CS-465 Computer Vision

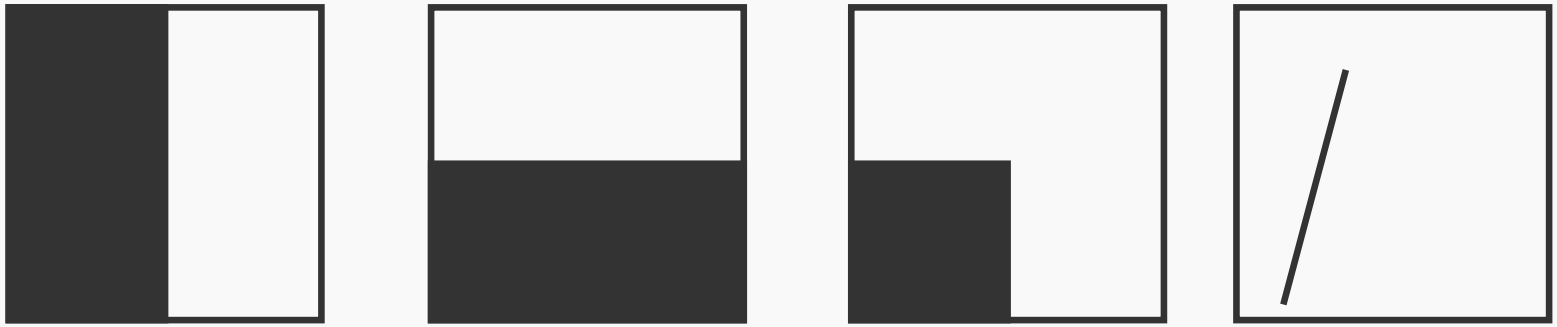
Nazar Khan

PUCIT

4. Edge Detection

Derivatives

- Derivative represents the rate of change.
- Image derivatives represent the rate of color changes in images.
- Interesting features in images (and in the real world) have high derivatives.
- Therefore, derivatives are used for detecting semantically important features such as edges, corners and lines.



Vertical edge Horizontal edge Corner Line

Partial Derivatives and Gradient

- Let $f(x, y)$ be a 2D function.
- *Partial derivative* in x-direction is denoted by $f_x, \partial_x f$, or $\frac{\partial f}{\partial x}$.
- Higher order derivative can be computed in sequence

$$\frac{\partial^2 f}{\partial x \partial y} := \frac{\partial f}{\partial x}\left(\frac{\partial f}{\partial y}\right)$$

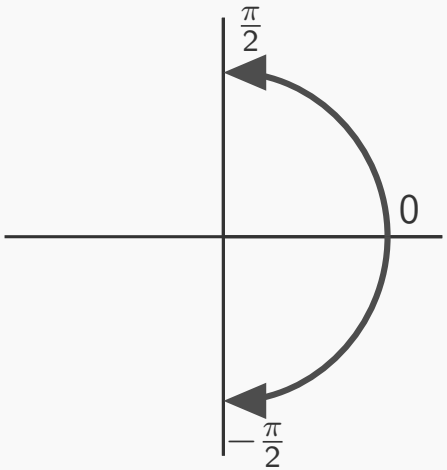
- Order of partial differentiation does not matter (under suitable smoothness assumptions)

$$f_{xy} = f_{yx}$$

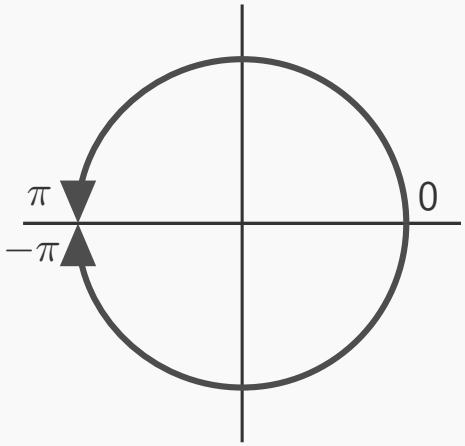
- The *gradient vector* $\nabla f = (f_x, f_y)^T$ always points in the direction of highest rate of change.
- *Gradient magnitude* $|\nabla f| = \sqrt{f_x^2 + f_y^2}$ is rotationally invariant.
- *Gradient direction* can be computed as $\theta = \arctan\left(\frac{f_y}{f_x}\right)$.

atan vs. atan2

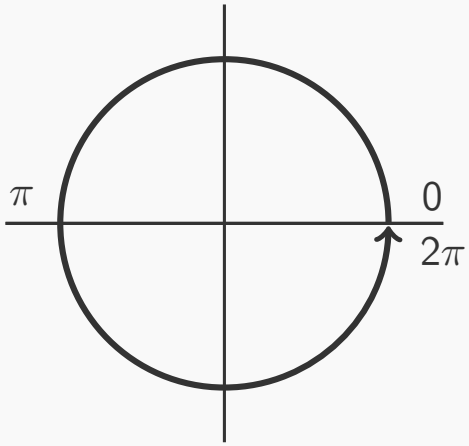
- The atan function returns angle in the range $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$ while atan2 returns angle in the range $\theta \in (-\pi, \pi)$.
  - Function atan does not differentiate between quadrants 1 and 3. Similarly for quadrants 2 and 4.
  - Function atan2 differentiates between all quadrants.
  - For example, $\text{atan}(\frac{1}{1}) = \text{atan}(\frac{-1}{-1}) = \text{atan2}(1,1) \neq \text{atan2}(-1,-1)$



$\theta = \text{atan}(\frac{y}{x})$



$\theta = \text{atan2}(y, x)$



$$\theta = \begin{cases} \text{atan2}(y,x) & \text{if } y \geq 0 \\ 2\pi + \text{atan2}(y,x) & \text{if } y < 0 \end{cases}$$

Numerical Approximation of Derivative

- Using 2nd order Taylor's expansion

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \quad (1)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \quad (2)$$

- Subtracting (2) from (1) and solving for $f'(x)$ yields

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (3)$$

- Adding (2) and (1) and solving for $f''(x)$ gives

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{2h^2} + O(h) \quad (4)$$

- Equations (3) and (4) are 1st and 2nd derivative approximations using *central differences*.

Numerical Approximation of Derivative

Take-home Quiz 2:

1. Prove that using a 1st order Taylor's expansion for $f(x+h)$ yields

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \tag{5}$$

This is a derivative approximation using *forward difference*.

2. Prove that using a 1st order Taylor's expansion for $f(x-h)$ yields

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \tag{6}$$

This is a derivative approximation using *backward difference*.

Derivative approximations using central differences are more accurate since they employ 2nd order Taylor approximations. Higher than order 2 approximations will be even better but computationally more expensive.

