Feature Extraction (Part 2)

Image Processing

CSE 166

Lecture 17

Announcements

- Assignment 6 is due today, 11:59 PM
- Final exam is Jun 12, 7:00 PM-9:59 PM
- Please complete TA and course evaluations
- Reading
  - Chapter 12: Feature extraction
    - Sections 12.7

Feature extraction

- Feature extraction is comprised of
  - Feature detection
  - Feature description
    - A feature descriptor is
      - **Invariant** with respect to a set of transformations if its value remains unchanged after the application of any transformation from the family
      - **Covariant** with respect to a set of transformations if applying any transformation from the set produces the same result in the descriptor

Features

- Features
  - Local (a member of a set)
  - Global (the entire set)
- Categories
  - Boundaries (not covered in CSE 166)
  - Regions
  - Whole images

Scale invariant feature transform (SIFT)

- SIFT features are called *keypoints*
- Keypoints are invariant to
  - Scale
  - Rotation
- Keypoints are robust to
  - Changes in viewpoint
  - Changes in illumination
  - Noise

Scale invariant feature transform (SIFT)

- SIFT feature descriptors are *n*-dimensional feature vectors
  - Elements are invariant feature descriptors

Scale invariant feature transform (SIFT)

- Steps
  1. Construct the scale space
  2. Obtain the initial keypoints
  3. Improve the accuracy of the location of the keypoints
  4. Delete unsuitable keypoints
  5. Compute keypoint orientations
  6. Compute keypoint descriptors

SIFT, construct the scale space

- Search for stable features across all possible scales
  - Use a function of scale known as *scale space*
- Achieves scale invariance

Scale space

- Scale space theory is a formal theory for image structures at different scales
  - Image is represented by a one-parameter *family* of Gaussian low pass filtered images
  - A Gaussian filter meets all scale space axioms
    - Linearity, shift invariance, semi-group structure, non-creation of local extrema (zero-crossings), non-enhancement of local extrema, rotational symmetry, and scale invariance
- The *scale parameter* is the variance of the Gaussian filter
  - Note: use border mirror padding on input image when applying filter
- Image details significantly smaller than (two times) the standard deviation (square root of variance) are removed from the image at that scale parameter



SIFT, construct the scale space

- Each octave corresponds to doubling the standard deviation
  - First image at each new octave is downsampled third image (*octave image*) from previous octave

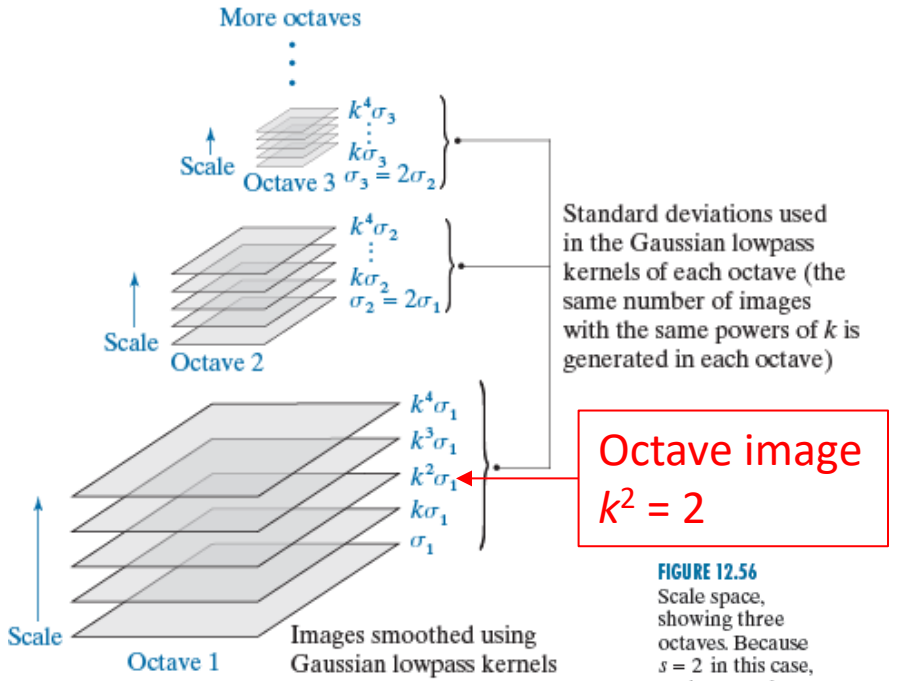


FIGURE 12.56 Scale space, showing three octaves. Because $s = 2$ in this case, each octave has five smoothed images. A Gaussian kernel was used for smoothing, so the space parameter is σ.

SIFT, construct the scale space

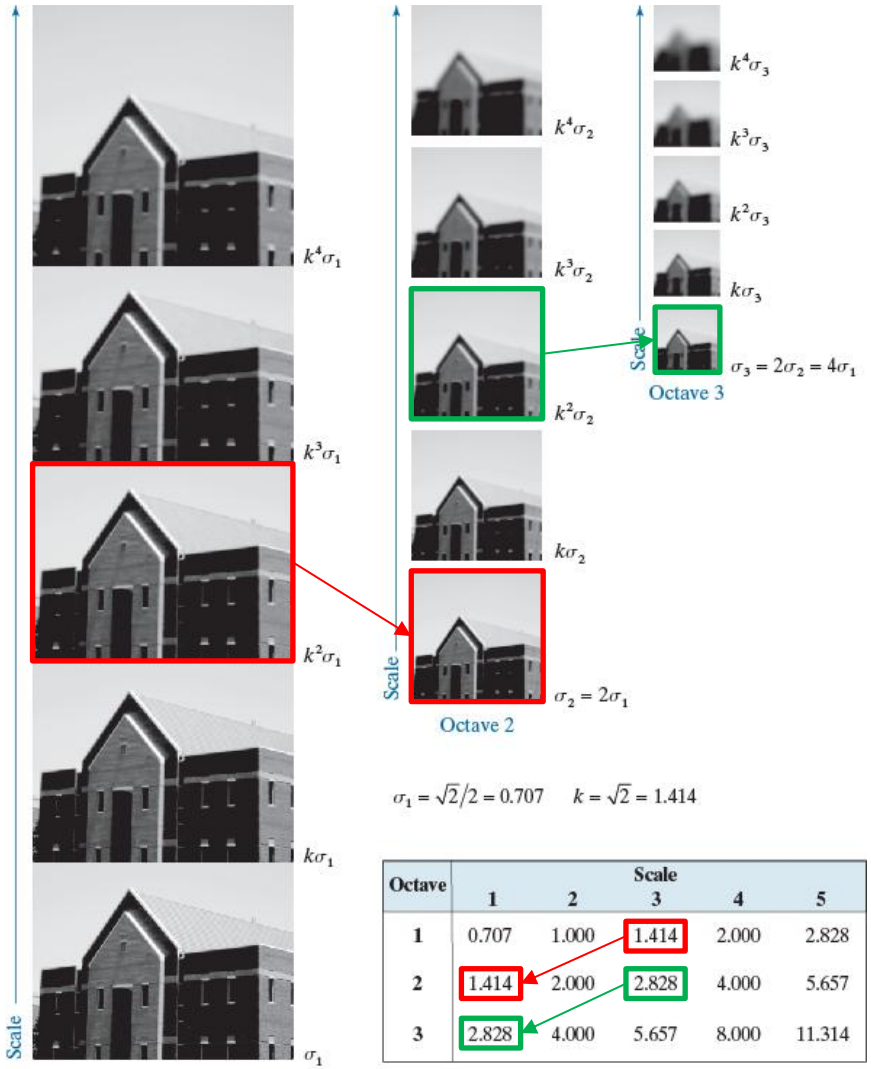


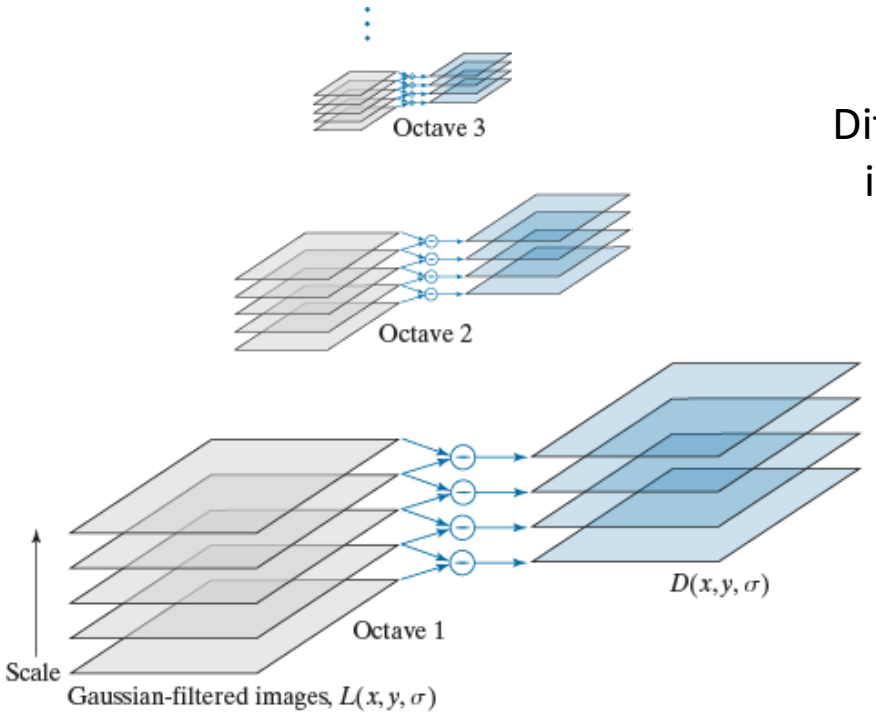
FIGURE 12.57 Illustration using images of the first three octaves of scale space in SIFT. The entries in the table are values of standard deviation used at each scale of each octave. For example the standard deviation used in scale 2 of octave 1 is $k\sigma_1$, which is equal to 1.0. (The images of octave 1 are shown slightly overlapped to fit in the figure space.)

$\sigma_1 = \sqrt{2}/2 = 0.707 \quad k = \sqrt{2} = 1.414$

Octave	Scale 1	2	3	4	5
1	0.707	1.000	1.414	2.000	2.828
2	1.414	2.000	2.828	4.000	5.657
3	2.828	4.000	5.657	8.000	11.314

SIFT, obtain the initial keypoints

- First, difference two adjacent scale-space images in an octave



Difference images

FIGURE 12.58 How Eq. (12-69) is implemented in scale space. There are $s+3$ $L(x, y, \sigma)$ images and $s+2$ corresponding $D(x, y, \sigma)$ images in each octave.

SIFT, obtain the initial keypoints

- Second, detect extrema in the differences

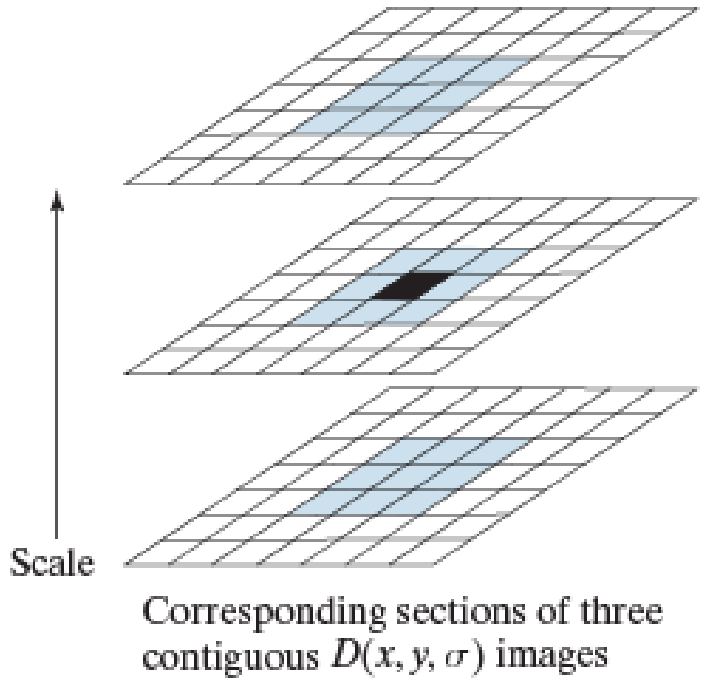


FIGURE 12.59 Extrema (maxima or minima) of the $D(x, y, \sigma)$ images in an octave are detected by comparing a pixel (shown in black) to its 26 neighbors (shown shaded) in 3×3 regions at the current and adjacent scale images.

