FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Departamento de Engenharia Eletrotécnica e de Computadores

Visão por Computador

2017/2018

Trabalho Prático Nº2

Filtragem e Deteção de Arestas

Eduardo Filipe da Fonseca Rodrigues
2007105369

Parte 1: *Filtragem*

3)

Para este trabalho foi selecionada a imagem “cameraman.tif”, não sendo necessário utilizar a função *rgb2gray*, foram aplicados ruídos gaussiano, com variância de 0.1, 0.5 e 0.8, “salt & Pepper” com densidade 0.5 e “Speckle” com variância de 0.8.

Após aplicação de ruído, cada uma das imagens foi tratada com filtro gaussiano e de mediana.

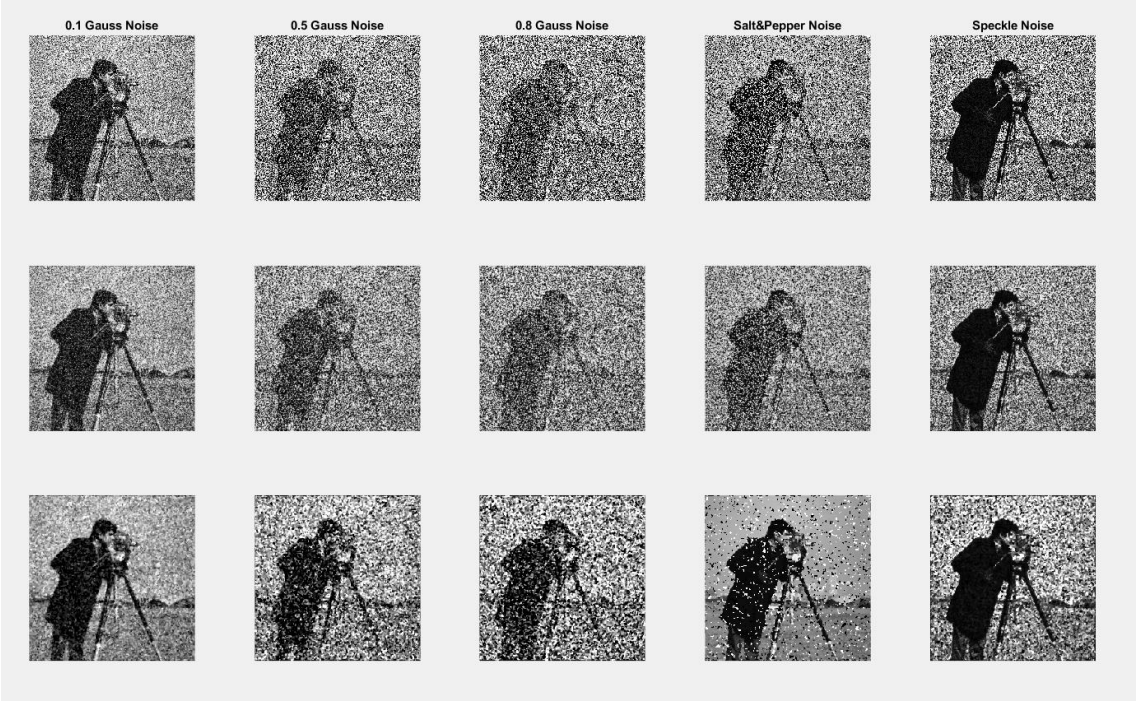

Figura 1 - Filtros e efeitos segundo ruído

a)

Como se pode observar, o filtro gaussiano, linha central na Figura 1, reduz ligeiramente a quantidade de ruído da imagem em todos os casos, tendo um melhor desempenho para ruídos gaussianos de baixa variância e “Speckle”.

No entanto, devido à natureza do filtro, atuando como um filtro passa-baixo, alguns detalhes da imagem são também perdidos.

O filtro de mediana, linha 3 na Figura 1, apresentou, no entanto, resultados mais satisfatórios para todos os casos, tendo quase restaurado à originalidade a imagem com ruído “Salt & Pepper”.

Para comparar o desempenho dos filtros foi então selecionada a imagem com ruído “Salt & Pepper” por apresentar uma maior variação após filtragem.

Observando a Figura 2, é possível comparar o desempenho de ambos os filtros.

O filtro gaussiano está apresentado a vermelho, sendo possível comprovar que não retira o ruído, apenas atenuando a sua intensidade, no entanto, zonas da imagem que sejam de alta frequência mas não tenham ruído são também atenuadas, o que pode provocar perdas de informação.

