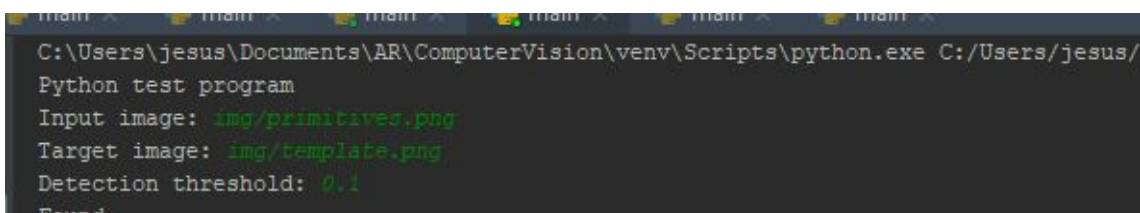- You have one week to complete the assignment.
- If the code has errors, the exercise won't be accepted for submission.
- Test images are provided to check if your code is correct. The application must find the target when using the "t1 images" (target 1) and threshold 0.1. Instead, no matching must be found when using "t2 images" and the same threshold.
- If a test does not run correctly, that exercise will not be accepted for submission.
- Code is expected to be readable, clean, and optimal.
- To submit the exercise, upload the python file and change its name so the student's name and surname can be seen directly **"lastname1_name1_and_lastname2_name2.py"**. Upload the file to the **"Project 1"** folder.
- No submissions will be accepted after **March 29th at 23:59:59**.

Statement

Implement the Template Matching algorithm from scratch using the **sum of squared differences metric**. Take into account the following considerations:

- The application loads the input and the target images and computes the matching map using the above mentioned metric. Then, if the minimum value of the matching map divided by the maximum one is lesser than a user-defined threshold (0.1), the target is considered to be in the image.

- To improve the performance of your program, compute the matching map by using the **grayscale** versions of the input and the target images. However, the results must be shown in color as shown in the image below.

- The input of the application must be:
    - The input image
    - The target image
    - The detection threshold

- The program must allow the user to introduce these values. For instance:

- As a result, the application must show in different windows:
    - The target image
    - The matching map
    - The input image with the target highlighted if it was found in it
    - A message telling whether or not the target was found

For instance, when the target is found, the application must show something similar to the following images:

On the other hand, when the target is not found, the output should be the following:
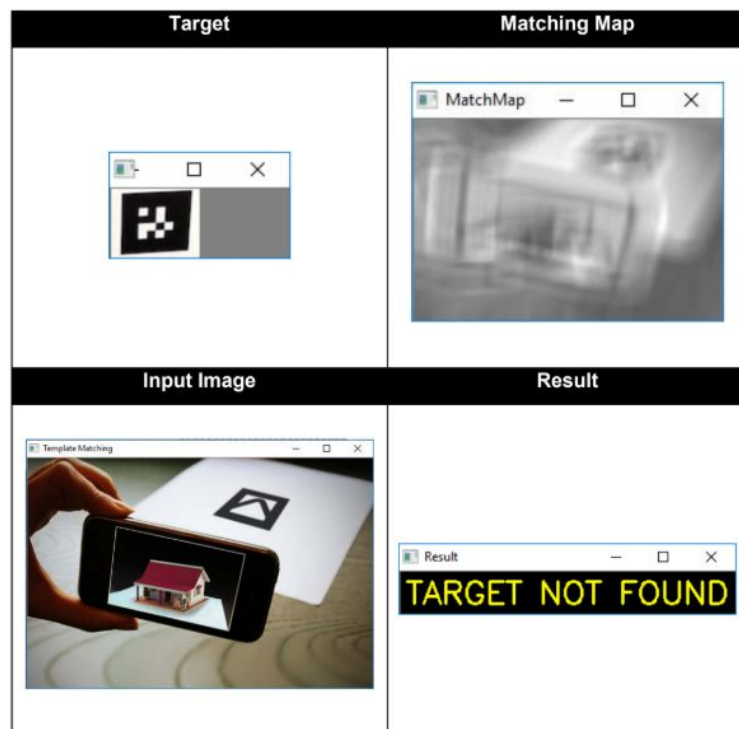
NOTE: Remember to normalize the images right before showing them.

- When the target is repeated in the input image, the application should detect and highlight all of them.

  For instance, if the target is , the result of the application must show:

- In order to highlight the target on the image when it is found, you can use the function **cv.rectangle(...)**. Look at the OpenCV API for additional information.

- Furthermore, to write the result of your application, you can use cv2.putText(...). For instance, you can use the following code:

```
# Create a black image
imgFound = np.zeros((40,245,3), np.uint8)

# Write the text
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(imgFound,"TARGET FOUND",(5,30), font, 1,(0,255,0),2)

#Display the image
cv2.imshow("Result",imgFound)
```

- The programming language used must be Python, and you cannot use the OpenCV methods (or other packages) other than the ones stated above, and those used to load and show images, and to manage keyboard input.

IMPORTANT!

You will see that this method is quite slow if you make intensive use of loops. In order to improve the performance of this technique, you can scale down both the target and the input images, and compute the matching map from them. Take into account that both images should be scaled to the same extent, aspect ratios must be preserved, and the main features of both images too (do not scale too much!). To resize the images you can use cv2.resize().

This feature is optional, but it must be implemented to get the maximum qualification!