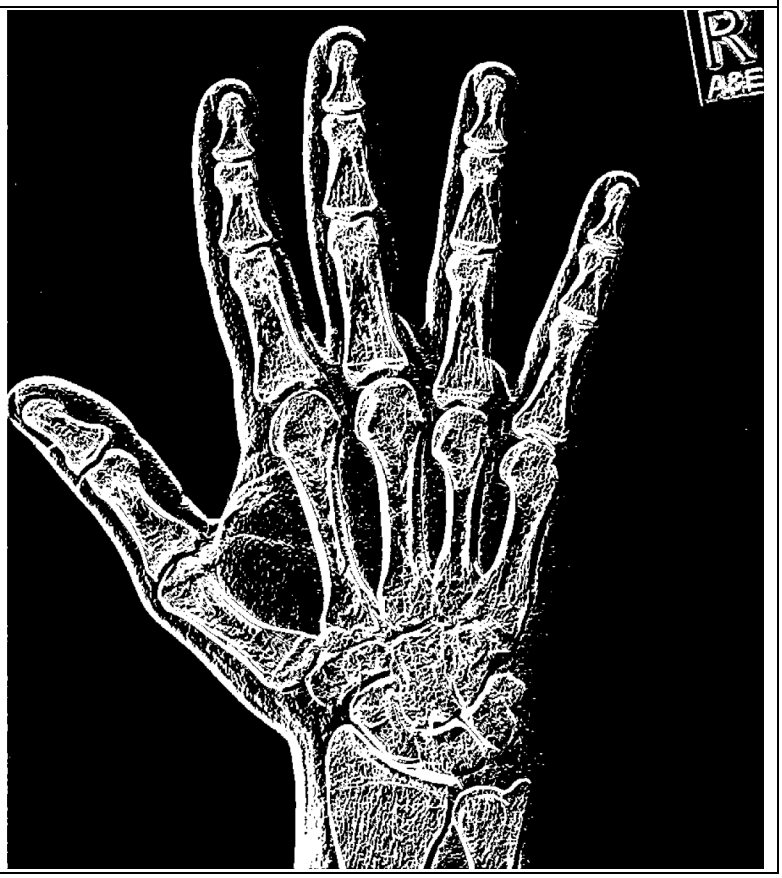
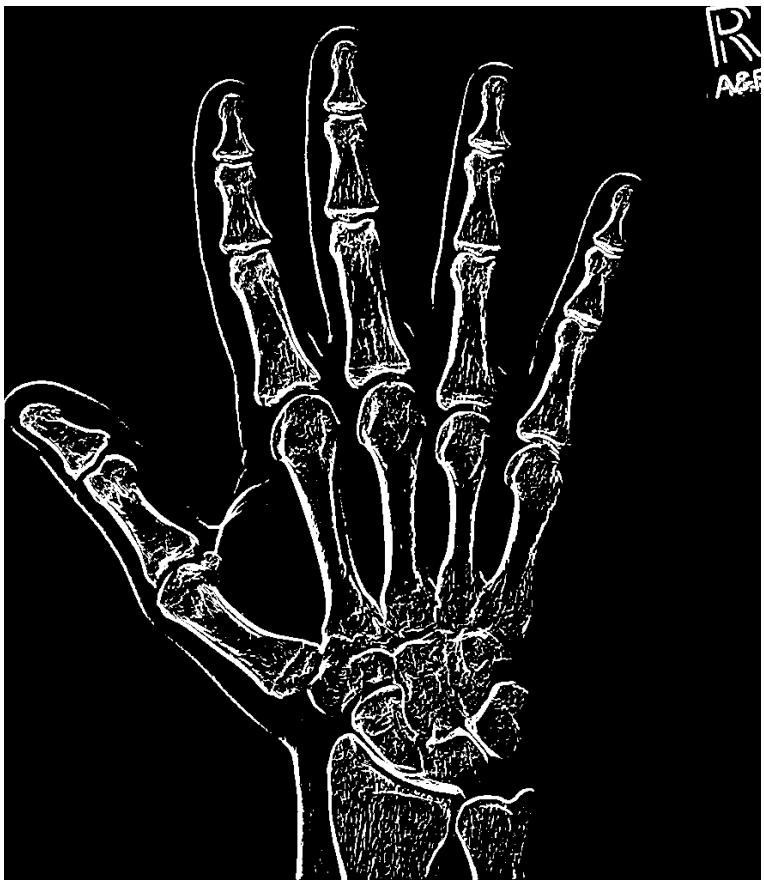
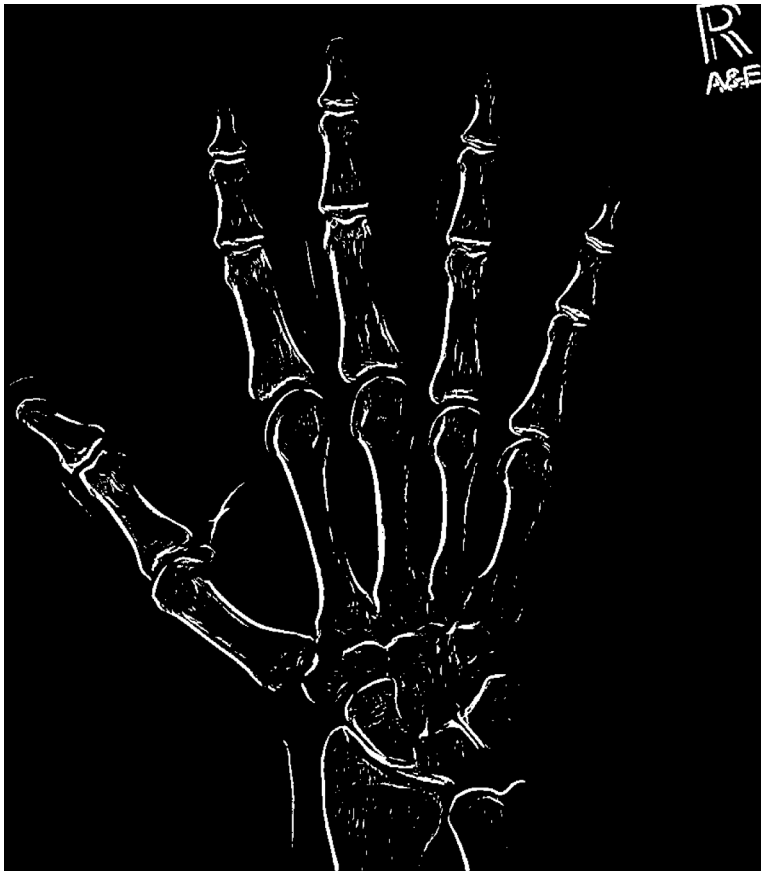
目標：找出影像中的邊緣