O filtro de mediana, apresentado a amarelo, reduz quase na totalidade os picos de alta frequência causados pelo ruído.

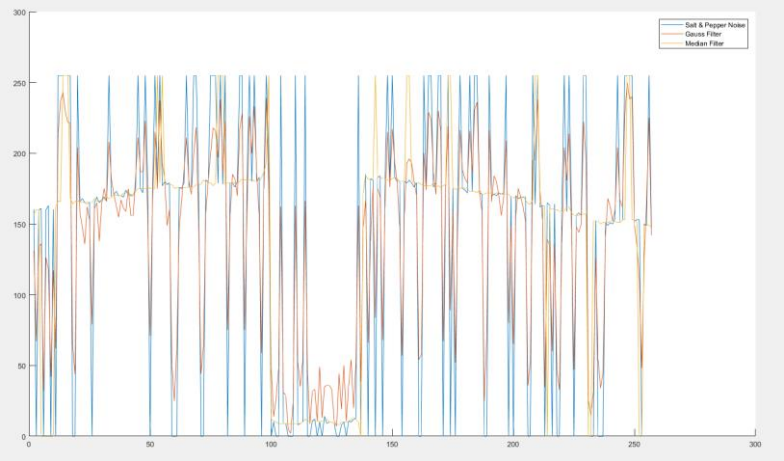

Figura 2 - Linha de perfil da imagem com ruído "Salt & Pepper"

b)

O filtro gaussiano funciona como um filtro de média, no entanto, o peso de cada pixel da vizinhança é dado pela função gaussiana.

Sendo assim, o desvio padrão, alargando a curva da função gaussiana, irá aumentar e peso dado aos pixéis da vizinhança.

Aumentando o desvio padrão o suficiente, todos os pixéis na máscara de convolução terão valores aproximadamente idênticos, aproximando assim o funcionamento do filtro gaussiano ao de um filtro de média.

Com um grande desvio padrão, o filtro elimina uma maior gama de altas frequências, aumentando o risco de perda de informação de alta frequência na imagem, com um desvio padrão demasiado baixo, no entanto, o peso dado aos pixéis da vizinhança pode não ser suficiente para reduzir o ruído.

4)

Foi implementado o filtro de mediana e aplicado a uma das imagens, comparando de seguida com o filtro de mediana fornecido pelo matlab, *medfilt2.*

Como método de comparação, foi analisado o tempo decorrido por ambas as funções e a disparidade entre os pixéis da imagem final.

Foi observada uma diferença de 1.43% entre as imagens, sendo esta diferença causada pelos pixéis da periferia, sendo apenas aplicado o valor “0” nos pixéis pertencentes às linhas e colunas pertencentes à periferia.

Em termos de tempo, a função implementada, teve uma duração média de 0.2 ms, no entanto, a diferença entre ambas foi de 0.197 ms.

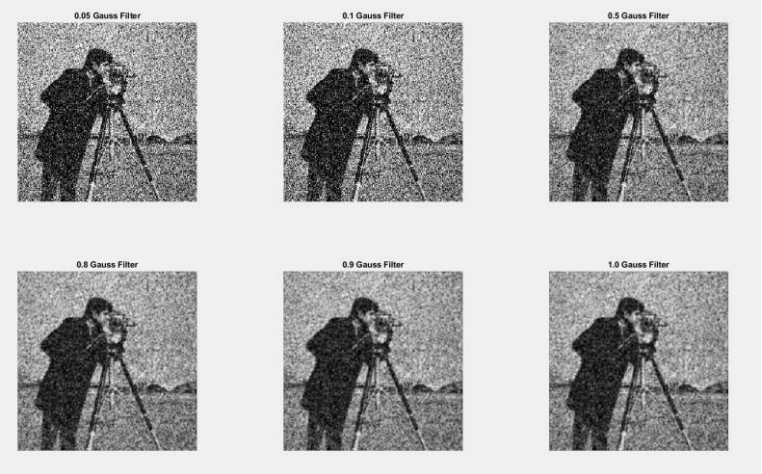

Figura 3 - Influência de desvio padrão no filtro gaussiano

Parte 2: *Deteção de Arestas*

5)

Para a segunda parte do trabalho, foi utilizada a imagem “ImgTest.jpg”.

Para aplicar os filtros *Sobel, Laplaciano e SoG*, foi utilizada a função *fspecial* para gerar as máscaras de convolução, e a função *imfilter* para as aplicar à imagem.

