“感知”

特征信息提取、对象识别、场景理解、运动分析、三维深度感知、
上下文感知 ……

“感知”

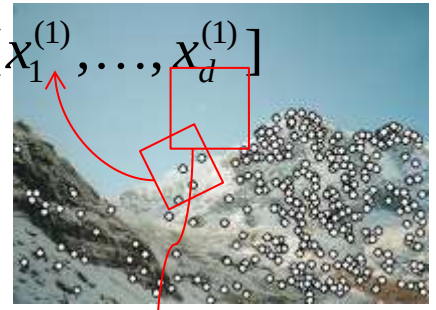
从 几何模型：基于几何原理和相对位置关系的模型

到 统计模型：基于数据分布和统计推断的模型

计算机的视觉认知：基于特征匹配的识别

1) 检测 Detection: 找到图中的关键点

2) 解释 Description: 在关键点周围提取特征



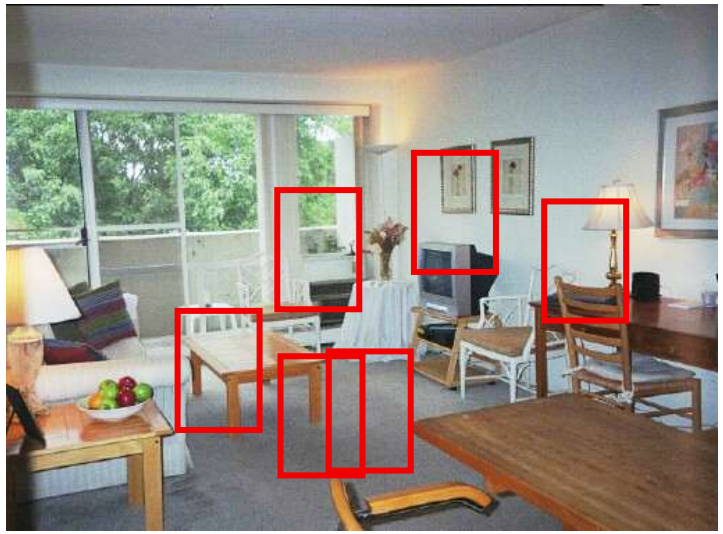
$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

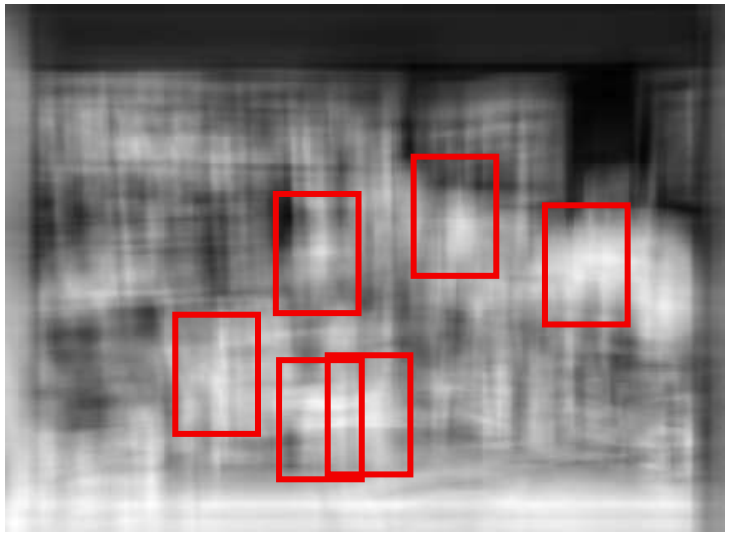
3) 匹配 Matching: 根据两个视角下的特征进行匹配

Kristen Grauman

挑战是什么？

找到图中的椅子

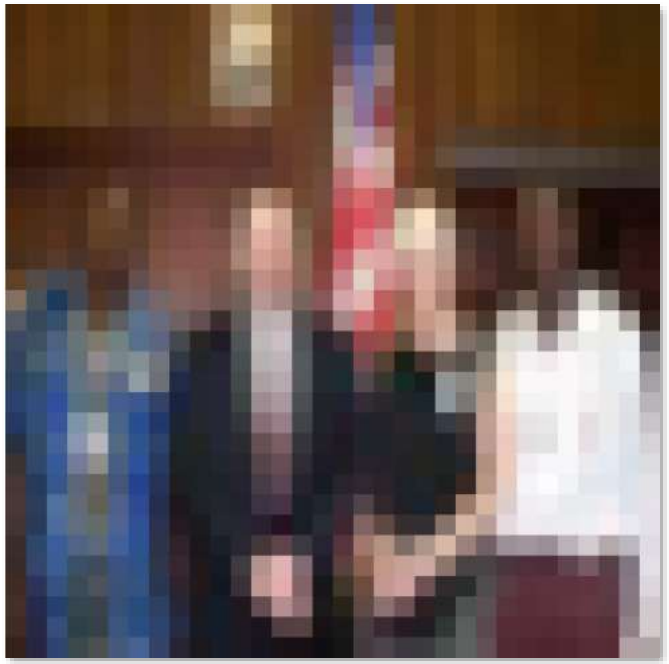




绝大多数目标框都没有目标物体

目标物体在新场景下不会一成不变

人类的优势：经验推理



Source: “80 million tiny images” by Torralba, et al.

计算机的优势：感知可以利用大量的冗余

一个房子也许认不出来…

一百个房子认出来了！

Source: S. Lazebnik

“感知”

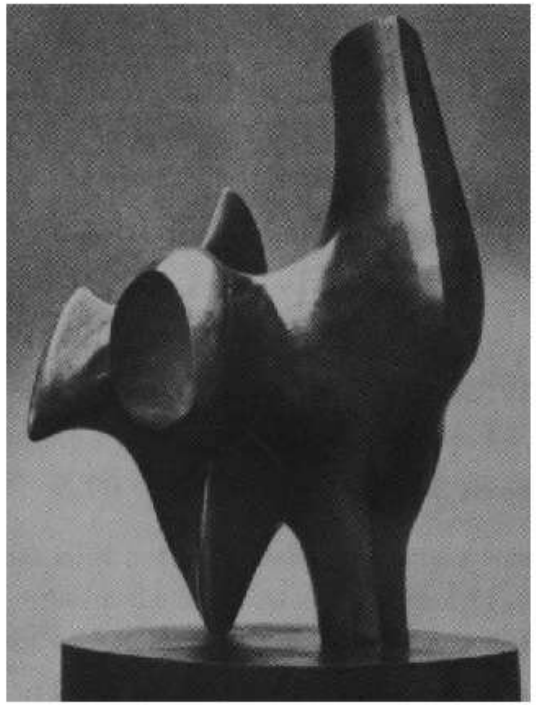
从 几何模型：基于几何原理和相对位置关系的模型

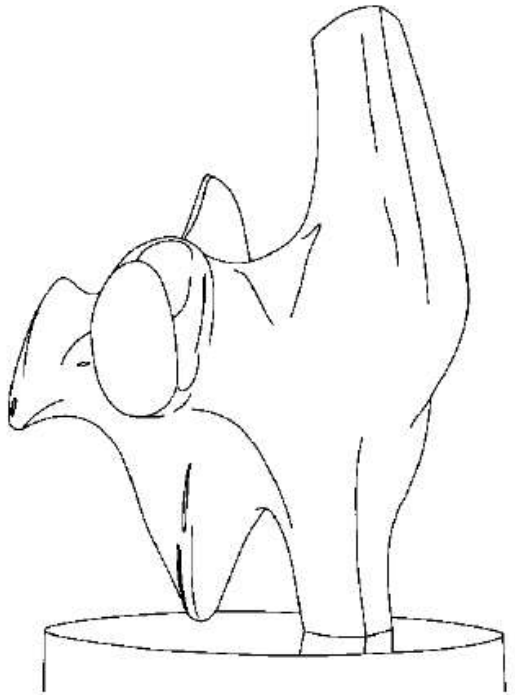
到 **统计模型：基于数据分布和统计推断的模型**

我们需要统计模型快速合理地进行感知推断

边缘检测

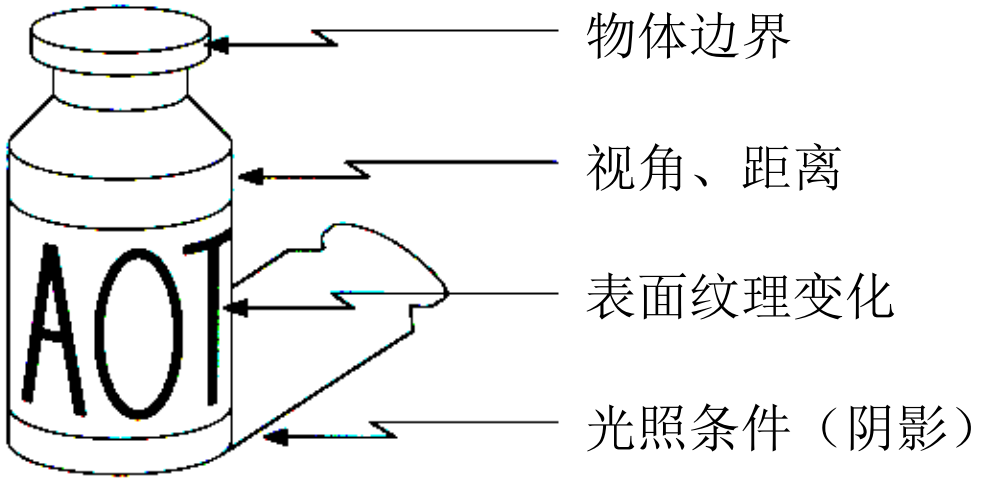
Edge detection 边缘检测





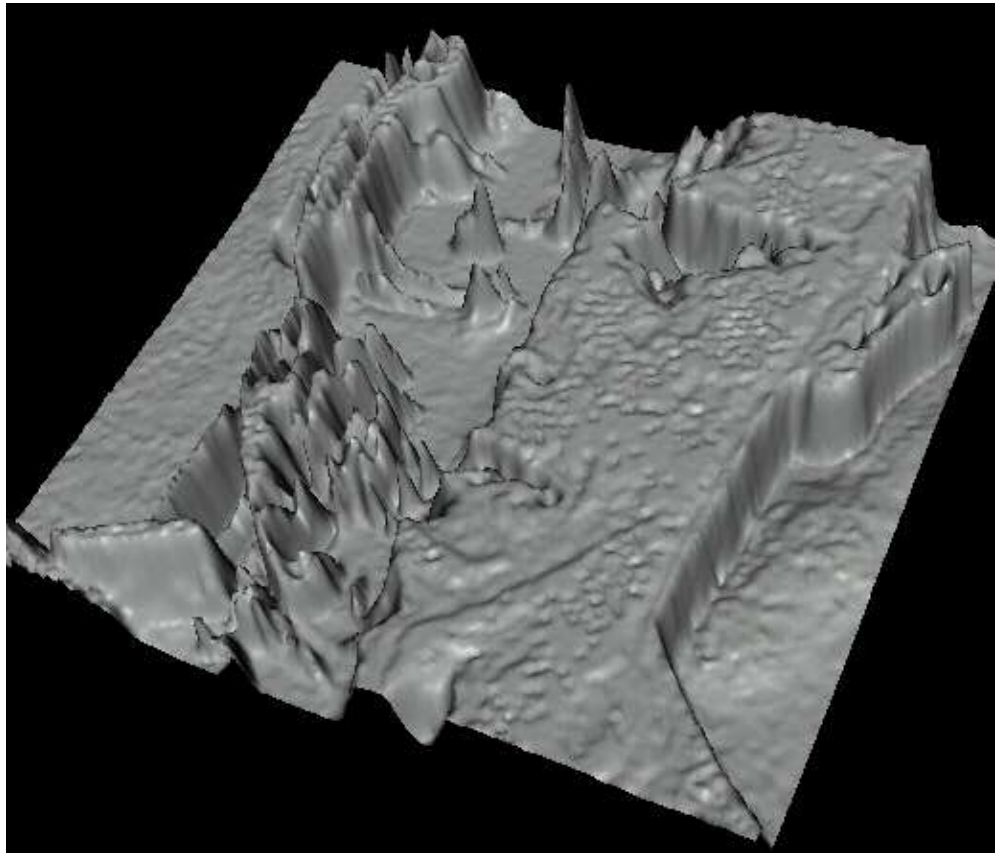
- 把2维图像转成曲线的集合
  - 可以提取场景中的显著特征
  - 比像素的信息量更大（简图）

什么是边缘？



- 很多因素可以导致边缘的出现

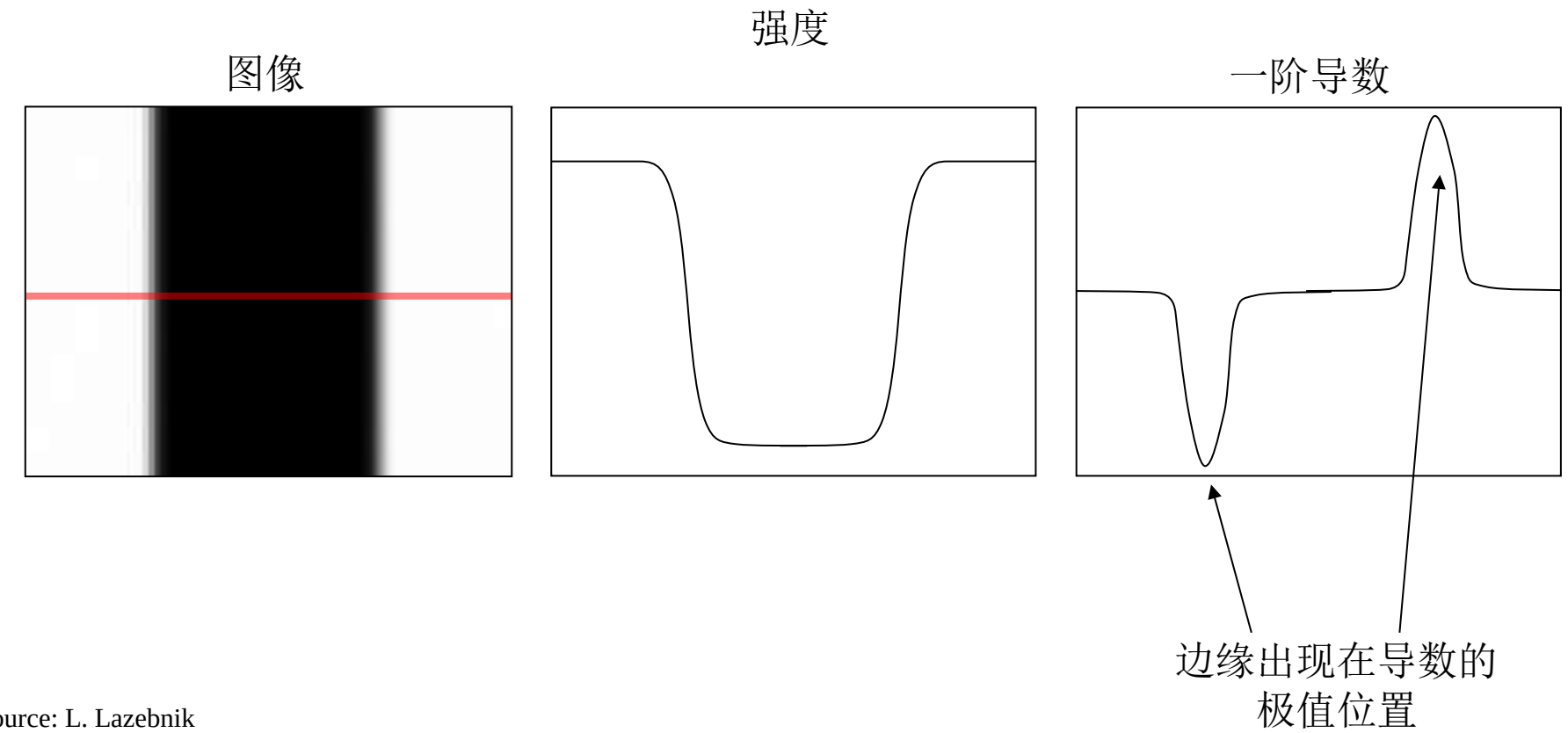
回顾：图像可以认为是一个像素位置到像素值的映射函数



- 边缘可以认为是那些“悬崖”

定义边缘

- 边缘时图像强度发生突变的区域



Source: L. Lazebnik

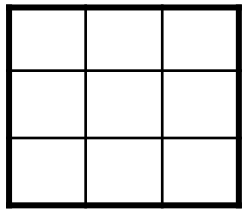
Image derivatives： 图像斜率

- 如何计算图像F[x,y]的“斜率”？
  - Option 1: 把F转换为连续函数 *f*
  - Option 2: 做离散梯度discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

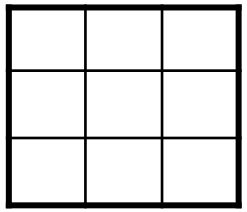
可以用卷积实现

$\frac{\partial f}{\partial x}$:



H_x

$\frac{\partial f}{\partial y}$:



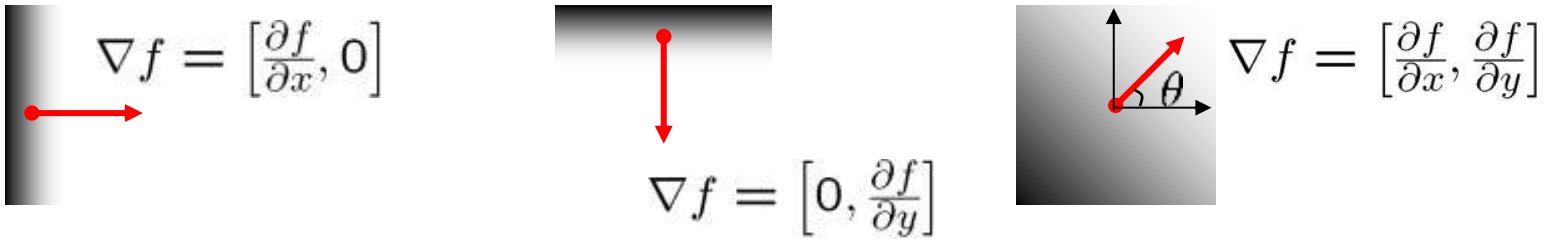
H_y

Source: S. Seitz

Image gradient： 图像梯度

- 图像梯度定义为: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

这些梯度会指出哪些方向图像会发生剧烈的变化



梯度的强度定义了边缘的强度:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

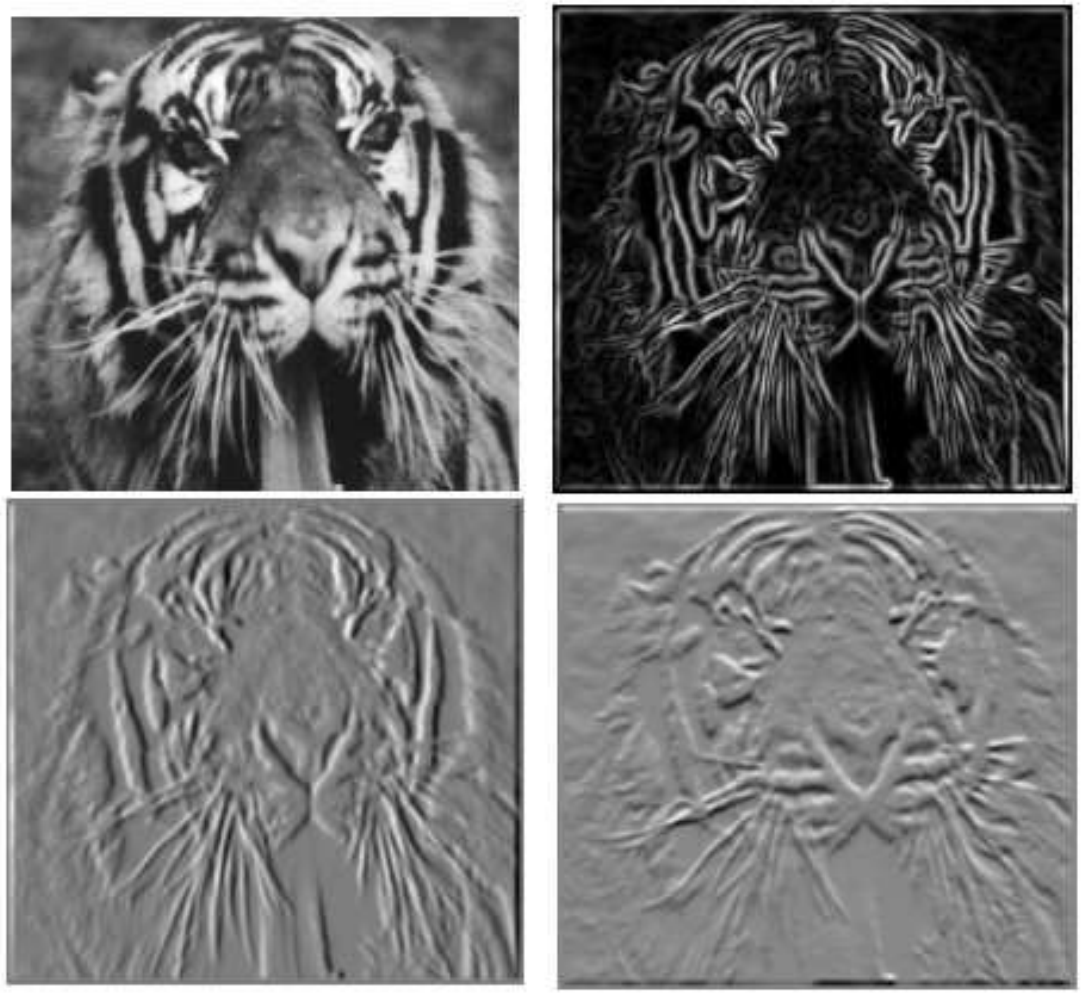
我们可以计算梯度的方向:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- 梯度的方向可以反应边缘的方向，他们是什么关系呢?

