Bilkent University

Semester of Fall 2024-2025

EEE 391
Basics of Signals and Systems

MATLAB Assignment 2

Image Signal Analysis and Processing

Deniz Tuna ONGUNER

Department of Computer Engineering

Bilkent ID: 22001788

Section: 3
Instructor: Prof. Haldun ÖZAKTAŞ

Wed, 11 Dec 2024

This document is submitted to the Department of Electrical and Electronics Engineering of Bilkent University in partial fulfillment of EEE 391, Basics of Signals and Systems, requirement.

Table of Contents

List of Figures

List of Listings

1 Introduction

The objective of this report is to document, demonstrate, and analyze the use of MATLAB in image signal analysis and processing via filtering by convolution operation and template matching. The main focus is to learn and apply the concepts of convolution, filtering, and template matching in image processing. Throughout the assignment, I applied filters of different sizes to a gray-scale image of a flower, added Gaussian noise to the image, and filtered the noised image with different filters. I also added two images together and filtered the resulting image with a 10x10 filter. Later, I filtered the two images separately with the same filter and added them together. Finally, I counted the number of horizontal "t" letters in a given text-containing image using template matching.

All the images generated and the MATLAB scripts used in this assignment are included in the Appendices A and B, respectively.

2 Analysis and Discussion

2.1 Question 1 Part 1: Applying Filters

In this part, I applied filters of different sizes to a gray-scale image of a flower. The filters used are 3x3, 10x10, and 50x50, respectively. The filters were applied to the image using two different methods: a self-implemented convolution function and the built-in convolution function of MATLAB. The results to the self-implemented convolution function are shown in Figure 1, and the results to the built-in convolution function are shown in Figure 2 of Appendix A. The code for the self-implemented convolution function is shown in Listing 1, and the test script for the function is in Listing 2 of Appendix B. The code for this part is shown in Listing 3 of Appendix B.
As one can observe, applying filters of different sizes to the image results in different outputs. The larger the filter size, the more blurred the image becomes. The 3x3 filter made the image smoother but, still, recognizable; on the other hand, in the case of the 50x50 filter, it is even hard to tell that it is an image to a flower. This is due to the fact that larger filters average the pixel values over a larger area, which results in a smoother image. The self-implemented convolution function and the built-in convolution function yield the exact same results, which is expected since the aim was to re-implement the built-in function.

2.2 Question 1 Part 2: Image Noising and Filtering

In this part, I added Gaussian noise to the gray-scaled flower image, the noised image is shown in Figure 3 of Appendix A. The code for this part is shown in Listing 4 of Appendix B. The noise was generated with a mean of 0 and a standard deviation of 0.5, as stated in the assignment document. Then, I filtered this noised image using 3x3, 10x10, and 50x50 filters. The results are shown in Figure 4 of Appendix A.
The 3x3 filter did not remove the noise effectively, but the 10x10 and 50x50 filters did a better job. The 50x50 filter removed the noise almost entirely, but it also blurred the image significantly, making it unrecognizable just like in the case of Part 1. The 10x10 filter, on the other hand, removed the noise relatively better than 3x3 one while preserving the details of it.

2.3 Question 2: Image Summing and Filtering

In this part, first, I added two images, image1.png and image2.png, together. Then, I filtered the resulting image with a 10x10 filter. Later, I filtered the two images separately with the same filter of 10x10 and added them together. The results are shown in Figure 5 of Appendix A. The code for this part is shown in Listing 5 of Appendix B.
Expectedly, the results are the same, as verified within the script. This is because the convolution operation is linear [1] and, hence, the order of the operations does not matter. The filter is applied to the sum of the images in both cases, and the results are the same. So, filtering the sum of two images, it's mathematically the same as summing the filtered versions of the individual images.

2.4 Question 3: Letter Counting in an Image

In this part, I counted the number of horizontal “t” letters in a given text-containing image. The code for this part is shown in Listing 6.
The aim was to detect the number of occurrences of the letter “t” in the given text-containing image. To do this, I used the letter “t” as a template and performed template matching with the other image. The number of horizontal “t” letters in the text was calculated by summing the thresholded image. The result was 5, as shown in Listing 7 of Appendix B.

(i) As preprocessing, the `letter_t` template is rotated by 180 degrees to align with the convention of cross-correlation. Convolution with a flipped kernel/template is equivalent to cross-correlation in image processing, which is typically used for template matching [2, 3].

