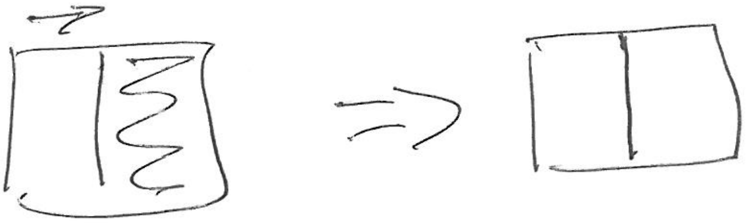
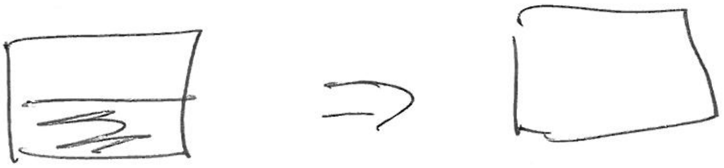
Edge Detection 140-153

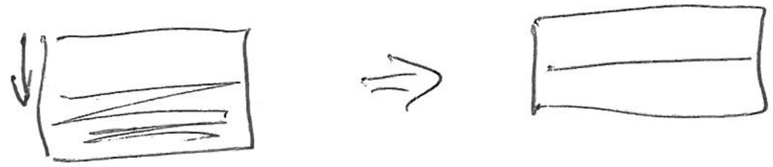
i). differentiation = differencing

$$edge_{x,y} = f_{x+1,y} - f_{x,y}$$





$$= f_{x,y+1} - f_{x,y}$$

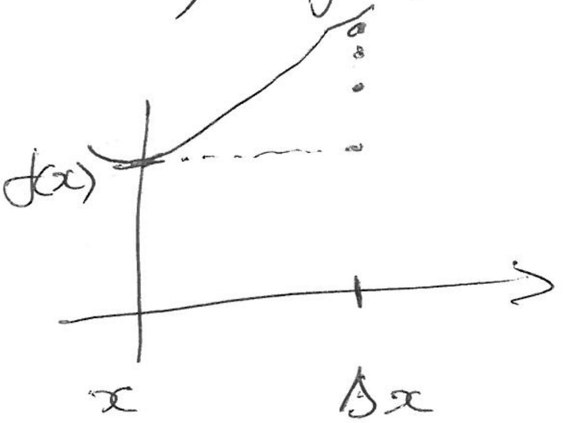


$$edge_{x,y} = f_{x+1,y} + f_{x,y+1} - 2f_{x,y}$$

-2	1
1	

ii). better operator? include smoothing.

$$① \quad f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) \ldots$$



1	-1

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x)$$

$$② \quad f(x-\Delta x) = f(x) - \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) \pm O(\Delta x^3)$$

$$①-② \quad f(x+\Delta x) - f(x-\Delta x) = 2\Delta x f'(x) + O(\Delta x^3).$$

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + O(\Delta x^2)$$

+1	0	-1

this is better if $\Delta x^2 \ll \Delta x$

& since $\Delta x \ll 1$, it is better

iii). better edge detection

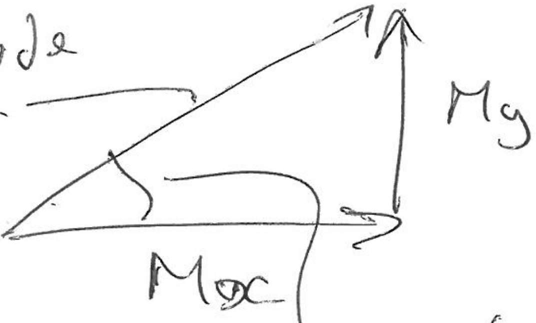
$$\begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

include averaging

ave →

diff^n ↓

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}}_{M_y} \qquad \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ \phi & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{M_x} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$



edge magnitude $\sqrt{Mx^2+My^2}$

edge direction $= \tan^{-1} \frac{My}{Mx}$

N/. Solvel operator

employs Gaussian averagily.

1	2	1
0	0	0
−1	−2	−1

M_y

1	0	−1
2	0	−2
1	0	−1

Good basic operator

5×5?

$$\begin{array}{lccccccccc} & & & & 1 & & 1 & & & \\ & & & 1 & & 2 & & 1 & & \\ & & 1 & & 3 & & 3 & & 1 & \\ M_x & 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

$$\begin{array}{lccccccccc} & & & & 1 & & -1 & & & \\ & & & 1 & & 0 & & -1 & & \\ & & 1 & & 1 & & -1 & & -1 & \\ M_y & 1 & & 2 & & 0 & & -2 & & -1 \end{array}$$

$M_x \times M_y^T$

```
function convolved = convolve(image,template)
%New image point brightness convolution of template with image
%  Usage: [new image] = convolve(image,template of point values)
%
%  Parameters: image      - array of points
%              template   - array of weighting coefficients

%get image dimensions
[irows,icols]=size(image);

%get template dimensions
[trows,tcols]=size(template);

%set a temporary image to black
temp(1:irows,1:icols)=0;

%half of template rows is
tr=floor(trows/2);

%half of template cols is
tc=floor(tcols/2);

%then convolve the template
for x = tr +1:icols- tr %address all columns except border
  for y = tc +1:irows- tc %address all rows except border
    sum=0;
    for iwin = 1:trows %address template columns
      for jwin = 1:tcols %address template rows
        sum=sum+image(y+jwin-tr-1,x+iwin-tc-1)*
                           template(tcols-jwin,trows-iwin);
      end
    end
    temp(y,x)=sum;
  end
end
```