VIETNAM NATIONAL UNIVERSITY,
HO CHI MINH CITY
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

REPORT AND USER GUIDE
Individual Lab 3

Computer Vision

Performed by:
Student's ID: 21127730
Student's name: Hoang Le Cat Thanh

Instructors:
Pham Minh Hoang

Ho Chi Minh City, March 15, 2024

Contents

REPORT

1 Self-assessment

	Requirements	Completion rate
1	Load two images	100%
2	Detect keypoints and compute descriptors	100%
3	Find the closest matches between descriptors	100%
4	Find the homography transformation	100%

2 Functions used in algorithm

2.1 Extract key points and compute descriptors

The function use the Scale-Invariant Feature Transform (SIFT) algorithm in OpenCV to extract the key points and descriptors from an image.

- Input: An input image represented by a matrix, a vector of *KeyPoint* object in OpenCV to store the key points and a matrix to store the descriptors.
- Output: returns 1 if the key points and descriptors are computed successfully and 0 if it failed.
- Begin:
  - Verify whether the source image is empty. If it is indeed empty, the function terminates and returns 0.
  - The function applies OpenCV library to create a SIFT detector. Then the key points are extracted from the image and their descriptors are computed by the *detectAndCompute* function of the SIFT detector.
    - In SIFT, key points are often corners or blobs that possess uniqueness and can be consistently detected across multiple images of the same scene. The determination of each key point's position and scale is achieved by identifying stable features within scale-space, which involves a sequence of Gaussian-blurred images.
    - After identifying key points, SIFT proceeds to compute a descriptor for each key point. This descriptor is a 128-dimensional vector that encapsulates the most significant characteristics of the local image patch surrounding the key point. Designed to be robust to minor variations

in viewpoint or lighting, the descriptor ensures that key points can be reliably matched across images.

- The key points and descriptors are saved respectively to the vector and matrix and the function returns 1 indicating that the operation has been carried out successfully.

- End.

2.2 Find closest matches

The function is used to find the closest matches between the descriptors of a template image and a scene image using the FLANN Feature Matching.

- Input: two descriptors for template image and scene image, and a vector of *DMatch* object in OpenCV to store the good matches it finds.
- Output: returns 1 if the good matches are found successfully and 0 if it failed.
- Begin:
  - Verify whether any of the descriptors is empty. If it is indeed empty, the function terminates and returns 0.
  - Create a *FlannBasedMatcher* matcher and use the *knnMatch* function in the matcher find the k nearest neighbors for each descriptor, which in this case, k is set to 2 and store them in a vector *knnMatches*.
  - Subsequently, the function filters the matches based on their distances. If the distance of the first match is less than 75% of the distance of the second match, the match is deemed satisfactory.
  - The matches are saved into the vector and the function returns 1 indicating that the operation has been carried out successfully.
- End.

2.3 Find homorgraphy transformation

The function is used to find a homography matrix between a template image and a scene image. A homography refers to a transformation that establishes a mapping between points in one image and their corresponding points in another image. This transformation can accommodate various geometric changes, including rotation, translation, scaling, and perspective distortion.

- Input: A template image and a scene image in form of matrix and a matrix to store the homography matrix.
- Output: returns 1 if the homography matrix is found successfully and 0 if it failed.
- Begin:
  - Verify whether any of the image is empty. If it is indeed empty, the function terminates and returns 0.
  - Extract key points and descriptors for both images.
  - Find the closest matches between the descriptors of the two images.
  - For every good match, the function aggregates the point coordinates from the template image and the scene image into two vector.
  - Use the *findHomography* function to find the homography matrix that maps two corresponding points in the two previous vector together with the RANSAC algorithm to robustly estimate the homography.
  - Store the homorgraphy and return 1 indicating that the operation has been carried out successfully.
- End.

3 Object Detection

The function is used to detect a specific object from the template image in the scene image using the SIFT algorithm from OpenCV.

- Input: The template image and scene image and a matrix to save the result of the function.
- Output: returns 1 if the object detection is success and 0 if it failed.
- Begin:
  - Convert the input images to grayscale since SIFT works on grayscale images.
  - Sequentially carry out the functions: extract key points and compute descriptors, find the closest matches between descriptors, find the homorgraphy matrix.

