计算机视觉

邬向前

计算学部

多模态智能及应用研究中心

电子邮箱: xqwu@hit.edu.cn

Lecture 2：

Image Filtering

Outline

- Point (Pixel) Operations
- Group (Neighborhood) Operations

Image Histograms

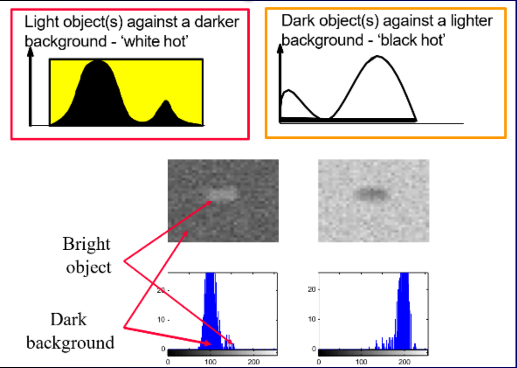
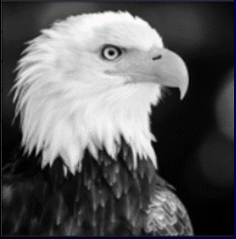


Image Histograms



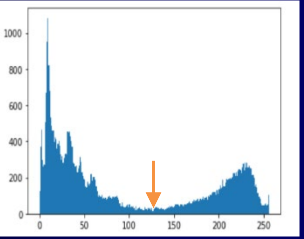


Image Histograms

上堂课的作业：

1. 打开任意一幅灰度图片，计算该图像的图像直方图；
2. 打开任意一幅灰度图片，对该图像进行处理，使得整幅图像变亮10个灰度级；
3. 打开任意一幅灰度图片，在该图像的中心位置画一个半径为半个图像高度或宽度的圆。

Point (Pixel) Operations

Examples of point processing

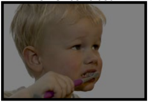
How would you implement these?

original

x

darken

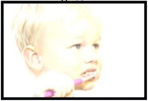
lower contrast



non-linear lower contrast

invert

lighten



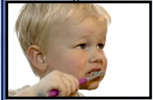
raise contrast

non-linear raise contrast

Examples of point processing

How would you implement these?

original

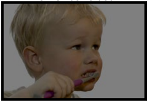


x

darken

$x - 128$

lower contrast



$\frac{x}{2}$

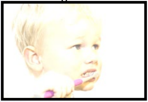
non-linear lower contrast

$\left(\frac{x}{255}\right)^{1/3} \times 255$

invert

$255 - x$

lighten



$x + 128$

raise contrast

$x \times 2$

non-linear raise contrast

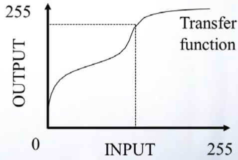
$\left(\frac{x}{255}\right)^{2} \times 255$

Point (Pixel) Operation

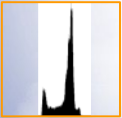
- Point operation: A function is applied to every pixel in an image, which operates only on the pixel's current value.
- Thresholding - A mask may be created by setting a pixel value to 1 or 0 depending upon if the current value is above or below a certain threshold value.

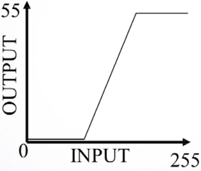
Input pixel value, *I*, mapped to output pixel value, *O*, via transfer function *T*.

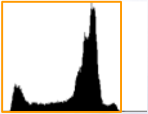
$$O=T(I)$$



Contrast Enhancement: Linear Stretching





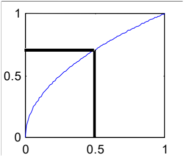


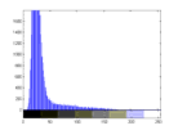
Contrast Enhancement: Power Law Function

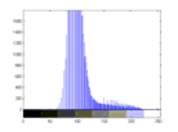
$$O = I^{\gamma}$$

- $\gamma < 1$ to enhance contrast in dark regions
- $\gamma > 1$ to enhance contrast in bright regions.

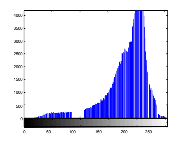
Contrast Enhancement: Power Law Function ($\gamma = 0.5$)

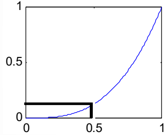


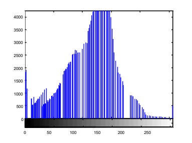




Contrast Enhancement: Power Law Function ($\gamma = 3.0$)







Contrast Enhancement: Power Law Function

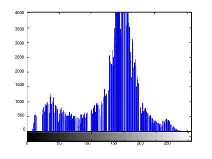
- Look-up Table
  - Transfer function implemented as a look-up table (LUT).
  - Implemented in hardware or software.

$\gamma=2$

I	*O*
0	0
0.10	0.01
0.20	0.04
0.30	0.03
0.40	0.16
0.50	0.25
0.60	0.36
0.70	0.49
0.80	0.64
0.90	0.81
1.00	1.00

Contrast Enhancement: Histogram Equalisation

- Image histograms consisting of peaks and low plains.
- Peaks = many pixels concentrated in a few grey levels
- Plains = small number of pixels distributed over a wider range of grey levels



Contrast Enhancement: Histogram Equalisation

- 直方图均衡化的理论基础
- 前提：如果一幅图像占有全部可能的灰度级，并且均匀分布
- 结论：该图像具有高对比度和多变的灰色色调
- 外观：图像细节主富，质量更高

Contrast Enhancement: Histogram Equalisation

- 直方图均衡化的步骤

1. 计算累计直方图

2. 将累计直方图进行区间转换

3. 在累计直方图中，概率相近的原始值，会被处理为相同的值。

Contrast Enhancement: Histogram Equalisation

- 直方图均衡化的步骤

1. 计算累计直方图

0	1	4	1	7	3	3
0	0	4	0	0	1	3
1	2	7	5	7	4	6
0	4	0	1	1	6	6
7	1	2	2	7	3	3
4	5	7	4	2	7	2
0	7	1	5	2	0	1

原始图像

像素级	个数
0	9
1	9
2	6
3	5
4	6
5	3
6	3
7	8

统计直方图

Contrast Enhancement: Histogram Equalisation

■ 直方图均衡化的步骤

1. 计算累计直方图

像素级	个数
0	9
1	9
2	6
3	5
4	6
5	3
6	3
7	8

统计直方图

像素级	百分比
0	0.18
1	0.18
2	0.12
3	0.10
4	0.12
5	0.06
6	0.06
7	0.16

归一化直方图

像素级	百分比
0	0.18
1	0.37
2	0.49
3	0.59
4	0.71
5	0.78
6	0.84
7	1.00

四舍五入

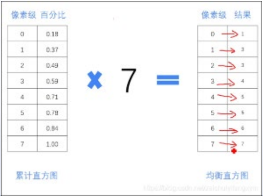
累计直方图

Contrast Enhancement: Histogram Equalisation

- 直方图均衡化的步骤

2. 将累计直方图进行区间转换

3. 在累计直方图中，概率相近的原始值，会被处理为相同的值。



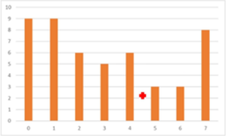
像素级	百分比
0	0.18
1	0.37
2	0.49
3	0.59
4	0.71
5	0.79
6	0.84
7	1.00

像素级	结果
0	1
1	3
2	3
3	4
4	5
5	5
6	6
7	7

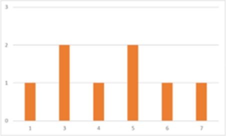
Contrast Enhancement: Histogram Equalisation

- 直方图均衡化的步骤

2. 将累计直方图进行区间转换

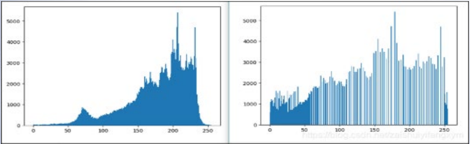


原始直方图



均衡直方图

Histogram Equalisation



Contrast Enhancement: Comparison

Original

$\gamma > 1$

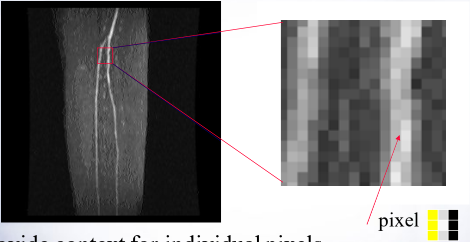
Histogram equalisation

Group (Neighborhood) Operations

Neighbourhood Operations

- Replace each pixel by a *linear* combination of its neighbors (and possibly itself).
- The combination is determined by the filter's *kernel*.
- The same kernel is *shifted* to all pixel locations so that all pixels use the same linear combination of their neighbors.

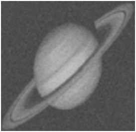
Why are Neighbourhoods Important?

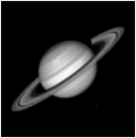


- Provide context for individual pixels.
- Relationships between neighbours determine image features.

Neighbourhood Operations

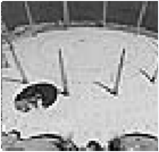
Noise Reduction





Edge Enhancement

Zooming



Convolution

Convolution for 1D continuous signals

Definition of filtering as convolution:

notice the flip

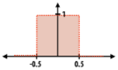
$$(f * g)(x) = \int_{-\infty}^{\infty} f(y) g(x - y) dy$$

filtered signal   filter   input signal

Consider the box filter example:

1D continuous box filter

$$f(x) = \begin{cases} 1 & |x| \leq 0.5 \\ 0 & otherwise \end{cases}$$



filtering output is a blurred version of g

$$(f * g)(x) = \int_{-0.5}^{0.5} g(x - y) dy$$

连续卷积的通俗解释：做馒头

楼下早点铺子生意太好了，供不应求，就买了一台机器，不断的生产馒头。

假设馒头的生产速度是 $f(t)$ ，那么一天后生产出来的馒头总量为：

$$\int_0^{24} f(t)dt$$

馒头生产出来之后，就会慢慢腐败，假设腐败函数为 $g(t)$ ，比如，10个馒头，24小时会腐败：

$$10 * g(t)$$

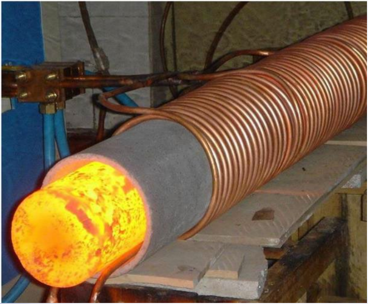
想想就知道，第一个小时生产出来的馒头，一天后会经历24小时的腐败，第二个小时生产出来的馒头，一天后会经历23小时的腐败。

如此，我们可以知道，一天后，馒头总共腐败了：

$$\int_0^{24} f(t)g(24-t)dt$$

这就是连续的卷积。

连续卷积的通俗解释：铁棒加热



$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-y)dy$$

Convolution for 2D discrete signals

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image filter input image

If the filter $f(i, j)$ is non-zero only within $-1 \leq i, j \leq 1$, then

$$(f * g)(x, y) = \sum_{i,j=-1}^{1} f(i, j) I(x - i, y - j)$$

The kernel we saw earlier is the 3x3 matrix representation of $f(i, j)$.

In general, the kernel for image processing is symmetry, so the flip processing can be skipped.

Convolution

- Consists of filtering an image A using a filter (mask) B.
- Mask is a small image whose pixel values are called weights.
- Weights modify relationships between pixels.

Filter, mask or template **B**

$$\begin{array}{|c|c|}\hline B_{1,1} & B_{1,2} \\ \hline B_{2,1} & B_{2,2} \\ \hline \end{array}$$

2×2

Input image **A**

$$\begin{array}{|c|c|c|c|}\hline A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\ \hline A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\ \hline A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\ \hline A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} \\ \hline \end{array} =$$

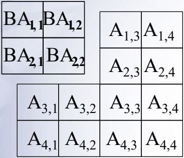
4×4

Convolved Image **C**

$$\begin{array}{|c|c|c|}\hline C_{1,1} & C_{1,2} & C_{1,3} \\ \hline C_{2,1} & C_{2,2} & C_{2,3} \\ \hline C_{3,1} & C_{3,2} & C_{3,3} \\ \hline \end{array}$$

3×3

Convolution



$A_{1,1} \times B_{1,1}$	$A_{1,2} \times B_{1,2}$
$A_{2,1} \times B_{2,1}$	$A_{2,2} \times B_{2,2}$

$$\mathbf{C_{1,1}} = \boxed{A_{1,1} \times B_{1,1}} + \boxed{A_{1,2} \times B_{1,2}} + \boxed{A_{2,1} \times B_{2,1}} + \boxed{A_{2,2} \times B_{2,2}}$$

Convolution

$A_{1,1}$	$A_{1,2}$	$BA_{1,3}$	$BA_{1,4}$
$A_{2,1}$	$A_{2,2}$	$BA_{2,3}$	$BA_{2,4}$
$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	$A_{3,4}$
$A_{4,1}$	$A_{4,2}$	$A_{4,3}$	$A_{4,4}$

$A_{1,3} \times B_{1,1}$ $A_{1,4} \times B_{1,2}$

$A_{2,3} \times B_{2,1}$ $A_{2,4} \times B_{2,2}$

$$\mathbf{C_{1,3}} = \boxed{A_{1,3} \times B_{1,1}} + \boxed{A_{1,4} \times B_{1,2}} + \boxed{A_{2,3} \times B_{2,1}} + \boxed{A_{2,4} \times B_{2,2}}$$

Convolution

$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$
$BA_{2,1}$	$BA_{2,2}$	$A_{2,3}$	$A_{2,4}$
$BA_{3,1}$	$BA_{3,2}$	$A_{3,3}$	$A_{3,4}$
$A_{4,1}$	$A_{4,2}$	$A_{4,3}$	$A_{4,4}$

$A_{2,1} \times B_{1,1}$ $A_{2,2} \times B_{1,2}$

$A_{3,1} \times B_{2,1}$ $A_{3,2} \times B_{2,2}$

$$\mathbf{C_{2,1}} = \boxed{A_{2,1} \times B_{1,1}} + \boxed{A_{2,2} \times B_{1,2}} + \boxed{A_{3,1} \times B_{2,1}} + \boxed{A_{3,2} \times B_{2,2}}$$

Mathematical Notation

$$C_{1,1} = \boxed{A_{1,1} \times B_{1,1}} + \boxed{A_{1,2} \times B_{1,2}} + \boxed{A_{2,1} \times B_{2,1}} + \boxed{A_{2,2} \times B_{2,2}}$$

$$B = M \times N$$

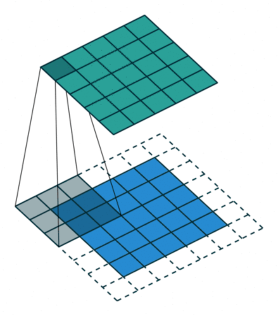
$$C_{k,l} = \sum_{i=k}^{k+M-1} \sum_{j=l}^{l+N-1} A_{i,j} \times B_{i-k+1,j-l+1}$$

$$\sum_{i=1}^{2} i = 1 + 2$$

$$\sum_{i=1}^{2} A_i = A_1 + A_2$$

$$\sum_{i=1}^{2} \sum_{j=1}^{2} A_{i,j} = \sum_{i=1}^{2} \left(A_{i,1} + A_{i,2}\right) = A_{1,1} + A_{1,2} + A_{2,1} + A_{2,2}$$

Convolution



Convolution

Filter, mask or template

B

-1	2
-1	2

2×2

Input image

A

4	4	7	9
4	3	8	9
3	5	9	9
3	6	10	9

4×4

=

Convolved Image

C

6	23	21
9	26	19
16	27	17

3×3

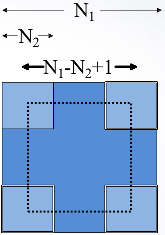
Convolution Size

Image size = $M_1 \times N_1$

Mask size = $M_2 \times N_2$

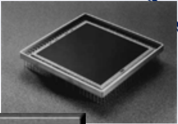
Convolution size = $(M_1 - M_2 + 1) \times (N_1 - N_2 + 1)$

Typical Mask sizes = 3×3, 5×5, 7×7, 9×9, 11×11

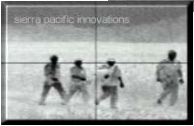


What is the convolved image size for a 128×128 image and 7×7 mask?

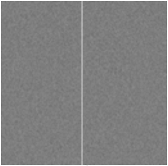
Noise



- Source of noise = CCD chip.
- Electronic signal fluctuations in detector.
- Caused by thermal energy.
- Worse for infra-red sensors.



image

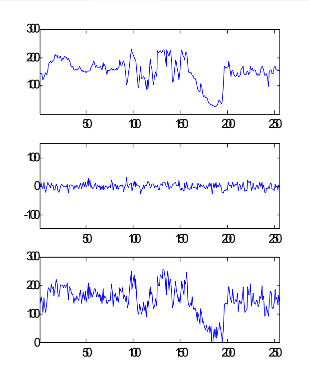


noise

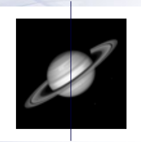
'grainy' image

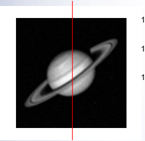
Noise

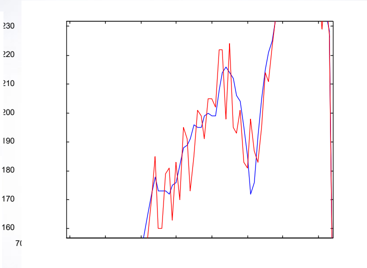
- Plot of image brightness.
- Vertical slice through image.
- Noise is additive.
- Noise fluctuations are rapid, i.e, high frequency.



Noise Reduction



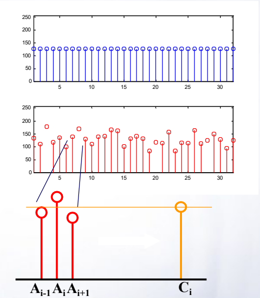




- Noise varies above and below uncorrupted image.

Noise Reduction-1st principles

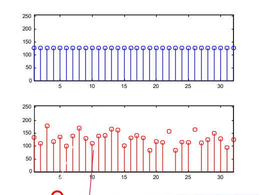
- How do we reduce noise?
- Consider a uniform 1-d image A and add noise.
- Focus on a pixel neighbourhood.
- Central pixel has been increased and <u>neighbouring</u> pixels have decreased.

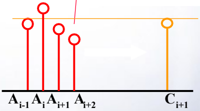


Noise Reduction-1st principles

- Averaging ‘smoothes’ the noise fluctuations.
- Consider the next pixel A_{i+1}
- Repeat for remainder of pixels.

$$C_{i+1} = \frac{A_i + A_{i+1} + A_{i+2}}{3}$$





Noise Reduction-
Neighborhood Operations

- All pixels can be averaged by convolving *1-d* image A with mask **B** to give enhanced image C.

$$\mathbf{C} = \mathbf{A} * \mathbf{B}$$

$$\mathbf{B} = \begin{bmatrix} B_1 & B_2 & B_3 \end{bmatrix}$$

$$C_i = A_{i-1} \times B_1 + A_i \times B_2 + A_{i+1} \times B_3$$

$$B = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$C_i = \frac{A_{i-1} + A_i + A_{i+1}}{3}$$

- Weights of B must equal one when added together.

$$\mathbf{C} = \mathbf{A} * \mathbf{B}$$

$$\mathbf{B} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

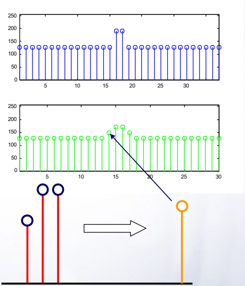
- Extend to two dimensions.

Noise Reduction

- Technique relies on high frequency noise fluctuations being ‘blocked’ by filter. Hence, low-pass filter.
- Fine detail in image may also be smoothed.
- Balance between keeping image fine detail and reducing noise.
- Example:
  - Saturn image coarse detail
  - Boat image contains fine detail
  - Noise reduced but fine detail also smoothed

Noise Reduction

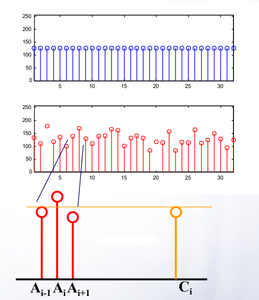
- Consider a uniform 1-d image A with a step function.
- Step function corresponds to fine image detail such as an edge.
- Low-pass filter ‘blurs’ the edge.



Noise Reduction-1st principles

- How do we reduce noise without averaging?
- Consider a uniform 1-d image A and add noise.
- Focus on a pixel neighbourhood.
- Non-linear operator?

Median filter!



Noise Reduction- Neighborhood Operations

- All pixels can be replaced by neighbourhood median by convolving 1-d image A with median filter B to give enhanced image C.

$$\mathbf{C} = \mathbf{A} * \mathbf{B}$$
$$\mathbf{B} = [B_1 \quad B_2 \quad B_3]$$
$$C_i = \text{median}\{A_{i-1} \times B_1, A_i \times B_2, A_{i+1} \times B_3\}$$
$$\mathbf{B} = [1 \quad 1 \quad 1]$$
$$C_i = \text{median}\{A_{i-1}, A_i, A_{i+1}\}$$

- Extend to two dimensions.

$$C_{k,l} = \underset{i=k:k+M-1,\, j=l:l+N-1}{\text{median}} \{A_{i,j} \times B_{i-k+1,\, j-l+1}\}$$
$$B_{i,j} = 1 \text{ for all } i,j$$

Noise reduction

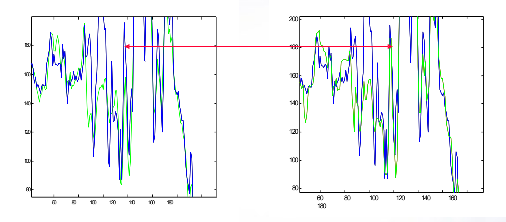
Original

Low-pass

Median

Noise reduction

Low-pass　　　　Median



- Low-pass: fine detail smoothed by averaging
- Median: fine detail passed by filter

Filter Operations

1 Low-Pass Filter

- Class: Image Enhancement/Restoration
- Implementation: Pixel group process and smooth an image

$$\begin{matrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{matrix}\Big/9 \qquad \begin{matrix}1 & 1 & 1\\1 & 2 & 1\\1 & 1 & 1\end{matrix}\Big/10 \qquad \begin{matrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{matrix}\Big/16$$

2 High-Pass Filter

- Implementation: Pixel group process and sharpen an image

$$\begin{matrix}-1 & -1 & -1\\-1 & 9 & -1\\-1 & -1 & -1\end{matrix} \qquad \begin{matrix}0 & -1 & 0\\-1 & 5 & -1\\0 & -1 & 0\end{matrix} \qquad \begin{matrix}1 & -2 & 1\\-2 & 5 & -2\\1 & -2 & 1\end{matrix}$$

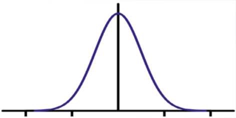
Filter Operations

3 The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance
- Any heuristics for selecting where to truncate?
- usually at 2-3σ

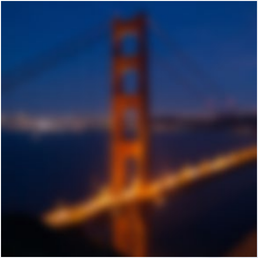


Is this a separable filter?

kernel $\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Gaussian filtering example

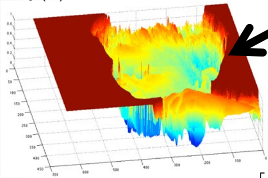
Edge Enhancement Neighborhood Operations

- Sobel Filter
- Derivative of Gaussian (DoG) filter
- Derivative of Laplacian (LoG) filter

What are image edges?

grayscale image

$f(\boldsymbol{x})$



Very sharp discontinuities in intensity.

domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

1D derivative filter

-1	0	1

1	0	-1

The Sobel filter

Horizontal Sober filter:

