ANEXOS

ANEXO 1

CÓDIGO DETALLADO DE LA FUNCIÓN POCS:

```
function [ y,E ] = funcion_pocs( im,im_ref,X,registro,k,iteraciones )
%[ y,E ] = FUNCION_POCS(im,im_ref,X,registro,k,iteraciones)
%  Realiza el proceso de Super-Resolución de un banco de imágenes
%  especificado, mediante el algoritmo POCS (Projections Onto Convex
Sets)
%  devuelve, adicionalmente, el Error por iteración.
% Parámetros de entrada:
%   im -> Arreglo tipo celda que contiene las 'n' imágenes
%   im_ref -> Imagen de referencia para el error , de no tener una,
colocar 0
%   X  -> Imagen de referencia inicial sobre el cual inicia el algoritmo
%   registro -> Matriz de n X 2, 'n' es el número de imágenes que tiene
'im'
%   k -> Factor de magnificación de la imagen deseada
%   iteraciones -> Número de veces que se desea realizar las proyecciones
% Parámetros de salida:
%   y -> Imagen deseada de alta resolución
%   E -> Error por iteración

%Parámetros de inicialización
psf=fspecial('gauss',5,1);
alfa=0;
beta=255;
phi=1;

%Se sobre-muestrean las imágenes de baja resolución (arreglo 'im')
%según el factor de magnificación 'k'
ISA=sobre_muestrear_imagenes(im,k);

%Se determina la imagen inicial (con la cual parte el algoritmo) en base
%a la especificada por el usuario en 'X'
if(size(X)~=size(ISA{1}))
    X=zeros(size(ISA{1}));
    for i=length(im):-1:1
        coordenadas=find(ISA{i}>-1);
        X(coordenadas)=ISA{i}(coordenadas);
    end
end
X(X==-1) = 0;

%Inicialización del arreglo de imágenes simuladas
E=zeros(size(1,iteraciones));
imagen_simulada=cell(size(ISA));
dim1=size(X);

%Asignación de la imagen de referencia para el cálculo del error por
%iteración
if(size(X)~=size(im_ref))
    bandera=1;
```

```
    im_ref=X;
else
    bandera=0;
end

%Inicio de las iteraciones del algoritmo
for n=1:iteraciones

    %Generación de imágenes simuladas a partir de la imagen de alta
    %resolución
    for i=1:length(imagen_simulada)
        imagen_simulada{i}=desplazamiento_subpixel(X,-k*registro(i,2),-
k*registro(i,1),0);
        imagen_simulada{i}=imfilter(imagen_simulada{i},psf);
        imagen_simulada{i}=padarray(imagen_simulada{i},[4 4],0,'both');
    end

    %Por cada pixel de cada imagen de baja resolución se realiza la
    %proyección en base al cálculo del residuo
    for LR=1:length(im)

            for i=1:dim1(1)
                for j=1:dim1(2)
                    if(ISA{LR}(i,j)==-1)
                        continue
                    end

                        r=ISA{LR}(i,j)-imagen_simulada{LR}(i+4,j+4);
                        if(r>phi)
                            imagen_simulada{LR}(i-2+4:i+2+4,j-
2+4:j+2+4)=imagen_simulada{LR}(i-2+4:i+2+4,j-2+4:j+2+4)+(r-
phi).*psf./(sum(sum(psf.*psf)));

                        elseif(r<-phi)
                            imagen_simulada{LR}(i-2+4:i+2+4,j-
2+4:j+2+4)=imagen_simulada{LR}(i-2+4:i+2+4,j-
2+4:j+2+4)+(r+phi).*psf./(sum(sum(psf.*psf)));
                        end
                end
            end
            imagen_simulada{LR}= imagen_simulada{LR}(5:end-4,5:end-4);
    end

    %Se realiza el proceso de compensación o ensamblaje de la imagen de
    %alta resolución a partir de las imágenes simuladas actualizadas por
POCS.

    imagen_simulada=alinear_imagenes(imagen_simulada,k*registro);
for i=length(imagen_simulada):-1:1
    coordenadas=find(imagen_simulada{i}~=-1);
    X(coordenadas)=imagen_simulada{i}(coordenadas);
end

    %Aplicación de la segunda restricción (amplitud)
    X(X<alfa) = alfa;
```

```
    X(X>beta) = beta;

    %Cálculo del porcentaje de error por iteración
    dim2=size(X);
    error= norm(reshape(im_ref,dim2(1)*dim2(2),1)-
reshape(X,dim2(1)*dim2(2),1))./norm(reshape(im_ref,dim2(1)*dim2(2),1));
    E(n)= error;
    if(bandera==1)
        im_ref=X;
    end

    %Se muestra el proceso de reconstrucción de forma interactiva
    figure(n)
    imshow(uint8(X))
    drawnow
end

y=X;
end
```

ANEXO 2

CÓDIGO DE FUNCIONES AUXILIARES.

Función que extrae imágenes desde un directorio y les da formato.

```
function [ y ] = extraer_imagenes( x )
%Y=ETRAER_IMAGENES(X)
% Recibe como parámetro de entrada la ubicación (carpeta) de las imágenes
correspondientes a las tomas de una escena
% El parámetro de salida de esta función es una arreglo tipo celda (cell)
% que contiene todas las imágenes encontradas en la carpeta especificada,
% adicionalmente, a cada imagen del arreglo de celda se le ha dado el
% formato de escala de grises (si es que estaba en RGB) y tipo de dato
doble precisión (double) para
% que pueda ser utilizada por funciones que lo requieran .
%
% X-> Ubicación de la carpeta que contiene las imágenes a usar
% Y-> Arreglo tipo celda que contiene todas las imágenes encontradas en
la carpeta especificada


directorio = dir(x);
archivos=cell(size(dir(x)));
imagenes=cell(size(dir(x)));
j=1;

for i=1:length(dir(x))

    archivos{i}=directorio(i).name;
    [a ,b ,c]=fileparts(directorio(i).name);


if(strcmp(c,'.bmp')||strcmp(c,'.tiff')||strcmp(c,'.jpeg')||strcmp(c,'.jpg
'))
        imagenes{j}=archivos{i};
        j=j+1;
    end

end

imagenes=imagenes(1:j-1);

% imagenes
% archivos
for i=1:length(imagenes)

    temp=strcat(x,'\');
    temp=strcat(x,imagenes{i});
    temp=imread(temp);
    if(length(size(temp))==3)
        temp=rgb2gray(temp);
    end
```

```
    temp=double(temp);
    imagenes{i}=temp;
end

    y=imagenes;
end
```

Función que calcula del desplazamiento sub-pixel entre imágenes:

```
function [ registro] = registro_imagenes( im,k )
%REGISTRO_IMAGENES(im,k)
%Calcula el registro de un arreglo especificado de imágenes
%Parámetros de entrada:
%   im -> Arreglo tipo celda que contiene 'n' imágenes
%   k -> Factor de magnificación de la imagen deseada
%Parámetros de salida:
%   registro -> Matriz de n X 2, 'n' es el número de imágenes que tiene
'im'
registro=zeros(length(im),2);

for i=1:length(im)
    temp=dftregistration(fft2(im{1}),fft2(im{i}),k);
    registro(i,:)=temp(end-1:1:end);
end


end
```

Función que desplaza imágenes a nivel sub-pixel:

```
function [ out ] = desplazamiento_subpixel( im,deltax,deltay,val )
%Función que deslaza imágenes a nivel sub-pixel
%Parámetros de entrada:
%   im -> Imagen que se desea desplazar
%   deltax -> Desplazamiento en el eje x
%   deltay -> Desplazamiento en el eje y
%   val -> Valor de relleno para los pixeles "muertos"
%Parámetros de entrada:
%   out -> Imagen desplazada a nivel sub-pixel

[M ,N]=size(im);
[yy ,xx]=meshgrid(1:N,1:M);
out=interp2(yy,xx,im,yy-deltay,xx-deltax,'cubic',val);

end
```

