1. Funções Implementadas e corpo principal do programa

1.1 Main

close all;

filename = 'titan'; %Imagem 1

img = imread(strcat(filename,'.jpg'));

[sx,sy,ch] = size(img);

fac = 2;

nn\_img = resize\_nn(sx,sy,fac,img); %Interpolação Nearest Neighbor

bilinear\_img = resize\_bilinear(sx,sy,fac,img); %Interpolação Bilinear

cubic\_img = resize\_bicubic(sx,sy,fac,img); %Interpolação Bicúbica

imwrite(nn\_img, strcat(filename,'\_nn.jpg'));

imwrite(bilinear\_img,strcat(filename,'\_bilinear.jpg'));

imwrite(cubic\_img,strcat(filename,'\_cubic.jpg'));

filename = 'passaro'; %Imagem 2

img = imread(strcat(filename,'.jpg'));

nn\_img = resize\_nn(sx,sy,fac,img); %Interpolação Nearest Neighbor

bilinear\_img = resize\_bilinear(sx,sy,fac,img); %Interpolação Bilinear

cubic\_img = resize\_bicubic(sx,sy,fac,img); %Interpolação Bicúbica

imwrite(nn\_img, strcat(filename,'\_nn.jpg'));

imwrite(bilinear\_img,strcat(filename,'\_bilinear.jpg'));

imwrite(cubic\_img,strcat(filename,'\_cubic.jpg'));

1.2 resize\_nn

function nn\_img = resize\_nn(sx,sy,fac,img)

nn\_img = uint8(zeros(size(img).\* [fac,fac,1]));

for x = 1:sx

for y = 1:sy

nn\_img(fac\*x,fac\*y-1,:) = img(x,y,:);

nn\_img(fac\*x-1,fac\*y,:) = img(x,y,:);

nn\_img(fac\*x,fac\*y,:) = img(x,y,:);

nn\_img(fac\*x-1,fac\*y-1,:) = img(x,y,:);

end

end

end

1.3 resize\_bilinear

function [bilinear\_img] = resize\_bilinear(sx,sy,fac,img)

bilinear\_img = uint8(zeros(size(img).\* [fac,fac,1]));

for i=1:sx

for j=1:sy

bilinear\_img(1+(i-1)\*fac,1+(j-1)\*fac,:)=img(i,j,:);

end

end

for i=1:1+(sx-2)\*fac

for j=1:1+(sy-2)\*fac

if not(((rem(i-1,fac)==0) && (rem(j-1,fac)==0)) )

%calcula as coordenadas a serem interpoladas

inx = ceil(i/fac);

iny = ceil(j/fac);

interp00=double(bilinear\_img( inx\*fac-1,iny\*fac-1,:));

interp10=double(bilinear\_img( inx\*fac+1,iny\*fac-1,:));

interp01=double(bilinear\_img( inx\*fac-1,iny\*fac+1,:));

interp11=double(bilinear\_img( inx\*fac+1,iny\*fac+1,:));

x=rem(i-1,fac); %coordenadas do pixel interpolado

y=rem(j-1,fac);

dx=x/fac; %

dy=y/fac;

c1=interp00; %constantes da interpolação

c2=interp10-interp00;

c3=interp01-interp00;

c4=interp00-interp10-interp01+interp11;

bilinear\_img(i,j,:)=c1+c2\*dx+c3\*dy+c4\*dx\*dy; %pixel resultante

end

end

end

end

1.4 resize\_cubic

function cubic\_img = resize\_bicubic(sx,sy,fac,img)

cubic\_img = uint8(zeros(size(img).\* [fac,fac,1]));

for i=1:(sx\*fac)-6

for j=1:(sy\*fac)-6

inx = ceil(i/fac);

iny = ceil(j/fac);

interp = double(zeros(4,4,3));

for polx = -1:2

for poly = -1:2

interp(polx+2,poly+2,:) = double(img((inx+polx)+1,(iny+poly)+1,:));

end

end

for channel = 1:3

cubic\_img(i,j,channel) = BicubicInterpolate(interp(:,:,channel),1,1);

end

end

end

end

1.4.1 BicubicInterpolate

function [output] = BicubicInterpolate(points, coefx, coefy)

x1 = CubicInterpolate( points(1,1), points(2,1), points(3,1), points(4,1), coefx );

x2 = CubicInterpolate( points(1,2), points(2,2), points(3,2), points(4,2), coefx );

x3 = CubicInterpolate( points(1,3), points(2,3), points(3,3), points(4,3), coefx );

x4 = CubicInterpolate( points(1,4), points(2,4), points(3,4), points(4,4), coefx );

output = CubicInterpolate( x1, x2, x3, x4, coefy );

end

1.4.2 CubicInterpolate

function [output] = CubicInterpolate(v0, v1, v2, v3, coef )

A = (v3-v2)-(v0-v1);

B = (v0-v1)-A;

C = v2-v0;

D = v1;

output = D + coef \* (C + coef \* (B + coef \* A));

end

2. Imagens escolhidas

Escolheu-se as seguintes imagens para exemplificação dos métodos de interpolação:

Foto de um pássaro:

Foto do personagem Titã do jogo Final Fantasy XIV



3. Resultados obtidos

4. Discussão dos resultados

Pôde se perceber nas imagens interpoladas pelo método bilinear de primeira ordem o borramento dos detalhes das imagens e a forte suavização das bordas serrilhadas presentes anteriormente. Houve uma leve perda de detalhes na coloração da imagem.

Já no método NN(Nearest Neighbour), percebe-se a distorção nas bordas e o aparecimento mais forte de serrilhamento. Houve, comparado com os demais métodos de interpolação, uma perda maior de detalhes e da diversidade de coloração. Há inclusive um efeito aparente da formação de quadrados de mesma cor, resultado esperado de um método que copia pixeis para os espaços vizinhos.

O método bicúbico apresentou um resultado que equilibra suavização de bordas e manutenção dos detalhes e da coloração da imagem. Teve um certo grau de serrilhamento das bordas, mas não tão grave quanto a interpolação Nn e sem o borramento/perda de detalhes da interpolação bilinear.

5. Conclusões

Em termos de resultados, a interpolação bicúbica apresenta o melhor desempenho dos três métodos. Porém isso veio ao custo de um tempo significamente maior de execução. As interpolações NN e bilinear foram muito mais rápidas, e o resultado obtido por elas não foi significamente pior numa breve observação. Isto é, em termos de custo-benefício elas apresentarem desempenho satisfatório em tempo razoável de execução. O borramento do método bilinear, contudo, pode ser mais catastrófico dependendo da imagem e da importância da preservação dos detalhes dela.