Image Processing

Jitendra Malik

Different kinds of images

- Radiance images, where a pixel value corresponds to the radiance from some point in the scene in the direction of the camera.
- Other modalities
  - X-rays, MRI…
  - Light Microscopy, Electron Microscopy…
  - …

A function $I(x,y)$ or $I(x,y,t)$ or $I(x,y,z)$ …

Canonical Image Processing problems

- Image Restoration
  - denoising
  - deblurring
- Image Compression
  - JPEG, JPEG2000, MPEG..
- Computing Field Properties
  - orientation
  - optical flow
  - disparity
- Locating Structural Features
  - corners
  - edges

Image Restoration
(lookup Wikipedia for more details)

- Based on priors of what the “true” image should be like. Typically the world consists of opaque piecewise smooth surfaces, and illumination is also piecewise smooth, therefore the resulting radiance images are piecewise smooth.
- Some techniques
  - Median filtering
  - Gaussian smoothing
  - Anisotropic diffusion
  - Non-Local means
  - Deconvolution

Image Compression

- Based on prior distributions on natural images, as well as properties of the human visual system, which is more sensitive to some error than others

Computing field properties

these are defined at every pixel (x,y)

- Orientation
  - at every pixel, one can define a local orientation by computing the gradient of the image
- Optical Flow
  - at every pixel, a vector corresponding to the movement from one time frame to the next
- Binocular Disparity
  - at every pixel, a vector corresponding to the displacement of the corresponding point from the left to the right image

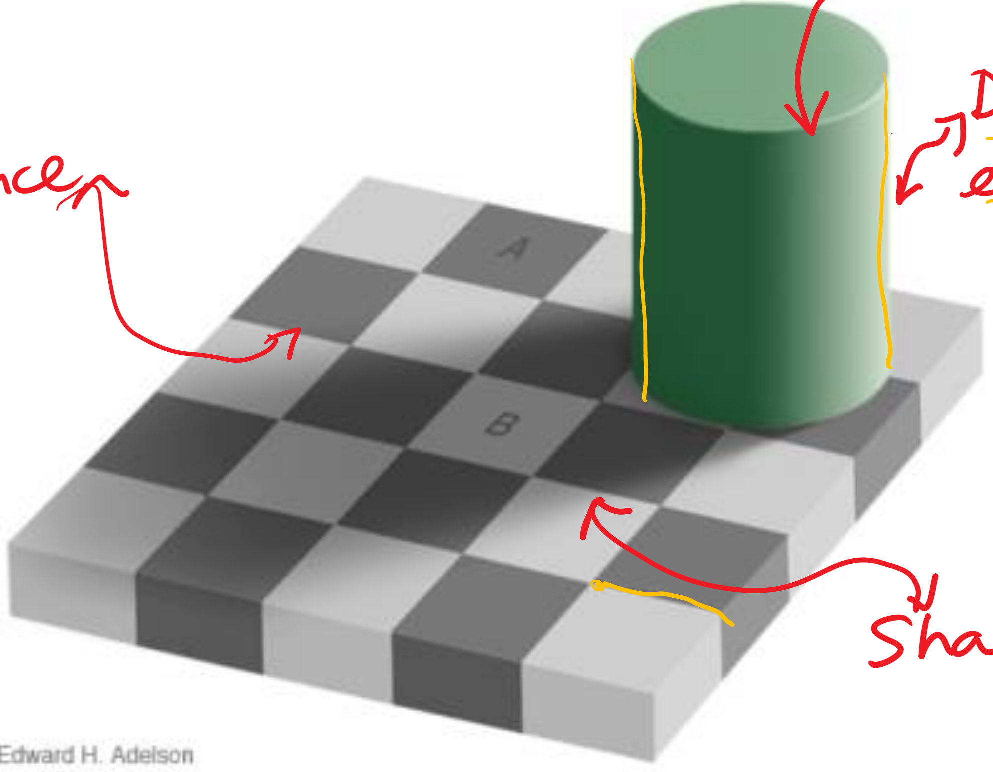
Locating Structural Features

- Edges are curves in the image, across which the brightness changes “a lot”
- Corners/Junctions