方法：參考 **Canny** 的步驟

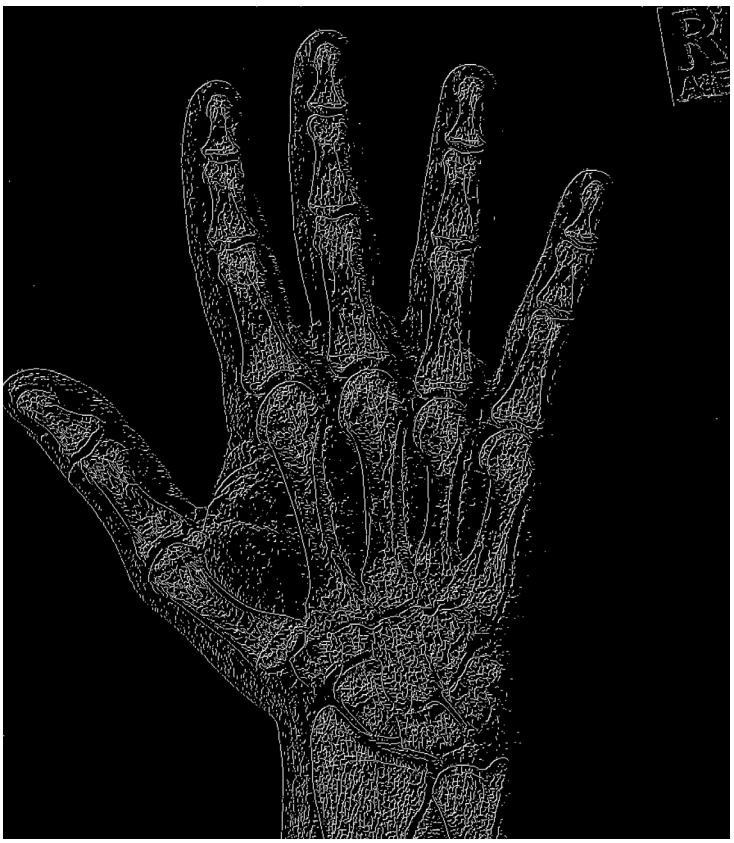
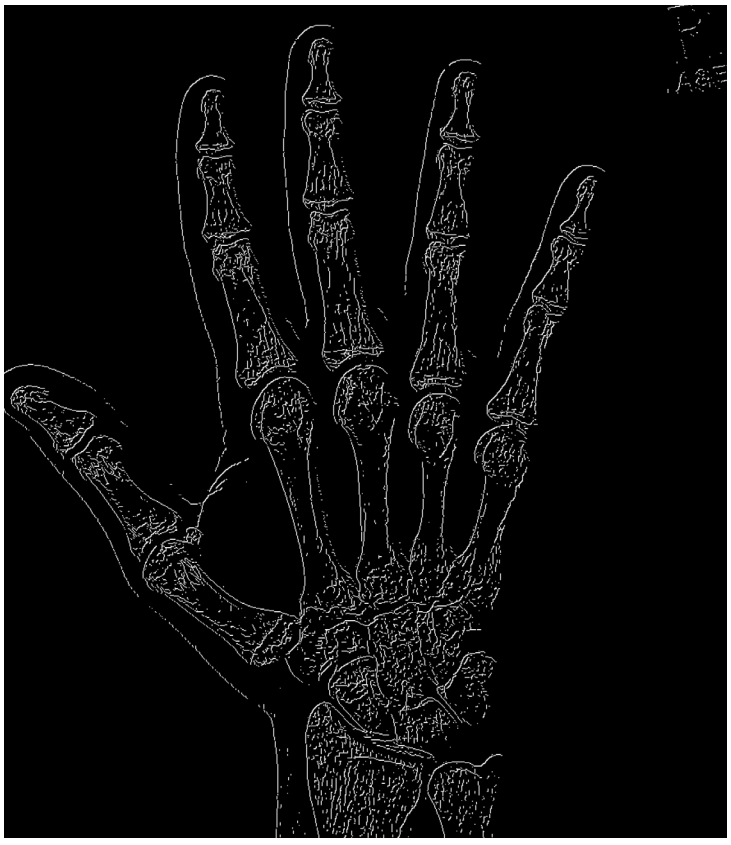
Step 1: Gaussian Blur
Step 2: Find Gradient Using Sobel
Step 3: NMS (Non-maximum suppression)
Step 4: Double Thresholding
Step 5: Hysteresis

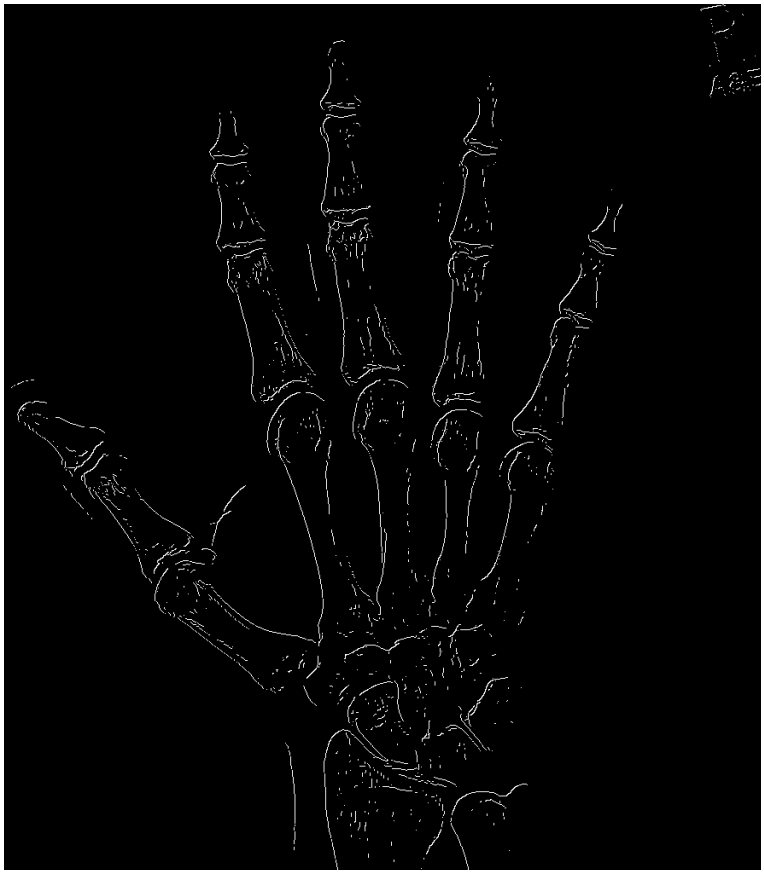
Condition: Without NMS and one threshold

Threshold	Result
10	

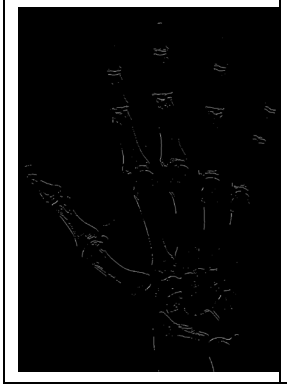
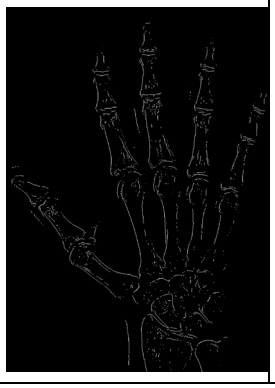
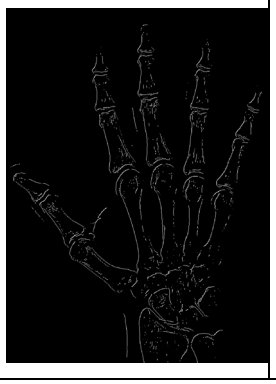
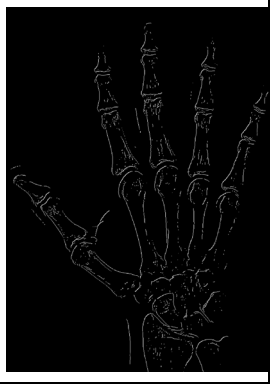
30	
60	

