Monday, February 11, 2019 8:12 AM

Image enhancement

Def: accentuating important image features or suppressing unwanted features to make visual information more accessible

Examples are edges, contrast, or texture.

Enhancement methods are motivated by a wide variety of goals:

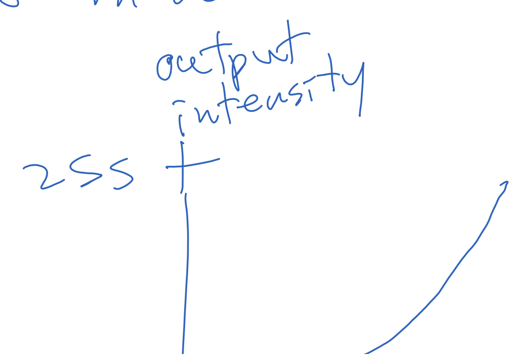
Ex: - contrast enhancement
- noise enhancement
- edge sharpening
- magnification

Tools:

- pointwise operations
- algebraic "
- spatial "
- combinations of these

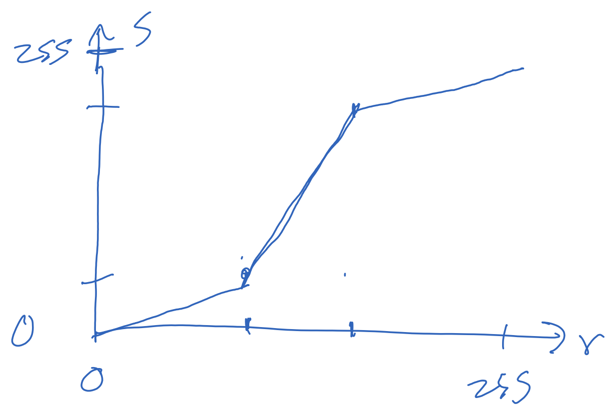
Pointwise operations

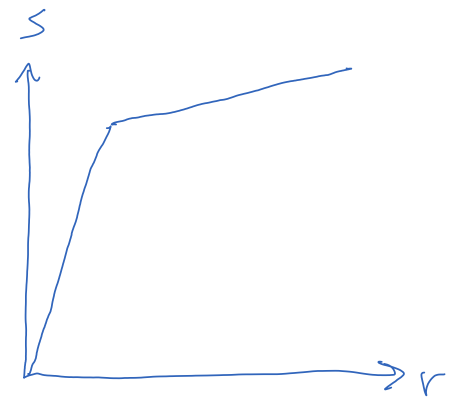
$$s = T(r)$$



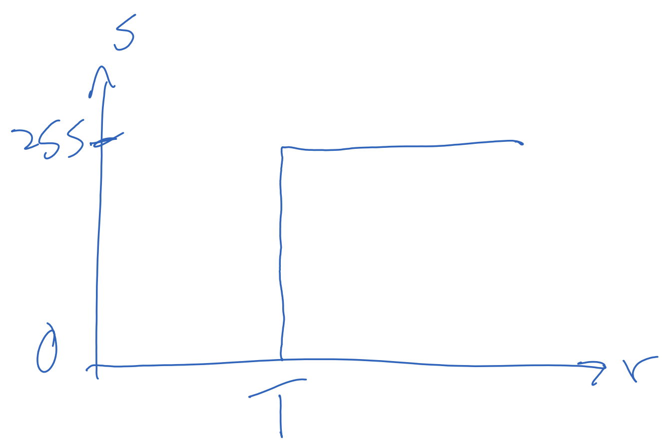
- pointwise transformation of intensity values

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





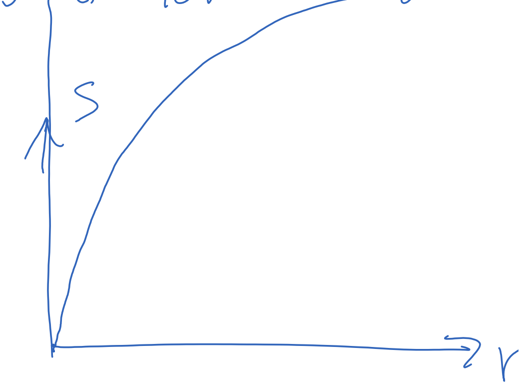
- thresholding



- 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

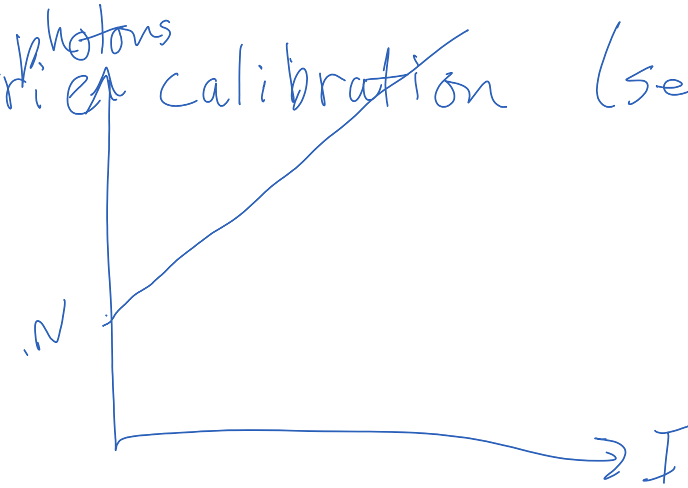


$$s = c \log(|r| + \varepsilon)$$

adjustable

---

photometric calibration (sensor or display)

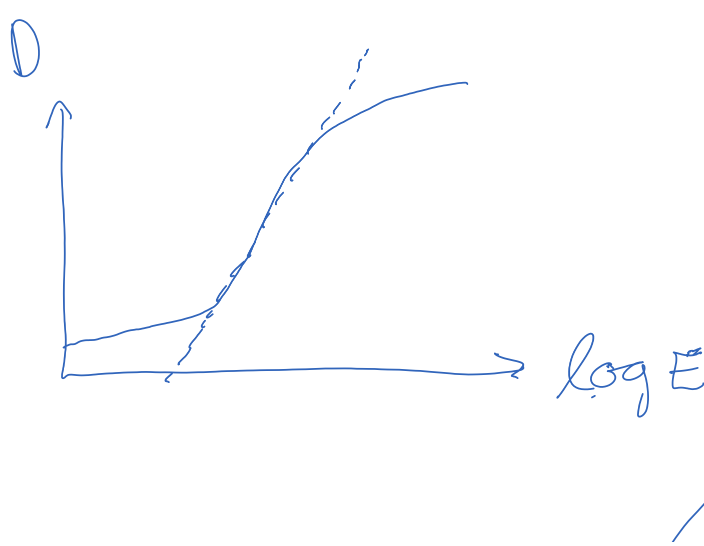


CCD

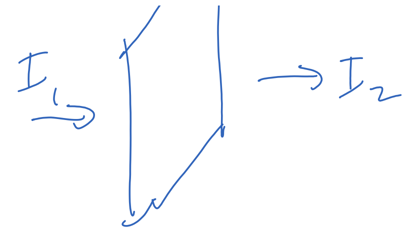
$$f(I) = \alpha I + N$$

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

film negative



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



exposure $E = T I_1$

where T = exposure time, I = intensity of light

Read 3.4, Skim 3.5-3.6

$D \simeq \gamma \log E + K$ in linear region

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

$$D = \gamma \log T I_0 + \log e^K$$ where I_0 = object intensity

$$= \log (T I_0)^\gamma e^K$$

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)

