

Computer Vision

Image Filtering and Edge Detection

A64282-Paulo Alves

University of Minho, Department of Informatics,
4710-057 Braga, Portugal

Resumo Este relatório trata da implementação em *MATLAB* e respectivos resultados de algoritmos no âmbito de visão por computador de filtros de imagem e detetores de contornos, bem como resultados experimentais da aplicação dos mesmos a várias imagens.

1 Introdução

O presente relatório, no âmbito da unidade curricular de **Visão por Computador** do curso de Mestrado Integrado em Engenharia informática da Universidade do Minho, vai tratar da implementação em *MATLAB* e experimentação de duas partes, uma onde se introduz ruído numa imagem convertida para *grey-scale*. Posteriormente à mesma imagem aplicam-se filtros, **Gaussianos, de média e mediana**, tanto em domínio espacial como, no caso do filtro **Gaussiano e Butterworth**, no domínio das frequências.

A segunda parte trata de um algoritmo de deteção de contornos do tipo *Canny*, onde se utilizaram implementações da primeira parte para o processo de introdução de ruído e depois filtragem com um filtro **Gaussiano** numa imagem *grey-scaled*.

2 I - Introdução de Ruído e Filtragem de Imagens

A primeira parte do trabalho trata das funções e scripts que introduzem ruído de diversos tipos nas imagens e aplicam filtros do tipo **Média, Mediana e Gaussiano**. Apresenta-se neste capítulo como foi feita a implementação, e como é o algoritmo, de cada um em *MATLAB*.

2.1 Introdução de Ruído

A primeira parte começa pelo função/script que introduz ruído numa imagem, o processo é o seguinte :

- Carregar imagem .
- Converter a imagem para *grey-scale* .
- Chamando a função que filtra, o utilizador tem várias opções.Gerar ruído do tipo **salt and pepper** , ou ruído do tipo **Gaussiano** . Caso escolha **salt and pepper** é ainda possível seleccionar uma densidade para o ruído de 0 até 1, quanto maior a densidade, maior ruído se gera.

Caso seja escolhido o tipo **Gaussiano**, é ainda possível seleccionar como é gerado esse ruído com um parâmetro extra , que pode ser com média, média e variância ou nenhum dos dois .

O que isto permite é que o utilizador use a função padrão **gaussiana**, ou use uma média ou média e variância ao seu gosto.

- Depois de gerado o ruído, a imagem é guardada.
- O formato para o nome da imagem que é guardada é :

(nome original da imagem)+(tipo de ruído)+(parâmetros extra escolhidos).png .

2.2 Filtros no Domínio espacial

Filtro de Média O processo para esse filtro é o seguinte :

- Depois de ter uma imagem já convertida para *grey-scale*,utilizamos um kernel de tamanho axb, por exemplo 3x3;5x5;7x7.
- No elemento central colocamos a media dos elementos vizinhos.
- Percorre-se toda a imagem com o kernel.

2.3 Filtro de Mediana

Igual ao filtro da média, onde temos um kernel de tamanho axb e depois de ordenar os valores da vizinhança, o valor médio no caso de um kernel ímpar, ou a media dos dois valores centrais no caso do kernel ser par. Esse valor central é o valor que vai ter o pixel, e aplica-se a toda a imagem tal como o da média.

2.4 Filtro Gaussiano

Semelhante ao filtro da média, porém com a forma da função **Gaussiana** para os valores do kernel. A função **Gaussiana** tem a forma.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figura 1: Função Gaussiana para 2D

O processo então é o seguinte :

- Escolher o tamanho do kernel axb.
- Escolher um desvio padrão "sigma".
- Normalizar os valores do kernel utilizando um somatório.
- Introduzir padding de zeros na imagem original.
- Aplicar o kernel para a imagem utilizando convolução .

2.5 DFT - Transformada de Fourier

Antes de continuar e passar para filtros no domínio das frequências, temos que obviamente converter os valores que estão no domínio espacial para o domínio da frequencia. Para isso foi criada uma função para calcular a transformada de Fourier .

Vale tambem notar que depois para se representar outra vez as imagens vai ser necessário tambem calcular a IFDT, ou seja a inversa da transformada de Fourier para retornar ao domínio espacial. Também vale notar que nestas aplicações para melhor visualização foram feitas separações nos espectros pois contém números complexos e muitas vezes não são fáceis de interpretar.

