CSE 3048
HW#1 REPORT

Implementing 2D Discrete Convolution with MATLAB

Lecturer: Mehmet Kadir BARAN

Assistant: Lokman ALTIN

Student Names - IDs:

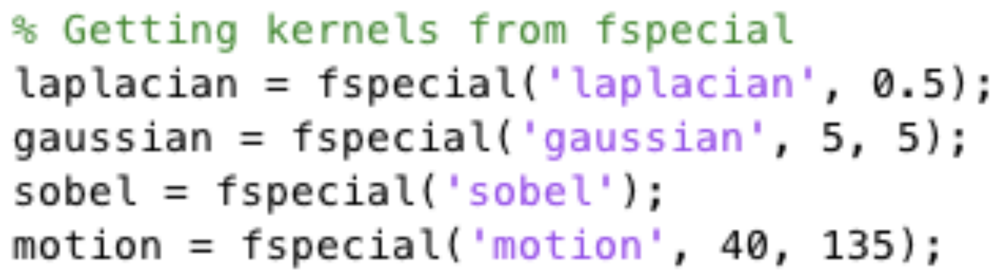
- Mert KELKİT - 150115013
- Rümeysa ELİÖZ - 150114016

Sample Image

We used “cameraman.tif” grayscale image which can be found in MATLAB’s directory.

Used Filters

We used Gaussian, Laplacian, Sobel and Motion filters which can be generated from fspecial function in MATLAB.

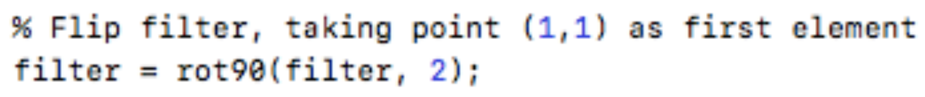


```
% Getting kernels from fspecial
laplacian = fspecial('laplacian', 0.5);
gaussian = fspecial('gaussian', 5, 5);
sobel = fspecial('sobel');
motion = fspecial('motion', 40, 135);
```

- Gaussian filter is used for blurring images.
- Laplacian filter can be used for edge detection.
- Sobel filter can be used for edge detection.
- Motion filtered images are looking like they are moving.

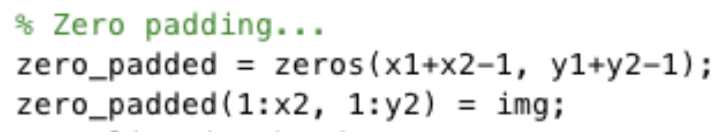
Flipping Kernels and Padding Images

We called rot90 function in order to flip the kernel up-down and left-right. Flipping should be done for convolution.

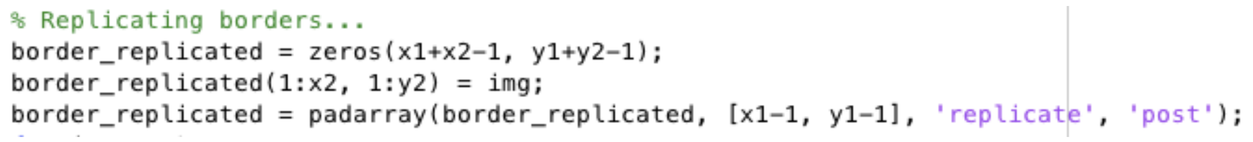


```
% Flip filter, taking point (1,1) as first element
filter = rot90(filter, 2);
```

In this homework, we have to use 2 padding types, first one is zero padding, second one is border replication.



```
% Zero padding...
zero_padded = zeros(x1+x2-1, y1+y2-1);
zero_padded(1:x2, 1:y2) = img;
```



```
% Replicating borders...
border_replicated = zeros(x1+x2-1, y1+y2-1);
border_replicated(1:x2, 1:y2) = img;
border_replicated = padarray(border_replicated, [x1-1, y1-1], 'replicate', 'post');
```

