện cho blob) lên ảnh.

**Cải thiện layout và hiển thị**

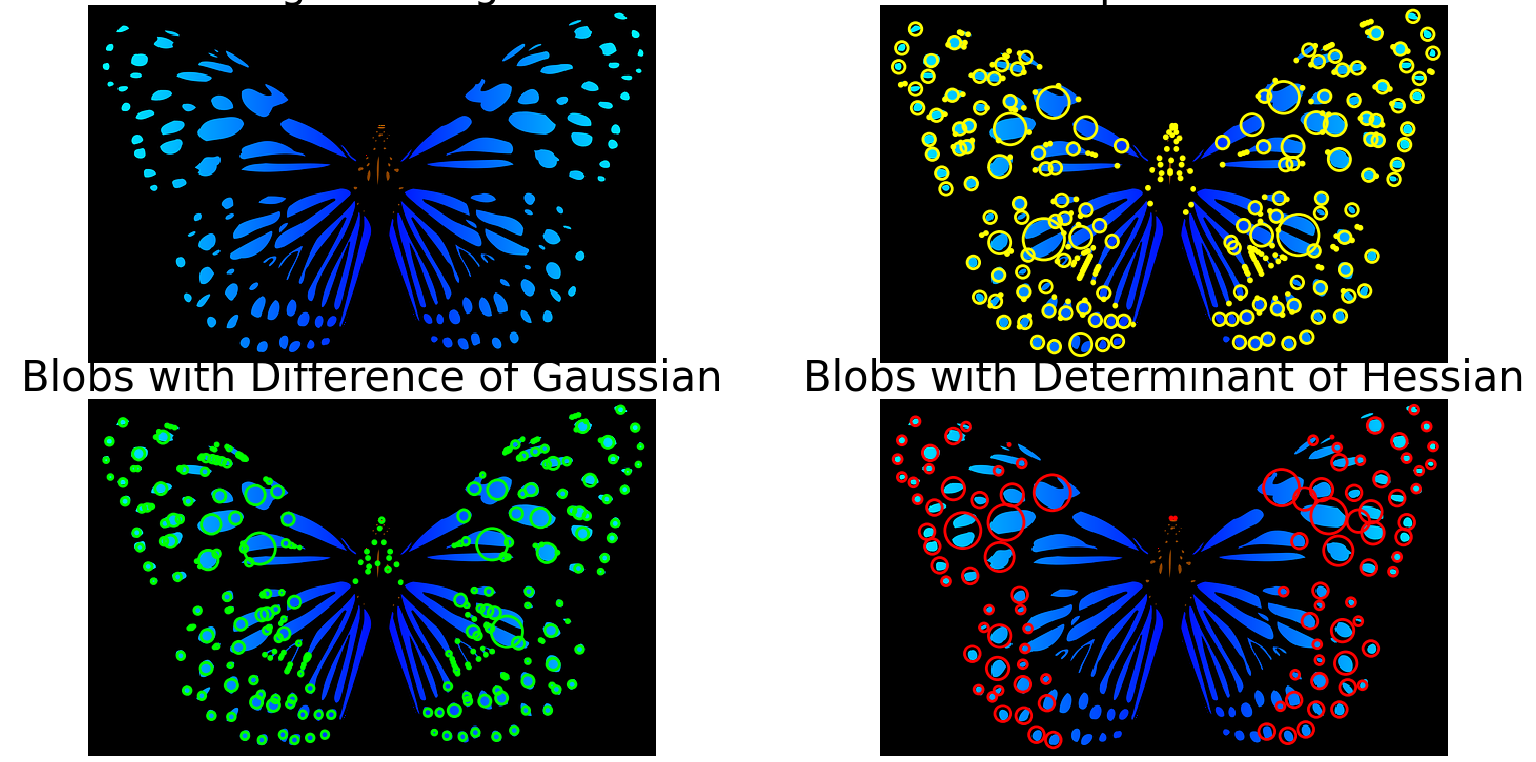
pylab.tight\_layout()