Función que genera imágenes degradadas.

```
function [s] = generar_imagenes_LR(im,delta,k,n)

psf=fspecial('gauss',5,1); %Generación de psf
im2=cell(size(1,n));

im=resample(im,2,1)'; % Sobre-muestreo
im=resample(im,2,1)';

for i=1:n
    im2{i} = shift(im,-delta(i,2)*2*k,-delta(i,1)*2*k); % Desplazar
imágenes
end

%Proceso de degradación de una imagen digital de alta resolución

for i=1:n
    im2{i} = lowpass(im2{i},[0.12 0.12]);
    im2{i}=imfilter(im2{i},psf);
    im2{i} = downsample(im2{i},2*k)';
    im2{i} = downsample(im2{i},2*k)';
end
s=im2;
end
```

Función que calcula el error cuadrático medio:

```
function [ y ] = mse( f,g )
%MSE(f,g)
%Calcula el error cuadrático medio entre dos imágenes
%Parámetros de entrada:
%   f -> Imagen de restaurada
%   g -> Imagen de referencia
%Parámetros de salida:
%   y -> ECM

dim=size(f);

y = (1/(dim(1)*dim(2)))*sum(sum((f-g).^2));
end
```

Función que calcula el PSNR:

```
function [ PSNR ] = psnr( X,im_ref)
%PSNR(X,im_ref)
%Calcula el PSNR entre dos imágenes
%Parámetros de entrada:
%   X -> Imagen de restaurada
%   im_ref -> Imagen de referencia
%Parámetros de salida:
%   PSNR -> PSNR
m=size(X);
n=m(2);
```

```
m=m(1);
MSE = (1/(m*n))*sum(sum((im_ref-X).^2));
disp(MSE);
PSNR = 20*log10((max(max(im_ref)))/((MSE)^0.5));
% dim=size(X);
% N=dim(1).*dim(2);
% num=(max(max(im_ref))^2)*N;
% den= norm(reshape(X,dim(1)*dim(2),1)-reshape(im_ref,dim(1)*dim(2),1));
% den=den.*den;
% PSNR=10*log10(num/den);

end
```

Código de pruebas sobre Lena

```
% Pruebas sobre Lena
close all
clear all
clc

k=2;
iteraciones=30;
psf=fspecial('gauss',5,1);
lena=double(imread('lena.png'));
registro=[0 0 ;-10.5 -0.5 ;10.5 0.5;14.5 12.5;0 -0.5;13.5 -0.5;-11.5 -
0.5;7.5 14.5];
registro=[registro ;registro];
registro=registro(1:k^2,:);

im=generar_imagenes_LR(lena,registro,k,k.^2);

[f1,E1]=funcion_pocs(im,lena,zeros(size(lena)),registro,k,iteraciones);

y1=interpolacion(im,registro,0*registro,k,'linear');
y2=interpolacion(im,registro,0*registro,k,'cubic');
close all
clc

figure(1)
for i=1:length(im)
    subplot(1,length(im),i);
    imshow(uint8(im{i}));
end

figure(2)
imshow(uint8(f1));
% figure(3)
% imshow(uint8(f2));
figure(4)
imshow(uint8(y1));
figure(5)
imshow(uint8(y2));
figure(6)
imshow(uint8(lena));
figure(7)
```

```
hold on
plot(1:iteraciones,100*E1,'b-o')
% plot(1:iteraciones,100*E2,'r--*')
axis([1 iteraciones 0 inf])
legend('imagen inicial cero','imagen inicial distinta a
cero','Location','northeast')
title('Curva de porcentaje de error por iteración')
xlabel('Iteracion')
ylabel('Porcentaje de error')
figure(8)
hold on
title('Histograma de imagen de referencia (Lena)')
xlabel('Número de intensidad')
ylabel('Cantidad de pixeles')
imhist(uint8(lena))
```

Pruebas de PSNR realizadas sobre imágenes de campos de cultivo:

```
%Pruebas PSNR en campos de cultivo
close all
clear all
clc

k=2;
iteraciones=30;
psf=fspecial('gauss',5,1);
carpeta1='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_1\';
carpeta2='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_2\';
carpeta3='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_3\';
carpeta4='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_4\';

im1=extraer_imagenes(carpeta1);
im2=extraer_imagenes(carpeta2);
im3=extraer_imagenes(carpeta3);
im4=extraer_imagenes(carpeta4);
registro1=registro_imagenes(im1,k);
registro2=registro_imagenes(im2,k);
registro3=registro_imagenes(im3,k);
registro4=registro_imagenes(im4,k);
imagen=im1{1};
registro=registro1;

im=generar_imagenes_LR(imagen,registro,k,k.^2);

[f1,E1]=funcion_pocs(im,imagen,zeros(size(imagen)),registro,k,iteraciones
);

y1=interpolacion(im,registro,0*registro,k,'linear');
[y1blind,psf_rest1]=deconvblind(y1,psf);
[y1lucy]=deconvlucy(y1,psf);
[y1reg]=deconvreg(y1,psf);

y2=interpolacion(im,registro,0*registro,k,'cubic');
[y2blind,psf_rest2]=deconvblind(y2,psf);
```

```
[y2lucy]=deconvlucy(y2,psf);
[y2reg]=deconvreg(y2,psf);

close all
clc

figure(1)
for i=1:length(im)
    subplot(1,length(im),i);
    imshow(uint8(im{i}));
end
figure(2)
imshow(uint8(f1));
figure(3)
imshow(uint8(y1));
figure(4)
imshow(uint8(y2));
figure(5)
imshow(uint8(imagen));
figure(6)
hold on
plot(1:iteraciones,100*E1,'b-o')
% plot(1:iteraciones,100*E2,'r--*')
axis([1 iteraciones 0 inf])
% legend('imagen inicial cero','imagen inicial distinta a
cero','Location','northeast')
title('Curva de porcentaje de error por iteración')
xlabel('Iteracion')
ylabel('Porcentaje de error')
figure(7)
hold on
title('Histograma de imagen de referencia')
xlabel('Número de intensidad')
ylabel('Cantidad de pixeles')
imhist(uint8(imagen))
figure(8)
hold on
title('Histograma de imagen restaurada')
xlabel('Número de intensidad')
ylabel('Cantidad de pixeles')
imhist(uint8(f1))
```

Código de pruebas en imágenes de campos de cultivo:

```
%Este archivo .M realiza la aplicación del algoritmo POCS a las imágenes
de
%campos de cultivo, en cada carpeta se debe especificar la ruta donde se
%encuentran las imágenes de cada escena (imágenes de baja resolución)

close all
clear all
clc

%Inicialización de parámetros generales
k=2;
phi=1;
iteraciones=30;

%Ubicación de las imágenes de una misma escena
carpeta1='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_1\';
carpeta2='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_2\';
carpeta3='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_3\';
carpeta4='E:\Daniel\Tesis DP\TESIS\Pruebas POCS\Campo_cultivo_4\';

%Bloque de extracción de imágenes y formateo de las mismas
im1=extraer_imagenes(carpeta1);
im2=extraer_imagenes(carpeta2);
im3=extraer_imagenes(carpeta3);
im4=extraer_imagenes(carpeta4);
%Cálculo del registro de las imágenes
registro1=registro_imagenes(im1,k);
registro2=registro_imagenes(im2,k);
registro3=registro_imagenes(im3,k);
registro4=registro_imagenes(im4,k);

%Aplicación de POCS a cada una de las escenas que se desea magnificar
[ y1,E1 ] = funcion_pocs(
im1,0,zeros(k*size(im1{1})),registro1,k,iteraciones );
[ y2,E2 ] = funcion_pocs(
im2,0,zeros(k*size(im2{1})),registro2,k,iteraciones );
[ y3,E3 ] = funcion_pocs(
im3,0,zeros(k*size(im3{1})),registro3,k,iteraciones );
[ y4,E4 ] = funcion_pocs(
im4,0,zeros(k*size(im4{1})),registro4,k,iteraciones );
```

ANEXO 3

POCS APLICADO A CAMPOS DE CULTIVO

Imagen de campo de cultivo.