$\hat{n}$ edges

Depth edges

Reflectance edges

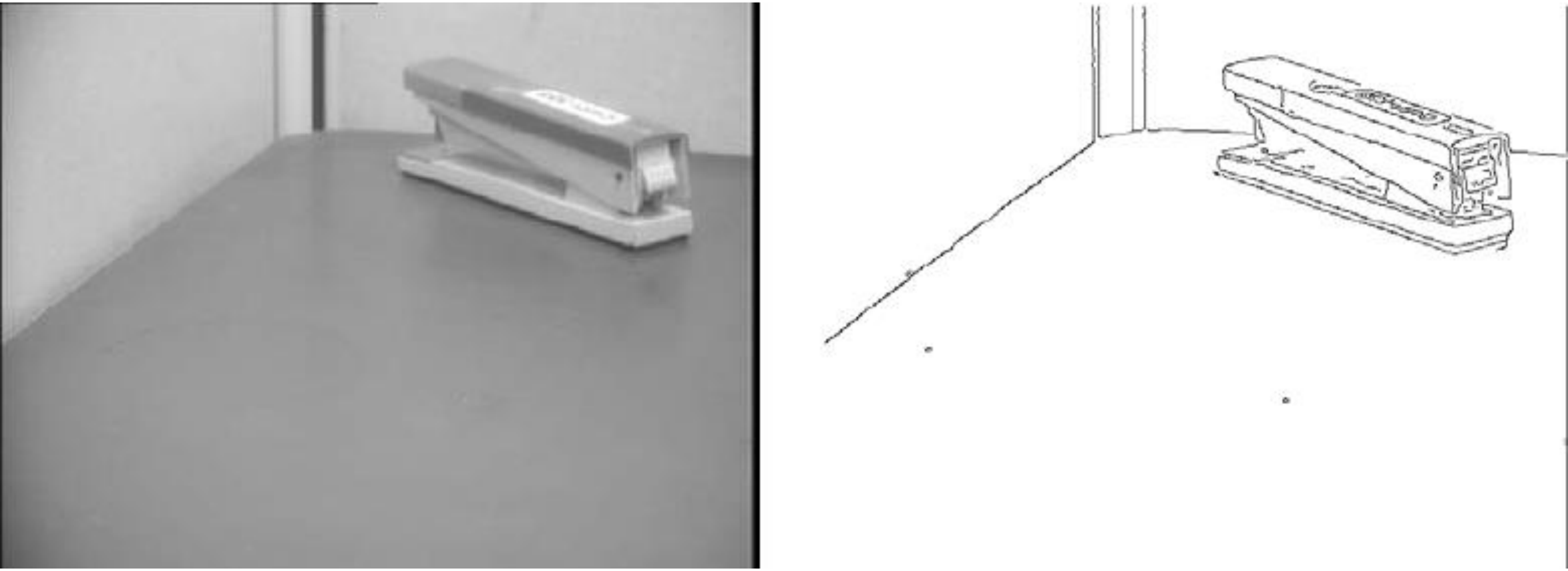


Edward H. Adelson

Shadows

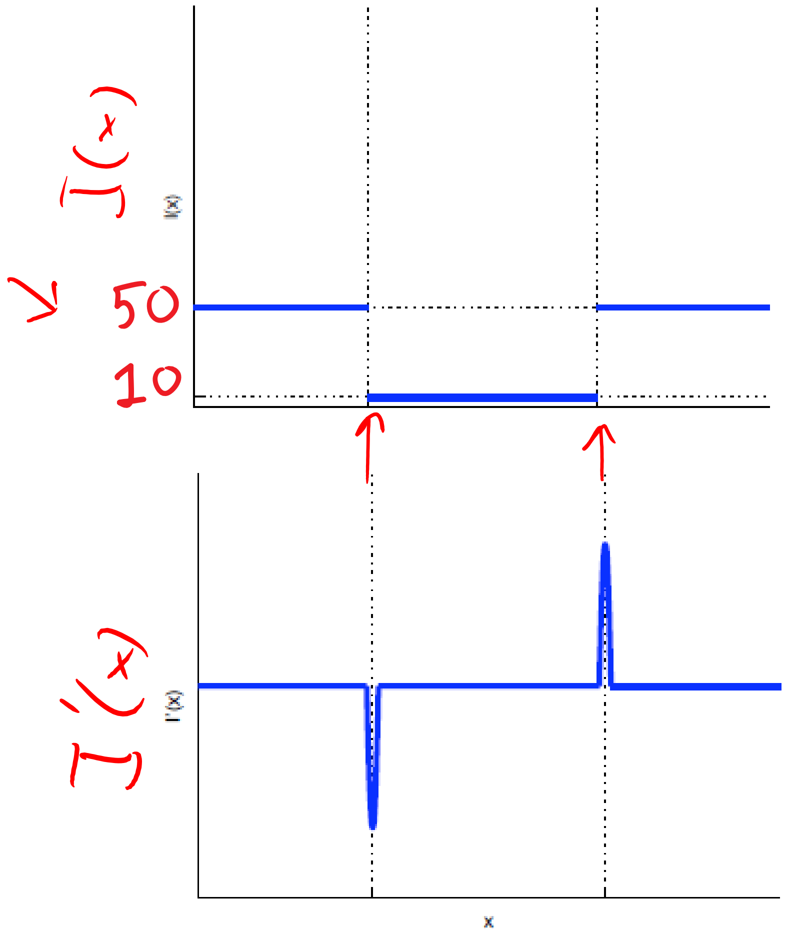
Adelson's checkerboard illusion

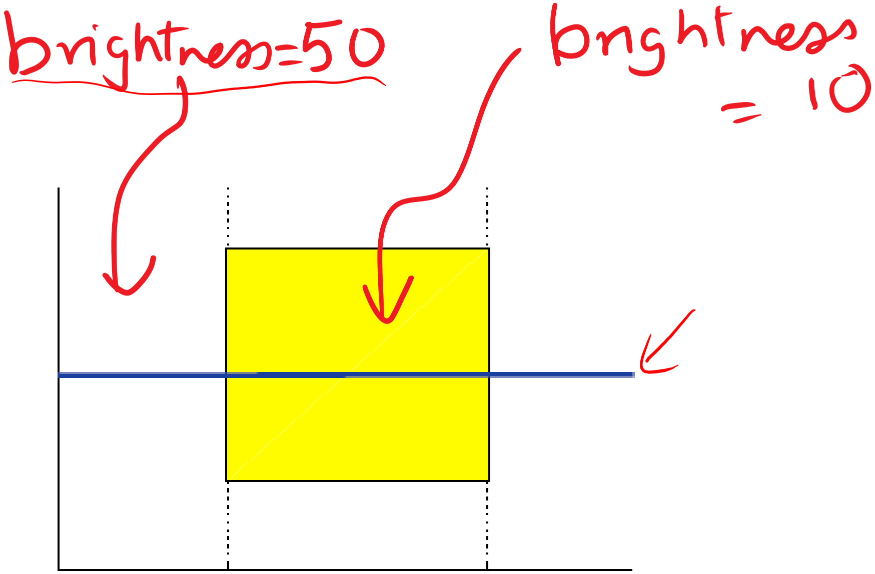
Edges detected in an image



Edge Detection

Consider a one dimensional image, which is a scanline of a two dimensional image:





We can detect edges by differentiation.

