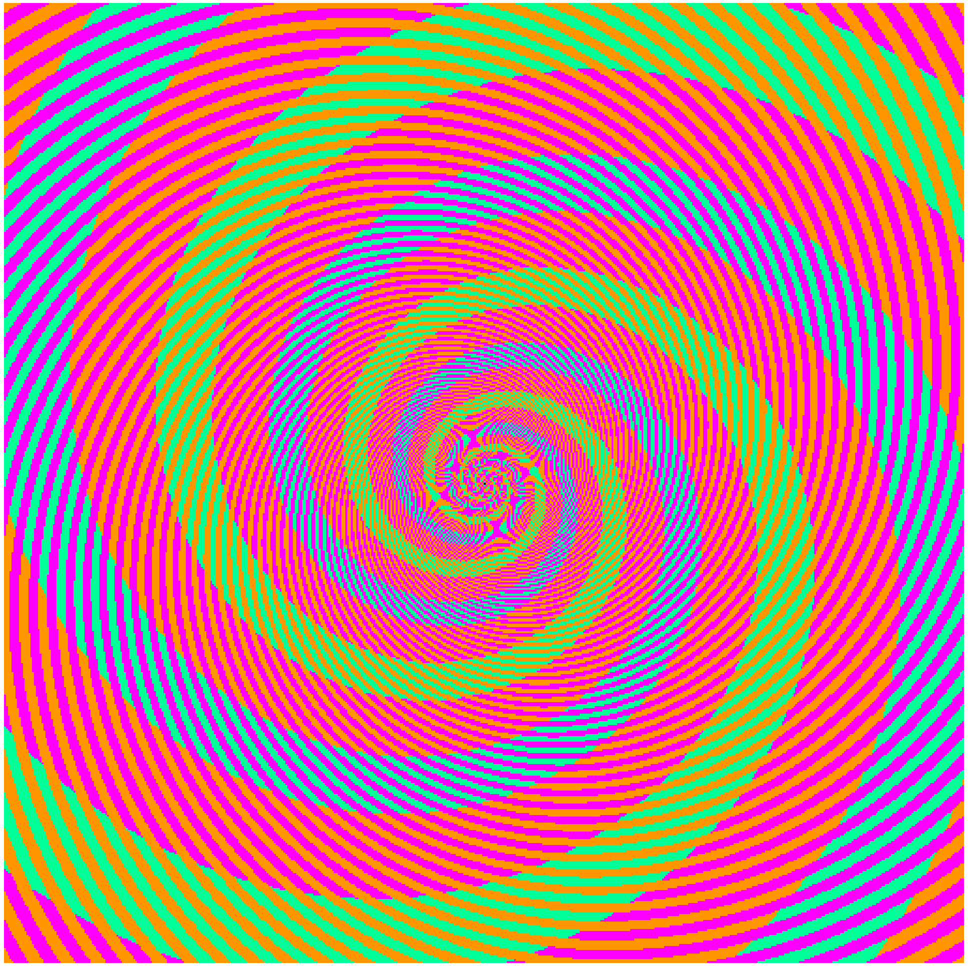
Edge Detection

边缘检测

潘 纲
浙江大学计算机学院
gpan@zju.edu.cn



http://blogs.discovermagazine.com/badastronomy/2009/06/24/the-blue-and-the-green/

Outline

- 边缘及边缘检测概念
- 用模板实现卷积
- 基于一阶导数的边缘检测
- 基于二阶导数的边缘检测
  – Laplacian算子
  – LoG算子 （Marr&Hildreth算子）
- **Canny边缘检测**

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

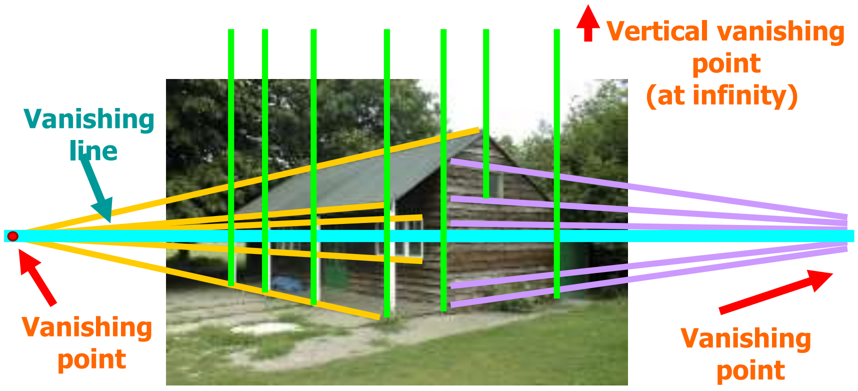
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

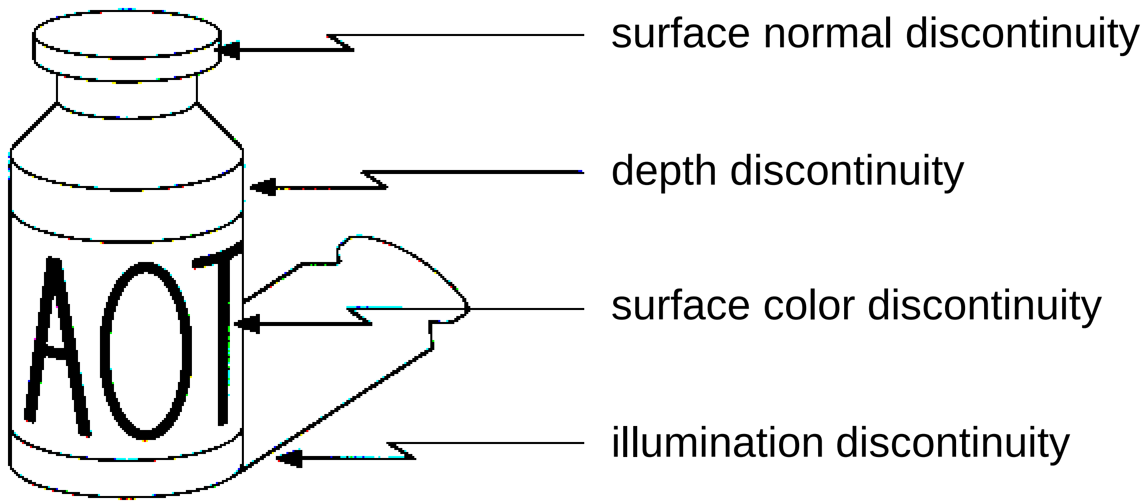
Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



Origin of Edges



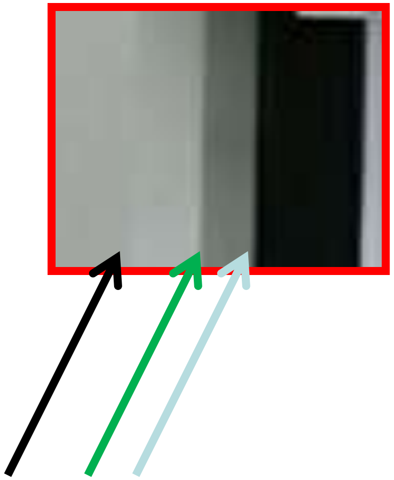
- Edges are caused by a variety of factors

Source: Steve Seitz

Closeup of edges

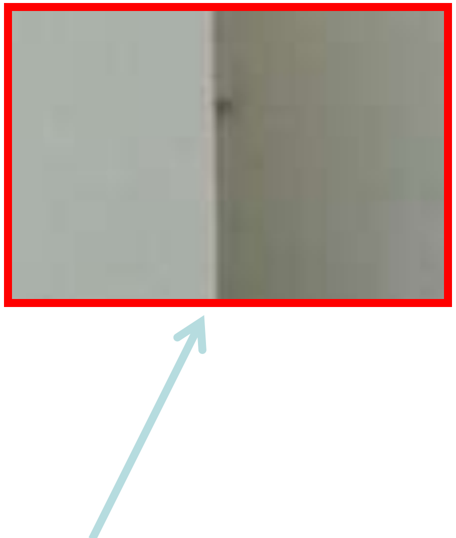
Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges



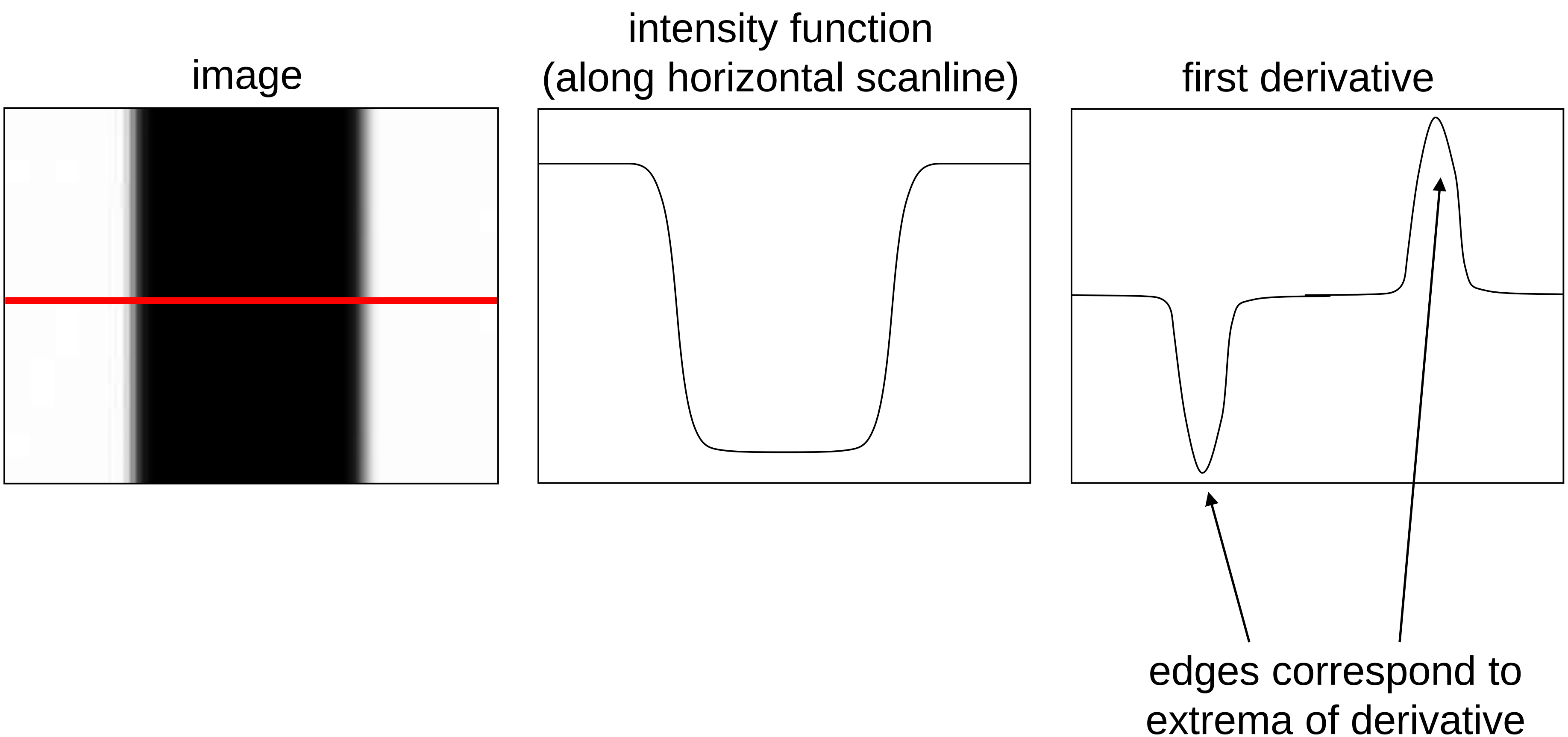
Source: D. Hoiem

Closeup of edges

Source: D. Hoiem

Characterizing edges

- **An edge is a place of rapid change in the image intensity function**



边缘检测

- **Goal:**

  Identify sudden changes (discontinuities) in an image

- **Solution**

  Zero-crossing of second-order derivative

  Local maxima of first-order derivative

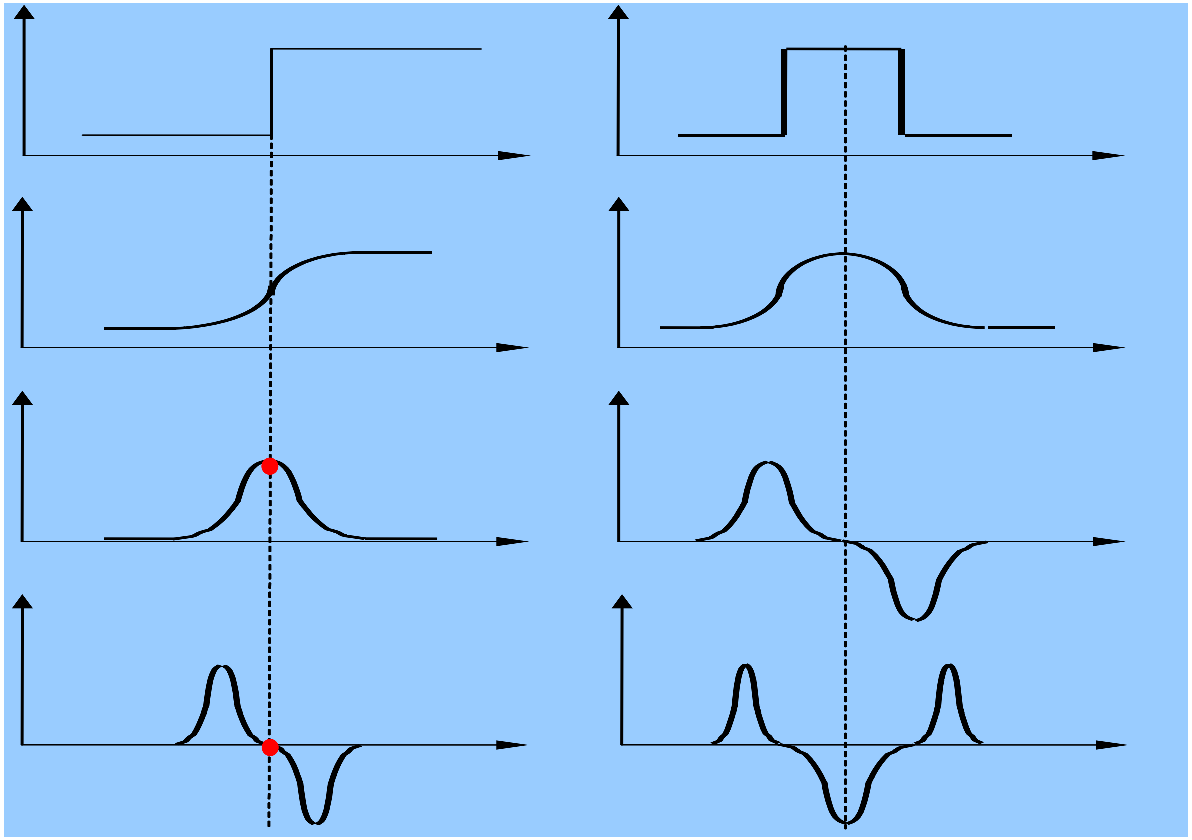
示意图

理论曲线

实际曲线

一阶导数

二阶导数



(a)阶跃函数　　(b)线条函数

对噪音的敏感性比较

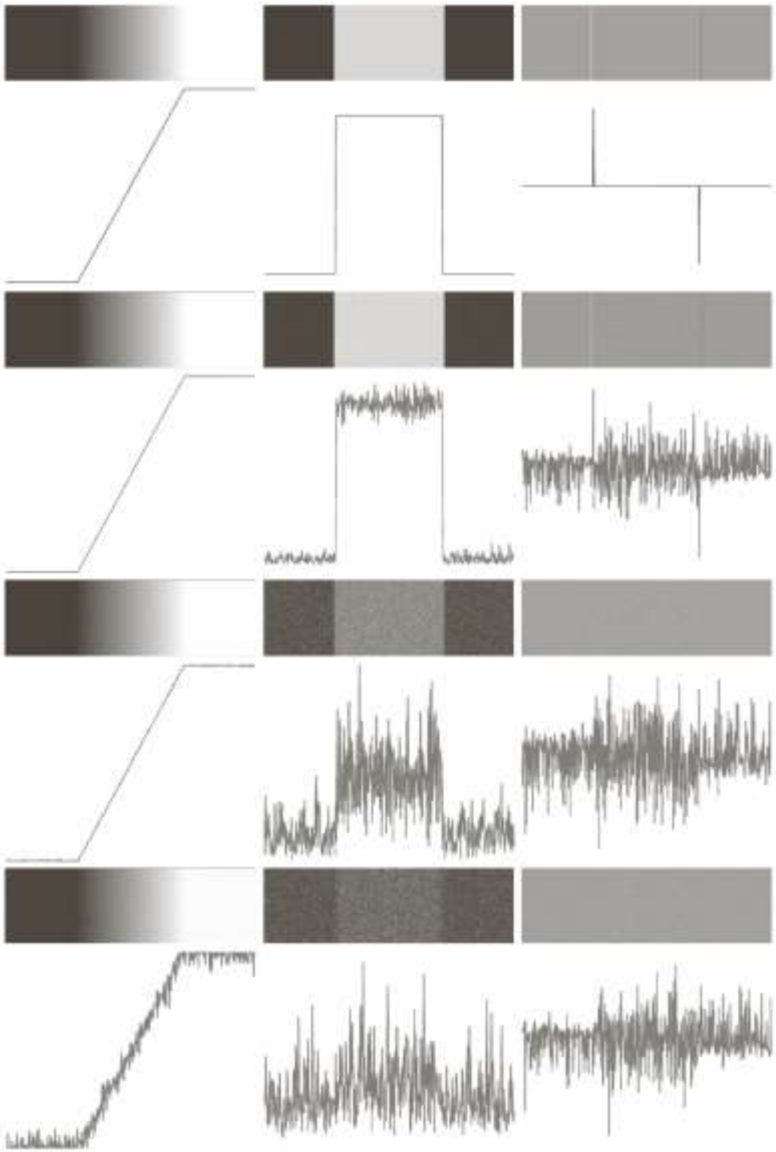
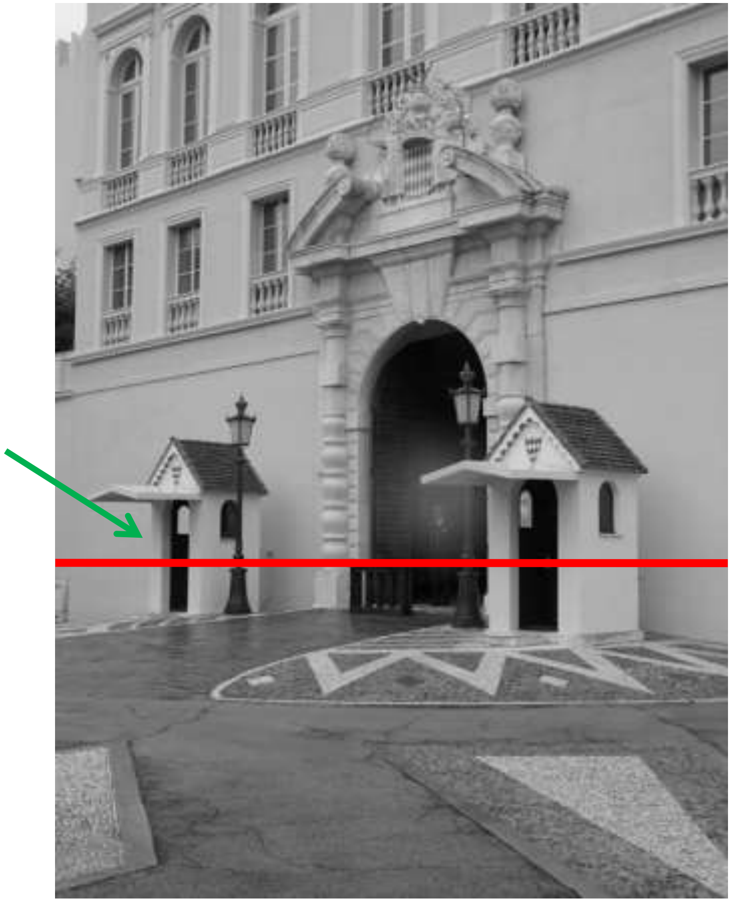
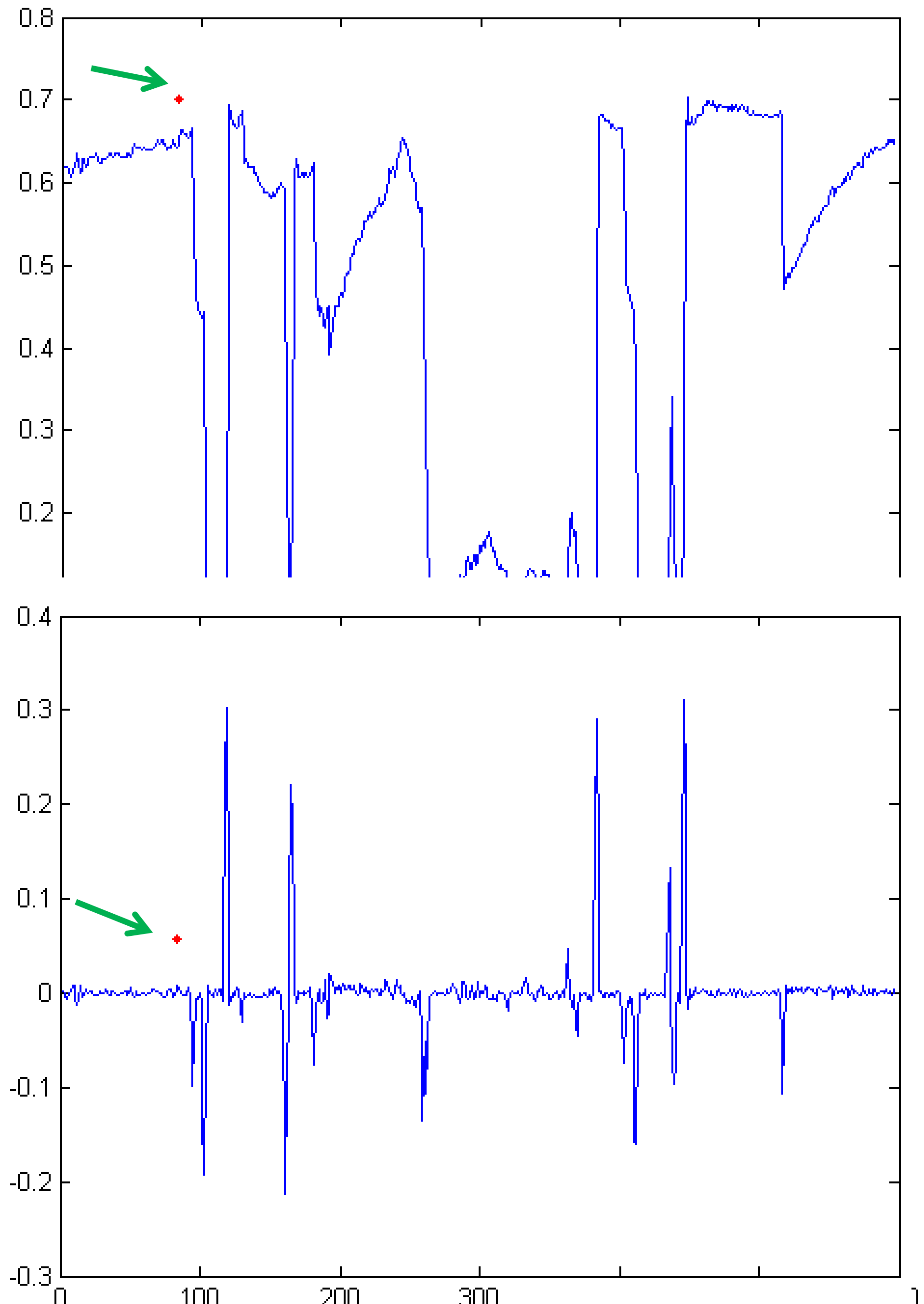


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

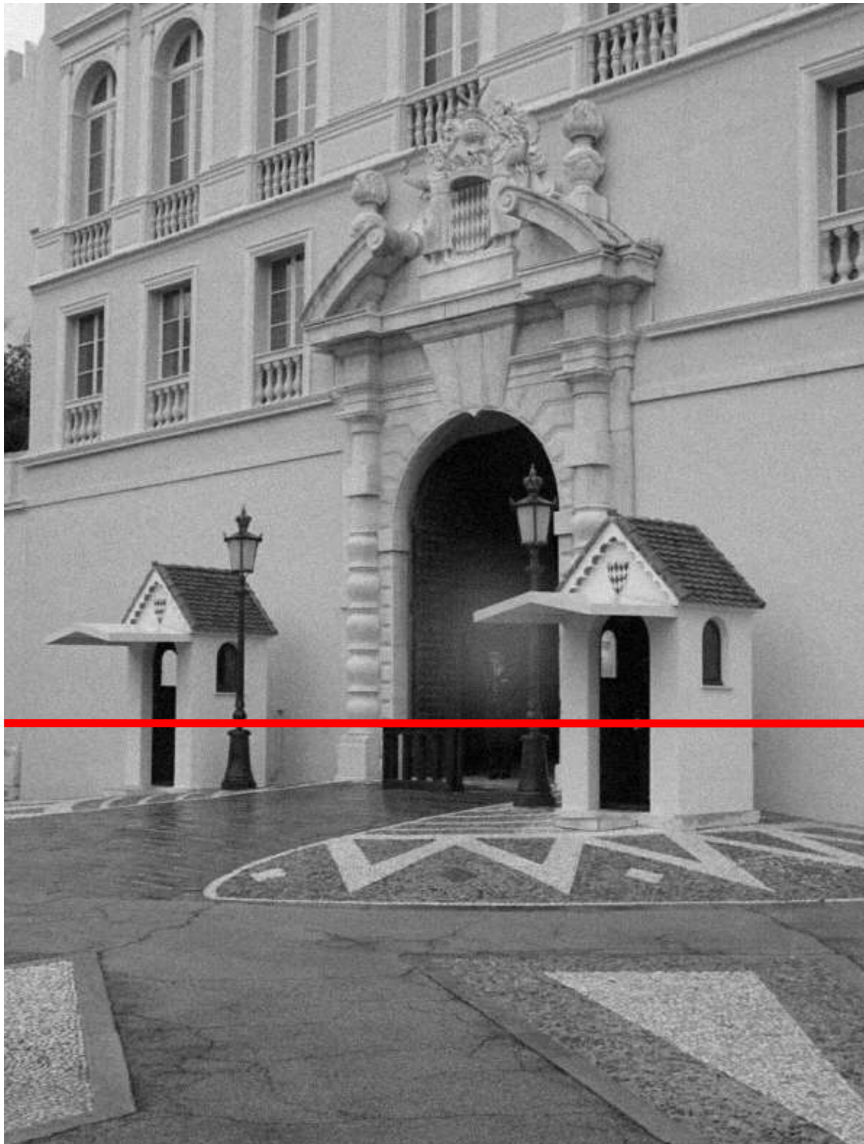
Intensity profile

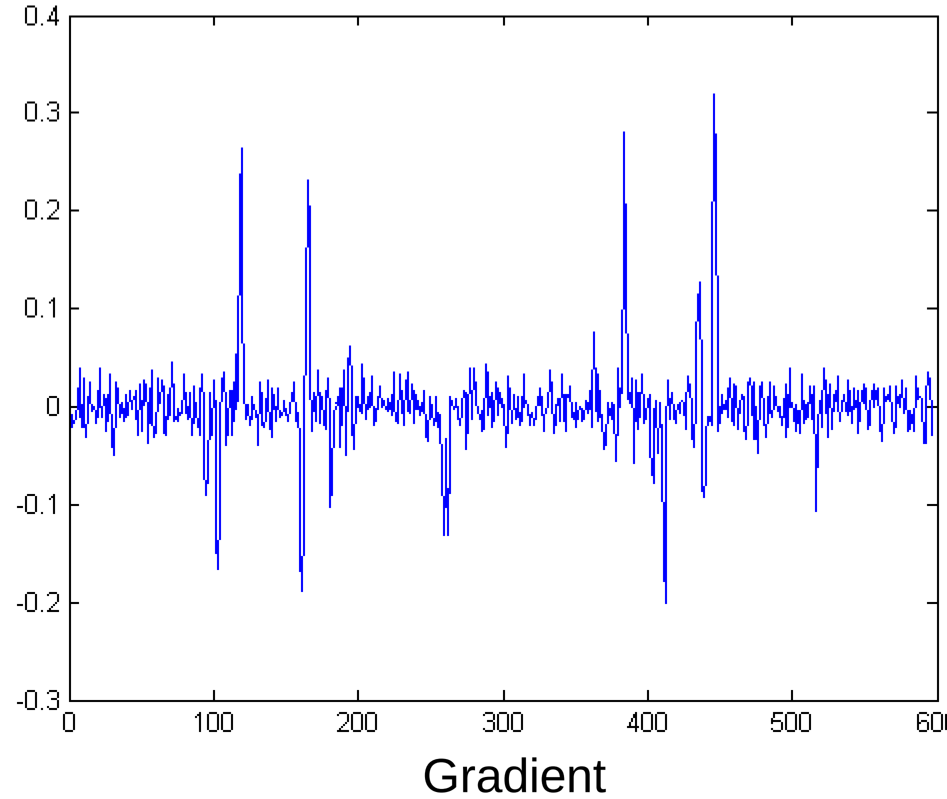




Source: D. Hoiem

With a little Gaussian noise





Source: D. Hoiem

Outline

- 边缘及边缘检测概念
- 用模板实现卷积
- 基于一阶导数的边缘检测
- 基于二阶导数的边缘检测
  – Laplacian算子
  – LoG算子 （Marr&Hildreth算子）
- **Canny边缘检测**

Convolution with Template/Kernel

- 模板(Template/Kernel): A matrix represents an operator. A convolution template centers on each pixel in an image and generates new output pixels.

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 4 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

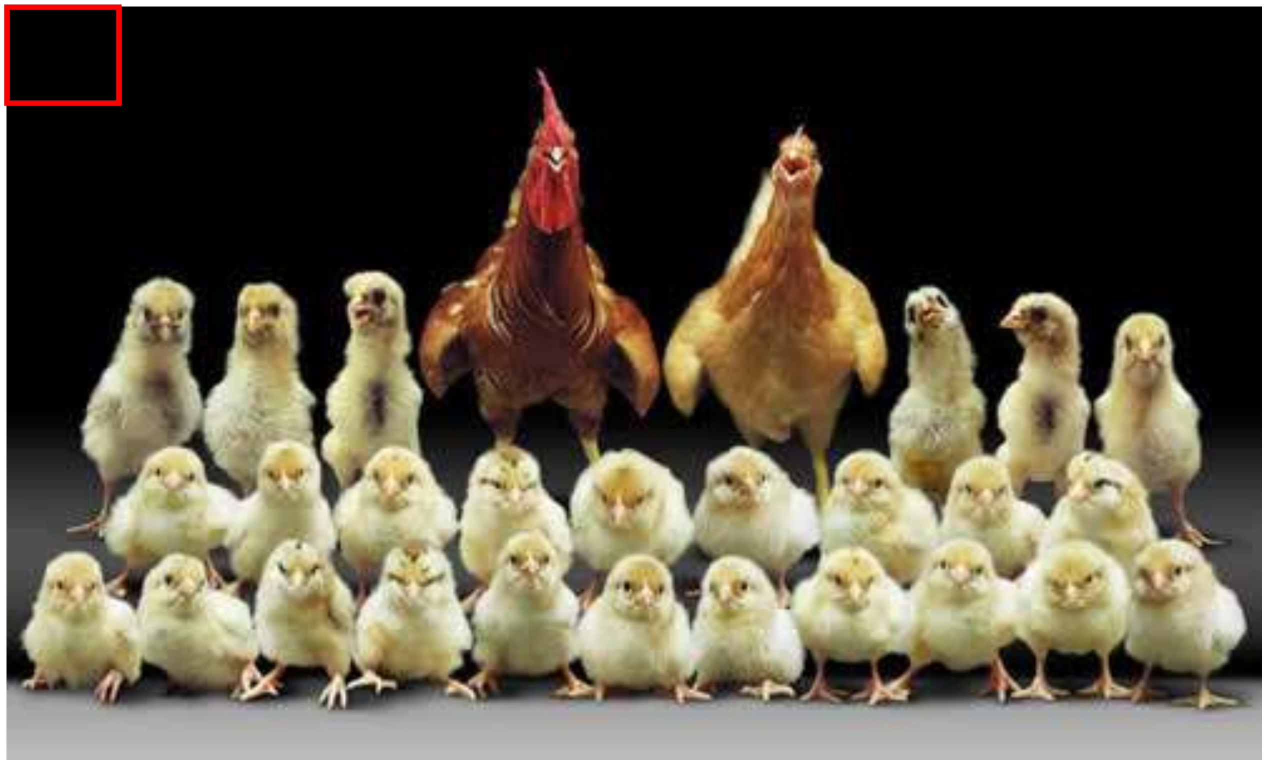
- 卷积(Convolution): by using the template, the new pixel value is computed by multiplying each pixel value in the neighborhood with the corresponding weight in the convolution mask and summing these products.

Convolution with Template/Kernel

34	20	10	30	38	198	246
28	45	0	1	4	9	2
0	9	0	0	0	2	0
238	5	5	2	9	3	9
8	98	1	8	2	8	2
2	5	4	7	1	6	2
9	3	6	5	3	1	4

-1	1	-1
1	4	1
-1	1	-1

Convolution with Template/Kernel



Convolution with Template/Kernel

- ***T*(*x*,*y*)** is a template (*n*x*m*), **I(x,y)** is an image (*M*x*N*), then the convoluting of *T* with *I* is

$$T * I(X,Y) = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} T(i,j)I(X+i,Y+j)$$

- Usually, the template is not allowed to shift off the edge of the image, so the resulting image will **normally be smaller** than the original image.

$$\begin{array}{|cc|}\hline 1 & 0 \\ 0 & 1 \\ \hline\end{array} * \begin{array}{|ccccc|}\hline 1 & 1 & 3 & 3 & 4 \\ 1 & 1 & 4 & 4 & 3 \\ 2 & 1 & 3 & 3 & 3 \\ 1 & 1 & 1 & 4 & 4 \\ \hline\end{array} = \begin{array}{|ccccc|}\hline 2 & 5 & 7 & 6 & * \\ 2 & 4 & 7 & 7 & * \\ 3 & 2 & 7 & 7 & * \\ * & * & * & * & * \\ \hline\end{array}$$

Outline

- 边缘及边缘检测概念
- 用模板实现卷积
- 基于一阶导数的边缘检测
- 基于二阶导数的边缘检测
  – Laplacian算子
  – LoG算子 （Marr&Hildreth算子）
- **Canny边缘检测**

基于一阶导数的边缘检测

- **梯度(Gradient)：** 是图像对应二维函数的一阶导数

$$G(x,y)=\begin{bmatrix} G_x \\ G_y \end{bmatrix}=\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- **梯度的幅值：** isotropic operator

anisotropic

$$|G(x,y)| = \sqrt{G_x^2 + G_y^2}$$

$$|G(x,y)| = |G_x| + |G_y|$$

$$|G(x,y)| \approx \max(|G_x|, |G_y|)$$

基于一阶导数的边缘检测

- 梯度方向:

$$a(x,y) = \arctan(G_y / G_x)$$

梯度方向为函数最大变化率方向

- 图像中用差分近似偏导数

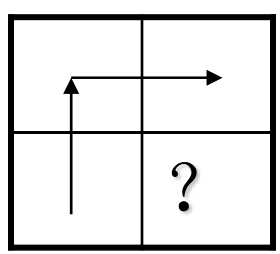
用差分近似偏导数

$$G_x = f[x+1, y] - f[x, y]$$
$$G_y = f[x, y] - f[x, y+1]$$

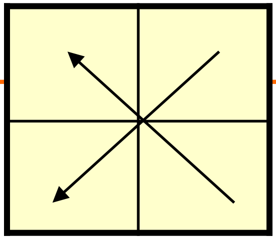
一般用卷积模板进行计算：

$$G_x = \boxed{-1 \mid 1} \qquad G_y = \begin{array}{|c|} \hline 1 \\ \hline -1 \\ \hline \end{array}$$

上述表示？



(1) Roberts交叉算子



$$G[i,j] = |G_x| + |G_y|$$

$$G[i,j] = \left| f[i,j] - f[i+1,j+1] \right| + \left| f[i+1,j] - f[i,j+1] \right|$$

$$G_x = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|} \hline 0 & -1 \\ \hline 1 & 0 \\ \hline \end{array}$$

2X2梯度算子？ **3X3梯度算子！**

(2)Sobel算子

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

(3)Prewitt算子：运算较快

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Sobel算子实例

均值差分：一定邻域内灰度平均值之差

3×3邻域加权

$$G_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$G_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4)$$

a_0	a_1	a_2
a_7	[i,j]	a_3
a_6	a_5	a_4

C=1: Prewitt算子
C=2: Sobel算子
C=3: Sethi算子

OpenCV函数

```
void cvSobel( const CvArr* src, CvArr* dst,
              int xorder, int yorder, int aperture_size=3 );
```

aperture_size: -1, 1, 3, 5, 7

X方向：xorder=1, yorder=0, aperture_size=3
Y方向：xorder=0, yorder=1, aperture_size=3

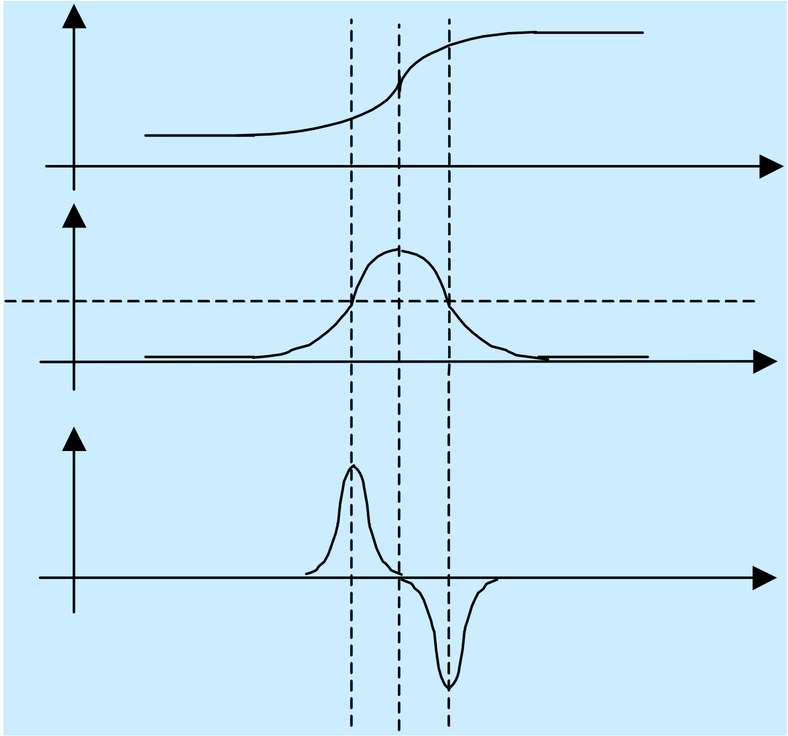
注意： **8bit depth 溢出！ 用 cvAbs cvConvertScale**
dst: IPL_DEPTH_16S or IPL_DEPTH_32F

```
void cvFilter2D( const CvArr* src, CvArr* dst,
                 const CvMat* kernel,
                 CvPoint anchor=cvPoint(-1,-1));
float k[9]={  1,   2,  1,
              0,   0,  0,
             -1,  -2, -1 };
CvMat kernel=cvMat(3,3,CV_32FC1, k);  【kernel必须为浮点类型】
```

Outline

- 边缘及边缘检测概念
- 用模板实现卷积
- 基于一阶导数的边缘检测
- 基于二阶导数的边缘检测
  - Laplacian算子
  - LoG算子 （Marr&Hildreth算子）
- **Canny边缘检测**

基于二阶导数的边缘检测



二阶微分算子

图像灰度二阶导数的过零点对应边缘点.

拉普拉斯（Laplacian）算子

拉普拉斯算子是二阶导数的二维等效式:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

用差分近似微分:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} \\ &= \frac{\partial (f[i,j+1] - f[i,j])}{\partial x} \\ &= \frac{\partial f[i,j+1]}{\partial x} - \frac{\partial f[i,j]}{\partial x} \\ &= (f[i,j+1] - f[i,j]) - (f[i,j] - f[i,j-1])\end{aligned}$$

$$\frac{\partial^2 f}{\partial x^2} = (f[i,j+1] - 2f[i,j]) + f[i,j-1]$$

$$\frac{\partial^2 f}{\partial y^2} = (f[i+1,j] - 2f[i,j]) + f[i-1,j]$$

表示为卷积模板：

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

邻域中心点具有更大权值的近似算子：

$$\nabla^2 \approx \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

LoG边缘检测算法

LoG = Laplacian of Gaussian

高斯滤波+拉普拉斯边缘检测

基本特征：

- 平滑滤波器是高斯滤波器.
- 采用拉普拉斯算子计算二阶导数.
- 边缘检测判据是二阶导数零交叉点并对应一阶导数的较大峰值.
- 使用线性内插方法在子像素分辨率水平上估计边缘的位置.

（Marr & Hildreth 80）

LoG算子

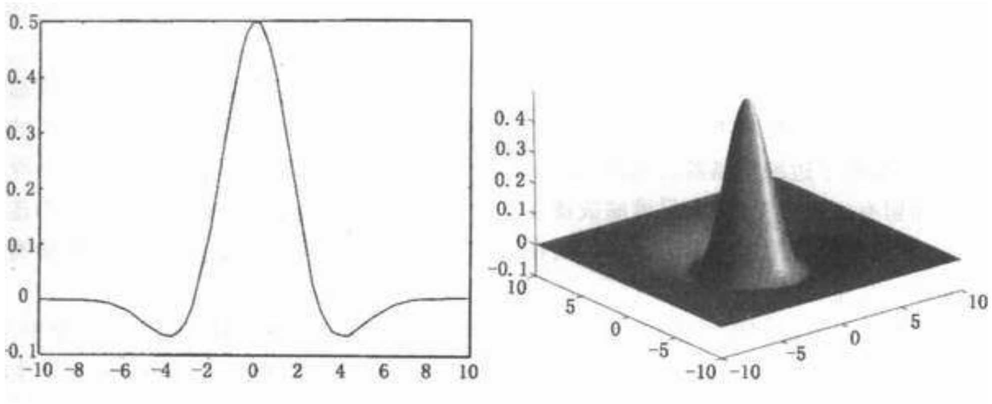
$$h(x,y)=\nabla^2[g(x,y)*f(x,y)]$$

根据卷积求导法

$$h(x,y)=[\nabla^2 g(x,y)]*f(x,y)$$

$$\nabla^2 g(x,y)=\left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

墨西哥草帽算子:



两种等效计算方法

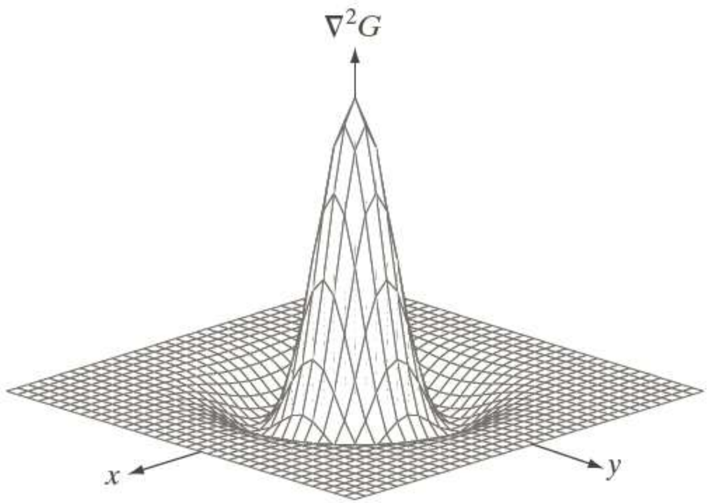
1. 图像与高斯函数卷积，再求卷积的拉普拉斯微分

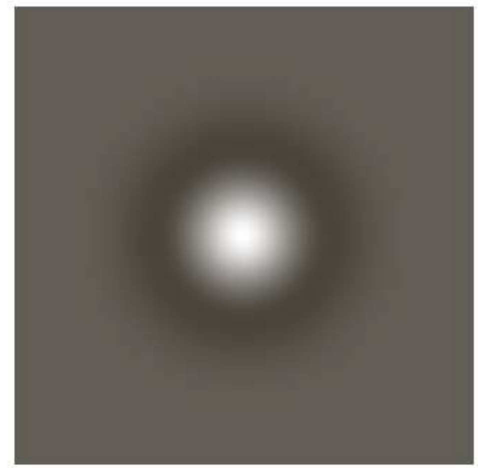
2. 求高斯函数的拉普拉斯微分，再与图像卷积

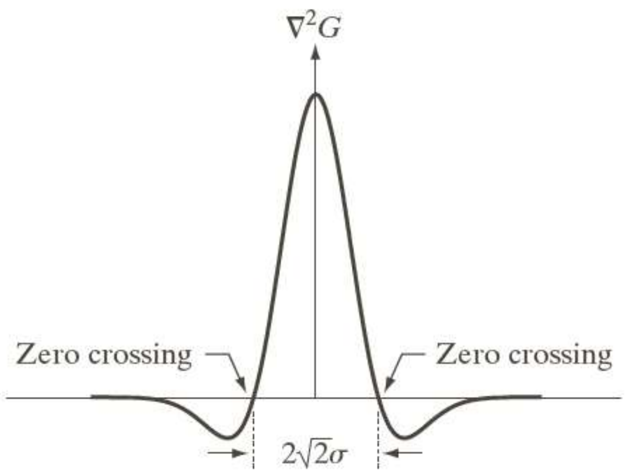
0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

5X5拉普拉斯高斯模板

LoG算子







0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

LoG算子

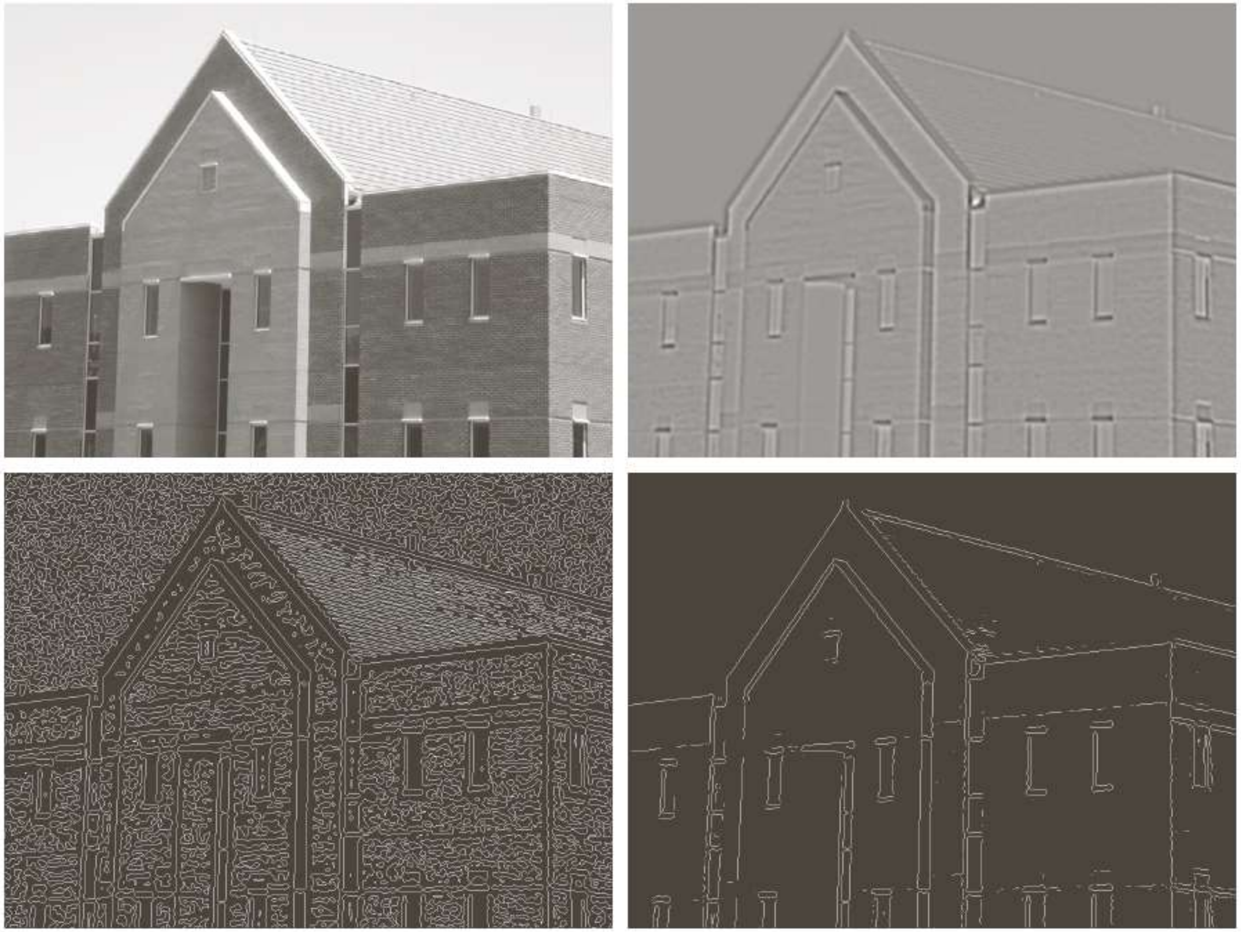


FIGURE 10.22 (a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

OpenCV相关函数

- **void cvLaplace**

  (CvArr *src, CvArr *dst, int aperture_size=3)

  – If aperture_size=1

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

  – 不进行Scale变换

- **cvFilter2D**

Outline

- 边缘及边缘检测概念
- 用模板实现卷积
- 基于一阶导数的边缘检测
- 基于二阶导数的边缘检测
- **Canny边缘检测**

A computational approach to edge detection

J Canny - IEEE Transactions on pattern analysis and machine ..., 1986 - ieeexplore.ieee.org

This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the ...

Cited by 33096 Related articles All 22 versions

Canny 边缘检测器

算法步骤：

1. 用高斯滤波器平滑图像.
2. 用一阶偏导有限差分计算梯度幅值和方向.
3. 对梯度幅值进行非极大值抑制（NMS）.
4. 用双阈值算法检测和连接边缘.

Why 高斯滤波器?

平滑去噪和边缘检测是一对矛盾，应用高斯函数的一阶导数，在二者之间获得最佳的平衡。

[Canny86]

步1. 图像与高斯平滑滤波器卷积:

$$S[i, j] = G[i, j; \sigma] * I[i, j]$$

步2a. 使用一阶有限差分计算偏导数阵列P与Q:

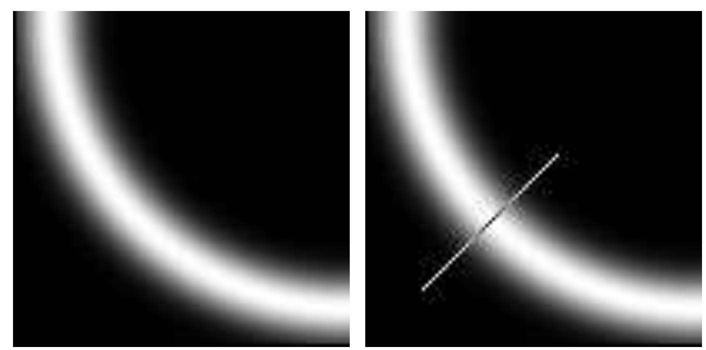
$$G_x[i, j] \approx (S[i, j+1] - S[i, j] + S[i+1, j+1] - S[i+1, j]) / 2$$
$$G_y[i, j] \approx (S[i, j] - S[i+1, j] + S[i, j+1] - S[i+1, j+1]) / 2$$

步2b. 计算梯度幅值与方向角:

$$M[i, j] = \sqrt{G_x[i, j]^2 + G_y[i, j]^2}$$

$$\theta[i, j] = \arctan(G_y[i, j] / G_x[i, j])$$

Before Non-max Suppression

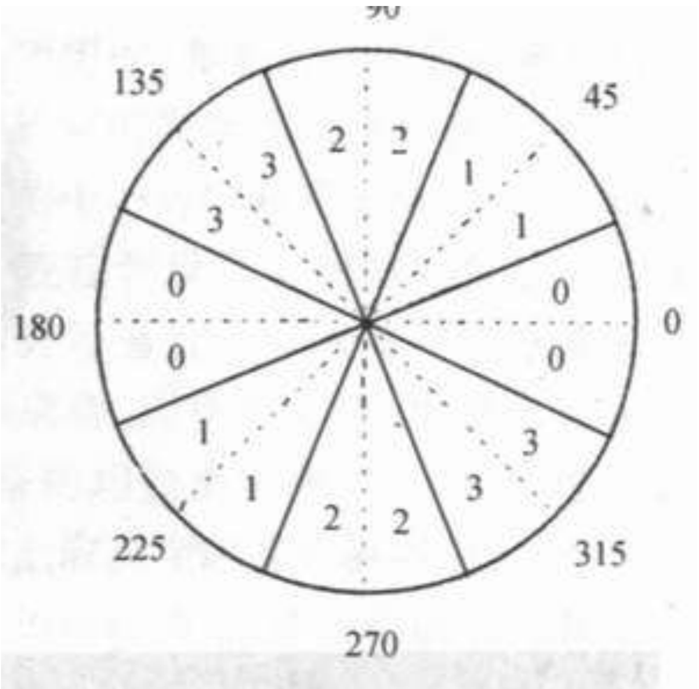


After non-max suppression

步3. 非极大值抑制（NMS ）：

去掉幅值局部变化非极大的点.

* 将梯度角离散为圆周的四个扇区之一，以便用3×3的窗口作抑制运算



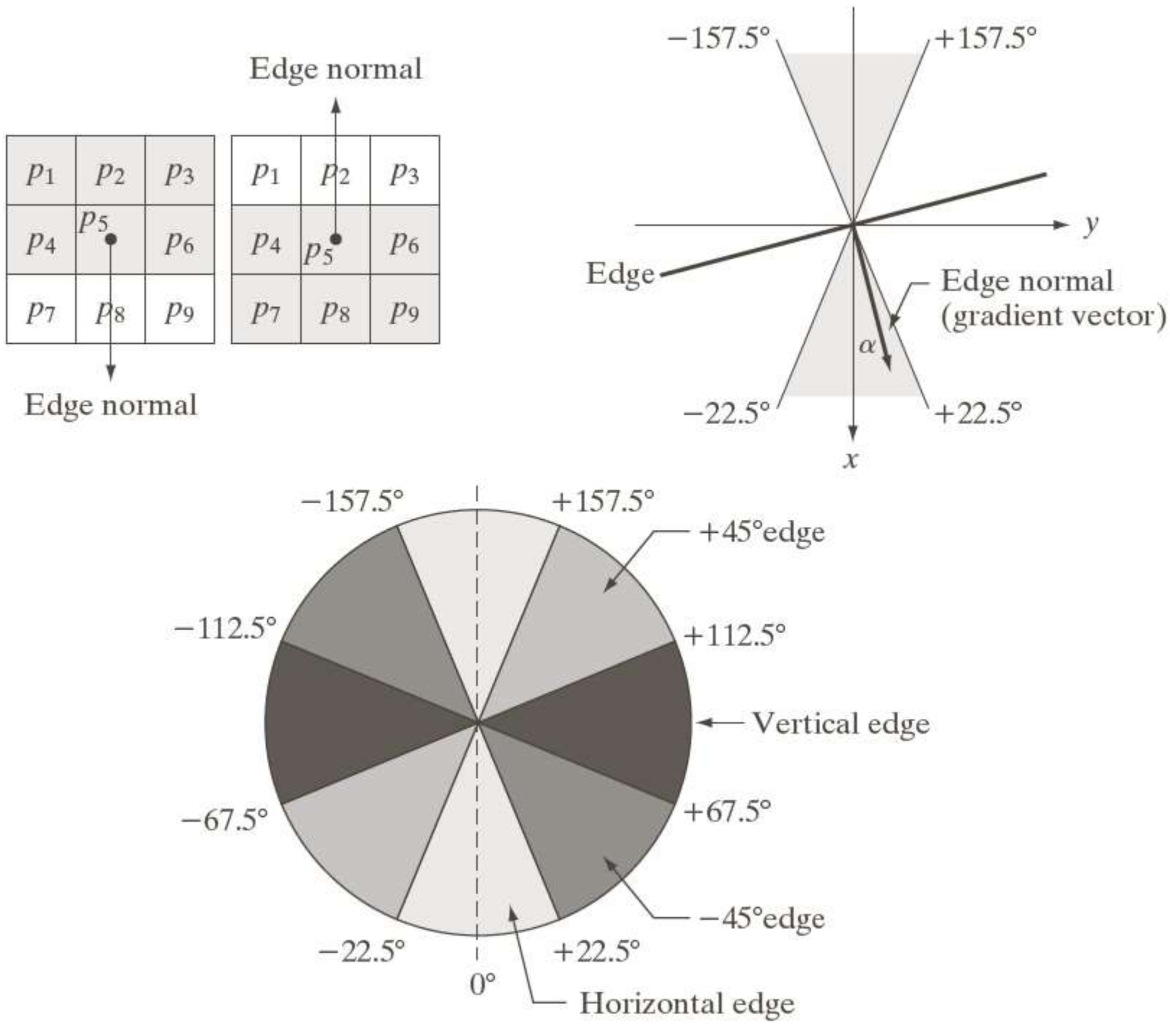
* 方向角离散化：

$$\zeta[i, j] = \mathrm{Sector}(\theta[\mathrm{i}, \mathrm{j}])$$

* 抑制，得到新幅值图：

$$N[i, j] = \mathrm{NMS}(M[i, j], \zeta[i, j])$$

How抑制？ 若M[i,j]不比沿梯度线方向上的两个相邻点幅值大，则N[i,j]=0



a b
c

FIGURE 10.24
(a) Two possible orientations of a horizontal edge (in gray) in a 3×3 neighborhood. (b) Range of values (in gray) of α, the direction angle of the *edge normal*, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3×3 neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.

After non-max suppression

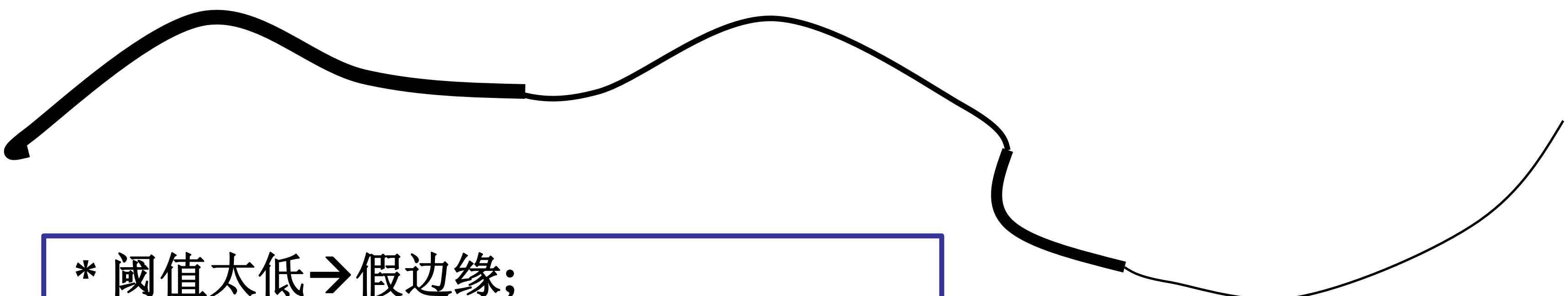
步4. 双阈值化并边缘链接

(a) 取高低两个阈值(T2, T1)作用于新幅值图N[i,j]，得到两个边缘图：高阈值和低阈值边缘图。

高阈值图：N[i,j] > T2;

低阈值图：N[i,j] > T1

(b) 连接高阈值边缘图，出现断点时，在低阈值边缘图中的8邻点域搜寻边缘点。



* 阈值太低→假边缘;
* 阈值太高→部分轮廓丢失.
* 选用两个阈值: 更有效的阈值方案.

最终结果

提取彩色边缘

最终边缘结果图（作为mask）覆盖在原彩色图像后，可提取出彩色边缘图

Canny 边缘检测器

算法步骤：

1. 用高斯滤波器平滑图像.
2. 用一阶偏导有限差分计算梯度幅值和方向.
3. 对梯度幅值进行非极大值抑制（NMS） .
4. 用双阈值算法检测和连接边缘.

[Canny86]

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

a	b
c	d

FIGURE 10.25 (a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

a	b
c	d

FIGURE 10.26 (a) Original head CT image of size 512×512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

OpenCV相关函数

- **cvCanny**

    (CvArr* src, CvArr* dst,
    double threshold1, double threshold2
    int aperture_size=3)

  - threshold2 ~= (2 or 3) * threshold1
  - src must: grayscale; dst must: grayscale
  - aperture_size: for cvSobel()

Review

- 边缘及边缘检测概念
- 用模板实现卷积
- 基于一阶的边缘检测
- 基于二阶的边缘检测
  - Laplacian算子
  - LoG算子 （Marr&Hildreth算子）
- **Canny边缘检测**