*Nguyễn Hoàng Thanh - 2151013087*

BÀI NỘP: LAB 3 - COMPUTER VISION

TASK 1: SIFT MATCHING

* Source code:

import cv2

img0 = cv2.imread("datasets/pic0.jpg")

img0 = cv2.resize(img0, (356, 270))

img1 = cv2.imread("datasets\pic1.jpg")

img1 = cv2.resize(img1, (960, 1280))

gray1 = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

gray0 = cv2.cvtColor(img0, cv2.COLOR\_BGR2GRAY)

sift = cv2.SIFT\_create()

sift.setContrastThreshold(0.03)

sift.setEdgeThreshold(5)

keypoints\_1, descriptors\_1 = sift.detectAndCompute(gray1, None)

keypoints\_0, descriptors\_0 = sift.detectAndCompute(gray0, None)

bf = cv2.BFMatcher(cv2.NORM\_L1, crossCheck=False)

matches = bf.match(descriptors\_0, descriptors\_1)

matches = sorted(matches, key=lambda x: x.distance)

img2 = cv2.drawMatches(

    gray0,

    keypoints\_0,

    gray1,

    keypoints\_1,

    matches[:50],

    None,

    matchColor=(0, 255, 0),

    singlePointColor=(255, 0, 0),

    flags=0,

)

matches = bf.knnMatch(descriptors\_0, descriptors\_1, k=2)

good = []

for m, n in matches:

    if m.distance < 0.7 \* n.distance:

        good.append([m])

img3 = cv2.drawMatchesKnn(

    gray0,

    keypoints\_0,

    gray1,

    keypoints\_1,

    good,

    None,

    matchColor=(0, 255, 0),

    matchesMask=None,

    singlePointColor=(255, 0, 0),

    flags=0,

)

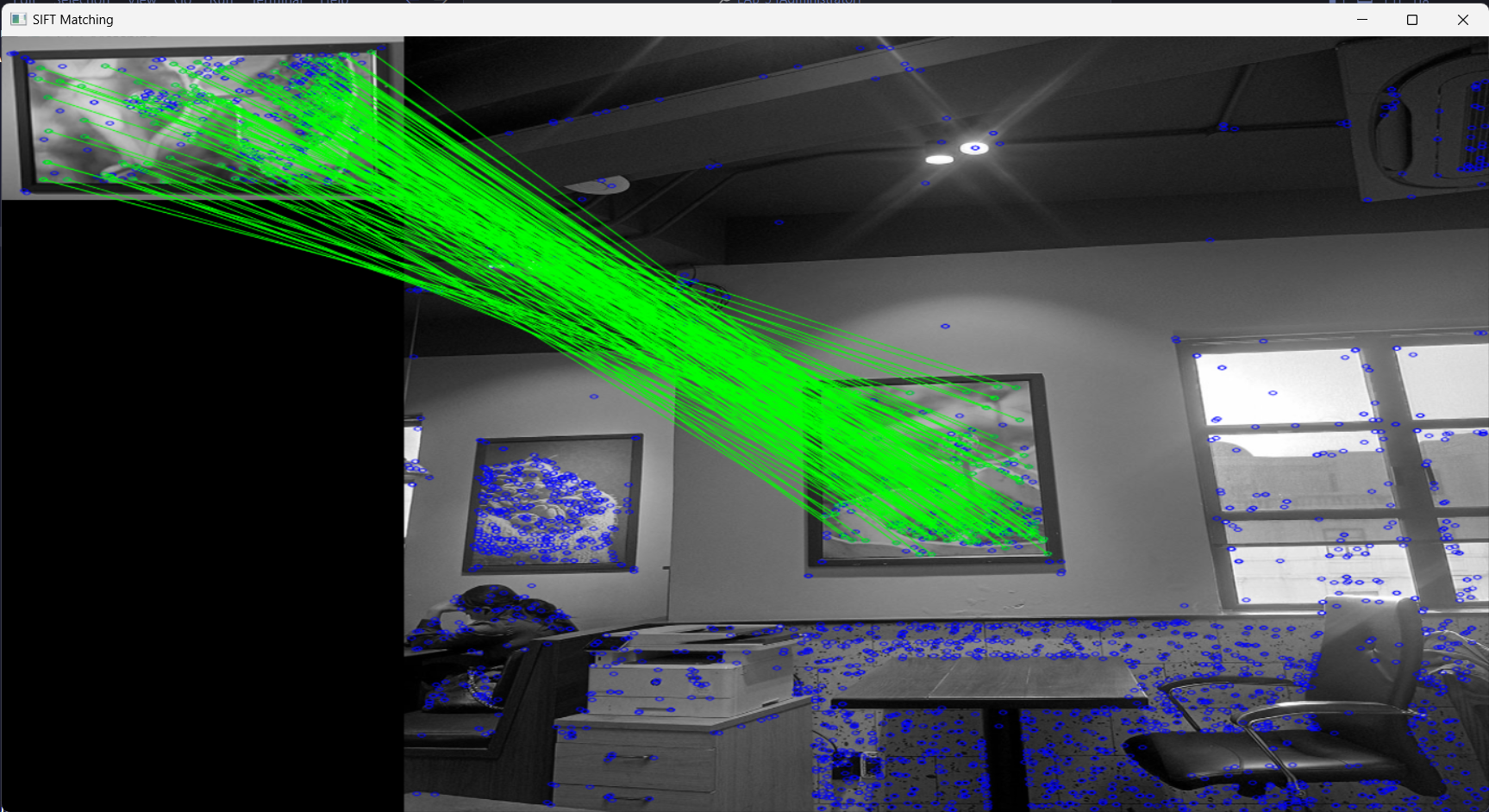
img3 = cv2.resize(img3, (1380, 720))

