

VISÃO POR COMPUTADOR

2º Semestre

Ano lectivo 14/15

Trabalho Prático Nº 1

FILTRAGEM e DETECÇÃO DE ARESTAS

FILTROS PASSA-BAIXO (*Filtragem de ruído*) e de DETECÇÃO DE ARESTAS

Em *Visão por Computador*, ruído corresponde às entidades na imagem que não são relevantes para o objectivo do processamento a efectuar (e que degradam o desempenho desse processamento).

A filtragem é uma técnica de processamento de sinal/imagem que modifica e/ou melhora a informação contida na imagem, realçando ou removendo características da imagem. Em geral a filtragem linear é uma operação de vizinhança, na qual um pixel da imagem resultado é obtido processando os pixels na vizinhança do pixel correspondente na imagem origem (normalmente por meio de uma convolução ou correlação).

Diferenciação Numérica

Considere-se a função real $f(x)$ da qual se conhecem N amostras efectuadas em pontos equidistantes x_1, x_2, \dots, x_N , de modo a que $x_i = x_{i-1} + h$ com $h > 0$.

Representando a função $f(x)$ nos pontos $x+h$ e $x-h$ através de uma série de Taylor obtem-se:

$$\begin{aligned}f(x+h) &= f(x) + hf'(x) + 1/2 h^2 f''(x) + O(h^3) \\f(x-h) &= f(x) - hf'(x) + 1/2 h^2 f''(x) + O(h^3)\end{aligned}$$

onde $O(h^3)$ representa o erro de truncatura.

Subtraindo as equações apresentadas e resolvendo em ordem a $f'(x)$ obtem-se a equação para a primeira derivada de $f(x)$:

$$f'(x) = (f(x+h) - f(x-h)) / 2h + O(h^2).$$

De modo análogo, somando as equações obtem-se a equação para a segunda derivada de $f(x)$:

$$f''(x) = (f(x+h) - 2f(x) + f(x-h)) / h^2 + O(h)$$

Primeiras derivadas

$$\begin{aligned}f'_i &= (f_{i+1} - f_{i-1}) / 2h + O(h^2) \\f''_i &= (-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}) / 12h + O(h^4)\end{aligned}$$

Segundas derivadas

$$\begin{aligned}f''_i &= (f_{i+1} - 2f_i + f_{i-1}) / h^2 + O(h) \\f'''_i &= (-f_{i+2} + 16f_{i+1} - 30f_i + 16f_{i-1} - f_{i-2}) / 12h^2 + O(h^3)\end{aligned}$$

As equações apresentadas são chamadas de *equações de diferenciação central* uma vez que calculam as derivadas em x_i com base em amostras de intervalos simétricos centrados em x_i .

As utilização de intervalos assimétricos origina soluções de diferenciação em avanço (forward) ou em atraso (backward), resultando para as primeiras derivadas as seguintes equações:

$$f'_i = (f_{i+1} - f_i) / h + O(h)$$

$$f'_i = (f_i - f_{i-1})/h + O(h)$$

Exemplo de Máscaras de Diferenciação:

Laplaciano (segunda derivada : aproxima a estimação da soma das 2^{as} derivadas)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Sobel (primeira derivada : aproxima o cálculo do gradiente numa imagem)

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

As duas máscaras apresentadas detectam as derivadas horizontais e verticais respectivamente.

Prewitt (primeira derivada : idêntico ao Sobel, com a diferença de que ambas as linhas contribuem com o mesmo peso)

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Roberts (primeira derivada assimétrica)

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Trabalho a realizar :

Parte I:

1. Efectuar a leitura de uma qualquer imagem existente em disco. (Ex: imagem RGB “peppers.png”).
Para converter esta imagem numa imagem em níveis de cinzento utilizar a função *rgb2gray*.
2. Adicionar à imagem diferentes tipos de ruído (separadamente, ou seja não se deve aplicar o ruído sequencialmente à mesma imagem—o ruído é sempre aplicado à imagem original).
 - a) ruído gaussiano com diferentes médias e variâncias.
 - b) ruído ‘salt & pepper’ com diferentes densidades.
 - c) ruído ‘speckle’
Utilizar a função *imnoise* disponível no MATLAB.
3. Efectuar a filtragem de cada uma das imagens (apenas em 3 das imagens obtidas na alínea anterior) utilizando o filtro linear de média, o filtro gaussiano e o filtro não linear de mediana (também aqui os filtros são aplicados separadamente e não sequencialmente—ou seja temos 9 imagens).
Utilize as função *imfilter* disponível no MATLAB.
As máscaras de convolução podem ser obtidas através da função *fspecial*.
A função *medfilt2* realiza a filtragem de mediana.
 - a) Comparar o desempenho dos diversos filtros na presença dos diferentes tipos de ruído. Que diferença existe entre a aplicação do filtro linear de média e do gaussiano?

Utilizar a função *improfile* para visualizar os valores de nível de cinzento ao longo de uma recta na imagem. Comparar os valores antes e depois da filtragem. Juntar os gráficos.

- b) Analisar a influência do desvio padrão na definição da máscara de convolução do filtro gaussiano (testar com vários filtros gaussianos).

Utilizar a função *fspecial* para obter a máscara de filtragem associada a um determinado desvio padrão.

4. Implementar o **filtro de mediana** e comparar o seu funcionamento e desempenho com o filtro de mediana disponível no MATLAB.

FILTRO DE MEDIANA

O filtro de mediana é um filtro de vizinhança onde o valor de um pixel (x,y) é substituído pela *mediana* dos pixels da sua vizinhança.

Algoritmo MED_FILTER

Considere-se **I** a imagem origem, **I_m** a imagem resultando e **n** um número ímpar.

Para cada pixel **I(x,y)** :

1 - Calcular a **mediana** $m(x,y)$ dos valores de cinzento numa vizinhança $n \times n$ de (x,y) , $\{I(x+h,j+k), h,k \in \{-n/2, n/2\}\}$, representando $n/2$ uma divisão inteira.

2 - Atribuir **I_m(x,y)=m(x,y)**.

O mediana de um conjunto de **k** valores corresponde ao valor central (ordem $k/2$) desse conjunto após o ordenamento dos seus valores.

Parte II:

Considere de novo a imagem original (sem as alterações efectuadas na Parte I):

5. Determinar os pixels pertencentes a arestas utilizando as máscaras correspondentes aos operadores de : **Sobel, Prewitt, Laplaciano, LoG (Laplaciano da Gaussiana) e Canny**. Fazer as filtrações separadamente e não sequencialmente: os filtros são aplicados directamente às imagens originais. Fazer o “display” das imagens de módulo e de fase (nos casos em que isso se aplique).

Estas máscaras podem ser obtidas através da função *fspecial* e a sua convolução com a imagem pode ser efectuada pela função *imfilter*.

6. Executar as operações definidas no ponto 5 **depois** de filtrar a imagem original (níveis de cinzento) com um filtro gaussiano. O que conclui?

7. Obtenha os contornos da imagem utilizando as máscaras acima referidas. Use a informação de orientação determinada pelo “edge detector” para obter uma imagem da orientação dos contornos. Faça um histograma com as orientações dos contornos. Quantifique os 360° em 8 intervalos.

O Matlab disponibiliza a função *edge* (com opção *sobel*) para obter os contornos de uma imagem.

8. Implemente o extractor de contornos baseado na utilização da máscara de Sobel e compare o seu desempenho com o apresentado pela função *edge*.

Algoritmo de extracção de contornos baseado no operador SOBEL

1 - Após a prévia filtragem da imagem I por meio de um filtro passa-baixo, convolucionar a imagem filtrada I_f com as máscaras de SOBEL apresentadas (horizontal e vertical), obtendo as imagens I_x e I_y .

2 - Calcular a amplitude do gradiente em cada pixel (i,j) da imagem através de:

$$G(i, j) = \sqrt{I_x^2 + I_y^2}$$

obtendo a imagem de amplitude do gradiente G .

3 – Defina um limiar δ . Marcar os pixels da imagem que constituem os contornos da imagem através da binarização da imagem de acordo com:

$$\begin{aligned} G(i,j) > \delta & \text{ então } O(i,j) = 255; \\ & \text{senão } O(i,j) = 0; \end{aligned}$$