**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**](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); |