Segmentación

Proceso Digital de Imágenes

David Guillermo Morales Sáez

Índice

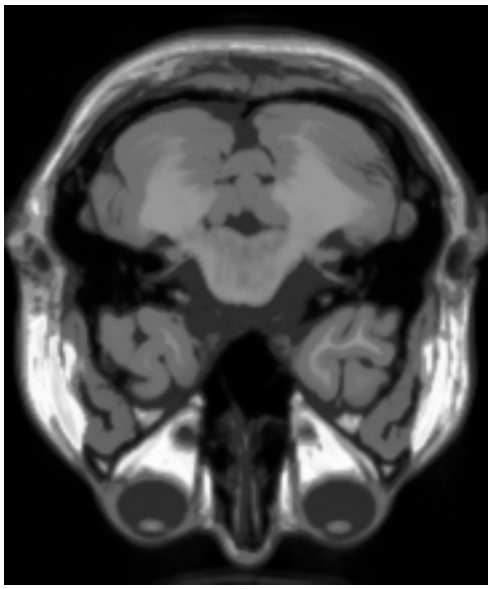
Gradientes

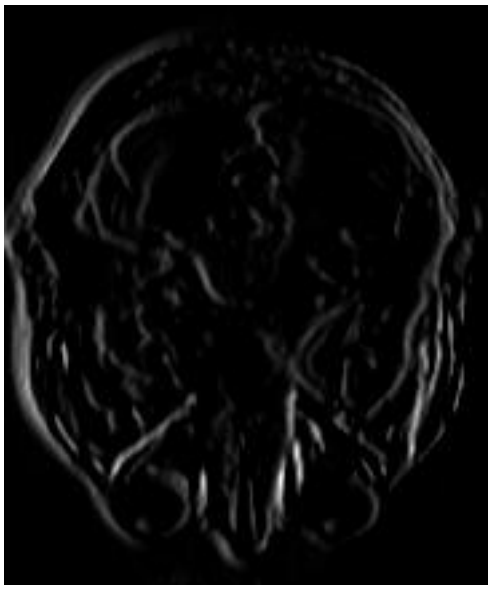
Gradiente de Prewitt

Para realizar el gradiente de Prewitt sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = GradientePrewitt( img )
    [x,y] = size(img);
    img2 = zeros(x+2,y+2);
    img2(2:x+1,2:y+1) = img;
    H = ([-1,0,1;
          -1,0,1;
          -1,0,1]./ 3);
    for i=1:x
        for j=1:y
            salida(i,j) = sum(sum(img2(i:1:i+2,j:1:j+2).*H));
        end
    end
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:



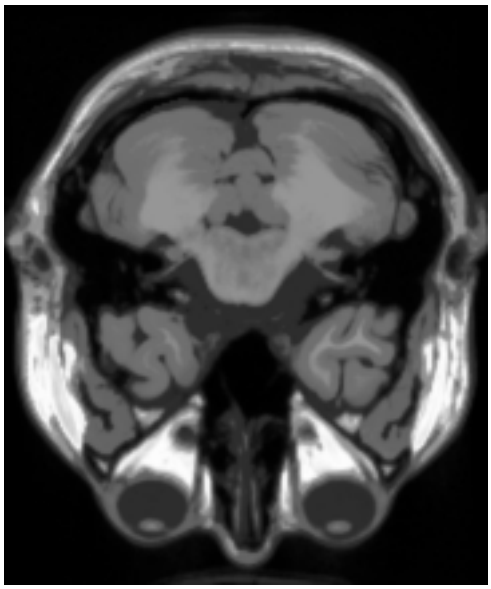


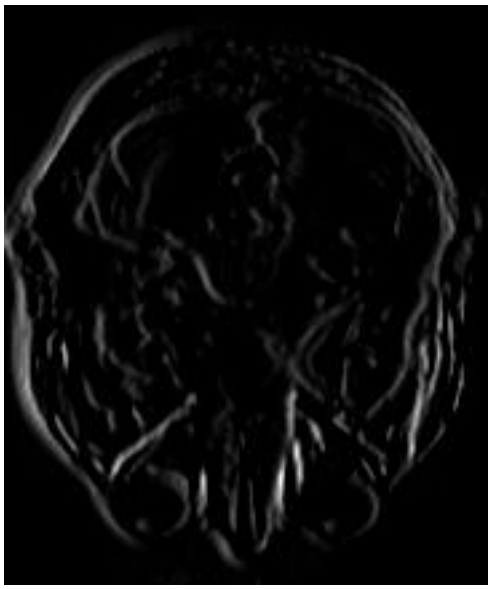
Gradiente de Sobel

Para realizar el gradiente de Sobel sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = GradienteSobel( img )
    [x,y] = size(img);
    img2 = zeros(x+2,y+2);
    img2(2:x+1,2:y+1) = img;
    H = ([-1,0,1;
          -2,0,2;
          -1,0,1]./ 4);
    for i=1:x
        for j=1:y
            salida(i,j) = sum(sum(img2(i:1:i+2,j:1:j+2).*H));
        end
    end
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:



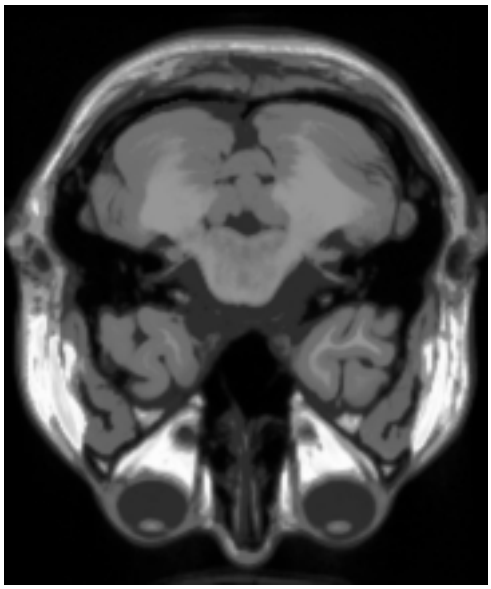


Gradiente de Frei-Chen

Para realizar el gradiente de Frei-Chen sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = GradienteFrei_Chen( img )
    [x,y] = size(img);
    img2 = zeros(x+2,y+2);
    img2(2:x+1,2:y+1) = img;
    raiz2=sqrt(2);
    H = ([         1,0,    -1;
          raiz2,0,-raiz2;
              1,0,    -1]./(2+raiz2));
    for i=1:x
        for j=1:y
            salida(i,j) = sum(sum(img2(i:1:i+2,j:1:j+2).*H));
        end
    end
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:

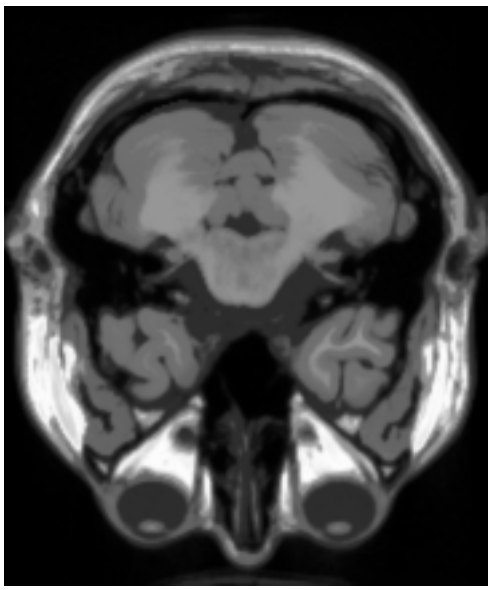


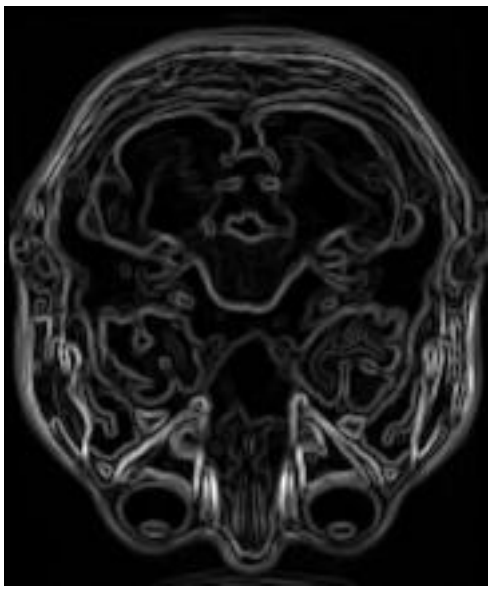
Gradiente de Kirsh

Para realizar el gradiente de Kirsh sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = GradienteKirsh( img )
    N=8;
    k = zeros (3,3,N);
    k(:,:,1) = [-3,-3,5;
                -3, 0,5;
                -3,-3,5];
    k(:,:,2) = [-3, 5, 5;
                -3, 0, 5;
                -3,-3,-3];
    for i=3:1:N
        k(:,:,i)=rot90(k(:,:,i-2));
    end
    G = zeros(size(img,1),size(img,2),N);
    [x,y] = size(img);
    img2 = zeros(x+4,y+4);
    img2(3:x+2,3:y+2) = img;
    for i=1:1:N
        for j=3:x
            for l=3:y
                salida(j,l,i) = sum(sum(img2(j:1:j+2,l:1:l+2).*k(:,:,i)));
            end
        end
    end
    % Se selecciona el maximo de cada filtro
    salida = max (abs(salida),[],3);
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:





Gradiente de Nevatia-Barbu

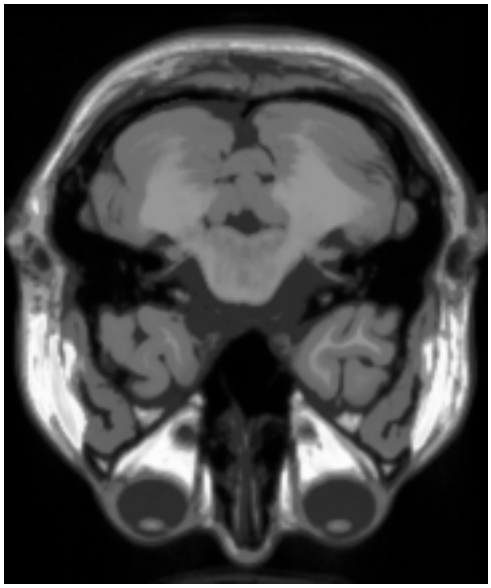
Para realizar el gradiente de Nevatia-Barbu sobre una imagen se ha utilizado la siguiente función en Matlab:

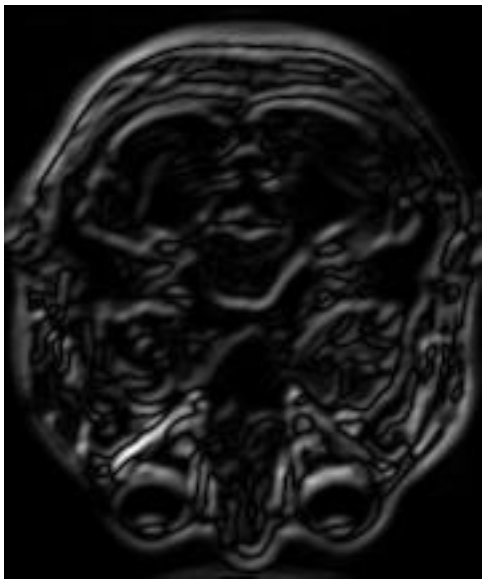
```
function [ salida ] = GradienteNevatia_Barbu( img )
     N = 12;

    k(:,:,1) = [  100/1000, 100/1000,         0,-100/1000,-100/1000;
                  100/1000, 100/1000,         0,-100/1000,-100/1000;
                  100/1000, 100/1000,         0,-100/1000,-100/1000;
                  100/1000, 100/1000,         0,-100/1000,-100/1000;
                  100/1000, 100/1000,         0,-100/1000,-100/1000];
    k(:,:,2) = [  100/1102, -32/1102,-100/1102,-100/1102,-100/1102;
                  100/1102,  78/1102, -92/1102,-100/1102,-100/1102;
                  100/1102, 100/1102,         0,-100/1102,-100/1102;
                  100/1102, 100/1102,  92/1102, -78/1102,-100/1102;
                  100/1102, 100/1102, 100/1102,  32/1102,-100/1102];
    k(:,:,3) = [ -100/1102,-100/1102,-100/1102,-100/1102,-100/1102;
                   32/1102, -78/1102,-100/1102,-100/1102,-100/1102;
                  100/1102,  92/1102,         0, -92/1102,-100/1102;
                  100/1102, 100/1102, 100/1102,  78/1102, -32/1102;
                  100/1102, 100/1102, 100/1102, 100/1102, 100/1102];
    k(:,:,4) = [ -100/1000,-100/1000,-100/1000,-100/1000,-100/1000;
                 -100/1000,-100/1000,-100/1000,-100/1000,-100/1000;
                         0,         0,         0,         0,         0;
                  100/1000, 100/1000, 100/1000, 100/1000, 100/1000;
                  100/1000, 100/1000, 100/1000, 100/1000, 100/1000];
    k(:,:,5) = [ -100/1102,-100/1102,-100/1102,-100/1102,-100/1102;
                 -100/1102,-100/1102,-100/1102, -78/1102,  32/1102;
                 -100/1102, -92/1102,         0,  92/1102, 100/1102;
                  -32/1102,  78/1102, 100/1102, 100/1102, 100/1102;
                  100/1102, 100/1102, 100/1102, 100/1102, 100/1102];
    k(:,:,6) = [ -100/1102,-100/1102,-100/1102, -32/1102, 100/1102;
                 -100/1102,-100/1102, -92/1102,  78/1102, 100/1102;