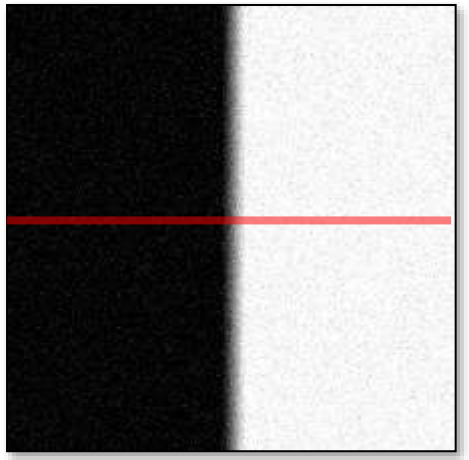
Source: Steve Seitz

图像梯度



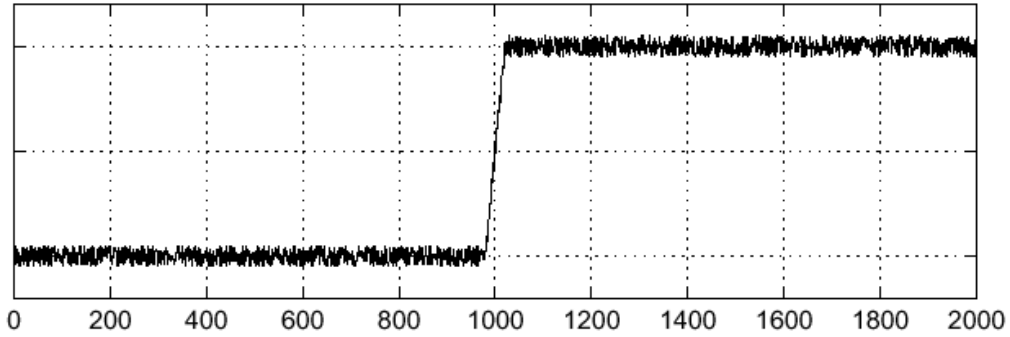
Source: L. Lazebnik

噪声的影响

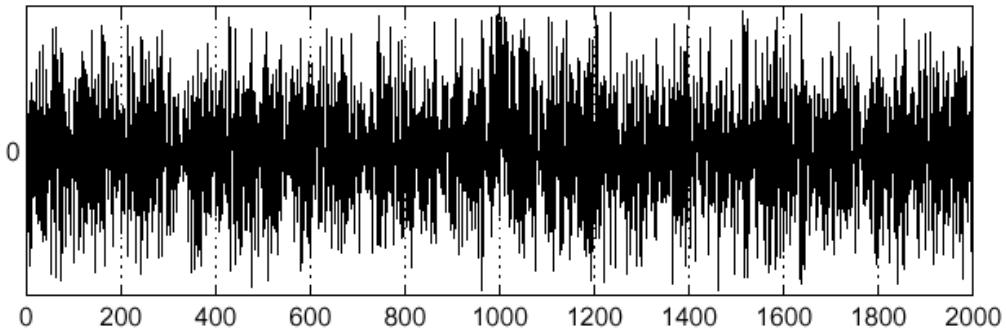


有噪声的图像

$f(x)$



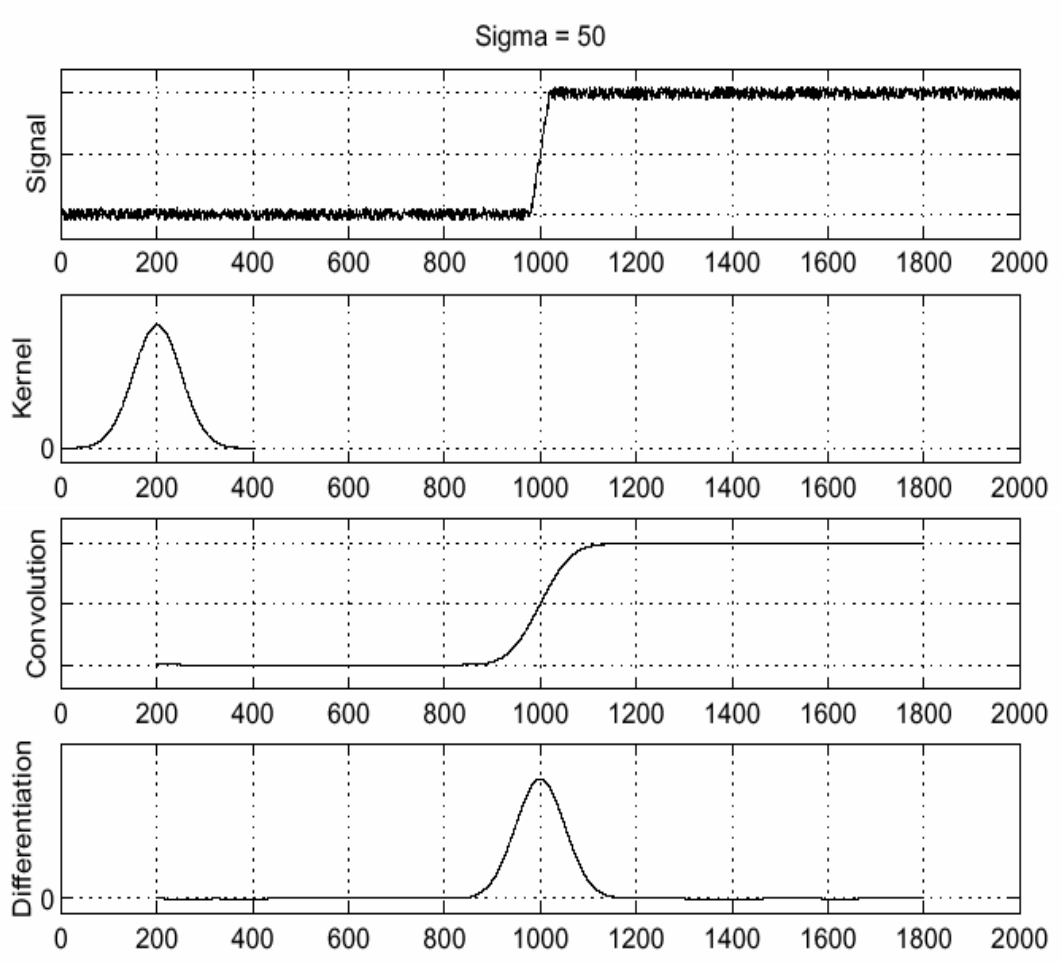
$\frac{d}{dx}f(x)$



哪里是边缘?

Source: S. Seitz

解决方案：高斯模糊



f

h

$f * h$

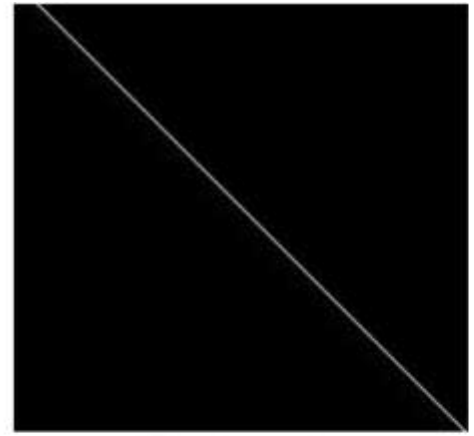
$\frac{d}{dx}(f * h)$

如果要找边缘，找峰值

$\frac{d}{dx}(f * h)$

Source: S. Seitz

Image with Edge



Edge Location

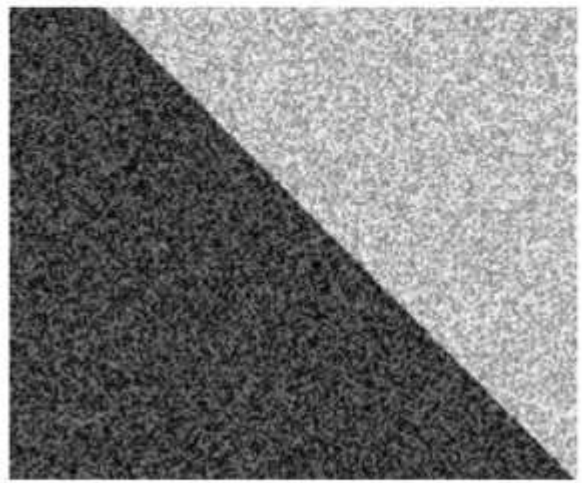
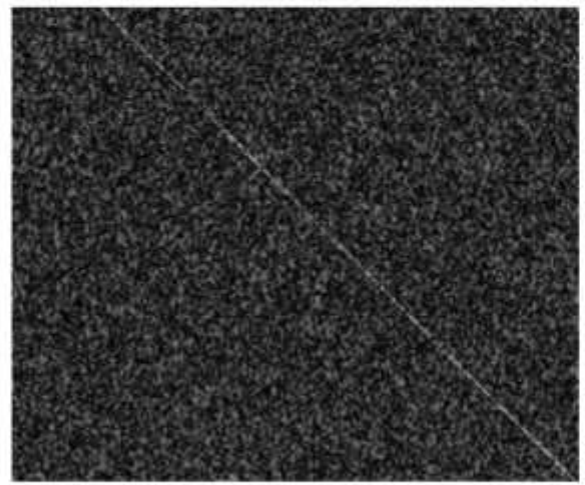
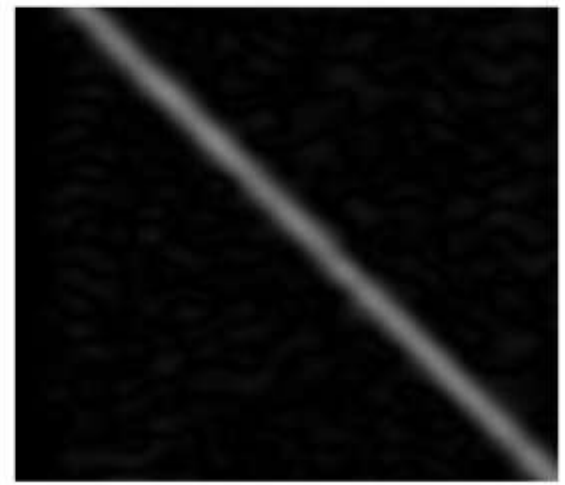


Image + Noise



Derivatives detect
edge *and* noise



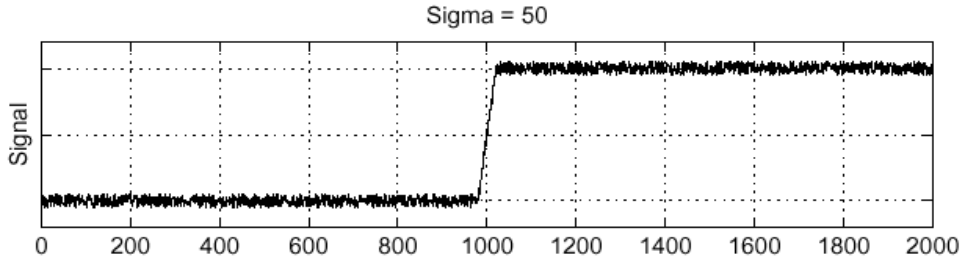
Smoothed derivative removes
noise, but blurs edge

卷积是符合交换律的

- 计算梯度可以用卷积实现，而卷积有交换律: $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$

- 我们可以把计算梯度和模糊结合在一起:
  - Derivative of Gaussian （DoG）

f

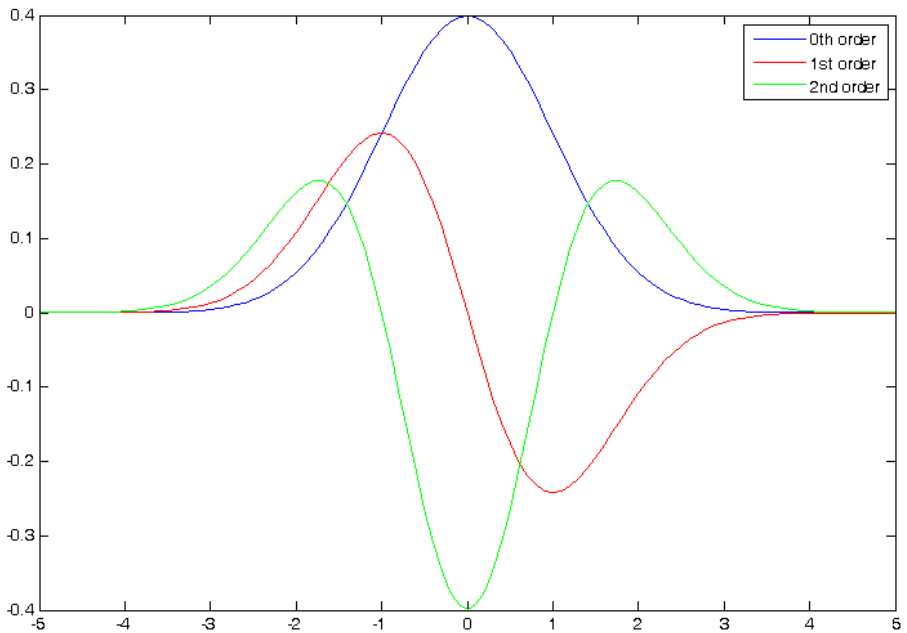


Source: S. Seitz

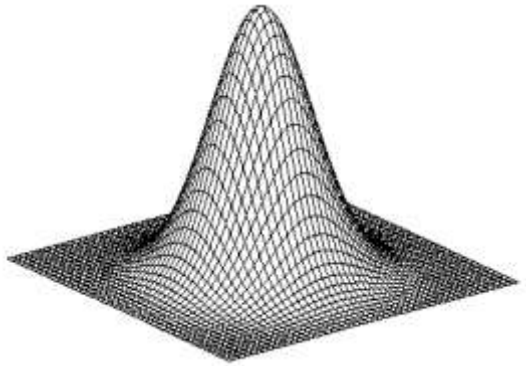
一维高斯函数和DoG

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma}\left(\frac{x}{\sigma}\right) G_\sigma(x)$$

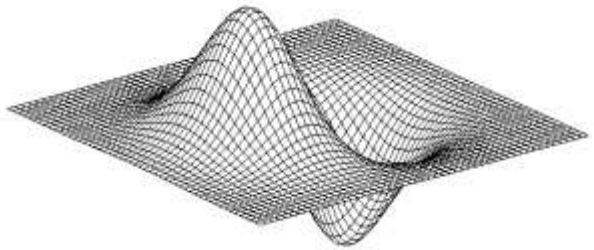


二维高斯函数和DoG



Gaussian

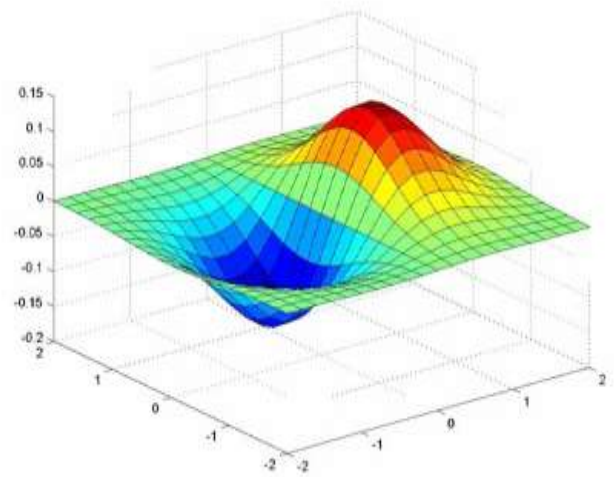
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



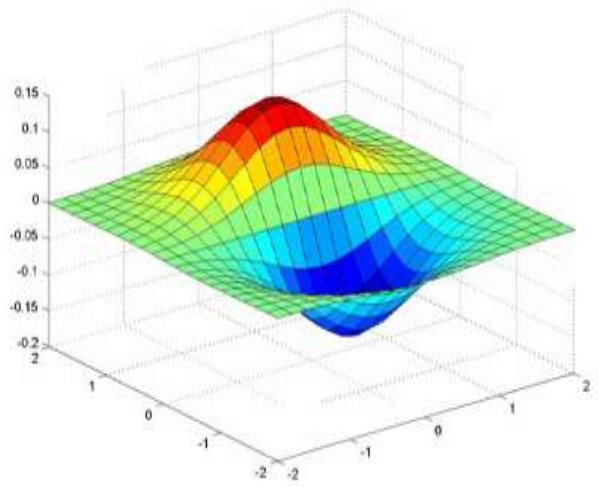
derivative of Gaussian (*x*)

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

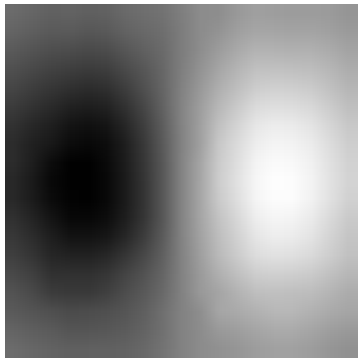
不同方向的DoG

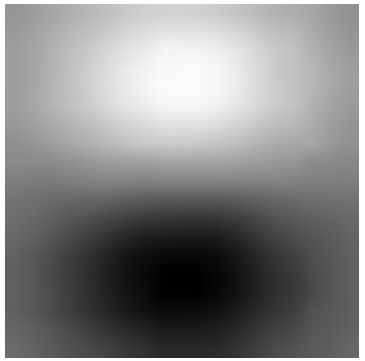


x-direction



y-direction





Sobel 算子

- 经常用来替换DoG

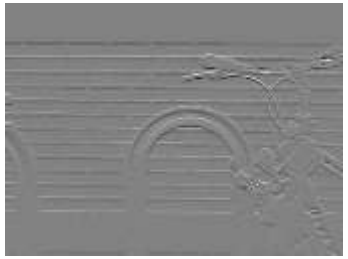
$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad s_x$$

$$\frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad s_y$$

- Sobel算子通常比DoG更加计算效率，因为它使用的是简单的3x3卷积核，而DoG通常需要更大的卷积核和更多的计算。
  - 计算简单，速度快。对于基础的边缘检测任务通常足够有效。
  - 对噪声比较敏感。可能无法检测到更复杂或微妙的边缘。

Sobel算子





Source: Wikipedia

效果

原图

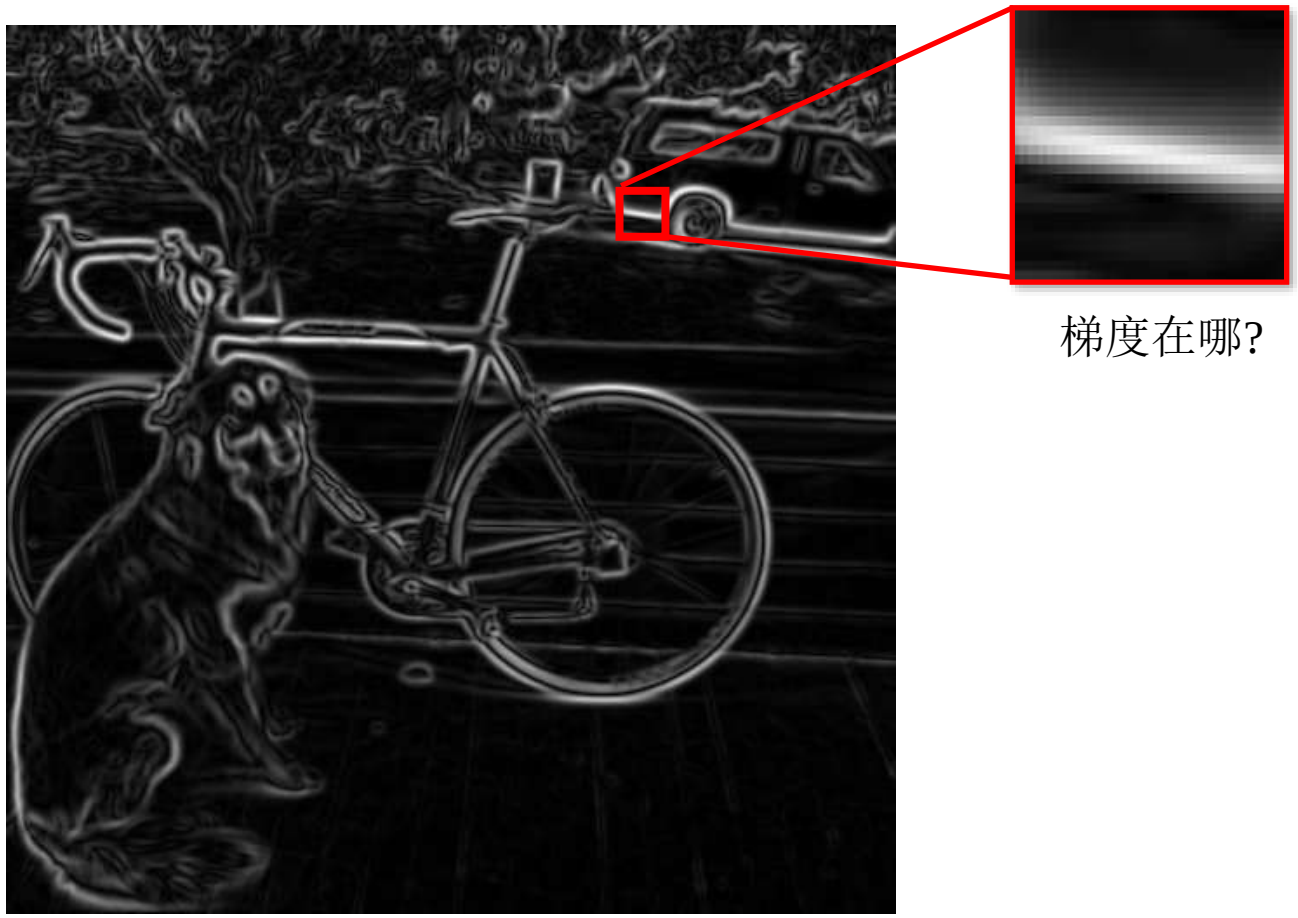
Demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Image credit: Joseph Redmon

找出边缘

平滑梯度幅度

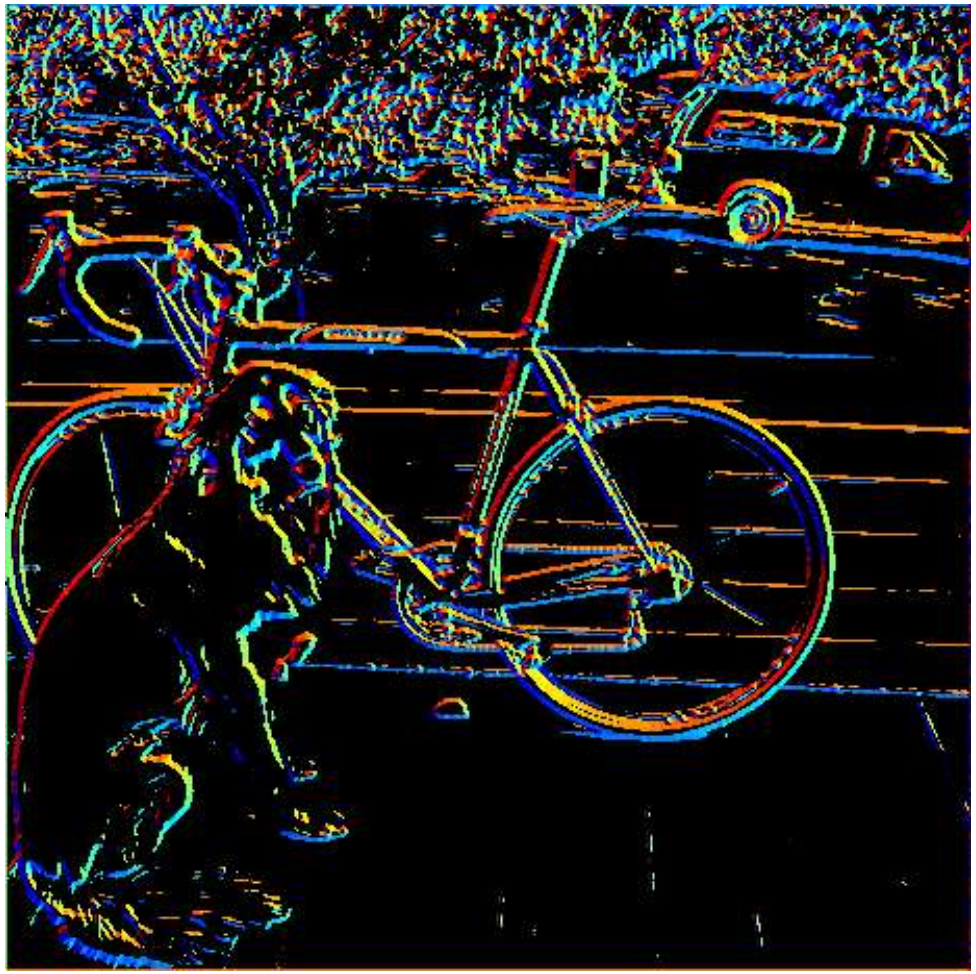
找到梯度



梯度在哪?