  - Define the template image's corners and apply the homography using the *perspectiveTransform* function to maps the corners of the template image to the corresponding corners in the scene image and store the transformed corners.
  - From the transformed corners, lines are drawn between them to form a bounding box around the detected object in the scene image.
  - Finally, the matches between the key points in two images are drawn and the result image is stored to be shown and save later.

- End.

4 Proof images and result

4.1 Proof images

Figure 1: Extract the key points

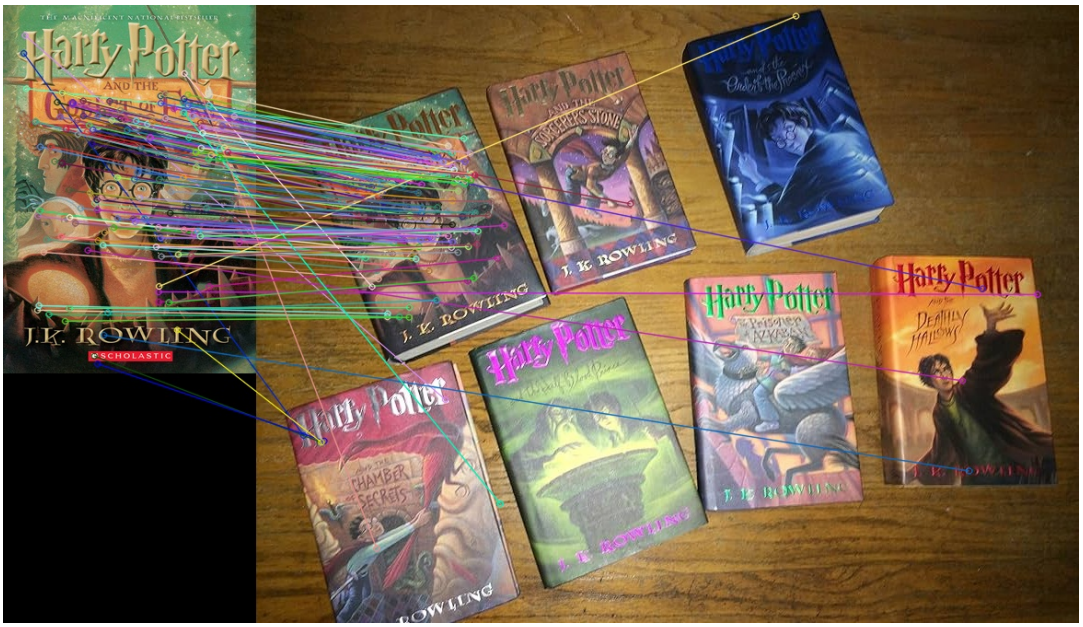
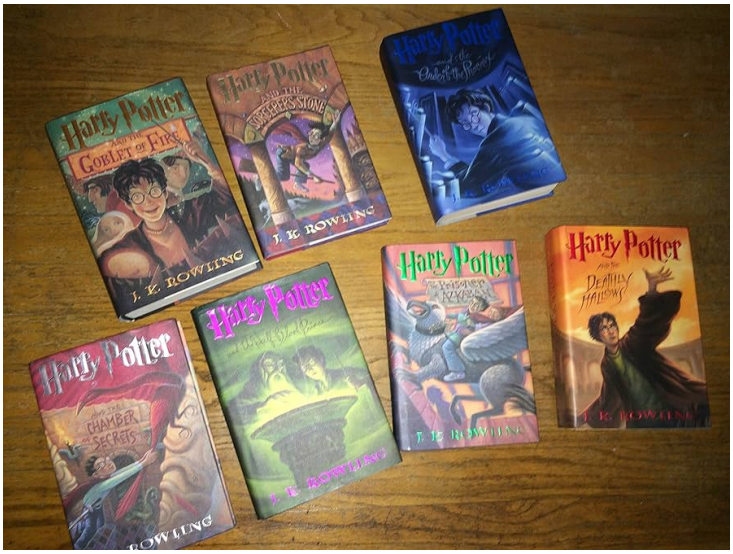