Condition: With NMS and one threshold

Threshold	Result
10	
30	

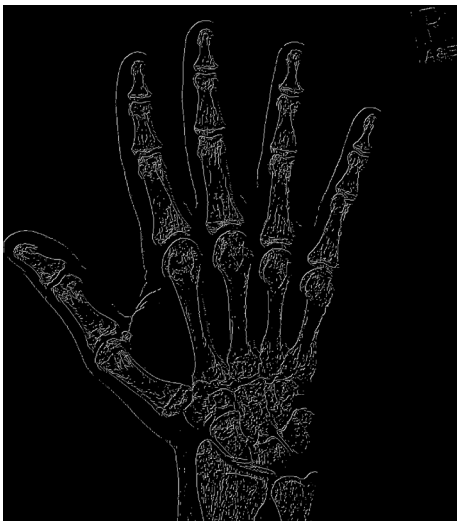
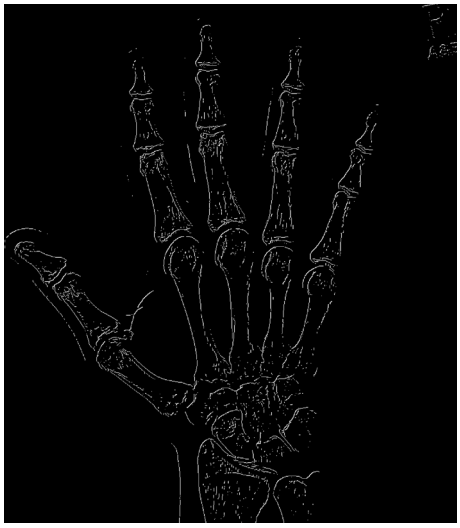
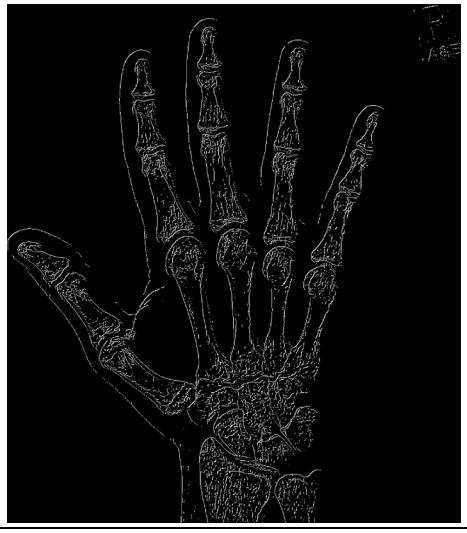
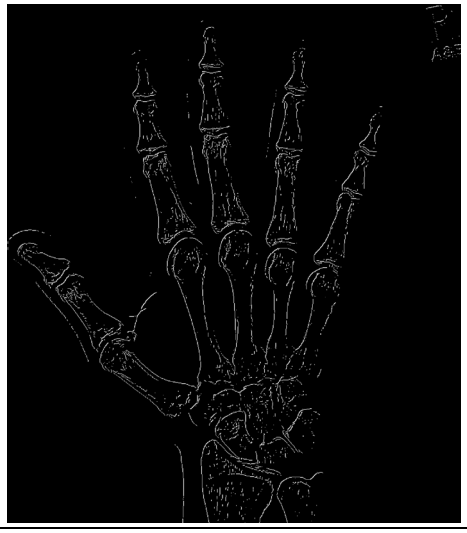
60	

Hysteresis effect (weak edge -> strong edge):

VERTICAL	HORIZON	SLASH	BACK_SLASH
			

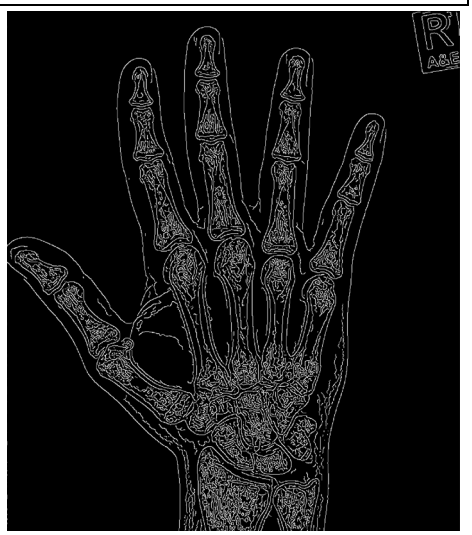
Condition: With NMS and double thresholding and hysteresis

- Columns and rows represent first thresholding and second thresholding respectively.

	30	50
80		
120		

Compare with matlab's canny:

對比 Matlab 的 Canny，實作中只用四個方向的偵測(即上下、左右、斜線、反斜線)，在結果裡手部的右側邊緣都偵測不出來，可能是 Code 的部分有些 Bug。



Reference:

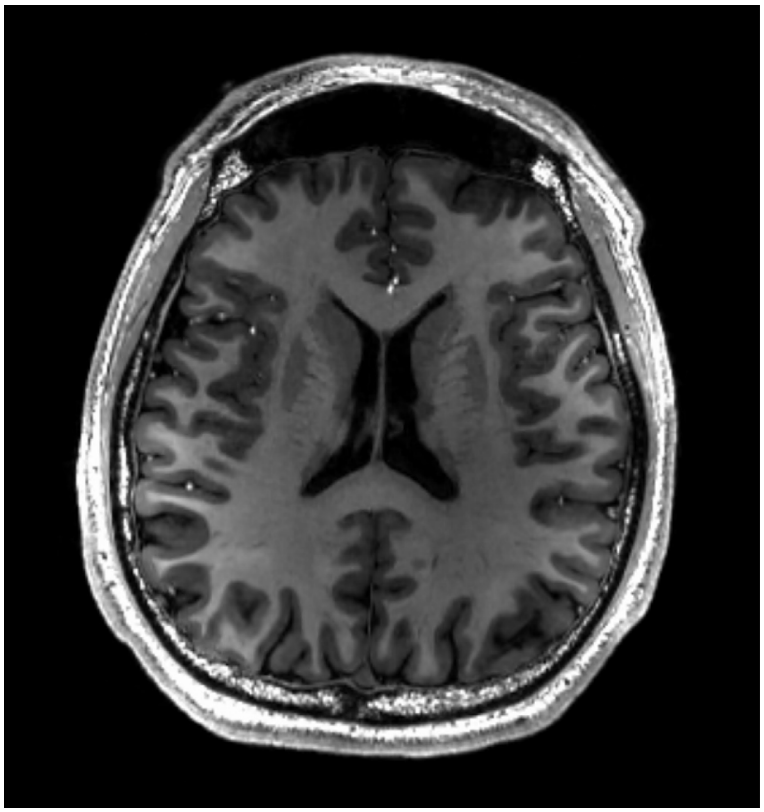
- https://medium.com/@pomelyu5199/canny-edge-detector-%E5%AF%A6%E4%BD%9C-opencv-f7d1a0a57d19

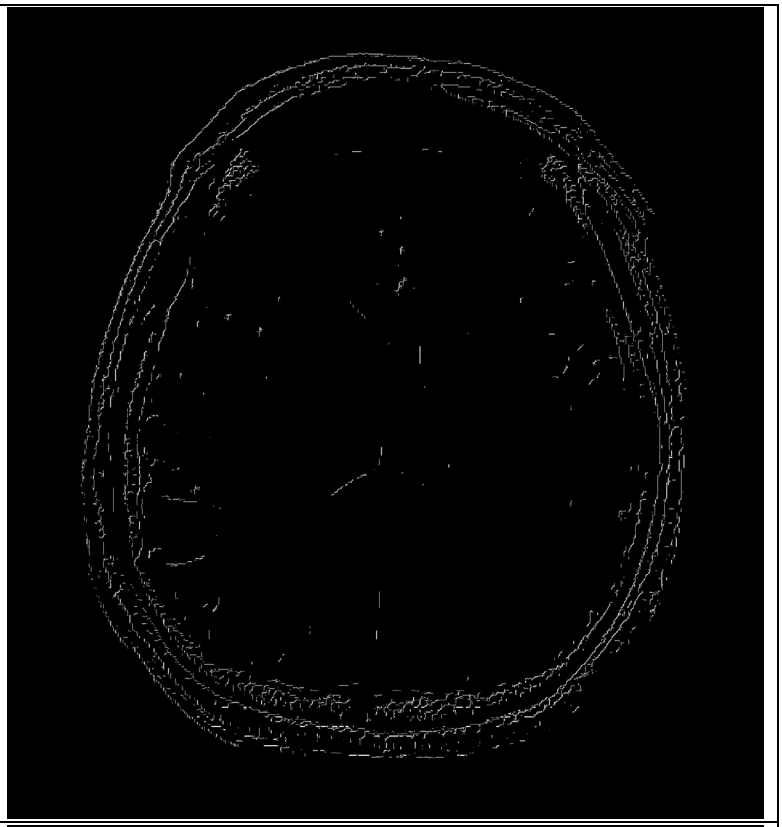
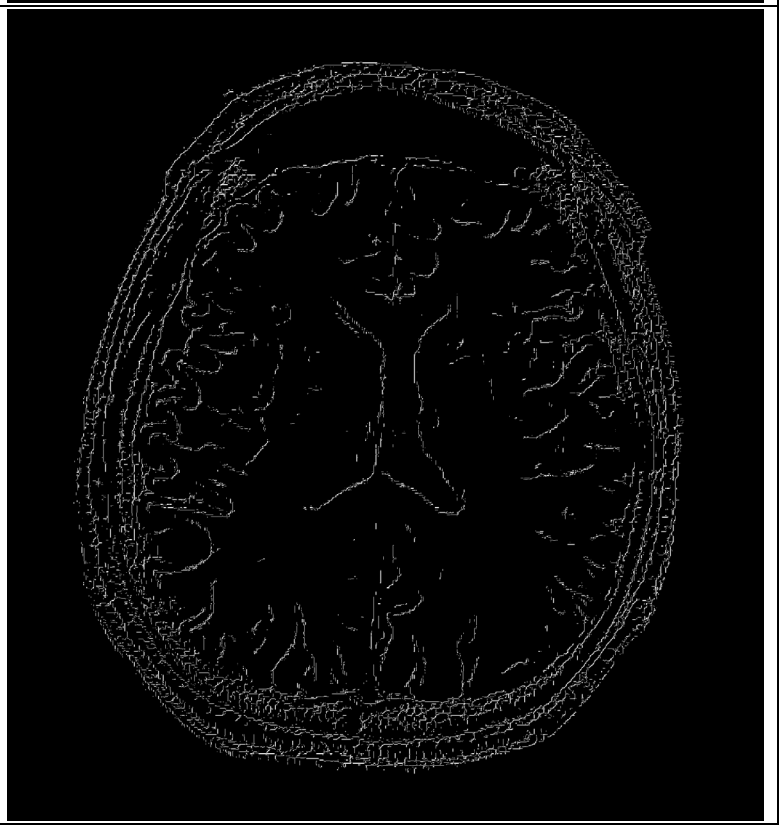
Other Images: teddy.png

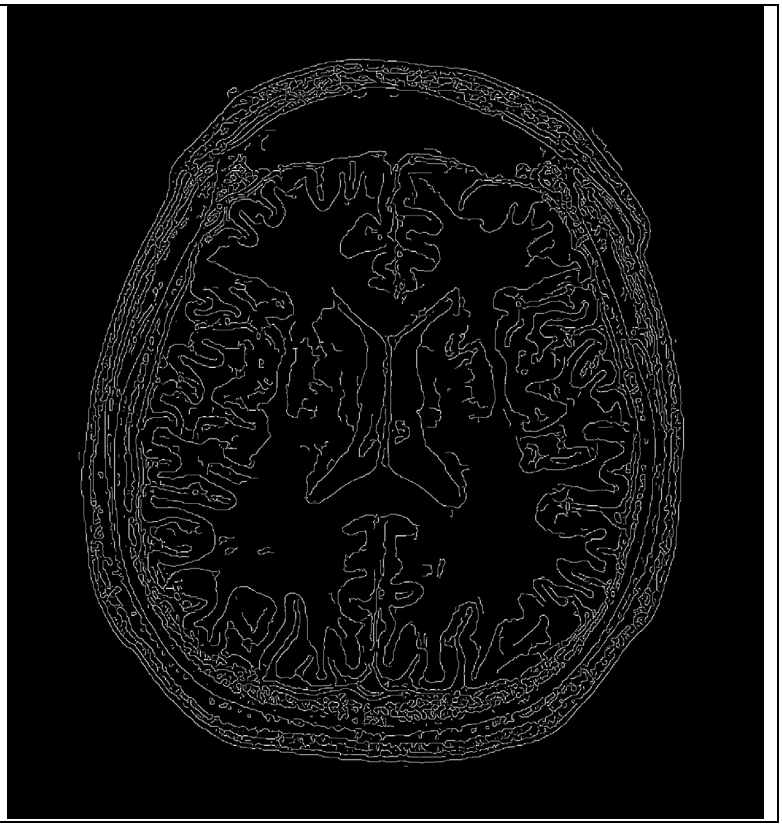
Original	
Matlab's canny	

My canny	

Other Images: mri.jpg

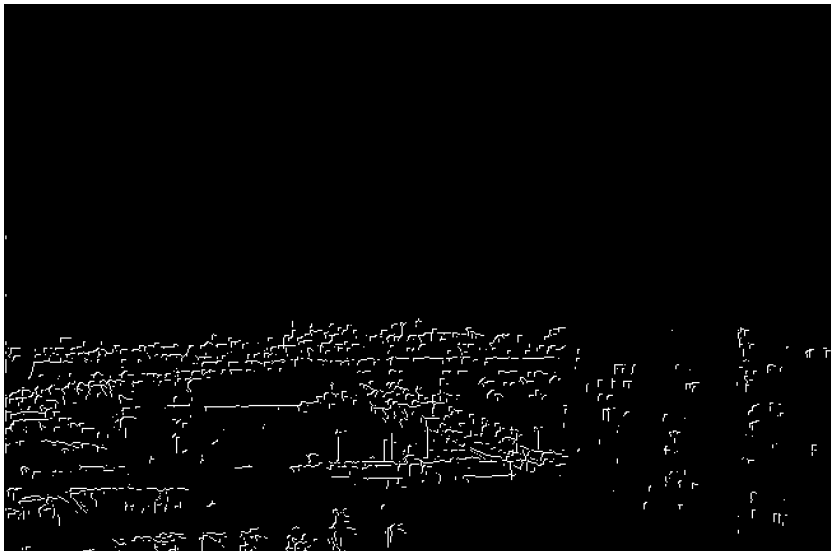
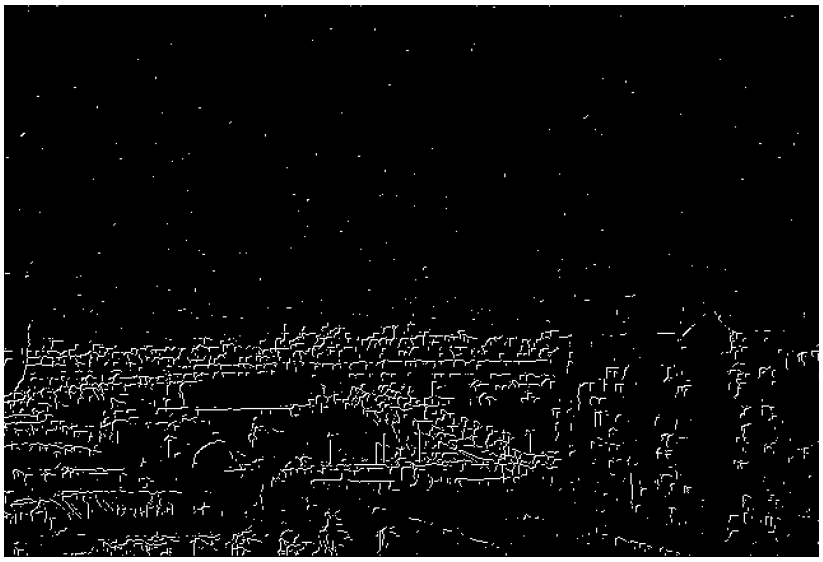
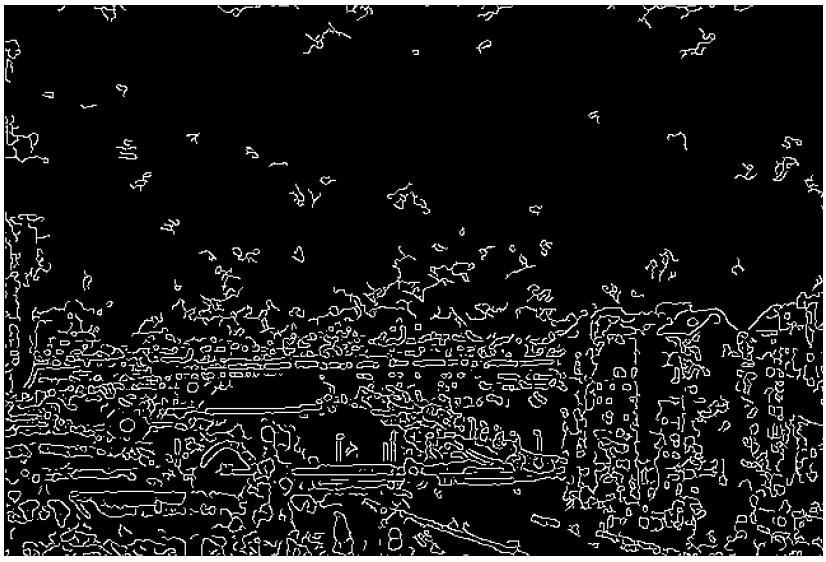
Original	

My canny with one threshold	
My canny with two threshold (Most detail is turned into strong edge from weak edge.)	

Matlab's canny	

Other Images: night_view.png

Original	

My canny with one threshold	
My canny with two threshold (More noise be detected.)	
Matlab's canny	

Filter.m

```
function imgResult = Filter(img, kernel)
    imgResult = img;
    sizeImg = size(img);
    sizeKernel = size(kernel);
    tmpImg = zeros(sizeImg+fix(sizeKernel/2)*2);
    tmpImg(fix(sizeKernel(1)/2)+1:fix(sizeKernel(1)/2)+sizeImg(1),
fix(sizeKernel(2)/2)+1:fix(sizeKernel(2)/2)+sizeImg(2)) = img;
    for row = 1:sizeImg(1)
        for col = 1:sizeImg(2)
            imgResult(row, col) = sum(tmpImg(row:row+sizeKernel(1)-1,
col:col+sizeKernel(2)-1) .* kernel, "all");
        end
    end
end
```

Canny.m

```
function out = Canny(in, threshold1, threshold2, L2gradient)
   %Gaussian Blur
   %Processing========================================================
   StandardDeviation_GaussianBlur = 0.707;
   KernelSize_GaussianBlur = 3;
   [gridX, gridY] = meshgrid(-(KernelSize_GaussianBlur-
1)/2:(KernelSize_GaussianBlur-1)/2,-(KernelSize_GaussianBlur-
1)/2:(KernelSize_GaussianBlur-1)/2);
   Kernel_GaussianBlur =
(1/(2*pi*StandardDeviation_GaussianBlur^2))*exp(1).^(-
((gridX.^2+gridY.^2)/(2*StandardDeviation_GaussianBlur^2)));
   Kernel_GaussianBlur = Kernel_GaussianBlur /
sum(Kernel_GaussianBlur,"all");
   in_After_GaussianBlur = Filter(in, Kernel_GaussianBlur);

   %Sobel Gradient
Direction==========================================================
   KernelSize_Sobel = 3;
   coefficients = [2 20 780 132600];
   [gridX, gridY] = meshgrid(-(KernelSize_Sobel-1)/2:(KernelSize_Sobel-
1)/2,-(KernelSize_Sobel-1)/2:(KernelSize_Sobel-1)/2);
```

```
   %kernel for Vertical
   Kernel_Sobel_Vertical = gridY ./ (gridX .* gridX + gridY .* gridY);
   Kernel_Sobel_Vertical((KernelSize_Sobel+1)/2, (KernelSize_Sobel+1)/2)
= 0;
   Kernel_Sobel_Vertical = Kernel_Sobel_Vertical *
coefficients((KernelSize_Sobel-1)/2);

   %kernel for Horizon
   Kernel_Sobel_Horizon = gridX ./ (gridX .* gridX + gridY .* gridY);
   Kernel_Sobel_Horizon((KernelSize_Sobel+1)/2, (KernelSize_Sobel+1)/2) =
0;
   Kernel_Sobel_Horizon = Kernel_Sobel_Horizon *
coefficients((KernelSize_Sobel-1)/2);

   Gx = double(Filter(in_After_GaussianBlur, Kernel_Sobel_Horizon));
   Gy = double(Filter(in_After_GaussianBlur, Kernel_Sobel_Vertical));

   TAN22_5 = 0.414;
   TAN67_5 = 2.414;
   VERTICAL = 0;
   HORIZON = 1;
   SLASH = 2;
   BACK_SLASH = 3;
   ZERO = 4;

   eta = 10e-6;
   theta = ((Gy./(Gx + eta)));
   thetaDirection = zeros(size(theta));
   thetaDirection(theta < TAN67_5 & theta >= TAN22_5) = SLASH;
   thetaDirection(theta < TAN22_5 & theta >= -TAN22_5) = HORIZON;
   thetaDirection(theta < -TAN22_5 & theta >= -TAN67_5) = BACK_SLASH;

   G = (Gx.*Gx + Gy.* Gy).^0.5;

   %NMS=================================================
   [sDy, sDx] = size(thetaDirection);
   thetaDirection(1,:) = ZERO;
```

```
thetaDirection(sDy,:) = ZERO;
thetaDirection(:, 1) = ZERO;
thetaDirection(:, sDx) = ZERO;

%VERTICAL
direction = thetaDirection==VERTICAL;
direction = direction(2:sDy-1, 2:sDx-1);
G_NM = true(size(G));
pixel = G(2:sDy-1, 2:sDx-1);
v1 = G(1:sDy-2, 2:sDx-1);
v2 = G(3:sDy, 2:sDx-1);
G_NM(2:sDy-1, 2:sDx-1) = direction & (pixel <= v1 | pixel <= v2);
G(G_NM) = 0;

%HORIZON
direction = thetaDirection==HORIZON;
direction = direction(2:sDy-1, 2:sDx-1);
G_NM = true(size(G));
pixel = G(2:sDy-1, 2:sDx-1);
v1 = G(2:sDy-1, 1:sDx-2);
v2 = G(2:sDy-1, 3:sDx);
G_NM(2:sDy-1, 2:sDx-1) = direction & (pixel <= v1 | pixel <= v2);
G(G_NM) = 0;

%SLASH
direction = thetaDirection==SLASH;
direction = direction(2:sDy-1, 2:sDx-1);
G_NM = true(size(G));
pixel = G(2:sDy-1, 2:sDx-1);
