

Visão por computador
Computação Gráfica

Mestrado Integrado em Engenharia Informática - 4º ano
Tutorial 1: Image Filtering and Edge Detection
Universidade do Minho

Bernardo Silva - a77230
Nuno Reis - a77310

Contents

1 Design a program to smooth images	2
1.1 Create a function to introduce noise in the image	2
1.2 Smoothing using Average, Gaussian or Median filters in Spatial Domain	3
1.2.1 Average filter Smoothing	4
1.2.2 Gaussian filter Smoothing	6
1.2.3 Median filter Smoothing	8
1.3 Compute the Discrete Fast Fourier Transform (DFT)	10
1.4 Smoothing using Butterworth and Gaussian filters in frequency domain	12
1.4.1 Gaussian filter Smoothing	12
1.4.2 Butterworth filter Smoothing	14
2 Design a program to detect edges using Canny Detector	16
2.1 Gaussian Smoothing	16
2.2 Computing the horizontal and vertical gradients	17
2.3 Computing the magnitude of the gradient	17
2.4 Non-Maximum Suppression	18
2.5 Double Thresholding	18
2.6 Hysteresis Thresholding	19
3 Conclusion	19

1 Design a program to smooth images

Nesta fase foi-nos pedido o desenvolvimento que gerasse imagens com ruído "Salt and Pepper" ou "Gaussian" e que investigássemos os efeitos dos filtros de smoothing e da aplicação da transformada de fourier nestas imagens.

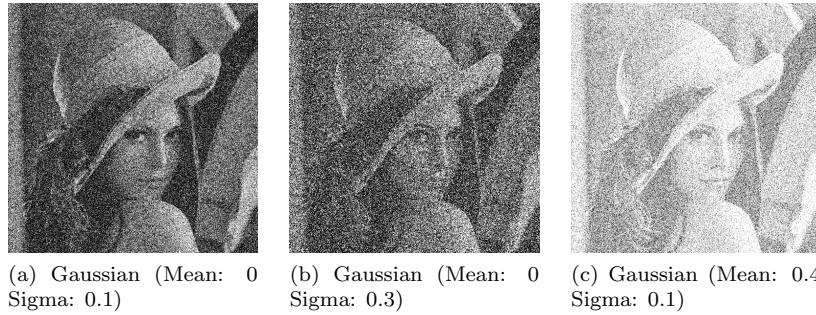
1.1 Create a function to introduce noise in the image

Utilizando a função presente no matlab *imnoise()* gerou-se imagens com "Salt and Pepper" e "Gaussian" noise, tal como definidas nos seguintes exemplos:



(a) Salt and Pepper (Density: 0.1) (b) Salt and Pepper (Density: 0.2)

Como se pode observar, no filtro Salt and Pepper, aumentar a densidade aumenta a probabilidade de um ponto ser preto ou branco.



No caso do filtro Gaussiano, modificar o valor da média faz com que a curva normal do filtro se afaste do valor atual do pixel, ou seja, sendo a média positiva, o filtro faz com que a imagem se torne mais iluminada. O sigma, que neste caso é a raíz quadrada da variância, faz com que píxeis mais afastados do pixel original sejam mais comuns, dando um tom mais ríspido ao ruído.

Desta maneira, decidiu-se que as propriedades de ruído que iríamos tratar de alisar no próximo passo seriam **Salt and Pepper: Density - 0.1** e **Gaussian: Mean - 0 Sigma - 0.1**

1.2 Smoothing using Average, Gaussian or Median filters in Spatial Domain

Para ao desenvolvimento das funções de smoothing, utilizámos funções pre-definidas do Image Processing Toolbox do matlab. Aplicámos cada tipo de smoothing aos noises predefinidos no passo anterior e compará-mo-los, de forma a observarmos as diferenças e a escolher a melhor opção.

