CÓDIGOS

SISTEMAS DE PERCEPCIÓN

@iNanoxTM

Contenido

BLOQUE 2. TRATAMIENTO DE IMÁGENES

1. SUAVIZADO

```
f = imread ('cameraman.tif');

% Ejemplo de suavizado
h = double(f);
[M,N] = size(f);

for i=2:M-1
    for j=2:N-1
    g( i,j) = uint8( (...
    h( i-1, j-1)+ h( i-1, j)+ h( i-1, j+1) + ...
    h( i , j-1)+ h( i , j)+ h( i , j+1) + ...
    h( i+1, j-1)+ h( i+1, j)+ h( i+1, j+1) )/9 );
    end
end
imshow(g)

%También
gMedia = zeros(size(f));

for i=2:M-1
    for j=2:N-1
        gMedia(i,j)=1/9*sum(sum(h(i-1:i+1,j-1:j+1)));
    end
end

imshow([g uint8(gMedia)]);

% Mediana

gMediana = zeros(size(f));

% Máscara 3x3
for i = 2:M-1
    for j = 2:N-1
        gMediana(i,j) = median(reshape(h(i-1:i+1,j-1:j+1),9,1));
    end
end

%Máscara 5x5
gMediana5 = zeros(size(f));

for i = 3:M-2
    for j = 3:N-2
        gMediana5(i,j) = median(reshape(h(i-2:i+2,j-2:j+2),25,1));
    end
end

imshow([uint8(gMediana) uint8(gMediana5)]);
```

2. CAMBIO DE BRILLO

```
f = imread ('tire.tif');

h = double(f);
g = uint8 (h + 100);
close all; figure;imshow([f,g])

%Lo que ocurre en el histograma es que se desplzada hacia la derecha. Si
%sobrepasa el limite de 255 (uint) todos los puntos se quedan en 255.Todos
%los puntos que pasen pierden información
```

3. CAMBIO DE CONTRASTE

```
%Le quito brilloa  los que menos intensidad luminosa tiene y le aumento
%brillo a los que mas intensidad tienen.
f = imread ('pout.tif'); %Le quito brilloa  los que menos intensidad luminosa
tiene y le aumento
%brillo a los que mas intensidad tienen.
f = imread ('pout.tif');
[M,N] = size(f);
for i=1:M
    for j=1:N
    orig = double(f(i,j));
    resu = 255*(1/(1+exp(-0.045*(orig-127))));
    g(i,j) = uint8(resu);
    end
end
%Visualizacion antes y despues del cambio
close all; figure;imshow([f,g])
figure
subplot(2,1,1);imhist(f);subplot(2,1,2);imhist(g);

[M,N] = size(f);
for i=1:M
    for j=1:N
    orig = double(f(i,j));
    resu = 255*(1/(1+exp(-0.045*(orig-127))));
    g(i,j) = uint8(resu);
    end
end
%Visualizacion antes y despues del cambio
close all; figure;imshow([f,g])
figure
subplot(2,1,1);imhist(f);subplot(2,1,2);imhist(g);
```

4. ROTACIÓN IMÁGENES

```
f = imread ('cameraman.tif');
g = imrotate(f,30);
imshow(g);
```

5. MODIFICACIÓN INSTOGRAMA

```
[g, T] = histeq(f);


% Histograma deseado

[g, T] = histeq(f,[0.5 1 1 2 2 2 3 1]);

%Esos números solo tienen sentido de manera relativa entre ellos. Cualquier
%combinación lineal entre ellos da el mismo resultado. No debo poner muchos
%números debido a las limitaciones, cada punto establece una restricción y
%si aumento mucho el número de restricciones puedo tener problemas a la
%hora de encontrar T.
```

BLOQUE 3. SEPARACIÓN DE REGIONES

1. GRADIENTE

```
% Lectura de la imagen
forig = imread('coins.png');
f = double(forig);
[M,N] = size (f);

%Valores inciiales
Gx = zeros(M,N);
Gy = zeros(M,N);

%Calculo de la aproximacion discreta del gradiente
for i=2:M
    for j=2:N
        Gx(i,j) = f(i,j) - f(i,j-1);
        Gy(i,j) = f(i,j) - f(i-1,j);
    end
end

GM = sqrt(Gx.^2 + Gy.^2);


%Imagen binaria de contornos mediante el umbral del gradiente
U = 56;


b = uint8 ( 255*( GM > U ) );


imshow(b);


[ModuloGradiente,DireccGradiente] = imgradient(I);


imshowpair(ModuloGradiente, DireccGradiente, 'montage');
```

2. GRADIENTE CON PLANTILLA

```
qx = [0 0 -1;...
      0 1 0;...
      0 0 0];

qy = [0 0 0;...
      0 1 0;...
      0 0 -1];

U = 70;

Gx = zeros(N,M);
Gy = zeros(N,M);

for i = 2:N-1
    for j = 2:M-1
        ty=f(i-1:i+1,j-1:j+1);
        Gx(i,j) = sum(sum(qx.*ty));
        Gy(i,j) = sum(sum(qy.*ty));
    end
end

G = sqrt(Gx.^2 + Gy.^2);
bc = G > U;

figure(2);
subplot(1,2,1)
imshow(h)
subplot(1,2,2)
imshow(bc)
```

3. CONTORNOS (EDGE)

```
I = imread('circuit.tif');
imshow(I)
SW1 = edge(I, 'prewitt');
figure;imshow(SW1);title('contornos');
%Estas funciones hace una busqueda de los umbrales mas adecuados.
%Para averiguar el umbral (está escalado);
[BW,U] = edge(I,'Prewitt')
```