(ii) The Procedure:

1. Load the images: The two given images are loaded; the image containing the text and the letter_t image, which serves as the template for detecting the letter “t”.

2. Normalize the images: Images are converted to double precision and normalized for uniform processing.

3. Rotate the template (Refer to (i) for the reasoning).

4. Perform convolution: The text image is convolved with the rotated template to detect the letter “t” in the text.

5. Rescale the result: The result of the convolution is rescaled to be in the range [0, 1], this step is requested in the assignment document.

6. Thresholding: The rescaled result is thresholded at 0.95 to detect the letter “t”, this step is also requested in the assignment document.

7. Count the letters: The number of horizontal “t” letters in the text is calculated by summing the thresholded image.

(iii) The name of the application performed is “Template Matching” [3, 4]. The letter “t” is used as the template, and it is matched with the image containing the text.

3 Conclusion

Throughout this assignment, I have learned and applied the concepts of convolution, filtering, and template matching in image processing using MATLAB. To conclude, I have learned how convolution operation works exactly as I have implemented it myself, and how filtering can be used to smooth images and remove noise. I have also learned that the order of operations does not matter in linear systems, as the results are the same. Finally, I have learned how template matching can be used to detect objects in images. After all, I have gained a better understanding of image signal analysis and processing, and I have improved my MATLAB skills.

4 Overview of Codes

The MATLAB scripts used in this assignment are all included in the Appendix B. R2024b version of MATLAB was used to run the scripts. Each script, except for the one containing convolution function implementation, starts with clearing the workspace and the command window via `clc` and `clear` commands, respectively.
Then, using the `addpath` command, the directory containing the images and the convolution function is added to the script path, so that the images can be read and the function can be used. They all could be typed on a single script file, but I explicitly preferred to separate them for clarity, ease of understanding, and debugging purposes, as well as for the sake of modularity and presentability, staying true to the software engineering principles.

References

[1] McClellan, J. H., Schafer, R. W., & Yoder, M. A. (2003). Signal Processing First (Chapter 5). Pearson/Prentice Hall.

[2] Jacobs, D. (2005). Correlation and Convolution. https://www.cs.umd.edu/ djacob-s/CMSC426/Convolution.pdf.

[3] Snavely, N. (2009). CS6670: Computer Vision Lecture 2: Edge detection and resampling. https://www.cs.cornell.edu/courses/cs6670/2009fa/lectures/lec02_edge.pdf

[4] Girod, B. (2013). Template Matching. https://web.stanford.edu/class/ee368/ Handouts/Lectures/2014_Spring/Combined_Slides/9-Template-Matching-Combined.pdf.

Appendix A

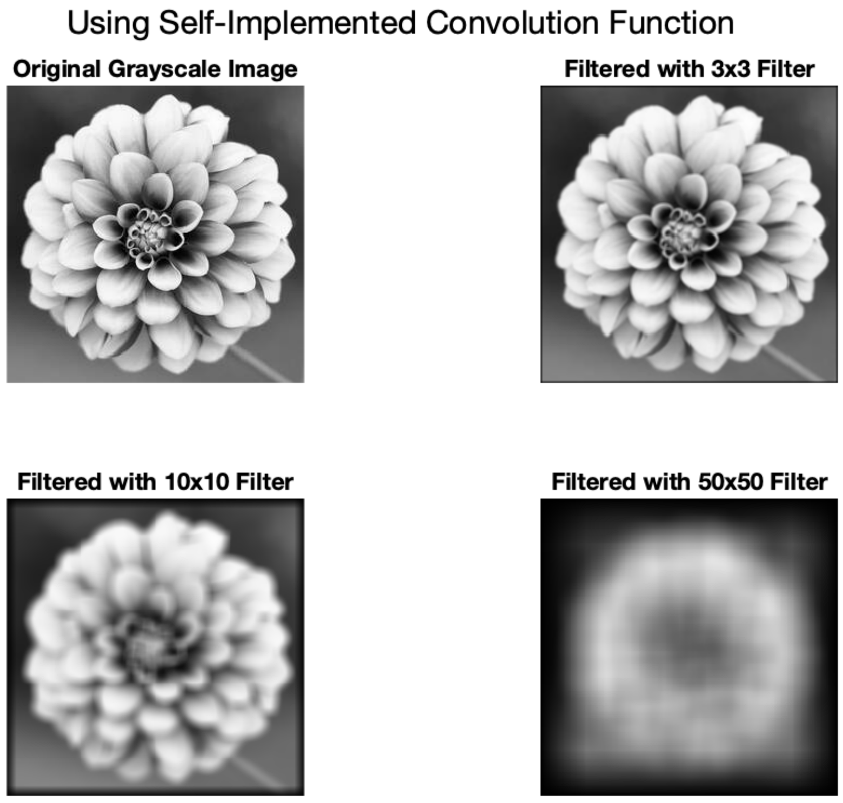
Using Self-Implemented Convolution Function
Original Grayscale Image
Filtered with 3x3 Filter
Filtered with 10x10 Filter
Filtered with 50x50 Filter