The point (in black) is selected as an extremum point if its value is **larger** than the values of **all** its neighbors (in blue) *or* **smaller** than the values of **all** its neighbors (in blue)

SIFT, obtain the initial keypoints

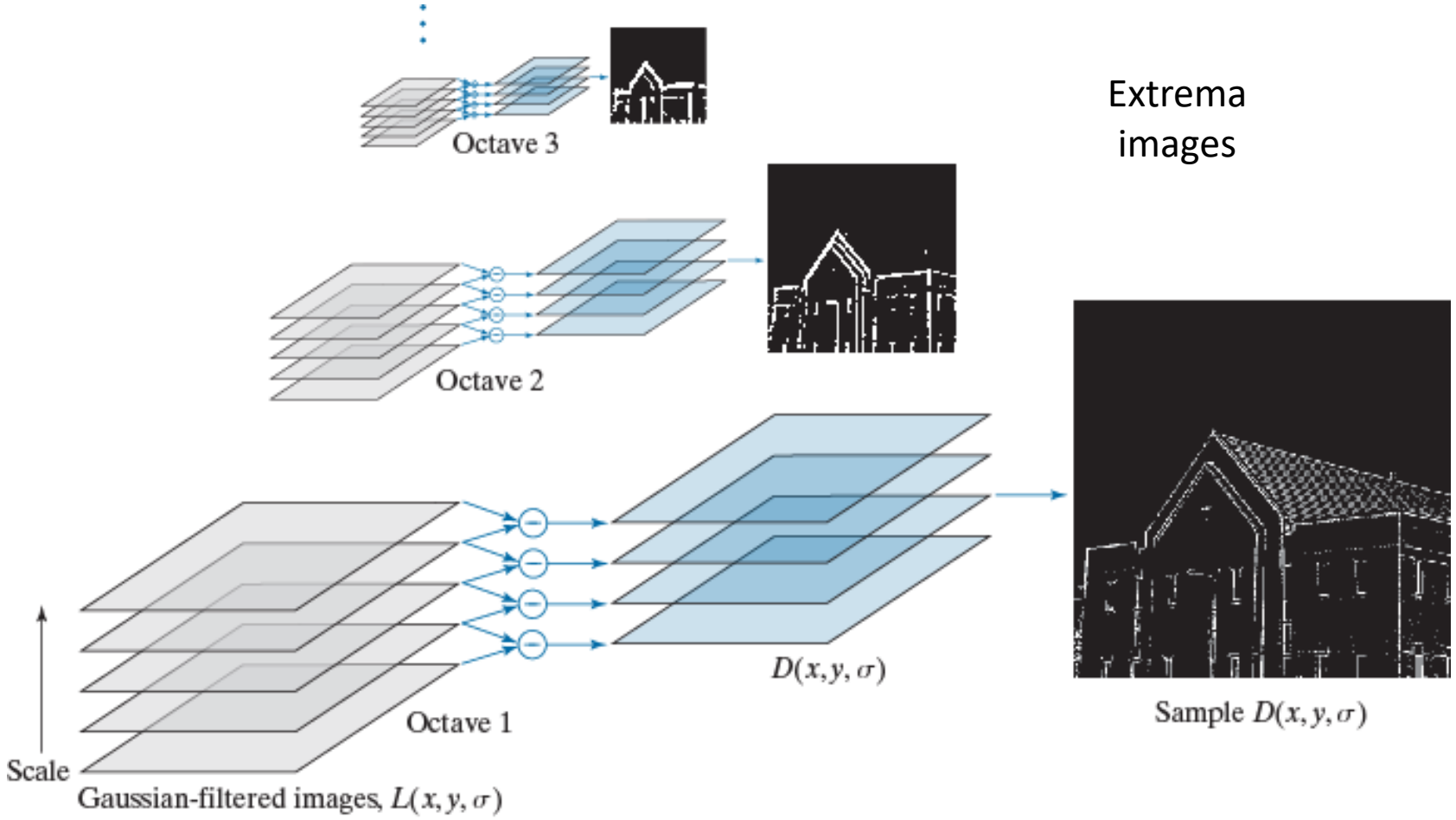
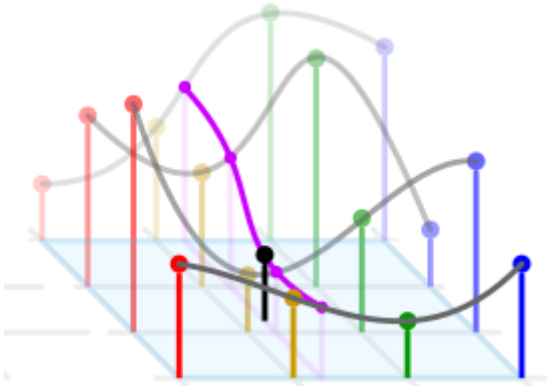


FIGURE 12.58 How Eq. (12-69) is implemented in scale space. There are $s+3$ $L(x, y, \sigma)$ images and $s+2$ corresponding $D(x, y, \sigma)$ images in each octave.

SIFT, improve the accuracy of the location of the keypoints

- Interpolate the values of the difference images about extrema
- Determine subpixel coordinates of extrema using interpolated values



Bicubic

SIFT, delete unsuitable keypoints

- Determine difference at subpixel keypoints
- Eliminate keypoints with low contrast and/or are poorly localized
- Additionally, delete keypoints associated with edges
  - Only keep corner-like features
    - Equivalent to thresholding the minor eigenvalue

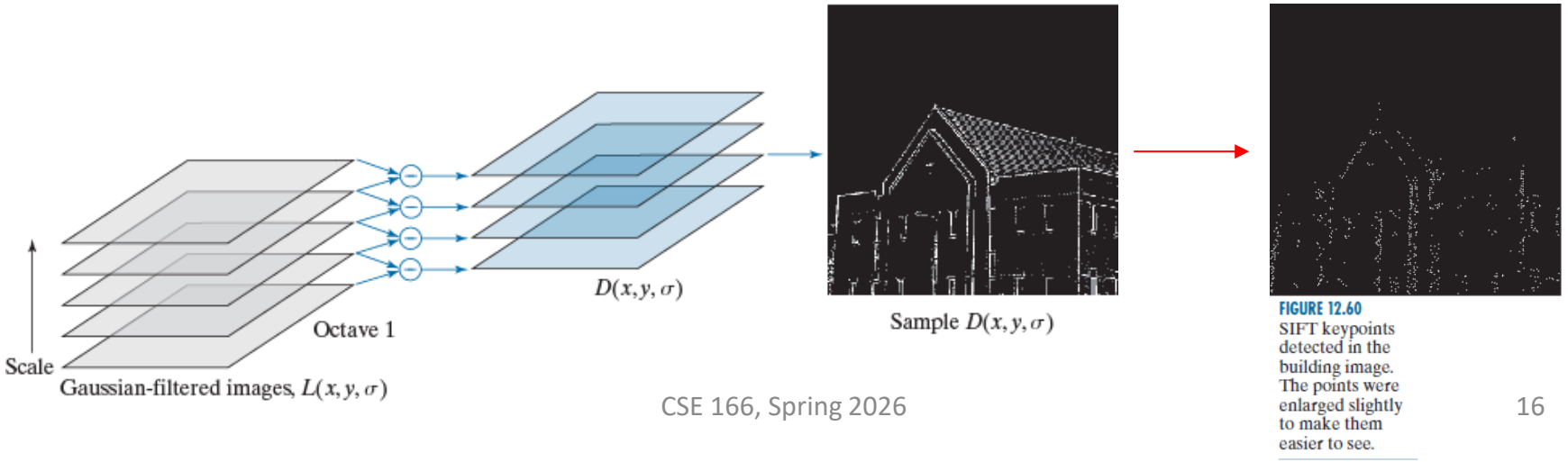


FIGURE 12.60 SIFT keypoints detected in the building image. The points were enlarged slightly to make them easier to see.

Scale invariant feature transform (SIFT)

- Steps
  1. Construct the scale space
  2. Obtain the initial keypoints
  3. Improve the accuracy of the location of the keypoints
  4. Delete unsuitable keypoints
  - So far, we have computed the location of each keypoint in scale space (i.e., location and scale of each keypoint)
    - Scale invariance
  - Next is rotation invariance
  5. Compute keypoint orientations
  6. Compute keypoint descriptors

SIFT, compute keypoint orientations

- For each keypoint
  - At its scale, compute the gradient magnitude and orientation of points in region about keypoint
  - Form a **histogram of orientations** from points in region about keypoint
    - 36 bins (10 degrees each)
    - Weight an orientation by its associated magnitude **and** a Gaussian, when adding it to an orientation bin
  - Initial orientation is largest bin
    - Create an additional keypoint for other bins within 80% the size of the largest bin
  - Improve orientation estimate using interpolation
    - Fit a parabola to values of the largest bin and its two neighboring bins

SIFT, compute keypoint orientations

- Keypoints
  - Location and scale (scale invariant)
  - Orientation (rotation invariant)
    - Length of arrow is histogram of orientations interpolated bin value
    - (Useful in matching keypoints across images)



FIGURE 12.60 SIFT keypoints detected in the building image. The points were enlarged slightly to make them easier to see.



FIGURE 12.61 The keypoints from Fig. 12.60 superimposed on the original image. The arrows indicate keypoint orientations.

Local regions

- The local region about each **oriented** keypoint is invariant to
  - Scale, orientation, illumination, and image viewpoint



FIGURE 12.61
The keypoints from Fig. 12.60 superimposed on the original image. The arrows indicate keypoint orientations.

SIFT, compute keypoint descriptors

- Feature descriptor
  - 16x16 region about keypoint
    - Gradient magnitude (Gaussian weighted) and direction at each point in region
    - Quantize gradient directions in each 4x4 subregion to 45 degree increments
      - Interpolate each of the 16 gradients directions to distribute it over all 8 bins (8 * 45 degrees = 360 degrees)
    - Concatenate the 16 8-directional histograms bins to form a 128-dimensional feature vector

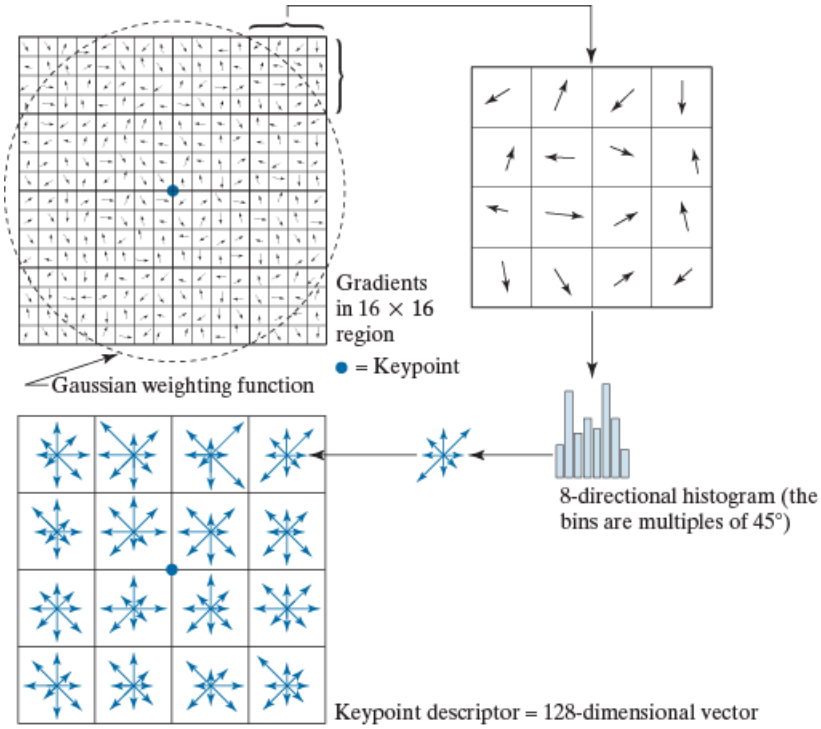
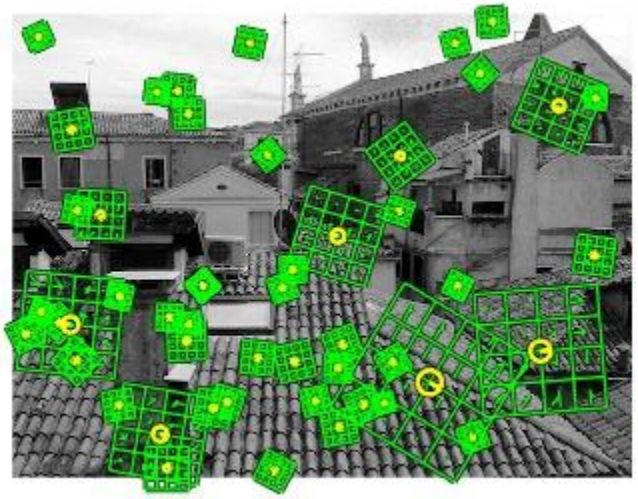


FIGURE 12.62
Approach used to compute a keypoint descriptor.

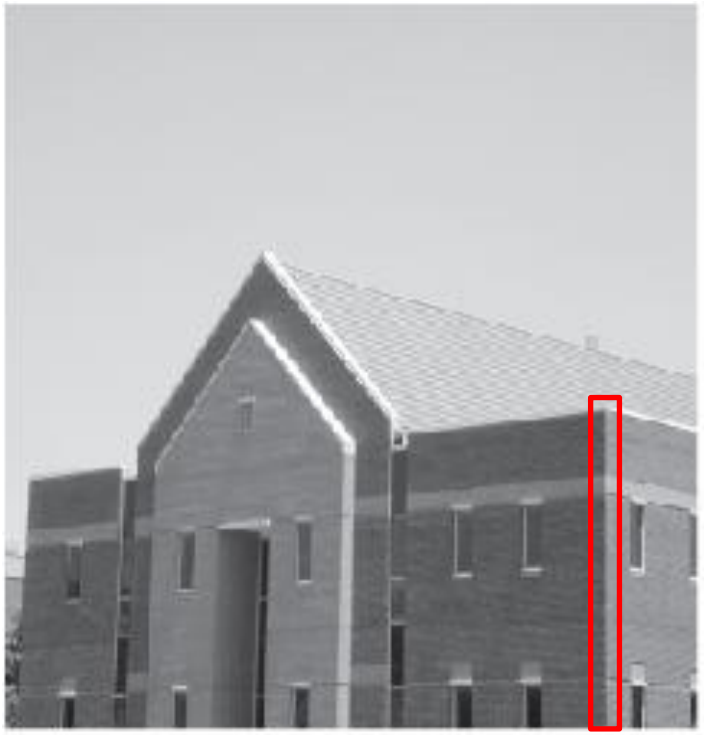
SIFT, compute keypoint descriptors

- Rotation invariance
  - Rotate the 8-directional histograms relative to the keypoint orientation
- Robustness to changes in illumination
  - Unitize the 128-dimensional feature vector
  - Threshold to reduce the influence of large gradient magnitudes
  - Unitize again



Matching keypoints across images

- First image is whole image
- Second image is darker version of red rectangle



Matching keypoints across images using SIFT features and feature descriptors



a b

FIGURE 12.63 (a) Keypoints and their directions (shown as gray arrows) for the building image and for a section of the right corner of the building. The subimage is a separate image and was processed as such. (b) Corresponding key points between the building and the subimage (the straight lines shown connect pairs of matching points). Only three of the 36 matches found are incorrect.

Application: panorama

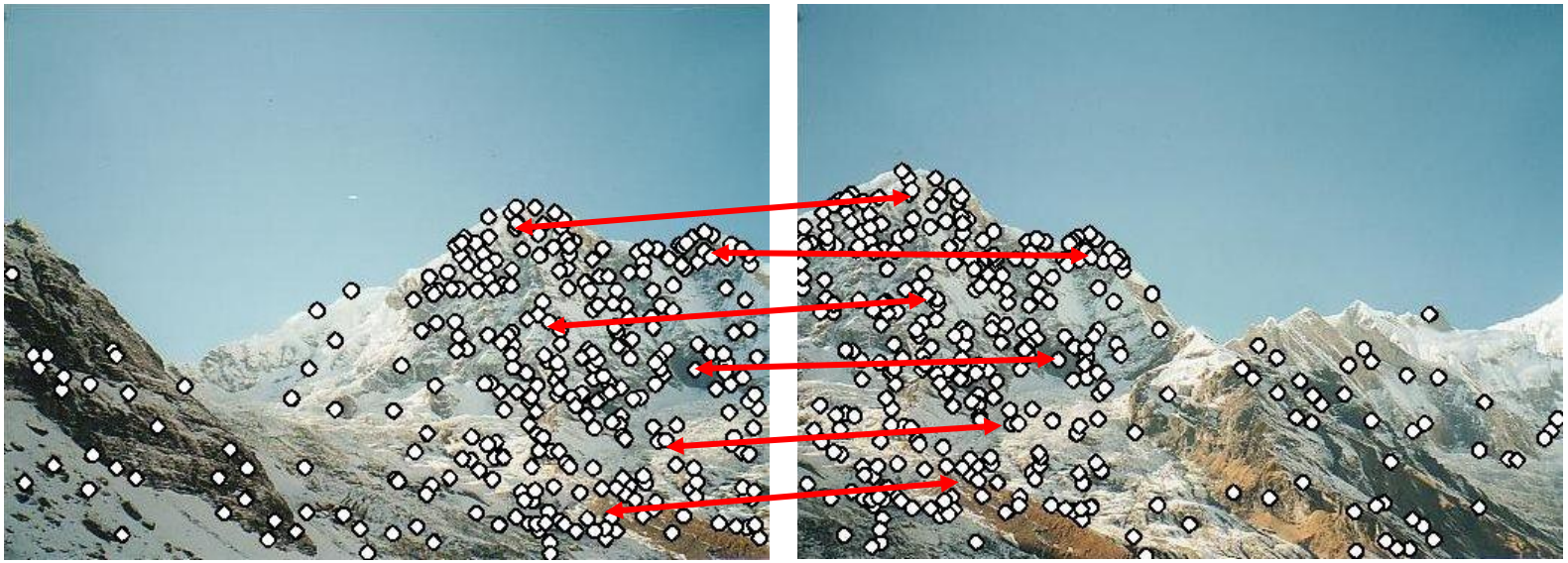
- How do we align these two images?





Application: panorama

- How do we align these two images?



Step 1: extract features in each image

Step 2: match features across images

Application: panorama

- How do we align these two images?



Step 1: extract features in each image

Step 2: match features across images

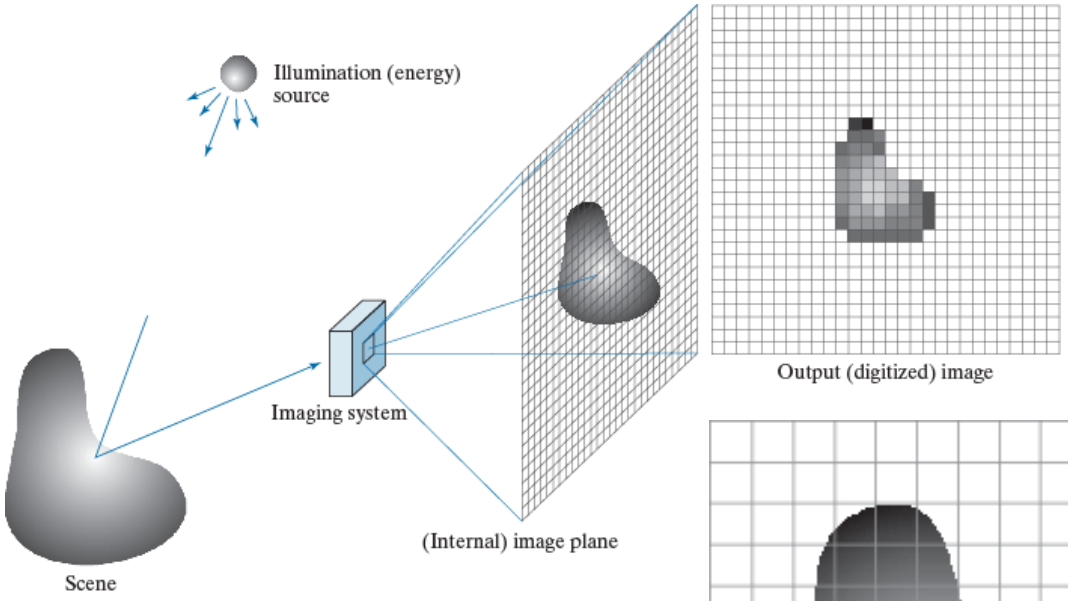
Step 3: estimate and apply a 2D transformation

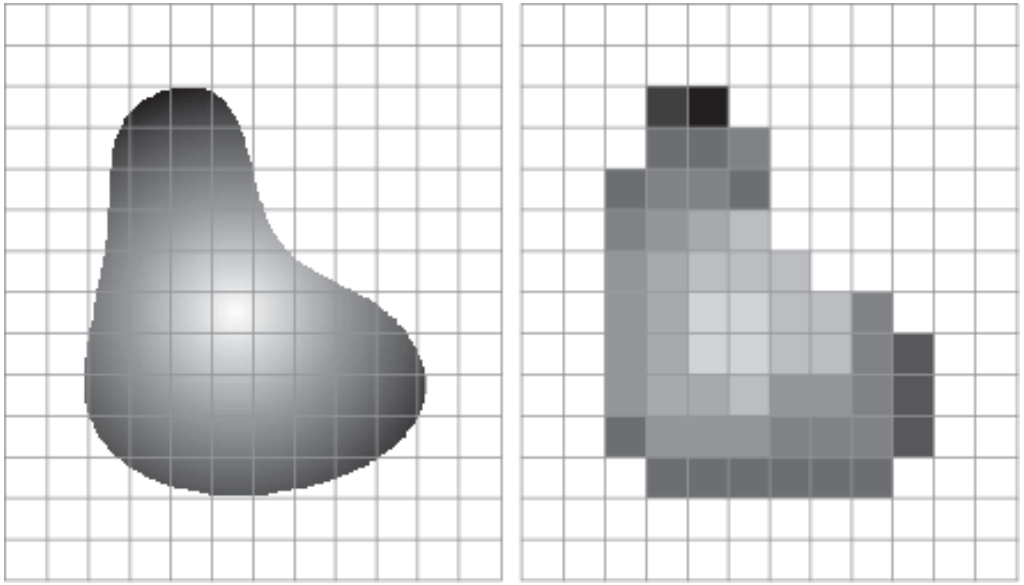
Review

Midterm exam topics

- Geometric transformations
- Intensity transformations
- Spatial filtering
- Fourier transform and filtering in the frequency domain

Image acquisition





Sampling and quantization

Geometric transformations

Euclidean transformation

$$x' = x\cos\theta - y\sin\theta + t_x$$
$$y' = x\sin\theta + y\cos\theta + t_y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Similarity transformation

$$x' = sx\cos\theta - sy\sin\theta + t_x$$
$$y' = sx\sin\theta + sy\cos\theta + t_y$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine transformation

$$x' = a_{11}x + a_{12}y + a_{13}$$
$$y' = a_{21}x + a_{22}y + a_{23}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image filtering and enhancement

- Intensity transformations
- Spatial filtering



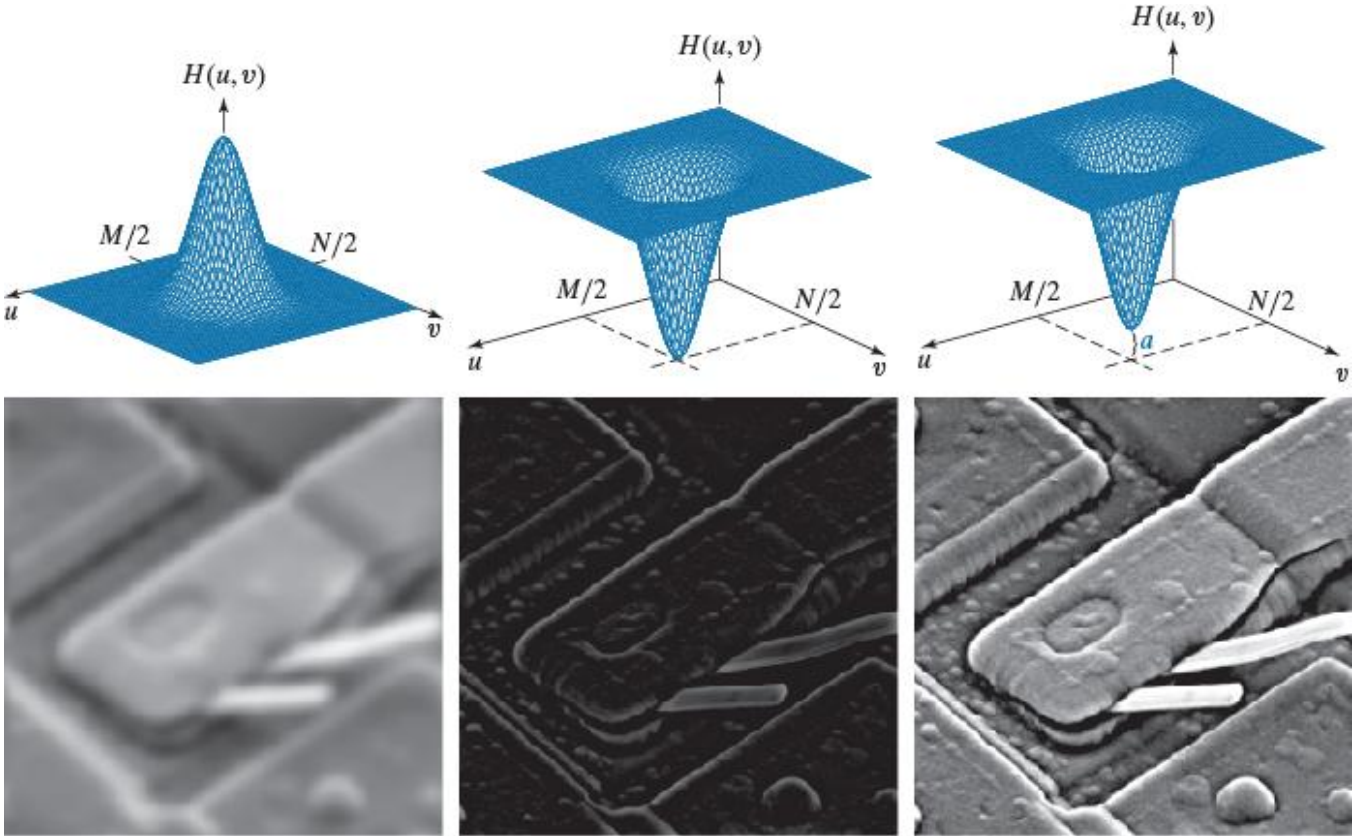
Low-pass filter



Gamma correction

Image filtering and enhancement

- Filtering in the frequency domain

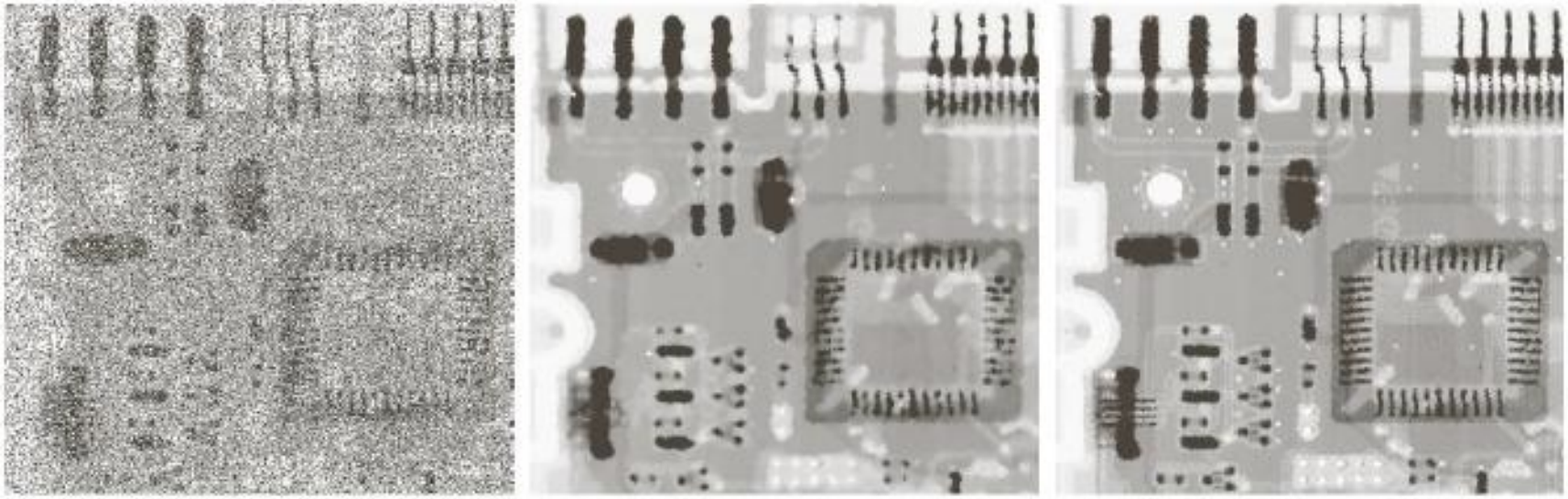


Final exam topics

- Image restoration
- Color image processing
- Basis vector and matrix-based transforms
- Multiscale image representations and the wavelet transform
- Image compression and watermarking
- Morphological image processing
- Image segmentation
- Feature extraction

Image restoration

- Noise models
- Noise reduction



Model of image degradation

- Spatial domain

$$g(x,y) = h(x,y) \bigstar f(x,y) + \eta(x,y)$$

Degraded image | Degradation function | Original image | Noise image

- Frequency domain

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

Degraded image | Degradation function | Original image | Noise image

Image restoration

- Inverse filtering

$$G(u,v) = H(u,v)\hat{F}(u,v)$$

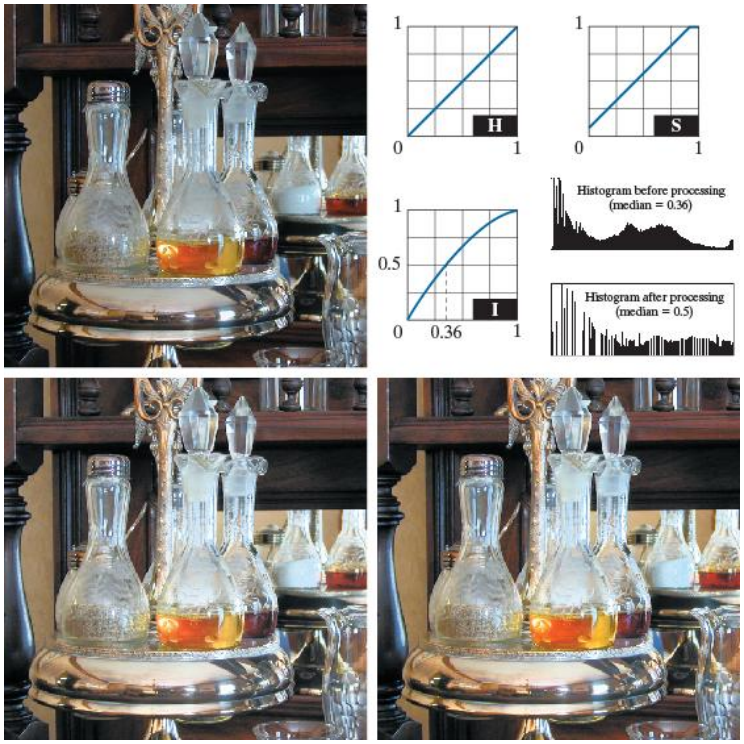
$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

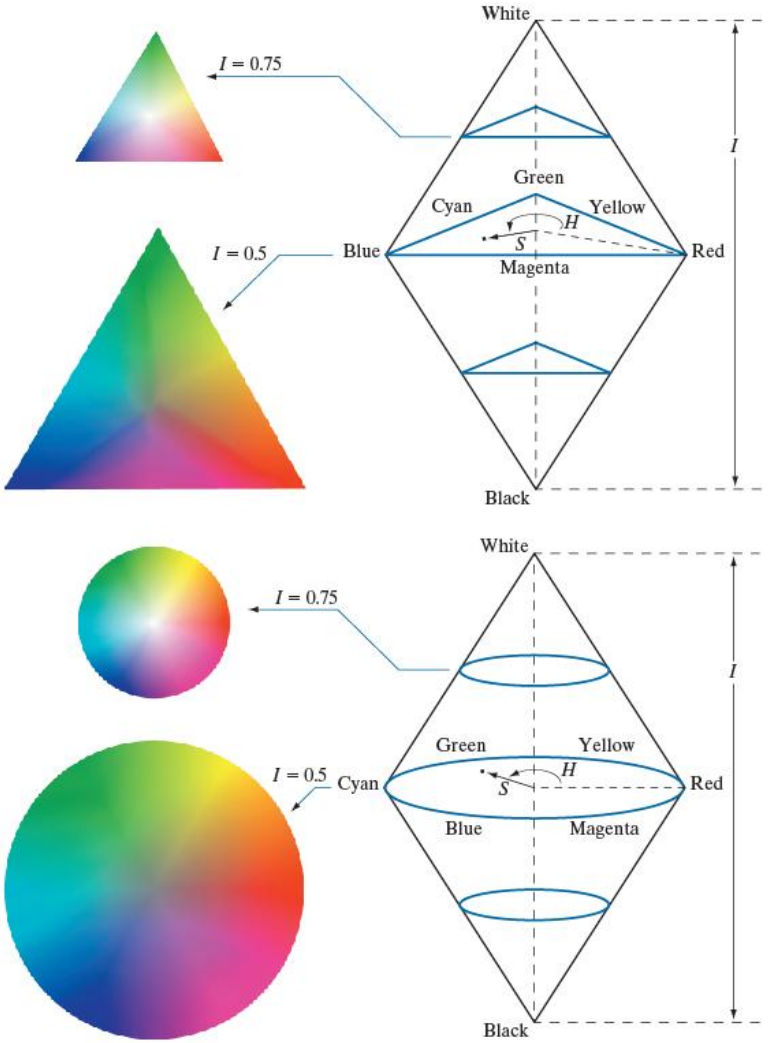
  - Mitigate divide by zero
    - Threshold *H(u,v)*
    - Ideal lowpass filter *H(u,v)*

$$\hat{F}(u,v) = \begin{cases} \frac{G(u,v)}{H(u,v)} & \text{if } |H(u,v)| \geq t \\ 0 & \text{otherwise} \end{cases}$$

Color image processing

- Color models
- Color transformations





Matrix-based transforms

- Transform pair

$$T(u) = \sum_{x=0}^{N-1} f(x) r(x,u)$$

$$f(x) = \sum_{u=0}^{N-1} T(u) s(x,u)$$

where

x is a spatial variable
u is a transform variable
$T(u)$ is the transform of $f(x)$
$f(x)$ is the inverse transform of $T(u)$
$r(x,u)$ is a forward transformation kernel
$s(x,u)$ is an inverse transformation kernel

Note the transformation kernels $r(x,u)$ and $s(x,u)$ depend only on x and u, not values of $f(x)$ and $T(u)$

Matrix-based transforms using orthonormal basis vectors

$$\langle s(x,u), f(x)\rangle = T(0)\langle s(x,u), s(x,0)\rangle + T(1)\langle s(x,u), s(x,1)\rangle + \cdots + T(u)\langle s(x,u), s(x,u)\rangle + \cdots$$

$$\langle s(x,u), f(x)\rangle = T(u)\langle s(x,u), s(x,u)\rangle$$

$$\frac{\langle s(x,u), f(x)\rangle}{\langle s(x,u), s(x,u)\rangle} = T(u)$$

If $s(x,0), s(x,1), s(x,2), \ldots$ are orthonormal basis vectors, then

$$\langle s(x,u), f(x)\rangle = T(u)$$

- In vector form

$$T(u) = \langle s(x,u), f(x)\rangle$$

$$T(u) = \langle \mathbf{s}_u, \mathbf{f}\rangle$$

$$\text{where } \mathbf{s}_u = \begin{bmatrix} s(0,u) \\ s(1,u) \\ \vdots \\ s(N-1,u) \end{bmatrix} \text{ and } \mathbf{f} = \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix}$$

(x indexes the rows of $\mathbf{s}_u$ and $\mathbf{f}$)

$$T(u) = \mathbf{s}_u^H \mathbf{f} \text{ for complex vectors}$$

$$T(u) = \mathbf{s}_u^T \mathbf{f} \text{ for real vectors}$$

Matrix-based transforms using orthonormal basis vectors

- In matrix form

$$\begin{bmatrix} T(0) \\ T(1) \\ \vdots \\ T(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{s}_0^H \\ \mathbf{s}_1^H \\ \vdots \\ \mathbf{s}_{N-1}^H \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix} \quad \text{for complex vectors}$$

$$\begin{bmatrix} T(0) \\ T(1) \\ \vdots \\ T(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_{N-1}^T \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix} \quad \text{for real vectors}$$

$$\mathbf{t} = \mathbf{A}\mathbf{f}$$

where transformation matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{s}_0^H \\ \mathbf{s}_1^H \\ \vdots \\ \mathbf{s}_{N-1}^H \end{bmatrix} \quad \text{for complex vectors} \qquad N \times N$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_{N-1}^T \end{bmatrix} \quad \text{for real vectors} \qquad N \times N$$

Matrix-based transforms using orthonormal basis vectors

- Forward and inverse transformations

$$\mathbf{t} = \mathtt{A}\mathbf{f}$$
$$\mathtt{A}^{-1}\mathbf{t} = \mathbf{f}$$
$$\mathtt{A}^{H}\mathbf{t} = \mathbf{f} \text{ for complex } \mathtt{A}$$
$$\mathtt{A}^{T}\mathbf{t} = \mathbf{f} \text{ for real } \mathtt{A}$$

Note forward transformation kernel $r(x,u)$ is not needed

Note $\mathtt{A}^H$ for complex $\mathtt{A}$ and $\mathtt{A}^T$ for real $\mathtt{A}$ is

$$\begin{matrix} & u \\ x & \end{matrix}\begin{bmatrix} \mathbf{r}_0^\top \\ \mathbf{r}_1^\top \\ \vdots \\ \mathbf{r}_{N-1}^\top \end{bmatrix}$$

where

$$\mathbf{r}_x = \begin{matrix} u \\ \downarrow \end{matrix}\begin{bmatrix} r(x,0) \\ r(x,1) \\ \vdots \\ r(x,N-1) \end{bmatrix}$$

- Properties

Unitary matrix $\mathtt{A}$

$$\mathtt{A}^H\mathtt{A} = \mathtt{A}\mathtt{A}^H = \mathtt{A}^*\mathtt{A}^T = \mathtt{A}^T\mathtt{A}^* = \mathtt{I}$$
$$\mathtt{A}^{-1} = \mathtt{A}^H$$

Orthogonal matrix $\mathtt{A}$

$$\mathtt{A}^T\mathtt{A} = \mathtt{A}\mathtt{A}^T = \mathtt{I}$$
$$\mathtt{A}^{-1} = \mathtt{A}^T$$

Matrix-based transforms in two dimensions

$$T(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) r(x,y,u,v)$$ Forward transform

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v) s(x,y,u,v)$$ Inverse transform

where

x, y are spatial variables

u, v are transform variables

$T(u,v)$ is the transform of $f(x,y)$

$f(x,y)$ is the inverse transform of $T(u,v)$

$r(x,y,u,v)$ is a forward transformation kernel

$s(x,y,u,v)$ is an inverse transformation kernel

Matrix-based transforms in two dimensions

If $s(x, y, u, v) = s_1(x, u)s_2(y, v)$, then s is called seperable.
If $s(x, y, u, v) = s_1(x, u)s_1(y, v)$, then s is called symmetric.
If r and s are separable and symmetric, then

$$\mathtt{T} = \mathtt{A}_M \mathtt{F} \mathtt{A}_N^T$$

$$\mathtt{F} = \mathtt{A}_M^{*T} \mathtt{T} \mathtt{A}_N^* \text{ for complex } \mathtt{A}_M \text{ and } \mathtt{A}_N$$

$$\mathtt{F} = \mathtt{A}_M^T \mathtt{T} \mathtt{A}_N \text{ for real } \mathtt{A}_M \text{ and } \mathtt{A}_N$$

Note forward transformation kernel is not needed

where

$$\mathtt{T} = \begin{array}{c} \\ u \\ \\ \\ \end{array} \overset{v}{\begin{bmatrix} T(0,0) & T(0,1) & \cdots & T(0,N-1) \\ T(1,0) & T(1,1) & \cdots & T(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ T(M-1,0) & T(M-1,1) & \cdots & T(M-1,N-1) \end{bmatrix}}, \qquad M \text{ x } N$$

$$\mathtt{A}_M = \begin{array}{c} \\ u \\ \\ \\ \end{array} \overset{x}{\begin{bmatrix} s_1(0,0) & s_1(1,0) & \cdots & s_1(M-1,0) \\ s_1(0,1) & s_1(1,1) & \cdots & s_1(M-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ s_1(0,M-1) & s_1(1,M-1) & \cdots & s_1(M-1,M-1) \end{bmatrix}}, \text{ and } \quad M \text{ x } M$$

Transformation matrices

$$\mathtt{A}_N = \begin{array}{c} \\ v \\ \\ \\ \end{array} \overset{y}{\begin{bmatrix} s_1(0,0) & s_1(1,0) & \cdots & s_1(N-1,0) \\ s_1(0,1) & s_1(1,1) & \cdots & s_1(N-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ s_1(0,N-1) & s_1(1,N-1) & \cdots & s_1(N-1,N-1) \end{bmatrix}} \qquad N \text{ x } N$$

Matrix-based transforms in two dimensions

- If r and s are separable and symmetric, and $M = N$, then
  - For orthonormal basis vectors

$$\mathtt{T} = \mathtt{A}\mathtt{F}\mathtt{A}^T$$

$$\mathtt{F} = \mathtt{A}^{*T}\mathtt{T}\mathtt{A}^* \text{ for complex } \mathtt{A}$$

$$\mathtt{F} = \mathtt{A}^T\mathtt{T}\mathtt{A} \text{ for real } \mathtt{A}$$

Note $\mathtt{A}^H$ for complex $\mathtt{A}$ and $\mathtt{A}^T$ for real $\mathtt{A}$ is

$$\begin{matrix} & u \\ x & \begin{bmatrix} \mathbf{r}_0^\top \\ \mathbf{r}_1^\top \\ \vdots \\ \mathbf{r}_{N-1}^\top \end{bmatrix} \end{matrix}$$

where

$$\mathbf{r}_x = \begin{matrix} u \\ \downarrow \\ \end{matrix} \begin{bmatrix} r(x,0) \\ r(x,1) \\ \vdots \\ r(x,N-1) \end{bmatrix}$$

  - For biorthonormal basis vectors

$$\mathtt{T} = \tilde{\mathtt{A}}\mathtt{F}\tilde{\mathtt{A}}^T$$

$$\mathtt{F} = \mathtt{A}^{*T}\mathtt{T}\mathtt{A}^* \text{ for complex } \mathtt{A}$$

$$\mathtt{F} = \mathtt{A}^T\mathtt{T}\mathtt{A} \text{ for real } \mathtt{A}$$

Use dual vectors for forward transformation only

Matrix-based transforms in two dimensions

- Inverse transform

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)s(x,y,u,v)$$ Using transformation kernel

$$\mathrm{F} = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)\mathrm{S}_{u,v}$$ Using basis images

where

$$\mathrm{F} = \begin{array}{c} \\ x \\ \\ \end{array}\!\!\begin{array}{c} \overset{y}{\downarrow} \\ \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,N-1) \end{bmatrix} \end{array}$$

Each $S_{u,v}$ is a basis image

$$\mathrm{S}_{u,v} = \begin{array}{c} \\ x \\ \\ \end{array}\!\!\begin{array}{c} \overset{y}{\downarrow} \\ \begin{bmatrix} s(0,0,u,v) & s(0,1,u,v) & \cdots & s(0,N-1,u,v) \\ s(1,0,u,v) & s(1,1,u,v) & \cdots & s(1,N-1,u,v) \\ \vdots & \vdots & \ddots & \vdots \\ s(N-1,0,u,v) & s(N-1,1,u,v) & \cdots & s(N-1,N-1,u,v) \end{bmatrix} \end{array}$$

Matrix-based transforms in two dimensions using basis images

- Forward transform

$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)r(x,y,u,v)$$ Using transformation kernel

$$\mathtt{T} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\mathtt{R}_{x,y}$$ Using basis images

where

$$\mathtt{T} = \begin{matrix} u \\ \\ \\ \\ \end{matrix}\overset{v}{\begin{bmatrix} T(0,0) & T(0,1) & \cdots & T(0,N-1) \\ T(1,0) & T(1,1) & \cdots & T(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ T(N-1,0) & T(N-1,1) & \cdots & T(N-1,N-1) \end{bmatrix}}$$

Each $\mathrm{R}_{x,y}$ is a basis image

$$\mathtt{R}_{x,y} = \begin{matrix} u \\ \\ \\ \\ \end{matrix}\overset{v}{\begin{bmatrix} r(x,y,0,0) & r(x,y,0,1) & \cdots & r(x,y,0,N-1) \\ r(x,y,1,0) & r(x,y,1,1) & \cdots & r(x,y,1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(x,y,N-1,0) & r(x,y,N-1,1) & \cdots & r(x,y,N-1,N-1) \end{bmatrix}}$$

Matrix-based transforms in two dimensions using basis images

- Forward transformation

Each $\mathbf{R}_{x,y}$ is a basis image

$$\mathbf{R}_{x,y} = \begin{matrix} u \downarrow \\ \end{matrix} \begin{bmatrix} r(x,y,0,0) & r(x,y,0,1) & \cdots & r(x,y,0,N-1) \\ r(x,y,1,0) & r(x,y,1,1) & \cdots & r(x,y,1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ r(x,y,N-1,0) & r(x,y,N-1,1) & \cdots & r(x,y,N-1,N-1) \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \end{matrix}$$

$v \rightarrow$

If $r(x,y,u,v)$ is separable and symmetric, then

$$\mathbf{R}_{x,y} = \mathbf{r}_x \mathbf{r}_y^T$$

where

$$\mathbf{r}_x = \begin{matrix} u \downarrow \\ \end{matrix} \begin{bmatrix} r(0,y) \\ r(1,y) \\ \vdots \\ r(N-1,y) \end{bmatrix} \quad \text{and} \quad \mathbf{r}_y = \begin{matrix} v \downarrow \\ \end{matrix} \begin{bmatrix} r(0,y) \\ r(1,y) \\ \vdots \\ r(N-1,y) \end{bmatrix}$$

Matrix-based transforms in two dimensions using basis images

- Inverse transform

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)s(x,y,u,v)$$ Using transformation kernel

$$\mathbf{F} = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v)\mathbf{S}_{u,v}$$ Using basis images

where

$$\mathbf{F} = \begin{matrix} x \\ \\ \\ \\ \end{matrix}\overset{y}{\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,N-1) \end{bmatrix}}$$

Each $\mathbf{S}_{u,v}$ is a basis image

$$\mathbf{S}_{u,v} = \begin{matrix} x \\ \\ \\ \\ \end{matrix}\overset{y}{\begin{bmatrix} s(0,0,u,v) & s(0,1,u,v) & \cdots & s(0,N-1,u,v) \\ s(1,0,u,v) & s(1,1,u,v) & \cdots & s(1,N-1,u,v) \\ \vdots & \vdots & \ddots & \vdots \\ s(N-1,0,u,v) & s(N-1,1,u,v) & \cdots & s(N-1,N-1,u,v) \end{bmatrix}}$$

Matrix-based transforms in two dimensions using basis images

- Inverse transformation

Each $S_{u,v}$ is a basis image

$$\mathbf{S}_{u,v} = \begin{bmatrix} s(0,0,u,v) & s(0,1,u,v) & \cdots & s(0,N-1,u,v) \\ s(1,0,u,v) & s(1,1,u,v) & \cdots & s(1,N-1,u,v) \\ \vdots & \vdots & \ddots & \vdots \\ s(N-1,0,u,v) & s(N-1,1,u,v) & \cdots & s(N-1,N-1,u,v) \end{bmatrix}$$

(x: rows, y: columns)

If $s(x,y,u,v)$ is separable and symmetric, then

$$\mathbf{S}_{u,v} = \mathbf{s}_u \mathbf{s}_v^T$$

where

$$\mathbf{s}_u = \begin{bmatrix} s(0,u) \\ s(1,u) \\ \vdots \\ s(N-1,u) \end{bmatrix} \quad \text{and} \quad \mathbf{s}_v = \begin{bmatrix} s(0,v) \\ s(1,v) \\ \vdots \\ s(N-1,v) \end{bmatrix}$$

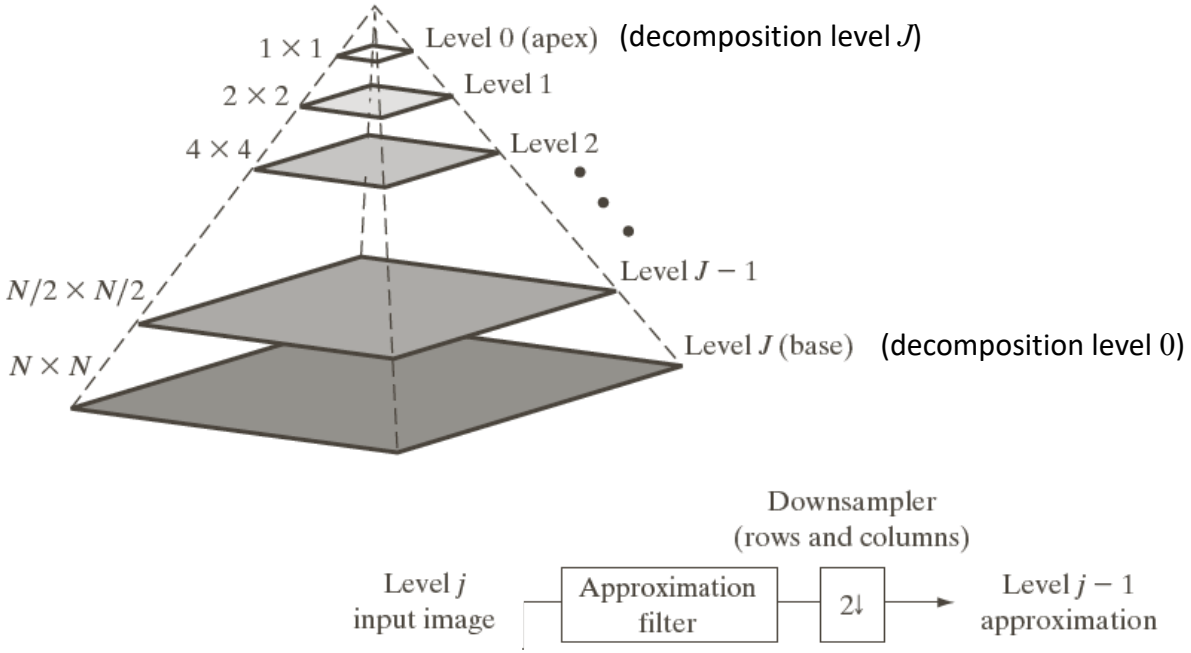
(x: rows, y: columns)

Multiscale image representations

- Image pyramids
  - Gaussian pyramid (approximation pyramid)
  - Laplacian pyramid (prediction residual pyramid)
- Scale space
- Wavelet decomposition

Image pyramids

- Gaussian pyramid (approximation pyramid)



Laplacian pyramid

- Full name is Laplacian of Gaussian (LoG)
  - Apply Gaussian (low pass) filter, then apply Laplacian (second derivative) filter
  - Approximates second derivative
  - Approximate with difference of Gaussians (DoG)

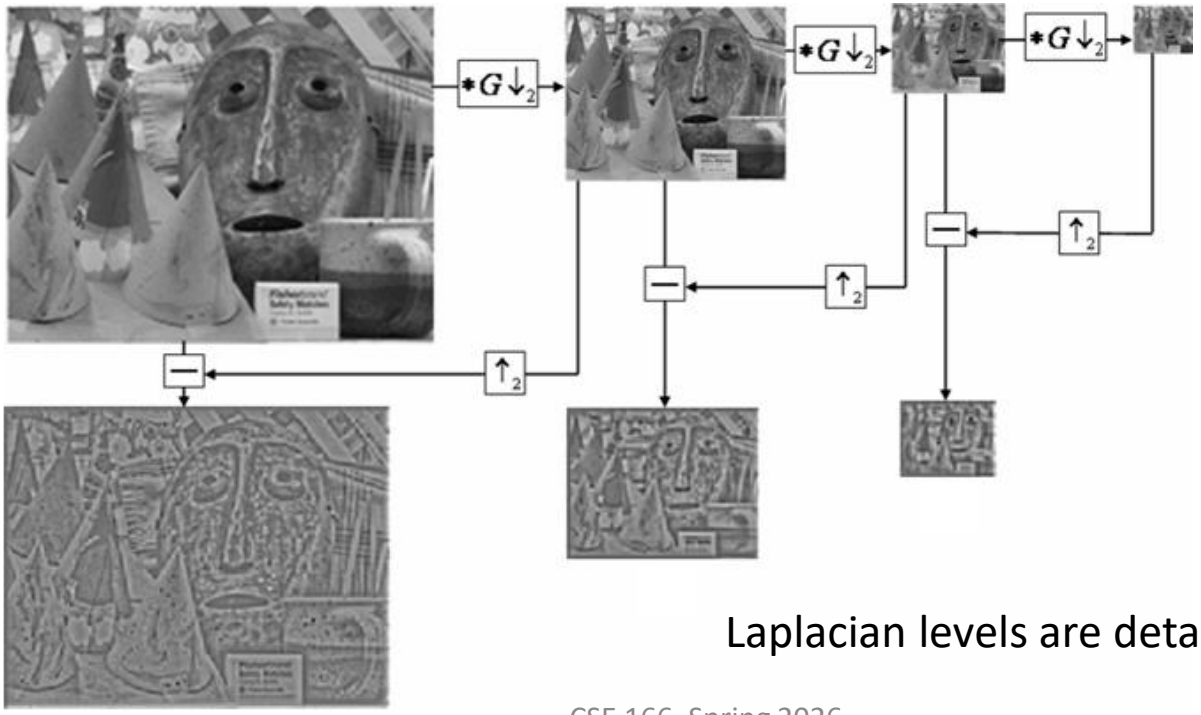
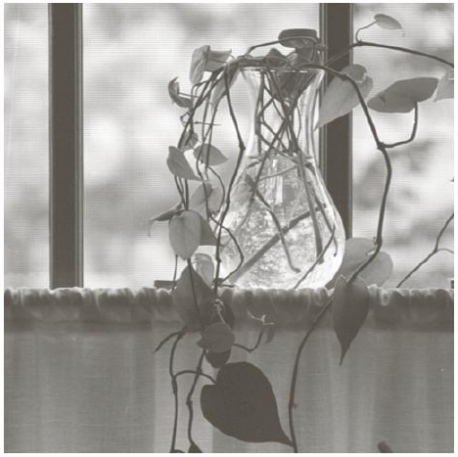


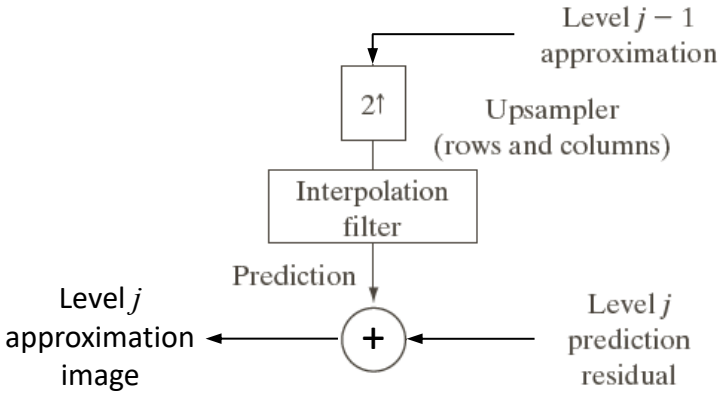
Image pyramids

- Original image can be reconstructed from smallest Gaussian approximation and Laplacian pyramid





Reconstructed original image



Scale space

- Pyramid representation is a predecessor to scale space representation
- Scale space theory is a formal theory for image structures at different scales
  - Image is represented by a one-parameter *family* of Gaussian low pass filtered images
  - A Gaussian filter meets all scale space axioms
    - Linearity, shift invariance, semi-group structure, non-creation of local extrema (zero-crossings), non-enhancement of local extrema, rotational symmetry, and scale invariance
- The *scale parameter* is the variance of the Gaussian filter
  - Note: use border mirror padding on input image when applying filter
- Image details significantly smaller than (two times) the standard deviation (square root of variance) are removed from the image at that scale parameter



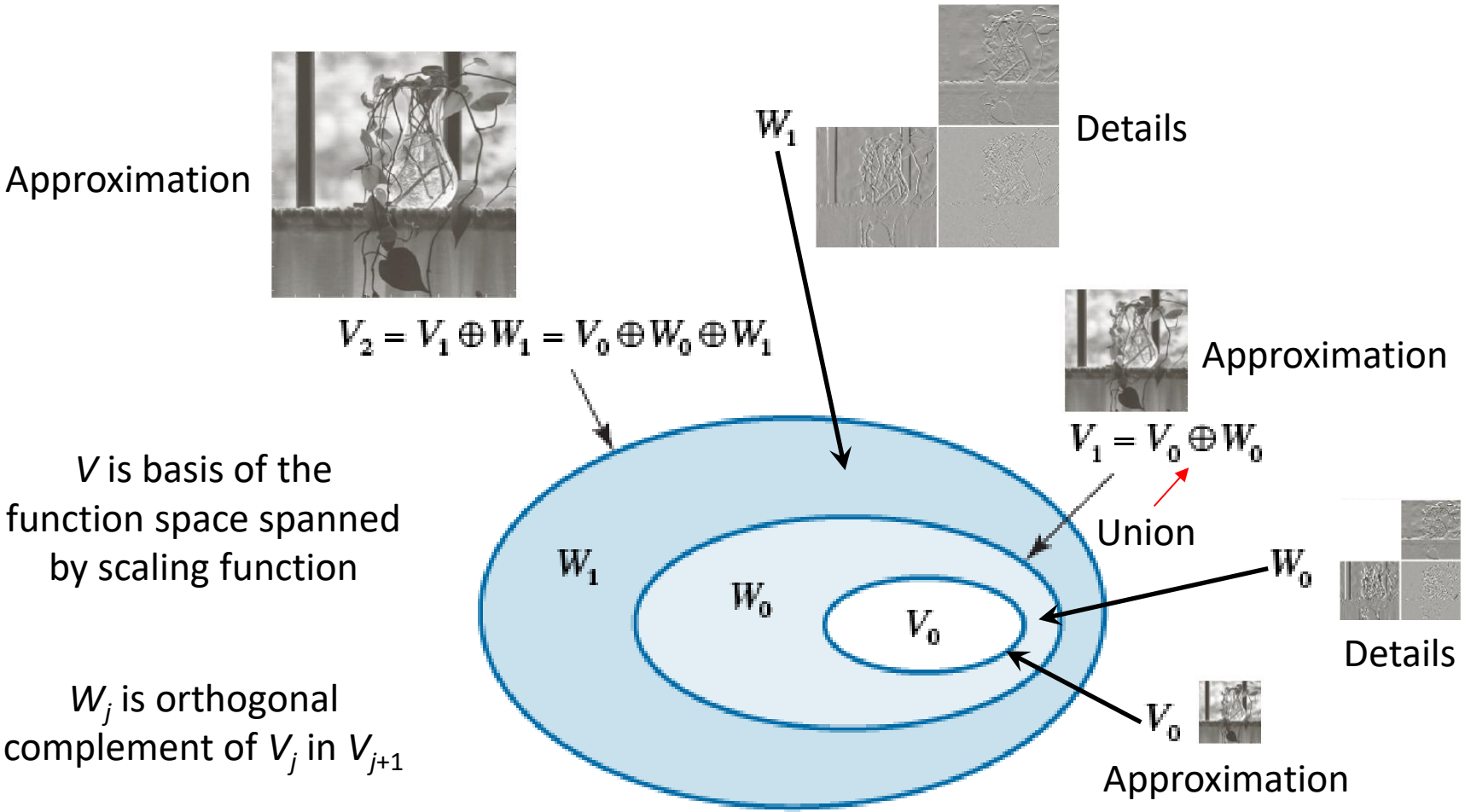
Scale space vs Gaussian pyramid

- Note that Gaussian pyramid (of rate 2) decomposition levels corresponds to Gaussian kernel with standard deviations of 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512
  - Rate of Gaussian pyramid dictates standard deviations of decomposition levels
  - Scale space does not have this imposition; all standard deviations available
- In scale space, since all images are the same size, features are precisely located in the coordinates of the original image
  - Points in levels of Gaussian pyramid must be scaled up to coordinates of the original image, which is imprecise

Wavelet transforms

- A scaling function is used to create a series of approximations of a function or image, each differing by a factor of 2 in resolution from its nearest neighboring approximations
- Wavelet functions (wavelets) are then used to encode the differences between adjacent approximations
- The discrete wavelet transform (DWT) uses those wavelets, together with a single scaling function, to represent a function or image as a linear combination of the wavelets and scaling function

Relationship between scaling and wavelet function spaces

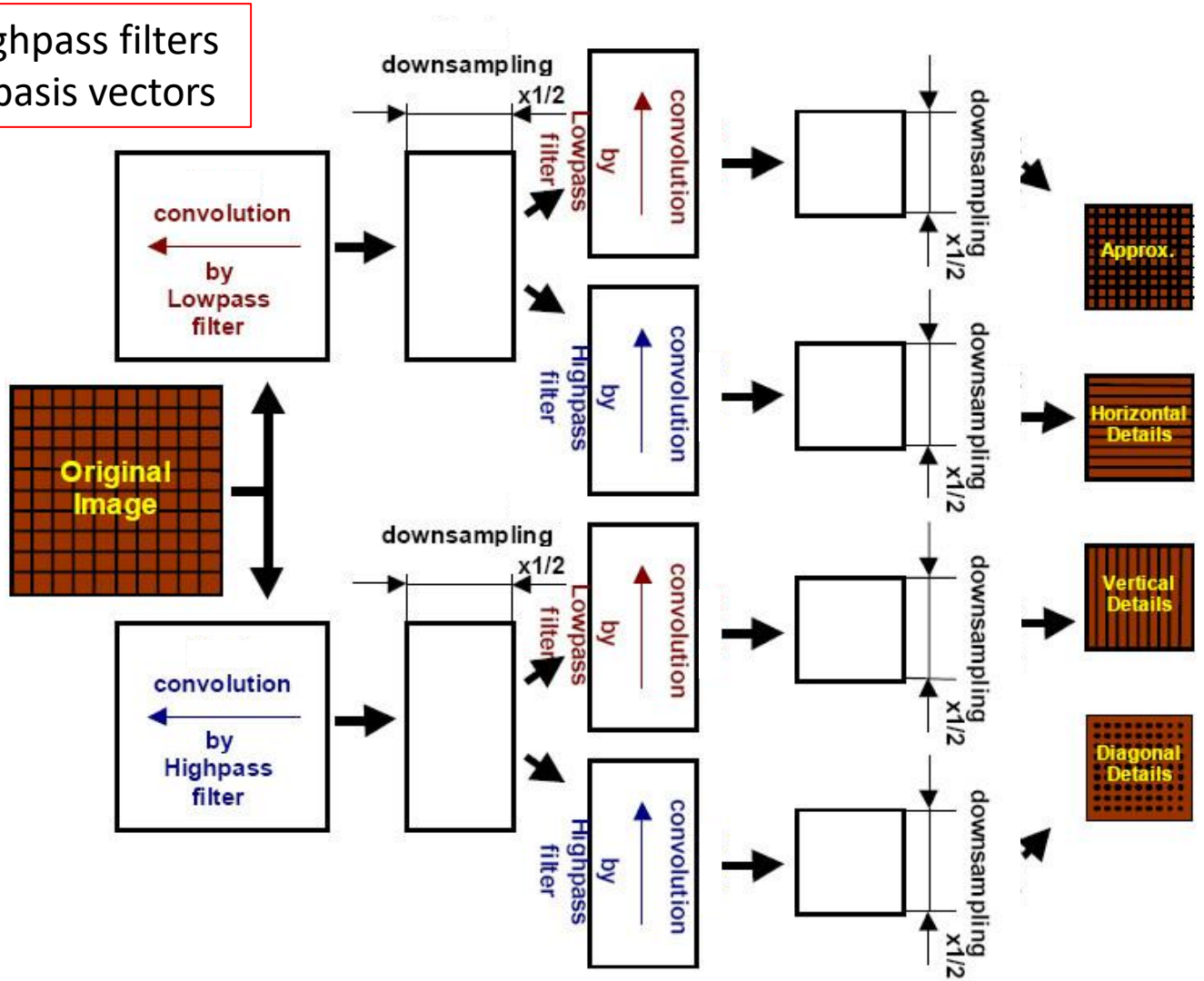


V is basis of the function space spanned by scaling function

W_j is orthogonal complement of V_j in V_{j+1}

Wavelet decomposition

Lowpass and highpass filters are orthogonal basis vectors



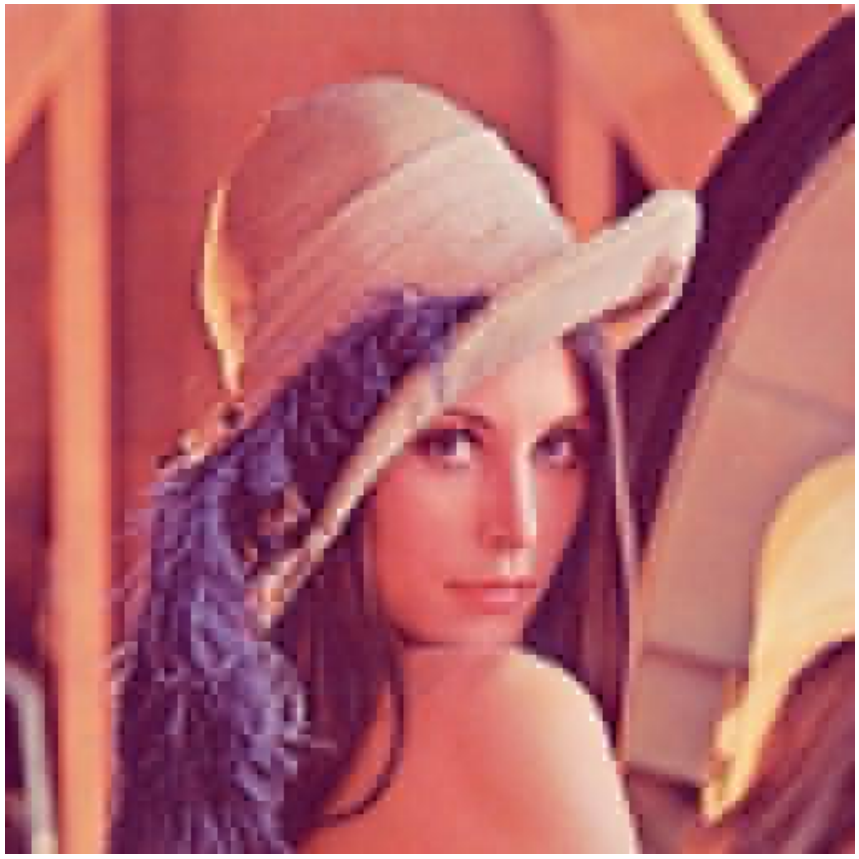
Wavelets in image processing

1. Wavelet transform
2. Alter transform
3. Inverse wavelet transform

Image compression

- Lossless vs lossy compression





Data compression

- Data redundancy

$$R = 1 - \frac{1}{C}$$

where compression (ratio)

$$C = \frac{b}{b'}$$

where

b and b' are the number of bits in two different representations of the same information