4. LAPLACIANA

```
%paso por cero: Si L1L2<0. Cruce por 0 ahi, donde L = f*p

f = imread('coins.png');
[M,N]=size(f);
L= zeros(M,N);
U=1900; %UMBRAL DE CRUCES POR 0
h=double(f);
pL=[0 -1 0;-1 4 -1;0 -1 0];

for i=2:M-1
    for j=2:N-1
        ty = h(i-1:i+1,j-1:j+1);
        L(i,j) = sum(sum(pL.*ty));
    end
end



b= uint8(zeros(M,N));
for i=2:M-1
    for j=2:N-1
       L1 = L(i,j); L2=L(i-1,j); L3=L(i,j-1);
       if((L1*L2<-U) || (L1*L3<-U))
         b(i,j)=255;
       end
    end
end


imshow(L);
```

5. RECORRIDO CONTORNOS CON CADENAS

```
cadena = [0 0 0 3 0 3 3 2 2 2 2 1 1 1];
imagen = uint8(zeros(10,10));
f = double(imagen);
[M,N] = size(f);
for i=2:M-2
    for j=2:N-2
        f(i,j) = 255;   %He creado un rectangulo
    end
end

%busqueda punto inicial
for i=1:M
    for j=1:N
        if f(i,j)>0
            i_ini = i;
            j_ini = j;
            break;
        end
    end
    if i_ini>0, break; end
end

%bucle del recorrido
ic_ant = i_ini; jc_ant = j_ini;     %valores anteriores
cont = zeros(M,N);                  %matriz que tendrá el contorno
cont(i_ini,j_ini) = 1;
```

```
for k=1:length(cadena)
    switch (cadena(k))
        case 0
            ic = ic_ant;
            jc = jc_ant + 1;
        case 1
            ic = ic_ant - 1;
            jc = jc_ant;
        case 2
            ic = ic_ant;
            jc = jc_ant - 1;
        case 3
            ic = ic_ant + 1;
            jc = jc_ant;
    end
    cont(ic,jc) = 1;
    ic_ant = ic; %actualizo valores anteriores
    jc_ant = jc;
end

micont = uint8(255*cont);
subplot(1,3,2)
imshow(micont)
title('Contorno realizado con cadena')
```

6. SIN CADENA

```
imagen = uint8(zeros(10,10));

f = double(imagen); [M,N] = size(f);

%Busqueda del punto inicial
for i=1:M
    for j=1:N
        if f(i,j)>0
            i_ini = i;
            j_ini = j;
            break;
        end
    end
    if i_ini>0, break; end
end

%Bucle para el recorrido
ic_ant = i_ini; jc_ant = j_ini;
jc = 0; ic = 0;

while (jc~=j_ini && ic~=i_ini)
    jc = jc_ant+1; ic = ic_ant; %a la derecha
    if f(ic,jc)~=0
        cont(ic,jc)=1;
        ic_ant = ic; jc_ant = jc;
    else
        ic = ic_ant + 1; jc = jc_ant; %abajo
        if f(ic,jc)~=0
        cont(ic,jc)=1;
        ic_ant = ic; jc_ant = jc;
        else
            ic = ic_ant; jc = jc_ant - 1 %izquierda
            if f(ic,jc)~=0
            cont(ic,jc)=1;
            ic_ant = ic; jc_ant = jc;
            else
                ic = ic_ant - 1; jc = jc_ant; %arriba
                if f(ic,jc)~=0
                cont(ic,jc)=1;
                ic_ant = ic; jc_ant = jc;
                end
            end
        end
    end
end

micont = uint8(255*cont);
subplot(1,3,3)
imshow(micont)
title('Contorno realizado sin cadena')
```

7. EROSIÓN ACREECIÓN

```
close all;
%Ejemplo erosión
forig = imread('coins.png');
umbral = 115;
plant=double(forig)>umbral;
f=uint8(255*plant);  %imagen de silueta
imshow(f);title('Silueta antes de erosión');

%Erosion con plantilla p
p=[0 1 0;1 1 1;0 1 0];
[M,N]=size(f);

%valor inicial
g=uint8(zeros(M,N));
for i=2:M-1
    for j=2:N-1
        cont=0; %contador de elementos en el vector
        for di=-1:1
            for dj=-1:1
                if(p(2+di,2+dj)>0)
                    cont=cont+1;
                    vector_de_elementos(cont)=f(i+di,j+dj);
                end
            end
        end
        g(i,j)=min(vector_de_elementos); %Si quisera acreecion coloco max.
    end
end

figure;imshow(g);
```

8. CÁLCULO PERÍMETRO

```
% valor inicial del perímetro antes del recorrido
perim = 0;
%variable que se pone a cero cuando haya dado la vuelta completa  al contorno
para así parar
seguir = 1;
visitado = zeros(M, N);
% valor inicial del punto actual
%busqueda punto inicial
for i=1:M
    for j=1:N
        if f(i,j)>0
            iini = i;
            jini = j;
            break;
        end
    end
    if i_ini>0, break; end
end

ic = iini; jc = jini;
while seguir
        %cálculo del punto vecino siguiente
        for di=-1:1
                for dj=-1:1
                        %vecino considerado
                        ivec = ic+di; jvec = jc+dj;

            % si el vecino no ha sido visitado y es del contorno
                        if(visitado(ivec,jvec) < 1 && b(ivec,jvec) > 0)
                                isig = ivec; jsig = jvec; % se marca como el
siguiente
                        end
                end
        end

        % se suma al perímetro el segmento actual
        perim = perim+sqrt((isig-ic)^2+(jsig-jc)^2);

    % se comprueba si se ha dado la vuelta
    if(isig == iini && jsig == jini)
        %se desactiva la marca que permite seguir
        seguir = 0;
    end

    %se actualiza la posición
    ic = isig; jc = jsig;

    %se actualiza la matriz de visitados
    visitado(isig, jsig) = 1;
end
```