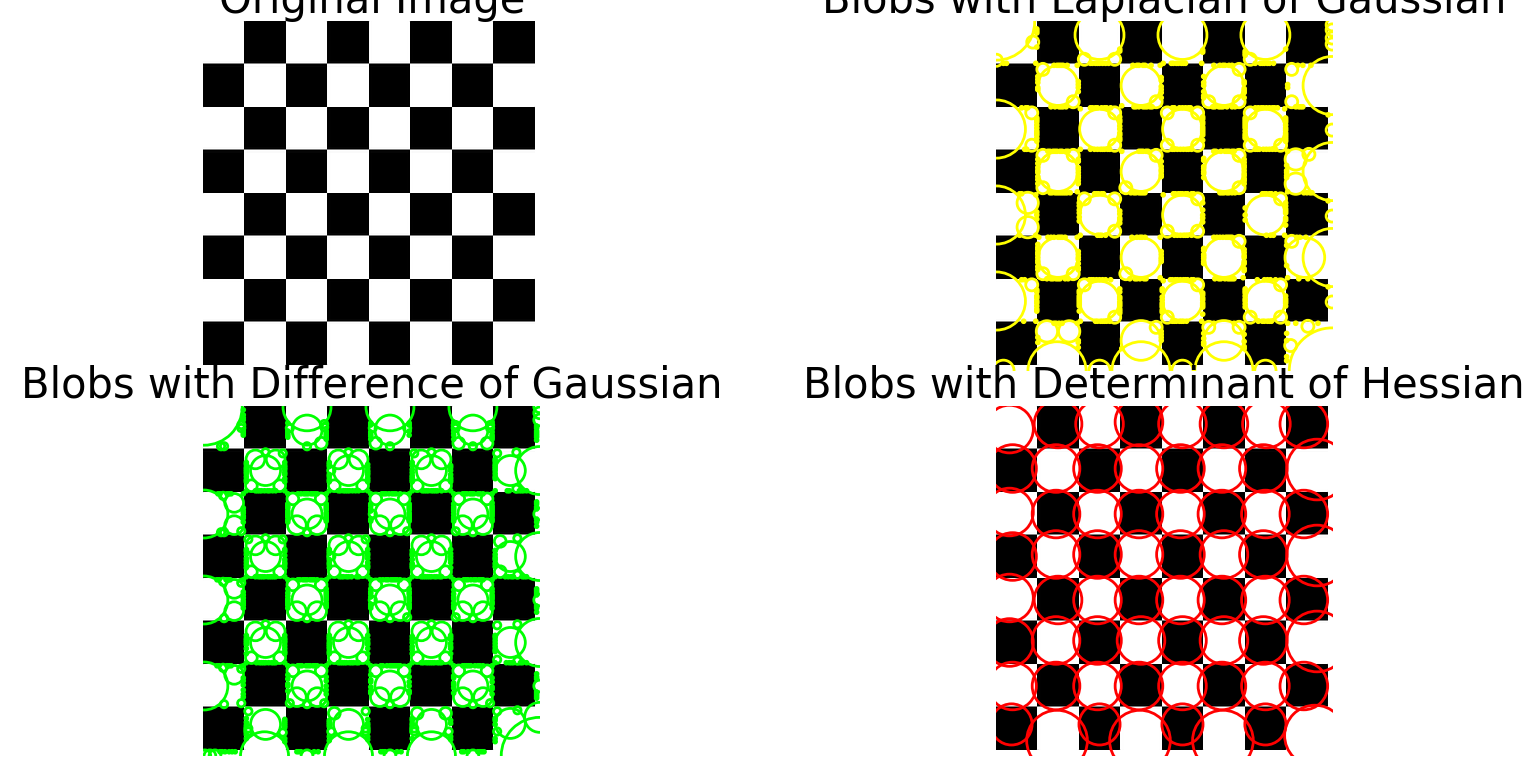
pylab.subplots\_adjust(wspace=0.1, hspace=0.1) # Tăng khoảng cách giữa các subplot