Image information

- Entropy

$$\tilde{H} = -\sum_{k=0}^{L-1} p_r(r_k) \log_2(p_r(r_k))$$

Average information (in bits) per pixel

where

L is the number of intensity or gray levels
r_k is input image intensity or gray level value k
$p_r(r_k)$ is normalized histogram of input image

  - It is not possible to encode input image with fewer than $\tilde{H}$ bits/pixel

Visible watermark



Watermark



Watermarked image



Original image minus watermark

Alpha blending

$f_w = (1 - \alpha) f + \alpha w$

Invisible watermark

Example: DCT-based watermarking

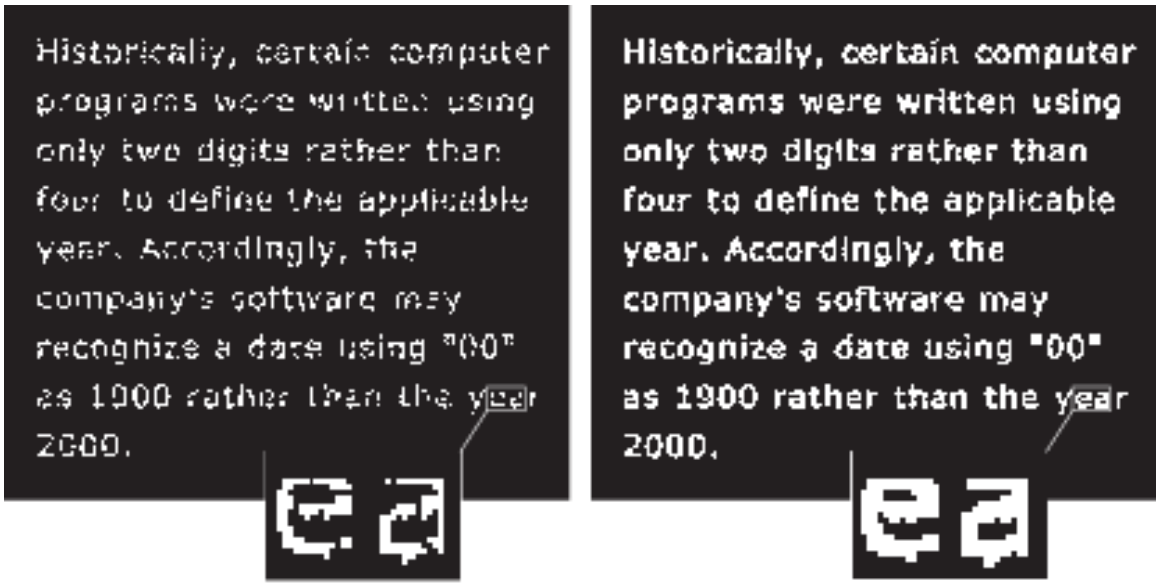
Watermarked images (different watermarks)



Extracted robust invisible watermark

Morphological image processing

- Dilation and erosion
- Opening and closing



1	1	1
1	1	1
1	1	1

Image segmentation

- General approach
  1. Spatial filtering
  2. Additional processing
  3. Thresholding

Image segmentation

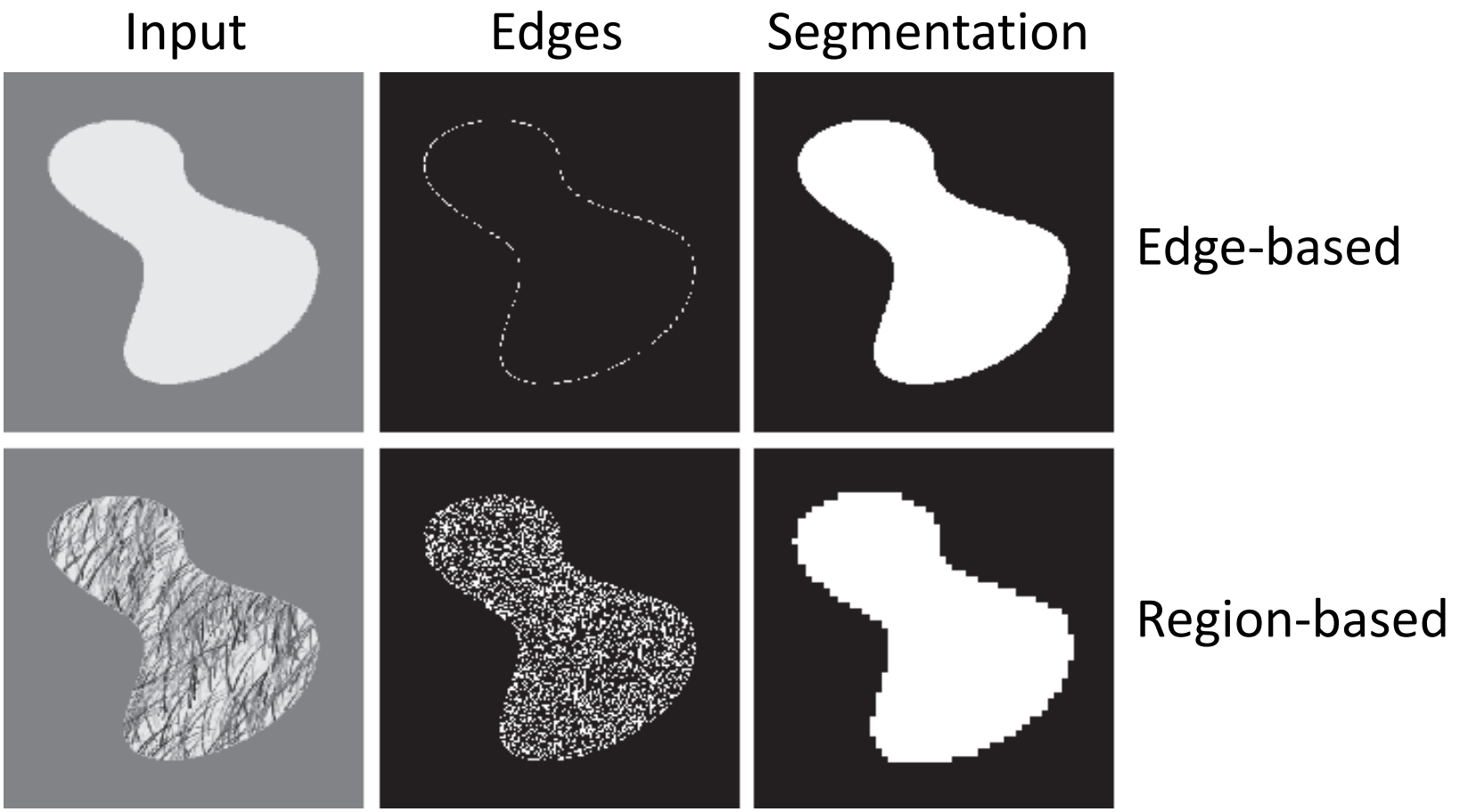
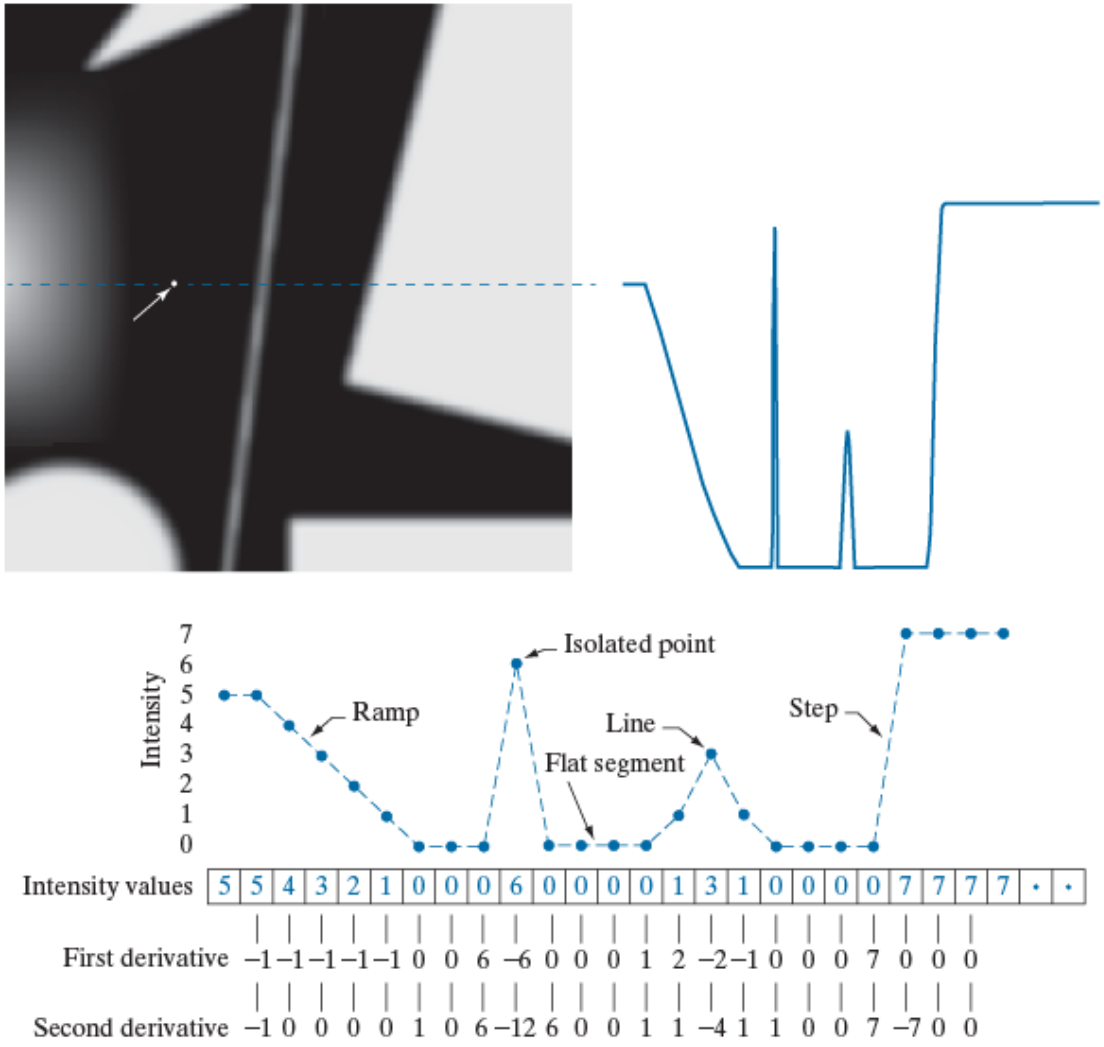
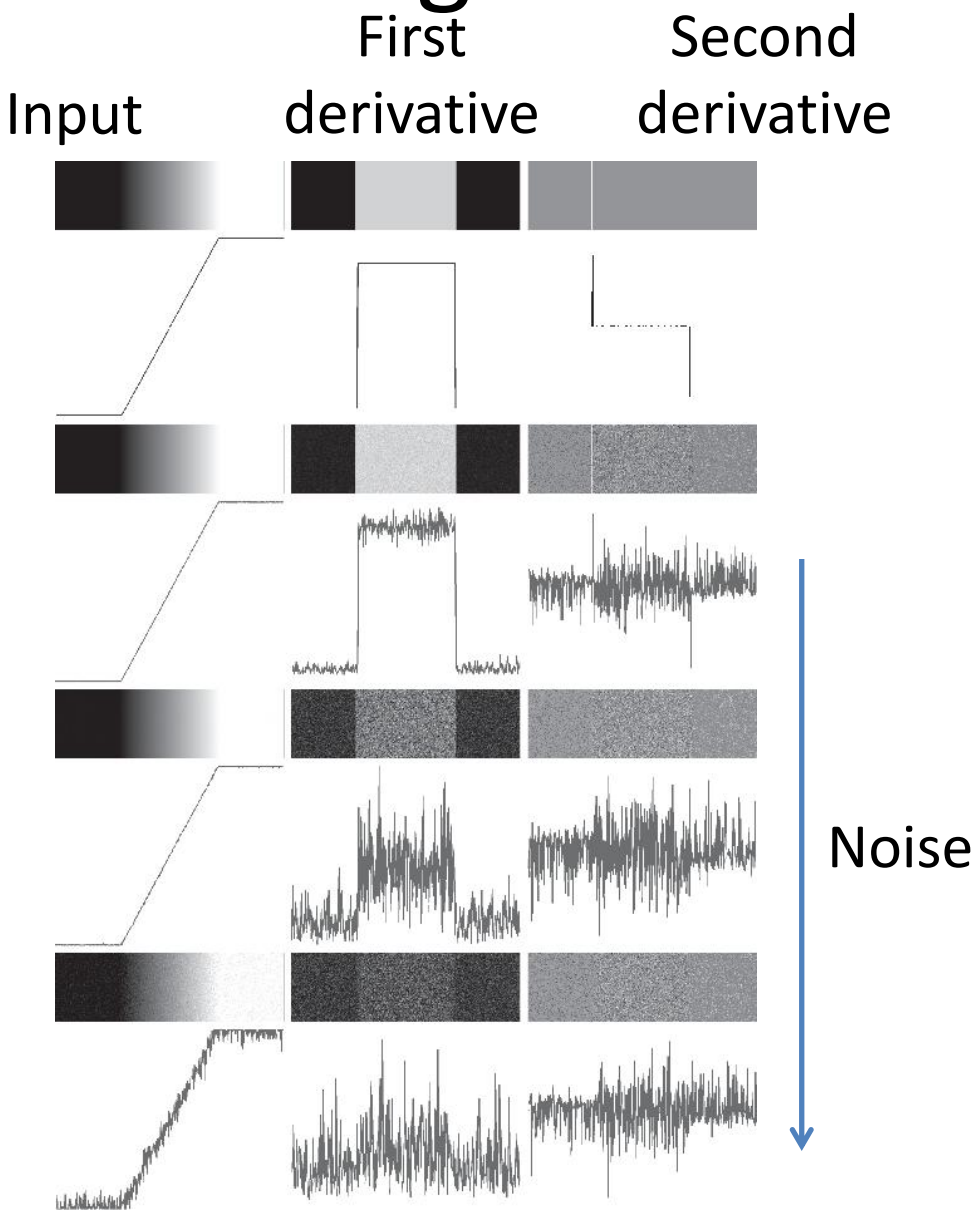


