
Introduction to Image Processing

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Course Description

- Intense introduction to image processing.
- Intended for advanced undergraduate and graduate students.
- Topics include:
 - Image enhancement
 - Digital filtering theory, Fourier transforms
 - Image reconstruction, resampling, antialiasing
 - Scanline algorithms, geometric transforms
 - Warping, morphing, and visual effects

Syllabus

Week Topic

- 1 Introduction / overview
- 2-3 Point operations
- 4 Neighborhood operations
- 5-6 Fast Fourier transforms (FFT)
- 7-8 Sampling theory
- 9 Midterm, Image reconstruction
- 10 Fast filtering for resampling
- 11 Spatial transformations, texture mapping
- 12-14 Separable warping algorithms; visual effects

Required Text

Rafael Gonzalez and Richard Woods, *Digital Image Processing*, 3rd Edition, Prentice Hall, Wesley, 2008.

Supplementary Texts

Milan Sonka, Vaclav Hlavac, and Roger Boyle,
Image Processing, Analysis, and Machine Vision,
Cengage Learning, 2014.

George Wolberg, *Digital Image Warping*, IEEE
Computer Society Press, 1990.

Grading

- The final grade is computed as follows:
 - Midterm exam: 25%
 - Final exam: 25%
 - Homework programming assignments: 50%
- Substantial programming assignments are due every three weeks.
- Proficiency in C/C++ is expected.
- Prereqs: CSc 22100

Contact Information

- Prof. Wolberg
 - Office hours: After class and by appointment
 - Email: wolberg@cs.ccny.cuny.edu
- Teaching Assistant (TA): Siavash Zokai
 - Email: ccny.cs470@gmail.com
- See class web page for all class info such as homework and sample source code:
www-cs.ccny.cuny.edu/~wolberg/cs470

Objectives

- These notes accompany the textbooks:
 - “Digital Image Processing” by Gonzalez/Woods
 - “Digital Image Warping” by George Wolberg
- They form the basis for approximately 14 weeks of lectures.
- Programs in C/C++ will be assigned to reinforce understanding of the material.
 - Four homework assignments
 - Each due in 3 weeks and requiring ~4 programs

What is Image Processing?

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we:
 - Explore what image processing is about
 - Compare it against related fields
 - Provide historical introduction
 - Survey some application areas

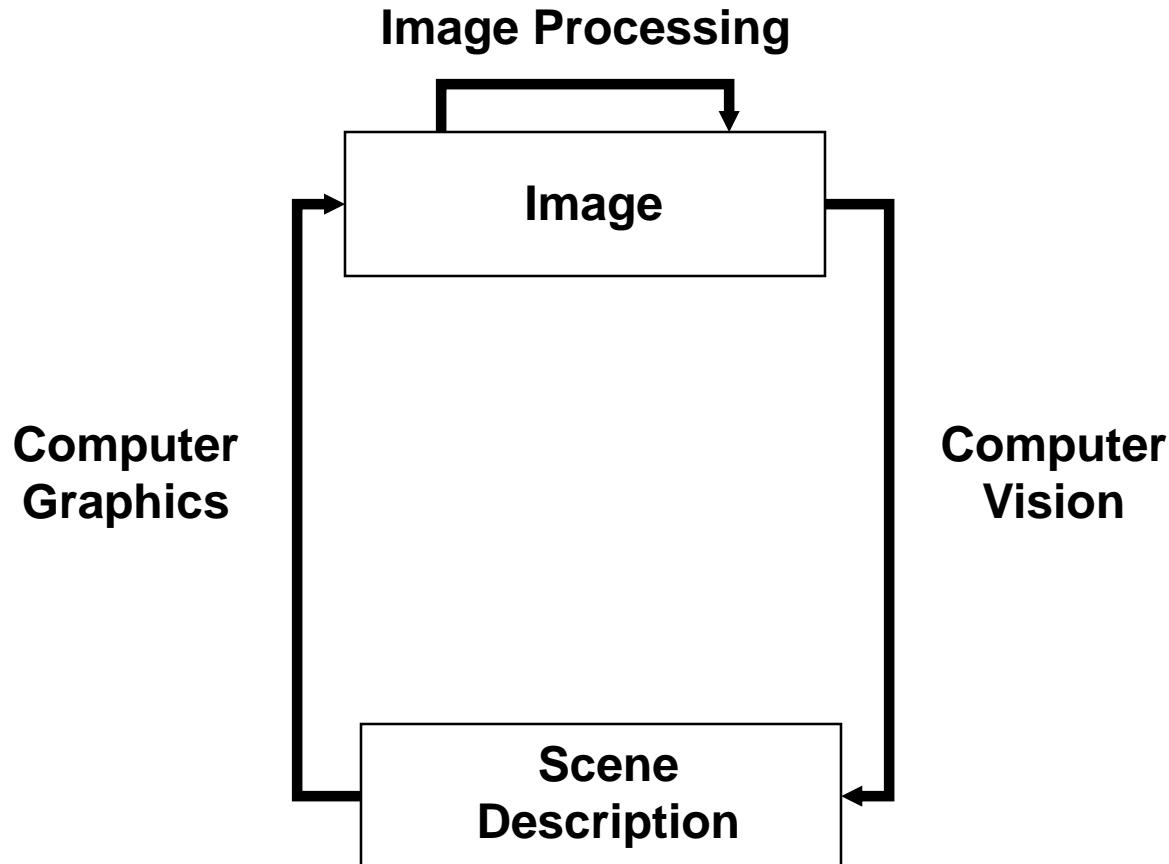
What is Digital Image Processing?

- Computer manipulation of pictures, or images, that have been converted into numeric form.
Typical operations include:
 - Contrast enhancement
 - Remove blur from an image
 - Smooth out graininess, speckle, or noise
 - Magnify, minify, or rotate an image (image warping)
 - Geometric correction
 - Image compression for efficient storage/transmission

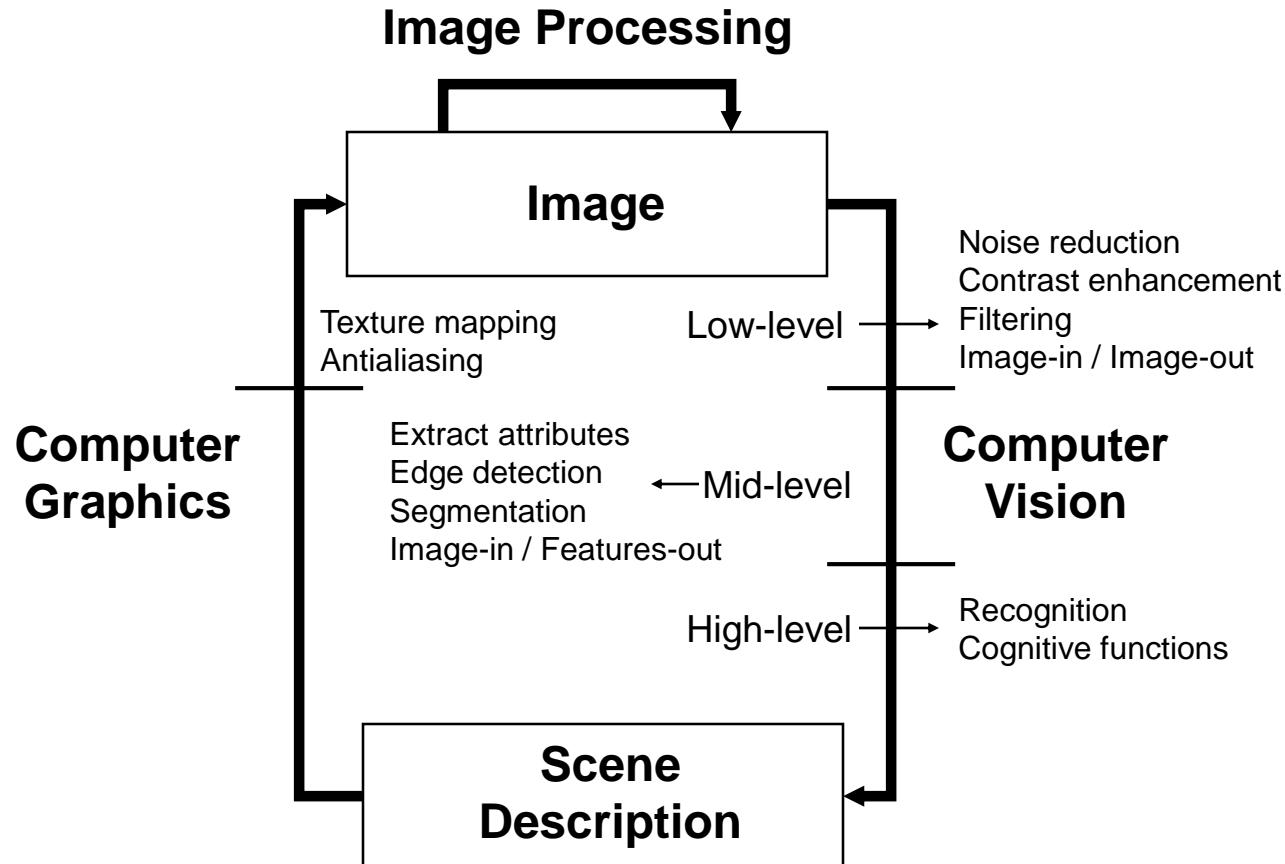
Image Processing Goals

- Image processing is a subclass of signal processing concerned specifically with pictures
- It aims to improve image quality for
 - human perception: subjective
 - computer interpretation: objective
- Compress images for efficient storage/transmission

Related Fields



Overlap with Related Fields



Distinctions

- No clear cut boundaries between image processing on the one end and computer vision at the other
- Defining image processing as image-in/image-out does not account for
 - computation of average intensity: image-in / number-out
 - image compression: image-in / coefficients-out
- Nevertheless, image-in / image-out is true most of time

Input \ Output	Image	Description
Image	Image Processing	Computer Vision
Description	Computer Graphics	Artificial Intelligence

Image Processing: 1960-1970

Geometric correction and image enhancement applied to Ranger 7 pictures of the moon.

Work conducted at the Jet Propulsion Laboratory.

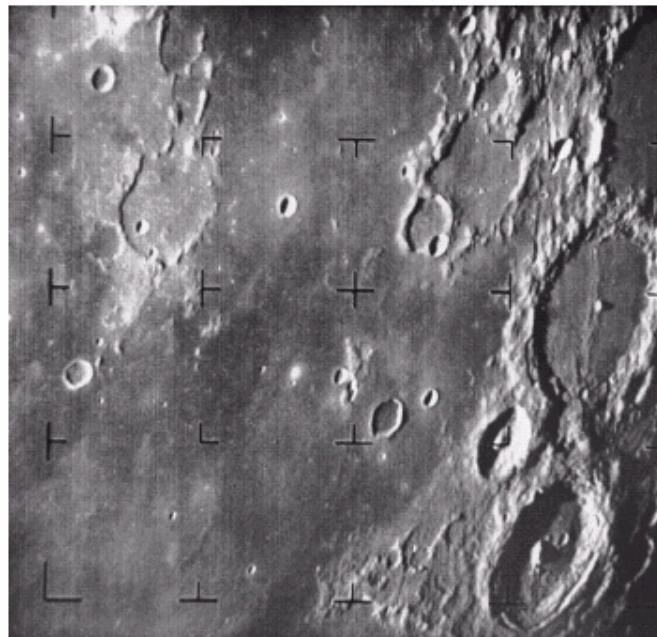
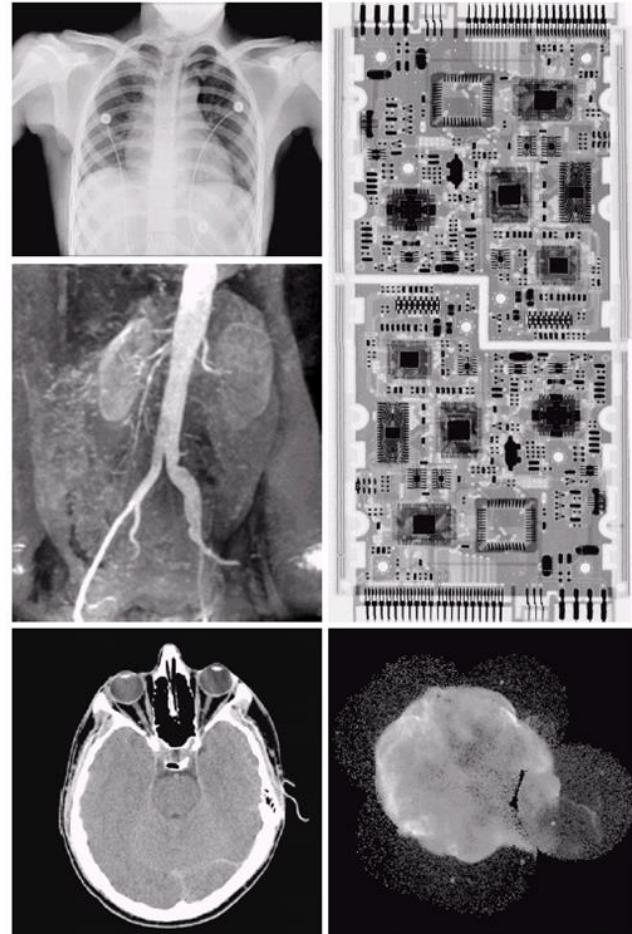


FIGURE 1.4 The first picture of the moon by a U.S. spacecraft. *Ranger 7* took this image on July 31, 1964 at 9:09 A.M. EDT, about 17 minutes before impacting the lunar surface. (Courtesy of NASA.)

Image Processing: 1970-1980

- Invention of computerized axial tomography (CAT)
- Emergence of medical imaging
- Rapid growth of X-ray imaging for CAT scans, inspection, and astronomy
- LANDSAT earth observation



a
b
c
d
e

FIGURE 1.7 Examples of X-ray imaging. (a) Chest X-ray. (b) Aortic angiogram. (c) Head CT. (d) Circuit boards. (e) Cygnus Loop. (Images courtesy of (a) and (c) Dr. David R. Pickens, Dept. of Radiology & Radiological Sciences, Vanderbilt University Medical Center, (b) Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, (d) Mr. Joseph E. Pascente, Lixi, Inc., and (e) NASA.)

Image Processing: 1980-1990

- Satellite infrared imaging: LANDSAT, NOAA
- Fast resampling and texture mapping

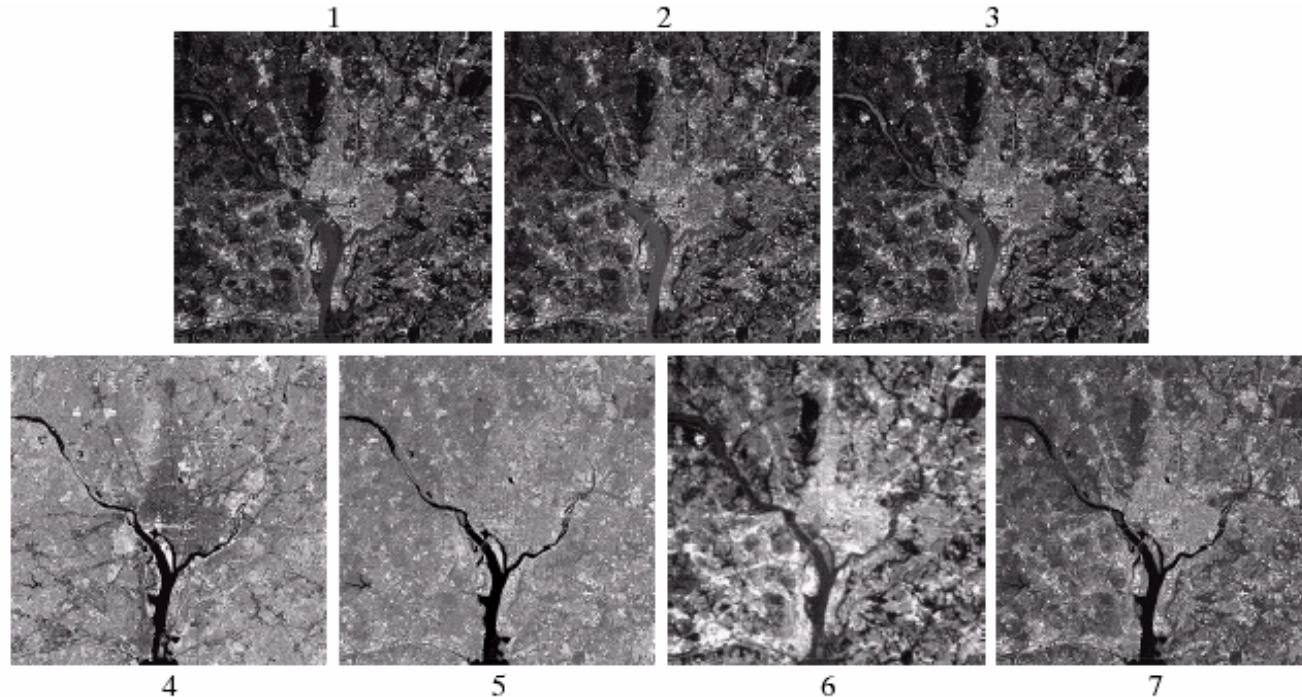


FIGURE 1.10 LANDSAT satellite images of the Washington, D.C. area. The numbers refer to the thematic bands in Table 1.1. (Images courtesy of NASA.)

Image Processing: 1990-2000

- Morphing / visual effects algorithms
- JPEG/MPEG compression, wavelet transforms
- Adobe PhotoShop



Image Processing: 2000-

- Widespread proliferation of fast graphics processing units (GPU) from nVidia and ATI to perform real-time image processing
- Ubiquitous digital cameras, camcorders, and cell phone cameras rely heavily on image processing and compression

Sources of Images

- The principal energy source for images is the electromagnetic energy spectrum.
- EM waves = stream of massless (proton) particles, each traveling in a wavelike pattern at the speed of light. Spectral bands are grouped by energy/photon
 - Gamma rays, X-rays, UV, Visible, Infrared, Microwaves, radio waves
- Other sources: acoustic, ultrasonic, electronic

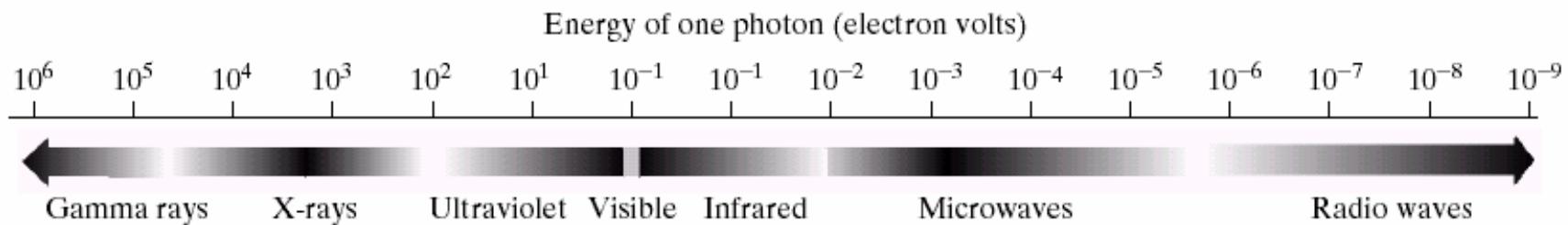


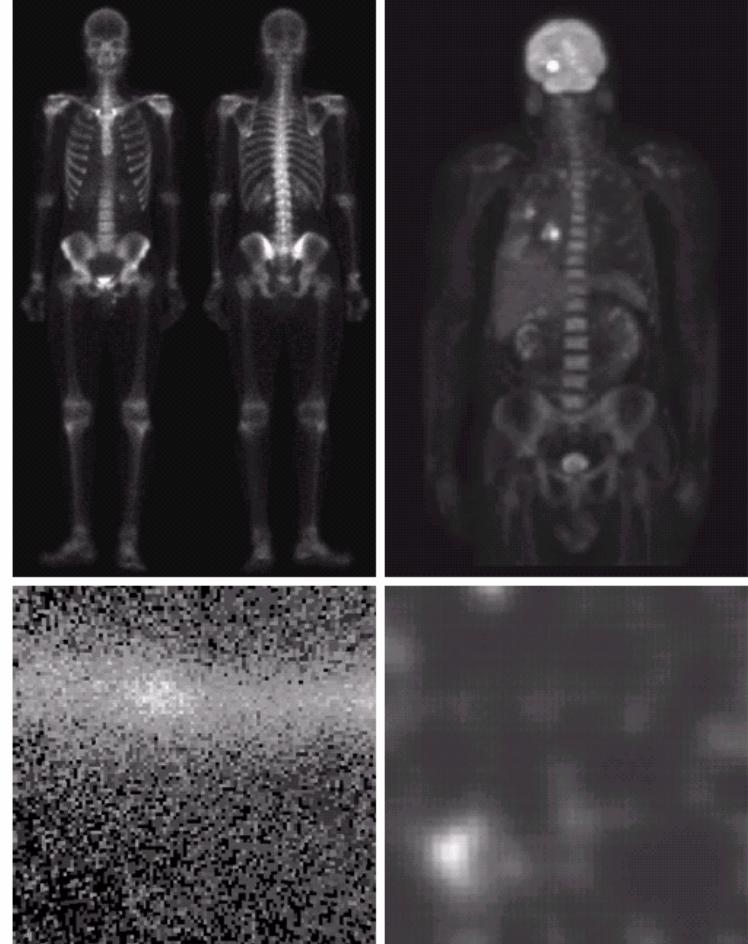
FIGURE 1.5 The electromagnetic spectrum arranged according to energy per photon.

Gamma-Ray Imaging

- Used in nuclear medicine, astronomy
- Nuclear medicine:
patient is injected with radioactive isotope
that emits gamma rays as it decays.
Images are produced from emissions collected by detectors.

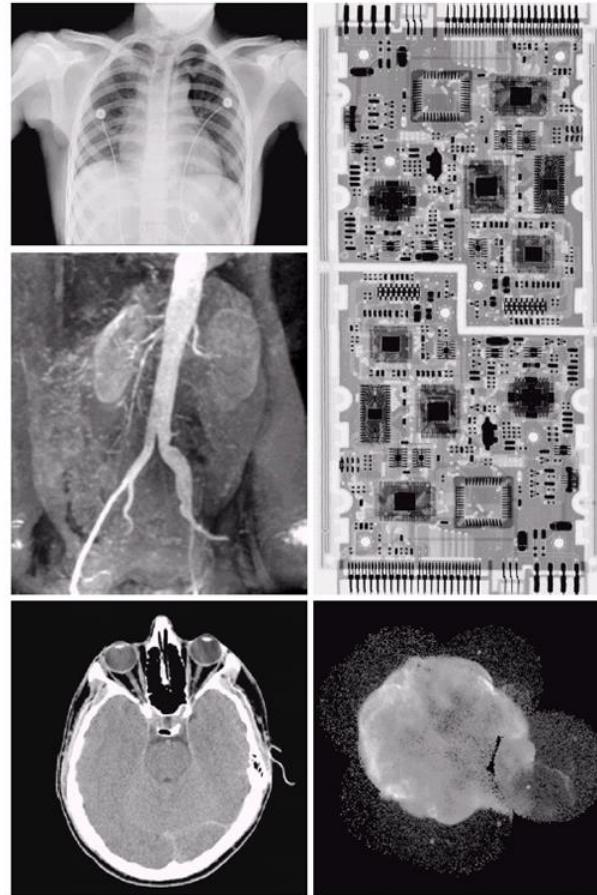
a
b
c
d

FIGURE 1.6
Examples of gamma-ray imaging. (a) Bone scan. (b) PET image. (c) Cygnus Loop. (d) Gamma radiation (bright spot) from a reactor valve.
(Images courtesy of (a) G.E. Medical Systems, (b) Dr. Michael E. Casey, CTI PET Systems, (c) NASA, (d) Professors Zhong He and David K. Wehe, University of Michigan.)



X-Ray Imaging

- Oldest source of EM radiation for imaging
- Used for CAT scans
- Used for angiograms where X-ray contrast medium is injected through catheter to enhance contrast at site to be studied.
- Industrial inspection



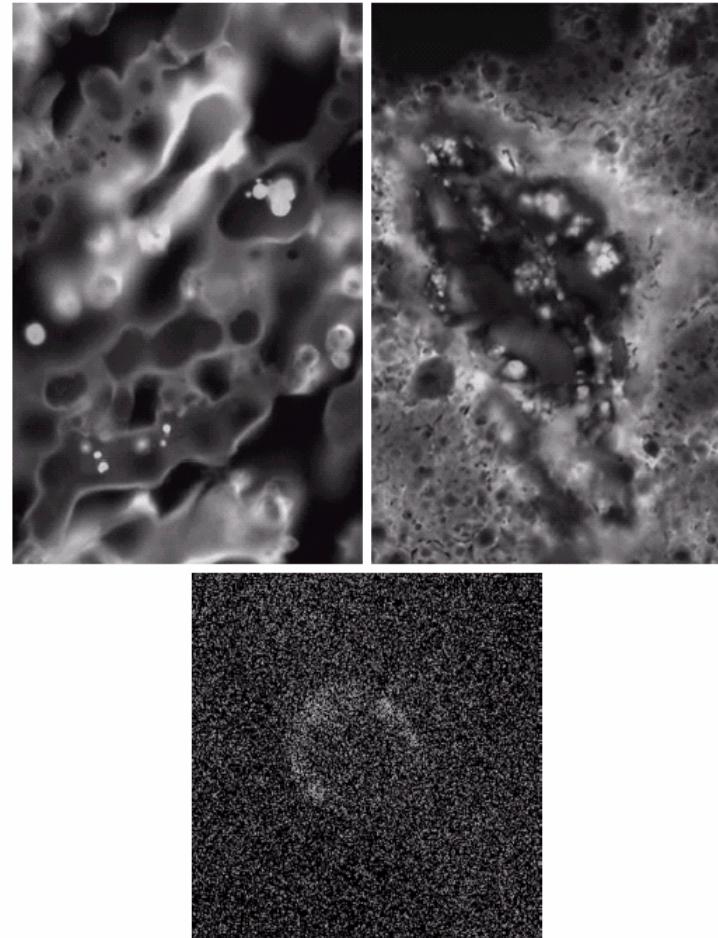
a b c d e
FIGURE 1.7 Examples of X-ray imaging. (a) Chest X-ray. (b) Aortic angiogram. (c) Head CT. (d) Circuit boards. (e) Cygnus Loop. (Images courtesy of (a) and (c) Dr. David R. Pickens, Dept. of Radiology & Radiological Sciences, Vanderbilt University Medical Center, (b) Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, (d) Mr. Joseph E. Pascente, Lixi, Inc., and (e) NASA.)

Ultraviolet Imaging

- Used for lithography, industrial inspection, fluorescence microscopy, lasers, biological imaging, and astronomy
- Photon of UV light collides with electron of fluorescent material to elevate its energy. Then, its energy falls and it emits red light.

a
b
c

FIGURE 1.8
Examples of ultraviolet imaging.
(a) Normal corn.
(b) Smut corn.
(c) Cygnus Loop.
(Images courtesy of (a) and
(b) Dr. Michael W. Davidson,
Florida State University,
(c) NASA.)



Visible and Infrared Imaging (1)

- Used for astronomy, light microscopy, remote sensing

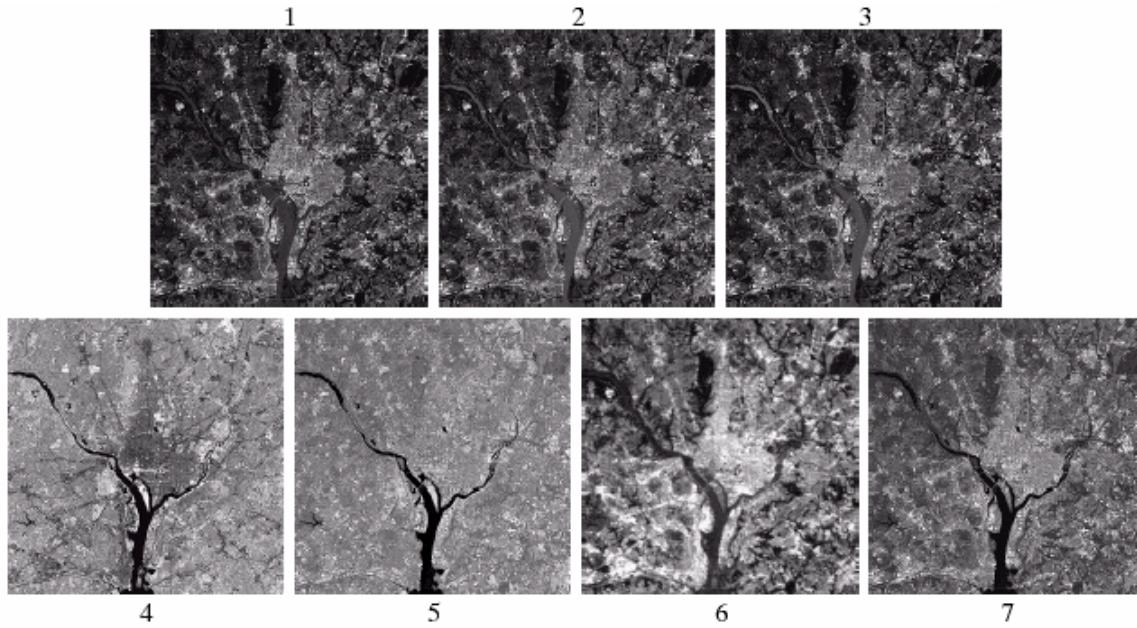


FIGURE 1.10 LANDSAT satellite images of the Washington, D.C. area. The numbers refer to the thematic bands in Table 1.1. (Images courtesy of NASA.)

TABLE 1.1
Thematic bands
in NASA's
LANDSAT
satellite.

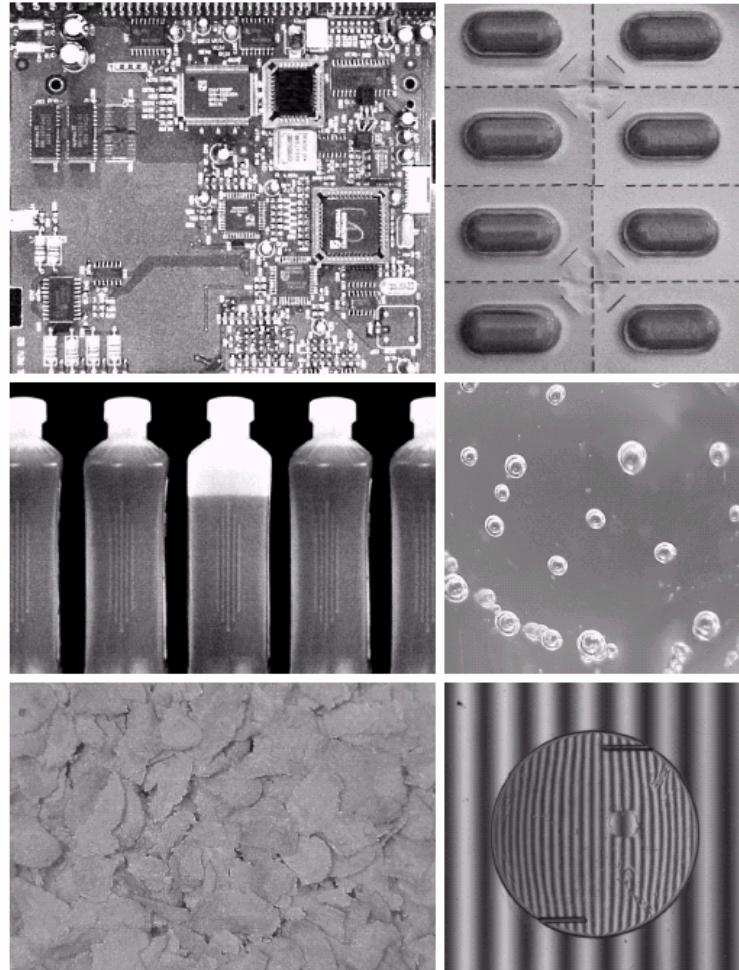
Band No.	Name	Wavelength (μm)	Characteristics and Uses
1	Visible blue	0.45–0.52	Maximum water penetration
2	Visible green	0.52–0.60	Good for measuring plant vigor
3	Visible red	0.63–0.69	Vegetation discrimination
4	Near infrared	0.76–0.90	Biomass and shoreline mapping
5	Middle infrared	1.55–1.75	Moisture content of soil and vegetation
6	Thermal infrared	10.4–12.5	Soil moisture; thermal mapping
7	Middle infrared	2.08–2.35	Mineral mapping

Visible and Infrared Imaging (2)

- Industrial inspection
 - inspect for missing parts
 - missing pills
 - unacceptable bottle fill
 - unacceptable air pockets
 - anomalies in cereal color
 - incorrectly manufactured replacement lens for eyes

a
b
c
d
e
f

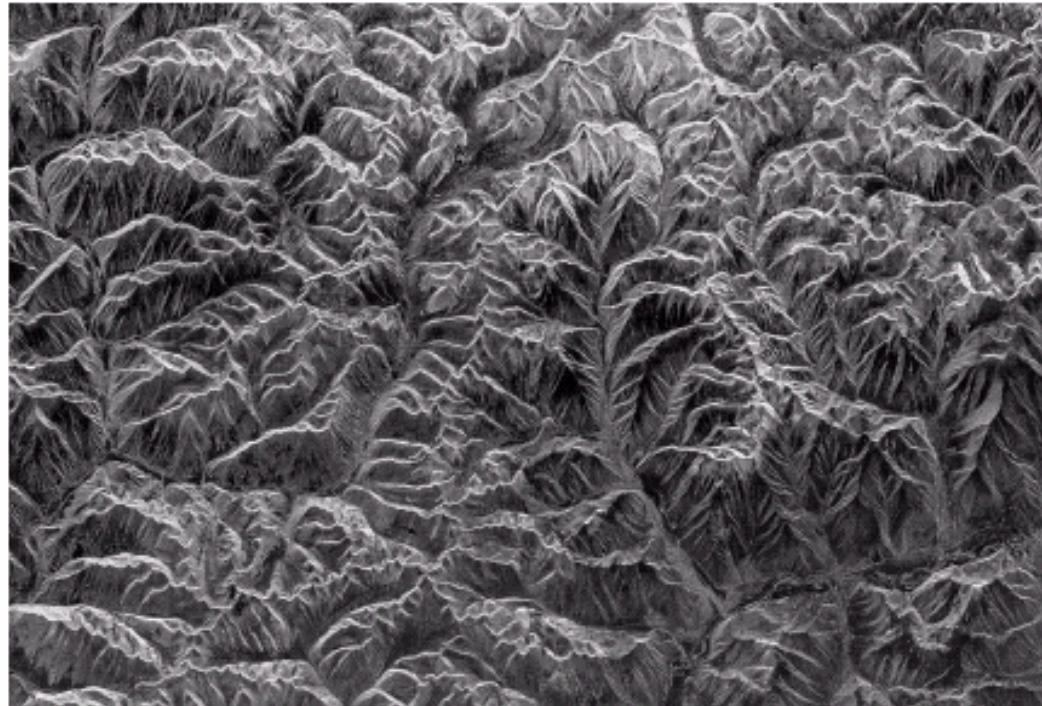
FIGURE 1.14
Some examples of manufactured goods often checked using digital image processing. (a) A circuit board controller.
(b) Packaged pills.
(c) Bottles.
(d) Bubbles in clear-plastic product.
(e) Cereal.
(f) Image of intraocular implant.
(Fig. (f) courtesy of Mr. Pete Sites, Perceptics Corporation.)



Microwave Imaging

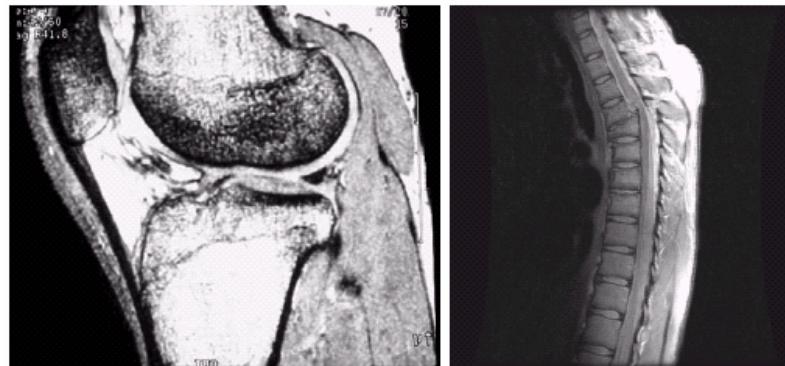
- Radar is dominant application
- Microwave pulses are sent out to illuminate scene
- Antenna receives reflected microwave energy

FIGURE 1.16
Spaceborne radar
image of
mountains in
southeast Tibet.
(Courtesy of
NASA.)



Radio-Band Imaging

- Magnetic resonance imaging (MRI):
 - places patient in powerful magnet
 - passes radio waves through body in short pulses
 - each pulse causes a responding pulse of radio waves to be emitted by patient's tissues
 - Location and strength of signal is recorded to form image



a b

FIGURE 1.17 MRI images of a human (a) knee, and (b) spine. (Image (a) courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School, and (b) Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Images Covering EM Spectrum

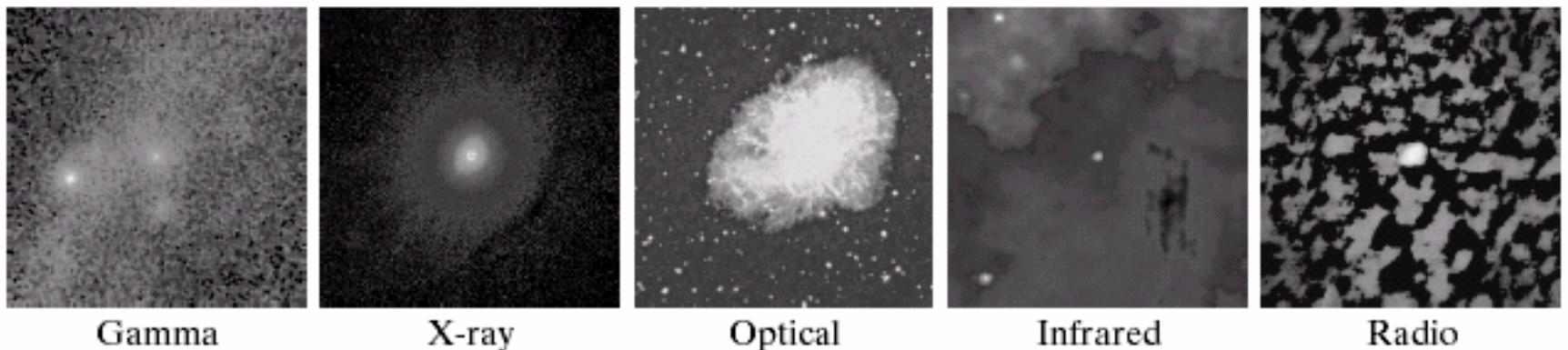


FIGURE 1.18 Images of the Crab Pulsar (in the center of images) covering the electromagnetic spectrum. (Courtesy of NASA.)

Non-EM modality: Ultrasound

- Used in geological exploration, industry, medicine:
 - transmit high-freq (1-5 MHz) sound pulses into body
 - record reflected waves
 - calculate distance from probe to tissue/organ using the speed of sound (1540 m/s) and time of echo's return
 - display distance and intensities of echoes as a 2D image

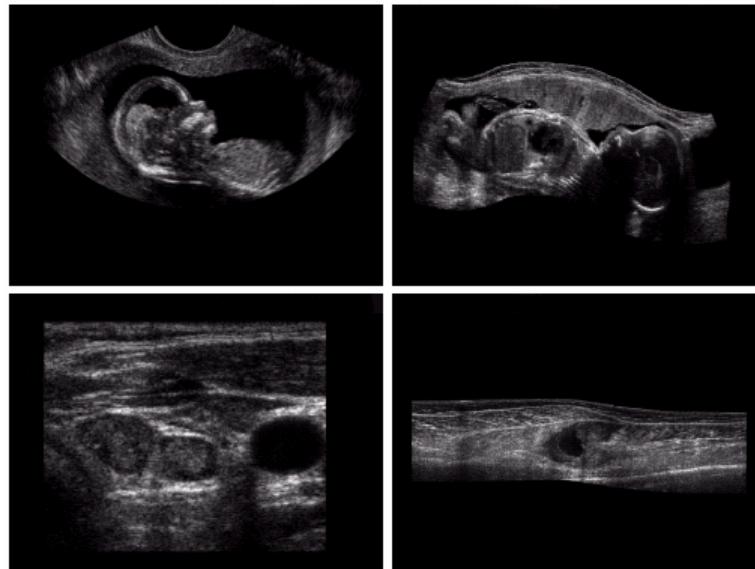
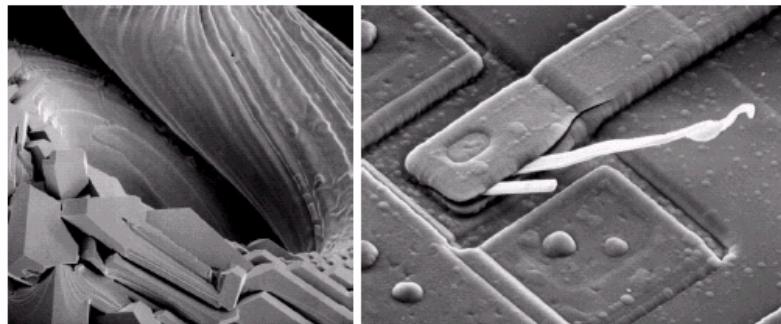


FIGURE 1.20
Examples of ultrasound imaging. (a) Baby.
(b) Another view of baby.
(c) Thyroids.
(d) Muscle layers showing lesion.
(Courtesy of Siemens Medical Systems, Inc., Ultrasound Group.)

Non-EM modality: Scanning Electron Microscope

- Stream of electrons is accelerated toward specimen using a positive electrical potential
- Stream is focused using metal apertures and magnetic lenses into a thin beam
- Scan beam; record interaction of beam and sample at each location (dot on phosphor screen)



a b

FIGURE 1.21 (a) 250 \times SEM image of a tungsten filament following thermal failure. (b) 2500 \times SEM image of damaged integrated circuit. The white fibers are oxides resulting from thermal destruction. (Figure (a) courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene; (b) courtesy of Dr. J. M. Hudak, McMaster University, Hamilton, Ontario, Canada.)

Visible Spectrum

- Thin slice of the full electromagnetic spectrum

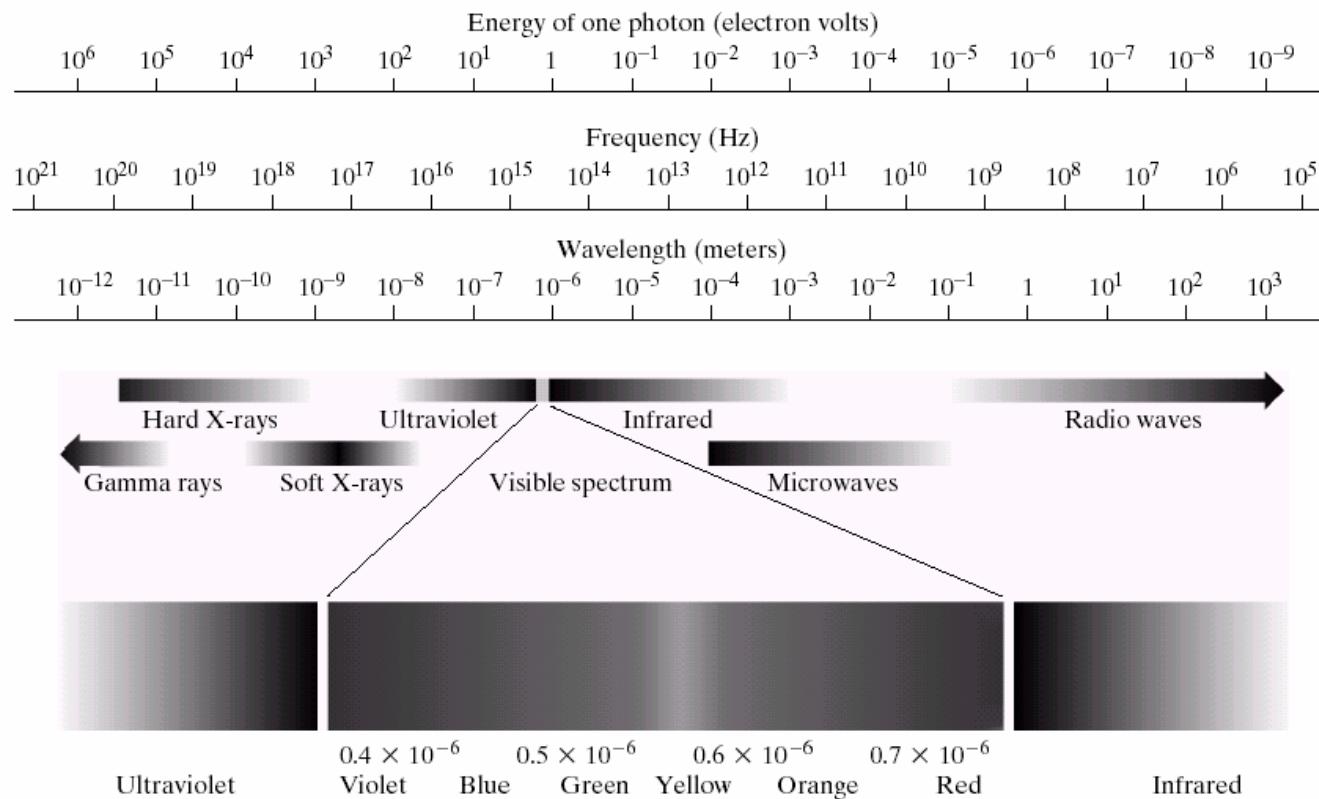


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.

Human Visual System

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

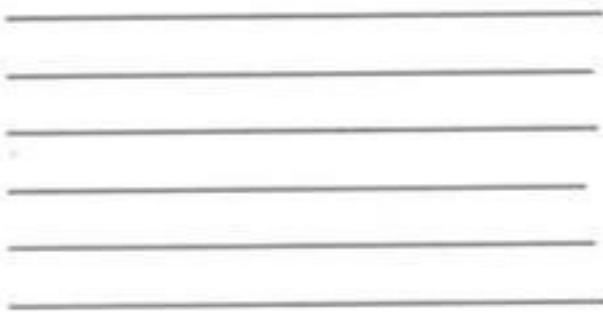
- In this lecture we discuss:
 - Structure of human eye
 - Mechanics of human visual system (HVS)
 - Brightness adaptation and discrimination
 - Perceived brightness and simultaneous contrast

Human and Computer Vision

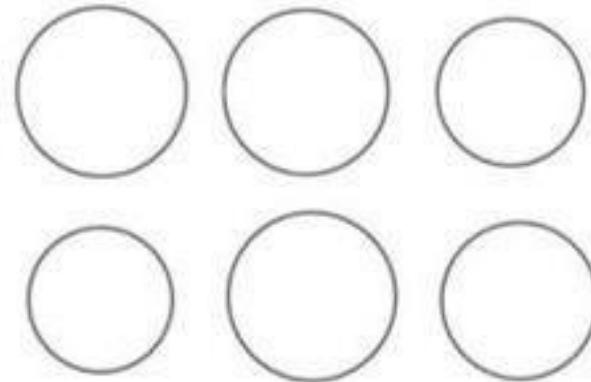
- We observe and evaluate images with our visual system
- We must therefore understand the functioning of the human visual system and its capabilities for brightness adaptation and discrimination:
 - What intensity differences can we distinguish?
 - What is the spatial resolution of our eye?
 - How accurately do we estimate distances and areas?
 - How do we sense colors?
 - By which features can we detect/distinguish objects?

