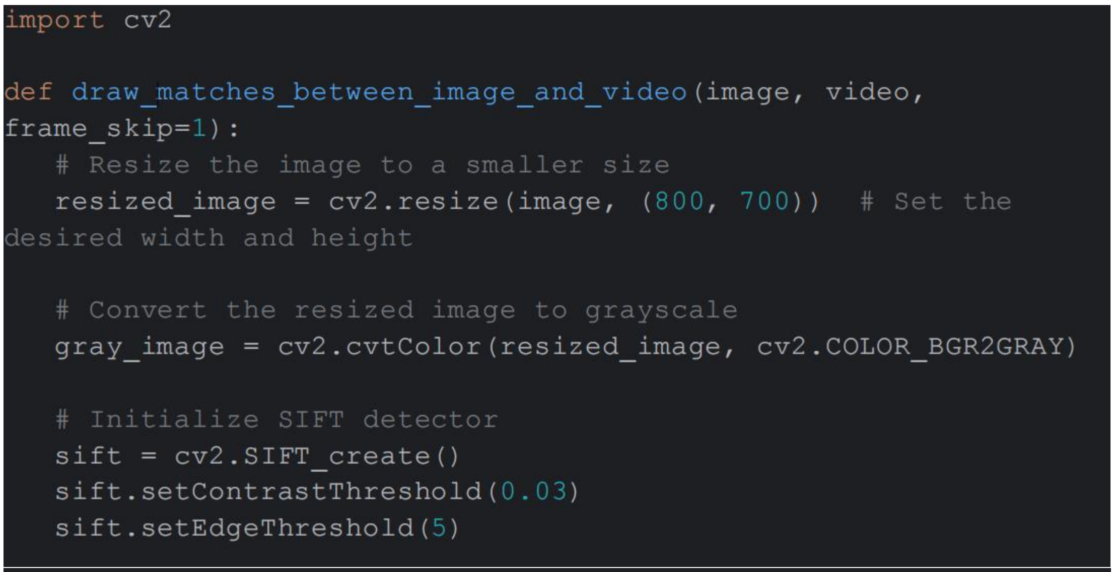
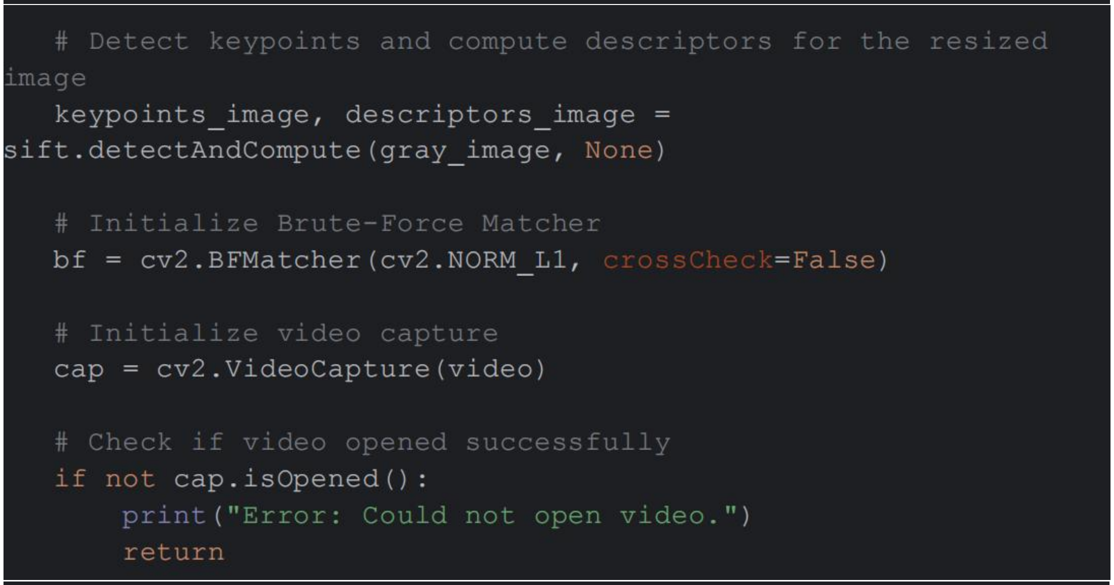
Nguyễn Thảo Nhi -2151010266

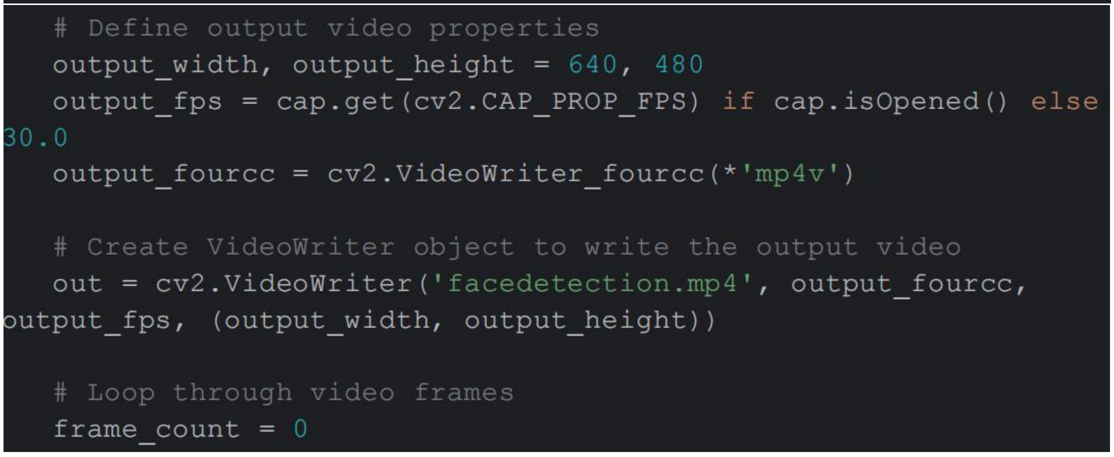
FACE DETECTION



```
import cv2

def draw_matches_between_image_and_video(image, video,
frame_skip=1):
    # Resize the image to a smaller size
    resized_image = cv2.resize(image, (800, 700))  # Set the
desired width and height

    # Convert the resized image to grayscale
    gray_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)

    # Initialize SIFT detector
    sift = cv2.SIFT_create()
    sift.setContrastThreshold(0.03)
    sift.setEdgeThreshold(5)
```



```
    # Detect keypoints and compute descriptors for the resized
image
    keypoints_image, descriptors_image =
sift.detectAndCompute(gray_image, None)

    # Initialize Brute-Force Matcher
    bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=False)

    # Initialize video capture
    cap = cv2.VideoCapture(video)

    # Check if video opened successfully
    if not cap.isOpened():
        print("Error: Could not open video.")
        return
```

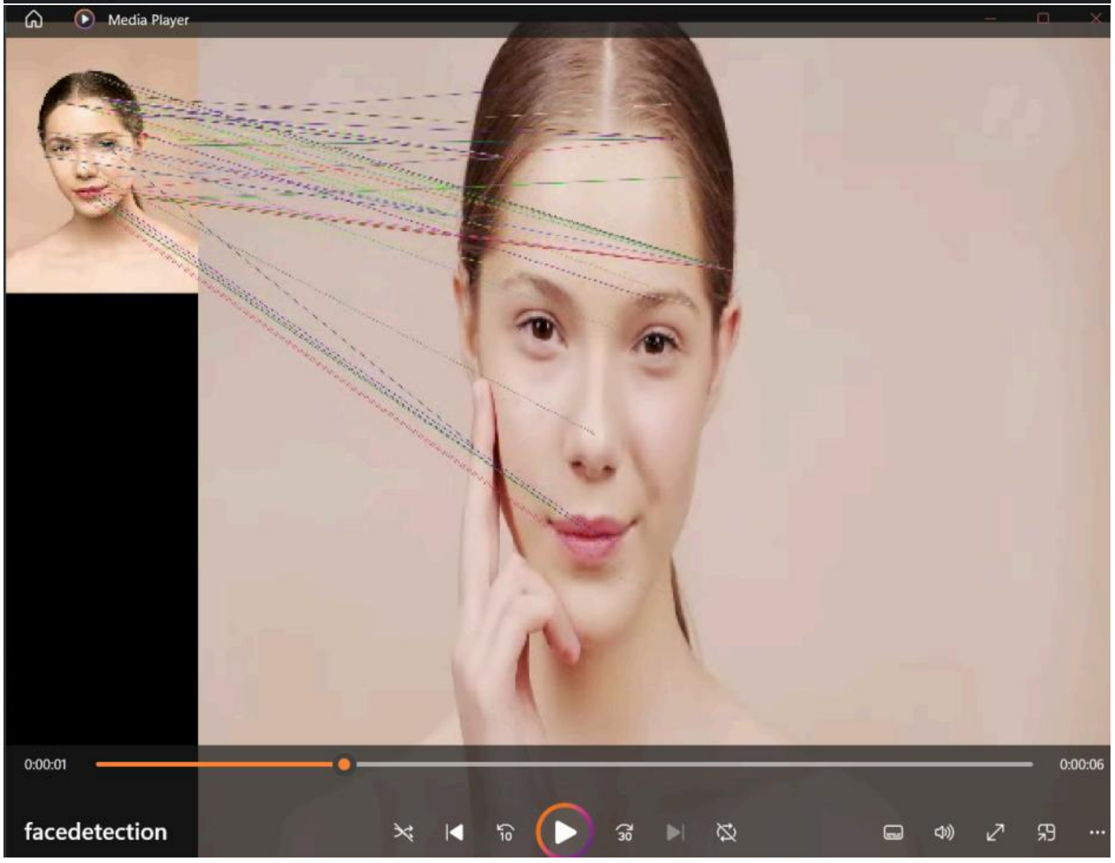