Figure 1: Using Self-Implemented Convolution Function with Different Filters

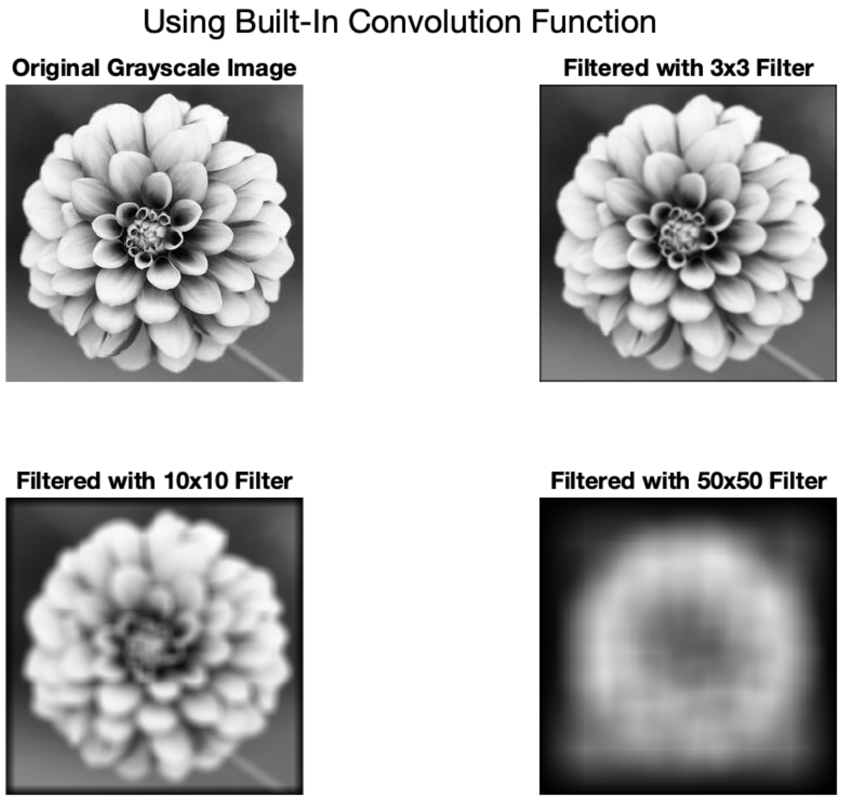
Using Built-In Convolution Function
Original Grayscale Image
Filtered with 3x3 Filter
Filtered with 10x10 Filter
Filtered with 50x50 Filter