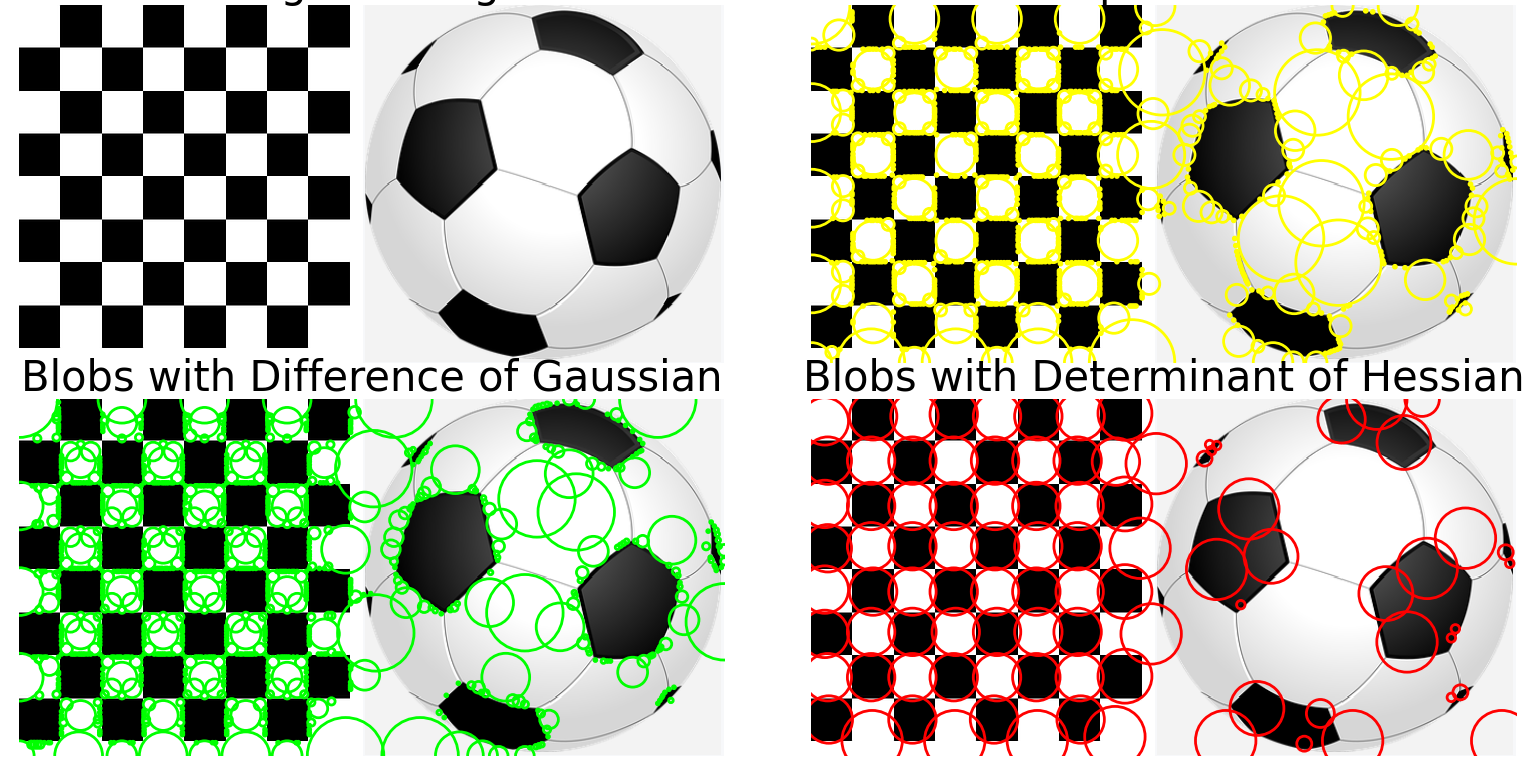
pylab.show()

* tight\_layout(): Tự động điều chỉnh layout để tránh chồng lấn giữa các subplot.
* subplots\_adjust(wspace=0.1, hspace=0.1): Điều chỉnh khoảng cách giữa các subplot theo chiều rộng và chiều cao.
* show(): Hiển thị cửa sổ hình ảnh.

4, Kết quả sau khi thực hiện :

Ví dụ 1 : 

Ví dụ 2 : 

Ví dụ 3 : 

Nhận xét ảnh đầu ra :

Đoạn mã cung cấp sử dụng ba phương pháp phát hiện blob (Laplacian of Gaussian - LoG, Difference of Gaussian - DoG, và Determinant of Hessian - DoH) để phát hiện các blob (vùng sáng/tối) trong ảnh. Các phương pháp này áp dụng trên ảnh xám (grayscale) của một bức ảnh màu (RGB).

**. Ảnh gốc**

Ảnh gốc (hiển thị ở subplot đầu tiên) sẽ là một bức ảnh màu nguyên gốc. Trong trường hợp này, vì ảnh gốc được tải về từ tệp .png mà không có kênh alpha (hoặc được xử lý nếu có), ảnh sẽ được chuyển đổi thành ảnh RGB nếu cần. Nếu ảnh có độ phân giải cao và màu sắc rõ ràng, ta sẽ thấy các đặc điểm trong ảnh dễ dàng hơn khi các blob được đánh dấu.

