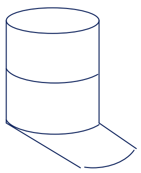
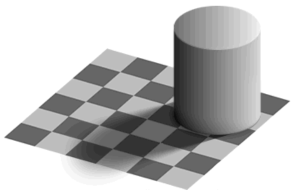
Edge Detection

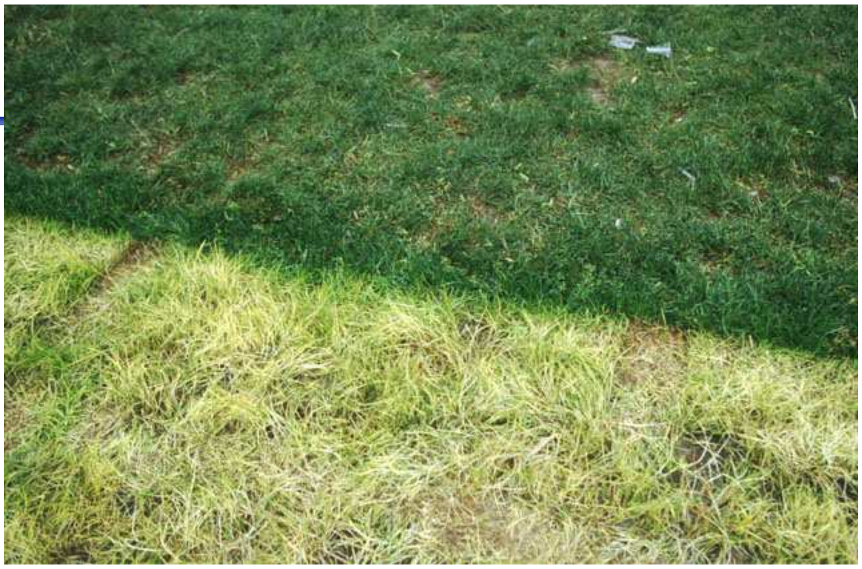
- Why Detect Edges?
  - Information reduction
    - Replace image by a cartoon in which objects and surface markings are outlined $\Rightarrow$ create line drawing description
    - These are the most informative parts of the image
  - Biological plausibility
    - Initial stages of mammalian vision systems involve detection of edges and local features
  - Applications
    - Object recognition, stereo, texture analysis, motion analysis, image enhancement, image compression

What Causes Image Intensity Changes?

- Many types of physical events cause intensity changes
  - **Surface reflectance discontinuity** - change in the fraction of light incident on the surface that is reflected to viewer
  - **Illumination discontinuity** - shadow
  - **Surface orientation (normal) discontinuity**
  - **Depth discontinuity** - at occluding contour, where surface orientation is perpendicular to line of sight

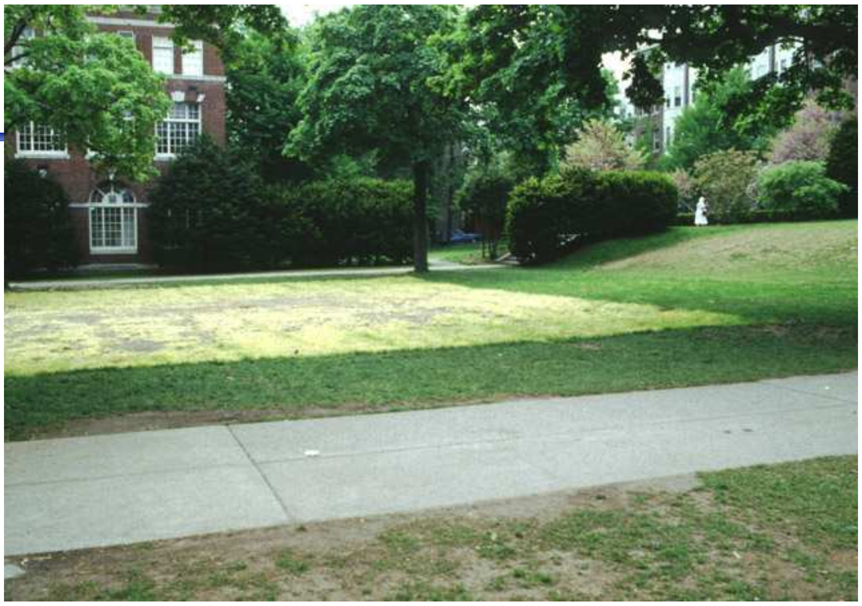






What type of real-world edge is this contrast border?

3

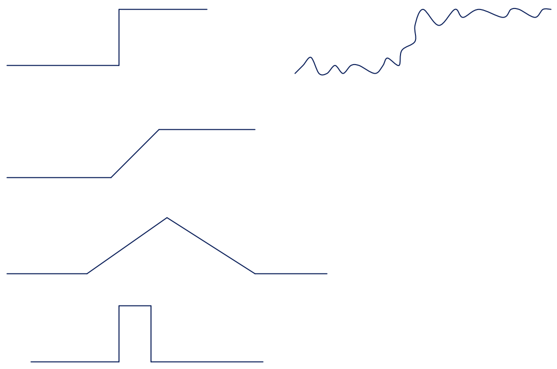


Need global information too

4

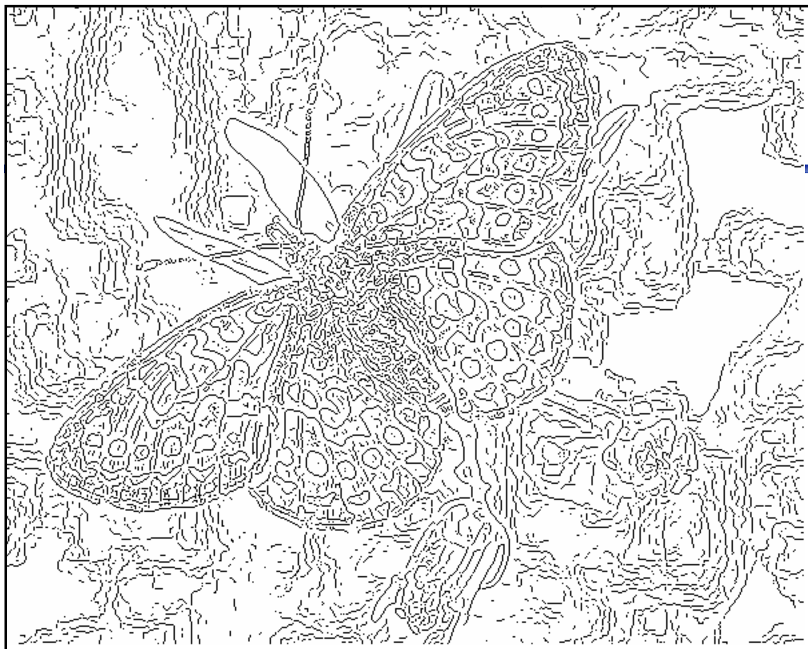
What Does an Edge Look Like?

- Step
- Ramp
- Roof
- Line (bar)



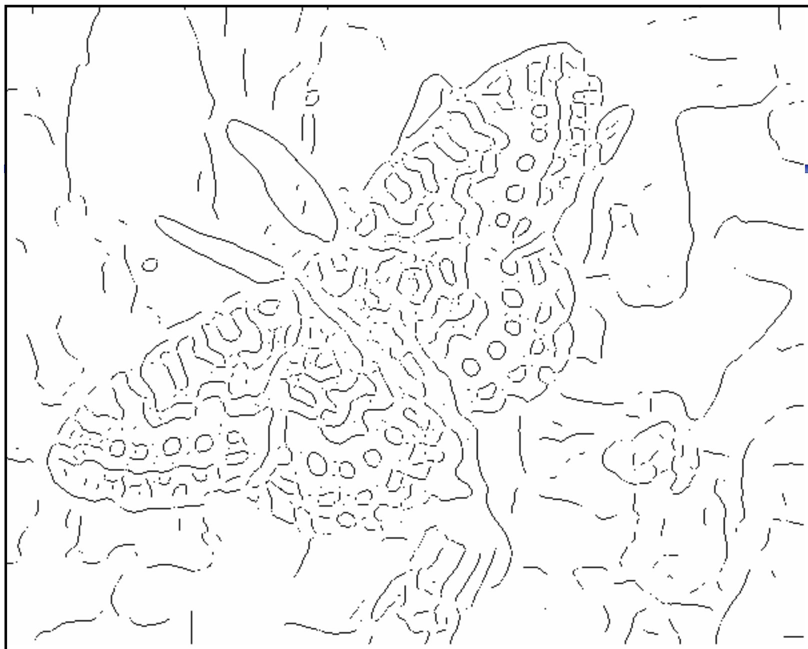
5

6



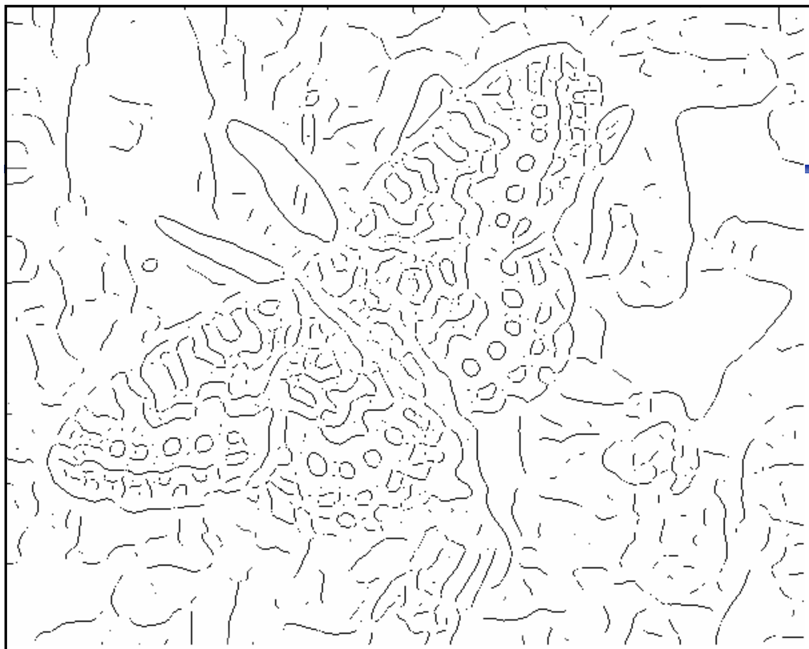
fine scale
high
threshold

7



coarse
scale,
high
threshold

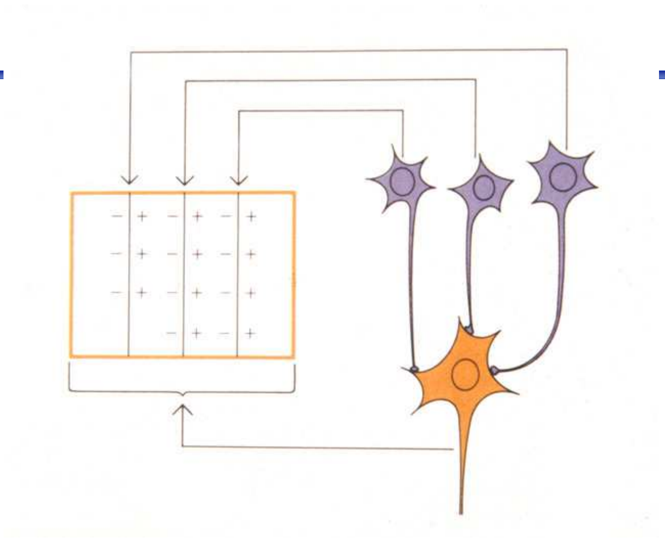
8



coarse
scale
low
threshold

9

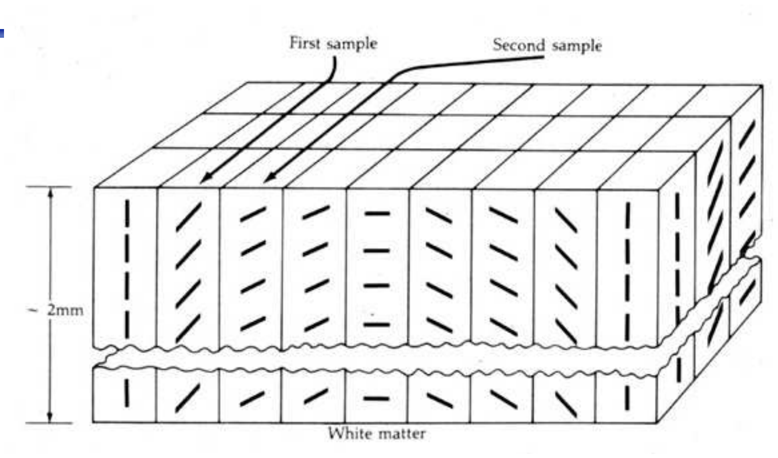
Edge Detectors in the HVS



- Sum activity from an array of oriented simple cells

10

Orientation Columns in V1



- A “hypercolum” = complete set of orientations

11

Craik-O’Brien-Cornsweet Illusion

12

Koffka Ring

13

Luminance Differences Affect our Perceptions

- Artists use the technique of “equiluminance” to *blur* outlines and suggest *motion*. We cannot perceive the edges of objects where object and background have the same luminance. If parts of a painting are equiluminant, their positions become ambiguous. They may seem to shift position or to float
- Equiluminant colors have special properties, e.g., they can make a painting appear unstable

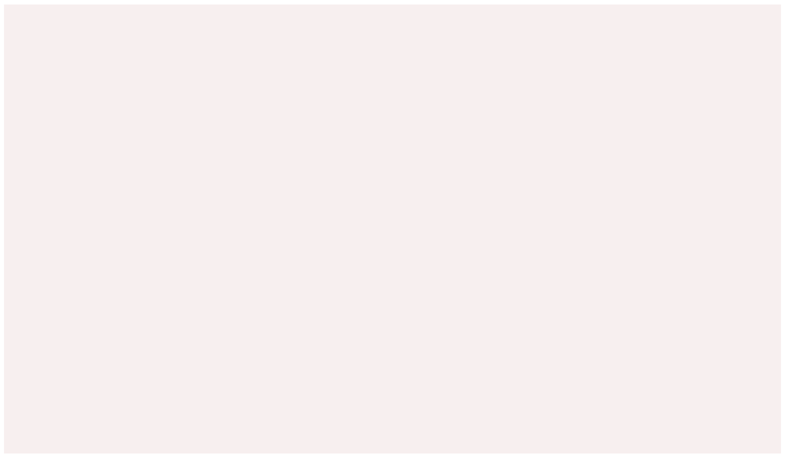
14

Equiluminant Colors

- An object that can be seen by both subdivisions of the visual system will be perceived accurately. But if the two subdivisions are not balanced in their response to an object, it may look peculiar.
- For example, an object defined by equiluminant colors can be seen by the What system but is invisible (or poorly seen) by the Where system. It may seem flat, it may seem to shift position, or it may seem to float ambiguously because there is too little luminance contrast to provide adequate information about its three-dimensional shape, its location in space, or its motion (or lack of it).
- Conversely, something defined by very low contrast contours is seen by the Where system but not the What system and may seem to have depth and spatial organization but no clear shape.