Figure 2: Using Built-In Convolution Function with Different Filters

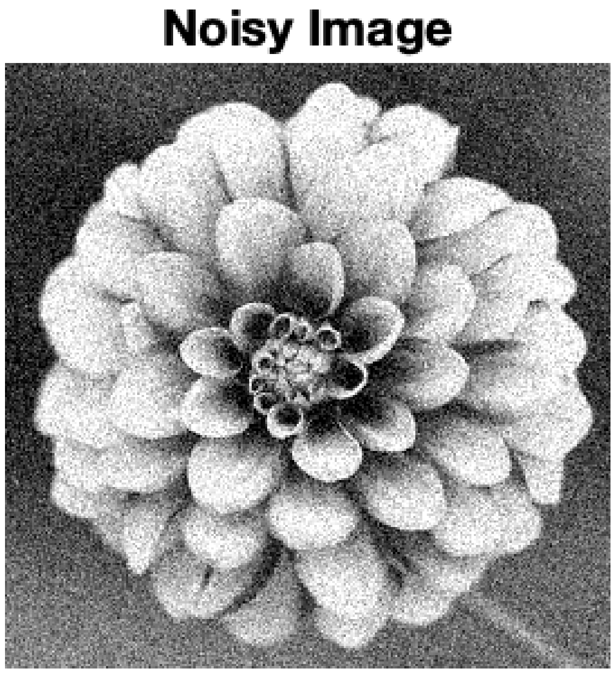

Figure 3: Generated Noisy Image

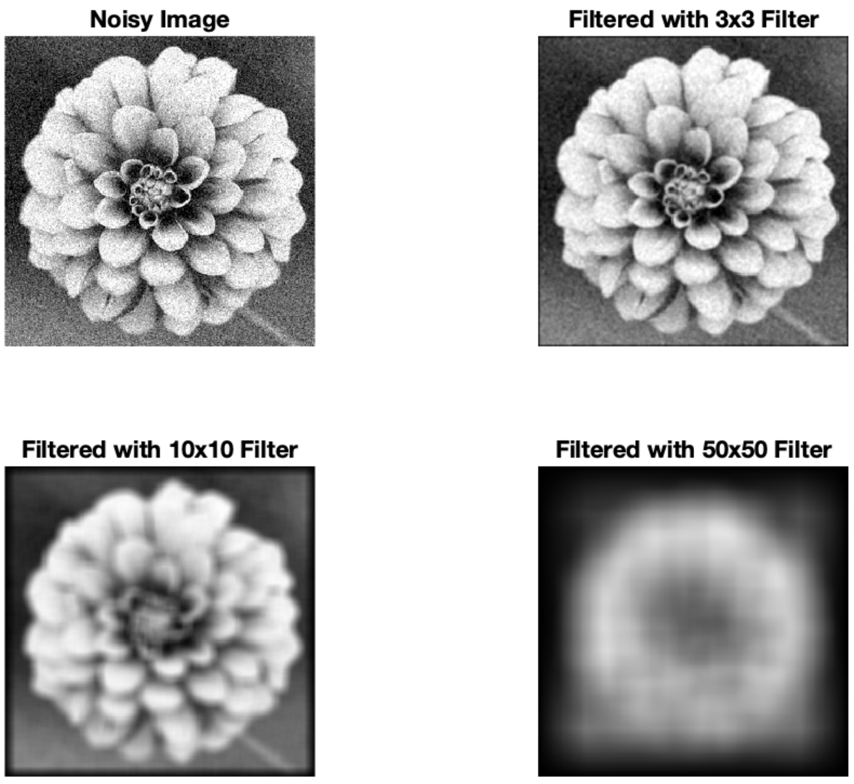

Figure 4: Filtered Noisy Image with Different Filters

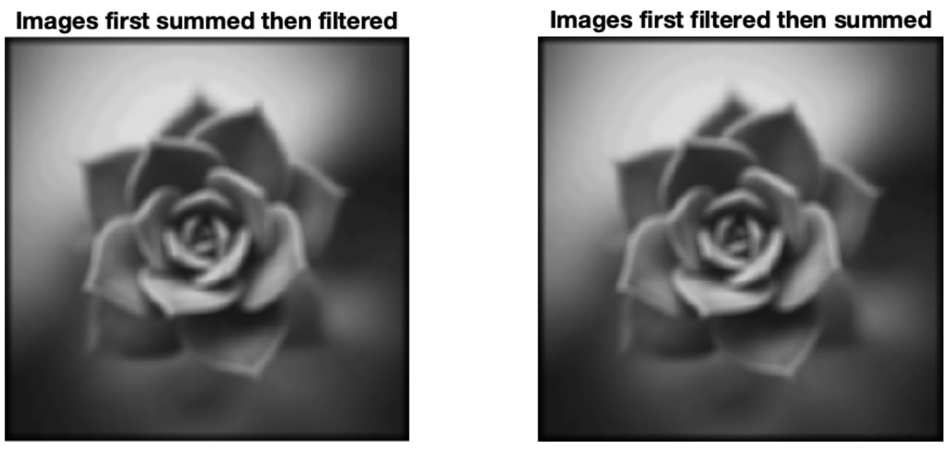
Images first summed then filtered
Images first filtered then summed