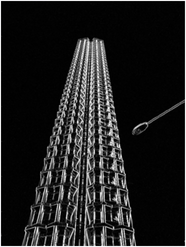
$$\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

Vertical Sober filter:

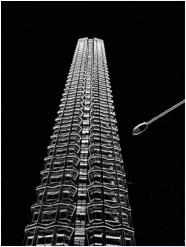
$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

Sobel filter example

original

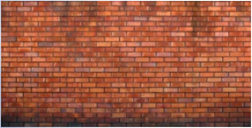


horizontal Sobel filter



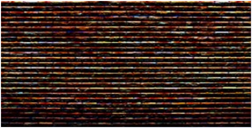
vertical Sobel filter

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

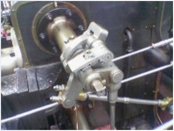
3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

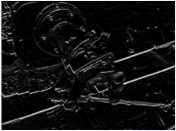
gradient | direction | amplitude

Image gradient example

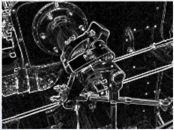
original



vertical derivative



gradient amplitude

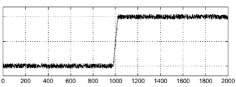


horizontal derivative

How does the gradient direction relate to these edges?

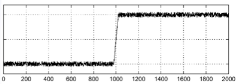
How do you find the edge of this signal?

intensity plot



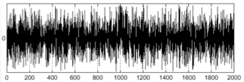
How do you find the edge of this signal?

intensity plot



Using a derivative filter:

derivative plot



What's the problem here?

Differentiation is very sensitive to noise

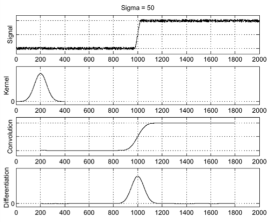
When using derivative filters, it is critical to blur first!

input

Gaussian

blurred

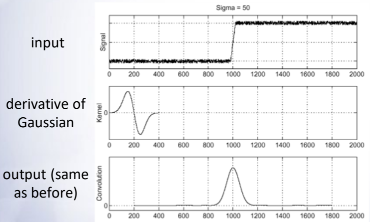
derivative of blurred



How much should we blur?

Derivative of Gaussian (DoG) filter

Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$



- How many operations did we save?
- Any other advantages beyond efficiency?

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference $f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$ $\longrightarrow$ 1D derivative filter

1	0	-1

second-order finite difference $f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$ $\longrightarrow$ Laplace filter

1	-2	1

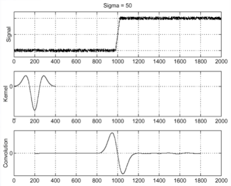
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



"zero crossings" at edges

Laplace and LoG filtering examples

Laplacian of Gaussian filtering

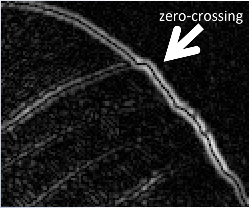
Laplace filtering

Laplacian of Gaussian vs Derivative of Gaussian

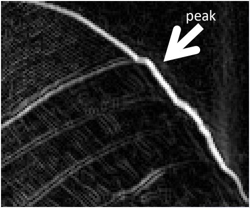
Laplacian of Gaussian filtering

Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian



Laplacian of Gaussian filtering

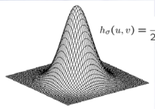


Derivative of Gaussian filtering

Zero crossings are more accurate at localizing edges (but not very convenient).

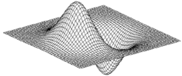
2D Gaussian based filters

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



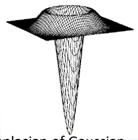
Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Derivative of Gaussian（DOG）

$$\nabla^2 h_\sigma(u,v)$$



Laplacian of Gaussian（LOG）

作业：

1. 编程实现图像的卷积操作函数；
2. 给定一方差，编制三个函数，分别生成高斯滤波器、DOG和LOG滤波器
3. 打开任意一幅灰度图片，用上面的卷积函数和生成的滤波器，对其进行滤波并显示

So much for today!

Thank you !!!