Figure 2: Good matches between template image and scene image

Figure 3: Bounding box around the detected object

4.2 Result image

(a) Template image



(b) Scene image

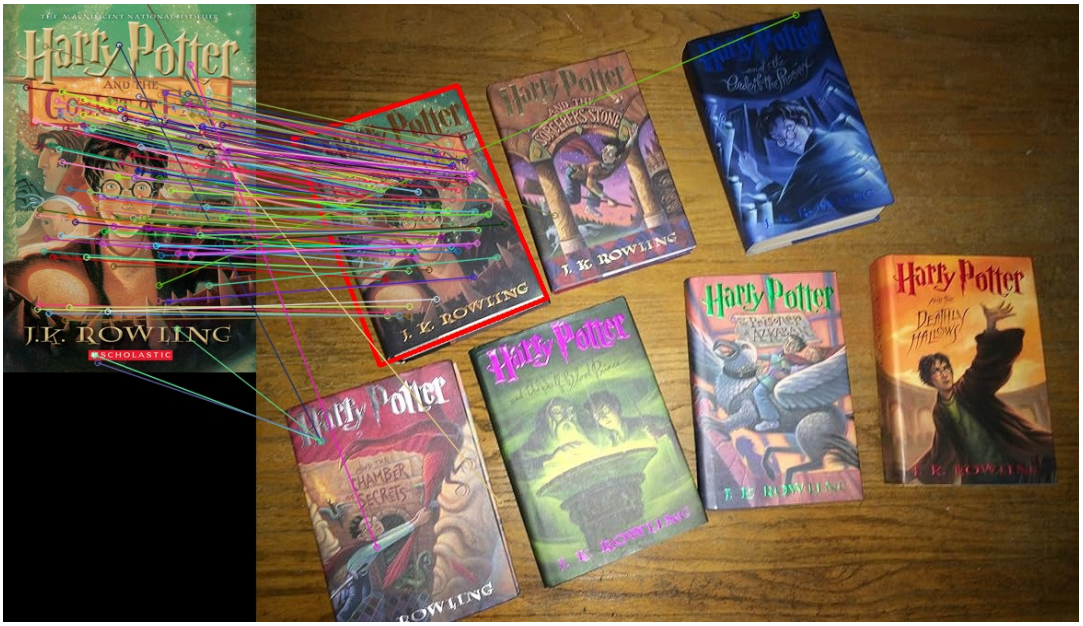


Figure 5: Result image

5 References

[0] Lecture slide
[1] Practice's references
[2] SIFT algorithm
[3] Feature detection
[4] Feature Detection and Description
[5] FLANN Feature matching
[6] Homorgraphy
[7] Object Detection with SIFT

USER GUIDE

6 Introduction

The project is the implementation of object detection using SIFT algorithm using OpenCV library in C++. The program perform the following functions

- Load two images: template and scene image from their paths
- Detect key points in both images and compute descriptors for each of the key points
- Find the closest matches between descriptors from the first image to the second
- Find the homography transformation between two sets of points
- Detect the object

7 Installation and set up

- Install OpenCV library
- Set up environment variables
- Configure project
  - Open project's *Properties*, add *Include Directories* and *Library Directories* as well as lib file.
  - Copy *opencv world341d.dll* to the Debug folder containing the exe file.

8 Running program

8.1 Command line

⟨Executable file⟩ -sift ⟨TemplateImagePath⟩ ⟨SceneImagePath⟩ ⟨OutputFilePath⟩

where

- *-sift*: command for the function
- *TemplateImagePath*: the path of template image

- *SceneImagePath*: the path of scene image
- *OutputFilePath*: the path of output image with the detected object highlighted by lines representing a bounding box

8.2 Call the program

- Run the program once to create exe file. (Assuming that the *opencv world341d.dll* has been copied to the project)
- The executable file is located in the debug folder after the project run once.
- In the *Debug* folder, right click and choose *Open in Terminal*.
- After open the Terminal, enter the command line with the *Executable file* is named *21127730*.
- If there is no error, the input and output image will be shown. After that, press Enter to close the images and save the output. If save successfully, there will be a notification on where the image is saved.

