Image Analysis
Exercise 04 - Filtering

2019

Introduction

The purpose of this exercise is to illustrate different image filtering techniques. You will try methods for smoothing and enhancing different structures in a noisy image.

You are encouraged to create an .m file for each exercise. To create an empty M-file write:

```
edit Filtering.m
```

Write the code to solve the exercises in the M-file. The M-file could look like this:

```
clear; clc; close all; % Clear workspace and figures
% Exercise 1
f = [zeros(5,3),ones(5,2)]
h = ones(3,3)
g = imfilter(f,h)

% Exercise 2
...
```

Hint: You can run a part of an M-file by hitting `F9` after selecting the part you want to run. Pressing `ctrl+enter` (`cmd+enter` for Mac) will run the current session and pressing `F5` will run the entire M-file.

You can use `%%` to define a new Matlab section.

Filtering using Matlab

Matlab presents a large variety of different image filtering functions. The Matlab filtering function `imfilter` function performs general image filtering. The function takes a filter-kernel `h` and an image `f`. Take a look of the documentation for `imfilter`.

Exercise 1 *Try:*

```
f = [zeros(5,3),ones(5,2)]
h = ones(3,3)
```

Now, `f` *is a (very small) 2D image and* `h` *is a* $3x3$ *filter kernel. Try filtering* `f` *with* `h`*:*

```
g = imfilter(f,h)
```

What is the result at $(r, c) = (2, 4)$*. Explain why.*

Alternatively, try to use Matlabs array editor to edit your image `f` *or your filter* `h`*. This is done by double clicking the variable in the variable workspace. This opens the editor and you can change the values of each element (pixel).*

General filtering

Border Handling

Initially, we will work with an artificial image to get an understanding of filtering. Start by reading it an showing it:

```
im1 = imread('Gaussian.png');
imshow(im1);
```

Exercise 2 *Create a simple kernel for a mean filter:*

```
fsize = 5;
h = ones(fsize)/fsize^2;
```

Examine the matrix `h`*. Why should the size of the kernel be odd?*

Exercise 3 *Apply the filter* `h` *on the image* `im1`*. Store the filtered image in a variable called* `meanim1`*. Display both images in the same figure by using:*

```
figure
subplot(1,2,1);
imshow(im1), colormap(gca,gray), axis image off;
title('Original image')
subplot(1,2,2);
imshow(meanim1), colormap(gca,gray), axis image off;
title('Filtered image, mean filter')
```

Try to do the same with different colormaps (jet, pink, hot etc.) and find one where you can clearly see the edges.

When the value of an output pixel at the boundary of the image is computed, a portion of the filter is usually outside the edge of the input image. The `imfilter` normally fills in these *off-the-edge pixels* by assuming they are 0. This is called zero padding. Since 0 is the value of a black pixel, the output image will have a dark edge as seen in the previous result.

Exercise 4 *Find out how to use* border replication *with* `imfilter`. *Filter* `im1` *with* `h` *using border replication. Store the filtered image in a variable called* `meanim2`.

Show `meanim1` *and* `meanim2` *in the same figure and compare them. Where are they different and why?*

Noise reduction

Show the original image `im1` and one of the filtered images `meanim1` or `meanim2`.

Exercise 5 *Compare the two images: What do you see? Is there noise in the original image? Has it been reduced in the mean filtered version?*

Exercise 6 *Vary the size of* `h` *from 5x5 to 15x15 and filter im1. What happens with the noise when the size increases?*

Exercise 7 *What happens with the edges in the image when the mean kernel is chosen to be large* `[15x15]`*? **Remember that an edge is where there is a large change in the image intensity** (an edge is not the same as the border of the image).*

In Matlab a median filter is written: `medfilt2`. Take a look at the help.

Exercise 8 *Try using the median filter on* `im1`. *Compare the result with the result of the mean filter. What difference can you see?*

Exercise 9 *Try changing the size of the kernel for the median filter and the mean filter. For the median filter this is done by:*

```
medim2 = medfilt2(im1, [15 15]);
```

for a `[15x15]` *kernel.*

Exercise 10 *What happens when the kernels gets larger? What happens with the mean, and median filtered images. Explain what you see!*

Exercise 11 *Try to use both the mean filter and the median filter on the image called* `SaltPepper.png` *(an image with salt and pepper noise). Show the results and explain the difference.*

Standard filter routines

Matlab includes the built-in function `fspecial` which provides various standard filter types.

Exercise 12 *Take a look at the help file for* `fspecial`*. Try to call* `fspecial` *with 'average' and check if the produced filter compares to your previously defined mean filter,* `h`*. Try to vary the* `hsize` *argument to get different sizes of filter kernels.*

Edge detection

Filtering can often be applied for edge detection. Take a look at the Sobel and Prewitt operators by calling `fspecial`. Both of these kernels perform a gradient estimation that can be used to segment the edges in the image. This is because edges correspond to large gradients in the image. Each filter applied once calculates the gradient in a single image direction.

Try to take a look at the filter kernel and its values so you have an understanding of what it does.

In this exercise we will work with slice from a CT scan of a human elbow.

```
CT = imread('ElbowCTSlice.png');
```

the goal of the exercise is to find a filter that highlights the edges of the bones the best possible.

Exercise 13 *Use the Sobel or Prewitt operators to calculate the magnitude of the gradient in the original image* `CT`*.*

Show the output images using

```
imshow(CT2, []);
```

where the `[]` *makes the command scale the gray values automatically. Try playing around with the colormap (*`colormap(gca, jet)` *for example). What edges are most visible?*

Exercise 14 *Now use* `h'` *and* `-h'` *to rotate the kernels 90 and -90 degrees. Take a look at the kernels and the values to get an understanding of the differences. What edges are now visible after filtering?*

Exercise 15 *Apply the edge filters to mean filtered version of* `CT`*. Try with kernels of size* `[5x5]` *and* `[13x13]`*. What influence do the mean filter have on the output of the edge filters?*

Exercise 16 *Apply the edge filters to median filtered version of* `CT`*. Try with kernels of size* `[5x5]` *and* `[13x13]`*. Does it give better results than the average filter?*

Take a look at the help file for Matlabs `edge` function. This function provides edge detection using various filtering routines, among others the Sobel and Prewitt operators.

Exercise 17 *Try to run the* `edge` *on the original image. Also try to play around with the options for the edge filter.*

Gaussian Filtering

The Gaussian filter is widely used in image processing. It is a smoothing filter that removes high frequencies from the image.

Exercise 18 *Try to create a Gaussian filter kernel using* `fspecial` *with* `hsize=17` *and* `sigma=3` *and call it* `G`*. Visualise the kernel using the* `surf` *function:*

```
surf(G);
```

The plot can be rotated by the mouse by pressing the `rotate 3D` *icon in the menu.*

Exercise 19 *Filter the CT slice image using Gaussian filters with* `sigma=1`*,* `sigma=3`*, and* `sigma=11`*. Use* `hsize=51` *for the last sigma. How do the sigma change the output image?*

Filtering your own image

Take some images with your camera and download them to your Matlab directory.

Exercise 20 *Convert your image to grayscale using* `rgb2gray` *and resize the image (to make it smaller and faster to work with):*

```
my3 = imresize(my2, [1000 NaN])
```

where `my2` *is the output of the colour-grayscale conversion.*

Exercise 21 *Try using the* `Prewitt`*,* `Sobel`*,* `edge`*, and* `Gaussian` *filters on your image.*