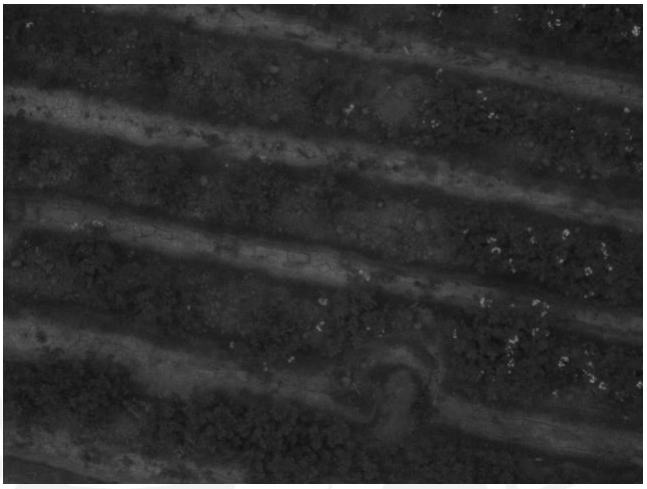

Imagen magnificada con POCS.

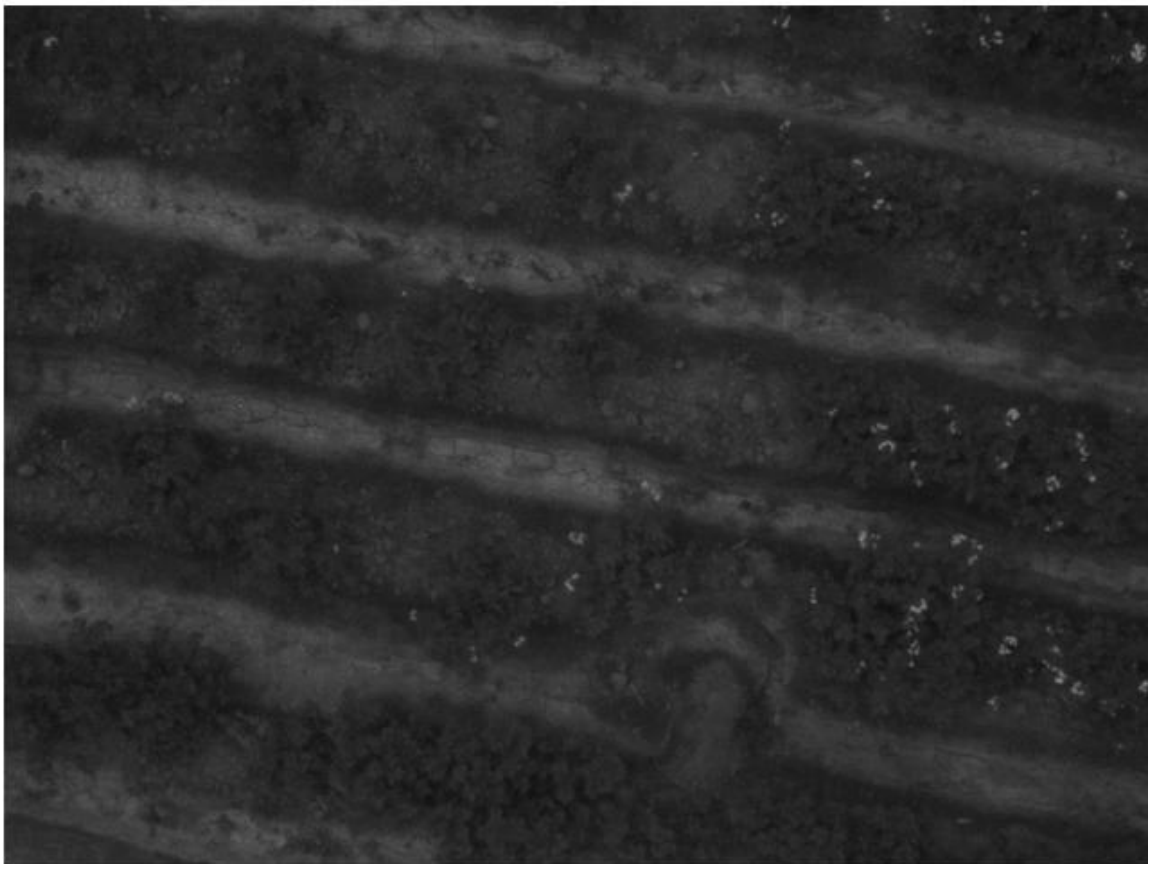

Imagen de campo de cultivo.

Imagen magnificada con POCS.