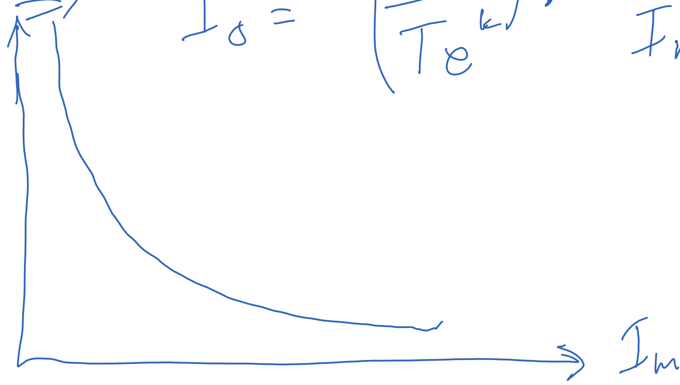
$$\log \frac{I_s}{I_m} = \log \frac{(T I_0)^{\gamma} e^{k}}{I_s}$$

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

To correct, solve for I_0 in terms of I_m:

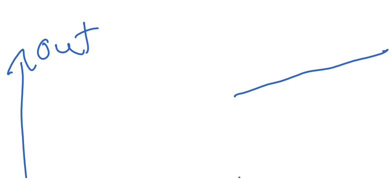
$$I_0 = \frac{1}{T}\left(\frac{I_s}{I_m e^{k}}\right)^{\frac{1}{\gamma}}$$

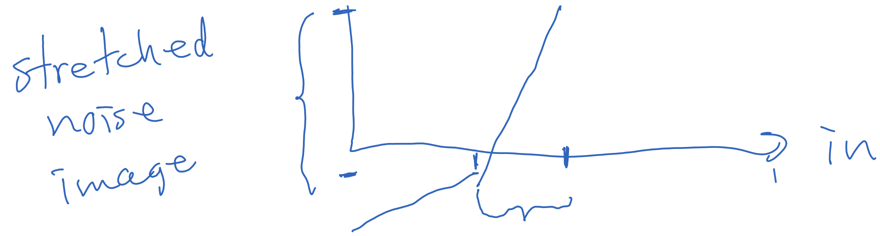
Let $\frac{1}{\gamma} = 1$ $I_0 \Rightarrow$ $\quad I_0 = \left(\frac{I_s}{T e^{k}}\right) \cdot \frac{1}{I_m}$



Noise effects in pointwise operations

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





If the transformation can be locally approximated by a straight line

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

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

then

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

$$= \underbrace{\alpha \bar{r} + \beta} + \alpha u$$

αu is the new noise term with

variance $\alpha^2 \sigma_u^2$, std. dev. $= |\alpha| \sigma_u$

Algebraic operations

* Enhancement is sometimes performed by combining images

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

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

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

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

- 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, with σ_u^2

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

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

$$\mathrm{var}\left[f_{AVE}(m,n)\right] = E\left\{\left[f_{AVE}(m,n) - \bar{f}(m,n)\right]^2\right\}$$

$$= E\left\{\left[\frac{1}{N}\sum_{i=1}^{N} u_i(m,n)\right]^2\right\}$$

$$= \frac{1}{N}\sigma_u^2$$

- image subtraction
  - ideal for highlighting subtle differences between similar images

* motion detection
* change detection in medical images

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

- 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

Read 3.7
HW will be posted
Project 3 due Fri.

histograms

- a histogram is a plot of relative freq. of all gray levels
- pointwise operations modify the histogram
- histograms can be used to define

transformations

<u>histogram equalization</u> 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 $\sum_{i=0}^{L-1} h(x_i)$ = total # of pixels = MN

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

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

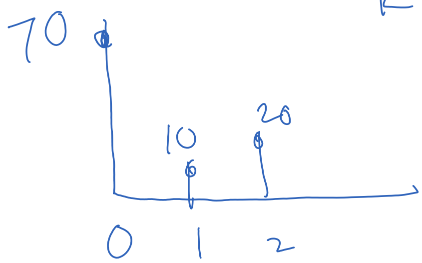
Can define a transformation between j + k such that

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

Solving for k:

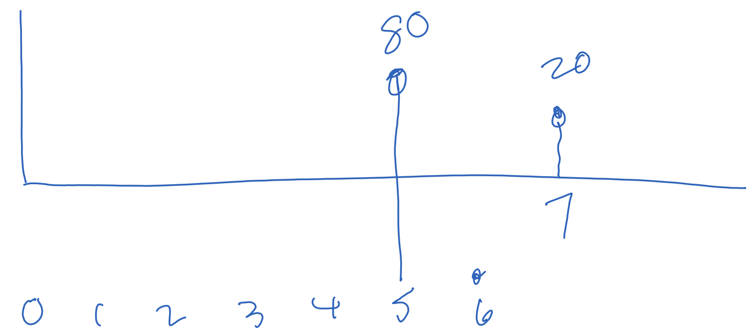
$$\sum^{j} h(x_i) - 1)$$

$k = \text{round}\left(\frac{L}{MN}\right.$ $T=8$



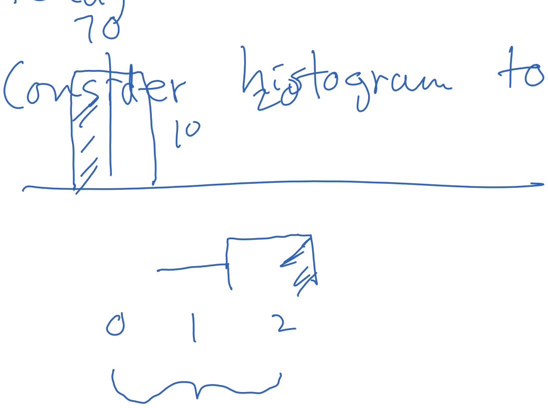
$L = 8$

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,

1) Consider histogram to be pulses



2) Force histogram to fill range
$\Rightarrow$ only the right half of the left pulse & left half of the right pulse affect the spread

3) Spread depends on total area between

the pulse midpoints

4) total region will fill $L_d - 1$

where L_d = desired # of gray levels

sum right half of $i-1$ pulse and left half of $\sum_{i=1}$ i pulse

begin with $i=1$ to leave out left half of $h(x_0)$

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

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

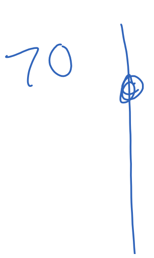
$L_d = 8$

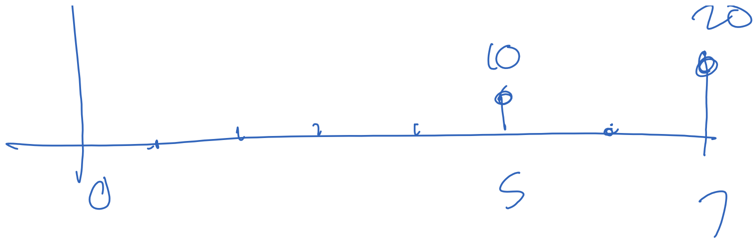
$N - \frac{1}{2}h(x_0) - \frac{1}{2}h(x_{L-1})$

$100 - \frac{1}{2}70 - \frac{1}{2}(20)$

55

j	formula	K
0	-	0
1	$\frac{7}{55}(40) = 5.1$	5
2	$\frac{7}{55}(55) = 7$	7





patial filtering

* can be linear or nonlinear

* spatial averaging (lowpass filter)

