//**Convolution**

img=imread(‘bird.jpg’);

imshow(img);

title(‘original image’);

g=fspecial(‘gaussian’,[5,5],2);

ig=imfilter(img,g,’same’);

pause(2);

figure,imshow(ig);

title(‘gaussian blur image’);

sharp=[0 0 0 0 0;0 0 -1 0 0;0 -1 5 -1 0;0 0 -1 0 0;0 0 0 0 0];

sharpimg=convn(img,sharp);

pause(2);

figure,imshow(sharpimg);

title(‘sharpen kernel’);

edge=[0 0 0 0 0;0 0 1 0 0;0 1 -4 1 0;0 0 1 0 0;0 0 0 0 0];

edgeimg=convn(img,edge);

pause(2);

figure,imshow(edgeimg);

title(‘edge detection’);

horizontal=[0 0 0 0 0;0 -1 -2 -1 0;0 0 0 0 0;0 1 2 1 0;0 0 0 0 0];

horizontalimg=convn(img,horizontal);

pause(2);

figure,imshow(horizontalimg);

title(‘horizontal edge detection’);

vertical=[0 0 0 0 0;0 -1 0 1 0;0 -2 0 2 0;0 -1 0 1 0 ;0 0 0 0 0];

verticalimg=convn(img,vertical);

pause(2);

figure,imshow(verticalimg);

title(‘vertical edge detection’);

//**2D Convolution**

A=imread(‘blur.jpg’);

pause(2);

figure,imshow(A);

A=rgb2gray(A);

B=[0 0 0 0 0 ;0 -1 -2 -1 0;0 0 0 0 0;0 1 2 1 0;0 0 0 0 0];

C=conv2((A),(B),’same’);

pause(2);

figure,imshow(C);

A=rgb2gray(A);

B=[0 0 0 0 0 ;0 -1 -2 -1 0;0 0 0 0 0;0 1 2 1 0;0 0 0 0 0];

C=conv2((A),(B),’same’);

pause(2);

figure,imshow(C);

//**Algorithm for generating PSF**

%psf size

pf=[0 0 0 0 0;0 0 0 0 0;0 0 1 0 0;0 0 0 0 0;0 0 0 0 0];

[n,n]=size(pf);

%generate random values

r=rand((n+1)/2);

disp(r);

%create a vector

[p,q]=size(r);

vec=[];

for i=1:p

for j=1:q

vec=[vec,r(ij)];

end

end

disp(vec);

%sort values in vector

vecsize=size(vec);

disp(vecsize); bound=vecsize(2);

for i=1:bound-1

for j=1:bound- 1

if(vec(j)>vec(j+1))

temp=vec(j);

vec(j)=vec(j+1);

vec(j+1)=,temp;

end

end

end

disp(vec);

%flip values from left to right

flipvec=fliplr(vec);

disp(flipvec);

trans=transpose(flipvec);

disp(trans);

%convolution of flipped vector and transpose

D=conv2((flipvec),(trans),’same’);

disp(d);

%normalize result

nor=norm(d);

disp(nor);

// **EVOLUTIONARY ALGORITHM:**   
img=imread(‘bird.jpg’);

figure,imshow(img);

g=fspecial('gaussian’,[5 5],2000);

ig=imfilter(img,g,’same’);

figure,imshow(ig);

title('gaussian blur image’);

mse = mean(mean(im2double(ig) - im2double(img)).^2, 1), 2);

psnr = 10\* log 10(1/mean(mse,3));

disp(psnr);

%psf size

pf=[0 0 0 0 0;0 0 0 0 0;0 0 1 0 0;0 0 0 0 0;0 0 0 0 0];

[n,n]=size(pf);

psnr3=[];

nor=[];

nor1=[];

for 1=1:10

for k=1:10

%generate random values

r=rand((n+1)/2);

disp(r);

%create a vector

[p,q]=size(r);

vec=[];

for i= 1:p

for j= 1 :q

vec=[vec,r(i,j)];

end

end

%sort values in vector

vecsize=size(vec);

disp(vecsize);

bound=vecsize(2);

for i=1 :bound- 1

for j= 1 :bound-1

if (vec(j)>vec(j+ 1 ))

temp=vec(j);

vec(j)=vec(j+1);

vec(j+1 )=temp;

end

end

end

disp(vec);

%flip values from left to right

flipvec=fliplr(vec);

disp(flipvec);

trans=transpose(flipvec);

%convoltion of flipped vector and transpose

d=conv2((flipvec),(trans), ‘same’);

%normalize result

nor1(k)=norm(d);

if(l==1)

op=ig+nor1(k);

else

op1=op2;

op=op1 +nor1(k);

end

mse3 = mean(mean((im2double(op) - im2double(img)).^2, 1), 2);

psnr3(k)= 10 \* log10(1/ mean(mse3,3));

nor(k)=nor1(k);

end

for i=1:9

for j=1:9

if(psnr3(j)>psnr3(j+1))

temp=psnr3(j);

temp1=nor(j);

psnr3(j)=psnr3(j+1);

nor(j)=nor(j+1);

psnr3(j+1)=temp;

nor(j+1)=temp1;

end

end

end

disp(psnr3);

pd(1)=psnr3(10);

md(1)=nor(10);

if(l==1)

op2=ig+nor(10);

else

op2=op1+nor(10);

end

figure.imishow(op2);

end

mse4 = mean(mean((im2double(op2) – im2double(ig)).^2, 1), 2);

psnr4= 10 \* log10(1/mean(mse4,3));

disp(psnr4);

if(psnr4>psnr)

per1 = ((psnr4-psnr)/psnr)\*100;

else

per1 = ((psnr4-psnr)/psnr)\*100;

end

disp(per1);

figure,imshow(op2);

title(' restored image’);