9. DIFERENCIAR CÍRCULOS DE CUADRADOS

```
xc=130; yc=100;for R=0:0.5:40


    for ang=0:0.001:2*pi
        x = xc + R*cos(ang);
        y = yc + R*sin(ang);
        pB(fix(y),fix(x))=1; %Quitamos los decimales
    end

end

[M, N] = size(pA);
pA=double(pA);
m00 = 0; m11 = 0;
m01 = 0; m10 = 0;
m02 = 0; m20 = 0;
for i=1:M
    for j=1:N
    m00 = m00 + pA( i,j);
    m10 = m10 + i*pA( i,j);
    m11 = m11 + i*j*pA( i,j);
    m02 = m02 + j^2*pA( i,j);
    m20 = m20 + i^2*pA( i,j);
    end
end

%momentos centrados para A
mu20a = m20 - m10*m10/m00;
mu02a = m02 - m01*m01/m00;


[M, N] = size(pB);
pB = double(pB);
m00 = 0; m11 = 0;
m01 = 0; m10 = 0;
m02 = 0; m20 = 0;
for i=1:M
    for j=1:N
    m00 = m00 + pB( i,j);
    m01 = m01 + j*pB( i,j);
    m10 = m10 + i*pB( i,j);
    m11 = m11 + i*j*pB( i,j);
    m02 = m02 + j^2*pB( i,j);
    m20 = m20 + i^2*pB( i,j);
    end
end

%momentos centrados para B
mu20b = m20 - m10*m10/m00;
mu02b = m02 - m01*m01/m00;

DMA = abs(mu02a-mu02a);
DMB = abs(mu20b-mu02b);

%Clasificacion
if DMA>DMB
     disp('La Pieza A es el rectangulo');
else
     disp('La pieza B es un circulo');
end
```

4. RECONOCIMIENTO DE FORMAS

1. CLASIFICADOR LINEAL

```
% Grupo training para ajuste
x1=[0.5, 2.0, 3.6, 1.2, 3.4, 3.9, 4.2, 4.9, 3.0, 9.0, 7.6, 8.5, 6.9 ];
x2=[1.5, 1.7, 1.0, 0.5, 4.3, 4.2, 4.7, 3.9, 4.1, 8.5, 7.0, 8.2, 8.5 ];
c= [ 1 , 1 , 1, 1, 2 , 2, 2, 2, 2, 3, 3, 3, 3 ];

% Representación datos
i1 = find(c==1);
i2 = find(c==2);
i3 = find(c==3);

plot(x1(i1),x2(i1),'or');
hold on;
plot(x1(i2),x2(i2),'*b');
hold on;
plot(x1(i3),x2(i3),'sk');
hold on;grid on;
axis([0 10 0 10]);

% Calculo discriminante
% fd(x) = vx - w

x1d = 0:0.01:10;
fd2 = -1/2*x1d + 4;                    % xd1=(40-2*x1d)/5;
fd1 = -2/5*x1d + 8;                    % xd2=(8-x1d)/2;

plot(x1d,fd1,'-.r',x1d,fd2,'-.b');

% Chequeo
D1=[2 5 -40];
D2=[1 2 -8];

cont=0;

for i=1:length(x1)
    checkv1 = D1 * [x1(i);x2(i);1];
    checkv2 = D1 * [x1(i);x2(i);1];

if checkv1<0 && checkv2<0 % Clase1
    plot(x1(i),x2(i),'.g');

elseif checkv1>0 && checkv2<0 % Clase 2
    plot(x1(i),x2(i),'.b');

elseif checkv1>0 && checkv2>0 % Clase 3

    plot(x1(i),x2(i),'.r');
end

end

% Indice de Bondad

x1o=[6, 7, 6, 6, 2, 3, 4, 6, 5];
x2o=[2, 2, 8, 6, 8, 1, 6, 9, 5];
co =[1, 2, 3, 2, 3, 1, 2, 3, 2];
cc = ones(size(co));
CI=0;
```

```
% Calculamos la clasificacion mediante los discriminantes

for i=1:length(x1o)
checkv1=D1*[x1o(i) x2o(i) 1]'; % linea superior
checkv2=D2*[x1o(i) x2o(i) 1]'; % linea inferior

if checkv1<0 && checkv2<0       % clase 1
    plot(x1o(i),x2o(i),'om');
    cc(i)=1;
elseif  checkv1<0 && checkv2>0  % clase 2
    plot(x1o(i),x2o(i),'*m');
    cc(i)=2;
elseif  checkv1>0 && checkv2>0  % clase 3
    plot(x1o(i),x2o(i),'sm');
    cc(i)=3;
end

if cc(i)~=co(i)
    CI=CI+1;
end

end

PCI=100*(CI/length(x1o));
```

```
load fisheriris

% meas : medidas o caractetisticas : se emplean 4 caracteristicas
% species : vector de clases : Hay 3 clases
% Cargamos una base de datos

PL = meas(:,3);
PW = meas(:,4);

% Nos quedamos con las dos ultimas  caracteristicas para la representacion
% Se emplea un  plot tipo scatter para representarlo
figure
h1 = gscatter(PL,PW,species,'krb','ov^',[],'off');
h1(1).LineWidth = 2;
h1(2).LineWidth = 2;
h1(3).LineWidth = 2;
legend('Setosa','Versicolor','Virginica','Location','best')
hold on

% Se crea el discriminante lineal
X = [PL,PW];
cls = fitcdiscr(X,species);

% Se plotean los discriminantes calculados
% Se obtienen los coeficientes para el discriminante lineal entre las
% clases 2 y 3.

w = cls.Coeffs(2,3).Const; % First retrieve the coefficients for the linear
v = cls.Coeffs(2,3).Linear;% boundary between the second and third classes
                           % (versicolor and virginica).

% Se dibuja la curva  [v1 v2]*[x1,x2]' + w  = 0.

f = @(x1,x2) w + v(1)*x1 + v(2)*x2;

h2 = ezplot(f,[.9 7.1 0 2.5]);  % Version facil del plot
h2.Color = 'r';
h2.LineWidth = 2;
```

```
% Discriminante entre la clase 1 y la 2
w2 = cls.Coeffs(1,2).Const;
v2 = cls.Coeffs(1,2).Linear;

% Se vuelve a pintar la curva discriminante [v1 v2]*[x1,x2]' + w  = 0:
f2 = @(x1,x2) w2 + v2(1)*x1 + v2(2)*x2;

h3 = ezplot(f2,[.9 7.1 0 2.5]);
h3.Color = 'k';
h3.LineWidth = 2;
axis([.9 7.1 0 2.5])

xlabel('Caracteristica 1')
ylabel('Caracteristica 2')
title('{\bf Ejemplo de Clasificacion lineal}')


% Índice de bondad

c=ones(size(species));


CI=0


for i=1:length(species)

 fd1=f(X(i,1),X(i,2));   % entre la clase 2 y 3
 fd2=f2(X(i,1),X(i,2));  % entre la clase 1 y 2

 if strcmp(species(i),'setosa')==1
    c(i)=1;
elseif strcmp(species(i),'versicolor')==1
    c(i)=2;
elseif strcmp(species(i),'virginica')==1
    c(i)=3;
end


 if fd2>0 && fd1>0 % pertenece a la clase 1
   if c(i)~=1
       CI=CI+1;
   end
 elseif fd2<0 && fd1>0 % pertenece a la clase 1
    if c(i)~=2
       CI=CI+1;
   end
 elseif fd2<0 && fd1<0 % pertenece a la clase 3
    if c(i)~=3
       CI=CI+1;
   end
 end

end

PCI=100*(CI/length(species))
```

