Computer vision 1

Homework 10

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 10

R05942060 蘇宛琳

Question :

Write the following programs to detect edge:

 Zero-crossing on the following four types of images to get edge images (choose proper thresholds), p.349

 Laplacian

 minimum-variance Laplacian

 Laplacian of Gaussian

 Difference of Gaussian, (use tk to generate D.O.G.)

dog (inhibitory , excitatory , kernel size = 11)

**\* Laplacian Mask Type1 kernel Concept \***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | x | x | x | x | x | x | x | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  | Lena |  |  |  | x |
|  |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x | x | x | x | x | x | x | x | x |

這裡用運用到的概念是:首先需要將 Kernel(這裡的 3\*3 值為 Mask 之值)放置 lean 圖最邊緣像素位置上，由於原點中心在(2,2)位置，因此會有一半的 Kernel 是跑出去 此 lean 圖的範圍，因此需要將跑出去的像素位置範圍接補零(這裡以 x 代替表示沒有數值)，因此需要補 lean 四周圍各多一行為零(x)的矩陣(如上圖)。

原本像素和 Mask 之間位置對應的值進行相乘(利用 im2double 可以取出影像的每點像素數值)。將輸出的影像在進行門檻值的選取，則可以求得最終的影像圖。

**Source code**

function lap1=lapaican\_1(image1,threshold)

image1=imread('lena.bmp');

b=im2double(image1);

[m,n]=size(image1);

newimage\_lap1=zeros(size(image1));

threshold=15;

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

for i=1:m-3;

for j=1:m-3;

L(i,j)=0+1\*b(i,j+1)+0+1\*b(i+1,j)-4\*b(i+1,j+1)+1\*b(i+1,j+2)+0+1\*b(i+2,j+1)+0;

end

end

for i=1:m-3;

for j=1:m-3;

newimage\_robert(i,j)= L(i,j);

end

end

figure;

imshow(newimage\_robert);

imwrite(newimage\_robert,'lapl.bmp')

figure;

lapl = imread('lapl.bmp');

[m,n]=size(lapl);

for i=1:m

for j=1:n

if lapl(i,j)>threshold

lapl(i,j)=0;

else

lapl(i,j)=1;

end

end

end

imshow(uint8(lapl)\*255);

imwrite(uint8(lapl)\*255,'lapl\_thres.bmp')

end

**Result**

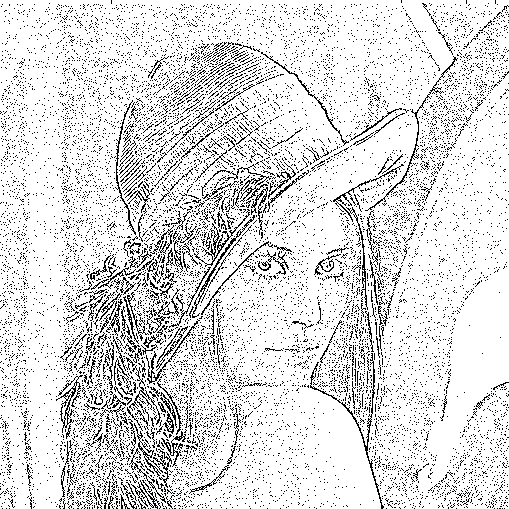
 

Laplacian Mask Type1

Original image Edge image

利用threshold = 15

來取得二值化影像。



Laplacian type1 Edge image

**\* Laplacian Mask Type2 kernel Concept \***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | x | x | x | x | x | x | x | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  | Lena |  |  |  | x |
|  |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x | x | x | x | x | x | x | x | x |

這裡用運用到的概念是:首先需要將 Kernel(這裡的 3\*3 值為 Mask 之值)放置 lean 圖最邊緣像素位置上，由於原點中心在(2,2)位置，因此會有一半的 Kernel 是跑出去 此 lean 圖的範圍，因此需要將跑出去的像素位置範圍接補零(這裡以 x 代替表示沒有數值);因此需要補 lean 四周圍各多一行為零(x)的矩陣(如上圖)。

原本像素和 Mask 之間位置對應的值進行相乘(利用 im2double 可以取出影像的每點像素數值)。將輸出的影像在進行門檻值的選取，則可以求得最終的影像圖。

**Source code**

function lap2=lapaican\_2(image2,threshold)

image2=imread('lena.bmp');

b=im2double(image2);

[m,n]=size(image2);

newimage\_lap2=zeros(size(image2));

threshold=15;

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

for i=1:m-3;

for j=1:m-3;

L(i,j)=1/3\*(1\*b(i,j)+1\*b(i,j+1)+1\*b(i,j+2)+1\*b(i+1,j)-8\*b(i+1,j+1)+1\*b(i+1,j+2)+1\*b(i+2,j)+1\*b(i+ 2,j+1)+1\*b(i+2,j+2));

end

end

for i=1:m-3;

for j=1:m-3;

newimage\_lap2(i,j)= L(i,j);

end

end

figure;

imshow(newimage\_lap2);

imwrite(newimage\_lap2,'lap2.bmp')

figure;

lap2 = imread('lap2.bmp');

[m,n]=size(lap2);

for i=1:m

for j=1:n

if lap2(i,j)>threshold

lap2(i,j)=0;

else

lap2(i,j)=1;

end

end

end

imshow(uint8(lap2)\*255);

imwrite(uint8(lap2)\*255,'lap2\_thres.bmp')

end

**Result**

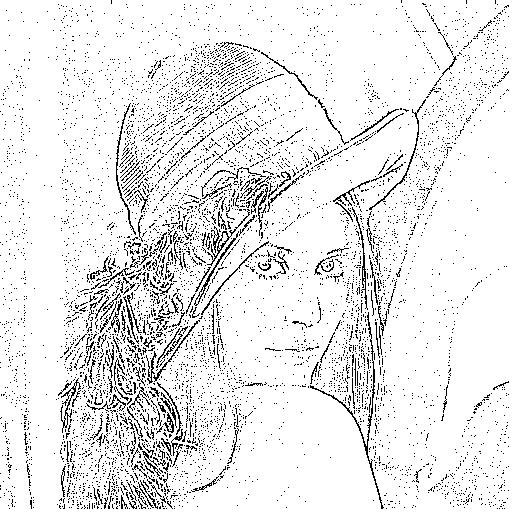
 

Laplacian Mask Type2

Original image Edge image

利用threshold = 15

來取得二值化影像。



Laplacian type2 Edge image

**\* Minimum variance Laplacian kernel Concept \***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | x | x | x | x | x | x | x | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  | Lena |  |  |  | x |
|  |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x | x | x | x | x | x | x | x | x |

這裡用運用到的概念是:首先需要將 Kernel(這裡的 3\*3 值為 Mask 之值)放置 lean 圖最邊緣像素位置上，由於原點中心在(2,2)位置，因此會有一半的 Kernel 是跑出去 此 lean 圖的範圍，因此需要將跑出去的像素位置範圍接補零(這裡以 x 代替表示沒有數值);因此需要補 lean 四周圍各多一行為零(x)的矩陣(如上圖)。

原本像素和 Mask 之間位置對應的值進行相乘(利用 im2double 可以取出影像的每點像素數值)。將輸出的影像在進行門檻值的選取，則可以求得最終的影像圖。

**Source code**

function mvl=MVL(image3,threshold)

image3=imread('lena.bmp');

b=im2double(image3);

[m,n]=size(image3);

newimage\_robert=zeros(size(image3));

threshold=15;

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

for i=1:m-3;

for j=1:m-3;

L(i,j)=1/3\*(2\*b(i,j)-1\*b(i,j+1)+2\*b(i,j+2)-1\*b(i+1,j)-4\*b(i+1,j+1)-1\*b(i+1,j+2)+2\*b(i+2,j)-1\*b(i+2,j +1)+2\*b(i+2,j+2));

end

end

for i=1:m-3;

for j=1:m-3;

newimage\_mvl(i,j)= L(i,j);

end

end

figure;

imshow(newimage\_mvl);

imwrite(newimage\_mvl,'mvl.bmp')

figure;

mvl = imread('mvl.bmp');

[m,n]=size(mvl);

for i=1:m

for j=1:n

if mvl(i,j)>threshold mvl(i,j)=0;

else

mvl(i,j)=1;

end

end

end

imshow(uint8(mvl)\*255);

imwrite(uint8(mvl)\*255,'mvl\_thres.bmp')

end

**Result**

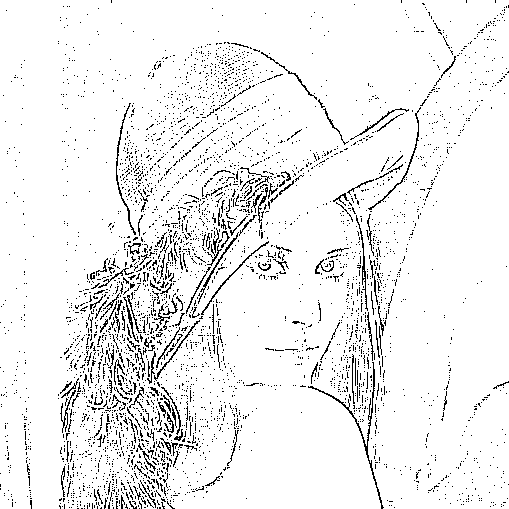
 

Minimum variance Laplacian

Original image Edge image

利用threshold = 15

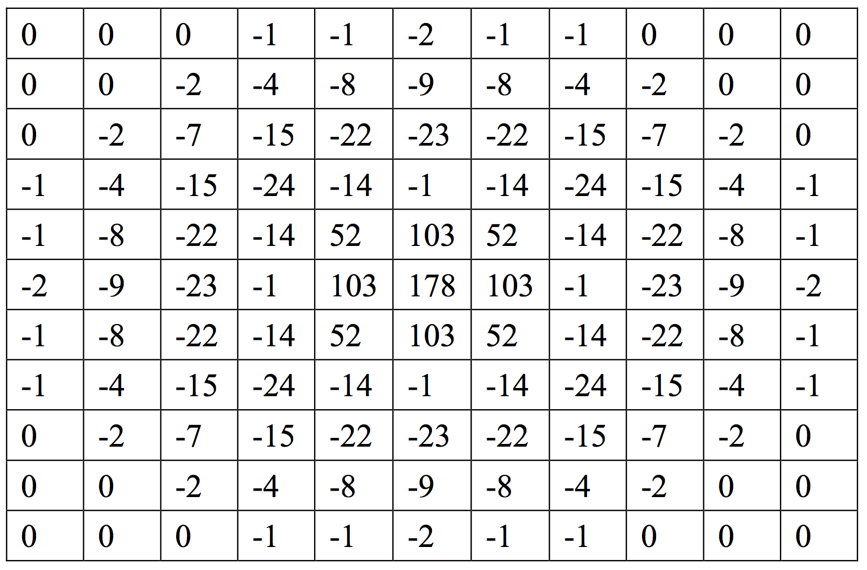
來取得二值化影像。



Minimum variance Laplacian Edge image

**\* Laplacian of Gaussian kernel Concept \***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | x | x | x | x | x | x | x | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  | Lena |  |  |  | x |
|  |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x | x | x | x | x | x | x | x | x |



這裡用運用到的概念是:首先需要將 Kernel(這裡的 11\*11 值為 Mask 之值)放置 lean 圖最邊緣像素位置上，由於原點中心在(6,6)位置，因此會有一半的 Kernel 是跑 出去此 lean 圖的範圍，因此需要將跑出去的像素位置範圍接補零(這裡以 x 代替表示沒有數值);因此需要補 lean 四周圍各多五行為零(x)的矩陣(如上圖)。

原本像素和 Mask 之間位置對應的值進行相乘。將輸出的影像在進行門檻值的選取， 則可以求得最終的影像圖。『newimage\_log(r-edge,c-edge)=sum(sum(Itemp(r-edge:r+edge,c-edge:c+edge).\*ker));』最後的 newimage\_log 即為邊緣影像;再去利用 threshold 去取得二元影像。

**Source code**

function log=LOG(image4,threshold)

image4=imread('lena.bmp');

%[m,n]=size(image4);

threshold=3000;

ker=[0 0 0 -1 -1 -2 -1 -1 0 0 0;...

0 0 -2 -4 -8 -9 -8 -4 -2 0 0;...

0 -2 -7 -15 -22 -23 -22 -15 -7 -2 0;...

-1 -4 -15 -24 -14 -1 -14 -24 -15 -4 -1;...

-1 -8 -22 -14 52 103 52 -14 -22 -8 -1;...

-2 -9 -23 -1 103 178 103 -1 -23 -9 -2;...

-1 -8 -22 -14 52 103 52 -14 -22 -8 -1;...

-1 -4 -15 -24 -14 -1 -14 -24 -15 -4 -1;...

0 -2 -7 -15 -22 -23 -22 -15 -7 -2 0;...

0 0 -2 -4 -8 -9 -8 -4 -2 0 0;...

0 0 0 -1 -1 -2 -1 -1 0 0 0;];

edge=floor(size(ker,1)/2);

output=ones(size(image4));

newimage\_log=ones(size(image4));

Itemp=double(wextend('2','symw',image4,edge)); % extend image

% Mask

for r=edge+1:size(Itemp,1)-edge

for c=edge+1:size(Itemp,2)-edge

newimage\_log(r-edge,c-edge)=sum(sum(Itemp(r-edge:r+edge,c-edge:c+edge).\*ker));

end

end

figure;

imshow(newimage\_log);

imwrite(newimage\_log,'log.bmp')

% Zero Crossing

zctemp=zeros(size(Itemp));

zctemp=double(wextend('2','symw',newimage\_log,edge));

for r=1:size(image4,1)

for c=1:size(image4,2)

mask=zctemp(r+edge-1:r+edge+1,c+edge-1:c+edge+1);

center=mask(2,2);

neighbors=mask([1 2 3 4 6 7 8 9]);

% condition 1

c1=center<-threshold;

c1=c1\*sum(neighbors>threshold);

% condition 2

c2=center>threshold;

c2=c2\*sum(neighbors<-threshold);

output(r,c)=~(c1||c2);

end

end

figure;

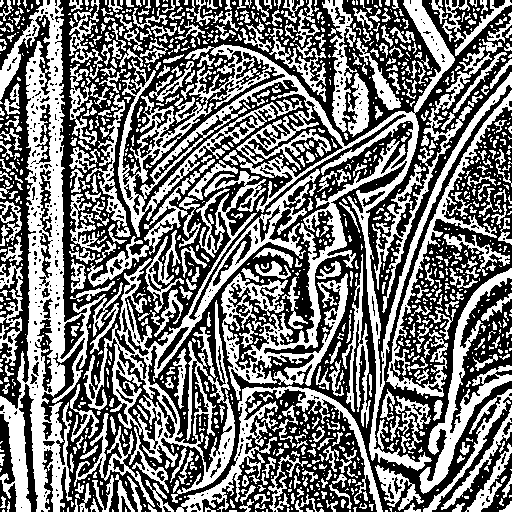
output=uint8(255\*output);

imshow(output);

imwrite(output,'log\_thres.bmp');

end

**Result**

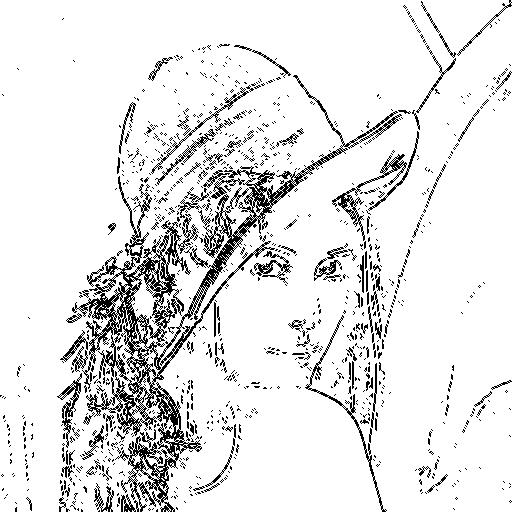
Laplacian of

Gaussian kerneln

Original image Edge image

利用threshold = 3000

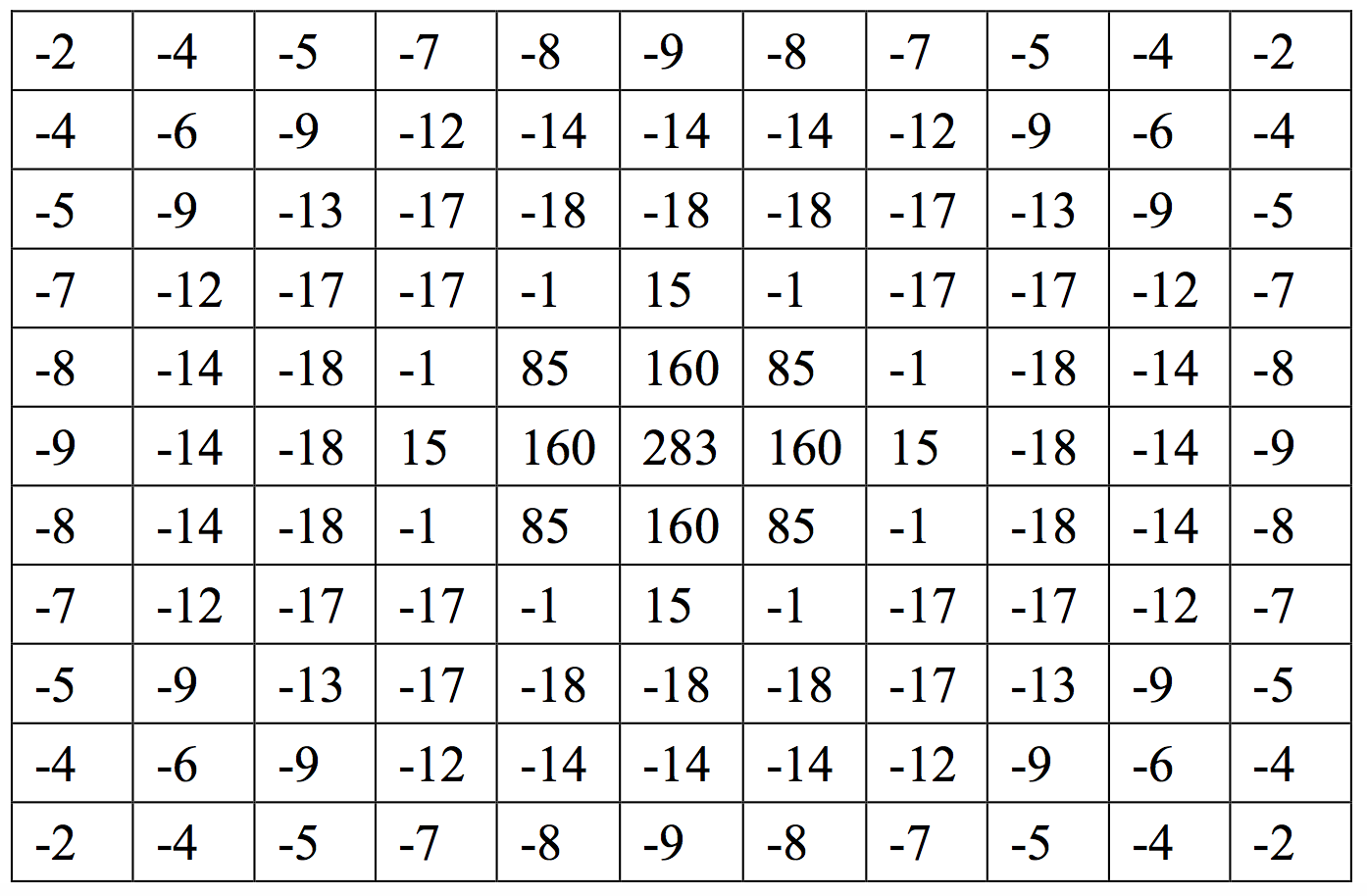
來取得二值化影像。



Laplacian of Gaussian kernel Edge image

**\* Difference of Gaussian kernel Concept \***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | x | x | x | x | x | x | x | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x |  |  |  | Lena |  |  |  | x |
|  |  |  |  |  |  |  |  | x |
| x |  |  |  |  |  |  |  | x |
| x | x | x | x | x | x | x | x | x |



這裡用運用到的概念是:首先需要將 Kernel(這裡的 11\*11 值為 Mask 之值)放置 lean 圖最邊緣像素位置上，由於原點中心在(6,6)位置，因此會有一半的 Kernel 是跑 出去此 lean 圖的範圍，因此需要將跑出去的像素位置範圍接補零(這裡以 x 代替表示沒有數值);因此需要補 lean 四周圍各多五行為零(x)的矩陣(如上圖)。

原本像素和 Mask 之間位置對應的值進行相乘。將輸出的影像在進行門檻值的選取， 則可以求得最終的影像圖。

『newimage\_log(r-edge,c-edge)=sum(sum(Itemp(r-edge:r+edge,c-edge:c+edge).\*ker));』最後的 newimage\_log 即為邊緣影像;再去利用 threshold 去取得二元影像。

\*唯一值得注意的，這個是利用兩個高斯函數進行相減;因此設定兩個高斯函數， 進行相減;得到高斯差!----(Laplacian)

**Source code**

function dog=DOG(image5,threshold)

image5=imread('lena.bmp');

%[m,n]=size(image5);

threshold=1;

% Difference of Gaussian

gaussian1=zeros(11); % Gaussian 1

gaussian2=gaussian1; % Gaussian 2

sig1=1; % sigma 1

sig2=3; % sigma 2

for r=-5:5

for c=-5:5

gaussian1(r+6,c+6)=1/(2\*pi\*sig1^2)\*exp(-(r^2+c^2)/(sig1^2)/2);

gaussian2(r+6,c+6)=1/(2\*pi\*sig2^2)\*exp(-(r^2+c^2)/(sig2^2)/2);

end

end

ker=round(2000\*(gaussian1-gaussian2));

edge=floor(size(ker,1)/2);

output=ones(size(image5));

newimage\_log=ones(size(image5));

Itemp=double(wextend('2','symw',image5,edge)); % extend image

% Mask

for r=edge+1:size(Itemp,1)-edge

for c=edge+1:size(Itemp,2)-edge

newimage\_dog(r-edge,c-edge)=sum(sum(Itemp(r-edge:r+edge,c-edge:c+edge).\*ker));

end

end

figure;

imshow(newimage\_dog);

imwrite(newimage\_dog,'dog.bmp')

% Zero Crossing

zctemp=zeros(size(Itemp));

zctemp=double(wextend('2','symw',newimage\_dog,edge));

for r=1:size(image5,1)

for c=1:size(image5,2)

mask=zctemp(r+edge-1:r+edge+1,c+edge-1:c+edge+1);

center=mask(2,2);

neighbors=mask([1 2 3 4 6 7 8 9]);

% condition 1

c1=center<-threshold;

c1=c1\*sum(neighbors>threshold);

% condition 2

c2=center>threshold;

c2=c2\*sum(neighbors<-threshold);

output(r,c)=~(c1||c2);

end

end

figure;

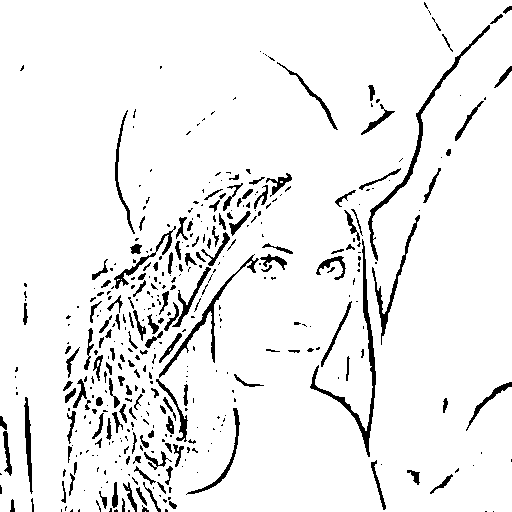
output=uint8(255\*output);

imshow(output);

imwrite(output,'dog\_thres.bmp');

end

**Result**

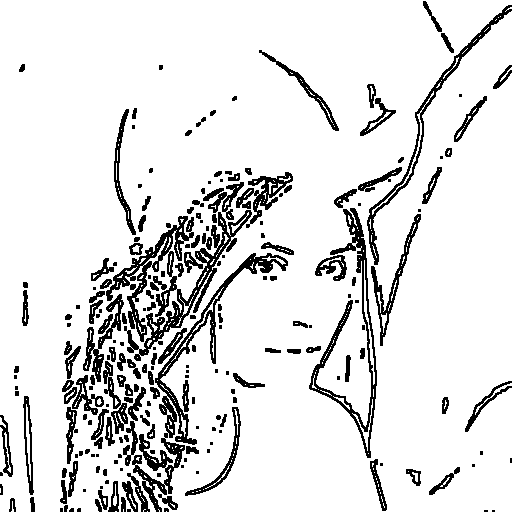
Difference of

Gaussian kernel

Original image Edge image

利用threshold = 1

來取得二值化影像。



Difference of Gaussian kernel Edge image