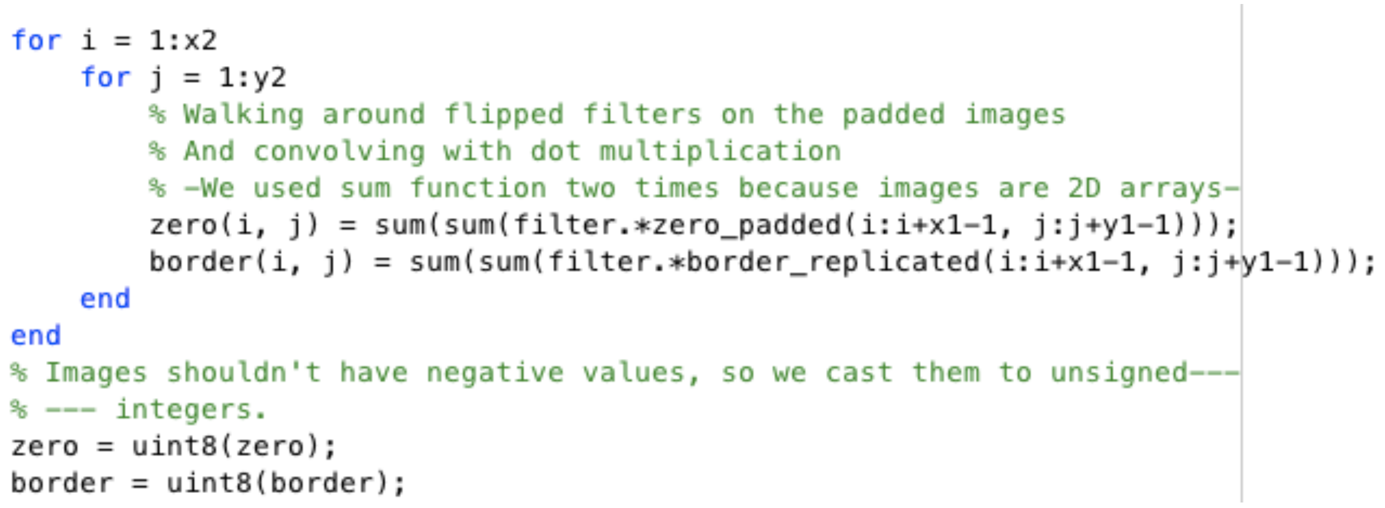
In this part, we create two matrices with all zeros at first with size:

(image_row_size + filter_row_size - 1) x (image_column_size + filter_column_size - 1)

In order to apply convolution, image should have extra columns and rows as shown. Then, we put our original image to the matrix to up-left corner. Because we assumed that our kernel's initial point is (1, 1).

For border replication, we need to repeat borders on bottom and right sides instead of zero. Thus, border replication has an extra line for padding.

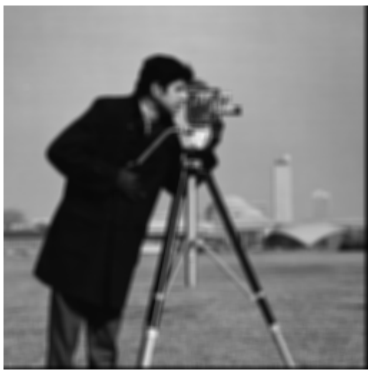
Convolution



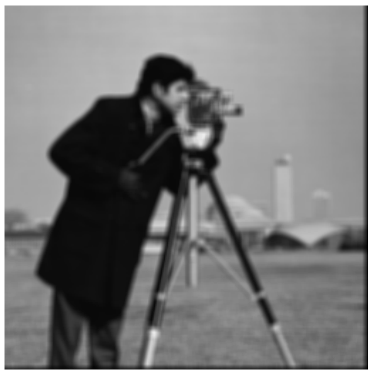
```
for i = 1:x2
    for j = 1:y2
        % Walking around flipped filters on the padded images
        % And convolving with dot multiplication
        % -We used sum function two times because images are 2D arrays-
        zero(i, j) = sum(sum(filter.*zero_padded(i:i+x1-1, j:j+y1-1)));
        border(i, j) = sum(sum(filter.*border_replicated(i:i+x1-1, j:j+y1-1)));
    end
end
% Images shouldn't have negative values, so we cast them to unsigned---
% --- integers.
zero = uint8(zero);
border = uint8(border);
```

To convolve the image with given kernel, we need 2 nested for loops in order to walk around and apply the filter to the image. We multiplied the flipped filter matrix with the corresponding filter-sized part of the image and sum the elements of product. We used sum function two times in order to sum all of the elements because dot product is a 2D array.

