2013

Proceso Digital de Imágenes

David Morales Sáez

[TÉCNICAS DE FILTRADO]

Contenido

Introducción

Una de las primeras etapas del proceso digital de imágenes consiste en la mejora o realce de las imágenes adquiridas por un sistema de visión artificial, siendo el objetivo de las técnicas de realce procesar una imagen determinada de forma que la imagen resultante sea más adecuada que la original para una aplicación concreta.

Filtrado en el Dominio Espacial

Las técnicas en el dominio espacial se basan en operaciones sobre la propia imagen digital basadas en máscaras de convolución creadas al efecto o bien en métodos algorítmicos entre los cuales cabe destacar el filtro mediana. Se trata de métodos para resaltar o suprimir, de forma selectiva, información contenida en una imagen a diferentes escalas espaciales, para destacar algunos elementos de la imagen, o también para ocultar valores anómalos.

Filtros de paso bajo

Filtro de la media

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroMedia( img )
    [x, y] = size(img);
    for i=1:x
        for j=1:y
            if(i>1)
                if(j>1)
                    media = double(img(i-1,j-1));
                else
                    media = 0;
                end
                media = media + double(img(i-1,j));
                if(j<y)
                    media = media + double(img(i-1,j+1));
                end
            else
                media = 0;
            end
            if(j>1)
                media = media + double(img(i,j-1));
            end
            media = media + double(img(i,j));
            if(j<y)
                media = media + double(img(i,j+1));
            end
            if(i<x)
                if(j>1)
                    media = media + double(img(i+1,j-1));
                end
                media = media + double(img(i+1,j));
                if(j<y)
                    media = media + double(img(i+1,j+1));
                end
            end
            aux(i,j) = media/9;
        end
    end
    salida = im2uint8(mat2gray(aux));
end
```

Obteniendo los siguientes resultados:

Filtro de la media ponderada

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroMediaPonderada( img, pesos )
    [x, y] = size(img);
    for i=1:x
        for j=1:y
            if(i>1)
                if(j>1)
                    media = double(pesos(1)*img(i-1,j-1));
                else
                    media = 0;
                end
                media = media + double(pesos(2)*img(i-1,j));
                if(j<y)
                    media = media + double(pesos(3)*img(i-1,j+1));
                end
            else
                media = 0;
            end
            if(j>1)
                media = media + double(pesos(4)*img(i,j-1));
            end
            media = media + double(pesos(5)*img(i,j));
            if(j<y)
                media = media + double(pesos(6)*img(i,j+1));
            end
            if(i<x)
                if(j>1)
                    media = media + double(pesos(7)*img(i+1,j-1));
                end
                media = media + double(pesos(8)*img(i+1,j));
                if(j<y)
                    media = media + double(pesos(9)*img(i+1,j+1));
                end
            end
            aux(i,j) = media/9;
        end
    end
    salida = im2uint8(mat2gray(aux));
end
```

Obteniendo los siguientes resultados utilizando diferentes pesos:

Filtro de la mediana

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroMediana( img )
    [x, y] = size(img);
    for i=1:x
        for j=1:y
            if(i>1)
                if(j>1)
                    valores(1) = double(img(i-1,j-1));
                    n = 2;
                else
                    n = 1;
                end
                valores(n) = double(img(i-1,j));
```

```
                n = n+1;
                if(j<y)
                    valores(n) = double(img(i-1,j+1));
                    n = n+1;
                end
            else
                n = 1;
            end
            if(j>1)
                valores(n) = double(img(i,j-1));
                n = n+1;
            end
            valores(n) = double(img(i,j));
            n = n+1;
            if(j<y)
                valores(n) = double(img(i,j+1));
                n = n+1;
            end
            if(i<x)
                if(j>1)
                    valores(n) = double(img(i+1,j-1));
                    n = n+1;
                end
                valores(n) = double(img(i+1,j));
                n = n+1;
                if(j<y)
                    valores(n) = double(img(i+1,j+1));
                    n = n+1;
                end
            end
            valores = sort(valores);
            aux(i,j) = valores(round(n/2));
            n = 0;
            clear valores;
        end
    end
    salida = im2uint8(mat2gray(aux));
end
```

Con el cuál, se han obtenido los siguientes resultados:

Filtros de paso alto

Filtro del Gradiente de Roberts

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroGradienteRoberts( img )
    [x, y] = size(img);
    for i=1:x
        for j=1:y
            if(i<x)
                if(j<y)
                    salida(i,j) = abs(img(i,j)-img(i+1, j+1)) + abs(img(i+1, j)-img(i, j+1));
                else
                    salida(i,j) = abs(img(i,j)) + abs(img(i+1, j));
                end
            else
                if(j<y)
                    salida(i,j) = abs(img(i,j)) + abs(img(i, j+1));
                else
                    salida(i,j) = abs(img(i,j));
                end
            end
        end
    end
end
```

Obteniéndose los siguientes resultados:

Filtro del Gradiente de Sobel

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroGradienteSobel( img )
    [x, y] = size(img);
    for i=1:x
        for j=1:y
            if(1<i && i<x)
                if(1<j && j<y)
```

```
                    p1 = abs(img(i-1, j+1) + 2*img(i, j+1) + img(i+1, j+1) - img(i-1, j-1) +
2*img(i, j-1) + img(i+1, j-1));
                    p2 = abs(img(i+1, j-1) + 2*img(i+1, j) + img(i+1, j+1) - img(i-1, j-1) +
2*img(i-1, j) + img(i-1, j+1));
                else
                    if(j == 1)
                        p1 = abs(img(i-1, j+1) + 2*img(i, j+1) + img(i+1, j+1));
                        p2 = abs(2*img(i+1, j) + img(i+1, j+1)+ 2*img(i-1, j) + img(i-1, j+1));
                    else
                        p1 = abs(-img(i-1, j-1) + 2*img(i, j-1) + img(i+1, j-1));
                        p2 = abs(img(i+1, j-1) + 2*img(i+1, j) - img(i-1, j-1) + 2*img(i-1, j));
                    end
                end
            else
                if(i == 1)
                   if(1<j && j<y)
                        p1 = abs(2*img(i, j+1) + img(i+1, j+1) + 2*img(i, j-1) + img(i+1, j-1));
                        p2 = abs(img(i+1, j-1) + 2*img(i+1, j) + img(i+1, j+1));
                    else
                       if(j == 1)
                           p1 = abs(2*img(i, j+1) + img(i+1, j+1));
                           p2 = abs(2*img(i+1, j) + img(i+1, j+1));
                       else
                           p1 = abs(2*img(i, j-1) + img(i+1, j-1));
                           p2 = abs(img(i+1, j-1) + 2*img(i+1, j));
                       end
                   end
                else
                   if(1<j && j<y)
                        p1 = abs(img(i-1, j+1) + 2*img(i, j+1) - img(i-1, j-1) + 2*img(i, j-1));
                        p2 = abs(-img(i-1, j-1) + 2*img(i-1, j) + img(i-1, j+1));
                    else
                       if(j == 1)
                           p1 = abs(img(i-1, j+1) + 2*img(i, j+1));
                           p2 = abs(2*img(i-1, j) + img(i-1, j+1));
                       else
                           p1 = abs(-img(i-1, j-1) + 2*img(i, j-1));
                           p2 = abs(-img(i-1, j-1) + 2*img(i-1, j));
                       end
                   end
                end
            end
            salida(i,j) = p1-p2;
        end
    end
end
```

y se han obtenido los siguientes resultados:

Filtro Laplaciano

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroLaplaciano( img, mascara)
    [x, y] = size(img);
    for i=1:x
        for j=1:y
            if(1<i && i<x)
                if(1<j && j<y)
                    gradiente = img(i+1, j)*mascara(3,2) + img(i-1, j)*mascara(1,2) + img(i,
j+1)*mascara(2,3) + img(i, j-1)*mascara(2,1) + img(i,j)*mascara(2,2);
                    gradiente = gradiente + img(i-1, j-1)*mascara(1,1) + img(i-1,
j+1)*mascara(1,3) + img(i+1, j-1)*mascara(3,1) + img(i+1, j+1)*mascara(3,3);
                else
                    if(j == 1)
                        gradiente = img(i+1, j)*mascara(3,2) + img(i-1, j)*mascara(1,2) + img(i,
j+1)*mascara(2,3) + img(i,j)*mascara(2,2);
                        gradiente = gradiente + img(i-1, j+1)*mascara(1,3) + img(i+1,
j+1)*mascara(3,3);
                    else
                        gradiente = img(i+1, j)*mascara(3,2) + img(i-1, j)*mascara(1,2) + img(i,
j-1)*mascara(2,1) + img(i,j)*mascara(2,2);
                        gradiente = gradiente + img(i-1, j-1)*mascara(1,1) + img(i+1, j-
1)*mascara(3,1);
                    end
                end
            else
                if(i == 1)
                    if(1<j && j<y)
                        gradiente = img(i+1, j)*mascara(3,2) + img(i, j+1)*mascara(2,3) + img(i,
j-1)*mascara(2,1) + img(i,j)*mascara(2,2);
                        gradiente = gradiente + img(i+1, j-1)*mascara(3,1) + img(i+1,
j+1)*mascara(3,3);
                    else
                        if(j == 1)
                            gradiente = img(i+1, j)*mascara(3,2) + img(i, j+1)*mascara(2,3) +
img(i,j)*mascara(2,2);
                            gradiente = gradiente + img(i+1, j+1)*mascara(3,3);
                        else
                            gradiente = img(i+1, j)*mascara(3,2) + img(i, j-1)*mascara(2,1) +
img(i,j)*mascara(2,2);
                            gradiente = gradiente + img(i+1, j-1)*mascara(3,1);
                        end
                    end
                else
                    if(1<j && j<y)
                        gradiente = img(i-1, j)*mascara(1,2) + img(i, j+1)*mascara(2,3) + img(i,
j-1)*mascara(2,1) + img(i,j)*mascara(2,2);
                        gradiente = gradiente + img(i-1, j-1)*mascara(1,1) + img(i-1,
j+1)*mascara(1,3);
                    else
                        if(j == 1)
                            gradiente = img(i-1, j)*mascara(1,2) + img(i, j+1)*mascara(2,3) +
img(i,j)*mascara(2,2);
                            gradiente = gradiente + img(i-1, j+1)*mascara(1,3);
                        else
                            gradiente = img(i-1, j)*mascara(1,2) + img(i, j-1)*mascara(2,1) +
img(i,j)*mascara(2,2);
                            gradiente = gradiente + img(i-1, j-1)*mascara(1,1);
                        end
                    end
                end
```

```
            end
            if (mascara(2,2)<0)
                salida(i,j) = img(i,j) - gradiente;
            else
                salida(i,j) = img(i,j) + gradiente;
            end
        end
    end
end
```

y se han obtenido los siguientes resultados con distintas máscaras:

Filtro de Promediado Direccional

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroPromediadioDireccional( img )
    [x, y] = size(img);
    for i=1:x
        for j=1:y
```

```
            if (1<i && i<x)
                if (1<j && j<y)
                    v(1) = (img(i-1, j-1) + img(i,j) + img(i+1, j+1))/3;
                    v(2) = (img(i-1, j) + img(i,j) + img(i+1, j))/3;
                    v(3) = (img(i, j-1) + img(i,j) + img(i, j+1))/3;
                else
                    if (j==1)
                        v(1) = (img(i,j) + img(i+1, j+1))/2;
                        v(2) = (img(i-1, j) + img(i,j) + img(i+1, j))/3;
                        v(3) = (img(i,j) + img(i, j+1))/2;
                    else
                        v(1) = (img(i-1, j-1) + img(i,j))/2;
                        v(2) = (img(i-1, j) + img(i,j) + img(i+1, j))/3;
                        v(3) = (img(i, j-1) + img(i,j))/2;
                    end
                end
            else
                if (i==1)
                    if (1<j && j<y)
                        v(1) = (img(i,j) + img(i+1, j+1))/2;
                        v(2) = (img(i,j) + img(i+1, j))/2;
                        v(3) = (img(i, j-1) + img(i,j) + img(i, j+1))/3;
                    else
                        if (j==1)
                            v(1) = (img(i,j) + img(i+1, j+1))/2;
                            v(2) = (img(i,j) + img(i+1, j))/2;
                            v(3) = (img(i,j) + img(i, j+1))/2;
                        else
                            v(1) = img(i,j);
                            v(2) = (img(i,j) + img(i+1, j))/2;
                            v(3) = (img(i, j-1) + img(i,j))/2;
                        end
                    end
                else
                    if (1<j && j<y)
                        v(1) = (img(i-1, j-1) + img(i,j))/2;
                        v(2) = (img(i-1, j) + img(i,j))/2;
                        v(3) = (img(i, j-1) + img(i,j) + img(i, j+1))/3;
                    else
                        if (j==1)
                            v(1) = img(i,j);
                            v(2) = (img(i-1, j) + img(i,j))/2;
                            v(3) = (img(i,j) + img(i, j+1))/2;
                        else
                            v(1) = (img(i-1, j-1) + img(i,j))/2;
                            v(2) = (img(i-1, j) + img(i,j))/2;
                            v(3) = (img(i, j-1) + img(i,j))/2;
                        end
                    end
                end
            end
            salida(i,j) = max(v);
        end
    end
end
```

Obteniéndose los siguientes resultados:

Filtrado en el Dominio Frecuencial

Las técnicas desarrolladas en el dominio frecuencial se refieren mayoritariamente al diseño de filtros, tanto paso bajo cuya meta es llevar a cabo un suavizado de la imagen con vistas a la reducción de ruido como paso alto cuyo efecto es el opuesto, es decir, la acentuación de la imagen para resaltar los cambios de intensidades que se producen en la misma.

Filtros de paso bajo

Filtro ideal

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroIdealBajo( img, dist )
   imgFrec = fftshift(fft2(img));
   [x, y] = size(img);
   xcent = x/2;
   ycent = y/2;
   im2 = zeros(x,y);
   for i=1:x
      for j=1:y
         dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
         if (dist2<=dist)
            im2(i,j) = 1;
         end
      end
   end
   im2 = im2.*imgFrec;
   salida = uint8(ifft2(ifftshift(im2)));
end
```

y se han obtenido los siguientes resultados:


Distancia: 10


Distancia: 30


Distancia: 50

Filtro Butterworth

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroButterworthBajo( img, dist, n )
    imgFrec = fftshift(fft2(img));
    [x, y] = size(img);
    xcent = x/2;
    ycent = y/2;
    img2 = zeros(x,y);
    for i=1:x
        for j=1:y
            dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
            img2(i,j) = 1/(1+power(dist2/dist, n*2));
        end
    end
    img2 = img2.*imgFrec;
    salida = uint8(ifft2(ifftshift(img2)));
end
```

con el cuál, se han obtenido los siguientes resultados:


Distancia: 10


Distancia: 30


Distancia: 50

Filtro Exponencial

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroExponencial( img, dist, n )
    imgFrec = fftshift(fft2(img));
    [x, y] = size(img);
    xcent = x/2;
    ycent = y/2;
    img2 = zeros(x,y);
    for i=1:x
        for j=1:y
            dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
            img2(i,j) = imgFrec(i,j)*exp(-power(dist2/dist, n));
        end
    end
    salida = uint8(ifft2(ifftshift(img2)));
end
```

Obteniéndose los siguientes resultados:


Distancia: 10


Distancia: 30


Distancia: 50

Filtro Gausiano

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroGausianoBajo( img, dist )
    imgFrec = fftshift(fft2(img));
    [x, y] = size(img);
    xcent = x/2;
    ycent = y/2;
    img2 = zeros(x,y);
    for i=1:x
        for j=1:y
            dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
            img2(i,j) = exp(-dist2*dist2/(2*dist*dist));
        end
    end
    img2 = img2.*imgFrec;
    salida = uint8(ifft2(ifftshift(img2)));
end
```

y se han obtenido los siguientes resultados:


Distancia: 10


Distancia: 30


Distancia: 50

Filtros de paso alto

Filtro ideal

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroIdealBajo( img, dist )
    imgFrec = fftshift(fft2(img));
    [x, y] = size(img);
    xcent = x/2;
    ycent = y/2;
    im2 = zeros(x,y);
    for i=1:x
        for j=1:y
            dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
            if (dist2>dist)
                im2(i,j) = 1;
            end
        end
    end
    im2 = im2.*imgFrec;
    salida = uint8(ifft2(ifftshift(im2)));
```

```
end
```

Obteniéndose los siguientes resultados:

Distancia: 10 Distancia: 30 Distancia: 50

Filtro Butterworth

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroButterworthAlto( img, dist, n )
    imgFrec = fftshift(fft2(img));
    [x, y] = size(img);
    xcent = x/2;
    ycent = y/2;
    img2 = zeros(x,y);
    for i=1:x
        for j=1:y
            dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
            img2(i,j) = 1/(1+power(dist/dist2, n*2));
        end
    end
    img2 = img2.*imgFrec;
    salida = uint8(ifft2(ifftshift(img2)));
end
```

Con el cuál se han obtenido los siguientes resultados:


Distancia: 10


Distancia: 30


Distancia: 50

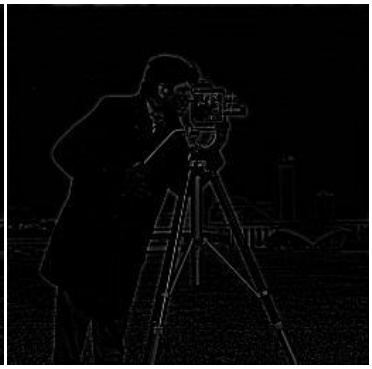
Filtro Gausiano

Para realizar este filtro, se ha creado la siguiente función en MatLab:

```
function [ salida ] = FiltroGausianoAlto( img, dist )
    imgFrec = fftshift(fft2(img));
    [x, y] = size(img);
    xcent = x/2;
    ycent = y/2;
    img2 = zeros(x,y);
    for i=1:x
        for j=1:y
            dist2 = sqrt((i-xcent)*(i-xcent) + (j-ycent)*(j-ycent));
            img2(i,j) = 1-exp(-dist2*dist2/(2*dist*dist));
        end
    end
    img2 = img2.*imgFrec;
    salida = uint8(ifft2(ifftshift(img2)));
end
```

y se han obtenido los siguientes resultados:

Distancia: 10



Distancia: 30

Distancia: 50