8.3 Sample process

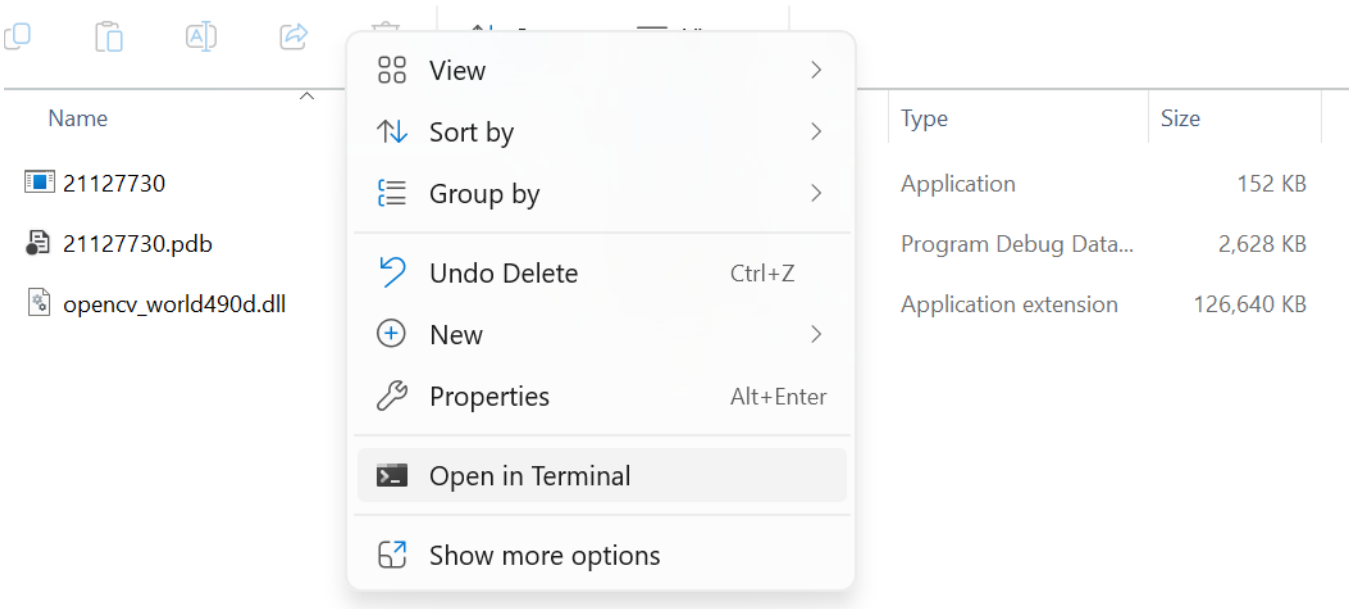


Figure 6: Open Terminal

