Processamento e Analise de Imagens (MC940)
Analise de Imagens (MO445)
Universidade Estadual de Campinas
Instituto de Computação
Professor: Helio Pedrini
Aluno: Renan Gomes Pereira 103927

Trabalho 1

Introdução

Este relatório tem o objetivo de apresentar a realização alguns processamentos básicos em imagens digitais: combinação de imagens; planos de bits; comparação entre imagens; filtragem; e mosaicos.

Procedimentos

Foi criado um programa utilizando o software MATLAB R2014a para o sistema operacional Windows 8.1 64 bits. Este programa lê um conjunto de imagens em um diretório (input), realiza os procedimentos desejados nas imagens selecionadas e devolve as imagens resultantes em outro diretório denominado output.

Para rodar o programa: extrair o .zip fornecido e **modificar as variáveis nas linhas 13 e 14** com o path da em que as pastas de input e output (subpastas do .zip) estão na máquina em que de se deseja rodar o programa.

1 - Combinação de Imagens

Neste procedimento, o programa lê duas imagens do diretório input e combina duas imagens de mesmo tamanho por meio da média dos seus tons de cinza.

Primeiramente converte-se a imagem de RGB para escalas de cinza e em seguida chama uma subfunção do programa que realiza a seguinte operação:

```
RESULT = 0.5*IMG1 + 0.5*IMG2;
```

Inicialmente aplicou-se a operação acima para as duas imagens presentes no enunciado do trabalho1 [1] para verificar se a imagem resultante estava coerente com a presente em [1] e o resultado foi **positivo**.

As *imagem 1* abaixo mostra o resultado obtidos com a aplicação da operação descrita acima:

Imagem 1 (a): Imagem Original 1

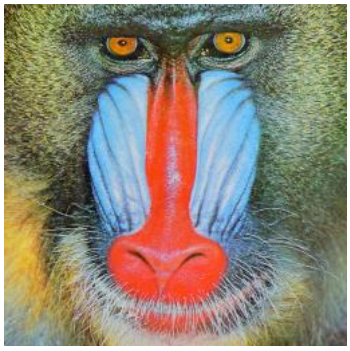

Imagem 1 (b): Imagem Original 2

*Imagem 1 (c): Imagem obtida com o uso da operação 0.5***IMG1** + 0.5***IMG2** para combinar as imagens.*

Por fim, ainda nesta parte de combinação de imagens, foi feito um teste com duas imagens de formas diferentes, nas quais uma delas era retangular e a outra era quadrada. Para essa operação ser possível, primeiro foi feita a conversão da dimensão das duas imagens para 256x256.

A converter a imagem retangular para 256x256, obteve-se uma imagem distorcida porque a conversão de não respeitou o aspect ratio dessa imagem.

A *imagem 2* mostra as imagens originais e o resultado obtido:

Imagem 2 (a): Imagem inicial de dimensões 768x512

Imagem 2 (b): Imagem inicial de dimensões 512x512

Imagem 2 (c): Imagem final obtida pela combinação

O último teste realizado nesta parte foi tentar somar duas imagens de dimensões diferentes e o MATLAB apontou o seguinte erro:

```
Error using  + Matrix dimensions must agree.
```

Logo, converter as imagens para a mesma dimensão se mostra necessário.

2 – Planos de Bits

Um plano de bits de uma imagem é um conjunto de bits correspondendo a uma posição dos números binários representando essa imagem [2]. Eles são muito usados em diversas aplicações decompondo a imagem para tornar a análise mais rápida [3]. No nosso caso, cada

pixel de uma imagem em níveis de cinza é representado por 8 bits (0 a 255), portanto temos 8 planos de bits. O plano de bits número 1 contém os bits menos significativos de todos os pixels da imagem, por outro lado, o plano de bits número 8 contém os bits mais significativos de todos os pixels da imagem.

Nesta parte do trabalho, o programa converte uma imagem colorida para uma imagem em níveis de cinza, extrai os 8 planos de bits e mostra cada um na tela.