2.6 Filtros no Domínio da frequência

Foram implementados filtros que trabalham não sobre os pixeis em si, mas sim sobre as frequências dos sinais que foram convertidos com a tranformada de Fourier .

Filtro Butterworth Neste trabalho foi implementado um filtro Butterworth do tipo HP(*high-pass*), para se fazer o mesmo para LP(*low-pass*) simplesmente teríamos que fazer (1-HP) . Na prática o que vamos fazer é de forma "suave" atenuar frequências acima de um valor designado por *cutoff*.

A função a utilizar é a seguinte :

$$H(u, v) = \frac{1}{1 + [D_0/\sqrt{u^2 + v^2}]^{2n}}$$

Do is the cutoff frequency
n is the order of the filter

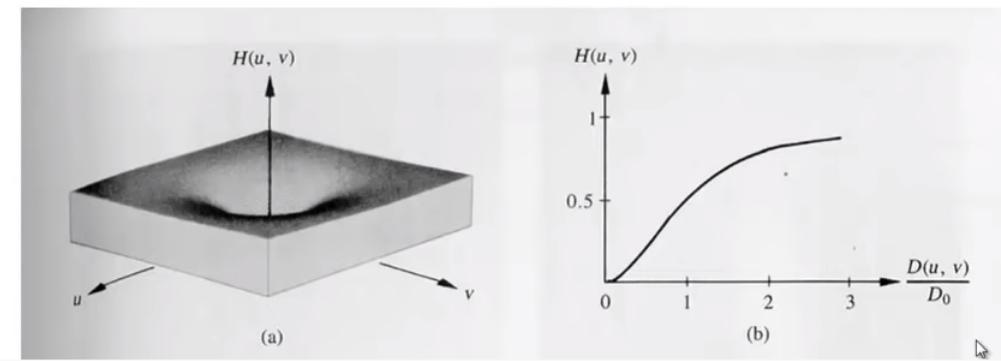


Figura 2: Butterworth - High Pass

Temos então o seguinte processo :

- Receber uma imagem *grey-scale* .
- Escolher um valor para a *cutoff frequency*
- Escolher um valor para a ordem do filtro ('n')
- Aplicar a formula e calcular a matriz axb(a e b são a altura e largura da imagem) que tem os valores.
- Aplicar DFT a imagem(a componente DC é centrada na origem)
- Convolver os valores do filtro com as frequências da imagem .
- fazer IDFT (Inverso da transformada de Fourier para voltar ao domínio espacial).

3 II - Canny Edge Detector

Nesta parte foi feito um script para fazer deteção de contornos, do tipo *Canny*. O processo é o seguinte :

- Tendo uma imagem *grey-scale* onde ja foi aplicado, ou a imagem por algum motivo tinha ruído, fazemos o *smoothing* utilizando os scripts criados anteriormente para reduzir o ruido e suavizar a imagem, o que vai fazer com que o detetor de contornos seja mais efectivo .

Neste caso sera usado um filtro **Gaussiano** para fazer smoothing

- O detetor de contornos(Canny) diz que a melhor maneira, mais perto do filtro ideal, de se calcular os contornos vai ser usando os máximos locais da magnitude do gradiente da imagem na direção do gradiente . Isso vai nos dar um maior número de contornos "reais".

Isto significa que o primeiro passo é calcular o a magnitude do gradiente e orientação do gradiente da imagem usando derivadas parciais.

- Aplicar o que se chama de *non-maximum suppression*, que so escolhe pixeis que são maximos locais seguindo o gradiente da imagem.

Utiliza-se isso para afinar os contornos pois contornos "grossos"não vão dar bons resultados na deteção .

Para isso temos que escolher a melhor orientação para os contornos em relação com a orientação do gradiente . O calculo das orientações é feito com a função *atan* do *MATLAB*.

Orientações negativas são ajustadas para ficarem positivas .