2. VECINOS MÁS CERCANOS

```
load fisheriris
% meas : medidas o caractetisticas : se emplean 4 caracteristicas
% species : vector de clases : Hay 3 clases

PL = meas(:,3);
PW = meas(:,4);

% calculamos los prototipos
P1_PL=0;P1_PW=0;
P2_PL=0;P2_PW=0;
P3_PL=0;P3_PW=0;
n1=0;n2=0;n3=0;

for i=1:length(species)

if strcmp(species(i),'setosa')==1
    c(i)=1;
    P1_PL=P1_PL+PL(i);
    P1_PW=P1_PW+PW(i);
    n1=n1+1;
elseif strcmp(species(i),'versicolor')==1
    c(i)=2;
    P2_PL=P2_PL+PL(i);
    P2_PW=P2_PW+PW(i);
    n2=n2+1;
elseif strcmp(species(i),'virginica')==1
    c(i)=3;
    P3_PL=P3_PL+PL(i);
    P3_PW=P3_PW+PW(i);
    n3=n3+1;
end

end

P1_PL=P1_PL/n1;
P1_PW=P1_PW/n1;
P2_PL=P2_PL/n2;
P2_PW=P2_PW/n2;
P3_PL=P3_PL/n3;
P3_PW=P3_PW/n3;


% Nos quedamos con las dos ultimas  caracteristicas para la representacion

% Se emplea un  plot tipo scatter para representarlo

figure
h1 = gscatter(PL,PW,species,'krb','ov^',[],'off');
h1(1).LineWidth = 2;
h1(2).LineWidth = 2;
h1(3).LineWidth = 2;
legend('Setosa','Versicolor','Virginica','Location','best')
hold on
plot(P1_PL,P1_PW,'xm','LineWidth',3);
plot(P2_PL,P2_PW,'xm','LineWidth',3);
plot(P3_PL,P3_PW,'xm','LineWidth',3);

% Montamos los discriminantes

f1 = @(x1,x2) sqrt((x1-P1_PL).^2+(x2-P1_PW).^2);
f2 = @(x1,x2) sqrt((x1-P2_PL).^2+(x2-P2_PW).^2);
f3 = @(x1,x2) sqrt((x1-P3_PL).^2+(x2-P3_PW).^2);

speciesnew=species;

[value,clas]=min([f1(PL,PW) f2(PL,PW) f3(PL,PW)],[],2);
```

```
for i=1:length(species)

if clas(i)==1
    speciesnew{i}='setosa';
elseif clas(i)==2
    speciesnew{i}='versicolor';
elseif clas(i)==3
    speciesnew{i}='virginica';
end

end

figure
h2 = gscatter(PL,PW,speciesnew,'krb','ov^',[],'off');
h2(1).LineWidth = 2;
h2(2).LineWidth = 2;
h2(3).LineWidth = 2;
legend('Setosa','Versicolor','Virginica','Location','best')
axis([.9 7.1 0 2.5])

xlabel('Caracteristica 1')
ylabel('Caracteristica 2')
title('{\bf Ejemplo de Clasificacion lineal}')
```

3. NAIVES BAYES

```
load fisheriris
X = meas(:,3:4);
Y = species;
tabulate(Y)

Md1 = fitcnb(X,Y);

Mdl = fitcnb(X,Y,...
    'ClassNames',{'setosa','versicolor','virginica'})
```

```
% Añada en el hueco el cogido MATLAB que se requiere para clasificar la
% forma de la region indicada por p. Se usara un clasificador de Bayes para
% decidir si la region pertenece a las clases ci=1,2,3. Se tiene que:

% Imagen f dada
f=imread('coins.png');

[n,m]=size(f);

P1=[20;0.5;2];
P2=[50;2;0.1];
P3=[30;2;1.5];

p=f>40;

Pc1=0.5;
Pc2=0.3;
Pc3=0.2;

m00=0;
m30=0;
```

```
m10=0;
m01=0;
mu20=0;
mu03=0;

for i=1:n
    for j=1:m
       m00=m00+p(i,j);
       m10=m10+p(i,j)*i;
       m01=m01+p(i,j)*j;
       m30=m30+p(i,j)*(i^3);
    end
end

% centro de gravedad

ig=m10/m00;
jg=m01/m00;

for i=1:n
    for j=1:m
       mu20=mu20+p(i,j)*((i-ig)^2);
       mu03=mu03+p(i,j)*((j-jg)^3);
    end
end


xo=[mu20/m00^2 mu03/m00^2 m30/m00^3]';

% Probabilidades condicionales

dP1=norm(xo-P1,2);
dP2=norm(xo-P2,2);
dP3=norm(xo-P3,2);
dtotal=dP1+dP2+dP3;

[val,clase]=min([dP1 dP2 dP3])


Pcond1=dP1/dtotal;
Pcond2=dP2/dtotal;
Pcond3=dP3/dtotal;

Px=Pcond1*Pc1+Pcond2*Pc2+Pcond3*Pc3;

Pcn1=Pcond1*Pc1/Px;
Pcn2=Pcond2*Pc2/Px;
Pcn3=Pcond3*Pc3/Px;

[val,clase]=max([Pcn1 Pcn2 Pcn3]);
```