Sample Outputs

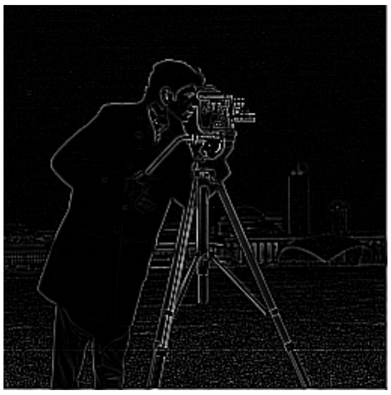


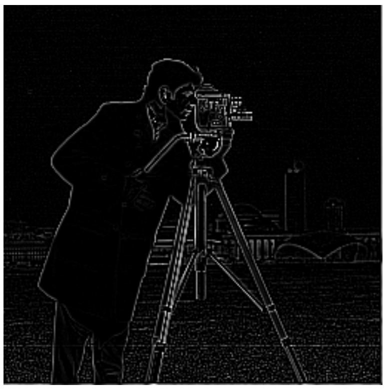
Gaussian filter with zero-padding



Gaussian filter with border replication

*Gaussian filter blurred this image.

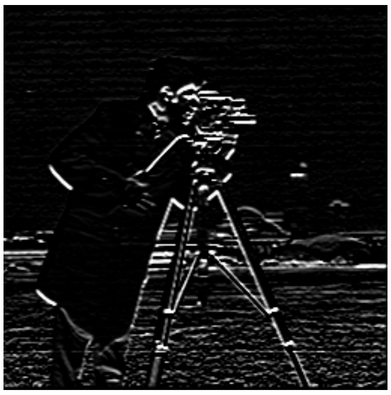


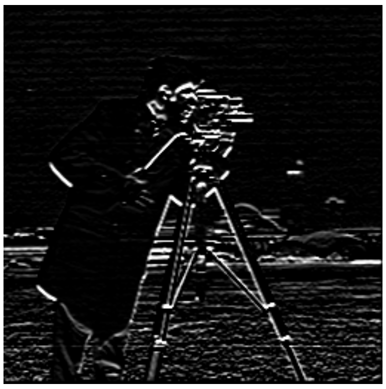


Laplacian filter with zero-padding

Laplacian filter with border replication

*Laplacian filter detected edges in this image very well.

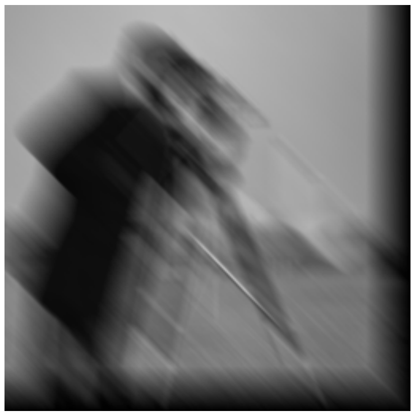


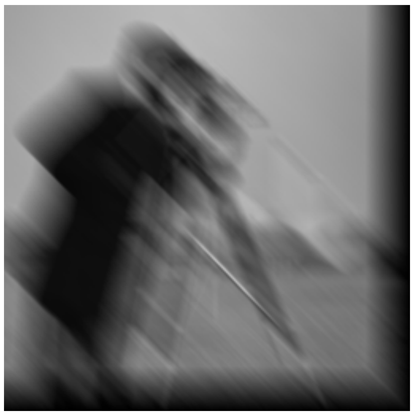


Sobel filter with zero-padding

Sobel filter with border replication

*Sobel filter detected edges but not as good as Laplacian filter, also it looks like it gave a 3D-looking to this image.





Motion filter with zero-padding

Motion filter with border replication

*Motion filter gave a moving looking to this image.

*** Black shades on the right and bottom sides are caused from we choose (1, 1) as the initial point of the kernel. Because of that we have paddings on the right and bottom sides.