O filtro de Sobel foi aplicado previamente utilizando a função *imgradientxy* de modo a poder ser extraída a informação de módulo e fase do gradiente.

O filtro Canny foi aplicado utilizando a função *edge*.

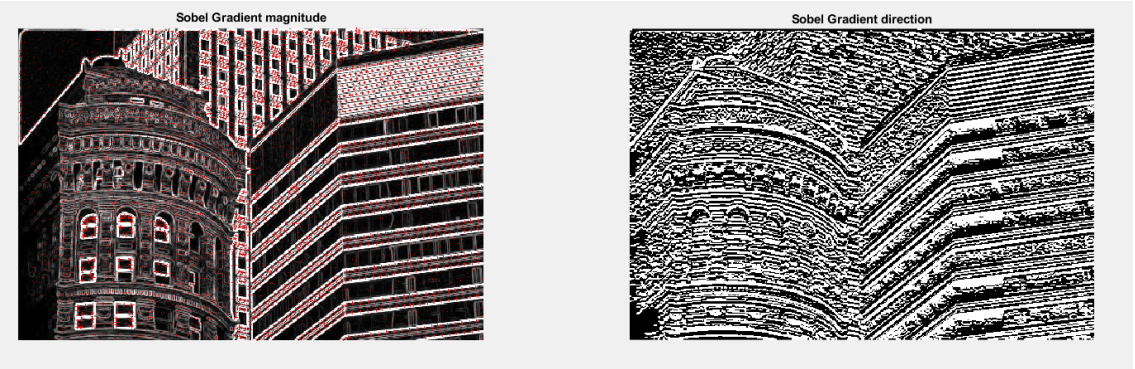

Figura 4 - Módulo e fase de filtro Sobel

Na Figura 4, pode observar-se o módulo e fase do gradiente da imagem calculado através do filtro Sobel. Na primeira metade da Figura 4, pode observar-se, sobreposto ao módulo do gradiente, o próprio gradiente a vermelho, apenas onde o seu módulo é máximo local.

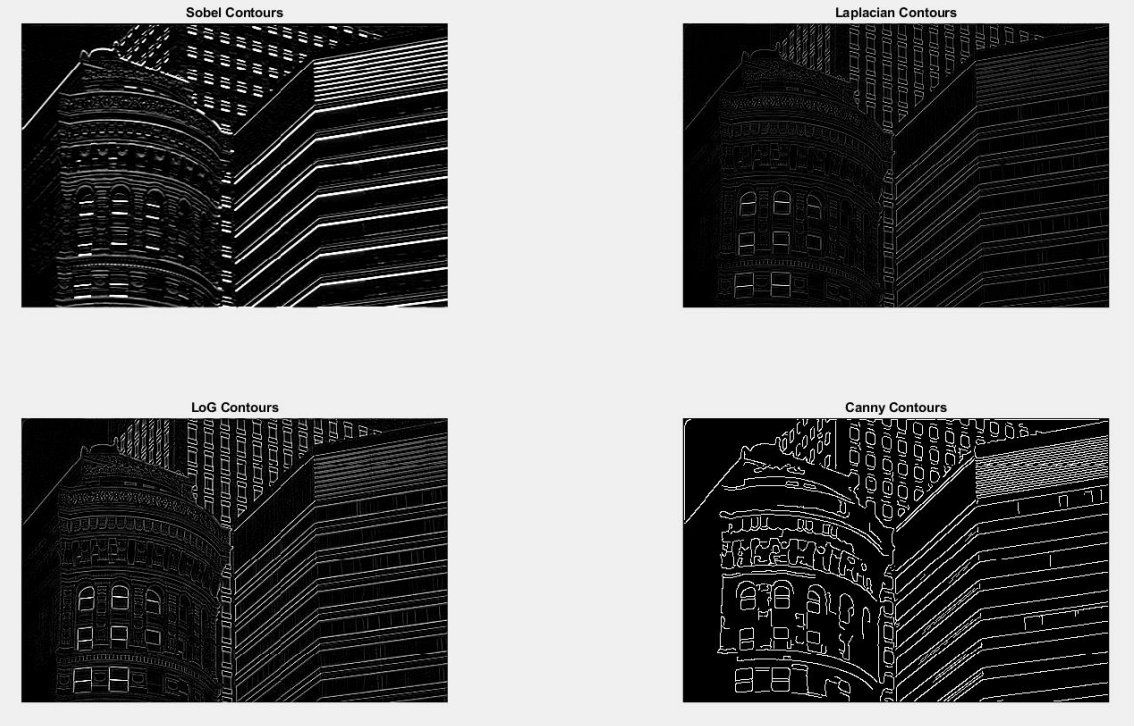

Figura 5 - Contornos obtidos por cada filtro

6)

Para determinar a orientação dos contornos, foi utilizada uma imagem com contornos com orientação pré determinada para facilitar a análise.