Find locations where $|I'(x)|$ is > some threshold.

However…

- Differentiation amplifies noise

$$I(x) + \epsilon \sin(wx)$$

$$I'(x) + \boxed{\epsilon w} \cos(wx)$$

can be large if w is large

- Compensate by Gaussian smoothing

- Both of these are examples of convolution

Edge detection in 1D

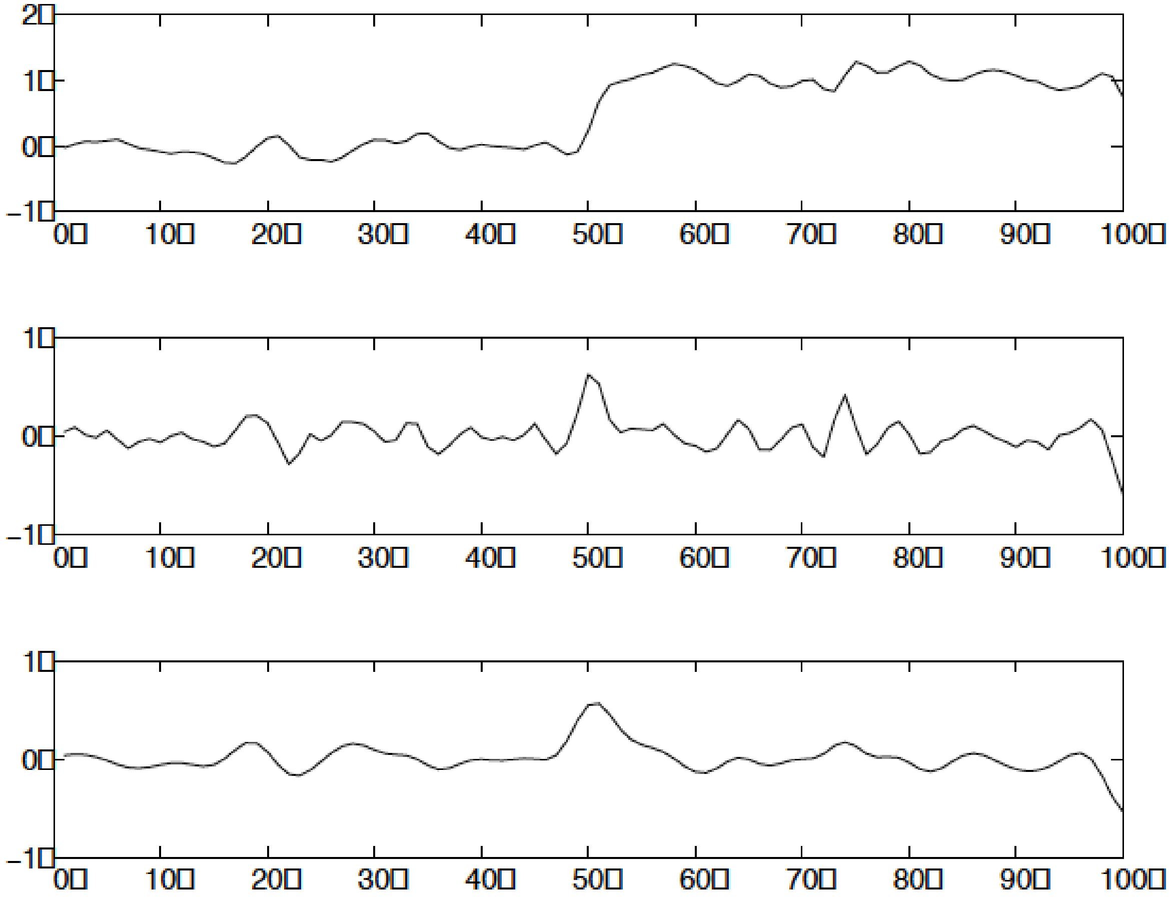
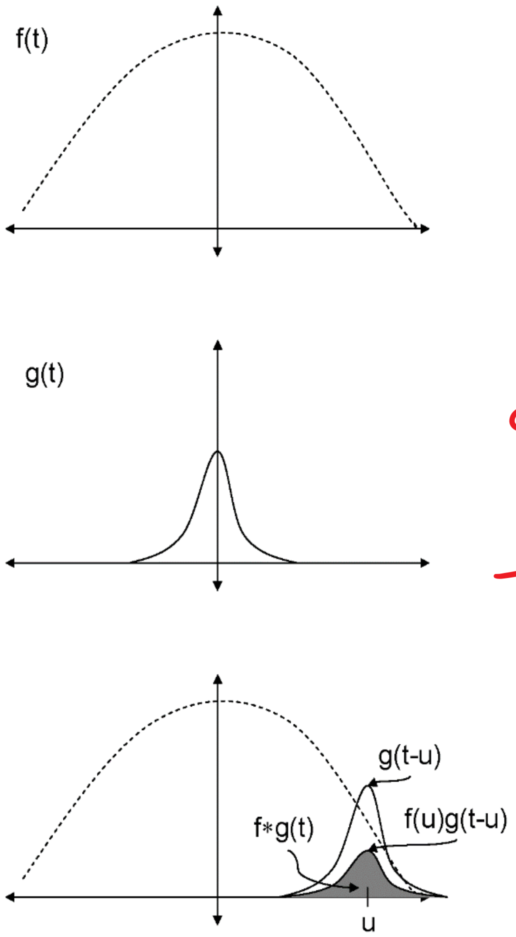