5. OTRAS FORMAS DE PERCEPCIÓN

1. RADIANCIA ESPECTRAL

```
h = 6.626068e-34;                    % Units are m^2 kg / s
c=299792458;                         % velocidad de la luz (m/s)
k=1.3806488e-23;                     % constante de boltzman


Ird = @(T,delta) (2.*h.*c^2./(delta.^5)).*(1./(-1+exp(h.*c./(delta.*k.*T))));

%plot(273:10:2000,Ird(273:10:2000,1000e-9))
T1=30;


T2=60;




IRIm=imread('Catedral1.JPG');
imshow(IRIm)

[N,M]=size(IRIm(:,:,1));
IRIT=zeros(N,M);
for i=1:N
    for j=1:M
       IRIT(i,j)=ColorVirtual(IRIm(i,j,1),IRIm(i,j,2),IRIm(i,j,3));
    end
end
figure(2)
imshow(IRIT)

% calculamos las zonas que no cumplen

f1=IRIT<T1;
f2=IRIT>T2;



function T = ColorVirtual(R,G,B)

T=0;

if (R==0)&&(B==255)&&(G>0)&&(G<255)
     T=G*64/255;
elseif (R==0)&&(G==255)&&(B>0)&&(B<255)
     T=128+B*((-128+64)/255);
elseif (B==0)&&(G==255)&&(R>0)&&(R<255)
     T=128+R*(191-128)/255;
elseif (B==0)&&(R==255)&&(G>0)&&(G<255)
     T=255+G*((-255+191)/255);
end


end
```

2. CORRESPONDENCIA

```
Ic1=double(imread('Catedral1.JPG'));
Ic2=double(imread('Catedral2.JPG'));

Ic2=Ic2(1:size(Ic1,1),1:size(Ic1,2),:);

[Ny,Mx]=size(Ic1(:,:,1));

tam=floor(round([Ny,Mx]./20)/2);

Ic1bn=rgb2gray(uint8(Ic1));
Ic2bn=rgb2gray(uint8(Ic2));

figure
imshow(Ic1bn);
figure
imshow(Ic2bn);

% Detector de puntos singulares

[Ix, Iy] = imgradientxy(Ic1bn,'prewitt');

figure;
imshowpair(Ix, Iy, 'montage');
title('Gradientes direccionales: Direccion x, Gx (Izq), Direccion y, Gy (Der),
metodo Prewitt')
axis off;

Autoval1=zeros(Ny,Mx);
Autoval2=zeros(Ny,Mx);

for i=1:Ny
    for j=1:Mx
        % Montamos el tensor por cada punto de la imagen
        eigval=eig([Ix(i,j)^2 Ix(i,j)*Iy(i,j);Iy(i,j)*Ix(i,j) Iy(i,j)^2]);
        Autoval1(i,j)=eigval(1);
        Autoval2(i,j)=eigval(2);
    end
end

% buscamos puntos singulares

[fila2,columna2]=find(Autoval2>1e5);
[fila1,columna1]=find(Autoval1>1e5);

if isempty(fila2)&&isempty(fila1)
    error('No existen puntos singulares');
end

fila=[fila1;fila2];
columna=[columna1;columna2];

figure
imshow(Ic1bn);
hold on
plot(columna,fila,'or');

% Calculo de correspondencia entre imagenes
Corresp=[];
npsing=length(fila);
np=4;
for i=1:20
   % Para cada punto singular
   % Calculo SD para un entorno de np cuadrados R en la imagen Ic2 y
   % me quedo con el minimo
  vsolmin=Inf;
  vsolposmin=[];
```

```
  for j=fila(i)-np:fila(i)+np
      for k=columna(i)-np:columna(i)+np
        % Calculamos SD
        if (j>=1)&&(k>=1)
        vsol=CalculaSD(Ic1bn,Ic2bn,tam,[fila(i),columna(i)],[j,k]);
        if vsol<vsolmin
            vsolmin=vsol;
            vsolposmin=[j,k];
        end
        end
      end
  end

  Corresp=[Corresp;fila(i),columna(i),vsolposmin(1),vsolposmin(2)];

end

point=30;
figure(11)
subplot(1,2,1)
imshow(Ic1bn);
hold on
plot(Corresp(point,2),Corresp(point,1),'or');
subplot(1,2,2)
imshow(Ic2bn);
hold on
plot(Corresp(point,4),Corresp(point,3),'xr');

% Funcion Corner

meth2='MinimumEigenvalue';
C = corner(Ic1bn);
figure;
imshow(Ic1bn);
hold on
plot(C(:,1), C(:,2), 'r*');

C = corner(Ic1bn,meth2);
figure;
imshow(Ic1bn);
hold on
plot(C(:,1), C(:,2), 'r*')
```

```
function sum=CalculaSD(Im1,Im2,tam,cr,cs)

sum=0;
[Nmax,Mmax]=size(Im1);
for i=cr(1) - tam(1):cr(1)+tam(1)
    for j=cr(2) - tam(2):cr(2)+tam(2)
        k= i-cr(1)+cs(1);
        l= j-cr(2)+cs(2);

        if (i>=1)&&(j>=1)&&(k>=1)&&(l>=1)...
                &&(i<=Nmax)&&(j<=Mmax)&&(k<=Nmax)&&(l<=Mmax)
            sum = sum+abs(Im1(i,j)-Im2(k,l));
        end

    end
end
```

