VIET NAM NATIONAL UNIVERSITY OF HO CHI MINH CITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

****

**EXERCISE-01**

**Computer Vision**

**Sinh viên thực hiện: 21127043 - Lư Trung Hậu**

**Lớp CN: 21CNTThức**

**Lớp BM: 21TGMT**

**Giảng viên hướng dẫn: Phạm Minh Hoàng**

**Nguyễn Trọng Việt**

**Võ Hoài Việt**

Contents

[I. User Guide: using Microsoft visual studio 2](#_Toc161678175)

[II:Self evaluation: 4](#_Toc161678176)

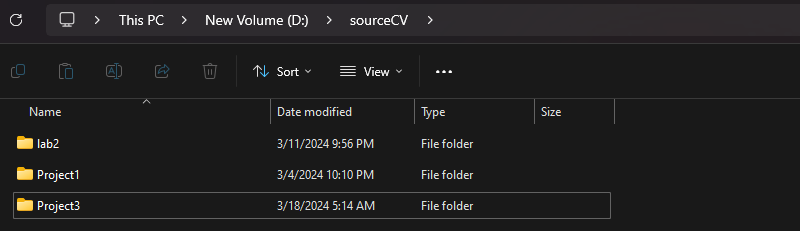
[III:Coding: 4](#_Toc161678177)

[IV: Experimental example: 6](#_Toc161678178)

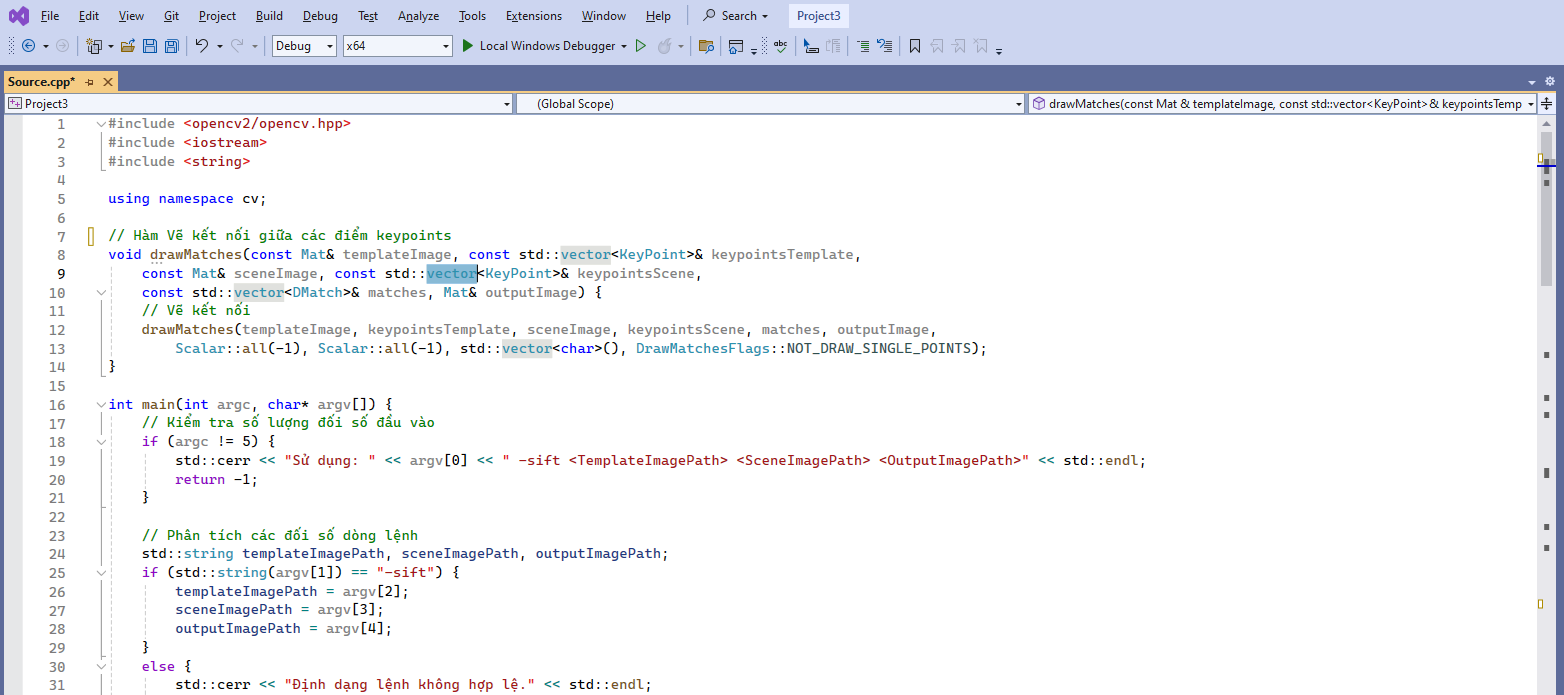
[V.References: 8](#_Toc161678179)

