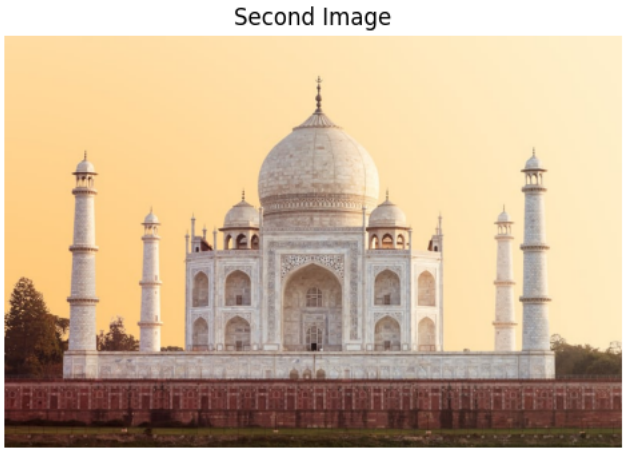
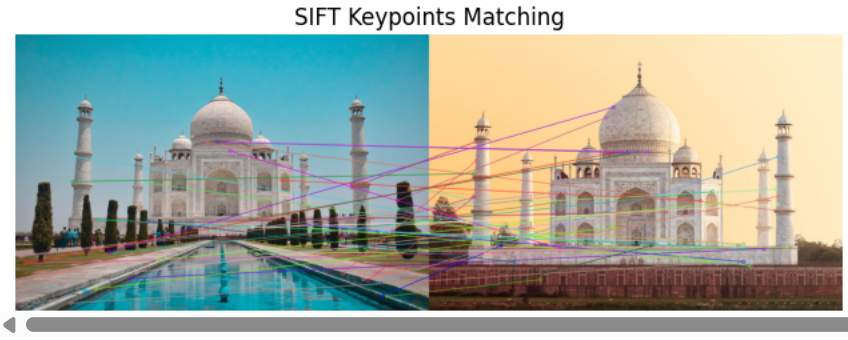
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt  # Import matplotlib

def detect_and_match_sift(image1, image2):
    # Convert both images to grayscale
    gray_img1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
    gray_img2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

    # Initialize the SIFT detector
    sift = cv2.SIFT_create()

    # Detect keypoints and compute descriptors
    kp1, des1 = sift.detectAndCompute(gray_img1, None)
    kp2, des2 = sift.detectAndCompute(gray_img2, None)

    # Use a Brute Force Matcher to match descriptors
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(des1, des2)

    # Sort matches based on distance (best matches first)
    matches = sorted(matches, key = lambda x: x.distance)

    # Draw matches on the images
    matched_img = cv2.drawMatches(image1, kp1, image2, kp2, matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

    return matched_img


# Load the two images
image1 = cv2.imread('taj1.jpg')
image2 = cv2.imread('taj2.jpg')

if image1 is None or image2 is None:
    print("Error: Could not open or find one or both of the images.")
else:
    # Display the first image with title
    plt.figure(figsize=(6, 4))  # Adjust figure size if needed
    plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.title("First Image")
    plt.axis('off')  # Turn off axis ticks and labels
    plt.show()

    # Display the second image with title
    plt.figure(figsize=(6, 4))
    plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
    plt.title("Second Image")
    plt.axis('off')
    plt.show()

    # Call the SIFT keypoints detection and matching function
    matched_img = detect_and_match_sift(image1, image2)

    # Display the result with matches
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(matched_img, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.title("SIFT Keypoints Matching")
    plt.axis('off')
    plt.show()
```


```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt  # Import matplotlib

def detect_and_match_sift(image1, image2):
    # Convert both images to grayscale
    gray_img1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
    gray_img2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

    # Initialize the SIFT detector
    sift = cv2.SIFT_create()

    # Detect keypoints and compute descriptors
    kp1, des1 = sift.detectAndCompute(gray_img1, None)
    kp2, des2 = sift.detectAndCompute(gray_img2, None)

    # Use a Brute Force Matcher to match descriptors
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(des1, des2)

    # Sort matches based on distance (best matches first)
    matches = sorted(matches, key = lambda x: x.distance)

    # Draw matches on the images
    matched_img = cv2.drawMatches(image1, kp1, image2, kp2, matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

    return matched_img
```

```
# Load the two images
image1 = cv2.imread('darkknight1.jpg')
image2 = cv2.imread('darkknight2.jpg')

if image1 is None or image2 is None:
    print("Error: Could not open or find one or both of the images.")
else:
    # Display the first image with title
    plt.figure(figsize=(6, 4))  # Adjust figure size if needed
    plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.title("First Image")
    plt.axis('off')  # Turn off axis ticks and labels
    plt.show()

    # Display the second image with title
    plt.figure(figsize=(6, 4))
    plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
    plt.title("Second Image")
    plt.axis('off')
    plt.show()

    # Call the SIFT keypoints detection and matching function
    matched_img = detect_and_match_sift(image1, image2)

    # Display the result with matches
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(matched_img, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.title("SIFT Keypoints Matching")
    plt.axis('off')
    plt.show()
```

First Image


Second Image


SIFT Keypoints Matching


```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt  # Import matplotlib

def detect_and_match_sift(image1, image2):
    # Convert both images to grayscale
    gray_img1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
    gray_img2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
```

```
    # Initialize the SIFT detector
    sift = cv2.SIFT_create()

    # Detect keypoints and compute descriptors
    kp1, des1 = sift.detectAndCompute(gray_img1, None)
    kp2, des2 = sift.detectAndCompute(gray_img2, None)

    # Use a Brute Force Matcher to match descriptors
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
    matches = bf.match(des1, des2)

    # Sort matches based on distance (best matches first)
    matches = sorted(matches, key = lambda x: x.distance)

    # Draw matches on the images
    matched_img = cv2.drawMatches(image1, kp1, image2, kp2, matches[:50], None, flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

    return matched_img


# Load the two images
image1 = cv2.imread('eiffel1.jpg')
image2 = cv2.imread('eiffel2.jpg')

if image1 is None or image2 is None:
    print("Error: Could not open or find one or both of the images.")
else:
    # Display the first image with title
    plt.figure(figsize=(6, 4))  # Adjust figure size if needed
    plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.title("First Image")
    plt.axis('off')  # Turn off axis ticks and labels
    plt.show()

    # Display the second image with title
    plt.figure(figsize=(6, 4))
    plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
    plt.title("Second Image")
    plt.axis('off')
    plt.show()

    # Call the SIFT keypoints detection and matching function
    matched_img = detect_and_match_sift(image1, image2)

    # Display the result with matches
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(matched_img, cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for matplotlib
    plt.title("SIFT Keypoints Matching")
    plt.axis('off')
    plt.show()
```