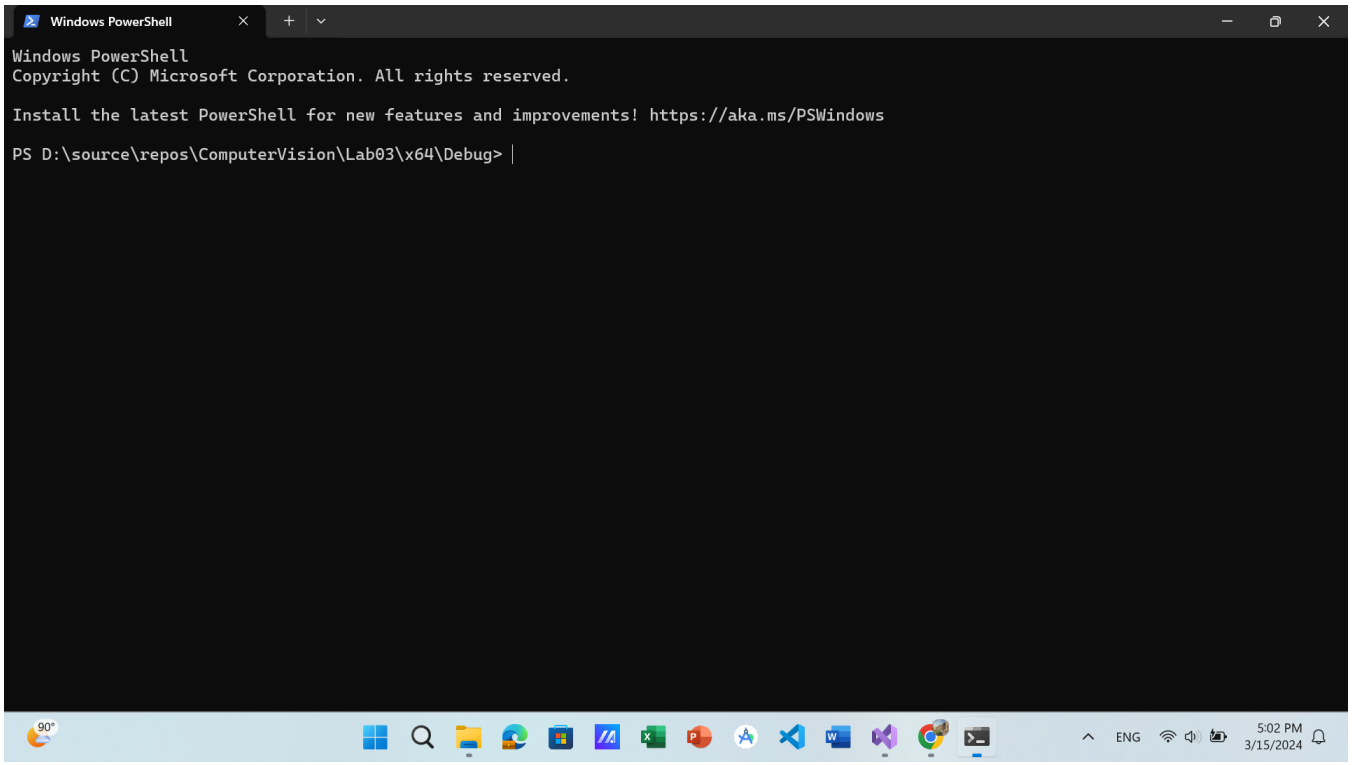


Figure 7: Terminal screen

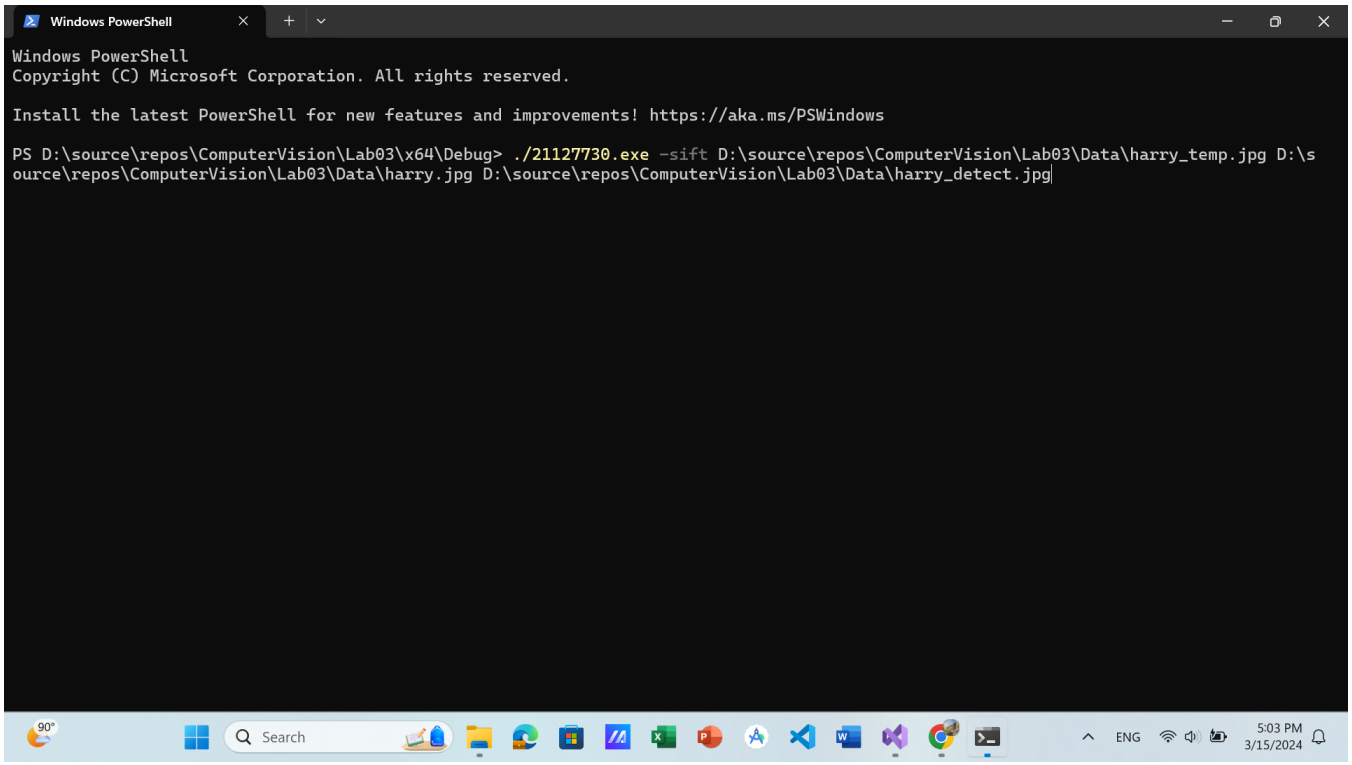


