Homework 02

Gustavo Teodoro Laureano

Como esta lista de exercícios deve ser respondida?

Esta lista contém exercícios que exigem respostas discursivas, apresentação de imagens/gráficos e códigos. Portanto, o formato "Relatório" de respostas é mais apropriado. Para cada questão elabore um relatório que apresenta uma discussão incremental do raciocínio, seguido dos resultados e a implementação ao final do texto. Use referências para os algoritmos se necessário. Os algoritmos podem ser implementados usando as seguintes linguagens de programação:

- Octave/Matlab
- Python
- C/C++

1 Questão 1: Detecção de bordas

Escreva uma função que receba uma imagem em escala de cinza e calcule as imagens dos gradientes G_x e G_y, da magnitude $M = \sqrt{G_x^2 + G_y^2}$ e dos ângulos $\Theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$ (em radianos) usando os coeficientes derivativos e proporcionais de Hany Farid e Eero P. Simoncelli d e p.

$$p = \begin{bmatrix} 0.037659 & 0.249153 & 0.426375 & 0.249153 & 0.037659 \end{bmatrix}$$
$$d = \begin{bmatrix} 0.109604 & -0.276691 & 0.00000 & -0.276691 & 0.109604 \end{bmatrix}$$

Crie outra função que recebe as images M e Θ e calcule a imagem de borde E usando a *non-max suppression* para uma janela 3×3. Para isso, assuma que os ângulos sejam discretizados em 0^o, 45^o, 90^o, 135^o e 180^o. Apresente as imagens G_x, G_y, M, O e E para a Figura 1. Para atingir melhores resultados, suavize a imagem original com um filtro gaussiano 3×3.

```
%%
% imgradient
% Input:
% I - input image
% Output:
% Gx - Output image of x gradient
% Gy - Output image of y gradient
% M - Output image of magnitude
% O - Output image of gradient orientation
%%
function [Gx, Gy, M, O] = imgradient(I)
    % ... your code here
end
```

```
%%
% imedge
% Input:
% M - Magnitude image
% O - Gradient orientation image
% Output:
% E - Edge image
%%
function [E] = imedge(M, O)
    % ...  your code here
end
```

Figura 1: Imagem de teste para a questão 1.

2 Questão 2: Pirâmide Gaussiana

Uma pirâmide Gaussiana é um conjunto de cópias de uma mesma imagem em escalas diferentes. Sendo G_l a imagem do nível l e W o *kernel* de suavização, pode-se definir o processo de redução de escala ($REDUCE$) de acordo com a Equação 1.

$$REDUCE\{G_l\} = G_{l+1} \Rightarrow G_{l+1}(i,j) = \sum_{n=-2}^{2}\sum_{m=-2}^{2} W(m,n)G_l(2i+m, 2j+n) \tag{1}$$

Sendo $W(m,n) = w(m)w(n)$ e $w = \begin{bmatrix}\frac{1}{4} - \frac{a}{2} & \frac{1}{4} & a & \frac{1}{4} & \frac{1}{4} - \frac{a}{2}\end{bmatrix}$, onde o valor de $a = 0.375$ apresenta melhor ajuste ao *kernel* gaussiano de tamanho 5.

Uma pirâmide Gaussiana pode ser construída usando qualquer *kernel* Gaussiano, no entanto, Burt e Adelson em 1983, com o trabalho *The Laplacian Pyramid as a Compact Image Code*, concluiram que a estrutura do *kernel* w apresenta menor perda no processo de *downsample*.

Escreva as funções `imreduce()` que calcula a imagem do próximo nível da pirâmide e `impyramid()` que cria a piramide de imagens. As entradas devem ser uma imagem em escala de cinza, um filtro formado por um *kernel* separável (p e d) e a quantidade de níveis (L) da pirâmide. Como técnica de *downsampling*, descarte as linhas e colunas ímpares.

Construa o duas pirâmides. Para a primeira use o *kernel* $p = d = w$ dado acima e a para a segunda use o filtro da média dado por $p = d = \frac{1}{7}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$. Compare e comente o resultado para uma pirâmide com 5 níveis para a Figura 2.

```
%%
% imreduce
% Input:
% G0 - input image
% p - filter
% d - filter
% Output:
% G1 - upper level pyramid image
%%
function [G1] = imreduce(G0, p, d)
    % ... your code here
end
```

```
%%
% impyramid
% Input:
% I - input image
% p - filter
% d - filter
% L - number of levels
% Output:
% P - Pyramid
%%
function [P] = impyramid(I, p, d, L)
    % ... your code here
end
```

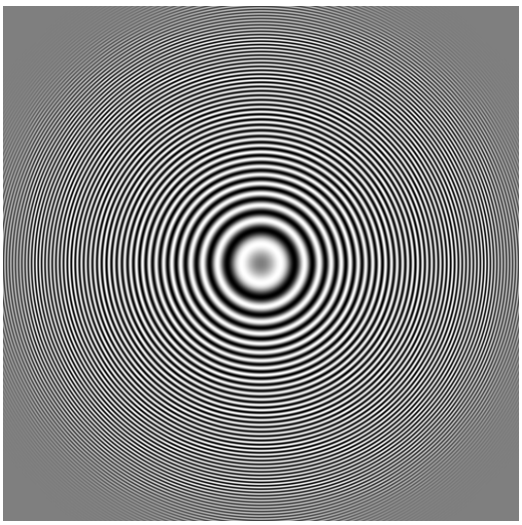

Figura 2: Imagem de teste para a questão 2.

3 Questão 3: Difference of Gaussian (DoG) Pyramids

A pirâmide de imagem DoG é uma aproximação da pirâmide *Laplacian of Gaussian* (LoG), obtida pela diferença entre os níveis da Pirâmide Gaussiana. Nesse tipo de transformação a imagem pode ser reconstruída fielmente usando os coeficientes de cana nível. Crie funções que permitam construir a pirâmide DoG e a reconstrução da imagem original a partir de uma pirâmide.

Um nível da pirâmide DoG é dado pela Equação 2.

$$L_l = G_l - EXPAND\{G_{l+1}\} \tag{2}$$

Escreva a função `imLoG()` que receba uma imagem I em escala de cinza e a quantidade de níveis L e que retorna a pirâmide DoG. Lembre-se que o último nível da pirâmide DoG é formado pela imagem em menor escala. O processo de expansão da escala ($EXPAND$) é dado pela Equação 3.

$$EXPAND\{G_l\} = G_{l-1} \Rightarrow G_{l-1}(i,j) = \sum_{n=-2}^{2}\sum_{m=-2}^{2} W(m,n)G_l\left(\frac{i+m}{2}, \frac{j+n}{2}\right) \tag{3}$$

, onde $W(m,n) = w(m)w(n)$ e w é dado na Questão 2. Os valores não inteiros gerados pelos termos $\frac{i+m}{2}$ e $\frac{j+n}{2}$ são desconsiderados.

```
%%
% imexpand
% Input:
% G1 - upper level gaussian pyramid image
% D0 - Difference DoG pyramid image
% Output:
% G0 - down level gaussian image
%%
function [G0] = imexpand(G1, D0)
    % ... your code here
end
```

```
%%
% imDoG
% Input:
% I - input image
% p - filter
% d - filter
% L - number of levels
% Output:
% DoG - DoG Pyramid
%%
function [DoG] = imDoG(I, p, d, L)
    % ... your code here
end
```

Apresente a pirâmide DoG e a imagem reconstruída para a Figura 3.

Figura 3: Imagem de teste para a questão 3.