**Blobs với Laplacian of Gaussian (LoG)**

* **Blobs phát hiện**: LoG là phương pháp phát hiện blob đầu tiên. Các blob sẽ được hiển thị bằng vòng tròn màu **vàng**.
* **Đặc điểm**: LoG rất hiệu quả trong việc phát hiện các blob sáng trên nền tối, và sử dụng các mức sigma để tìm kiếm blobs trên nhiều kích thước khác nhau.
* **Hiệu suất**: LoG có thể chậm khi xử lý các blob lớn, nhưng nó rất chính xác trong việc xác định blob sáng trên nền tối. Hình ảnh sẽ có các vòng tròn vàng xung quanh các khu vực nổi bật hoặc các đặc trưng mạnh trong ảnh. **Blobs với Difference of Gaussian (DoG)**
* **Blobs phát hiện**: DoG là phương pháp thứ hai được áp dụng. Các blob sẽ được hiển thị bằng vòng tròn màu **xanh lá cây**.
* **Đặc điểm**: DoG là sự xấp xỉ của LoG, nhanh hơn và đơn giản hơn, nhưng độ chính xác có thể không cao như LoG, đặc biệt đối với các blob lớn. Tuy nhiên, nó vẫn rất hữu ích trong việc phát hiện các đặc trưng vùng sáng trong ảnh.
* **Hiệu suất**: DoG nhanh hơn LoG, nhưng nó có thể bị thiếu sót khi phát hiện những blob rất nhỏ hoặc rất lớn, vì sự chênh lệch Gaussian được tính toán không hoàn toàn chính xác như LoG.

**Blobs với Determinant of Hessian (DoH)**

* **Blobs phát hiện**: DoH là phương pháp thứ ba. Các blob sẽ được hiển thị bằng vòng tròn màu **đỏ**.
* **Đặc điểm**: DoH được sử dụng để phát hiện cả blob sáng và tối. Nó nhanh hơn LoG và DoG và phát hiện các blob sáng/tối trên nền sáng/tối rất hiệu quả. Tuy nhiên, phương pháp này không chính xác trong việc phát hiện các blob rất nhỏ.
* **Hiệu suất**: DoH là phương pháp nhanh nhất trong ba phương pháp, và có thể phát hiện các blob rất hiệu quả mà không bị ảnh hưởng bởi kích thước blob.

**. Nhận xét tổng quan**

* **Các vòng tròn đại diện cho blob**: Trong mỗi subplot, các blob sẽ được thể hiện dưới dạng vòng tròn vẽ trên ảnh gốc. Màu sắc của các vòng tròn tương ứng với các phương pháp khác nhau.
  + **Vòng tròn vàng** (LoG): Dự đoán chính xác các blob sáng trên nền tối.
  + **Vòng tròn xanh lá cây** (DoG): Phát hiện các blob sáng nhưng có thể thiếu chính xác ở các blob lớn.
  + **Vòng tròn đỏ** (DoH): Phát hiện các blob sáng và tối, tuy nhiên có thể bỏ qua những blob nhỏ.
* **Khoảng cách giữa các subplot**: Nhờ vào tight\_layout() và subplots\_adjust(), các subplot không bị chồng lấn và dễ dàng quan sát từng kết quả từ ba phương pháp.
* **Kết quả phát hiện blob**: Các blob được phát hiện bởi mỗi phương pháp có thể sẽ khác nhau về độ chính xác và phạm vi kích thước blob mà chúng phát hiện. LoG có thể phát hiện được các blob sáng với độ chính xác cao, trong khi DoG và DoH có thể nhận diện các blob nhanh hơn nhưng với ít chi tiết hơn.

THUẬT TOÁN 2 : KHỚP CÁC HÌNH ẢNH VỚI ĐĂC TRƯNG NHỊ PHÂN (BRIEF) VỚI SCIKIT-IMAGE

**1.Giới thiệu về thuật toán BRIEF**

BRIEF là một phương pháp mô tả đặc trưng hình ảnh dựa trên các phép toán so sánh độ sáng giữa các điểm ảnh trong vùng lân cận của các điểm góc (corner points). Các đặc trưng BRIEF được biểu diễn dưới dạng nhị phân, giúp tiết kiệm bộ nhớ và cho phép việc khớp đặc trưng trở nên nhanh chóng. Tuy nhiên, BRIEF không thể cung cấp tính bất biến xoay (rotation-invariance) mặc dù có thể đạt được tính bất invariance tỷ lệ thông qua việc phát hiện đặc trưng ở các tỷ lệ khác nhau.

**2.Các bước thực hiện thuật toán**

**B1 :Đọc ảnh và chuyển sang ảnh xám**:

* + Đọc hình ảnh đầu vào và chuyển từ ảnh màu sang ảnh xám (grayscale) để giảm độ phức tạp tính toán.

**B2:Áp dụng phép biến đổi affine và xoay ảnh**:

* + Để mô phỏng các thay đổi hình học, ảnh gốc được biến đổi affine (thay đổi tỷ lệ và dịch chuyển) và xoay một góc 25 độ.

**B3: Phát hiện các điểm góc (corner detection)**:

* + Phương pháp Harris được sử dụng để phát hiện các điểm góc trên mỗi ảnh.

**B4: Trích xuất đặc trưng BRIEF**:

* + Sau khi phát hiện các điểm góc, BRIEF được sử dụng để trích xuất đặc trưng nhị phân từ các điểm này.

**B5: Khớp các đặc trưng**:

* + Sau khi trích xuất các đặc trưng nhị phân, chúng được so sánh và khớp với nhau bằng cách sử dụng khoảng cách Hamming, giúp xác định sự tương đồng giữa các điểm đặc trưng.

**B6: Hiển thị kết quả khớp**:

* + Cuối cùng, các đặc trưng khớp được hiển thị trên các ảnh đầu vào, để có thể so sánh các đặc trưng giữa các ảnh gốc và ảnh biến đổi.

### **3.Giải thích các câu lệnh mã nguồn**

from skimage import transform as transform  
from skimage.feature import (match\_descriptors, corner\_peaks,  
 corner\_harris, plot\_matches, BRIEF)  
from skimage.io import imread  
from skimage.color import rgb2gray  
import matplotlib.pylab as pylab  
  
# Đọc ảnh  
img1 = rgb2gray(imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\me18.jpg"))  
  
# Áp dụng phép biến đổi affine và xoay  
affine\_trans = transform.AffineTransform(scale=(1.2, 1.2), translation=(0, -100))  
img2 = transform.warp(img1, affine\_trans)  
img3 = transform.rotate(img1, 25)  
  
# Phát hiện góc bằng phương pháp Harris  
coords1, coords2, coords3 = corner\_harris(img1), corner\_harris(img2), corner\_harris(img3)  
coords1[coords1 > 0.01 \* coords1.max()] = 1  
coords2[coords2 > 0.01 \* coords2.max()] = 1  
coords3[coords3 > 0.01 \* coords3.max()] = 1  
  
# Lọc các góc có thể chấp nhận  
keypoints1 = corner\_peaks(coords1, min\_distance=5)  
keypoints2 = corner\_peaks(coords2, min\_distance=5)  
keypoints3 = corner\_peaks(coords3, min\_distance=5)  
  
# Sử dụng BRIEF để trích xuất đặc trưng  
extractor = BRIEF()  
extractor.extract(img1, keypoints1)  
keypoints1, descriptors1 = keypoints1[extractor.mask], extractor.descriptors  
extractor.extract(img2, keypoints2)  
keypoints2, descriptors2 = keypoints2[extractor.mask], extractor.descriptors  
extractor.extract(img3, keypoints3)  
keypoints3, descriptors3 = keypoints3[extractor.mask], extractor.descriptors  
  
# Khớp các mô tả từ ảnh gốc và ảnh đã biến đổi  
matches12 = match\_descriptors(descriptors1, descriptors2, cross\_check=True)  
matches13 = match\_descriptors(descriptors1, descriptors3, cross\_check=True)  
  
# Vẽ kết quả khớp các đặc trưng  
fig, axes = pylab.subplots(nrows=2, ncols=1, figsize=(20, 20))  
pylab.gray()  
plot\_matches(axes[0], img1, img2, keypoints1, keypoints2, matches12)  
axes[0].axis('off')  
axes[0].set\_title("Original Image vs. Transformed Image")  
plot\_matches(axes[1], img1, img3, keypoints1, keypoints3, matches13)  
axes[1].axis('off')  
axes[1].set\_title("Original Image vs. Rotated Image")  
pylab.show()

**Đọc và chuyển ảnh sang xám**:

img1 = rgb2gray(imread(r"C:\Users\Dell\Desktop\Sandipan\_Dey\_2018\_Sample\_Images\images\me18.jpg"))

Đọc ảnh từ đường dẫn chỉ định và chuyển sang ảnh xám bằng hàm rgb2gray.

**Áp dụng phép biến đổi affine**:

affine\_trans = transform.AffineTransform(scale=(1.2, 1.2), translation=(0, -100))

img2 = transform.warp(img1, affine\_trans)

Dùng AffineTransform để thay đổi tỷ lệ ảnh và dịch chuyển nó.

**Áp dụng phép xoay ảnh**:

img3 = transform.rotate(img1, 25)

Xoay ảnh gốc 25 độ.

**Phát hiện các điểm góc**:

coords1, coords2, coords3 = corner\_harris(img1), corner\_harris(img2), corner\_harris(img3)

Phương pháp Harris được sử dụng để phát hiện các điểm góc trong các ảnh gốc và ảnh biến đổi.

**Lọc các điểm góc có thể chấp nhận**:

keypoints1 = corner\_peaks(coords1, min\_distance=5)

keypoints2 = corner\_peaks(coords2, min\_distance=5)

keypoints3 = corner\_peaks(coords3, min\_distance=5)

Chọn các điểm góc có độ mạnh cao và cách nhau ít nhất 5 pixel.

**Trích xuất đặc trưng BRIEF**:

extractor = BRIEF()

extractor.extract(img1, keypoints1)

keypoints1, descriptors1 = keypoints1[extractor.mask], extractor.descriptors

Trích xuất các đặc trưng nhị phân BRIEF từ các điểm góc trong ảnh. extractor.mask là một chỉ số các điểm góc hợp lệ.

**Khớp các đặc trưng**:

matches12 = match\_descriptors(descriptors1, descriptors2, cross\_check=True)

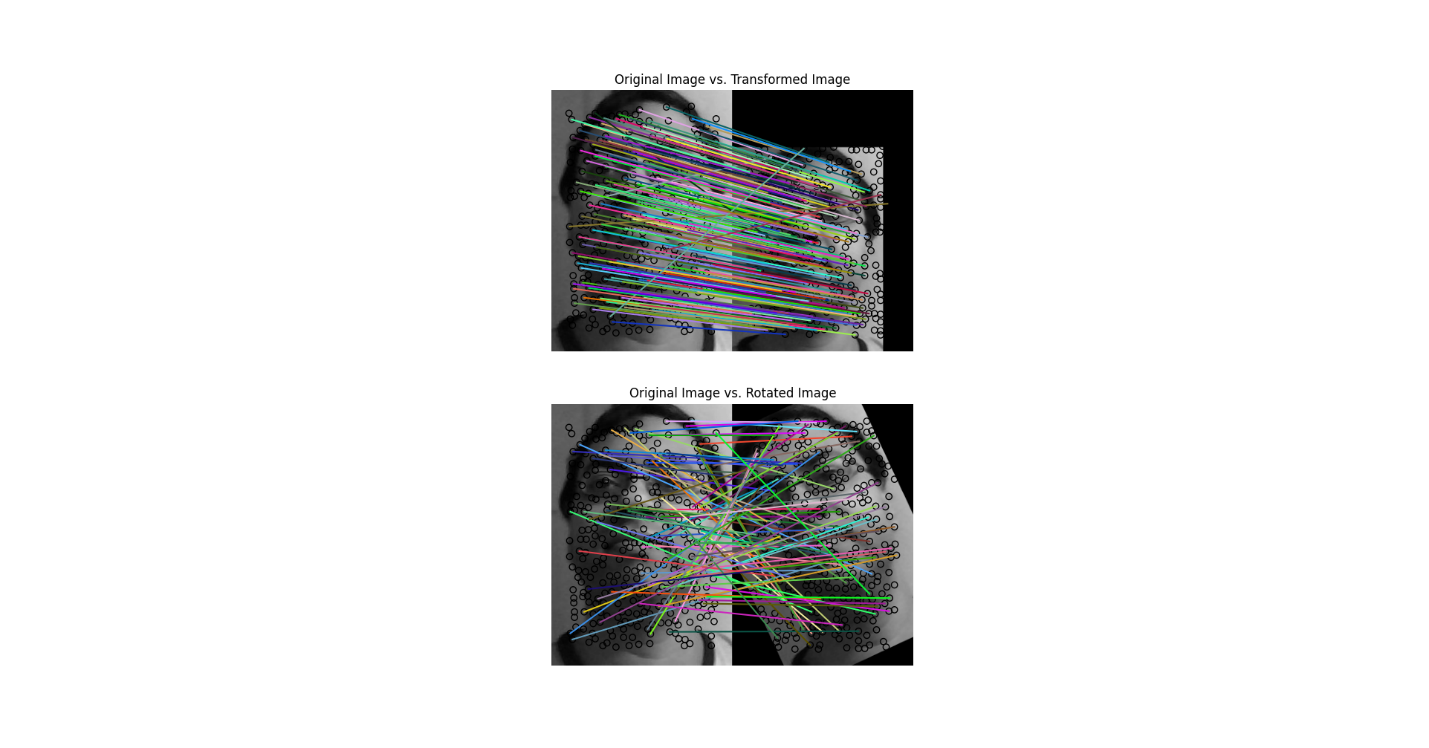
Dùng hàm match\_descriptors để khớp các đặc trưng giữa hai ảnh.