Figure 8: Enter the command line

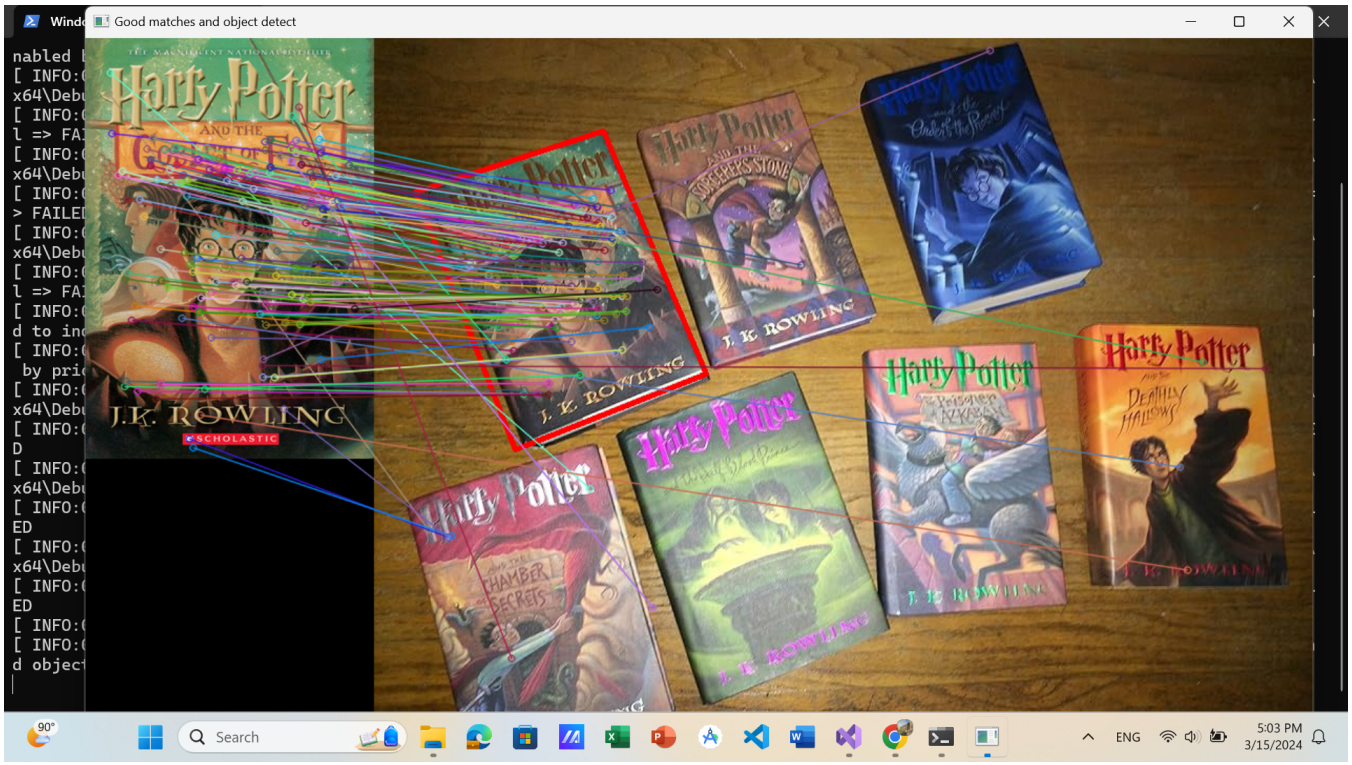


Figure 9: Show the result image