Figure 24.8 Top: Intensity profile $I(x)$ along a one-dimensional section across a step edge. Middle: The derivative of intensity, $I'(x)$. Large values of this function correspond to edges, but the function is noisy. Bottom: The derivative of a smoothed version of the intensity, $(I * G_\sigma)'$, which can be computed in one step as the convolution $I * G'_\sigma$. The noisy candidate edge at $x = 75$ has disappeared.

Convolution

$$f * g(t) = \int_{-\infty}^{\infty} f(u)g(t-u)du$$



$$f * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v) \cdot g(x-u, y-v)\, du\, dv$$

$$(\alpha f_1 + \beta_2 f_2) * g = \alpha f_1 * g + \beta_2 f_2 * g$$

$$f * g = g * f$$

$$f * (g * h) = (f * g) * h$$

Look up on Wikipedia for more...

Implementation Details

- Images are 2D arrays of numbers, so how does one implement the process of computing derivatives, gradients etc?
- The solution: use discrete convolution. In the formula for convolution, replace integral by sum. You can find an exposition in the Wikipedia entry on convolution, also in Wolfram MathWorld

An example

10	20	20	20
10	20	20	20
10	20	20	20
10	20	20	20

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

Result is a new array

Convolution is implemented by "flip and drag". Here let us flip

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

An example

I

$\overset{1}{10}$	$\overset{0}{20}$	$\overset{-1}{20}$	20
10	20	20	20
10	20	20	20
10	20	20	20

multiply pointwise and add

$$1 \times 10 + 0 \times 20 - 1 \times 20$$

$$= -10$$

I_x

	-10		

An example

1 10	0 20	-1 20	20
10	20	20	20
10	20	20	20
10	20	20	20

multiply pointwise and add

$1 \times 10 + 0 \times 20 - 1 \times 20$

$= -10$

	-10		

An example

10	$\overset{1}{20}$	$\overset{0}{20}$	$\overset{-1}{20}$
10	20	20	20
10	20	20	20
10	20	20	20

	-10		

Slide mask & repeat

An example

10	$\overset{1}{20}$	$\overset{0}{20}$	$\overset{-1}{20}$
10	20	20	20
10	20	20	20
10	20	20	20

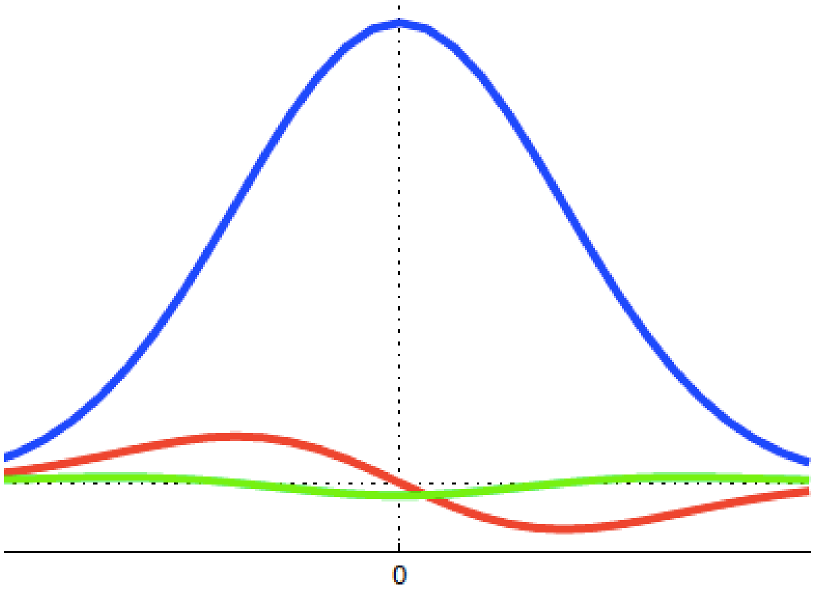
	-10	0	

Slide mask & repeat

$1 \times 20 + 0 \times 20 - 1 \times 20$
$= 0$

and so on..

The 1D Gaussian and its derivatives



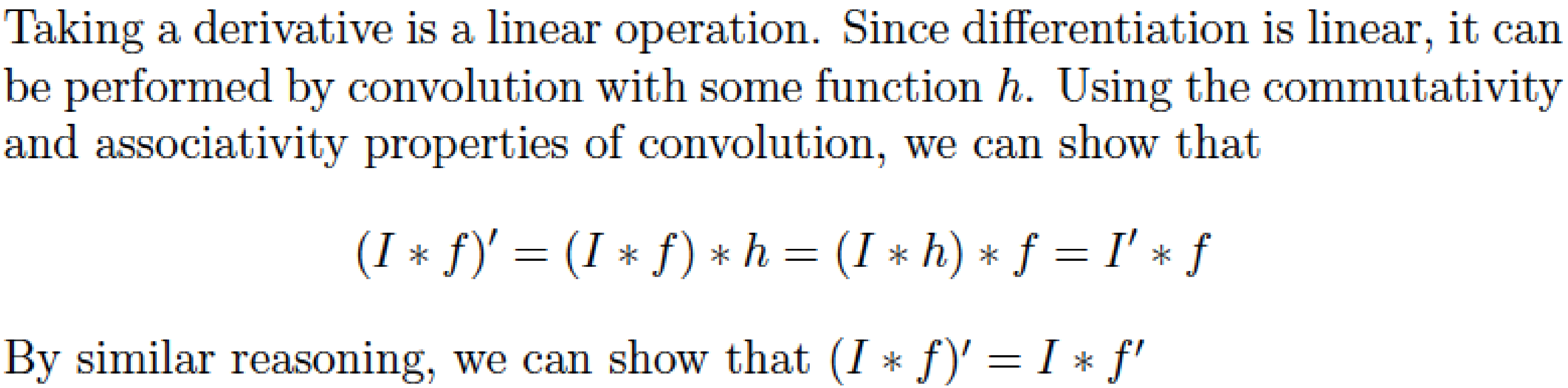
$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

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

$$G''_\sigma(x) = \frac{d^2}{dx^2} G_\sigma(x) = \frac{1}{\sigma^2}\left(\frac{x^2}{\sigma^2} - 1\right) G_\sigma(x)$$

$G'_\sigma(x)$'s maxima/minima occur at $G''_\sigma(x)$'s zeros. And, we can see that $G'_\sigma(x)$ is an odd symmetric function and $G''_\sigma(x)$ is an even symmetric function.

An important observation



Taking a derivative is a linear operation. Since differentiation is linear, it can be performed by convolution with some function h. Using the commutativity and associativity properties of convolution, we can show that

$$(I * f)' = (I * f) * h = (I * h) * f = I' * f$$

By similar reasoning, we can show that $(I * f)' = I * f'$

Edge detection in 1D

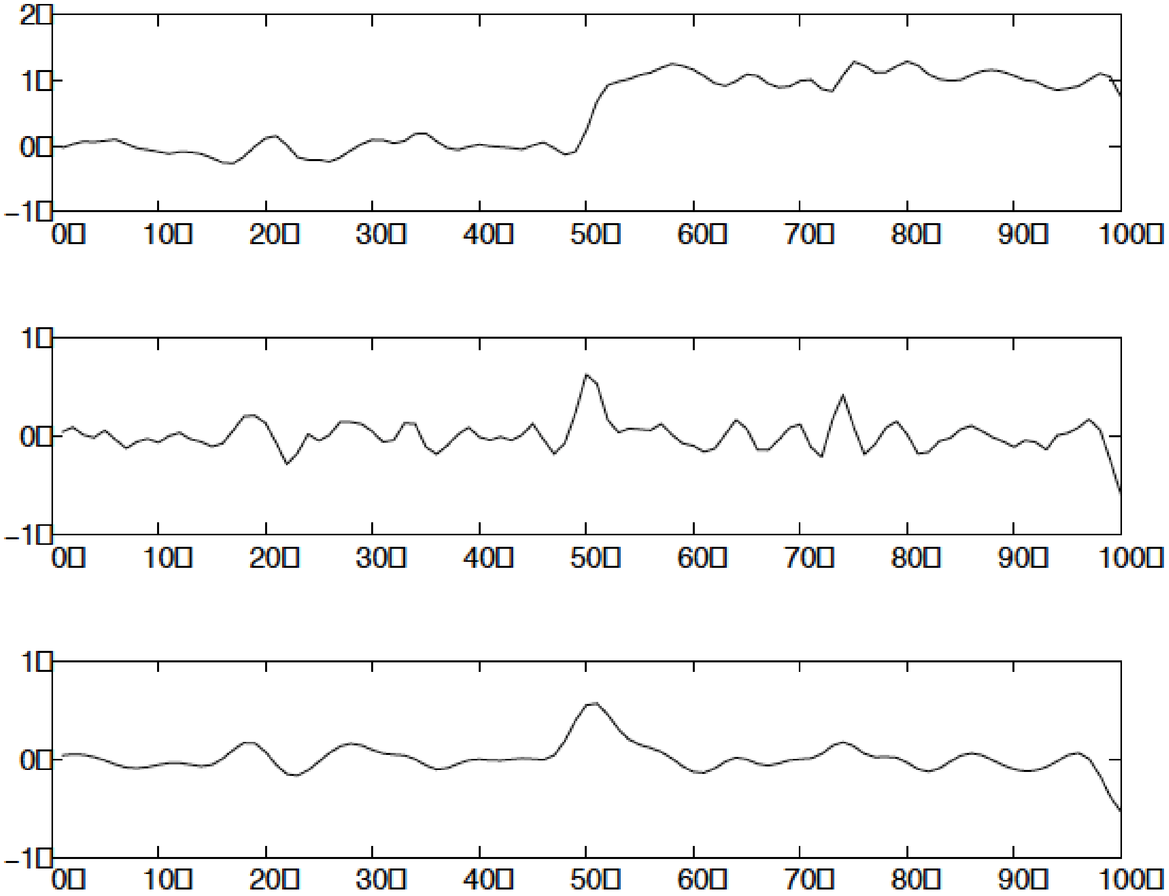
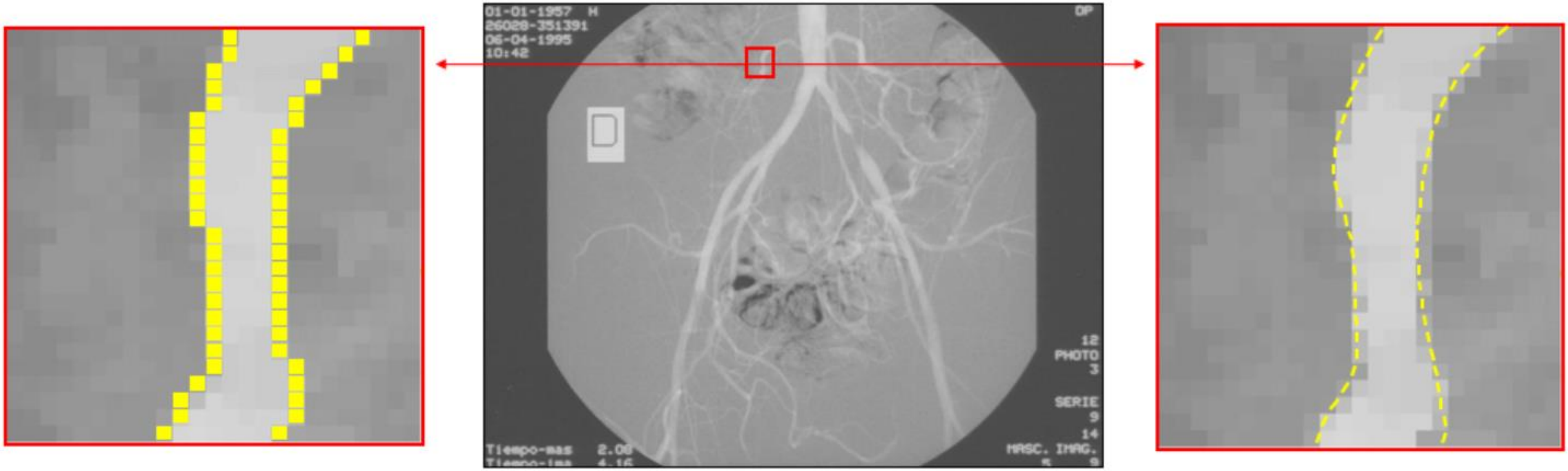