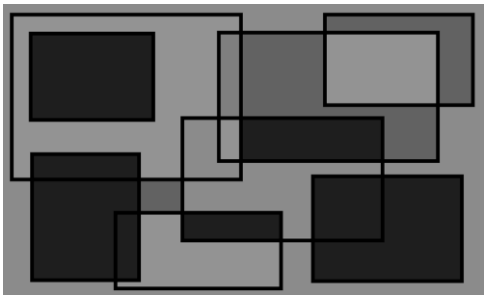

Figura 6 - Imagem para análise de contornos

Aplicando os filtro de *Canny, LoG, Sobel e ZeroCross* através da função *edge* foram obtidas as seguintes imagens.

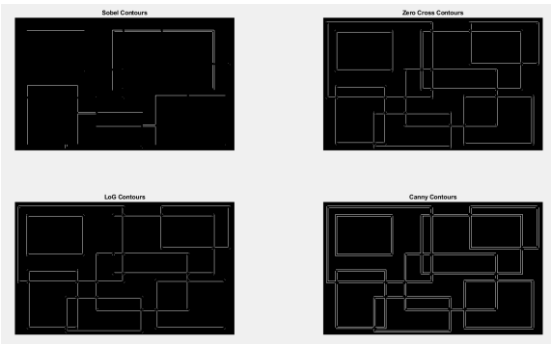

Figura 7 - Contornos

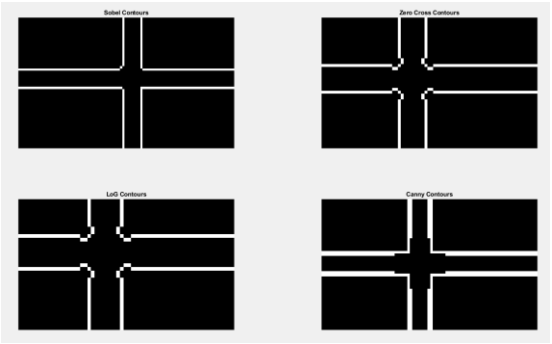

Figura 8 - Contornos ampliados

Observando as imagens da Figura 7, alguns contornos não são visíveis, no entanto, estão presentes. Para observação dos mesmos as imagens foram ampliadas na mesma zona, comprovando assim que todos os contornos foram detetados.

Fazendo um histograma da direção dos contornos na imagem após filtro *Sobel*, obteve-se o seguinte resultado:

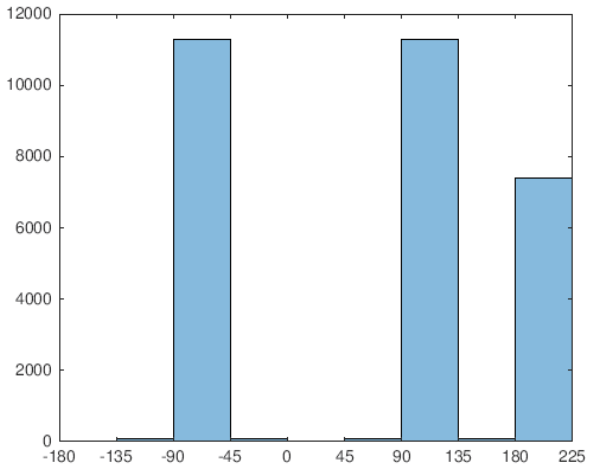

Figura 9 - histograma de orientação de contornos

Tal como esperado, o histograma apresenta orientações de -90º, 90º e 180º, no entanto, apresente também algumas arestas com orientações de -135º, -45º e 45º. Para estudar a causa de tais contornos, foram utilizadas duas imagens, uma apenas com

traços verticais e outra com traços horizontais, onde se calculou de igual modo um histograma para ambas.

