EEL 7825 - Projeto Nível II em Cont e Proc de Sinais I
Daniel Augusto Figueiredo Collier
Exercícios - Aula 2

1. Escreva um programa para detectar objetos por conectividade, escreva inicialmente para que funcione com vizinhança 4 e com imagens binárias, se possível, incremente para que funcione com vizinhança 8 e imagens em tons de cinza. As informações sobre o método podem ser encontradas no livro. Você pode investigar os códigos das funções “neighborhood.sci” e “labelconnection.sci” para escrever o seu próprio programa.

O que eu fiz foi utilizar as funções existentes para a dectcção de algum objeto na imagem. Não fiz funções novas, embora tenha tentado vetoriza-las, porque as disponíveis funcionam para a aplicação e por você ter dito depois que no SIP já existem funções mais eficientes não sendo necessário vetoriza-las.
As imagens originais são:

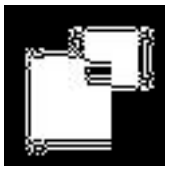
Binária

Em tons de cinza

É apresentado um script de como foi feita à detecção do objeto.

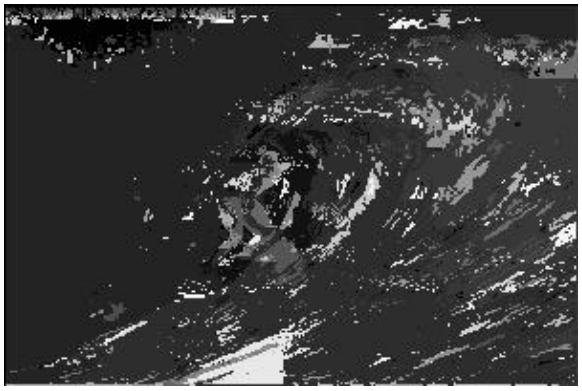
```
// Script
//
// leitura das imagens
B = imread('daniel_ex2_2/daniel_ex2_2_xorAB.jpg');
C = imread('surf1.jpg');
// Detecção em imagem binária
B1 = connection(B, [1 3], 4, 1);
```

```
B2 = connection(B, [30 3], 8, 1);
```



```
// Detecção em imagens em tons de cinza
C1 = labelconnection(C, 4, 64);
C1 = connection(C1, [100,200], 4, 8);
```

```
C2 = labelconnection(C, 8, 64);
```

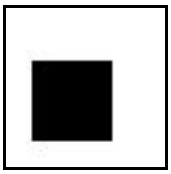


2. Utilizando os operadores lógicos para imagens, gere simulações de forma a obter as mesmas imagens e relações apresentadas na página 49 do livro texto. (obs.: você deverá gerar a imagem original utilizando zeros e uns)

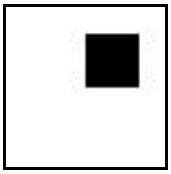
Como as operações no livro eram feitas com uma imagem binária em que o número 1 corresponde ao preto e o número 0 ao branco, e ao trabalhar com imagens em tons de cinza essa escala é invertida tive que mudar algumas operações do livro para ficar coerente com os resultados obtidos. A listagem do programa para realizar as operações está abaixo.

```
// Script
// comprimento dos quadrados gerados
a1 = 20; b1 = 30; c1 = 10;
a2 = 10; b2 = 20; c2 = 30;

d1 = 10; e1 = 30; f1 = 20;
d2 = 30; e2 = 20; f2 = 10;
// matrizes de intensidades correspondentes às imagens A e B da página 49 do livro
A = [ones(a1,(d1+e1+f1)); ones(b1,d1) zeros(b1,e1) ones(b1,f1); ones(c1,(d1+e1+f1))];
B = [ones(a2,(d2+e2+f2)); ones(b2,d2) zeros(b2,e2) ones(b2,f2); ones(c2,(d2+e2+f2))];
// operações lógicas
notA = 1*( ~A ); // not(A)
notB = 1*( ~B ); // not(B)
andAB = 1*( ~(notA&notB) ); // (A) and (B)
orAB = 1*( ~(notA|notB) ); // (A) or (B)
xorAB = 1*( ~xor(notA,notB) ); // (A) xor (B)
notAandB = 1*( ~(A&notB) ); // (~A) and (B)
// armazena imagens
imwrite(A, 'daniel_ex2_2_A.jpg');
```

