# **CVI620/DPS920- Lab 10**

# **Matching features and templates**

| Total Mark: | 10 marks (2.5% of the total course grade)   * 7 out of 10: Blackboard submission * 3 out of 10: Lab demo |
| --- | --- |
| Submission file(s): | * Lab10\_1.cpp * Lab10\_2.cpp * Lab10\_3.cpp * Lab10.docx including the **result images and answers** |

Please work in **groups** to complete this lab. This lab is worth 2.5% of the total course grade and will be evaluated through your written submission ***(include all results)***, as well as the lab demo. During the lab demo, group members are *randomly* selected to explain the submitted solution. Group members not present during the lab demo will lose the demo mark.

1. Add this declaration to your file:

We, ------------ (mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.

## **Part I: Keypoint matching**

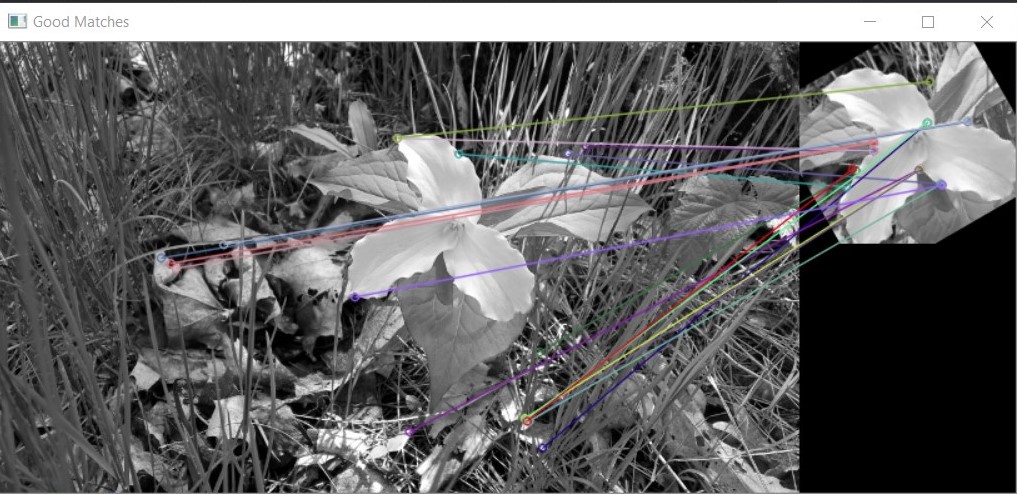
1. Download the code for SURF features and FLANN matcher into Lab10\_1.cpp:

<https://docs.opencv.org/3.0.0/d5/d6f/tutorial_feature_flann_matcher.html>

Modify the code as follows:

1. Remove include lines with xFeatures2d
2. Change the SURF detector to ORB detector and set the maximum features parameter for this detector to 50.
3. Change the definition of matcher to

cv::Ptr<cv::DescriptorMatcher> matcher = cv::makePtr<cv::FlannBasedMatcher>(cv::makePtr<cv::flann::LshIndexParams>(12, 20, 2));

1. Open the Trillium\_s image (posted) as the first image and Trillium\_t image (posted) as the second image.
2. Run and debug the code. Are the keypoints matched properly? ****
3. Set the second image to be a rotated version of the first image (e.g. use Trillium\_t and rotate by 30 degrees around the center of the image). Run the code. How are the keypoints matched this time?

## **Part II: Template matching**

1. Write code in Lab10\_2.cpp to:
   1. Open Trillium\_s image as query image, and Trillium\_t image as template image.
   2. Use matchTemplate to find the match location.
   3. Draw a rectangle around the match in the query image.
   4. Show the match in the matching space. Use normalization similar to: <https://docs.opencv.org/3.1.0/de/da9/tutorial_template_matching.html>



* 1. Now rotate the query image by 30 degrees (and keep the template the same), and do the template matching. How is the performance?



* 1. Assuming the first location found (in part b) is the ground truth location, calculate the number of TP, TN, FP, and FN pixels. Then calculate precision and recall and output the results.

## **Part III: Motion Detection**

1. Write a code in Lab10\_3.cpp to use the webcam video stream (similar to the photo booth application in Lab 2). Apply the following changes:
2. Convert the captured frames to grayscale.
3. Save the first frame (converted to grayscale) as background image.
4. Change code to display the difference images, instead of captured frames.
5. Threshold the difference image using a threshold of 128.
6. Count the number of pixels changed by at least 128 gray levels, by finding the sum of the threshold image in part d.
7. If the count is larger than 1% of the frame size,
   1. Make a beep sound
   2. Output the time the activity was detected
   3. Save the current color frame in a file (with a counter or time stamp)
   4. Update the background image
   5. Wait for 2 seconds before returning to capturing loop.
8. Exit the program by entering esc or space.
9. Adjust the motion sensitivity (1% of the frame size) to your desired sensitivity.
10. Specify what each member has done towards the completion of this work:

|  | Name | Task(s) |
| --- | --- | --- |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |