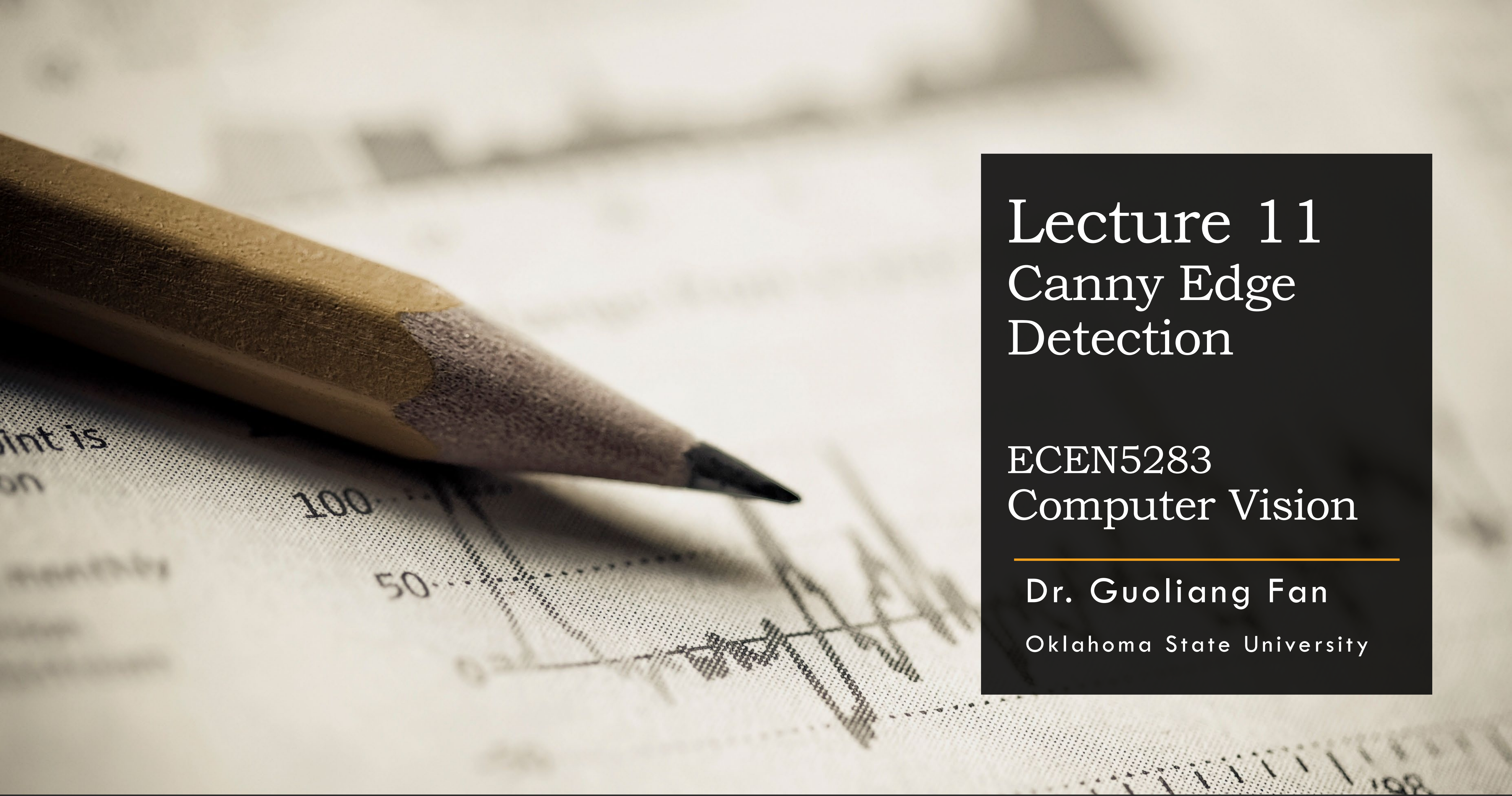


Lecture 11
Canny Edge Detection

ECEN5283 Computer Vision

Dr. Guoliang Fan

Oklahoma State University

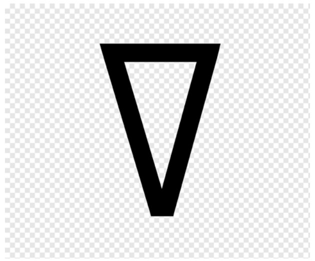
Goals

To review the LoG edge detector.

To implement the gradient-based Canny edge detector.

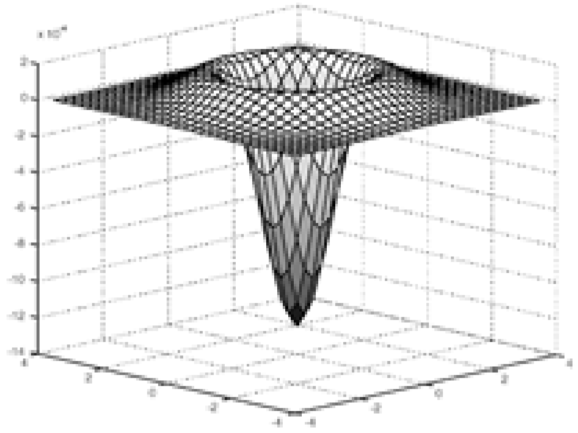
To present some deep learning approaches to Canny edge detection.