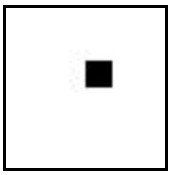


```
imwrite(B, 'daniel_ex2_2_B.jpg');
```

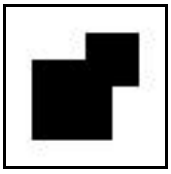


```
imwrite(notA, 'daniel_ex2_2_notA.jpg');
```

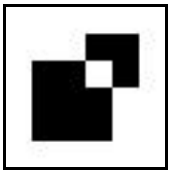
```
imwrite(andAB, 'daniel_ex2_2_andAB.jpg');
```



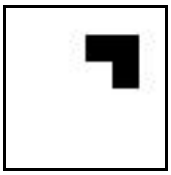
```
imwrite(orAB, 'daniel_ex2_2_orAB.jpg');
```



```
imwrite(xorAB, 'daniel_ex2_2_xorAB.jpg');
```



```
imwrite(notAandB, 'daniel_ex2_2_notAandB.jpg');
```



3. A seção 2.4 do livro texto (Image Geometry) apresenta a relação de transformações geométricas básicas sobre uma imagem.

    I. Escreva um programa que simule os efeitos de translação, rotação e escalonamento em uma imagem (um de cada vez).
    II. Aplique uma transformação reversa (se possível) para desenvolver as imagens do item anterior ao seu estado original.
    III. Gere transformações sucessivas em uma imagem (ex. translação + rotação, ou escalonamento + translação, etc) e tente revertê-la. Gere todas as combinações de transformações 2 a 2 possíveis e conclua quais são possíveis e conclua quais são possíveis de serem revertidas e quais não são. Apresente uma BREVE conclusão dos seus experimentos.

Imagem original.

Os programas escritos simulam os efeitos requisitados, mas os resultados obtidos têm perda de qualidade como a aparecimento de pontos pretos e perda de informação. Em breve, será possível utilizar algumas técnicas de melhoria nas imagens.

Nas transformações reversas:

- a translação inversa (TT) deixa a imagem original com uma borda preta;

- o escalonamento inverso (EE) não apresenta perdas desde que sempre o primeiro escalonamento seja maior que 1, senão sempre ocorrerá perda de dados;

- a rotação inversa (RR) tem o efeito de borda preta e o perdas com inserção de pontos pretos na imagem.

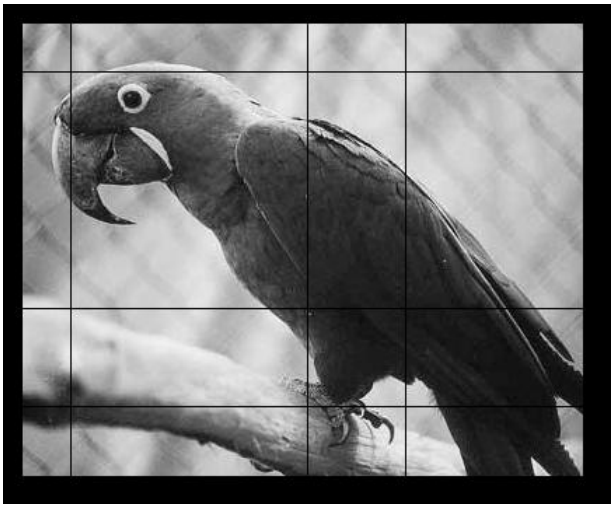
Quanto às transformações sucessivas ficam os seguintes comentários:

- Translação + Escalonamento (TE): a imagem obtida não tem perda de detalhes, mas ficou com alguns riscos pretos que se deve à compressão da imagem expandida anteriormente. Com algum tratamento é possível obter a imagem original.

- Translação + Rotação (TR): alguns pontos são perdidos nas rotações, mas é possível fazer algum tipo de interpolação para recuperar alguns detalhes da imagem original.