6. EXÁMENES

1. SUAVIZADO SELECTIVO

```
f = imread('coins.png');

UII = 46;           %Umbral intensidad minima
UIS = 150;          %Umbral Intensidad maxima
UMG = 30;           %Umbral gradiente

%Para calcular el gradiente en el examen lo usamos a manos a mano. No
%podemos usar imgradiente. Lo usamos aqui para no levarnos mucho tiempo

ffond=(f>UII).*(f<UIS);
fobj=not(ffond);

figure
subplot(1,2,1)
imshow(f)
subplot(1,2,2)
imshow(fobj)


% Calculamos gradiente
[Gx,Gy] = imgradient(f,'prewitt'); %Intentar hacerla usando plantilla

figure;
imshowpair(Gx,Gy,'montage');

[Gmag,Gdir]=imgradient(Gx,Gy);

fgsel=Gmag>UMG;
figure;
imshow(fgsel);

% Calculamos la imagen final
fsel=fgsel.*fobj;
close all;
figure
imshow(fsel)


% Calculamos suavizado

gfinal=SuavizaV4(double(f)); %Le mando suvizado a todo pero despues me quedo
solo con la zona de los puntos que me ha pedido

g=uint8(fsel).*gfinal+uint8(not(fsel)).*f;
```

2. PLANTILLA GRADIENTE

```
h = imread('coins.png');

% B. Obtenga la imagen binaria de la silueta de la imagen almacenada
% en h y almacenela en b, utilizando como umbral la intensidad de gris
% media de la mitan inferior de h

[N,M] = size(h);
f = double(h);
cont = 0;
for i = floor(N/2):N % Asegurarnos de manejar numeros enteros
    for j = 1:M
        cont = cont + f(i,j);
    end
end

total = floor((M*N/2)); % Asegurarnos de que los numeros sean enteros
intmedia = cont/total;
b = f > intmedia;


figure(1)
subplot(1,2,1)
imshow(h)
subplot(1,2,2)
imshow(b)
qx = [0 0 -1;...
      0 1 0;...
      0 0 0];

qy = [0 0 0;...
      0 1 0;...
      0 0 -1];

U = 70;

Gx = zeros(N,M);
Gy = zeros(N,M);

for i = 2:N-1
    for j = 2:M-1
        ty=f(i-1:i+1,j-1:j+1);
        Gx(i,j) = sum(sum(qx.*ty));
        Gy(i,j) = sum(sum(qy.*ty));
    end
end

G = sqrt(Gx.^2 + Gy.^2);
bc = G > U;

figure(2);
subplot(1,2,1)
imshow(h)
subplot(1,2,2)
imshow(bc)


q = ones(3);
```

```
% Calcular b-(b-q). Doble erosión

gi = zeros(N,M);
for i = 2:N-1
    for j = 2:M-1
        bi = b(i-1:i+1,j-1:j+1);
        gi(i,j) = min(min(q.*bi));
    end
end

t=zeros(N,M);
ti = b-gi;
t(2:N-1,2:M-1) = ti(2:N-1,2:M-1);

figure(3)
subplot(1,2,1)
imshow(h)
subplot(1,2,2)
imshow(t)
```

3. SEPARACIÓN CLASIFICADOR

```
Una camara toma imagenes digitales en las que siempre existe un fondo
% oscuro homogeneo y un unico objeto de tono mas claro. Se desea clasificar
% el objeto en una de tres posibles clases. Se proporcionan los tres
% prototipos para el discriminante por distancias P!,P2,P3.
% El vector de caracteristicas es: x=(mur maxv muc)^T siendo
% mur: valor medio de rojo.
% maxv: valor maximo de verde.
% muc: valor medio de (r-a)/(a+1).
% Codifique en MATLAB el programa.
clear all;close all;
% tomamos una imagen de ejemplo

fc = imread('Catedral1.jpg');
figure;
imshow(fc);

% Prototipos

P1=[0.2 10 50]';
P2=[100 40 200]';
P3=[50 200 50]';

% calculamos el vector de caracteristicas

rojo=double(fc(:,:,1));
verde=double(fc(:,:,2));
azul=double(fc(:,:,3));
aux=(rojo-azul)./(azul+1);

mur=mean(rojo(:));
maxv=max(verde(:));
muc=mean(aux(:));

xo=[mur maxv muc]';

[val,clase]=min([norm(xo-P1,2),norm(xo-P2,2),norm(xo-P3)])
```