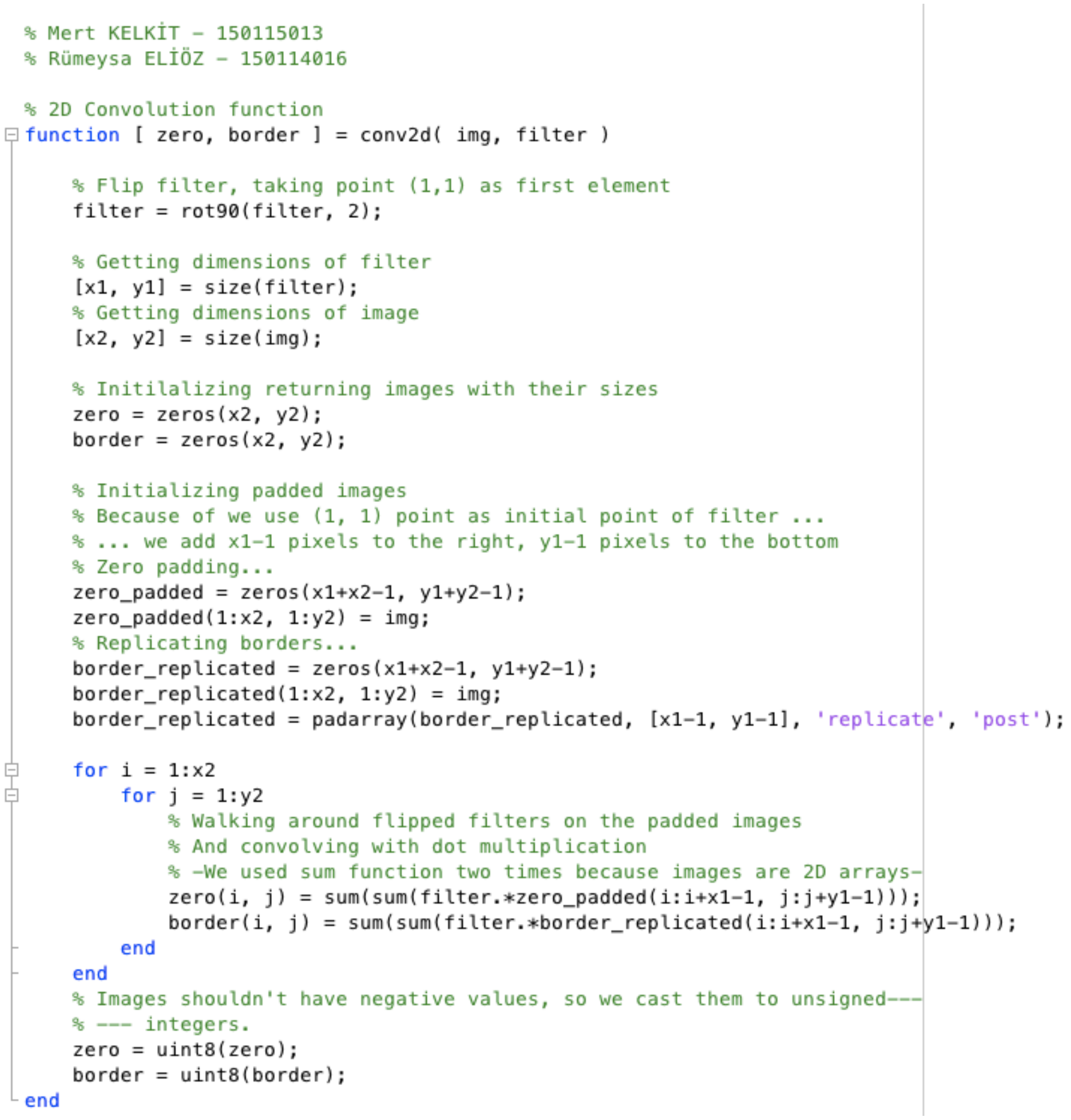
Source Codes

- myConv.m - Test script for conv2d function

```
% Mert KELKİT - 150115013
% Rümeysa ELİÖZ - 150114016

% Test script for conv2d function

% Getting kernels from fspecial
laplacian = fspecial('laplacian', 0.5);
gaussian = fspecial('gaussian', 5, 5);
sobel = fspecial('sobel');
motion = fspecial('motion', 40, 135);

% Sample image
img = imread('cameraman.tif');

% Getting images which are filtered by gaussian filter
% Gaussian filter blurs image
% 1 -> image with zero padding , 2 -> image with border replication
[gaussian_zero, gaussian_border] = conv2d(img, gaussian);

% Getting images which are filtered by laplacian filter
% Laplacian filter extracts edges efficiently
% 1 -> image with zero padding , 2 -> image with border replication
[laplacian_zero, laplacian_border] = conv2d(img, laplacian);

% Getting images which are filtered by sobel filter
% Sobel filter can be used for edge detection, but not as good as laplacian
% filter for this image
% 1 -> image with zero padding , 2 -> image with border replication
[sobel_zero, sobel_border] = conv2d(img, sobel);

% Getting images which are filtered by motion filter
% Image looks like moving with this filter
% 1 -> image with zero padding , 2 -> image with border replication
[motion_zero, motion_border] = conv2d(img, motion);

% Original filtered images, used MATLAB built-in imfilter function
original_laplacian = imfilter(img, laplacian);
original_gaussian = imfilter(img, gaussian);
original_sobel = imfilter(img, sobel);
original_motion = imfilter(img, motion);
```

```matlab
% Displaying images with our convolutions
figure()
subplot(2,4,1), imshow(gaussian_zero);
title('Gaussian filter with zero padding', 'Color', 'b');
subplot(2,4,2), imshow(laplacian_zero);
title('Laplacian filter with zero padding', 'Color', 'b');
subplot(2,4,3), imshow(sobel_zero);
title('Sobel filter with zero padding', 'Color', 'b');
subplot(2,4,4), imshow(motion_zero);
title('Motion filter with zero padding', 'Color', 'b');
subplot(2,4,5), imshow(gaussian_border);
title('Gaussian filter with replicated borders', 'Color', 'r');
subplot(2,4,6), imshow(laplacian_border);
title('Laplacian filter with replicated borders', 'Color', 'r');
subplot(2,4,7), imshow(sobel_border);
title('Sobel filter with replicated borders', 'Color', 'r');
subplot(2,4,8), imshow(motion_border);
title('Motion filter with replicated borders', 'Color', 'r');

% Displaying images which have been filtered with MATLAB built-in function
figure()
subplot(1,4,1), imshow(original_gaussian);
title('Original Gaussian filtered', 'Color', 'b');
subplot(1,4,2), imshow(original_laplacian);
title('Original Laplacian filtered', 'Color', 'b');
subplot(1,4,3), imshow(original_sobel);
title('Original Sobel filtered', 'Color', 'b');
subplot(1,4,4), imshow(original_motion);
title('Original Motion filtered', 'Color', 'b');

% Saving sample outputs
imwrite(gaussian_zero, 'gaussian_zero.png');
imwrite(gaussian_border, 'gaussian_border.png');
imwrite(laplacian_zero, 'laplacian_zero.png');
imwrite(laplacian_border, 'laplacian_border.png');
imwrite(sobel_zero, 'sobel_zero.png');
imwrite(sobel_border, 'sobel_border.png');
imwrite(motion_zero, 'motion_zero.png');
imwrite(motion_border, 'motion_border.png');
```

- conv2d.m - conv2d function for convolution



```matlab
% Mert KELKİT - 150115013
% Rümeysa ELİÖZ - 150114016

% 2D Convolution function
function [ zero, border ] = conv2d( img, filter )

    % Flip filter, taking point (1,1) as first element
    filter = rot90(filter, 2);

    % Getting dimensions of filter
    [x1, y1] = size(filter);
    % Getting dimensions of image
    [x2, y2] = size(img);

    % Initilalizing returning images with their sizes
    zero = zeros(x2, y2);
    border = zeros(x2, y2);

    % Initializing padded images
    % Because of we use (1, 1) point as initial point of filter ...
    % ... we add x1-1 pixels to the right, y1-1 pixels to the bottom
    % Zero padding...
    zero_padded = zeros(x1+x2-1, y1+y2-1);
    zero_padded(1:x2, 1:y2) = img;
    % Replicating borders...
    border_replicated = zeros(x1+x2-1, y1+y2-1);
    border_replicated(1:x2, 1:y2) = img;
    border_replicated = padarray(border_replicated, [x1-1, y1-1], 'replicate', 'post');

    for i = 1:x2
        for j = 1:y2
            % Walking around flipped filters on the padded images
            % And convolving with dot multiplication
            % -We used sum function two times because images are 2D arrays-
            zero(i, j) = sum(sum(filter.*zero_padded(i:i+x1-1, j:j+y1-1)));
            border(i, j) = sum(sum(filter.*border_replicated(i:i+x1-1, j:j+y1-1)));
        end
    end
    % Images shouldn't have negative values, so we cast them to unsigned---
    % --- integers.
    zero = uint8(zero);
    border = uint8(border);
end
```

References

- mathworks.com
- stackoverflow.com