John F. Canny
Professor
EECS, UC-Berkeley



Nabla Symbol for the Differential Operator

Determining a Discrete LoG Kernel



$$\nabla^2 G_\sigma(x,y) = \left(\frac{1}{2\pi\sigma^4}\right)\left[\frac{x^2+y^2}{\sigma^2} - 2\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{(continuous LoG)}$$

$$K[i,j] = \left(\frac{1}{2\pi\sigma^4}\right)\left[\frac{(i-k-1)^2+(j-k-1)^2}{\sigma^2} - 2\right] e^{-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}} \quad \text{(Discrete LoG)}$$

The dimension of the kernel is $(2k+1)\times(2k+1)$.

The variance σ^2 will determine the dimension of the kernel.

π is usually ignored in computing the LoG coefficients.

Some rounding may be involved to make coefficient integers to speed up the 2D convolution.

The average of all kernel coefficients must be zero (why?).

The Steps of LoG Edge Detection

Applying the LoG to the image.

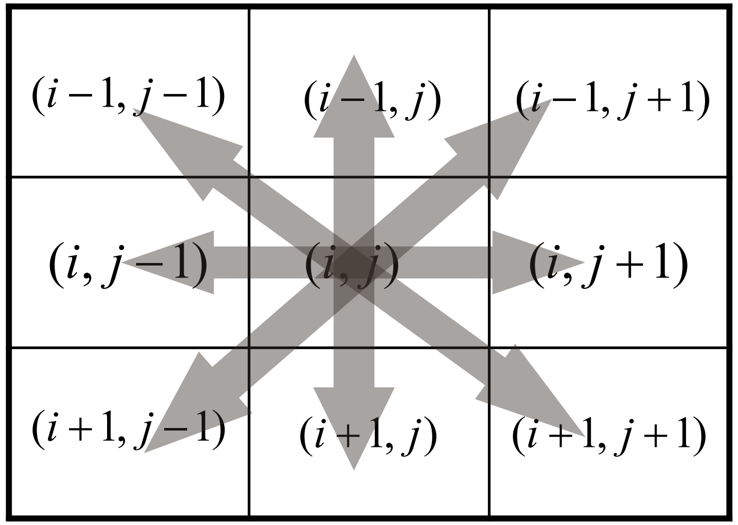
Detection of zero-crossings (ZCs) in the image (how?).

Threshold the ZCs to keep only strong ones with significant difference between the positive and negative values.

Remove isolated edge points caused by noise by counting the number of pixels in each connected component and removing smaller components.

- (To be discussed in more details later.)

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$



Gradient-based Edge Detection

The image gradient is a vector defined as

$$\nabla f = grad(f) = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Magnitude: $M(x, y) = \sqrt{f_x^2 + f_y^2} \approx |f_x| + |f_x|$

Direction: $\alpha(x, y) = \tan^{-1}\left(\frac{f_y}{f_x}\right)$ (with possible $\pm\pi$)

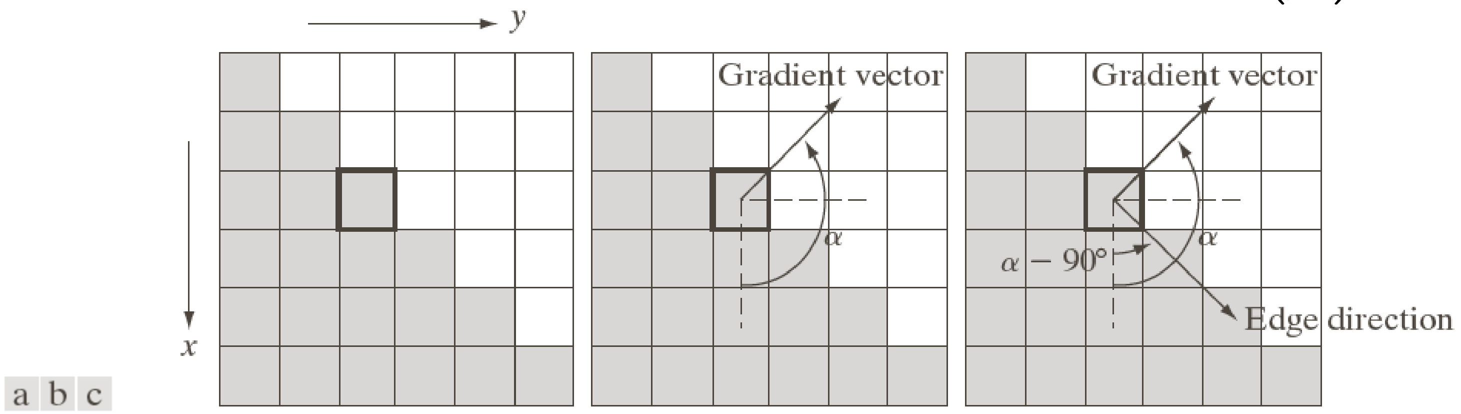


FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

2D Gradient Operators

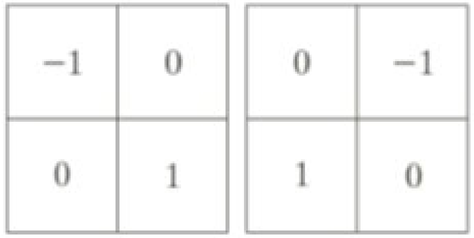
f_x

-1
1

f_y

-1	1

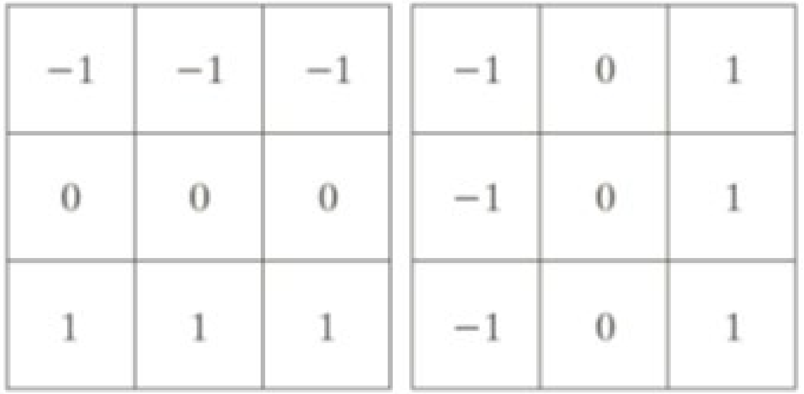
$$\leftarrow \begin{cases} f_x = \dfrac{\partial f}{\partial x} = f(x,y) - f(x-1,y) \\ f_y = \dfrac{\partial f}{\partial y} = f(x,y) - f(x,y-1) \end{cases}$$



−1	0
0	1

0	−1
1	0

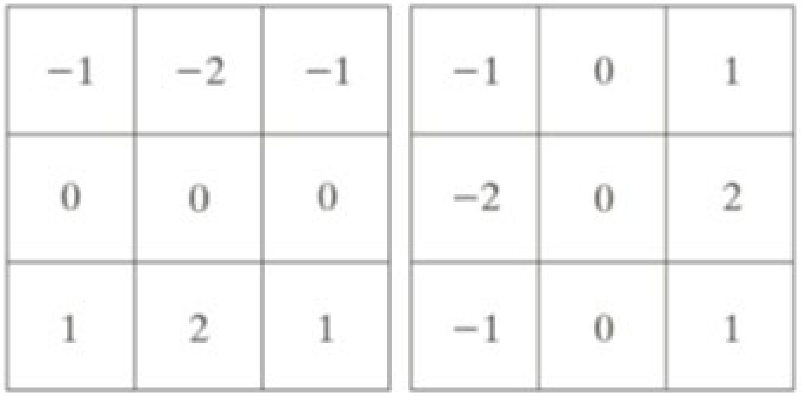
Roberts



−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

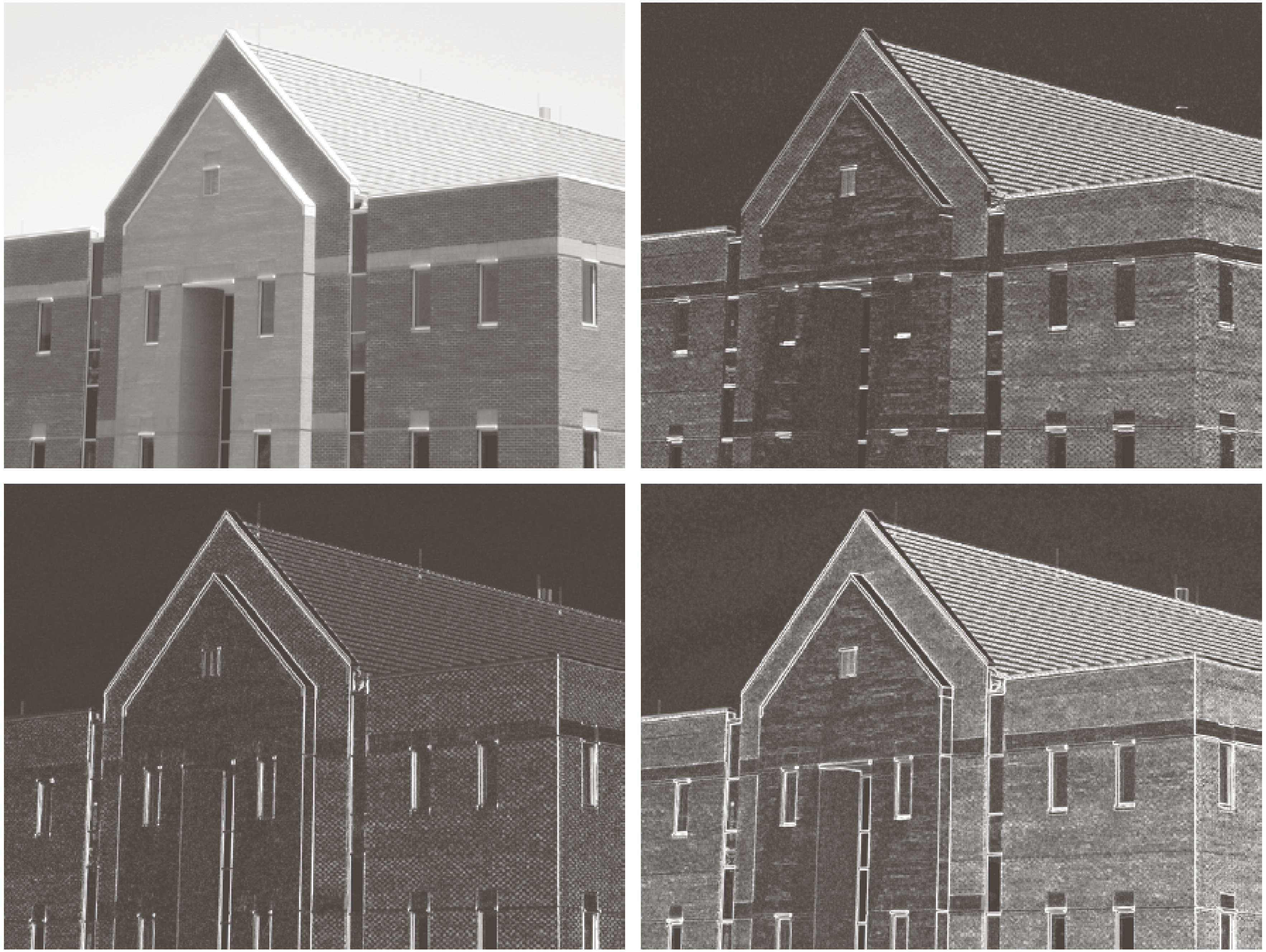
Prewitt



−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel



a	b
c	d

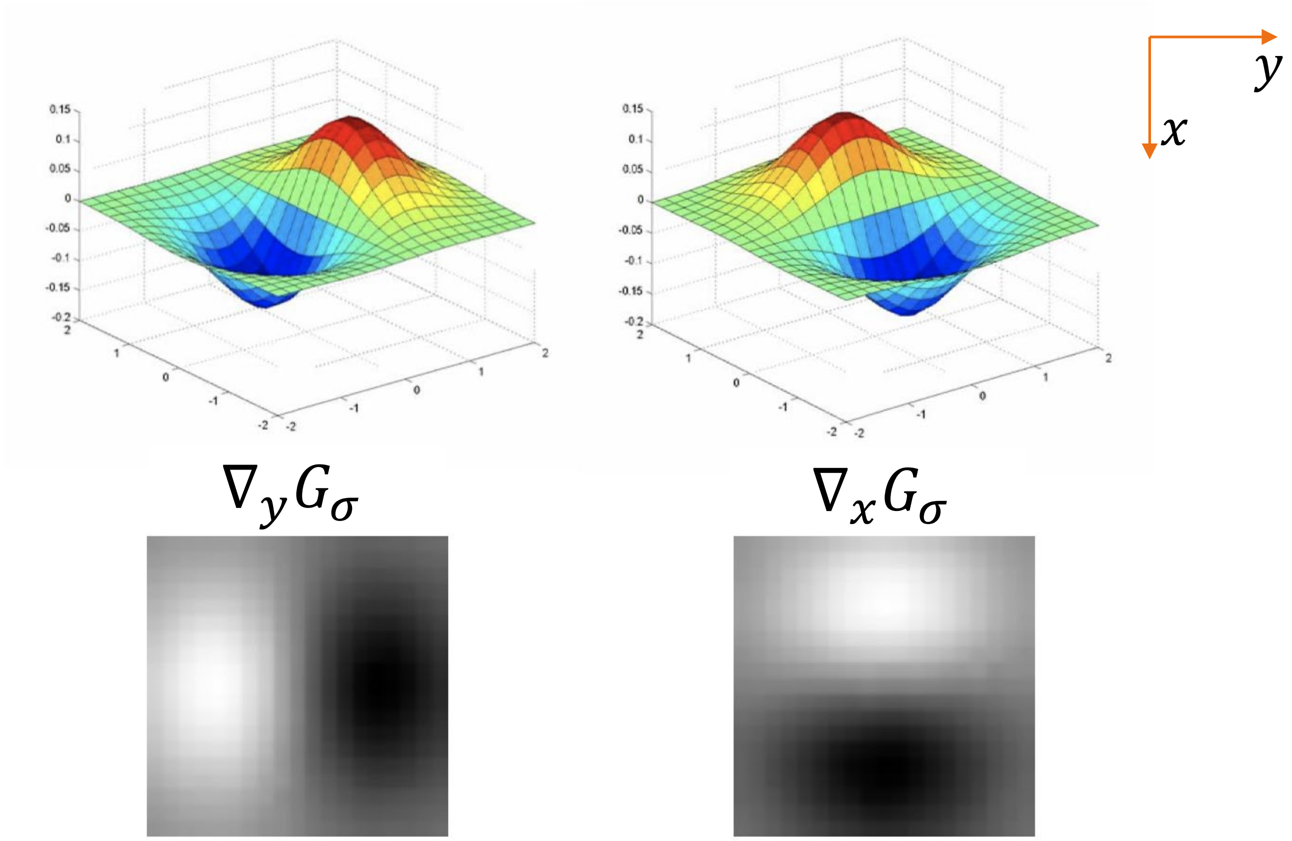
FIGURE 10.16 (a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

Canny Edge Detector

We can apply the first order derivative of a Gaussian to find the image gradient:

$$\nabla_y G_\sigma = \frac{-y}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$\nabla_x G_\sigma = \frac{-x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



Canny Edge Detection

Step 1: We first compute the two first-order derivative maps ($f_x(x,y), f_y(x,y)$).

$$f_x = \nabla_x G_\sigma * f(x,y)$$

$$f_y = \nabla_y G_\sigma * f(x,y)$$

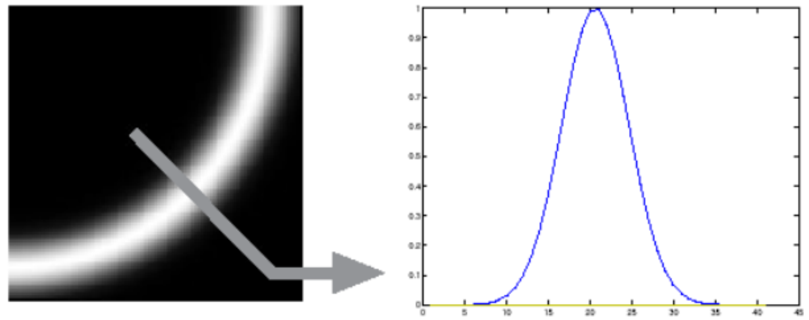
Step 2: We can compute the magnitude and orientation of the gradient $M(x,y)$ and direction $\alpha(x,y)$ maps for each pixel.

$$M(x,y) = \sqrt{f_x^2 + f_y^2}$$

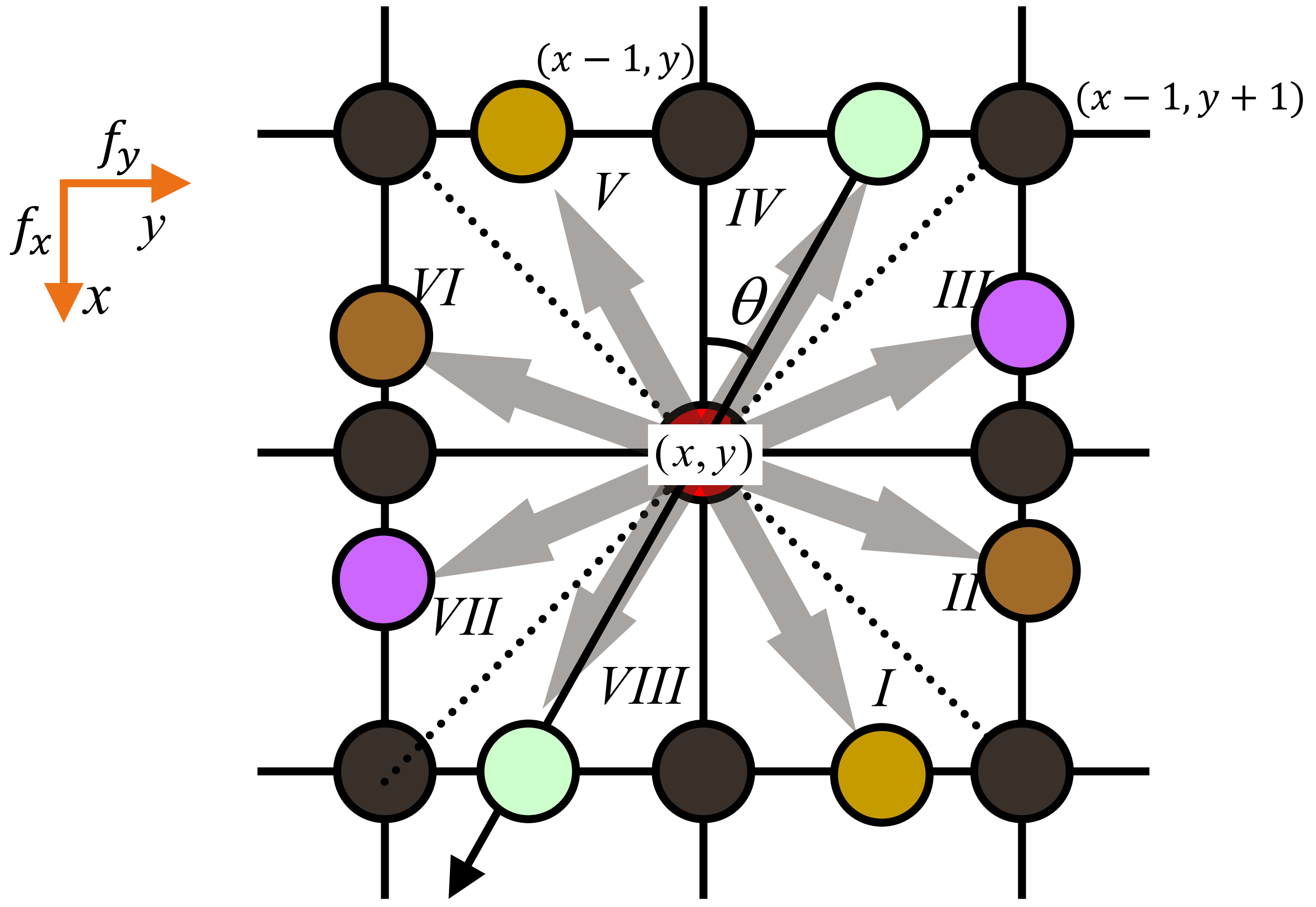
$$\alpha(x,y) = \tan^{-1}\left(\frac{f_y}{f_x}\right)$$

Step 3: Apply non-maximum suppression to the gradient magnitude image and find the local maximum along the gradient direction.

Step 4: Use connectivity analysis to detection and link weak edges.



Non-maximum Suppression



An edge point (x,y) is declared if it is a larger than the two neighboring pixels along the gradient direction (by linear interpolation)

Case 1 (I/V): $f_x f_y > 0, |f_x| \geq |f_y|$

Case 2 (II/VI): $f_x f_y > 0, |f_x| < |f_y|$

Case 3 (VIII/IV): $f_x f_y < 0, |f_x| \geq |f_y|$

Case 4 (VII/III): $f_x f_y < 0, |f_x| < |f_y|$

Special Case: $f_x = 0$ or $f_y = 0$

Edge Linking and Tracking: Double Thresholds and Hysteresis

Edge linking by double thresholds:

- It is essential to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value.
- We use a high-threshold to start edge curves and a low threshold to continue them. The two threshold values are empirically determined

Edge tracking by hysteresis:

- To preserve edge connection, window-based analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels.
- As long as there is one strong edge pixel that is involved in the blob, that weak edge point can be identified as one that should be preserved.

Canny-Net:

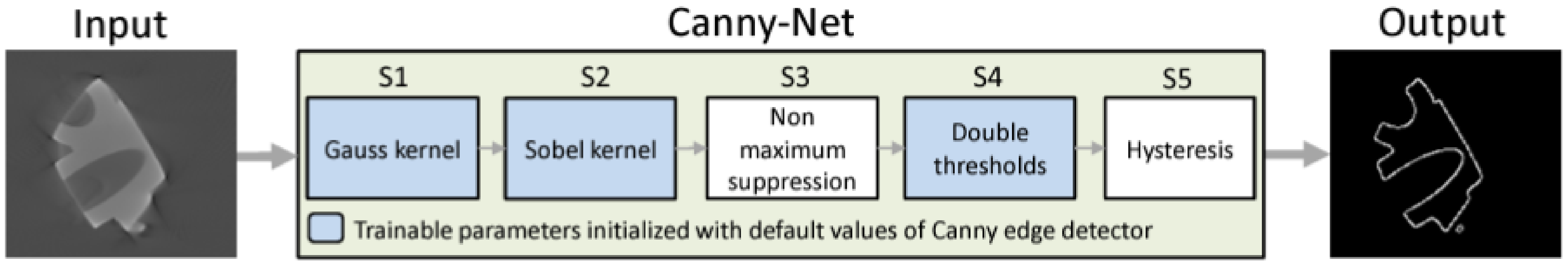


Figure 2: Complete architecture of Canny-Net consisting of five steps (S1-S5). The input image contains noticeable artifacts. Canny-Net's architecture is equivalent to the Canny edge detector and the trainable parameters (blue) of Canny-Net are initialized with the default values of the Canny edge detector. The network yields a binary image with background and edge pixels.

Wittmann J, Herl G. Canny-Net: Known Operator Learning for Edge Detection.

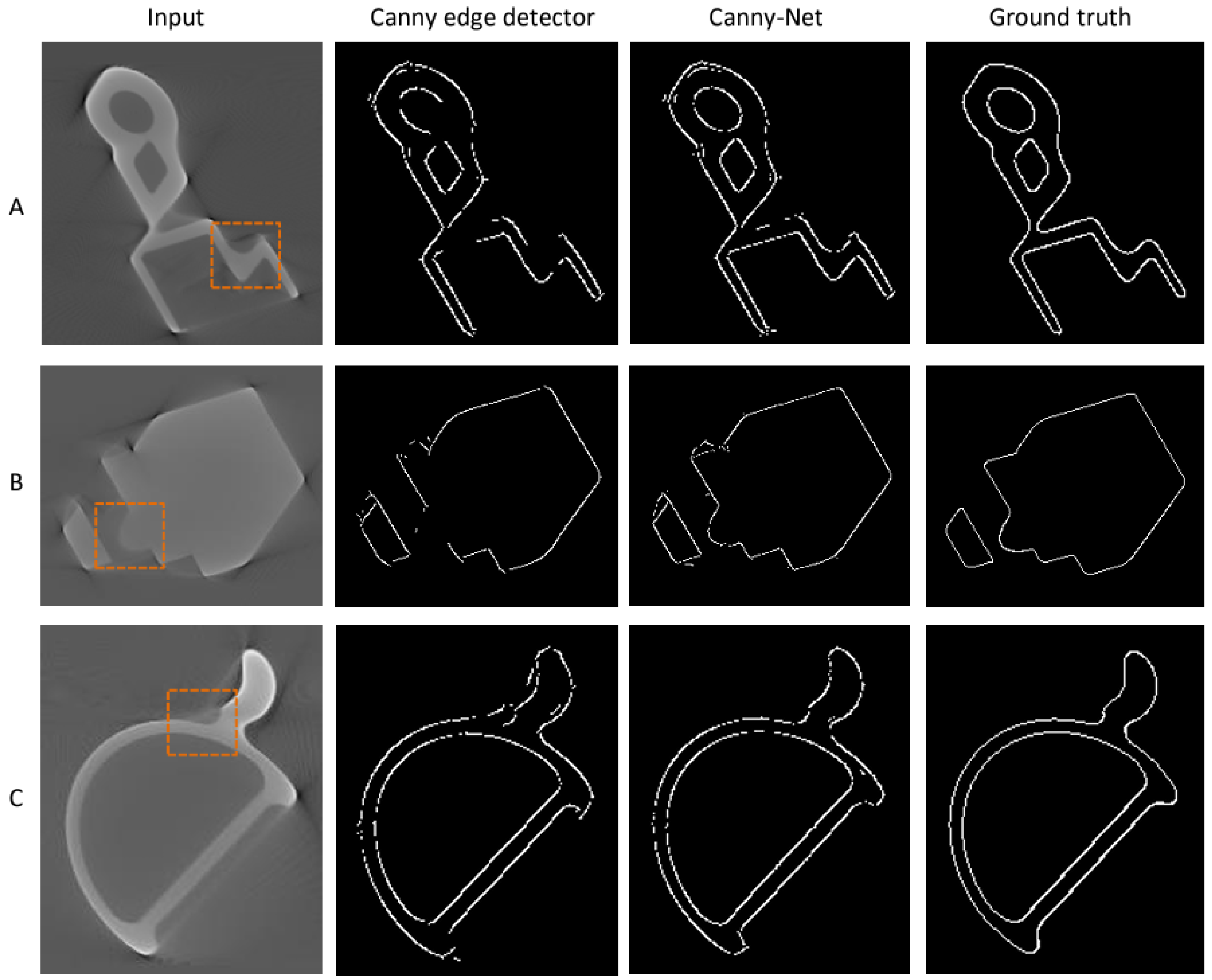


Figure 4: Comparison of the proposed methods on three mono-material metal objects A,B and C. Edge detection is performed on the input image using Canny edge detector and Canny-Net. When compared to the ground truth edges, both approaches yield similar results on sharp edges. In addition, Canny-Net performs better on edges affected by metal-induced artifacts, as illustrated by the areas located inside the orange boxes.

Holistically-Nested Edge Detection

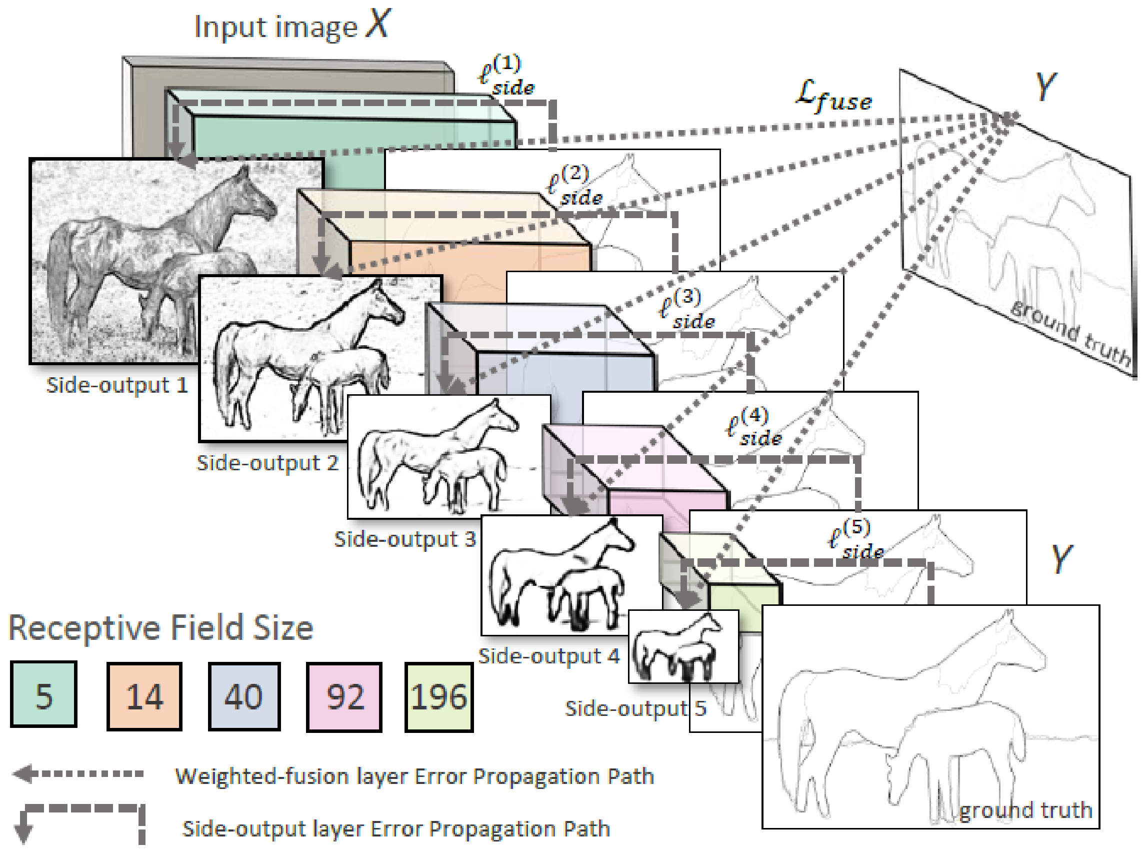
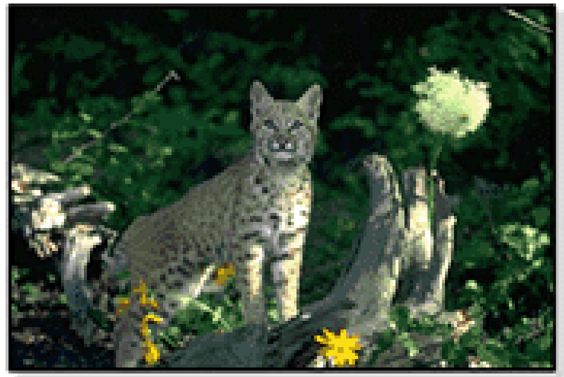
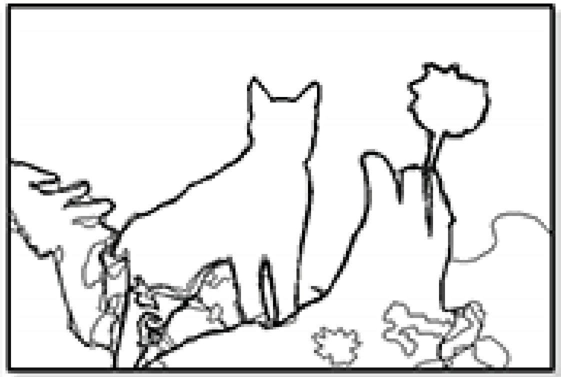
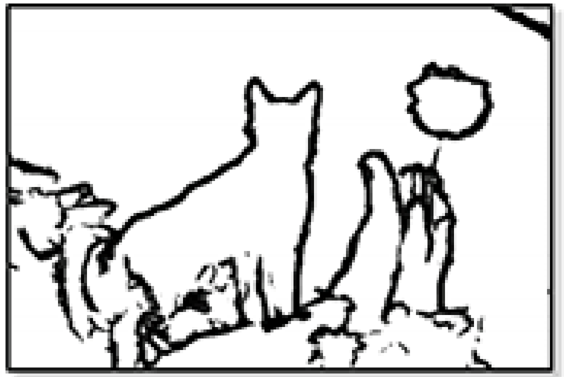


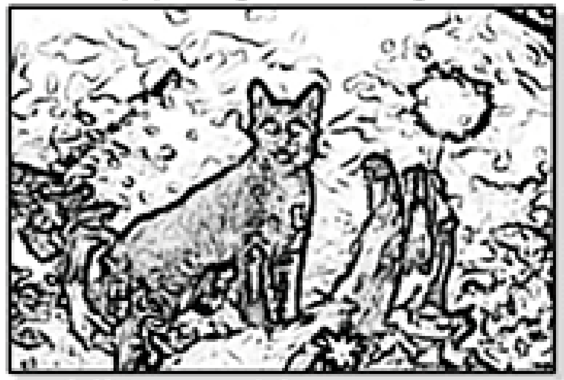
Figure 3. Illustration of our network architecture for edge detection, highlighting the error backpropagation paths. Side-output layers are inserted after convolutional layers. Deep supervision is imposed at each side-output layer, guiding the side-outputs towards edge predictions with the characteristics we desire. The outputs of HED are multi-scale and multi-level, with the side-output-plane size becoming smaller and the receptive field size becoming larger. One weighted-fusion layer is added to automatically learn how to combine outputs from multiple scales. The entire network is trained with multiple error propagation paths (dashed lines).

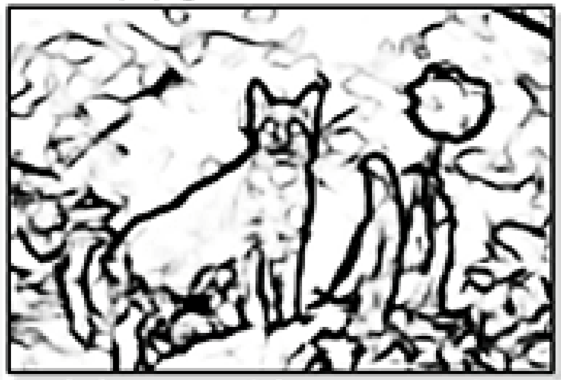
Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 1395-1403).

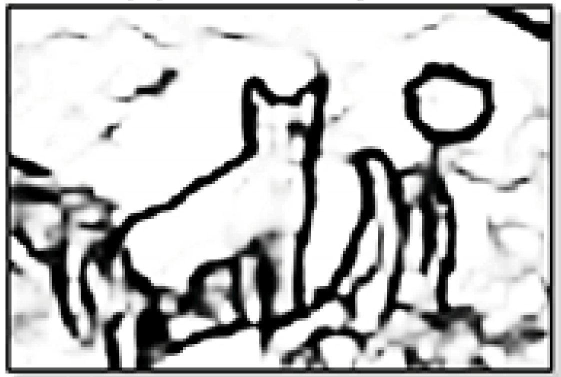

(a) original image

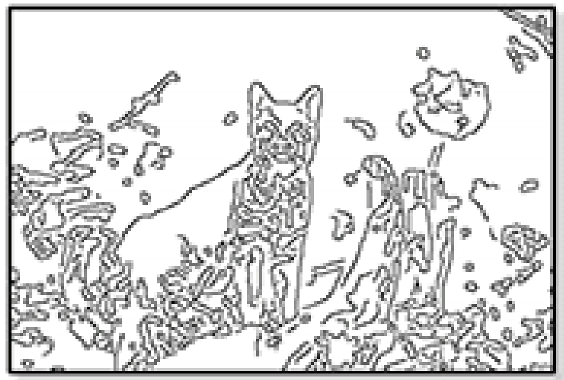

(b) ground truth

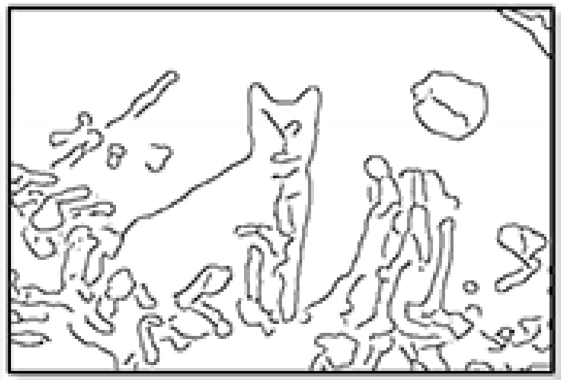

(c) HED: output

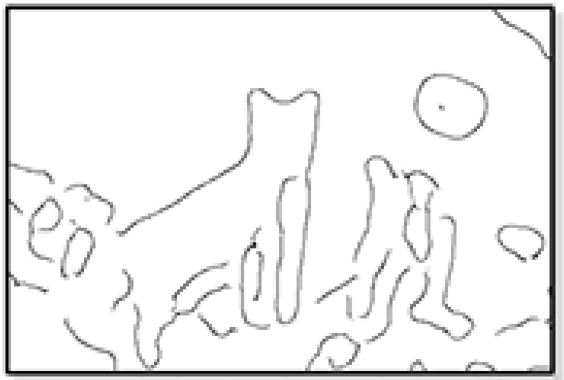

(d) HED: side output 2


(e) HED: side output 3


(f) HED: side output 4


(g) Canny: $\sigma = 2$


(h) Canny: $\sigma = 4$


(i) Canny: $\sigma = 8$

Do you see anything Interesting from Canny results under different Variances?