

Relatório de Trabalho 2

David Clifte da Silva Vieira

2014, 21 de outubro

Introdução

Este trabalho apresenta o resultados obtidos durante o desenvolvimento da segunda lista de exercícios propostos pelo professor Pedro Pedrosa no curso de mestrado em Ciências da Computação do IFCE. Parte do código fonte é exibido na forma de Apêndice ao fim do trabalho. O código fonte detalhado será enviado junto ao trabalho.

1 Translação, Rotação e Redimensionamento: Tópico 11

Translação, Rotação e Redimensionamento são transformações lineares que podem ser aplicadas a um ponto, linha ou qualquer forma representável por um hiperplano. A forma original do objeto é comumente chamada de pré imagem e após as transformações realizadas na forma ou no posicionamento do objeto é simplesmente chamada de imagem. Os tipos de transformações que foram supracitados possuem uma característica entre si. Não importa a ordem de aplicação ou mesmo a intensidade dela, sempre será possível recuperar a forma original a partir da combinação das mesmas três transformações. Os códigos fonte utilizados nesta sessão encontram-se no Apêndice A.

1.1 Translação

A operação de translação move cada ponto da forma em um fator constante em uma direção específica. A forma do objeto não sofre qualquer alteração com essa transformação, apenas o posicionamento no espaço. A translação pode ser vista e interpretada como a adição de vetores constantes a cada ponto da forma ou como o deslocamento do centro de coordenadas. A ideia de se transladar um ponto de coordenadas (x, y) é calcular a nova posição (x', y') onde $(x', y') = (x + x_0, y + y_0)$ onde x_0 e y_0 é a quantidade transladada em cada eixo da coordenada ou simplesmente a nova coordenada da origem. Apesar de ser exibido a translação apenas para duas coordenadas esta operação pode ser feita em qualquer número de dimensões. A translação pode ser expressa de forma matricial. Abaixo temos a representação para a translação de uma forma geométrica em um espaço de três dimensões. De forma generalizada, para qualquer número de dimensões, podemos pensar na matriz de transformação em que todos os elementos da diagonal principal possuem o valor 1 e a última coluna com os valores da nova origem de coordenadas.

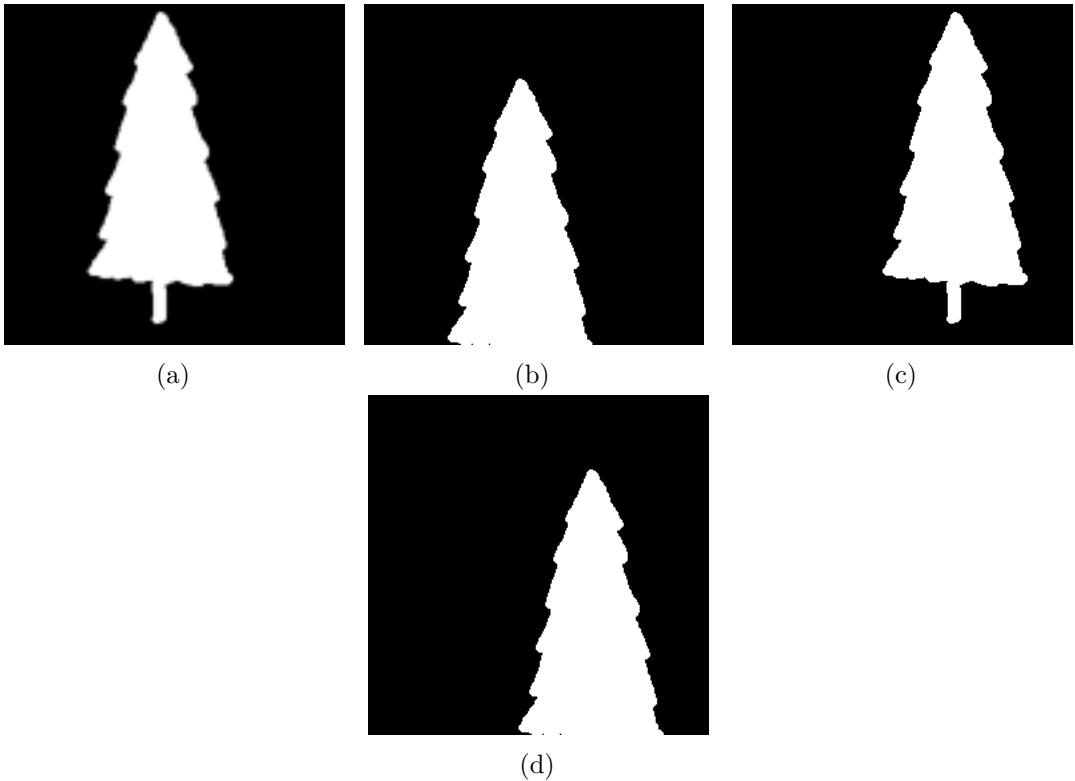


Figura 1 – 1a. Image original 1b. Imagem com translação vertical de 50 pixels 1c. Imagem com translação horizontal de 50 pixels 1d. Imagem com translação vertical e horizontal de 50 pixels

$$Im' = T * Im \quad (1)$$

$$Im' = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

Na figura 1 temos o resultado da translação na imagem.

1.2 Rotação

A operação de rotação é uma transformação geométrica que mapeia a posição de uma forma em um espaço levando em consideração o ângulo desejado na transformação. Todos os pontos da forma são rotacionados a partir de um ângulo constante em relação ao ponto de rotação, para rotação no \mathbb{R}^2 , reta no \mathbb{R}^3 ou hiperplano para o R^n . No R^2 os pontos de destino da forma podem ser calculados através das seguintes equações:

$$x' = \cos(\theta) * (x_1 - x_0) - \sin(\theta) * (y_1 - y_0) + x_0 \quad (3)$$

$$y' = \sin(\theta) * (x_1 - x_0) + \cos(\theta) * (y_1 - y_0) + y_0 \quad (4)$$

(5)

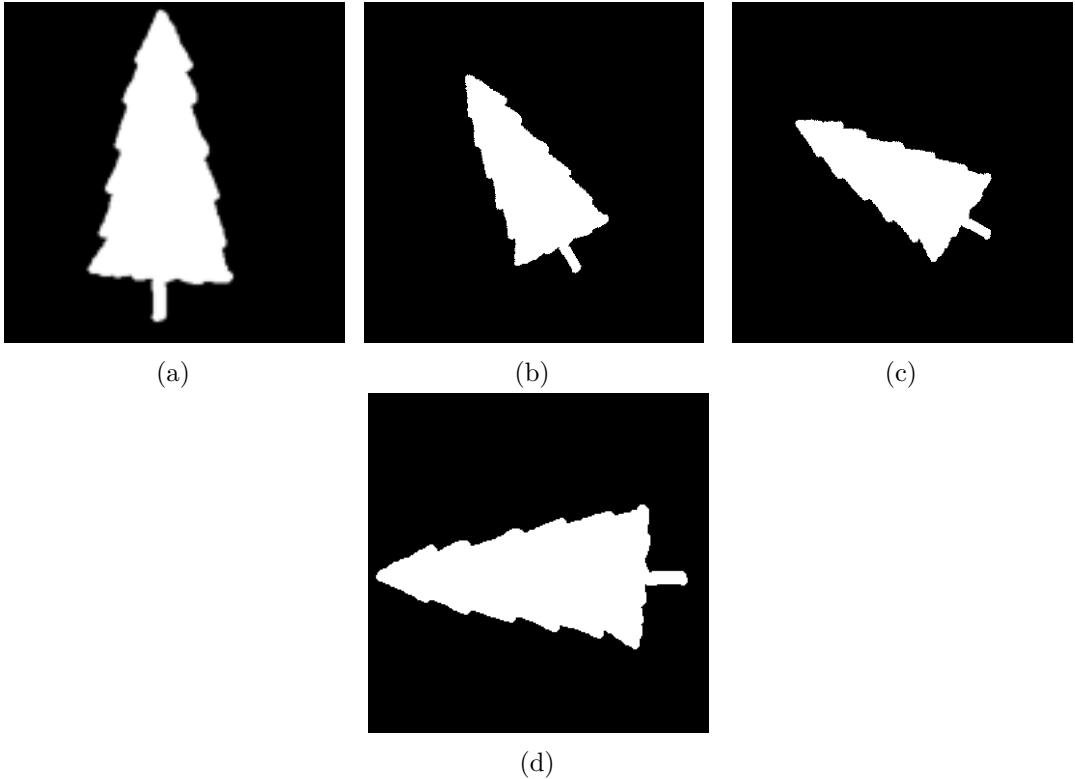


Figura 2 – **2a.** Image original **2b.** Imagem com rotação de 30° **2c.** Imagem com rotação de 60° **2d.** Imagem com rotação de 90°

ou em termos matriciais.

$$Im' = T * Im \quad (6)$$

$$Im' = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} \quad (7)$$

Ao fazer uso da forma matricial deve-se perceber que a rotação ocorrerá somente sobre a origem. Para contornar esta característica faz-se necessário o uso do operador de translação, deslocando assim o ponto desejado para o centro das coordenadas para só então realizar a rotação. Na figura 2 podemos ver o resultado da rotação.

1.3 Redimensionamento

O redimensionamento é uma transformação linear que aumenta ou diminui o tamanho de uma forma em um fator constante em todas as direções da forma. Normalmente este aumento ou redução é feita de forma uniforme, mas pode ser feita de forma diferente em cada eixo da representação da forma. Um fator de redimensionamento maior que 1 reflete em um aumento na imagem, menor que 1 em uma redução e um fator igual a 1 não é feita nenhuma transformação. É importante ressaltar que valores menor que 0 não são válidos. Bem como as operações de translação e rotação o redimensionamento também pode ser representado por um produto matricial. O redimensionamento de um vetor $\vec{v} = (x, y, z)$ por um fator f resulta em um vetor $\vec{w} = (fx, fy, fz)$, representado matricialmente por:

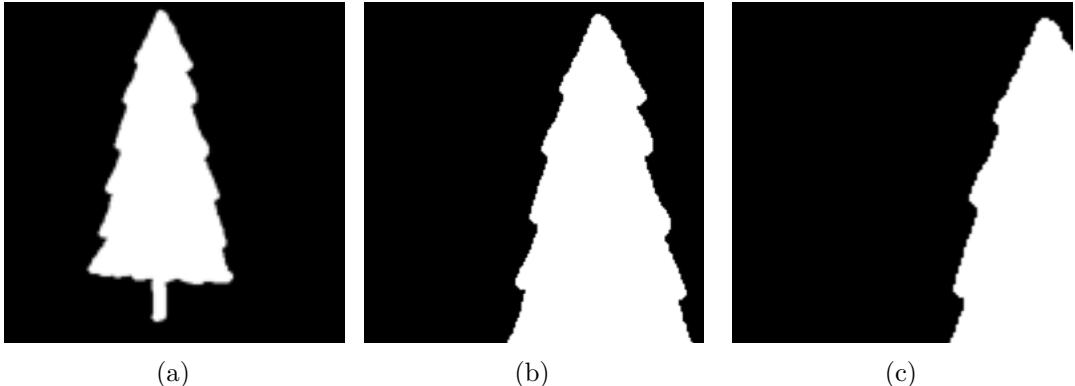


Figura 3 – **3a.** Image original **3b.** Redimensionamento com fator 1.5 **3c.** Redimensionamento com fator 2

$$Im' = T * Im \quad (8)$$

$$Im' = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & f_z \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (9)$$

f_x, f_y, f_z são os fatores de redimensionamento para cada dimensão do vetor.

2 Contraste, Brilho e Gama: Tópico 12

As técnicas de modificação de histograma são conhecidas como técnicas ponto-a-ponto, uma vez que o valor de tom de cinza de um certo pixel após o processamento depende apenas de seu valor original. Dentre as principais técnicas temos o ajuste de brilho, contraste e gama.

2.1 Brilho

O Brilho é um atributo de percepção visual no qual determina a intensidade de energia emitida ou refletida por uma fonte. O brilho é uma característica importante pois é ela que determina se a quantidade de luz é perceptível pelos sensores ou pela aplicação. O ajuste de intensidade do brilho é feito a partir da seguinte equação:

$$Im'(x, y) = Im(x, y) + b \quad (10)$$

onde b é o valor de ajuste do brilho.

2.2 Contraste

O contraste é um atributo que permite que objetos sejam discriminados dentro de um mesmo campo visual. Dois objetos com intensidades semelhantes são dificilmente distinguidos por sensores como o olho humano. O ajuste no contraste tem como princípio aumentar a diferença de intensidade entre os objetos.

$$Im'(x, y) = c * Im(x, y) \quad (11)$$

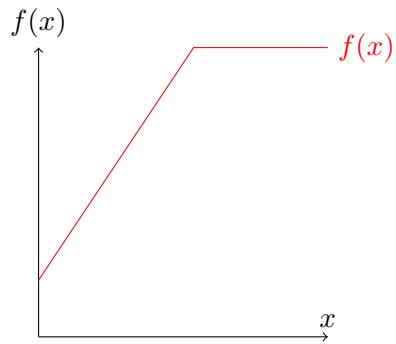


Figura 4 – Função de mapeamento com brilho=50 e contraste=1.5

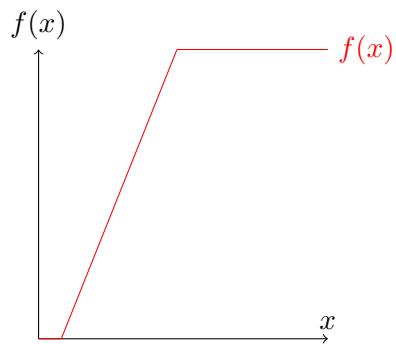


Figura 5 – Função de mapeamento com brilho=-50 e contraste=2.5

onde c é o valor de ajuste do contraste.

2.3 Gama

O gama, assim como o ajuste de contraste é utilizado para aumentar a diferença de intensidade entre os objetos porém de forma não linear. A regulação do gama pode ser feita com base na seguinte função.

$$Im'(x, y) = c * Im(x, y)^\gamma \quad (12)$$

onde c é um fator de correção do gama e γ é o fator de gama. O fator de correção é importante pois geralmente o função descrita acima retorna valores muito altos.

2.4 Resultado Obtidos

Na figura 8 temos os resultados obtidos após a realização de uma série de ajustes na figura 7. Podemos perceber através dos histogramas que a alteração do contraste 8 a,b, faz com que o hisograma seja alongado ou comprimido ao longo dos possíveis valores da imagem. Diferentemente do ajuste de contraste, o brilho apenas desloca o histograma, figura 8 c,d. Caso o valor do brilho seja positivo o histograma é deslocado para a região mais clara, caso negativo para a região mais escura. O ajuste de gama,figura 8 e,f, é um pouco mais peculiar que os dois anteriores. Por ser uma operação exponencial, caso o valor

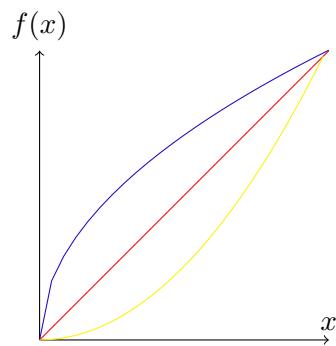


Figura 6 – Função de mapeamento do gama. vermelho, azul e amarelo possuem 1, 0,5 e 2 valores de gama respectivamente



Figura 7 – Imagem utilizada como teste para as operações de brilho, contraste e gama

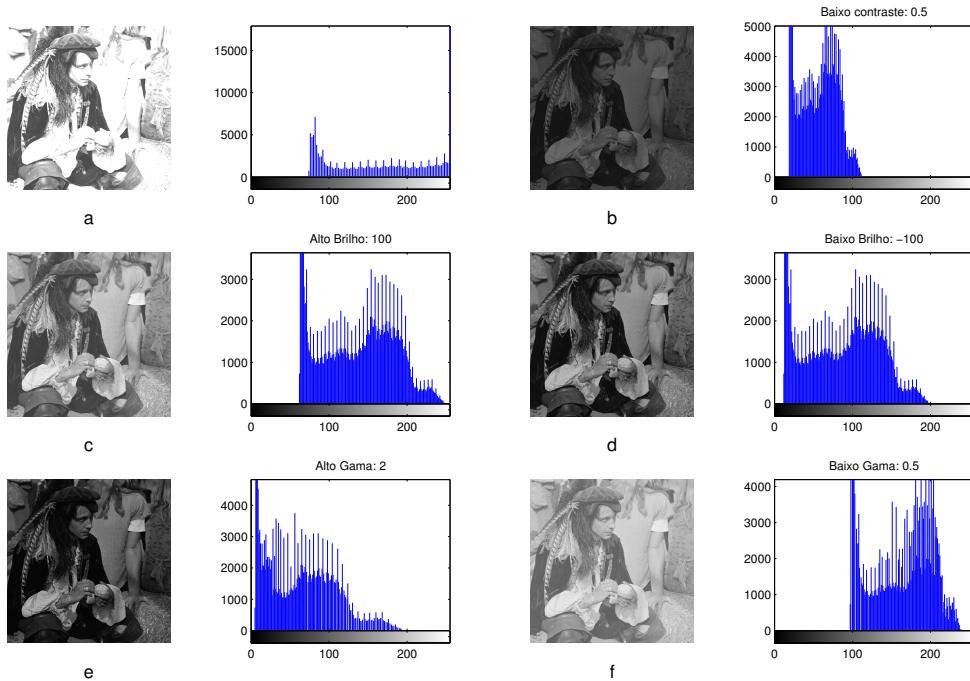


Figura 8 – Resultado da aplicação das técnicas de ajuste de histograma. a)Contraste aplicado 2 b)Contraste aplicado 0.5 c)Brilho aplicado 25 d)Brilho aplicado -25 e)Gama aplicado 2 f)Gama aplicado 0.5

seja maior que 1 a imagem como um todo terá a intensidade de seus pixels reduzidas, porém os pixels com valores de intensidade menor serão ainda mais afetados pelo operador. caso o valor seja menor que 1 a intensidade da imagem aumentará e da mesma forma os pixels com valores de intensidade menor sofrerão maiores mudanças. O código fonte deste exemplo pode ser visualizado no Apêndice B

3 Filtragem no domínio da frequência: Tópico 13 e 14

A transformada de Fourier é uma transformação matemática que permite que funções possam ser expressas em componentes senoidais. O somatório destas componentes tem como resultado a função original. A informação de frequencia destas componentes são muito importantes tendo em vista que com elas podemos identificar e filtrar ruídos que atuam em frequencias conhecidas. Também pode-se fazer a filtragem de componentes de maiores ou menores frequencias a partir de um limiar, filtros passa-baixa e filtros passa-alta respectivamente.

Para um sinal de uma dimensão temos a seguinte equação utilizada para calcular a transformada de fourier de $s(t)$:

$$S(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-i2\pi ft} dt. \quad (13)$$

Para duas dimensões temos a seguinte equação:

$$\hat{f}(\xi_x, \xi_y) = \iint f(x, y) e^{-2\pi i(\xi_x x + \xi_y y)} dx dy \quad (14)$$

ξ_x e ξ_y são as frequências calculadas sobre a imagem. À medida que nos aproximamos da origem da transformada, imagem calculada a partir da equação anterior, as baixas frequências correspondem aos componentes de intensidade de variação lenta em uma imagem como mudanças suaves de intensidade na parede ou em outras regiões uniformes da imagem. À medida que nos distanciamos da origem da transformada, as frequências mais altas começam a corresponder a variações de intensidade cada vez mais rápidas como bordas de objetos e outros elementos, figura 9. De acordo com as características citadas anteriormente podemos construir as matrizes para a filtragem no domínio da frequencia, figura 13. Uma das características do domínio da frequência é que a filtragem espacial pode ser realizada utilizando apenas um produto de matrizes. O resultado da filtragem pode ser visualizado na figura. O código fonte utilizado encontra-se no Apêndice C.

4 Segmentação Baseada em cores

4.1 Espaço de cores RGB

O espaço de cores RGB é formado por todas as combinações possíveis das cores R(Red), G(Green) e B(Blue), vermelho, verde e azul respectivamente. Esse espaço de cores é normalmente representado por um cubo onde cada eixo representa uma das componentes, ou canais, de cor, a intensidade de cada componente indica a cor resultante, figura 13a. Em imagens coloridas baseadas nesta representação o valor máximo de cada componente é 255. Na figura 12 temos o resultado de algumas combinações deste espaço de cores. Neste espaço de cores, vale ressaltar que a cor branca é formada pela combinação da máxima intensidade de todas as três componentes, a cor preta pela combinação da intensidade mínima das três componentes.



Figura 9 – 9a. Image original 9b. Espectro de Fourier da imagem.

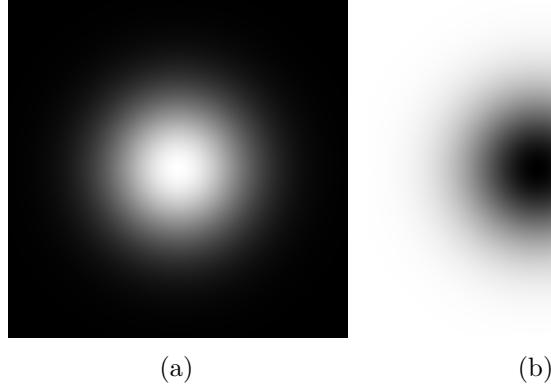


Figura 10 – 11c. Filtro passa baixa. 11d. Filtro passa alta.

4.2 Espaço de cores HSV

O espaço de cores HSV também conhecido como HSB, assim como o RGB possue 3 componentes para representar as cores. As três componentes estão organizadas de forma que suas componentes armazenem informação da matiz, saturação e brilho separadamente. O lugar geométrico deste espaço de cores é um cilindro, as vezes também representado por um cone, onde a altura é o nível de brilho, o raio a intensidade da cor e o ângulo a cor no espectro luminoso, matiz. A matiz, componente H, descreve a cor encontrada no espectro eletromagnético, é dessa componente que descrevemos as cores eg. vermelho, verde, laranja. A saturação, componente S, descreve a pureza da cor em relação a cor branca. Uma cor totalmente impura, valor mínimo desta componente, tem como cor resultante a cor branca. A última componente V ou B, o brilho, descreve quanta luz é emitida do objeto. A conversão entre os espaços de cores HSV e RGB pode ser feita facilmente com os seguintes comandos no Matlab.

```

1 % Converter RGB para HSV
2 hsv_image = rgb2hsv(rgb_image)
3 % Converter HSV para RGB
4 rgb_image = hsv2rgb(hsv_image)

```

No Matlab, por padrão as imagens coloridas são alocadas em matrizes onde cada canal representa uma das componentes do RGB. Cada elemento da matriz representa uma

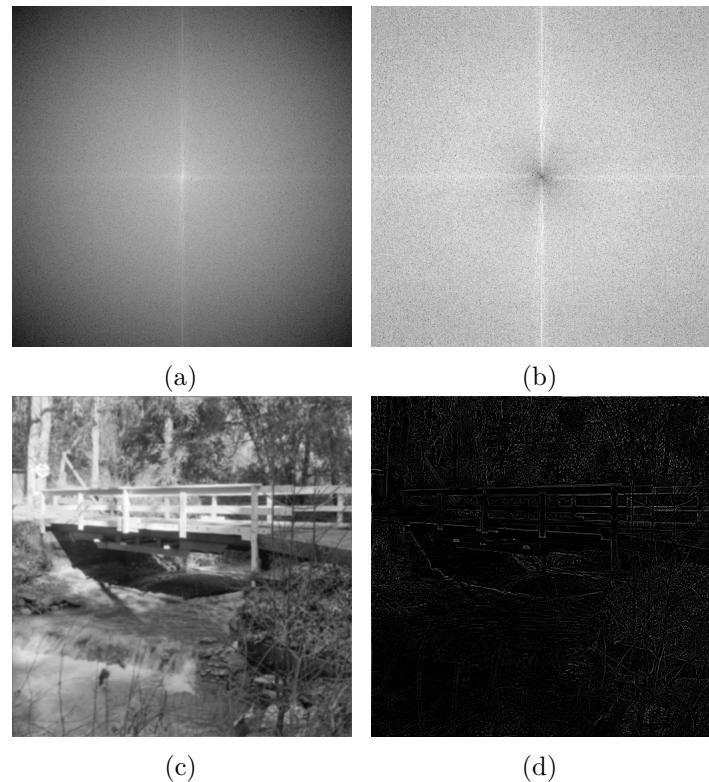


Figura 11 – [11c](#). Filtro passa baixa. [11d](#). Filtro passa alta.

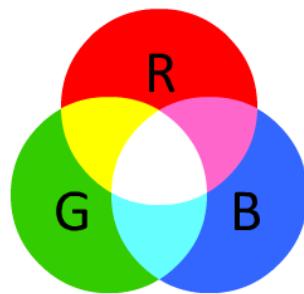


Figura 12 – Algumas combinações possíveis do espaço de cores RGB. Perceba que na interseção das cores são produzidas cores diferentes.

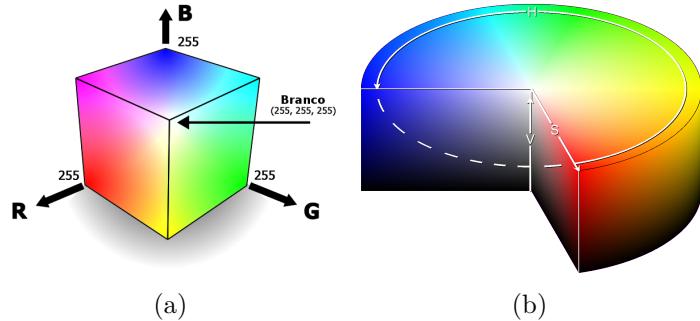


Figura 13 – 13a. Representação do espaço de cores RGB 13b. Representação do espaço de cores HSV.

componente de um pixel da imagem com valores variando entre $[0, 255]$. Após a conversão da imagem para o formato HSV as matrizes de cada componente têm seus valores variando entre $[0, 1]$.

4.3 Segmentação baseada na cor da pele: Tópico 16

A definição do limiar de segmentação baseado em cores é feita utilizando o espaço de cores HSV. Como dito anteriormente este espaço de cores possui como característica a separação da informação de cor de outras informações como luminosidade e pureza. A figura 14a foi utilizada para a definição dos limiares. Foi verificado que os valores H que melhores descrevem os tons da pele estão no intervalo $[0, 0.1]$, intervalo este que representa 10% da faixa de valores possíveis do canal H. Foi verificado também que para o canal S os valores que melhor representam o tom da pele estão na faixa de $[0.4, 0.8]$. O resultado da combinação dos dois limiares em S e H com a imagem de entrada pode ser visualizado na imagem 14d. O código fonte utilizado encontra-se no Apêndice D.

4.4 Rastreamento baseado em cor: Tópico 17 e 19

Da mesma forma que a sessão anterior foi feito a segmentação dos objetos baseada nas cores dos mesmos. Na figura 15 temos os objetos encontrados, três bolas vermelhas. Cada objeto tem como identificador o centróide do objeto. A identificação foi feita através da combinação de métodos de multi-limiarização, segmentação e contorno. Para esta atividade foi utilizado o OpenCV. O código fonte encontra-se no Apêndice G.

5 Transformada Haar: Tópico 15

A maneira mais fácil dentre as transformadas Wavelets disponível temos a transformada Haar(HWT). A transformada Haar é uma matriz ortogonal, logo $H^{-1} = H^T$, utilizada para calcular cada uma das componentes Wavelets, a saber: Vertical, horizontal, diagonal e aproximação. Abaixo temos a definição matemática da matriz de Haar.

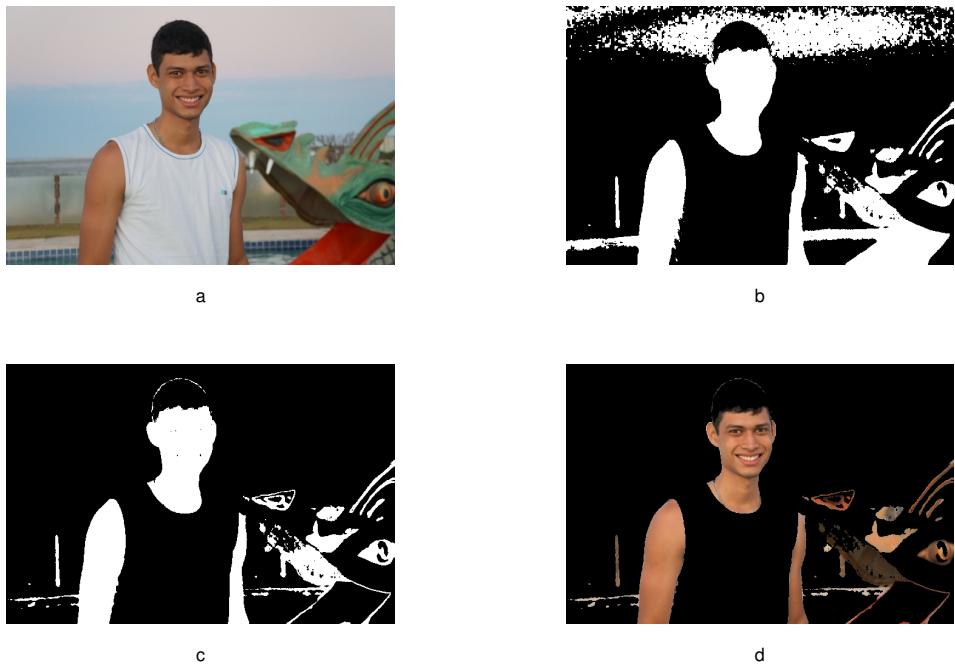


Figura 14 – Resultado da aplicação das técnicas de segmentação. a)Imagem Original
b)Filtro aplicado ao canal H. c)Filtro aplicado ao canal H e S d)Imagen resultante após os filtros.



Figura 15 – Rastreamento de objetos conectados baseado na cor.

$$H_N = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & \sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \sqrt{2}/2 & \sqrt{2}/2 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & -\sqrt{2}/2 & \sqrt{2}/2 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & -\sqrt{2}/2 & \sqrt{2}/2 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \quad (15)$$

Como podemos observar as primeiras $N/2$ linhas resultam em uma média ponderada com fator $\sqrt{2}$ da entrada. As últimas $N/2$ linhas resultam em uma diferença ponderada da entrada também com fator $\sqrt{2}$.

Na equação 15 temos a definição dos filtros passa baixa na primeira linha da matriz. $h = (h_0, h_1) = (\sqrt{2}/2, \sqrt{2}/2)$. É importante que além de realizar a média, esse filtro passa baixa amplia a sua intensidade em $\sqrt{2}$.

Na equação 15 temos também a definição do filtro passa alta na linha central da matriz. $g = (g_0, g_1) = (-\sqrt{2}/2, \sqrt{2}/2)$. Fazendo exatamente o contrário do filtro anterior, esse filtro amplia a intensidade do sinal apenas se houver valores distintos.

Se a imagem é uma matriz Im , com dimensões $m \times n$, temos que garantir que a imagem possuirá dimensões iguais e de potência de base 2, isso pode ser feito cortando ou adicionando elementos à matriz. Então após realizar estes ajustes, a HWT pode ser computada como.

$$W_H = H_N * Im * H_M^T = \begin{bmatrix} H \\ G \end{bmatrix} A \begin{bmatrix} H^T \\ G^T \end{bmatrix} = \begin{bmatrix} HA \\ GA \end{bmatrix} \begin{bmatrix} H^T \\ G^T \end{bmatrix} \quad (16)$$

$$W_H = \begin{bmatrix} HAH^T \\ HAG^T \end{bmatrix} \begin{bmatrix} GAH^T \\ GAG^T \end{bmatrix} \quad (17)$$

É importante notar o significado de cada bloco na equação 17. O bloco superior esquerdo calcula a média da imagem pois a matriz H é um filtro passa baixa, por esse motivo este bloco recebe o nome de componente de aproximação. O bloco superior direito é também uma média, porém combinado com um filtro passa baixa atuando na direção vertical, daí o nome componente vertical. O bloco inferior esquerdo, assim como o superior direito é uma combinação de filtros passa baixa e passa alta, desta vez com um filtro passa alta atuando na direção horizontal. O bloco inferior direito é uma combinação dos filtros passa alta nas duas direções, vertical e horizontal, sendo chamado por essa característica de componente diagonal. O resultado desta operação e a imagem obtida de cada bloco é exibido na Figura 16. O código fonte utilizado pode ser visualizado no apêndice E

Uma prática comum, principalmente em atividades de compressão de imagem e reconhecimento de padrões é a reaplicação desta transformada na componente de aproximação da transformada. Nas figura 16, 17 e 18 temos o resultado destas aplicações sucessivas. perceba que ao longo destas aplicações a imagem de aproximação tem suas informações distribuídas nas componentes verticais, horizontais e diagonais. As técnicas de compressão se valem dessa característica e valores nas componentes que tendem a zero são

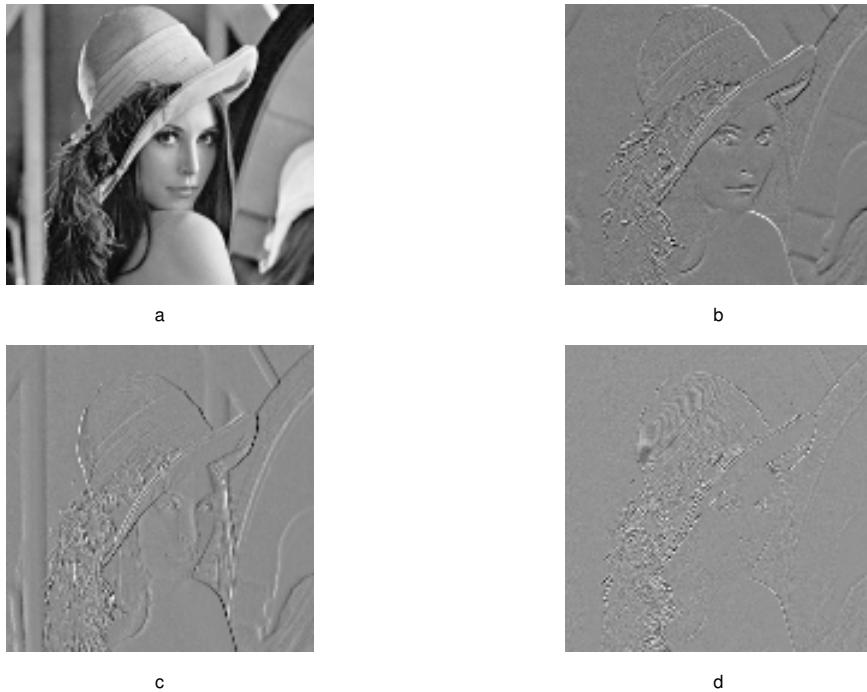


Figura 16 – Resultado da transformada de Haar na imagem Original, Nível 1.
 a) Componente de aproximação. b) Componente de detalhes verticais
 c) Componente de detalhes horizontais d) Componente de detalhes diagonais.

simplesmente zerados encaminhados para o codificador. Já em reconhecimento de padrões as informações presentes nestas componentes podem descrever características específicas dos objetos.

6 Morfologia Matemática: Tópico 18

A palavra Morfologia é originalmente um ramo da biologia que estuda as formas e estruturas dos animais e plantas. Usamos esta palavra no contexto de Morfologia Matemática como um instrumento para extração de componentes da imagem que sejam úteis para representação e descrição da forma de uma região, como fronteiras, esqueletos e fecho convexo.

A morfologia matemática é o estudo quantificado da forma e estrutura de conjuntos de pontos (no caso de imagens, conjunto de pixels), cujo principal objetivo é revelar a estrutura dos objetos formados pelos pontos através da transformação dos conjuntos que os modelam. Isto é realizado através de OMs. A partir desta estreita relação da morfologia matemática com a forma, torna-se natural aplicá-la como técnica de processamento de imagens (Cardoso, 1999).

Um elemento estruturante é um conjunto de coordenadas de pixel. Por exemplo, o elemento cruz é definido por $E = \{(0, 0), (-1, 0), (1, 0), (0, -1), (0, 1)\}$. Uma transformação morfológica requer uma operação não-linear entre a imagem e o elemento estruturante, o qual desliza sobre a imagem de forma similar a uma convolução.

Os operadores utilizados no processamento morfológico de imagem formam uma

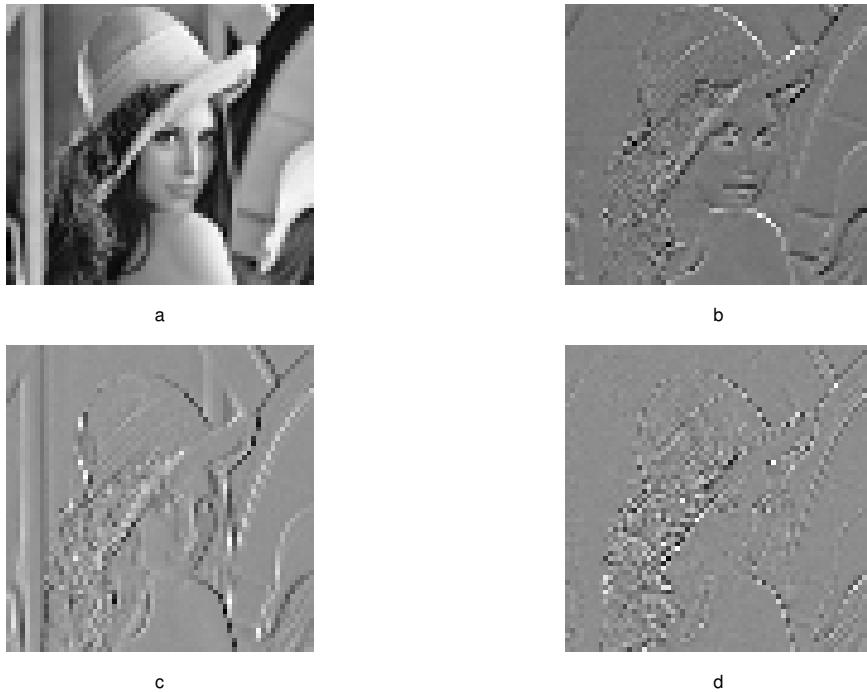


Figura 17 – Resultado da transformada de Haar aplicada na componente de aproximação, figura 16a, Nível 2. a)Componente de aproximação. b)Componente de detalhes verticais c)Componente de detalhes horizontais d)Componente de detalhes diagonais.

ampla classe de operadores não-lineares. Porém, todos estes são construídos pela interação de duas operações básicas: a erosão e a dilatação.

A erosão(\ominus) de uma imagem F para um dado pixel x é definida como o valor mínimo da imagem em uma janela definida pelo elemento estruturante, estando a origem do elemento estruturante na posição de x (Soille, 1999). Por outro lado, a dilatação(\oplus) é definida como o valor máximo da imagem em uma janela definida também por um elemento estruturante. Na figura 20a, 20b e 20b temos o resultado da aplicação para três elementos estruturantes diferentes 6. Podemos perceber nos resultados que a mudança do elemento estruturante provoca uma modificação significante na imagem. O primeiro elemento estruturante permite que a imagem seja dilatada ou erodida em todos os sentidos enquanto os dois últimos atuam somente nas diagonais, uma em cada direção.

$$A \ominus B = \{x | (B)_x \cap A \subseteq A\} \quad (18)$$

$$A \oplus B = \{x | (B^{-1})_x \cap A \neq \emptyset\} \quad (19)$$

7 Gradiente Morfológico: Tópico 20

Esta operação é composta de três outras operações básicas da morfologia: a dilatação, erosão e a subtração e é definida da seguinte forma:

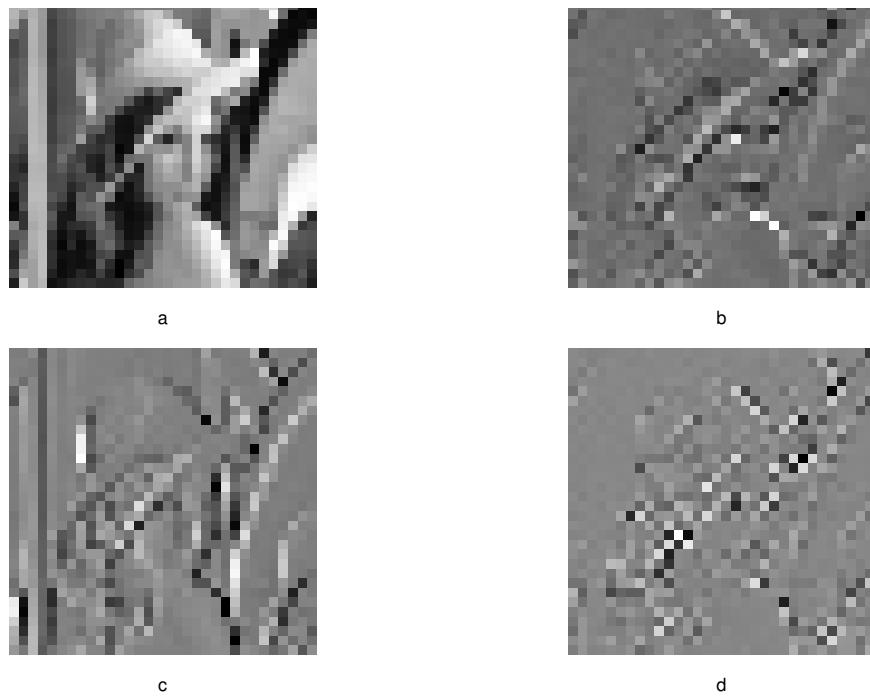


Figura 18 – Resultado da transformada de Haar aplicada na componente de aproximação, figura 17a, Nível 3. a)Componente de aproximação. b)Componente de detalhes verticais c)Componente de detalhes horizontais d)Componente de detalhes diagonais.

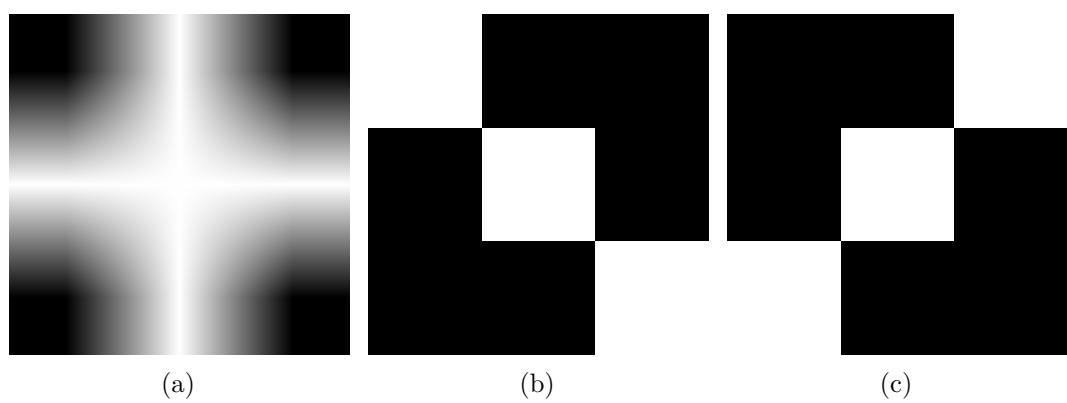
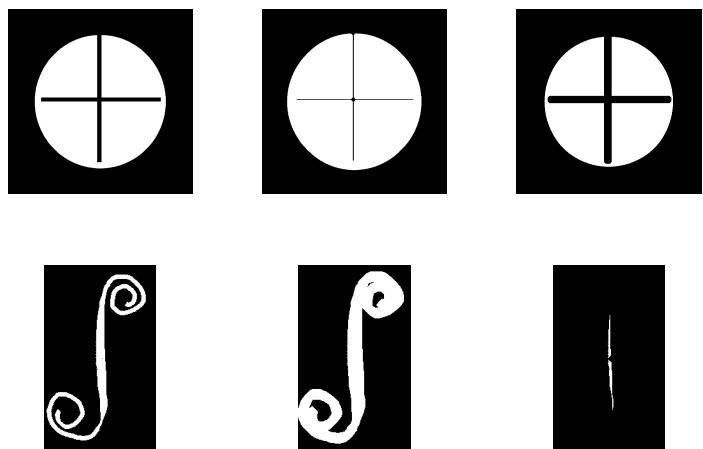
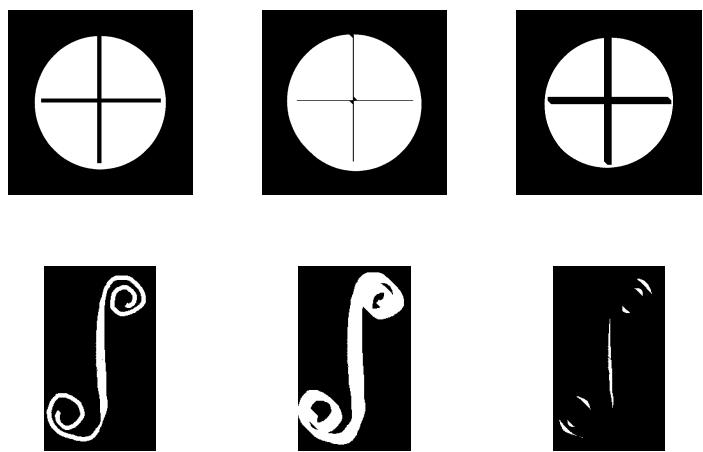


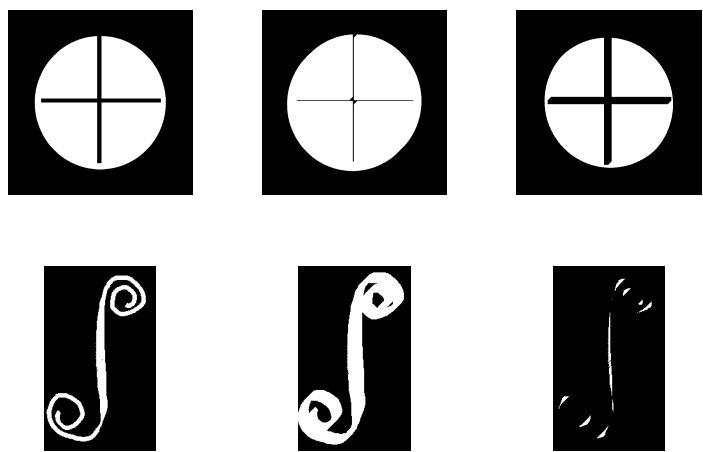
Figura 19 – 19a. Elemento estruturante em forma de cruz 19b. Elemento estruturante 135° 19c. Elemento estruturante 45°



(a) Resultado com elemento estruturante em forma de cruz



(b) Resultado com elemento estruturante 135°



(c) Resultado com elemento struturante 45°

Figura 20 – Na primeira coluna das figuras 20a, 20b e 20c temos a imagem orinal, ao centro temos o resultado da dilatação e na última coluna o resultado da erosão.



Figura 21 – Resultado da aplicação do gradiente morfológico.

$$\beta(A) = (A \oplus B) - (A \ominus B) \quad (20)$$

O processo destaca as transições entre os níveis de cinza (contornos). O resultado da operação depende do elemento estruturante utilizado, um elemento pequeno privilegia contornos delgados. É interessante notar que a simetria do elemento vai fazer com que o resultado da operação tenda a depender menos dos aspectos direcionais das bordas. Na figura 21 temos o resultado da aplicação do gradiente morfológico em 2 imagens tons de cinza. O código utilizado nesta implementação pode ser visualizado no Apêndice F

Referências

Cardoso, E.S.J. (1999), Compressão de Imagens Utilizando Decomposições em Multi-resolução Morfológicas, Tese de Mestrado, Programa de Engenharia Elétrica, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 114 p.

Soille, P. (1999): Morphological Image Analysis, Heidelberg: Springer-Verlag.

Alvarenga, A.V. , Application of morphological operators on the segmentation and contour detection of ultrasound breast images, Revista Brasileira de Engenharia Biomédica, v. 19, n. 2, p. 91-101, agosto 2003

APÊNDICE A – Código fonte: Translação, Rotação e Redimensionamento

```
1 clear;
2 clc;
3 image = imread('../bd/tree.gif');
4 [m n] = size(image);
5 imshow(image);
6
7 %%
8 %Operao de rotao
9 image30 = imrotate(image,30);
10 image60 = imrotate(image,60);
11 image90 = imrotate(image,90);
12
13
14 imwrite(image30,'image30.png');
15 imwrite(image60,'image60.png');
16 imwrite(image90,'image90.png');
17
18 %%
19 %Operao de redimensionamento
20 imageScale1 = imresize(image, 0.5);
21 imageScale2 = imresize(image, 1.5);
22 imageScale3 = imresize(image, 2);
23
24 imwrite(imageScale1,'imageScale1.png');
25 imwrite(imageScale2(1:m,1:n),'imageScale2.png');
26 imwrite(imageScale3(1:m,1:n),'imageScale3.png');
27
28
29 %%
30 %Operao de translao
31 imageT1 = imtranslate( image, [50, 0] );
32 imageT2 = imtranslate( image, [0 50] );
33 imageT3 = imtranslate( image, [50 50] );
34
35 imwrite(imageT1,'imageT1.png');
36 imwrite(imageT2,'imageT2.png');
37 imwrite(imageT3,'imageT3.png');
```

APÊNDICE B – Código fonte: Ajuste de Brilho, Contraste e Gama

```
1 function res = ajustaBrilho(image, b)
2     res = image + b;
```

```
3 | end
```

```
1 | function res = ajustaContraste(image, c)
2 |     res = image*c;
3 | end
```

```
1 |
2 | function res = ajustaGama(image,g)
3 |     res = uint8(255*((double(image)/255).^g));
4 | end
```

```
1 | close all;clear;
2 | %Carregando e exibindo Imagem Original
3 | image = imread('../bd/pirate.tif');
4 | figure
5 | subplot(1,2,1);
6 | imshow(image);
7 | subplot(1,2,2);
8 | imhist(image);
9 |
10 | figure;
11 | res = ajustaContraste(image,2);
12 | subplot(3,4,1);
13 | imshow(res);
14 | xlabel('a', 'FontSize', 15)
15 | subplot(3,4,2);
16 | imhist(res);
17 | imwrite(res,'altoContraste.png');
18 |
19 |
20 | res = ajustaContraste(image,0.5);
21 | subplot(3,4,3);
22 | imshow(res);
23 | xlabel('b', 'FontSize', 15)
24 | subplot(3,4,4);
25 | imhist(res);
26 | title('Baixo contraste: 0.5')
27 | imwrite(res,'baixoContraste.png');
28 |
29 |
30 |
31 | res = ajustaBrilho(image,25);
32 | subplot(3,4,5);
33 | imshow(res);
34 | xlabel('c', 'FontSize', 15)
35 | subplot(3,4,6);
36 | imhist(res);
37 | title('Alto Brilho: 100')
38 | imwrite(res,'altoBrilho.png');
39 |
40 |
41 | res = ajustaBrilho(image,-25);
42 | subplot(3,4,7);
43 | imshow(res);
44 | xlabel('d', 'FontSize', 15)
45 | subplot(3,4,8);
```

```

46 imhist(res);
47 title('Baixo Brilho: -100')
48 imwrite(res, 'baixoBrilho.png');
49
50 res = ajustaGama(image,2);
51 subplot(3,4,9);
52 imshow(res);
53 xlabel('e', 'FontSize', 15)
54 subplot(3,4,10);
55 imhist(res);
56 title('Alto Gama: 2')
57 imwrite(res, 'altoGama.png');
58
59 res = ajustaGama(image,0.5);
60 subplot(3,4,11);
61 imshow(res);
62 xlabel('f', 'FontSize', 15)
63 subplot(3,4,12);
64 imhist(res);
65 title('Baixo Gama: 0.5')
66 imwrite(res, 'baixoGama.png');

```

APÊNDICE C – Código fonte: Filtragem no domínio da frequência

```

1 close all;clear
2
3 %Carregando imagem
4 image = imread('../bd/walkbridge.tif');
5 image = image(:,:,1);
6
7 subplot(4,2,1);
8 imshow(image);
9 title('Original');
10
11 %Aplicando transformada de Fourier
12 spec = fft2(image);
13
14 %Exibindo Spectro de Fourier
15 ;
16 subplot(4,2,2)
17 surf(log(abs(spec))), 'EdgeColor', 'None')
18 imwrite(mat2gray(log(abs(spec))), 'espectroOriginal.png')
19 title('Espectro');
20
21
22 %% Passa Baixa
23
24 var = 3;
25 d = 20/length(spec(1,:));
26
27 %Calculando filtro gaussiano
28 filtro = gaussmf(-10:d:10,[var 0])' * gaussmf(-10:d:10,[var 0]);
29 filtro = filtro( 1:length(spec(1,:)), 1:length(spec(1,:)));
30 imwrite(mat2gray(filtro), 'passaBaixa.png');
31 %filtro = double(filtro > 0.5);
32

```

```

33    ;
34 subplot(4,2,3)
35 surf(filtro,'EdgeColor','None')
36 title('Filtro Passa Baixa');
37
38 %Filtrando
39 filtrada = filtro .* fftshift(spec);
40
41 ;
42 subplot(4,2,5)
43 surf(log(abs(filtrada)), 'EdgeColor','None')
44 imwrite(mat2gray(log(abs(filtrada))), 'filtradaPBSpec.png');
45 title('Filtrada Baixa');
46
47 %Exibindo imagem resultante
48
49 espaco = uint8(ifft2(fftshift(filtrada)));
50
51 imwrite(espaco, 'filtradaPB.png')
52 subplot(4,2,7)
53 imshow(espaco);
54 title('Resultado Passa Baixa');
55
56
57
58
59
60 %% Passa Alta
61
62 var = 3;
63 d = 20/length(spec(1,:));
64
65 %Calculando filtro gaussiano invertido
66 filtro = gaussmf(-10:d:10,[var 0])' * gaussmf(-10:d:10,[var 0]);
67 filtro = filtro( 1:length(spec(1,:)), 1:length(spec(1,:)));
68 filtro = 1 - filtro;
69 imwrite(mat2gray(filtro), 'passaAlta.png');
70
71 %filtro = double(filtro < 0.5);
72
73 ;
74 subplot(4,2,4)
75 surf(filtro, 'EdgeColor','None')
76 title('Filtro Passa Alta');
77
78 %Filtrando
79 filtrada = filtro .* fftshift(spec);
80
81 ;
82 imwrite(mat2gray(log(abs(filtrada+1))), 'filtradaPASpec.png');
83 subplot(4,2,6)
84 surf(log(abs(filtrada)), 'EdgeColor','None')
85 title('Filtrada Passa Alta');
86
87 %Exibindo imagem resultante
88
89 espaco = mat2gray(uint8(ifft2(fftshift(filtrada))));
90
91 imwrite(espaco, 'filtradaPA.png')
92 subplot(4,2,8)

```

```

93 | imshow(espaco);
94 | title('Resultado Passa Alta');

```

APÊNDICE D – Código fonte: Segmentação de Pele

```

1 close all;
2 clc;
3 %Carregando imagem
4 image = imread('../bd/eu.JPG');
5
6 subplot(2,2,1)
7 imshow(image)
8 xlabel('a', 'FontSize', 15)
9
10 %Separando canais HSV
11 hsv_image = rgb2hsv(image) ;
12 h = hsv_image(:,:,1);
13 s = hsv_image(:,:,2);
14 v = hsv_image(:,:,3);
15
16 %Filtrando tons da pele
17 h = h<0.1 & h>0;
18 s = s> 0.4 & s < 0.8;
19 skinMask = double(s & h);
20
21 %Canal H filtrado
22 subplot(2,2,2)
23 imshow(h)
24 xlabel('b', 'FontSize', 15)
25
26 %Canal S filtrado combinado com H filtrado
27 subplot(2,2,3)
28 imshow(skinMask)
29 xlabel('c', 'FontSize', 15)
30
31 %Imagem resultante
32 res = []
33 res(:,:,:,1) = double(image(:,:,:)) .* skinMask;
34 res(:,:,:,2) = double(image(:,:,:)) .* skinMask;
35 res(:,:,:,3) = double(image(:,:,:)) .* skinMask;
36
37 subplot(2,2,4)
38 imshow(mat2gray(res))
39 xlabel('d', 'FontSize', 15)

```

APÊNDICE E – Código fonte: Transformada de Haar

```

1 clear all;
2 close all;
3
4 %%Carregando imagem

```

```

5 N = 256;
6 lowLevel = 2;
7 image = imread('../bd/lena_gray_256.tif');
8
9 %Pegando imagem quadrada
10 image = image(1:N,1:N);
11 figure;
12 imshow(image);
13 title 'Original'
14
15 %mask = ConstructHaarWaveletTransformationMatrix(N);
16 %Calculando decomposio nvel 1
17 [dH dV dD dA] = haarDecomposition(image)
18 figure
19 exibeDecomposicao( dH, dV, dD, dA );
20
21
22 %Calculando decomposio nvel 2
23 figure
24 [dH dV dD dA] = haarDecomposition(dA)
25 exibeDecomposicao( dH, dV, dD, dA );
26
27 %Calculando decomposio nvel 3
28 figure
29 [dH dV dD dA] = haarDecomposition(dA)
30 exibeDecomposicao( dH, dV, dD, dA );

```

APÊNDICE F – Código fonte: Gradiente Morfológico

```

1 clear all;
2 close all;
3 clc;
4 %Carregando imagem
5 A = imread('../bd/lena_gray_256.tif');
6 A2 = imread('../bd/pirate.png');
7 %Elemento estruturante
8 B = [ 0 1 0
9         1 1 1
10        0 1 0
11    ]
12 %Calculando gradiente morfologico
13 G = imdilate(A,B) - imerode(A,B);
14 subaxis(2,2,1, 'Spacing', 0.08, 'Padding', 0, 'MB', 0.1, 'MT', 0.01);
15 imshow(A)
16 subaxis(2,2,2, 'Spacing', 0.08, 'Padding', 0, 'MB', 0.1, 'MT', 0.01);
17 imshow(G)
18
19 %Calculando gradiente morfologico
20 G2 = imdilate(A2,B) - imerode(A2,B);
21 subaxis(2,2,3, 'Spacing', 0.08, 'Padding', 0, 'MB', 0.1, 'MT', 0.01);
22 imshow(A2)
23 subaxis(2,2,4, 'Spacing', 0.08, 'Padding', 0, 'MB', 0.1, 'MT', 0.01);
24 imshow(G2)

```

APÊNDICE G – Código fonte: Rastreamento baseado em cor

```
1  public static void detectObjects(Mat srcImage, Mat dstImage) {
2      Mat edges = new Mat(srcImage.size(), CvType.CV_8U);
3
4      Imgproc.Canny(srcImage, edges, 128, 255);
5
6      //Encontra objetos conectados na imagem segmentada por Canny
7      List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
8      Imgproc.findContours(edges, contours, new Mat(),
9                          Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
10
11     for (int i = 0; i < contours.size(); i++) {
12         MatOfPoint pts = (MatOfPoint) contours.get(i);
13
14         //Filtrando pontos pequenos
15         if (pts.rows() < 10)
16             continue;
17
18         //Filtrando formas quadradas
19         Rect rect = Imgproc.boundingRect(contours.get(i));
20         float k = Math.abs(1 - rect.width/rect.height);
21         if(k>0.1 || rect.height < 30 ) continue;
22
23
24         double area = Imgproc.contourArea(pts);
25
26         System.out.println(k + " " + rect);
27         // System.out.println(rect.x
28         // +" , "+rect.y+", "+rect.height+", "+rect.width);
29         Point pt1 = new Point(rect.x, rect.y);
30         Point pt2 = new Point(rect.x + rect.width, rect.y + rect.height);
31         Point center = new Point(rect.x + rect.width / 2, rect.y
32                               + rect.height / 2);
33
34         Scalar cor = new Scalar(0, 0, 255);
35
36         Core.rectangle(dstImage, pt1, pt2, cor, 3);
37
38         Core.putText(dstImage, "(" + center.toString() + ")", pt2,
39                      Core.FONT_HERSHEY_SIMPLEX, 0.5, cor);
40     }
41 }
```

```
1  //Realiza a filtragem baseada em cores de acordo com as variáveis globais
2  //sValue,Limiar de saturação
3  // hueValue e hueWidth, valores do tamanho da janela e do centro
4  //da cor utilizada na filtragem
5
6  public void getFilteredImage(Mat frame, Mat dst) {
7
8      if(hsv == null){
9          Size d = new Size(frame.cols(), frame.height());
10         hsv = new Mat(d, CvType.CV_8UC3);
```

```

11
12     h = new Mat(d, CvType.CV_8U);
13     s = new Mat(d, CvType.CV_8U);
14     v = new Mat(d, CvType.CV_8U);
15
16     h1 = new Mat(d, CvType.CV_8U);
17     h2 = new Mat(d, CvType.CV_8U);
18
19 }
20
21
22 Imgproc.cvtColor(frame, hsv, Imgproc.COLOR_BGR2HSV);
23 Core.split(hsv, lst);
24
25 h = lst.get(0);
26 s = lst.get(1);
27 v = lst.get(2);
28
29 Imgproc.threshold(v, v, 20, 255, Imgproc.THRESH_BINARY);
30 Imgproc.threshold(s, s, sValue, 255, Imgproc.THRESH_BINARY);
31
32
33 int ls = hueValue+hueWidth;
34 int li = hueValue-hueWidth;
35
36 if( li>=0 && ls<=180){
37     Core.inRange(h, new Scalar(li),new Scalar(ls), dst);
38     System.out.println("^L1 = " + (li) + "      L2=" + (ls));
39 }else{
40     if( ls > 180){
41         Core.inRange(h, new Scalar(ls-180),new Scalar(li), dst);
42         Core.bitwise_not(dst, dst);
43         System.out.println("^L1 = " + (ls-180) + "      L2=" + (li));
44     }else{
45         Core.inRange(h, new Scalar(ls),new Scalar(180+li), dst);
46         Core.bitwise_not(dst, dst);
47         System.out.println("^L1 = " + (ls) + "      L2=" + (180+li));
48     }
49 }
50
51
52 Core.bitwise_and(dst, v, dst);
53 Core.bitwise_and(dst, s, dst);
54
55 }

```