```
    # Define output video properties
    output_width, output_height = 640, 480
    output_fps = cap.get(cv2.CAP_PROP_FPS) if cap.isOpened() else
30.0
    output_fourcc = cv2.VideoWriter_fourcc(*'mp4v')

    # Create VideoWriter object to write the output video
    out = cv2.VideoWriter('facedetection.mp4', output_fourcc,
output_fps, (output_width, output_height))

    # Loop through video frames
    frame_count = 0
```

```
   while cap.isOpened():
       ret, frame = cap.read()
       if not ret:
           break

       frame_count += 1
       if frame_count % frame_skip != 0:
           continue

       # Convert the frame to grayscale
       gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

       # Detect keypoints and compute descriptors for the frame
       keypoints_frame, descriptors_frame =
sift.detectAndCompute(gray_frame, None)

       # Match keypoints between resized image and frame
       matches = bf.match(descriptors_image, descriptors_frame)
       matches = sorted(matches, key=lambda x: x.distance)

       # Draw matches between resized image and frame on the
original-sized frame
       img_matched = cv2.drawMatches(resized_image,
keypoints_image, frame, keypoints_frame, matches[:50], None,
flags=2)

       # Resize the display window
       cv2.namedWindow('Matches Between Image and Video',
cv2.WINDOW_NORMAL)  # Create window with resizable flag
       cv2.resizeWindow('Matches Between Image and Video', 800,
600)  # Set window size (width, height)
```

```
        # Write the frame to the output video
        out.write(cv2.resize(img_matched, (output_width,
output_height)))

        # Display the result
        cv2.imshow('Matches Between Image and Video', img_matched)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('s'):
            break

    # Release video capture and VideoWriter
    cap.release()
    out.release()
    cv2.destroyAllWindows()
```

```
# Load the image and video
image_path = 'face.jpg'
video_path = 'vdface.mp4'
image = cv2.imread(image_path)

# Set frame_skip to control video speed
frame_skip = 5  # Adjust this value to control video speed

draw_matches_between_image_and_video(image, video_path,
frame_skip)
```


Media Player
0:00:01
0:00:06
facedetection