1.2.1 Average filter Smoothing

Para o filtro average utilizámos a função *fspecial* com a tag 'average', aplicando posteriormente o filtro à imagem com a função *imfilter*.

De seguida vão-se poder observar os efeitos causados por Average Filter Smoothing usando tamanhos de filtro de 5, 10 e 20.



Figure 1: Imagem com Ruído Salt and Pepper com Desidade - 0.1



(a) Average Smoothing Filter Size: 5



(b) Average Smoothing Filter Size: 10



(c) Average Smoothing Filter Size: 20

Pode-se observar que o filtro consegue esconder o ruído decentemente mas em contrapartida a imagem também fica muito desfocada.



Figure 2: Imagem com Ruído Gaussian com Mean - 0 e Sigma - 0.1



(a) Average Smoothing -
Filter Size: 5



(b) Average Smoothing -
Filter Size: 10



(c) Average Smoothing -
Filter Size: 20

Por outro lado, este consegue lidar decentemente com gaussian noise, aproximando cada pixel do centro da curva normal causada pelo ruído.

1.2.2 Gaussian filter Smoothing

Para o filtro gaussian utilizámos a função *imgaussfilt*, com as tags 'FilterSize' e 'FilteredDomain', sendo 'FilterDomain' spatial.

De seguida vão-se poder observar os efeitos causados por Gaussian Filter Smoothing usando tamanhos de filtro de 5, 11 e 21. Era necessária a utilização de valores ímpares para o tamanho do filtro, por isso foram utilizados os mais aproximados possíveis.



Figure 3: Imagem com Ruído Salt and Pepper com Desidade - 0.1



(a) Gaussian Smoothing -
Filter Size: 5 Sigma: 3



(b) Gaussian Smoothing -
Filter Size: 11 Sigma: 3



(c) Gaussian Smoothing -
Filter Size: 21 Sigma: 3



Figure 4: Imagem com Ruído Gaussian com Mean - 0 e Sigma - 0.1



(a) Gaussian Smoothing -
Filter Size: 5 Sigma: 3



(b) Gaussian Smoothing -
Filter - Size: 11 Sigma: 3



(c) Gaussian Smoothing -
Filter - Size: 21 Sigma: 3

1.2.3 Median filter Smoothing

Para o filtro median utilizámos a função *medfilt2*, definindo apenas o tamanho do filtro.

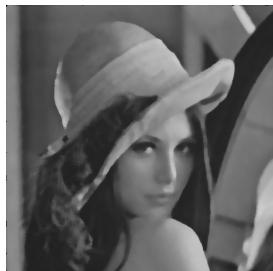
De seguida vão-se poder observar os efeitos causados por Median Filter Smoothing usando tamanhos de filtro de 5, 10 e 20.



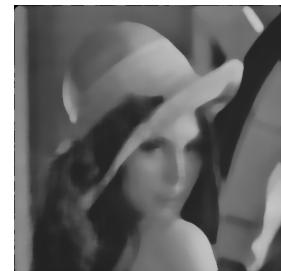
Figure 5: Imagem com Ruído Salt and Pepper com Desidade - 0.1



(a) Median Smoothing -
Filter Size: 5



(b) Median Smoothing -
Filter Size: 10



(c) Median Smoothing -
Filter Size: 20

Como se pode observar este foi o filtro que lidou melhor com o ruído salt and pepper, tendo-o eliminado completamente da imagem, estando esta completamente nítida, quando o tamanho do filtro é 5.



Figure 6: Imagem com Ruído Gaussian com Mean - 0 e Sigma - 0.1



(a) Median Smoothing -
Filter Size: 5



(b) Median Smoothing -
Filter Size: 10



(c) Median Smoothing -
Filter Size: 20

Por outro lado, este filtro não consegue lidar tão bem com o ruído gaussiano, desfocando a imagem devido a aleatoriedade do algoritmo de geração deste ruído.