利用阈值

找到每个像素的方向



theta = atan2(gy, gx)

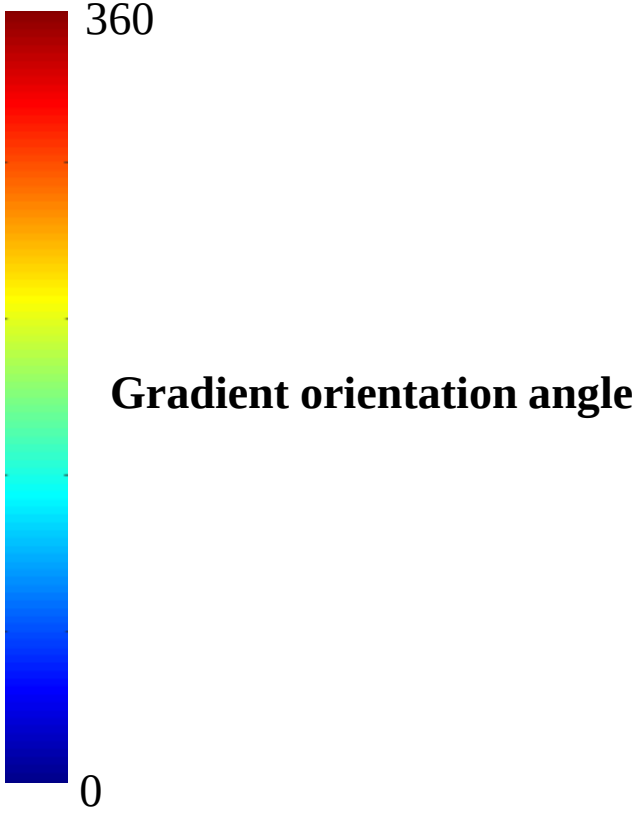
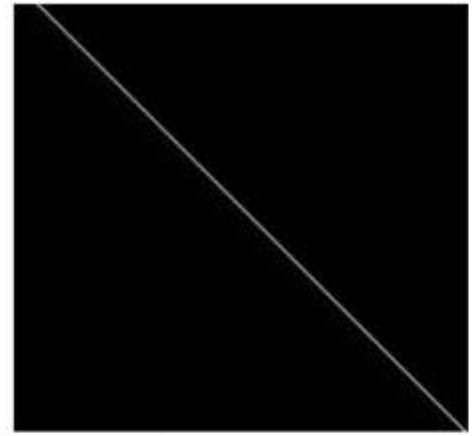


Image with Edge



Edge Location

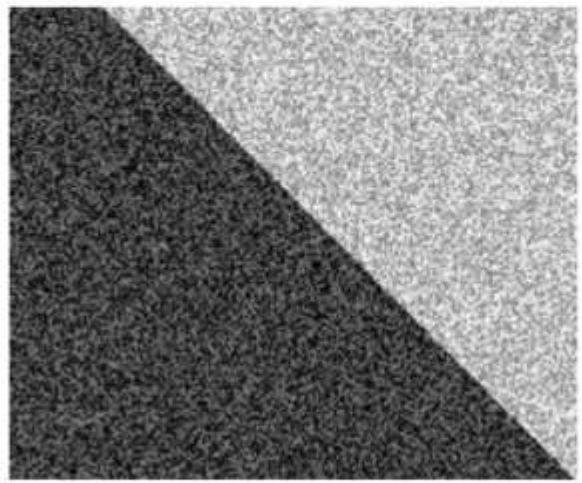
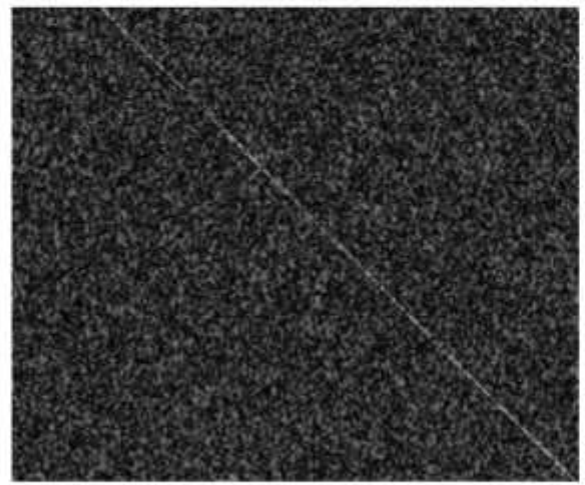
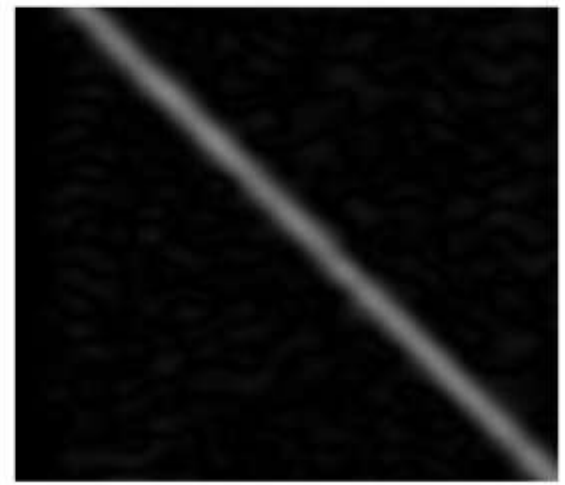


Image + Noise



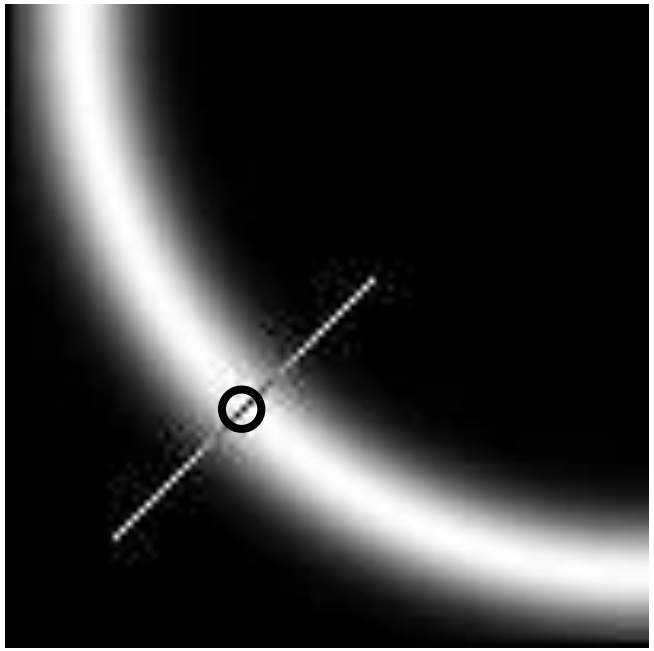
Derivatives detect
edge *and* noise

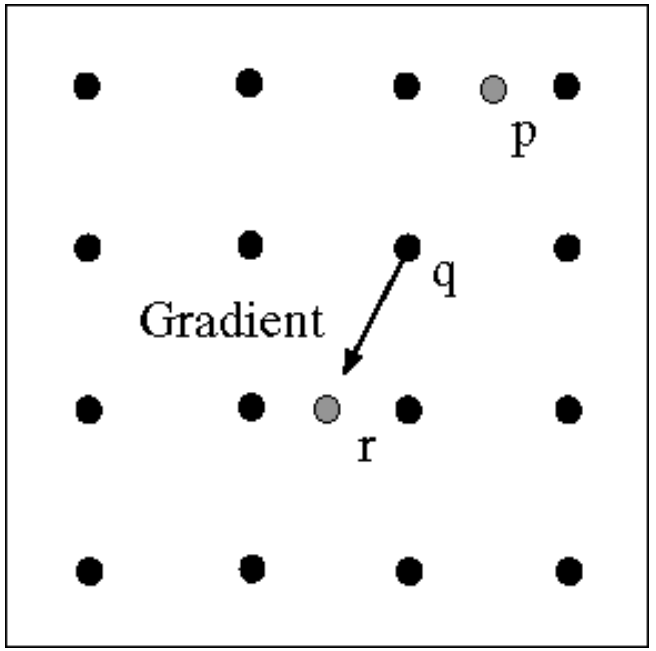


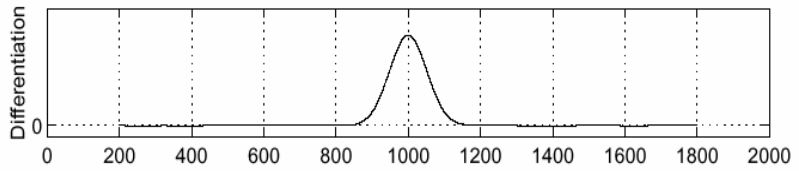
Smoothed derivative removes
noise, but blurs edge

Non-maximum suppression: 非极大抑制

将“厚边缘”细化为“细边缘”







- 对于图像中的每一个像素点，确定其梯度的方向，计算该方向子区域的梯度幅度，在该方向上，如果该像素点的梯度幅度不是局部最大值，则将其设置为0。

Before Non-max Suppression

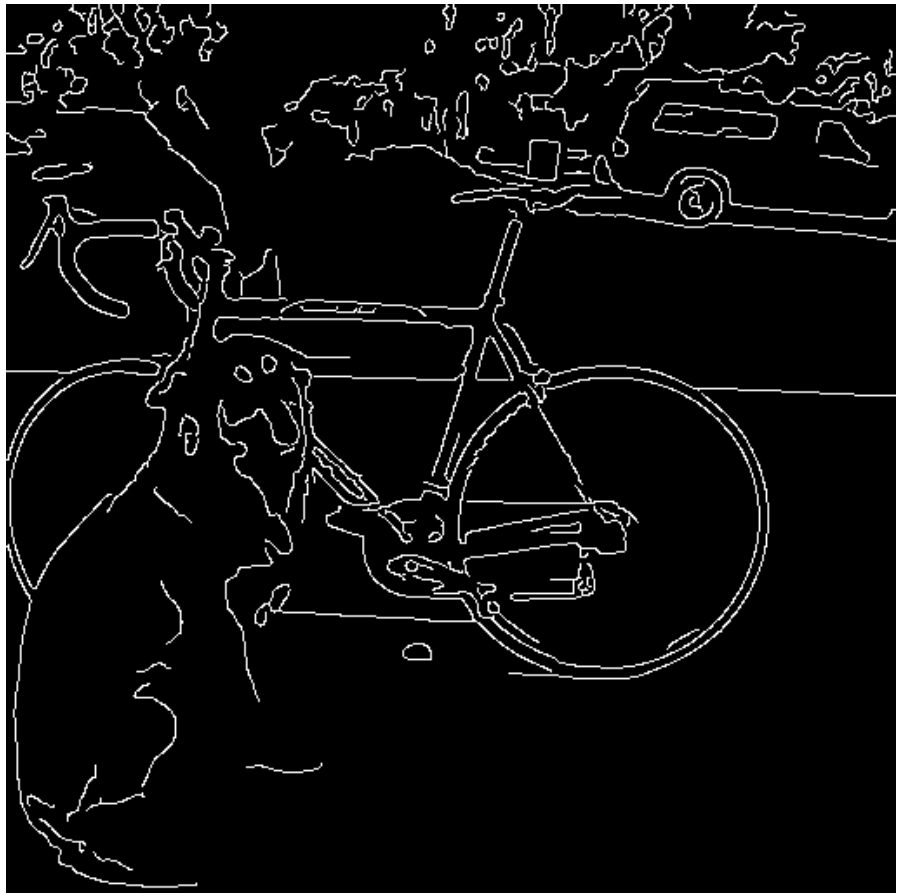
After Non-max Suppression

通过阈值得到边缘

- 噪声依然存在
- 我们想要一些明显边缘
- 2 个阈值, 3 种情况
  - R > T: 强边缘
  - R < T but R > t: 弱边缘
  - R < t: 非边缘
- 为什么要两个阈值?

连接边缘

- 强边缘一定是边缘
- 如果弱边缘与强边缘联通则也是边缘
- 通过周围八个像素点进行搜索



J. Canny, *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

MATLAB: **edge(image,'canny')**

Canny边缘检测

1. 使用DoG对图像预处理
2. 找到梯度的幅度和方向
3. Non-maximum suppression
4. 连接边缘:
   - 定义高低两个阈值，得到弱边缘和强边缘
   - 用强边缘作为初始边缘，用弱边缘连接强边缘

Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny边缘检测器

original

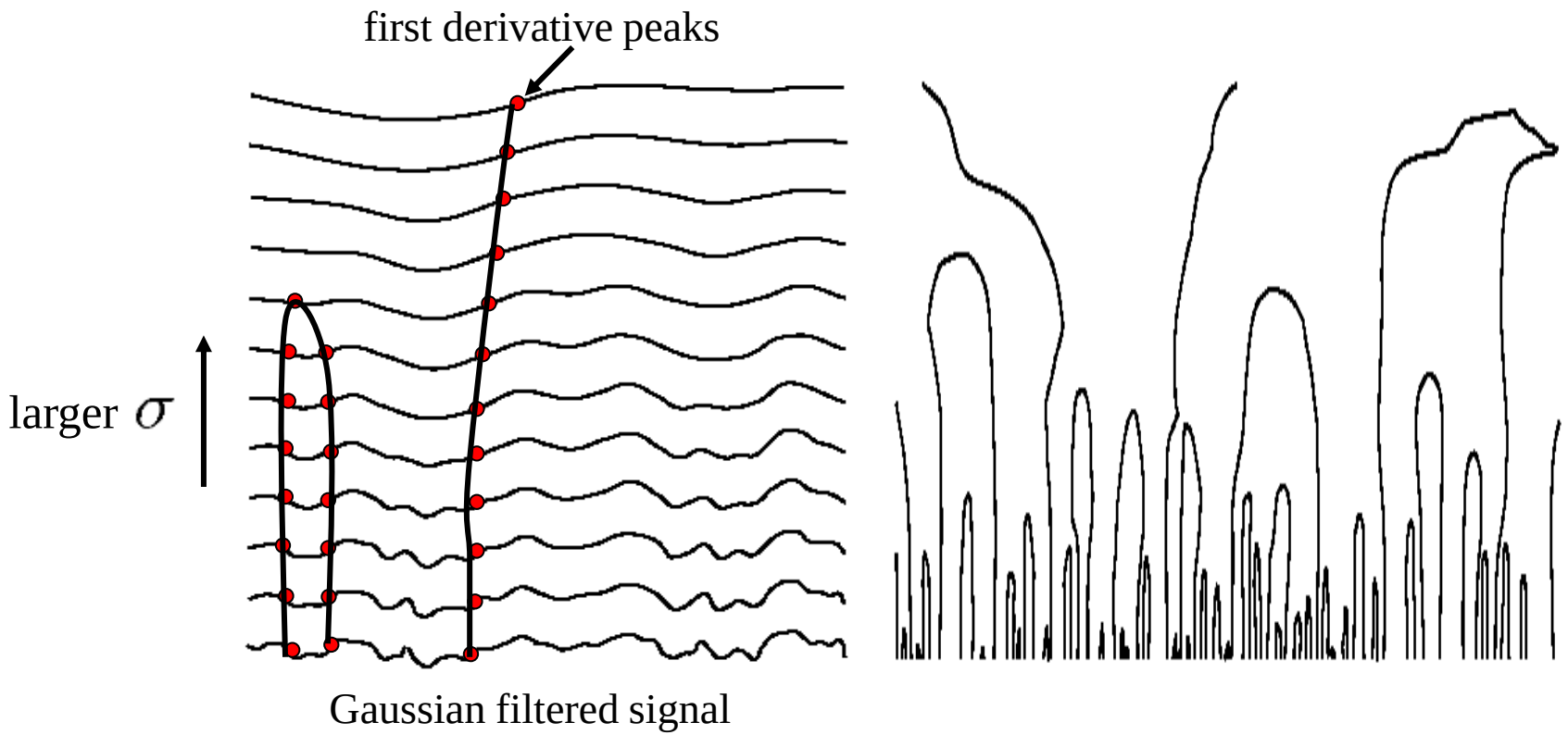
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- 高斯滤波带宽σ影响最终结果
  - 高带宽边缘更“大尺度”
  - 低带宽边缘更“细粒度”

Source: S. Seitz

尺度空间[Witkin 83]

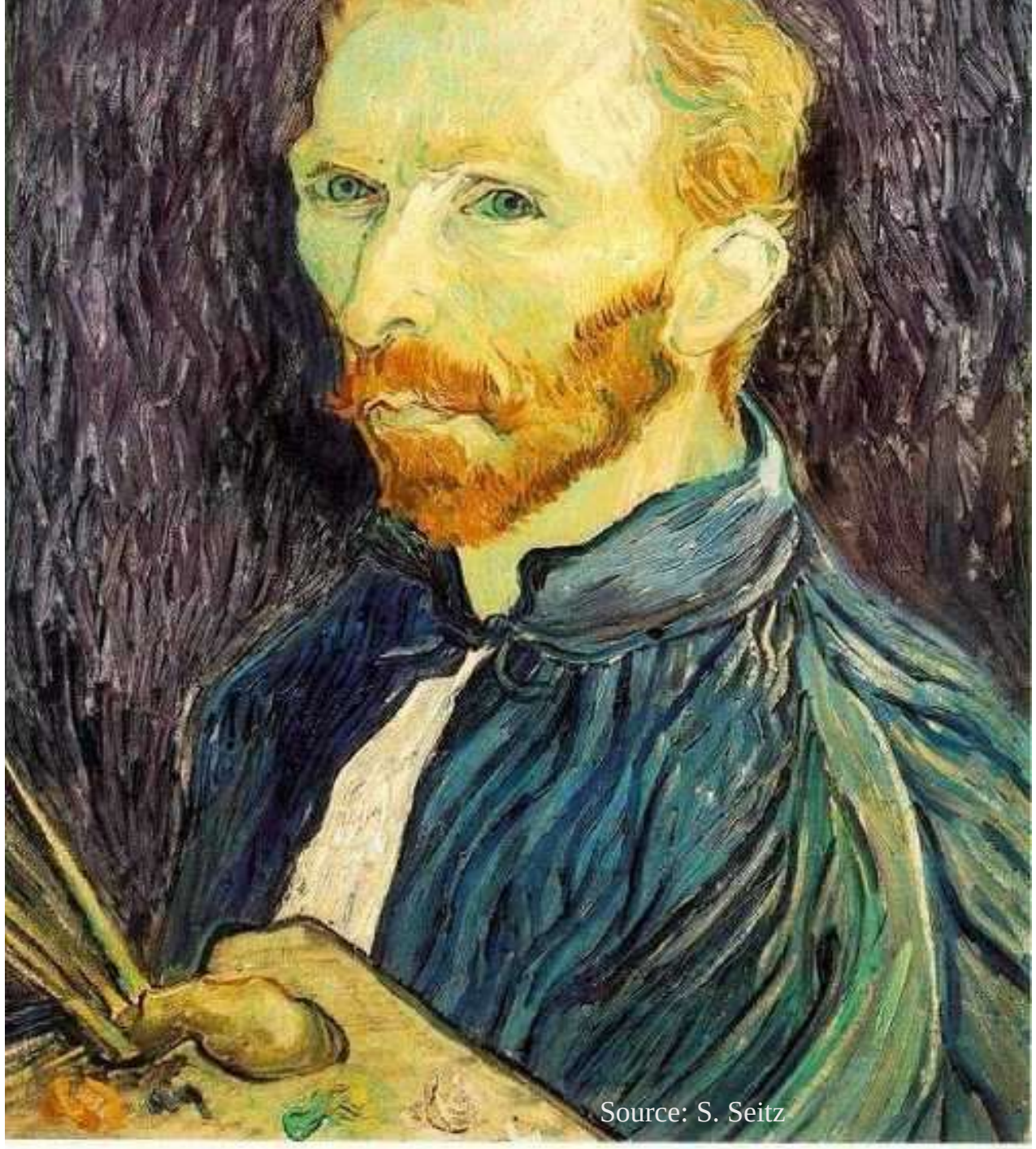


- 更高的尺度下（更大的带宽）：
  - 边缘的位置可能改变
  - 边缘可能合并
  - 但边缘不会再分开

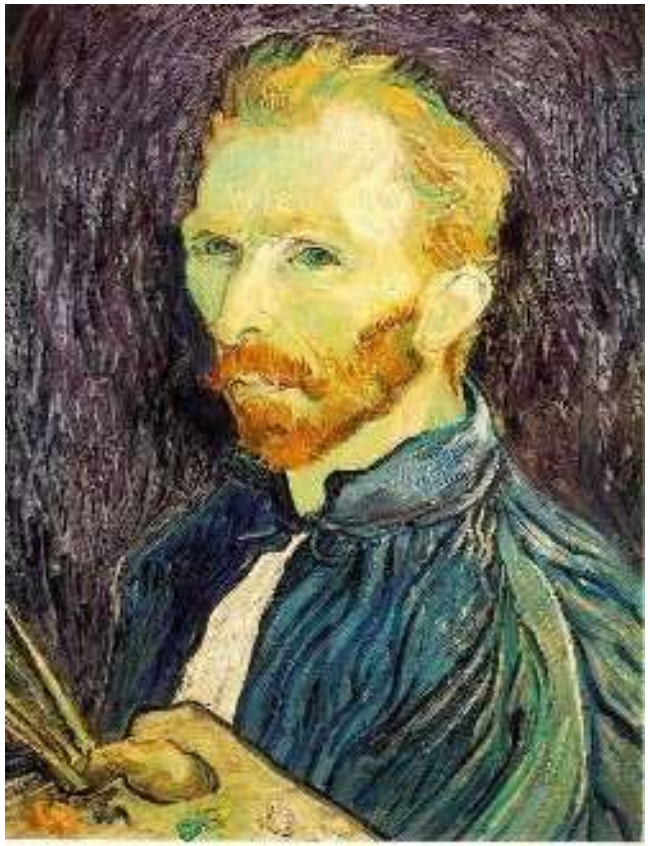
采样与插值

图像尺度

如果图像太大了放不到屏幕里，我们怎样制作一张更小的图像？



Sub-sampling 下采样



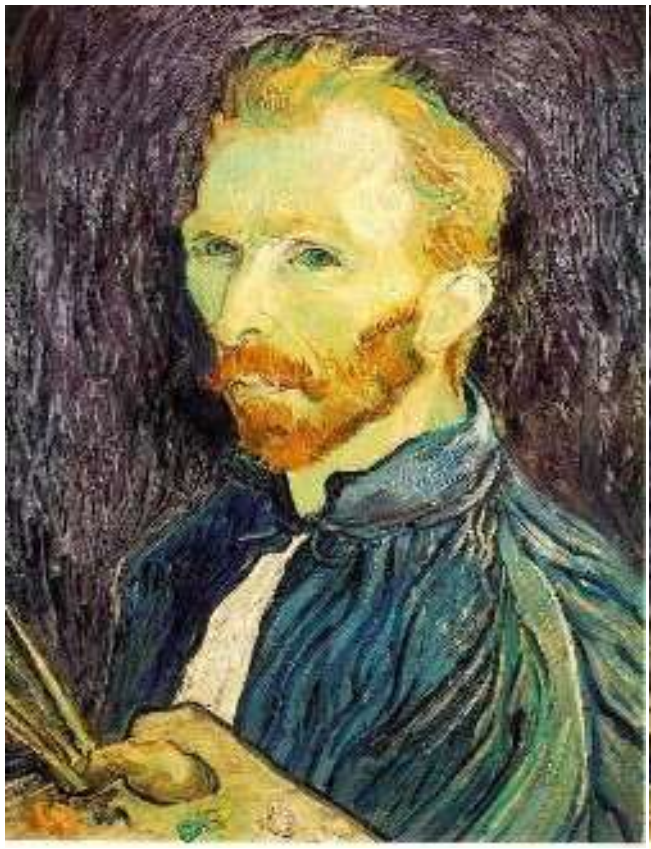
1/4

1/8

隔一行隔一列扔掉一半的图像 - 图像下采样 *image sub-sampling*

Source: S. Seitz

图像下采样



1/2

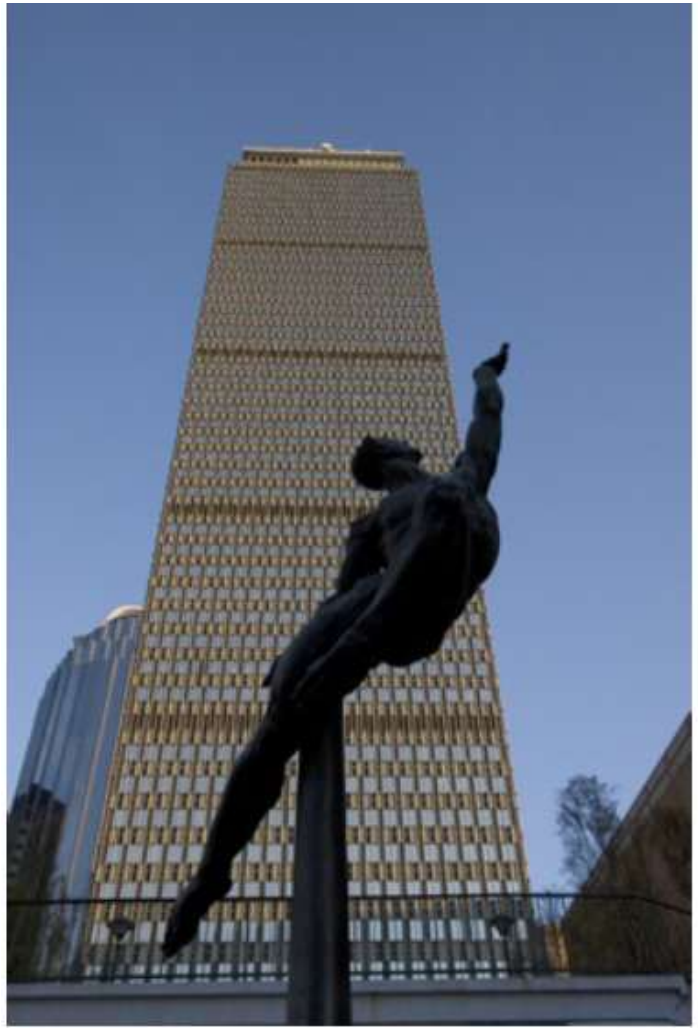
1/4 (2x zoom)

1/8 (4x zoom)

如果再上采样回来，为什么看起来这么粗糙?

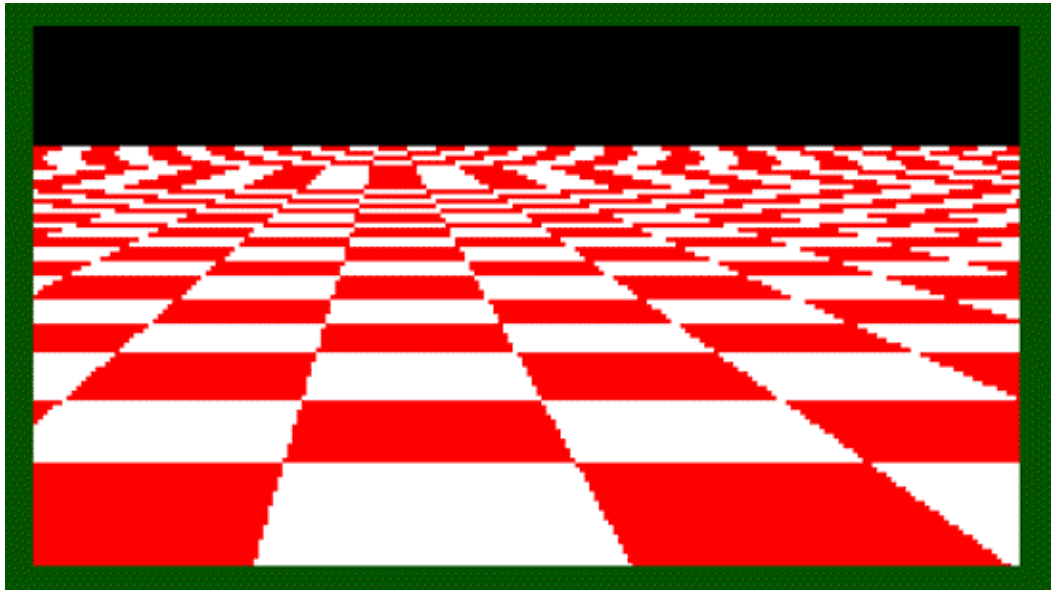
Source: S. Seitz

图像下采样：变形



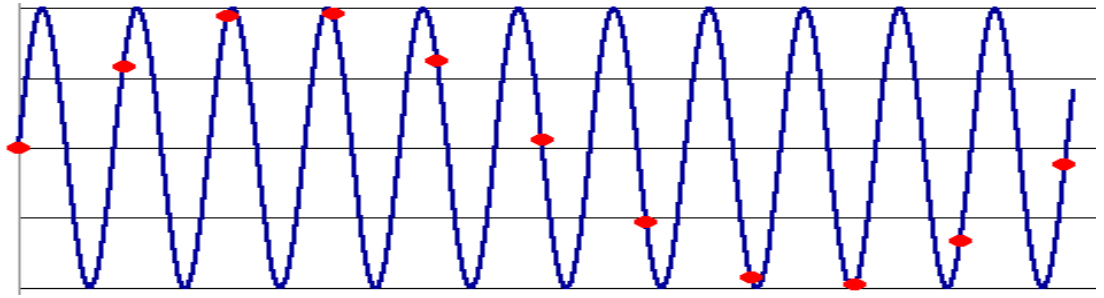
Source: F. Durand

生成图像下采样：变形



Source: L. Zhang

Aliasing：混淆

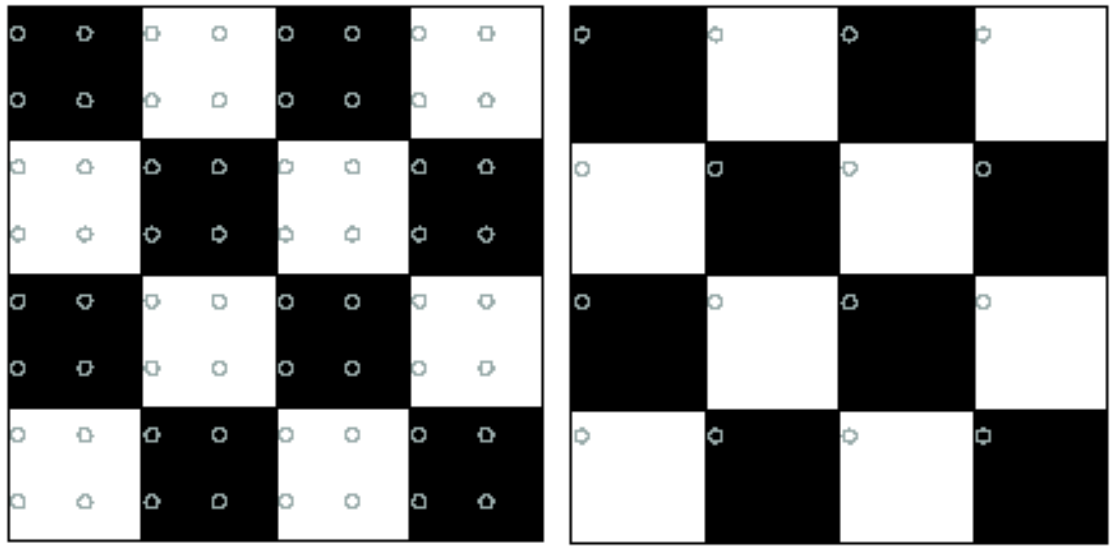


- 当采样率不够高的时候，有可能会导致采样的信息有问题
- 导致得到的信号出现问题（频率变低）导致 *alias 混淆*

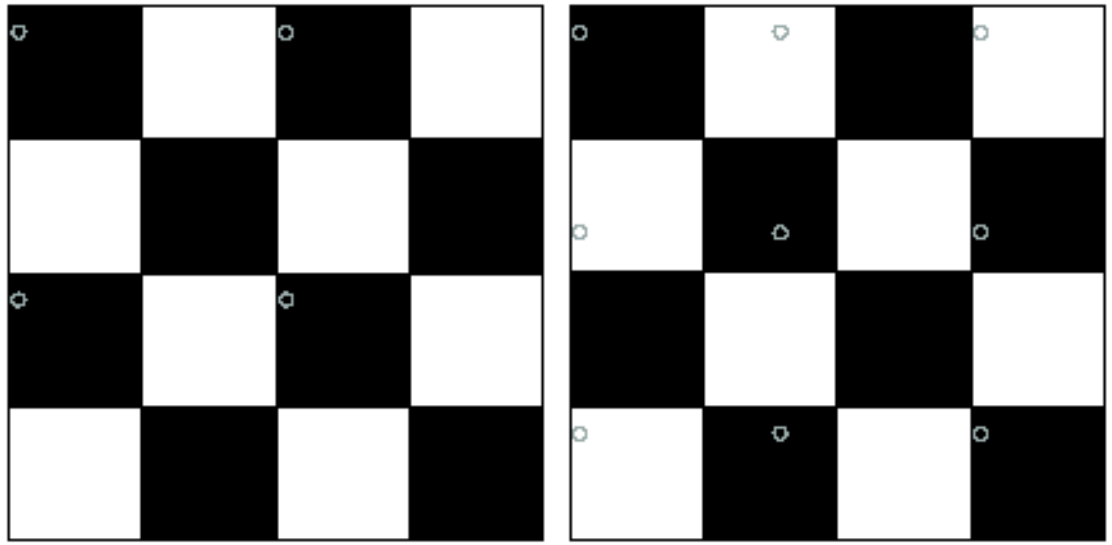
- 如何正确地采样？我们需要正确地理解信号、图像的结构
- 傅里叶变换的相关知识
  - “But what is the Fourier Transform? A visual introduction.” https://www.youtube.com/watch?v=spUNpyF58BY
- 为了避免混淆:
  - 采样率要不小于图像最大频率的2倍
    - 也就是说，每个周期至少两个采样
  - **最小采样率也被称为Nyquist rate**

Source: L. Zhang

Nyquist limit – 2D的例子



好的采样



坏的采样

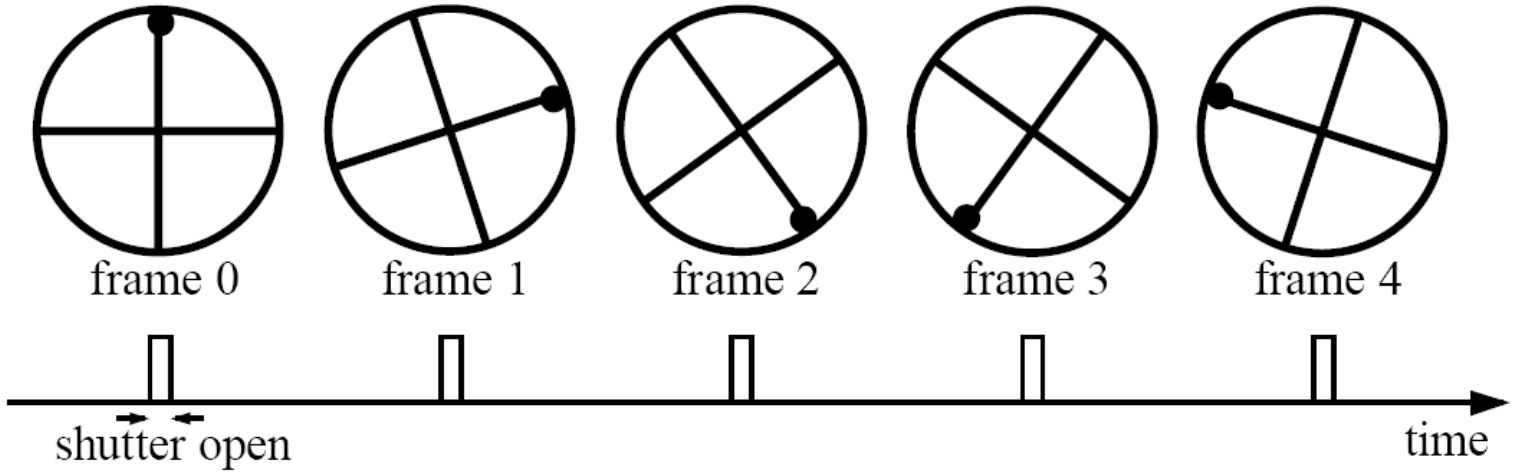
Wagon-wheel effect

https://en.wikipedia.org/wiki/Wagon-wheel_effect

Wagon-wheel effect

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

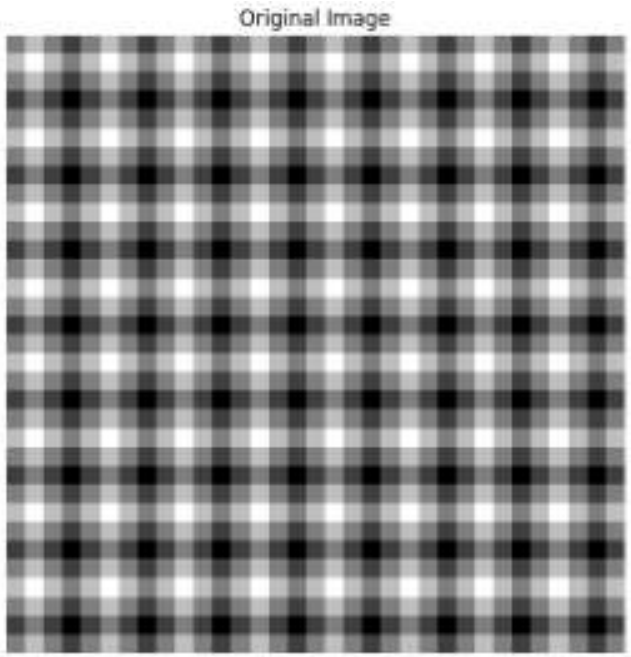
(See http://www.michaelbach.de/ot/mot-wagonWheel/index.html)

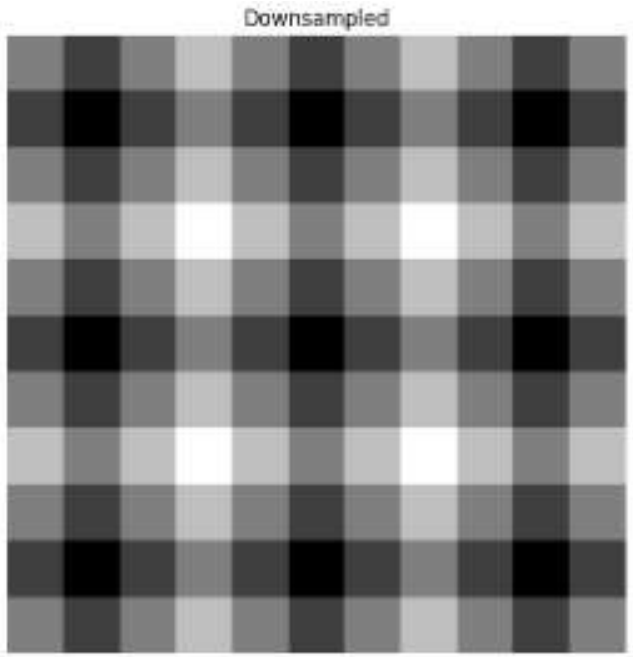
Source: L. Zhang

Aliasing 混淆

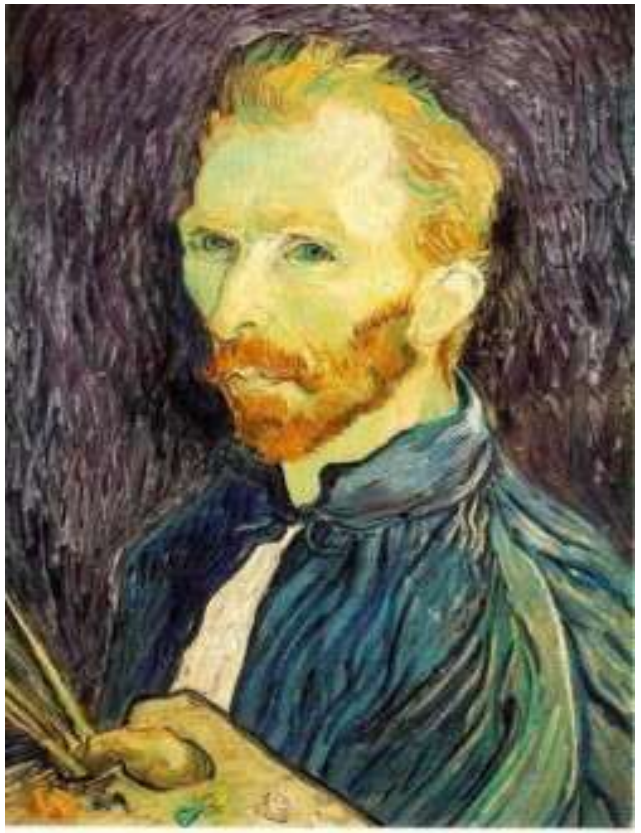
- 当下采样的比例大于2的时候
  - 原来的像中的有些内容频率可能过高，导致出现混淆

- 怎么解决这个问题?





高斯模糊预处理



Gaussian 1/2

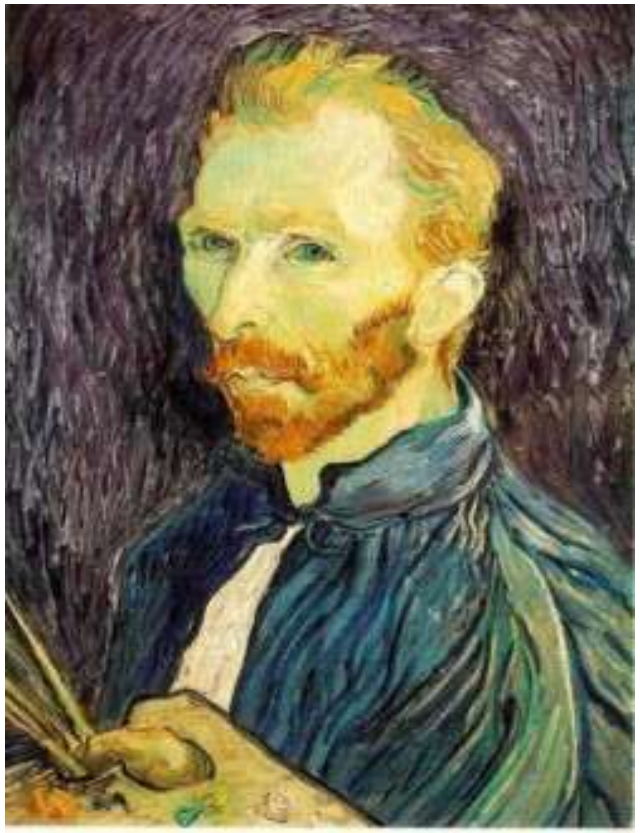
G 1/4

G 1/8

- 解决方案：先模糊，再下采样

Source: S. Seitz

使用高斯模糊以后的下采样



Gaussian 1/2

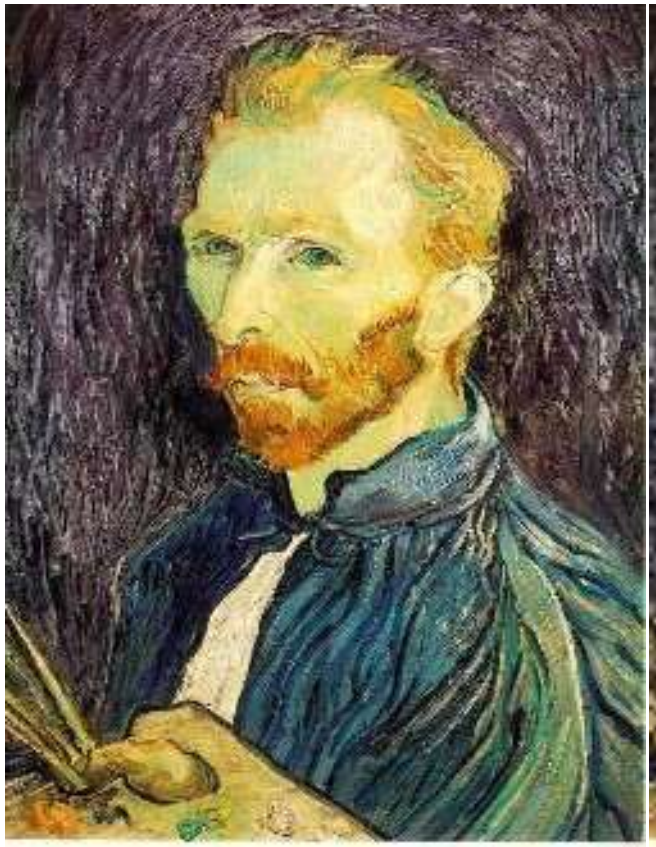
G 1/4

G 1/8

- 解决方案：先模糊，再下采样

Source: S. Seitz

如果不这么做...



1/2

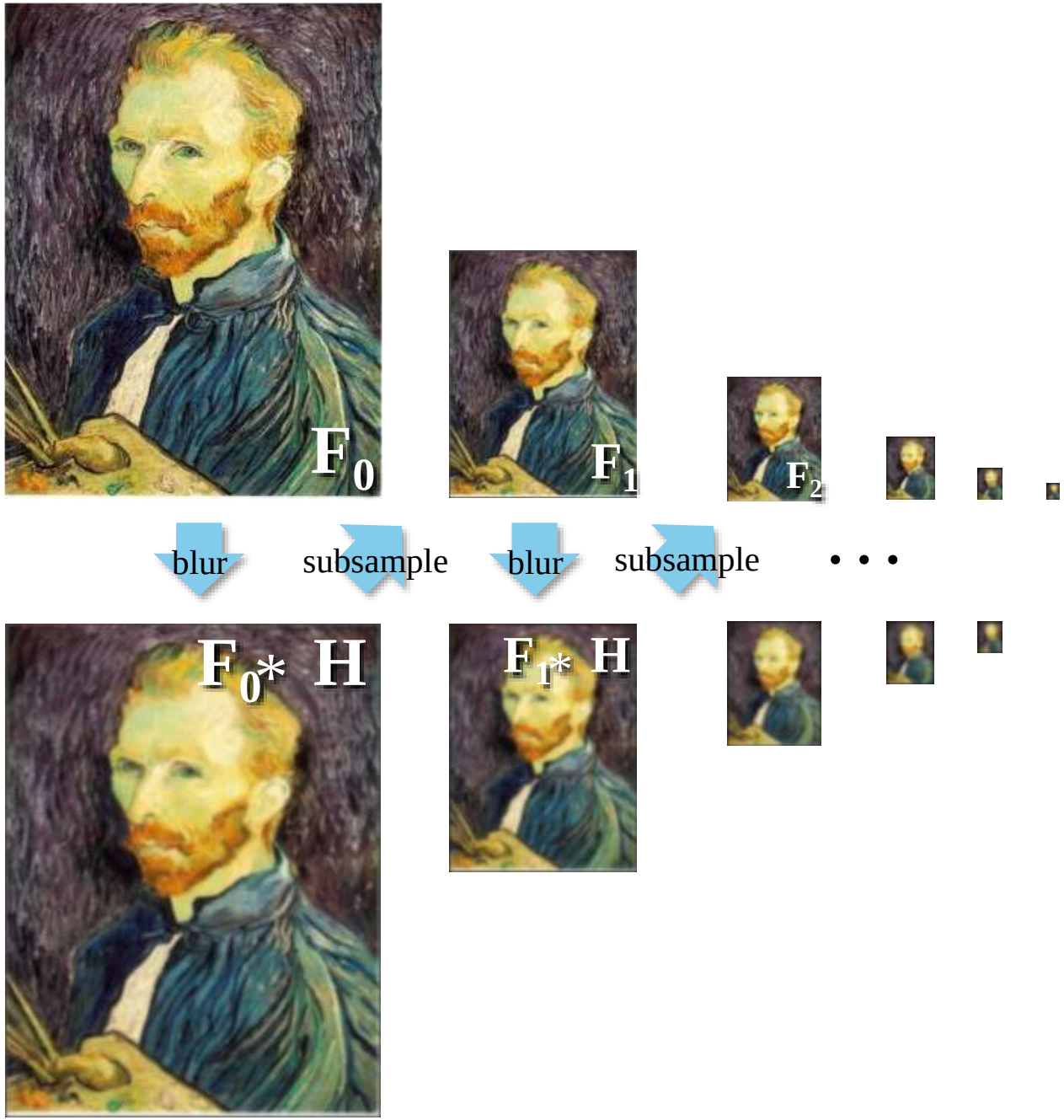
1/4 (2x zoom)

1/8 (4x zoom)

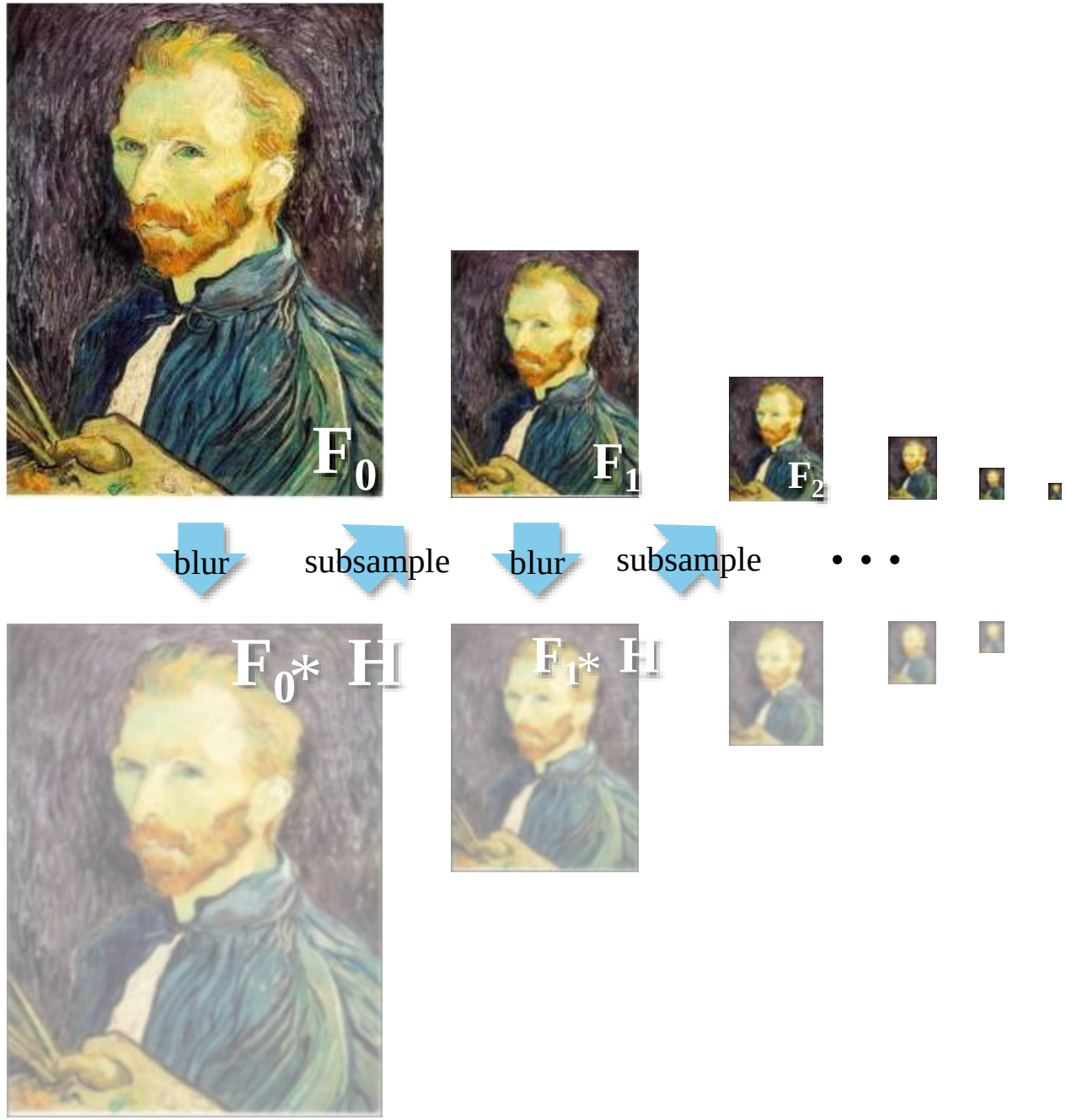
Source: S. Seitz

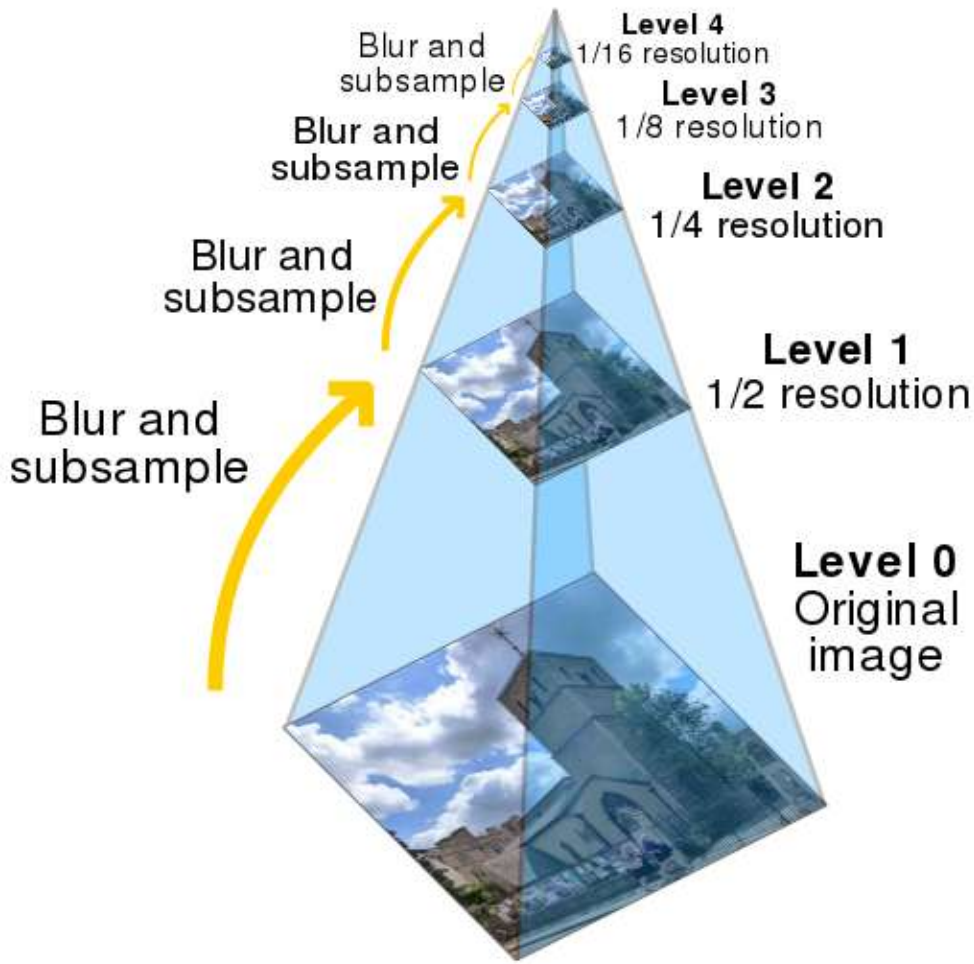
高斯模糊

- 解决方案：先模糊，再下采样

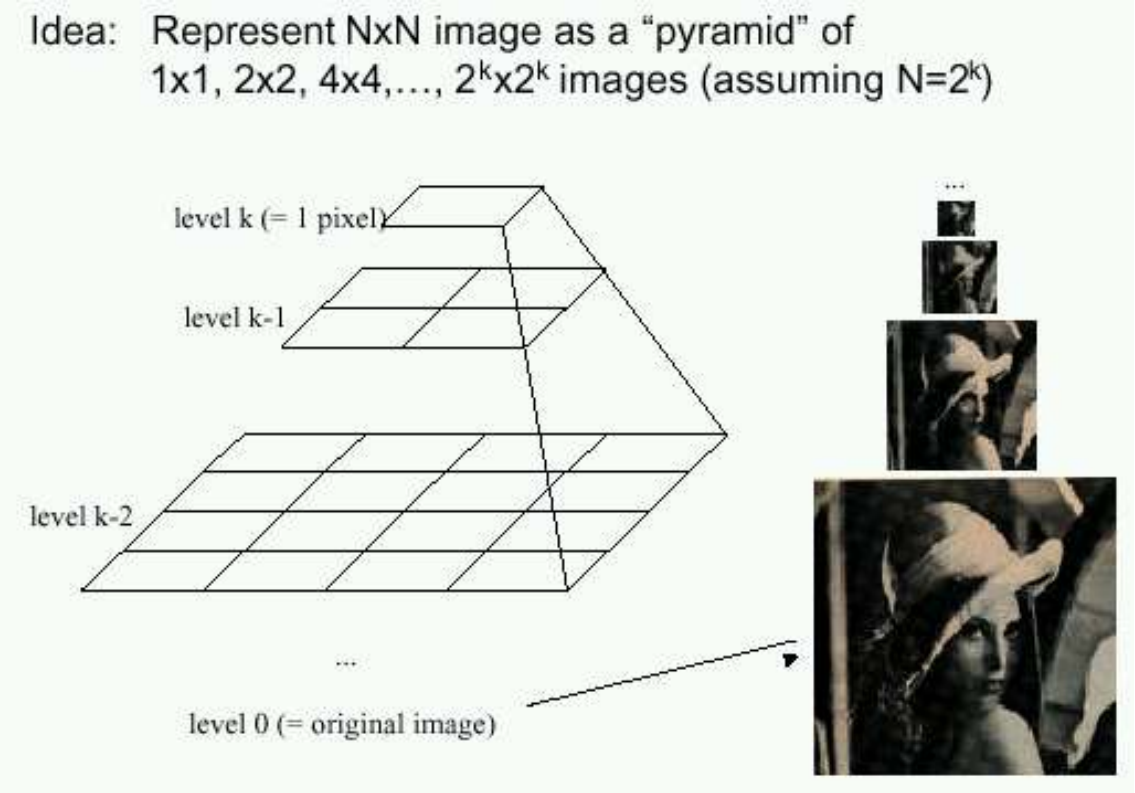


Gaussian pyramid
高斯金字塔




Blur and subsample
Blur and subsample
Blur and subsample
Blur and subsample
Level 4
1/16 resolution
Level 3
1/8 resolution
Level 2
1/4 resolution
Level 1
1/2 resolution
Level 0
Original image

Gaussian pyramids [Burt and Adelson, 1983]

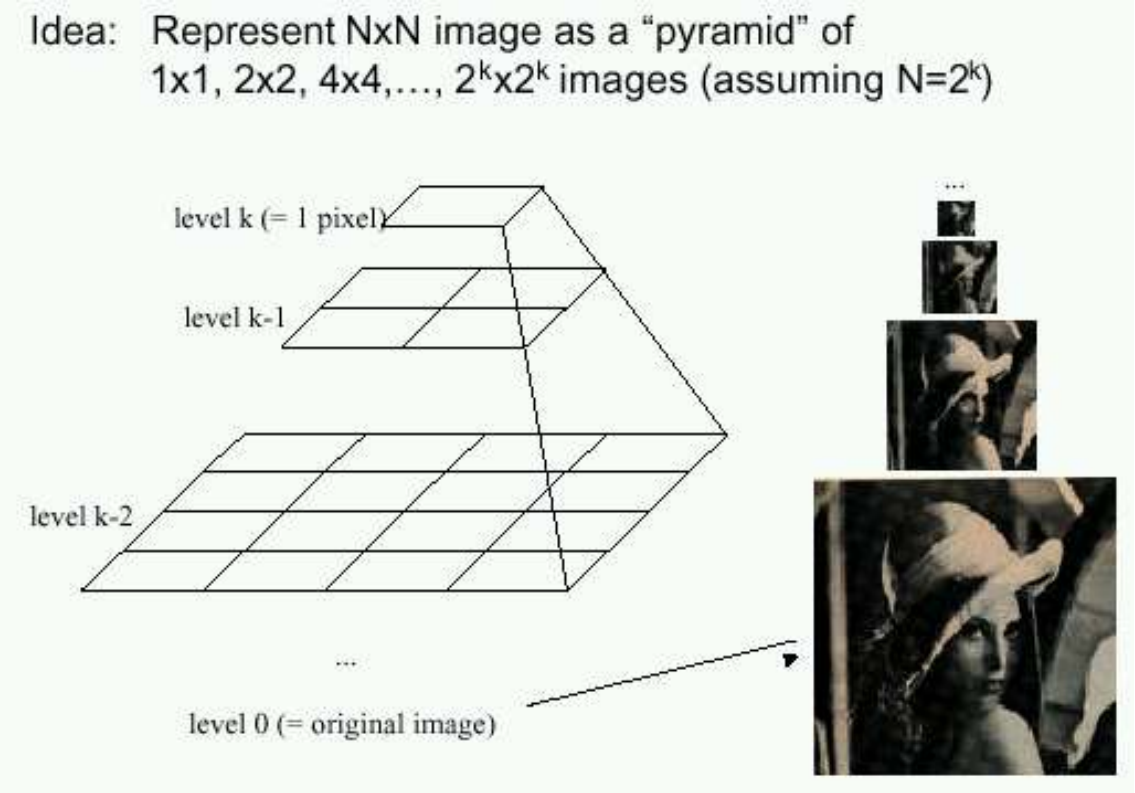


- *mip map* [Williams, 1983] 纹理映射和反走样的技术。它包含了一系列预先计算和存储的图像，每个图像都是原始纹理的一个下采样版本。
- *wavelet transform在不同频率成分和不同位置的数学变换。*

我们未来还会再遇到这个金字塔结构

Source: S. Seitz

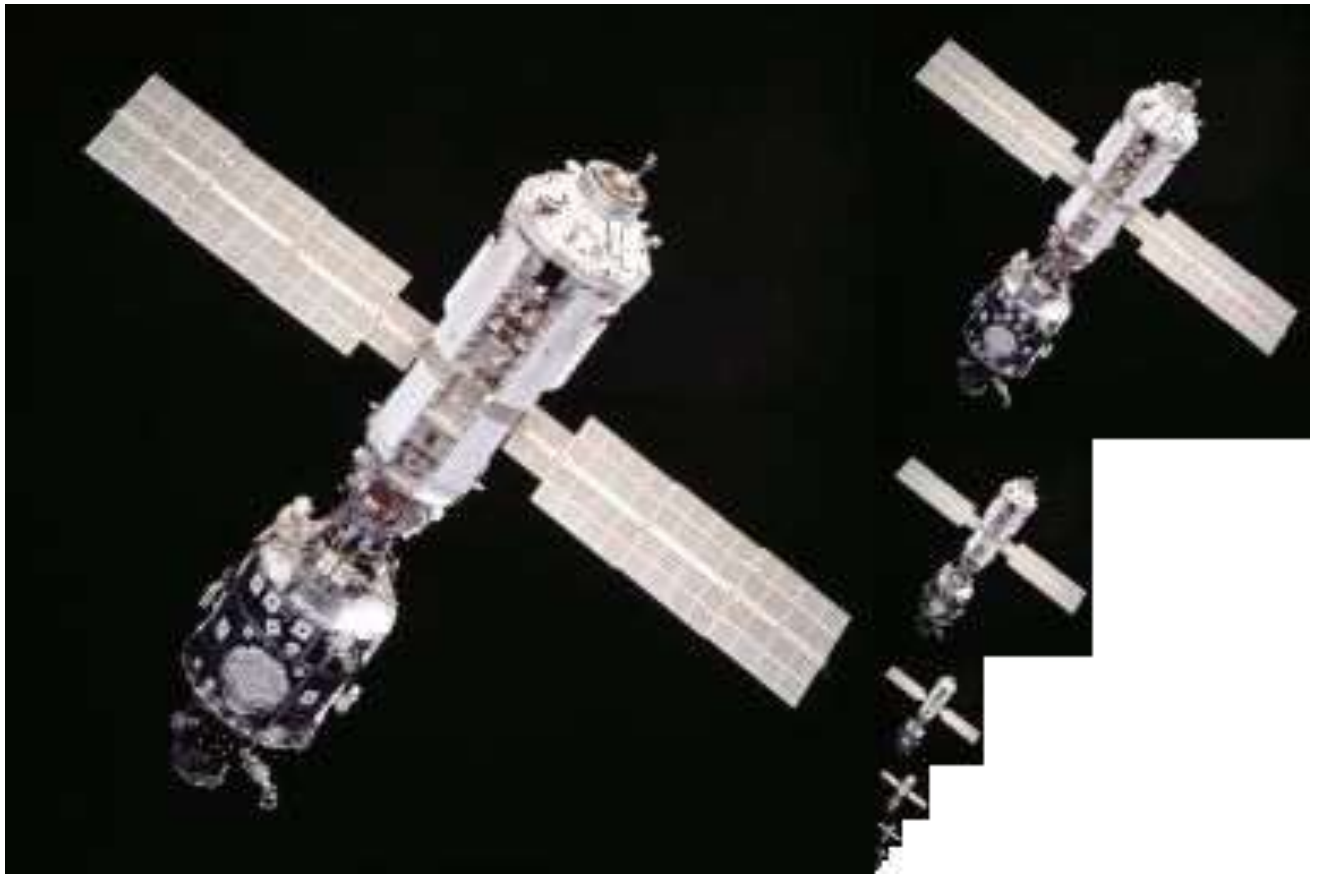
Gaussian pyramids [Burt and Adelson, 1983]



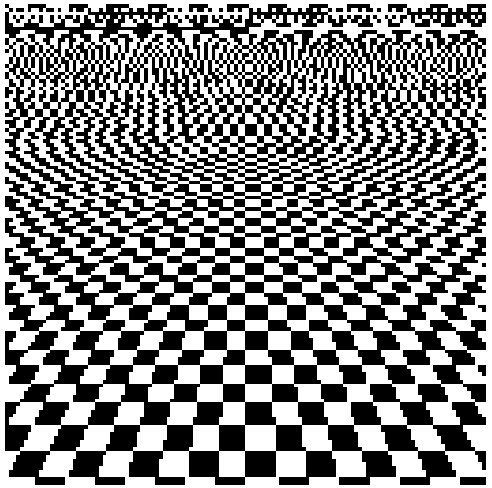
- 高斯金字塔相比原来多占用了多少空间?

Source: S. Seitz

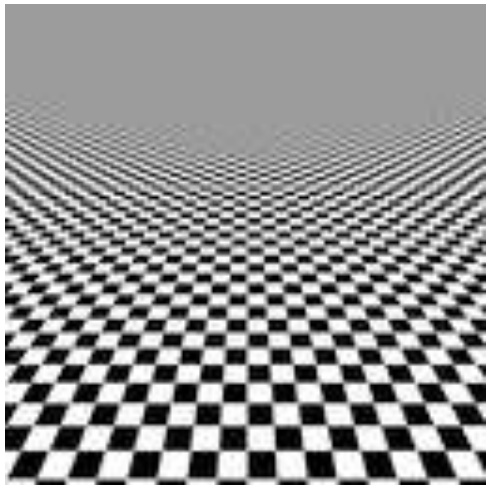
Gaussian pyramid



使用高斯金字塔的效果



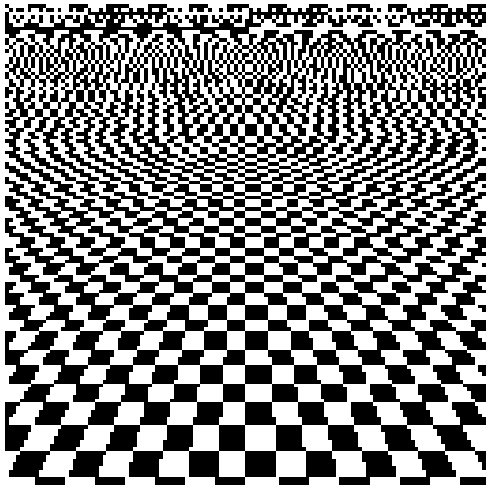
Naïve subsampling



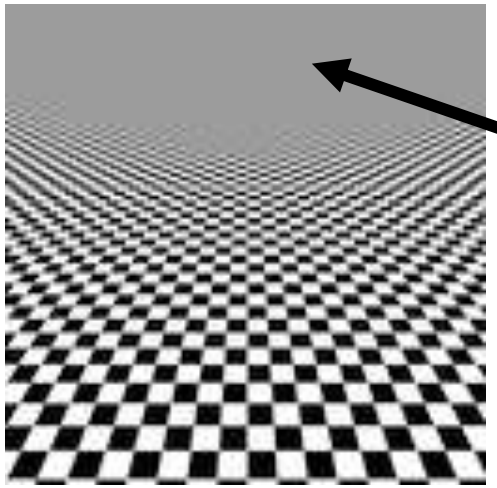
Proper prefiltering
(“antialiasing”)

使用高斯金字塔的效果

- What should happen when you make the checkerboard smaller and smaller?



Naïve subsampling



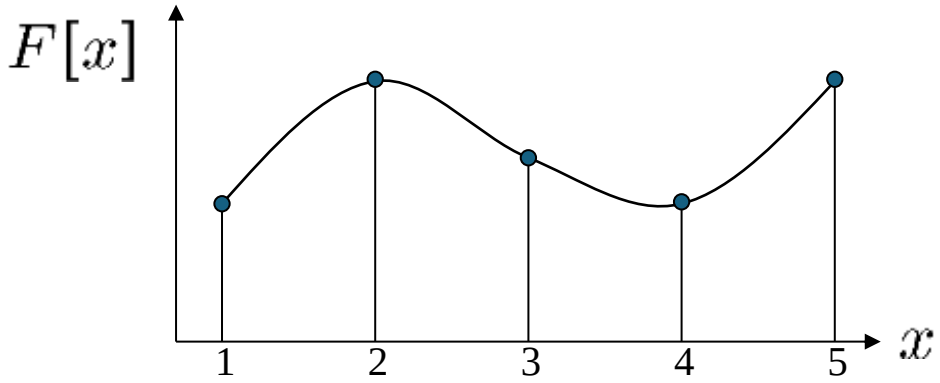
Proper prefiltering
(“antialiasing”)

Image turns grey!
(Average of black and white squares, because each pixel contains both.)

Upsampling 上采样

- 如果图像太小了:
- 怎么把它放大十倍?
- 最简单的方法:
  重复每行每列
  十次
- (“Nearest neighbor interpolation”)
  最近邻插值

图像插值



d = 1

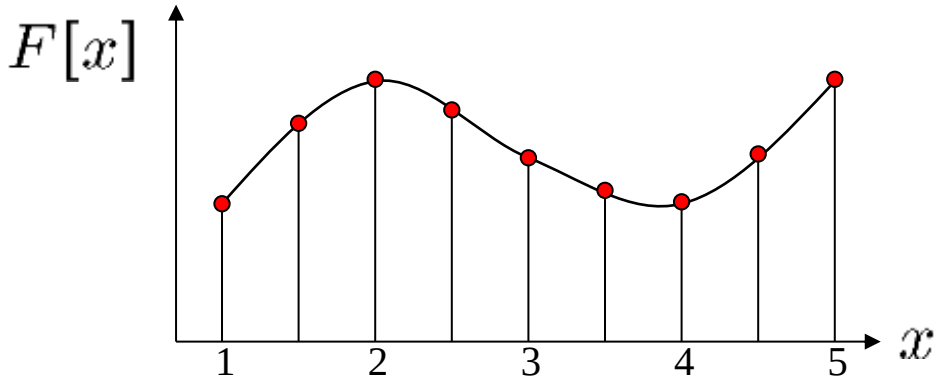
数字图像可以认为是连续信号的“量化”：

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- 每一个点都是连续信号上的离散采样
- 我们是否可以用采样的值还原整个函数？

Adapted from: S. Seitz

图像插值



d = 0.5

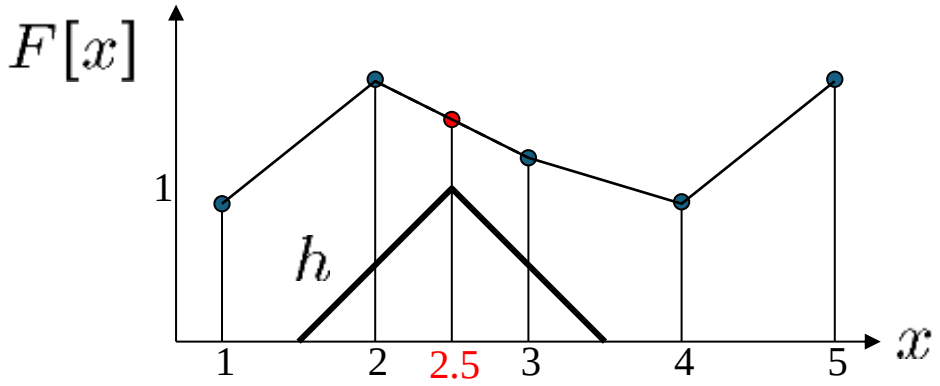
数字图像可以认为是连续信号的“量化”：

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- 每一个点都是连续信号上的离散采样
- 我们是否可以用采样的值还原整个函数？

Adapted from: S. Seitz

重构滤波器



d = 1 in this example

- 当我们不知道 f 的时候
  - 猜一个近似函数: $\tilde{f}$
  - 依然可以用滤波的形式来做
  - 把F 转换为连续函数，保留整数点的值:

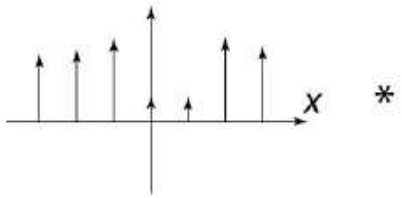
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - *用整数点估计非整数点的值，卷积h 来实现*，称之为 重建滤波

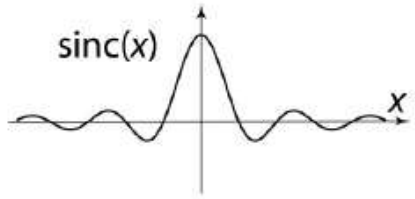
$$\tilde{f} = h * f_F$$

Adapted from: S. Seitz

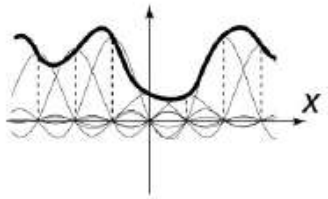
重构滤波器



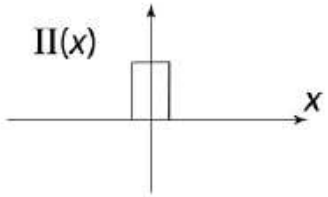
*



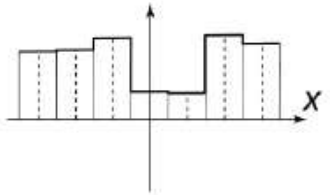
⇒



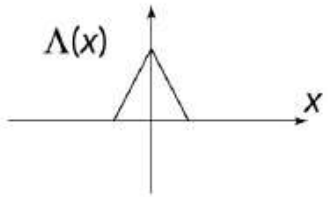
理想效果



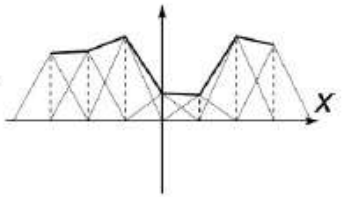
⇒



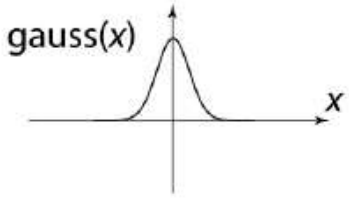
Nearest-neighbor interpolation
最近邻插值



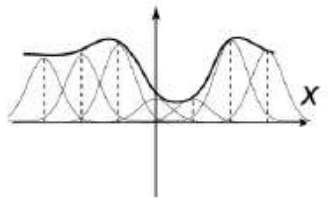
⇒



Linear interpolation
线性插值



⇒

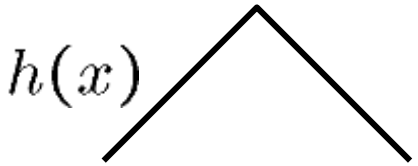


Gaussian reconstruction
高斯重建

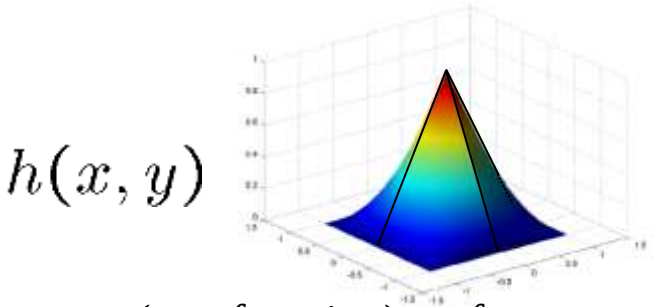
Source: B. Curless

重构滤波器

- 二维情形：



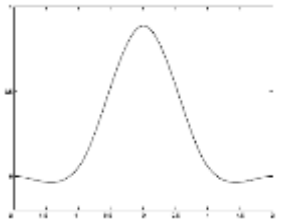
performs
linear interpolation



(tent function) performs
bilinear interpolation

更好的滤波产生更好的图像

- **Bicubic** 一般是最好的选择
- 请选择 数值分析 相关资料来学习具体内容



Cubic reconstruction filter

$$r(x)=\frac{1}{6}\begin{cases}(12-9B-6C)|x|^3+(-18+12B+6C)|x|^2+(6-2B) & |x|<1\\ ((-B-6C)|x|^3+(6B+30C)|x|^2+(-12B-48C)|x|+(8B+24C) & 1\le|x|<2\\ 0 & otherwise\end{cases}$$

图像插值

原图:  x 10

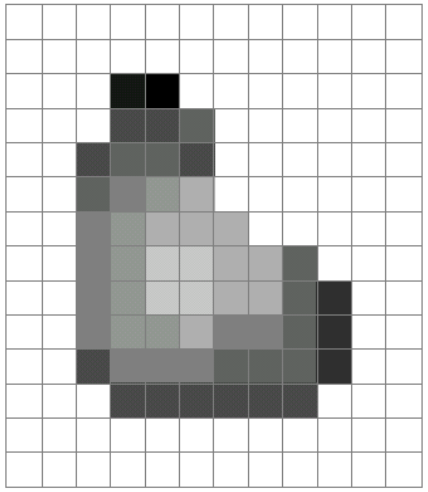
Nearest-neighbor interpolation
最近邻插值

Bilinear interpolation
双线性插值

Bicubic interpolation
双三次插值

什么是图像?

- 代表光强度的矩阵



=

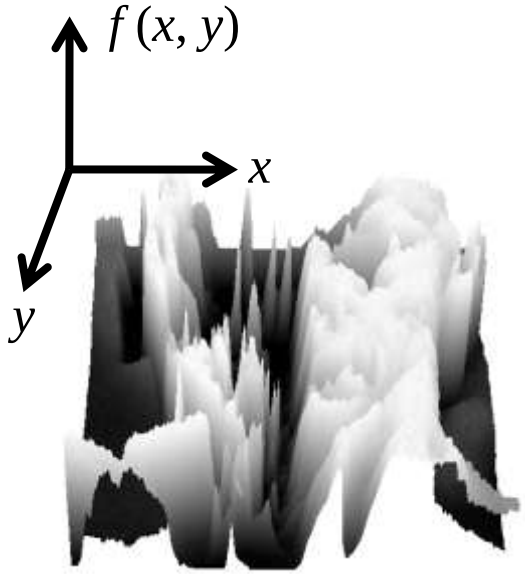
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(一般用8-bit（byte）整数表示每个值: 0 = black, 255 = white)

什么是图像?

- 可以认为是R^2 到 R的映射:
  - $f(x,y)$ 给出(x,y)的强度

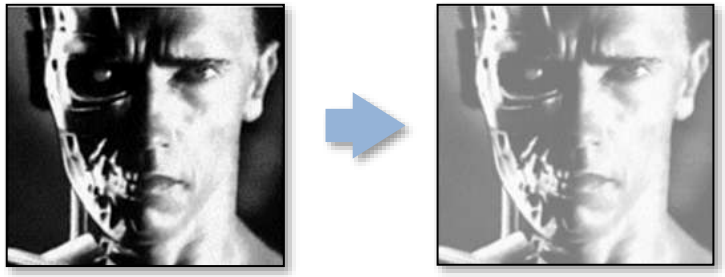
snoop



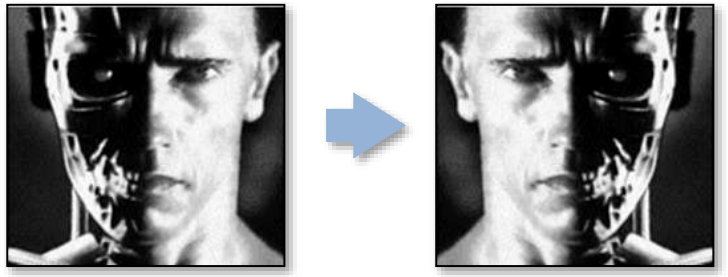
3D view

图像变换

- 可以应用到图像的变换函数



$g\,(x,y) = f\,(x,y) + 20$



$g\,(x,y) = f\,(-x,y)$

- 接下来我们会具体介绍一个十分常用的变换 卷积 *convolution* (线性滤波)
  - 是图像处理的基础操作
  - 也是后续处理实时性和通用性的基础单元

图像滤波

- 基于局部图像和某个函数修改像素的值

10	5	3
4	5	1
1	1	7

局部图像

某个函数

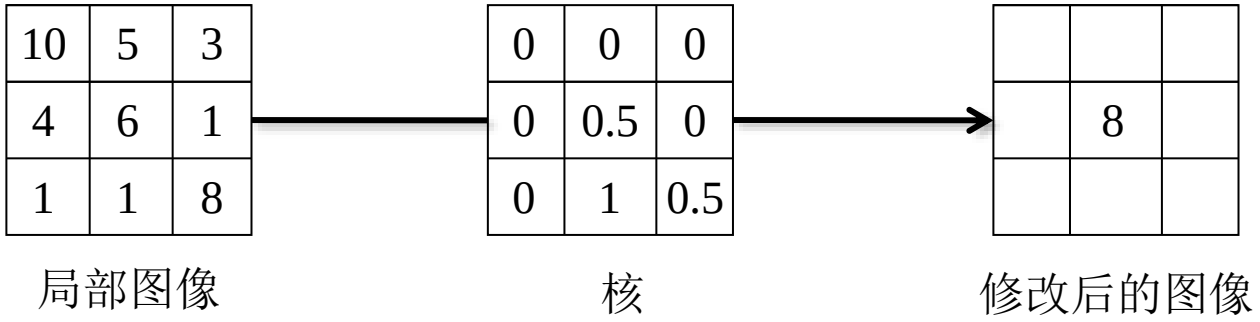
	7	

修改后的图像

Source: L. Zhang

线性图像滤波

- 一种简单的图像滤波：线性滤波(cross-correlation, convolution)
  - 用像素点周围的值的线性组合来替代自身的像素值
- 函数可以被 核“kernel” (or “mask”, “filter”)表示



Source: L. Zhang

Cross-correlation 互相关

F ： 图像, H ： 核 (大小 $2k+1$ x $2k+1$), G ： 输出图像

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

这就是 **cross-correlation 互相关** :

$$G = H \otimes F$$

- 可以认为是图像局部与核的相关性（内积）

Convolution 卷积

- 与互相关类似，不过水平垂直反转

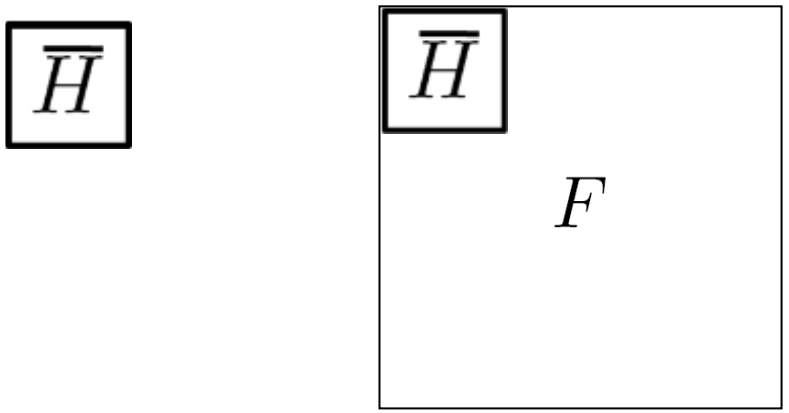
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

这就是 **convolution** 卷积:

$$G = H * F$$

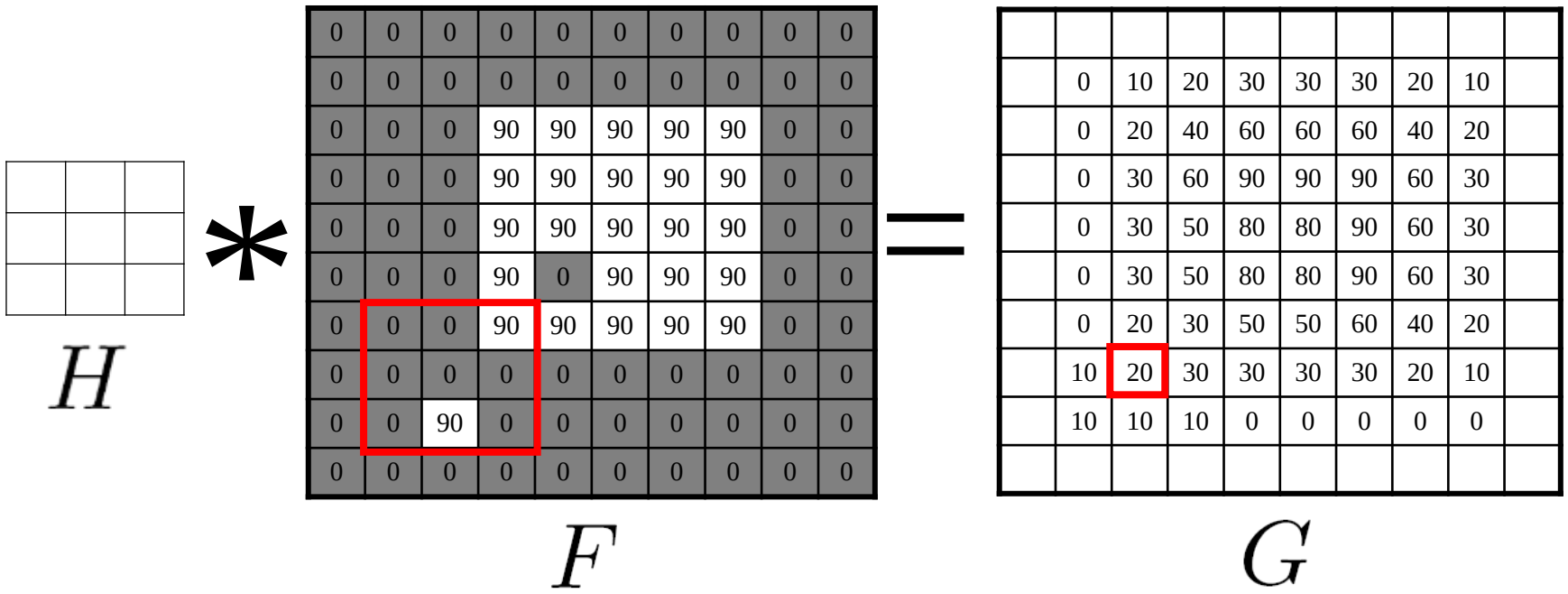
- 卷积具有 **交换律** 和 **结合律**

卷积

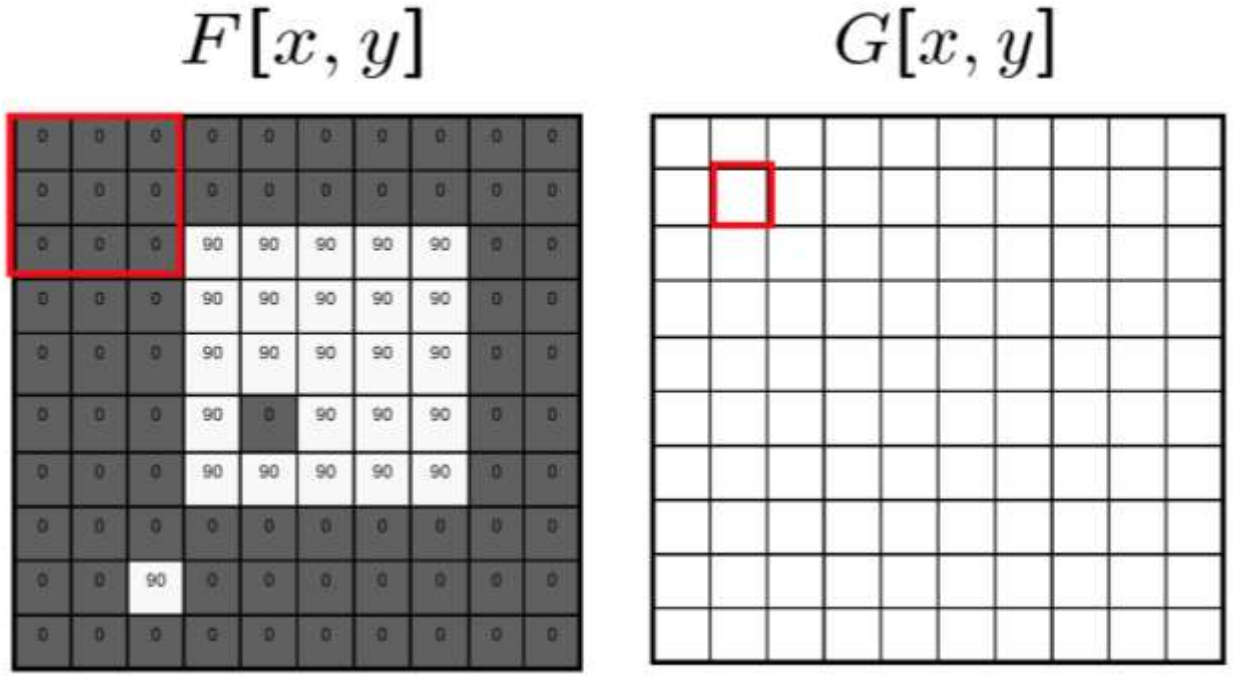


Adapted from F. Durand

Mean filtering 平均滤波

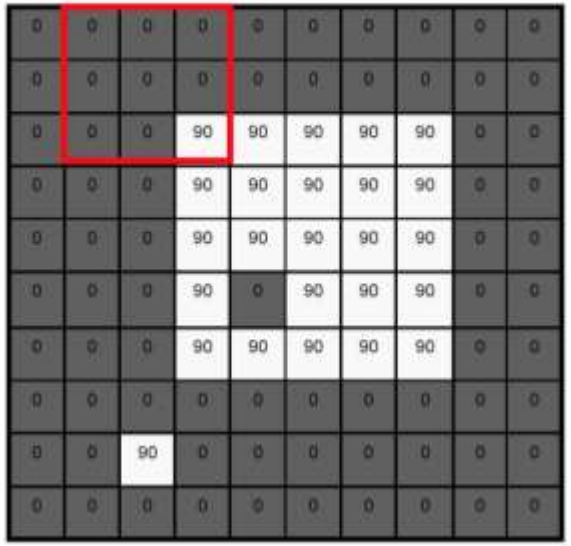


平均滤波

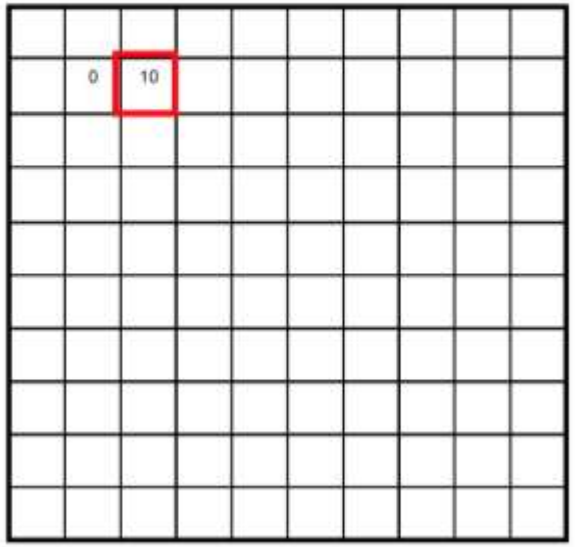


平均滤波

$F[x, y]$

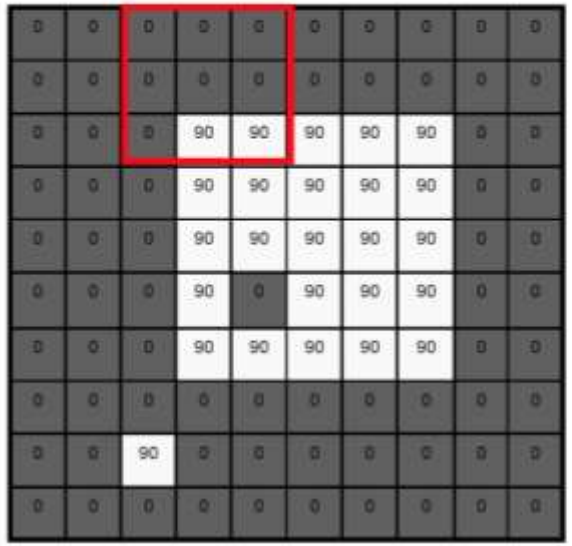

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

$G[x, y]$

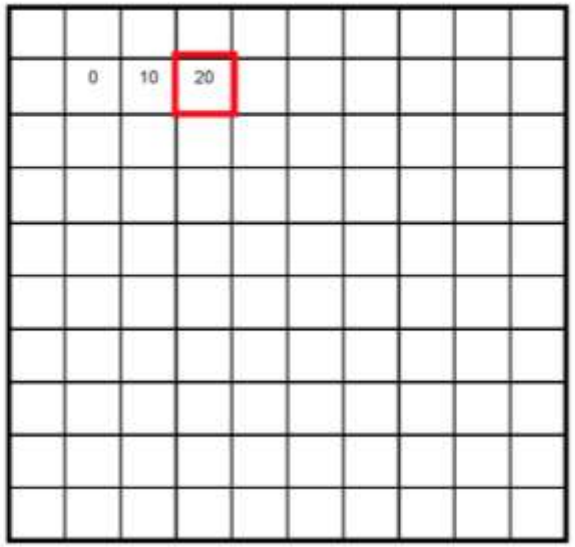

0
10

平均滤波

$F[x, y]$

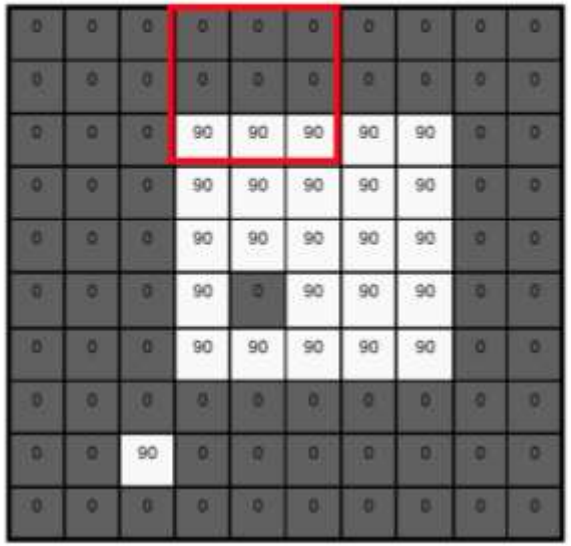

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

$G[x, y]$

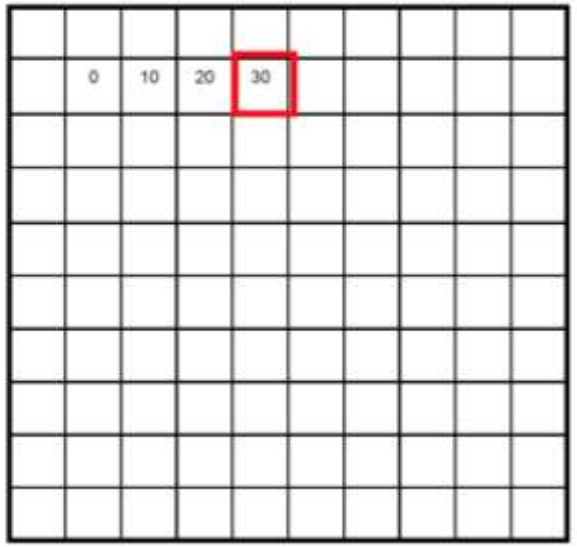

0 10 20

平均滤波

$F[x, y]$

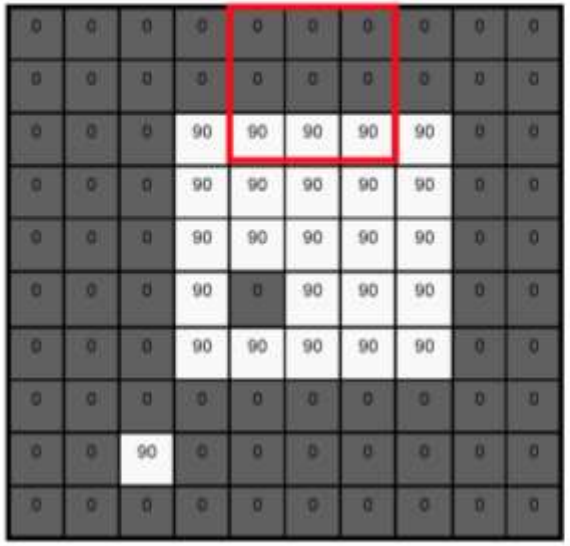

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

$G[x, y]$

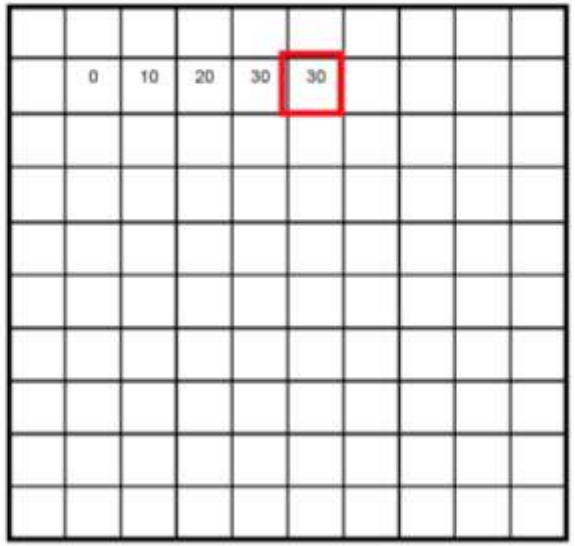

0 10 20 30

平均滤波

$F[x, y]$


0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 90 0 90 90 90 0 0
0 0 0 90 90 90 90 90 0 0
0 0 0 0 0 0 0 0 0 0
0 0 90 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

$G[x, y]$


0 10 20 30 30

平均滤波

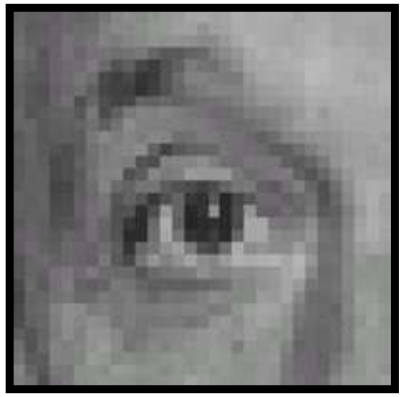
$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

线性滤波

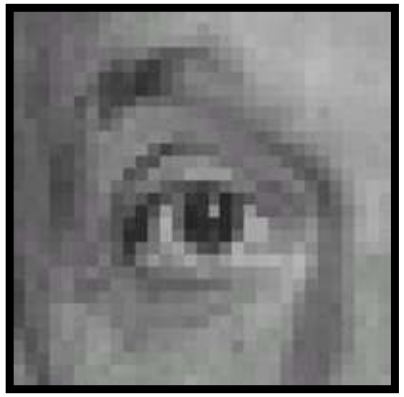


原图

$$*\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

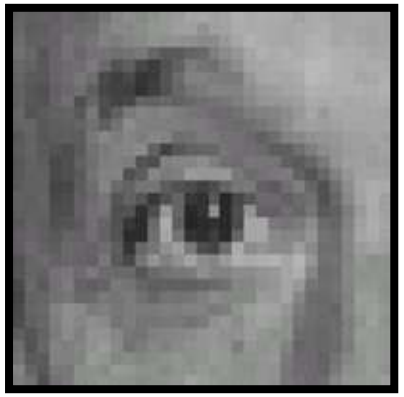
Source: D. Lowe

线性滤波



原图

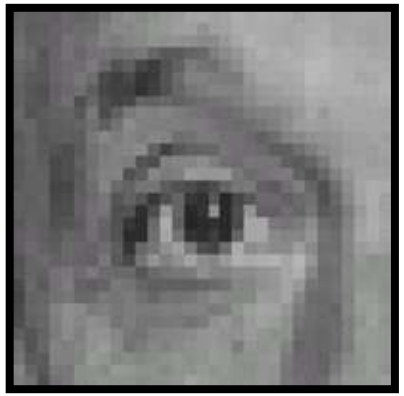
$$*\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}=$$



同一张图

Source: D. Lowe

线性滤波

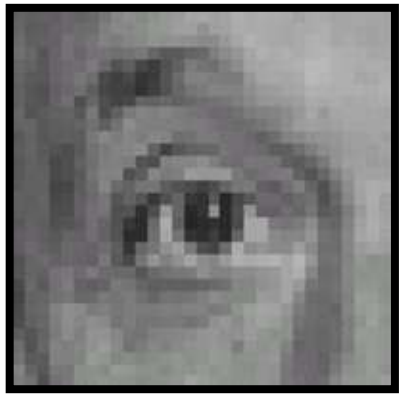


原图

$$*\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

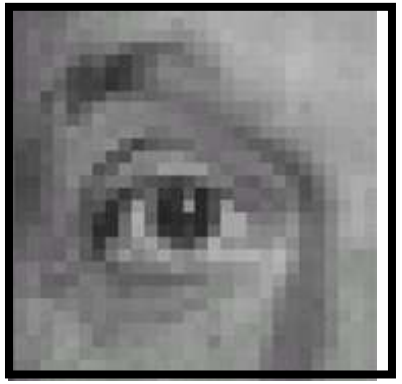
Source: D. Lowe

线性滤波



原图

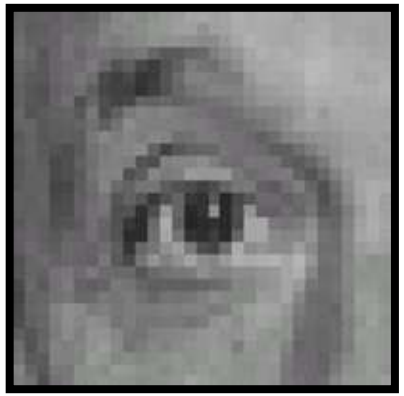
$$*\begin{bmatrix}0 & 0 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}=$$



向左移动一个像素

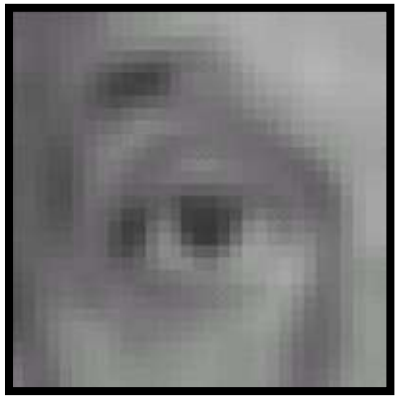
Source: D. Lowe

线性滤波



原图

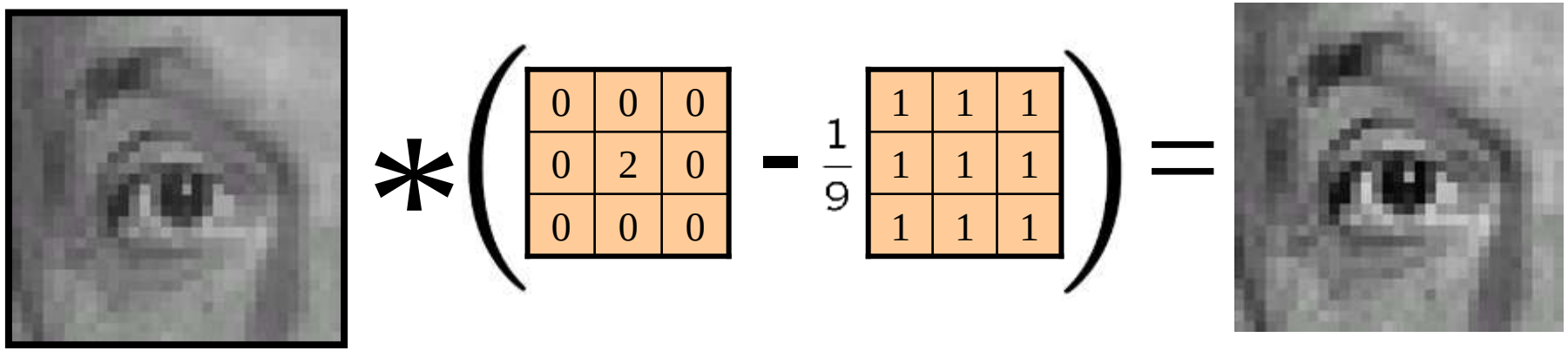
$$*\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} =$$



模糊（平均滤波）

Source: D. Lowe

线性滤波

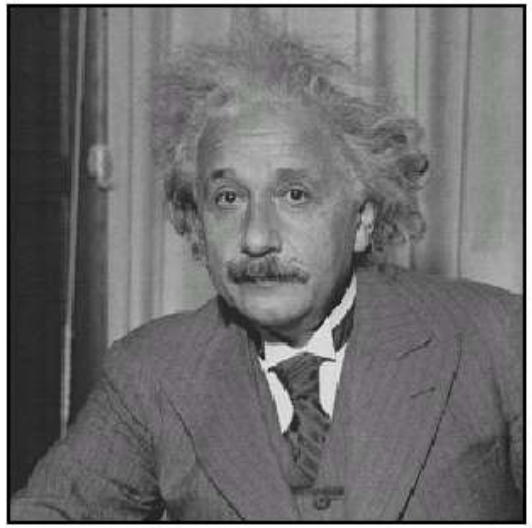


原图

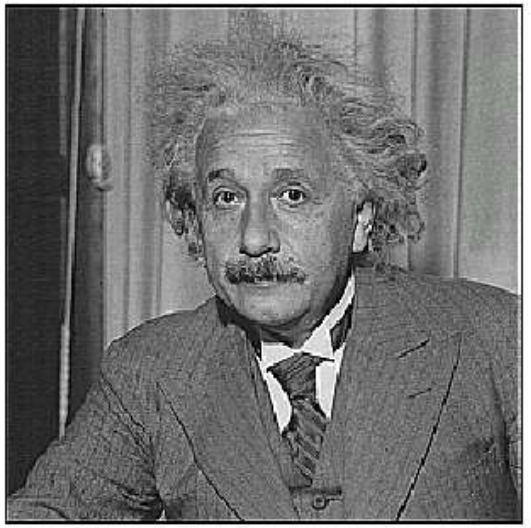
锐化后
(突出边缘)

Source: D. Lowe

图像锐化



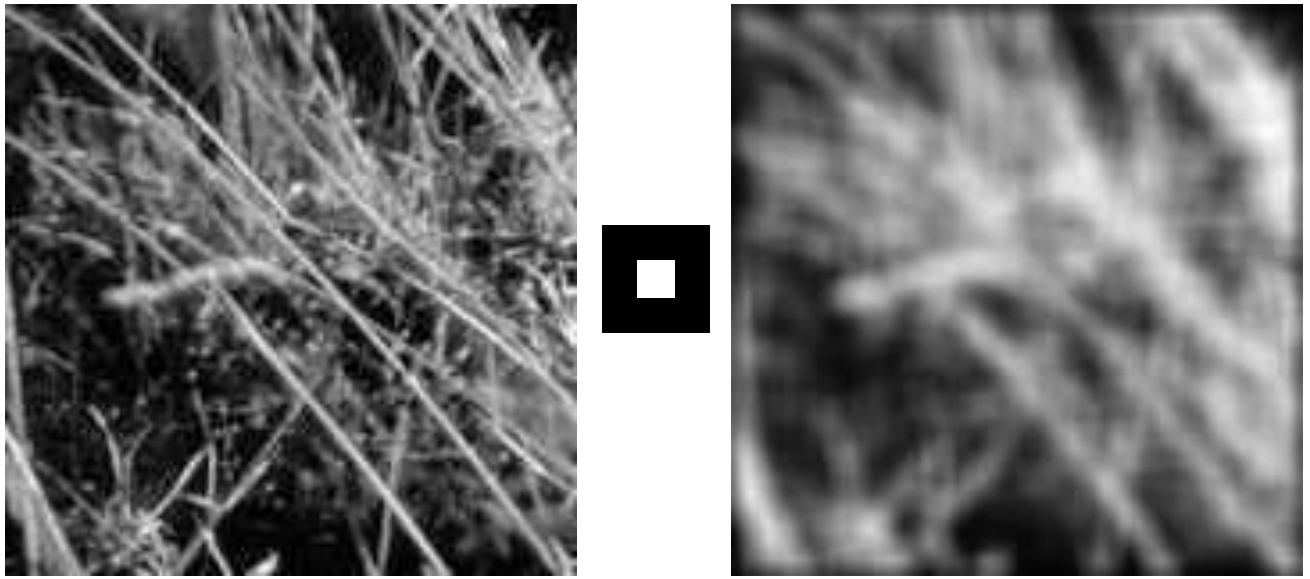
before



after

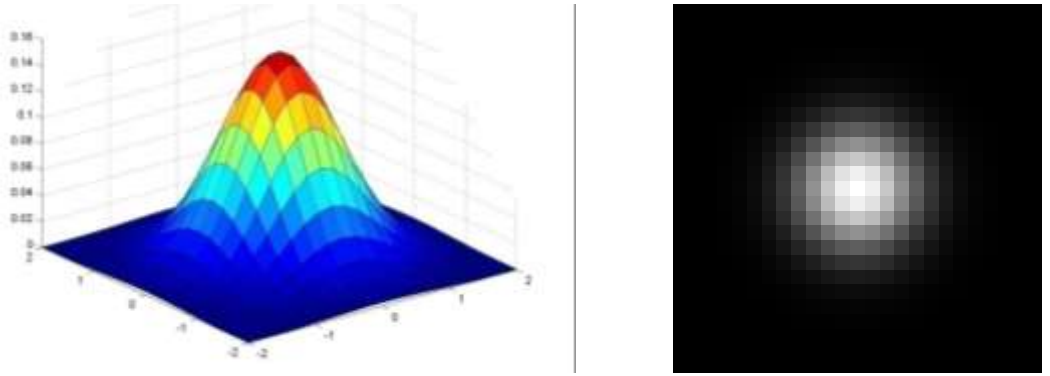
Source: D. Lowe

图像模糊



Source: D. Forsyth

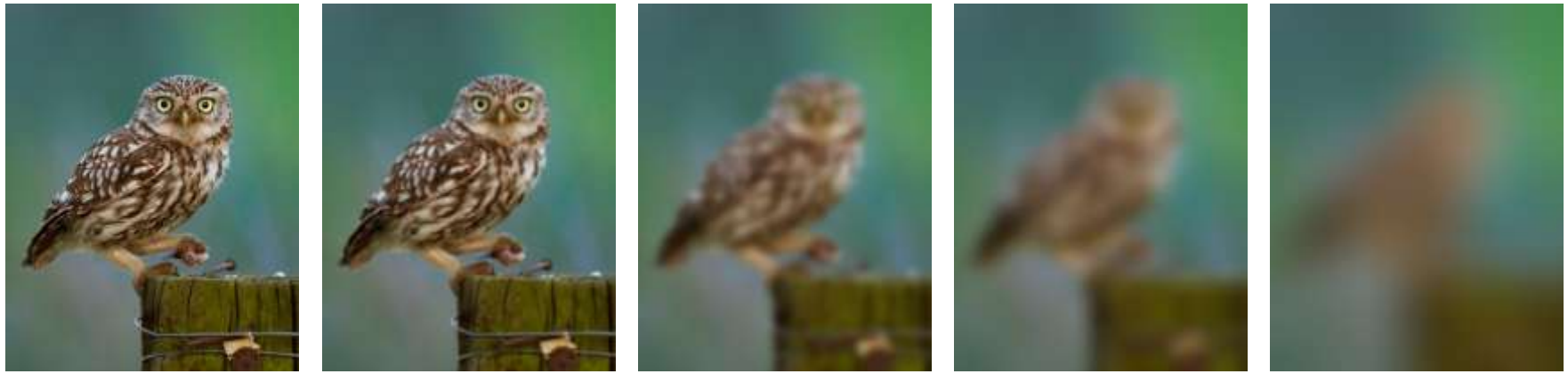
高斯核



$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

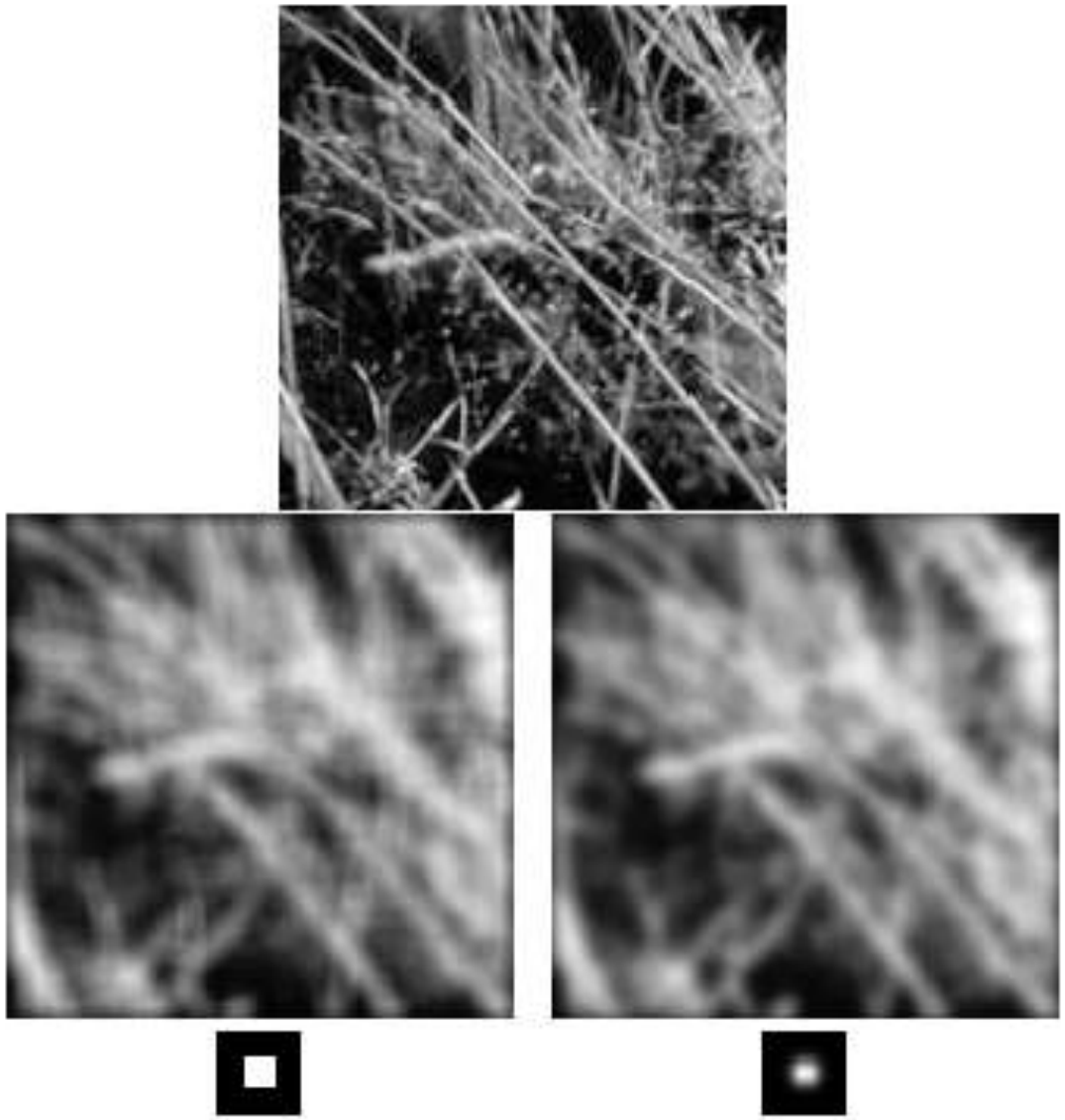
Source: C. Rasmussen

高斯滤波



σ = 1 pixel　　σ = 5 pixels　　σ = 10 pixels　　σ = 30 pixels

平均滤波 vs 高斯滤波



高斯滤波

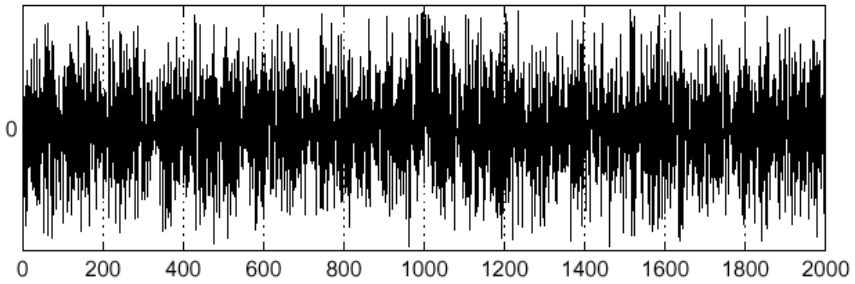
- 去掉 “high-frequency 高频” 图像 (low-pass filter 低通滤波)
- 如果高斯滤波自我叠加