15

Detail from Richard Anuszkiewicz's *Plus Reversed*, 1960



The red and blue seem to move around because they are equiluminant

16

Edge Detection Goals

- **Good detection**: Low false alarm rate and low false dismissal rate $\Rightarrow$ maximize signal-to-noise (S/N) ratio
- **Good localization**: Mark point closest to “center” of true edge $\Rightarrow$ minimize distance between marked point and center
- **Uniqueness**: Only one response to a single edge
- **Good property measurement**: Orientation, contrast, etc.

17

Edge Operator Properties

- **Shift invariant** (translation invariant, position invariant)
  - If $g(x,y) = Op[f(x,y)]$ then $g(x-a, y-b) = Op[f(x-a, y-b)]$
- **Isotropic** (rotation invariant) vs. non-isotropic
- Derivative order (if differentiation-based method)
- **Linear** vs. non-linear
  - $Op[a\, f_1(x,y) + b\, f_2(x,y)] = a\, Op[f_1(x,y)] + b\, Op[f_2(x,y)]$
    $= a\, g_1(x,y) + b\, g_2(x,y)$
- **Scale** (operator neighborhood size)
  - $g(x,y) = Op[f(x+a, y+b), \forall -k < a, b < k]$
- **Convolution** (linear and shift-invariant)

18

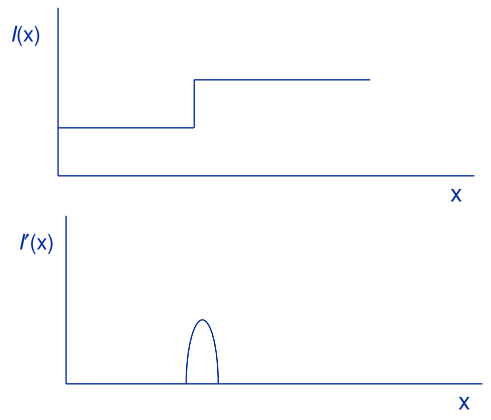
Edge and Local Feature Detection Methods

- Gradient-based edge detection
- Edge detection by function fitting
- Second derivative edge detectors
- Edge linking and the construction of the chain graph

19

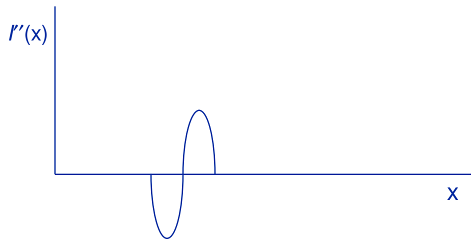
1D Edge Detection

- An ideal edge is a step function



20

1D Edge Detection

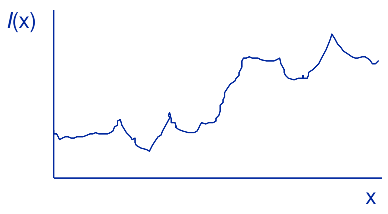


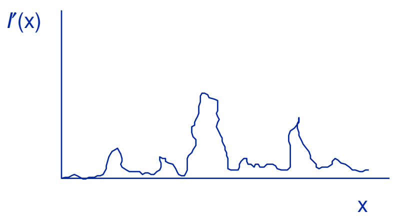
- The first derivative of I(x) has a **peak** at the edge
- The second derivative of I(x) has a **zero crossing** at the edge

21

1D Edge Detection

- More realistically, image edges are **blurred** and the regions that meet at those edges have **noise** or variations in intensity
  - Blur - high first derivatives near edges
  - Noise - high first derivatives within regions that meet at edges





22

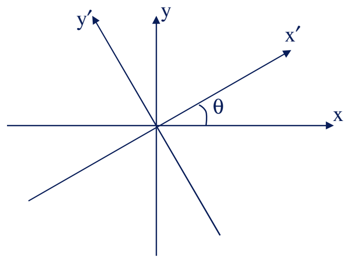
Edge Detection in 2D

- Let I(x,y) be the image intensity function. It has derivatives in all directions
  - $\partial I(x, y)/\partial x = \lim I(x+\Delta x, y) - I(x, y) / \Delta x \approx I(u+1, v) - I(u,v)$
  - **Gradient** of I(x, y) is a vector $\nabla I(x, y) = [\partial I/\partial x, \partial I/\partial y]^T$ specifying the direction of greatest rate of change in intensity (i.e., perpendicular to the edge's direction)
  - From gradient can determine the **direction** in which the first derivative is highest, and the **magnitude** of the first derivative in that direction
  - Magnitude = $[(\partial I/\partial x)^2 + (\partial I/\partial y)^2]^{1/2}$
  - Direction = $\tan^{-1} (\partial I/\partial y)/(\partial I/\partial x)$

23

Computing First Derivative

- To compute first derivative in direction θ, calculate from linear combination of derivatives from any two non-collinear directions



$x = x'\cos\theta - y'\sin\theta$

$y = x'\sin\theta + y'\cos\theta$

$\partial I/\partial x' = \partial I/\partial x\, \partial x/\partial x' + \partial I/\partial y\, \partial y/\partial y'$
$= \partial I/\partial x \cos\theta + \partial I/\partial y \sin\theta$

$\partial I/\partial y' = -\partial I/\partial x \sin\theta + \partial I/\partial y \cos\theta$

$(\partial I/\partial x')^2 + (\partial I/\partial y')^2 = (\partial I/\partial x)^2 + (\partial I/\partial y)^2$

So, sum of squares of first derivative is isotropic, non-linear

Similarly, all derivatives of odd ordcr raised to an even power are isotropic

24

Gradient

- Gradient = $[\partial I/\partial x, \partial I/\partial y]^T$
- What direction is first derivative a maximum?
  - Set $\partial/\partial\theta\,(\partial I/\partial x') = 0$, and solve for θ

    $\Rightarrow\ \partial/\partial\theta\,(\partial I/\partial x \cos\theta\ +\ \partial I/\partial y \sin\theta)\ =\ 0$

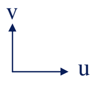
    $\Rightarrow\ \theta\ =\ \tan^{-1}(\partial I/\partial y\ /\ \partial I/\partial x)$
- Gradient direction is perpendicular to "edge direction"
- Gradient magnitude is isotropic

$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

25

Edge Detection in 2D

- With a digital image, the partial derivatives are replaced by finite differences:
  - $\Delta_x I = I(u+1, v) - I(u, v)$
  - $\Delta_y I = I(u, v) - I(u, v+1)$



- An alternative (Sobel)
  - $\Delta_{sobel\_X} I = I(u+1, v+1) + 2I(u+1, v) + I(u+1, v-1) - I(u-1, v+1) - 2I(u-1, v) - I(u-1, v-1)$
  - $\Delta_{sobel\_Y} I = I(u-1, v-1) + 2I(u, v-1) + I(u+1, v-1) - I(u-1, v+1) - 2I(u, v+1) - I(u+1, v+1)$
- Roberts's "Cross"
  - $\Delta_+ I = I(u, v) - I(u+1, v-1) \longrightarrow \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$
  - $\Delta_- I = I(u, v-1) - I(u+1, v) \longrightarrow \begin{matrix} 0 & -1 \\ 1 & 0 \end{matrix}$

26

Gradient Operator Example

- I = 0 0 0 1 2 3 4 4 4 8 8 8 3 3 3
- Δ_x = -1 1
- $\Delta_x I$ = * 0 0 1 1 1 1 0 0 4 0 0 -5 0 0

27

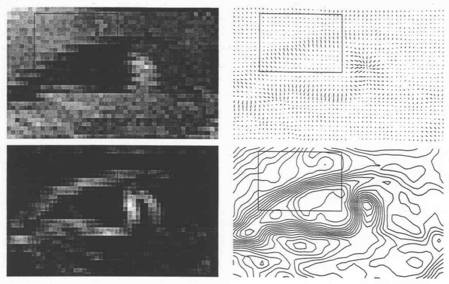


Figure 1: Detail of an eye (top left), the corresponding gradient field (top right), the magnitude of the gradient (bottom left), and the isocontours of the image intensity (bottom right). The area in the box is analyzed further in the text.

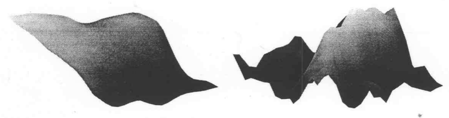


Figure 2: The image intensity function (left) in the box shown in figure 1 and the corresponding gradient magnitude (right).

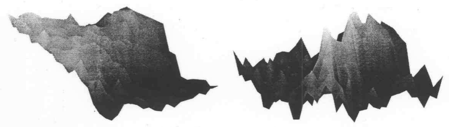
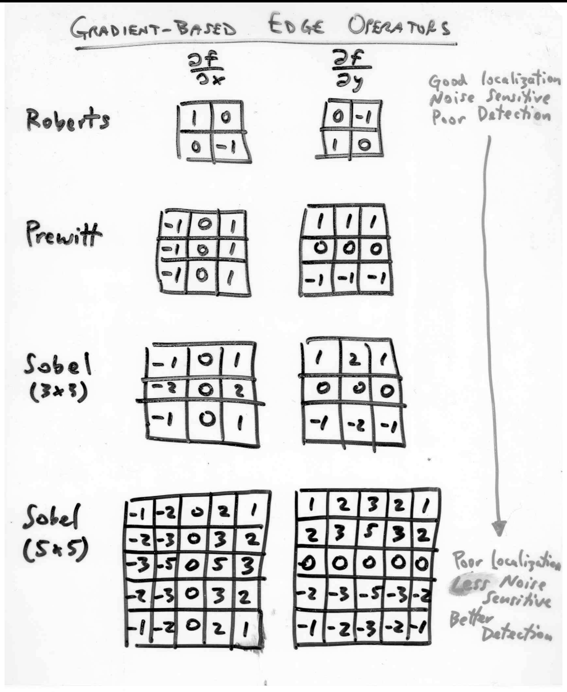


Figure 3: The same plots as in figure 2 but without smoothing. Noise is very visible.



GRADIENT-BASED EDGE OPERATORS

	$\frac{\partial f}{\partial x}$	$\frac{\partial f}{\partial y}$	
Roberts	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	Good localization Noise Sensitive Poor Detection
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	
Sobel (3x3)	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	
Sobel (5x5)	$\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 5 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & -5 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \end{bmatrix}$	Poor localization Less Noise Sensitive Better Detection

Directional Edge Operators

- Kirsch 8-direction masks

$$k_0 = \begin{matrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{matrix} \qquad k_1 = \begin{matrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{matrix}$$

$$k_2 = \begin{matrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{matrix} \qquad k_7 = \begin{matrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{matrix}$$

- Gradient magnitude = $\max\limits_{n=0,\dots 7} k_n$
- Gradient direction = $45°$ argmax k_n

30

Directional Edge Operators

- Robinson masks

$$r_0 = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad r_7 = \begin{matrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{matrix}$$

- Nevatia and Babu used 6 5x5 masks, detecting orientations at multiples of 30°

31

Edge Detection in 2D

- How do we combine the directional derivatives to compute the gradient magnitude?
  - Use the root mean square (RMS) as in the continuous case
  - Sum of the absolute values of the directional derivatives
  - Maximum of the absolute values of the directional derivatives
- Advantages of the latter
  - Avoids computing square root (although this can be done using table lookups)
  - Keeps result in the same range as the original image
  - Gives a magnitude that is invariant with respect to orientation of the edge in the image (Δ_{sobel_X} overestimates diagonal edges and Δ_+ overestimates horizontal and vertical edge magnitudes)

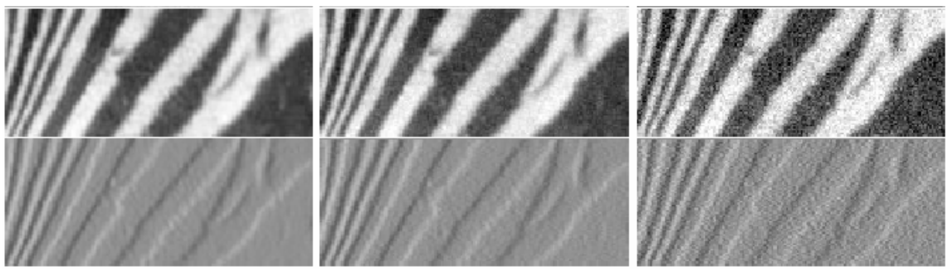
32

Finite Differences and Noise

- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbors
- Generally, the larger the noise the stronger the response
- What is to be done?
  - intuitively, most pixels in images look quite a lot like their neighbors
  - this is true even at an edge; along the edge they're similar, across the edge they're not
  - suggests that smoothing the image should help, by forcing pixels different from their neighbors (=noise pixels?) to look more like their neighbors