v1 = G(3:sDy, 3:sDx);
v2 = G(1:sDy-2, 1:sDx-2);
G_NM(2:sDy-1, 2:sDx-1) = direction & (pixel <= v1 | pixel <= v2);
G(G_NM) = 0;

%BACK_SLASH
direction = thetaDirection==BACK_SLASH;
direction = direction(2:sDy-1, 2:sDx-1);
G_NM = true(size(G));
```

```
   pixel = G(2:sDy-1, 2:sDx-1);
   v1 = G(1:sDy-2, 3:sDx);
   v2 = G(3:sDy, 1:sDx-2);
   G_NM(2:sDy-1, 2:sDx-1) = direction & (pixel <= v1 | pixel <= v2);
   G(G_NM) = 0;

   %ZERO
   direction = thetaDirection==ZERO;
   direction = direction(2:sDy-1, 2:sDx-1);
   G_NM = true(size(G));
   G_NM(2:sDy-1, 2:sDx-1) = direction;
   G(G_NM) = 0;

   %Double
Thresholding=============================================================
   StrongEdge = threshold2 <= G;

   %Hysteresis
   while true
       WeakEdge = threshold1 <= G & threshold2 > G & (StrongEdge == 0);
       Cnt_Hysteresis = 0;

       %VERTICAL
       direction = thetaDirection==VERTICAL;
       direction = direction(2:sDy-1, 2:sDx-1);
       StrongEdge_Hysteresis = false(size(G));
       p1 = WeakEdge(3:sDy, 2:sDx-1);
       p2 = WeakEdge(1:sDy-2, 2:sDx-1);
       StrongEdge_Hysteresis(3:sDy, 2:sDx-1) =
StrongEdge_Hysteresis(3:sDy, 2:sDx-1) | (direction & p1);
       StrongEdge_Hysteresis(1:sDy-2, 2:sDx-1) =
StrongEdge_Hysteresis(1:sDy-2, 2:sDx-1) | (direction & p2);
       StrongEdge(StrongEdge_Hysteresis) = 1;
       Cnt_Hysteresis = Cnt_Hysteresis + sum(StrongEdge_Hysteresis,"all");
%        figure,imshow(StrongEdge);

       %HORIZON
       direction = thetaDirection==HORIZON;
```

```
       direction = direction(2:sDy-1, 2:sDx-1);
       StrongEdge_Hysteresis = false(size(G));
       p1 = WeakEdge(2:sDy-1, 3:sDx);
       p2 = WeakEdge(2:sDy-1, 1:sDx-2);
       StrongEdge_Hysteresis(2:sDy-1, 3:sDx) =
StrongEdge_Hysteresis(2:sDy-1, 3:sDx) | (direction & p1);
       StrongEdge_Hysteresis(2:sDy-1, 1:sDx-2) =
StrongEdge_Hysteresis(2:sDy-1, 1:sDx-2) | (direction & p2);
       StrongEdge(StrongEdge_Hysteresis) = 1;
       Cnt_Hysteresis = Cnt_Hysteresis + sum(StrongEdge_Hysteresis,"all");
%        figure,imshow(StrongEdge);

       %SLASH
       direction = thetaDirection==SLASH;
       direction = direction(2:sDy-1, 2:sDx-1);
       StrongEdge_Hysteresis = false(size(G));
       p1 = WeakEdge(1:sDy-2, 3:sDx);
       p2 = WeakEdge(3:sDy, 1:sDx-2);
       StrongEdge_Hysteresis(1:sDy-2, 3:sDx) =
StrongEdge_Hysteresis(1:sDy-2, 3:sDx) | (direction & p1);
       StrongEdge_Hysteresis(3:sDy, 1:sDx-2) =
StrongEdge_Hysteresis(3:sDy, 1:sDx-2) | (direction & p2);
       StrongEdge(StrongEdge_Hysteresis) = 1;
       Cnt_Hysteresis = Cnt_Hysteresis + sum(StrongEdge_Hysteresis,"all");
%        figure,imshow(StrongEdge);

       %BACK_SLASH
       direction = thetaDirection==BACK_SLASH;
       direction = direction(2:sDy-1, 2:sDx-1);
       StrongEdge_Hysteresis = false(size(G));
       p1 = WeakEdge(3:sDy, 3:sDx);
       p2 = WeakEdge(1:sDy-2, 1:sDx-2);
       StrongEdge_Hysteresis(3:sDy, 3:sDx) = StrongEdge_Hysteresis(3:sDy,
3:sDx) | (direction & p1);
       StrongEdge_Hysteresis(1:sDy-2, 1:sDx-2) =
StrongEdge_Hysteresis(1:sDy-2, 1:sDx-2) | (direction & p2);
       StrongEdge(StrongEdge_Hysteresis) = 1;
       Cnt_Hysteresis = Cnt_Hysteresis + sum(StrongEdge_Hysteresis,"all");
```

```
%       figure,imshow(StrongEdge);

      if Cnt_Hysteresis == 0
          break;
      end
   end

   out = StrongEdge;

end
```

main.m

```
clear
clc
close all;
img= imread("xray.png");

%
img_f = double(img);
% img_f = 255.0./(1+exp(1).^(-2.*img_f+4));
% img_r = uint8(img_f);

BW1 = edge(img,'Canny');

r = Canny(img, 30, 40);

figure,imshow(BW1);
figure,imshow(r);
```