# I. User Guide: using Microsoft visual studio

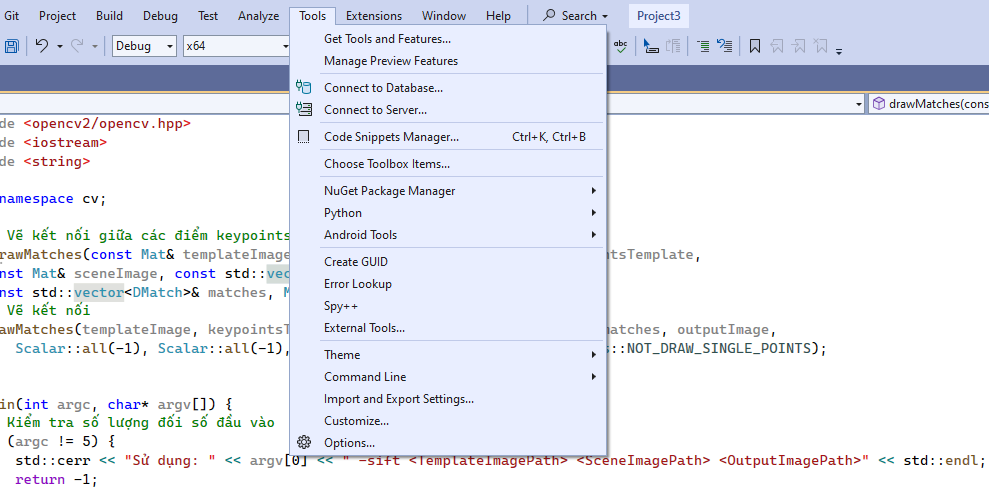
- Save the project to the available drive in the device. (Project3=lab3)



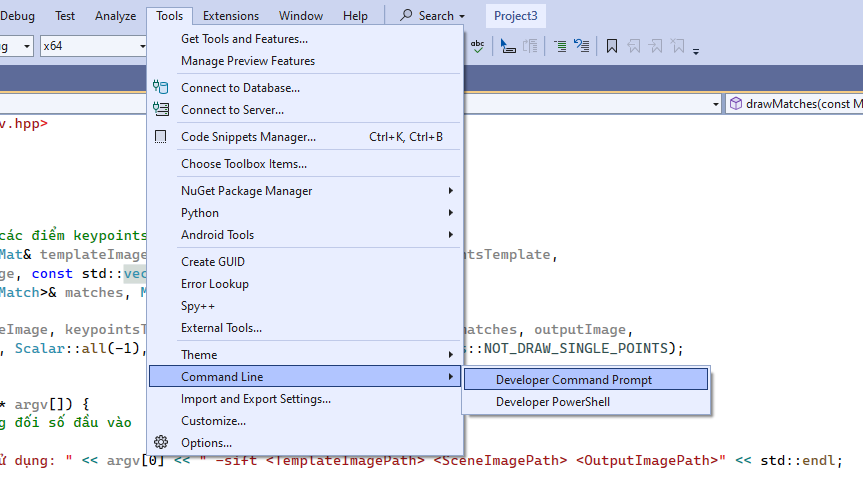
- Once you have the project in your computer, open the program



- In the menu bar, select Tools

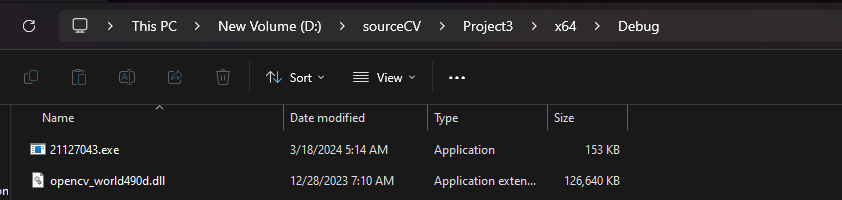


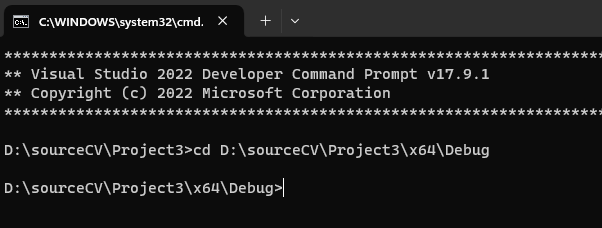
- Select Command Line.