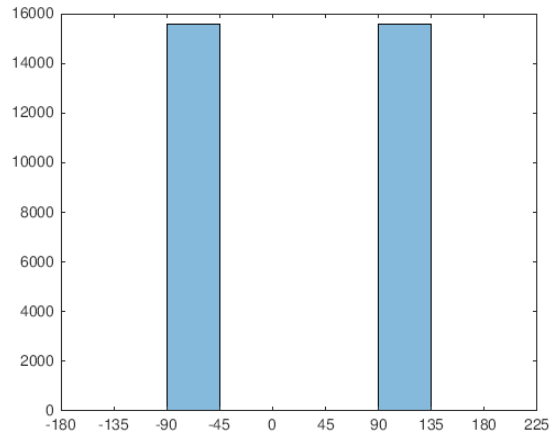

Figura 11 - histograma de contornos verticais

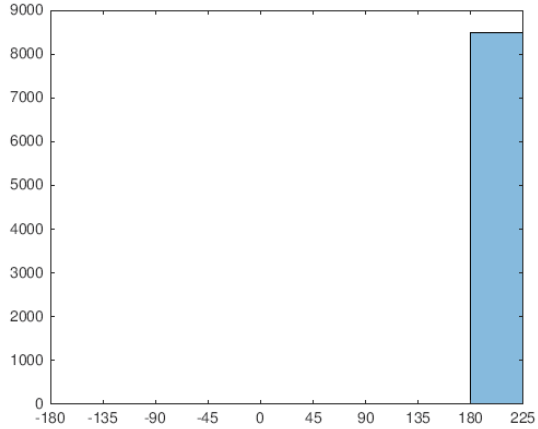

Figura 10 - histograma de contornos horizontais

Como se pode observar, ambos os histogramas apresentam apenas orientações verticais (-90º e 90º) ou horizontais (180º), logo, as orientações calculadas utilizando a imagem “quadrados.png” que não correspondem a estes ângulos terão sido calculadas nos cantos dos quadrados, sendo este o único elemento não comum entre as três imagens.

Os contornos foram então apresentados numa escala de cinza de acordo com o seu ângulo, tendo sido obtida a seguinte imagem:

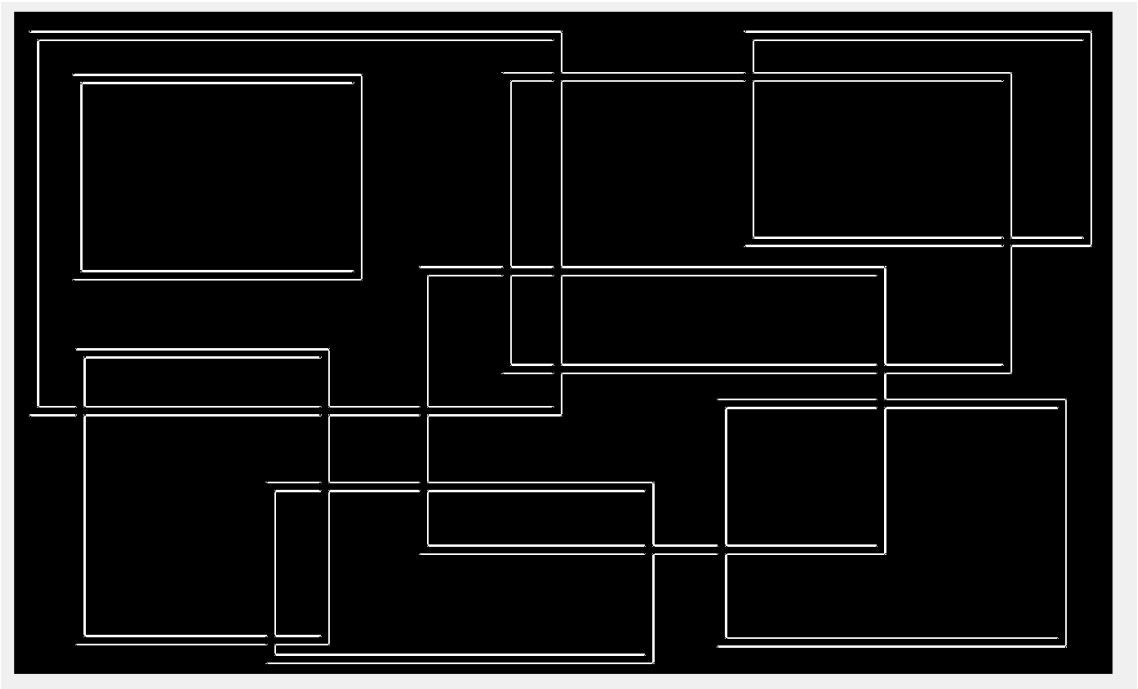

Figura 12 - Contornos em escala de cinza