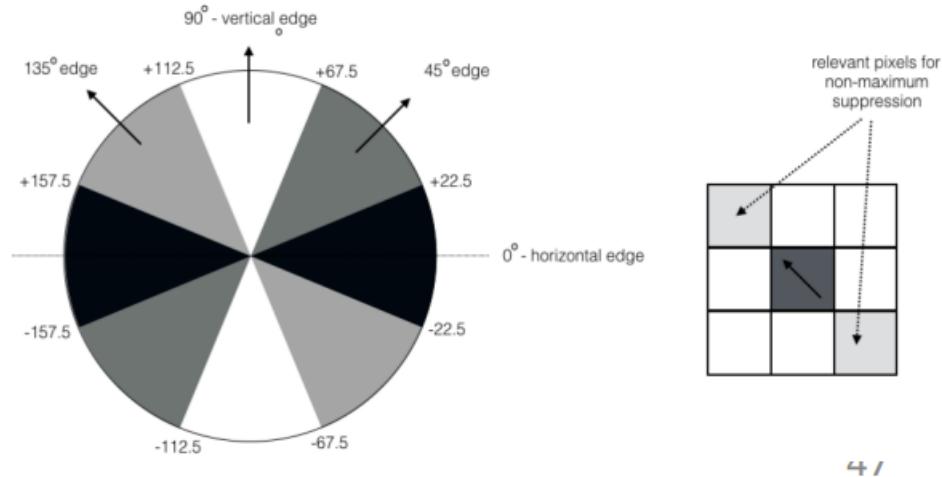


Figura 3: Os pixels relevantes dada a orientação do gradiente.

- Para evitar a captura de falsos contornos usamos um *hysteresis thresholding*, que diz que, um contorno real mas que ficou muito "fraco/fino" por causa do processo de *non-maximum suppression*, pode ser diferenciado em relação a contornos falsos gerados por ruidos e etc. porque os falsos não estão conectados a outros contornos "reais".
Ou seja os contornos são separados em regiões e caso hajam contornos "fracos" conectados a contornos "fortes", estes são escolhidos como contornos reais e os que não tem conexão são descartados.

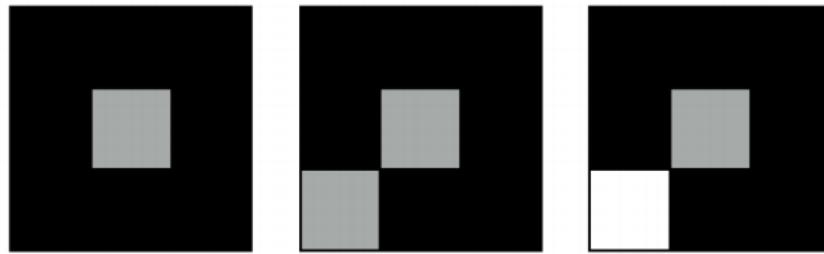


Figura 4: Na figura do meio como o pixel não esta conectado a nenhum "contorno forte", é des-
cartado como parte do contorno real, enquanto que na ultima figura é contabilizado por estar
conectado com um pixel que faz parte de um contorno "forte"(pixel mais claro/branco)

4 Resultados



Figura 5: Figura original convertida para *grey-scale*

Ruido



Figura 6: Ruído Salt and Pepper com intensidade 0.3



Figura 7: Ruído Salt and Pepper com intensidade 0.04



Figura 8: Ruído Salt and Pepper com intensidade 0.008



Figura 9: Ruido Gaussiano



Figura 10: Ruido Gaussiano com média 0.3



Figura 11: Ruido Gaussiano com média 0.3 e desvio padrão de 0.01



Figura 12: Ruido Gaussiano com média 0.3 e desvio padrão de 0.4

4.1 Filtros no domínio espacial

Vale notar que aumentar o sigma no filtro *Gaussiano* aumenta o "blur" da imagem .O mesmo acontece no filtro da média ao aumentar o kernel.

Em termos de diminuir o efeito de blurring e ao mesmo tempo eliminar bastante ruido, o da mediana é um dos melhores entre estes 3.



Figura 13: Filtro da média numa imagem com ruido SP de intensidade 0.4



Figura 14: Filtro da média numa imagem com ruido SP de intensidade 0.08



Figura 15: Filtro da mediana numa imagem com ruido SP de intensidade 0.4



Figura 16: Filtro da mediana numa imagem com ruido SP de intensidade 0.08



Figura 17: Filtro gaussiano de kernel 3x3 e sigma 1 numa imagem com ruido SP de intensidade 0.08