4. CORRESPONDENCIA

```
close all;clear all;

Ic1=double(imread('Catedral1.JPG'));
Ic2=double(imread('Catedral2.JPG'));

Ic2=Ic2(1:size(Ic1,1),1:size(Ic1,2),:);

[Ixr, Iyr]=imgradientxy(Ic1(:,:,1),'sobel');
[Ixg, Iyg]=imgradientxy(Ic1(:,:,2),'sobel');
[Ixb, Iyb]=imgradientxy(Ic1(:,:,3),'sobel');

[M,N]=size(Ic1(:,:,1));
tam=floor(round([M,N]./20)/2);
%Montamos tensor


for i=1:M
    for j=1:N
        eigvalr=eig([Ixr(i,j)^2 Ixr(i,j)*Iyr(i,j);Iyr(i,j)*Ixr(i,j)
Iyr(i,j)^2]);
        eigvalg=eig([Ixg(i,j)^2 Ixg(i,j)*Iyg(i,j);Iyg(i,j)*Ixg(i,j)
Iyg(i,j)^2]);
        eigvalb=eig([Ixb(i,j)^2 Ixb(i,j)*Iyb(i,j);Iyb(i,j)*Ixb(i,j)
Iyb(i,j)^2]);

        Autovalr1(i,j)=eigvalr(1);
        Autovalr2(i,j)=eigvalr(2);
        Autovalg1(i,j)=eigvalg(1);
        Autovalg2(i,j)=eigvalg(2);
        Autovalb1(i,j)=eigvalb(1);
        Autovalb2(i,j)=eigvalb(2);
    end
end


[filr1,colr1]=find(Autovalr1>1e5);
[filr2,colr2]=find(Autovalr2>1e5);

[filg1,colg1]=find(Autovalg1>1e5);
[filg2,colg2]=find(Autovalg2>1e5);

[filb1,colb1]=find(Autovalb1>1e5);
[filb2,colb2]=find(Autovalb2>1e5);

filr=[filr1;filr2];
filg=[filg1;filg2];
filb=[filb1;filb2];

colr=[colr1;colr2];
colg=[colg1;colg2];
colb=[colb1;colb2];

figure
subplot(1,3,1)
imshow(uint8(Ic1(:,:,1)));
hold on
plot(colr,filr,'or');
title('rojo');
subplot(1,3,2)
imshow(uint8(Ic1(:,:,2)));
hold on
plot(colr,filr,'or');
title('verde');
subplot(1,3,3)
imshow(uint8(Ic1(:,:,3)));
hold on
```

```
plot(colr,filr,'or');
title('azul');
Corresp=[];

npsing=length(filr);

np=4;

vsolminr=Inf;vsolming=Inf;vsolminb=Inf;

vsolposminr=[];vsolposming=[];vsolposminb=[];

Correspr=[];Correspg=[];Correspb=[];

for i=1:20

    for j=filr(i)-np:filr(i)+np
        for k=colr(i)-np:colr(i)+np
            if(j>=1)&&(k>=1)
                vsolr =
CalculaSD(Ic1(:,:,1),Ic2(:,:,1),tam,[filr(i),colr(i)],[j,k]);
                if vsolr < vsolminr
                    vsolminr=vsolr;
                    vsolposminr=[j,k];
                end
            end
        end
    end
    Correspr=[Correspr;filr(i),colr(i),vsolposminr(1),vsolposminr(2)];



end

% Verde
for i=1:20
    for j=filg(i)-np:filg(i)+np
        for k=colg(i)-np:colg(i)+np
            if(j>=1)&&(k>=1)
                vsolg =
CalculaSD(Ic1(:,:,2),Ic2(:,:,2),tam,[filg(i),colg(i)],[j,k]);
                if vsolg < vsolming
                    vsolming=vsolg;
                    vsolposming=[j,k];
                end
            end
        end
    end
    Correspg=[Correspg;filg(i),colg(i),vsolposming(1),vsolposming(2)];
end


for i=1:20
    for j=filb(i)-np:filb(i)+np
        for k=colb(i)-np:colb(i)+np
            if(j>=1)&&(k>=1)
                vsolb =
CalculaSD(Ic1(:,:,3),Ic2(:,:,3),tam,[filb(i),colb(i)],[j,k]);
                if vsolb < vsolminb
                    vsolming=vsolb;
                    vsolposminb=[j,k];
                end
            end
        end
```

```
    end
    Correspb=[Correspb;filb(i),colb(i),vsolposminb(1),vsolposminb(2)];
End

point=15;

figure(11)subplot(1,2,1)

imshow(uint8(Ic1(:,:,1)));

hold on
plot(Correspr(point,2),Correspr(point,1),'or');
subplot(1,2,2)
imshow(uint8(Ic2(:,:,1)));
hold on
plot(Correspr(point,4),Correspr(point,3),'or');


%Reconstrucción 3D

f = 0.03;        %Distancia focal
b = 0.025;       %Distancia entre centros
dpix=3e-3;

disparidad = double(Correspr(:,2)-Correspr(:,4));

profundidad = f*b./(disparidad.*dpix);
```

5. CLASIFICADOR POR DISTANCIA

```
f = imread('cuadrado.jpg');
h = double(f);

[M,N] = size(f);

% Prototipos
Mp = [ 1, 5, 4, 6;...
       25, 40, 60, 90;...
       44, 66, 10, 30];
p1 = Mp(:,1);
p2 = Mp(:,2);
p3 = Mp(:,3);
p4 = Mp(:,4);


MaxI = 0;
MinI = Inf;
Imax = 0;
Imin = 0;

for i=1:M
    for j=1:N

        if(h(i,j) > MaxI)
            Imax = h(i,j);
            MaxI = Imax;
        end

        if(h(i,j) < MinI)
            Imin = h(i,j);
            MinI = Imin;
        end
```

```
    end
end

Beta = Imax - Imin;
% Beta = max(max(h))-min(min(h));

% Calculamos momentos
m00 = 0;
m01 = 0; m10 = 0;
m02 = 0; m20 = 0;
m30 = 0; m03 = 0;
for i=1:M
    for j=1:N
        m00 = m00+h(i,j);
        m10 = m10+i*h(i,j);
        m01 = m01+j*h(i,j);
        m20 = m20+i^2*h(i,j);
        m02 = m02+j^2*h(i,j);
        m30 = m30+i^3*h(i,j);
        m03 = m03+j^3*h(i,j);
    end
end

iG = m10/m00; jG = m01/m00;

u31 = 0; u11 = 0;

for i=1:M
    for j=1:N
        u11 = u11 + (i-iG)*(j-jG)*h(i,j);
        u31 = u31 + (i-iG)^3*(j-jG)*h(i,j);
    end
end

alfa = u31 - u11;


% Gradiente

for i=2:M-1
    for j=2:N-1
        Gx = h(i,j) - h(i,j-1);
        Gy = h(i,j) - h(i-1,j);
    end
end

G = sqrt(Gx.^2+Gy.^2);

gamma = mean(G);

x = [alfa Beta gamma]';

% Clasificador por distancias
d1 = sqrt((x(1)-p1(1))^2+(x(2)-p1(2))^2+(x(3)-p1(3))^2);
d2 = sqrt((x(1)-p2(1))^2+(x(2)-p2(2))^2+(x(3)-p2(3))^2);
d3 = sqrt((x(1)-p3(1))^2+(x(2)-p3(2))^2+(x(3)-p3(3))^2);
d4 = sqrt((x(1)-p4(1))^2+(x(2)-p4(2))^2+(x(3)-p4(3))^2);

[value, class]=min([d1 d2 d3 d4]);
```