Figure 24.8 Top: Intensity profile $I(x)$ along a one-dimensional section across a step edge. Middle: The derivative of intensity, $I'(x)$. Large values of this function correspond to edges, but the function is noisy. Bottom: The derivative of a smoothed version of the intensity, $(I * G_\sigma)'$, which can be computed in one step as the convolution $I * G'_\sigma$. The noisy candidate edge at $x = 75$ has disappeared.

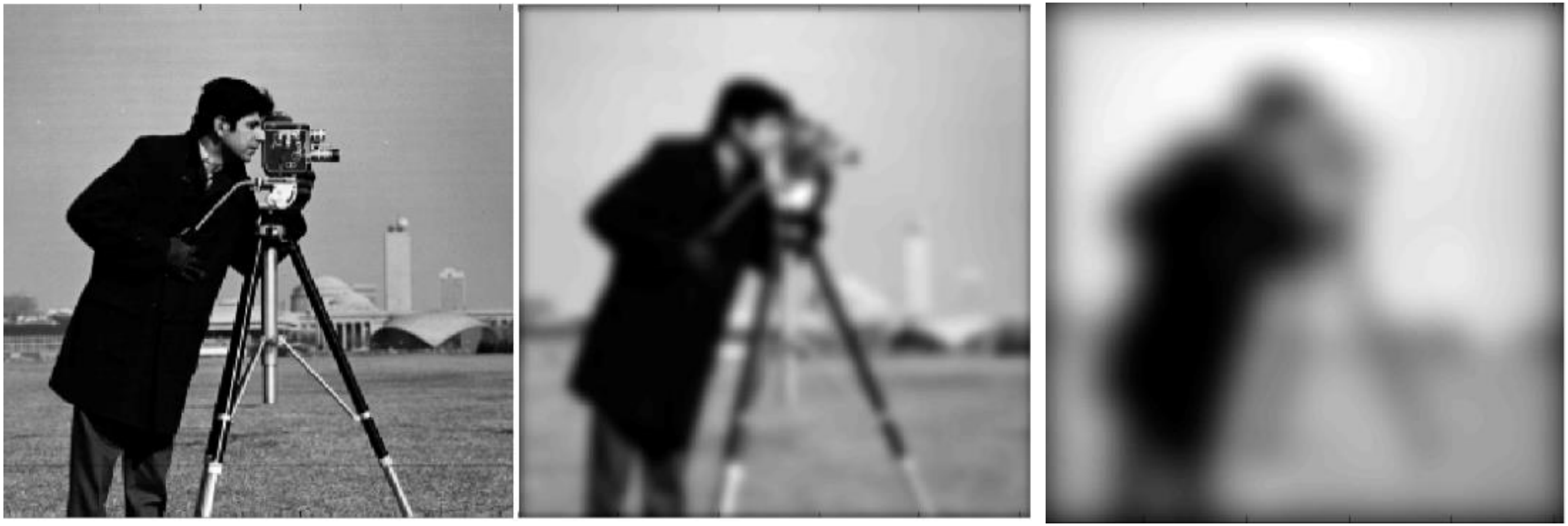

01-01-1957 M
26028-351391
06-04-1995
10:42
DP
D
12
PHOTO
3
SERIE
9
14
MASC. IMAG.
5 9
Tiempo-mas 2.08
Tiempo-ima 4.16

Two Dimensional Gaussian

$$\text{Anisotropic: } G_{\sigma_x,\sigma_y}(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)}$$

$$\text{Isotropic: } G_{\sigma}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}}$$

Image convolved with 2D Gaussian



Oriented Gaussian Derivatives in 2D

$$f_1(x, y) = G'_{\sigma_1}(x) G_{\sigma_2}(y) \tag{10.4}$$

$$f_2(x, y) = G''_{\sigma_1}(x) G_{\sigma_2}(y) \tag{10.5}$$

We also consider rotated versions of these Gaussian derivative functions.

$$Rot_\theta f_1 = G'_{\sigma_1}(u) G_{\sigma_2}(v) \tag{10.6}$$

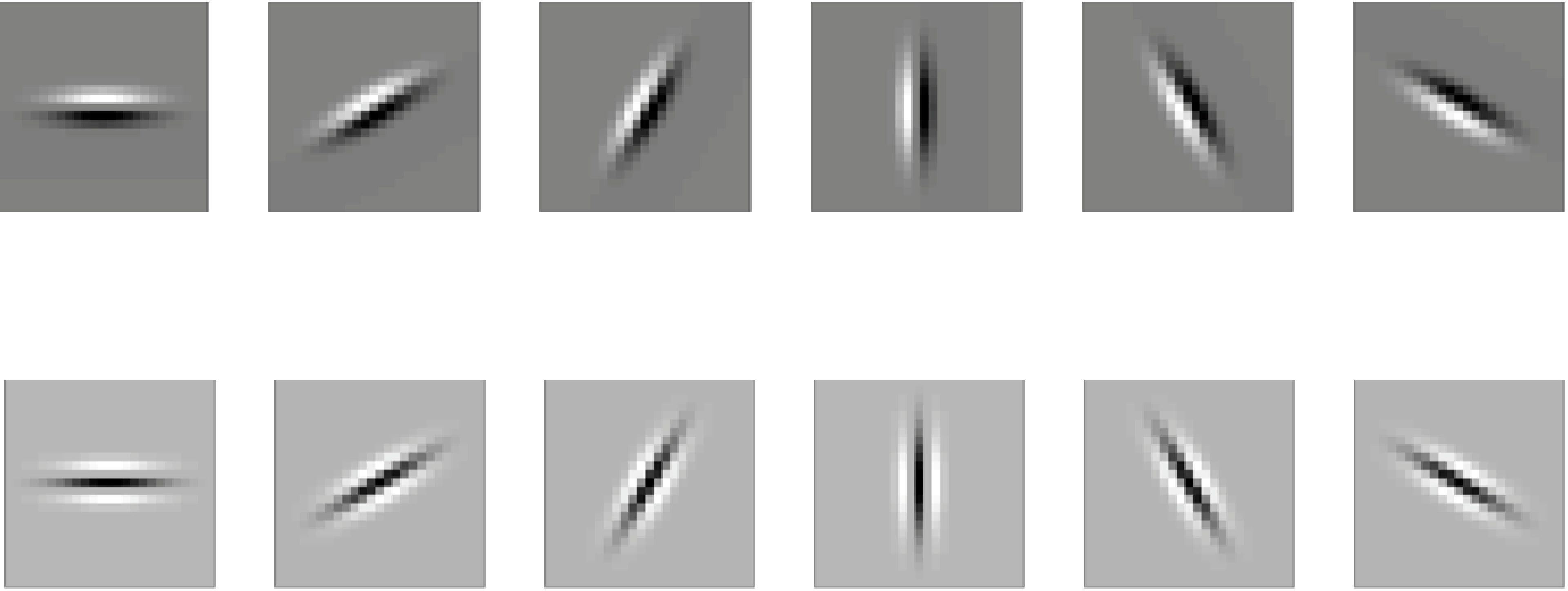
$$Rot_\theta f_2 = G''_{\sigma_1}(u) G_{\sigma_2}(v) \tag{10.7}$$