Derivative Filters

- In images, colours are stored at every pixel. Therefore, h is (almost) always taken to be 1.
- Commonly used approximate derivative filters via convolution are

Approximation	Mask	Difference
$f'(x) = f(x+1) - f(x)$	$\begin{array}{\|c\|c\|}\hline 1 & \mathbf{-1} \\ \hline \end{array}$	Forward
$f'(x) = f(x) - f(x-1)$	$\begin{array}{\|c\|c\|}\hline \mathbf{1} & -1 \\ \hline \end{array}$	Backward
$f'(x) = \frac{f(x+1)-f(x-1)}{2}$	$\frac{1}{2}\begin{array}{\|c\|c\|c\|}\hline 1 & \mathbf{0} & -1 \\ \hline \end{array}$	Central
$f''(x) = \frac{f(x+1)-2f(x)+f(x-1)}{2}$	$\frac{1}{2}\begin{array}{\|c\|c\|c\|}\hline 1 & \mathbf{-2} & 1 \\ \hline \end{array}$	Central

where each mask will be flipped during convolution and the origin (in bold blue) will be placed over location x.

Derivative Filters

- Derivative filters are very important examples of linear shift invariant (LSI) filters.
- 1D filters can be applied on 2D images.

x-direction

$$\frac{1}{2}\begin{bmatrix} 1 \\ \mathbf{0} \\ -1 \end{bmatrix}$$

y-direction

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

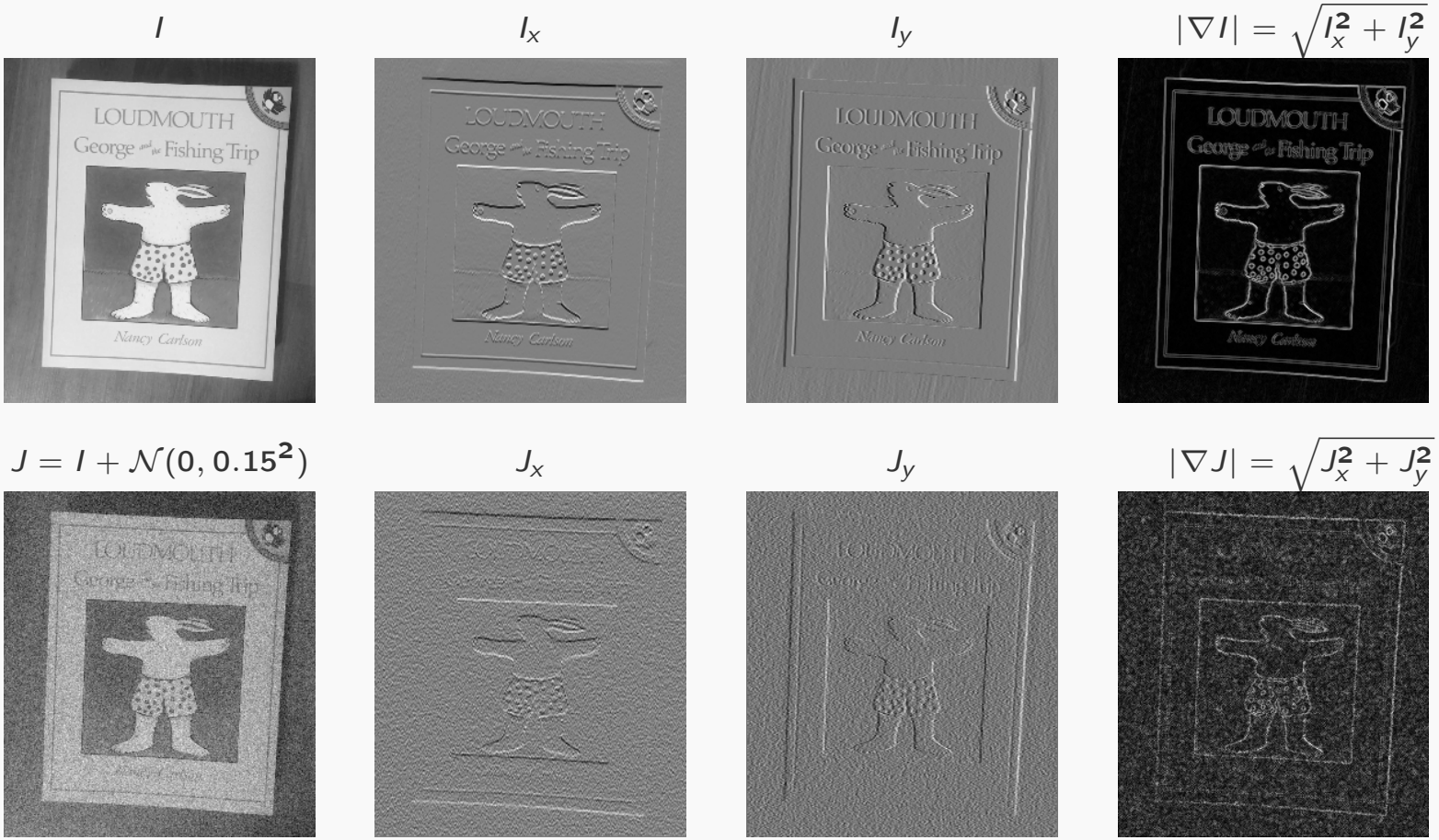


Figure: Derivative filters are sensitive to noise. Convolving with derivative filters yields high responses on edges as well as noise. Author: N. Khan (2018)

Smoothing before Derivative Filtering

- Noise affects *every* signal in the real world.
- If there is noise in the signal, its effect on the derivative will be amplified. For example, derivative computed from two noisy pixels can have twice the noise.
- Always smooth a signal before computing derivatives.
- This requires two convolutions – one for smoothing and one for derivatives.
- We can exploit associativity of convolution

$$(I * M_1) * M_2 = I * (M_1 * M_2)$$

to obtain a cheaper solution.

  - Apply the small derivative kernel on the small smoothing kernel. Then apply resulting kernel on the large image.

Derivative of Gaussian Filter

- For the case of Gaussian smoothing followed by derivative filtering, we can instead differentiate the smoothing kernel and then convolve with the result.
- This yields the *Derivative of Gaussian* filter.

$$1D: \quad \frac{\partial g_\sigma(x)}{\partial x} = \frac{-x}{\sqrt{2\pi}\sigma^3} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

$$2D: \quad \frac{\partial G_\sigma(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

  where all means have been set to 0.
- This filter computes gradients at smoothing scale σ.
- To make a 5×1 mask, evaluate the $1D$ Derivative of Gaussian at $x = -2, -1, 0, 1, 2$.

Python Example

- The python function `scipy.ndimage.filters.gaussian_filter` in the `scipy` library can perform filtering with
  1. Gaussian
  2. 1st-derivative of Gaussian
  3. 2nd-derivative of Gaussian
  4. 3rd-derivative of Gaussian

  with different boundary handling mechanisms.

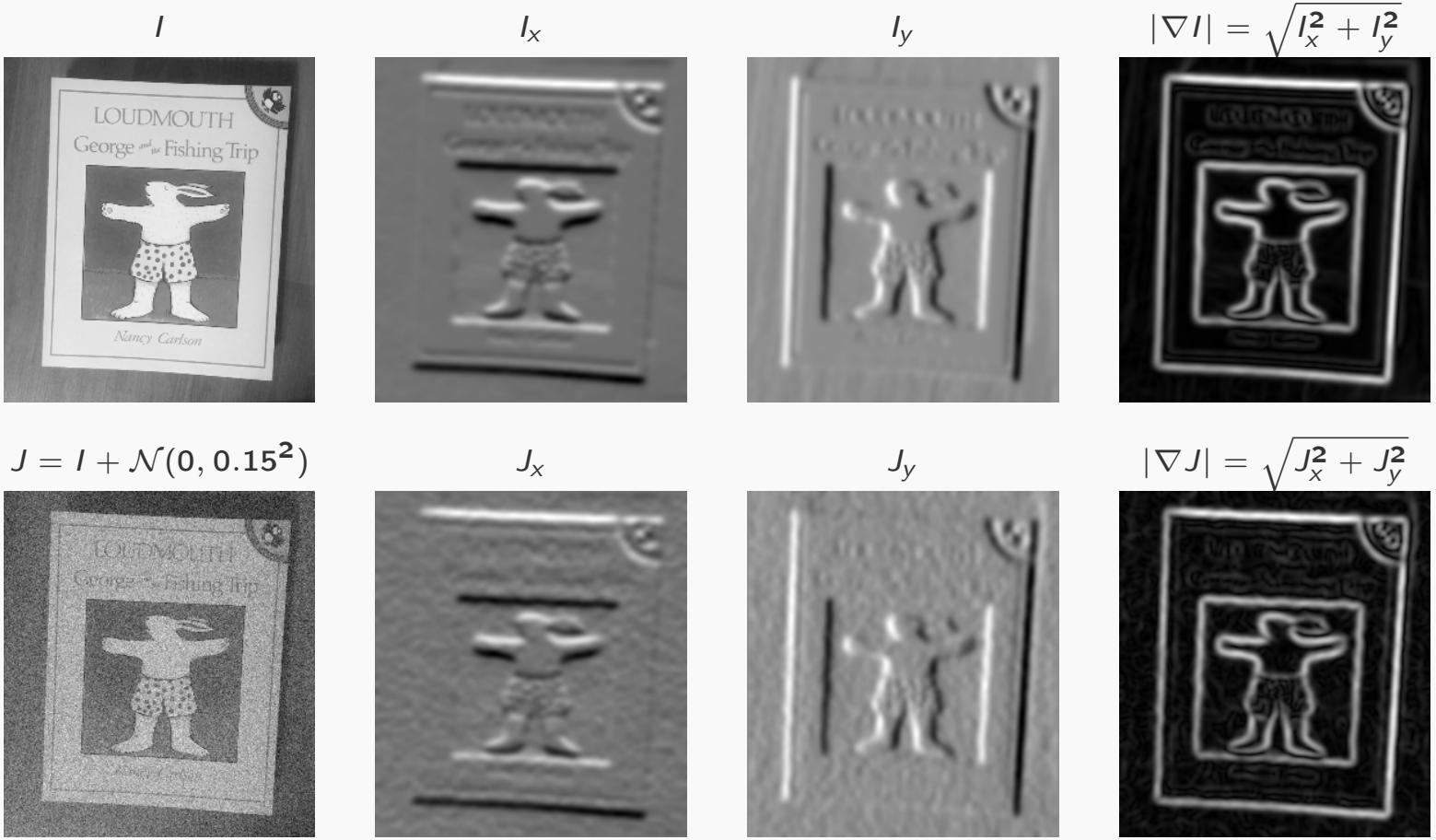


Figure: Convolving with derivative of Gaussian smooths out noise and computes derivatives. Results without noise are similar to results with noise. This demonstrates robustness to noise. Smoothing scale for all results was $\sigma = 3$. Author: N. Khan (2018)

Sobel Filter

- *Sobel operator* is a 2D filter that applies an integer valued smoothing filter before applying a derivative filter.
- Once again, the filters can be convolved with each other first to obtain a single derivative filter that is robust to noise.

x-direction

$$\frac{1}{2}\begin{bmatrix}1\\ \mathbf{0}\\ -1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1 & \mathbf{2} & 1\end{bmatrix} = \frac{1}{8}\begin{bmatrix}1 & 2 & 1\\ 0 & \mathbf{0} & 0\\ -1 & -2 & -1\end{bmatrix}$$

y-direction

$$\frac{1}{2}\begin{bmatrix}1 & \mathbf{0} & -1\end{bmatrix} * \frac{1}{4}\begin{bmatrix}1\\ \mathbf{2}\\ 1\end{bmatrix} = \frac{1}{8}\begin{bmatrix}1 & 0 & -1\\ 2 & \mathbf{0} & -2\\ 1 & 0 & -1\end{bmatrix}$$

- This is an approximation of the Derivative of Gaussian filter but with coarser control over the smoothing scale σ.