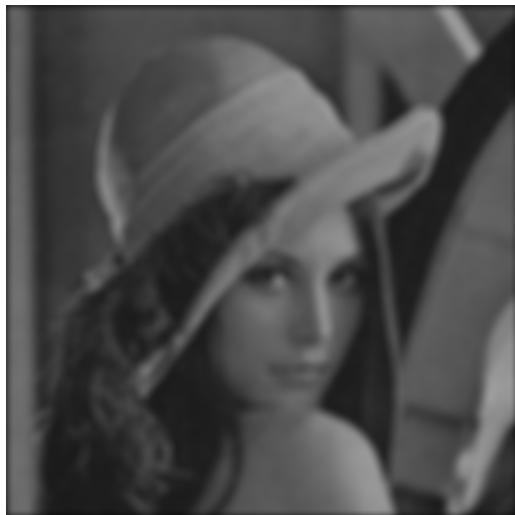


Figura 18: Filtro gaussiano de kernel 7x7 e sigma 4 numa imagem com ruido SP de intensidade 0.08

4.2 Canny Edge detection

Imagens à esquerda são do Canny Edge detector com *non-maximum suppression*, e à direita com *non-maximum suppression* e também *hysteresis thresholding*



Figura 19: Canny na imagem Original passada para *grey-scale*

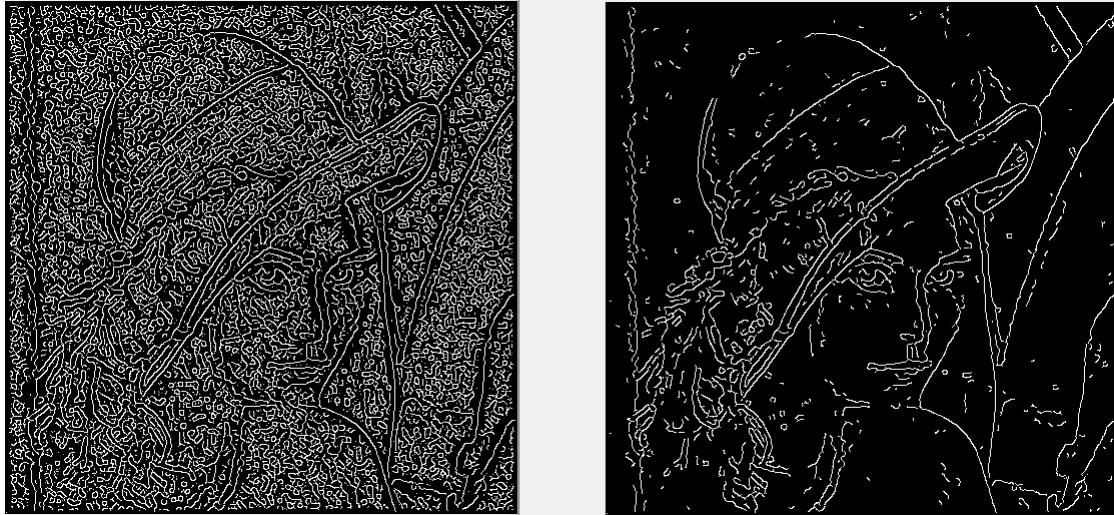


Figura 20: Canny na imagem que recebeu ruido SP de intensidade 0.04 e smoothing com filtro Gaussiano com kernel 3x3 e sigma 2



Figura 21: Canny na imagem que recebeu ruido SP de intensidade 0.04 e smoothing com filtro Gaussiano com kernel 5x5 e sigma 4



Figura 22: Canny na imagem que recebeu ruido SP de intensidade 0.04 e smoothing com filtro Gaussiano com kernel 7x7 e sigma 8