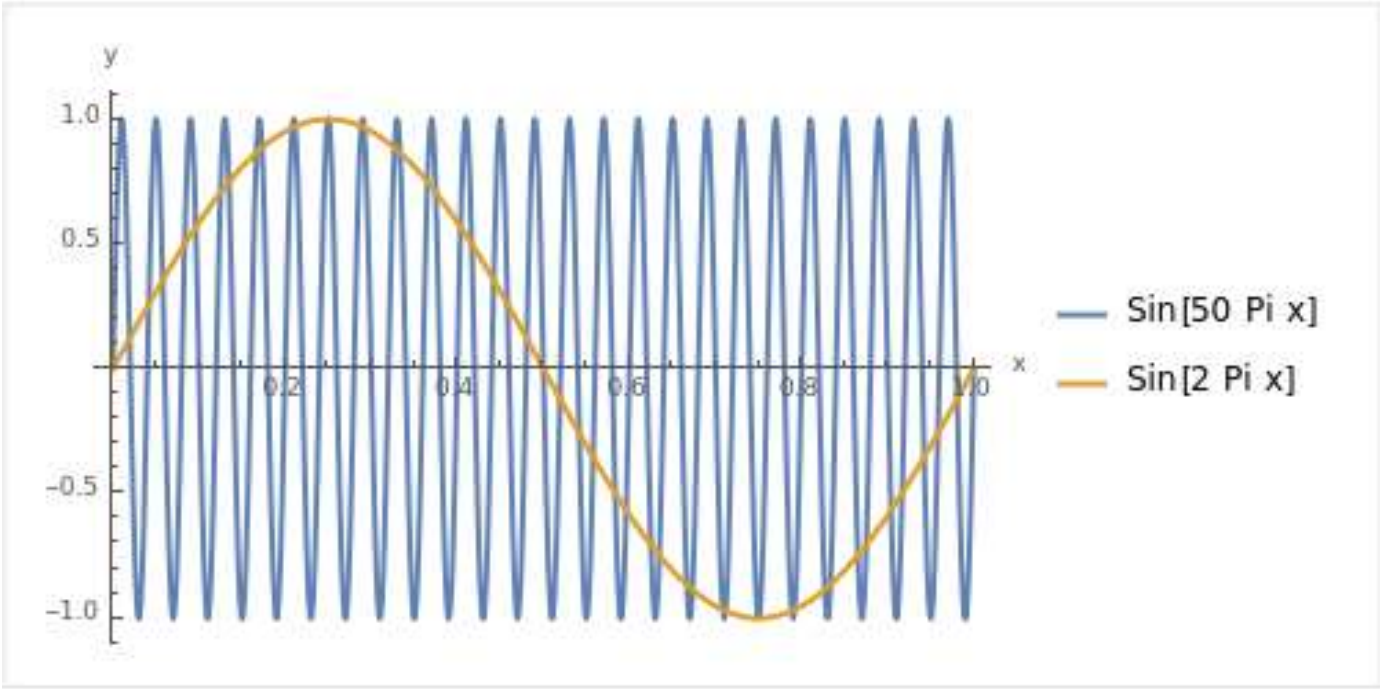
- 核宽为σ　卷积后核宽为$\sigma\sqrt{2}$

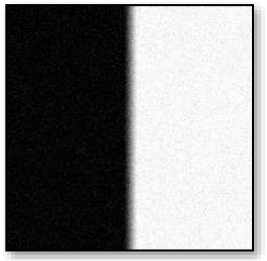
Source: K. Grauman

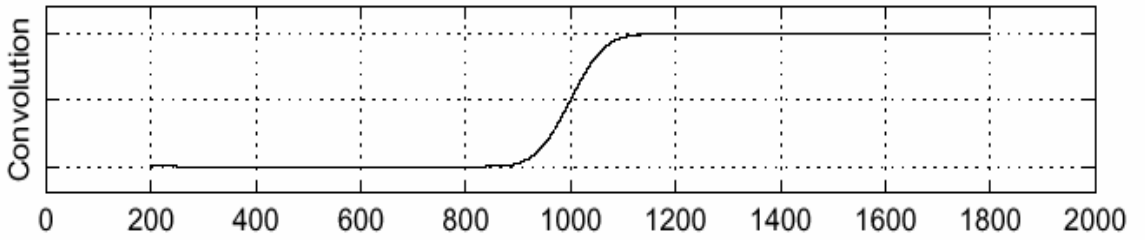
低频信号 vs 高频信号

在图像处理中，高频信号通常代表图像中的细节和边缘，而低频信号则代表图像的基础结构和平滑区域。

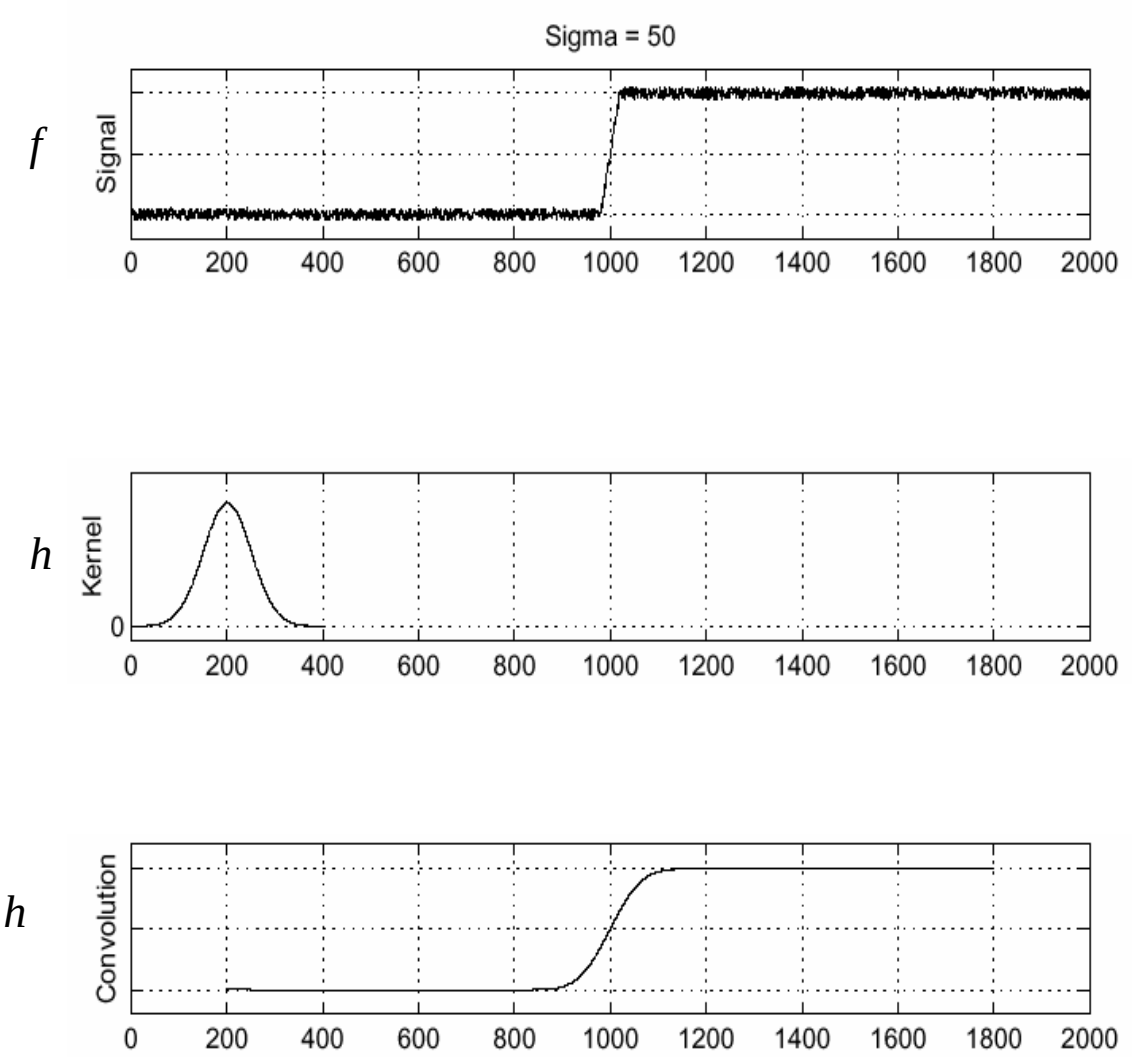








高斯滤波的平滑效果



Source: S. Seitz

锐化

- 模糊带走了什么?

(“细节提取”：高通滤波 ***high-pass filter***)

把细节加回来:

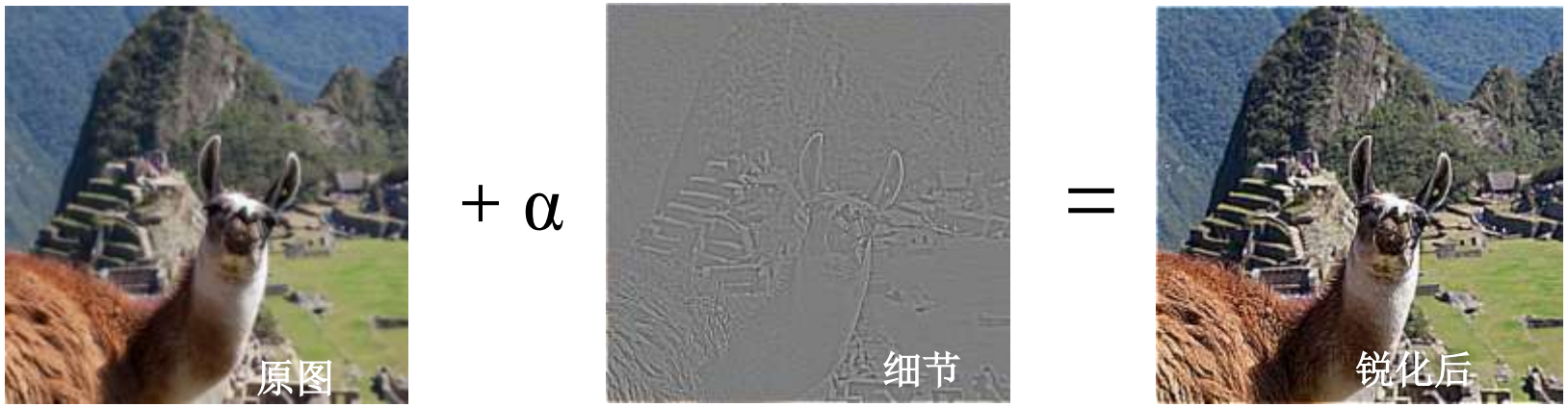


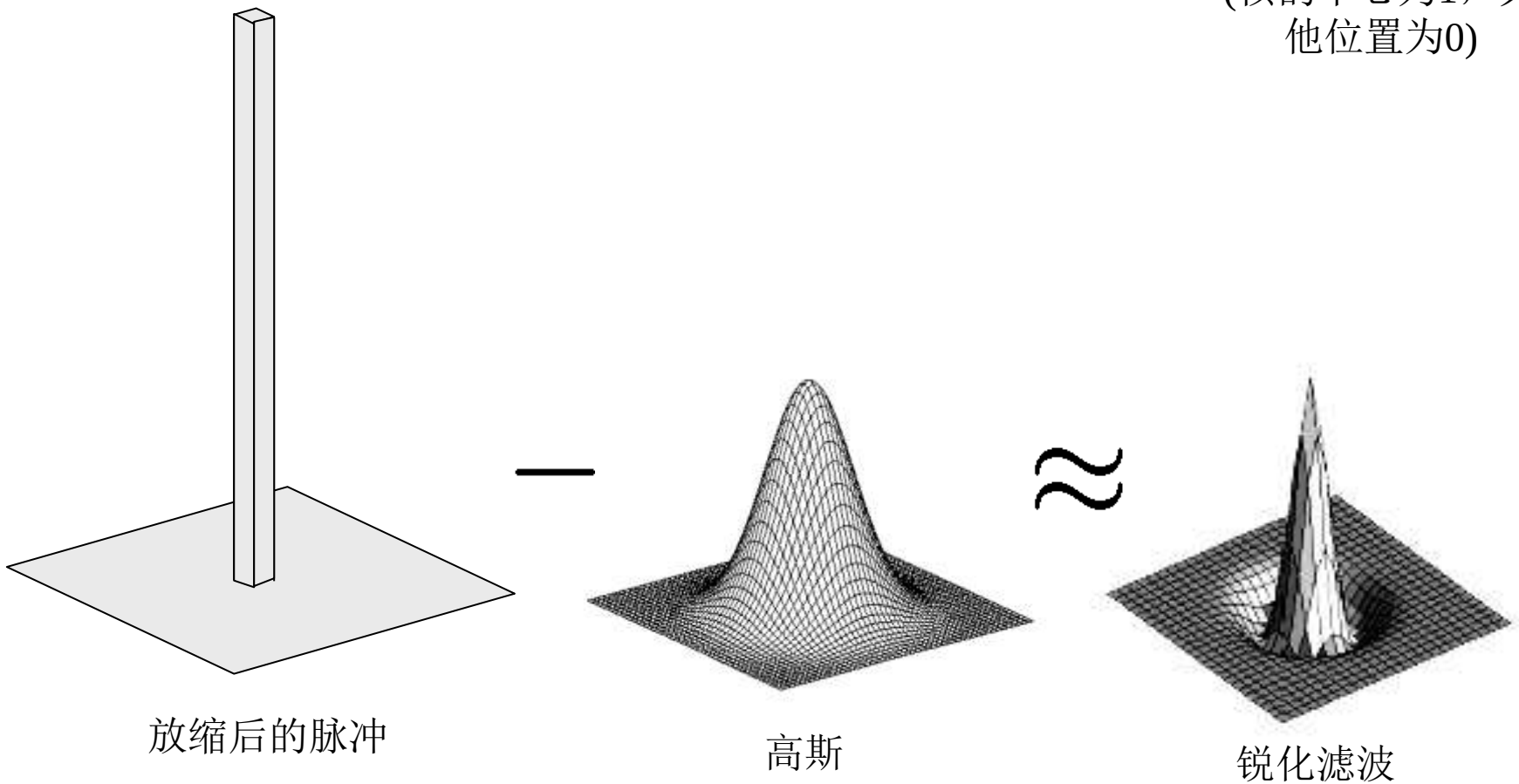
Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

锐化滤波

$$F + \alpha\,(F - \underbrace{F * H}_{\text{模糊后的图像}}) =$$

图像

单位脉冲
(核的中心为1，其他位置为0)



放缩后的脉冲

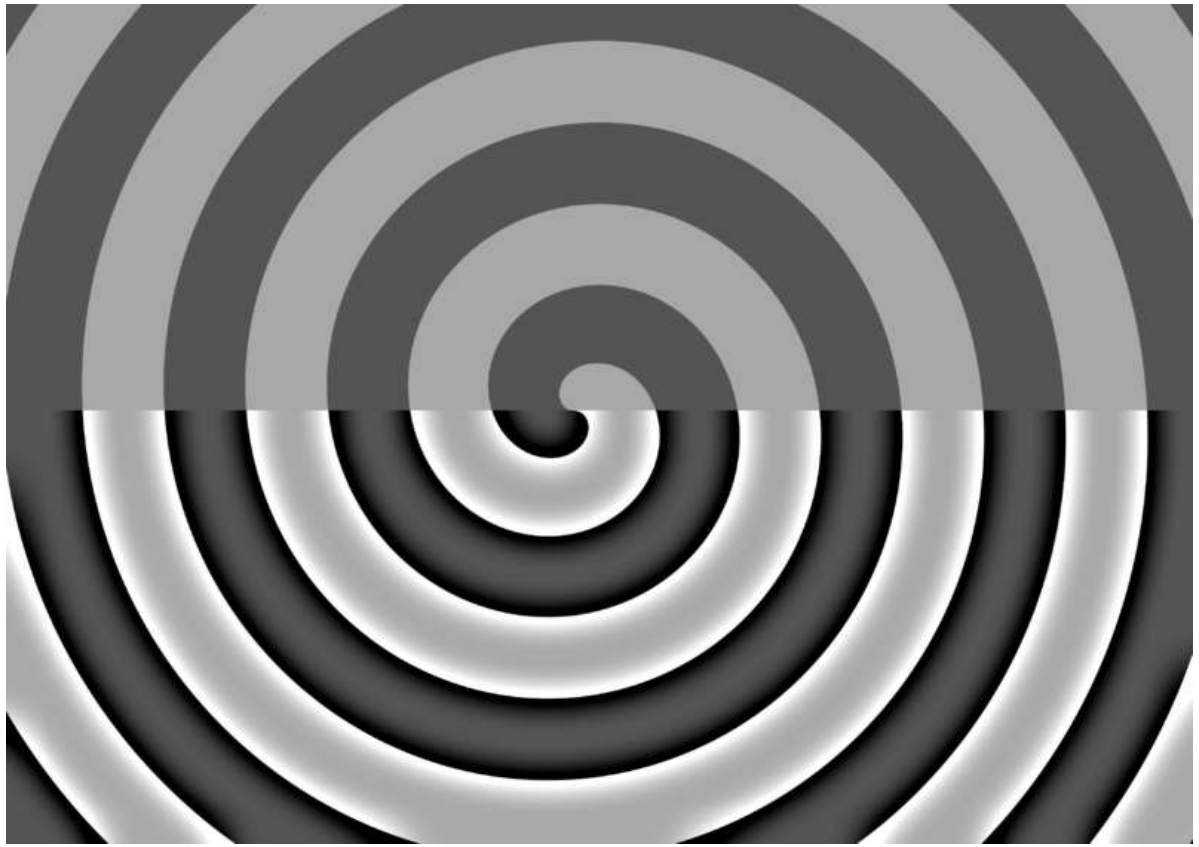
高斯

锐化滤波

锐化滤波

锐化前

锐化后



“光学”卷积

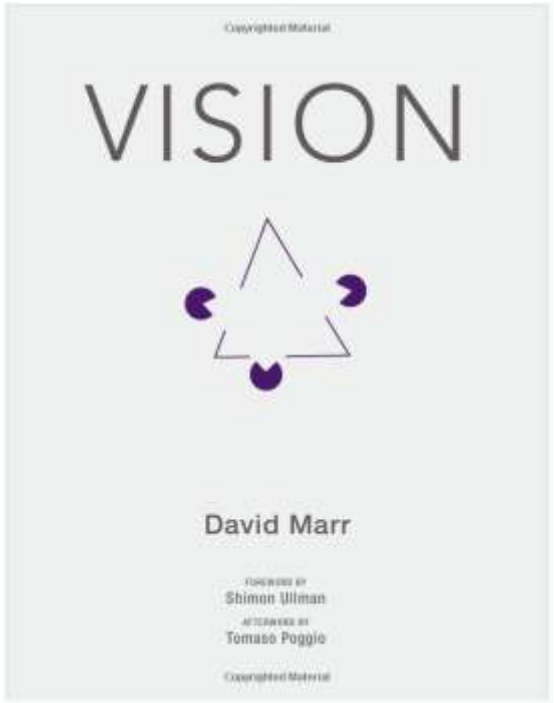
镜头晃动

Source: Fergus, *et al.* “Removing Camera Shake from a Single Photograph”, SIGGRAPH 2006

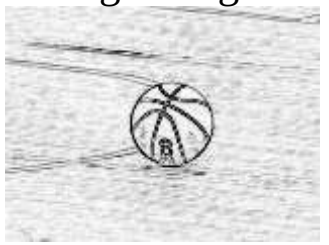
虚化模板

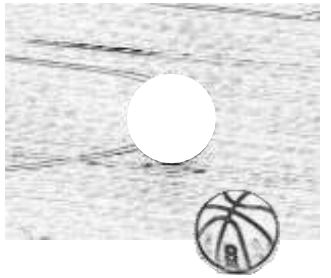
Source: https://www.diyphotography.net/diy_create_your_own_bokeh/

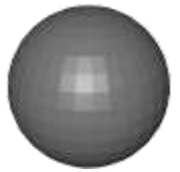
角点检测

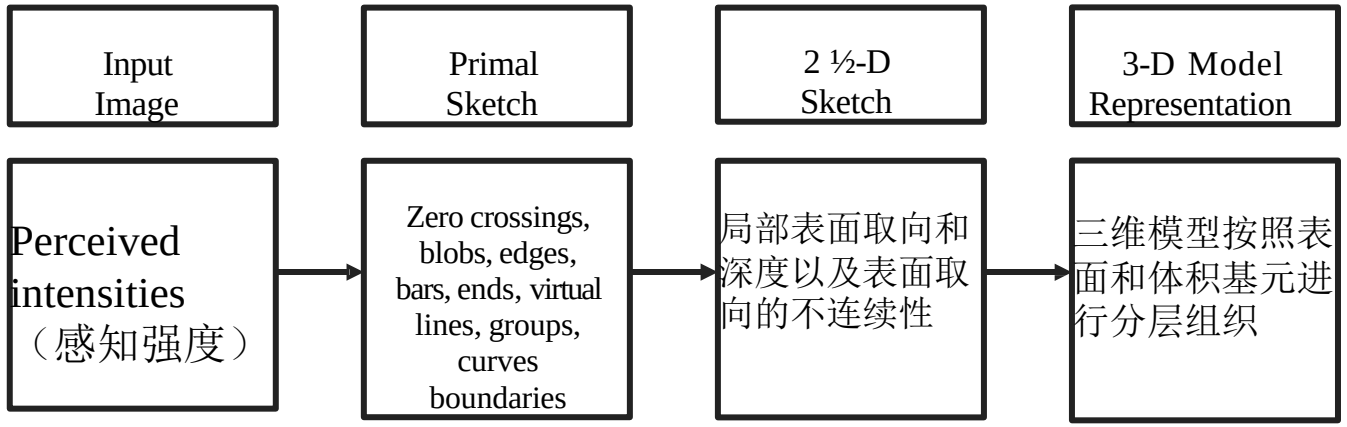


This image is CC0 1.0 public domain





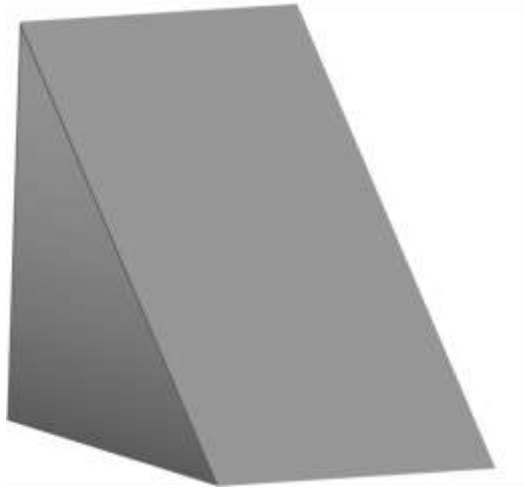




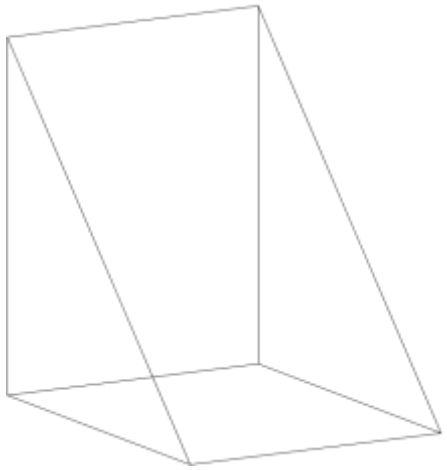
“二维半素描”：通过从不同视角获取的二维图像推断出的一些关于物体表面形状和结构的简化信息

Stages of Visual Representation, David Marr, 1970s

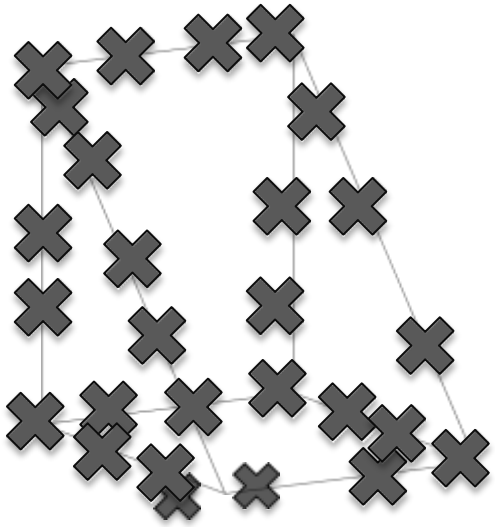
特征点



(a) Original picture



(b) Differentiated picture



(c) Feature points selected

从二维图像提取形状和三维信息

Lawrence Gilman Roberts, “Machine Perception of Three-Dimensional Solids”, 1963