- can be performed by convolution with uniform $(2M+1) \times (2M+1)$ kernel

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

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

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

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

ead 3.7

Effect: reduces corruption due to

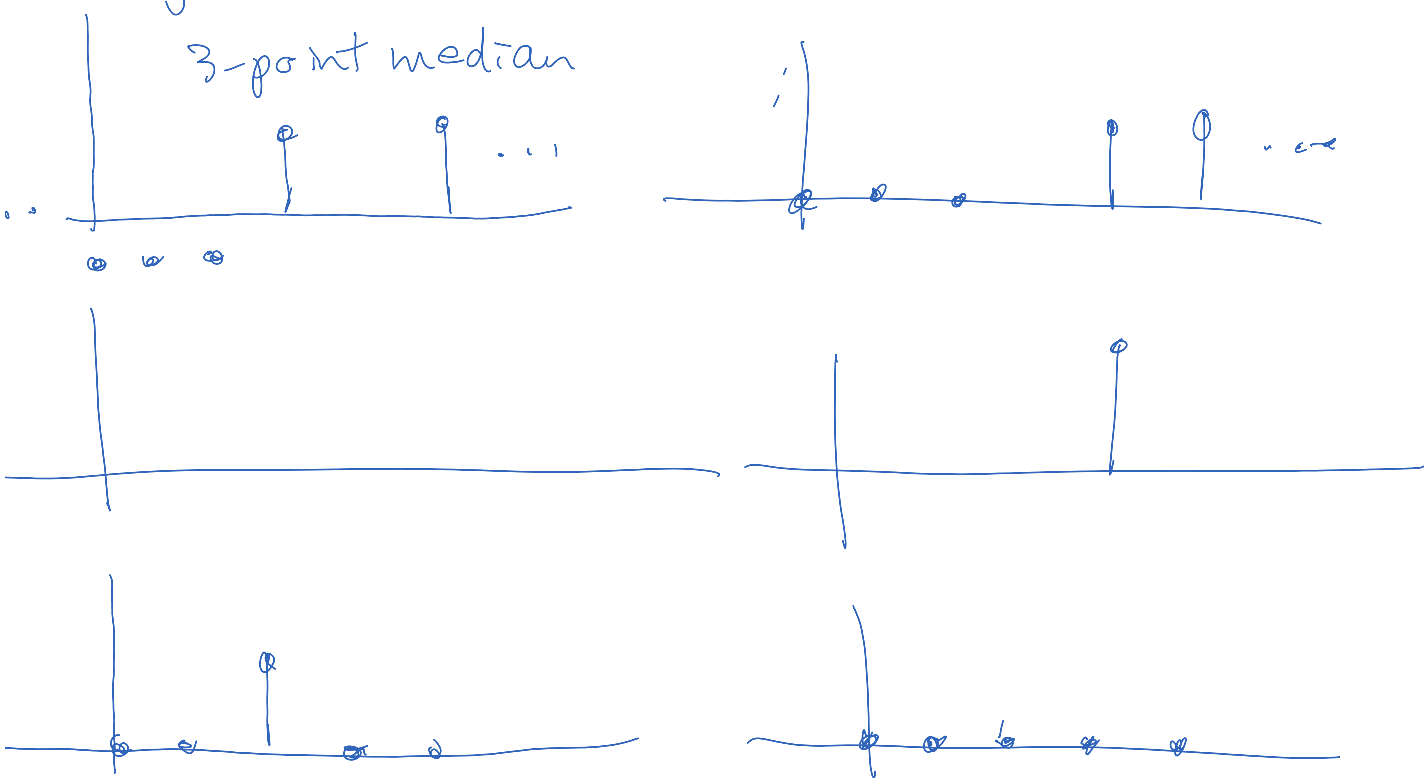
noise. If noise is white with variance σ_n^2, Then $(2M+1)\times(2M+1)$ local averaging decreases variance to $\dfrac{\sigma_n^2}{(2M+1)^2}$

However, it also produces blurring of underlying image.

- median filtering

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

3-point median



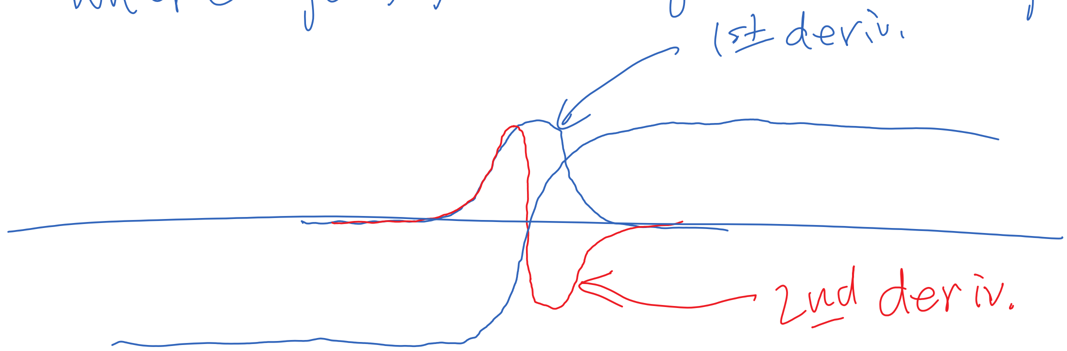
* sorting to choose middle value requires many computations