- Select Developer Command Prompt:

- Navigate the path to the Debug file containing (21127043.exe)





- Enter the command line as requested in the lab:



21127043.exe -sift D:\\temp.jpg D:\\scene.jpg D:\\output.jpg



- After running, the image file output.jpg is located in drive D

- After completing the above step, we see the line:



- So we have succeeded in running the project.

# II:Self evaluation:

|  |  |
| --- | --- |
| Requirements | Completion rate |
| Load Images | 100% |
| Detect the keypoint | 100% |
| Calculate descriptors | 100% |
| Draw matches | 100% |
| Draw boundaries  (highlighted bydrawing lines) | 100% |
| Save Sift Image | 100% |

# III:Coding:

1. Initialization:

**// Initialize SIFT feature detector**

**SIFT sift = initializeSIFT()**

* In this part, we initialize the SIFT feature detector from the OpenCV library, which will be used for detecting keypoints and computing descriptors in the subsequent steps.

2. Keypoint Detection and Description:

**// Detect keypoints and compute descriptors for template and scene images**

**keypointsTemplate,descriptorsTemplate=detectAndCompute(sift,templateImage)**

**keypointsScene, descriptorsScene = detectAndCompute(sift, sceneImage)**

* Detect keypoints and compute descriptors for both the template and scene images using the SIFT algorithm. We use the initialized SIFT feature detector (sift) to perform this task.

3. Matching Descriptors:

**// Match descriptors between template and scene images**

**matches = matchDescriptors(descriptorsTemplate, descriptorsScene)**

**// Filter good matches**

**goodMatches = filterMatches(matches)**

* This section involves matching descriptors between the template and scene images. Utilizing a suitable matching algorithm (such as Brute Force) to find correspondences between descriptors. Then, filter out the good matches based on a specified criteria (e.g., ratio test) to ensure robustness in subsequent steps.

4. Drawing Connections:

**// Draw connections between keypoints on template and scene images**

**outputImage=drawMatches(templateImage,keypointsTemplate,sceneImage, keypointsScene, goodMatches)**

* Draw connections between keypoints on the template and scene images using the good matches obtained in the previous step. The drawMatches function visualizes these connections on the output image.

5.Localization:

**// Obtain corresponding points between template and scene images**

**obj, scene = getCorrespondingPoints(goodMatches, keypointsTemplate, keypointsScene)**

**// Find homography matrix between template and scene images**

**H = findHomography(obj, scene, RANSAC)**

* This part involves localizing the object in the scene image.Firstly, obtain corresponding points between the template and scene images based on the good matches. Then, we find the homography matrix (H) using the RANSAC algorithm, which represents the transformation between the template and scene images.

6. Drawing Object Boundaries:

**// Get object corners from template image**

**objCorners = getObjectCorners(templateImage)**

**// Perspective transform object corners to scene image**

**perspectiveTransform(objCorners, H)**

**// Draw object boundaries on scene image**

**drawObjectBoundaries(outputImage, sceneCorners)**

* In this section, Draw the boundaries of the object on the scene image. Firstly obtain the corners of the object from the template image. Then, Transform these corners to the scene image using the homography matrix. Finally, Draw the object boundaries on the scene image.

# IV: Experimental example:

Load images:

a.Scene:



b.Template:



2. Results after drawing matches and bounding:



Successful Execution:

* If the program runs without any errors, it indicates successful execution.
* The program performs SIFT feature detection, descriptor matching, and object localization on the provided template and scene images.
* Keypoint matches between the template and scene images are identified and visualized through drawn connections.
* The object in the template image is localized within the scene image, and its boundaries are marked.
* The resulting image with drawn connections and object boundaries is saved to the specified output file.

Output Image:

* The output image contains visual representations of the keypoint matches and the localized object boundaries.
* Keypoint matches are depicted by lines connecting corresponding keypoints between the template and scene images.
* The localized object boundaries in the scene image are outlined by lines connecting the corners of the template image after transformation.
* These visual cues help in understanding the alignment and positioning of the object within the scene.

# V.References:

1. <https://www.opencv.org.cn/opencvdoc/2.3.1/html/doc/tutorials/features2d/feature_homography/feature_homography.html#feature-homography>
2. <https://docs.opencv.org/4.x/d9/d97/tutorial_table_of_content_features2d.html>