                 -100/1102,-100/1102,         0, 100/1102, 100/1102;
                 -100/1102, -78/1102,  92/1102, 100/1102, 100/1102;
                 -100/1102,  32/1102, 100/1102, 100/1102, 100/1102];
    k(:,:,7) = [ -100/1000,-100/1000,         0, 100/1000, 100/1000;
                 -100/1000,-100/1000,         0, 100/1000, 100/1000;
                 -100/1000,-100/1000,         0, 100/1000, 100/1000;
                 -100/1000,-100/1000,         0, 100/1000, 100/1000;
                 -100/1000,-100/1000,         0, 100/1000, 100/1000];
    k(:,:,8) = [ -100/1102,  32/1102, 100/1102, 100/1102, 100/1102;
                 -100/1102, -78/1102,  92/1102, 100/1102, 100/1102;
                 -100/1102,-100/1102,         0, 100/1102, 100/1102;
                 -100/1102,-100/1102, -92/1102,  78/1102, 100/1102;
                 -100/1102,-100/1102,-100/1102, -32/1102, 100/1102];
    k(:,:,9) = [  100/1102, 100/1102, 100/1102,100/1102 , 100/1102;
                  -32/1102,  78/1102, 100/1102,100/1102 , 100/1102;
                 -100/1102, -92/1102,         0,92/1102  , 100/1102;
                 -100/1102,-100/1102,-100/1102,-78/1102 ,  32/1102;
                 -100/1102,-100/1102,-100/1102,-100/1102,-100/1102];
    k(:,:,10) = [ 100/1000, 100/1000, 100/1000, 100/1000, 100/1000;
                  100/1000, 100/1000, 100/1000, 100/1000, 100/1000;
                          0,         0,         0,         0,         0;
                 -100/1000,-100/1000,-100/1000,-100/1000,-100/1000;
                 -100/1000,-100/1000,-100/1000,-100/1000,-100/1000];
    k(:,:,11) = [ 100/1102, 100/1102, 100/1102, 100/1102, 100/1102;
                  100/1102, 100/1102, 100/1102,  78/1102, -32/1102;
                  100/1102,  92/1102,         0, -92/1102,-100/1102;
                   32/1102, -78/1102,-100/1102,-100/1102,-100/1102;
                 -100/1102,-100/1102,-100/1102,-100/1102,-100/1102];
