COMPUTER VISION 1

Homework 9

姓名	：	蘇宛琳
系所	：	電信所碩一
學號	：	R05942060
指導教授	：	傅楸善老師

Computer Vision Report – Homework 9

R05942060 蘇宛琳

Question :

Write programs to generate the following gradient magnitude images and choose proper thresholds to get the binary edge images:

1. Roberts operator
2. Prewitt edge detector
3. Sobel edge detector
4. Frei and Chen gradient operator
5. Kirsch compass operator
6. Robinson compass operator
7. Nevatia-Babu 5X5 operator

* Roberts operator Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Roberts operators with threshold 10 : two 2X2 masks to calculate gradient

$$2\times2\ masks：r_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\ and\ r_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$gradient\ magnitude：\sqrt{{r_1}^2 + {r_2}^2} > Threshold$$

利用每點影像的像素值和 mask 分別相乘，達到邊緣偵測的結果值，在依照 gradient 定義把影像值寫回入新的影像矩陣中。利用『im2double』將原本 Lena 影像中的每點像素轉換成 doubles，方便和 mash 數值做運算。最後的 threshold 判斷將影像變成『二值化』的步驟即可求得最後結果。

Source code

```
function robert=RobertOperator(image,threshold)
image=imread('lena.bmp');
b=im2double(image);
[m,n]=size(image);

newimage_robert=zeros(size(image));
threshold=10;
L(1:m,1:n)=0;
for i=1:m-2;
    for j=1:m-2;
        r1=-1*b(i,j)+0+0+1*b(i+1,j+1);
        L(i,j)=r1*r1;
    end
end
M(1:m,1:n)=0;
for i=1:m-2;
```

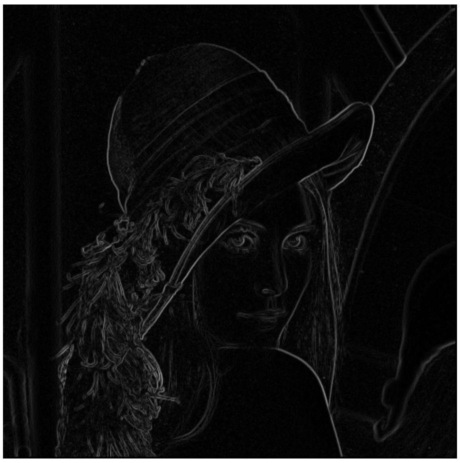
```
   for j=1:m-2;
       r2=0-1*b(i,j+1)+1*b(i+1,j)+0;
       M(i,j)=r2*r2;
   end
end
for i=1:m-2;
   for j=1:m-2;
       gradient =sqrt(L(i,j)+M(i,j));
       newimage_robert(i,j)= gradient;
   end
end
figure;
imshow(newimage_robert);
imwrite(newimage_robert,'robert1.bmp')
figure;
robert=imread('robert.bmp');
[m,n]=size(robert);

for i=1:m
   for j=1:n
       if robert(i,j)>threshold
         robert(i,j)=0;
       else
          robert(i,j)=1;
       end
   end
end
imshow(uint8(robert)*255);
imwrite(uint8(robert)*255,'robert_thres.bmp')
end
```

Result

Original image

Edge

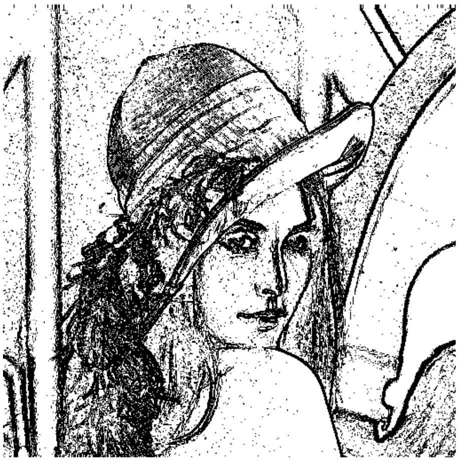
operator



Edge image

利用threshold = 10

來取得二值化影像。



Robert's Operator Edge image

* Prewitt edge detector Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Prewitt edge detector with threshold 25 : two 3X3 masks in row column direction

$$3\times3\ masks:\ p_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}\ and\ p_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$gradient\ magnitude:\ \sqrt{{p_1}^2 + {p_2}^2} > Threshold$$

利用每點影像的像素值和 mask 分別相乘，達到邊緣偵測的結果值，在依照 gradient 定義把影像值寫回入新的影像矩陣中。利用『im2double』將原本 Lena 影像中的每點像素轉換成 doubles，方便和 mash 數值做運算。最後的 threshold 判斷將影像變成『二值化』的步驟即可求得最後結果。這裡和上一個 operator 不一樣的地方為:此 為 3*3 矩陣。因此要修改矩陣的大小即可。

Source code

```
function prewitt=PrewittOperator(image,threshold)
image=imread('lena.bmp');

b=im2double(image);
[m,n]=size(image);
newimage_robert=zeros(size(image));
threshold=25;
L(1:m,1:n)=0;
for i=1:m-3;
    for j=1:m-3;
        r1=-1*b(i,j)+-1*b(i,j+1)+-
1*b(i,j+2)+0+0+0+1*b(i+2,j)+1*b(i+2,j+1)+1*b(i+2,j+2);
        L(i,j)=r1*r1;
    end
end
M(1:m,1:n)=0;
for i=1:m-3;
```

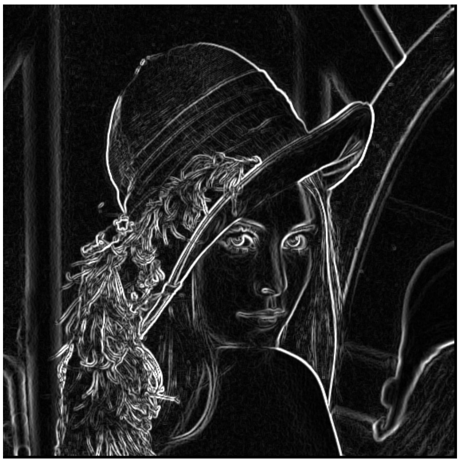
```
    for j=1:m-3;
        r2=-1*b(i,j)+0+1*b(i,j+2)-1*b(i+1,j)+0+1*b(i+1,j+2)-
1*b(i+2,j)+0+1*b(i+2,j+2);
        M(i,j)=r2*r2;
    end
end
for i=1:m-3;
    for j=1:m-3;
        gradient =sqrt(L(i,j)+M(i,j));
        newimage_robert(i,j)= gradient;
    end
end

figure;
imshow(newimage_robert);
imwrite(newimage_robert,'prewitt.bmp')
figure;
prewitt = imread('prewitt.bmp');
[m,n]=size(prewitt);
for i=1:m
    for j=1:n
        if prewitt(i,j)>threshold
          prewitt(i,j)=0;
        else
           prewitt(i,j)=1;
        end
    end
end
imshow(uint8(prewitt)*255);
imwrite(uint8(prewitt)*255,'prewitt_thres.bmp')
end
```

Result

Original image

Edge

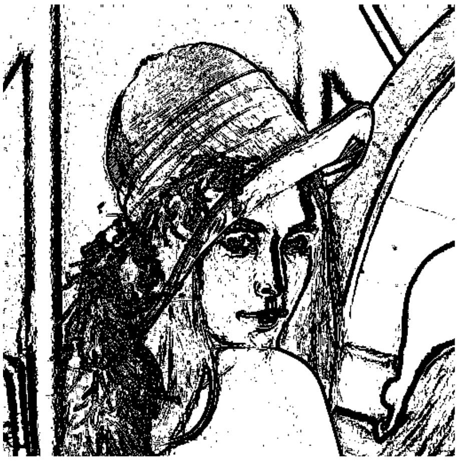
operator



Edge image

利用threshold = 25

來取得二值化影像。



prewitt's Operator Edge image

* Sobel's Edge Detector Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Sobel’s edge detector with threshold 36 : two 3X3 masks in row column direction

$$3\times3\ masks : S_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}\ and\ S_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$gradient\ magnitude : \sqrt{{s_1}^2 + {s_2}^2}\ > Threshold$$

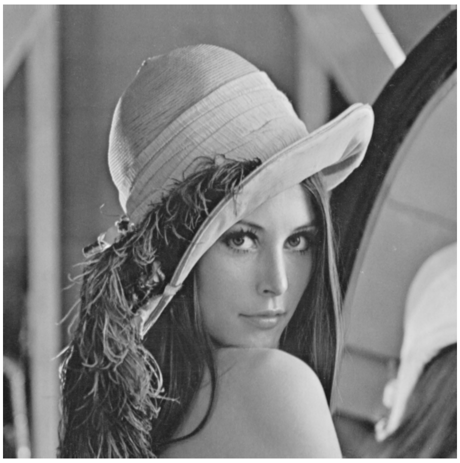
利用每點影像的像素值和 mask 分別相乘，達到邊緣偵測的結果值，在依照 gradient 定義把影像值寫回入新的影像矩陣中。利用『im2double』將原本 Lena 影像中的每點像素轉換成 doubles，方便和 mash 數值做運算。最後的 threshold 判斷將影像變成『二值化』的步驟即可求得最後結果。這裡和上一個 operator 運用到的概念幾乎一樣，唯一差別在 mash 的值。

Source code

```
function sobel=SobelOperator(image3,threshold)

image3=imread('lena.bmp');

b=im2double(image3);

[m,n]=size(image3);

newimage_robert=zeros(size(image3));

threshold=36;

%Sobel

L(1:m,1:n)=0;

for i=1:m-3;

    for j=1:m-3;

        r1=-1*b(i,j)+-2*b(i,j+1)+-
1*b(i,j+2)+0+0+0+1*b(i+2,j)+2*b(i+2,j+1)+1*b(i+2,j+2);

        L(i,j)=r1*r1;

    end

end

M(1:m,1:n)=0;

for i=1:m-3;
```

```
    for j=1:m-3;
        r2=-1*b(i,j)+0+1*b(i,j+2)-2*b(i+1,j)+0+2*b(i+1,j+2)-1*b(i+2,j)+0+1*b(i+2,j+2);
        M(i,j)=r2*r2;
    end
end
for i=1:m-3;
    for j=1:m-3;
        gradient =sqrt(L(i,j)+M(i,j));
        newimage_robert(i,j)= gradient;
    end
end
figure;
imshow(newimage_robert);
imwrite(newimage_robert,'Sobel.bmp')
figure;
Sobel = imread('Sobel.bmp');
[m,n]=size(Sobel);
for i=1:m
    for j=1:n
        if Sobel(i,j)>threshold
          Sobel(i,j)=0;
        else
          Sobel(i,j)=1;
        end
    end
end
imshow(uint8(Sobel)*255);
imwrite(uint8(Sobel)*255,'Sobel_thres.bmp')
end
```

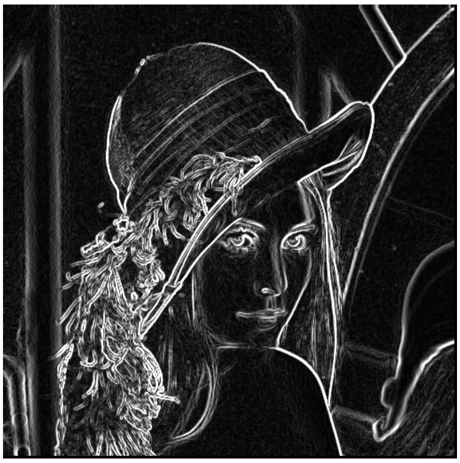
Result



Original image

Edge

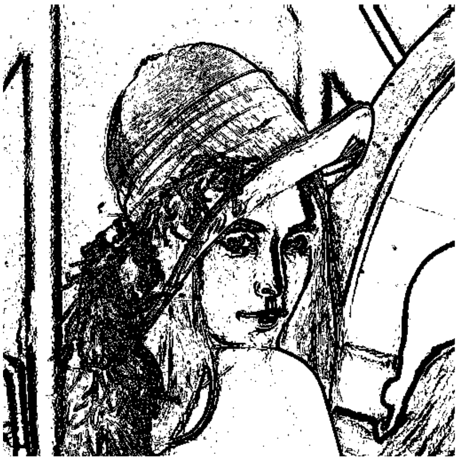
operator



Edge image

利用threshold = 36

來取得二值化影像。



Sobel's Operator Edge image

* Frei and Chen′s Gradient Operator Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Frei and Chen's Gradient Operator with threshold 30 : two 3X3 masks in row column direction

$$3\times3\ masks:\ f_1 = \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}\ and\ f_2 = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$$

$$gradient\ magnitude:\ \sqrt{{f_1}^2 + {f_2}^2} > Threshold$$

利用每點影像的像素值和 mask 分別相乘，達到邊緣偵測的結果值，在依照 gradient 定義把影像值寫回入新的影像矩陣中。利用『im2double』將原本 Lena 影像中的每點像素轉換成 doubles，方便和 mash 數值做運算。最後的 threshold 判斷將影像變成『二值化』的步驟即可求得最後結果。這裡和上一個 operator 運用到的概念幾乎一樣，唯一差別在 mash 的值。

Source code

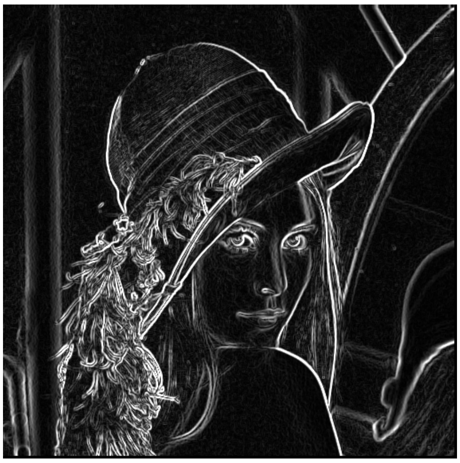
```
%Frei and Chen's Gradient Operator
function FCG = Frei_and_Chen(image4,threshold)
image4 = imread('lena.bmp');

b = im2double(image4);
[m,n] = size(image4);
newimage_robert = zeros(size(image4));
threshold = 30;

% Frei and Chen's Gradient Operator
L(1:m,1:n) = 0;
for I = 1 : m-3;
    for j = 1 : m-3;
        r1=-1*b(i,j)-sqrt(2)*b(i,j+1)-
1*b(i,j+2)+0+0+0+1*b(i+2,j)+sqrt(2)*b(i+2,j+1)+1*b(i+2,j+2);
        L(i,j) = r1 * r1;
    end
end
M(1:m,1:n) = 0;
for I = 1 : m-3;
    for j = 1 : m-3;
        r2 = -1*b(i,j)+0+1*b(i,j+2)-
sqrt(2)*b(i+1,j)+0+sqrt(2)*b(i+1,j+2)-1*b(i+2,j)+0+1*b(i+2,j+2);
        M(i,j) = r2*r2;
    end
end
for i=1:m-3;
    for j=1:m-3;
        gradient =sqrt(L(i,j)+M(i,j));
        newimage_robert(i,j)= gradient;
    end
end
figure;
imshow(newimage_robert);
imwrite(newimage_robert,'FCG.bmp')
figure;
FCG=imread('FCG.bmp');
[m,n]=size(FCG);

for i=1:m
    for j=1:n
        if FCG(i,j)>threshold
          FCG(i,j)=0;
        else
          FCG(i,j)=1;
        end
    end
end
imshow(uint8(FCG)*255);
imwrite(uint8(FCG)*255,'FCG_thres.bmp')
end
```

Result

Original image

Edge

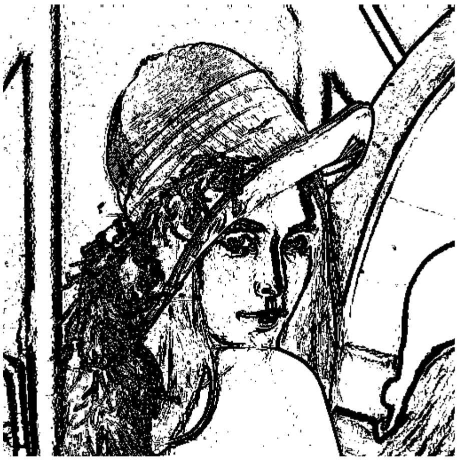
operator



Edge image

利用threshold = 30

來取得二值化影像。



FCG's Operator Edge image

* Kirsch′s Compass Operator Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Kirsch’s Compass Operator with threshold 135 : eight 3*3 compass template edge masks

$$\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix}\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix}\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$$

$$\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix}\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix}\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix}\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}$$

$$gradient\ magnitude：\ g = \underset{n=0,\ldots,7}{max}\, k_n$$

這裡和上一個 operator 運用到的概念不一樣，這裡將輸入 8 個 3*3 的 Mask 矩陣。Gradient 不再是 mask 乘上矩陣後值平方再開根號。而是將這八個 mask 分別去運算後找出最大的值(max)。

我利用到『imfilter』來分別將八的 mask 去和影像 lena 就運算;再利用『max』函式將運算最大值寫入新的影像矩陣中當作我們的輸出像素值。

Source code

```
function ks = Kirsch(image5,threshold)

image5 = imread('lena.bmp');

image5= im2double(image5);

threshold=135;

m = zeros(3,3,8);

m(:,:,1) = [-3 -3 5; -3 0 5; -3 -3 5];

m(:,:,2) = [-3 5 5; -3 0 5; -3 -3 -3];

m(:,:,3) = [5 5 5; -3 0 -3; -3 -3 -3];

m(:,:,4) = [5 5 -3; 5 0 -3; -3 -3 -3];

m(:,:,5) = [5 -3 -3; 5 0 -3; 5 -3 -3];
```

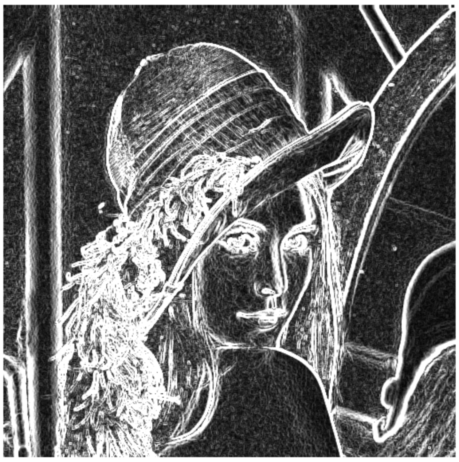
```
m(:,:,6) = [-3 -3 -3; 5 0 -3; 5 5 -3];
m(:,:,7) = [-3 -3 -3; -3 0 -3; 5 5 5];
m(:,:,8) = [-3 -3 -3; -3 0 5; -3 5 5];
Am=zeros(size(image5,1), size(image5,2),8);
for i=1:8
   Am(:,:,i) = imfilter(image5,m(:,:,i));
end
Ak = max(Am,[],3);
figure;
imshow(Ak);
imwrite(Ak,'Kirsch.bmp')
figure;
Kirsch=imread('Kirsch.bmp');
[m,n]=size(Kirsch);

for i=1:m
   for j=1:n
       if Kirsch(i,j)>threshold
         Kirsch(i,j)=0;
       else
         Kirsch(i,j)=1;
       end
   end
end
imshow(uint8(Kirsch)*255);
imwrite(uint8(Kirsch)*255,'Kirsch_thres.bmp')
end
```

Result

Original image

Edge

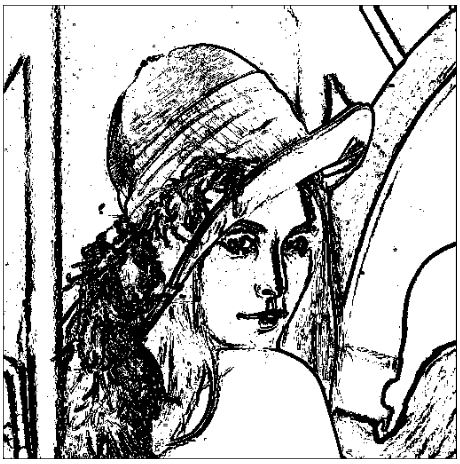
operator



Edge image

利用threshold = 135

來取得二值化影像。



Kirsch's Operator Edge image

* Robinson’s Compass Operator Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Robinson's Compass Operator with threshold 45 : eight 3*3 compass template edge masks

$$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

$$gradient\ magnitude : \ g = \underset{n=0,\dots,7}{max}\ k_n$$

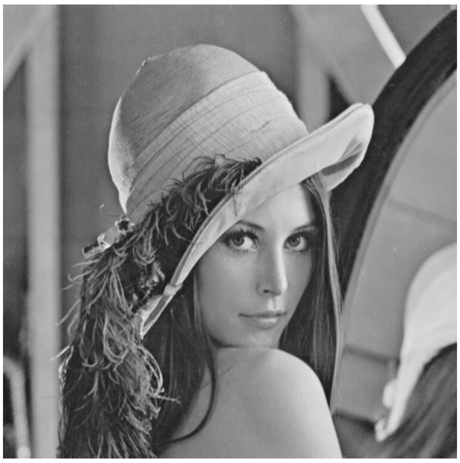
這裡和上一個 operator 運用到的概念一樣，這裡一樣輸入8個 3*3的 Mask 矩陣。將這八個 mask 分別去運算後找出最大的值(max)，作為 gradient。

Source code

```
function Robins = Robinson(image6,threshold)


image6= imread('lena.bmp');

image6 = im2double(image6);

threshold=45;


m = zeros(3,3,8);

m(:,:,1) = [-1 0 1 ; -2 0 2; -1 0 1];

m(:,:,2) = [0 1 2 ;-1 0 1 ; -2 -1 0];

m(:,:,3) = [1 2 1 ; 0 0 0 ; -1 -2 -1];

m(:,:,4) = [2 1 0 ; 1 0 -1 ;0 -1 -2];

m(:,:,5) = [1 0 -1 ; 2 0 -2 ;1 0 -1];

m(:,:,6) = [0 -1 -2;1 0 -1 ; 2 1 0];
```

```
m(:,:,7) = [-1 -2 -1 ;0 0 0 ; 1 2 1];
m(:,:,8) = [-2 -1 0 ; -1 0 1 ; 0 1 2];

Am=zeros(size(image6,1), size(image6,2),8);
for i=1:8
   Am(:,:,i) = imfilter(image6,m(:,:,i));
end
Ak = max(Am,[],3);
%n = 255 / (max(Ak(:)) - min(Ak(:)));
%pic = uint8(n * Ak);

figure;
imshow(Ak);
imwrite(Ak,'Robinson.bmp')
figure;
Robinson=imread('Robinson.bmp');
[m,n]=size(Robinson);

for i=1:m
   for j=1:n
       if Robinson(i,j)>threshold
         Robinson(i,j)=0;
       else
         Robinson(i,j)=1;
       end
   end
end
imshow(uint8(Robinson)*255);
imwrite(uint8(Robinson)*255,'Robinson_thres.bmp')
end
```

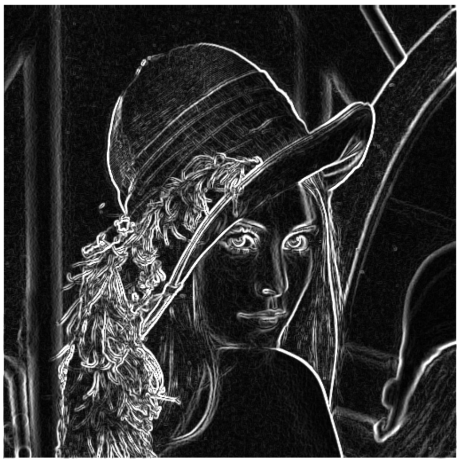
Result



Original image

Edge

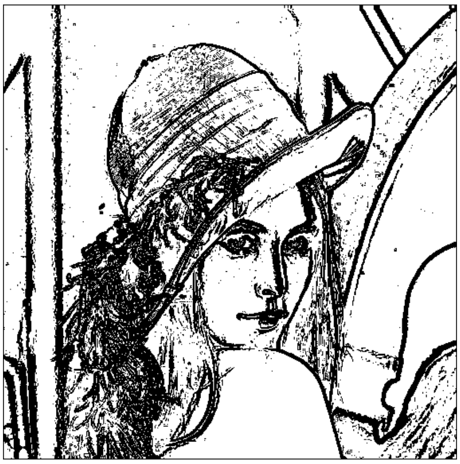
operator



Edge image

利用threshold = 45

來取得二值化影像。



Robinson's Operator Edge image

* Nevatia-Babu 5x5 Operator Concept *

When processing the gradient image, there should be a value which you used for zero crossing. -> Nevatia-Babu 5x5 Operator with threshold 155 : eight 3*3 compass template edge masks

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 0 & 0 & 0 \\ -100 & -100 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix} \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 78 & -32 \\ 100 & 92 & 0 & -92 & -100 \\ 32 & -78 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

0度 30度

$$\begin{bmatrix} 100 & 100 & 100 & 32 & -100 \\ 100 & 100 & 92 & -78 & -100 \\ 100 & 100 & 0 & -100 & -100 \\ 100 & 78 & -92 & -100 & -100 \\ 100 & -32 & -100 & -100 & -100 \end{bmatrix} \qquad \begin{bmatrix} -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \end{bmatrix}$$

60度 -90度

$$\begin{bmatrix} -100 & 32 & 100 & 100 & 100 \\ -100 & -78 & 92 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & -92 & 78 & 100 \\ -100 & -100 & -100 & -32 & 100 \end{bmatrix} \qquad \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ -32 & 78 & 100 & 100 & 100 \\ -100 & -92 & 0 & 92 & 100 \\ -100 & -100 & -100 & -78 & 32 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

-60度 -30度

$$gradient\ magnitude：g = \max_{n=0,\ldots,5} N_n$$

這裡和上一個 operator 運用到的概念一樣，這裡改成輸入 6 個 5*5 的 Mask 矩陣。將這六個 mask 分別去運算後找出最大的值(max)，作為 gradient。

這裡比較麻煩和上面不一樣，需要加上下面的程式碼，才能運作。

『n =(max(Ak(:)))/255; pic = uint8(n*Ak);』

(因為這裡的 gradient 和 contour direction 差 90 度。)

Source code

```
function Nev = Nevati(image7,threshold)
image7 = imread('lena.bmp');
image7 = im2double(image7);
threshold=155;
m = zeros(5,5,6);
m(:,:,1) = [100 100 100 100 100 ; 100 100 100 100 100; 0 0 0 0 0 ;-
100 -100 -100 -100 -100 ;-100 -100 -100 -100 -100];
m(:,:,2) = [100 100 100 100 100 ; 100 100 100 78 -32; 100 92 0 -92 -
100 ;32 -78 -100 -100 -100 ;-100 -100 -100 -100 -100];
m(:,:,3) = [100 100 100 32 -100 ; 100 100 92 -78 -100; 100 100 0 -100
-100 ;100 78 -92 -100 -100 ;100 -32 -100 -100 -100];
m(:,:,4) = [-100 -100 0 100 100 ; -100 -100 0 100 100; -100 -100 0
100 100 ;-100 -100 0 100 100 ;-100 -100 0 100 100];
m(:,:,5) = [-100 32 100 100 100 ; -100 -78 92 100 100; -100 -100 0
100 100 ;-100 -100 -92 78 100 ;-100 -100 -100 -32 100];
m(:,:,6) = [100 100 100 100 100 ; -32 78 100 100 100; -100 -92 0 92
100 ;-100 -100 -100 -78 32 ;-100 -100 -100 -100 -100];
Am=zeros(size(image7,1),size(image7,2),6);
for i=1:6
    Am(:,:,i) = imfilter(image7,m(:,:,i),'same');
end
Ak = max(Am,[],3);
```

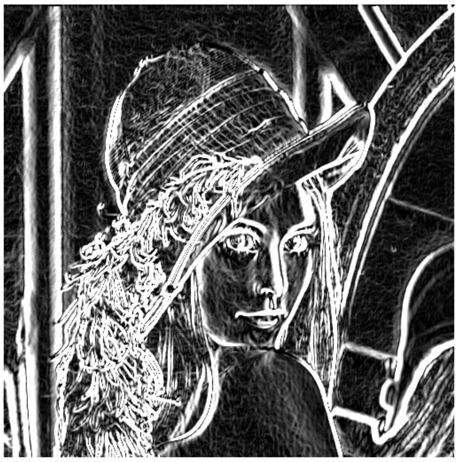
```
n =(max(Ak(:)))/255;

pic = uint8(n*Ak);

figure;

imshow(pic);

imwrite(pic,'Nevatin.bmp')

pic

figure;

Nevatin = imread('Nevatin.bmp');

[m,n]=size(Nevatin);

for i=1:m

   for j=1:n

      if Nevatin(i,j)>threshold

         Nevatin(i,j)=0;

      else

         Nevatin(i,j)=1;

      end

   end

end

imshow(uint8(Nevatin)*255);

imwrite(uint8(Nevatin)*255,'Nevatin_thres.bmp')

end
```

Result

Original image

Edge

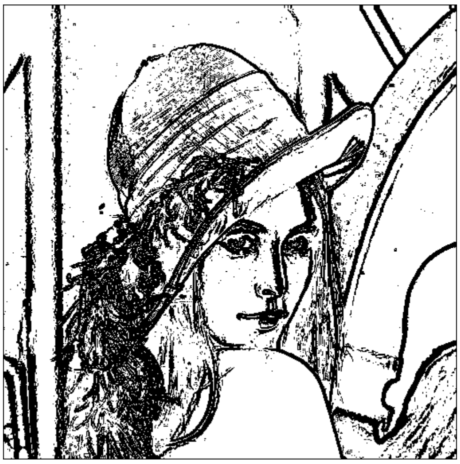
operator



Edge image

利用threshold = 155

來取得二值化影像。



Nevatin ‘s Operator Edge image