A *imagem 3* abaixo mostra os resultados obtidos:

Imagem 3 (a): Imagem Original

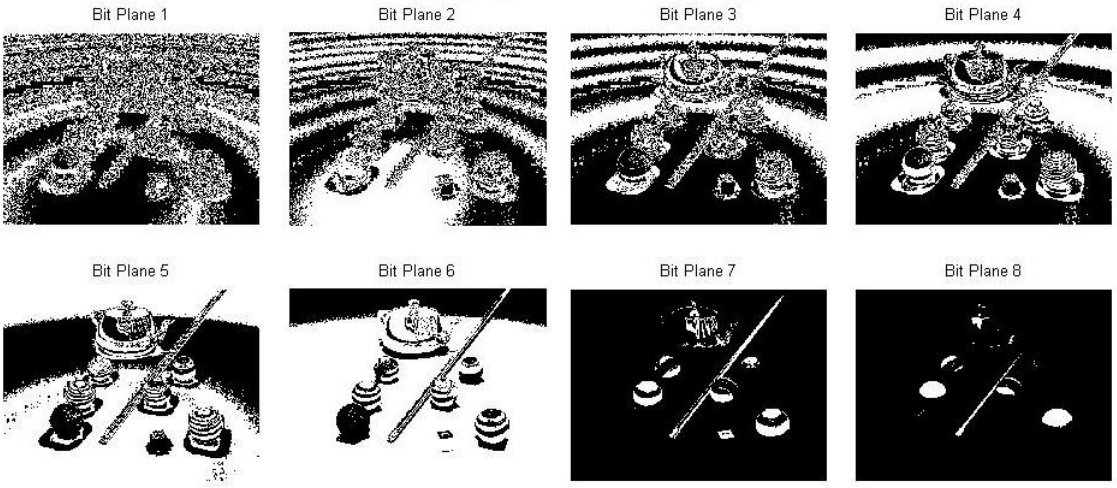

Imagem 3 (b): Planos de bit

É um pouco difícil fazer uma análise geral das diferenças entre os planos de bit da imagem 3. Pode-se notar que nos dois primeiros planos de bits não conseguimos ver muito os detalhes dos objetos da imagem, mas podemos identificar o fundo e as sombras das bolas de sinuca e do fundo. No terceiro e no quarto, os detalhes começam a ficar mais evidentes. No quinto e no sexto o fundo começa a ficar branco, possivelmente pelo fato de que como o fundo é de apenas uma cor (logo uma faixa de níveis de cinza próximos), os quintos e sextos bits dessa faixa de níveis de cinza são 1, tornando essa parte dos planos 5 e 6 branca. Já nos planos 7 e 8 o fundo volta a ser preto e perdemos a separações dos degrades de sombras do fundo.

Outras imagens foram analisadas e uma que teve um resultado muito bom foi a imagem da Lena, mostrada na *imagem 4*.