* sliding window can reduce comparisons required

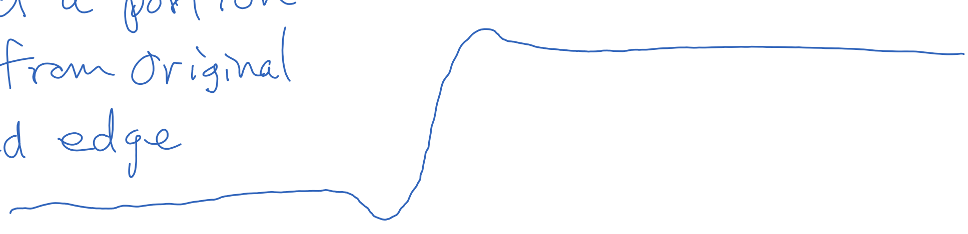
- unsharp masking
  - subtract a smoothed version of the image from the original. This will accentuate sharp changes in the image.
  - equivalently, we can add a gradient or highpass image

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

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



t: subtract a portion
nd deriv. from original
ccentuated edge



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

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

In discrete case: $\dfrac{u(m+1) - u(m)}{1}$

Approx. 2nd deriv.: $u(m+1) - u(m) - \big(u(m) - u(m-1)\big)$

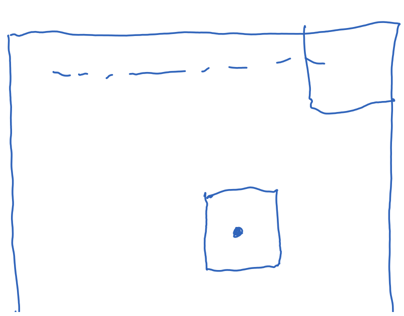
$$= u(m+1) - 2u(m) + u(m-1)$$

or 2-D,

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

$$g(m,n) = u(m,n) * \underbrace{\begin{array}{|c|c|c|}\hline & & \\ \hline & -4 & \\ \hline & & \\ \hline \end{array}}_{\text{discrete Laplacian}}$$

Boundaries in spatial operations



- neighborhood will hang ~~off~~ the image near the edges of the image

-ions:

irror the image over
he boundaries (symmetric
xtension)

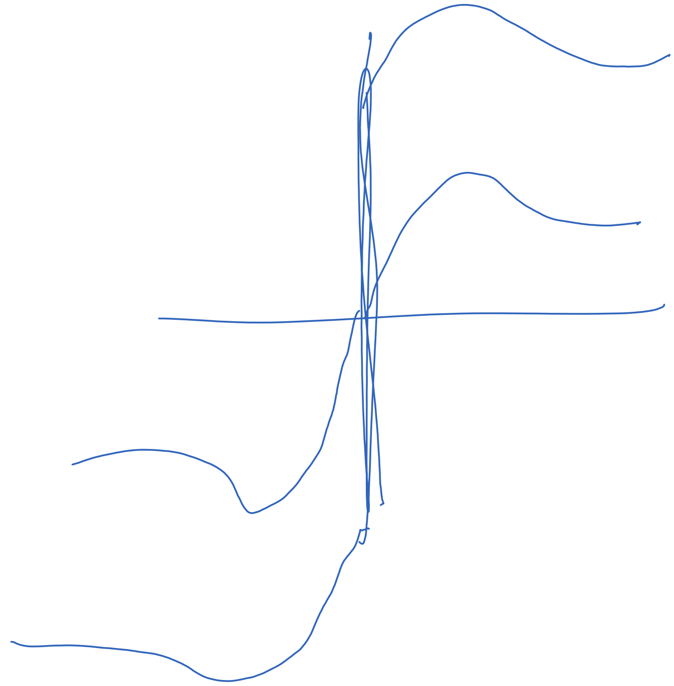
eplicate boundary pixels

se average value as
onstant background

eplicate first difference

hange algorithm at
oundaries to avoid
using pixels outside ROS

- If pixels outside region of
support (ROS) are assumed to
be zero, this creates a
false edge around the
image that can lead to
artifacts in processing