Figure 5: Summing and Filtering Images in Different Orders

Appendix B

Listing 1: Self-Implemented Matrix Convolution Function

```matlab
% @author: Deniz T. Onguner
% MATLAB version: R2024b
% This is the script for the self-implemented matrix convolution function.
% Please note that the performance of the function is disregarded.
% The four-level nested loop is likely to cause slow runtime performance.
function convolution = my_conv2(image, kernel)
    % Y := convolution
    % X := image
    % H := kernel

    % Getting the dimensions of the parameters
    [n_image_rows, n_image_cols] = size(image);
    [n_kernel_rows, n_kernel_cols] = size(kernel);

    % Deciding the output matrix size
    n_conv_rows = n_image_rows - 1 + n_kernel_rows;
    n_conv_cols = n_image_cols - 1 + n_kernel_cols;

    % Padding the input image with zeros
    padded_image = zeros(n_conv_rows, n_conv_cols);
    padded_image((1 : n_image_rows), (1 : n_image_cols)) = image;

    % Initializing the output matrix, initially filled with zeros
    convolution = zeros(n_conv_rows, n_conv_cols);

    % Performing convolution using nested loops
    % Y[i, j] = Σ_nΣ_m(X[m, n]H[i - m, j - n])
    for i = (1 : n_conv_rows)
        for j = (1 : n_conv_cols)
            for m = (1 : n_kernel_rows)
                for n = (1 : n_kernel_cols)
                    if (i - m + 1 > 0) && (j - n + 1 > 0) && ...
                        (i - m + 1 <= n_image_rows) && ...
                        (j - n + 1 <= n_image_cols)
                        convolution(i, j) = convolution(i, j) + ...
                        padded_image(i - m + 1, j - n + 1) * kernel(m, n);
                    end % end if
                end % end for
            end % end for
        end % end for
    end % end for
end % end of function
% end of script
```

Listing 2: Test Script for the Self-Implemented Convolution Function

```matlab
% @author: Deniz T. Onguner
% MATLAB version: R2024b
% This is the script for testing the self-implemented convolution function.
clc; clear;
addpath('EEE 391/Assignment 2/');

% Test 1: Basic 3x3 image with a 2x2 kernel
input_image = [1, 2, 3; 4, 5, 6; 7, 8, 9];
conv_kernel = [1, 0; 0, -1];

output_my_conv2 = my_conv2(input_image, conv_kernel);
output_builtin = conv2(input_image, conv_kernel, 'full');

assert(isequal(output_my_conv2, output_builtin), 'Test 1 Failed!');
disp('Test 1 Passed');

% Test 2: Larger kernel
conv_kernel = [1, 2, 1; 0, 0, 0; -1, -2, -1];

output_my_conv2 = my_conv2(input_image, conv_kernel);
output_builtin = conv2(input_image, conv_kernel, 'full');

assert(isequal(output_my_conv2, output_builtin), 'Test 2 Failed!');
disp('Test 2 Passed');

% Test 3: Larger input image with a small kernel
input_image = randi(10, 5, 5); % Random 5x5 image
conv_kernel = [1, -1; -1, 1];

output_my_conv2 = my_conv2(input_image, conv_kernel);
output_builtin = conv2(input_image, conv_kernel, 'full');

assert(isequal(output_my_conv2, output_builtin), 'Test 3 Failed!');
disp('Test 3 Passed');

% Test 4: Edge case - kernel larger than image
input_image = [1, 2; 3, 4];
conv_kernel = [1, 0, -1; 2, 0, -2; 1, 0, -1];

output_my_conv2 = my_conv2(input_image, conv_kernel);
output_builtin = conv2(input_image, conv_kernel, 'full');

assert(isequal(output_my_conv2, output_builtin), 'Test 4 Failed!');
disp('Test 4 Passed');

% All tests passed
disp('All tests passed successfully!');

% End of script
```

Listing 3: Script for Part 1 of Question 1

```matlab
% @author: Deniz T. Onguner
% MATLAB version: R2024b
% This is the script for the Part 1 of the Question 1
clc;
clear;

addpath('EEE 391/Assignment 2/');

I = imread('flower.jpg'); % Reading the image
I = rgb2gray(I); % Converting the image to grayscale since it is colorful
A = mat2gray(I); % Converting to J matrix

% Declaring the filters
filter03 = ones(03) / 03^2;
filter10 = ones(10) / 10^2;
filter50 = ones(50) / 50^2;

% Using self-implemented conv2 function, i.e. my_conv2,
%   to perform convolution and to apply filters
I_filtered03 = my_conv2(A, filter03);
I_filtered10 = my_conv2(A, filter10);
I_filtered50 = my_conv2(A, filter50);

% Displaying the results
figure;
subplot(2,2,1); imshow(A); title('Original Grayscale Image');
subplot(2,2,2); imshow(I_filtered03); title('Filtered with 3x3 Filter');
subplot(2,2,3); imshow(I_filtered10); title('Filtered with 10x10 Filter');
subplot(2,2,4); imshow(I_filtered50); title('Filtered with 50x50 Filter');
sgtitle('Using Self-Implemented Convolution Function');

% Doing the same but using built-in conv2 function this time
I_filtered03 = conv2(A, filter03);
I_filtered10 = conv2(A, filter10);
I_filtered50 = conv2(A, filter50);

% Again, displaying the results
figure;
subplot(2,2,1); imshow(A); title('Original Grayscale Image');
subplot(2,2,2); imshow(I_filtered03); title('Filtered with 3x3 Filter');
subplot(2,2,3); imshow(I_filtered10); title('Filtered with 10x10 Filter');
subplot(2,2,4); imshow(I_filtered50); title('Filtered with 50x50 Filter');
sgtitle('Using Built-In Convolution Function');

% End of script
```

Listing 4: Script for Part 2 of Question 1

```matlab
% @author: Deniz T. Onguner
% MATLAB version: R2024b
% This is the script for the Part 2 of the Question 1
clc;
clear;

addpath('EEE 391/Assignment 2/');

I = imread('flower.jpg'); % Reading the image
I = rgb2gray(I); % Converting the image to grayscale since it is colorful
A = mat2gray(I); % Converting it to J matrix

[rows, cols] = size(A); % Getting the image sizes for output matrix

noise = randn(rows, cols) * 0.5; % Gaussian noise with mean 0 and std 0.5
noise = noise * 0.2; % Multiplying the new matrix by 0.2 to scale it down

image_noised = A + noise; % Adding generated matrix to input image

% Displaying the noisy image
figure;
imshow(image_noised); title('Noisy Image');

% Declaring the filters
filter03 = ones(03) / 03^2;
filter10 = ones(10) / 10^2;
filter50 = ones(50) / 50^2;

I_filtered03 = my_conv2(image_noised, filter03);
I_filtered10 = my_conv2(image_noised, filter10);
I_filtered50 = my_conv2(image_noised, filter50);

% Displaying the results
figure;
subplot(2,2,1); imshow(image_noised); title('Noisy Image');
subplot(2,2,2); imshow(I_filtered03); title('Filtered with 3x3 Filter');
subplot(2,2,3); imshow(I_filtered10); title('Filtered with 10x10 Filter');
subplot(2,2,4); imshow(I_filtered50); title('Filtered with 50x50 Filter');

% End of script
```

Listing 5: Script for Question 2

```matlab
% @author: Deniz T. Onguner
% MATLAB version: R2024b
% This is the script for the Question 2
clc;
clear;

addpath('EEE 391/Assignment 2');

filter10 = ones(10) / 10^2;

image_1 = imread('image1.png');
image_2 = imread('image2.png');

image_1 = im2double(image_1);
image_2 = im2double(image_2);

image_1 = image_1 * 10;
image_2 = image_2 * 5;

images_added_v1 = image_1 + image_2;
images_filtered = my_conv2(images_added_v1, filter10);

image_1 = my_conv2(image_1, filter10);
image_2 = my_conv2(image_2, filter10);

images_added_v2 = image_1 + image_2;

figure;

subplot(1,2,1);
imshow(images_filtered);
title('Images first summed then filtered');

subplot(1,2,2);
imshow(images_added_v2);
title('Images first filtered then summed');

assert(isequal(images_filtered, images_added_v2), ...
    'The matrices are NOT equal!');

% End of script
```

Listing 6: Script for Question 3

```matlab
% @author: Deniz T. Onguner
% MATLAB version: R2024b
% This is the script for the Question 3
clc;
clear;

addpath('EEE 391/Assignment 2/');

text = im2double(imread('Text.png'));
letter_t = im2double(imread('lettert.png'));

letter_t = rot90(letter_t, 2); % Rotate the letter t by 180 degrees

result_conv = my_conv2(text, letter_t);
result_conv = rescale(result_conv);

treshold = result_conv >= 0.95;
n_ts = sum(treshold(:));

disp(['The number of horizontal t letters in the text is: ', ...
    num2str(n_ts)]);

% End of script
```

Listing 7: Output of the Script for Question 3

```
The number of horizontal t letters in the text is: 5
>>
```