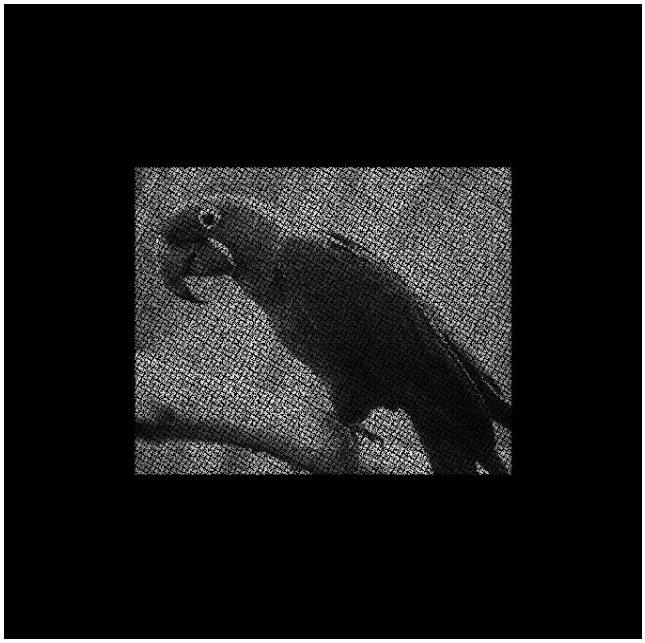
- Escalonamento + Rotação (ER): muitos pontos são perdidos depende das escalas usadas. Na imagem que obtive fica muito difícil recuperar o original, mas isso depende muito do ângulo de rotação e do fator de escalonamento.

- Escalonamento + Translação (ET): o resultado obtido foi parecido com a transformação “TE” com menos riscos pretos tendo a imagem um pouco mais de qualidade. Também seria possível recuperar a imagem.



- Rotação + Translação (RT): o resultado obtido foi parecido com da rotação + inversa isso porque na translação não há perda de dados embora faça muda a configuração de pontos perdidos. Também seria possível recuperar a imagem.

- Rotação + Escalonamento (RE): em comparação com a transformação “ER” a imagem obtida tem maior qualidade, mas ainda difícil recupera-la devido a grande quantidade de dados perdidos.



Os resultados variaram muito e têm que analisados caso à caso. Embora o livro adote uma seqüência para as transformações sucessivas isso é relativo no processamento de imagens, pois o que determina a qualidade do processamento é a aplicação. Esses resultados também não são definitivos visto que durante as transformações não foi usado nenhum processo de melhoria o que modificar bastante os resultados obtidos.