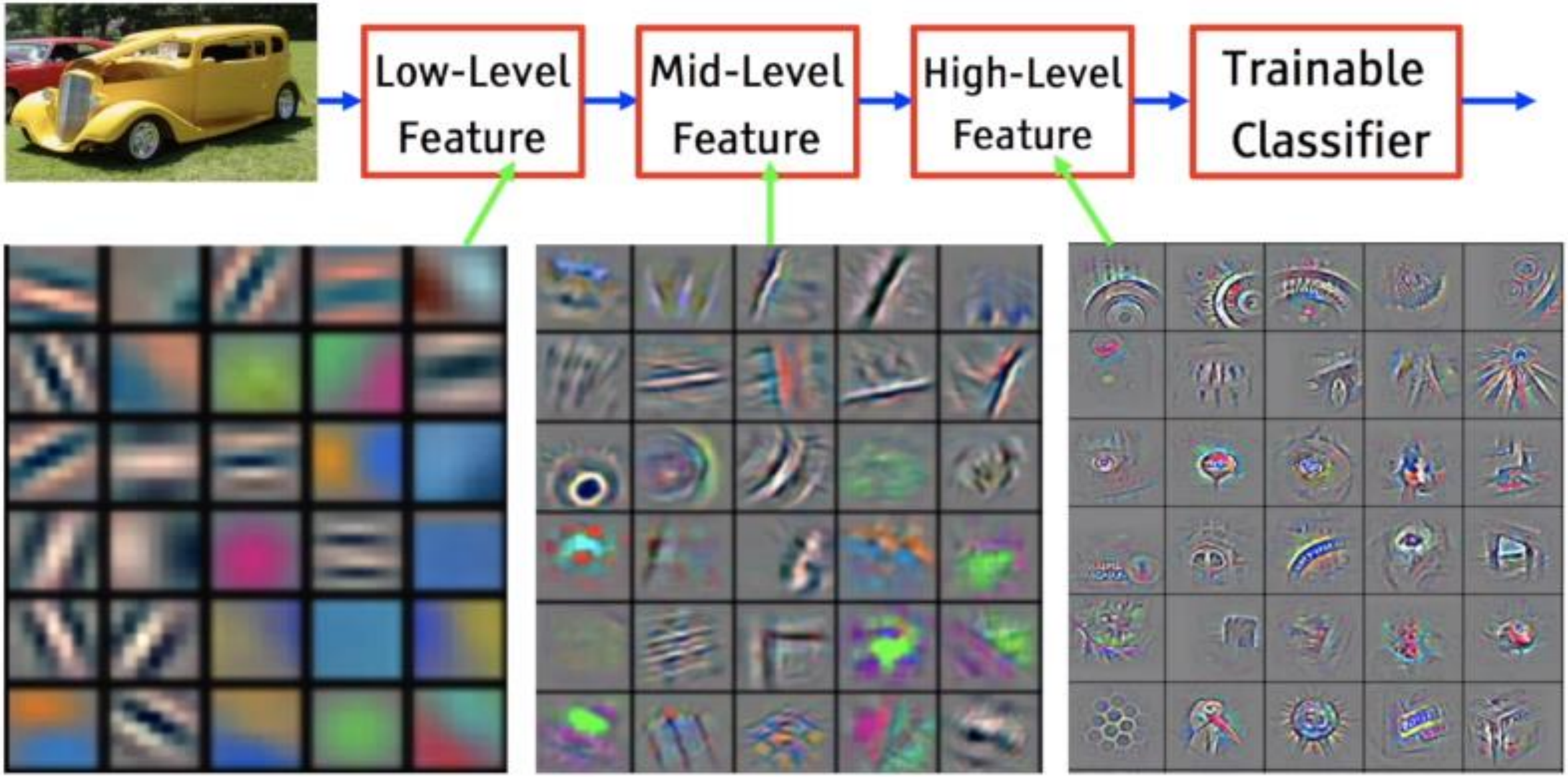
where we set

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

These are useful when we convolve with 2D images, e.g. to detect edges at different orientations.

Oriented Gaussian First and Second Derivatives





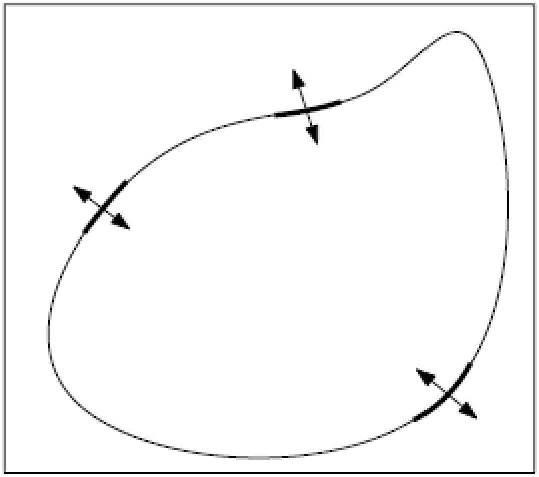
Low-Level Feature

Mid-Level Feature

High-Level Feature

Trainable Classifier

Computing Orientation



$$\nabla I = \begin{pmatrix} I_x \\ I_y \end{pmatrix}$$

can be computed at every pixel

At a horizontal edge, $\nabla I = \begin{pmatrix} k_1 \\ 0 \end{pmatrix}$, and at a vertical edge, $\nabla I = \begin{pmatrix} 0 \\ k_2 \end{pmatrix}$. In general, we have

$$\frac{\nabla I}{\|\nabla I\|} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

This gives us a $\theta = \theta(x, y)$ at every pixel, which defines the edge orientation at that pixel.

If ∇I is null or very close to 0, then the information given by θ is not reliable: we typically use $\|\nabla I\|$ as a confidence measure, and for $\|\nabla I\|$ below some threshold, we do not declare a direction θ.

Typically, we compute $\nabla(I * G_\sigma)$ to handle noise.