    k(:,:,12) = [ 100/1102, 100/1102, 100/1102,  32/1102,-100/1102;
                  100/1102, 100/1102,  92/1102, -78/1102,-100/1102;
                  100/1102, 100/1102,         0,-100/1102,-100/1102;
                  100/1102,  78/1102, -92/1102,-100/1102,-100/1102;
                  100/1102, -32/1102,-100/1102,-100/1102,-100/1102];
    for i=3:1:N
        k(:,:,i)=rot90(k(:,:,i-2));
    end
    G = zeros(size(img,1),size(img,2),N);
```

```
    [x,y] = size(img);
    img2 = zeros(x+4,y+4);
    img2(3:x+2,3:y+2) = img;
    for i=1:1:N
        for j=3:x
            for l=3:y
                salida(j-2,l-2) = sum(sum(img2(j-2:1:j+2,l-2:1:l+2).*k(:,:,i)));
            end
        end
    end
    % Se selecciona el maximo de cada filtro
    salida = max (abs(salida),[],3);
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:



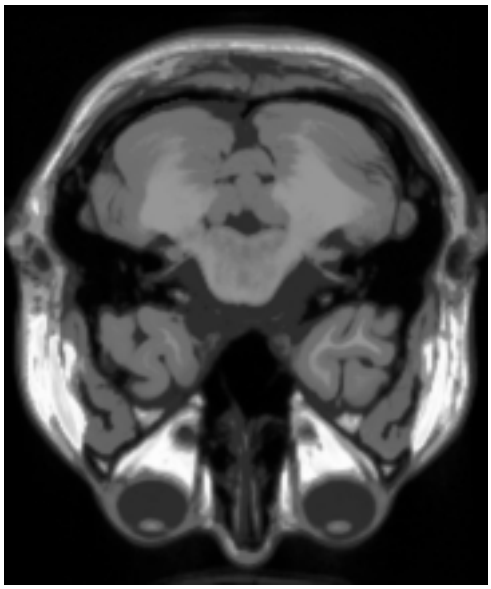


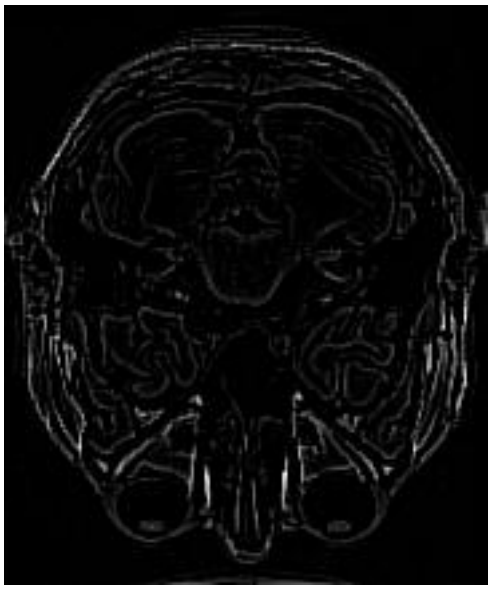
Gradiente Laplaciano-8

Para realizar el gradiente Laplaciano-8 sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = Laplaciana8( img )
    [x,y] = size(img);
    img2 = zeros(x+2,y+2);
    img2(2:x+1,2:y+1) = img;
    H = [-2  1 -2;
          1  4  1;
         -2  1 -2]./8;
    for i=1:x
        for j=1:y
            salida(i,j) = sum(sum(img2(i:1:i+2,j:1:j+2).*H));
        end
    end
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:



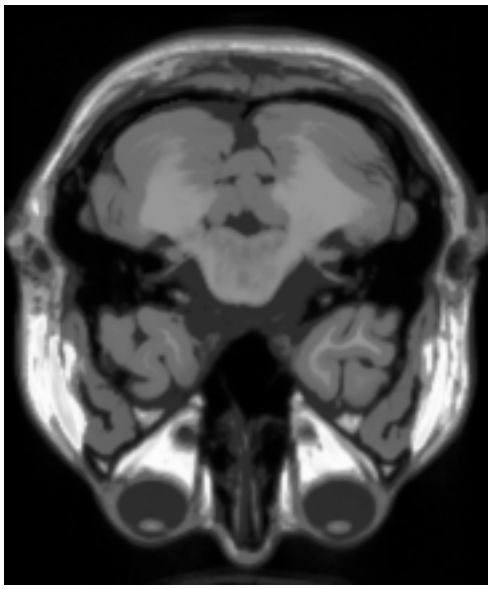


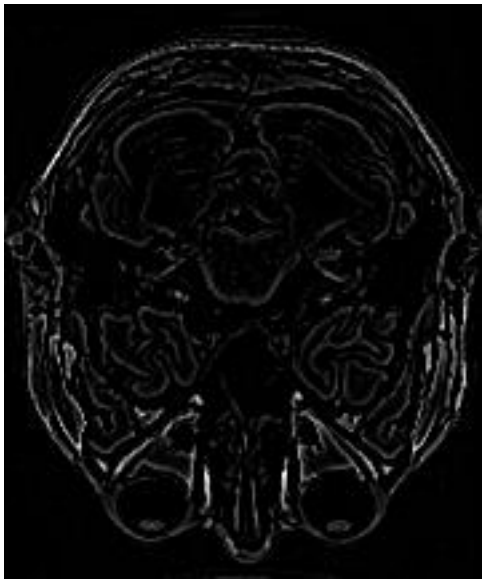
Gradiente LoG

Para realizar el gradiente LoG sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = LoG( img )
    [x,y] = size(img);
    img2 = zeros(x+4,y+4);
    img2(3:x+2,3:y+2) = img;
    H =[ 0  0 -1  0  0;
         0 -1 -2 -1  0;
        -1 -2 16 -2 -1;
         0 -1 -2 -1  0;
         0  0 -1  0  0];
    for i=3:x
        for j=3:y
            salida(i-2,j-2) = sum(sum(img2(i-2:1:i+2,j-2:1:j+2).*H));
        end
    end
    salida = uint8(salida.*255./max(max(salida)));
end
```