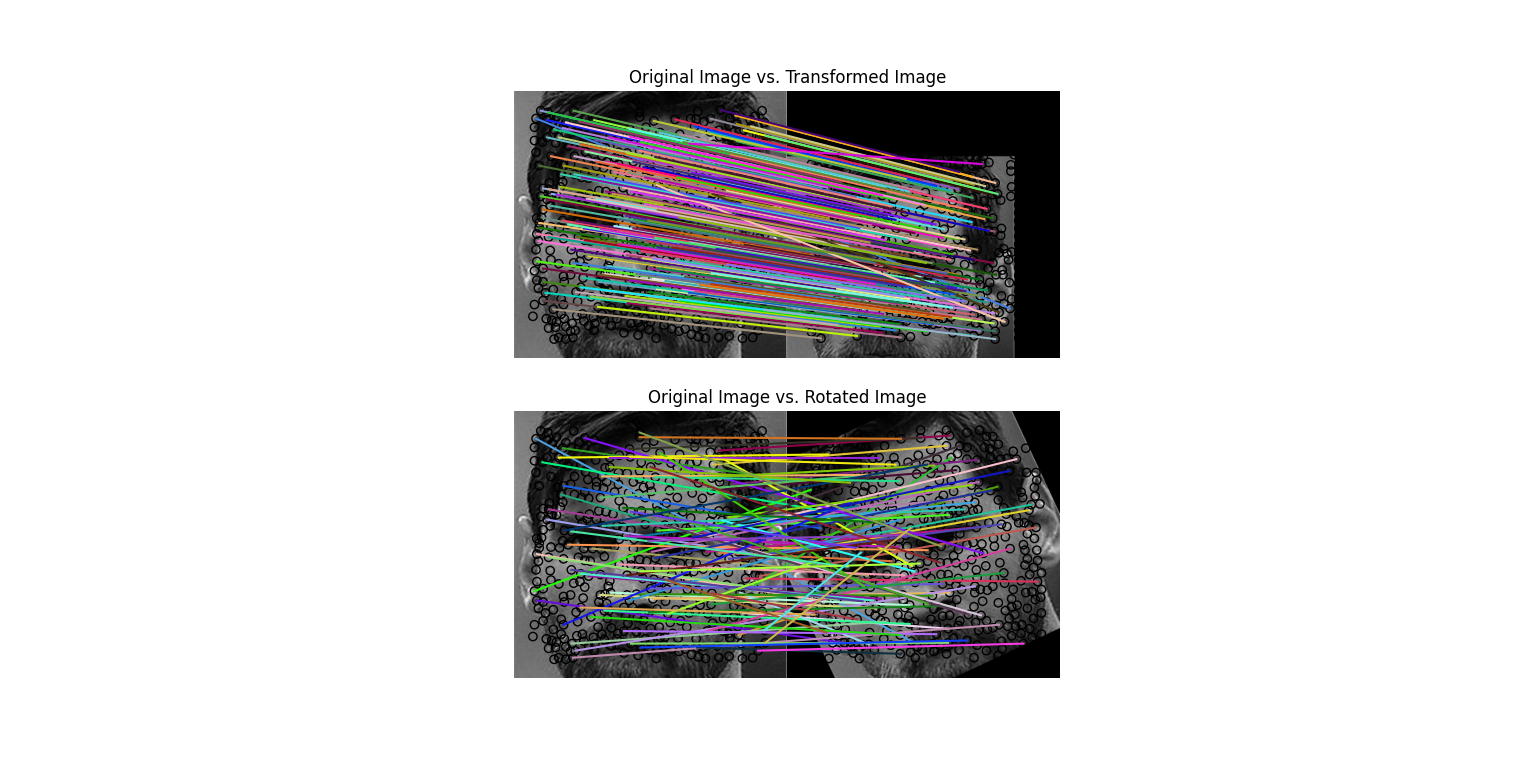
**Hiển thị kết quả**:

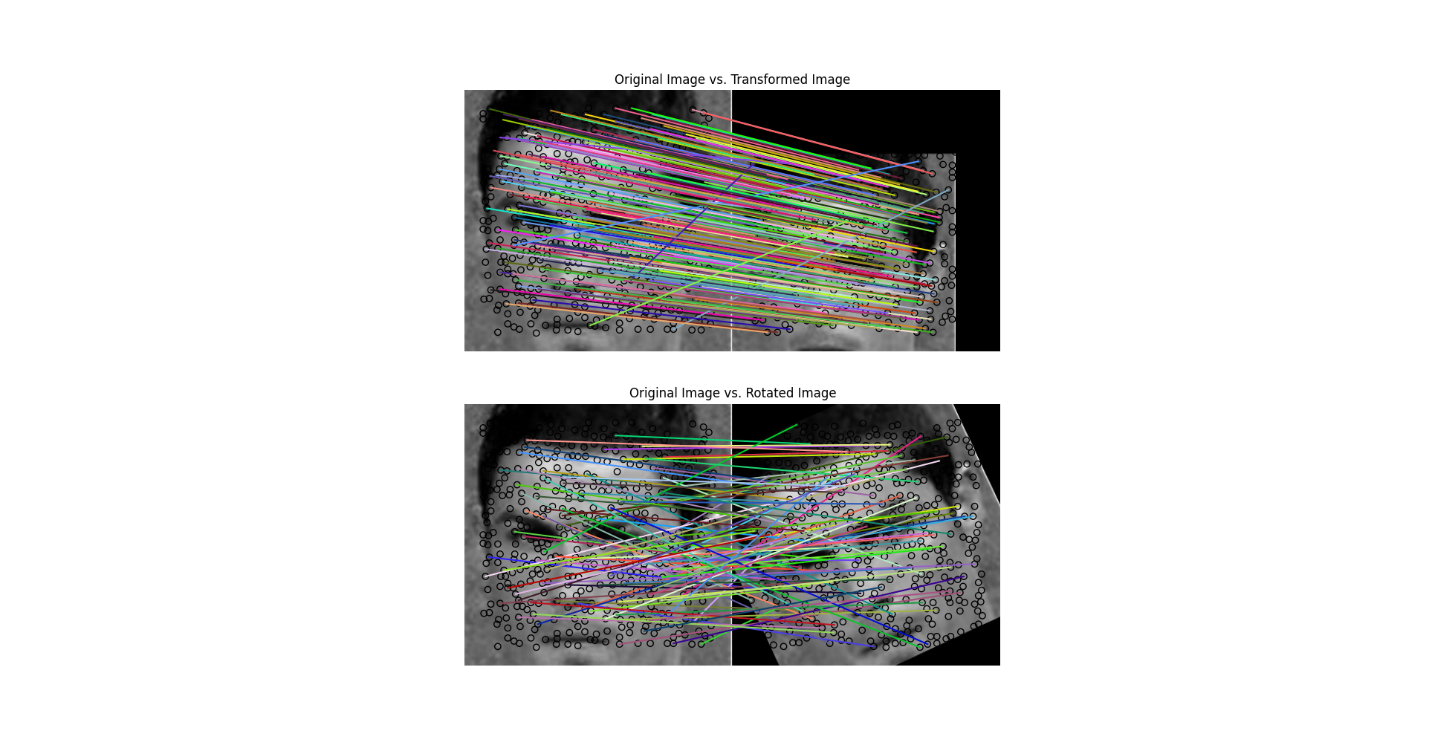
plot\_matches(axes[0], img1, img2, keypoints1, keypoints2, matches12)

Hiển thị các điểm đặc trưng khớp trên các ảnh gốc và biến đổi.

**4.Kết quả sau khi thực hiện :**

Ví dụ 1 : 

Ví dụ 2 : 

Ví dụ 3 : 

**Nhận xét về ảnh đầu ra**

* **Khớp ảnh gốc và ảnh đã biến đổi affine**:
  + Trong ảnh đầu tiên, bạn sẽ thấy các điểm đặc trưng BRIEF (được thể hiện bằng các dấu chấm) được khớp giữa ảnh gốc và ảnh đã biến đổi affine. Những điểm này thể hiện các điểm tương đồng mặc dù ảnh đã bị thay đổi tỷ lệ và dịch chuyển.
* **Khớp ảnh gốc và ảnh đã xoay**:
  + Trong ảnh thứ hai, bạn sẽ thấy các điểm đặc trưng BRIEF được khớp giữa ảnh gốc và ảnh đã xoay. Mặc dù BRIEF không thể xử lý sự thay đổi về xoay (vì nó không có tính bất invariance xoay), nhưng vẫn có thể thấy rằng các điểm khớp tốt ở những vùng có các đặc trưng mạnh, và BRIEF giúp giảm thiểu lỗi khớp nhờ tính toán nhanh chóng của mình.