33

Finite Differences Responding to Noise



Increasing noise ->
(this is zero mean additive gaussian noise)

34

Combining Smoothing and Differentiation - Fixed Scale

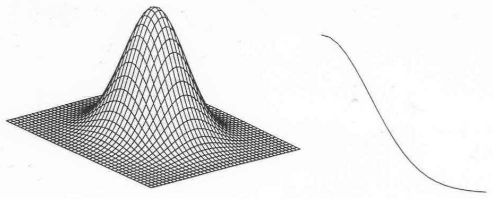
- Local operators like the Roberts operator give high responses to any intensity variation
  - local surface texture
  - noise from the sensing process
- If the picture is first smoothed by an averaging process, then these local variations are removed and what remains are the "prominent" edges
- Example: $I_{2x2}(u,v) = 1/4[I(u,v) + I(u+1,v) + I(u,v+1) + I(u+1,v+1)]$

35

Smoothing - Basic Problems

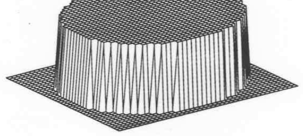
- What function should be used to smooth or average the image before differentiation?
  - **Mean** smoothing (*aka* box filters or uniform smoothing or averaging)
    - easy to compute (linear)
    - for large smoothing neighborhoods assigns too much weight to points far from an edge
  - **Median** smoothing
    - doesn't blur corners
    - unaffected by outliers
  - **Gaussian** smoothing
    - $[1/(2\pi\sigma^2)]\ e^{-(u^2 + v^2)/2\sigma^2}$

36



Gaussian

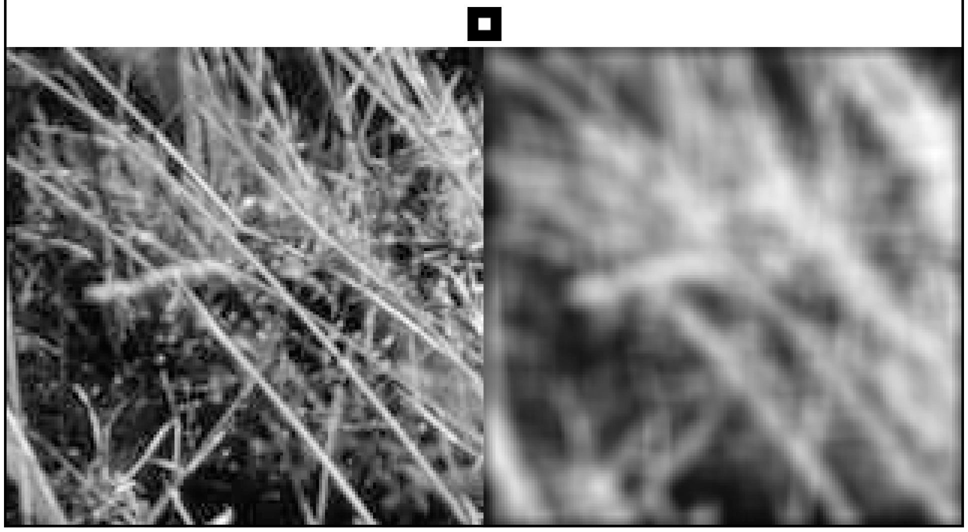
Figure 1: The two dimensional kernel on the left can be obtained by rotating the function $\gamma(r)$ on the right around a vertical axis through the maximum of the curve ($r = 0$).



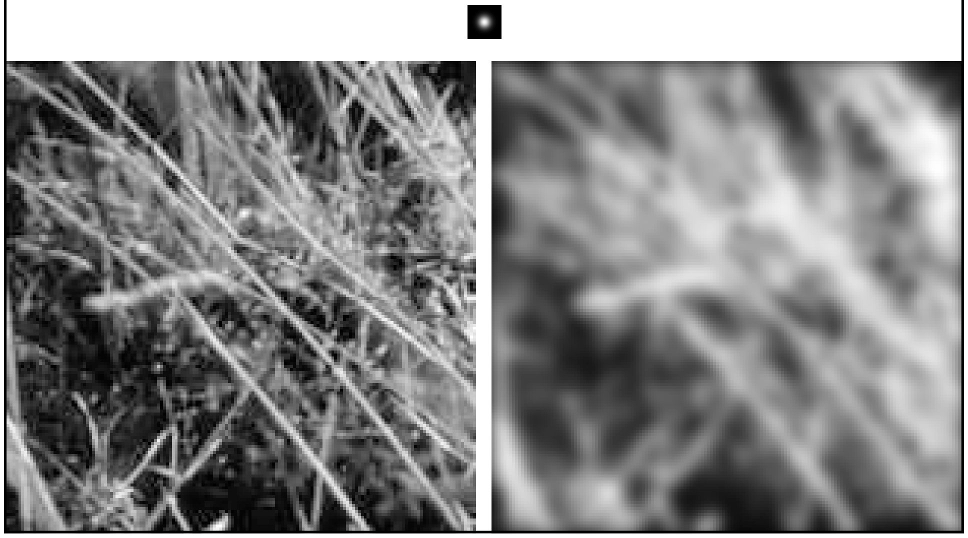
Mean

Figure 2: The pillbox function.

Example: Smoothing by Averaging

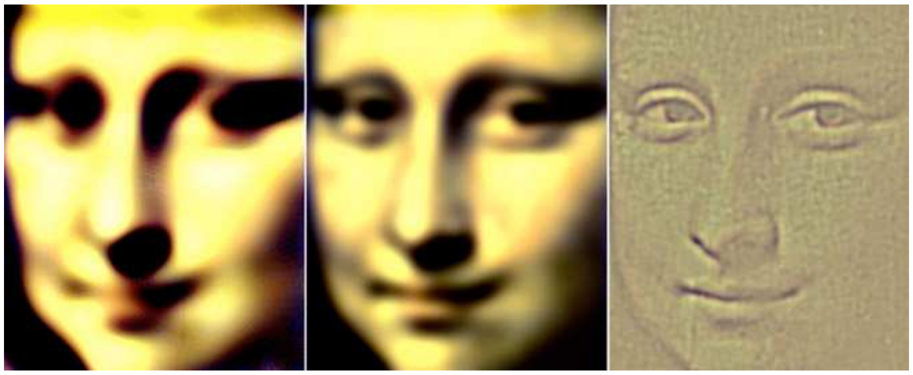


Smoothing with a Gaussian



The Mystery of the Mona Lisa Smile

- Margaret Livingstone has conjectured that the elusive smile is because of the differences in resolution in the fovea and the periphery of the retina



Low-pass filtered Middle-pass High-pass

40

Changing small details can affect the mood we perceive. A few pixels of change switches her from happy to sad. Adding random “noise” also changes our perception of her expression.

41

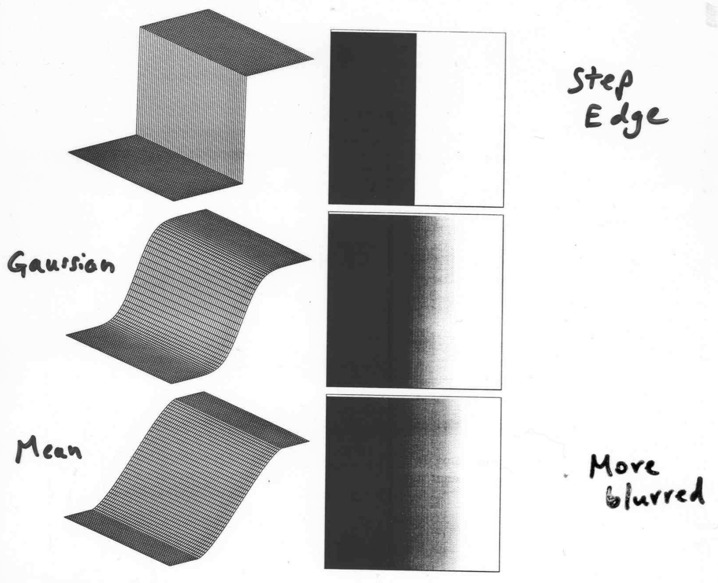


Figure 3: Intensity graphs (left) and images (right) of a vertical step function (top), and of the same step function smoothed with a Gaussian (middle), and with a pillbox function (bottom). Gaussian and pillbox have the same support and the same integral.

Smoothing and Convolution

- The convolution of two functions, $f(x)$ and $g(x)$, is defined as

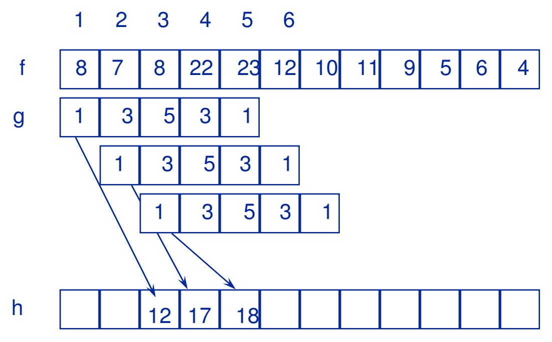
$$h(x) = \int g(x') f(x-x')\, dx' = g(x) * f(x)$$

- When f and g are discrete and when g is nonzero only over a finite range $[-n, n]$, this integral is replaced by the summation

$$h(u) = \sum g(i) f(u-i)$$

43

Example of 1D Convolution



Note: Values in h have been divided by $\sum g = 13$

$$h(4) = \sum_{i=-2}^{2} g(i) f(4-i)$$

$$= g(-2) f(6) + g(-1) f(5) + g(0) f(4) + g(1) f(3) + g(2) f(2)$$

44

Smoothing and Convolution

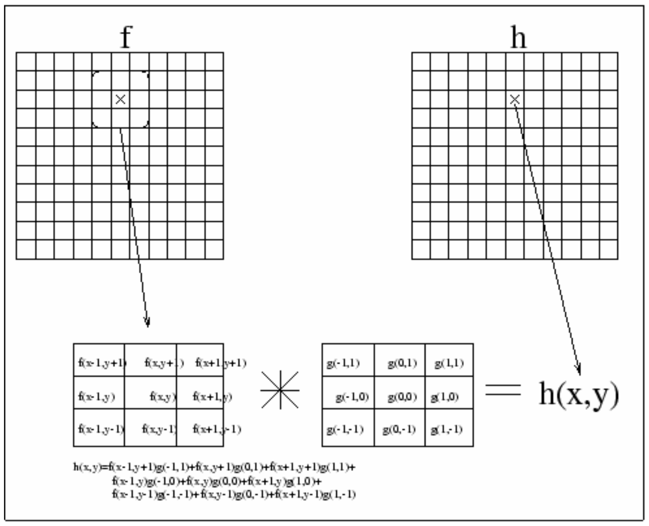
- These integrals and summations extend simply to functions of two variables:

$$h(u,v) = g(u,v) * f(u,v) = \sum\sum g(i,j)\, f(u-i, v-j)$$

- Convolution computes the weighted sum of the gray levels in each *n* x *n* neighborhood of the image, *f*, using the "kernel" matrix of weights, *g*
- Convolution is a **linear** operator because
  - $g*(af_1 + bf_2) = a(g*f_1) + b(g*f_2)$

45

Convolution



$$\begin{aligned}h(x,y) = {} & f(x-1,y+1)g(-1,1)+f(x,y+1)g(0,1)+f(x+1,y+1)g(1,1)+\\ & f(x-1,y)g(-1,0)+f(x,y)g(0,0)+f(x+1,y)g(1,0)+\\ & f(x-1,y-1)g(-1,-1)+f(x,y-1)g(0,-1)+f(x+1,y-1)g(1,-1)\end{aligned}$$

46

2D Convolution

$$h(5,5)=\sum_{i=-1}^{1}\sum_{j=-1}^{1} g(i,j)f(5-i,5-j)$$

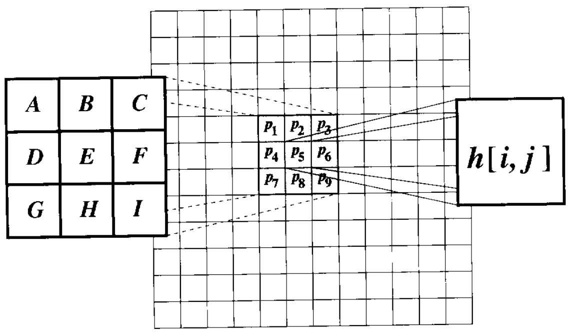
$$=g(-1,-1)f(6,6)+g(-1,0)f(6,5)+g(-1,1)f(6,4)$$

$$+g(0,-1)f(5,6)+g(0,0)f(5,5)+g(0,1)f(5,4)$$

$$+g(1,-1)f(4,6)+g(1,0)f(4,5)+g(1,1)f(4,4)$$

Smoothing and Convolution

4.2. LINEAR SYSTEMS *117*



$$h[i,j]=A\,p_1+B\,p_2+C\,p_3+D\,p_4+E\,p_5+F\,p_6+G\,p_7+H\,p_8+I\,p_9$$

Properties of Convolution

- Commutative
- Associative
- Cascadable
- Linear, shift-invariant
- Any linear, shift-invariant operation can be implemented as a convolution
- Convolution in spatial domain = multiplication in frequency domain. I.e., $g = f * h$ iff $G = F \cdot H$, where * denotes convolution and $F = \mathcal{F}(f)$
- An alternative for computing g: $g = \mathcal{F}^{-1}(F \cdot H)$

49

Combining Smoothing and Edge Detection

- Let $S = \begin{matrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{matrix}$
- $\Delta_x = -1\ 1$
- Smooth, then differentiate:

  $\Delta_x * (S * I)$, where *I* is original image and * is convolution

  $= (\Delta_x * S) * I$
- $\Delta_x * S = \begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} = \Delta_{\text{Prewitt\_x}}$

50

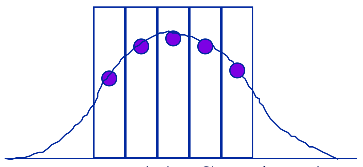
Back to Smoothing Functions

- To smooth an image using a Gaussian filter, $e^{-(u^2 + v^2)/2\sigma^2}$, we must
  - Choose an appropriate value for σ, which controls how quickly the Gaussian falls to near 0
    - Small σ produces filter which drops to near 0 quickly - can be implemented using small digital array of weights
    - Large σ produces a filter which drops to near 0 slowly - will be implemented using a larger size digital array of weights
  - Determine the size weight array needed to adequately represent that Gaussian
    - To represent 98.76% of the area under the Gaussian using $\sigma=\sigma_0$ use a mask of size $5\sigma_0$ x $5\sigma_0$
    - In practice, size of mask is often closer to size 3σ x 3σ
    - Weight array needs to be of odd size ($2k$+1 x $2k$+1) to allow for symmetry

51

Gaussian Smoothing

- To smooth an image using a Gaussian filter we must
  - Sample the Gaussian by integrating it over the square pixels of the array of weights and multiplying by a scale factor to obtain integer weights



  - Because we have truncated the Gaussian, the weights will not sum to 1.0 x scale factor
    - In "flat" areas of the image we expect our smoothing filter to leave the image unchanged
    - But if the filter weights do not sum to 1.0 x scale factor, it will either amplify (> 1.0) or de-amplify (< 1.0) the image

52

Gaussian Smoothing

- Gaussian smoothing steps
  - Normalize the weight array by dividing each entry by the sum of the all of the entries (integral equal to 1)
  - Convert to integers
- Advantages of Gaussian filtering
  - Rotationally symmetric (for large filters)
  - Filter weights decrease monotonically from central peak, giving most weight to central pixels
  - Simple and intuitive relationship between size of σ and size of objects whose edges will be detected in image.
  - The Gaussian is separable

53

Building Discrete Gaussian Kernels

- Given σ and w, compute real-valued weights for w x w kernel array
- Divide all entries by minimum weight and round result to nearest integer, obtaining kernel G
- Compute normalization factor $n = \sum$ weights
- $I' = 1/n \ (G * I)$

54

Separability

- A function $g(x, y)$ is **separable** if $g(x, y) = g_1(x)\, g_2(y)$
- The Gaussian function is separable:

$$e^{-[(x^2 + y^2)/2\sigma^2]} = e^{-[x^2/2\sigma^2]} * e^{-[y^2/2\sigma^2]}$$

- First convolve the image with a 1D horizontal filter
- Then convolve the result of the first convolution with a 1D vertical filter
- For a k x k Gaussian filter, 2D convolution requires k^2 multiplications and k^2-1 additions per pixel
- But using the separable filters, we reduce this to 2k multiplications and 2k-1 additions per pixel

55

Separability

$$I = \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

$$g_1 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} \qquad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

$$g_2 = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} \qquad \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

=2 + 6 + 3 = 11
= 6 + 20 + 10 = 36
= 4 + 8 + 6 = 18

65

56

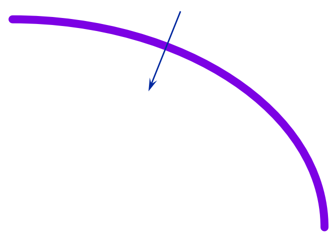
Advantages of Gaussians

- Cascading Gaussians: Convolution of a Gaussian with itself is another Gaussian
  - So, we can first smooth an image with a small Gaussian
  - Then, we convolve that smoothed image with another small Gaussian and the result is equivalent to smoother the original image with a larger Gaussian
  - If we smooth an image with a Gaussian having standard deviation σ twice, then we get the same result as smoothing the image with a Gaussian having standard deviation $\sqrt{2}\sigma$

57

Combining Smoothing and Differentiation - Fixed Scale

- Non-maxima suppression - Retain a point as an edge point if
  - its gradient magnitude is higher than a threshold
  - its gradient magnitude is a local maximum in the gradient direction
- "*Lateral inhibition*" process



58

Summary of Basic Edge Detection Steps

- Smooth the image to reduce the effects of local intensity variations
  - Choice of smoothing operator practically important
- Differentiate the smoothed image using a digital gradient operator that assigns a magnitude and direction of the gradient at each pixel
  - Mathematically, we can apply the digital gradient operator to the digital smoothing filter, and then just convolve the differentiated smoothing filter to the image
  - Requires using a slightly larger smoothing array to avoid border effects
- Threshold the gradient magnitude to eliminate low contrast edges

59

Summary of Basic Edge Detection Steps

- Apply a non-maximum suppression step to thin the edges to single pixel wide edges
  - Smoothing will produce an image in which the contrast at an edge is spread out in the neighborhood of the edge
  - Thresholding operation will produce thick edges

60

Canny Edge Operator

- Design operator that is optimal with respect to good detection, good localization, and unique response criteria
- **Good detection criterion**: Maximize SNR, which is the ratio of the step size of a step edge over the standard deviation of the 0-mean, additive Gaussian noise
- **Good localization criterion**: Maximize inverse distance of detected edge from true edge
- Optimal 1D step-edge detector = detector that maximizes the product of the two terms above subject to third criterion
- First derivative of a Gaussian is within 20% of optimal

61

Directional Derivatives

- Isotropic operators smooth **across** edges

  $\Rightarrow$ Poor localization

- Instead, average pixels **along** edge direction only

  $\Rightarrow$ Use a set of directional derivatives at a large number of orientations

$$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix} \qquad \text{etc.}$$

62

Canny Edge Operator

- Gaussian smooth image: $G * I$
- Estimate gradient direction (i.e., edge normal), **n**, at each pixel:

$$\mathbf{n} = \frac{\nabla(G * I)}{\left\|\nabla(G * I)\right\|}$$

- Compute first derivative in the gradient direction, **n**. That is, $\partial G/\partial \mathbf{n} * I$
- Suppress non-maxima
  - If value is not a local max in direction **n,** then change value to 0
  - Or, find zero-crossings in direction **n**. I.e., zero-crossings of $\partial([\partial G/\partial \mathbf{n}] * I)/\partial \mathbf{n} = \partial^2(G * I)/\partial \mathbf{n}^2$

63

Canny Edge Operator

- To get rid of spurious edges created by noise (false positives) and to fill in missing edges (false negatives) use **hysteresis thresholding**:
  - Define two thresholds, t_{high} and t_{low}
  - Mark all pixels with edge magnitude > t_{high} as edge pixels
  - For each marked pixel, search in edge direction, both ways, marking all pixels with edge magnitude > t_{low} as edge pixels. Stop search when a pixel is reached with edge magnitude < t_{low}

64

Laplacian Edge Detectors

- Directional second derivative in direction of gradient has a zero crossing at gradient maximum
- Can approximate directional second derivative with Laplacian:

  $\nabla^2 I = \partial^2 I/\partial x^2 + \partial^2 I/\partial y^2$
- Laplacian is lowest order linear isotropic operator
- Digital approximation (2nd forward difference) is
  - $\nabla^2 I(u,v) = [I(u+1,v) + I(u-1,v) + I(u,v+1) + I(u,v-1)] - 4I(u,v)$

    $= [I(u+1,v) - I(u,v)] - [I(u,v) - I(u-1,v)] + [I(u,v+1) - I(u,v)] - [I(u,v) - I(u,v-1)]$

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

65

Laplacian Examples

- I = ... 2 2 2 8 8 8 ...
- ∇^2 = 1 -2 1

  $\Rightarrow \nabla^2 I$ = ... 0 0 0 6 -6 0 0 0 ...

- I = ... 2 2 2 5 8 8 8 ...

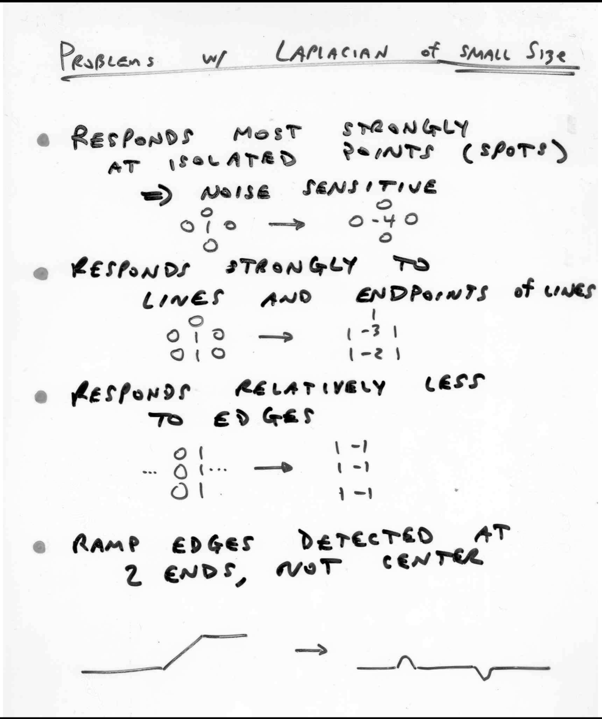
  $\Rightarrow \nabla^2 I$ = ... 0 0 0 3 0 -3 0 0 0 ...

66

Problems with Laplacian of Small Size

- Responds most strongly at isolated points (spots)
  - ⇒ noise sensitive
- Responds strongly at lines and endpoints of lines
- Responds relatively less at edges
- Ramp edges detected at two ends, not once at center

67



PROBLEMS W/ LAPLACIAN of SMALL SIZE

- RESPONDS MOST STRONGLY AT ISOLATED POINTS (SPOTS)
  - ⇒ NOISE SENSITIVE

$$\begin{array}{ccc} & 0 & \\ 0 & 1 & 0 \\ & 0 & \end{array} \longrightarrow \begin{array}{ccc} & 0 & \\ 0 & -4 & 0 \\ & 0 & \end{array}$$

- RESPONDS STRONGLY TO LINES AND ENDPOINTS of LINES

$$\begin{array}{ccc} & 0 & \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{array} \longrightarrow \begin{array}{ccc} & 1 & \\ 1 & -3 & 1 \\ 1 & -2 & 1 \end{array}$$

- RESPONDS RELATIVELY LESS TO EDGES

$$\begin{array}{cccc} & 0 & 1 & \\ \cdots & 0 & 1 & \cdots \\ & 0 & 1 & \end{array} \longrightarrow \begin{array}{cc} 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{array}$$

- RAMP EDGES DETECTED AT 2 ENDS, NOT CENTER

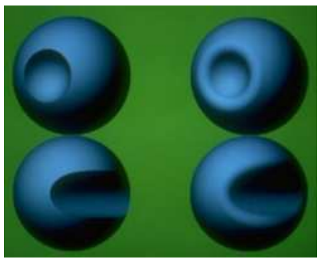
Using Laplacian for Edge Enhancement

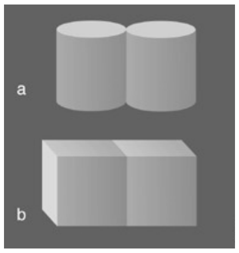
- Given a blurred image $g \approx I' = 1/5 \left(I * \begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{vmatrix} \right)$
- $\nabla^2 I \propto I - I'$
- $g - k\nabla^2 g \approx I' - k(-1/5\,(I - I'))$
  $= k/5\, I + I'(1 - k/5)$
  $\approx I$ when $k=5$
- ==> Can deblur an image by subtracting a multiple of its Laplacian
- Simulates **Mach Band** effect in human visual system
- Analogous to **unsharp masking** technique in photography

69

Mach Bands

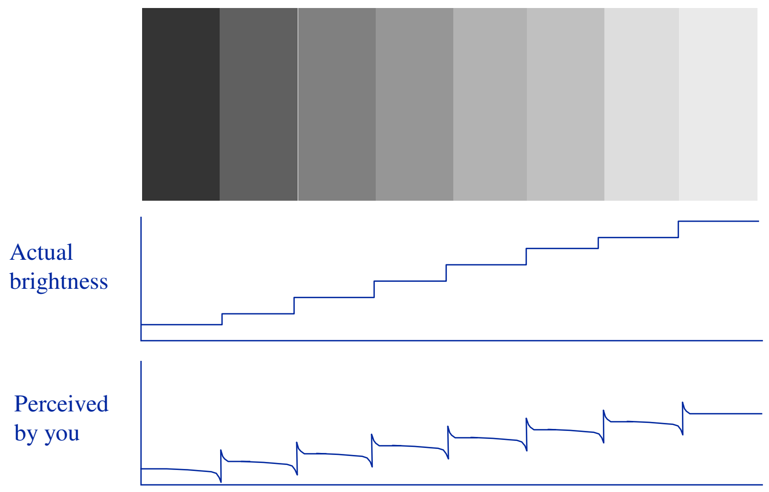
- In 1865, Mach discovered that we **perceive** an edge (**change of intensity**) at lines that separate surface regions with identical intensity on both sides but different intensity **derivatives**
- It is caused by the lateral inhibition of the receptors in the eyes





70

Mach Bands

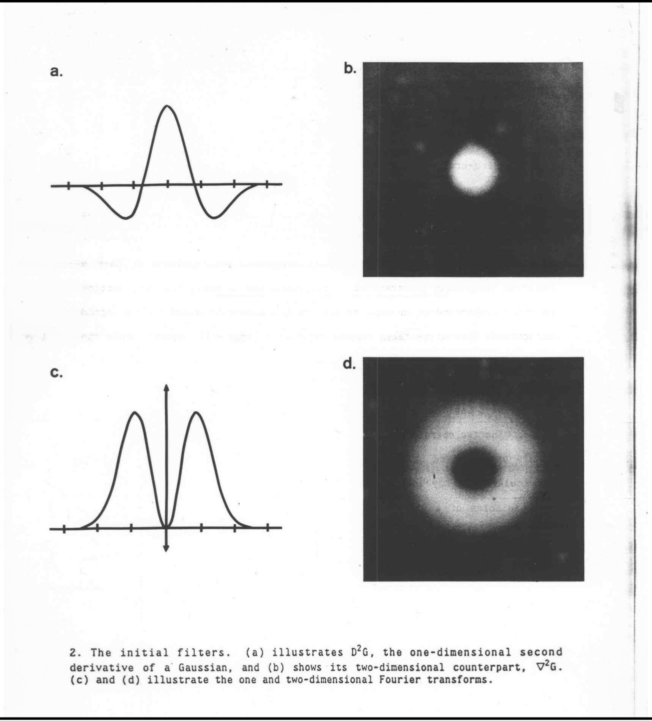


71

Laplacian Edge Detectors

- Laplacians are also combined with smoothing for edge detectors
  - Take the Laplacian of a Gaussian-smoothed image - called the Laplacian-of-Gaussian (**LoG**), Mexican Hat operator, Difference-of-Gaussians (**DoG**), Marr-Hildreth, $\boldsymbol{\nabla^2 G}$ operator
  - Locate the zero-crossing of the operator
    - these are pixels whose LoG is positive and which have neighbor's whose LoG is negative or zero
  - Usually, measure the gradient or directional first derivatives at these points to eliminate low contrast edges

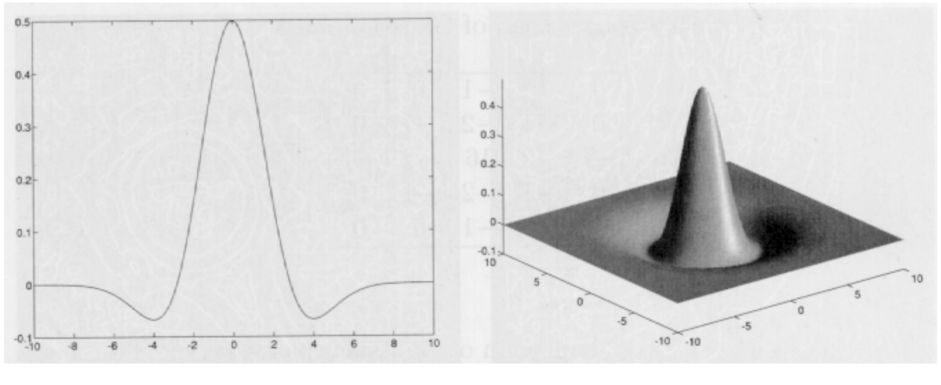
72



2. The initial filters. (a) illustrates D^2G, the one-dimensional second derivative of a Gaussian, and (b) shows its two-dimensional counterpart, $\nabla^2 G$. (c) and (d) illustrate the one and two-dimensional Fourier transforms.

Laplacian of Gaussian (LoG)

$$\nabla^2 G_\sigma(x, y) = -[1/2\pi\sigma^4]\,(2 - (x^2 + y^2)/\sigma^2)\, e^{-(x^2 + y^2)/2\sigma^2}$$



74

LoG Properties

- Linear, shift invariant $\Rightarrow$ convolution
- Circularly symmetric $\Rightarrow$ isotropic
- Size of LoG operator approximately 6σ x 6σ
- LoG is separable
- LoG $\approx G_{\sigma_1} - G_{\sigma_2}$, where $\sigma_1 = 1.6\sigma_2$

- Analogous to spatial organization of receptive fields of retinal ganglion cells, with a central excitatory region and an inhibitory surround

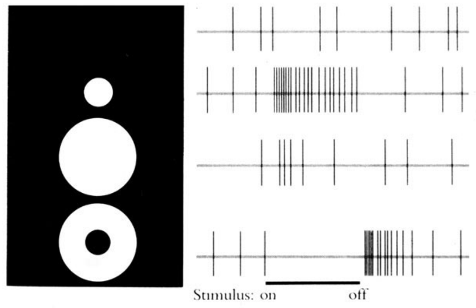
75

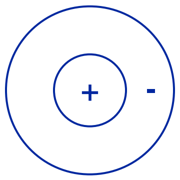
Zero-Crossing Detection

```
if $((\nabla^2 G(p) < -T) \wedge$
        $(\exists$ 8-neighbor $q \ni \nabla^2 G(q) > T))$
    $\vee ((\nabla^2 G(p) > T) \wedge$
        $(\exists q \ni \nabla^2 G(q) < -T))$
then  mark p as an edge point
```

76

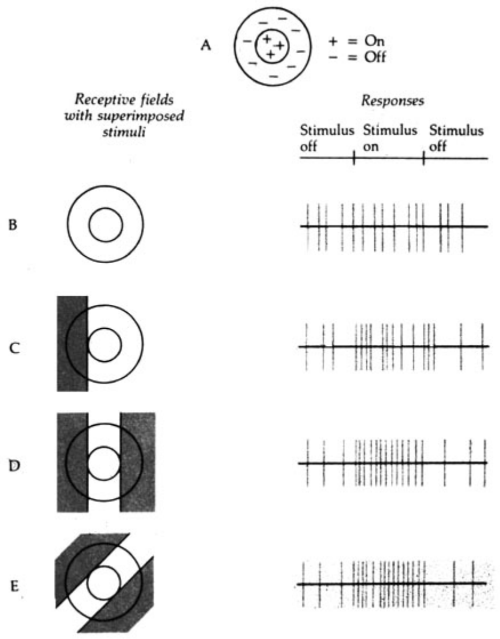
Receptive Field





- Region on retina that influences activity of a ganglion cell
- Firing rate of neuron
- Excitatory center, inhibitory surround

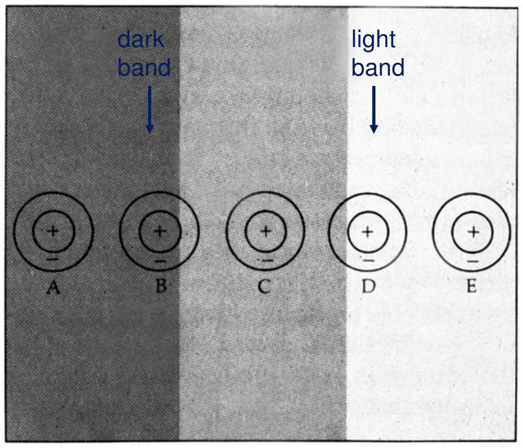
77



- Ganglion cells detect *differences* in luminance at a border, line, or spot
- Retina is ***NOT*** a spot-meter

78

Lateral Inhibition Explanation

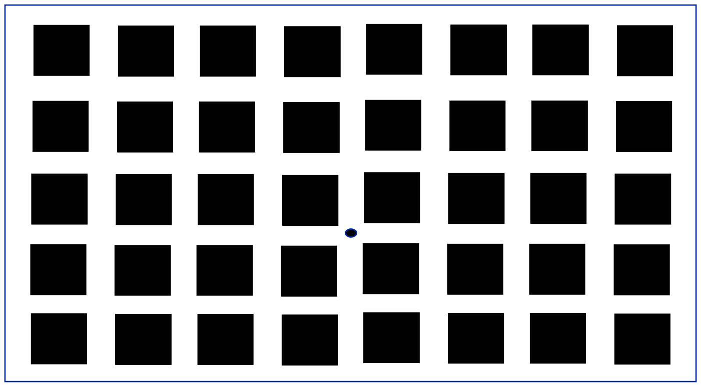


B < A
B inhibited by surround

D > E
D less inhibited by surround

79

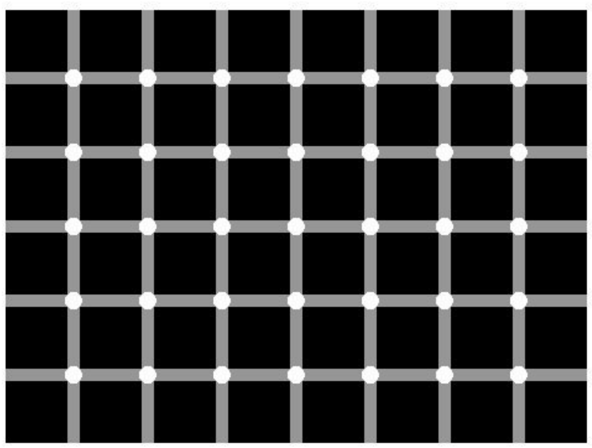
Hermann Grid



- Illustrates lateral inhibition

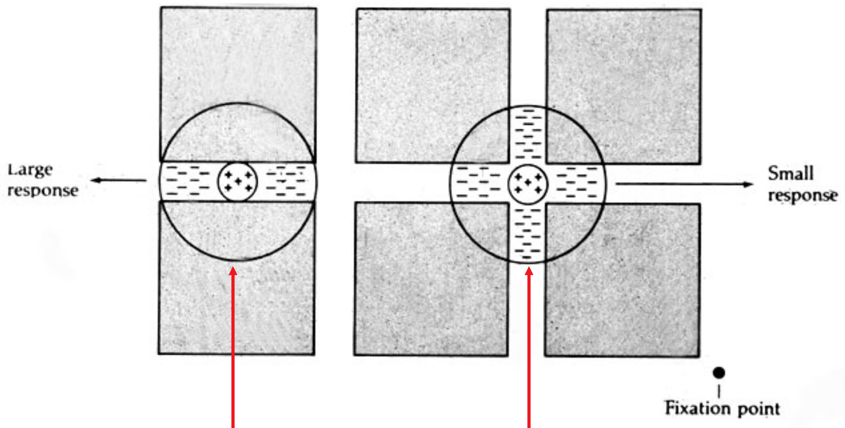
80

Visual illusions



81

Lateral Inhibition Explanation



Inhibited less by white stripes

Inhibited more by white stripes

82

The Scale Space Problem

- Usually, any single choice of σ does not produce a good edge map
  - a large σ will produce edges from only the largest objects, and they will not accurately delineate the object because the smoothing reduces shape detail
  - a small σ will produce many edges and very jagged boundaries of many objects
- Scale-space approaches
  - detect edges at a range of scales [σ_1, σ_2]
  - combine the resulting edge maps
    - trace edges detected using large σ down through scale space to obtain more accurate spatial localization

83

10

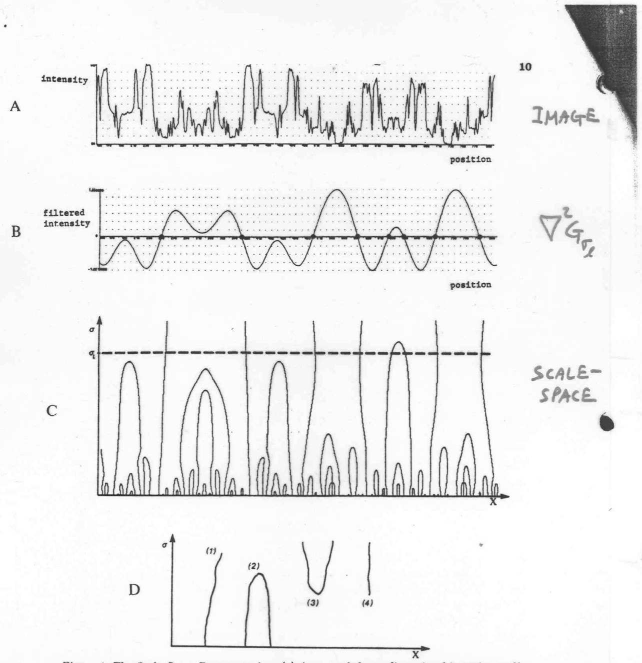
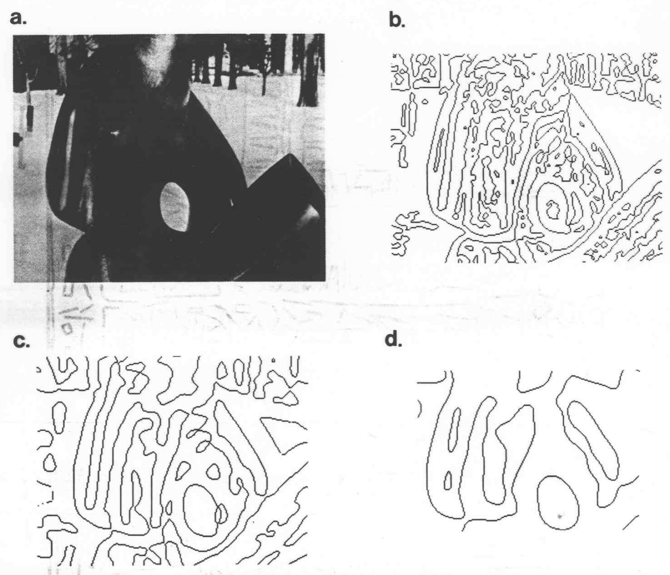
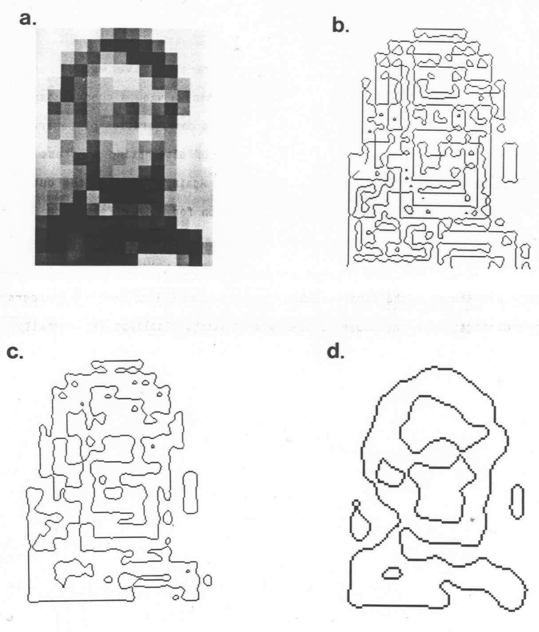
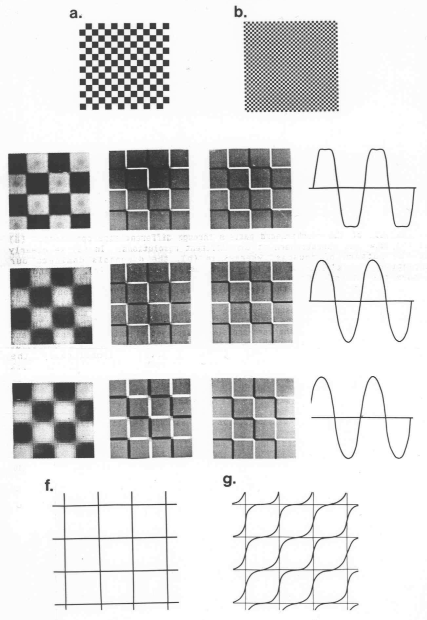


Figure 4. The Scale-Space Representation. (a) An extended one-dimensional intensity profile. (b) The result of convolving the profile in (a) with a $G''(x)$ operator with large σ. The zero-crossings are marked with heavy dots. (c) The scale-space representation of the positions of



6. Zero-crossings from different size operators. The image in (a) has been





(24)

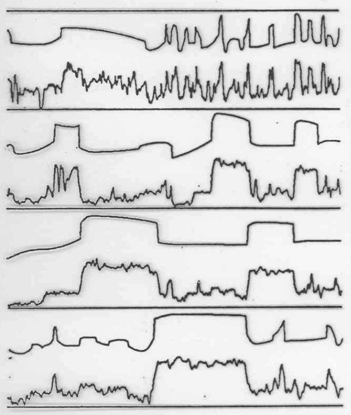


Figure 5. Several signals, with their maximum-stability descriptions. These are "top-level" descriptions, generated automatically and without thresholds. You should compare the descriptions to your own first-glance "top-level" percepts. (the noisy sine and square waves are synthetic signals.)

tend to leap out at the eye, while the most ephemeral are

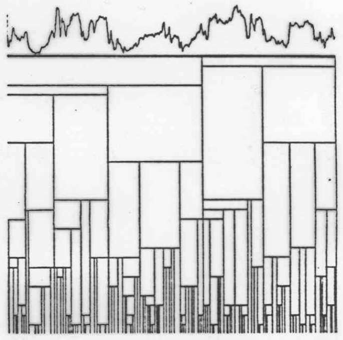


Figure 4. A signal with its interval tree, represented as a rectangular tesselation of scale-space. Each rectangle is a node, indicating an interval on the signal, and the scale interval over which the signal interval exists.

tesselate the (x, σ)-plane. See fig. 4 for an illustration of the tree.

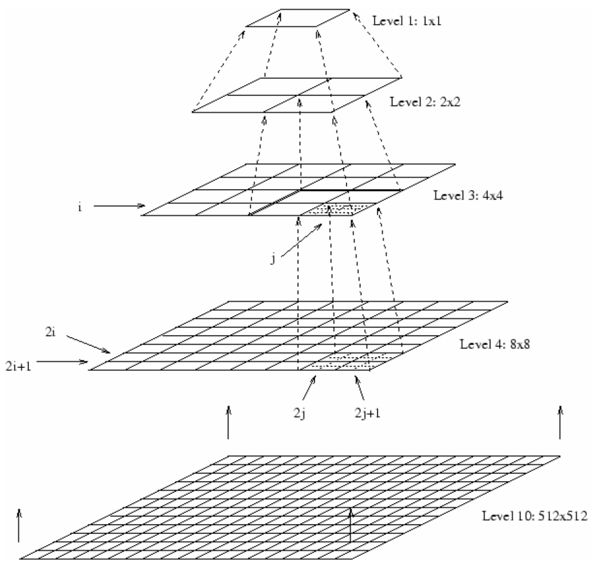
This *interval tree* may be viewed in two ways: as describing the signal simultaneously at all scales, or as generating a family of single-scale descriptions, each defined by a subset of nodes in the tree that cover the x-axis. On the second interpretation, one may move through the famil-

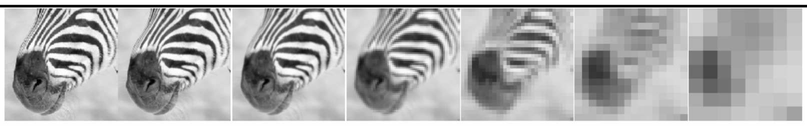
Pyramids

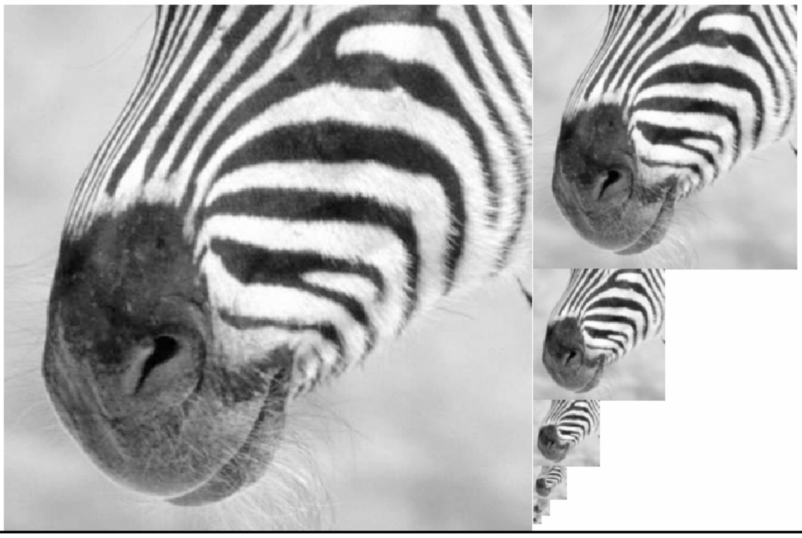
- Very useful for representing images
- Pyramid is built by using multiple copies of image
- Each level in the pyramid is 1/4 the area of previous level, i.e., each dimension is 1/2 resolution of previous level

89

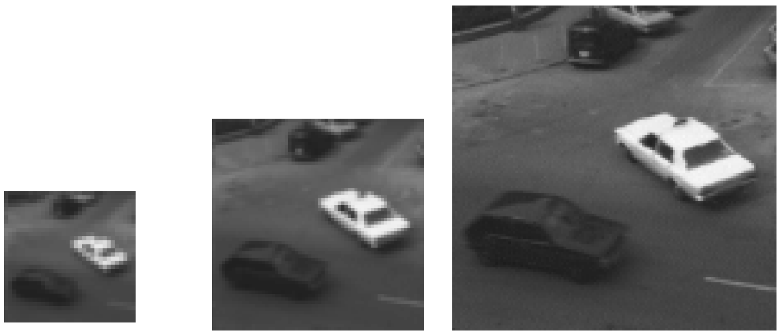
Pyramids







Gaussian Pyramid



Gaussian Pyramid

- Multiresolution, low-pass filter
- Hierarchical convolution
  - G0 = input image
  - $G'_k(u, v) = \sum\sum w(m, n)\, G_{k-1}(u-m, v-n)$; smooth
  - $G_k(u, v) = G'_k(2u, 2v)$, $0 < u, v < 2^{N-k}$; sub-sample
- *w* is a small (e.g., 5 x 5) separable generating kernel, e.g., 1/16 [1 4 6 4 1]
- Cascading *w* equivalent to applying large kernel
  - Effective size of kernel at level k = $2M(2^k - 1) + 1$, where *w* has width 2M+1
  - Example: Let M=1. If k=1 then equivalent size = 5; k=2 then equivalent size = 13; k=3 then equivalent size = 27

93

Gaussian Pyramids

$$g_l(u,v) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)\, g_{l-1}(2u+m, 2v+n)$$

$$g_l = REDUCE\ [g_{l-1}]$$

94

Reduce (1D)

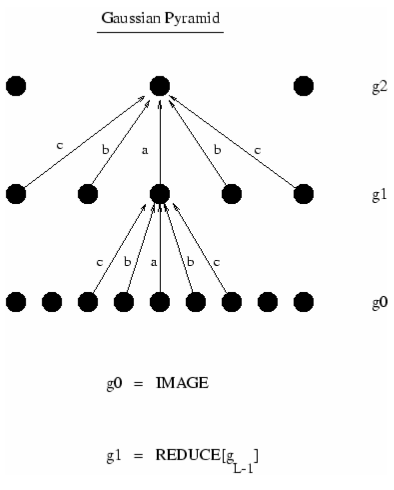
$$g_l(u) = \sum_{m=-2}^{2} \hat{w}(m) g_{l-1}(2u+m)$$

$$g_l(2) = \hat{w}(-2) g_{l-1}(4-2) + \hat{w}(-1) g_{l-1}(4-1) + \hat{w}(0) g_{l-1}(4) + \hat{w}(1) g_{l-1}(4+1) + \hat{w}(2) g_{l-1}(4+2)$$

$$g_l(2) = \hat{w}(-2) g_{l-1}(2) + \hat{w}(-1) g_{l-1}(3) + \hat{w}(0) g_{l-1}(4) + \hat{w}(1) g_{l-1}(5) + \hat{w}(2) g_{l-1}(6)$$

95

Reduce



96

Gaussian Pyramids

$$g_{l,n}(u,v) = \sum_{p=-2}^{2}\sum_{q=-2}^{2} w(p,q)\, g_{l,n-1}\left(\frac{u-p}{2}, \frac{v-q}{2}\right)$$

$$g_{l,n} = EXPAND[g_{l,n-1}]$$

97

Expand (1D)

$$g_{l,n}(u) = \sum_{p=-2}^{2} \hat{w}(p)\, g_{l,n-1}\left(\frac{u-p}{2}\right)$$

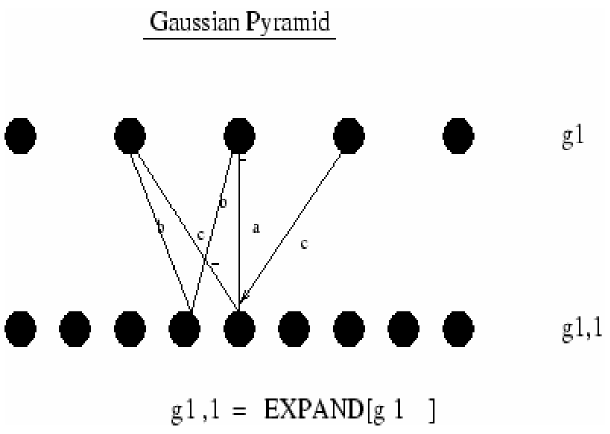
$$g_{l,n}(4) = \hat{w}(-2)\, g_{l,n-1}\left(\frac{4-2}{2}\right) + \hat{w}(-1)\, g_{l,n-1}\left(\frac{4-1}{2}\right) +$$

$$\hat{w}(0)\, g_{l,n-1}\left(\frac{4}{2}\right) + \hat{w}(1)\, g_{l,n-1}\left(\frac{4+1}{1}\right) + \hat{w}(2)\, g_{l,n-1}\left(\frac{4+2}{2}\right)$$

$$g_{l,n}(4) = \hat{w}(-2)\, g_{l,n-1}(1) + \hat{w}(0)\, g_{l,n-1}(2) + \hat{w}(2)\, g_{l,n-1}(3)$$

98

Expand



99

Convolution Mask

$$[w(-2), w(-1), w(0), w(1), w(2)]$$

100

Convolution Mask

- Separable

$$w(m,n) = \hat{w}(m)\hat{w}(n)$$

- Symmetric

$$\hat{w}(u) = \hat{w}(-u)$$

$$[c, b, a, b, c]$$

101

Convolution Mask

- The sum of mask should be 1:

$$a + 2b + 2c = 1$$

- All nodes at a given level must contribute the same total weight to the nodes at the next higher level:

$$a + 2c = 2b$$

102

Convolution Mask

$$\hat{w}(0) = a$$

$$\hat{w}(-1) = \hat{w}(1) = \frac{1}{4}$$

$$\hat{w}(-2) = \hat{w}(2) = \frac{1}{4} - \frac{a}{2}$$

a=.4 $\Rightarrow$ Gaussian   a=.5 $\Rightarrow$ Triangular

Algorithm

- Apply 1D mask to alternate pixels along each row of image
- Apply 1D mask to each pixel along alternate columns of resultant image from previous step

