Enhancement

Monday, January 6, 2020 10:46 AM

Read 3.3

Def: accentuating important image features or suppressing unwanted features to make information more accessible for display or analysis

Examples are edges, contrast, or texture

Enhancement methods are motivated by a wide variety of goals:

- contrast enhancement
- noise reduction
- edge sharpening
- magnification / resolution enhancement

Tools:

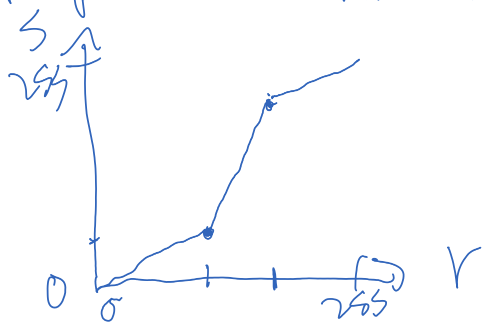
- pointwise operations
- algebraic "
- spatial "
- combinations of the above

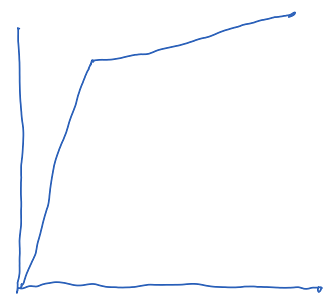
Pointwise operations

$s = T(r)$

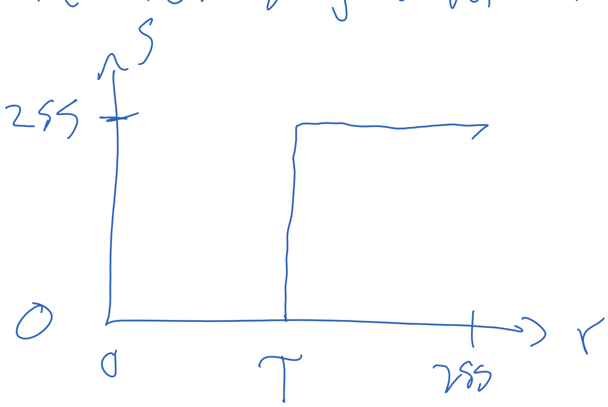
- pointwise transformation of intensity values

- contrast stretching — interval of pixels where intensity values are concentrated is stretched over larger intensity range to improve contrast





- thresholding
  - useful for images known to be binary (e.g., faxes or digitized forms)
  - useful for segmentation



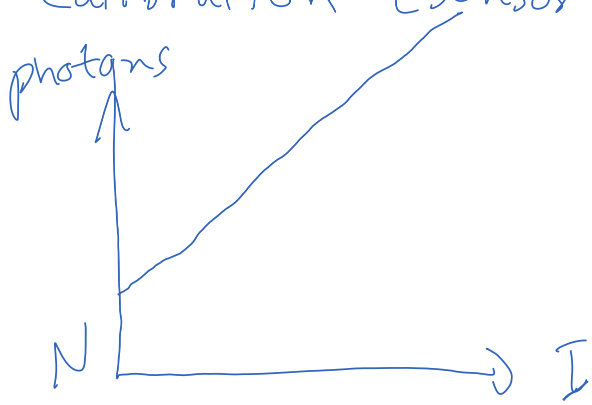
- range compression — if dynamic range is too large, we lose details at lower end, e.g. DFT magnitude

$$s = c\log(|r| + \varepsilon)$$

adjustable

~~Photometric calibration~~ (sensor or display)

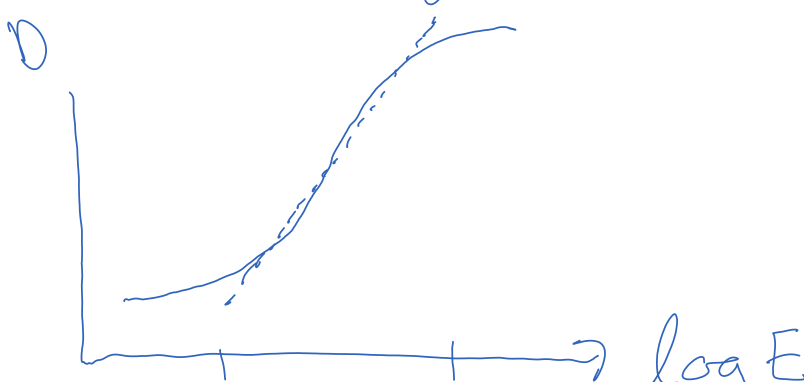
CCD



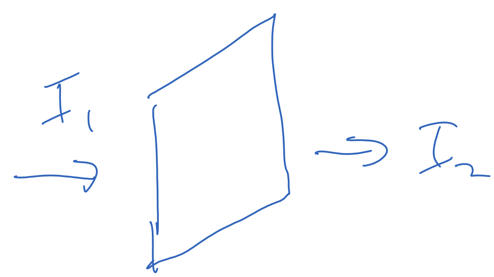
$$f(I) = \alpha I + N$$

$$\Rightarrow s(r) = \frac{r - N}{\alpha}$$

film negative



density $D = \log \frac{I_1}{I_2}$



exposure $E = T\,I$

where T = length of exposure, I = intensity of light

$D \approx \gamma \log E + k$ in linear region

How do we correct for this response if we measure light intensity (in the lab) passing through negative?

$$D = \gamma \log T I_0 + \log e^k = \log (T I_0)^\gamma e^k$$, where I_0 = intensity

fixed (property of negative)

In lab,

$$D = \log \frac{I_s}{I_m}$$

where I_s = source lamp intensity (constant over image)

I_m = measured intensity on other side of negative (function of spatial coordinates)

$$\log \frac{I_s}{I_m} = \log (T I_0)^\gamma e^k$$

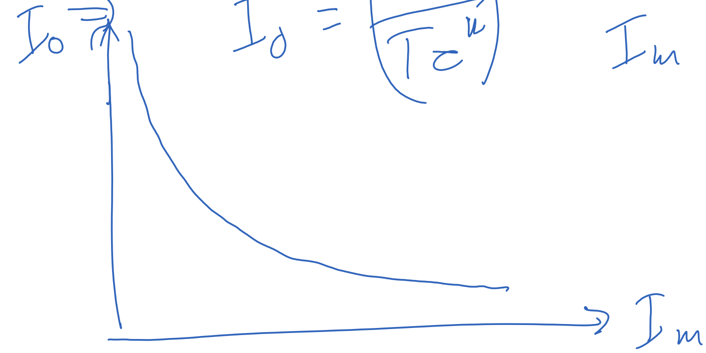
$$I_m = \frac{I_s}{\;}$$

$(TI_0)^\gamma e^k$

To correct, solve for I_0 in terms of I_m.

$$I_0 = \frac{1}{T}\left(\frac{I_s}{I_m e^k}\right)^{\frac{1}{\gamma}}$$

Let $\frac{1}{\gamma} = 1$ $\quad I_0 = \left(\frac{I_s}{T e^k}\right) \cdot \frac{1}{I_m}$



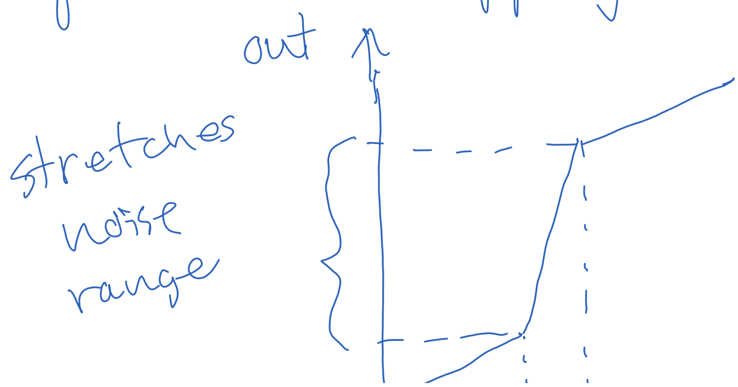
Project 3 due 2/21

Read 3.4. Skim 3.5-3.6

HW assignment posted

Noise effects in pointwise processing

Note that noise variance changes when a pointwise mapping is applied.



If the transformation can be locally approximated by a straight line;

$$s = \alpha r + \beta$$

then $\quad s = \alpha(\bar{r} + u) + \beta$

if $\quad r = \bar{r} + u$, $\quad$ where u is zero-mean noise, σ_u^2 variance

$$s = (\alpha \bar{r} + \beta) + \alpha u$$

αu ~~is~~ is the new noise term with variance $\alpha^2 \sigma_u^2$; std dev $= |\alpha| \sigma_u$

Histograms

- a histogram is a plot of relative frequencies of all gray levels
- pointwise operations modify the histogram

- histograms can be used to define transformations

Histogram equalization defines a pointwise transformation that tries to level out the histogram.

Let $h(x_i)$ = # of pixels with intensity x_i

L = # of gray levels

Then $\sum_{i=0}^{L-1} h(x_i) = MN$ = ~~total~~ # of pixels

Ideally, we want $\hat{h}(x_i) = \frac{MN}{L}$ at each x_i

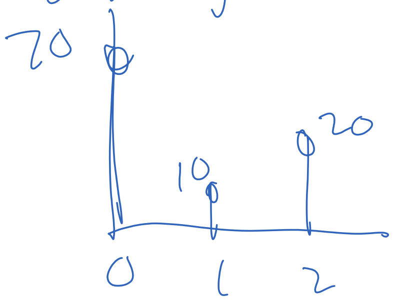
or $\sum_{i=0}^{k} h(x_i) \approx \frac{MN}{L}(k+1)$

can define a transformation between j & k such that

$$\sum_{i=0}^{j} h(x_i) = \sum_{i=0}^{k=\alpha(j)} \hat{h}(x_i) = \frac{MN}{L}(k+1)$$

Solving for k: $k = \text{round}\left(\frac{L}{MN}\sum^{j} h(x_i) - 1\right)$

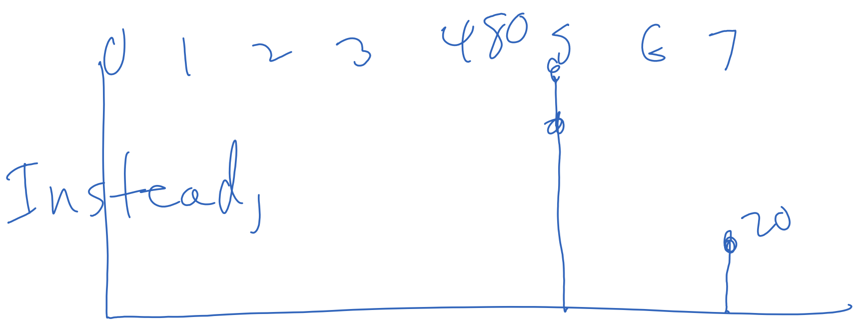
$L = 8$

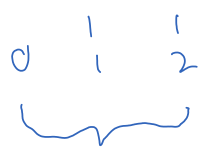


$\Rightarrow MN = 100$

j	formula	k
0	$\frac{8}{100}(70) - 1$	5
1	$\frac{8}{100}(80) - 1$	5
2	$\frac{8}{100}(100) - 1$	7

Instead,





2) Force histogram to fill range

$\Rightarrow$ only the right half of the left pulse & left half of right pulse affect the spread

3) Spread depends on total area between two pulse midpoints.

4) total region will fill $L_d - 1$

where L_d = desired # of gray levels

sum right of $i-1$ pulse and ~~left~~ half of i pulse

$$\frac{1}{2}\left(h(x_i) + h(x_{i-1})\right)$$

$\Rightarrow$ ~~begin~~ with $i=1$,

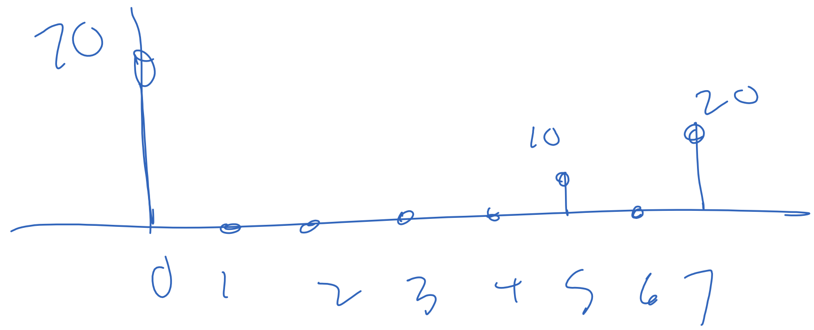
leave out left half of first pulse

$$= \frac{1}{MN - \frac{1}{2}h(x_0) - \frac{1}{2}h(x_{L-1})} \left[\sum_{i=1}^{L-1} \frac{1}{2}\left[h(x_i) + h(x_{i-1})\right] \right]$$

$$(L_d - 1)\left[\sum_{i=1}^{j} \frac{1}{2}\left(h(x_i) + h(x_{i-1})\right)\right] \Big] k$$

Read ~~B~~ round $\left[\ \text{formula}\ \right]$

j		k
0	—	0
1	$\frac{7}{55}(40) = 5.1$	5
2	$\frac{7}{55}(40+15) = 7$	7



Algebraic operations

* enhancement is sometimes performed by combining images:

$f(m,n) + g(m,n)$ $\qquad$ $f+g$, f_0+g

$f(m,n) - g(m,n)$ $\qquad$ $f-g$, f_0-g

$f(m,n) * g(m,n)$ $\qquad$ $f_0 * g$

$f(m,n) / g(m,n)$ $\qquad$ f_0 / g

- image averaging

Let $f_i(m,n) = \bar{f}(m,n) + u_i(m,n)$

where $u_i(m,n)$ is independent, identically distributed, zero mean, $var = \sigma_u^2$

$$f_{AVE}(m,n) = \frac{1}{N}\sum_{i=1}^{N} f_i(m,n) = \frac{1}{N}\sum_{i=1}^{N}\left[\bar{f}(m,n) + u_i(m,n)\right]$$

$$= \bar{f}(m,n) + \frac{1}{N}\sum_{i=1}^{N} u_i(m,n)$$

??

$$\mathrm{var}\{f_{AVE}(m,n)\} = E\{[f_{AVE}(m,n) - \bar{f}(m,n)]^2\}$$
$$= E\left\{\left[\frac{1}{N}\sum_{i=1} u_i(m,n)\right]^2\right\}$$
$$= \frac{1}{N}\sigma_u^2$$

- image subtraction
  - ideal for highlighting subtle differences between similar images
    - * motion detection
    - * change detection in medical imaging

$$g(m,n) = f_2(m,n) - f_1(m,n)$$

- image multiplication / division
  - * multiplication by binary image can mask out parts of image — 0's mask, and 1's retain
  - * division can correct for nonuniform sensor response

Spatial filtering

* can be linear or nonlinear

* spatial averaging (lowpass filter)
  - can be performed by convolution with uniform $(2M+1) \times (2M+1)$ kernel

$$g(m,n) = \sum_k \sum_l h(k,l) f(m-k, n-l)$$

$$= \frac{1}{(2M+1)^2} \sum_{k=-M}^{M} \sum_{l=-M}^{M} f(m-k, n-l)$$

  - linear
  - freq. domain : $G(\omega_m, \omega_n) = H(\omega_m, \omega_n) F(\omega_m, \omega_n)$

$$H(\omega_m, \omega_n) = \frac{\sin\left(\frac{2M+1}{2}\right)\omega_m}{\sin \frac{\omega_m}{2}} \cdot \frac{\sin\left(\frac{2M+1}{2}\right)\omega_n}{\sin \frac{\omega_n}{2}}$$

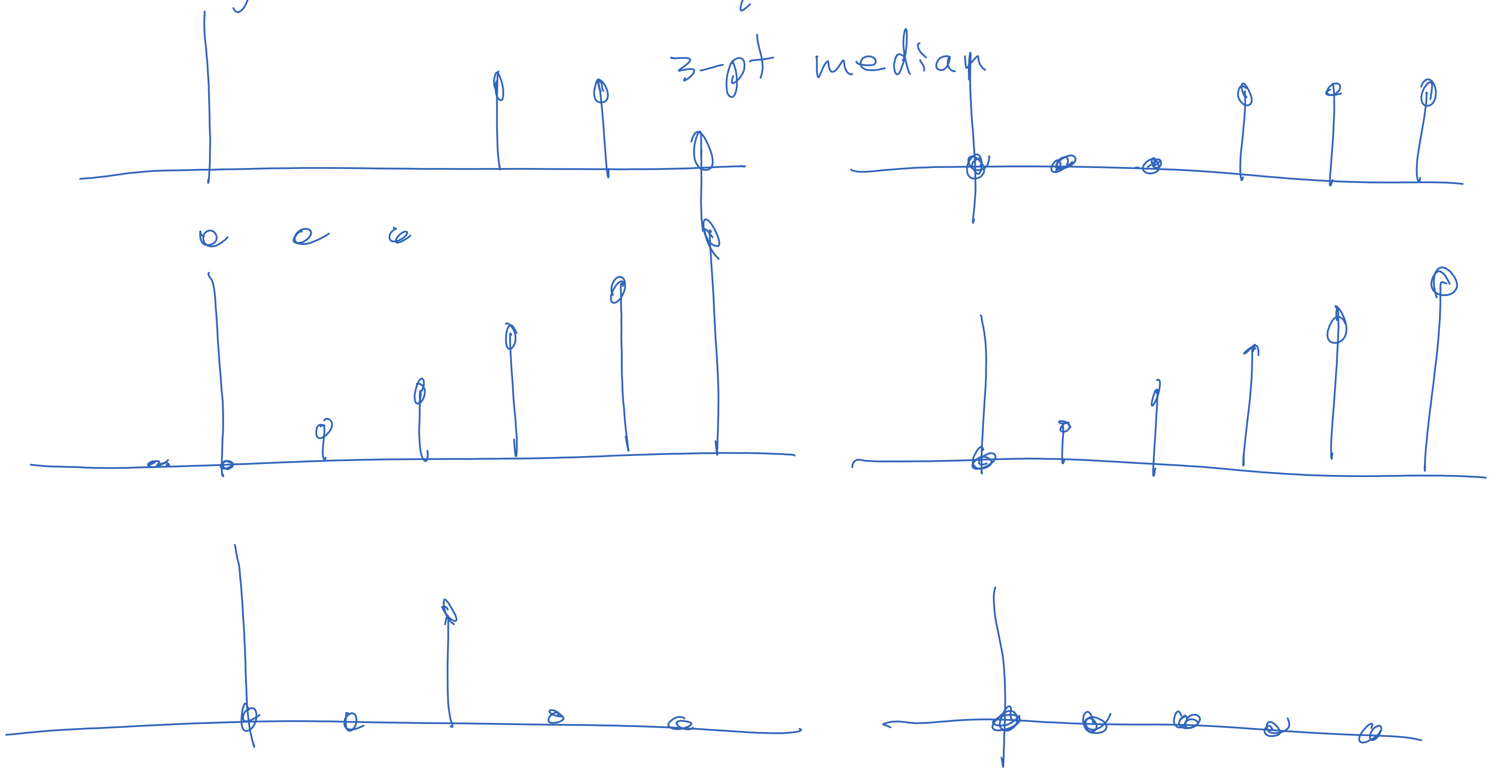
Effect: reduces corruption due to noise.
If noise is white with variance σ_n^2,
Then $(2M+1)\times(2M+1)$ local averaging decreases noise variance to

$$\frac{\sigma_n^2}{(2M+1)^2}$$

However, it also produces blurring of the underlying image.

- median filtering

$$g(m,n) = \text{median}\{f(m-k, n-l), (k,l) \in W\}$$



HW posted + 3.44, 3.48, 3.49

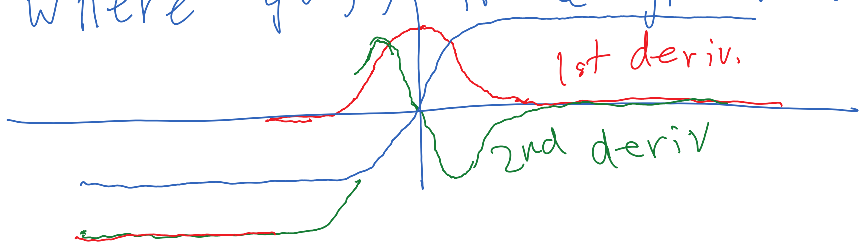
* sorting to choose middle value requires many comparisons
* sliding window can reduce comparisons required

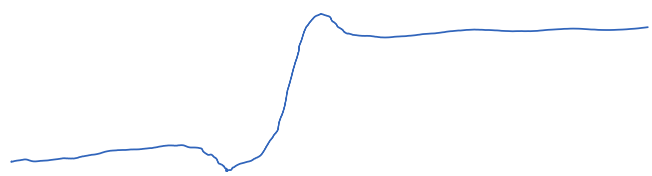
* unsharp masking
  - extract a smoothed version of the image from the original. This will accentuate sharp changes in the image

- equivalently, we can add a gradient or highpass image

$$v(m,n) = u(m,n) + \lambda g(m,n)$$

where $g(m,n)$ is a gradient image





subtract portion of 2nd derivative to get an accentuated edge

$$g(x,y) = \frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} \quad : \text{Laplacian}$$

$$\frac{du(x)}{dx} \approx \frac{u(x+\Delta) - u(x)}{\Delta}$$

In discrete case:

$$\overline{u(m+1) - u(m)}$$

approx 2nd deriv.