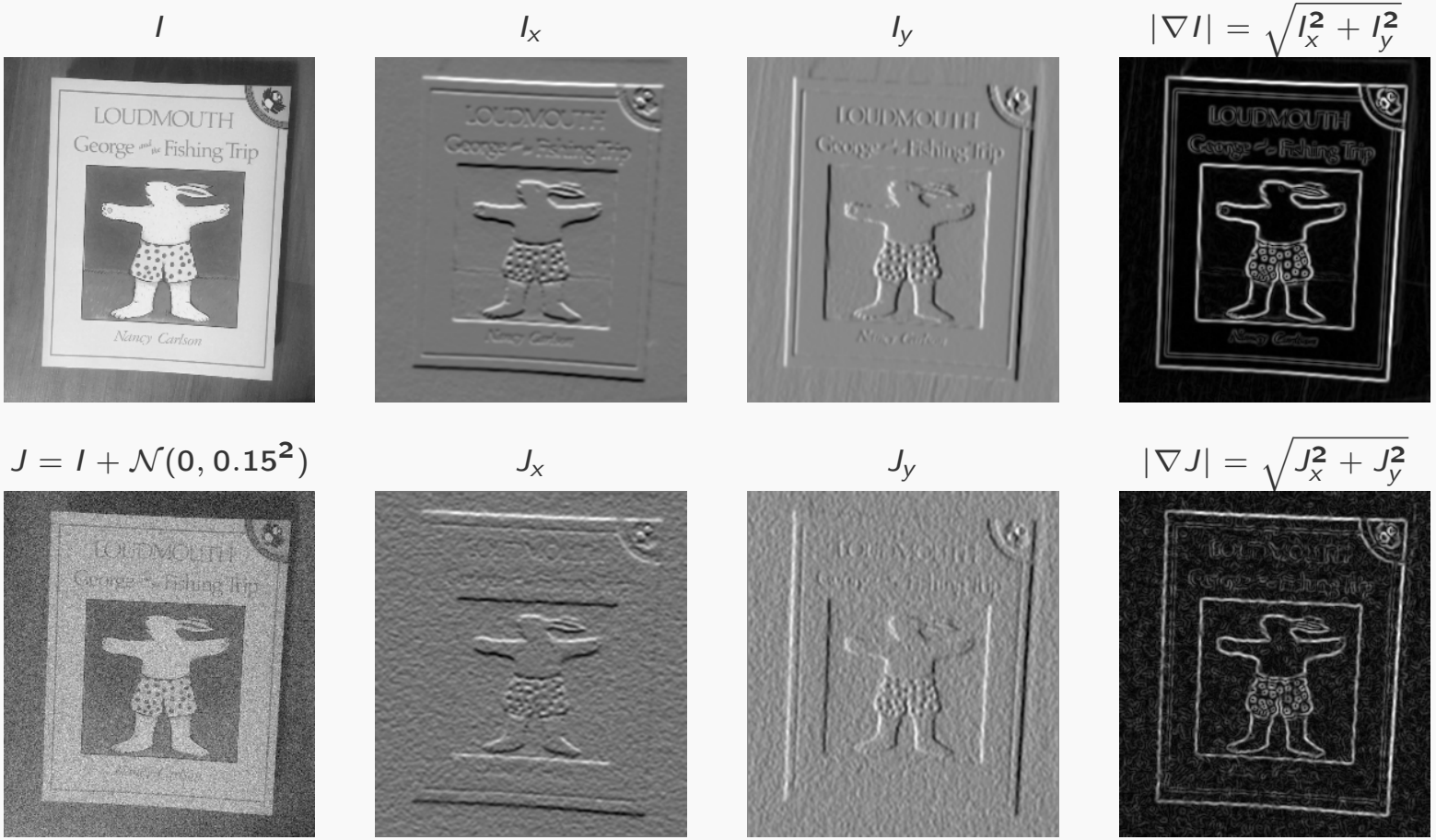


Figure: Convolving with Sobel kernels smooths out noise and computes derivatives. Results without noise are similar to results with noise. This demonstrates robustness to noise. Sobel kernel size for all results was 7×7. Author: N. Khan (2018)

A Simple Edge Detector

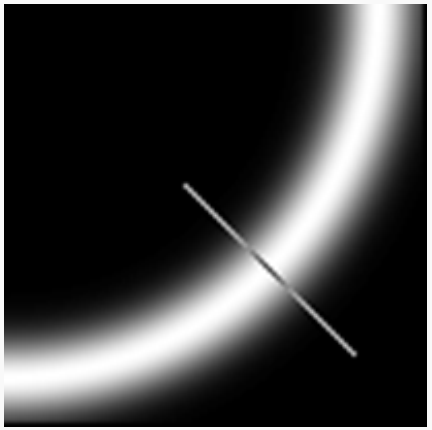
- Threshold gradient magnitude to obtain edge pixels.
- Percentiles can be used instead of fixed threshold.
  - The k-th percentile of a list of numbers is the value greater than $k\%$ of the numbers.
  - For example, if 70% values in a list of numbers are less than 48, then 48 is the 70-th percentile.
- For a threshold of, say, the 80-th percentile, 20% of the image pixels will be guaranteed to be marked as edge pixels.

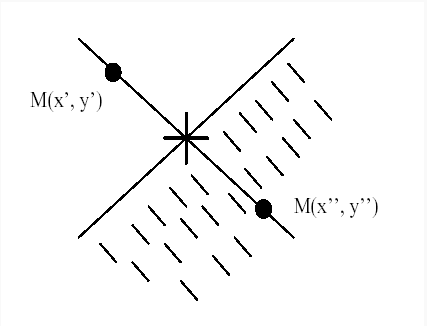
Canny Edge Detector

- Most popular edge detector.
- Three step procedure
  1. Compute derivatives of smoothed image
  2. Non-maxima supression
  3. Hysteresis thresholding

Non-maxima Supression

- Derivative magnitudes start increasing close to an edge and decrease gradually after crossing it.
- So pixels close to an edge pixel can also exceed the detection threshold and be detected as edges.
- Solution: The actual edge should have the highest gradient magnitude. So suppress the neighbours *across the edge*.





Non-maxima Supression

Quantization of Gradient Direction

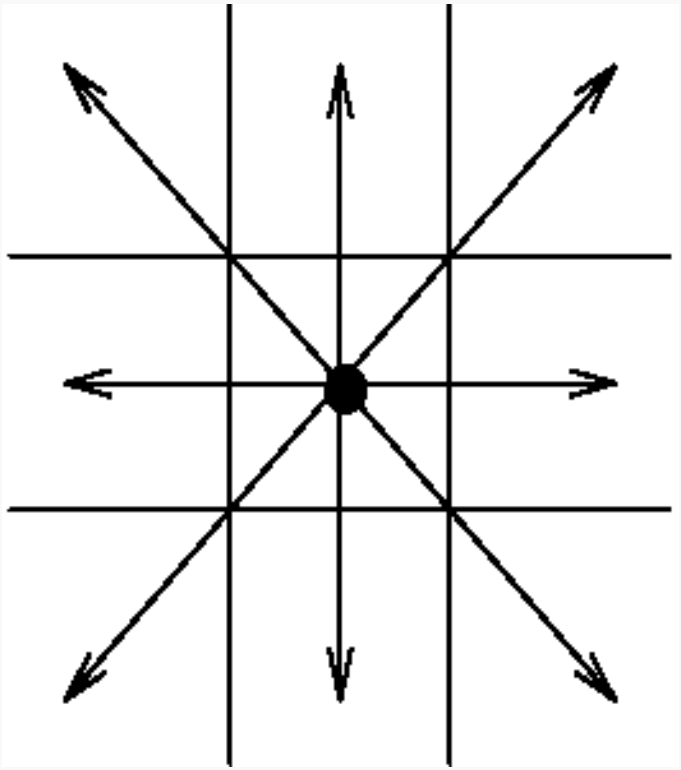


Figure: Gradient direction needs to be quantized into 1 of 4 directions in order to select neighbouring pixels across the edge.

Hysteresis Thresholding

- Basic Idea: If a weak edge lies adjacent to a strong edge, include the weak edge too.
- Use pixels above upper threshold T_{high} as seed points for relevant edges.
- Recursively add all neighbours of seed points that are above the lower threshold T_{low}.

Hysteresis Thresholding

1. Scan image from left to right, top to bottom.
2. If $|\nabla I|(x, y)$ is above T_{high}
   - **2.1** Mark pixel (x, y) as edge.
   - **2.2** Look at each unvisited neighbour of pixel (x, y). If gradient magnitude of neighbour is above T_{low}, mark it as edge and *recursively* look at unvisited neighbours of (x, y).

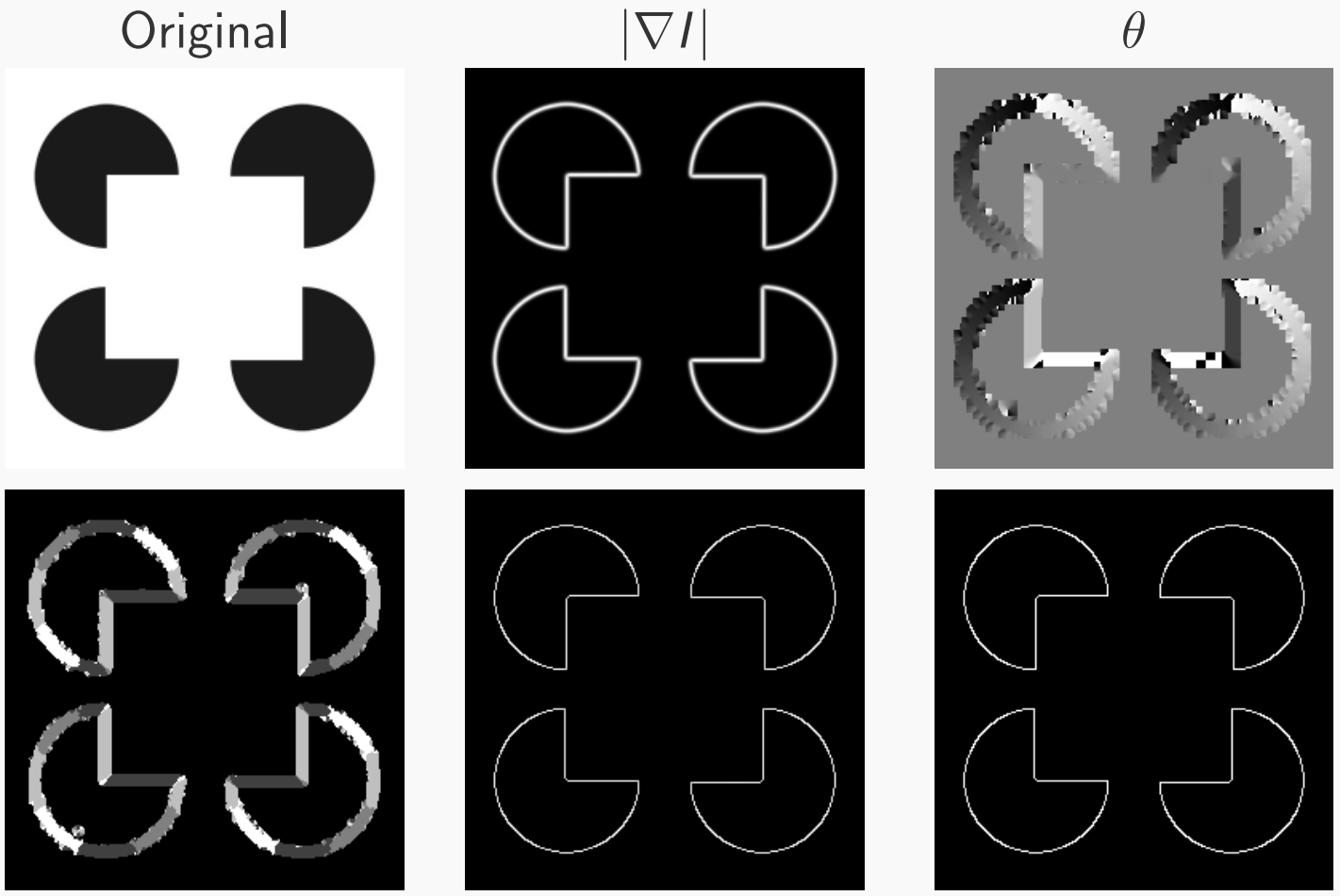


Figure: Canny edge detection pipeline. Gaussian derivatives were computed at smoothing scale $\sigma = 1$. Author: N. Khan (2018)

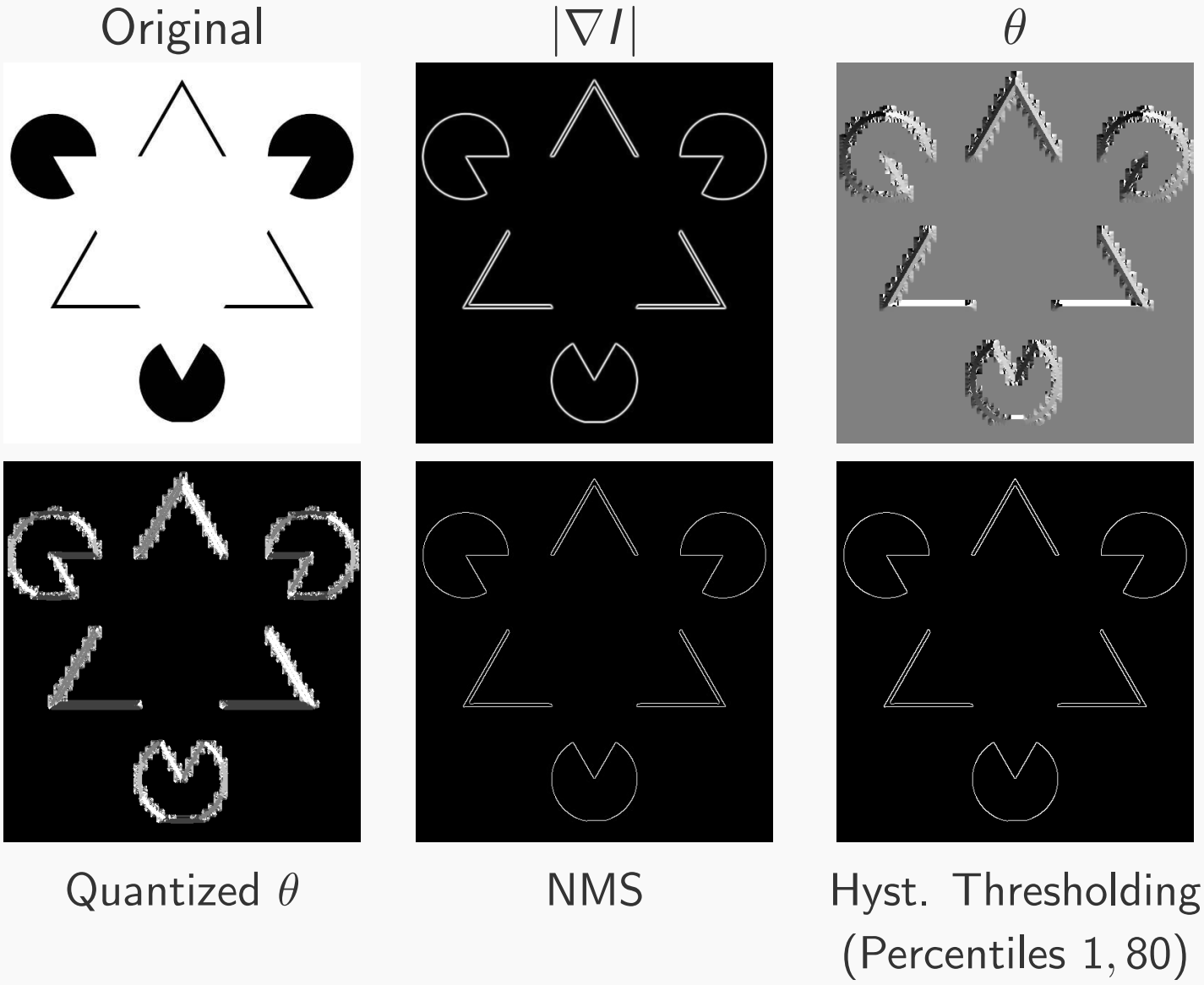


Figure: Canny edge detection pipeline. Gaussian derivatives were computed at smoothing scale $\sigma = 1$. Author: N. Khan (2018)

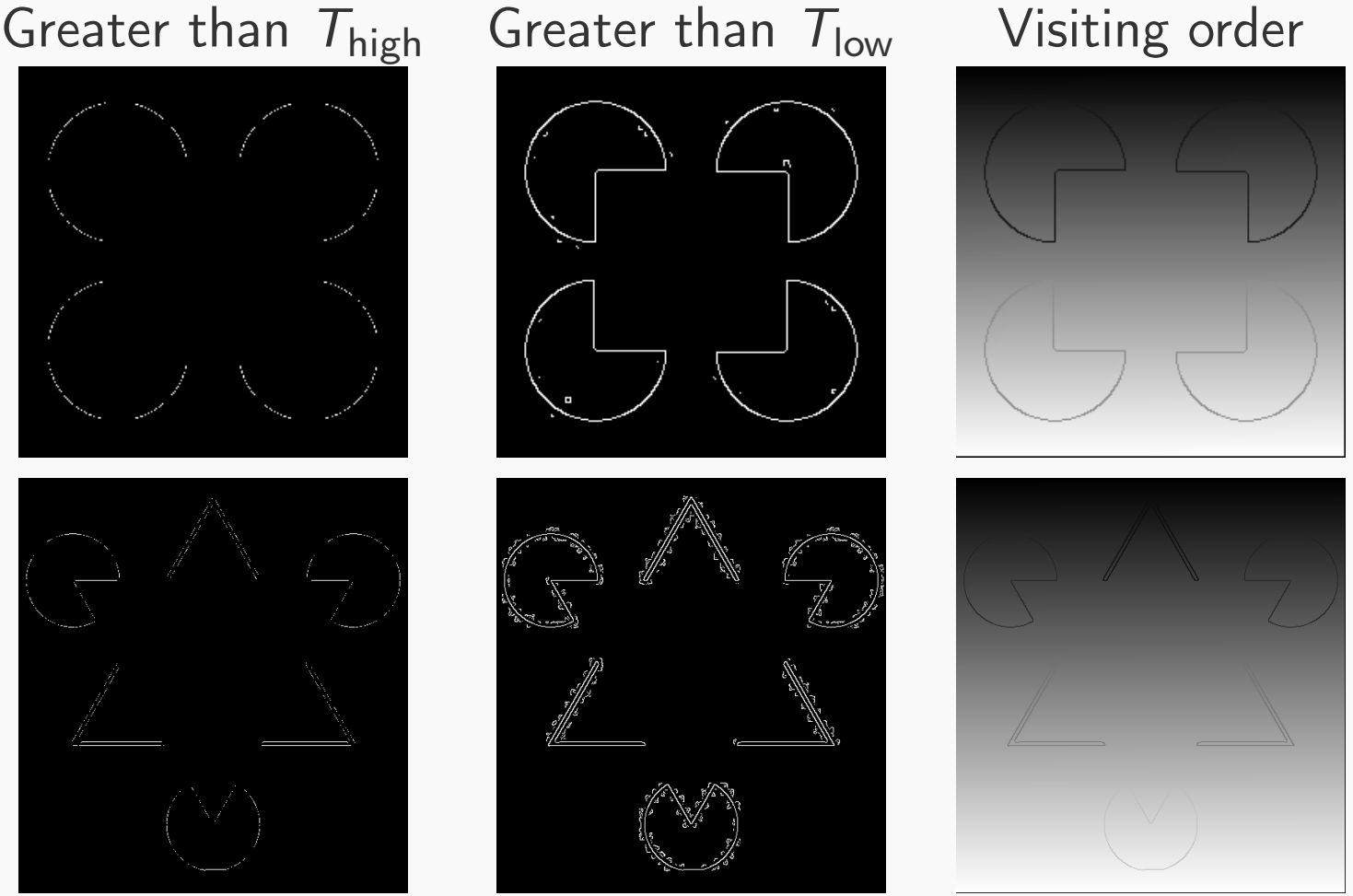


Figure: Neither the high threshold alone (**left**) nor the low threshold alone (**middle**) is sufficient for good edge detection. **Right**: Order of visiting pixels during hysteresis thresholding. Darker corresponds to pixels visited earlier. Author: N. Khan (2018)

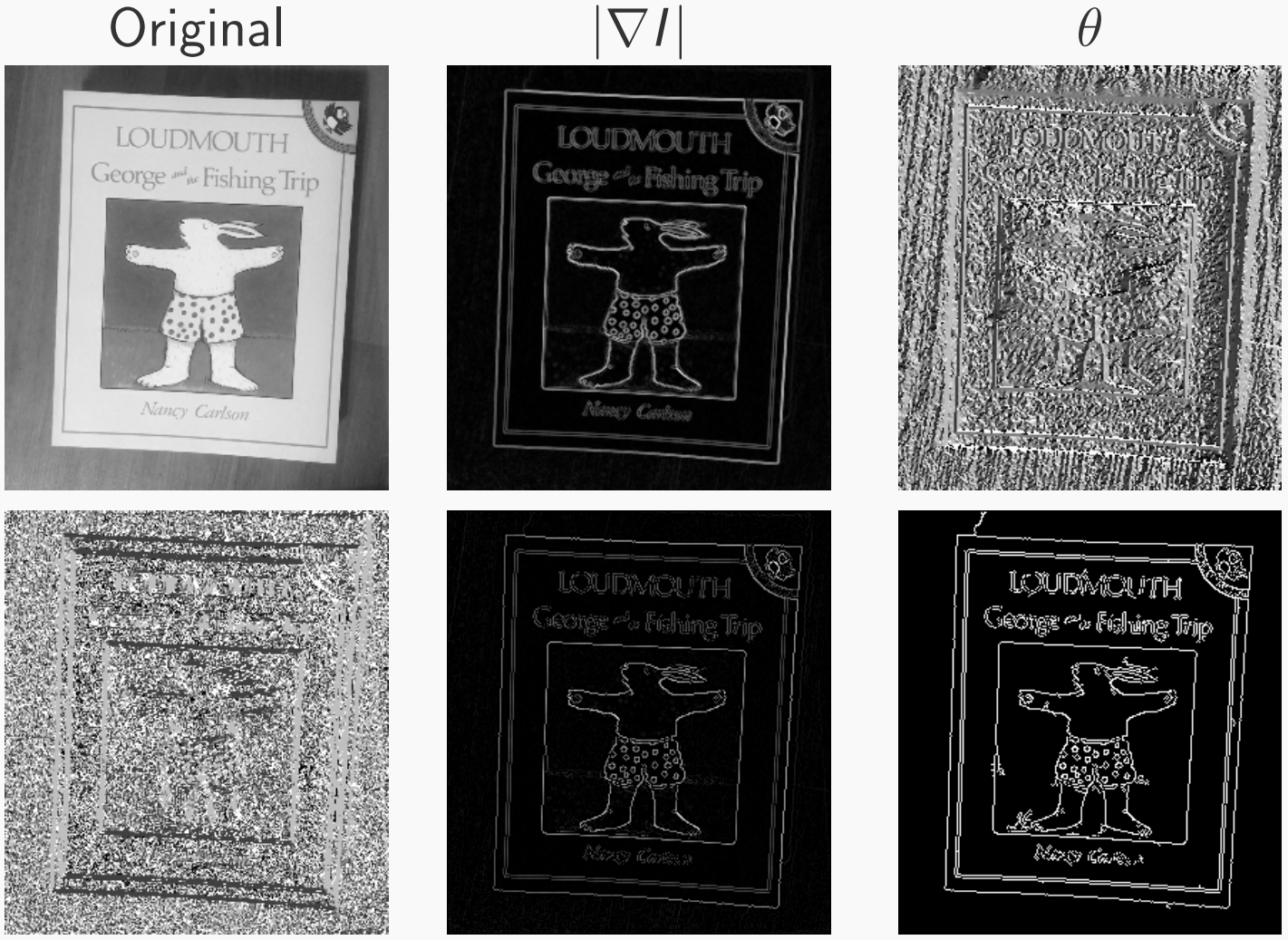


Figure: Canny edge detection pipeline. Gaussian derivatives were computed at smoothing scale $\sigma = 0.2$. Author: N. Khan (2018)

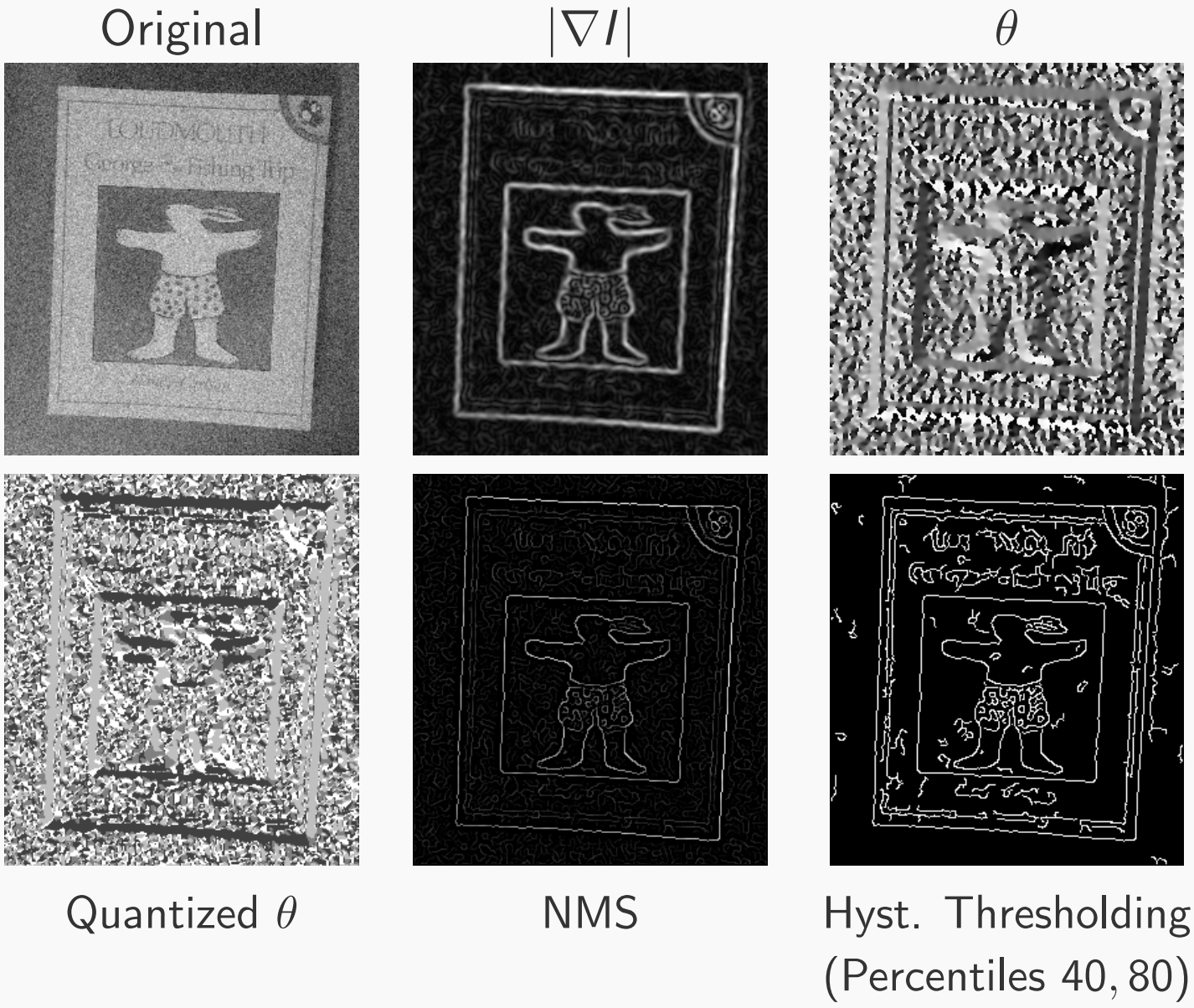


Figure: Canny edge detection on a noisy image. Gaussian derivatives were computed at smoothing scale $\sigma = 1.8$. Author: N. Khan (2018)

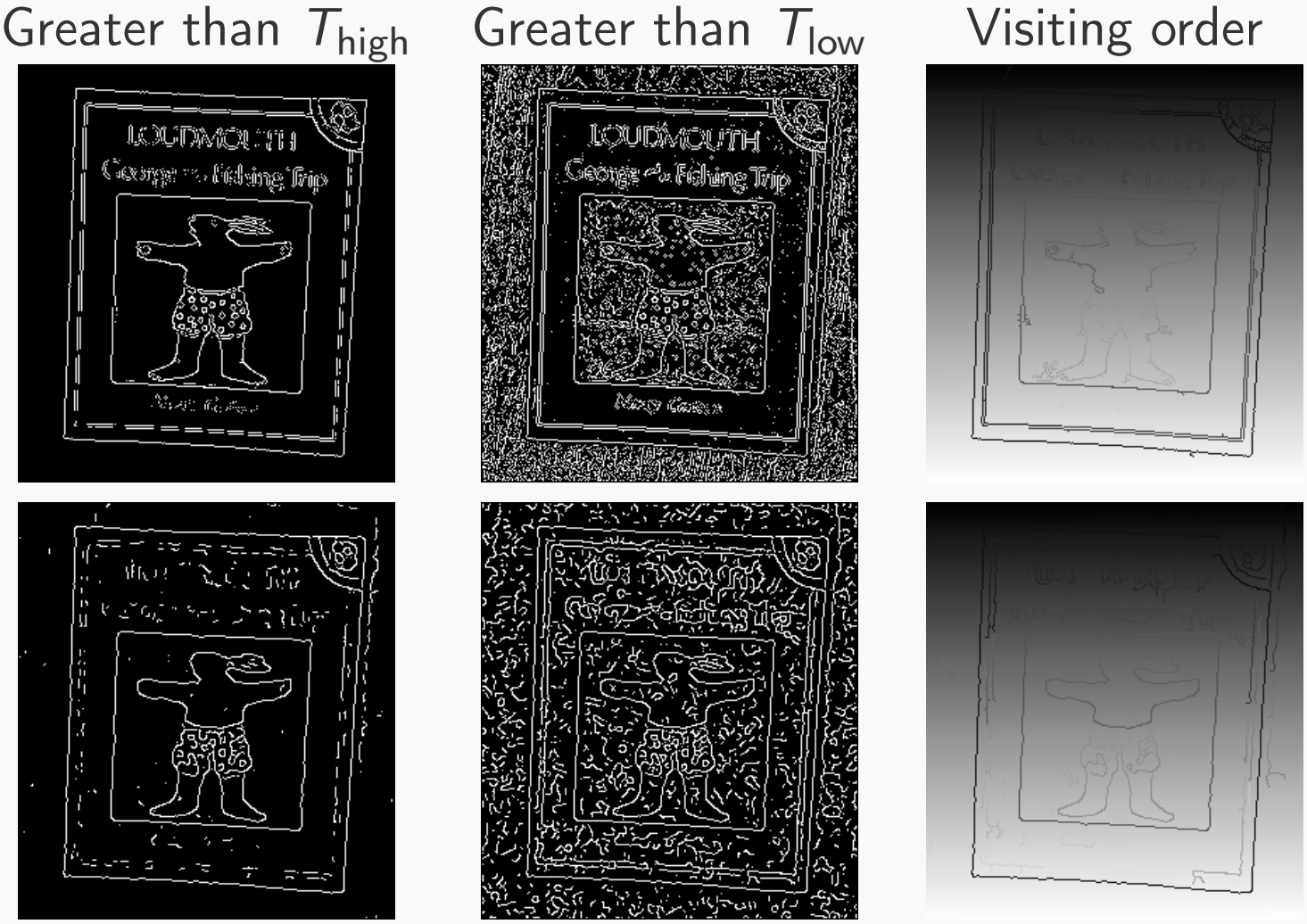
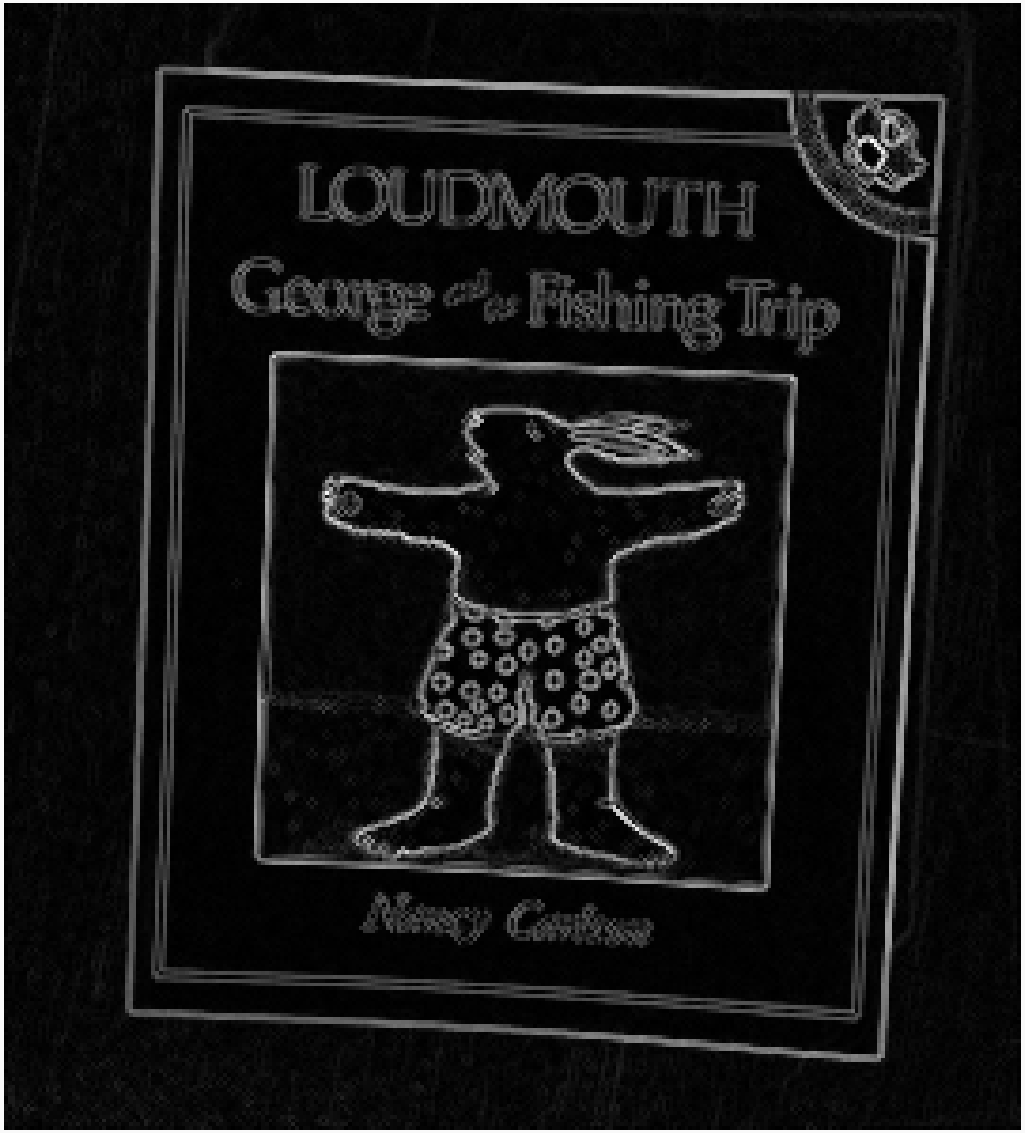


Figure: Neither the high threshold alone (**left**) nor the low threshold alone (**middle**) is sufficient for good edge detection. **Right**: Order of visiting pixels during hysteresis thresholding. Darker corresponds to pixels visited earlier. **Bottom** row is for the noisy image. Author: N. Khan (2018)


LOUDMOUTH
George and the Fishing Trip