6. CLASIFICADOR DISTANCIAS

```
f = imread('cuadrado.jpg');
h = double(f);

[M,N] = size(f);

% Prototipos
Mp = [ 1, 5, 4, 6;...
       25, 40, 60, 90;...
       44, 66, 10, 30];
p1 = Mp(:,1);
p2 = Mp(:,2);
p3 = Mp(:,3);
p4 = Mp(:,4);


MaxI = 0;
MinI = Inf;
Imax = 0;
Imin = 0;

for i=1:M
    for j=1:N

        if(h(i,j) > MaxI)
            Imax = h(i,j);
            MaxI = Imax;
        end

        if(h(i,j) < MinI)
            Imin = h(i,j);
            MinI = Imin;
        end
    end
end

Beta = Imax - Imin;
% Beta = max(max(h))-min(min(h));

% Calculamos momentos
m00 = 0;
m01 = 0; m10 = 0;
m02 = 0; m20 = 0;
m30 = 0; m03 = 0;
for i=1:M
    for j=1:N
        m00 = m00+h(i,j);
        m10 = m10+i*h(i,j);
        m01 = m01+j*h(i,j);
        m20 = m20+i^2*h(i,j);
        m02 = m02+j^2*h(i,j);
        m30 = m30+i^3*h(i,j);
        m03 = m03+j^3*h(i,j);
    end
end

iG = m10/m00; jG = m01/m00;

u31 = 0; u11 = 0;

for i=1:M
    for j=1:N
        u11 = u11 + (i-iG)*(j-jG)*h(i,j);
        u31 = u31 + (i-iG)^3*(j-jG)*h(i,j);
    end
end
```

```
alfa = u31 - u11;


% Gradiente

for i=2:M-1
    for j=2:N-1
        Gx = h(i,j) - h(i,j-1);
        Gy = h(i,j) - h(i-1,j);
    end
end

G = sqrt(Gx.^2+Gy.^2);

gamma = mean(G);

x = [alfa Beta gamma]';

% Clasificador por distancias
d1 = sqrt((x(1)-p1(1))^2+(x(2)-p1(2))^2+(x(3)-p1(3))^2);
d2 = sqrt((x(1)-p2(1))^2+(x(2)-p2(2))^2+(x(3)-p2(3))^2);
d3 = sqrt((x(1)-p3(1))^2+(x(2)-p3(2))^2+(x(3)-p3(3))^2);
d4 = sqrt((x(1)-p4(1))^2+(x(2)-p4(2))^2+(x(3)-p4(3))^2);

[value, class]=min([d1 d2 d3 d4]);
```

7. PLANTILLA NO BINARIA

```
% P2 plantilla no binaria
%Se ha realizado la separacion de regiones de una imagen con respecto al
fondo, obteninendose una
%plantilla de pertenencia p (0 para fondo, 1 para objetos). A modo de ejemplo
considere:
p=[
0 0 0 0 0 0 0 0 0 0 0 0;
0 1 1 0 0 0 0 0 0 1 1 0;
0 1 1 0 0 0 0 0 0 1 1 0;
0 0 0 0 1 1 1 0 0 1 1 0;
0 1 1 0 1 1 1 0 0 1 1 0;
0 1 1 0 0 0 0 0 1 1 1 0;
0 1 1 1 1 1 1 1 1 1 1 0;
0 0 0 0 0 0 0 0 0 0 0 0];
%Se sabe que puede haber mas de un objeto por lo que la separaciÓon ha podido
dar lugar a varias regiones
%disjuntas. Se busca una plantilla q (no binaria) que indique si el pixel
pertenece al fondo (valor 0), al primer
%objeto (valor 1), al segundo objeto (valor 2), etc. A modo de ejemplo
considere:
%{
q=[
0 0 0 0 0 0 0 0 0 0 0 0;
0 1 1 0 0 0 0 0 0 3 3 0;
0 1 1 0 0 0 0 0 0 3 3 0;
0 0 0 0 2 2 2 0 0 3 3 0;
0 3 3 0 2 2 2 0 0 3 3 0;
0 3 3 0 0 0 0 0 0 3 3 0;
0 3 3 3 3 3 3 3 3 3 3 0;
0 0 0 0 0 0 0 0 0 0 0 0];
%}
[M,N]=size(p);
clase=1;
recorrido=zeros(M,N);
q=zeros(M,N);
for i=1:M
    for j=1:M
        if(p(i,j)==1 && q(i,j)==0)
            q(i,j)=clase;
            hayvecino=1;
            while hayvecino
                hayvecino=0;
                for k=2:M-1
                    for l=2:N-1
                        if(p(k,l)==1 && q(k,l)==0)
                            if( q(k-1,l-1)==clase || q(k-1,l)==clase || q(k-
1,l+1)==clase || ...
                                q(k  ,l-1)==clase ||                    q(k
,l+1)==clase || ...
                                q(k+1,l-1)==clase || q(k+1,l)==clase ||
q(k+1,l+1)==clase )

                                q(k,l)=clase;
                                hayvecino=1;
                            end
                        end
                    end
                end
            end
            clase=clase+1;
        end
    end
end
```