Laplacian Pyramids

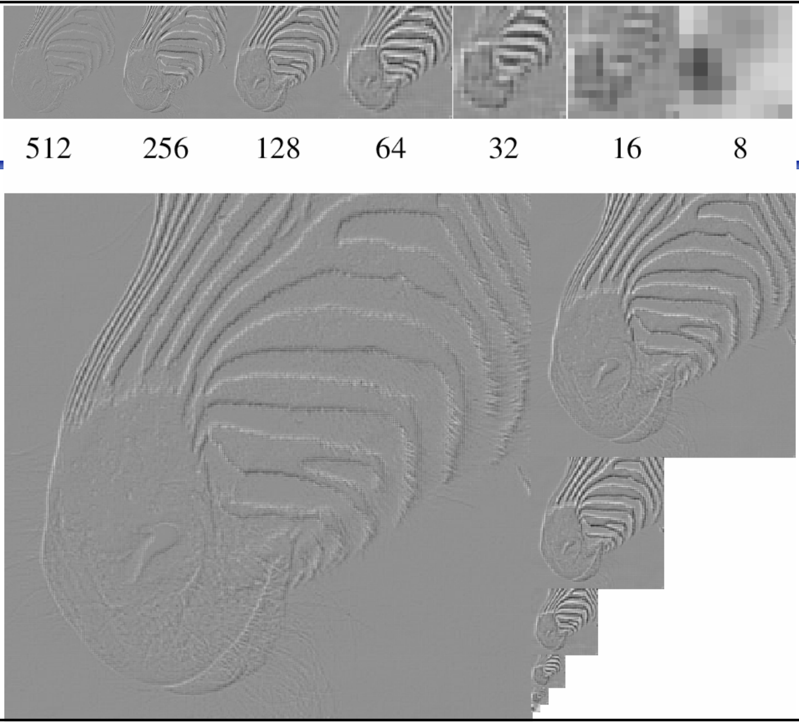
- Similar to results of edge detection
- Most pixels are 0
- Can be used for image compression:

$$L_1 = g_1 - EXPAND[g_2]$$

$$L_2 = g_2 - EXPAND[g_3]$$

$$L_3 = g_3 - EXPAND[g_4]$$

105



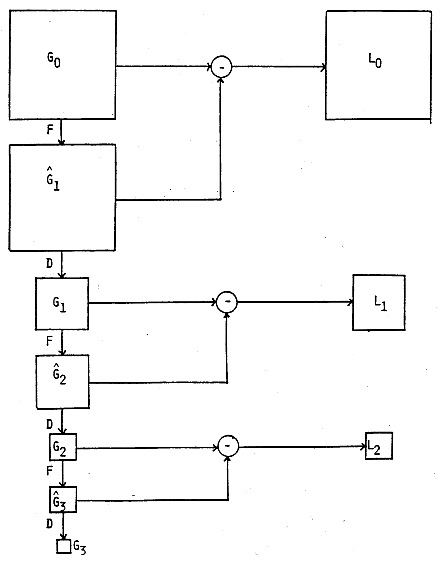
106

Laplacian Pyramid

- Computes a set of bandpass filtered versions of image
- $L_k = G_k - w * G_k$
  $= G_k - \text{Expand}(G_{k+1})$
- $L_N = G_N$ (apex of Laplacian pyr = apex of Gaussian pyr)
- Separates features by their scale (size)
- Enhances features
- Compact representation
- $\sum L_k = (G_0 - G_1) + (G_1 - G_2) + ... + (G_{N-1} - G_N) + G_N$
  $= G_0$

107

FSD Pyramid: (Filter, Subtract, Decimate)



Gaussian and Laplacian Pyramids

536

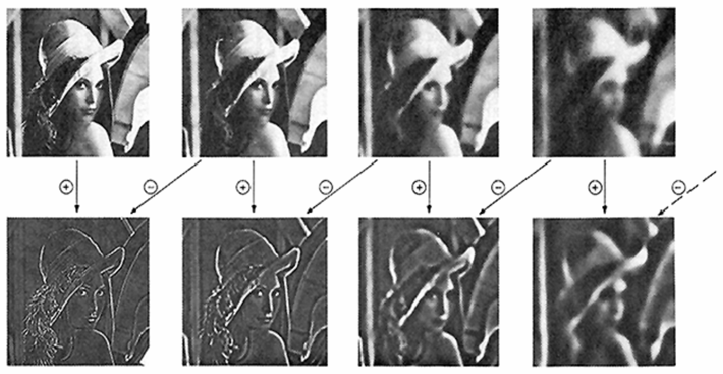


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4

109

Coding using the Laplacian Pyramid

- Compute Gaussian pyramid

$$g_1, g_2, g_3, g_4$$

- Compute Laplacian pyramid

$L_1 = g_1 - EXPAND[g_2]$

$L_2 = g_2 - EXPAND[g_3]$

$L_3 = g_3 - EXPAND[g_4]$

$L_4 = g_4$

- Code Laplacian pyramid

110

Decoding using Laplacian pyramid

- Decode Laplacian pyramid
- Compute Gaussian pyramid from Laplacian pyramid:

$$g_4 = L_4$$

$$g_3 = EXPAND[g_4] + L_3$$

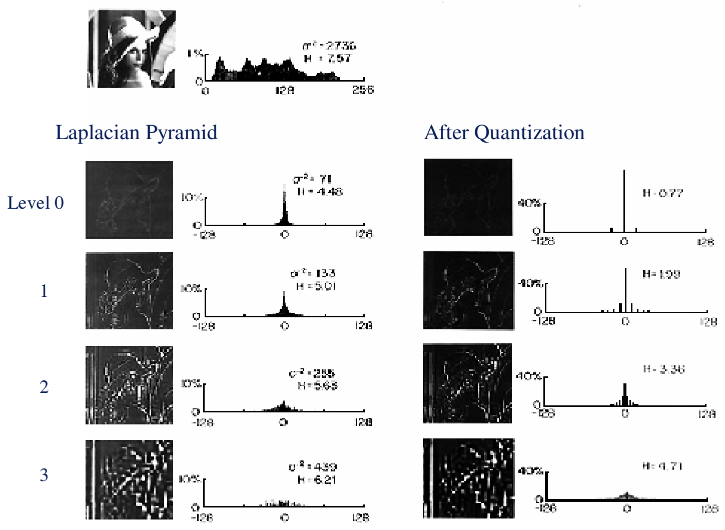
$$g_2 = EXPAND[g_3] + L_2$$

$$g_1 = EXPAND[g_2] + L_1$$

- g_1 is reconstructed image

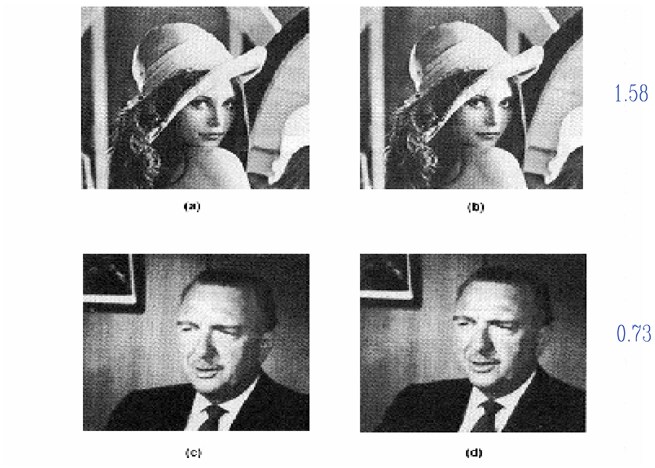
111

Image Compression



112

Image Compression



1.58 bits/pixel

0.73 bits/pixel

Originals Encoded

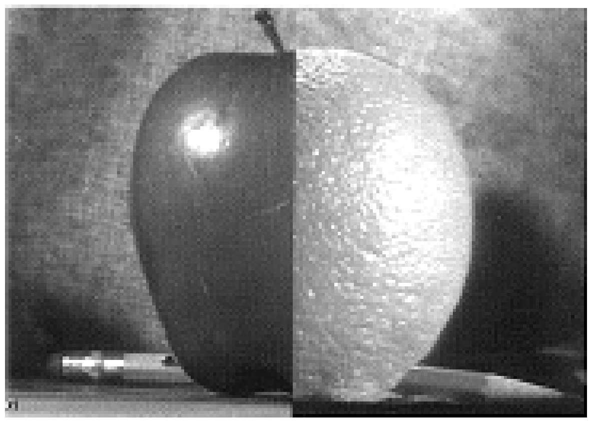
113

Image Compositing by Pyramid Blending

- **Given**: Two 2^n x 2^n images
- **Goal**: Create an image that contains left half of image A and right half of image B
- **Algorithm**
  - Compute Laplacian pyramids, LA and LB, from images A and B
  - Compute Laplacian pyramid LS by copying left half of LA and right half of LB. Pyramid nodes down the center line = average of corresponding LA and LB nodes $\Rightarrow$ blend along center line
  - Expand and sum levels of LS to obtain output image S

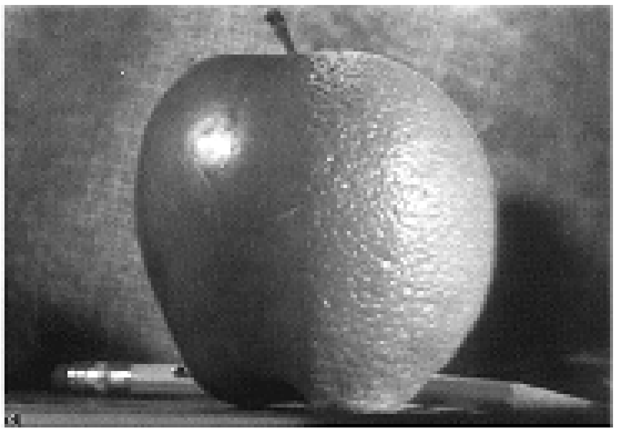
114

Combining Apple & Orange



115

Combining Apple & Orange (using Pyramids)

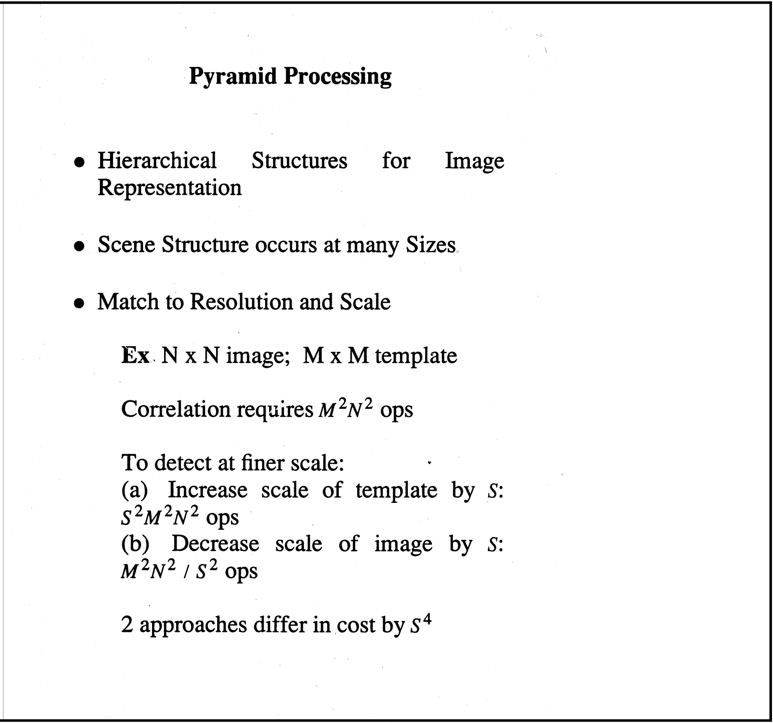


116

Image Compositing from Arbitrary Regions

- Given: Two 2^n x 2^n images and one 2^n x 2^n binary mask
- Goal: Output image containing image A where mask=1, and image B where mask=0
- Algorithm:
  - Construct Laplacian pyramids LA and LB from images A and B
  - Construct Gaussian pyramid GR from mask R
  - Construct Laplacian pyramid LS:
    $LS_k(u, v) = GR_k(u, v)\, LA_k(u, v) + (1 - GR_k(u, v))\, LB_k(u, v)$
  - Expand and sum levels of LS to obtain output image S

117



Pyramid Processing

- Hierarchical Structures for Image Representation
- Scene Structure occurs at many Sizes
- Match to Resolution and Scale

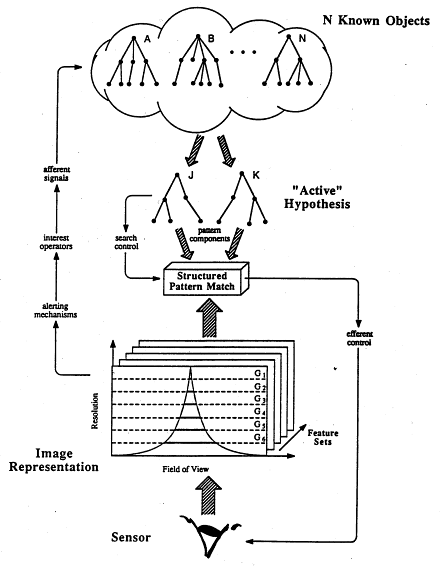
Ex N x N image; M x M template

Correlation requires M^2N^2 ops

To detect at finer scale:
(a) Increase scale of template by S: $S^2M^2N^2$ ops
(b) Decrease scale of image by S: M^2N^2 / S^2 ops

2 approaches differ in cost by S^4

MODEL-BASED OBJECT RECOGNITION



- Fast Feature Detection

  * Hierarchical, Fine-to-Coarse Operators

  Gaussian Pyramid
  Feature Pyramids (Texture, Motion)
  Image Moment Pyramids

  * Complex Features over large neighborhoods defined in terms of simpler features in small neighborhoods

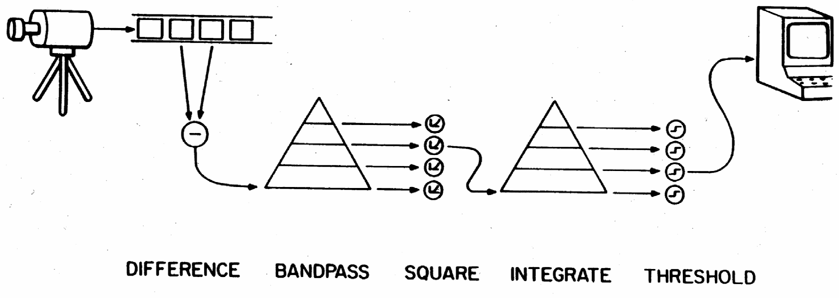
  * Hierarchical Grouping and Model-fitting

- Coarse-to-Fine Hierarchical Search

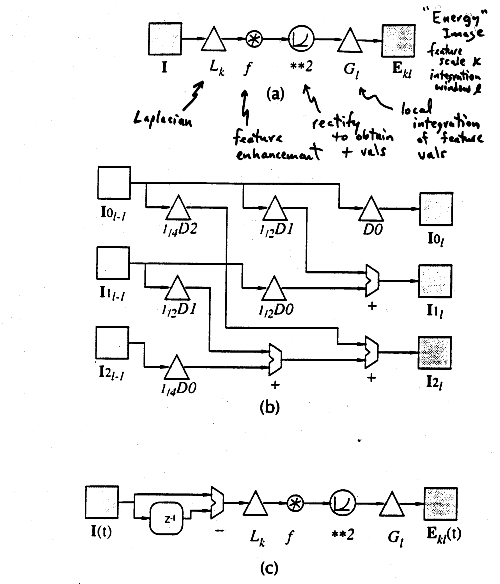
  * Selectively process only relevant areas and scales

  * Iterative refinement

  * Variable resolution analysis



Computing change energy within selected frequency

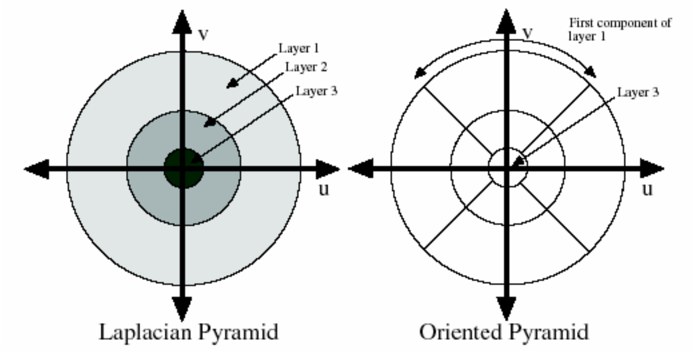


Basic algorithms for generating image feature sets. (a) Texture measure. (b) Moments. (c) Change energy measure.