set posted

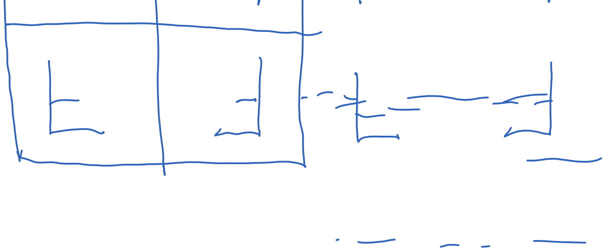
1, 3.48, 3.49

undaries 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 <u>before</u> FFT



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

in of neighborhood

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

$$\sum \sum f(m-k, n-\ell)\, h(k,\ell)$$

0

ination examples

ge detection

ges characterize object boundaries and are
erefore useful for segmentation, identification,
nd image registration (lining up two different
nages).

ixel values where abrupt grayscale changes
ccur in one direction are considered edges

ree steps:

Compute approximate gradi~~ents in~~ both directions

$$\left[\quad n \quad\right] \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$ Sobel operators

$\Downarrow$

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

compute gradient magnitude

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

strong gradients $\Rightarrow$ edges. So, threshold.

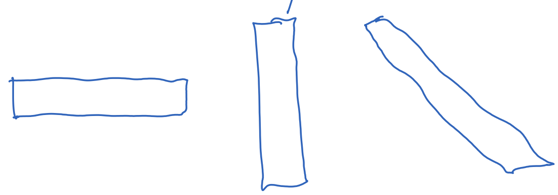
Anything over threshold is an edge.

4) remove isolated points & link edges

rectional averaging

- smoothing can be done in directions that don't cross over edges (prevents blurring of edges)

• image is smoothed in several directions independently



• select pixel from one of these directional images for which the value is closest to original noisy pixel — selection is made point by point.

nomorphic filtering

$$u(m,n) = i(m,n)\,r(m,n)$$

where $i(m,n)$ = illumination of scene

$r(m,n)$ = reflectance of scene

$$(m,n) = \log u(m,n)$$

$$= \log i(m,n) + \log r(m,n)$$

$$= \hat{i}(m,n) + \hat{r}(m,n)$$

mptions:

illumination varies slowly across a scene

reflectance varies rapidly across a scene

=> possible to partially separate the two by filtering

Would like a filter such that

$$\hat{y}(m,n) = \alpha\, \hat{i}(m,n) + \beta\, \hat{r}(m,n)$$

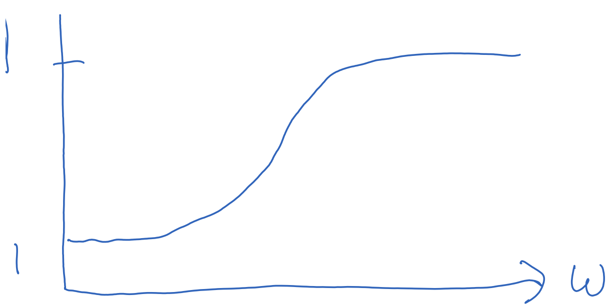
$\alpha < 1$

dynamic range compression

contrast enhancement

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

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



ilter $\hat{u}(m,n)$ to get $\hat{y}(m,n)$

$$y(m,n) = \exp\left[\hat{y}(m,n)\right]$$

- undoes log operation

pace-variant sharpening

- edge strength calculation $g(m,n)$
- let $s(m,n)$ be sharpened version of $f(m,n)$

$$y(m,n) = g(m,n)\, s(m,n) + (k - g(m,n))\, f(m,n)$$

nistogram equalization followed by median filtering