ATIVIDADE DE LABORATÓRIO Nº01

Ciência da Computação – Instituto de Ensino Superior de Brasília (IESB)
Brasília – DF – Brasil

INTRODUÇÃO

A Computação Gráfica é a área da ciência da computação que estuda a geração, manipulação e interpretação de imagens. No geral, são métodos e técnicas aplicadas para transformar dados em imagem através de um dispositivo gráfico. Processamento de Imagens é a subárea da computação gráfica que utiliza técnicas para manipulação de imagens como: cor, brilho, contrastes, aplicações de filtros etc.

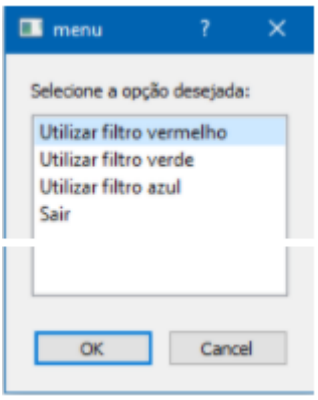
O seguinte relatório tem como objetivo detalhar a execução de dois algoritmos feito na linguagem GNU Octave, que é uma linguagem computacional desenvolvida para computação matemática. O primeiro algoritmo se refere a uma simples filtragem de cores e segundo em uma simples modificação de contraste.

PARTE 1 – ALGORITIMO SIMPLES DE FILTRO DE CORES

Primeiramente o algoritmo receberá uma imagem em cores representada no espaço RGB como entrada, à escolha do usuário.

```
1  # Receber nome da imagem
2  nome = input ("Informe o nome da imagem com extensão. Exemplo: imagem.png: ", "s");
3
4  # Leitura da imagem
5  img = imread(nome)
6
```

Por ser somente uma matriz, esta imagem será vista em escala de cinza. Após a leitura da imagem o programa mostrará um menu, para que o usuário escolha uma das três versões de filtros, sendo elas vermelho, verde ou azul:



Caso usuário selecione a primeira opção “Utilizar filtro vermelho”, para cada pixel da nova imagem, será atribuído o valor 1 (pixel branco), se a componente R (vermelha) for a maior entre as três (isto é, $R > G$ e $R > B$); ou será atribuído o valor 0 (pixel preto), se a componente vermelha não for a maior entre as três (isto é, $R \leq G$ ou $R \leq B$).

```
switch(type)
case 1
for i = 1:rows(img)
for j = 1:columns(img)
  string = int2str(img(i, j))

  if(length(string) == 3)
  if(string(1) > string(2) && string(1) > string(3))
  newimg(i, j) = 1;
else
  newimg(i,j) = 0;
endif

else
newimg(i,j) = 0;
endif
endfor
endfor
```

Caso o usuário escolha o filtro o verde ou o filtro azul, o procedimento análogo deve ser feito, tendo a componente verde (G) ou azul (B), respectivamente, como referência para comparação com as outras duas.

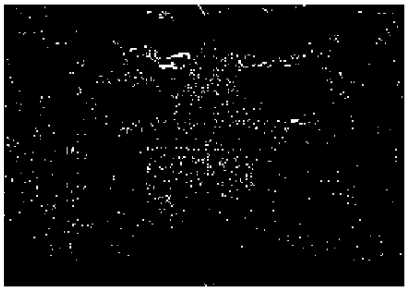
```
case 2
for i = 1:rows(img)
for j = 1:columns(img)
  string = int2str(img(i, j))

  if(length(string) == 3)
  if(string(2) > string(1) && string(2) > string(3))
  newimg(i, j) = 1;
else
  newimg(i,j) = 0;
endif

else
newimg(i,j) = 0;
endif
endfor
endfor
case 3
for i = 1:rows(img)
for j = 1:columns(img)
  string = int2str(img(i, j))

  if(length(string) == 3)
  if(string(3) > string(1) && string(3) > string(2))
  newimg(i, j) = 1;
else
  newimg(i,j) = 0;
endif

else
newimg(i,j) = 0;
endif
endfor
endfor
endswitch
```

RESULTADO – PARTE 01

Imagem de entrada:

Saída para o Filtro Verde:

Saída para o Filtro Vermelho:



Saída para o Filtro Azul:

As imagens resultantes além de serem representada em escala de cinzas, também serão binárias, ou seja, seus pixels serão necessariamente pretos ou brancos.

PARTE 2 – ALGORITIMO SIMPLES DE MODIFICAÇÃO DE CONTRASTE

A primeira parte do algoritmo solicita do usuário o nome da imagem preto e branco que ele deseja modificar. Logo após é feita a leitura da imagem:

```
1  # Receber nome da imagem
2  nome = input ("Informe o nome da imagem com extensão. Exemplo: imagem.png: ", "s");
3
4  # Leitura da imagem
5  img = imread(nome);
6
```

Após a leitura, é feita a média dos pixels da imagem:

```
7  # Soma de todas as posições da matriz
8  somal = sum(img);
9  soma = sum(somal);
10
11 # Pega o tamanho das linhas e colunas da matriz
12 linhas = rows(img);
13 colunas = columns(img);
14
15 # Calcula a quantidade de itens dentro da matriz
16 items = linhas * colunas;
17
18 # Calcula a média dos pixels
19 average = soma / items;
```

Com a média, é possível determinar se a imagem está clara ou escura. É utilizado o seguinte método para determinar: se a média for maior que 128, a imagem está clara e se for menor, ela está escura:

```
21 # Informa o gamma através da verificação da imagem entre claro e escuro
22 if(average > 128)
23   gamma = 0.9;
24 else
25   gamma = 1.05;
26 endif
```

O próximo passo é corrigir a imagem de acordo com o gamma definido. A correção gamma ajusta o contraste da imagem, tornando-a mais clara ou mais escura. O valor de gamma sempre será positivo. Caso o valor seja entre 0 e 1, a imagem será escurecida e se o valor for acima de 1, ela será clareada. Após testes com os valores, foram definidos os valores 0.9 e 1.05 para escurecer e clarear, respectivamente. Para fazer a correção, a matriz é potencializada ao valor gamma definido e as duas imagens são mostradas para averiguar a correção:

```
21  # Define gamma através da verificação da imagem entre claro e escuro
22  if(average > 128)
23    gamma = 0.9;
24  else
25    gamma = 1.05;
26  endif
27
28  # Altera a imagem de acordo com o gamma
29  alter = abs(img.^gamma)
30
31  # Mostra a imagem original e a alterada para comparação
32  subplot(221); imshow(img); title('Original');
33  subplot(222); imshow(alter); title('Alterada');
```

RESULTADO – PARTE 02

No primeiro caso, a imagem original foi determinada como escura. Sendo assim, a imagem foi clareada através da correção gamma. Já no segundo, a imagem foi definida como clara e obteve a correção gamma para escurecer:

CONCLUSÃO

Toda a computação gráfica é baseada em pixels que são pontos que fazem com que a imagem seja sintetizada visualmente em um monitor. Seja em 3D por modelagem tridimensional ou 2D, o profissional em computação gráfica trabalha direta ou indiretamente com pixels e suas compressões. Com esse projeto tivemos a oportunidade de pôr em prática a teoria estudada.

ALUNOS:
Geovana Cordeiro de Oliveira - 1722130073
Jorge L. C. O. Júnior - 1812130118
Milena N.M. Brito - 1722130027

Anexos

Link do GitHub: https://github.com/jjorge98/ComputacaoGrafica