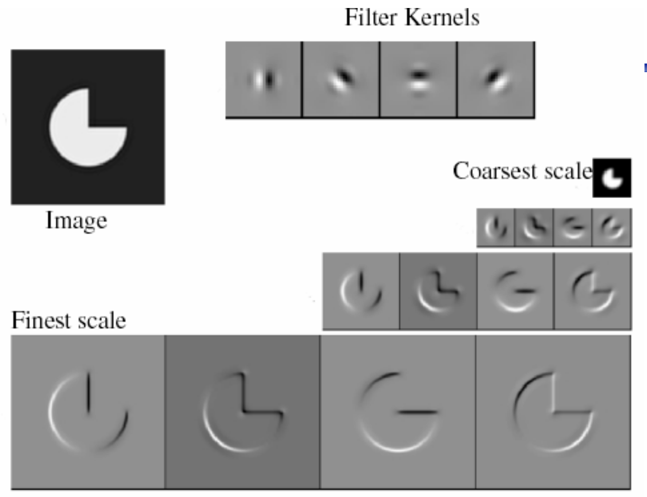
Oriented ("Steerable") Pyramids

- Laplacian pyramid is orientation-independent
- Apply an oriented filter to determine orientations at each layer
  - by clever filter design, we can simplify synthesis
  - this represents image information at a particular scale and orientation

124



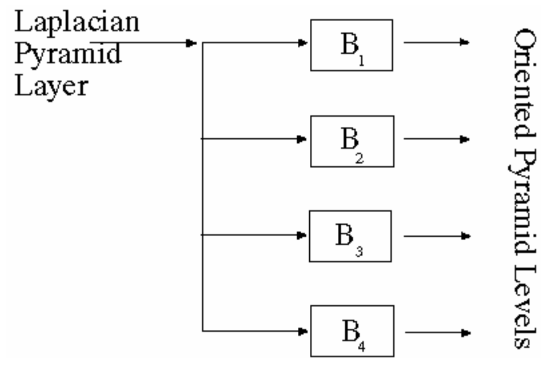
125



From “Shiftable MultiScale Transforms,” by Simoncelli *et al.*, *IEEE Transactions on Information Theory*, 1992

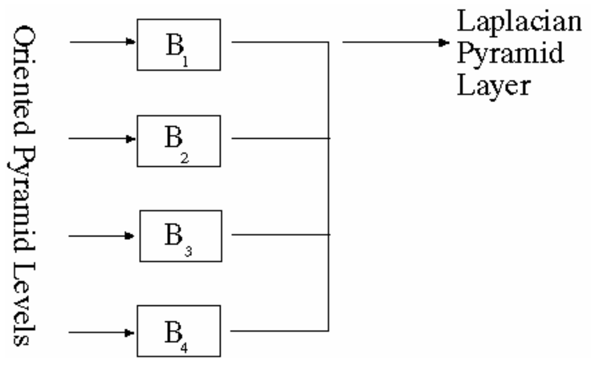
126

Analysis



127

Synthesis



128

Edge Detection by Function Fitting: Facet Model

- General approach
  - Fit a function to each pixel's neighborhood of the image
  - Use the gradient of the function as the digital gradient of the image neighborhood
- Example: Fit a plane to a 2 x 2 neighborhood
  - $z = ax + by + c$; z is gray level
  - Gradient is then $(a^2 + b^2)^{1/2}$
  - Neighborhood points are $I(x,y)$, $I(x+1,y)$, $I(x,y+1)$ and $I(x+1,y+1)$
- Need to minimize $E(a,b,c) = \Sigma\Sigma\, [a(x+i) + b(y+j) + c - I(x+i,y+j)]^2$
- Solve this and similar problems by:
  - Differentiating with respect to a,b,c, set results to 0, and
  - Solve for a,b,c in resulting system of equations

129

Edge Detection by Function Fitting

- $\partial E/\partial a = \Sigma\Sigma 2[a(x+i) + b(y+j) + c - I(x+i,y+j)](x+i)$
- $\partial E/\partial b = \Sigma\Sigma 2[a(x+i) + b(y+j) + c - I(x+i,y+j)](y+j)$
- $\partial E/\partial c = \Sigma\Sigma 2[a(x+i) + b(y+j) + c - I(x+i,y+j)]$
- It is easy to verify that

  $a = [I(x+1,y) + I(x+1,y+1) - I(x,y) - I(x,y+1)]/2$

  $b = [I(x,y+1) + I(x+1,y+1) - I(x,y) - I(x+1,y)]/2$
- a and b are the partial derivatives with respect to x and y

$$a = \begin{matrix} -1 & 1 \\ -1 & 1 \end{matrix} \qquad b = \begin{matrix} 1 & 1 \\ -1 & -1 \end{matrix}$$

130

Edge Detection by Function Fitting

- Could also fit a higher order surface than a plane
  - With a second order surface we could find the (linear) combination of pixel values that corresponds to the higher order derivatives, which can also be used for edge detection
- Would ordinarily use a neighborhood larger than 2 x 2
  - Better fit
  - For high degree functions need more points for the fit to be reliable

131

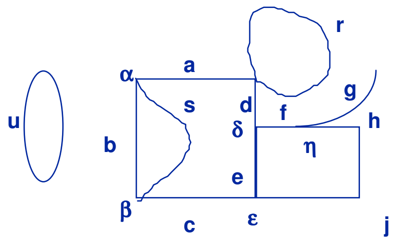
Edge Linking and Following

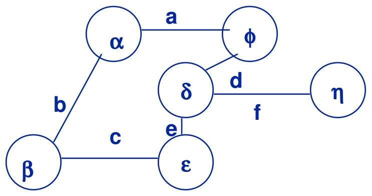
- Group edge pixels into chains and chains into large pieces of object boundary.
  - can use the shapes of long edge chains in recognition
    - slopes
    - curvature
    - corners
- Basic steps
  - thin connected components of edges to one pixel thick
  - find simply connected paths
  - link them at corners into a graph model of image contours
  - compute local and global properties of contours and corners

132

Finding Simply Connected Chains

- Goal: create a graph-structured representation (**chain graph**) of the image contours
  - vertex for each junction in the image
  - edge connecting vertices corresponding to junctions that are connected by an chain; edge labeled with chain





Creating the Chain Graph

- Algorithm: given binary image, E, of thinned edges
  - create a binary image, J, of junctions and end points
    - points in E that are 1 and have more than two neighbors that are 1 or exactly one neighbor that is a 1
  - create the image E-J = C(chains)
    - this image contains the chains of E, but they are broken at junctions
  - perform a connected component analysis of C. For each component store in a table T:
    - its end points (0 or 2)
    - the list of coordinates joining its end points
  - For each point in J:
    - create a node in the chain graph , G, with a unique label

Creating the Chain Graph

- For each chain in C
  - if that chain is a closed loop (has no end points)
    - choose one point from the chain randomly and create a new node in G corresponding to that point
    - mark that point as a “loop junction” to distinguish it from other junctions
    - create an edge in G connecting this new node to itself, and mark that edge with the name of the chain loop
  - if that chain is not a closed loop, then it has two end points
    - create an edge in G linking the two points from J adjacent to its end points

135

Creating the Chain Graph

- Data structure for creating the chain graph
- Biggest problem is determining for each open chain in C the points in J that are adjacent to its end points
  - sophisticated solution might use a hierarchical data structure, like a k-d tree, to represent the points in J
  - more direct solution is to create image J in which all 1’s are marked with their unique labels
  - For each chain in C
    - Examine the 3 x 3 neighborhood of each end point of C in J
    - Find the name of the junction or end point adjacent to that end point from this 3 x 3 neighborhood

136

Finding Internal "Corners" of Chains

- Chains are only broken at junctions
  - But important features of the chain might occur at internal points
  - Example: closed loop corresponding to a square - would like to find the natural corners of the square and add them as junctions to the chain graph (splitting the chains at those natural corners)
- Curve segmentation
  - similar to image segmentation, but in a 1D form
    - Local methods, like edge detectors
    - Global methods, like region analyzers

137

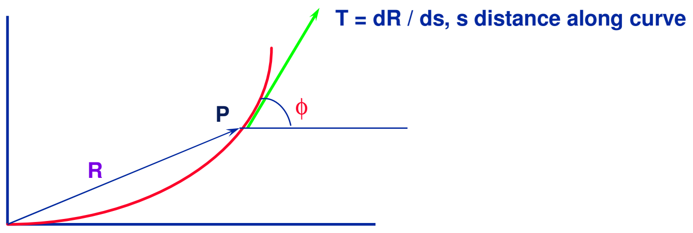
Local Methods of Curve Segmentation

- Natural locations to segment contours are points where the slope of the curve is changing quickly
  - These correspond, perceptually, to "corners" of the curve
- To measure the change in slope we are measuring the curvature of the curve
  - Straight line has 0 curvature
  - Circular arc has constant curvature corresponding to 1/r
- Can estimate curvature by fitting a simple function (circular arc, quadratic function, cubic function) to each neighborhood of a chain, and using the parameters of the fit to estimate the curvature at the center of the neighborhood

138

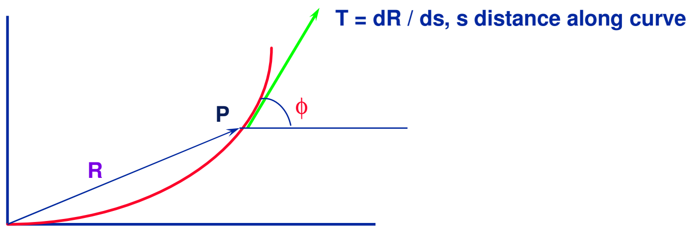
Formulas for Curvature

- Consider moving a point, P, along a curve
  - Let T be the unit tangent vector as P moves
    - T has constant length (1)
    - The direction of T, ϕ, changes from point to point unless the curve is a straight line
    - Measure this direction as the angle between T and the x-axis



Formulas for Curvature

- The **curvature**, κ, is the instantaneous rate of change of ϕ with respect to s, distance along the curve
  - $\kappa = d\phi / ds$
  - $ds = [dx^2 + dy^2]^{1/2}$
  - $\phi = \tan^{-1} dy/dx$



Formulas for Curvature

- Now

$$d\phi / dx = \frac{\frac{d^2 y}{dx^2}}{1 + (\frac{dy}{dx})^2}$$

and

$$ds / dx = \sqrt{1 + (\frac{dy}{dx})^2}$$

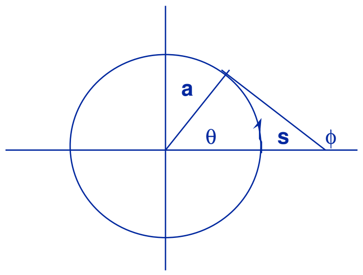
so

$$\kappa = d\phi / ds = \frac{d\phi / dx}{ds / dx} = \frac{d^2 y \big/ dx^2}{[1 + (\frac{dy}{dx})^2]^{3/2}}$$

141

Example - Circle

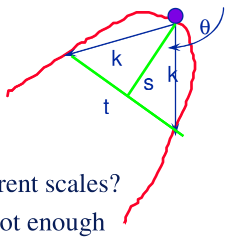
- For the circle
  - $s = a\theta$
  - $\phi = \theta + \pi/2$
  - so $\kappa = d\phi/ds = d\theta/ad\theta = 1/a$



142

Local Methods of Curve Segmentation

- There are also a wide variety of heuristic methods to estimate curvature like local properties
  - For each point, p , along the curve
  - Find the points k pixels before and after p on the curve (p^{+k}, p^{-k}) and then measure
    - the angle between pp^{+k} and pp^{-k}
    - the ratio s/t
- Similar problems to edge detection
  - what is the appropriate size for k?
  - how do we combine the curvature estimates at different scales?
  - boundary problems near the ends of open curves - not enough pixels to look out k in both directions



143