Imagem 4 (a): Lena

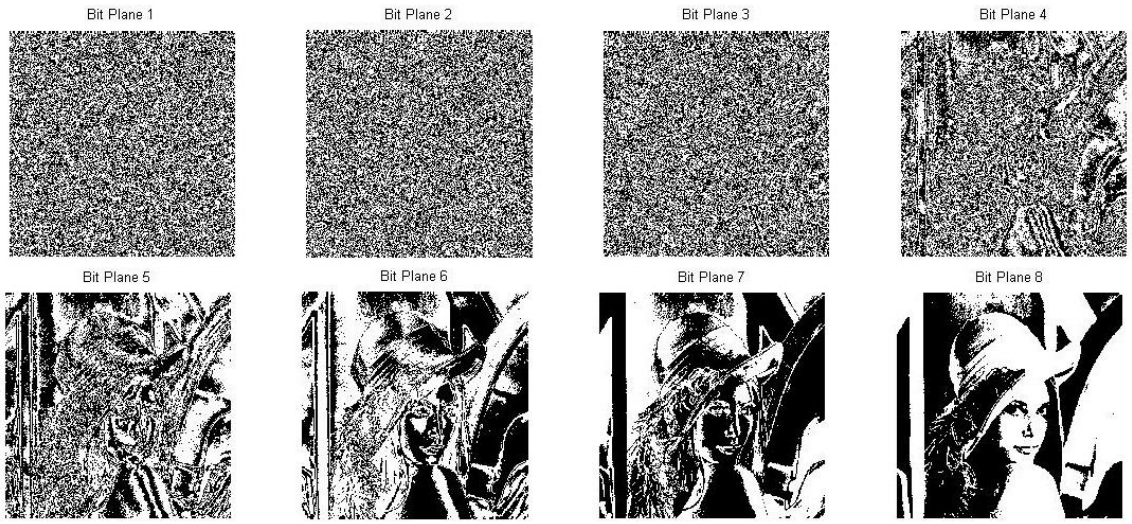

Imagem 4: planos de bit da imagem Lena.

Nos planos de bit na *imagem 4*, não conseguimos visualizar nenhum detalhe que possa identificar o conteúdo da imagem nos 4 primeiros planos de bit. Do quinto a imagem começa a tomar forma. No sexto e no sétimo já conseguimos identificar que é uma mulher. No oitavo temos uma maior presença de detalhes.

As análises das diferenças entre os planos de bit se mostraram muito dependentes da imagem, pois cada imagem apresenta um padrão diferente para os planos de bit.

Para a obtenção desses resultados foi utilizada a função do MATLAB bitget(A, i que retorna o i-ésimo bit mais significativo do array A [4].

Em seguida, foi utilizada a função logical() que converte valores numéricos para logicals (1 true, 0 false). Como o resultado de bitget é uma matriz com 0's e 1's, o uso de logical() é necessário, pois a função imshow() usada para mostrar as imagens espera uma imagem na qual o valor mínimo 0 é preto e o valor máximo 255 é branco. Aplicando logical() nessa matriz, convertemos os bits para true e false, deste modo imshow() mostra a imagem binária na tela.

3 - Comparações entre Imagens

A comparação entre Imagens foi feita utilizando os histogramas. O programa computa os histogramas (32 bins) de cores das imagens peppers.png e baboon.png para cada canal de cor e mostra as curvas na tela. As curvas obtidas estão presentes na *imagem 5*.

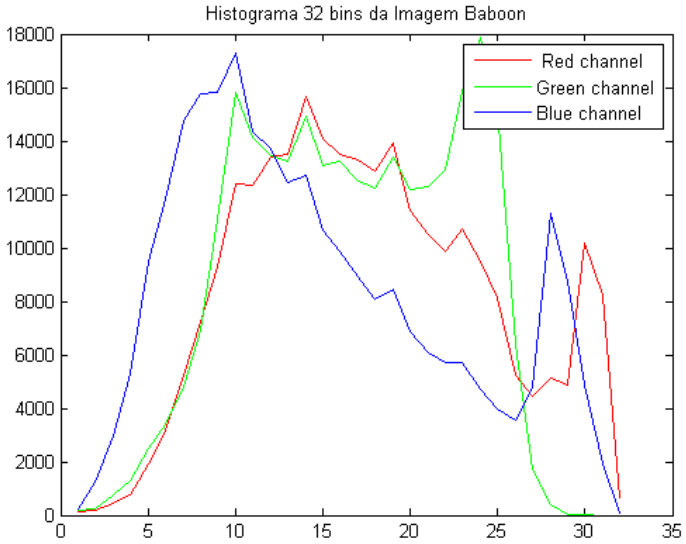

Imagem 5(a): Histogramas 32 bins da Imagem Baboon

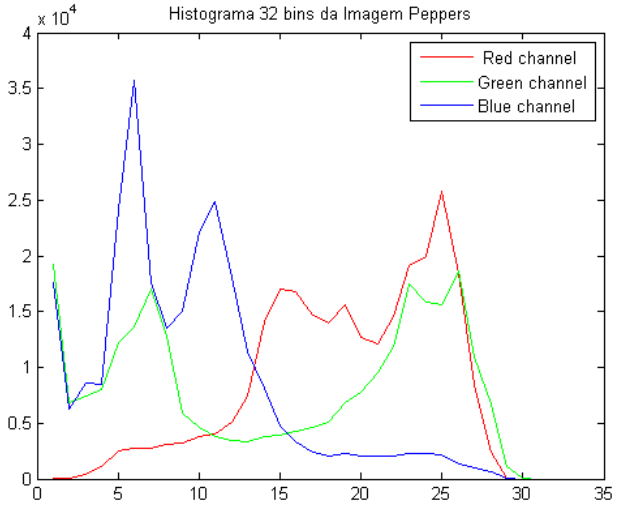

Imagem 5(b): Histogramas 32 bins da Imagem Peppers

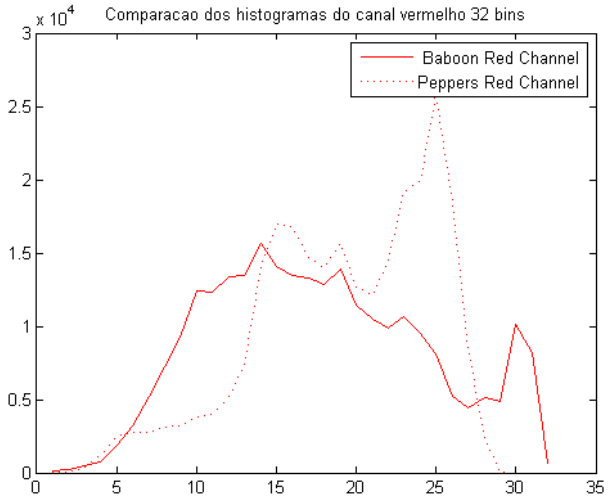

Imagem 5(c): Comparação dos histogramas do canal vermelho

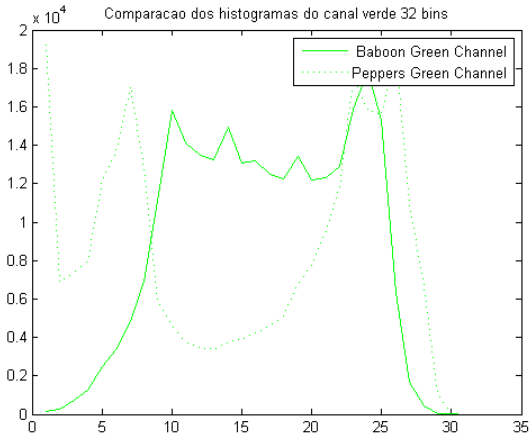

Imagem 5(d): Comparação dos histogramas do canal verde

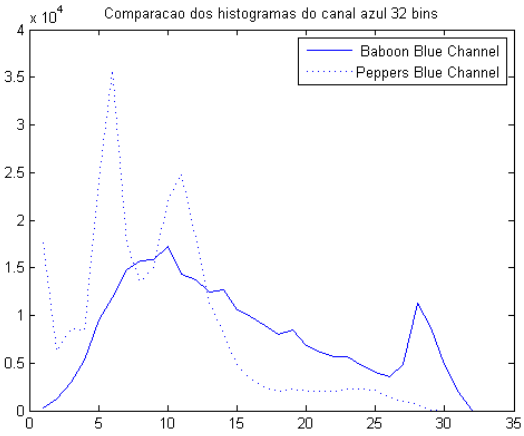

Imagem 5(d): Comparação dos histogramas do canal azul

Para comparar a similaridade das imagens, cada histograma de cor foi normalizado para que a soma dos elementos se torne 1 e então foi computada a distância Euclidiana dos dois histogramas de cor. Isso foi feito para cada canal com 4, 32, 128 e 256 bins. Os resultados se encontram na *Imagem 6* abaixo:

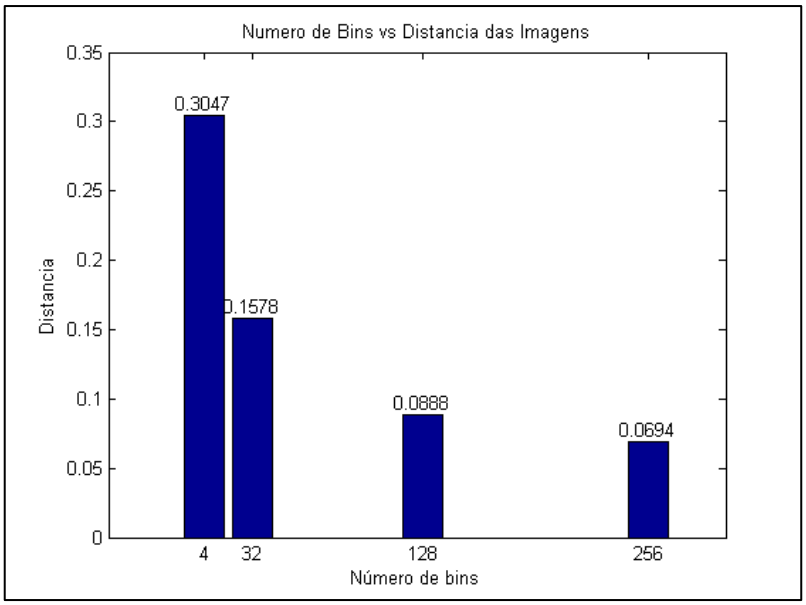

Imagem 6: Gráfico de barras do Número de Bins VS Distância das Imagens

Pela imagem acima podemos concluir que a distância entre as duas imagens **diminui** conforme aumentamos o número de bins. Isso pode ser explicado pelo fato de que aumentando o número de bins, cada bin se torna menos significativo já que os histograma estão normalizados.

4 – Filtragem

Para o procedimento de filtragem, foi adicionada ao programa uma função que aplica o filtro da média com a seguinte máscara de dimensões 3x3 pixels:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Inicialmente o programa lê uma imagem colorida de tamanho M x N e a converte para níveis de cinza. Em seguida, cria uma replica da matriz com dimensões M+2 x N+2 na qual as bordas possuem zeros e o restante possui a matriz da imagem. Depois, aplica a máscara do filtro da média por meio de um laço sobre todos os pixels da imagem com exceção dos pixels da borda. Cada pixel da matriz da imagem é substituído por R que é dado por:

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

Em que *wi* representa os coeficientes da máscara. No caso do filtro da média temos que $w_i = \frac{1}{9}, 1 \leq i \leq 9$. E zi representa a posição da submatriz 3x3 da imagem com centro em (x,y) para $2 \leq x \leq m-1,\ 2 \leq y \leq n-2$. [5]

Alguns detalhes da implementação: Para adicionar a borda de zeros foi usada a função padarray(A, size). A matriz com a imagem é convertida para Double para ser possível multiplicar seus elementos pela máscara. E por fim, ela foi convertida novamente para inteiros sem sinal de 8 bits. Existe uma função no matlab filter2(A, h) que, dada matriz (ou array) A e a máscara h, aplica o filtro na imagem, mas por razões didáticas ela não foi utilizada nesta parte do trabalho porque a intenção era aprender a aplicar um filtros pelo MATLAB.

A função de filtragem descrita acima foi aplicada para uma imagem sem ruídos e depois para essa mesma imagem com três tipos diferentes de ruídos: Poisson, Salt & Pepper e Speckles. Os resultados estão obtidos estão nas imagens 7 e 8.

Imagem7(a): Imagem Original

Imagem7(b): Imagem após a aplicação do filtro da média.

Agora com os ruídos na Imagem Original:

Imagem 8 (a): Imagem com ruído Salt & Pepper com d=0.01

Imagem 8 (b): Imagem com ruído Salt & Pepper filtrada

Imagem 8 (c): Imagem com ruído Poisson

Imagem 8 (d): Imagem com ruído Poisson filtrada

Imagem 8 (e): Imagem com ruído Speckle com v = 0.04

Imagem 8 (f): Imagem com ruído Speckle filtrada

O diretório *imagens_relatório* fornecido com o código possui as *Imagens 8 a,b,c,d,e,f* em um maior tamanho para a melhor visualização dos efeitos do filtro aplicado.

Como mostra a *imagem 7*, o efeito de aplicar o filtro na imagem original foi de um leve borramento na imagem.

Por outro lado, com a presença de ruído, o relativamente simples filtro da média mostrou bons resultados como podemos ver na imagem 8. Para o tipo de ruído Salt & Pepper (imagem 8(a) e (b)), apesar de ainda podemos identificar os pontos com ruídos, eles se tomaram bem menos expressivos. No ruído de Poisson, o resultado apresentado foi o melhor, pois a imagem com ruído apresentava um aspecto áspero e foi suavizada após a aplicação do filtro. Por outro lado para o ruído Speckle o resultado já não foi tão bom. Isso pode ser explicado pelo fato desse tipo de ruído com o parâmetro adotado de v = 0.04 ter atingido boa parte dos pixels da imagem.

Para a aplicação dos ruídos na imagem original foi utilizada a função *imnoise* do MATLAB.

5 – Mosaicos

Para a geração de um moisaco com blocos 4x4 de uma imagem, foi criada uma função que cria um array de mapeamento dos 16 blocos da imagem original para os 16 blocos da imagem destino. Esses blocos levam em conta o tamanho da imagem, portanto as dimensões M,N da imagem podem ser diferentes, contudo ambas dimensões devem ser múltiplos de 4. Esse array de mapeamento foi escolhido de forma arbitrária.

Em seguida cada bloco da imagem original é copiado para o bloco do moisaico de acordo com o especificado no array mapeamento.

Os resultados obtidos estão presentes na *imagem 9*:

Imagem 9(a): Imagem Original Quadrada

Imagem 9(b): Moisaco da Imagem Quadrada

Imagem 9(c): Imagem Original Retangular

Imagem 9(d): Moisaco da Imagem Retangular

Podemos ver que a função funcionou tanto para uma imagem quadrada quanto para uma imagem retangular.

Conclusões

Neste trabalho tive a oportunidade de ter um primeiro contato com o software MATLAB. O processo de aprendizado da utilização deste software, da sintaxe e de algumas funções básicas foi bem rápido devido a algumas semelhanças da linguagem MATLAB com algumas linguagens de programação como Python e C as quais já conheço. O uso do software se mostrou muito prático e eficiente na realização de operações com vetores e matrizes. Além disso, comparado com C, é muito mais fácil de ler e salvar arquivos, e mostrar as imagens na tela pelo MATLAB, sem falar na biblioteca extremamente poderosa do MATLAB.

A maior dificuldade foi descobrir que certos problemas na hora do desenvolvimento do programa se estavam ocorrendo devido a representação da imagem na forma de matriz (algumas vezes foi necessário converter para Double, inteiros sem sinal de 8 bits e logical).

Bibliografia

[1] Enunciado trabalho 1. Site processor Helio:
http://www.ic.unicamp.br/~helio/disciplinas/MC940/trabalho1.pdf

[2] Bit plane. Wikipedia.
https://en.wikipedia.org/wiki/Bit_plane

[3] Foggia, P. (2009). *Image analysis and processing ICIAP 2009 : 15th international conference, Vietri Sul Mare, Italy, September 8-11, 2009 : Proceedings*. Berlin: Springer. Pag. 894

[4] MathWorks.
http://www.mathworks.com/help/matlab/ref/bitget.html

[5] Aula 3. Realce Slides 60 ao 64.
http://www.ic.unicamp.br/~helio/disciplinas/MC940/aula_realce.pdf