Universidade Tecnológica Federal do Paraná - UTFPR
Campus Curitiba
Departamento Acadêmico de Eletrônica - DAELN
Curso de Engenharia Eletrônica
Prof. Gustavo B. Borba

Atividade #07
Vale nota, individual, observar prazo e instruções de entrega no moodle

Arquivos necessários [disponíveis no zip]
1. cameraman.tif [https://homepages.cae.wisc.edu/~ece533/images/]
2. makeImSynthHex.m

7.1) Nada a ver com bordas mas é interessante

Explique como foi gerado o círculo na imagem sintética obtida com a função *makeImSynthHex* (Notas de aula e prática Octave #07). Qual a maneira mais fácil de alterar o diâmetro do círculo? Responda no próprio .m, na forma de comentários.

Nome do .m: atv07_01.m

7.2) Detecção de bordas usando filtros de gradiente (Sobel na unha)

Nos códigos Octave mostrados nas Notas de aula e prática Octave #07, a convolução de uma máscara Sobel com a imagem de entrada é realizada assim:

```
gS = imfilter(g,S,'replicate','conv');
```

O que deve acontecer se retirarmos a opção `'conv'`? Justifique a sua resposta mostrando resultados e comparando-os com os resultados obtidos com a opção `'conv'`. Sabemos que o resultado final do detector de bordas Sobel é a imagem *magnitude do gradiente* (ou esta magnitude do gradiente *limiarizada*). A opção pela convolução ou pela correlação tem impacto neste resultado final? Pelo tipo da pergunta você já deve ter entendido que a resposta é não. Por que? Responda no próprio .m, na forma de comentários.

Nome do .m: atv07_02.m

7.3) Detecção de bordas usando filtros de gradiente (Sobel na unha)

Vimos que o cálculo da magnitude do gradiente, *M(i,j)*, é feito utilizando-se o operador *módulo de um vetor* (também chamado de *norma*). No entanto, para evitar o cálculo de uma raiz quadrada, economizando assim recursos computacionais, pode-se fazer uma simples soma dos valores absolutos (sem sinal) dos gradientes G_v e G_h. Teste esta afirmação comparando os resultados dos dois métodos.

Nome do .m: atv07_03.m

7.4) Detecção de bordas usando filtros de gradiente (Sobel na unha)

Sugira e teste outro método (diferente do módulo do vetor e diferente da soma) para obter a imgem magnitude do gradiente. Dica: ver tópico 7c em Notas de aula e prática Octave #07.

Nome do .m: atv07_04.m

7.5) Detecção de bordas usando o Laplaciano (função edge)

O objetivo deste exercício é testar a afirmação "a derivada de segunda ordem não costuma ser utilizada diretamente para a detecção de bordas porque é muito sensível ao ruído". Para isso, você pode comparar a saída o detector de bordas Laplaciano (derivada segunda) com o Sobel (derivada primeira). Como imagem de entrada use a *cameraman.tif* ou outra, se preferir. Lembrando: nos métodos que utilizam a derivada segunda, as bordas são obtidas detectando-se a passagem por zero.

O Laplaciano para a detecção de bordas pode ser implementado no Octave utilizando-se a função `edge` (para fazer a detecção de passagem por zero) da seguinte forma:

```
h = fspecial('laplacian',0);
lap = edge(img,'zerocross',t,h); %t é um limiar que atua na sensibilidade da
                                 %detecção das passagens por zero
```

Para o Sobel, vc pode utilizar a opção `edge` com os parâmetros default:

```
sob = edge(img,'sobel');
```

Para a comparação ser justa, tente ajustar o valor `t` da detecção das passagens por zero do Laplaciano, de modo que os falsos positivos na saída sejam tão poucos quanto os observados no Sobel. Qual é a conclusão? Responda no próprio .m, na forma de comentários.

Nome do .m: atv07_05.m

7.6) Detecção de bordas usando o LoG (função edge)

Já que Laplaciano puro é muito sensível ao ruído: "o operador baseado na derivada segunda utilizado para a detecção de bordas é o *Laplaciano do Gaussiano* (LoG), que incorpora no operador Laplaciano (derivada segunda) uma suavização através de uma função Gaussiana." Use a função `edge` para testar o LoG com diferentes valores de σ. Como imagem de entrada use a *cameraman.tif* ou outra, se preferir. O valor do limiar que atua na sensibilidade da detecção das passagens por zero deve ser mantido fixo. Para isso, entenda a sintaxe da função `edge` para o LoG, mostrada a seguir. Como o valor de σ influencia a saída do detector de bordas? Responda no próprio .m, na forma de comentários.

Nome do .m: atv07_06.m

Detecção de bordas com a função `edge`, método LoG
[https://octave.sourceforge.io/image/function/edge.html]

`[bw, thresh] = edge (im, "LoG", thresh, sigma)`

Find edges by convolving with the Laplacian of Gaussian (LoG) filter, and finding zero crossings.
Only zero crossings where the filter response is larger than thresh are retained. thresh is automatically computed as 0.75*M, where M is the mean of absolute value of LoG filter response.
sigma sets the spread of the LoG filter. Default to 2.

7.7) Detecção de bordas usando o método Kirsch

Implemente e teste o Kirsch em diferentes imagens, na unha. É simples: depois de definir todas as oito máscaras, convoluir cada uma com a imagem é moleza, é pra isso que existe a função `imfilter`. A dica é armazenar cada imagem convoluída em uma página diferente (dimensão 3) da mesma matriz. Por exemplo: `kirschT(:,:,1)` recebe o resultado da convolução da imagem de entrada com a máscara K_0; `kirschT(:,:,2)` recebe o resultado da convolução da imagem de entrada com a máscara K_1, e assim por diante. Com isso, será possível encontrar o máximo das saídas das oito máscaras numa paulada só, usando a seguinte sintaxe: `kirsch = max(kirschT,[],3);`

Nome do .m: atv07_07.m

7.8) Detecção de bordas usando o método Canny (função edge)

Compare as saída do Canny e do Sobel para uma ou mais imagens à sua escolha (com o cameraman é possível). Para o Canny, use os parâmetros default da função `edge`. Para a comparação ser (mais ou menos) justa, atue no valor do limiar que ajusta a sensibilidade do Sobel (sintaxe `BW = edge(I,'sobel',thresh)`), com o objetivo de diminuir a quantidade de falsos positivos do Sobel, se comparado com o Canny. Aponte falsos positivos e falsos negativos na saída do Sobel, que não ocorrem na saída do Canny.

Nome do .m: atv07_08.m