Computação Gráfica - Trabalho 01

Engenharia da Computação IFCE 2018.2
Professor: Lucas Sousa.
Aluna: Gabriela Bezerra.
Projeto no Github: https://github.com/GabrielaBezerra/ComputacaoGraficaTrabalho01

Objetivo

Ler uma imagem .PGM, Implementar e aplicar os seguintes algoritmos:

- Filtro da Media
- Filtro da Mediana
- Ajuste de Contraste
- Limiarização

Tecnologias utilizadas

OpenCV 4.0.0, C++

Imagem utilizada para testes

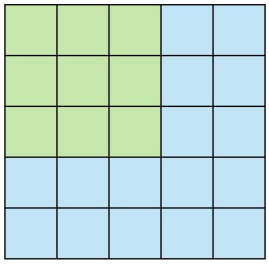
lena.pgm

Leitura da imagem

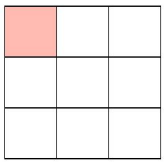
Para utilizar a imagem e processá-la de acordos com os algoritmos, foi utilizada a função *imread* do opencv que, em C++, retorna uma estrutura *Mat* contendo os valores de pixel divididos em *rows* e *cols*, a quantidade de canais da imagem, o tamanho, e tipo. Após esse processo, foi adicionada uma “borda preta” ao redor da imagem, proporcional ao tamanho da máscara aplicada sobre ela. Por exemplo, no caso de uma máscara 7x7:

1. Filtro da Media

O algoritmo consiste em varrer a matriz de pixels MxM de uma imagem sobrepondo-a a outra matriz menor NxN (o kernel, ou máscara), onde N é um valor ímpar, e multiplicar cada elemento da matriz maior pelo elemento correspondente na máscara (no caso deste trabalho, todos os elementos da máscara tem valor 1). Ao final de cada loop de multiplicação da máscara, é preciso calcular a média desses resultados e atribuir esse valor médio à uma nova matriz de mesmo tamanho da matriz maior, na posição correspondente. Para tentar ilustrar esse processo, a imagem ao lado mostra a máscara em verde, a matriz da imagem processada em azul, e a multiplicação entre os valores em rosa. Esse processo é feito de forma sequencial, como uma varredura, sempre posicionando o elemento central da máscara no elemento de cada iteração da varredura da imagem original.



Stride 1



Feature Map

Resultados esperados

O esperado é que o filtro da média gere uma nova imagem, de mesmo tamanho da imagem original, e que essa imagem esteja levemente borrada. Quanto maior o tamanho da mascara em relação ao tamanho da imagem, mais borrada a imagem ficará. Na verdade o que acontece é uma suavização e redução de ruídos do tipo sal/pimenta (pontos pretos e brancos sobre a imagem).

Resultados obtidos

Máscara 3x3. Reudção de ruídos ideal/eficaz.

Máscara 11x11

Máscara 7x7

2. Filtro da Mediana

O algoritmo segue o mesmo processo do filtro da média, porém em vez de calcular a média dos resultados obtidos pela multiplicação de uma parte da imagem original pelos valores da máscara, calcula-se a mediana, e essa mediana será atribuída a nova imagem, na sua respectiva posição.

Resultados esperados

O esperado é que o filtro da mediana gere uma nova imagem, de mesmo tamanho da imagem original, e que essa imagem esteja levemente borrada, assim como o filtro da média, porém preservando mais as bordas, e borrando as cores, ou tons de cores de cada região de pixels do tamanho da máscara. Quanto maior o tamanho da máscara em relação ao tamanho da imagem, mais borrada a imagem ficará. Na verdade o que acontece é uma suavização e redução de ruídos do tipo sal/pimenta (pontos pretos e brancos sobre a imagem).

Resultados obtidos

Máscara 3x3. Reudção de ruídos ideal/eficaz.

Mascara 7x7.

Máscara 7x7.

Máscara 11x11.

3. Ajuste de Contraste

O algoritmo consiste em deformar a linearidade dos valores de pixel de uma imagem, desbalanceando a diferença entre os valores de pixels existentes (tanto aumentando a diferença entre eles, quanto diminuindo). Pode ser feito de manualmente aplicando a fórmula:

$$g(x,y) = a * f(x,y) + b$$

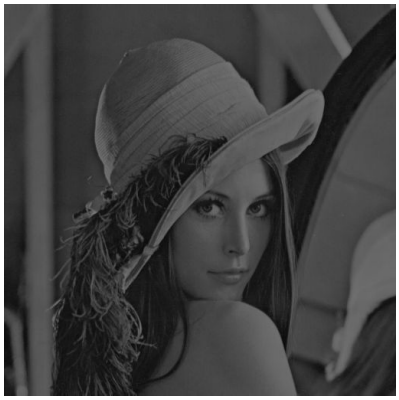
Onde *a* é o contraste, o balaceamento dos valores de cada pixel, e *b* é uma constante que representa o ajuste de brilho, podendo ser usada para corrigir um contraste extremo.

Fonte: https://docs.opencv.org/3.4.1/d3/dc1/tutorial_basic_linear_transform.html

Resultados esperados

A imagem gerada varia bastante dependendo dos parâmetros definidos, e da necessidade de quem aplica o ajuste de contraste. Neste trabalho, o valor de b (brilho) é sempre 0. O valor de equilíbrio do contraste é 1. Quanto mais o contraste variar para menos que 1, mais escura a imagem será, e menor será a diferença entre os tons de cinza. Quanto mais o contraste variar para mais de 1, mais clara a imagem será, e menor será a diferença entre os tons de cinza. Para destacar bordar ou relevo da imagem, o valor de contraste deve ser balanceado com o valor de brilho.

Resultados obtidos



Contraste: 0.5

Contraste: 2.5

Contraste: 1.5

4. Limiarização

O algoritmo consiste em “minimizar” uma imagem a partir de um determinado valor de pixel, que chamaremos de *limiar*. A imagem é varrida, e todo valor que for maior que o valor de limiar será tipo como valor máximo (branco), e todo valor que for menor será tido como valor mínimo (preto). A limiarização pode ser binária (só tem um limiar, e gera uma imagem com somente dois tons de cor), ou ter mais limiares, e mais tons de cores/valores de pixel. Para este trabalho, foi implementada a limiarização binária. O valor dos pixels varia entre 0 e 255.

Resultados esperados

A imagem gerada deverá conter somente pixels pretos e brancos. Quanto maior o valor limiar, mais pontos pretos a imagem terá e os pixels mais claros terão destaque. Quanto menor o valor do limiar, mais pontos brancos a imagem terá, destacando os pixels mais escuros. Apesar da simplicidade do algoritmo, existem aplicações muito interessantes, como a segmentação de imagens, e a vetorização de algumas imagens raster (funciona bem para vetorização de símbolos, letras, icones, cartoons, ou qualquer imagem *flat*).

Resultados obtidos

Limiar: 50.

Limiar: 180.

Limiar: 128. Valor médio.