1.3 Compute the Discrete Fast Fourier Transform (DFT)

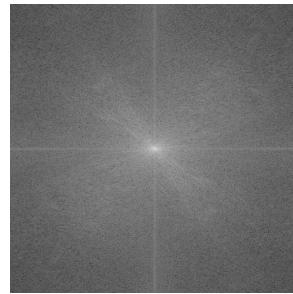
Para a geração da Transformada de Fourier, obteve-se a transformada usando a função predefinida `fft2()`, de seguida extraiendo a parte real da matriz e usando o `log(matriz+1)` para aumentar a visibilidade da transformada.

Desta maneira observaram-se os seguintes resultados:

Imagen Original:



(a) Imagem Original



(b) Transformada de Fourier

Imagens com ruído:



(a) Ruído Salt and Pepper



(b) Transformada de Fourier



(c) Ruído Gaussian

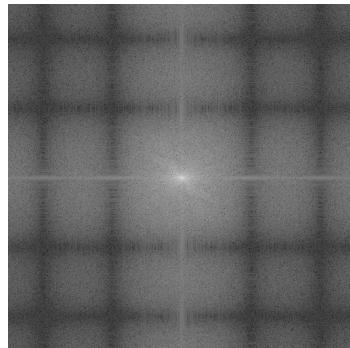


(d) Transformada de Fourier

Imagens com ruído após smoothing:



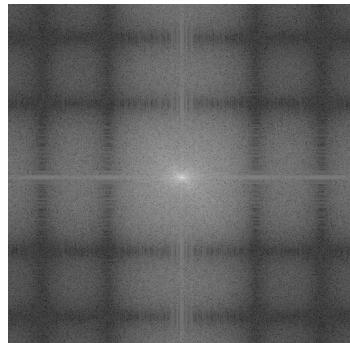
(a) Average Smoothing



(b) Transformada de Fourier



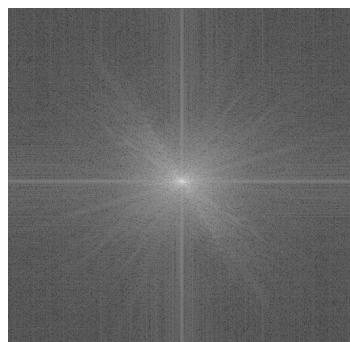
(c) Gaussian Smoothing



(d) Transformada de Fourier



(e) Median Smoothing



(f) Transformada de Fourier

Após estes testes, observa-se que nas imagens com ruído, a transformada é muito mais aleatória do que nas imagens originais. Por outro lado, nas imagens alisadas, observa-se uma ordem na transformada provavelmente proveniente do algoritmo usado para as alisar.

1.4 Smoothing using Butterworth and Gaussian filters in frequency domain

Para o desenvolvimento das funções de smoothing em frequency domain, utilizámos funções predefinidas do matlab para Gaussian Smoothing e criámos o algoritmo de Butterworth smoothing.

Aplicámos cada tipo de smoothing aos noises predefinidos anteriormente.

Em geral o processo de smoothing em frequency domain é mais lento pois é necessária a geração da transformada de fourier seguida do tratamento da matriz e inverter novamente.

1.4.1 Gaussian filter Smoothing

Para o filtro gaussian utilizámos a função *imgaussfilt*, com as tags 'FilterSize' e 'FilteredDomain', sendo 'FilterDomain' frequency.

De seguida vão-se poder observar os efeitos causados por Gaussian Filter Smoothing usando tamanhos de filtro de 5, 11 e 21.



Figure 7: Imagem com Ruído Salt and Pepper com Desidade - 0.1



(a) Gaussian Smoothing -
Filter Size: 5 Sigma: 3



(b) Gaussian Smoothing -
Filter Size: 11 Sigma: 3



(c) Gaussian Smoothing -
Filter Size: 21 Sigma: 3



Figure 8: Imagem com Ruído Gaussian com Mean - 0 e Sigma - 0.1



(a) Gaussian Smoothing -
Filter Size: 5 Sigma: 3



(b) Gaussian Smoothing -
Filter - Size: 11 Sigma: 3



(c) Gaussian Smoothing -
Filter - Size: 21 Sigma: 3

Em geral pode-se observar que o gaussian smoothing no domínio da frequência tendeu para obter mais sucesso do que no domínio espacial.

1.4.2 Butterworth filter Smoothing

Para o filtro Butterworth criámos o nosso próprio algoritmo.



Figure 9: Imagem com Ruído Salt and Pepper com Desidade - 0.1



(a) Butterworth Smoothing
- Cutoff Frequency: 50 Order: 1



(b) Butterworth Smoothing
- Cutoff Frequency: 50 Order: 5



(c) Butterworth Smoothing
- Cutoff Frequency: 20 Order: 5



Figure 10: Imagem com Ruído Gaussian com Mean - 0 e Sigma - 0.1



(a) Butterworth Smoothing
- Cutoff Frequency: 50 Order: 1



(b) Butterworth Smoothing
- Cutoff Frequency: 50 Order: 5



(c) Butterworth Smoothing
- Cutoff Frequency: 20 Order: 5

Neste tipo de filtro pode-se observar o ringing effect quando a ordem do algoritmo aumenta. Além disso pode-se observar que uma frequência demasiado baixa torna a imagem desfocada e aumentar a ordem torna o contraste entre as várias cores da imagem maior.

2 Design a program to detect edges using Canny Detector

2.1 Gaussian Smoothing

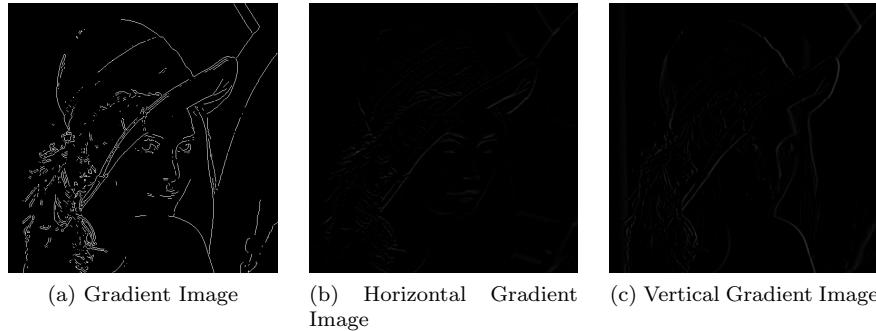
Utilizando uma das funções do programa anterior gerou-se uma imagem alisada com um filtro de Gauss com um tamanho de filtro de 5 pixels e sigma igual a 3.



Figure 11: Gaussian Smoothed Image

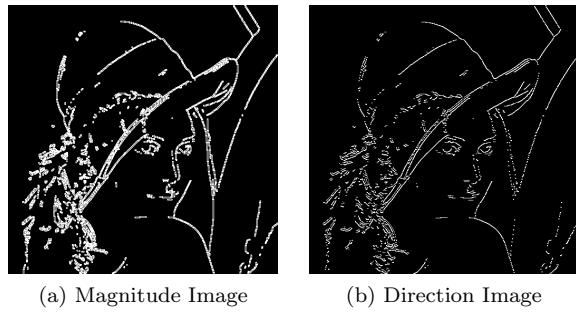
2.2 Computing the horizontal and vertical gradients

Usando os algoritmos nativos da ferramenta de MatLab, consegui-se, de forma simples chegar aos resultados pretendidos. A função *edge* facilita a resolução deste exercício visto que tem como output todos os valores pretendidos.



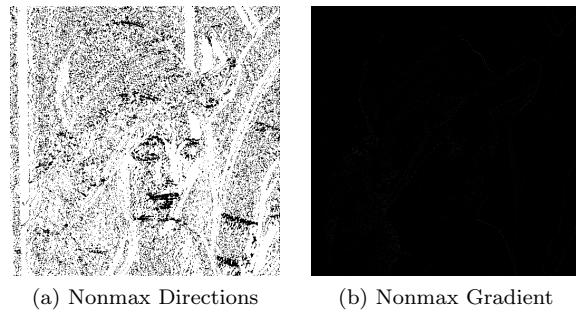
2.3 Computing the magnitude of the gradient

Para calcular a magnitude e direção do gradiente usou-se a função predefinida *imgradient* que retorna a matriz magnitude e a matriz direção correspondente à imagem.



2.4 Non-Maximum Suppression

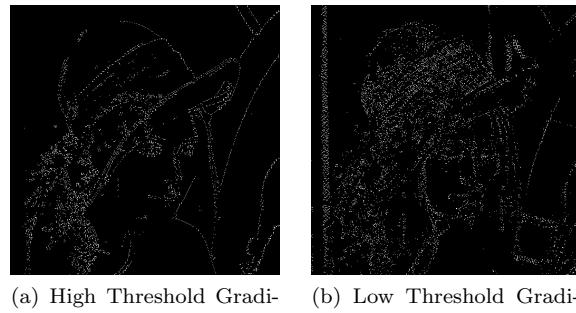
Usando os algoritmos desenvolvidos nas aulas, foi implementada a supressão de edges provenientes de possíveis artefactos gráficos(noise). O algoritmo, em si, consiste na normalização da matriz de direcções segundo certos angulos ($0,45,90,135$) e, a utilização destes em conjunto com a matriz magnitude para obter uma melhor deteção das chamadas "true edges".



(a) Nonmax Directions (b) Nonmax Gradient

2.5 Double Thresholding

Usando certos valores, no nosso caso 0.075 e 0.175, foram implementados limites superiores e inferiores de modo a identificar Soft e Harsh edges.



(a) High Threshold Gradient (b) Low Threshold Gradient

2.6 Hysteresis Thresholding

Usando os thresholds acima apresentados e os gradientes de Soft e Harsh edges, fez-se a interpolação das vizinhanças dos pontos correspondentes às Soft edges, de modo a saber se eram fortes o suficiente para fazerem parte do produto final. As Harsh edges, no entanto, são instantaneamente designadas como essenciais.

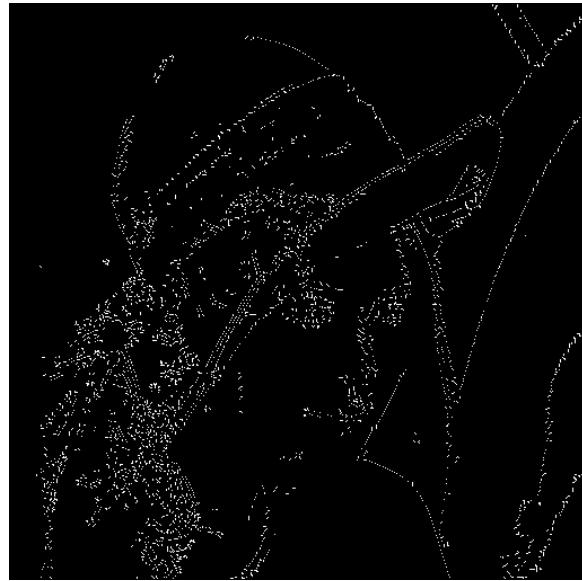


Figure 12: Hysteresis Thresholding

3 Conclusion

Devido às dificuldades apresentadas na segunda parte do trabalho não foi possível o cumprimento exato da data de entrega do tutorial, . No entanto, a pequena quantidade de tempo extra garantiu a completa integridade deste trabalho. Contudo, a realização deste tutorial permitiu-nos interiorizar a informação que nos é prescindida nas aulas, desta maneira criando uma fundação para situações futuras.