cv2.imshow("SIFT Matching", img3)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Result:



TASK 2: FACE DETECTION

Face\_detection\_Haar-like.py

* Source code:

import cv2

face\_cascade = cv2.CascadeClassifier(

    cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml"

)

cap = cv2.VideoCapture("datasets/video2.mp4")

while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        break

    gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    faces = face\_cascade.detectMultiScale(

        gray\_frame, scaleFactor=1.1, minNeighbors=8, minSize=(100, 100)

    )

    for x, y, w, h in faces:

        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2.imshow("Face Detection with Haar Cascade", frame)

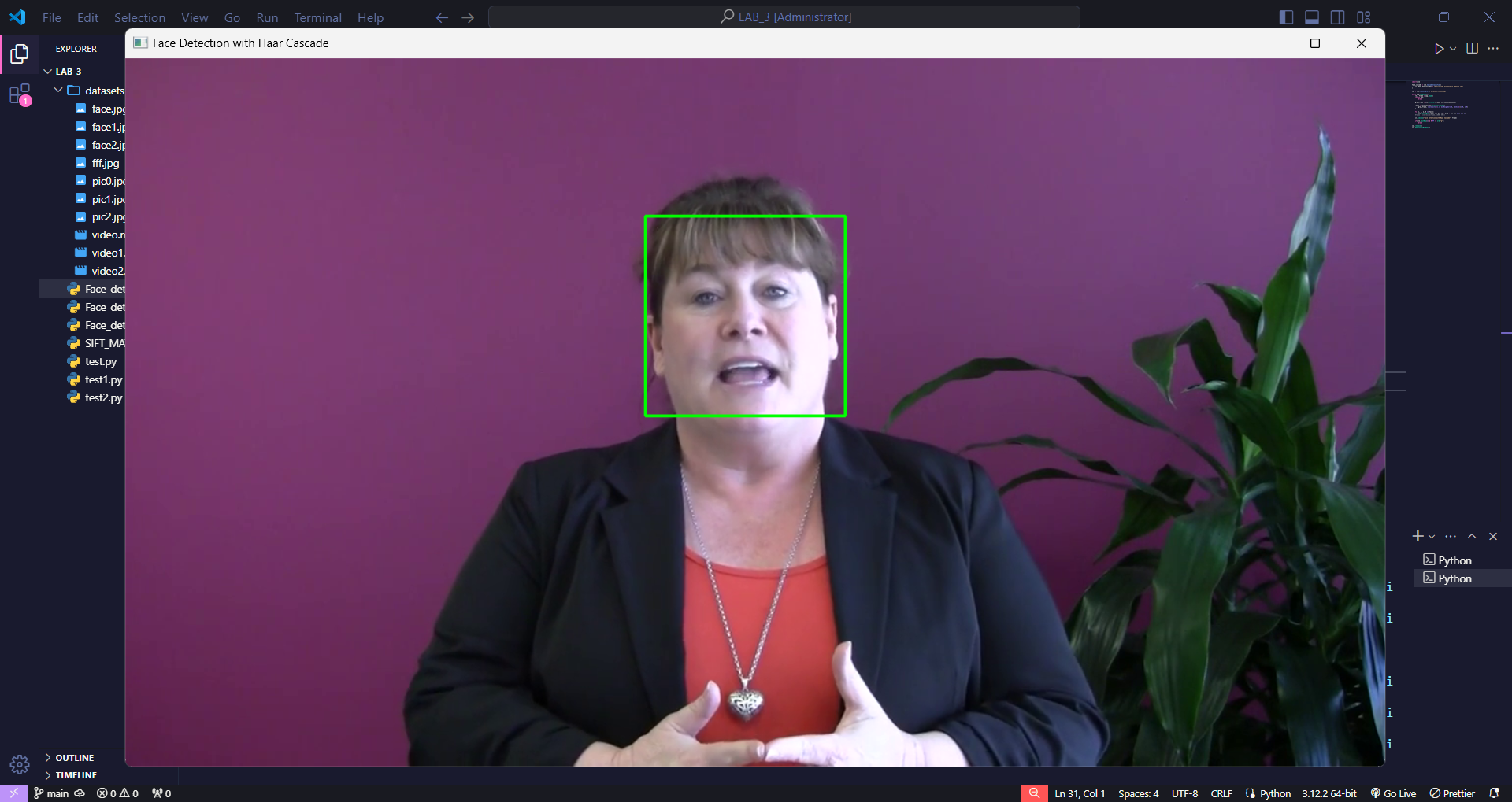
    if cv2.waitKey(1) & 0xFF == ord("q"):

        break

cap.release()

cv2.destroyAllWindows()

* Result:



Face\_detection\_SIFT\_casade.py

* Source code:

import cv2

import numpy as np

# Load cascade classifier for face detection

face\_cascade = cv2.CascadeClassifier(

    cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml"

)

# Create SIFT object

sift = cv2.SIFT\_create()

# Load image or video

cap = cv2.VideoCapture("datasets/video2.mp4")  # Or specify the path to your image

while cap.isOpened():

    # Capture frame from video

    ret, frame = cap.read()

    if not ret:

        break

    # Convert frame to grayscale

    gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    # Detect faces in the frame using cascade classifier

    faces = face\_cascade.detectMultiScale(

        gray\_frame, scaleFactor=1.1, minNeighbors=8, minSize=(100, 100)

    )

    # Draw green rectangles around detected faces

    for x, y, w, h in faces:

        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Detect keypoints and compute descriptors for frame

    keypoints, descriptors = sift.detectAndCompute(gray\_frame, None)

    # Draw keypoints on frame

    frame\_with\_keypoints = cv2.drawKeypoints(

        frame, keypoints, None, flags=cv2.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS

    )

    # Display the result

    cv2.imshow("Face Detection with SIFT and Cascade Classifier", frame\_with\_keypoints)

    if cv2.waitKey(1) & 0xFF == ord("q"):

        break

# Release resources

cap.release()

cv2.destroyAllWindows()

* Result:

