Vision Review: Image Processing

Course web page:

www.cis.udel.edu/~cer/arv

University of Delaware

Announcements

- Homework and paper presentation guidelines are up on web page
- Readings for next Tuesday: Chapters 6, 11.1, and 18
- For next Thursday: “Stochastic Road Shape Estimation”

Computer Vision
Review Outline

- Image formation
- **Image processing**
- Motion & Estimation
- Classification

Outline

- Images
- Binary operators
- Filtering
  - Smoothing
  - Edge, corner detection
- Modeling, matching
- Scale space

Images

- An image is a matrix of pixels $\mathbf{I}(x, y)$
  Note: Matlab uses $\mathbf{I}(r, c)$
- Resolution
  - Digital cameras: 1600 X 1200 at a minimum
  - Video cameras: ~640 X 480
- Grayscale: generally 8 bits per pixel $\rightarrow$ Intensities in range [0…255]
- RGB color: 3 8-bit color planes $\mathbf{I}_R, \mathbf{I}_G, \mathbf{I}_B$

UNIVERSITY OF DELAWARE

Image Conversion

- RGB $\rightarrow$ Grayscale: Mean color value, or weight by perceptual importance (Matlab: `rgb2gray`)

- Grayscale $\rightarrow$ Binary: Choose threshold based on histogram of image intensities (Matlab: `imhist`)

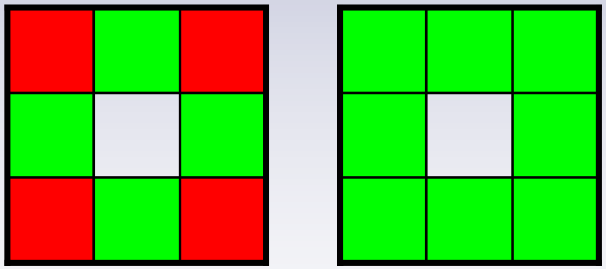
Color Representation

- RGB, HSV (hue, saturation, value), YUV, etc.
- Luminance: Perceived intensity
- Chrominance: Perceived color
  - HS(V), (Y)UV, etc.
  - Normalized RGB removes some illumination dependence:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}$$

Binary Operations

- Dilation, erosion (Matlab: `imdilate`, `imerode`)
  - Dilation: All 0's next to a 1 $\rightarrow$ 1 (Enlarge foreground)
  - Erosion: All 1's next to a 0 $\rightarrow$ 0 (Enlarge background)
- Connected components
  - Uniquely label each *n*-connected region in binary image
  - 4- and 8-connectedness
  - Matlab: `bwfill`, `bwselect`



- Moments: Region statistics
  - Zeroth-order: Size
  - First-order: Position (centroid)
  - Second-order: Orientation

Image Transformations

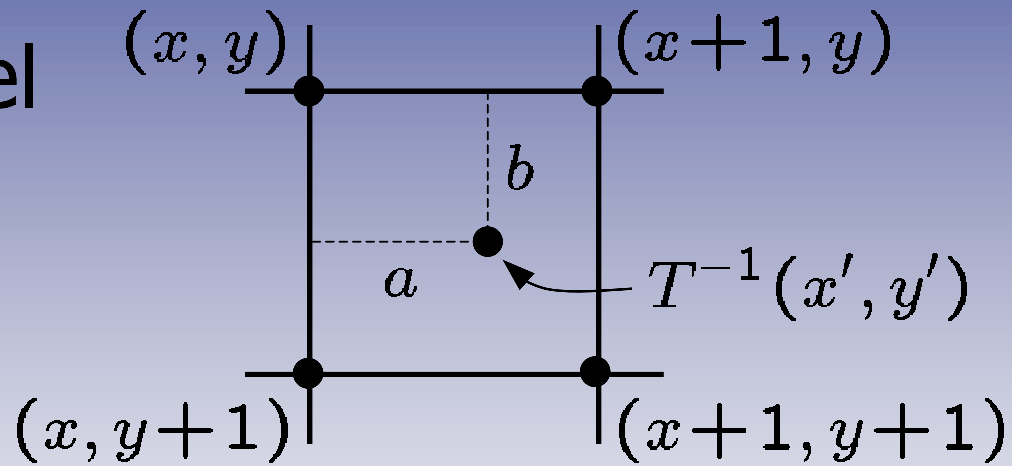
- Geometric: Compute new pixel locations
  - Rotate
  - Scale
  - Undistort (e.g., radial distortion from lens)

$$\boxed{T(x, y) \rightarrow (x', y')}$$

- Photometric: How to compute new pixel values when $T^{-1}(x', y')$ non-integral
  - Nearest neighbor: Value of closest pixel
  - Bilinear interpolation (2 x 2 neighborhood)
  - Bicubic interpolation (4 x 4)

Bilinear Interpolation

- Idea: Blend four pixel values surrounding source, weighted by nearness



$$\mathbf{I}(x', y') = (1-b, b)\begin{bmatrix} \mathbf{I}(x,y) & \mathbf{I}(x+1,y) \\ \mathbf{I}(x,y+1) & \mathbf{I}(x+1,y+1) \end{bmatrix}\begin{pmatrix} 1-a \\ a \end{pmatrix}$$

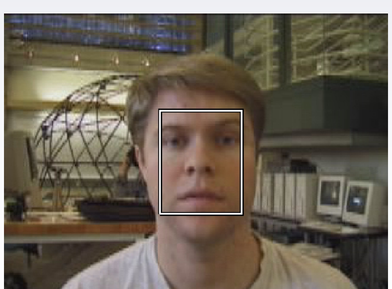
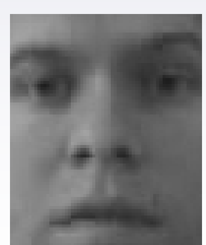
Vertical blend

Horizontal blend

Image Comparison: SSD

- Given a template image $\mathbf{I}_T$ and an image $\mathbf{I}$, how to quantify the similarity between them for a given alignment?
- Sum of squared differences (SSD)

$$\sum_{x,y} [\mathbf{I}_T(x,y) - \mathbf{I}(x,y)]^2$$

Cross-Correlation for Template Matching

- Note that SSD formula can be written:

$$\sum_{x,y} \mathbf{I}_T^2(x,y) + \mathbf{I}^2(x,y) - 2\mathbf{I}_T(x,y)\mathbf{I}(x,y)$$

- First two terms fixed $\rightarrow$ last term measures mismatch—the *cross-correlation:*

$$\sum_{x,y} \mathbf{I}_T(x,y) \cdot \mathbf{I}(x,y)$$

- In practice, normalize by image $\mathbf{I}$ magnitude when shifting template to search for matches

Filtering

- Idea: Analyze neighborhood around some point in image f with filter function h; put result in new image g at corresponding location
- System properties
  - Shift invariance: Same inputs give same outputs, regardless of location
  - Superposition: Output on sum of images =
  - Sum of outputs on separate images
  - Scaling: Output on scaled image = Scaled output on image
- Linear shift invariance $\rightarrow$ **Convolution**

Convolution

- Definition:

$$\boxed{g(x_0, y_0) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x_0 - x, y_0 - y)h(x, y)dxdy}$$

- Shorter notation: $g = f * h$
- Properties
  - Commutative
  - Associative
- Fourier theorem: Convolution in spatial domain = Multiplication in frequency domain
  - More on Fourier transforms on Thursday

Discrete Filtering

1	-1	-1
1	2	-1
1	1	1

- Linear filter: Weighted sum of pixels over rectangular neighborhood—*kernel* defines weights
- Think of kernel as template being matched by correlation (Matlab: `imfilter`, `filter2`)
- Convolution: Correlation with kernel rotated 180°
  - Matlab: `conv2`
- Dealing with image edges
  - Zero-padding
  - Border replication

Filtering Example 1: $\mathbf{I}' = \mathbf{K} * \mathbf{I}$

$\mathbf{K}$

1	-1	-1
1	2	-1
1	1	1

Rotate

1	1	1
-1	2	1
-1	-1	1

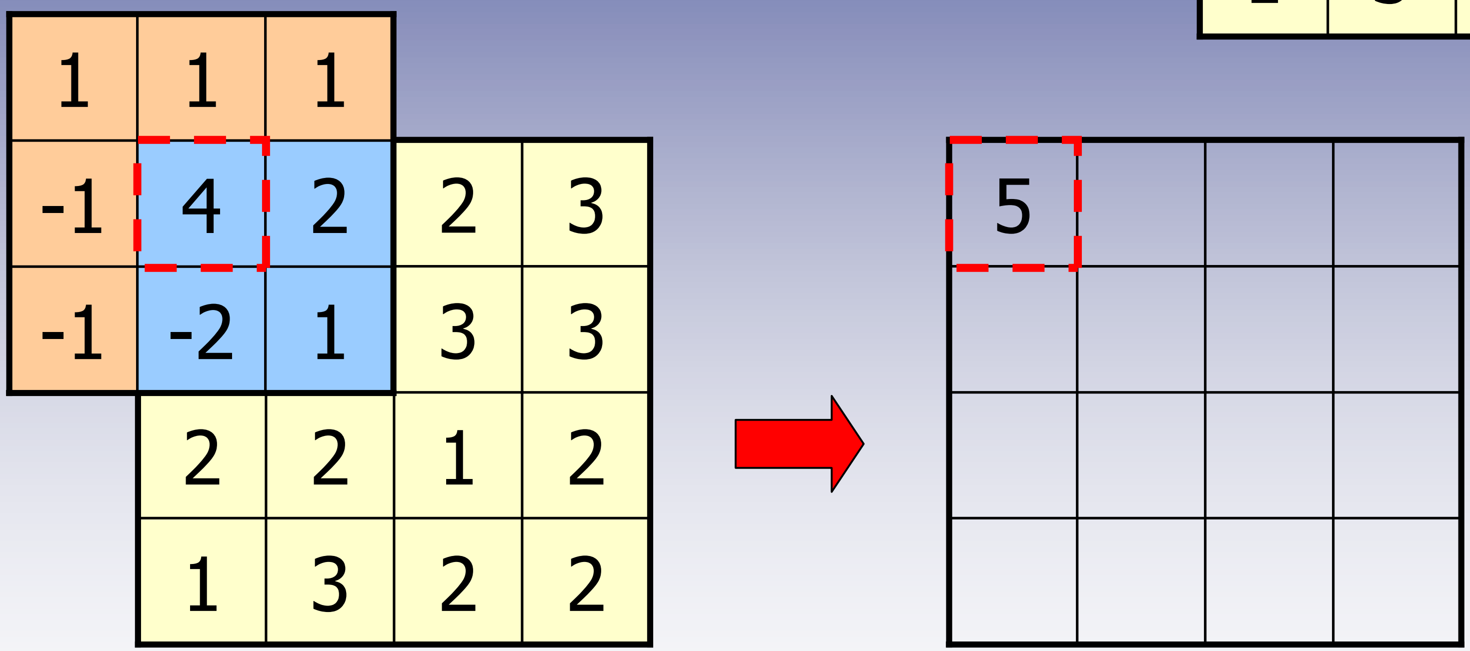
$\mathbf{I}$

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	1	1
-1	2	1
-1	-1	1

Step 1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

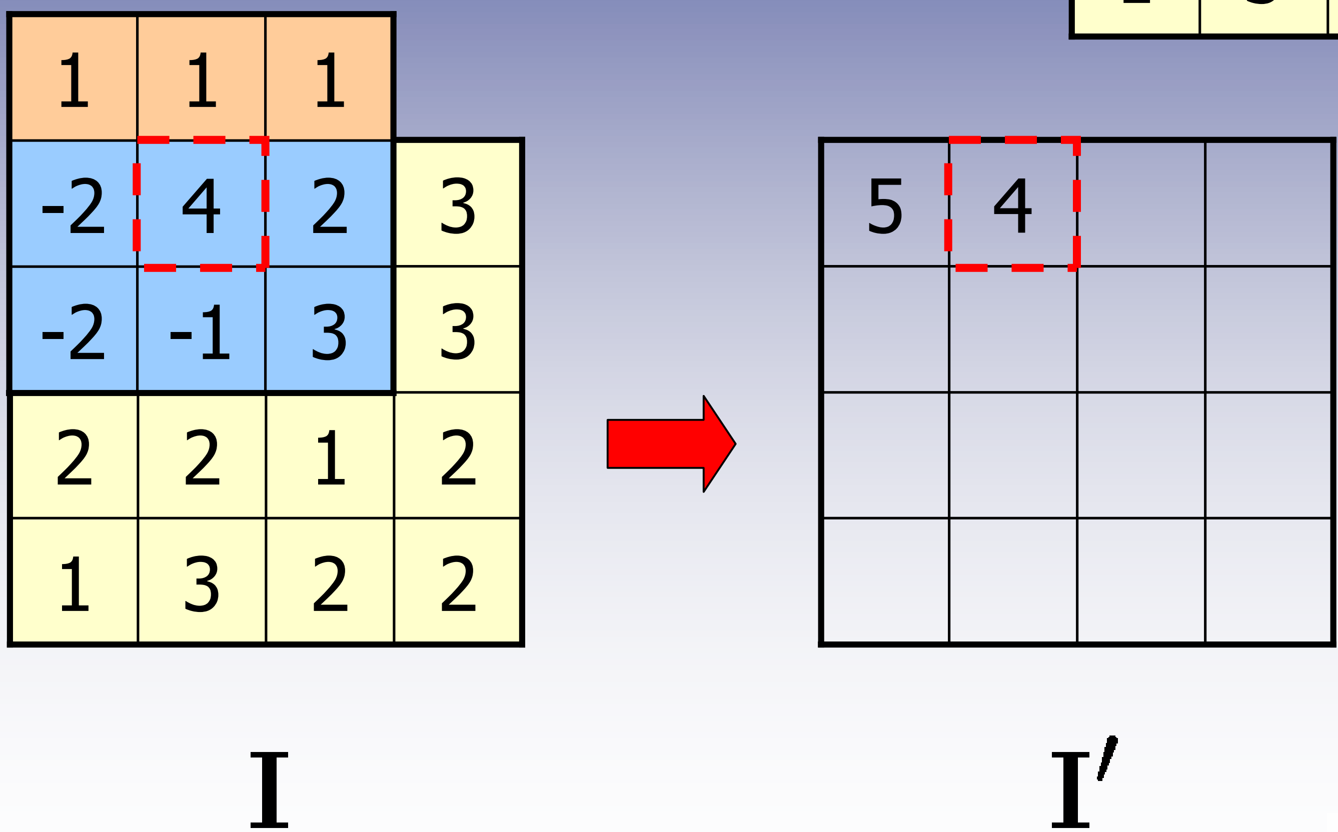


$\mathbf{I}$ $\mathbf{I}'$

Step 2

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



Step 3

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

	1	1	1
2	-2	4	3
2	-1	-3	3
2	2	1	2
1	3	2	2

$\mathbf{I}$

5	4	4	

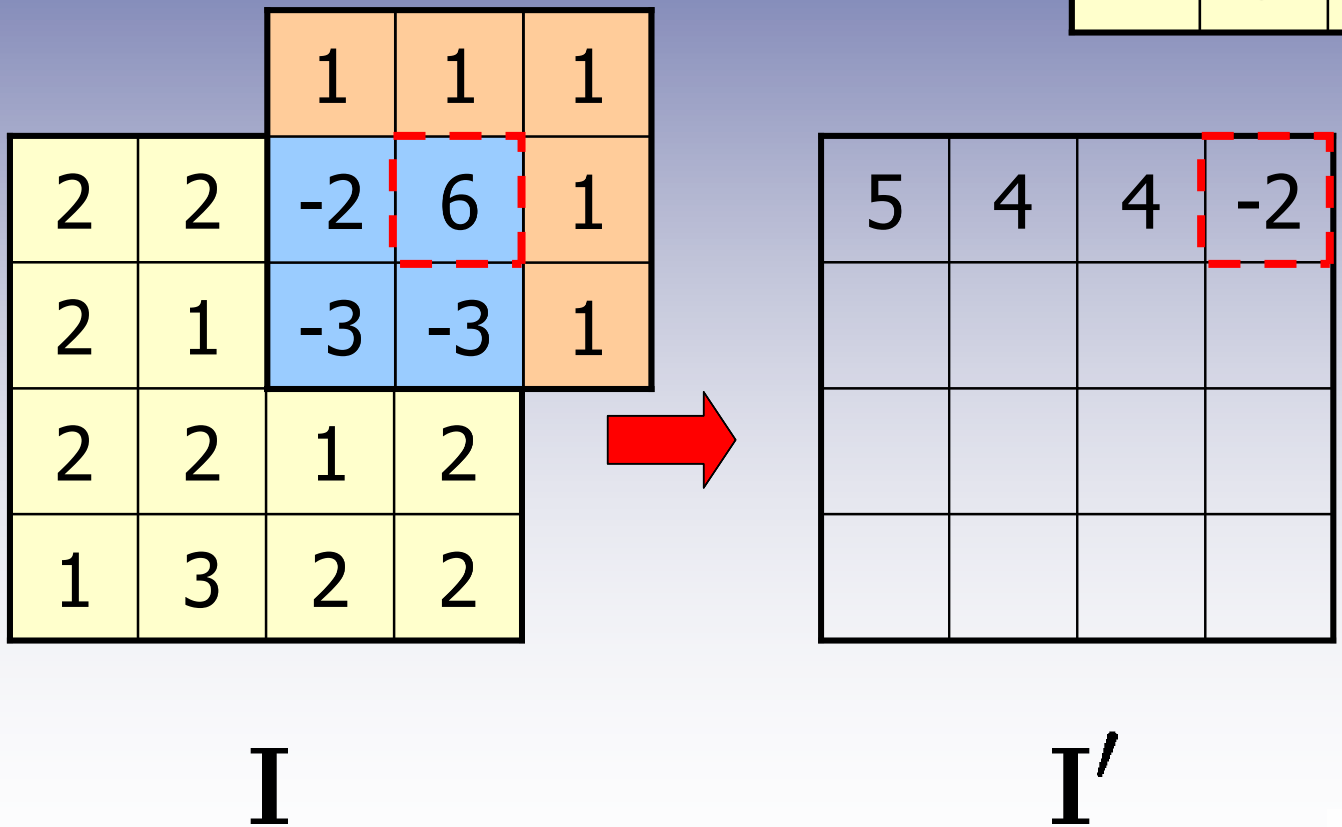
$\mathbf{I}'$

UNIVERSITY OF DELAWARE

Step 4

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



Step 5

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

1	2	2	2	3
-1	4	1	3	3
-1	-2	2	1	2
	1	3	2	2

$\mathbf{I}$

5	4	4	-2
9			

$\mathbf{I}'$

Step 6

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2

$\mathbf{I}$

5	4	4	-2
9	6		

$\mathbf{I}'$

Final Result

5	4	4	-2
9	6	14	5
11	7	6	5
9	12	8	5

$$\mathbf{I}'$$

Separability

- Definition: 2-D kernel can be written as convolution of two 1-D kernels
- Advantage: Efficiency—Convolving image with $n \times m$ kernel requires $n + m$ multiplies vs. nm for non-separable kernel

Smoothing (Low-Pass) Filters

- Replace each pixel with average of neighbors
- Benefits: Suppress noise, aliasing
- Disadvantage: Sharp features blurred
- Types
  - Mean filter (box)
  - Median (nonlinear)
  - Gaussian

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

3 x 3 box filter

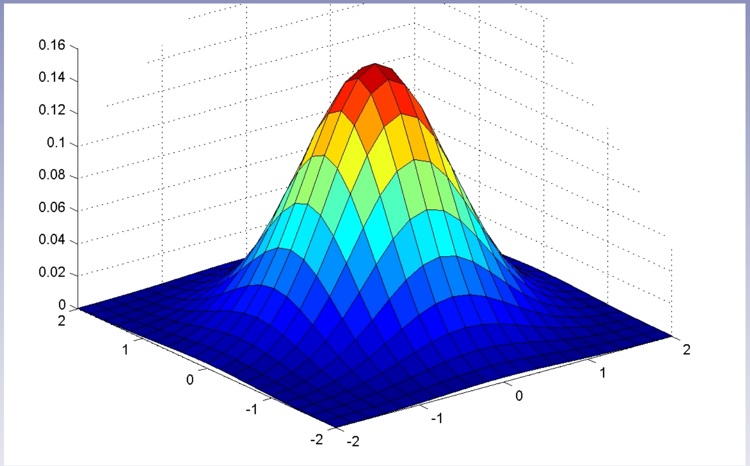
Box Filter: Smoothing

Original image

7 x 7 kernel

Gaussian Kernel

- Idea: Weight contributions of neighboring pixels by nearness



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

- Matlab: `fspecial('gaussian',…)`

UNIVERSITY OF DELAWARE

Gaussian: Benefits

- Rotational symmetry treats features of all orientations equally (isotropy)
- Smooth roll-off reduces "ringing"
- Efficient: Rule of thumb is kernel width $\geq 5\sigma$
  - Separable
  - Cascadable: Approach to large σ comes from identity

$$G_{\sigma_1} * G_{\sigma_2} = G_{\sqrt{\sigma_1^2+\sigma_2^2}}$$

Gaussian: Smoothing

Original image

σ = 1

7 x 7 kernel

σ = 3

Gradient

- Think of image intensities as a function $\mathbf{I}(x, y)$. Gradient of image is a vector field as for a normal 2-D height function:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y}\right)^T = (\mathbf{I}_x, \mathbf{I}_y)^T$$

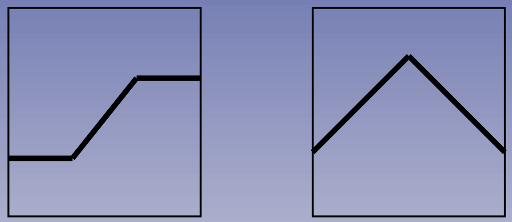
- **Edge**: Place where gradient magnitude is high; orthogonal to gradient direction

Edge Causes

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)

Edge Detection

- Edge Types
  - Step edge (ramp)
  - Line edge (roof)



- Searching for Edges:
  - **Filter**: Smooth image
  - **Enhance**: Apply numerical derivative approximation
  - **Detect**: Threshold to find strong edges
  - **Localize/analyze**: Reject spurious edges, include weak but justified edges

Step edge detection

- First derivative edge detectors: *Look for extrema*
  - Sobel operator (Matlab: `edge(I, 'sobel')`)
  - Prewitt, Roberts cross
  - Derivative of Gaussian

1	0	-1
2	0	-2
1	0	-1

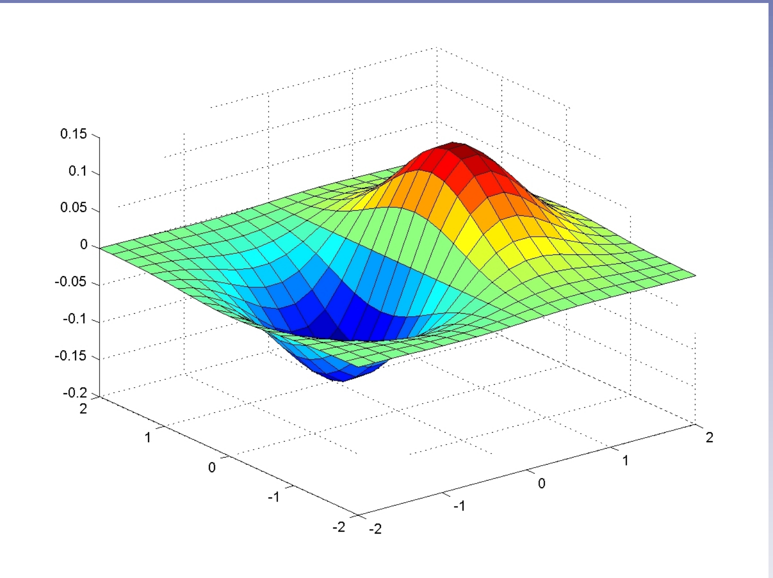
Sobel x

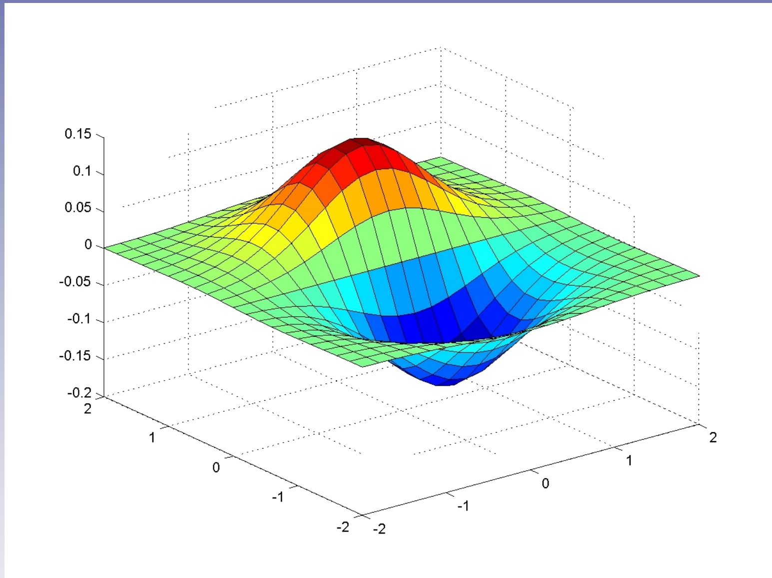
1	2	1
0	0	0
-1	-2	-1

Sobel y

- Second derivative: *Look for zero-crossings*
  - Laplacian $\nabla^2 \mathbf{I}$: Isotropic
  - Second directional derivative
  - Laplacian of Gaussian/Difference of Gaussians

Derivative of Gaussian



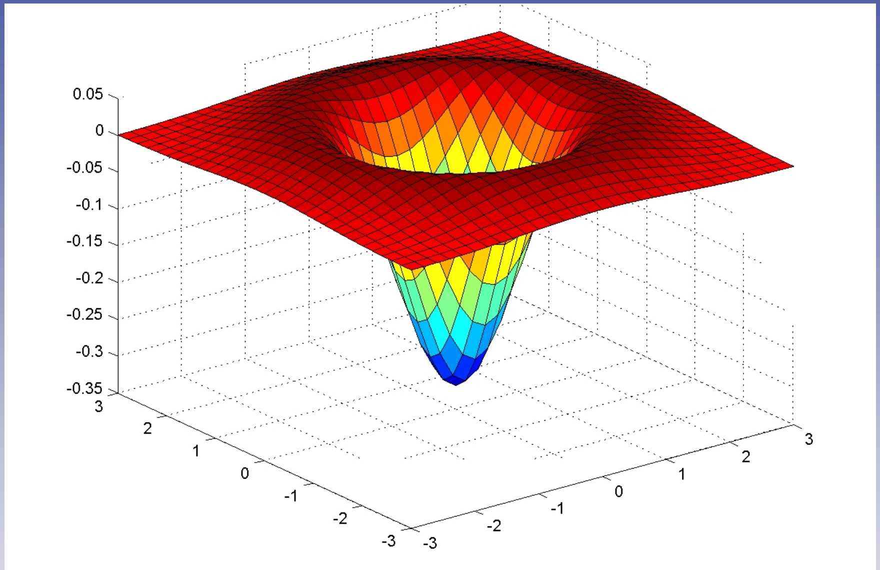


$$\frac{\partial}{\partial x} G_\sigma$$

$$\frac{\partial}{\partial y} G_\sigma$$

University of Delaware

Laplacian of Gaussian



$$LoG_\sigma = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Matlab: `fspecial('log',…)`

Edge Filtering Example

K

1	0	-1
2	0	-2
1	0	-1

Rotate

-1	0	1
-2	0	2
-1	0	1

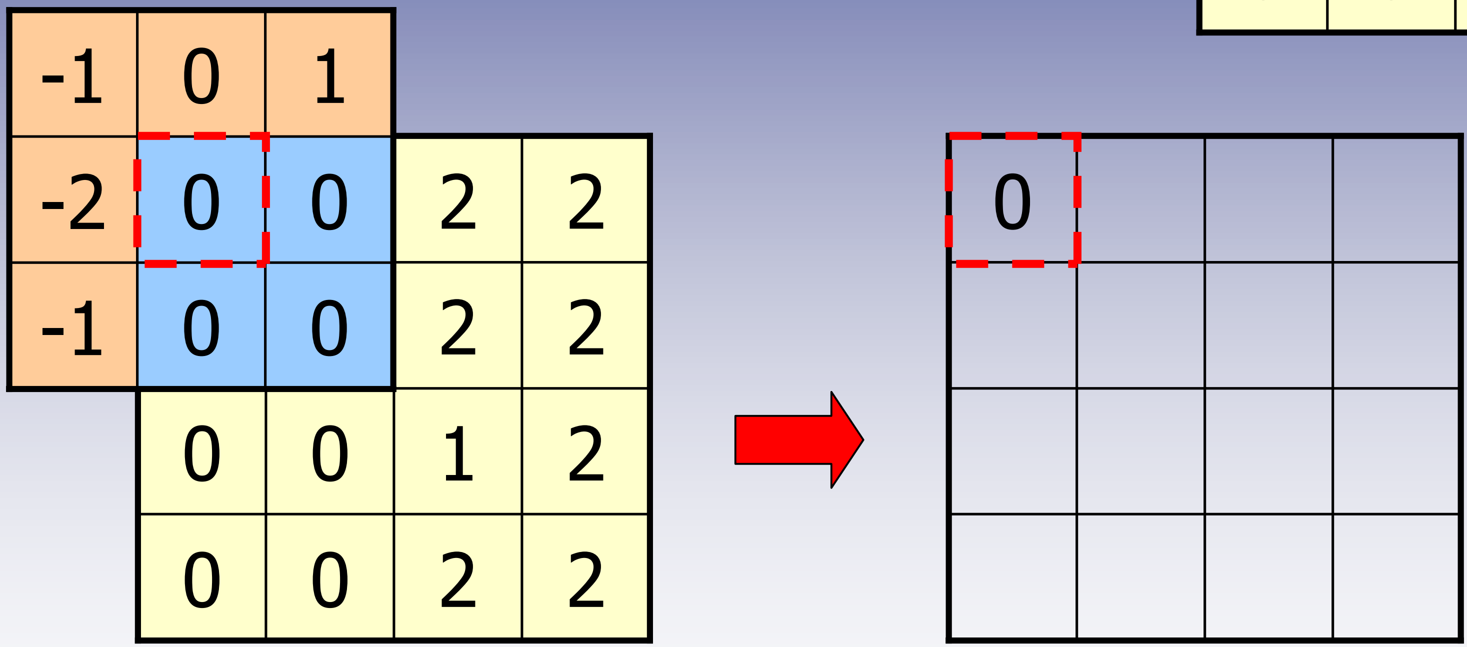
I

0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2

-1	0	1
-2	0	2
-1	0	1

Step 1

0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2

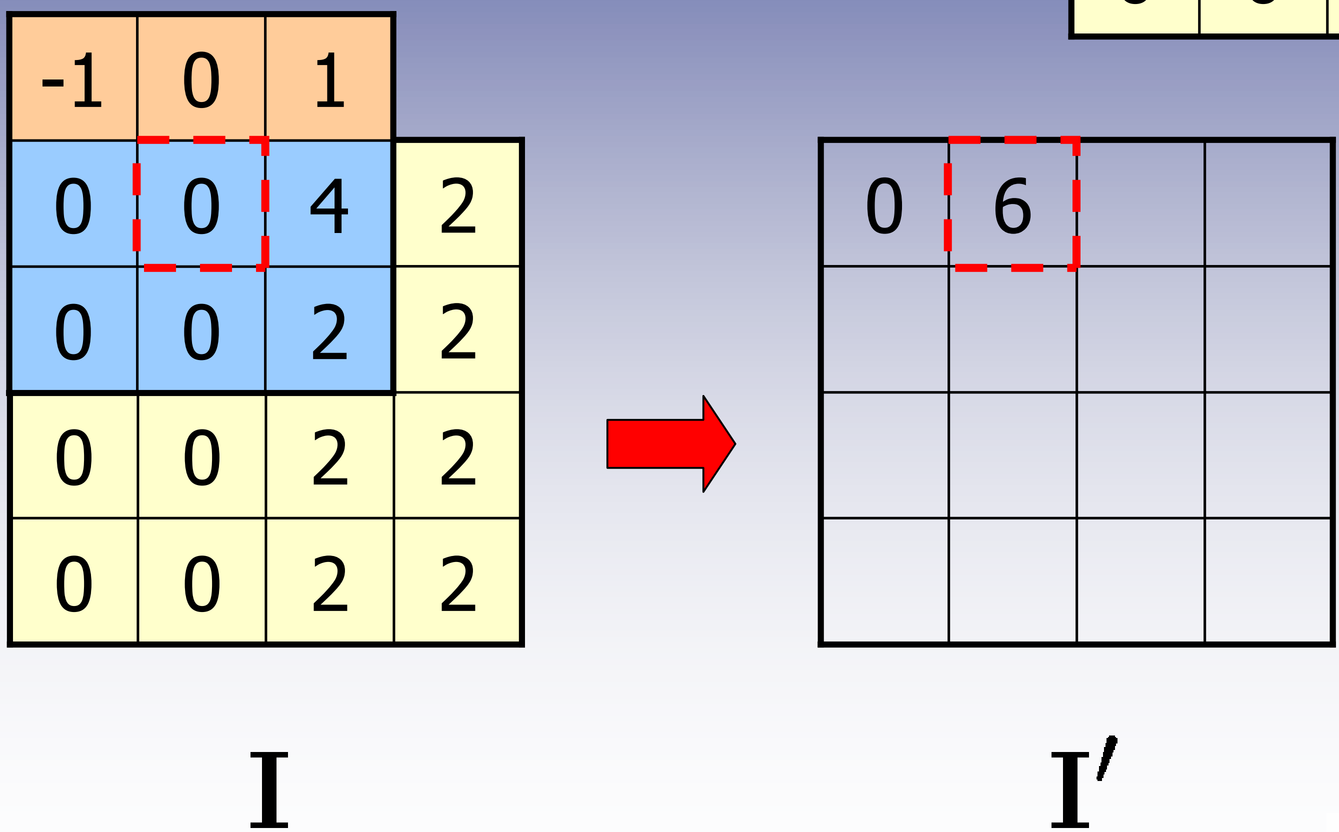


$\mathbf{I}$ $\mathbf{I}'$

Step 2

-1	0	1
-2	0	2
-1	0	1

0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2



Step 3

-1	0	1
-2	0	2
-1	0	1

0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2

	-1	0	1
0	0	0	4
0	0	0	2
0	0	2	2
0	0	2	2

$\mathbf{I}$

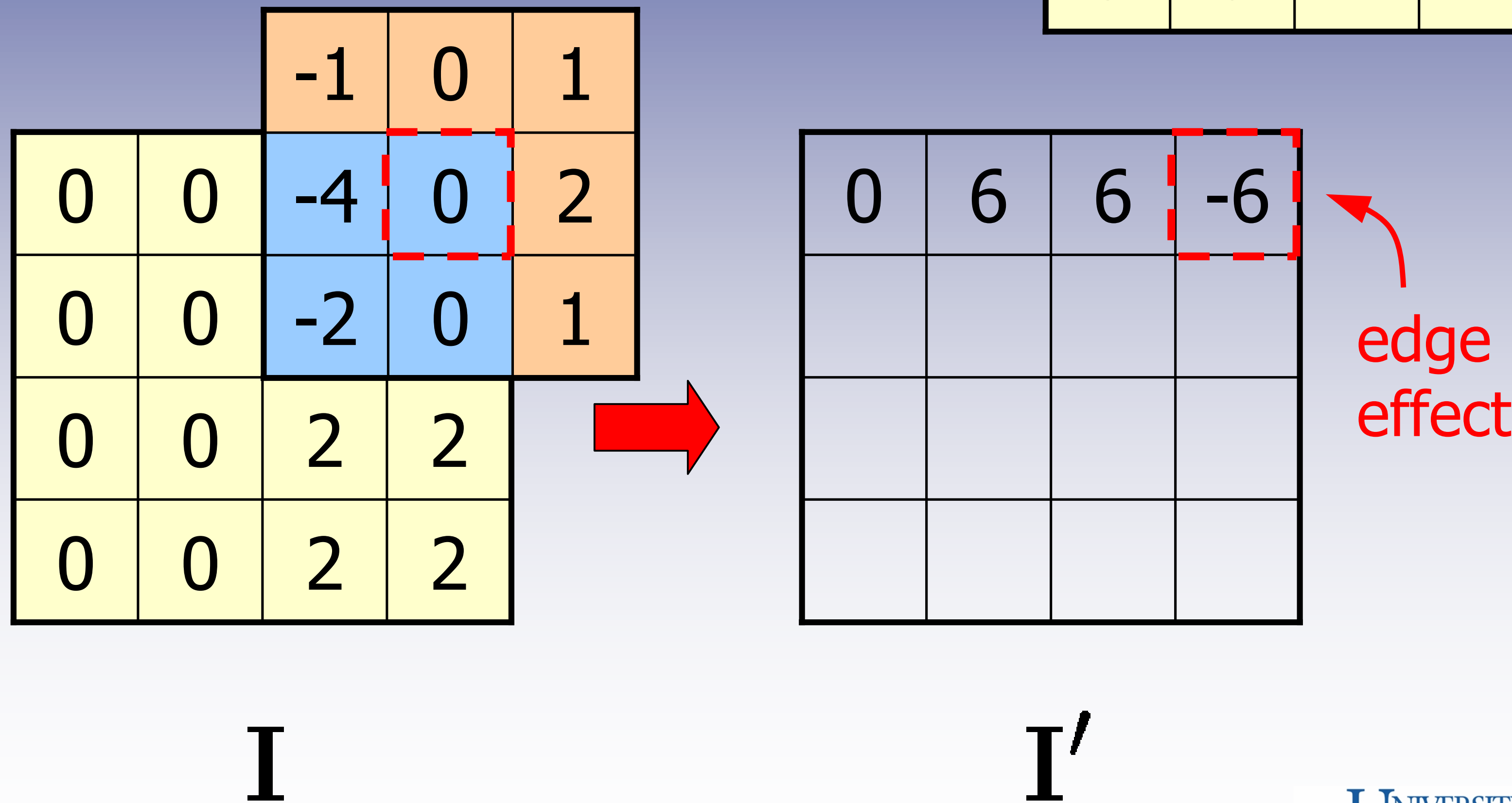
0	6	6	

$\mathbf{I}'$

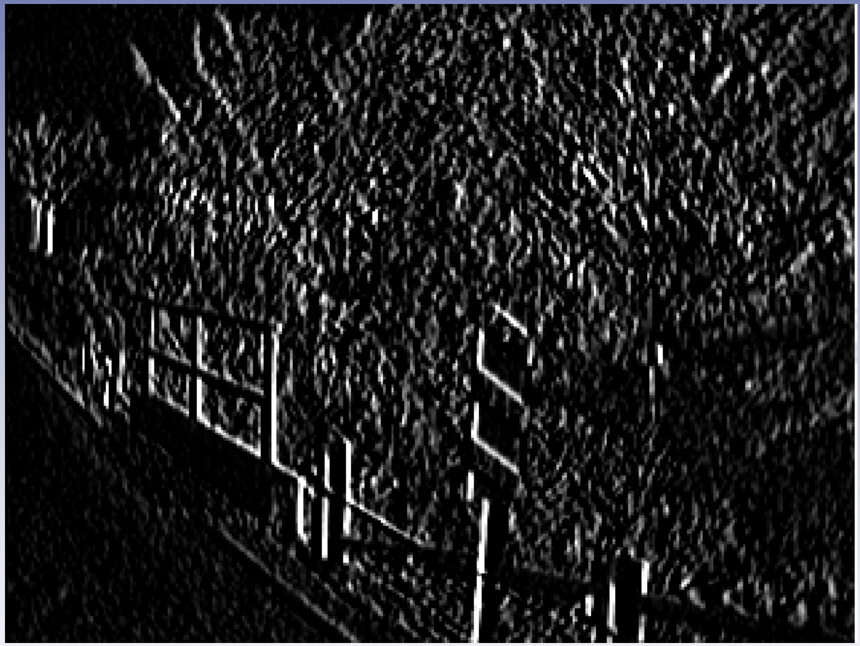
-1	0	1
-2	0	2
-1	0	1

Step 4

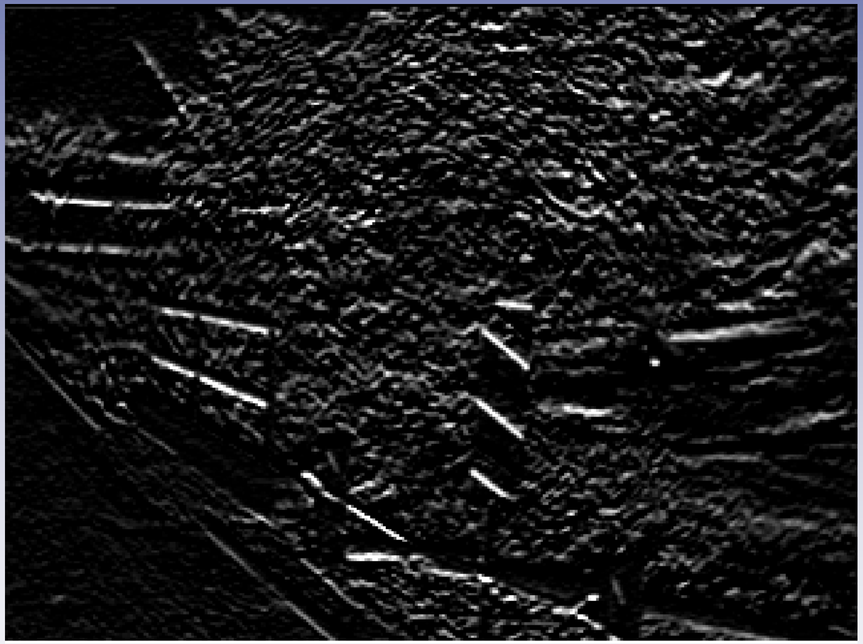
0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2



Sobel Edge Detection: Gradient Approximation



Horizontal



Vertical

Sobel vs. LoG Edge Detection: Matlab Automatic Thresholds

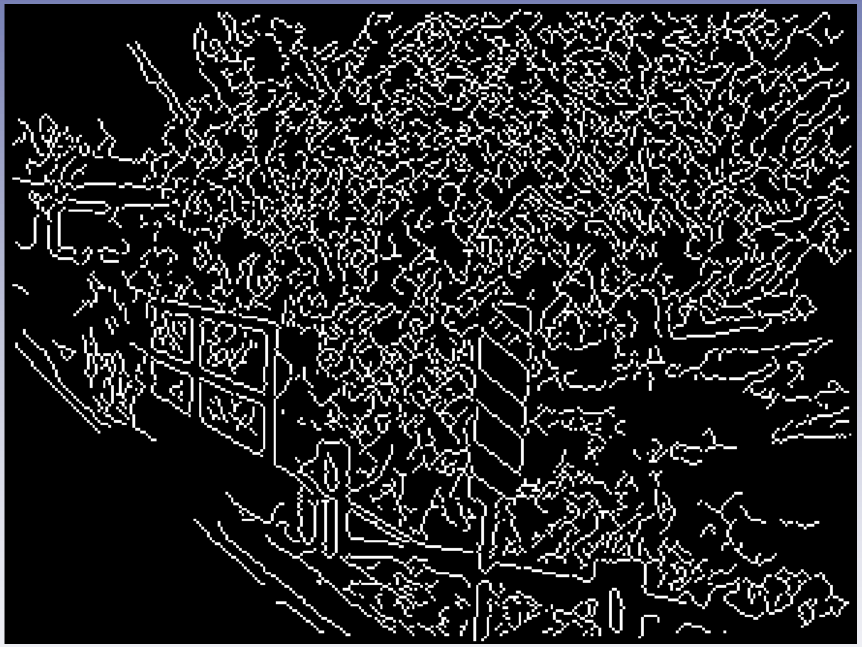
Sobel

LoG

Canny Edge Detection

- Derivative of Gaussian
- Non-maximum suppression
  - Thin multi-pixel wide "ridges" down to single pixel
- Thresholding
  - Low, high edge-strength thresholds
  - Accept all edges over low threshold that are connected to edge over high threshold
- Matlab: `edge(I, 'canny')`

Canny Edge Detection: Example



(Matlab automatically set thresholds)

Corner Detection

- Basic idea: Find points where two edges meet—i.e., high gradient in orthogonal directions
- Examine gradient over window (Shi & Tomasi, 1994)

$$C = \begin{pmatrix} \sum \mathbf{I}_x^2 & \sum \mathbf{I}_x \mathbf{I}_y \\ \sum \mathbf{I}_x \mathbf{I}_y & \sum \mathbf{I}_y^2 \end{pmatrix}$$

- Edge strength encoded by eigenvalues λ_1, λ_2; corner is where $\min(\lambda_1, \lambda_2)$ over threshold
- Harris corners (Harris & Stephens, 1988), Susan corners (Smith & Brady, 1997)

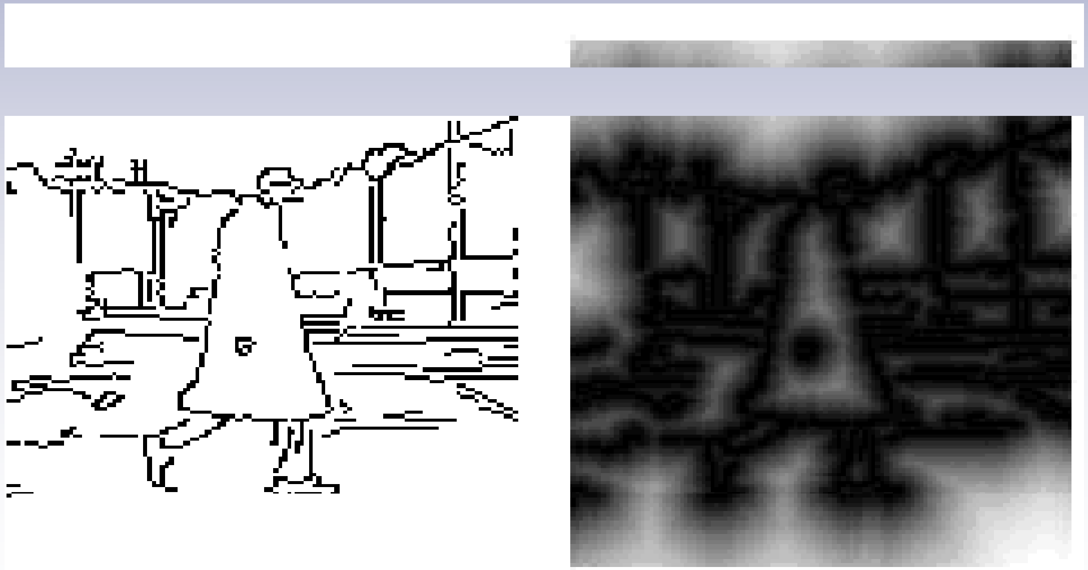
Example: Corner Detection

courtesy of S. Smith

SUSAN corners

Edge-Based Image Comparison

- Chamfer, Hausdorff distance, etc.
  - Transform edge map based on distance to nearest edge before correlating as usual



courtesy of D. Gavrila

Scale Space

- How thick an edge? How big a dot?
- Must consider what scale we are interested in when designing filters
- Efficiency a major consideration: Fine-grained template matching is expensive over a full image

Image Pyramids

- Idea: Represent image at different scales, allowing efficient *coarse-to-fine* search
- Downsampling: $S^{\downarrow}(\mathbf{I})(x, y) = \mathbf{I}(2x, 2y)$
- Simplest scale change: Decimation—just downsample

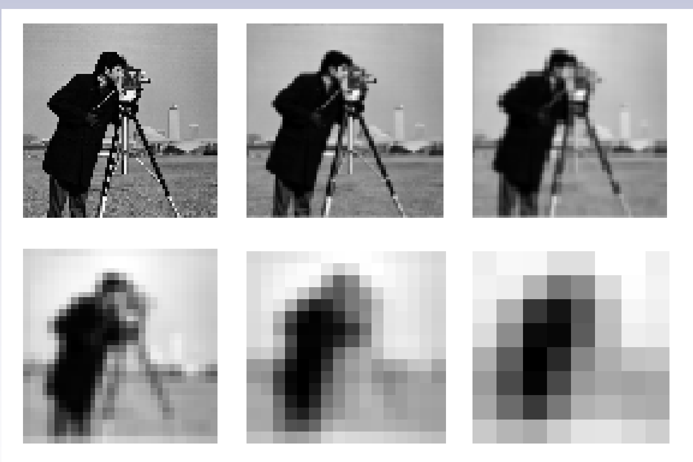
from Forsyth & Ponce

Gaussian, Laplacian Pyramids

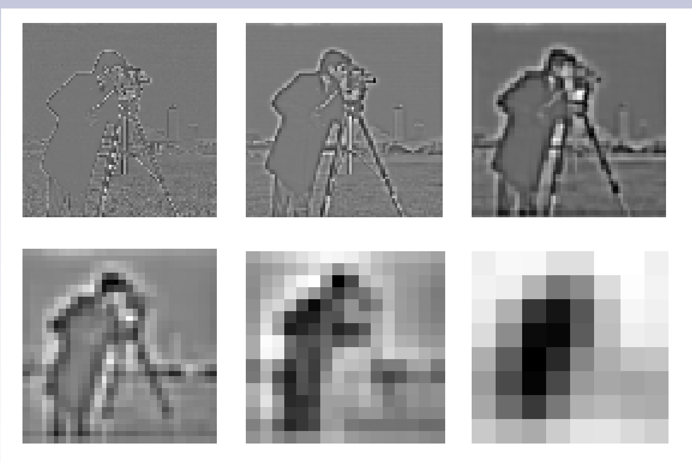
- Gaussian pyramid of image: $\mathbf{P}_0 = \mathbf{I}$ and

$$\mathbf{P}_i(\mathbf{I}) = S^{\downarrow}(G_\sigma * \mathbf{P}_{i-1}(\mathbf{I}))$$

- Laplacian pyramid
  - Difference of image and Gaussian at each level of $\mathbf{P}_i(\mathbf{I})$



Gaussian pyramid



Laplacian pyramid

courtesy of Wolfram

Color-based Image Comparison

- Color histograms (Swain & Ballard, 1991)
  - Steps
    - Histogram RGB/HSV triplets over two images to be compared
    - Normalize each histogram by respective total number of pixels to get frequencies
    - Similarity is Euclidean distance between color frequency vectors
  - Sensitive to lighting changes
  - Works for different-sized images
  - Matlab: `imhist, hist`

University of Delaware