A listagem dos programas segue abaixo:

```
function T = translada(I, x0, y0)
// TRASNSLADA_
// I: matriz de intensidade
// x0: deslocamento vertical
// y0: deslocamento horizontal
//
// Uso:
// I = imread('figura.jpg');
// T = translada(I, -20, 20);
// imshow(T, [])
//
[m n] = size(I);
//for i=1:m
//  for j=1:n
//    a(i,j) = i+x0;
//    b(i,j) = j+y0;
//  end
//end
// código vetorizado referente aos 'for'
x = [1:m]'*ones(1,n)+x0;
y = ones(m,1)*[1:n]+y0;
// arredonda para valores inteiros
x = round(x);
y = round(y);
// todos os valores no primeiro quadrante
if x0 < 0 then
  x = x - min(x) + 1;
end

if y0 < 0 then
  y = y - min(y) + 1;
end
// tamanho da nova matriz
mC = m + (x0/abs(x0))*x0;
nC = n + (y0/abs(y0))*y0 ;
// vetor com os novos indíces
//C = sub2ind([mC nC], x(:), y(:));
C = x(:) + mC*(y(:)-1);
//
clear x
clear y
//matriz com a imagem transladada
T = zeros(mC,nC);
T(C) = I(:);

endfunction
```

```
function E = escalona(I, ex, ey)
// ESCALONA_
// I: matriz de intensidade
// ex: escalonamento vertical
// ey: escalonamento horizontal
//
// Uso:
// I = imread('figura.jpg');
// E = escalona(I, 0.5, 2);
// imshow(E, [])
//
[m n] = size(I);
//for i=1:m
//  for j=1:n
//    a(i,j) = i*ex;
//    b(i,j) = j*ey;
//  end
//end
// código vetorizado referente aos 'for'
x = [1:m]'*ones(1,n)*ex;
y = ones(m,1)*[1:n]*ey;
// arredonda para valores inteiros
x = round(x);
y = round(y);
// tamanho da nova matriz
mC = max(x);
nC = max(y);
// vetor com os novos indíces
//C = sub2ind([mC nC], x(:), y(:));
C = x(:) + mC*(y(:)-1);
//
clear x
clear y
//matriz com a imagem escalonada
E = zeros(mC,nC);
E(C) = I(:);

endfunction

function R = rotaciona(I, teta)
// ROTACIONA_
// I: matriz de intensidade
// teta: angulo de rotação em graus
//
// Uso:
// I = imread('figura.jpg');
// R = rotaciona(I, 60);
// imshow(R, [])
//
teta = teta*%pi/180; // converte angulo para radianos
//
[m, n] = size(I);
//for i=1:3
//  for j=1:4
//    x(i,j) = cos(teta)*i + sin(teta)*j;
//    y(i,j) = cos(teta)*j - sin(teta)*i;
```

```
//  end
//end
// código vetorizado referente aos 'for'
x  =  cos(teta)*[1:m]'*ones(1,n)  +  sin(teta)*ones(m,1)*[1:n];
//subscrito de linhas
y  =  cos(teta)*ones(m,1)*[1:n]  -  sin(teta)*[1:m]'*ones(1,n);
//subscrito de colunas
//
minX = min(x); kX = minX / abs(minX);
minY = min(y); kY = minY / abs(minY);
// todos os valores no primeiro quadrante
x = round( -minX + x +1 );
y = round( -minY + y +1 );
// tamanho da nova matriz
mC = max(x);
nC = max(y);
// vetor com os novos indíces
//C = sub2ind([mC nC], x(:), y(:));
C = x(:) + mC*(y(:)-1);
//
clear x
clear y
//matriz com a imagem rotacionada
R = zeros(mC,nC);
R(C) = I(:);

endfunction

// Script utilizado para fazer as operações nos itens II e III.
//
// leitura da imagem
I = imread('ararauna.jpg');
// translação + inversa
TT = translada(I,-20,20);
TT = translada(TT,20,-20);
imwrite(TT/256, 'daniel_ex2_3_TT.jpg')
clear TT
// escalonamento + inversa
EE = escalona(I,2,2);
EE = escalona(EE,0.5,0.5);
imwrite(EE/256, 'daniel_ex2_3_EE.jpg')
clear EE
// rotação + inversa
RR = rotaciona(I,60);
RR = rotaciona(RR,-60);
imwrite(RR/256, 'daniel_ex2_3_RR.jpg')
clear RR
// translação + escalonamento
TE = translada(I,20,20);
TE = escalona(TE,sqrt(2),sqrt(2));
TE = translada(TE,-20,-20);
TE = escalona(TE,1/sqrt(2),1/sqrt(2));
imwrite(TE/256, 'daniel_ex2_3_TE.jpg')
clear TE
// translação + rotação
TR = translada(I,20,20);
```

```
TR = rotaciona(TR,60);
TR = translada(TR,-20,-20);
TR = rotaciona(TR,-60);
imwrite(TR/256, 'daniel_ex2_3_TR.jpg')
clear TR
// escalonamento + translação
ET = escalona(I,sqrt(2),sqrt(2));
ET = translada(ET,20,20);
ET = escalona(ET,1/sqrt(2),1/sqrt(2));
ET = translada(ET,-20,-20);
imwrite(ET/256, 'daniel_ex2_3_ET.jpg')
clear ET
// escalonamento + rotação
ER = escalona(I,sqrt(2),sqrt(2));
ER = rotaciona(ER,60);
ER = escalona(ER,1/sqrt(2),1/sqrt(2));
ER = rotaciona(ER,-60);
imwrite(ER/256, 'daniel_ex2_3_ER.jpg')
clear ER
// rotação + translação
RT = rotaciona(I,60);
RT = translada(RT,20,20);
RT = rotaciona(RT,-60);
RT = translada(RT,-20,-20);
imwrite(RT/256, 'daniel_ex2_3_RT.jpg')
clear RT
// rotação + escalonamento
RE = rotaciona(I,60);
RE = escalona(RE,1.2,1.2);
RE = rotaciona(RE,-60);
RE = escalona(RE,1/1.2,1/1.2);
imwrite(RE/256, 'daniel_ex2_3_RE.jpg')
clear RE
//
```