Image derivatives



Noise and image derivatives



Edge detection

- Clearly, noise must be reduced
- Approach (simple)
  1. Image smoothing for noise reduction
  2. Detection of image points (edge point candidates)
  3. Edge localization (select from candidates, set of edge points)

Edge detection

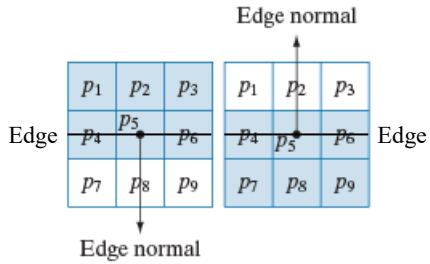
1. Smooth the input image
2. Compute the gradient magnitude and gradient direction images
3. Apply nonmaximal suppression to the gradient magnitude image using the gradient direction image
4. Threshold the resulting image

Nonmaxima suppression along the gradient direction

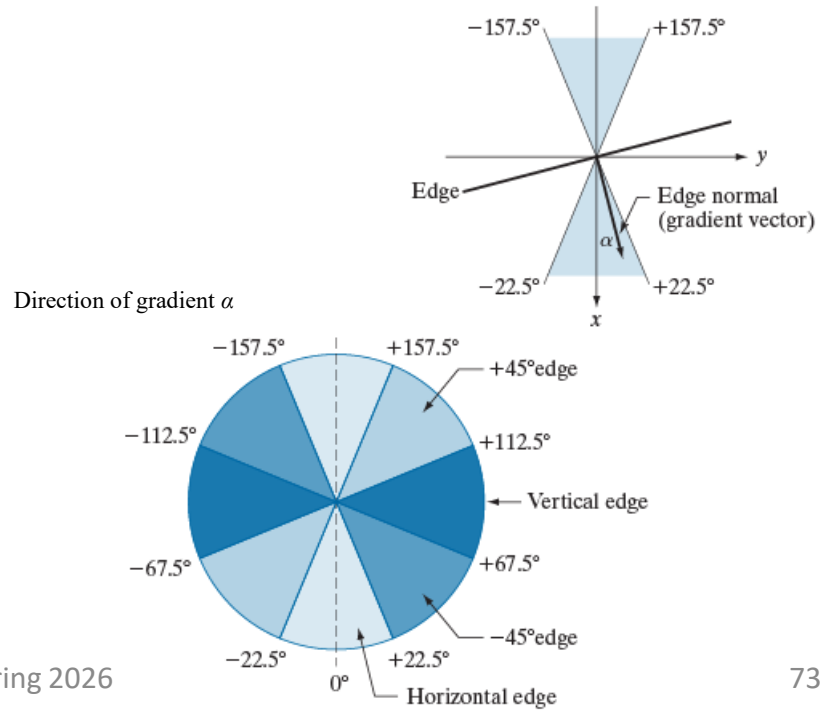
Specify a number of discrete orientations $d_1, d_2, \ldots$

1. Determine the direction d_k closest to $\alpha(x, y)$
2. Let K denote the value of $\|\nabla f\|$ at (x, y). If K is less than the value of $\|\nabla f\|$ at one or both of the neighbors of point (x, y) along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = K$.

Both of the neighbors means the two pixels corresponding to both of the edge normal vectors

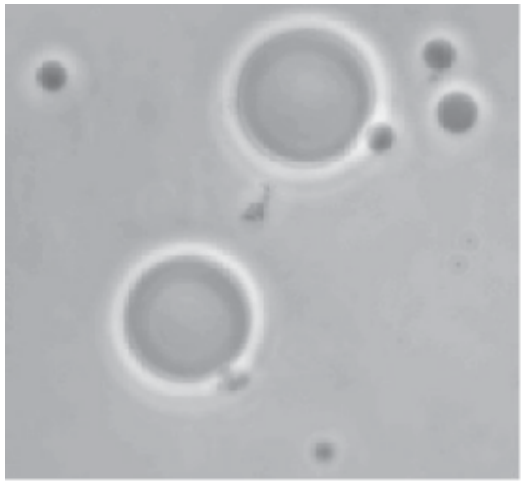


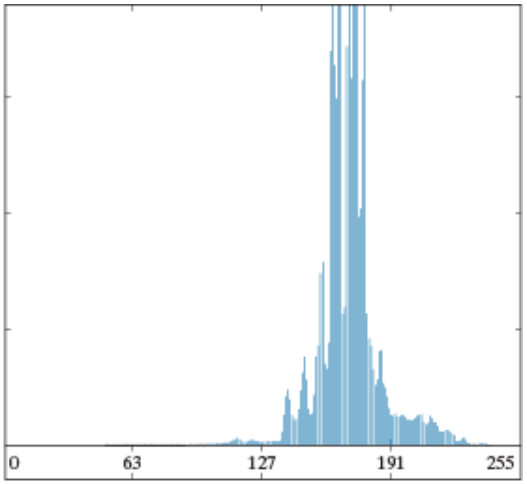
Example: horizontal edge



Optimum global thresholding

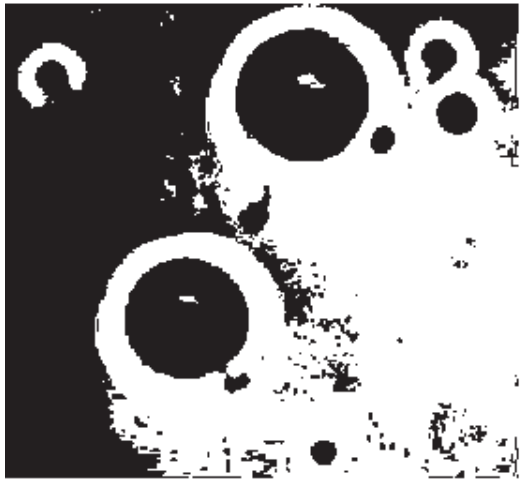
Input





Histogram

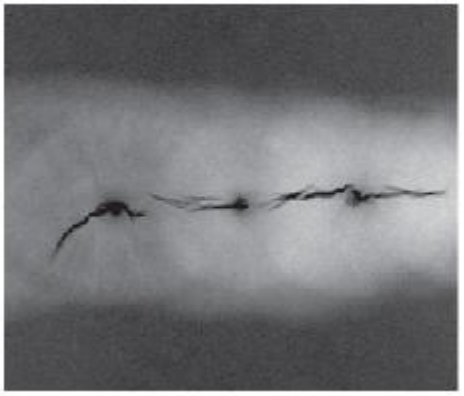
Basic global thresholding



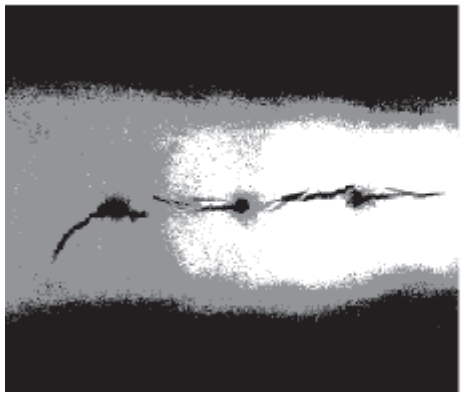


Optimum global thresholding using Otsu's method

Segmentation by region growing



Difference image



Difference image thresholded using dual thresholds

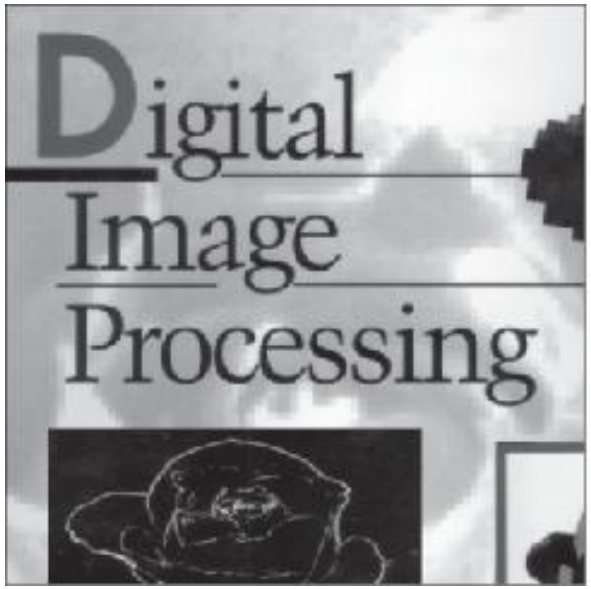


Difference image thresholded with the smallest of the dual thresholds



Segmentation by region growing

Segmentation using *k*-means clustering



Input



Segmentation using
k-means, *k* = 3

Superpixels

- Group pixels into primitive regions that are more perceptually meaningful that individual pixels. Results in less “pixels” to process.



Input image of 480,000 pixels

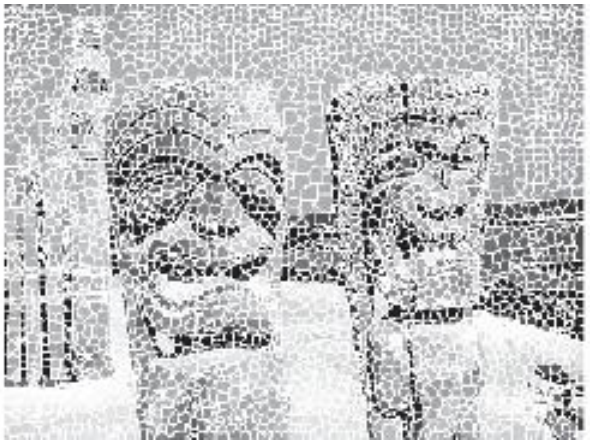


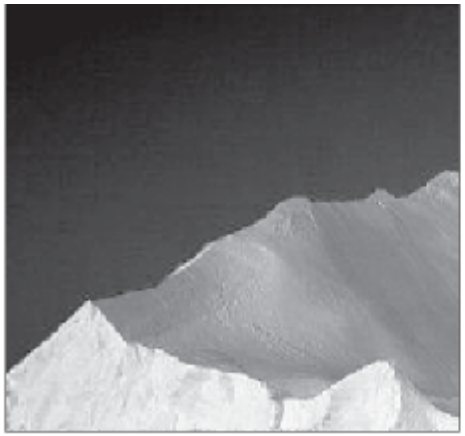
Image of 4,000 superpixels with boundaries



Image of 4,000 superpixels

Superpixels for image segmentation

Input image of 301,678 pixels





Segmentation using *k*-means, *k* = 3







Superpixel image (100 superpixels)

Segmentation using *k*-means, *k* = 3

Graph cuts for image segmentation



Input



Smoothed input



Graph cut segmentation

Feature extraction

- Feature detection



Feature extraction

- Feature description
  - A feature descriptor is
    - **Invariant** with respect to a set of transformations if its value remains unchanged after the application of any transformation from the family
    - **Covariant** with respect to a set of transformations if applying any transformation from the set produces the same result in the descriptor

Features

- Features
  - Local (a member of a set)
  - Global (the entire set)
- Categories
  - Boundaries (not covered in CSE 166)
  - Regions
  - Whole images

CSE 166

- Image acquisition
- Geometric transformations and image interpolation
- Intensity transformations
- Spatial filtering
- Fourier transform and filtering in the frequency domain
- Image restoration
- Color image processing
- Basis vectors and matrix-based transforms
- Multiscale image representations and the wavelet transform
- Image compression
- Image watermarking
- Morphological image processing
- Edge detection
- Image segmentation
- Feature extraction