obteniendo el siguiente resultado:





Umbralizado Adaptable

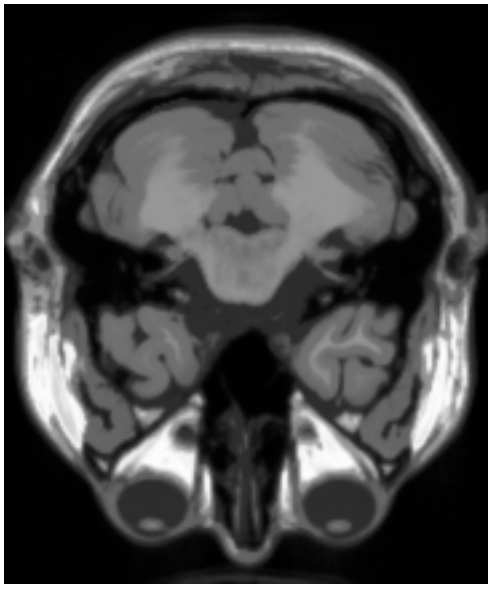
Para realizar el Umbralizado Adaptable sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = Umbralizado( img )
    [r c]=size(img);
    L = 256;
    k = 80;
    p=zeros(1,L);
    for level=1:1:L
        for i=1:1:r
            for j=1:1:c
                if(img(i,j)==level)
                    p(level)=p(level)+1;
                end
            end
            p(level) = p(level);
        end
    end
    p1 = zeros(1,L);
    p2 = zeros(1,L);
    m1 = zeros(1,L);
    m2 = zeros(1,L);
    var = zeros(1,L);
    temp=0;
    temp2 = 0;
    temp3 = 0;
    temp4 = 0;
    for thresh=1:k
        for ite= 1:thresh
            temp=temp+p(ite);
        end
        p1(thresh)=temp;
        temp=0;
    end
    for thresh2 = k+1: L
        for i = thresh2:L

            temp2 = temp2 + p(i);
        end
        p2(thresh2)=temp2;
        temp2=0;
    end
    for thresh3 =1:k
        for i =1:thresh3
            temp3 = temp3 + i* p(i);
        end
        m1(thresh3) = temp3/p1(thresh3);
    end
    for thresh4 =k+1:L
        for i =thresh4:L
            temp4 = temp4 + i* p(i);
        end
        m2(thresh4) = temp4/p2(thresh4);
    end
    mg = p1(k)*m1(k)+p2(k)*m2(k);
    for v =1:L
        var(v) = ( p1(v)*(m1(v)-mg)^2 )+( p2(v)*(m2(v)-mg)^2 );
    end
    [maxNum IndexofMaxNum] = max(var);
    output=img;
    for i=1:r
        for j=1:c
            if output(i,j) >IndexofMaxNum
                output(i,j)=255;
            else
                output(i,j)=0;
            end
        end
    end
```

```
        salida = output;
    end
```

obteniendo el siguiente resultado:



Crecimiento de Regiones

Para realizar el Crecimiento de Regiones sobre una imagen se ha utilizado la siguiente función en Matlab:

```
function [ salida ] = CrecimientoRegiones( img, x, y )
    Isizes = size(img);
    J = zeros(Isizes);
    reg_mean = img(x,y);
    reg_size = 1;
    neg_free = 10000;
    neg_pos=0;
    neg_list = zeros(neg_free,3);
    pixdist=0;
    reg_maxdist = 0.2;
    neigb=[-1  0;
            1  0;
            0 -1;
            0  1];
    while(pixdist<reg_maxdist&&reg_size<numel(img))
        for j=1:4,
            xn = x +neigb(j,1);
            yn = y +neigb(j,2);
 if((xn>=1)&&(yn>=1)&&(xn<=Isizes(1))&&(yn<=Isizes(2))&&(J(xn,yn)==0))
                    neg_pos = neg_pos+1;
                    neg_list(neg_pos,:) = [xn yn img(xn,yn)];
                    J(xn,yn)=1;
            end
        end
        if(neg_pos+10>neg_free)
            neg_free=neg_free+10000;
            neg_list((neg_pos+1):neg_free,:)=0;
        end
        a = neg_list(1:neg_pos,3);
        dist = abs(uint8(a)-reg_mean);
        [pixdist, index] = min(dist);
        J(x,y)=2;
        reg_size=reg_size+1;
        reg_mean= (reg_mean*reg_size + neg_list(index,3))/(reg_size+1);
        x = neg_list(index,1);
        y = neg_list(index,2);
        neg_list(index,:)=neg_list(neg_pos,:);
        neg_pos=neg_pos-1;
    end

    salida=J>1;
end
```

obteniendo el siguiente resultado: