# Assignment 6 - Template Matching

Start Assignment

**Due** Oct 12 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip
**Available** Oct 5 at 2pm - Oct 15 at 11:59pm 10 days

## Assignment Instructions

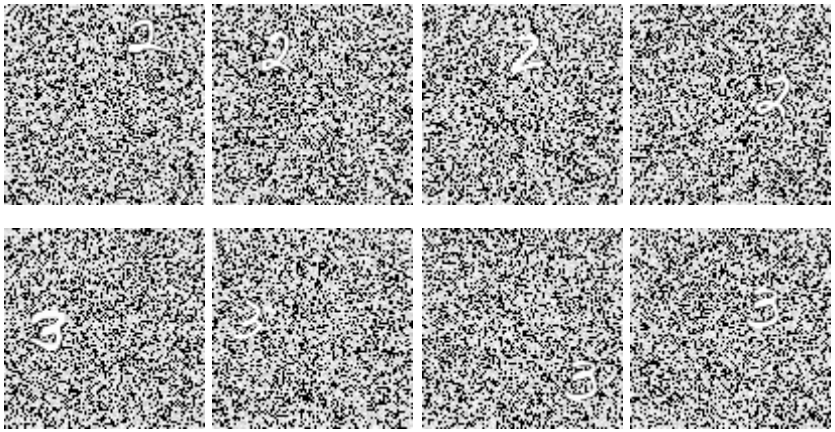

Figure 1. Example images on which your program will be tested.

Figure 2. A few examples from the MNIST dataset. You will use roughly 2000 images from this dataset in order to compute your average-2 and average3 images (about 1000 for each average image).

**Task 1 [50 points]:** Write a script, saved in a file called `task1.m` that computes an average two-image and an average three-image, by

taking the average over many images displaying the digit 2 and the digit 3.

Your training data comes from a public dataset that is called the *MNIST dataset*. You can download this training data from the attached **training_data.zip** (https://canvas.txstate.edu/courses/1798576/files/204435679/download?download_frd=1) file. Unzip that file, and put the resulting training_data directory into your Matlab path. Then, type

```
load_mnist
```

The above command loads 10,000 images of digits. To access, for example, the 434th image, you type the following:

```
%example value for i
i = 434;

% get the i-th image out of test digits
var = mnist_digits(:,:,i);

% show the i-th image in a figure
imshow(var, []);

% the label is a number between 0 and 9, telling us
% which digit is contained in the i-th image.
label = mnist_labels(i);

% print the label.
disp(label);
```

To compute the average two-image, you have to find (using the information in `mnist_labels`) all the images in `mnist_digits` that are displaying the number 2, and compute their average. Similarly, to compute the average three-image, you have to find (using the information in `mnist_labels`) all the images in `mnist_digits` that are displaying the number 3, and compute their average.

IMPORTANT: After executing `task1.m`, the average two-image and average three-image should be of size 28x28, and should be stored, respectively, in variables called `average2` and `average3` in the Matlab Workspace.

---

**Task 2 [50 points]:** Create a function called `detect_digit(image, template)`, that behaves as follows:

- It takes two inputs: the first input is a grayscale image like the images shown on Figure 1, and the second input is variable `average2` or variable `average3`, depending on whether we want to detect the digit 2 or the digit 3. As a reminder,

variables `average2` and `average3` are created by task 1.

- It computes normalized correlation scores between the image and the template, but it DOES NOT search over multiple scales or multiple orientations. You can assume that the digit will always appear at the standard scale and orientation in which the digit appears in the training images.
- It identifies the best-matching position, i.e., the pixel location where the best correlation score occurs.
- It creates a figure where it shows the original image, with a white 28x28 bounding box drawn centered on the best-matching position.
- It does not create any other figures or print out anything.
- It returns a 1x2 matrix (one row, two columns), containing the row (first) and column (second) of the best-matching position.

---

**Task 3 [10 points - extra credit]:** Write a function called `recognize_digit(image, average2, average3)` with the following specs:

- It takes three inputs: the first input is a grayscale image like the images shown in Figure 1. The second input is variable `average2`, and the third input is variable `average3`, from task 1.
- It does not create any figures or print out anything.
- It returns a single number, that is equal to either 2 or 3, depending on whether your function thinks that image displays a 2 or a 3.

You can implement this function in any way you like. I have a very simple solution, that attains 85% accuracy on the 40 images stored in the attached **test_data.zip** (https://canvas.txstate.edu/courses/1798576/files/204435689/download?download_frd=1) file (it gets correct results for 34 out of the 40 images). To get all the points, you must get correct results for at least 32 of the 40 images.

---

# How to submit

Submissions are only accepted via Canvas. The submission should include, as an attachment, a zip file containing your Matlab code and a README.txt file. The zip file should be named `NetId_lastname_firstname.zip`, with no spaces or other extra characters. Your solution should definitely contain:

- A Matlab file called task1.m that implements the solution for task 1.

- A Matlab file called detect_digit.m that implements the solution for task 2.
- A Matlab file called recognize_digit.m that implements the solution for task 3.
- In addition, your zip file should contain ALL files needed to run the code. If you are using files that are posted on the course web page, you still have to include them in your zip file. At least 5 points will be taken off otherwise.
- The README.txt file should contain the name and Net ID of the student, in addition to any additional comments/instructions useful for running the code and understanding the underlying ideas.

We try to automate the grading process as much as possible. Not complying precisely with the above instructions causes a significant waste of time during grading, and thus points will be taken off for failure to comply, and/or you may receive a request to resubmit.

## Additional resources for assignment

**training_data.zip** (https://canvas.txstate.edu/courses/1798576/files/204435679/download?download_frd=1)

**test_data.zip** (https://canvas.txstate.edu/courses/1798576/files/204435689/download?download_frd=1)