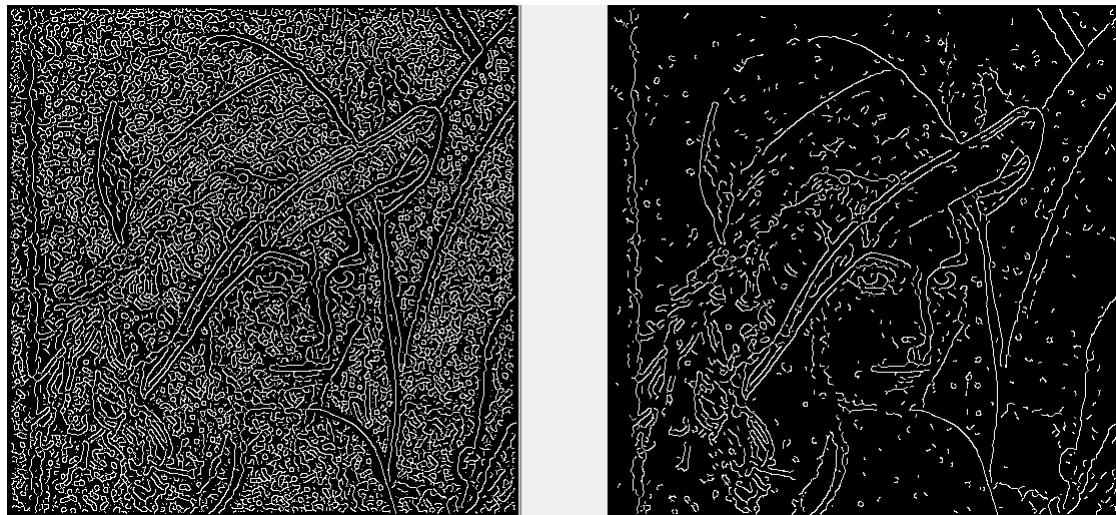


Figura 23: Canny na imagem que recebeu ruido SP de intensidade 0.04 e smoothing com filtro Gaussiano com kernel 3x3 e sigma 2 mas os valores do threshold foram modificados, o mínimo foi aumentado e o máximo diminuído

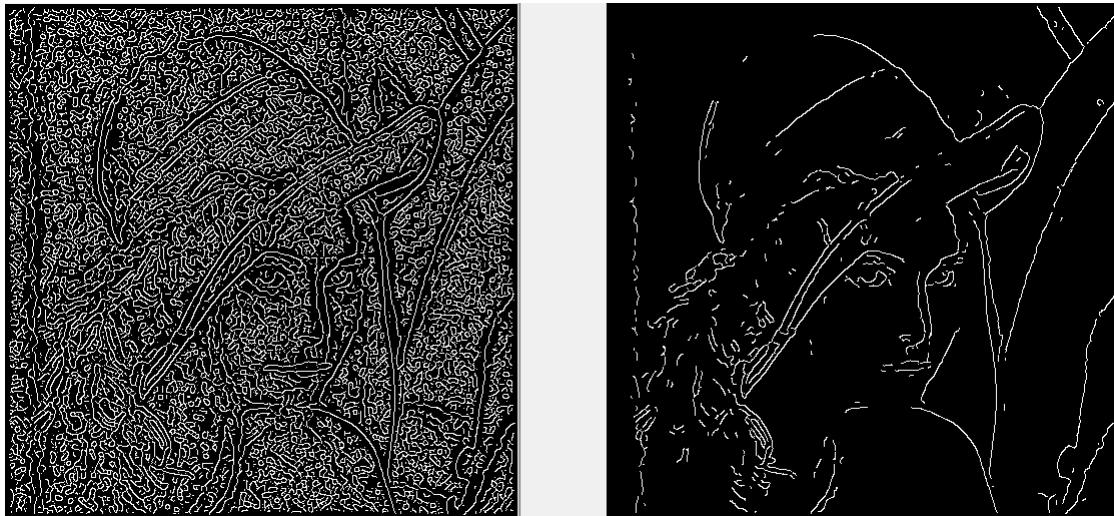


Figura 24: Canny na imagem que recebeu ruido SP de intensidade 0.04 e smoothing com filtro Gaussiano com kernel 3x3 e sigma 2 mas os valores do threshold foram modificados, o mínimo foi diminuido e o máximo aumentado



Figura 25: Canny na imagem que recebeu ruido SP de intensidade 0.04 e smoothing com filtro Gaussiano com kernel 5x5 e sigma 4 mas os valores do threshold foram modificados, o mínimo foi diminuido e o máximo diminuído

Em geral o que se observa é que o efeito do filtro de smoothing que se aplicou antes, neste caso o *Gaussiano* influência bastante no resultado do detetor de contornos, pois elimina ruído e variações bruscas geradas por ruído, o que torna o detetor mais eficiente no final das contas, porém se os valores de sigma e do tamanho do kernel aumentarem demais, o que faz com que o efeito de "blur" aumente muito, perde-se contornos no final da execução do detetor, e pior ainda, contornos que existiam na imagem original .

Referências

1. MATLAB help page.
<https://www.mathworks.com/help>.
2. Edge Detection, Cristina P Santos .
3. Filtering in Image Domain, Cristina P Santos .
4. Suavização de imagens .
<http://www.ic.uff.br/aconci/suavizacao.pdf>.
5. Gaussian Smoothing .
<https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>.