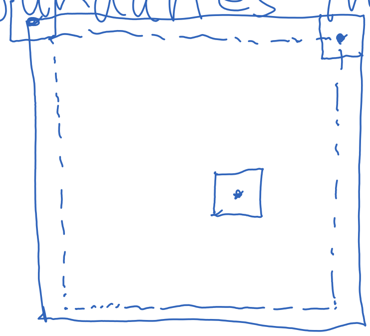
$$u(m+1) - 2u(m) + u(m-1)$$
$$= [u(m+1) - u(m)] - [u(m) - u(m-1)]$$

For 2-D case:

$$g(m,n) = u(m,n+1) - 2u(m,n) + u(m,n-1)$$
$$+ u(m+1,n) - 2u(m,n) + u(m-1,n)$$

$$= u(m,n) * \begin{array}{|c|c|c|} \hline & & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline \end{array}$$

Boundaries in spatial operations



- neighborhood will hang off image near the edges of image
- if pixels outside region of support (ROS) are assumed to be zero, this creates a false edge around the image that can cause artifacts in processing

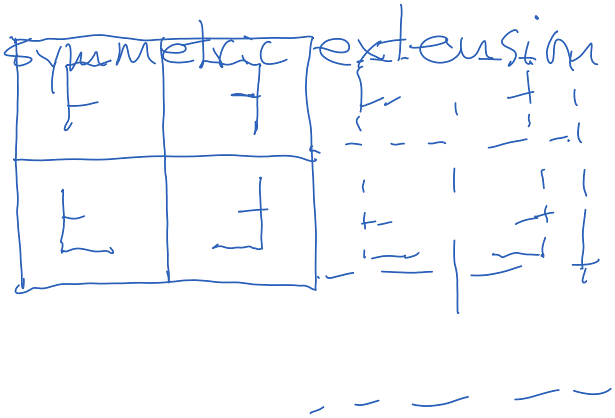
Options:

- modify algorithm slightly at boundary to use only available data
- replicate boundary pixels
- mirror (symmetrically extend) image
- periodically extend image
- \> replicate first difference of boundary

: boundaries in FFT-based processing

- replicate boundaries, then zeropad to obtain a linear convolution and/or to get an image up to power-of-2 size.

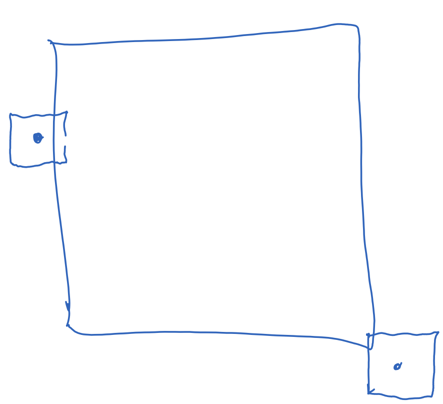
- Use symmetric extension before FFT



- after adding boundaries & processing, only keep part that is size of original

origin of neighborhood

Usually, we want to treat the center of the neighborhood as the origin. Otherwise, it will shift the output image.



Combination examples

edge detection

Edges characterize object boundaries and are therefore useful for segmentation, identification, and image registration (lining up two different images)

Pixel locations where abrupt grayscale changes occur in one direction are considered edges.

Three steps:

1) compute approximate gradients in both directions:

(convolve)

$$\begin{array}{ccc} & m & \\ \end{array} \qquad \begin{array}{ccc} & n & \end{array}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel operators

$$\Downarrow \qquad\qquad \Downarrow$$

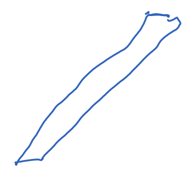
$$g_m(m,n) \qquad\qquad g_n(m,n)$$

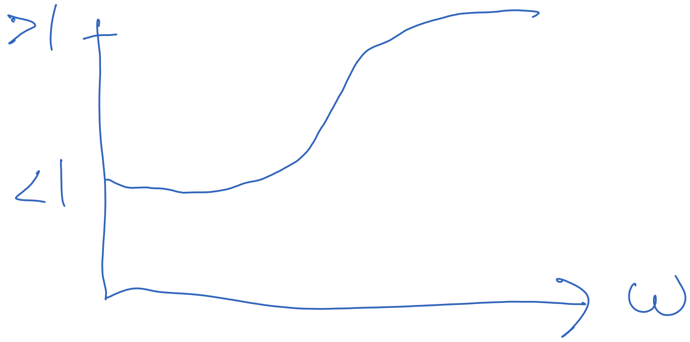
2) compute gradient magnitude

$$g(m,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$$

3) strong

Anything above threshold is an edge.





$\hat{l}(m,n)$ is lowpass

$\hat{r}(m,n)$ is highpass