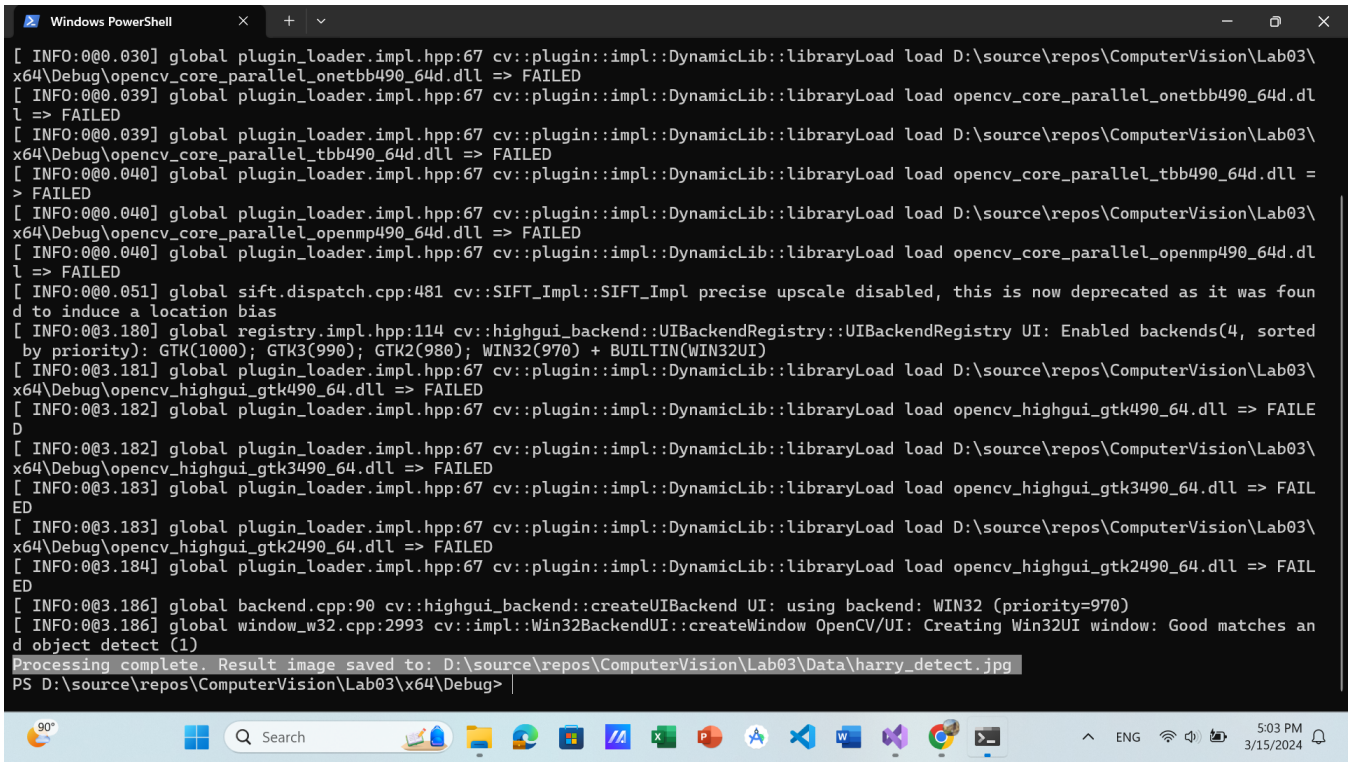


Figure 10: Output image saved successfully to the path