Examples

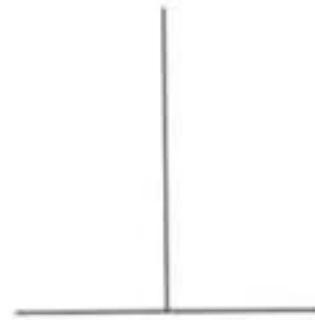
a Parallel lines : <5% variation in length



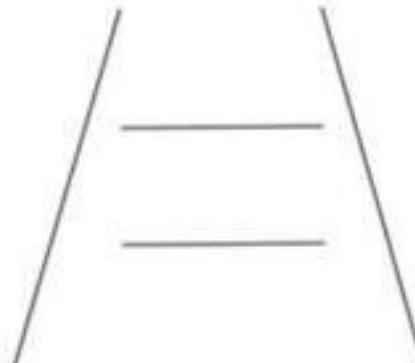
b Circles: <10% variation in radii



c Vertical line falsely appears longer



d Upper line falsely appears longer



Structure of the Human Eye

- Shape is nearly spherical
- Average diameter = 20mm
- Three membranes:
 - Cornea and Sclera
 - Choroid
 - Retina

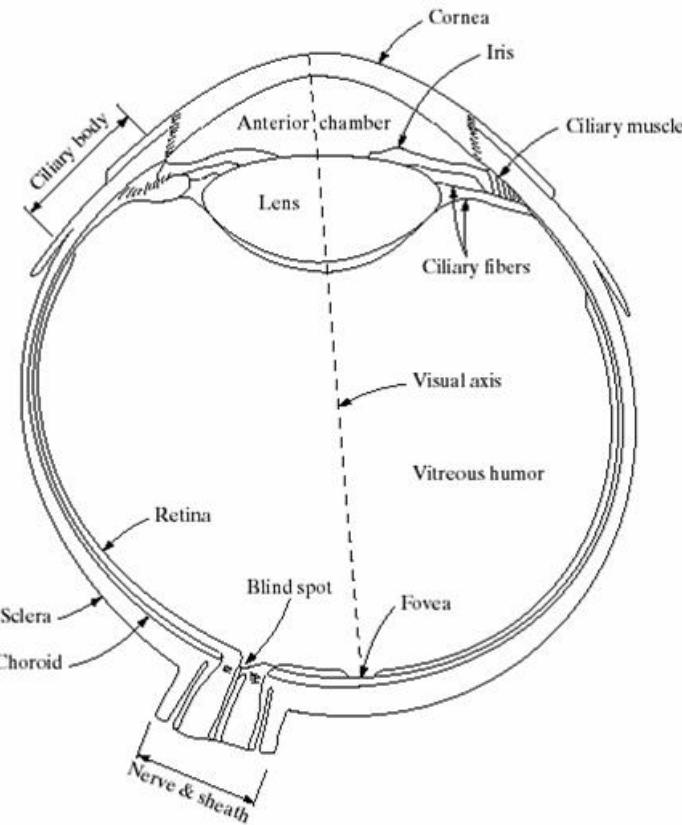
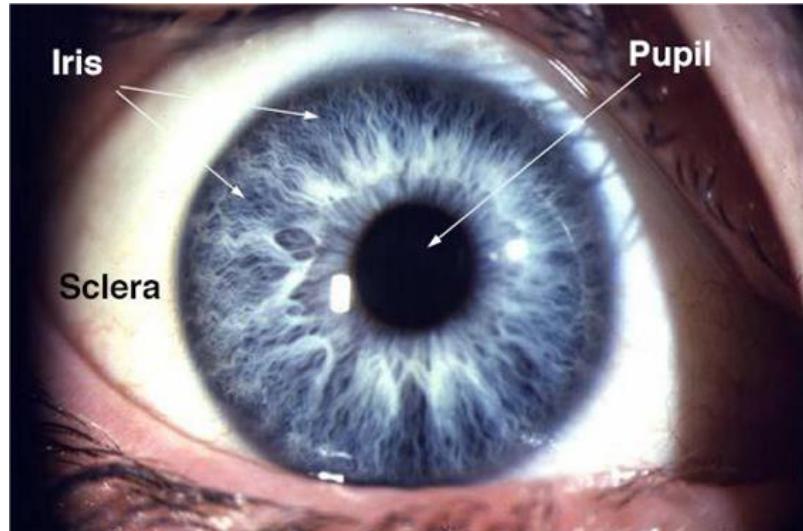


FIGURE 2.1
Simplified diagram of a cross section of the human eye.

Structure of the Human Eye: Cornea and Sclera

- Cornea
 - Tough, transparent tissue that covers the anterior surface of the eye
- Sclera
 - Opaque membrane that encloses the remainder of the optical globe

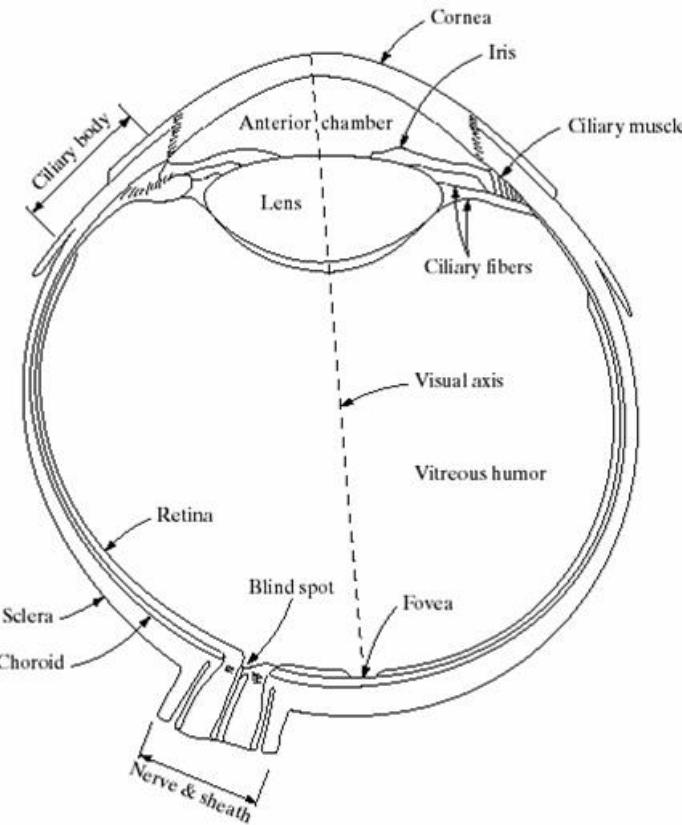


FIGURE 2.1
Simplified diagram of a cross section of the human eye.

Structure of the Human Eye: Choroid

- Choroid
 - Lies below the sclera
 - Contains network of blood vessels that serve as the major source of nutrition to the eye.
 - Choroid coat is heavily pigmented and hence helps to reduce the amount of extraneous light entering the eye and the backscatter within the optical globe

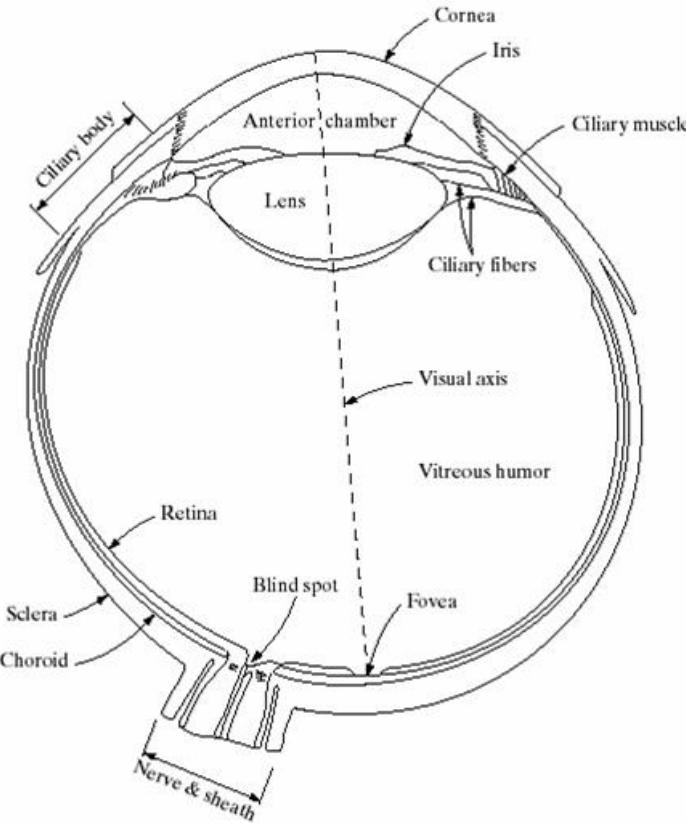


FIGURE 2.1
Simplified
diagram of a cross
section of the
human eye.

Lens and Retina

- Lens
 - Both infrared and ultraviolet light are absorbed appreciably by proteins within the lens structure and, in excessive amounts, can cause damage to the eye
- Retina
 - Innermost membrane of the eye which lines the inside of the wall's entire posterior portion. When the eye is properly focused, light from an object outside the eye is imaged on the retina.

Receptors

- Two classes of light receptors on retina: cones and rods
- Cones
 - 6-7 million cones lie in central portion of the retina, called the fovea.
 - Highly sensitive to color and bright light.
 - Resolve fine detail since each is connected to its own nerve end.
 - Cone vision is called photopic or bright-light vision.
- Rods
 - 75-150 million rods distributed over the retina surface.
 - Reduced amount of detail discernable since several rods are connected to a single nerve end.
 - Serves to give a general, overall picture of the field of view.
 - Sensitive to low levels of illumination.
 - Rod vision is called scotopic or dim-light vision.

Distribution of Cones and Rods

- Blind spot: no receptors in region of emergence of optic nerve.
- Distribution of receptors is radially symmetric about the fovea.
- Cones are most dense in the center of the retina (e.g., fovea)
- Rods increase in density from the center out to 20° and then decrease

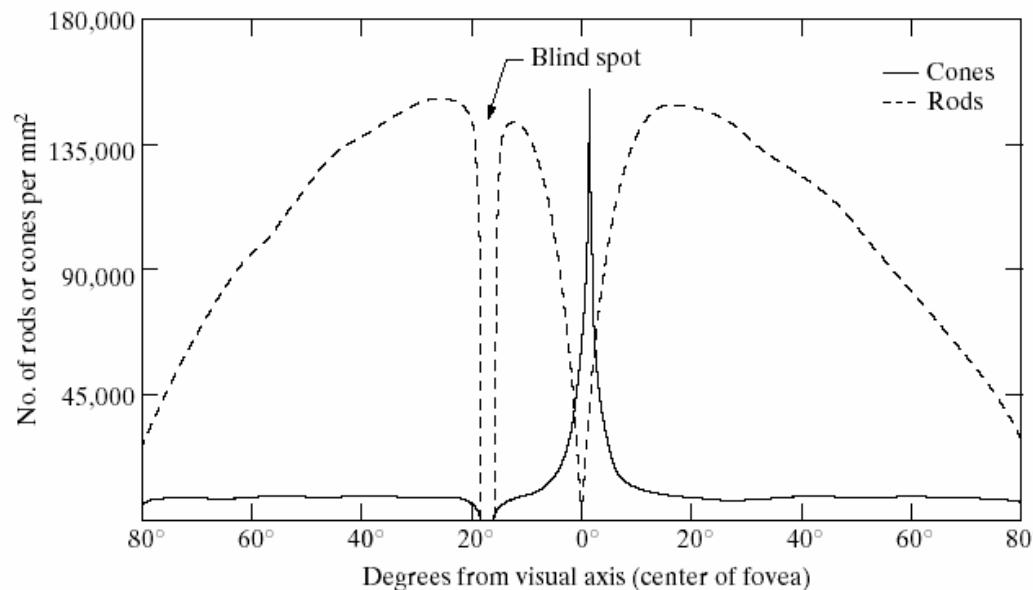


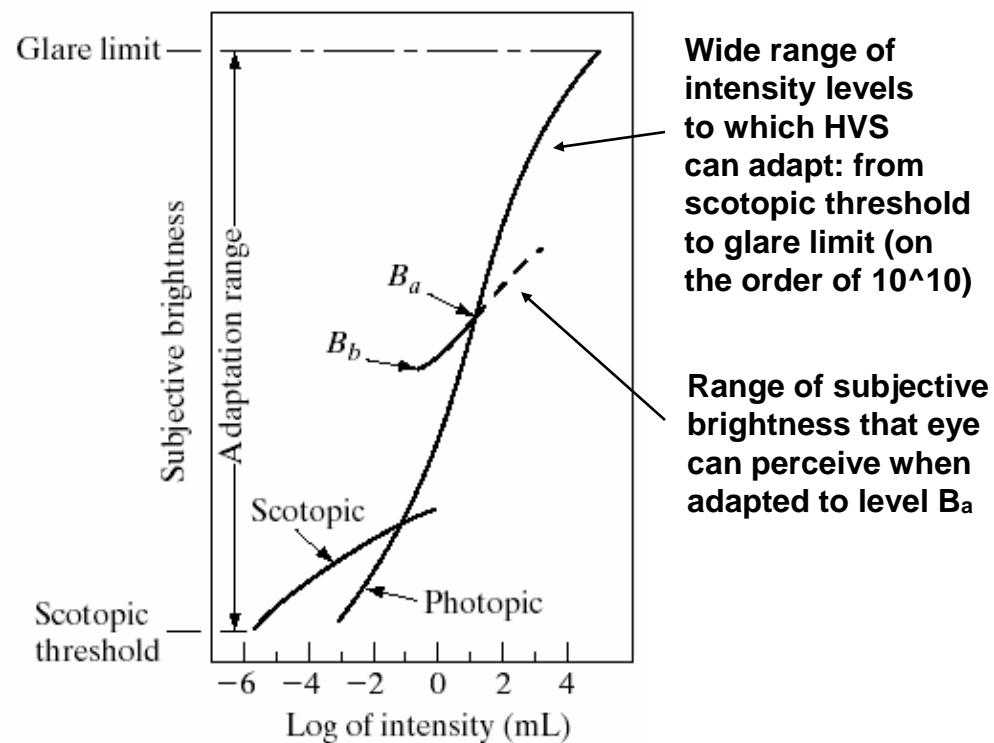
FIGURE 2.2
Distribution of rods and cones in the retina.

Brightness Adaptation (1)

- The eye's ability to discriminate between intensities is important.
- Experimental evidence suggests that subjective brightness (perceived) is a logarithmic function of light incident on eye.
Notice approximately linear response in log-scale below.

FIGURE 2.4

Range of subjective brightness sensations showing a particular adaptation level.



Brightness Adaptation (2)

- Essential point: the HVS cannot operate over such a large range *simultaneously*.
- It accomplishes this large variation by changes in its overall sensitivity: *brightness adaptation*.
- The total range of distinct intensity levels it can discriminate simultaneously is rather small when compared with the total adaptation range.
- For any given set of conditions, the current sensitivity level of the HVS is called the *brightness adaptation level* (B_a in figure).

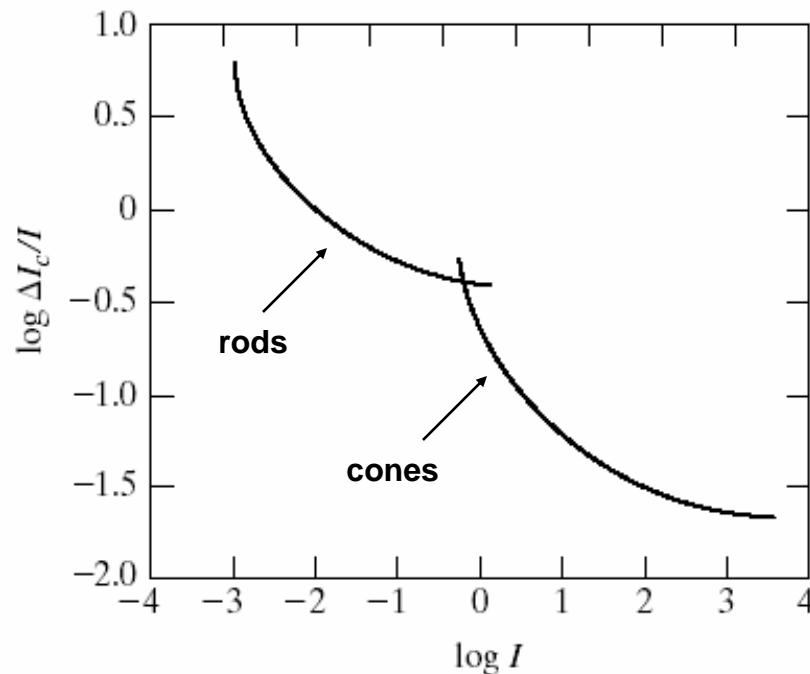
Brightness Discrimination (1)

- The ability of the eye to discriminate between intensity *changes* at any adaptation level is of considerable interest.
- Let I be the intensity of a large uniform area that covers the entire field of view.
- Let ΔI be the change in object brightness required to just distinguish the object from the background.
- Good brightness discrimination: $\Delta I / I$ is small.
- Bad brightness discrimination: $\Delta I / I$ is large.
- $\Delta I / I$ is called Weber's ratio.

Brightness Discrimination (2)

- Brightness discrimination is poor at low levels of illumination, where vision is carried out by rods. Notice Weber's ratio is large.
- Brightness discrimination improves at high levels of illumination, where vision is carried out by cones. Notice Weber's ratio is small.

FIGURE 2.6
Typical Weber
ratio as a function
of intensity.



Choice of Grayscales (1)

- Let I take on 256 different intensities:
 - $0 \leq I_j \leq 1$ for $j = 0, 1, \dots, 255$.
- Which levels we use?
 - Use eye characteristics: sensitive to *ratios* of intensity levels rather than to absolute values (Weber's law: $\Delta B/B = \text{constant}$)
 - For example, we perceive intensities .10 and .11 as differing just as much as intensities .50 and .55.

Choice of Grayscales (2)

- Levels should be spaced logarithmically rather than linearly to achieve equal steps in brightness:

$I_0, I_1 = rI_0, I_2 = rI_1 = r^2I_0, I_3 = rI_2 = r^3I_0, \dots, I_{255} = r^{255}I_0 = 1$,
where I_0 is the lowest attainable intensity.

- $r = (1/I_0)^{1/255}$,
- $I_j = r^j I_0 = (1/I_0)^{j/255} I_0 = I_0^{(1-j/255)} = I_0^{(255-j)/255}$
- In general, for $n+1$ intensities:

$$r = (1/I_0)^{1/n},$$

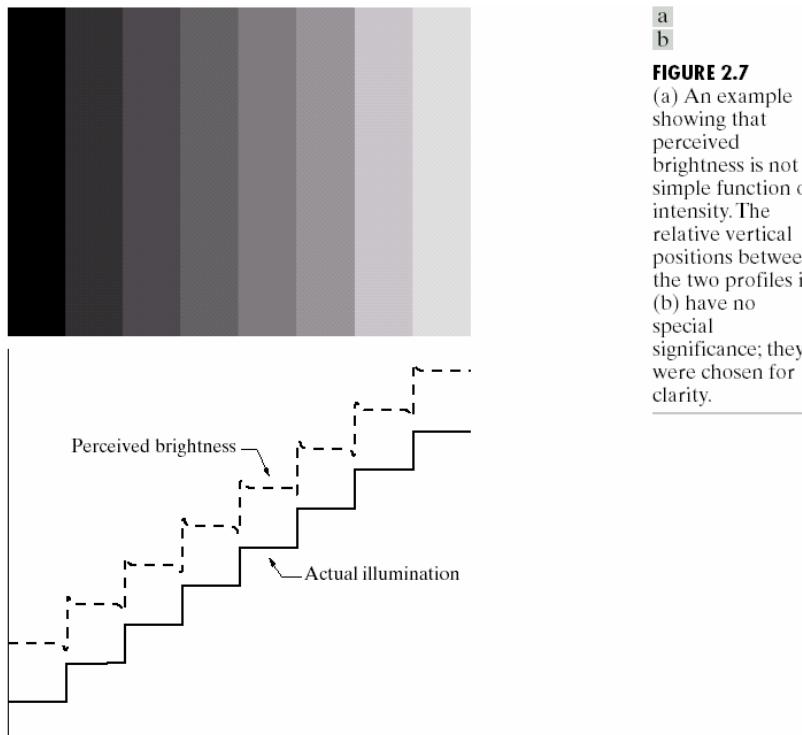
$$I_j = I_0^{(n-j)/n} \quad \text{for } 0 \leq j \leq n$$

Choice of Grayscales (3)

- Example: let $n=3$ and $I_0=1/8$:
 - $r = 2$
 - $I_0 = (1/8)^{(3/3)}$
 - $I_1 = (1/8)^{(2/3)} = \frac{1}{4}$
 - $I_2 = (1/8)^{(1/3)} = \frac{1}{2}$
 - $I_3 = (1/8)^{(0/3)} = 1$
 - For CRTs, $1/200 < I_0 < 1/40$.
 - $I_0 \neq 0$ because of light reflection from the phosphor within the CRT.
 - Linear grayscale is close to logarithmic for large number of graylevels (256).

Perceived Brightness

- Perceived brightness is not a simple function of intensity.
- The HVS tends to over/undershoot around intensity discontinuities.
- The scalloped brightness bands shown below are called *Mach bands*, after Ernst Mach who described this phenomenon in 1865.

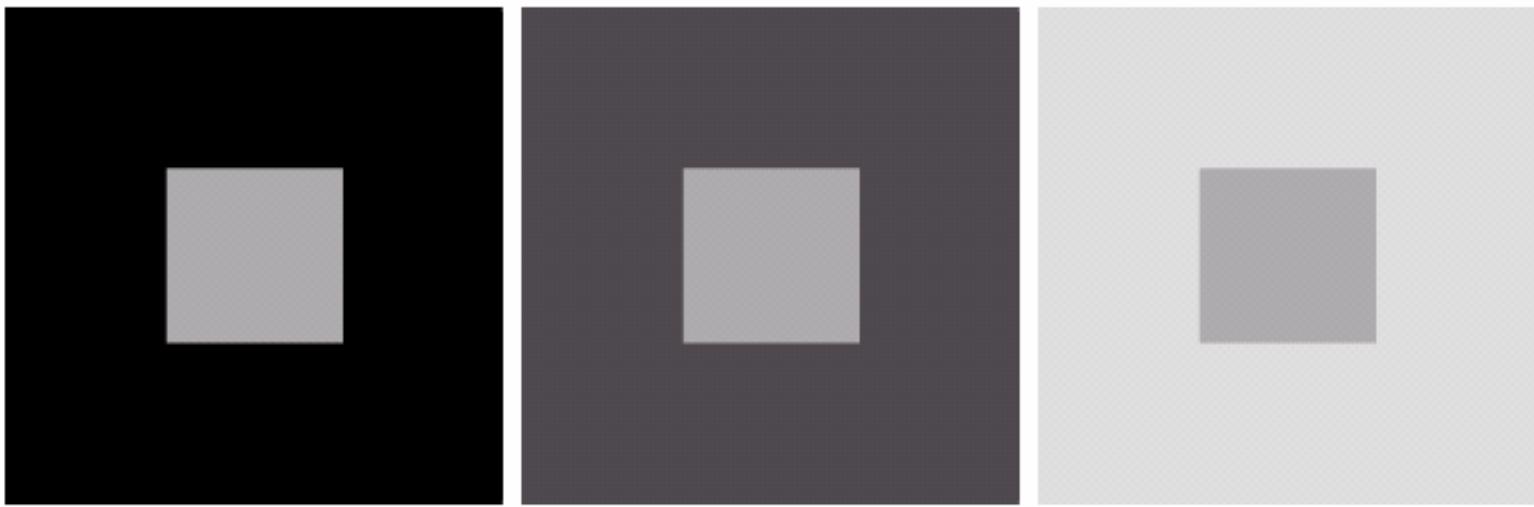


a
b

FIGURE 2.7
(a) An example showing that perceived brightness is not a simple function of intensity. The relative vertical positions between the two profiles in (b) have no special significance; they were chosen for clarity.

Simultaneous Contrast (1)

- A region's perceived brightness does not depend simply on its intensity. It is also related to the surrounding background.

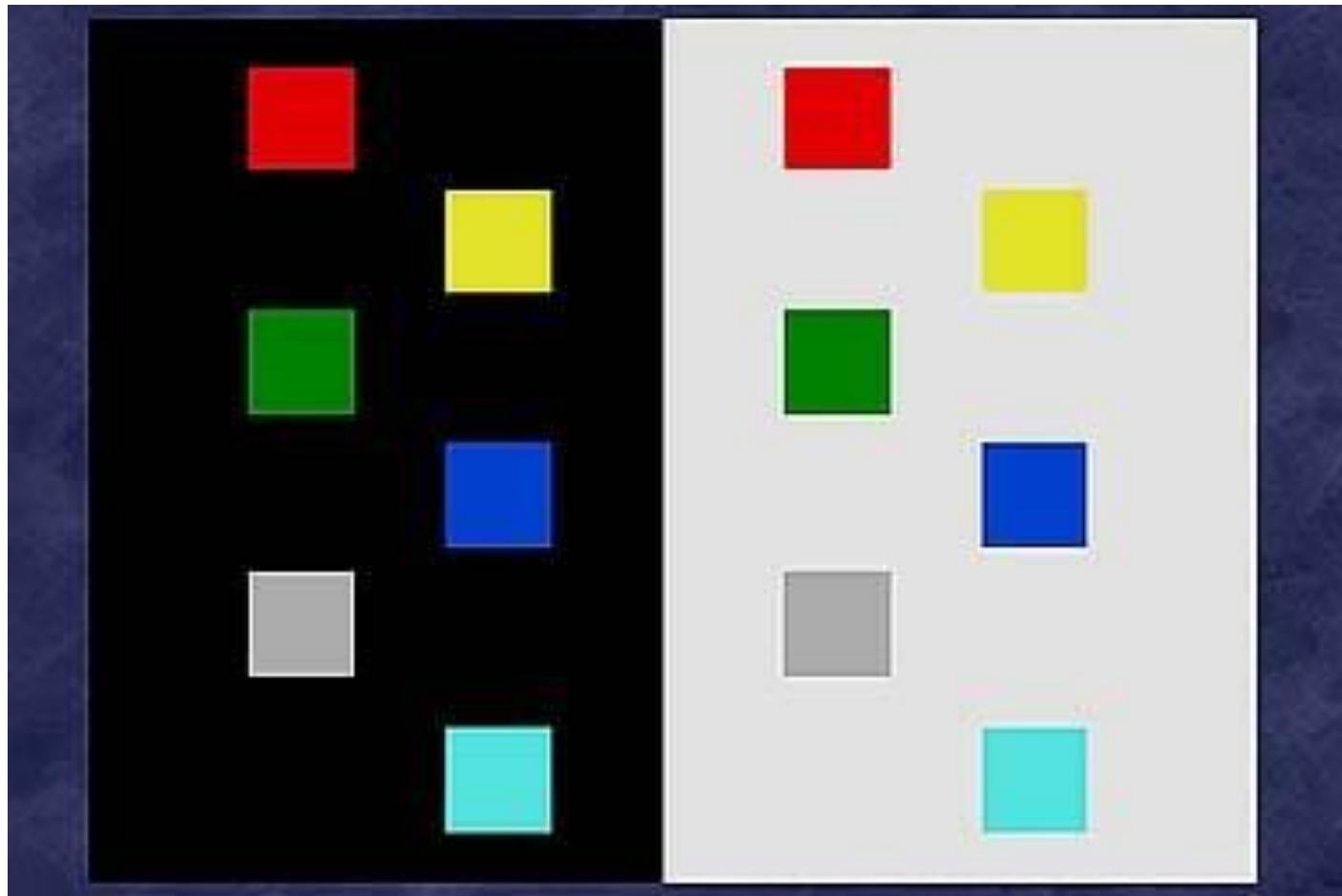


a b c

FIGURE 2.8 Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

Simultaneous Contrast (2)

- An example with colored squares.



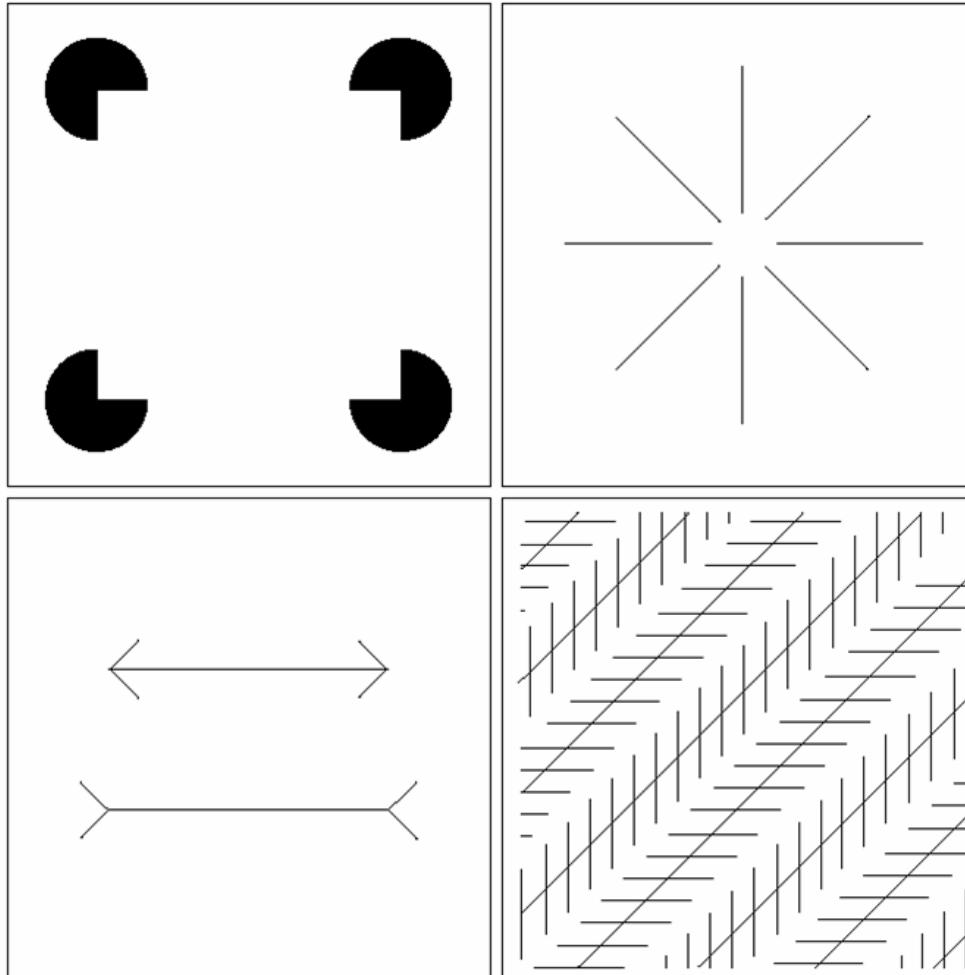
Projectors

- Why are projection screens white?
 - Reflects all colors equally well
- Since projected light cannot be negative, how are black areas produced?
 - Exploit simultaneous contrast
 - The bright area surrounding a dimly lit point makes that point appear darker

Visual Illusions (1)

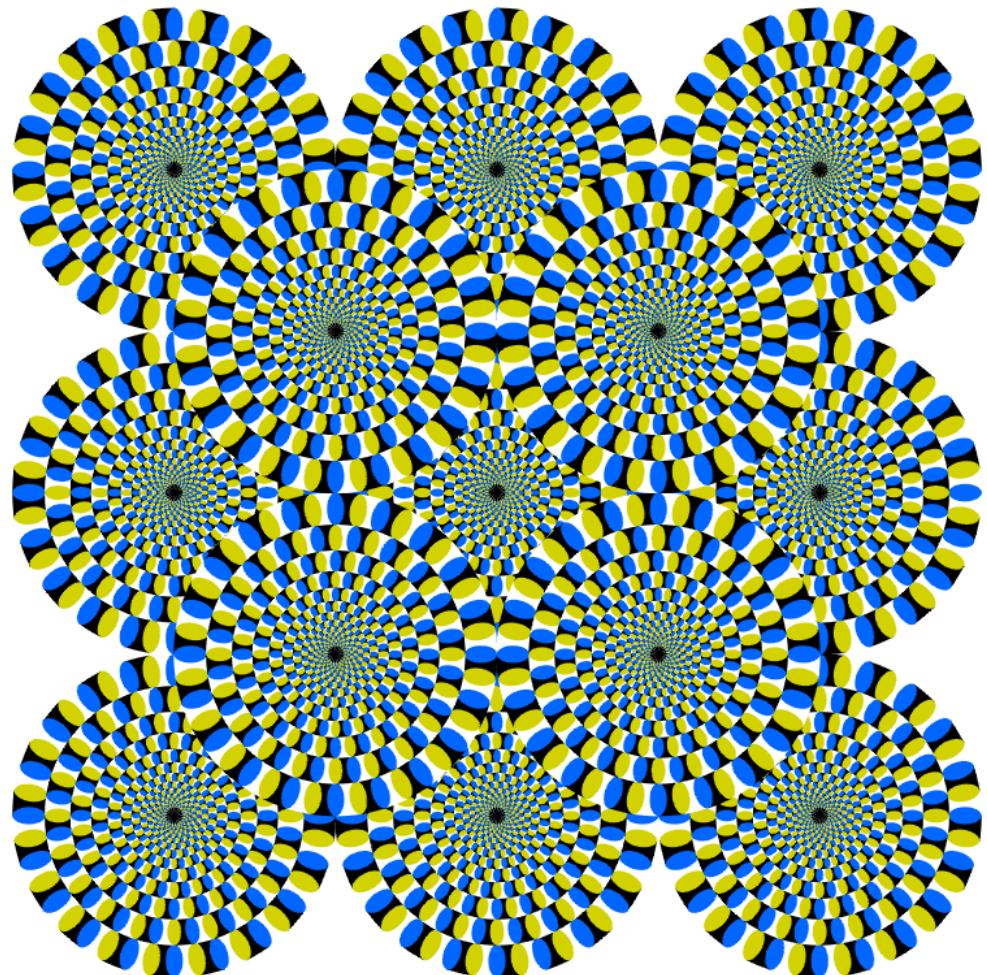
a
b
c
d

FIGURE 2.9 Some well-known optical illusions.



Visual Illusions (2)

- Rotating snake illusion
- Rotation occurs in relation to eye movement
- Effect vanishes on steady fixation
- Illusion does not depend on color
- Rotation direction depends on the polarity of the luminance steps
- Asymmetric luminance steps are required to trigger motion detectors



Digital Image Fundamentals

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

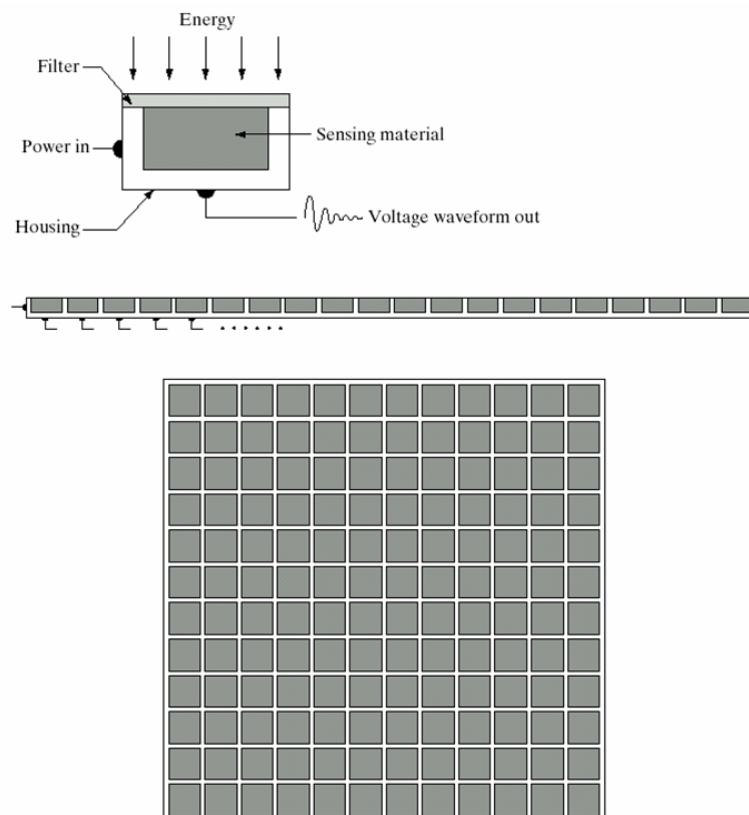
- In this lecture we discuss:
 - Image acquisition
 - Sampling and quantization
 - Spatial and graylevel resolution

Sensor Arrangements

- Three principal sensor arrangements:
 - Single, line, and array

a
b
c

FIGURE 2.12
(a) Single imaging
sensor.
(b) Line sensor.
(c) Array sensor.



Single Sensor

- Photodiode: constructed of silicon materials whose output voltage waveform is proportional to light.
- To generate a 2D image using a single sensor, there must be relative displacements in the horizontal and vertical directions between the sensor and the area to be imaged.
- Microdensitometers: mechanical digitizers that use a flat bed with the single sensor moving in two linear directions.

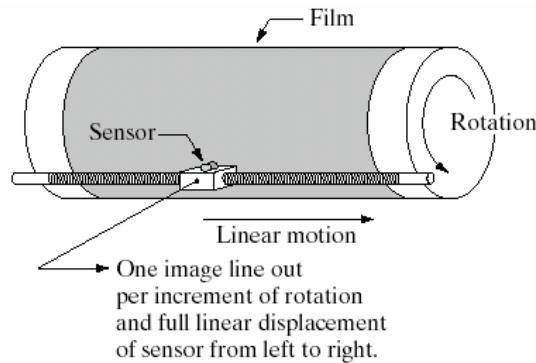
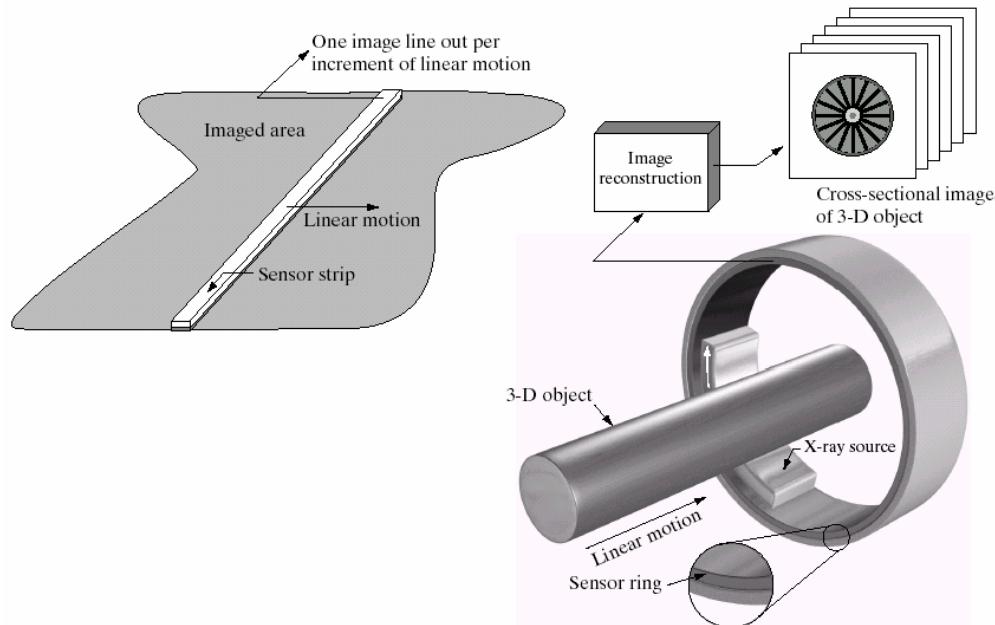


FIGURE 2.13 Combining a single sensor with motion to generate a 2-D image.

Sensor Strips

- In-line arrangement of sensors in the form of a sensor strip.
- The strip provides imaging elements in one direction.
- Motion perpendicular to strip images in the other direction.
- Used in flat bed scanners, with 4000 or more in-line sensors.



a b

FIGURE 2.14 (a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip.

Sensor Arrays

- Individual sensors are arranged in the form of a 2D array.
- Used in digital cameras and camcorders.
- Entire image formed at once; no motion necessary.

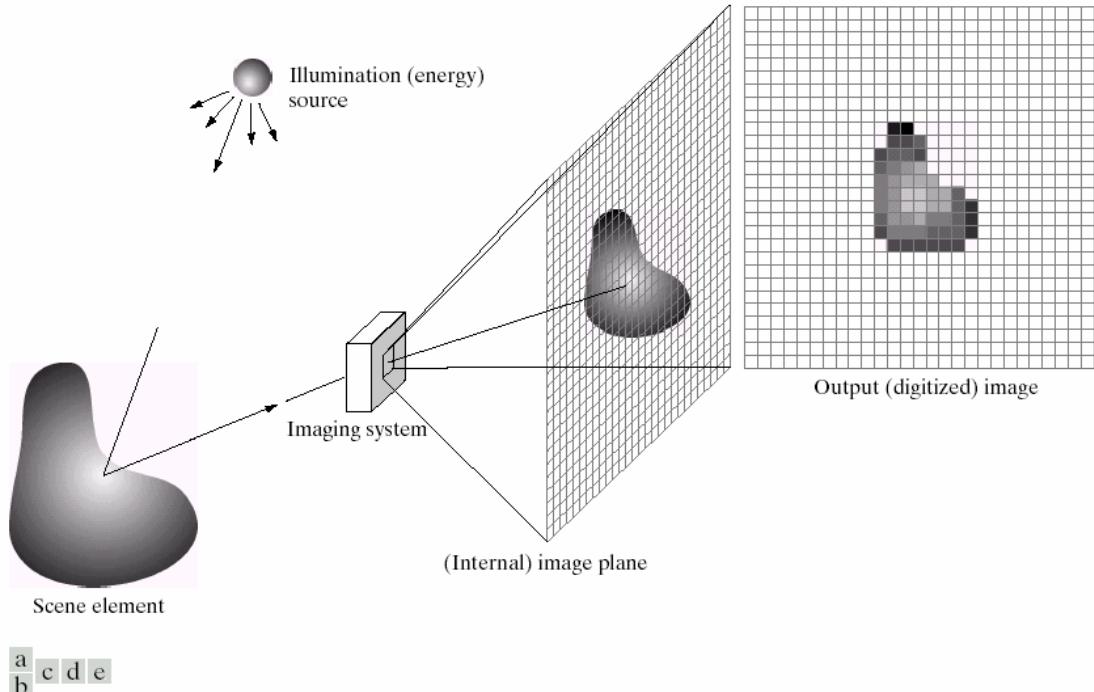


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Signals

- A signal is a function that conveys information
 - 1D signal: $f(x)$ waveform
 - 2D signal: $f(x,y)$ image
 - 3D signal: $f(x,y,z)$ volumetric data
 - or $f(x,y,t)$ animation (spatiotemporal volume)
 - 4D signal: $f(x,y,z,t)$ snapshots of volumetric data over time
- The dimension of the signal is equal to its number of indices.
- In this course, we focus on 2D images: $f(x,y)$
- Efficient implementation often calls for 1D row or column processing. That is, process the rows independently and then process the columns of the resulting image.

Digital Image

- Image produced as an array (the *raster*) of picture elements (*pixels* or *pel/s*) in the *frame buffer*.

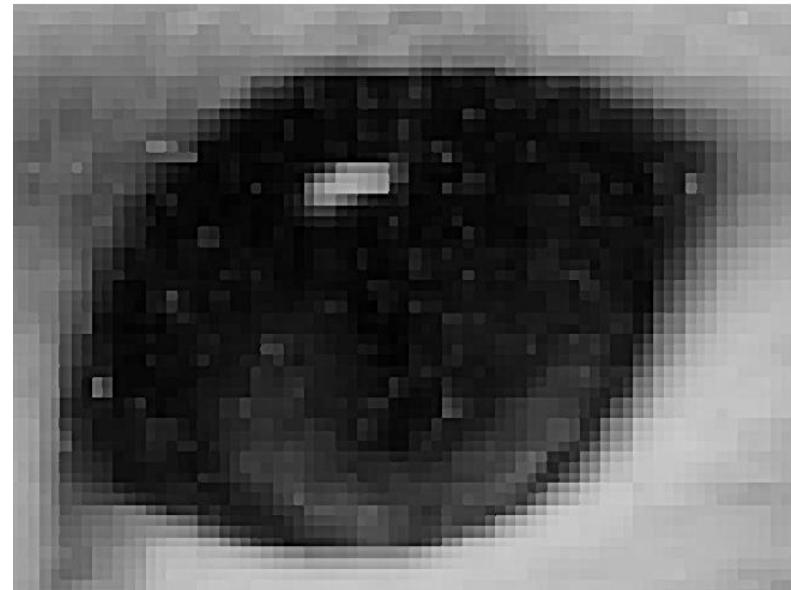
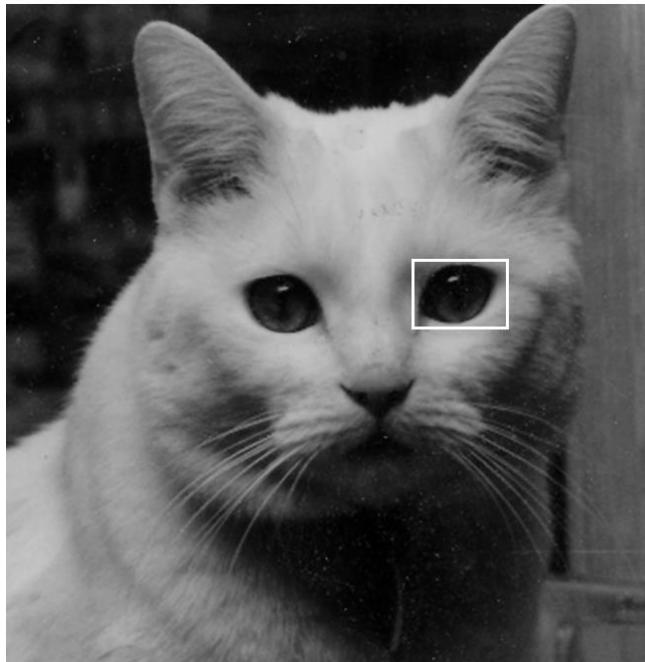


Image Classification (1)

- Images can be classified by whether they are defined over all points in the spatial domain and whether their image values have finite or infinite precision.
- If the position variables (x,y) are continuous, then the function is defined over all points in the spatial domain.
- If (x,y) is discrete, then the function can be sampled at only a finite set of points, i.e., the set of integers.
- The value that the function returns can also be classified by its precision, independently of x and y .

Image Classification (2)

- Quantization refers to the mapping of real numbers onto a finite set: a many-to-one mapping.
- Akin to casting from double precision to an integer.

Space	Image Values	Classification
continuous	continuous	analog (continuous) image
continuous	discrete	intensity quantization
discrete	continuous	spatial quantization
discrete	discrete	digital (discrete) image

Image Formation

- The values of an acquired image are always positive. There are no negative intensities: $0 < f(x,y) < \infty$
- Continuous function $f(x,y) = i(x,y) r(x,y)$, where
 - $0 < i(x,y) < \infty$ is the illumination
 - $0 < r(x,y) < 1$ is the reflectance of the object
- $r(x,y) = 0$ is total absorption and $r(x,y) = 1$ is total reflectance.
- Replace $r(x,y)$ with transmissivity term $t(x,y)$ for chest X-ray.

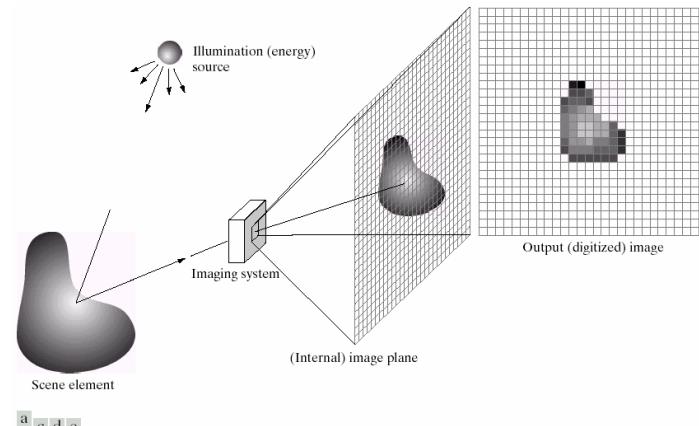


FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Typical Values of Illumination and Reflectance

- The following $i(x,y)$ illumination values are typical (in lumens/m²):
 - Illumination of sun on Earth on a clear day: 90,000
 - Illumination of sun on Earth on a cloudy day: 10,000
 - Illumination of moon on Earth on a clear night: 0.1
 - Illumination level in a commercial office: 1000
 - Illumination level of video projectors: 1000-1500
- The following $r(x,y)$ reflectance values are typical:
 - Black velvet: 0.01
 - Stainless steel: 0.65
 - Flat-white wall paint: 0.80
 - Silver-plated metal: 0.90
 - Snow: 0.93

Graylevels

- The intensity of a monochrome image at any coordinate (x,y) is called *graylevel* L , where $L_{\min} \leq L \leq L_{\max}$
- The office illumination example indicates that we may expect
 $L_{\min} \approx .01 * 1000 = 10$ (virtual black)
 $L_{\max} \approx 1 * 1000 = 1000$ (white)
- Interval $[L_{\min}, L_{\max}]$ is called the *grayscale*.
- In practice, the interval is shifted to the $[0, 255]$ range so that intensity can be represented in one byte (unsigned char).
- 0 is black, 255 is white, and all intermediate values are different shades of gray varying from black to white.

Generating a Digital Image

- Sample and quantize continuous input image.

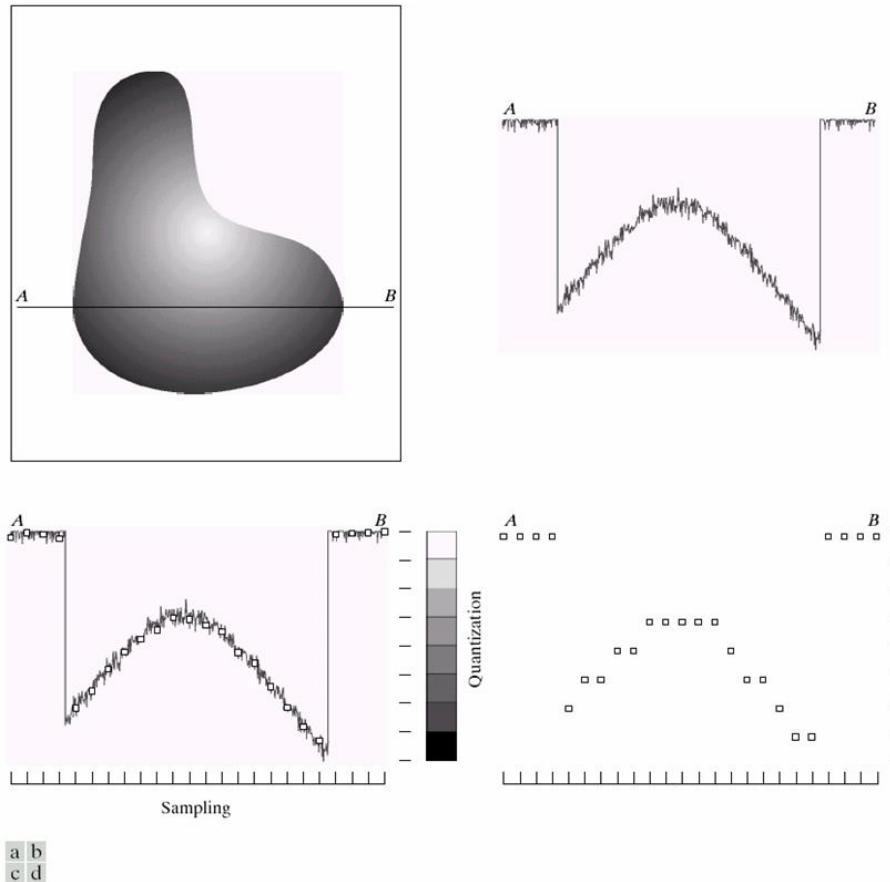


FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

Image Sampling and Quantization

- Sampling: digitize (discretize) spatial coordinate (x,y)
- Quantization: digitize intensity level L

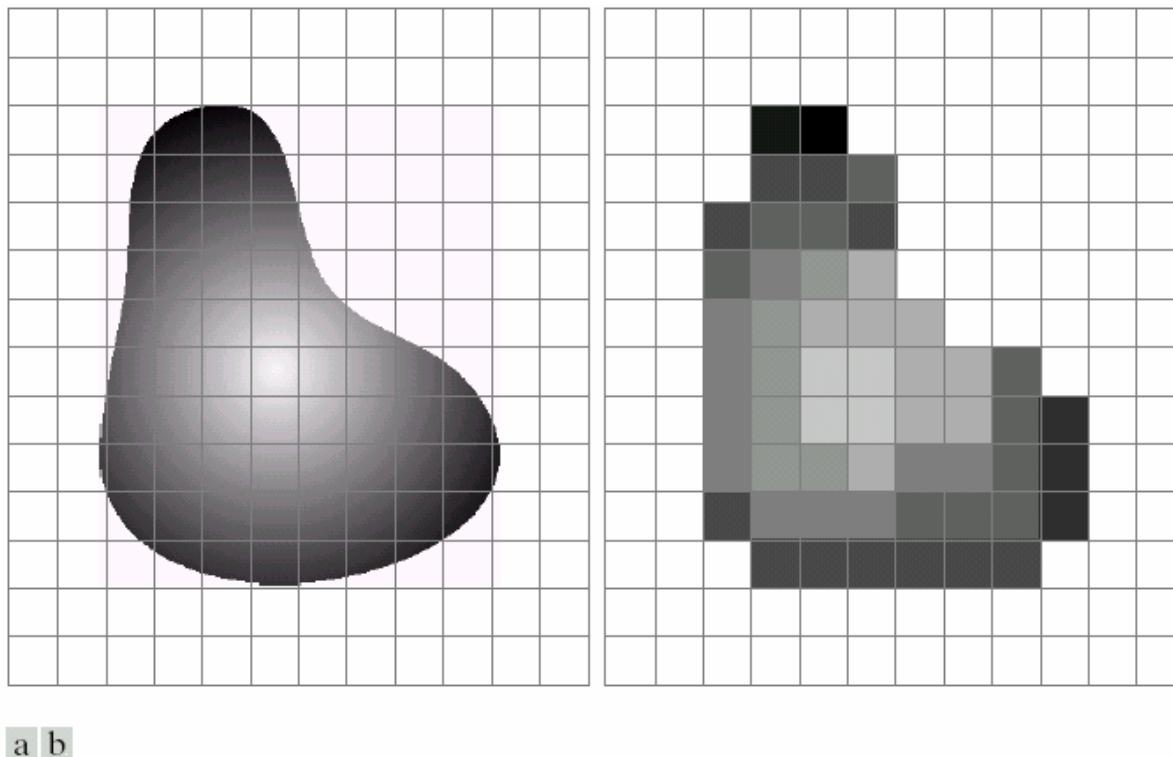


FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Effects of Varying Sampling Rate (1)

- Subsampling was performed by dropping rows and columns.
- The number of gray levels was kept constant at 256.



FIGURE 2.19 A 1024×1024 , 8-bit image subsampled down to size 32×32 pixels. The number of allowable gray levels was kept at 256.

Effects of Varying Sampling Rate (2)

- Size differences make it difficult to see effects of subsampling.

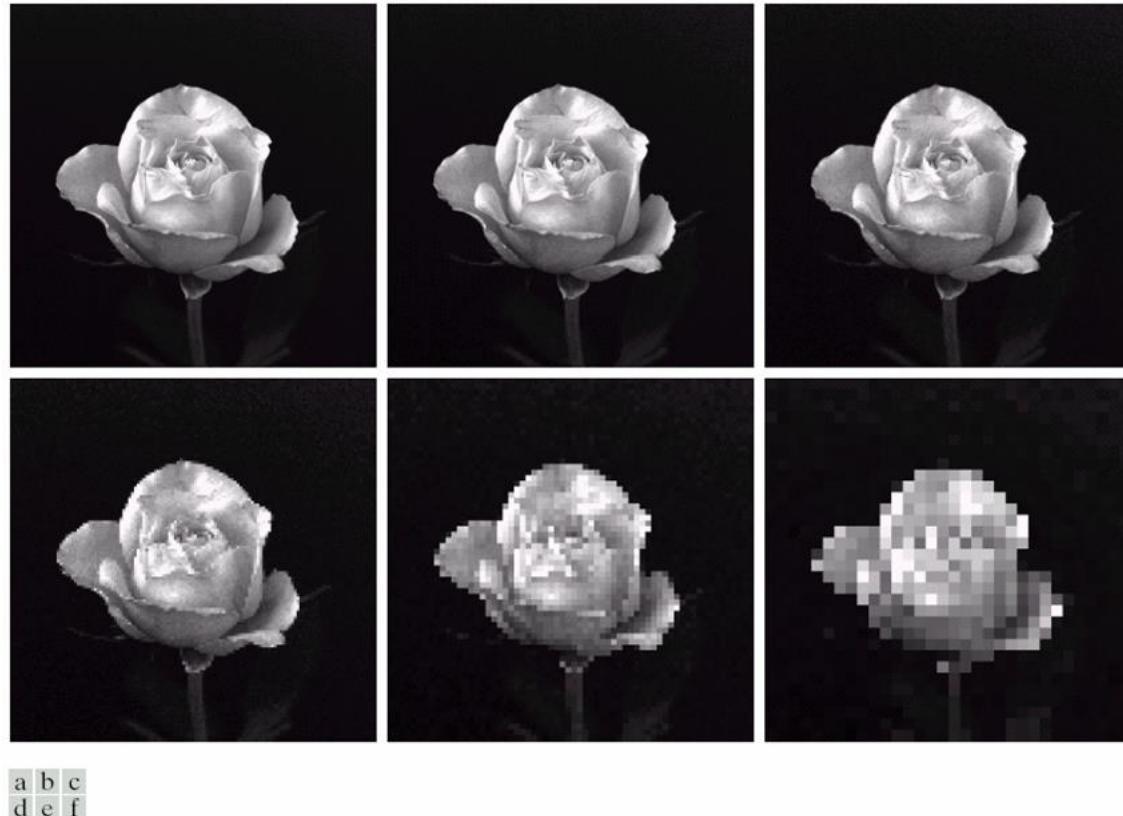


FIGURE 2.20 (a) 1024×1024 , 8-bit image. (b) 512×512 image resampled into 1024×1024 pixels by row and column duplication. (c) through (f) 256×256 , 128×128 , 64×64 , and 32×32 images resampled into 1024×1024 pixels.

Spatial Resolution

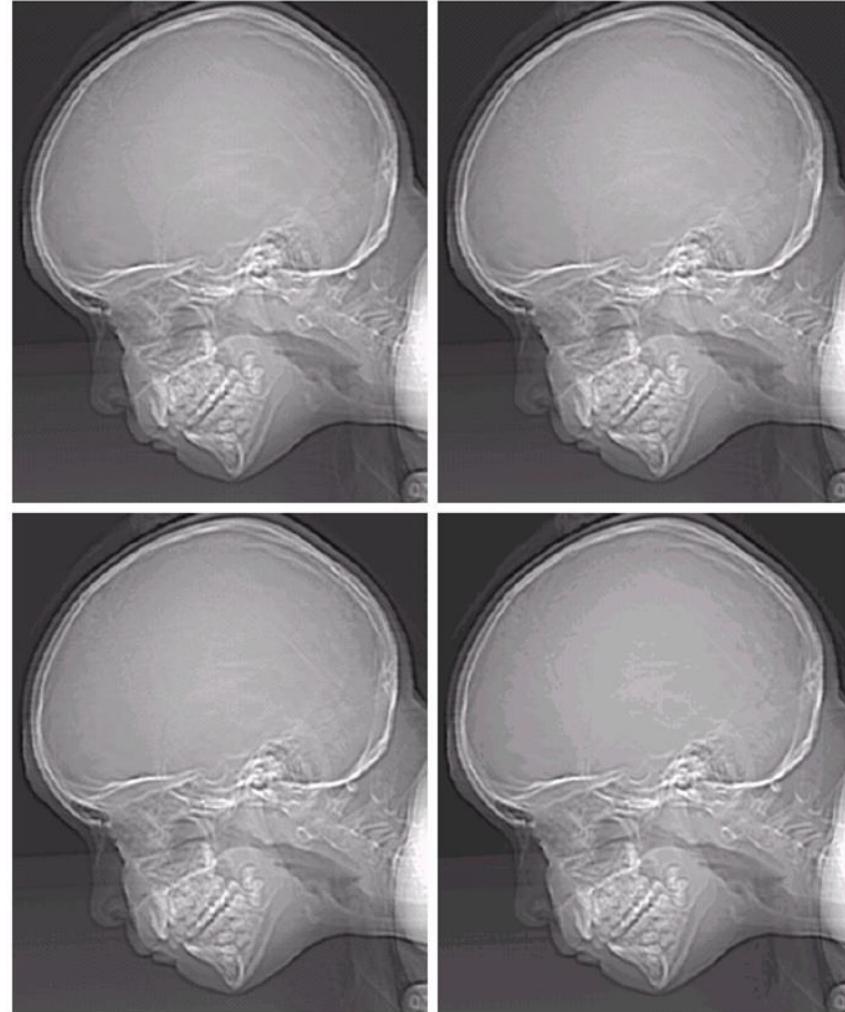
- Defined as the smallest discernable detail in an image.
- Widely used definition: smallest number of discernable line pairs per unit distance (100 line pairs/millimeter).
- A line pair consists of one line and its adjacent space.
- When an actual measure of physical resolution is not necessary, it is common to refer to an $M \times N$ image as having spatial resolution of $M \times N$ pixels.

Graylevel Resolution

- Defined as the smallest discernable change in graylevel.
- Highly subjective process.
- The number of graylevels is usually a power of two:
 - k bits of resolution yields 2^k graylevels.
 - When $k=8$, there are 256 graylevels ← most typical case
- Black-and-white television uses $k=6$, or 64 graylevels.

Effects of Varying Graylevels (1)

- Number of graylevels reduced by dropping bits from $k=8$ to $k=1$
- Spatial resolution remains constant.



a b
c d

FIGURE 2.21
(a) 452×374 , 256-level image.
(b)–(d) Image displayed in 128, 64, and 32 gray levels, while keeping the spatial resolution constant.

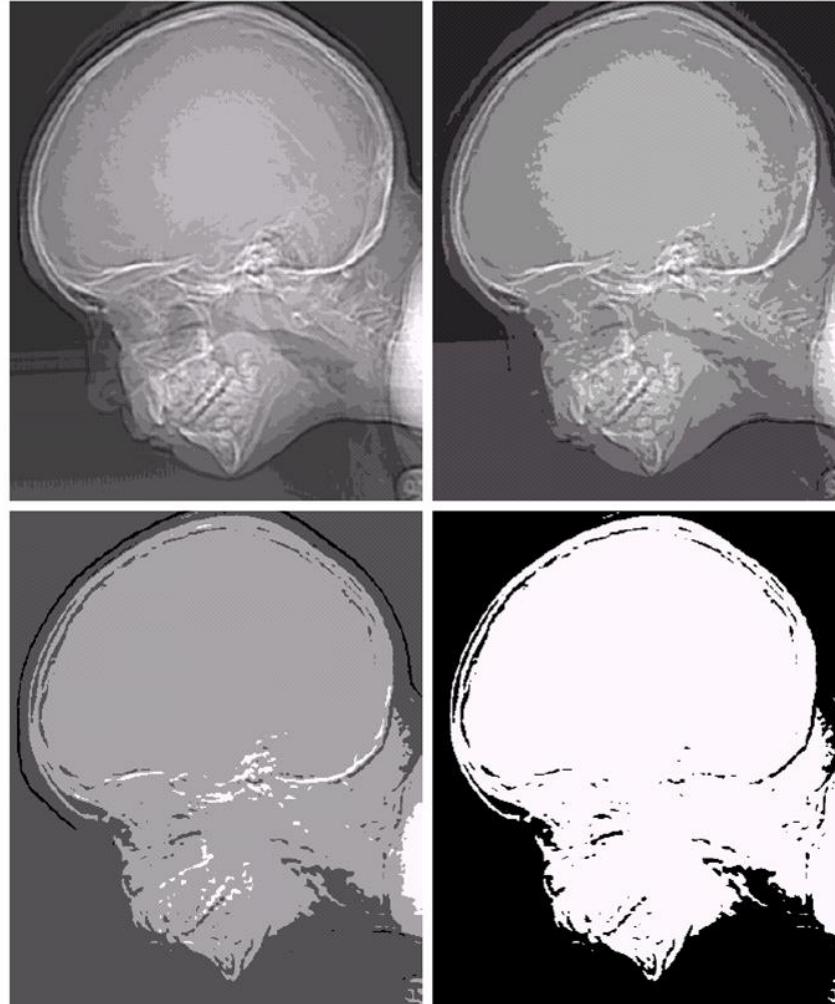
Effects of Varying Graylevels (2)

- Notice false contouring in coarsely quantized images.

- Appear as fine ridgelike structures in areas of smooth gray levels.

e f
g h

FIGURE 2.21
(Continued)
(e)–(h) Image displayed in 16, 8, 4, and 2 gray levels. (Original courtesy of Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)



Storage Requirements

- Consider an $N \times N$ image having k bits per pixel.
- Color (RGB) images require three times the storage (assuming no compression).

TABLE 2.1

Number of storage bits for various values of N and k .

N/k	1 ($L = 2$)	2 ($L = 4$)	3 ($L = 8$)	4 ($L = 16$)	5 ($L = 32$)	6 ($L = 64$)	7 ($L = 128$)	8 ($L = 256$)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

Large Space of Images

- Any image can be downsampled and represented in a few bits/pixel for use on small coarse displays.
- How many unique images can be displayed on an $N \times N$ k -bit display?
 - 2^k possible values at each pixel
 - N^2 pixels
 - Total: $(2^k)^{N^2}$
- This total is huge even for $k=1$ and $N=8$:
 $18,446,744,073,709,551,616 \quad \leftarrow 2^{64}$
- It would take 19,498,080,578 years to view this if it were laid out on video at 30 frames/sec.

Point Operations

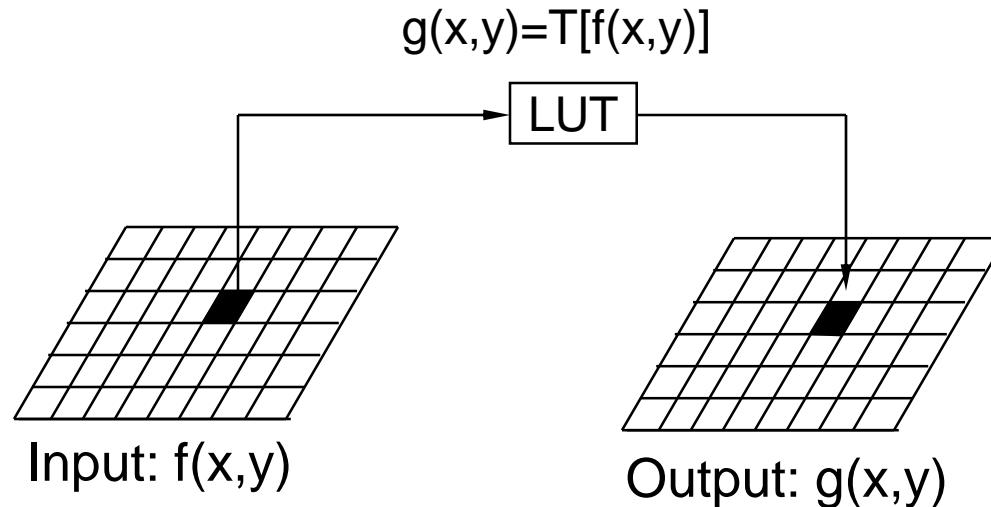
Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we describe point operations commonly used in image processing:
 - Thresholding
 - Quantization (aka posterization)
 - Gamma correction
 - Contrast/brightness manipulation
 - Histogram equalization/matching

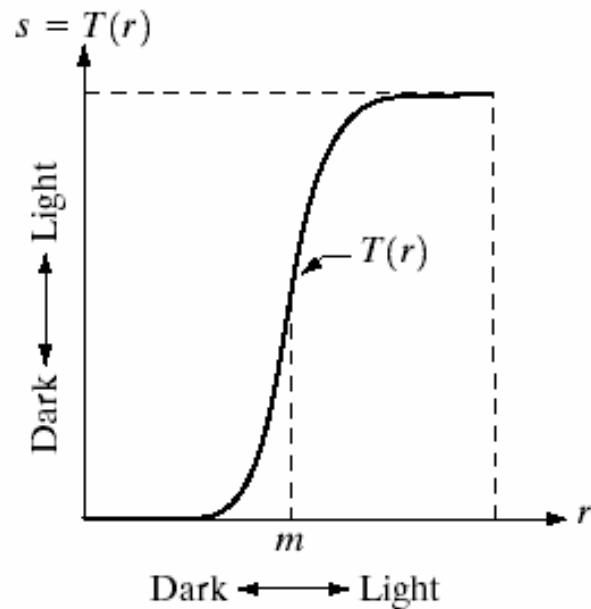
Point Operations

- Output pixels are a function of only one input point:
$$g(x,y) = T[f(x,y)]$$
- Transformation T is implemented with a lookup table:
 - An input value indexes into a table and the data stored there is copied to the corresponding output position.
 - The LUT for an 8-bit image has 256 entries.

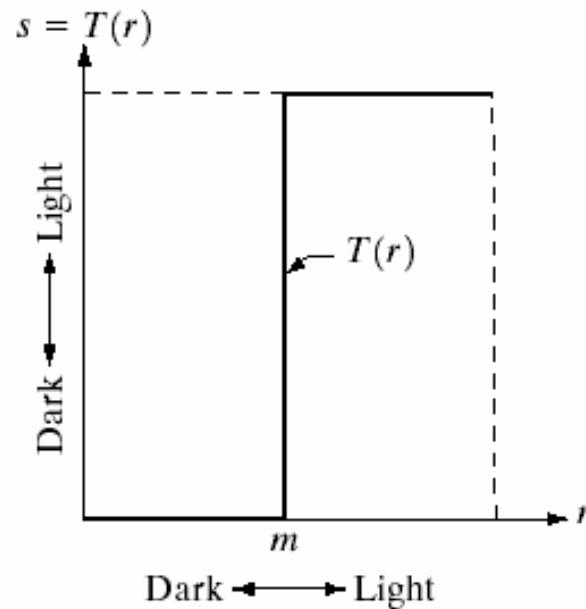


Graylevel Transformation T

Contrast enhancement:
Darkens levels below m
Brightens levels above m



Thresholding:
Replace values below m to black (0)
Replace values above m to white (255)

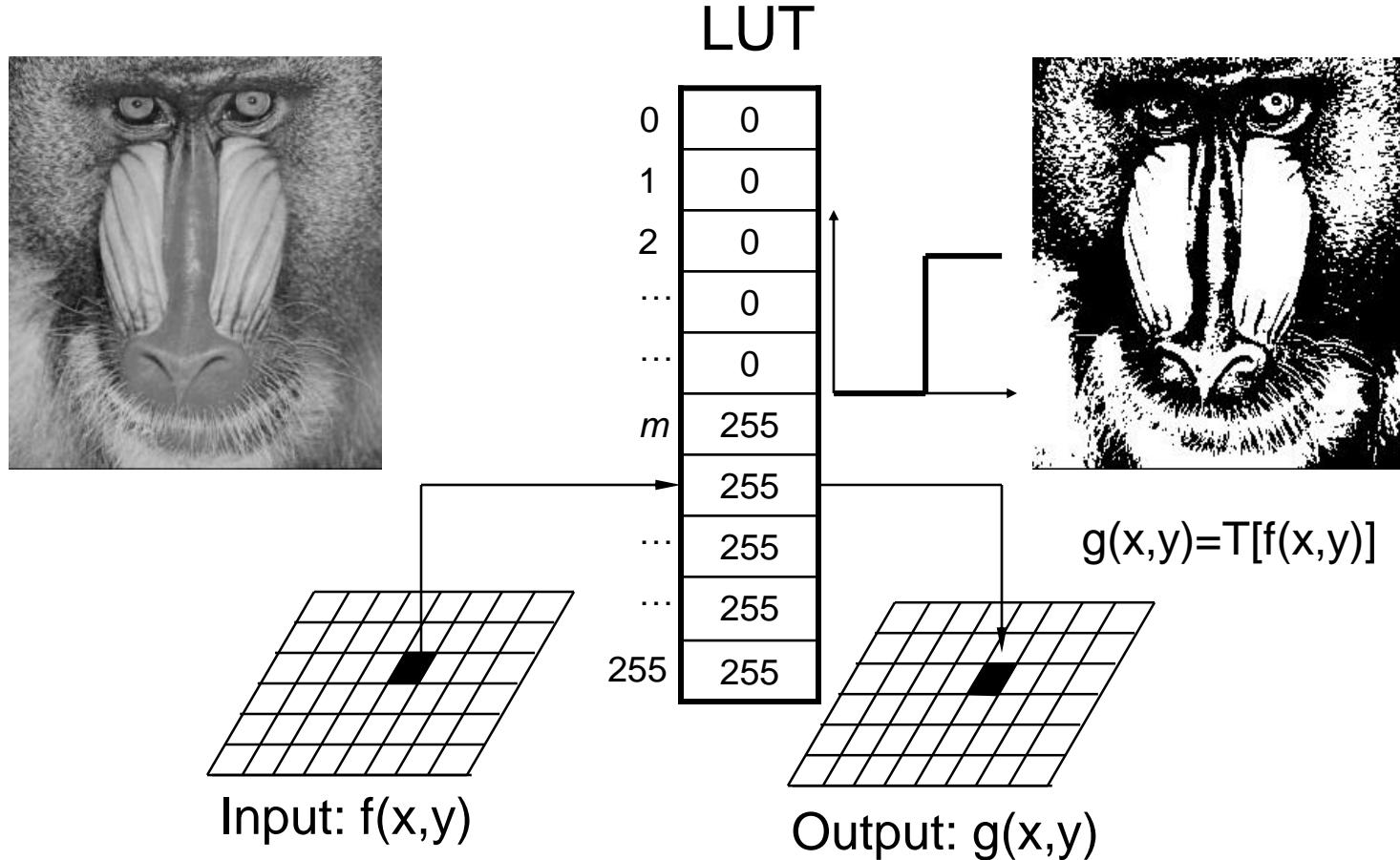


a b

FIGURE 3.2 Gray-level transformation functions for contrast enhancement.

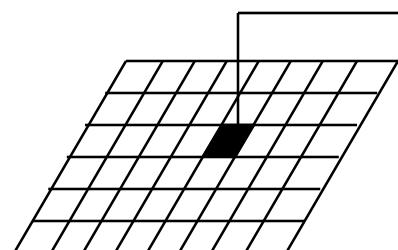
Lookup Table: Threshold

- Init LUT with samples taken from thresholding function T



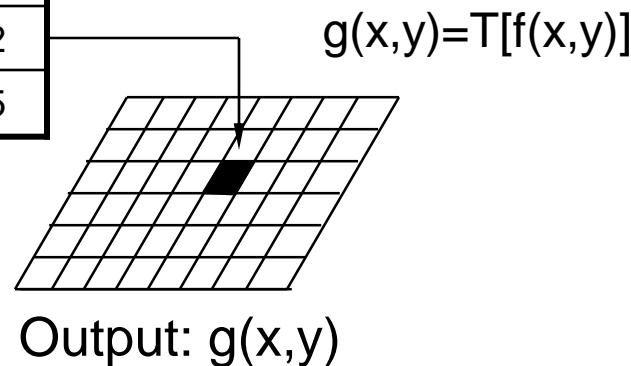
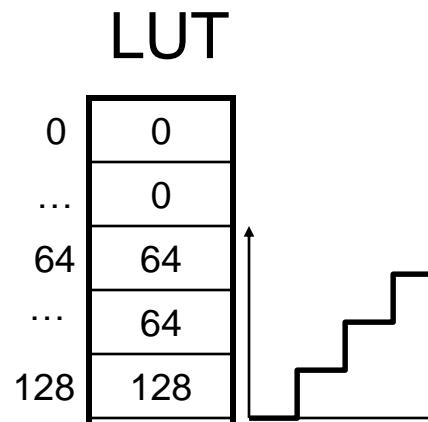
Lookup Table: Quantization

- Init LUT with samples taken from quantization function T



Input: $f(x,y)$

LUT
0
...
64
...
128
...
192
...
255



Output: $g(x,y)$

Threshold Program

- Straightforward implementation:

```
// iterate over all pixels
for(i=0; i<total; i++) {
    if(in[i] < thr)      out[i] = BLACK;
    else                  out[i] = WHITE;
}
```

- Better approach: exploit LUT to avoid **total** comparisons:

```
// init lookup tables
for(i=0; i<thr; i++) lut[i] = BLACK;
for(; i<MXGRAY; i++) lut[i] = WHITE;

// iterate over all pixels
for(i=0; i<total; i++) out[i] = lut[in[i]];
```

Quantization Program

- Straightforward implementation:

```
// iterate over all pixels
scale = MXGRAY / levels;
for(i=0; i<total; i++)
    out[i] = scale * (int) (in[i]/scale);
```

- Better approach: exploit LUT to avoid **total** mults/divisions:

```
// init lookup tables
scale = MXGRAY / levels;
for(i=0; i<MXGRAY; i++)
    lut[i] = scale * (int) (i/scale);

// iterate over all pixels
for(i=0; i<total; i++) out[i] = lut[in[i]];
```

Quantization Artifacts

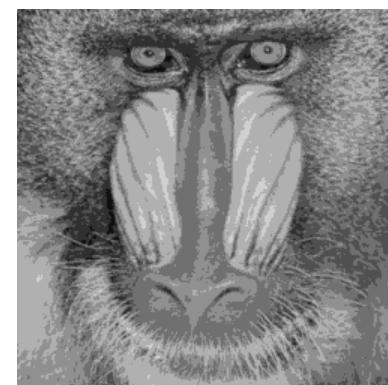
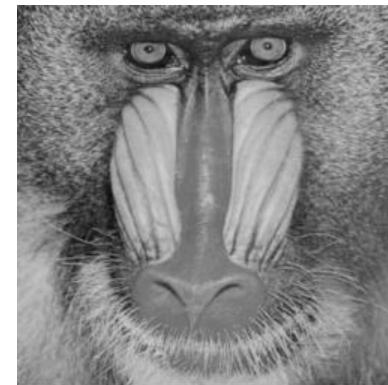
- The false contours associated with quantization are most noticeable in smooth areas.
- They are obscured in highly textured regions.



Original image



Quantized to 8 levels

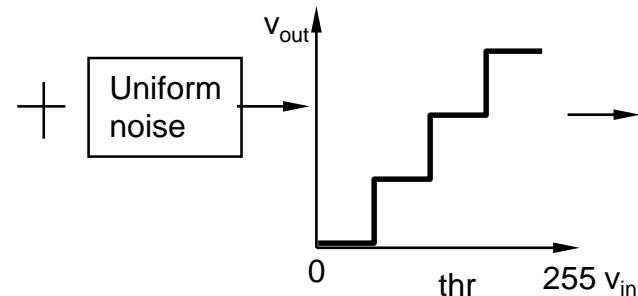


Dither Signal

- Reduce quantization error by adding uniformly distributed white noise (dither signal) to the input image prior to quantization.
- Dither hides objectional artifacts.
- To each pixel of the image, add a random number in the range $[-m, m]$, where m is MXGRAY/quantization-levels.



8 bpp (256 levels)



3 bpp (8 levels)

Comparison

1 bpp



2 bpp



3 bpp



4 bpp



Quantization



Dither/
Quantization

Enhancement

- Point operations are used to enhance an image.
- Processed image should be more suitable than the original image for a specific application.
- Suitability is application-dependent.
- A method which is quite useful for enhancing one image may not necessarily be the best approach for enhancing another image.
- Very subjective

Two Enhancement Domains

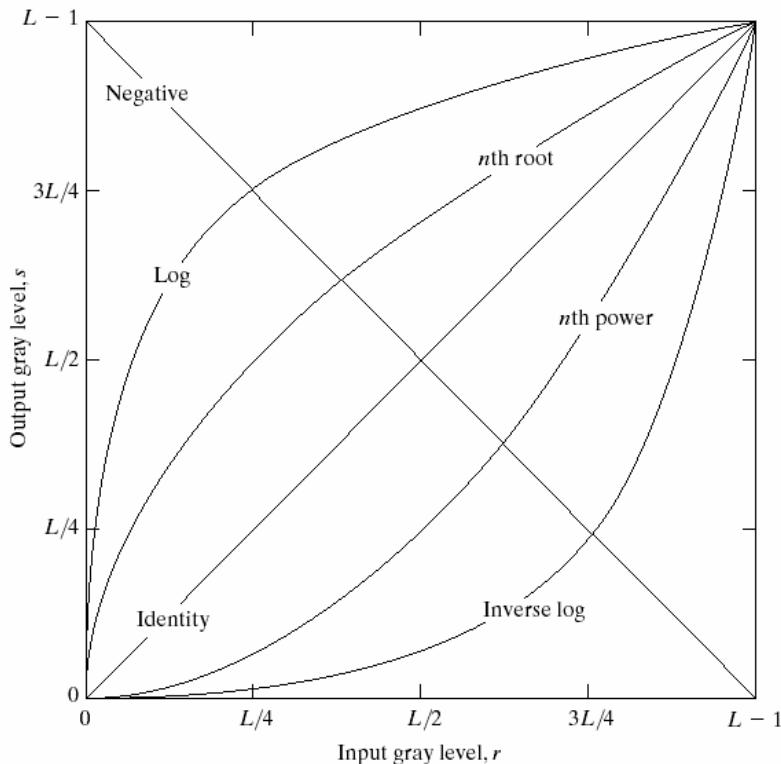
- Spatial Domain: (image plane)
 - Techniques are based on direct manipulation of pixels in an image
- Frequency Domain:
 - Techniques are based on modifying the Fourier transform of an image
- There are some enhancement techniques based on various combinations of methods from these two categories.

Enhanced Images

- For human vision
 - The visual evaluation of image quality is a highly subjective process.
 - It is hard to standardize the definition of a good image.
- For machine perception
 - The evaluation task is easier.
 - A good image is one which gives the best machine recognition results.
- A certain amount of trial and error usually is required before a particular image enhancement approach is selected.

Three Basic Graylevel Transformation Functions

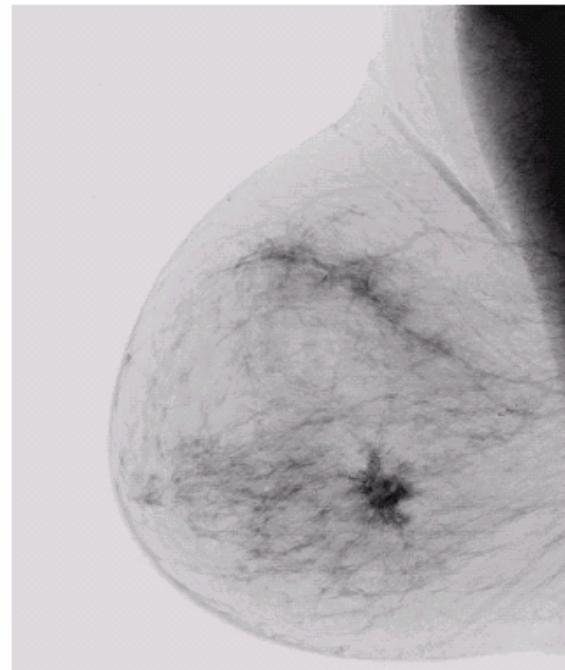
FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



- Linear function
 - Negative and identity transformations
- Logarithmic function
 - Log and inverse-log transformations
- Power-law function
 - n^{th} power and n^{th} root transformations

Image Negatives

- Negative transformation : $s = (L-1) - r$
- Reverses the intensity levels of an image.
- Suitable for enhancing white or gray detail embedded in dark regions of an image, especially when black area is large.

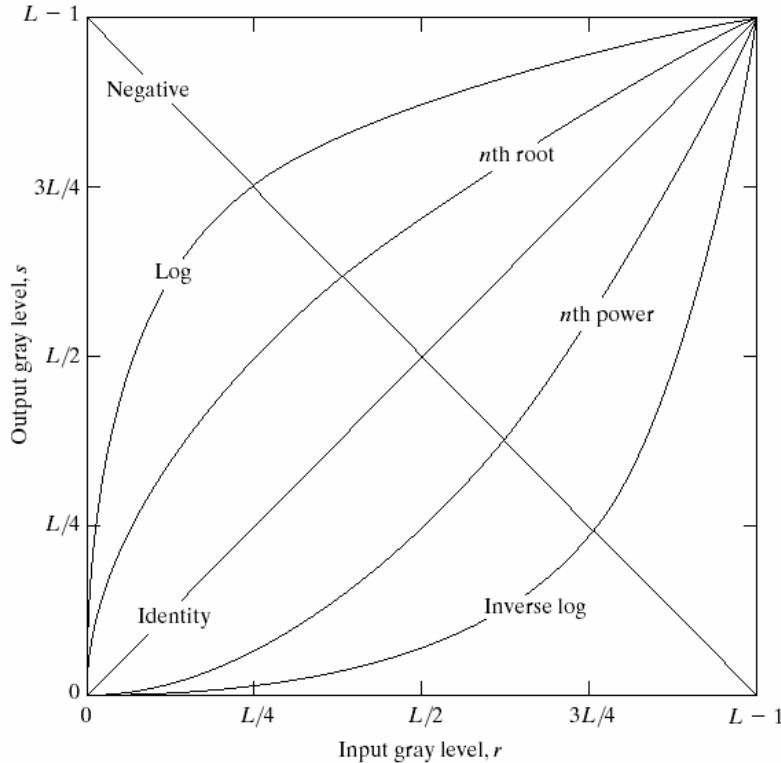


a | b

FIGURE 3.4
(a) Original digital mammogram.
(b) Negative image obtained using the negative transformation in Eq. (3.2-1).
(Courtesy of G.E. Medical Systems.)

Log Transformations

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



$$s = c \log (1+r)$$

- c is constant and $r \geq 0$
- Log curve maps a narrow range of low graylevels in input image into a wider range of output levels.
- Expands range of dark image pixels while shrinking bright range.
- Inverse log expands range of bright image pixels while shrinking dark range.

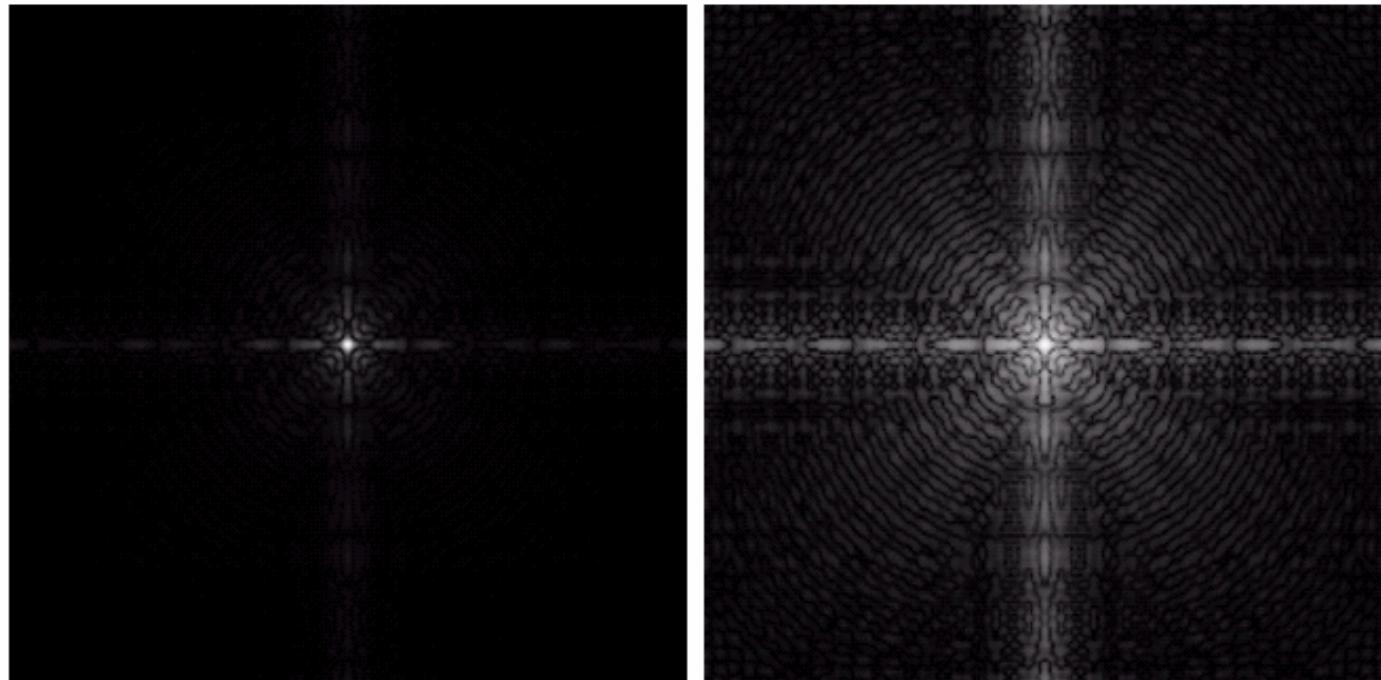
Example of Logarithm Image

- Fourier spectrum image can have intensity range from 0 to 10^6 or higher.
- Log transform lets us see the detail dominated by large intensity peak.
 - Must now display [0,6] range instead of [0, 10^6] range.
 - Rescale [0,6] to the [0,255] range.

a b

FIGURE 3.5

(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



Power-Law Transformations

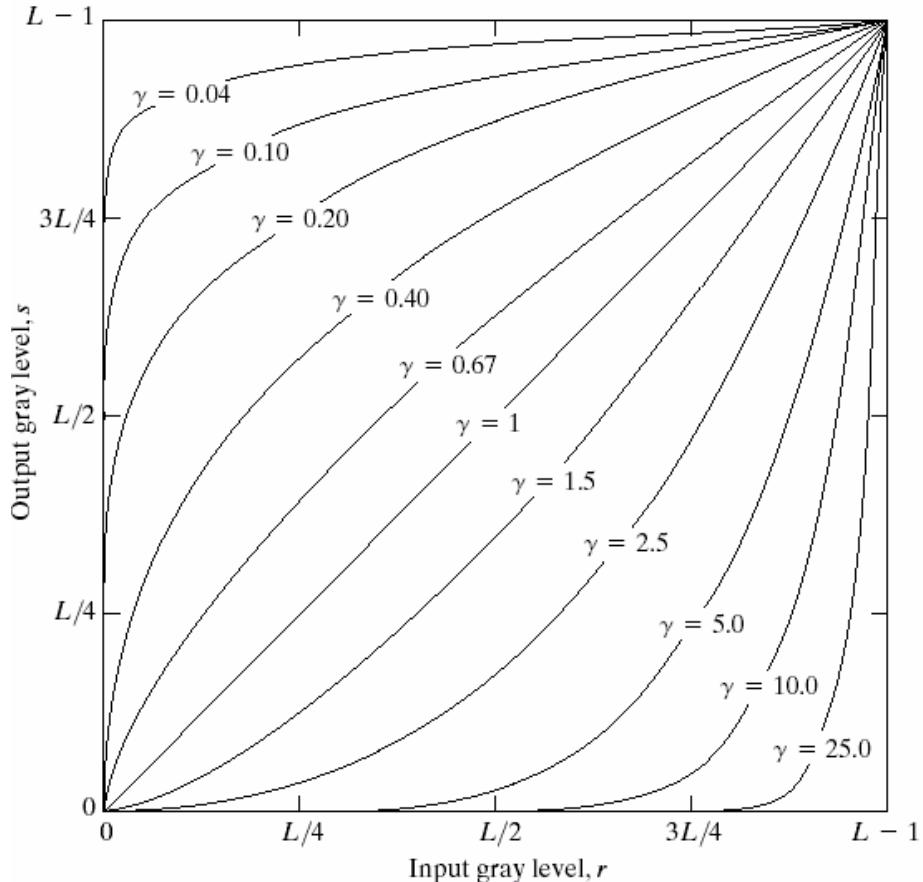


FIGURE 3.6 Plots of the equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases).

$$s = cr^\gamma$$

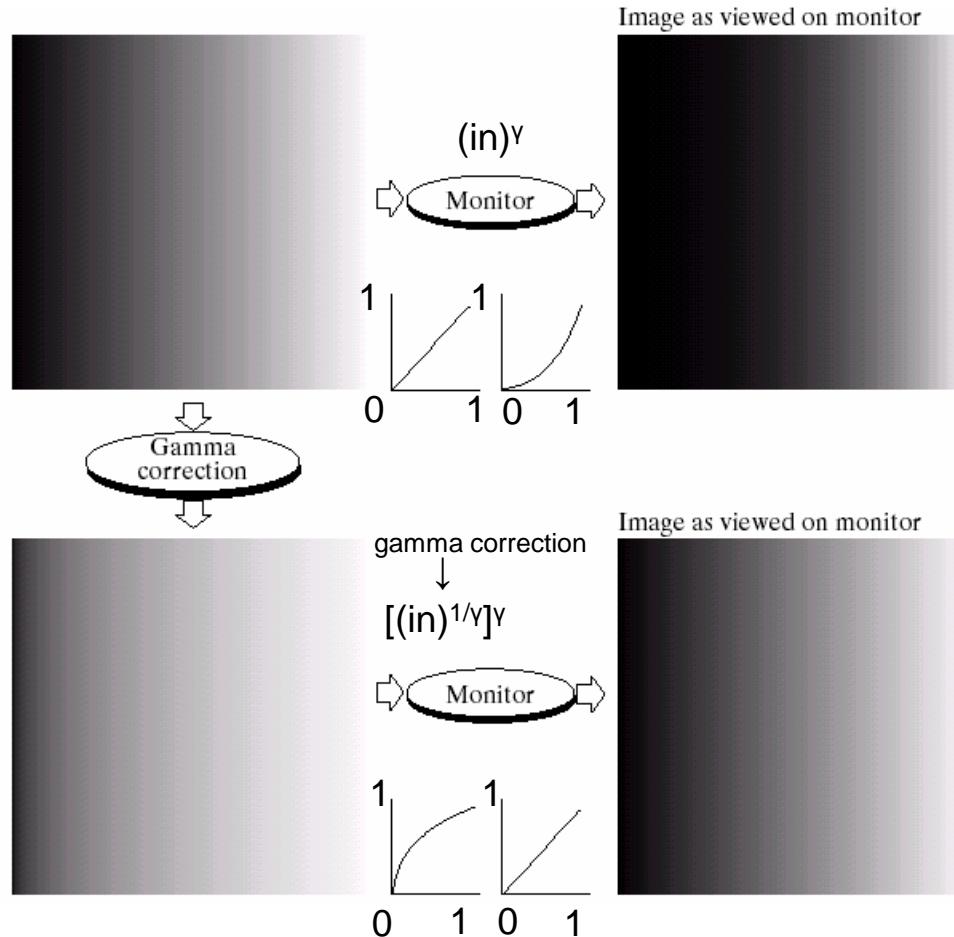
- c and γ are positive constants
- Power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels.
- $c = \gamma = 1 \Rightarrow$ identity function

Gamma Correction

a b
c d

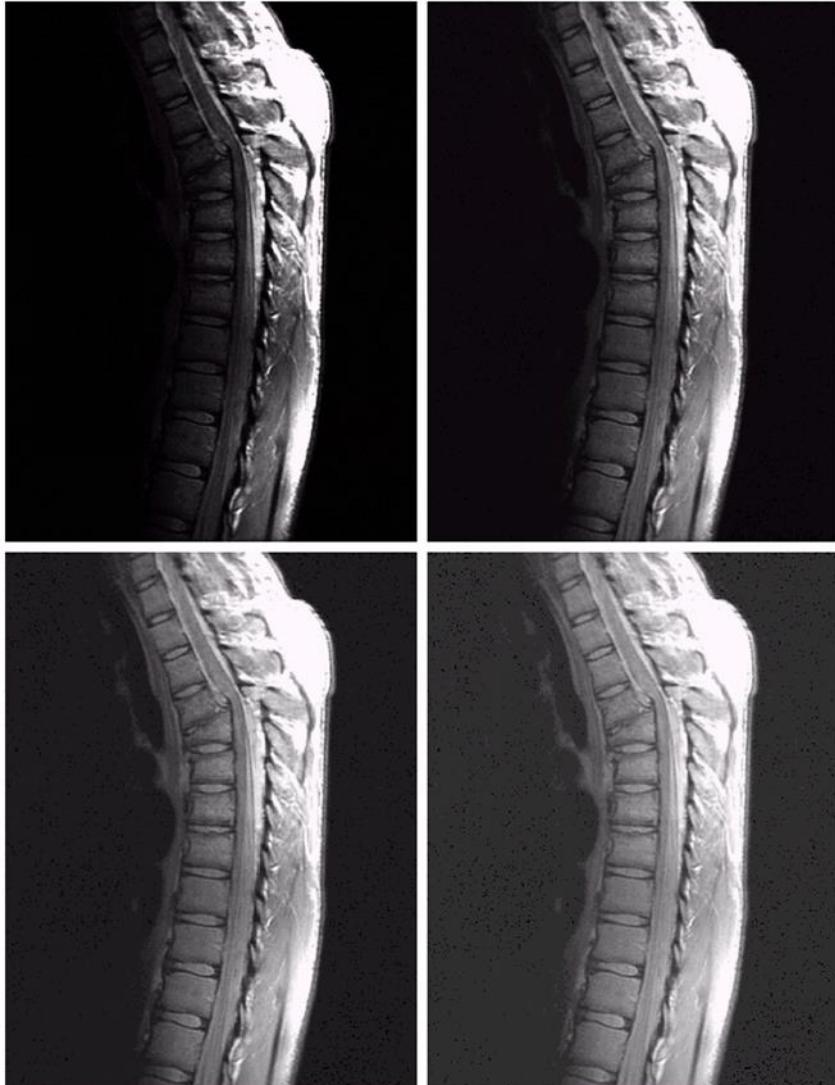
FIGURE 3.7

- (a) Linear-wedge gray-scale image.
- (b) Response of monitor to linear wedge.
- (c) Gamma-corrected wedge.
- (d) Output of monitor.



- Cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with γ varying from 1.8 to 2.5
- This darkens the picture.
- Gamma correction is done by preprocessing the image before inputting it to the monitor.

Example: MRI



a
b
c
d

FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3 , respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

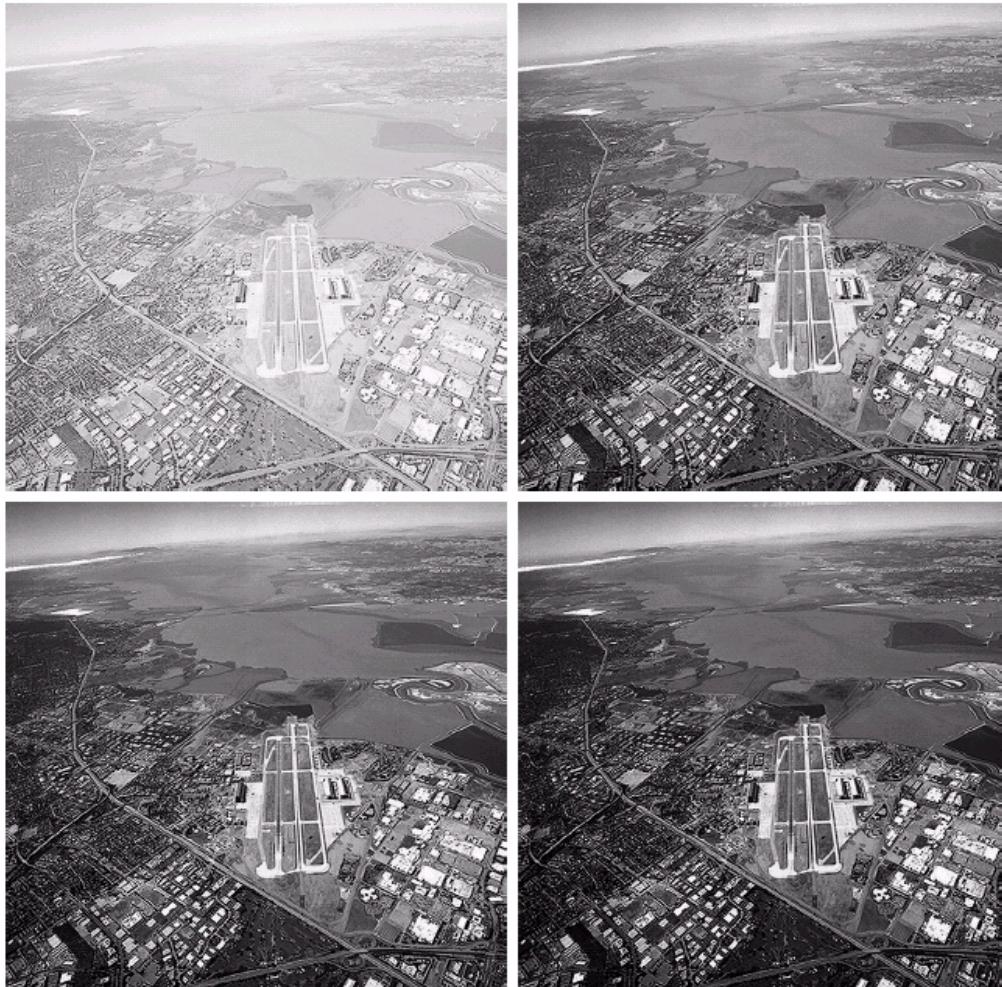
- (a) Dark MRI. Expand graylevel range for contrast manipulation
 $\Rightarrow \gamma < 1$
- (b) $\gamma = 0.6$, $c=1$
- (c) $\gamma = 0.4$ (best result)
- (d) $\gamma = 0.3$ (limit of acceptability)

When γ is reduced too much, the image begins to reduce contrast to the point where it starts to have a “washed-out” look, especially in the background

Example: Aerial Image

a
b
c
d

FIGURE 3.9
(a) Aerial image.
(b)–(d) Results of
applying the
transformation in
Eq. (3.2-3) with
 $c = 1$ and
 $\gamma = 3.0, 4.0,$ and
 $5.0,$ respectively.
(Original image
for this example
courtesy of
NASA.)



Washed-out
image. Shrink
graylevel range
 $\Rightarrow \gamma > 1$

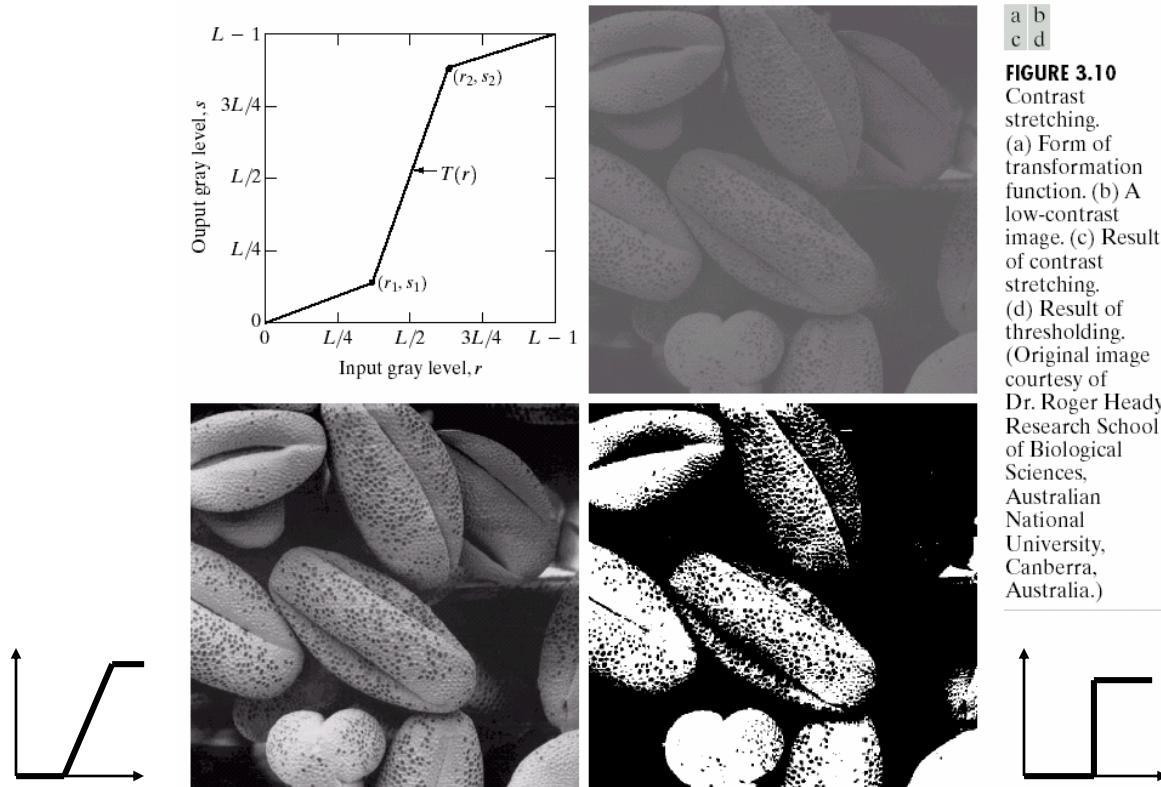
- (b) $\gamma = 3.0$
(suitable)
(c) $\gamma = 4.0$
(suitable)
(d) $\gamma = 5.0$
(High contrast;
the image has
areas that are
too dark; some
detail is lost)

Piecewise-Linear Transformation Functions

- Advantage:
 - The form of piecewise functions can be arbitrarily complex
- Disadvantage:
 - Their specification requires considerably more user input

Contrast Stretching

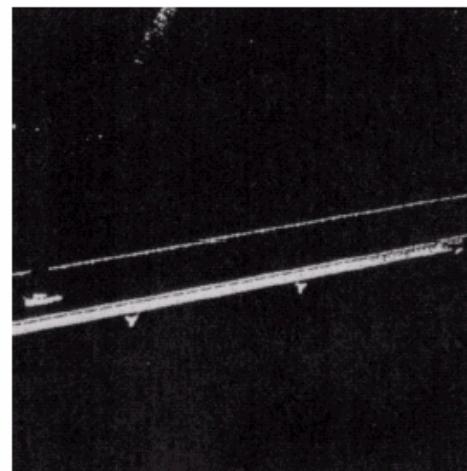
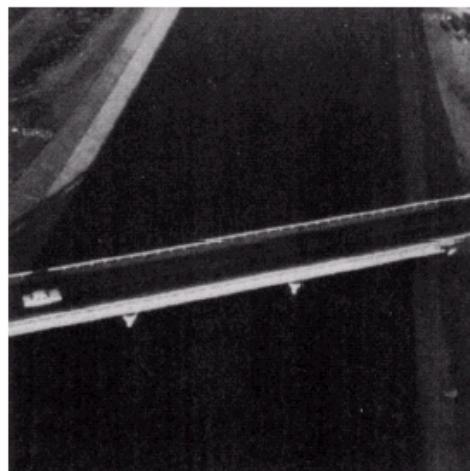
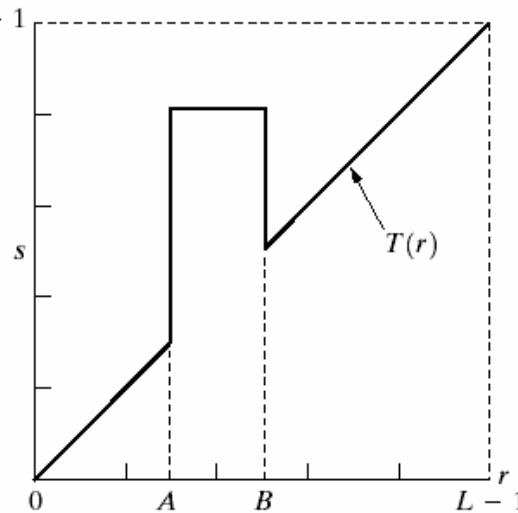
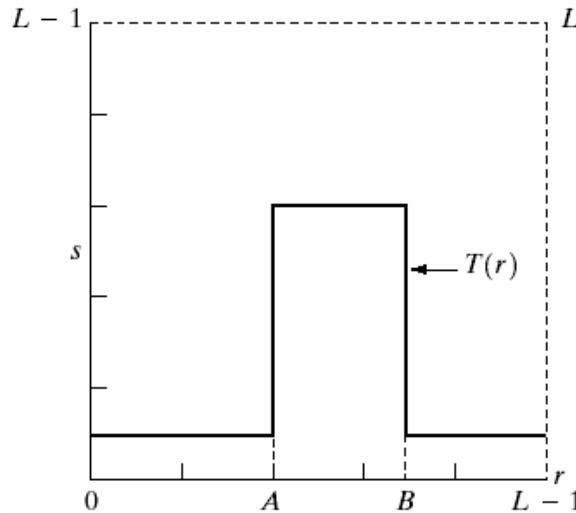
- Low contrast may be due to poor illumination, a lack of dynamic range in the imaging sensor, or even a wrong setting of a lens aperture during acquisition.
- Applied contrast stretching: $(r_1, s_1) = (r_{\min}, 0)$ and $(r_2, s_2) = (r_{\max}, L-1)$



a
b
c
d

FIGURE 3.10
Contrast stretching.
(a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Graylevel Slicing



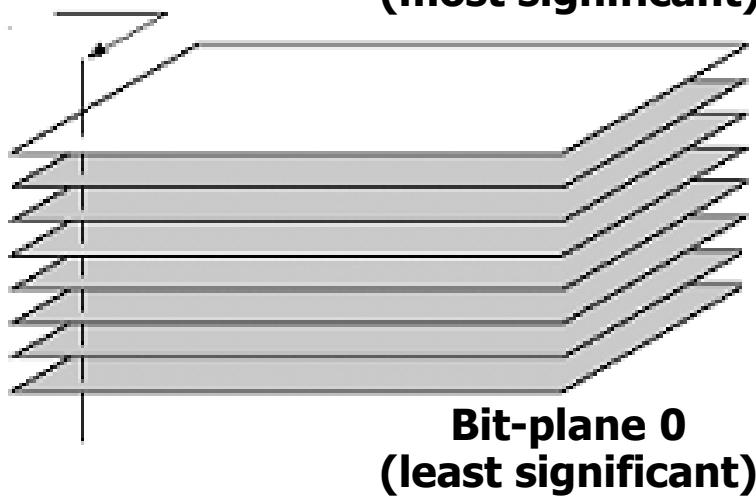
a	b
c	d

FIGURE 3.11

- (a) This transformation highlights range $[A, B]$ of gray levels and reduces all others to a constant level.
- (b) This transformation highlights range $[A, B]$ but preserves all other levels.
- (c) An image.
- (d) Result of using the transformation in (a).

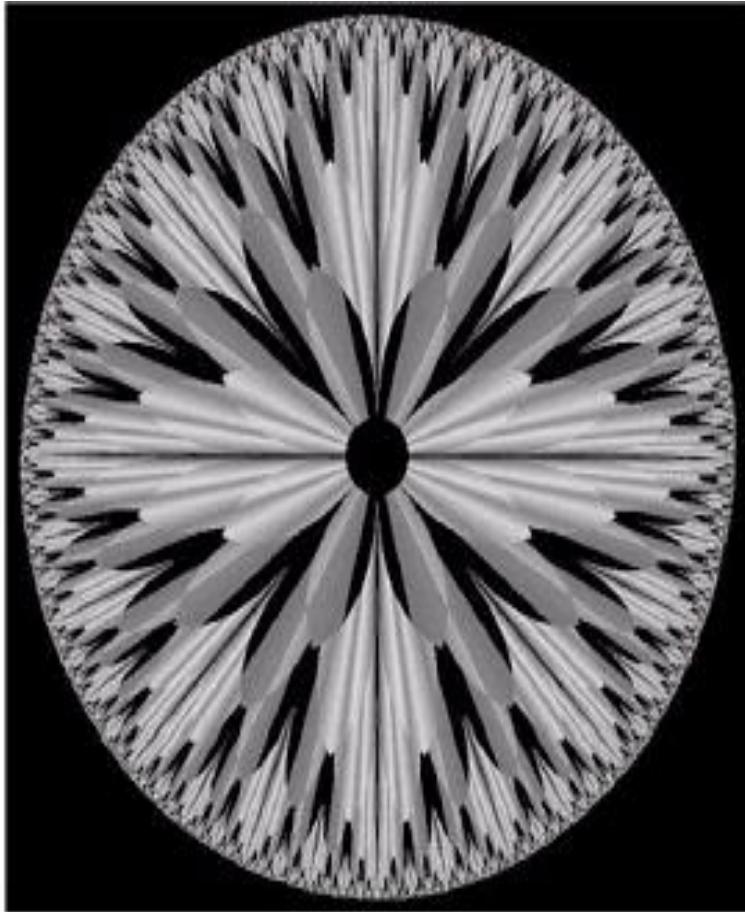
Bit-plane slicing

One 8-bit byte



- Highlighting the contribution made to total image appearance by specific bits
- Suppose each pixel is represented by 8 bits
Higher-order bits contain the majority of the visually significant data
Useful for analyzing the relative importance played by each bit of the image

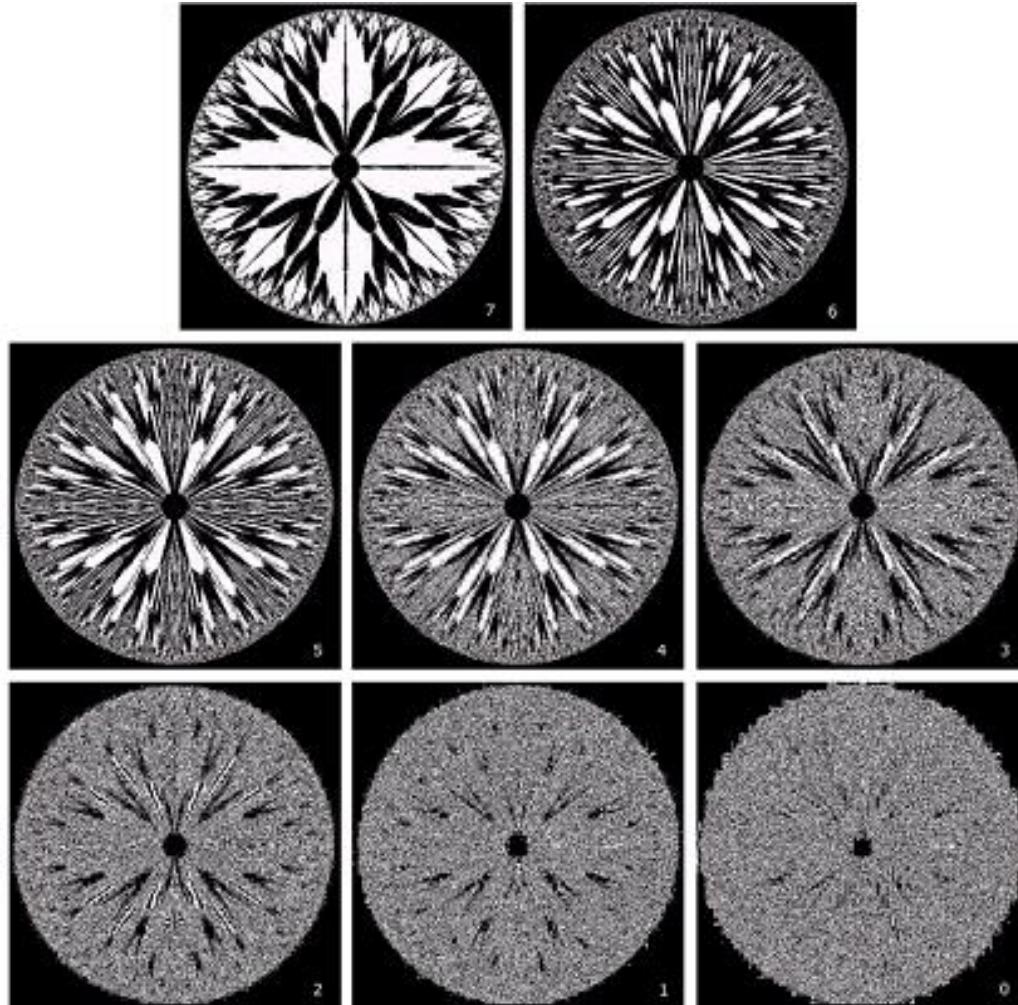
Example



An 8-bit fractal image

- The (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding graylevel transformation.
 - Map all levels between 0 and 127 to 0
 - Map all levels between 129 and 255 to 255

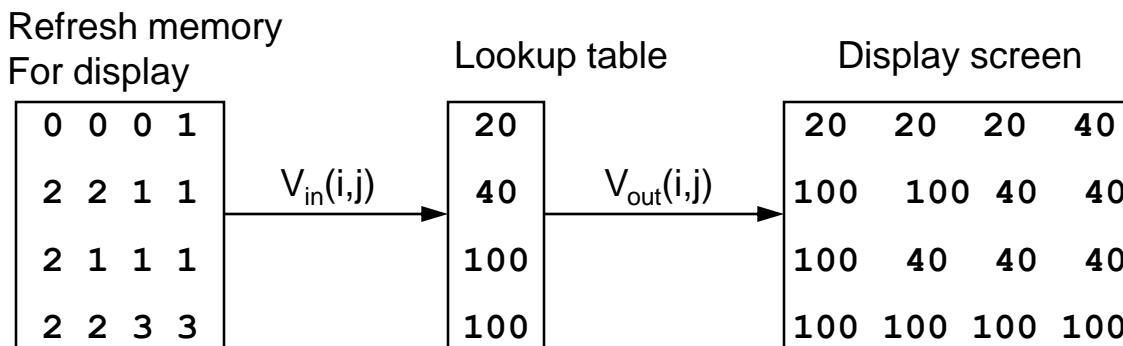
8-Bit Planes



	Bit-plane 7	Bit-plane 6
Bit-plane 5	Bit-plane 4	Bit-plane 3
Bit-plane 2	Bit-plane 1	Bit-plane 0

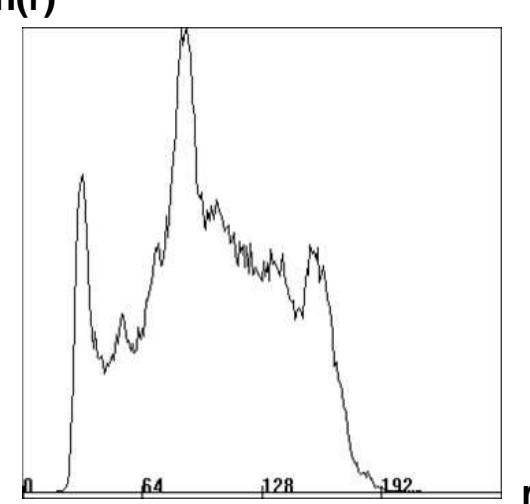
Hardware LUTs

- All point operations can be implemented by LUTs.
- Hardware LUTs operate on the data as it is being displayed.
- It's an efficient means of applying transformations because changing display characteristics only requires loading a new table and not the entire image.
- For a 1024x1024 8-bit image, this translates to 256 entries instead of one million.
- LUTs do not alter the contents of original image (nondestructive).



Histogram

- A histogram of a digital image with gray levels in the range $[0, L-1]$ is a discrete function $h(r_k) = n_k$
 - r_k : the k^{th} gray level
 - n_k : the number of pixels in the image having gray level r_k
- The sum of all histogram entries is equal to the total number of pixels in the image.



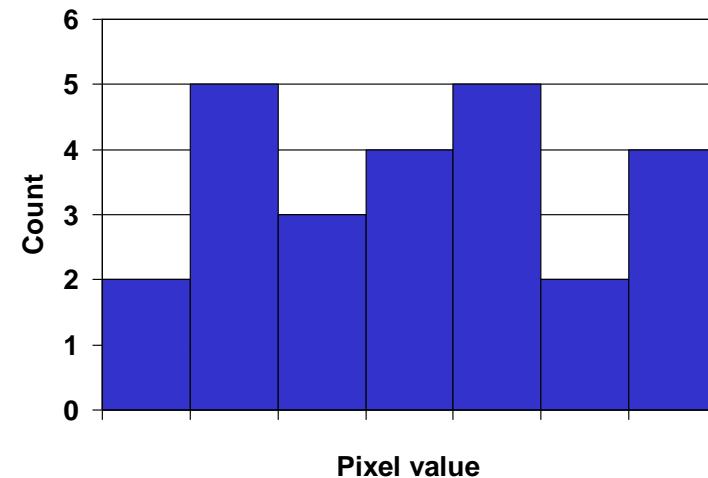
Histogram Example

5x5 image

2	3	4	4	6
1	2	4	5	6
1	1	5	6	6
0	1	3	3	4
0	1	2	3	4

Graylevel	Count
0	2
1	5
2	3
3	4
4	5
5	2
6	4
Total	25

Plot of the Histogram



Histogram evaluation:

```
for(i=0; i<MXGRAY; i++) H[i] = 0;  
for(i=0; i<total; i++) H[in[i]]++;
```

Normalized Histogram

- Divide each histogram entry at gray level r_k by the total number of pixels in the image, n

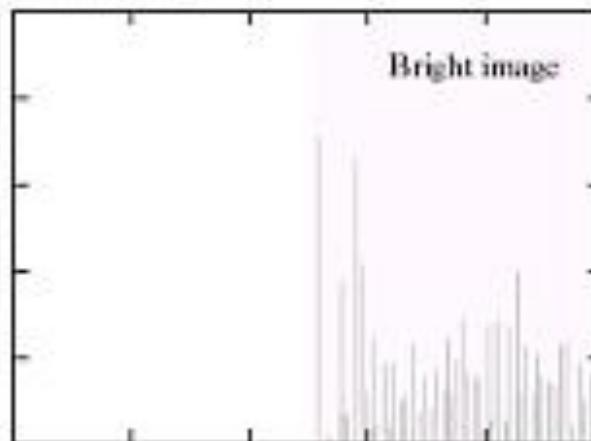
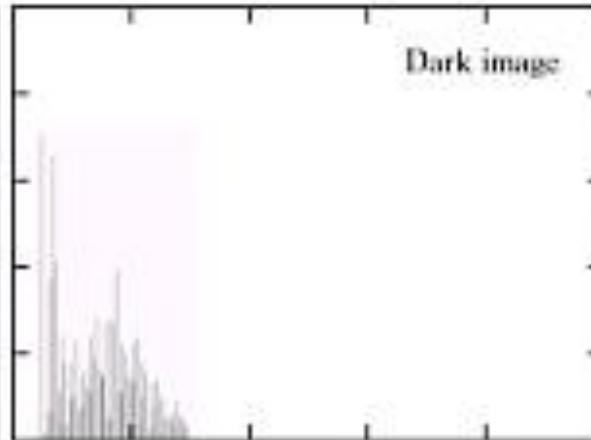
$$p(r_k) = n_k / n$$

- $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k
- The sum of all components of a normalized histogram is equal to 1.

Histogram Processing

- Basic for numerous spatial domain processing techniques.
- Used effectively for image enhancement:
 - Histogram stretching
 - Histogram equalization
 - Histogram matching
- Information inherent in histograms also is useful in image compression and segmentation.

Example: Dark/Bright Images



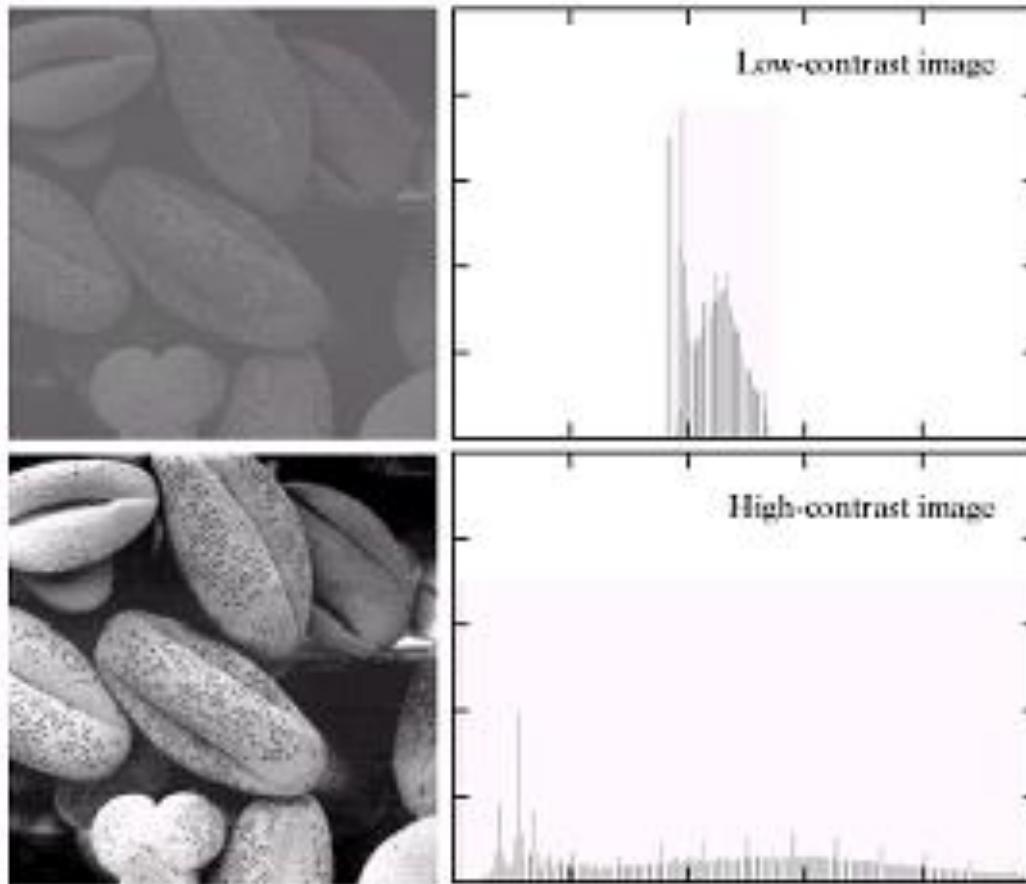
Dark image

Components of histogram are concentrated on the low side of the gray scale.

Bright image

Components of histogram are concentrated on the high side of the gray scale.

Example: Low/High Contrast Images



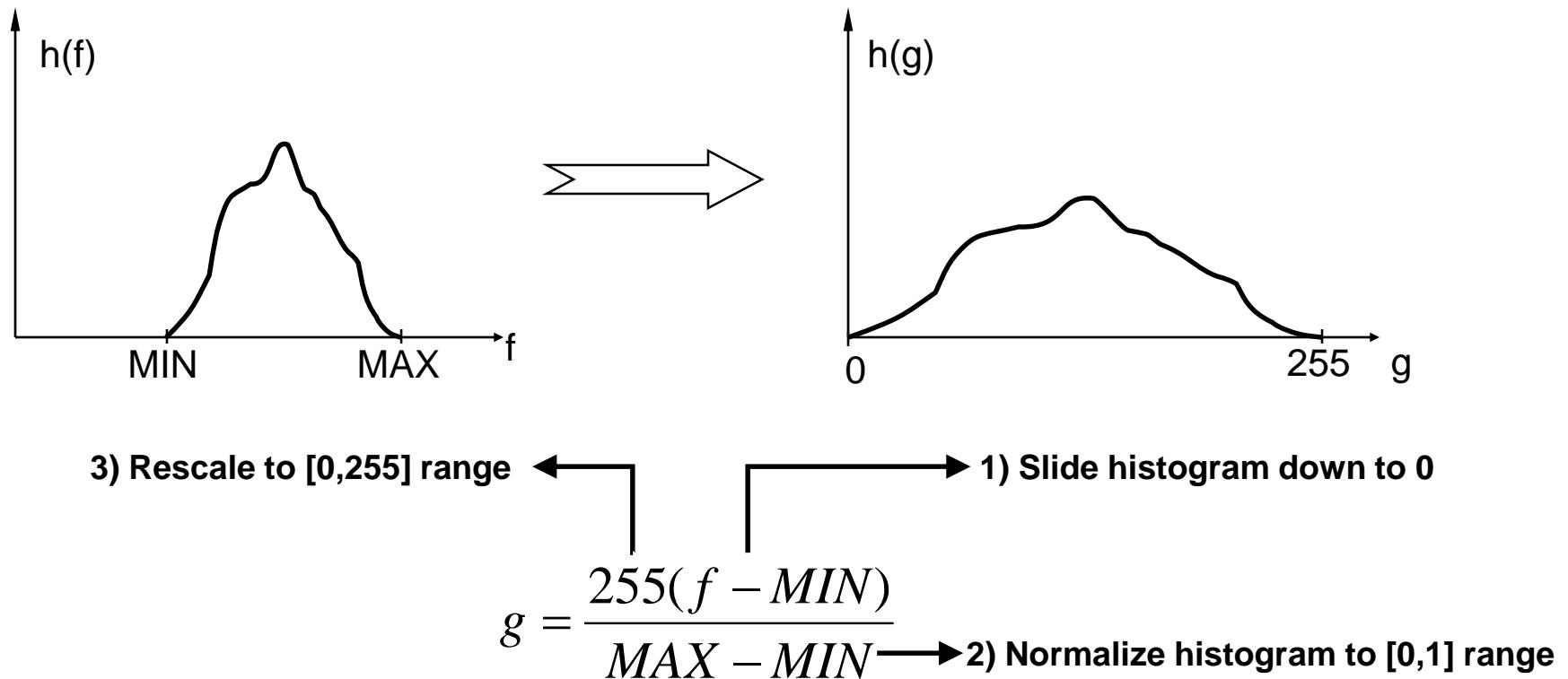
Low-contrast image

histogram is narrow
and centered toward
the middle of the
gray scale

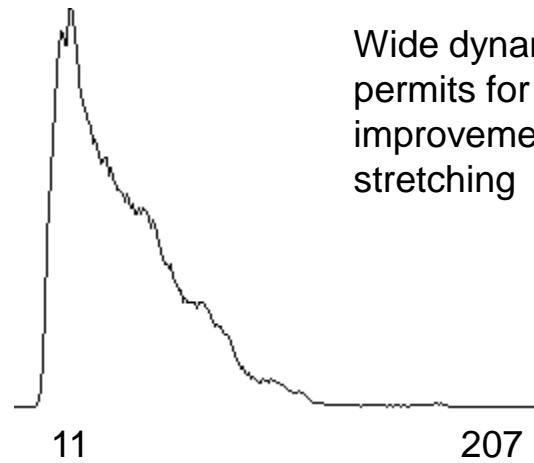
High-contrast image

histogram covers broad
range of the gray scale
and the distribution of
pixels is not too far from
uniform, with very few
vertical lines being much
higher than the others

Histogram Stretching



Example (1)



Wide dynamic range
permits for only a small
improvement after histogram
stretching

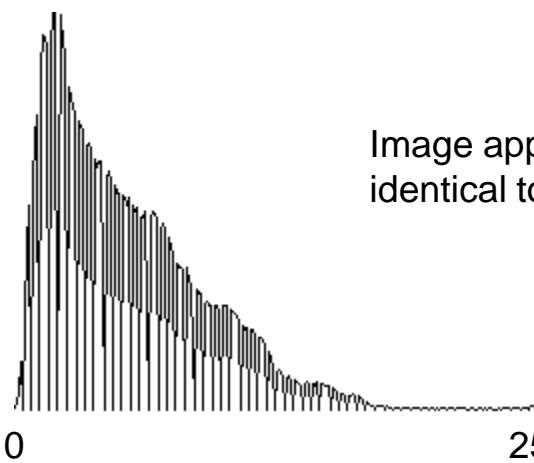
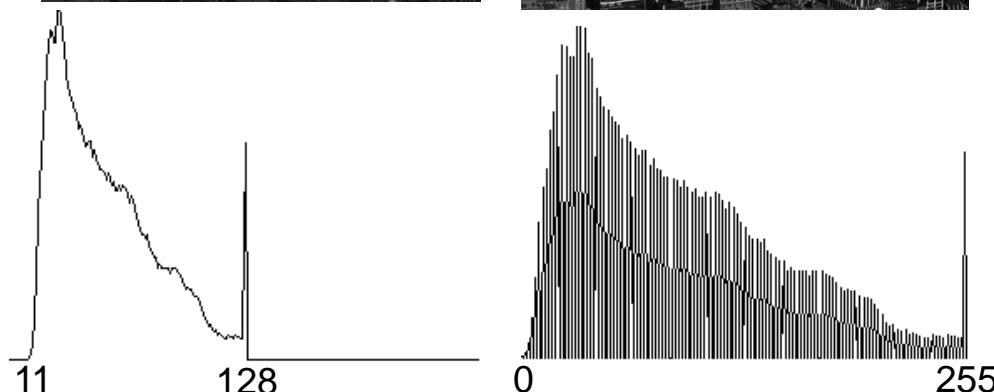


Image appears virtually
identical to original

Example (2)

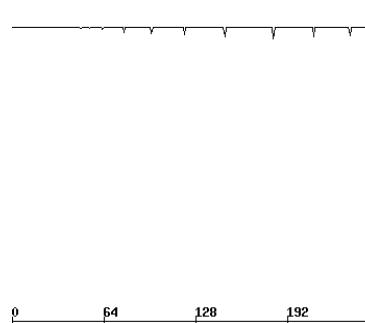
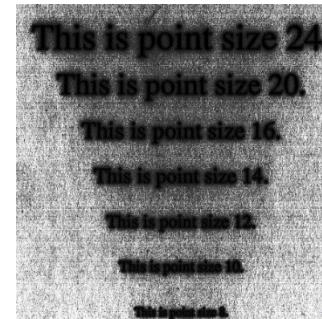
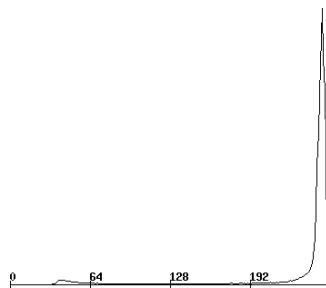
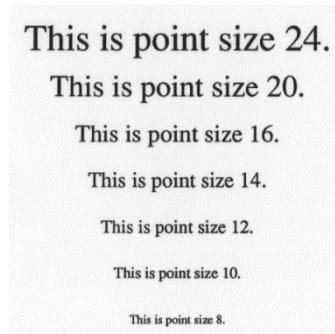
- Improve effectiveness of histogram stretching by clipping intensities first



Flat histogram: every graylevel
is equally present in image

Histogram Equalization (1)

- Produce image with flat histogram
- All graylevels are equally likely
- Appropriate for images with wide range of graylevels
- Inappropriate for images with few graylevels (see below)



Histogram Equalization (2)

Objective: we want a uniform histogram.

Rationale: maximize image entropy.

$$h_1(v_{out}) = \text{constant} = \frac{\text{total}}{MXGRAY}$$

$$= h_{avg}$$

$$c_1(v_{out}) = (v_{out} + 1) * h_{avg}$$

This is a special case of histogram matching.

Perfectly flat histogram: $H[i] = \text{total}/MXGRAY$ for $0 \leq i < MXGRAY$.

If $H[v] = k * h_{avg}$ then v must be mapped onto k different levels, from v_1 to v_k . This is a one-to many mapping.

Histogram Equalization Mappings

Rule 1: Always map v onto $(v_1 + v_k)/2$. (This does not result in a flat histogram, but one where brightness levels are spaced apart).

Rule 2: Assign at random one of the levels in $[v_1, v_k]$. This can result in a loss of contrast if the original histogram had two distinct peaks that were far apart (i.e., an image of text).

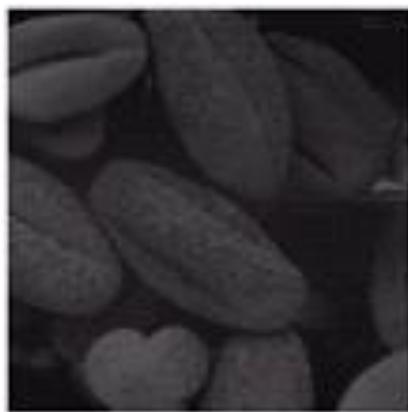
Rule 3: Examine neighborhood of pixel, and assign it a level from $[v_1, v_k]$ which is closest to neighborhood average. This can result in bluriness; more complex.

Rule (1) creates a lookup table beforehand.

Rules (2) and (3) are runtime operations.

Example (1)

before



after



Histogram
equalization



Example (2)

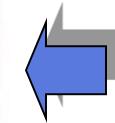
before



after



Histogram
equalization



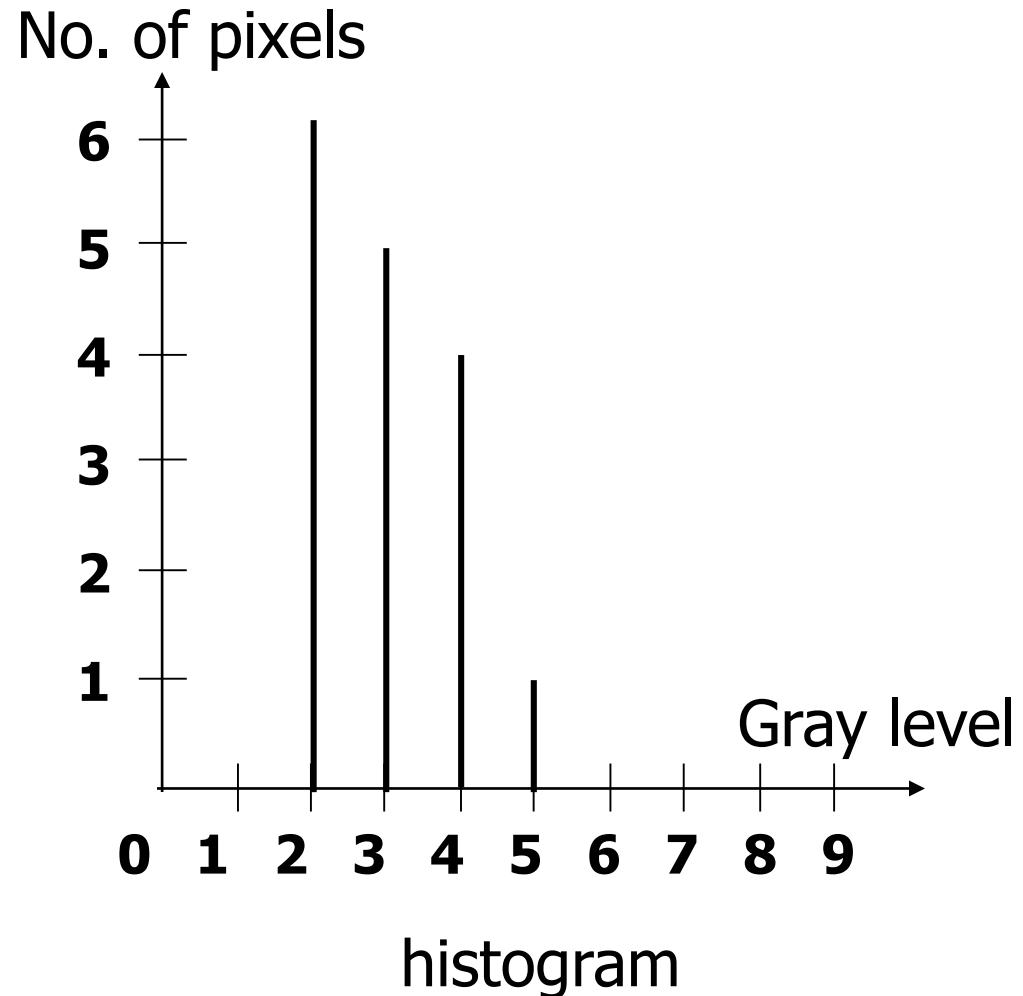
The quality is
not improved
much because
the original
image already
has a wide
graylevel scale

Implementation (1)

2	3	3	2
4	2	4	3
3	2	3	5
2	4	2	4

4x4 image

Gray scale = [0,9]



Implementation (2)

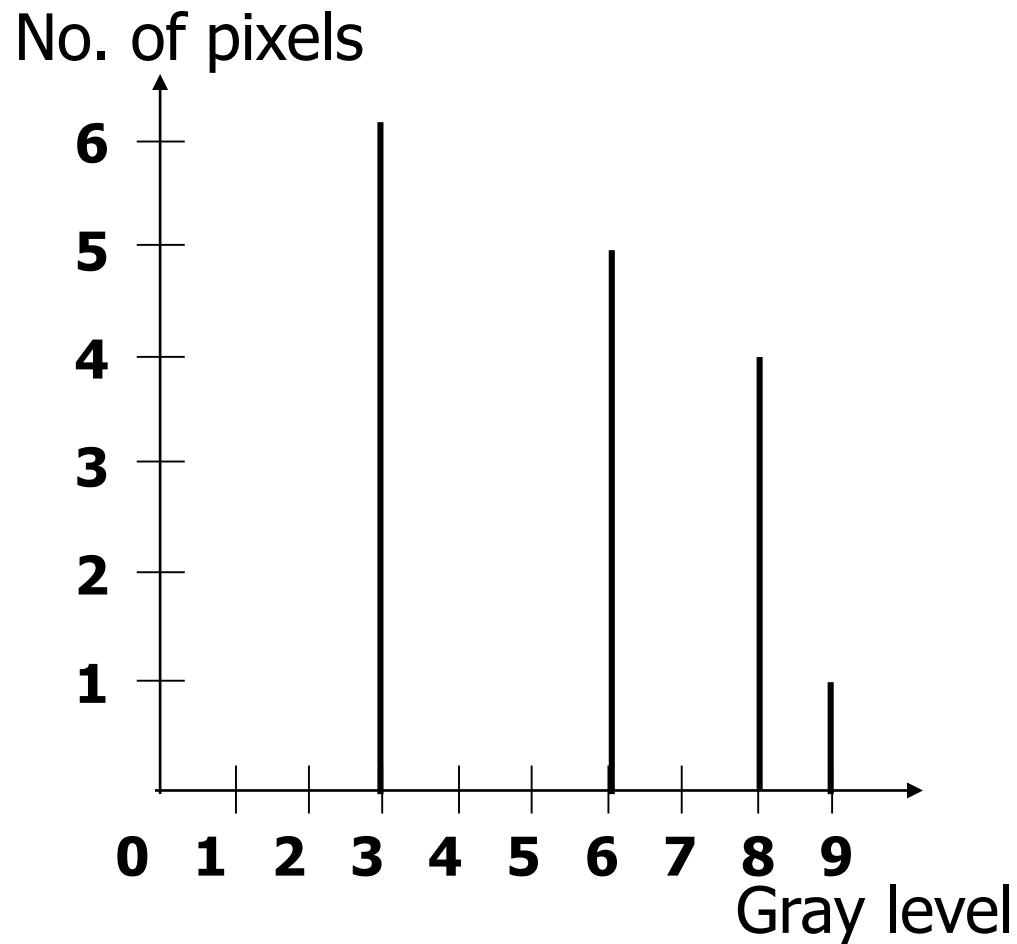
Gray Level(j)	0	1	2	3	4	5	6	7	8	9
No. of pixels	0	0	6	5	4	1	0	0	0	0
$\sum_{j=0}^k n_j$	0	0	6	11	15	16	16	16	16	16
$s = \sum_{j=0}^k \frac{n_j}{n}$	0	0	6/16	11/16	15/16	16/16	16/16	16/16	16/16	16/16
$s \times 9$	0	0	3.3 ≈3	6.1 ≈6	8.4 ≈8	9	9	9	9	9

Implementation (3)

3	6	6	3
8	3	8	6
6	3	6	9
3	8	3	8

Output image

Gray scale = [0,9]



Note (1)

- Histogram equalization distributes the graylevels to reach maximum gray (white) because the cumulative distribution function equals 1 when $0 \leq r \leq L-1$
- If $\sum_{j=0}^k n_j$ is slightly different among consecutive k , those graylevels will be mapped to (nearly) identical values as we have to produce an integer grayvalue as output
- Thus, the discrete transformation function cannot guarantee a one-to-one mapping

Note (2)

- The implementation described above is widely interpreted as histogram equalization.
- It is readily implemented with a LUT.
- It does not produce a strictly flat histogram.
- There is a more accurate solution. However, it may require a one-to-many mapping that cannot be implemented with a LUT.

Better Implementation (1)

```
void histeq(imageP I1, imageP I2)
{
    int i, R;
    int left[MXGRAY], width[MXGRAY];
    uchar *in, *out;
    long total, Hsum, Havg, histo[MXGRAY];

    /* total number of pixels in image */
    total = (long) I1->width * I1->height;

    /* init I2 dimensions and buffer */
    I2->width = I1->width;
    I2->height = I1->height;
    I2->image = (uchar *) malloc(total);

    /* init input and output pointers */
    in = I1->image;                      /* input image buffer */
    out = I2->image;                     /* output image buffer */

    /* compute histogram */
    for(i=0; i<MXGRAY; i++) histo[i] = 0; /* clear histogram */
    for(i=0; i<total; i++) histo[in[i]]++; /* eval histogram */

    R = 0;                                /* right end of interval      */
    Hsum = 0;                             /* cumulative value for interval */
    Havg = total / MXGRAY;                /* interval value for uniform histogram */

```

Better Implementation (2)

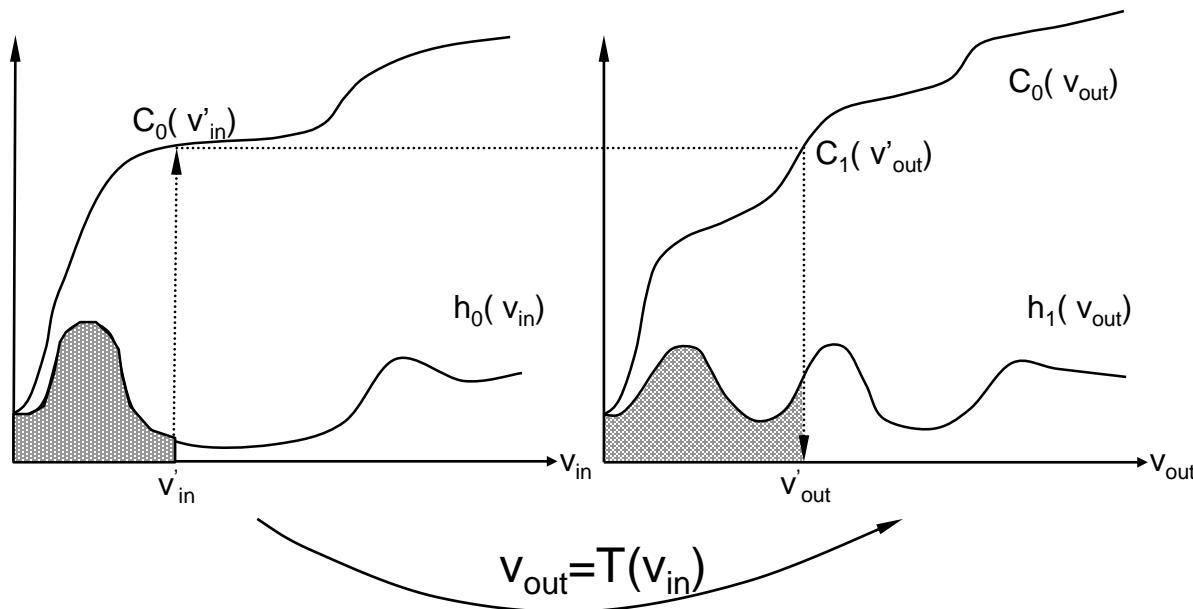
```
/* evaluate remapping of all input gray levels;
 * Each input gray value maps to an interval of valid output values.
 * The endpoints of the intervals are left[] and left[]+width[].
 */
for(i=0; i<MXGRAY; i++) {
    left[i] = R;                                /* left end of interval */
    Hsum += histo[i];                           /* cum. interval value */
    while(Hsum>Havg && R<MXGRAY-1) { /* make interval wider */
        Hsum -= Havg;                          /* adjust Hsum */
        R++;                                  /* update right end */
    }
    width[i] = R - left[i] + 1;                /* width of interval */
}

/* visit all input pixels and remap intensities */
for(i=0; i<total; i++) {
    if(width[in[i]] == 1) out[i] = left[in[i]];
    else {                                     /* in[i] spills over into width[] possible values */
        /* randomly pick from 0 to width[i] */
        R = ((rand()&0x7fff)*width[in[i]])>>15; /* 0 <= R < width */
        out[i] = left[in[i]] + R;
    }
}
```

Note

- Histogram equalization has a disadvantage:
it can generate only one type of output image.
- With histogram specification we can specify the shape of the histogram that we wish the output image to have.
- It doesn't have to be a uniform histogram.
- Histogram specification is a trial-and-error process.
- There are no rules for specifying histograms, and one must resort to analysis on a case-by-case basis for any given enhancement task.

Histogram Matching



In the figure above, $h()$ refers to the histogram, and $c()$ refers to its cumulative histogram. Function $c()$ is a *monotonically increasing function defined as:*

$$c(v) = \int_0^v h(u)du$$

Histogram Matching Rule

Let $v_{out} = T(v_{in})$ If $T()$ is a unique, monotonic function then

$$\int_0^{v_{out}} h_1(u)du = \int_0^{v_{in}} h_0(u)du$$

This can be restated in terms of the histogram matching rule:

$$c_1(v_{out}) = c_0(v_{in})$$

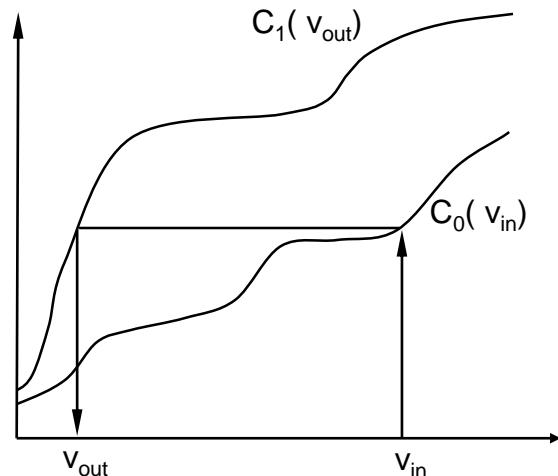
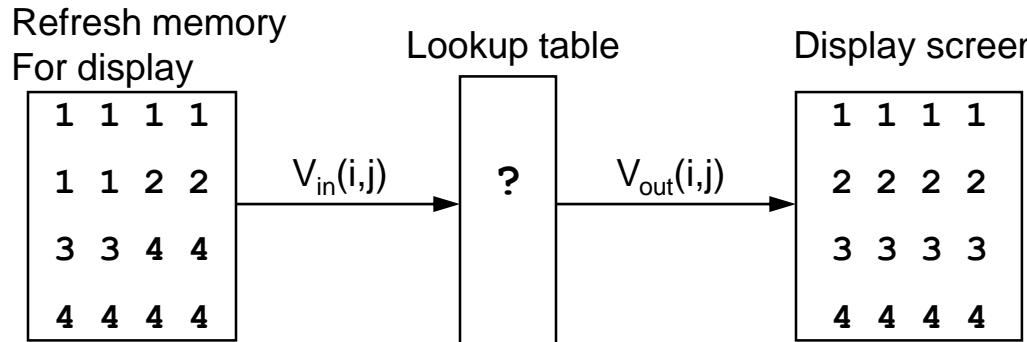
Where $c_1(v_{out}) = \# \text{ pixels } \leq v_{out}$, and $c_0(v_{in}) = \# \text{ pixels } \leq v_{in}$.

This requires that $v_{out} = c_1^{-1}(c_0(v_{in}))$

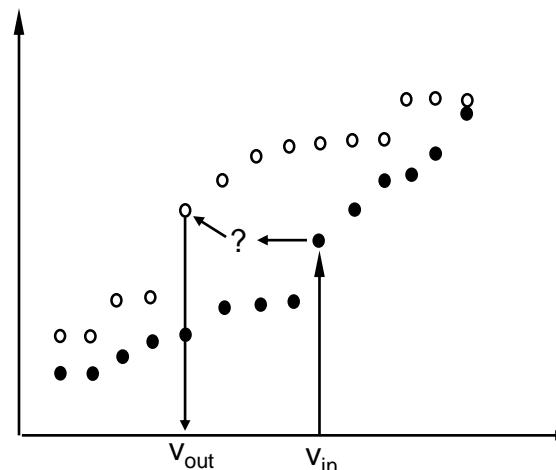
which is the basic equation for histogram matching techniques.

Histograms are Discrete

- Impossible to match all histogram pairs because they are discrete.



Continuous case



Discrete case

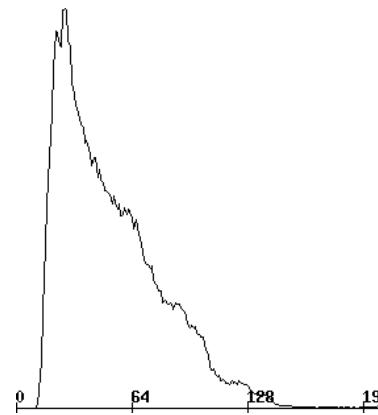
Problems with Discrete Case

- The set of input pixel values is a discrete set, and all the pixels of a given value are mapped to the same output value. For example, all six pixels of value one are mapped to the same value so it is impossible to have only four corresponding output pixels.
- No inverse for c_1 in $v_{out} = c_1^{-1}(c_0(v_{in}))$ because of discrete domain. Solution: choose v_{out} for which $c_1(v_{out})$ is closest to $c_0(v_{in})$.
- $v_{in} \rightarrow v_{out}$ such that $|c_1(v_{out}) - c_0(v_{in})|$ is a minimum

Histogram Matching Example (1)



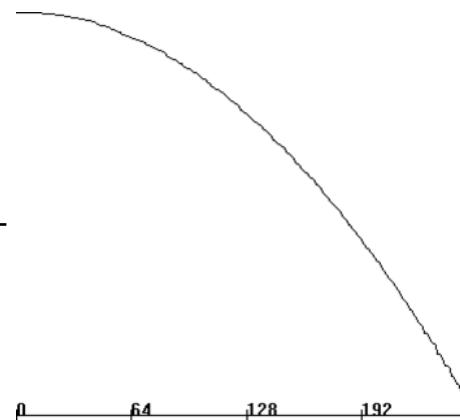
Input image



Input
Histogram



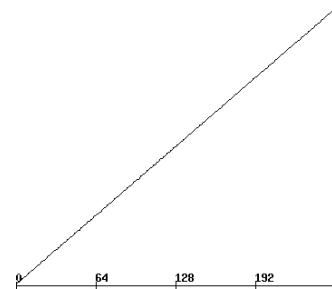
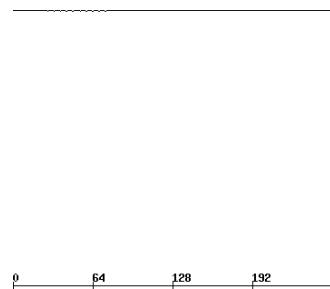
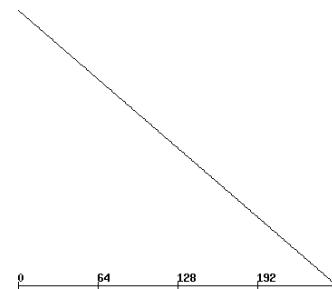
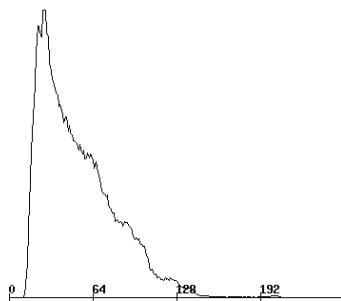
Output image



Histogram
match

Target
Histogram

Histogram Matching Example (2)



Implementation (1)

```
int histogramMatch(imageP I1, imageP histo, imageP I2)
{
    int i, p, R;
    int left[MXGRAY], right[MXGRAY];
    int total, Hsum, Havg, h1[MXGRAY], *h2;
    unsigned char *in, *out;
    double scale;

    /* total number of pixels in image */
    total = (long) I1->height * I1->width;

    /* init I2 dimensions and buffer */
    I2->width = I1->width;
    I2->height = I1->height;
    I2->image = (unsigned char *) malloc(total);

    in = I1->image;                      /* input image buffer */
    out = I2->image;                     /* output image buffer */

    for(i=0; i<MXGRAY; i++) h1[i] = 0; /* clear histogram */
    for(i=0; i<total; i++) h1[in[i]]++; /* eval histogram */
```

Implementation (2)

```
/* target histogram */
h2 = (int *) histo->image;

/* normalize h2 to conform with dimensions of I1 */
for(i=Havg=0; i<MXGRAY; i++) Havg += h2[i];
scale = (double) total / Havg;
if(scale != 1) for(i=0; i<MXGRAY; i++) h2[i] *= scale;

R = 0;
Hsum = 0;
/* evaluate remapping of all input gray levels;
   Each input gray value maps to an interval of valid output values.
   The endpoints of the intervals are left[] and right[] */
for(i=0; i<MXGRAY; i++) {
    left[i] = R;                      /* left end of interval */
    Hsum += h1[i];                    /* cumulative value for interval */
    while(Hsum>h2[R] && R<MXGRAY-1) { /* compute width of interval */
        Hsum -= h2[R];               /* adjust Hsum as interval widens */
        R++;                         /* update */
    }
    right[i] = R;                     /* init right end of interval */
}
```

Implementation (3)

```
/* clear h1 and reuse it below */
for(i=0; i<MXGRAY; i++) h1[i] = 0;

/* visit all input pixels */
for(i=0; i<total; i++) {
    p = left[in[i]];
    if(h1[p] < h2[p])      /* mapping satisfies h2 */
        out[i] = p;
    else          out[i] = p = left[in[i]] = MIN(p+1, right[in[i]]);
    h1[p]++;
}
}
```

Local Pixel Value Mappings

- Histogram processing methods are global, in the sense that pixels are modified by a transformation function based on the graylevel content of an entire image.
- We sometimes need to enhance details over small areas in an image, which is called a local enhancement.
- Solution: apply transformation functions based on graylevel distribution within pixel neighborhood.

General Procedure

- Define a square or rectangular neighborhood.
- Move the center of this area from pixel to pixel.
- At each location, the histogram of the points in the neighborhood is computed and histogram equalization, histogram matching, or other graylevel mapping is performed.
- Exploit easy histogram update since only one new row or column of neighborhood changes during pixel-to-pixel translation.
- Another approach used to reduce computation is to utilize nonoverlapping regions, but this usually produces an undesirable checkerboard effect.

Example: Local Enhancement

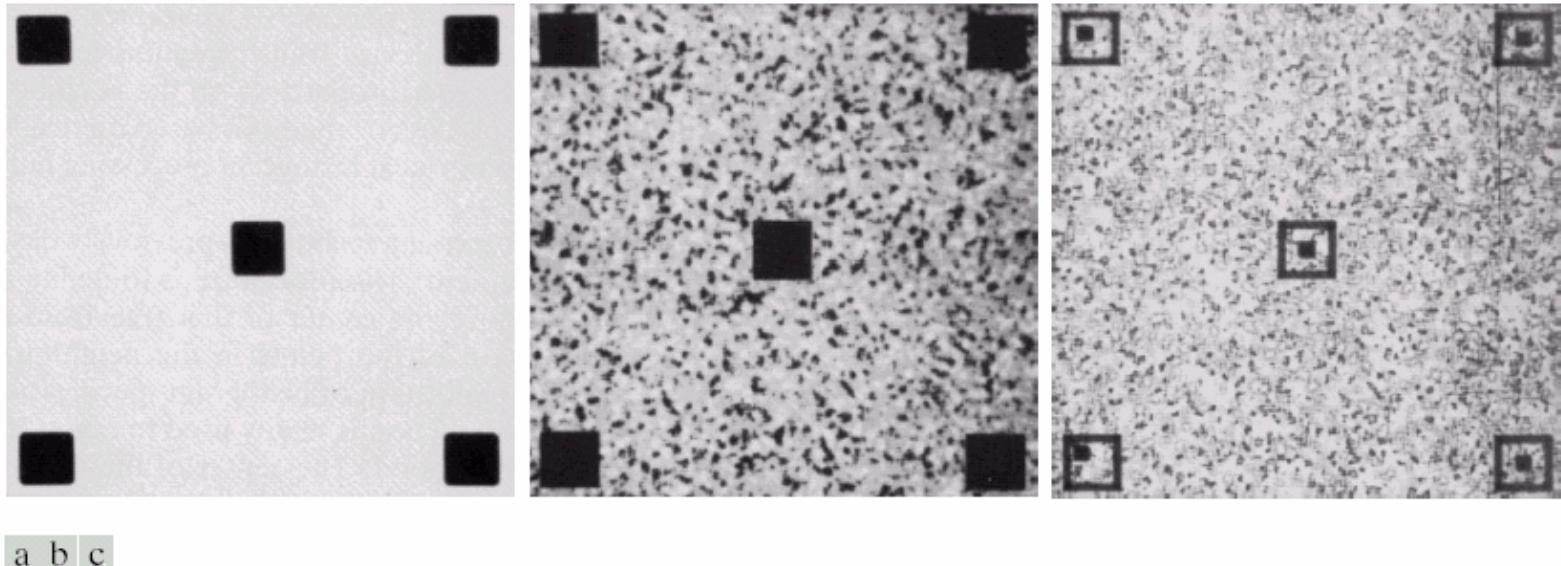


FIGURE 3.23 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a 7×7 neighborhood about each pixel.

- a) Original image (slightly blurred to reduce noise)
- b) global histogram equalization enhances noise & slightly increases contrast but the structural details are unchanged
- c) local histogram equalization using 7×7 neighborhood reveals the small squares inside of the larger ones in the original image.

Definitions (1)

$$\mu(x, y) = \frac{1}{n} \sum_{i,j} f(i, j) \quad \text{mean}$$

$$\sigma(x, y) = \sqrt{\frac{1}{n} \sum_{i,j} (f(i, j) - \mu(x, y))^2} \quad \text{standard deviation}$$

- Let $p(r_i)$ denote the normalized histogram entry for grayvalue r_i , for $0 \leq i < L$ where L is the number of graylevels.
- It is an estimate of the probability of occurrence of graylevel r_i .
- Mean m can be rewritten as

$$m = \sum_{i=0}^{L-1} r_i p(r_i)$$

Definitions (2)

- The nth moment of r about its mean is defined as

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n p(r_i)$$

- It follows that:

$$\mu_0(r) = 1 \quad \text{0th moment}$$

$$\mu_1(r) = 0 \quad \text{1st moment}$$

$$\mu_2(r) = \sum_{i=0}^{L-1} (r_i - m)^2 p(r_i) \quad \text{2nd moment}$$

- The second moment is known as variance $\sigma^2(r)$
- The standard deviation is the square root of the variance.
- The mean and standard deviation are measures of average grayvalue and average contrast, respectively.

Example: Statistical Differencing

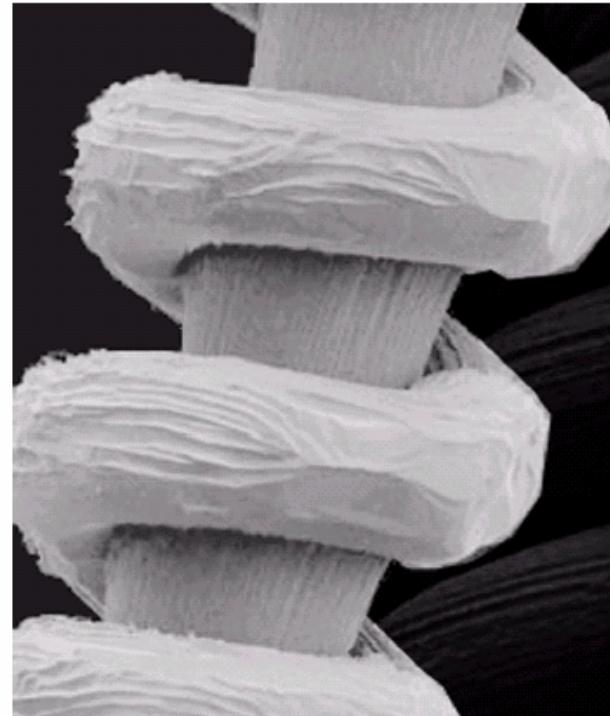
- Produces the same contrast throughout the image.
- Stretch $f(x, y)$ away from or towards the local mean to achieve a balanced local standard deviation throughout the image.
- σ_0 is the desired standard deviation and it controls the amount of stretch.
- The local mean can also be adjusted:

$$g(x, y) = \alpha m_0 + (1 - \alpha) \mu(x, y) + (f(x, y) - \mu(x, y)) \frac{\sigma_0}{\sigma(x, y)}$$

- m_0 is the mean to force locally and α controls the degree to which it is forced.
- To avoid problems when $\sigma(x, y) = 0$,
$$g(x, y) = \alpha m_0 + (1 - \alpha) \mu(x, y) + (f(x, y) - \mu(x, y)) \frac{\beta \sigma_0}{\sigma_0 + \beta \sigma(x, y)}$$
- Speedups can be achieved by dividing the image into blocks (tiles), exactly computing the mean and standard deviation at the center of each block, and then linearly interpolating between blocks in order to compute an approximation at any arbitrary position. In addition, the mean and standard deviation can be computed incrementally.

Example: Local Statistics (1)

FIGURE 3.24 SEM image of a tungsten filament and support, magnified approximately 130×. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene).

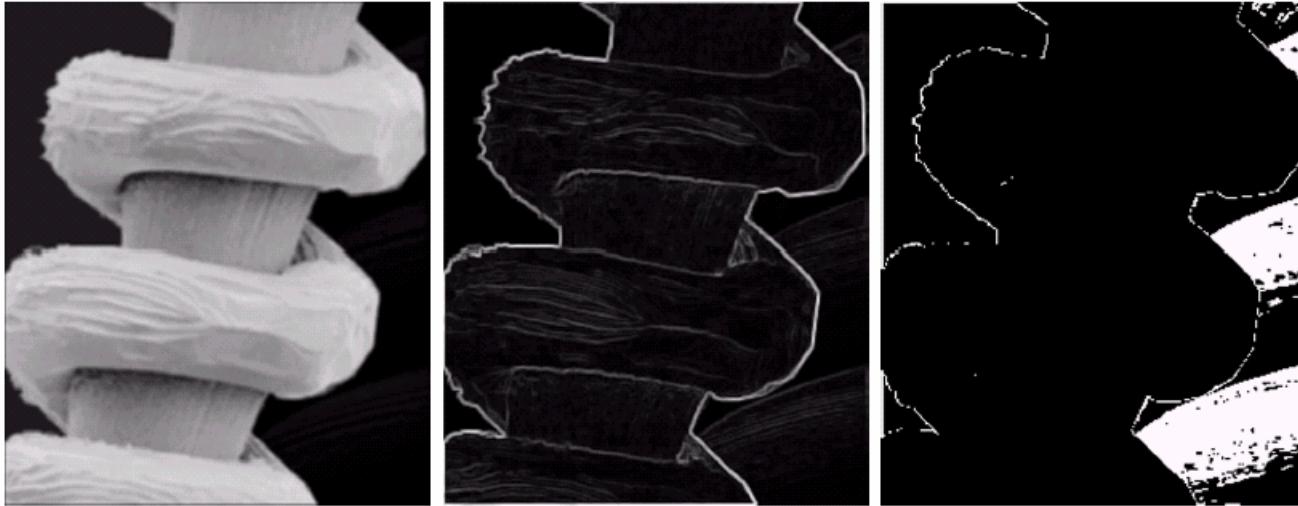


The filament in the center is clear.

There is another filament on the right side that is darker and hard to see.

Goal: enhance dark areas while leaving the light areas unchanged.

Example: Local Statistics (2)



a b c

FIGURE 3.25 (a) Image formed from all local means obtained from Fig. 3.24 using Eq. (3.3-21). (b) Image formed from all local standard deviations obtained from Fig. 3.24 using Eq. (3.3-22). (c) Image formed from all multiplication constants used to produce the enhanced image shown in Fig. 3.26.

Solution: Identify candidate pixels to be dark pixels with low contrast.
Dark: local mean $< k_0 * \text{global mean}$, where $0 < k_0 < 1$.
Low contrast: $k_1 * \text{global variance} < \text{local variance} < k_2 * \text{global variance}$, where $k_1 < k_2$.
Multiply identified pixels by constant $E > 1$. Leave other pixels alone.

Example: Local Statistics (3)



FIGURE 3.26
Enhanced SEM
image. Compare
with Fig. 3.24. Note
in particular the
enhanced area on
the right side of
the image.

Results for $E=4$, $k_0=0.4$, $k_1=0.02$, $k_2=0.4$. 3x3 neighborhoods used.

Arithmetic/Logic Operations

Prof. George Wolberg
Dept. of Computer Science
City College of New York

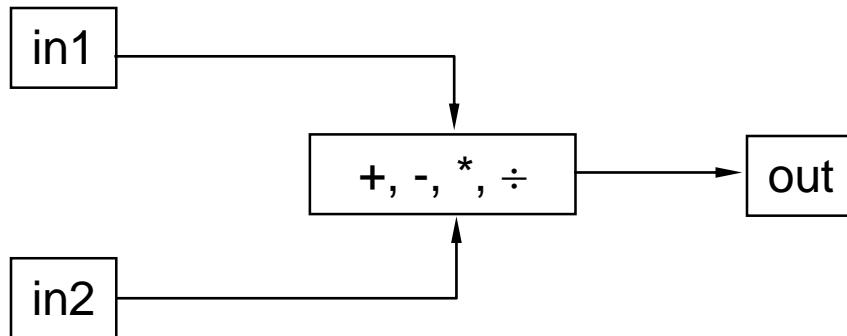
Objectives

- In this lecture we describe arithmetic and logic operations commonly used in image processing.
- Arithmetic ops:
 - Addition, subtraction, multiplication, division
 - Hybrid: cross-dissolves
- Logic ops:
 - AND, OR, XOR, BIC, ...

Arithmetic/Logic Operations

- Arithmetic/Logic operations are performed on a pixel-by-pixel basis between two images.
- Logic NOT operation performs only on a single image.
 - It is equivalent to a negative transformation.
- Logic operations treat pixels as binary numbers:
 - $158 \& 235 = 10011110 \& 11101011 = 10001010$
- Use of LUTs requires 16-bit rather than 8-bit indices:
 - Concatenate two 8-bit input pixels to form a 16-bit index into a 64K-entry LUT. Not commonly done.

Addition / Subtraction



Addition:

```
for(i=0; i<total; i++)
    out[i] = MIN(((int)in1[i]+in2[i]), 255);
```

Subtraction:

```
for(i=0; i<total; i++)
    out[i] = MAX(((int)in1[i]-in2[i]), 0);
```

Avoid overflow: clip result

Avoid underflow: clip result

Overflow / Underflow

- Default datatype for pixel is `unsigned char`.
- It is 1 byte that accounts for nonnegative range [0,255].
- Addition of two such quantities may exceed 255 (overflow).
- This will cause wrap-around effect:
 - 254: 11111110
 - 255: 11111111
 - 256: 100000000
 - 257: 100000001
- Notice that low-order byte reverts to 0, 1, ... when we exceed 255.
- Clipping is performed to prevent wrap-around.
- Same comments apply to underflow (result < 0).

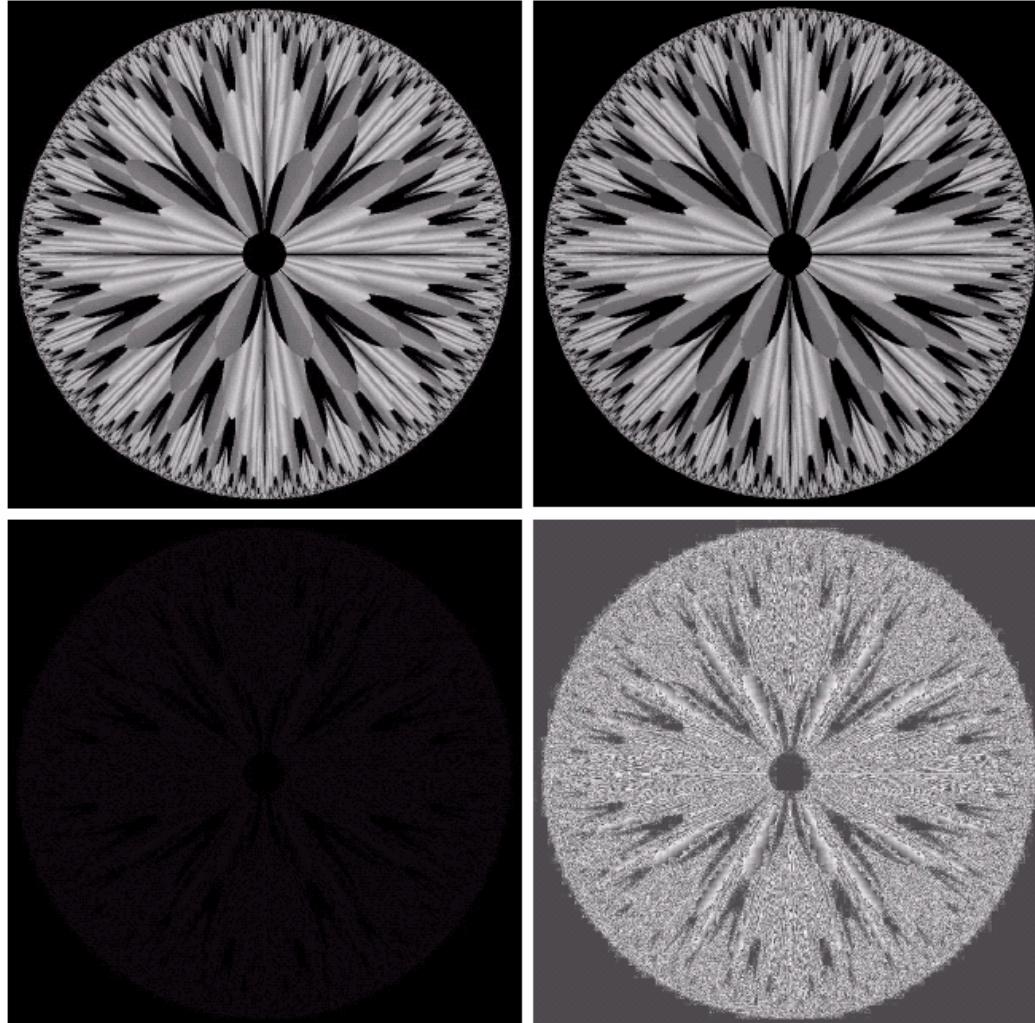
Implementation Issues

- The values of a subtraction operation may lie between -255 and 255. Addition: [0,510].
- Clipping prevents over/underflow.
- Alternative: scale results in one of two ways:
 1. Add 255 to every pixel and then divide by 2.
 - Values may not cover full [0,255] range
 - Requires **short** intermediate image
 - Fast and simple to implement
 2. Add negative of min difference (shift min to 0). Then, multiply all pixels by $255/(\text{max difference})$ to scale range to [0,255] interval.
 - Full utilization of [0,255] range
 - Requires **short** intermediate image
 - More complex and difficult to implement

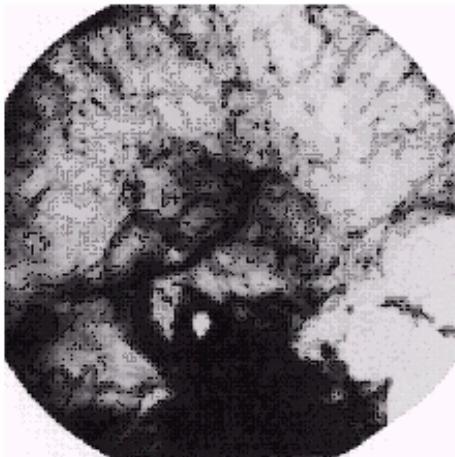
Example of Subtraction Operation

a	b
c	d

FIGURE 3.28
(a) Original fractal image.
(b) Result of setting the four lower-order bit planes to zero.
(c) Difference between (a) and (b).
(d) Histogram-equalized difference image.
(Original image courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA).



Example: Mask Mode Radiography



mask image $h(x,y)$



image $f(x,y)$ taken after
injection of a contrast
medium (iodine) into the
bloodstream, with mask
subtracted out.

Note:

- the background is dark because it doesn't change much in both images.
- the difference area is bright because it has a big change

- $h(x,y)$ is the mask, an X-ray image of a region of a patient's body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source
- $f(x,y)$ is an X-ray image taken after injection a contrast medium into the patient's bloodstream
- images are captured at TV rates, so the doctor can see how the medium propagates through the various arteries in an animation of $f(x,y)-h(x,y)$.

Arithmetic Operations: Cross-Dissolve

- Linearly interpolate between two images.
- Used to perform a fade from one image to another.
- Morphing can improve upon the results shown below.

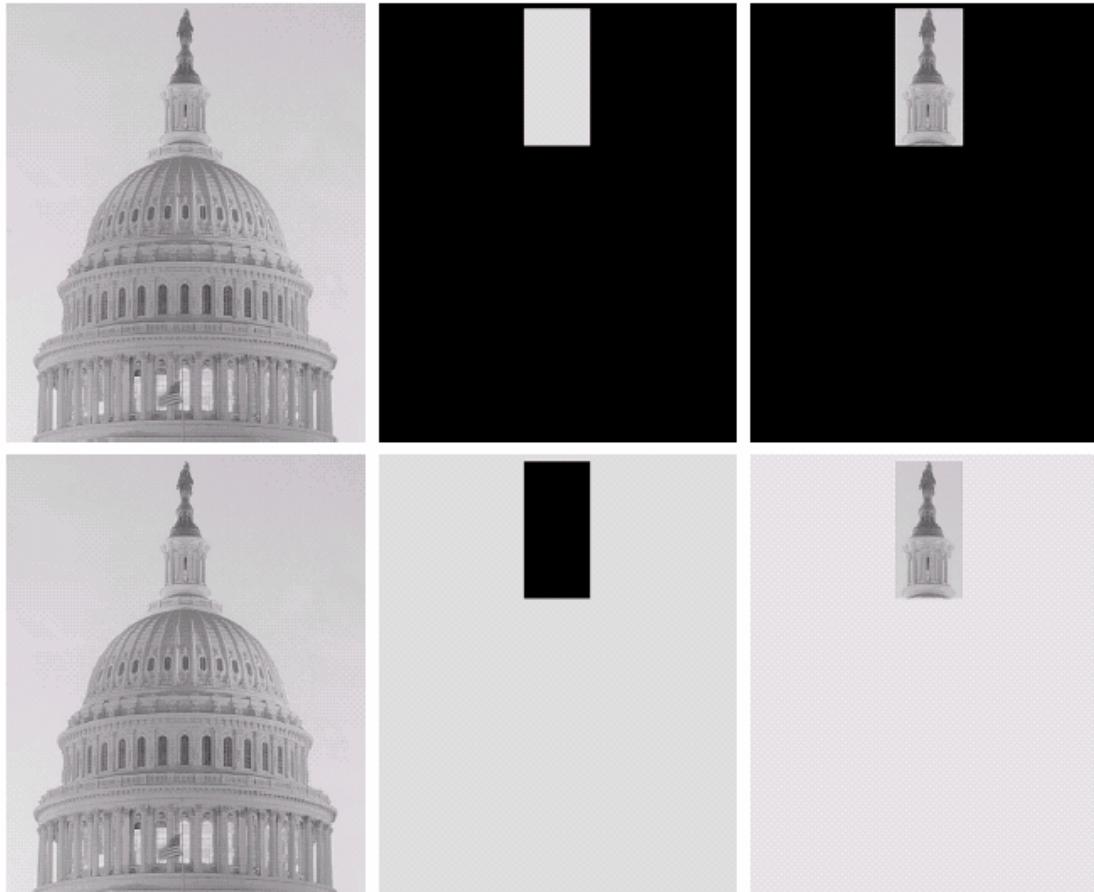
```
for(i=0; i<total; i++)
    out[i] = in1[i]*f + in2[i]*(1-f);
```



Masking

- Used for selecting subimages.
- Also referred to as region of interest (ROI) processing.
- In enhancement, masking is used primarily to isolate an area for processing.
- AND and OR operations are used for masking.

Example of AND/OR Operation



a	b	c
d	e	f

FIGURE 3.27
(a) Original image. (b) AND image mask.
(c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask.
(f) Result of operation OR on images (d) and (e).

Digital Halftoning

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we review digital halftoning techniques to convert grayscale images to bitmaps:
 - Unordered (random) dithering
 - Ordered dithering
 - Patterning
 - Error diffusion

Background

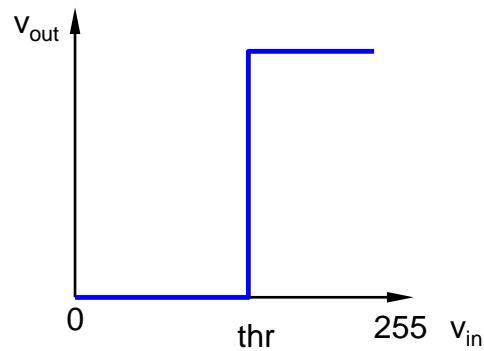
- An 8-bit grayscale image allows 256 distinct gray levels.
- Such images can be displayed on a computer monitor if the hardware supports the required number of intensity levels.
- However, some output devices print or display images with much fewer gray levels.
- In these cases, the grayscale images must be converted to binary images, where pixels are only black (0) or white (255).
- Thresholding is a poor choice due to objectionable artifacts.
- Strategy: sprinkle black-and-white dots to simulate gray.
- Exploit spatial integration (averaging) performed by eye.

Thresholding

- The simplest way to convert from grayscale to binary.



8 bpp (256 levels)



1 bpp (two-level)

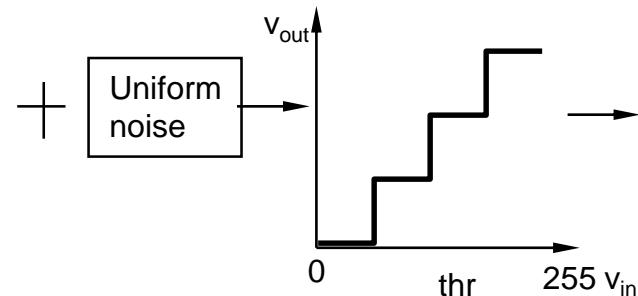
Loss of information is unacceptable.

Unordered Dither (1)

- Reduce quantization error by adding uniformly distributed white noise (dither signal) to the input image prior to quantization.
- Dither hides objectional artifacts.
- To each pixel of the image, add a random number in the range $[-m, m]$, where m is MXGRAY/quantization-levels.



8 bpp (256 levels)



3 bpp (8 levels)

Unordered Dither (2)

1 bpp



2 bpp



3 bpp



4 bpp



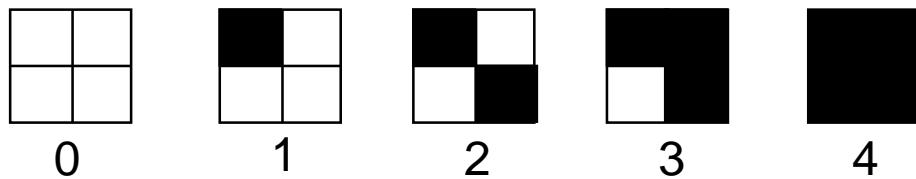
Quantization



Dither/
Quantization

Ordered Dithering

- Objective: expand the range of available intensities.
- Simulates n bpp images with m bpp, where $n > m$ (usually $m = 1$).
- Exploit eye's spatial integration.
 - Gray is due to average of black/white dot patterns.
 - Each dot is a circle of black ink whose area is proportional to $(1 - \text{intensity})$.
 - Graphics output devices approximate the variable circles of halftone reproductions.



- 2×2 pixel area of a bilevel display produces 5 intensity levels.
- $n \times n$ group of bilevel pixels produces $n^2 + 1$ intensity levels.
- Tradeoff: spatial vs. intensity resolution.

Dither Matrix (1)

- Consider the following 2x2 and 3x3 dither matrices:

$$D^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix} \quad D^{(3)} = \begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}$$

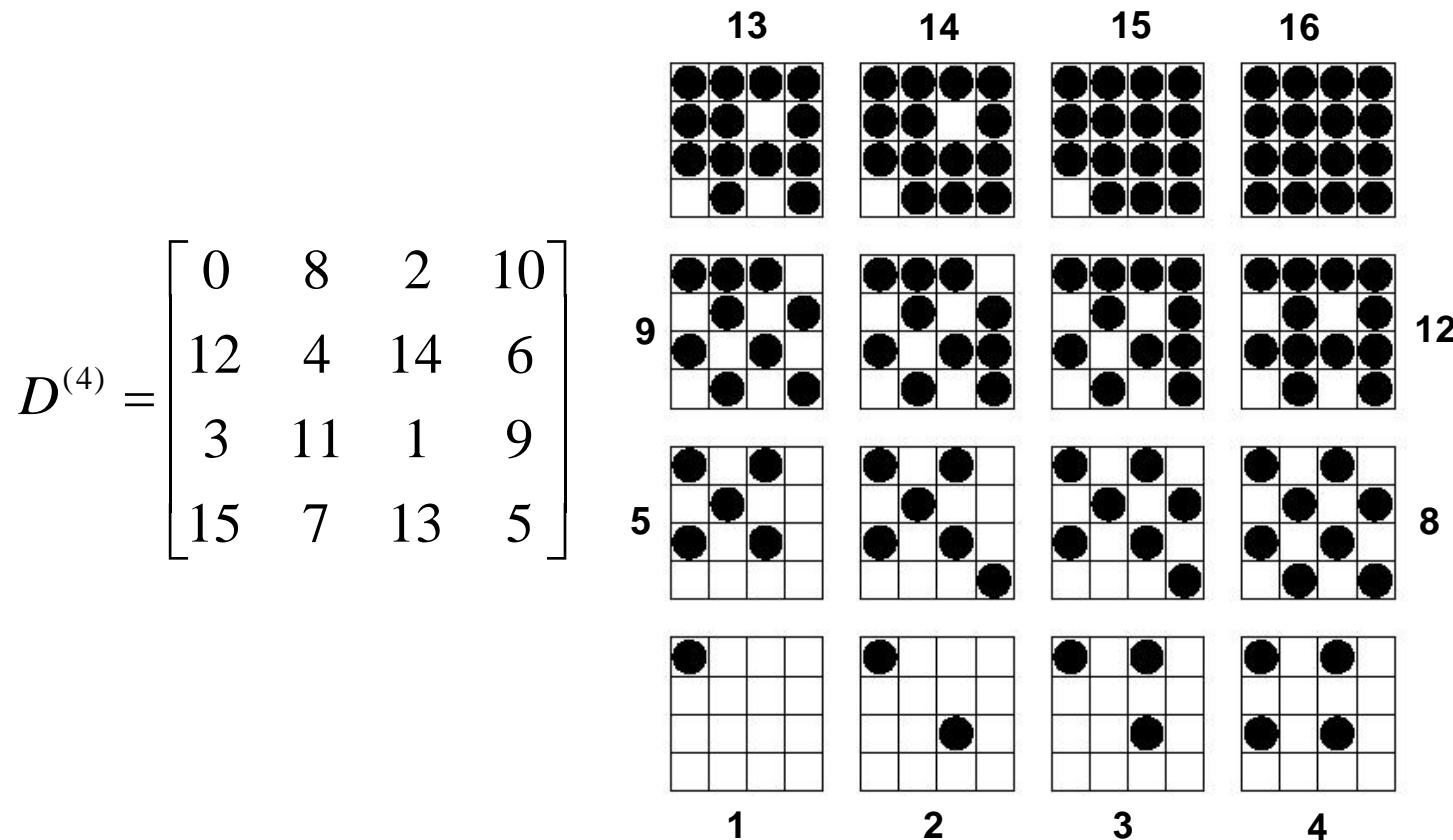
- To display a pixel of intensity I , we turn on all pixels whose associated dither matrix values are less than I .
- The recurrence relation given below generates larger dither matrices of dimension $n \times n$, where n is a power of 2.

$$D^{(n)} = \begin{bmatrix} 4D^{(n/2)} + D_{00}^{(2)}U^{(n/2)} & 4D^{(n/2)} + D_{01}^{(2)}U^{(n/2)} \\ 4D^{(n/2)} + D_{10}^{(2)}U^{(n/2)} & 4D^{(n/2)} + D_{11}^{(2)}U^{(n/2)} \end{bmatrix}$$

where $U^{(n)}$ is an $n \times n$ matrix of 1's.

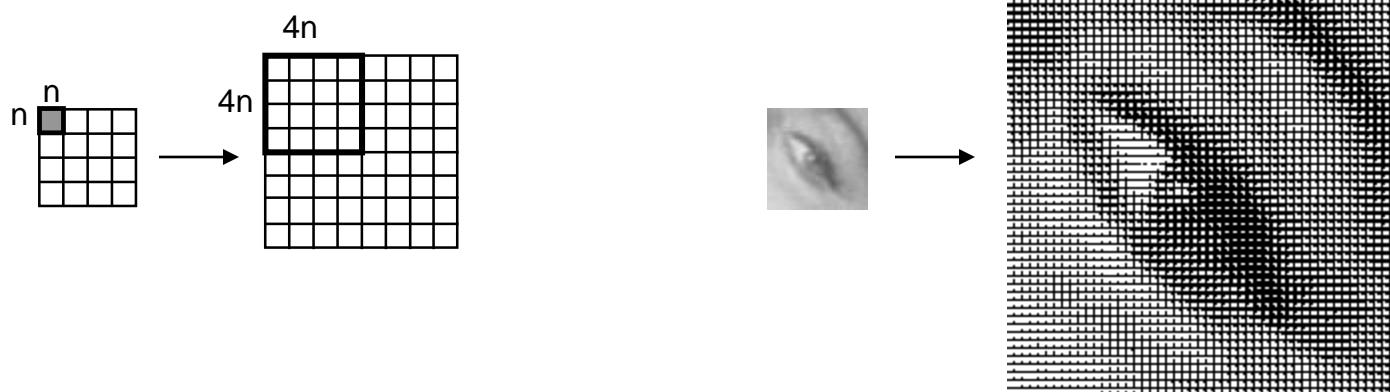
Dither Matrix (2)

- Example: a 4x4 dither matrix can be derived from the 2x2 matrix.



Patterning

- Let the output image be larger than the input image.
- Quantize the input image to $[0 \dots n^2]$ gray levels.
- Threshold each pixel against all entries in the dither matrix.
 - Each pixel forms a 4×4 block of black-and-white dots for a $D^{(4)}$ matrix.
 - An $n \times n$ input image becomes a $4n \times 4n$ output image.
- Multiple display pixels per input pixel.
- The dither matrix $D_{ij}^{(n)}$ is used as a spatially-varying threshold.
- Large input areas of constant value are displayed exactly as before.



Implementation

- Let the input and output images share the same size.
- First quantize the input image to $[0 \dots n^2]$ gray levels.
- Compare the dither matrix with the input image.

```
for(y=0; y<h; y++)          // visit all input rows
    for(x=0; x<w; x++) {
        i = x % n;           // dither matrix index
        j = y % n;           // dither matrix index

        // threshold pixel using dither value Dij(n)
        out[y*w+x] = (in[y*w+x] > Dij(n)) ? 255 : 0;
    }
```

Examples



8 bpp (256 levels)



1 bpp (D^3)



1 bpp (D^4)

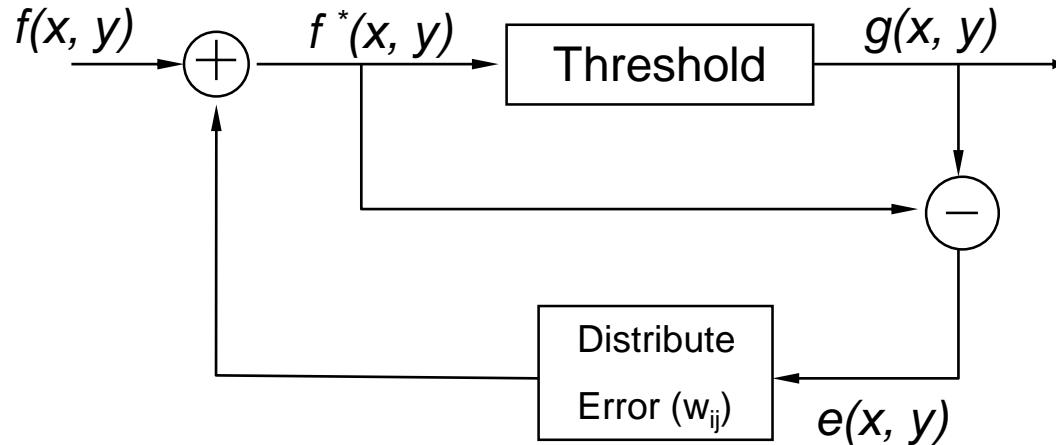


1 bpp (D^8)

Error Diffusion

- An error is made every time a grayvalue is assigned to be black or white at the output.
- Spread that error to its neighbors to compensate for over/undershoots in the output assignments
 - If input pixel 130 is mapped to white (255) then its excessive brightness (255-130) must be subtracted from neighbors to enforce a bias towards darker values to compensate for the excessive brightness.
- Like ordered dithering, error diffusion permits the output image to share the same dimension as the input image.

Floyd-Steinberg Algorithm



$$f^*(x, y) = f(x, y) + \sum_i \sum_j w_{ij} e(x-i, y-j) = \text{"corrected intensity value"}$$

$$g(x, y) = \begin{cases} 255 & \text{if } f^*(x, y) > MXGRAY/2 \\ 0 & \text{otherwise} \end{cases}$$

$$e(x, y) = f^*(x, y) - g(x, y)$$

$$\sum_i \sum_j w_{ij} = 1$$

Error Diffusion Weights

- Note that visual improvements are possible if left-to-right scanning among rows is replaced by serpentine scanning (zig-zag). That is, scan odd rows from left-to right, and scan even rows from right-to-left.
- Further improvements can be made by using larger neighborhoods.
- The sum of the weights should equal 1 to avoid emphasizing or suppressing the spread of errors.

$$\begin{array}{c} \longrightarrow \\ \rightarrow \quad x \quad 7/16 \\ 3/16 \quad 5/16 \quad 1/16 \end{array}$$

$$\begin{array}{ccccccccc} \longrightarrow & & \longrightarrow & x & 7/48 & 5/48 & \longrightarrow & x & 8/42 & 4/42 \\ & & \longrightarrow & 3/48 & 5/48 & 7/48 & 5/48 & 3/48 & 2/42 & 4/42 & 8/42 & 4/42 & 2/42 \\ & & & 1/48 & 3/48 & 5/48 & 3/48 & 1/48 & 1/42 & 2/42 & 4/42 & 2/42 & 1/42 \end{array}$$

Floyd-Steinberg

Jarvis-Judice-Ninke

Stucki

Examples (1)



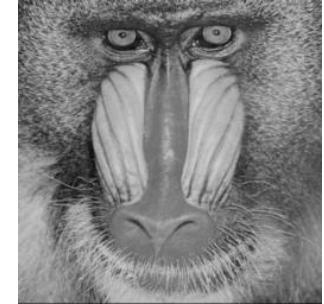
Floyd-Steinberg



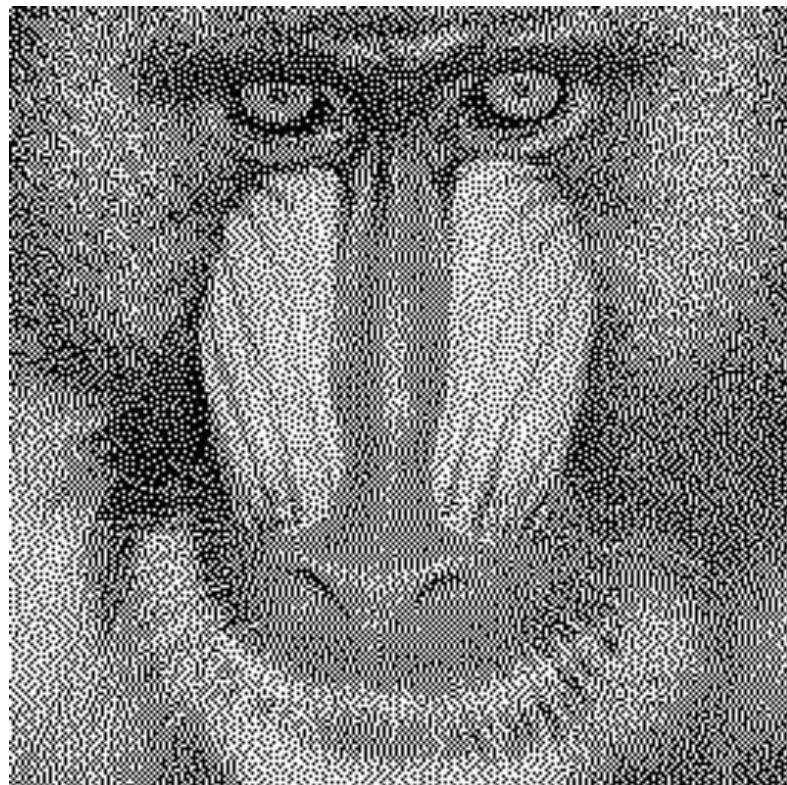
Jarvis-Judice-Ninke



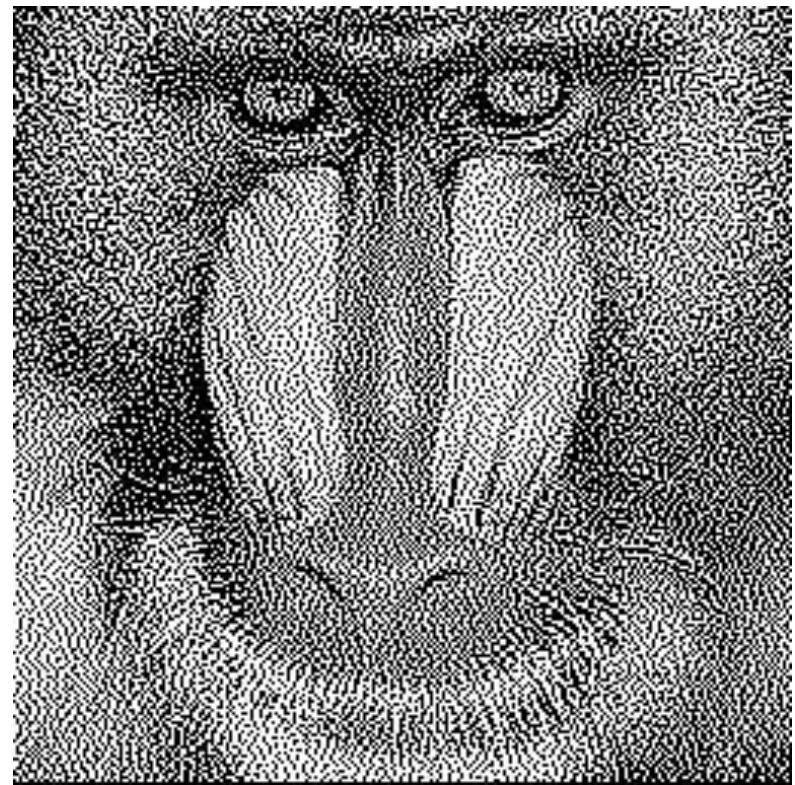
Examples (2)



Floyd-Steinberg



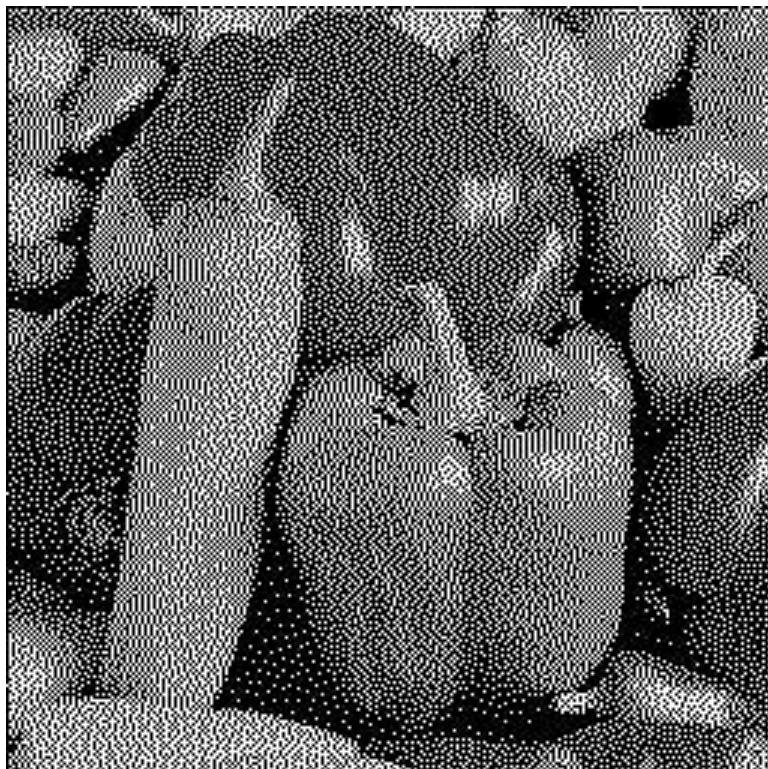
Jarvis-Judice-Ninke



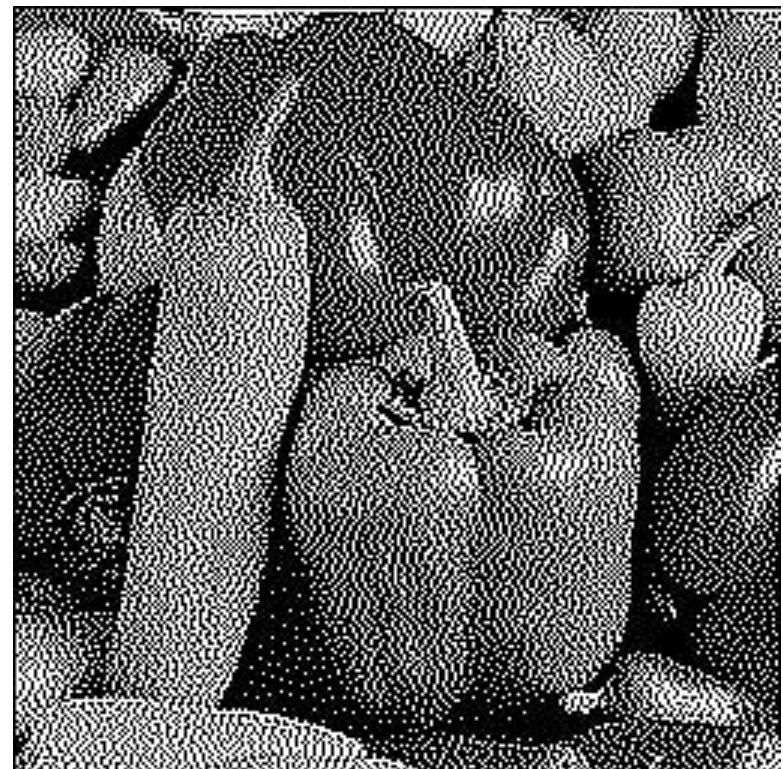
Examples (3)



Floyd-Steinberg



Jarvis-Judice-Ninke



Implementation

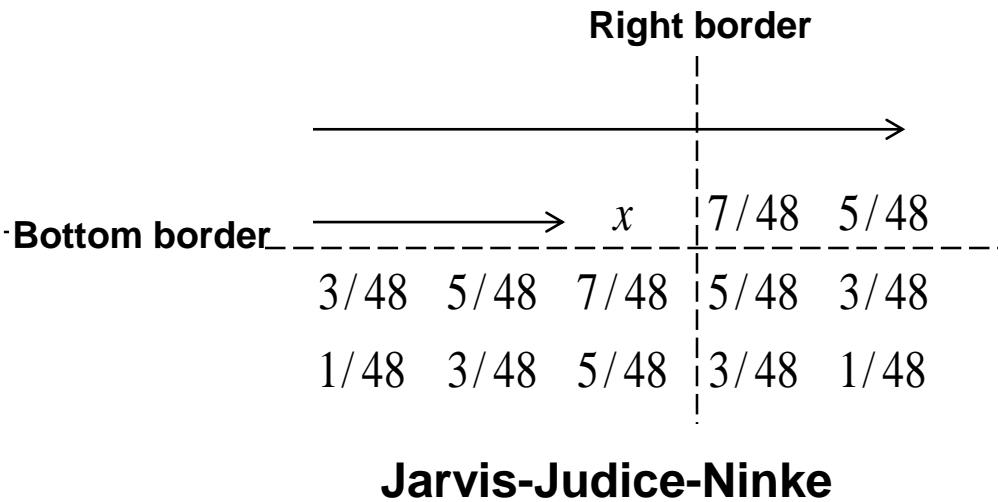
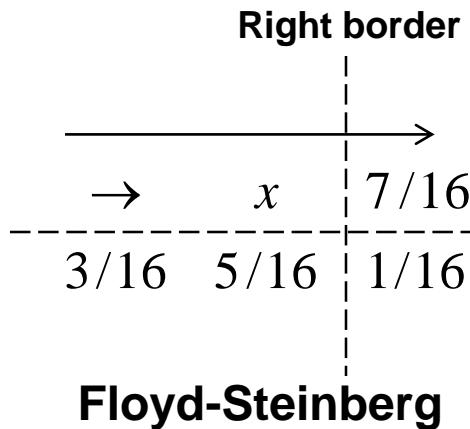
```
thr = MXGRAY /2;                                // init threshold value
for(y=0; y<h; y++) {                           // visit all input rows
    for(x=0; x<w; x++) {                         // visit all input cols
        *out = (*in < thr) ?                      // threshold
                BLACK : WHITE;                     // note: use LUT!

        e = *in - *out;                            // eval error
        in[ 1 ] +=(e*7/16.);                      // add error to E nbr
        in[w-1] +=(e*3/16.);                      // add error to SW nbr
        in[ w ] +=(e*5/16.);                      // add error to S nbr
        in[w+1] +=(e*1/16.);                      // add error to SE nbr

        in++;                                     // advance input ptr
        out++;                                    // advance output ptr
    }
}
```

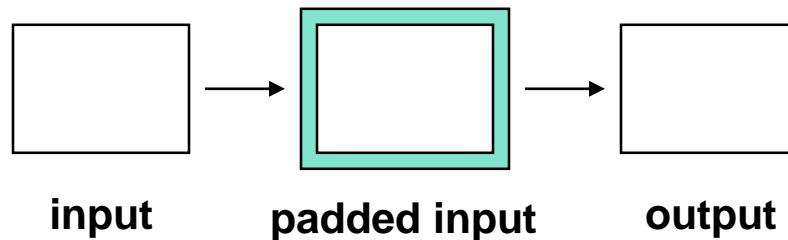
Comments

- Two potential problems complicate implementation:
 - errors can be deposited beyond image border ← True for all neighborhood ops
 - errors may force pixel grayvalues outside the [0,255] range



Solutions to Border Problem (1)

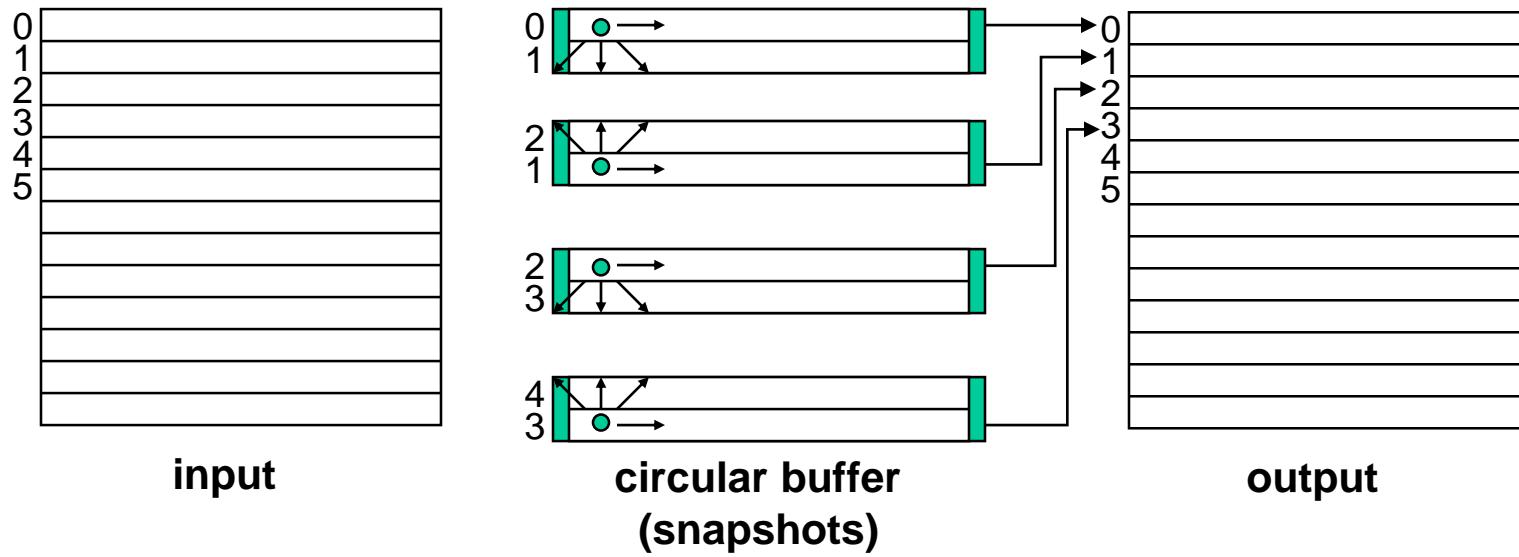
- Perform `if` statement prior to every error deposit
 - Drawback: inefficient / slow
- Limit excursions of sliding weights to lie no closer than 1 pixel from image boundary (2 pixels for J-J-N weights).
 - Drawback: output will be smaller than input
- Pad image with extra rows and columns so that limited excursions will yield smaller image that conforms with original input dimensions. Padding serves as placeholder.
 - Drawback: excessive memory needs for intermediate image



Solutions to Border Problem (2)

- Use of padding is further undermined by fact that 16-bit precision (**short**) is needed to accommodate pixel values outside [0, 255] range.
- A better solution is suggested by fact that only two rows are active while processing a single scanline in the Floyd-Steinberg algorithm (3 for JJN).
- Therefore, use a 2-row (or 3-row) circular buffer to handle the two (or three) current rows.
- The circular buffer will have the necessary padding and 16-bit precision.
- This significantly reduces memory requirements.

Circular Buffer



New Implementation

```
thr = MXGRAY /2;                      // init threshold value
copyRowToCircBuffer(0);                // copy row 0 to circular buffer
for(y=0; y<h; y++) {                  // visit all input rows
    copyRowToCircBuffer(y+1);          // copy next row to circ buffer
    in1 = buf[ y %2] + 1;             // circ buffer ptr; skip over pad
    in2 = buf[(y+1)%2] + 1;           // circ buffer ptr; skip over pad
    for(x=0; x<w; x++) {            // visit all input cols
        *out = (*in1 < thr)? BLACK : WHITE; // threshold

        e = *in1 - *out;              // eval error
        in1[ 1] +=(e*7/16.);          // add error to E nbr
        in2[-1] +=(e*3/16.);          // add error to SW nbr
        in2[ 0] +=(e*5/16.);          // add error to S nbr
        in2[ 1] +=(e*1/16.);          // add error to SE nbr

        in1++; in2++;                // advance circ buffer ptrs
        out++;                        // advance output ptr
    }
}
```

Neighborhood Operations

Prof. George Wolberg
Dept. of Computer Science
City College of New York

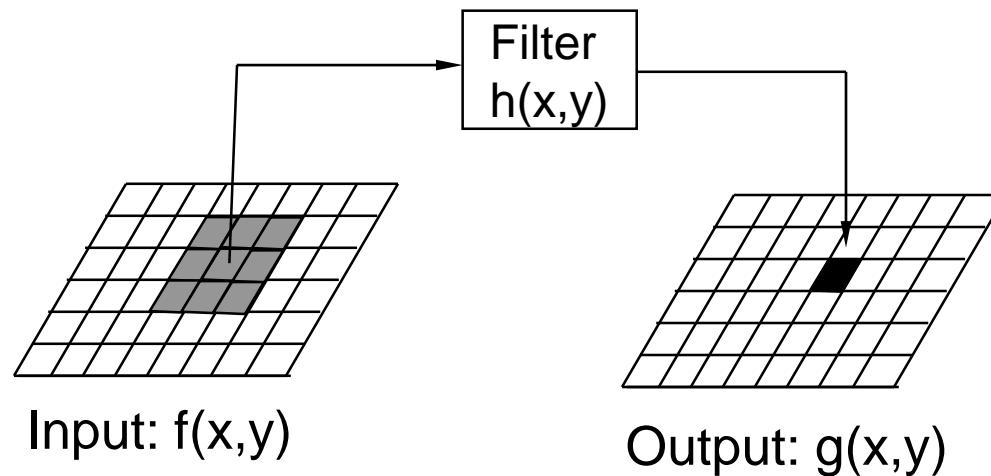
Objectives

- This lecture describes various neighborhood operations:
 - Blurring
 - Edge detection
 - Image sharpening
 - Convolution

Neighborhood Operations

- Output pixels are a function of several input pixels.
- $h(x,y)$ is defined to weigh the contributions of each input pixel to a particular output pixel.
- $g(x,y) = T[f(x,y); h(x,y)]$

$$g(x,y) = T[f(x,y); h(x,y)] = f(x,y) * h(x,y)$$



Spatial Filtering

- $h(x,y)$ is known as a *filter kernel*, *filter mask*, or *window*.
- The values in a filter kernel are coefficients.
- Kernels are usually of odd size: 3x3, 5x5, 7x7
- This permits them to be properly centered on a pixel
 - Consider a horizontal cross-section of the kernel.
 - Size of cross-section is odd since there are $2n+1$ coefficients: n neighbors to the left + n neighbors to the right + center pixel

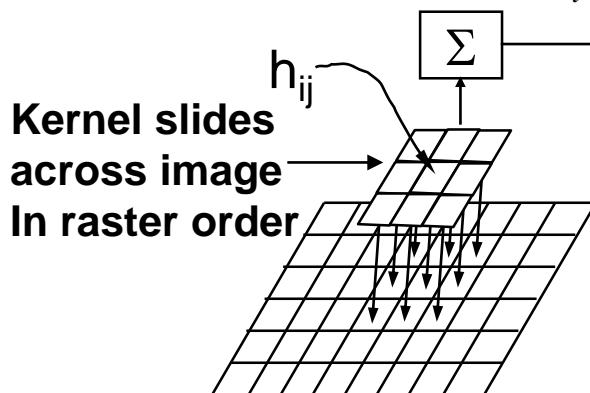
h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

Spatial Filtering Process

- Slide filter kernel from pixel to pixel across an image.
- Use raster order: left-to-right from the top to the bottom.
- Let pixels have grayvalues f_i .
- The response of the filter at each (x,y) point is:

$$R = h_1 f_1 + h_2 f_2 + \dots + h_{mn} f_{mn} \quad \leftarrow \text{1D indexing}$$

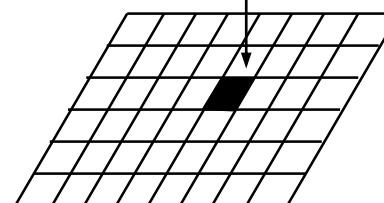
$$= \sum_{i=i}^{mn} h_i f_i$$



2D indexing

Window centered at (4,3)

$$\begin{aligned} g_{43} &= h_1 f_{32} + h_2 f_{33} + h_3 f_{34} \\ &+ h_4 f_{42} + h_5 f_{43} + h_6 f_{44} \\ &+ h_7 f_{52} + h_8 f_{53} + h_9 f_{54} \end{aligned}$$



Linear Filtering

- Let $f(x,y)$ be an image of size $M \times N$.
- Let $h(i,j)$ be a filter kernel of size $m \times n$.
- Linear filtering is given by the expression:

$$g(x, y) = \sum_{i=-s}^s \sum_{j=-t}^t h(i, j) f(x + i, y + j)$$

where $s = (m-1)/2$ and $t = (n-1)/2$

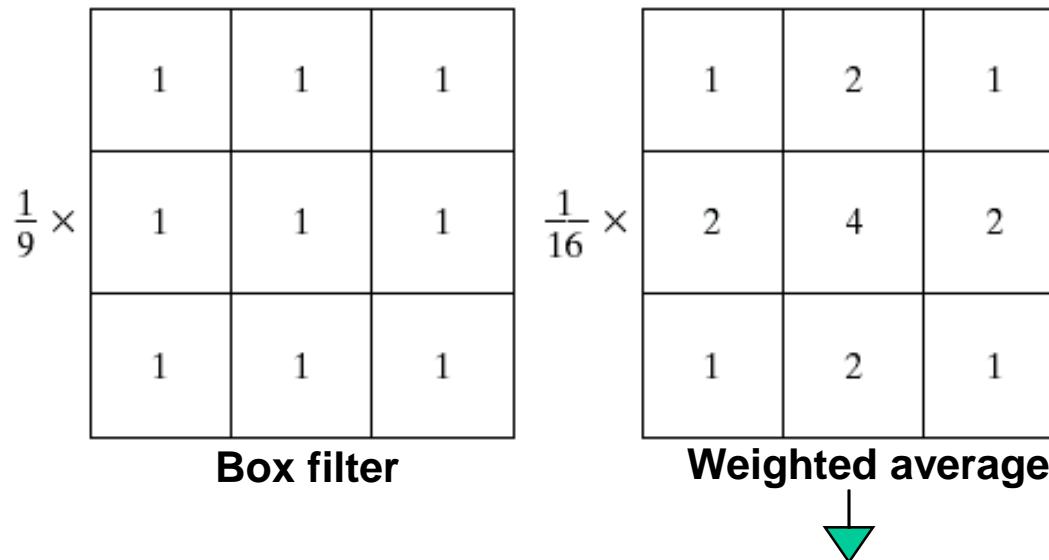
- For a complete filtered image this equation must be applied for $x = 0, 1, 2, \dots, M-1$ and $y = 0, 1, 2, \dots, N-1$.

Spatial Averaging

- Used for blurring and for noise reduction
- Blurring is used in preprocessing steps, such as
 - removal of small details from an image prior to object extraction
 - bridging of small gaps in lines or curves
- Output is average of neighborhood pixels.
- This reduces the “sharp” transitions in gray levels.
- Sharp transitions include:
 - random noise in the image
 - edges of objects in the image
- Smoothing reduces noise (good) and blurs edges (bad)

3x3 Smoothing Filters

- The constant multiplier in front of each kernel is equal to the sum of the values of its coefficients.
- This is required to compute an average.



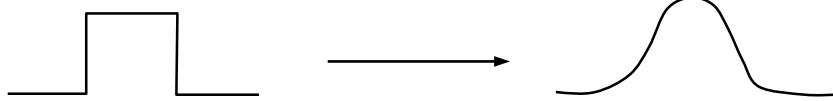
The center is the most important and other pixels are inversely weighted as a function of their distance from the center of the mask. This reduces blurring in the smoothing process.

Unweighted/Weighted Averaging

- Unweighted averaging (smoothing filter):

$$g(x, y) = \frac{1}{m} \sum_{i,j} f(i, j)$$

- Weighted averaging:



Original image



7x7 unweighted averaging



7x7 Gaussian filter



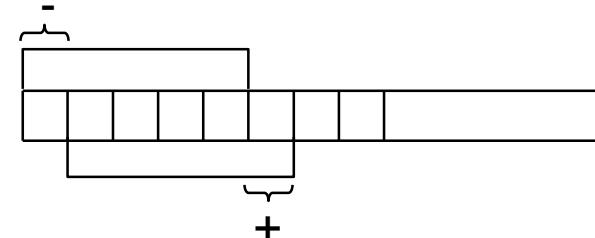
Unweighted Averaging

- Unweighted averaging over a 5-pixel neighborhood along a horizontal scanline can be done with the following statement:

```
for(x=2; x<w-2; x++)  
    out[x]=(in[x-2]+in[x-1]+in[x]+in[x+1]+in[x+2])/5;
```

- Each output pixel requires 5 pixel accesses, 4 adds, and 1 division. A simpler version (for unweighted averaging only) is:

```
sum=in[0]+in[1]+in[2]+in[3]+in[4];  
  
for(x=2; x<w-2; x++) {  
    out[x] = sum/5;  
    sum+=(in[x+3] - in[x-2]);  
}
```



Limited excursions reduce size of output

Image Averaging

- Consider a noisy image $g(x,y)$ formed by the addition of noise $\eta(x,y)$ to an original image $f(x,y)$:

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

- If the noise has zero mean and is uncorrelated then we can compute the image formed by averaging K different noisy images:

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y)$$

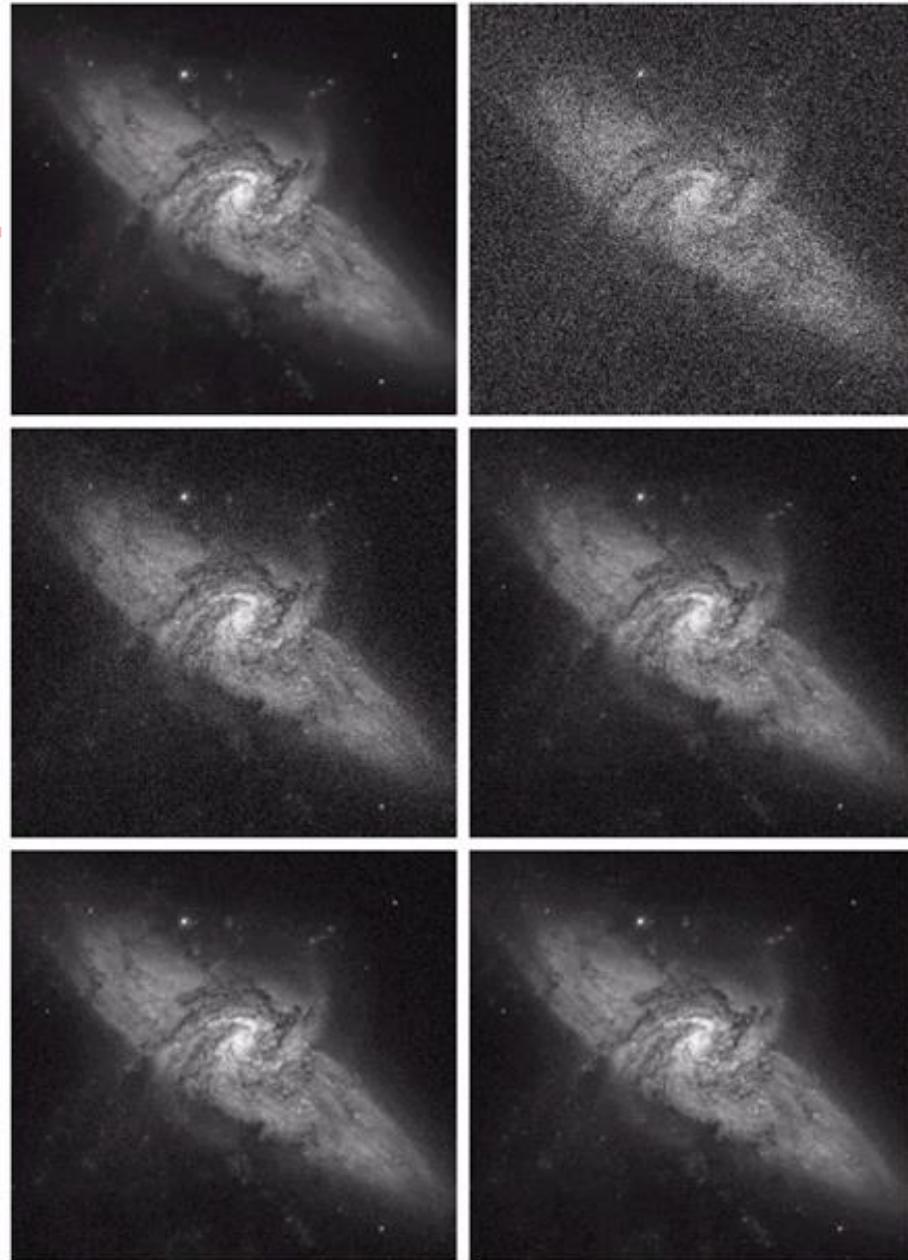
- The variance of the averaged image diminishes:

$$\sigma_{\bar{g}(x,y)}^2 = \frac{1}{K} \sigma_{\eta(x,y)}^2$$

- Thus, as K increases the variability (noise) of the pixel at each location (x,y) decreases assuming that the images are all registered (aligned).

Noise Reduction (1)

- Astronomy is an important application of image averaging.
- Low light levels cause sensor noise to render single images virtually useless for analysis.

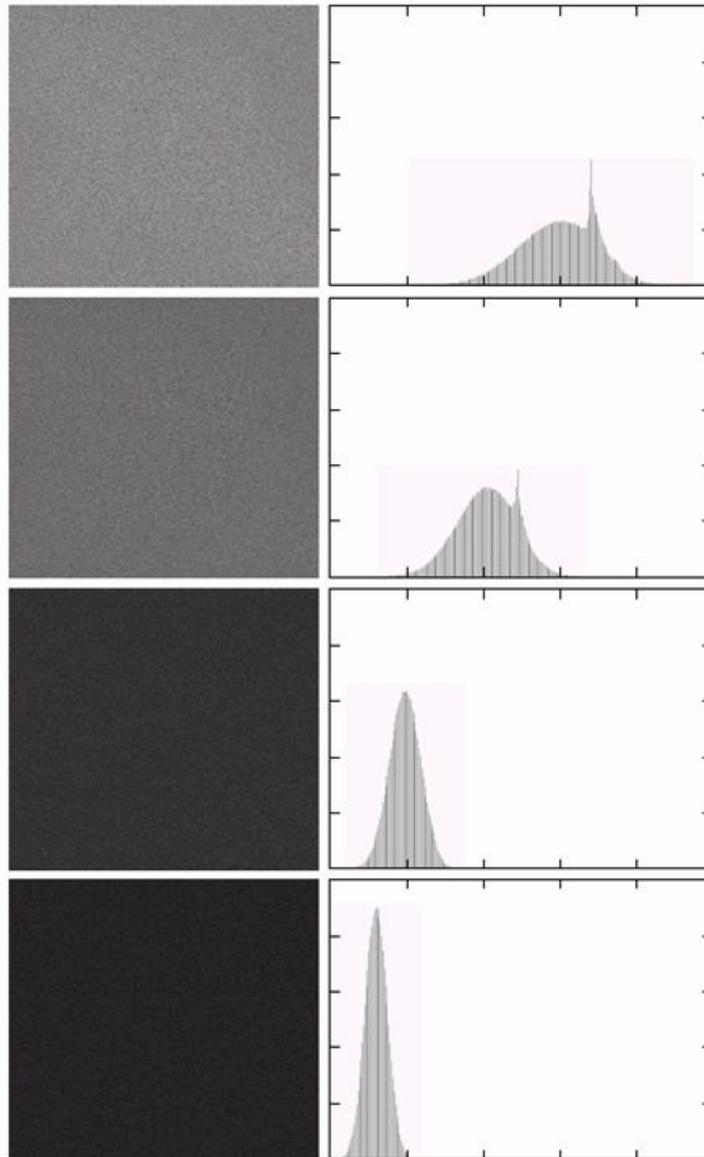


a b
c d
e f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64$, and 128 noisy images. (Original image courtesy of NASA.)

Noise Reduction (2)

- Difference images and their histograms yield better appreciation of noise reduction.
- Notice that the mean and standard deviation of the difference images decrease as K increases.



a b

FIGURE 3.31
(a) From top to bottom:
Difference images between
Fig. 3.30(a) and the four images in
Figs. 3.30(c) through (f),
respectively.
(b) Corresponding histograms

General Form: Smoothing Mask

- Filter of size $m \times n$ (where m and n are odd)

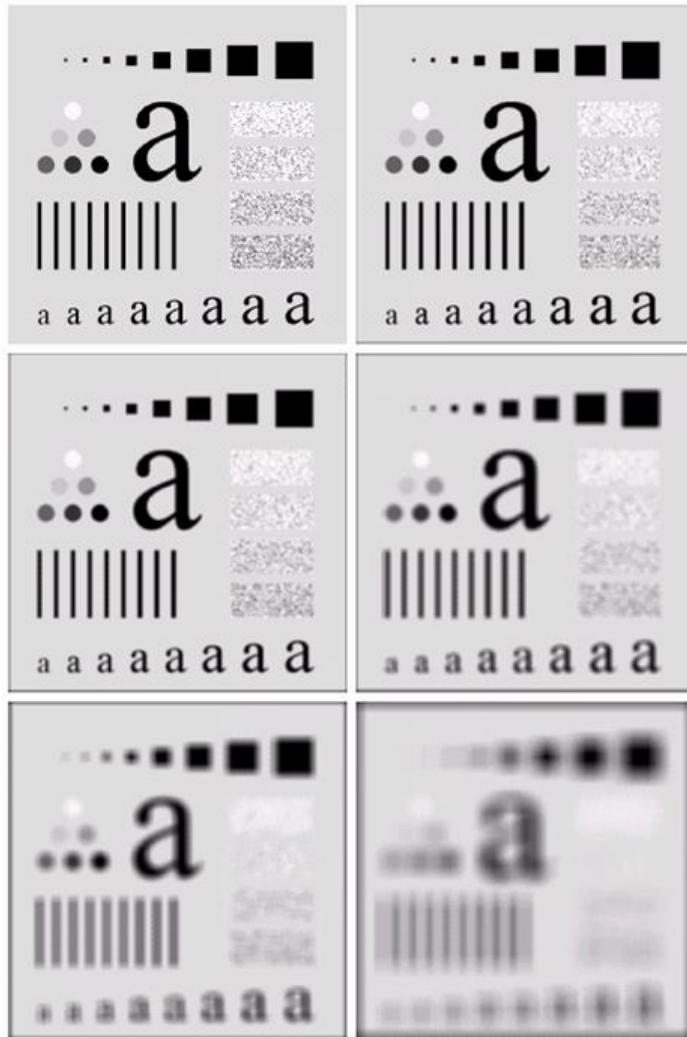
$$g(x, y) = \frac{\sum_{i=-s}^s \sum_{j=-t}^t h(i, j) f(x + i, y + j)}{\sum_{i=-s}^s \sum_{j=-t}^t h(i, j)}$$


summation of all coefficients of the mask

Note that $s = (m-1)/2$ and $t = (n-1)/2$

a	b
c	d
e	f

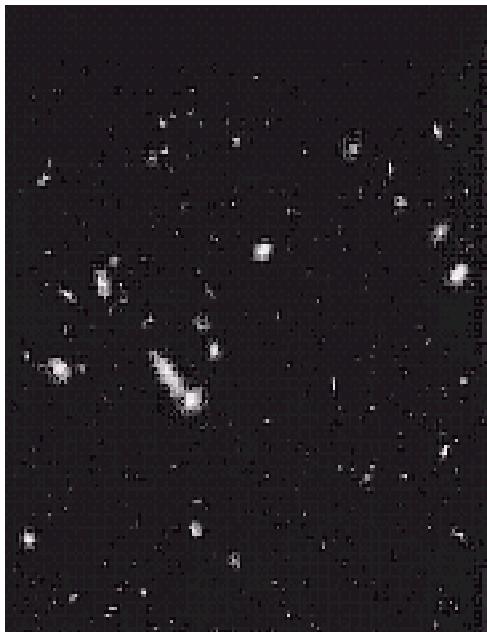
Example



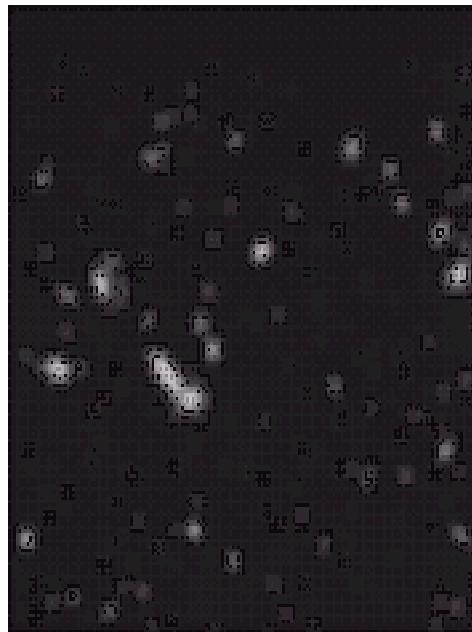
- a) original image 500x500 pixel
- b) - f) results of smoothing with square averaging filter of size $n = 3, 5, 9, 15$ and 35 , respectively.
- Note:
 - big mask is used to eliminate small objects from an image.
 - the size of the mask establishes the relative size of the objects that will be blended with the background.

Example

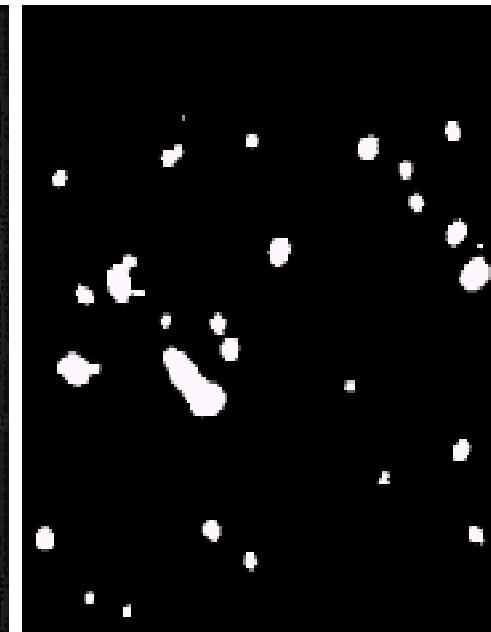
- Blur to get gross representation of objects.
- Intensity of smaller objects blend with background.
- Larger objects become blob-like and easy to detect.



original image



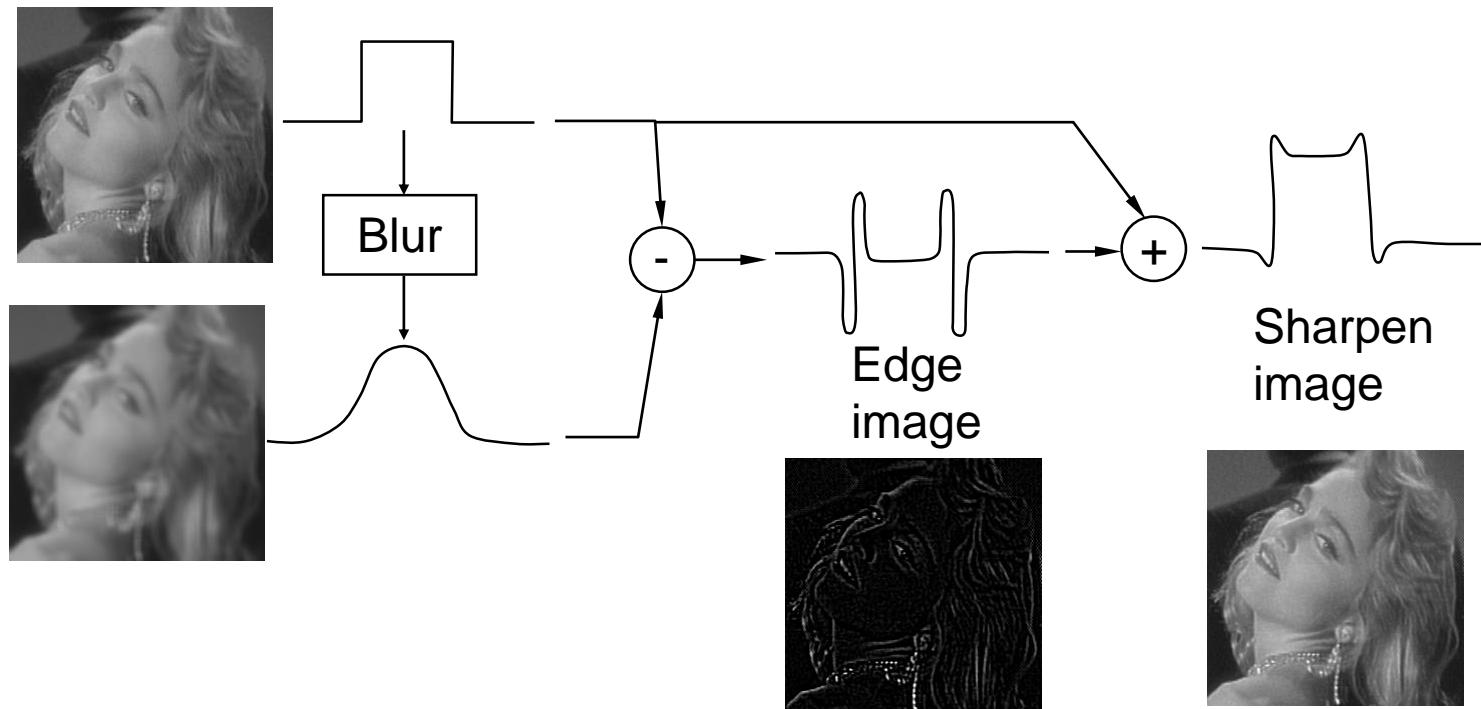
**result after smoothing
with 15x15 filter**



result of thresholding

Unsharp Masking

- Smoothing affects transition regions where grayvalues vary.
- Subtraction isolates these edge regions.
- Adding edges back onto image causes edges to appear more pronounced, giving the effect of image sharpening.



Order-Statistics Filters

- Nonlinear filters whose response is based on ordering (ranking) the pixels contained in the filter support.
- Replace value of the center pixel with value determined by ranking result.
- Order statistic filters applied to $n \times n$ neighborhoods:
 - median filter: $R = \text{median}\{z_k | k = 1, 2, \dots, n^2\}$
 - max filter: $R = \max\{z_k | k = 1, 2, \dots, n^2\}$
 - min filter: $R = \min\{z_k | k = 1, 2, \dots, n^2\}$

Median Filter

- Sort all neighborhood pixels in increasing order.
- Replace neighborhood center with the median.
- The window shape does not need to be a square.
- Special shapes can preserve line structures.
- Useful in eliminating intensity spikes: salt & pepper noise.

10	20	20
20	200	15
25	20	25

(10,15,20,20,20,20,25,25,200)

Median = 20

Replace 200 with 20

Median Filter Properties

- Excellent noise reduction
- Forces noisy (distinct) pixels to conform to their neighbors.
- Clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $n^2/2$ (one-half the filter area), are eliminated by an $n \times n$ median filter.
- k-nearest neighbor is a variation that blends median filtering with blurring:
 - Set output to average of k nearest entries around median

10	18	19
20	200	15
25	20	25

(10,15,18,19,20,20,25,25,200)

k=1: replace 200 with $(19+20+20)/3$

k=2: replace 200 with $(18+19+20+20+25)/5$

k=3: replace 200 with $(15+18+19+20+20+25+25)/7$

k=4: replace 200 with $(10+15+18+19+20+20+25+25+200)/9$

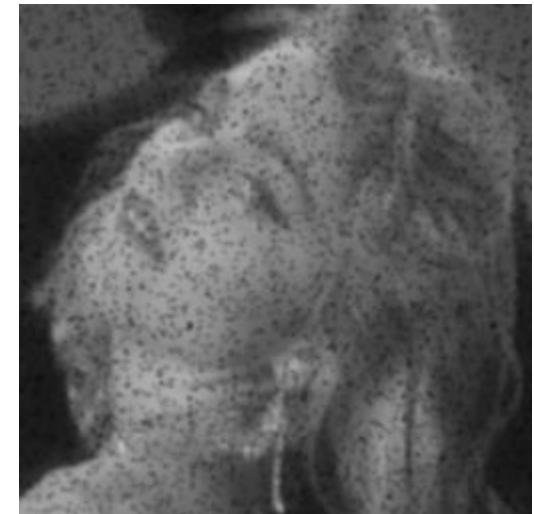
Examples (1)



Additive salt & pepper noise

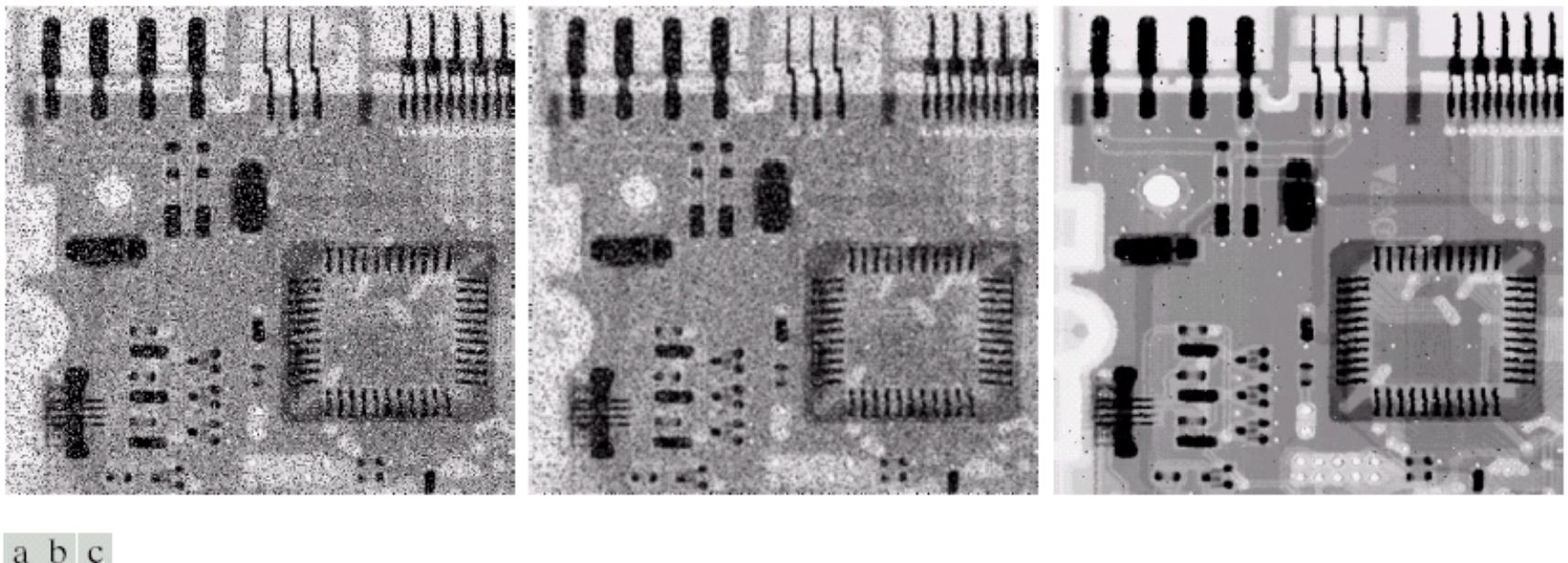


Median filter output



Blurring output

Examples (2)



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Derivative Operators

- The response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied.
- Image differentiation
 - enhances edges and other discontinuities (noise)
 - deemphasizes area with slowly varying graylevel values.
- Derivatives of a digital function are approximated by differences.

First-Order Derivative

- Must be zero in areas of constant grayvalues.
- Must be nonzero at the onset of a grayvalue step or ramp.
- Must be nonzero along ramps.

$$\frac{\partial f(x)}{\partial x} = f(x+1) - f(x)$$

Second-Order Derivative

- Must be zero in areas of constant grayvalues.
- Must be nonzero at the onset of a grayvalue step or ramp.
- Must be zero along ramps of constant slope.

$$\begin{aligned}\frac{\partial^2 f(x)}{\partial x^2} &= \partial f(x) - \partial f(x-1) \\ &= f(x+1) + f(x-1) - 2f(x)\end{aligned}$$

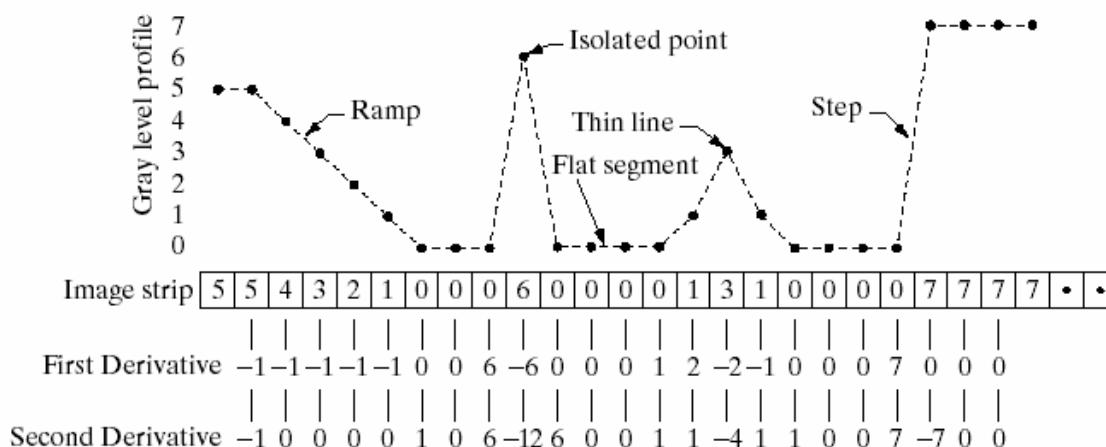
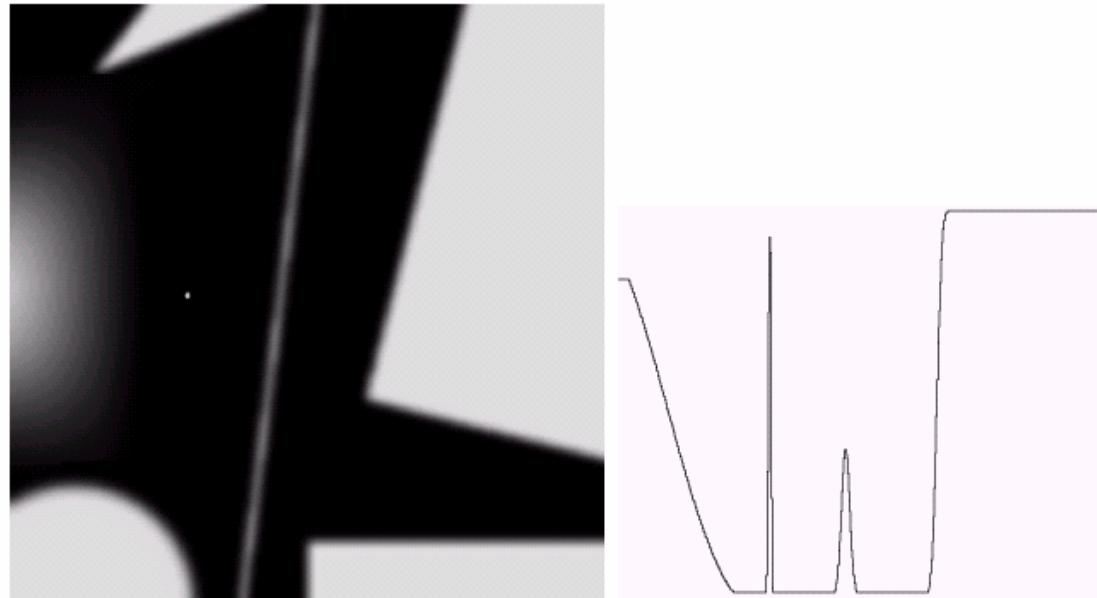
Example

a
b
c

FIGURE 3.38

(a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point.

(c) Simplified profile (the points are joined by dashed lines to simplify interpretation).



Comparisons

- 1st-order derivatives:
 - produce thicker edges
 - strong response to graylevel steps
- 2nd-order derivatives:
 - strong response to fine detail (thin lines, isolated points)
 - double response at step changes in graylevel

Laplacian Operator

- Simplest isotropic derivative operator
- Response independent of direction of the discontinuities.
- Rotation-invariant: rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.
- Since derivatives of any order are linear operations, the Laplacian is a linear operator.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Discrete Form of Laplacian

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

where

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\begin{aligned}\nabla^2 f = & [f(x+1, y) + f(x-1, y) \\ & + f(x, y+1) + f(x, y-1) - 4f(x, y)]\end{aligned}$$

Laplacian Mask

**Isotropic result
for rotations in
increments of 90°**

0	1	0
1	-4	1
0	1	0

**Isotropic result
for rotations in
increments of 45°**

1	1	1
1	-8	1
1	1	1

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

-1	-1	-1
-1	8	-1
-1	-1	-1

a b
c d

FIGURE 3.39

- (a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4).
(b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

Another Derivation

1/9*

1	1	1
1	1	1
1	1	1

Unweighted Average Smoothing Filter

0	0	0
0	1	0
0	0	0

Retain Original

1/9*

-1	-1	-1
-1	8	-1
-1	-1	-1

Original – Average (negative of Laplacian Operator)

In constant areas: 0



Summation of coefficients in masks equals 0.

Near edges: high values

Effect of Laplacian Operator

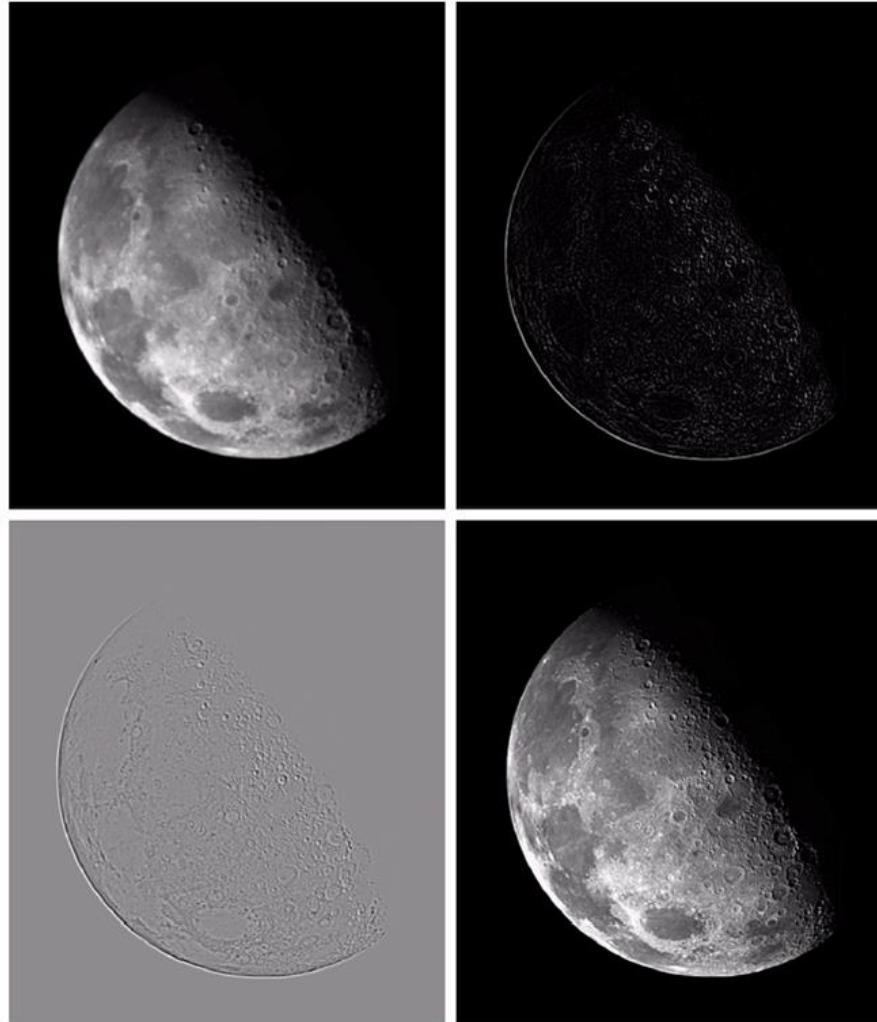
- Since the Laplacian is a derivative operator
 - it highlights graylevel discontinuities in an image
 - it deemphasizes regions with slowly varying gray levels
- The Laplacian tends to produce images that have
 - grayish edge lines and other discontinuities all superimposed on a dark featureless background

Example

a b
c d

FIGURE 3.40

- (a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)



-1	-1	-1
-1	8	-1
-1	-1	-1

Simplification

- Addition of image with Laplacian can be combined into one operator:

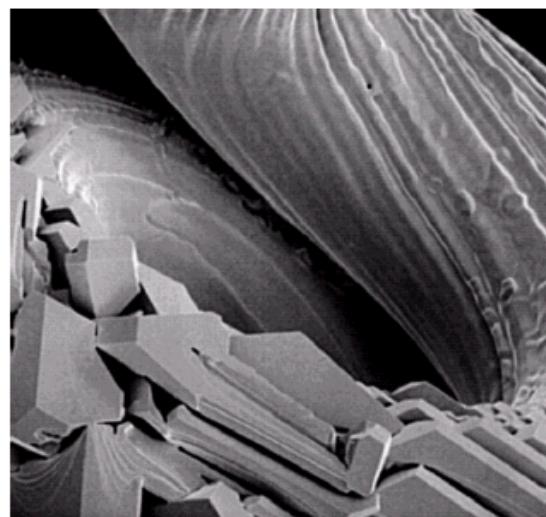
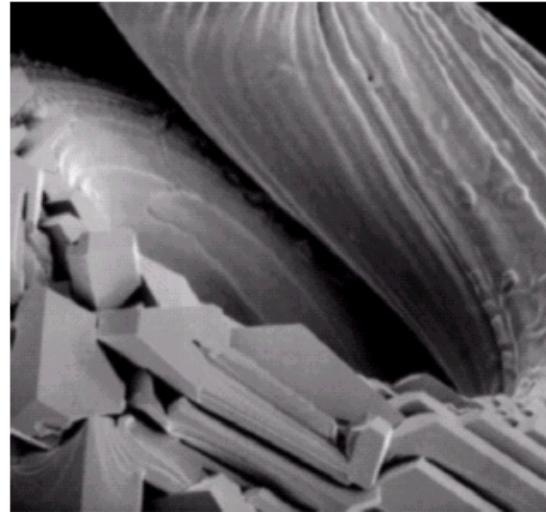
$$\begin{aligned}g(x, y) &= f(x, y) - [f(x+1, y) + f(x-1, y) \\&\quad + f(x, y+1) + f(x, y-1) - 4f(x, y)] \\&= 5f(x, y) - [f(x+1, y) + f(x-1, y) \\&\quad + f(x, y+1) + f(x, y-1)]\end{aligned}$$

$$\begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Example

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1



a b c
d e

FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Gradient Operator (1)

- The gradient is a vector of directional derivatives.

$$\nabla \mathbf{f} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- Although not strictly correct, the magnitude of the gradient vector is referred to as the gradient.
- First derivatives are implemented using this magnitude

$$\begin{aligned}\nabla f &= \text{mag}(\nabla \mathbf{f}) \\ &= [f_x^2 + f_y^2]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}\end{aligned}$$

approximation:

$$\nabla f \approx |f_x| + |f_y|$$

Gradient Operator (2)

- The components of the gradient vector are linear operators, but the magnitude is not (square,square root).
- The partial derivatives are not rotation invariant (isotropic), but the magnitude is.
- The Laplacian operator yields a scalar: a single number indicating edge strength at point.
- The gradient is actually a vector from which we can compute edge magnitude and direction.

$$f_{mag}(i, j) = \sqrt{f_x^2 + f_y^2} \quad \text{or} \quad f_{mag}(i, j) = |f_x| + |f_y|$$

$$f_{angle}(i, j) = \tan^{-1} \frac{f_y}{f_x}$$

where

$$\boxed{\begin{aligned} f_x(i, j) &= f(i+1, j) - f(i-1, j) \\ f_y(i, j) &= f(i, j+1) - f(i, j-1) \end{aligned}}$$

Summary (1)

Continuous

$$f(x)$$

$$f'(x)$$

$$f''(x) = \nabla^2 f(x)$$

Digital

$$v(i)$$

$$v'(i) = v(i) - v(i-1)$$

$$v''(i) = v'(i) - v'(i-1)$$

$$= [v(i) - v(i-1)] - [v(i-1) - v(i-2)]$$

$$= v(i-2) - 2v(i-1) + v(i)$$

$$= (1 \quad -2 \quad 1)(v(i-2) \quad v(i-1) \quad v(i))$$

$$= (1 \quad -2 \quad 1)(v(i-1) \quad v(i) \quad v(i+1))$$

	-1	
-1	4	-1
	-1	

-1	-1	-1
-1	8	-1
-1	-1	-1

The Laplacian is a scalar, giving only the magnitude about the change in pixel values at a point. The gradient gives both magnitude and direction.

Summary (2)

One dimensional:

$$\text{mask}_x = [-1 \ 0 \ 1]$$

$$\text{mask}_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$f_{mag}(i, j) = \sqrt{f_x^2 + f_y^2} \quad \text{or} \quad f_{mag}(i, j) = |f_x| + |f_y|$$

$$f_{angle}(i, j) = \tan^{-1} \frac{f_y}{f_x}$$

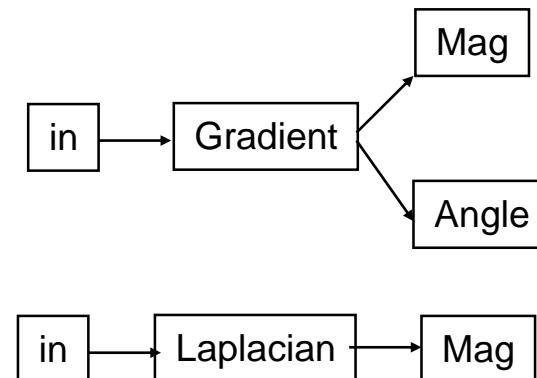
Two dimensional:

Sobel Operator :

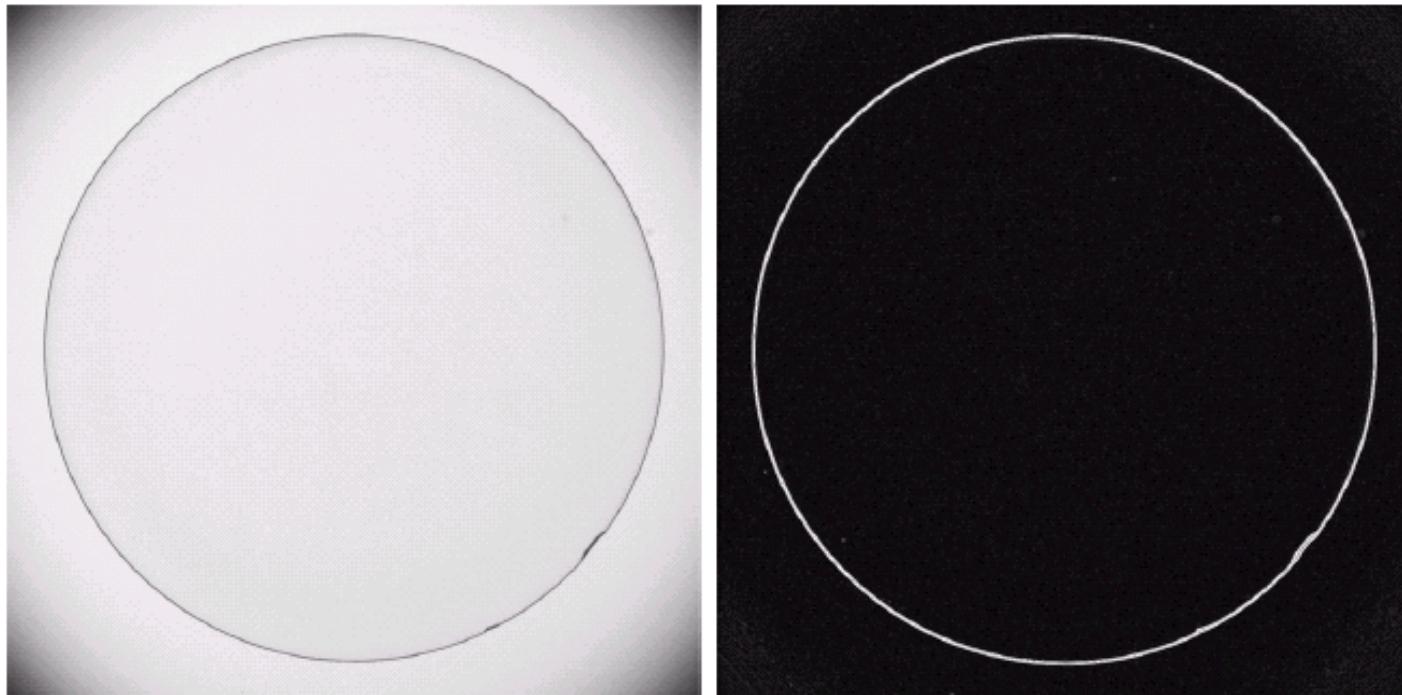
$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{mask}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Prewitt Operator :

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{mask}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



Example (1)

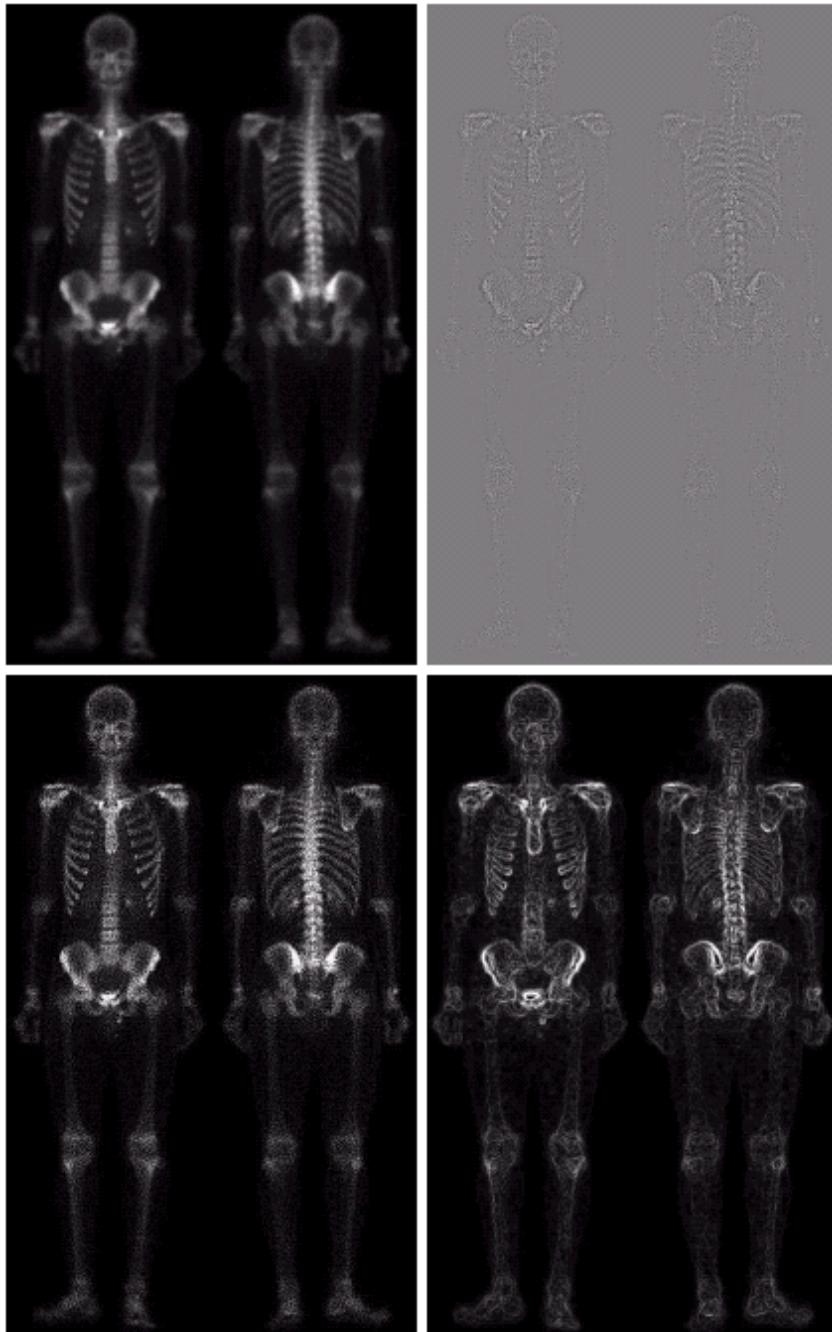


a b

FIGURE 3.45
Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).
(b) Sobel gradient.
(Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

Example (2)

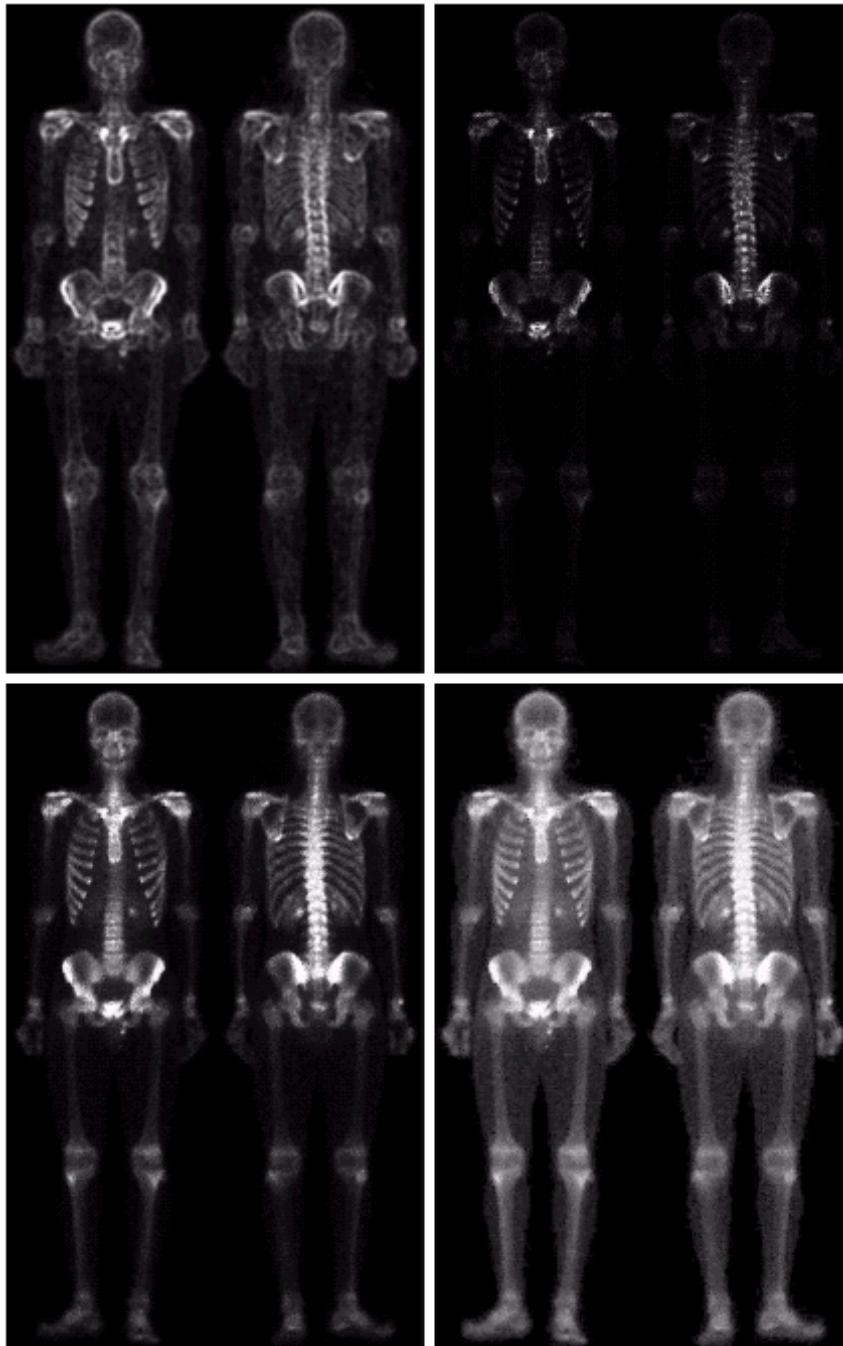
- **Goal:** sharpen image and bring out more skeletal detail.
- **Problem:** narrow dynamic range and high noise content makes the image difficult to enhance.
- **Solution:**
 1. Apply Laplacian operator to highlight fine detail
 2. Apply gradient operator to enhance prominent edges
 3. Apply graylevel transformation to increase dynamic range



a b
c d

FIGURE 3.46

- (a) Image of whole body bone scan.
(b) Laplacian of (a).
(c) Sharpened image obtained by adding (a) and (b).
(d) Sobel of (a).



e f
g h

FIGURE 3.46

(Continued)

(e) Sobel image smoothed with a 5×5 averaging filter. (f) Mask image formed by the product of (c) and (e).

(g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Filtering Theory

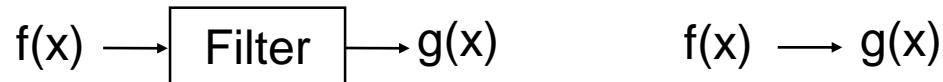
Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- This lecture reviews filtering theory.
 - Linearity and spatial-invariance (LSI)
 - Impulse response
 - Sifting integral
 - Convolution

Definitions

- We use two criteria for filters: linearity and spatial-invariance



Linear :

$$\alpha f(x) \rightarrow \alpha g(x) \quad \text{output is proportional to input}$$

$$f_1(x) + f_2(x) \rightarrow g_1(x) + g_2(x) \quad \text{superposition}$$

or simply,

$$\alpha f_1(x) + \beta f_2(x) \rightarrow \alpha g_1(x) + \beta g_2(x)$$

Space - invariant(shift - invariant) :

$$f(x - \lambda) \rightarrow g(x - \lambda)$$

LSI Filter

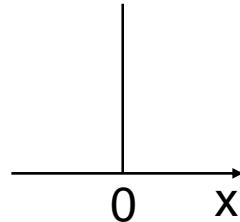
- Physically realizable filters (lenses) are rarely LSI filters.
 - Optical systems impose a limit in their maximum response.
 - Power can't be negative, imposing a limit on the minimum response.
 - Lens aberrations and finite image area prevent LSI property.
- Nevertheless, close enough to approximate as LSI.

Impulse Response

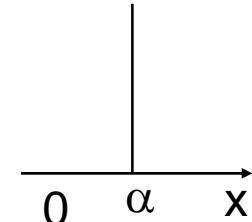


$$\delta(x) = \begin{cases} \lim_{\varepsilon \rightarrow 0} \int_{-\varepsilon}^{\varepsilon} f(x') dx' = 1, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

$\delta(x)$ Spike at $x=0$



$\delta(x - \alpha)$



$\delta(x)$ is the impulse function, or Dirac delta function which is a continuous function.

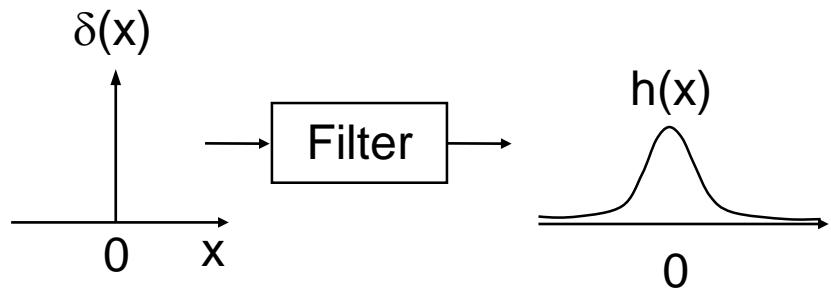
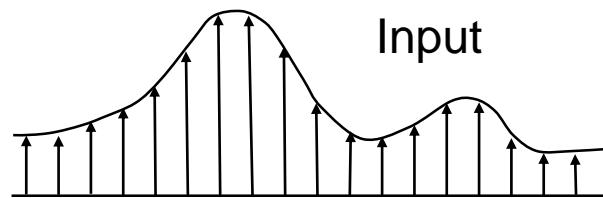
$$f(x_0) = \int_{-\infty}^{\infty} f(\lambda) \delta(x_0 - \lambda) d\lambda \quad \delta(x_0 - \lambda) \text{ samples } f(x) \text{ at } x_0$$

$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & x \neq 0 \end{cases} \quad \textbf{Kronecker delta function (discrete case: integer values of } x \text{)}$$

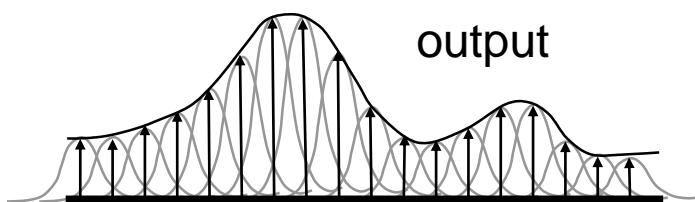
$$f(x) = \int_{-\infty}^{\infty} f(\lambda) \delta(x - \lambda) d\lambda \quad \textbf{Sifting integral}$$

Convolution

- Any input signal can be represented by an infinite sum of shifted and scaled impulses:



Convolution:



Blurred version of input

$$\text{continuous : } g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\lambda)h(x - \lambda)d\lambda$$

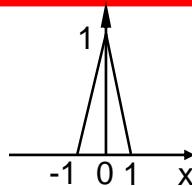
Discrete Case

- Output function is a scaled shifted version of impulse response.
 $\otimes, *$: convolution operator
 $h(x)$: convolution kernel, filter kernel
- If $h(x) = \delta(x)$ then we have an ideal filter: output = input.
- Usually $h(x)$ extends over several neighbors.
- Discrete convolution:

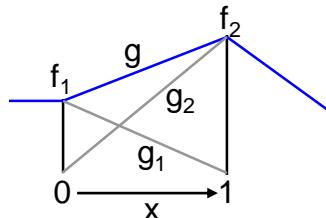
$$g(x) = f(x) \otimes h(x) = \sum_{-\infty}^{\infty} f(\lambda)h(x - \lambda)$$

Example: Triangle Filter

$$h(x) = \begin{cases} 1 - |x| & 0 \leq |x| < 1 \\ 0 & 1 \leq |x| \end{cases}$$



Input samples (impulses)



Addition of two adjacent impulse responses

Impulse responses (additive)

$$g_1 = f_1(1-x) \quad g_2 = f_2 x \quad 0 \leq x \leq 1$$
$$g = g_1 + g_2 = (f_2 - f_1)x + f_1$$

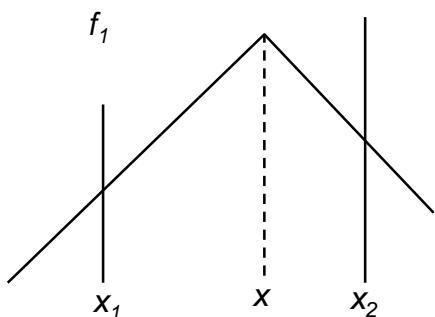
Straight line: $y=mx+b$

- Linearity rule: scale $h(x)$ according to $f(x)$ and add g_1, g_2 .
- Obtain $f(x)$ for any x by sampling the reconstructed $g(x)$.

Convolution Summation

- $g(x)$ is a continuous convolution summation to compute at particular values at x .

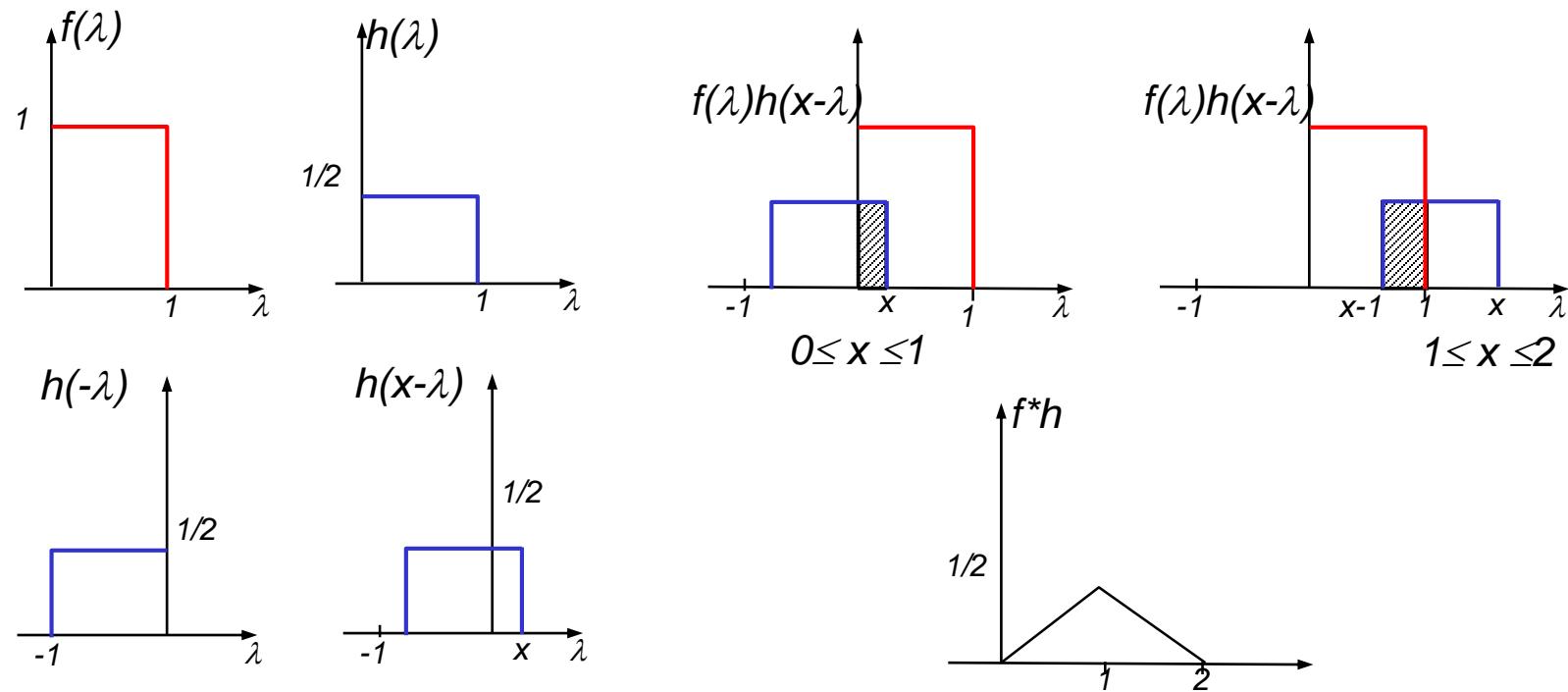
$$g(x) = f(x) \otimes h(x) = \sum_{-\infty}^{\infty} f(\lambda)h(x - \lambda)$$



$$g(x) = f_1 h(x-x_1) + f_2 h(x-x_2)$$

If $x_1=0$ and $x_2=1$ then $g(x)=f_1(1-x)+f_2x$

A Closer Look At the Convolution Integral



Mirrored Kernel

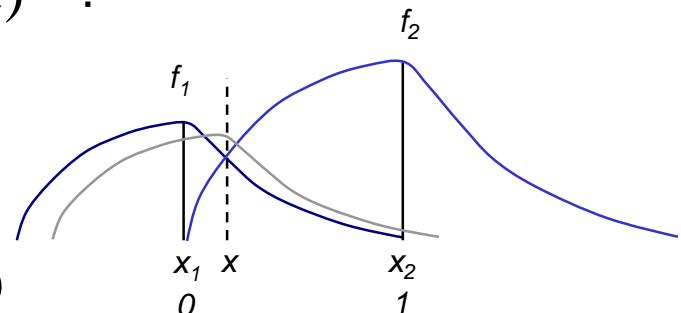
- Why fold over (mirror) kernel $h(x)$?
- Why not use $g(x) = \sum_{-\infty}^{\infty} f(\lambda)h(\lambda - x)$?

By construction : $f_1 h(x - x_1) + f_2 h(x - x_2)$

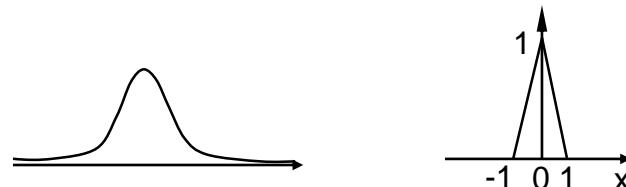
$$\sum f(\lambda)h(x - \lambda)$$

By centering h at x : $f_1 h(x_1 - x) + f_2 h(x_2 - x)$

(which is wrong) Therefore flip h before centering at x .



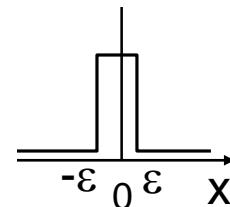
- We typically use symmetric kernels: $h(-x) = h(x)$



Impulse Function

- Impulse function (or Dirac delta function) is defined as

$$\delta(x) = \begin{cases} \lim_{\varepsilon \rightarrow 0} \int_{-\varepsilon}^{\varepsilon} f(x') dx' = 1, & x = 0 \\ 0, & x \neq 0 \end{cases}$$



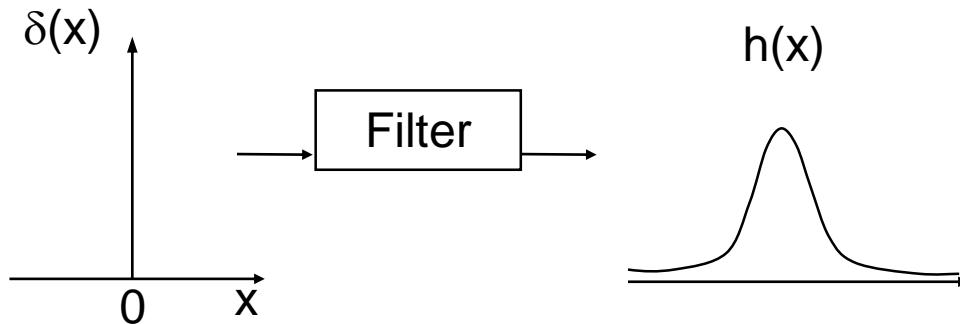
- It can be used to sample a continuous function $f(x)$ at any point x_0 , as follows:

$$f(x_0) = \int_{-\infty}^{\infty} f(\lambda) \delta(x_0 - \lambda) d\lambda = f(x_0) \delta(0) = f(x_0)$$

$$\delta(x_0 - \lambda) = 0 \text{ for } \lambda \neq x_0$$

Impulse Response

- When an impulse is applied to a filter, an altered impulse, (the *impulse response*) is generated at the output.



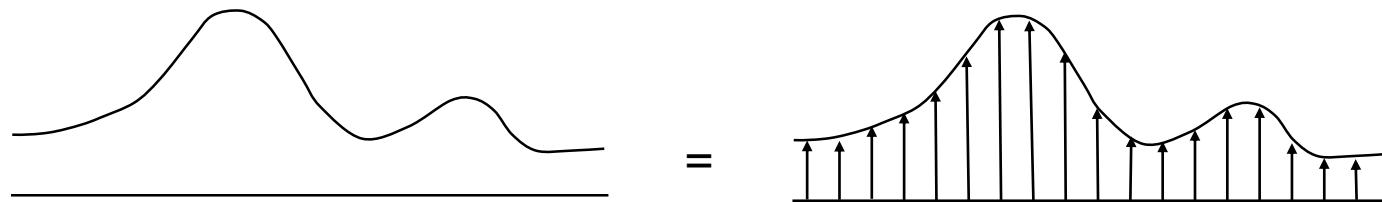
- The first direct outcome of LSI is that the filter can be uniquely characterized by its impulse response.

Sifting Integral

- Any continuous input signal can be represented in the limit by an infinite sum of shifted and scaled impulses.
- This is an outcome of the *sifting integral*:

$$f(x) = \int_{-\infty}^{\infty} f(\lambda) \delta(x - \lambda) d\lambda$$

which uses signal $f(x)$ to scale the collection of impulses:



Convolution Integral (1)

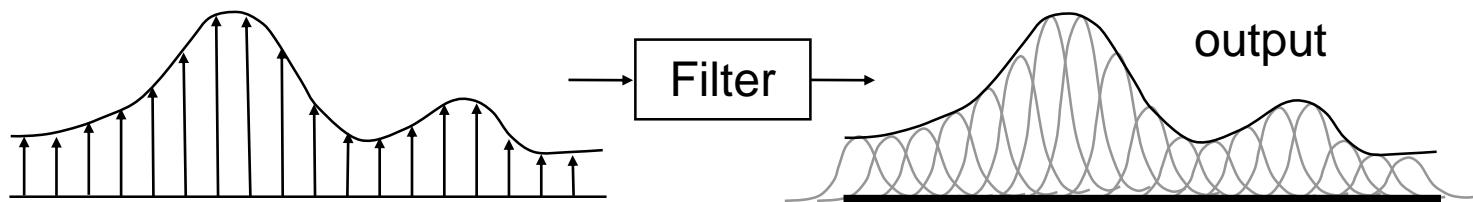
- The response $g(x)$ of a digital filter to an arbitrary input signal $f(x)$ is expressed in terms of the impulse response $h(x)$ of the filter by means of *convolution integral*:

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\lambda)h(x - \lambda)d\lambda$$

- where $*$ denotes the convolution operation,
 - $h(x)$ is used as the *convolution (filter) kernel*, and
 - λ is the dummy variable of integration.
- Kernel $h(x)$ is treated as a sliding window that is shifted across the entire input signal.
 - As $h(x)$ makes its way across $f(x)$, a sum of the pointwise products between the two functions is taken and assigned to output $g(x)$.

Convolution Integral (2)

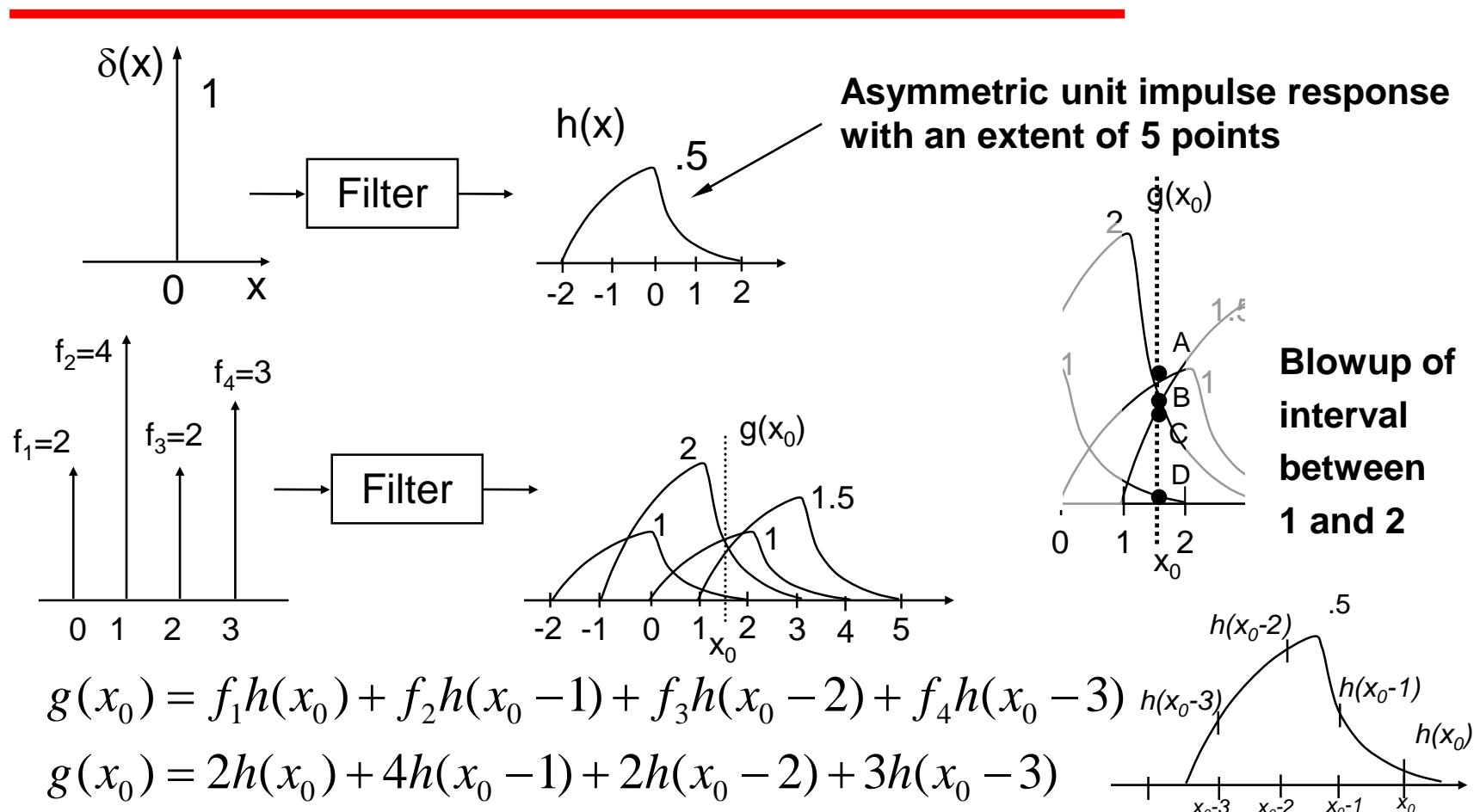
- This process, known as *convolution*, is of fundamental importance to linear filtering theory.
- It simply states that the output of an LSI filter will be a superposition of shifted and scaled impulse responses.
- This is used to explain how a continuous image is blurred by a camera as it passes through the lens.
 - In this context, $h(x)$ is known as the point spread function (PSF), reflecting the limitation of the camera to accurately resolve a small point without somewhat blurring it.



Evaluation

- We can arrive at $g(x)$ in two ways:
 - Graphical construction of a series of shifted/scaled impulse responses
 - Computing the convolution summation at all points of interest
- Graphical construction more closely follows the physical process as an input signal passes through a filter.
- Convolution summation more closely follows the practical evaluation of $g(x)$ at a finite number of desired points.
 - instead of adding scaled and shifted unit impulse responses (responses of unit impulses), we center the unit impulse response at the point of interest.

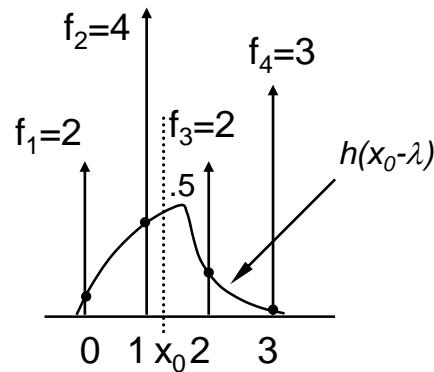
Graphical Construction



As we scan f_i from left to right we multiply samples with values taken from h in right to left order.

Convolution Summation (1)

- Instead of adding scaled/shifted unit impulse responses, center unit impulse response at point of interest:

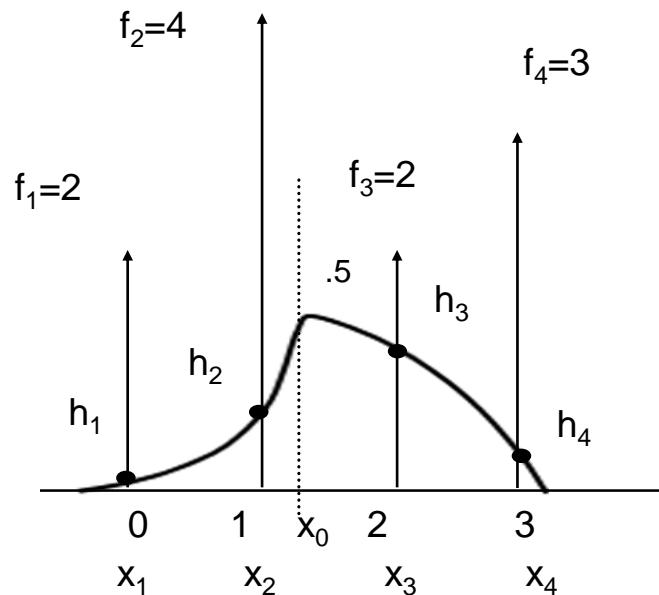


$$g(x_0) = \int_{-\infty}^{\infty} f(\lambda)h(x_0 - \lambda)d\lambda$$

- As λ increases, we scan $f()$ from left to right. However, $h(x_0 - \lambda)$ is scanned from right to left.
- Reason: points to the left of x_0 had contributed to x_0 through the right side of h (see graphical construction).

Convolution Summation (2)

- More straightforward: implement convolution if both the input and the kernel were scanned in the same direction.
- This permits direct pointwise multiplication among them.
- Thus, flip kernel before centering it on output position.



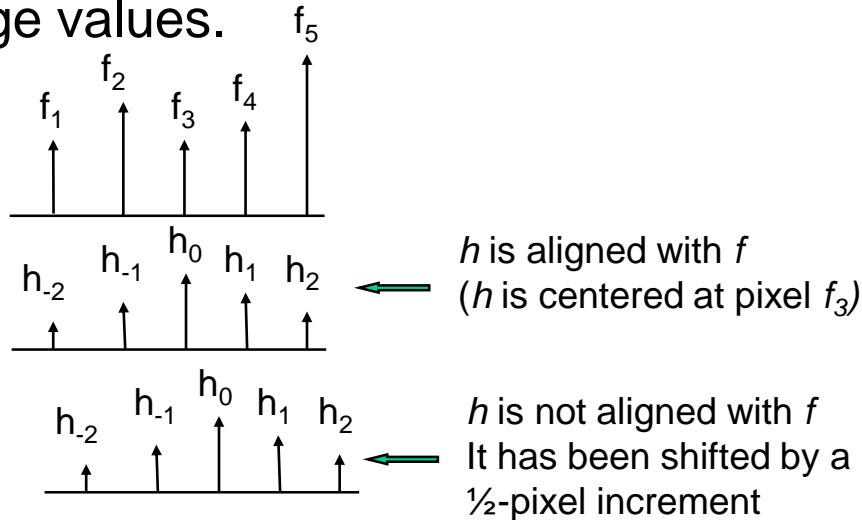
Convolution Summation (3)

- $\sum_{i=1}^4 f_i h_i$ will yield the value for $g(x_0)$, where $h_i = h(x_i - x_0)$.
- Rationale: multiplying the unit impulse response h with f_i , h is being scaled. The distance between x_0 and x_i accounts for the effect of a shifted response function on the current output.
- For most impulse response functions, h will taper off with increasing distance from its center.
- If the kernel is symmetric ($h(x) = h(-x)$), it is not necessary to flip the kernel before centering.

Discrete Convolution (1)

$$g(x) = \sum_{-\infty}^{\infty} f(\lambda)h(x - \lambda) \quad \text{for integer values of } \lambda \text{ and arbitrary values of } x$$

- The kernel is often a discrete set of weights, i.e., a 3×3 filter kernel.
- As long as kernel is shifted in pixel (integer) increments across the image, there is no alignment problem with underlying image.
- However, for noninteger pixel increments the kernel values may have no corresponding image values.



Discrete Convolution (2)

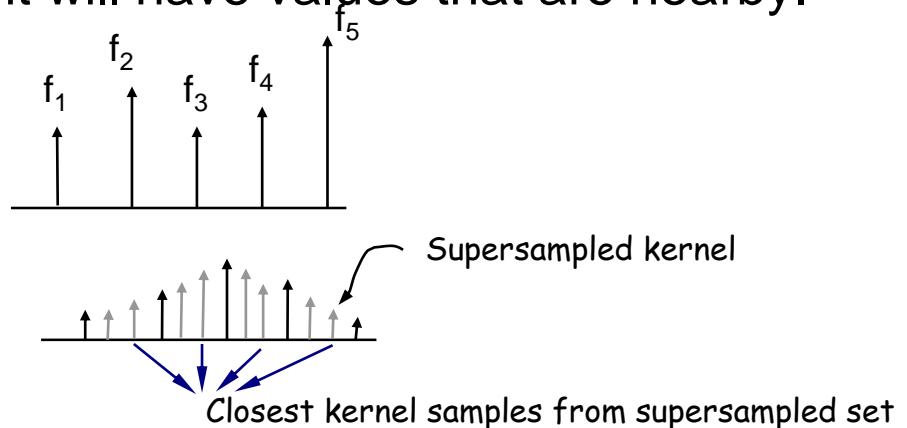
There are two possible solutions to this problem:

1. Represent h in analytic form for evaluation anywhere

Ex: Let bell-shape PSF be $h(x)=2^{-4x^2}$. The appropriate weight to apply to f_i can be obtained by setting x to the difference between the center of the bell (impulse response) and the position of the f_i .
2. Supersample h so that a dense set of samples are used to represent h . The supersampled version will then be aligned with the image data, or at least it will have values that are nearby.

If the kernel is known to be slid across the image at fixed increments, then the kernel can be sampled at known positions.

Ex: a 1/3 increment requires the kernel to be sampled three times per unit interval.



Fourier Transform

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- This lecture reviews Fourier transforms and processing in the frequency domain.
 - Definitions
 - Fourier series
 - Fourier transform
 - Fourier analysis and synthesis
 - Discrete Fourier transform (DFT)
 - Fast Fourier transform (FFT)

Background (1)

- Fourier proved that any periodic function can be expressed as the sum of sinusoids of different frequencies, each multiplied by a different coefficient. → *Fourier series*
- Even aperiodic functions (whose area under the curve is finite) can be expressed as the integral of sinusoids multiplied by a weighting function. → *Fourier transform*
- In a great leap of imagination, Fourier outlined these results in a memoir in 1807 and published them in *La Theorie Analytique de la Chaleur* (The Analytic theory of Heat) in 1822. The book was translated into English in 1878.

Background (2)

- The Fourier transform is more useful than the Fourier series in most practical problems since it handles signals of finite duration.
- The Fourier transform takes us between the spatial and frequency domains.
- It permits for a dual representation of a signal that is amenable for filtering and analysis.
- Revolutionized the field of signal processing.

Example

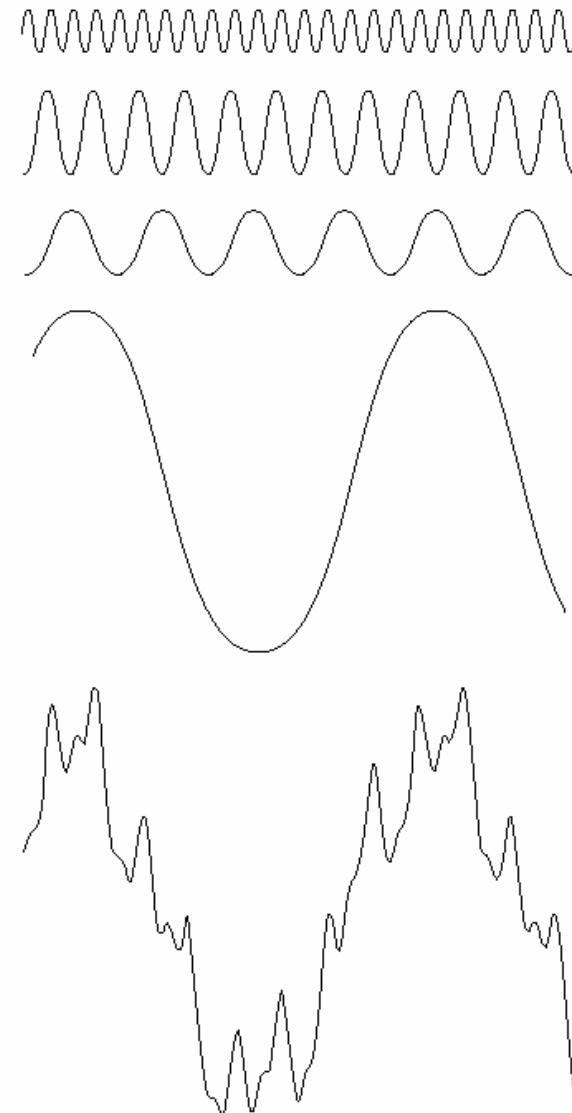
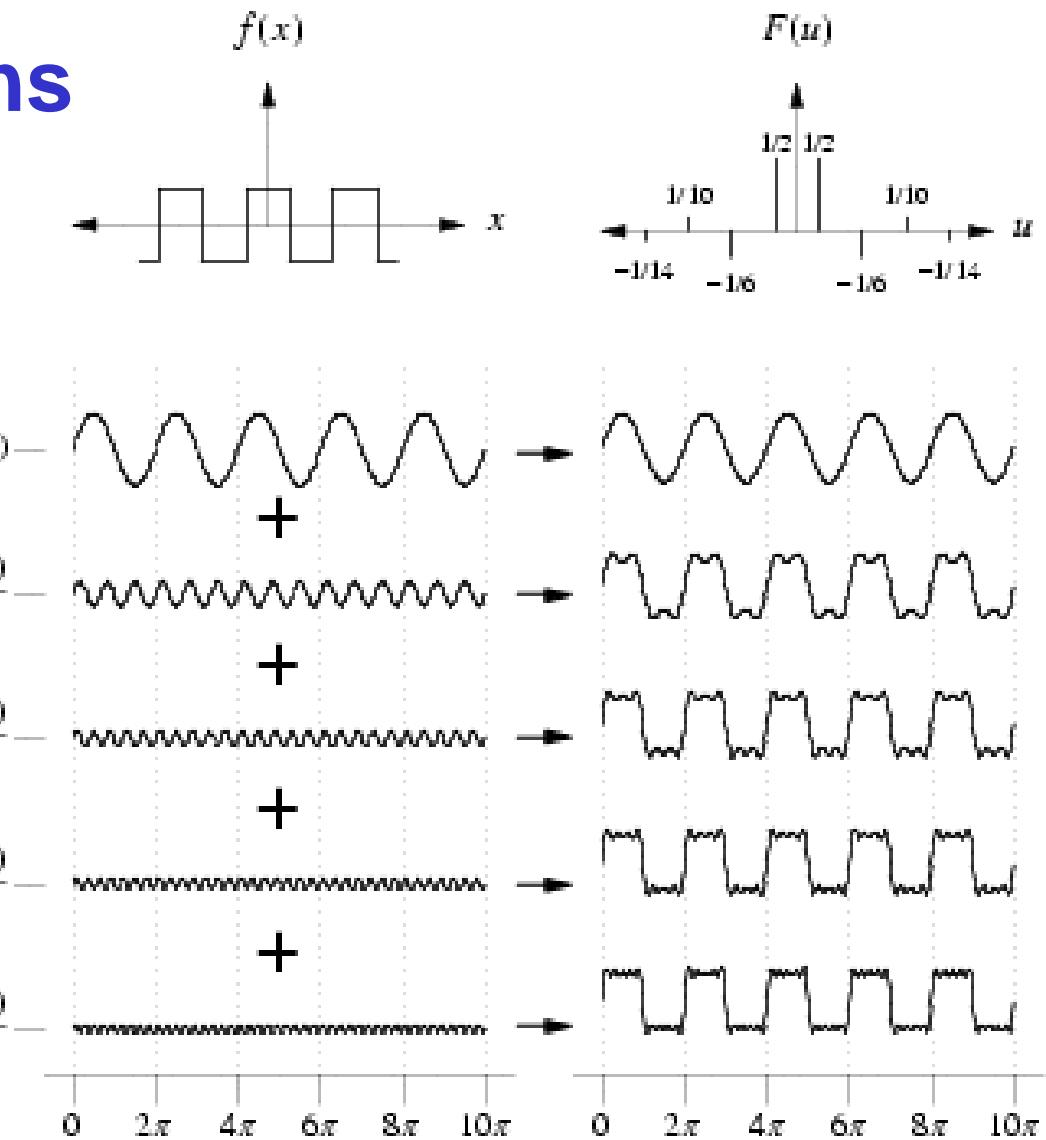


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

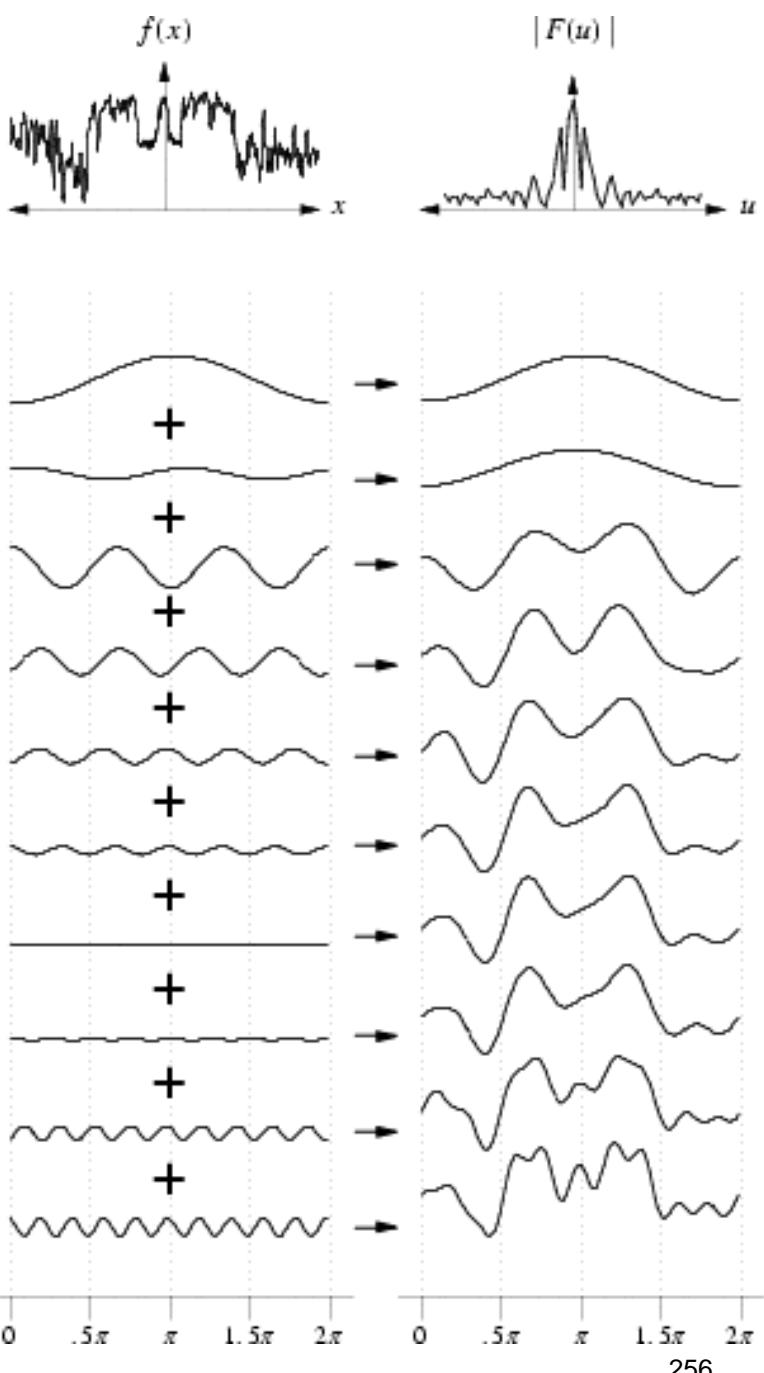
Useful Analogy

- A glass prism is a physical device that separates light into various color components, each depending on its wavelength (or frequency) content.
- The Fourier transform is a mathematical prism that separates a function into its frequency components.

Fourier Series for Periodic Functions



Fourier Transform for Aperiodic Functions



Fourier Analysis and Synthesis

- Fourier analysis: determine amplitude & phase shifts
- Fourier synthesis: add scaled and shifted sinusoids together
- Fourier transform pair:

$$\text{Forward F.T. } F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi ux} dx$$

$$\text{Inverse F.T. } f(x) = \int_{-\infty}^{\infty} F(u)e^{+i2\pi ux} dx$$

where $i = \sqrt{-1}$, and

$$e^{\pm i2\pi ux} = \cos(2\pi ux) \pm i \sin(2\pi ux) \leftarrow \text{complex exponential at freq. } u$$

Euler's formula
↑

Fourier Coefficients

- Fourier coefficients $F(u)$ specify, for each frequency u , the amplitude and phase of each complex exponential.
- $F(u)$ is the frequency spectrum.
- $f(x)$ and $F(u)$ are two equivalent representations of the same signal.

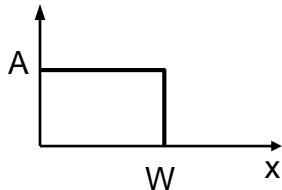
$$F(u) = R(u) + iI(u)$$

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \leftarrow \text{magnitude spectrum; aka Fourier spectrum}$$

$$\Phi(u) = \tan^{-1} \frac{I(u)}{R(u)} \leftarrow \text{phase spectrum}$$

$$\begin{aligned} P(u) &= |F(u)|^2 \\ &= R^2(u) + I^2(u) \leftarrow \text{spectral density} \end{aligned}$$

1D Example



$$f(x) = \begin{cases} A & 0 \leq x \leq W \\ 0 & x > W \end{cases}$$

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi u x} dx$$

note : $\int e^{ax} dx = \frac{1}{a} e^{ax}$

$$F(u) = \int_0^W A e^{-i2\pi u x} dx$$

$$F(u) = \frac{-A}{i2\pi u} [e^{-i2\pi u x}]_0^W = \frac{-A}{i2\pi u} [e^{-i2\pi u W} - 1] = \frac{A}{i2\pi u} [1 - e^{-i2\pi u W}]$$

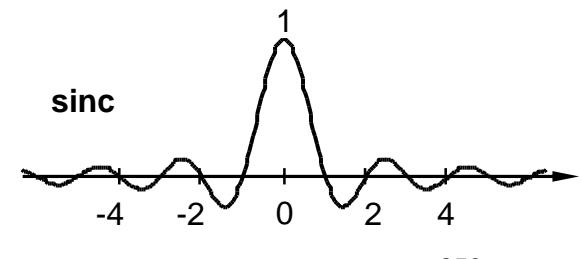
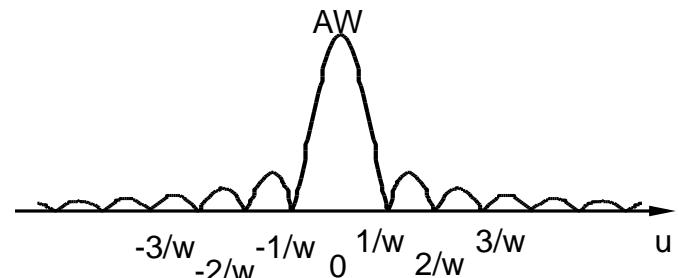
$$F(u) = \frac{A}{i2\pi u} [e^{i\pi u W} - e^{-i\pi u W}] e^{-i\pi u W} \quad \text{note : } \sin x = \frac{e^{ix} - e^{-ix}}{2i}$$

$$F(u) = \frac{A}{\pi u} \sin(\pi u W) e^{-i\pi u W} \leftarrow \text{complex function}$$

$$|F(u)| = \left| \frac{A}{\pi u} \right| \left| \sin(\pi u W) \right| |e^{-i\pi u W}| = AW \left| \frac{\sin(\pi u W)}{\pi u W} \right| = AW |\text{sinc}(\pi u W)|$$

where $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ (some books have $\text{sinc}(x) = \frac{\sin(x)}{x}$)

Wolberg: Image Processing Course Notes



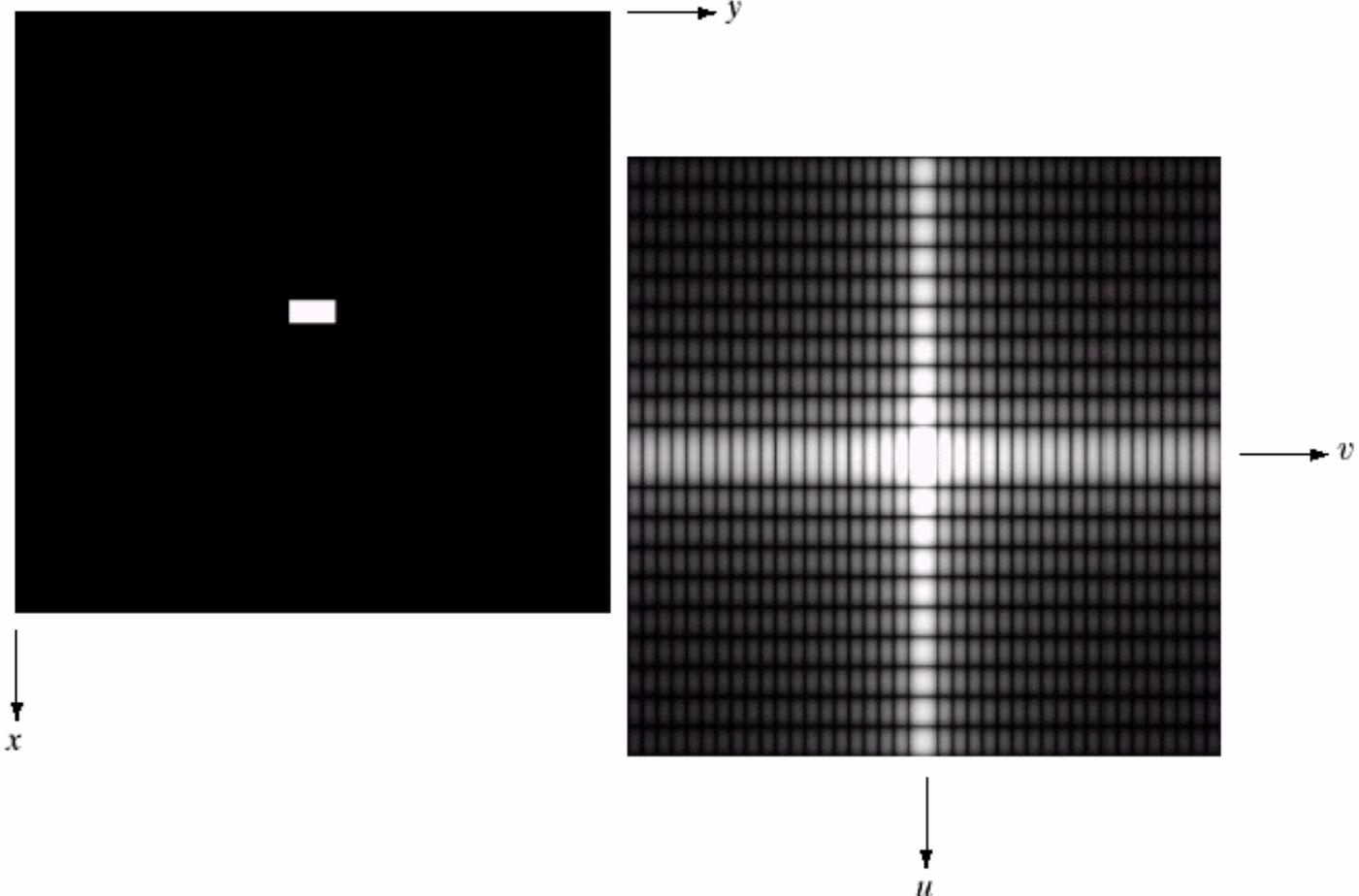
2D Example

a b

FIGURE 4.3

(a) Image of a 20×40 white rectangle on a black background of size 512×512 pixels.

(b) Centered Fourier spectrum shown after application of the log transformation given in Eq. (3.2-2). Compare with Fig. 4.2.



Fourier Series (1)

For periodic signals, we have the Fourier series:

$$f(x) = \sum_{n=-\infty}^{n=\infty} c(nu_0) e^{i2\pi u_0 x} \text{ where } c(nu_0) \text{ is the } n^{\text{th}} \text{ Fourier coefficient}$$

$$c(nu_0) = \frac{1}{x_0 - x_0/2} \int_{x_0/2}^{x_0/2} f(x) e^{-i2\pi u_0 x} dx$$

That is, the periodic signal contains all the frequencies that are harmonics of the fundamental frequency.

Fourier Series (2)

$$c(nu_0) = \frac{1}{x_0} \int_{-x_0/2}^{x_0/2} f(x) e^{-i2\pi n u_0 x} dx = \frac{1}{x_0} \int_{-W/2}^{W/2} A e^{-i2\pi n u_0 x} dx$$

$$c(nu_0) = \frac{A}{-i2\pi n u_0 x_0} (e^{-i\pi n u_0 W} - e^{+i\pi n u_0 W})$$

$$c(nu_0) = \frac{A}{\pi n} \sin(\pi n u_0 W) \leftarrow \sin x = \frac{e^{ix} - e^{-ix}}{2i}; u_0 x_0 = 1$$

$$c(nu_0) = \frac{A u_0 W}{\pi n u_0 W} \sin(\pi n u_0 W) = A u_0 W \text{sinc}(\pi n u_0 W)$$

Note that if $\frac{W}{2} = \frac{x_0}{2}$, then we have a square wave and

$$c(nu_0) = A u_0 x_0 \text{sinc}(\pi n u_0 x_0)$$

$$c(nu_0) = \begin{cases} A \text{sinc}(n) & n = \pm 1, \pm 3, \dots \\ 0 & n = 0, \pm 2, \pm 4, \dots \end{cases}$$

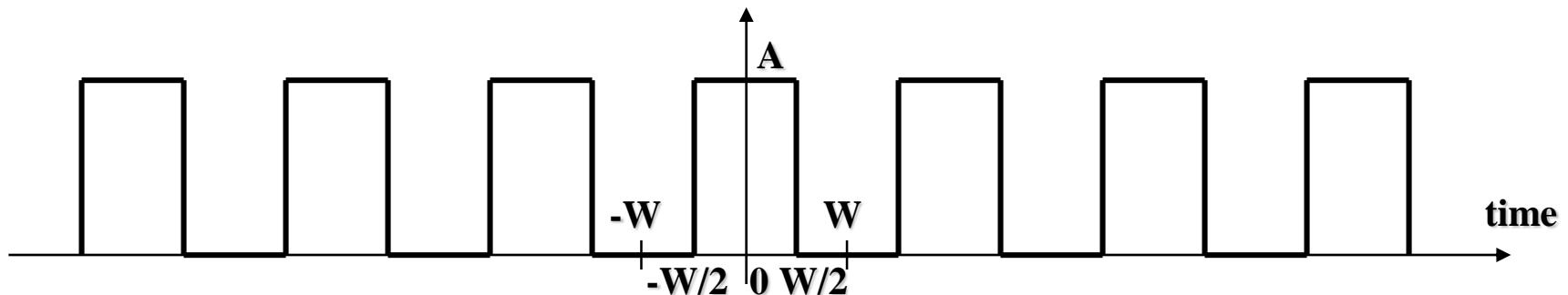
Fourier Series (3)

- The Fourier transform is applied for aperiodic signals.
- It is represented as an integral over a continuum of frequencies.
- The Fourier Series is applied for periodic signals.
- It is represented as a summation of frequency components that are integer multiples of some fundamental frequency.

Example

Ex : Rectangular Pulse Train

$$f(x) = \begin{cases} A & |x| < \frac{W}{2} \\ 0 & |x| > \frac{W}{2} \end{cases} \quad \text{in interval } [-W/2, W/2]$$



Discrete Fourier Transform

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-i2\pi \frac{ux}{N}} \quad \text{forward DFT}$$

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{+i2\pi \frac{ux}{N}} \quad \text{inverse DFT}$$

for $0 \leq u \leq N-1$ and $0 \leq x \leq N-1$ where N is the number of equi-spaced input samples.

The $1/N$ factor can be in front of $f(x)$ instead.

Fourier Analysis Code

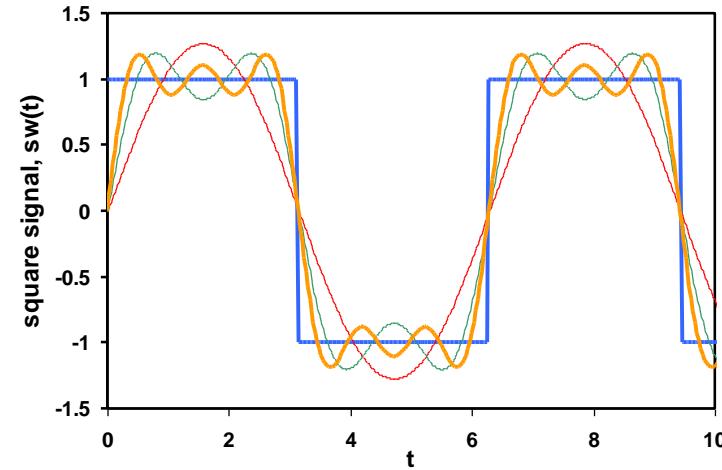
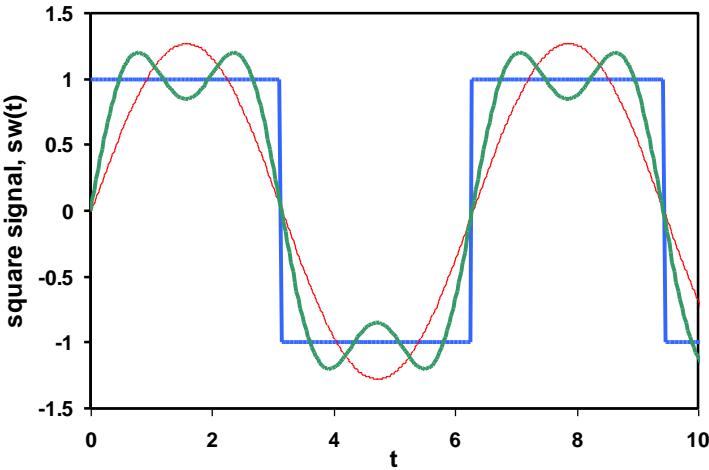
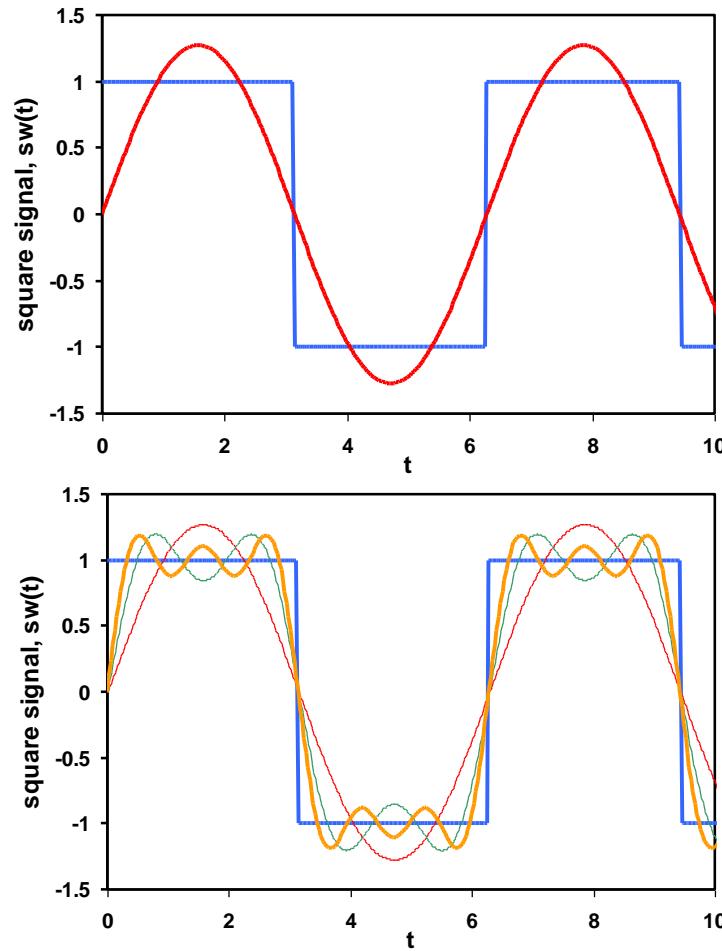
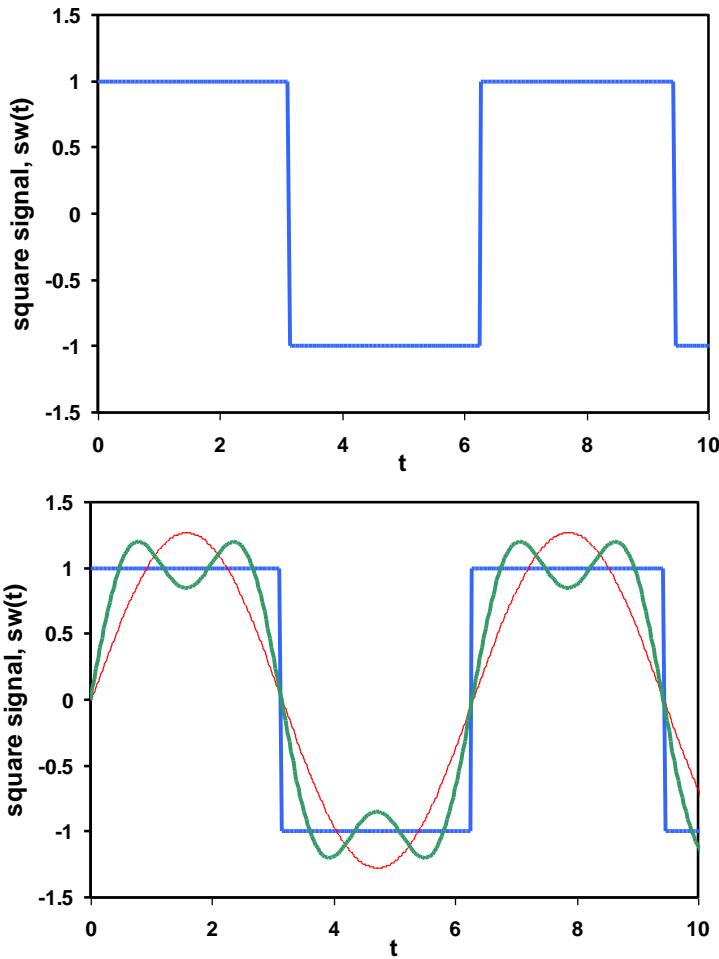
- DFT maps N input samples of f into the N frequency terms in F .

```
for(u=0; u<N; u++) { /*compute spectrum over all freq. u */
    real = imag = 0;      /*reset real, imag component of F(u)*/
    for(x=0; x<N; x++) { /* visit each input pixel */
        real += (f[x]*cos(-2*PI*u*x/N));
        imag += (f[x]*sin(-2*PI*u*x/N));
        /* Note: if f is complex, then
           real += (fr[x]*cos()-fi[x]*sin());
           imag += (fr[x]*sin()+fi[x]*cos());
           because (f_r+if_i)(g_r+ig_i)=(f_rg_r-f_ig_i)+i(f_ig_r+f_rg_i)
        */
    }
    Fr[u] = real / N;
    Fi[u] = imag / N;
}
```

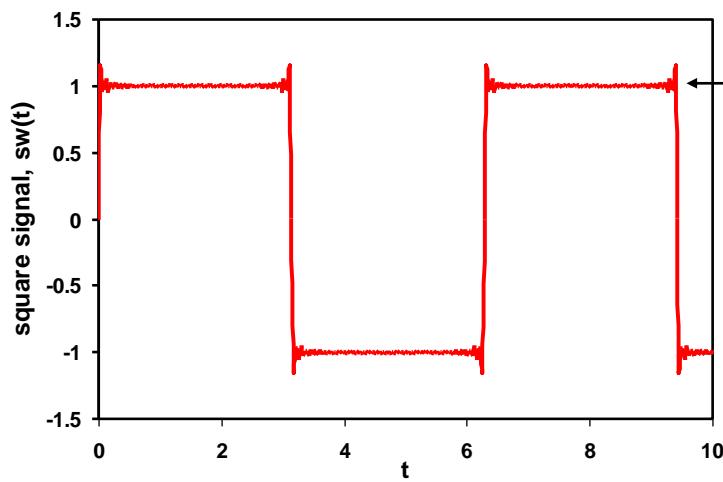
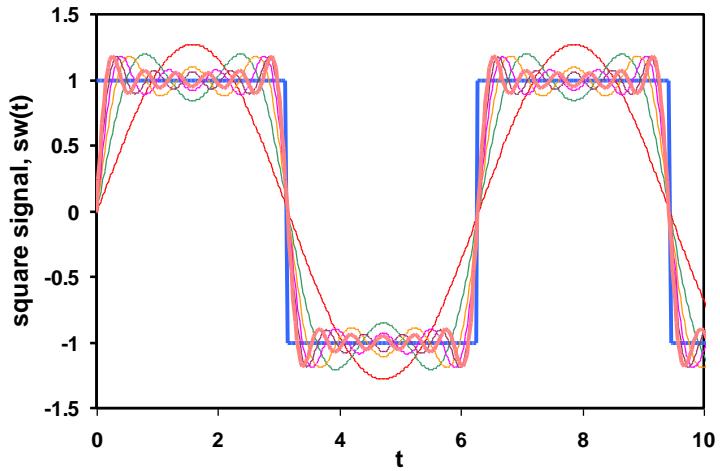
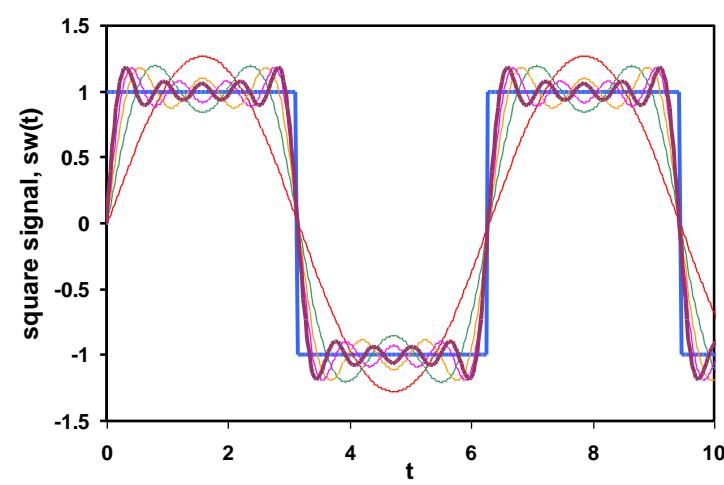
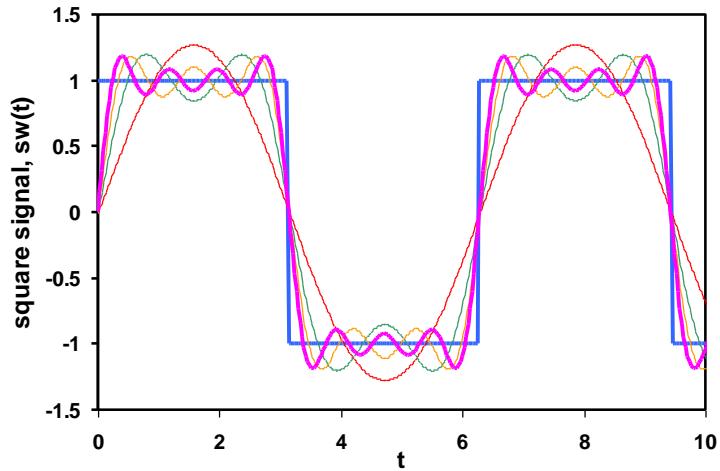
Fourier Synthesis Code

```
for(x=0; x<N; x++) /* compute each output pixel */
    real = imag = 0; /* reset real, imaginary component */
    for(u=0; u<N; u++) {
        c = cos(2*PI*u*x/N);
        s = sin(2*PI*u*x/N);
        real += (Fr[u]*c-Fi[u]*s);
        imag += (Fr[u]*s+Fi[u]*c);
    }
    fr[x] = real; /* OR f[x] = sqrt(real*real + imag*imag);
    fi[x] = imag;
}
```

Example: Fourier Analysis (1)

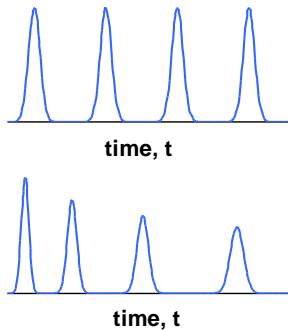


Example: Fourier Analysis (2)



Gibbs phenomenon

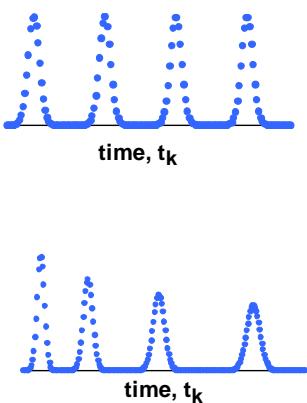
Summary



Continuous	<table border="0"> <tr> <td>Periodic <i>(period T)</i></td><td>FS</td><td>Discrete</td></tr> <tr> <td>Aperiodic</td><td>FT</td><td>Continuous</td></tr> </table>	Periodic <i>(period T)</i>	FS	Discrete	Aperiodic	FT	Continuous
Periodic <i>(period T)</i>	FS	Discrete					
Aperiodic	FT	Continuous					

$$c_k = \frac{1}{T} \cdot \int_0^T s(t) \cdot e^{-ik\omega t} dt$$

$$S(f) = \int_{-\infty}^{+\infty} s(t) \cdot e^{-i2\pi f t} dt$$



Discrete	<table border="0"> <tr> <td>Periodic <i>(period T)</i></td><td>D FS</td><td>Discrete</td></tr> <tr> <td>Aperiodic</td><td> <table border="0"> <tr> <td>DTFT</td><td>Continuous</td></tr> <tr> <td>DFT</td><td>Discrete</td></tr> </table> </td><td></td></tr> </table>	Periodic <i>(period T)</i>	D FS	Discrete	Aperiodic	<table border="0"> <tr> <td>DTFT</td><td>Continuous</td></tr> <tr> <td>DFT</td><td>Discrete</td></tr> </table>	DTFT	Continuous	DFT	Discrete	
Periodic <i>(period T)</i>	D FS	Discrete									
Aperiodic	<table border="0"> <tr> <td>DTFT</td><td>Continuous</td></tr> <tr> <td>DFT</td><td>Discrete</td></tr> </table>	DTFT	Continuous	DFT	Discrete						
DTFT	Continuous										
DFT	Discrete										

$$\tilde{c}_k = \frac{1}{N} \sum_{n=0}^{N-1} s[n] \cdot e^{-i\frac{2\pi kn}{N}}$$

$$S(f) = \sum_{n=-\infty}^{+\infty} s[n] \cdot e^{-i2\pi f n}$$

$$\tilde{c}_k = \frac{1}{N} \sum_{n=0}^{N-1} s[n] \cdot e^{-i\frac{2\pi kn}{N}}$$

Note: $i = \sqrt{-1}$, $\omega = 2\pi/T$, $s[n] = s(t_n)$, $N = \# \text{ of samples}$

2D Fourier Transform

Continuous:

$$F\{f(x, y)\} = F(u, v) = \iint f(x, y) e^{-i2\pi(ux+vy)} dx = \iint f(x, y) e^{-i2\pi ux} e^{-i2\pi vy} dx$$

$$F^{-1}\{F(u, v)\} = f(x, y) = \iint F(u, v) e^{+i2\pi(ux+vy)} dx = \iint F(u, v) e^{+i2\pi ux} e^{+i2\pi vy} dx$$

Separable: $F(u, v) = F(u)F(v)$

Discrete:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) e^{-i2\pi(\frac{ux}{N} + \frac{vy}{M})} dx = \frac{1}{M} \sum_{y=0}^{M-1} \left[\frac{1}{N} \sum_{x=0}^{N-1} f(x, y) e^{-i2\pi(\frac{ux}{N})} \right] e^{-i2\pi(\frac{vy}{M})}$$

$$f(x, y) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} F(u, v) e^{+i2\pi(\frac{ux}{N} + \frac{vy}{M})} dx = \sum_{y=0}^{M-1} \left[\sum_{x=0}^{N-1} F(u, v) e^{+i2\pi(\frac{ux}{N})} \right] e^{+i2\pi(\frac{vy}{M})}$$

Separable Implementation

$$\begin{aligned} F(u, v) &= \frac{1}{N} \sum_{y=0}^{N-1} e^{-j2\pi vy/N} \boxed{\frac{1}{M} \sum_{x=0}^{M-1} f(x, y) e^{-j2\pi ux/M}} \\ &= \frac{1}{N} \sum_{y=0}^{N-1} F(u, y) e^{-j2\pi vy/N} \quad \text{transform each row} \\ &\quad \text{transform each column of intermediate result} \end{aligned}$$

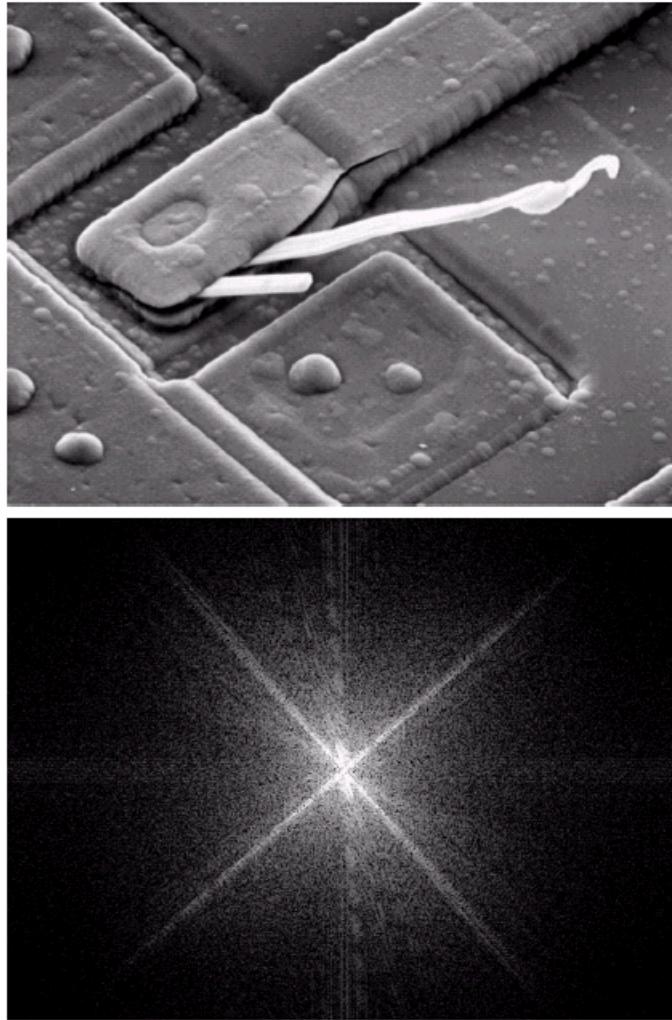
where $F(u, y) = \frac{1}{M} \sum_{x=0}^{M-1} f(x, y) e^{-j2\pi ux/M}$

The 2D Fourier transform is computed in two passes:

- 1) Compute the transform along each row independently.
- 2) Compute the transform along each column of this intermediate result.

Properties

- Edge orientations in image appear in spectrum, rotated by 90°.
- 3 orientations are prominent: 45°, -45°, and nearly horizontal long white element.



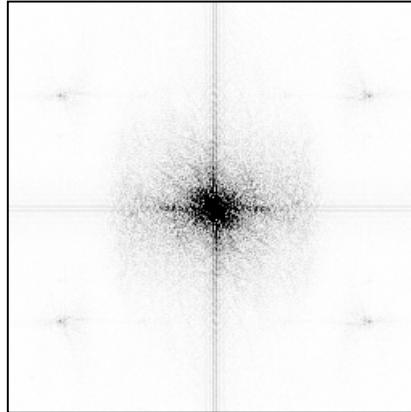
a
b

FIGURE 4.4
(a) SEM image of a damaged integrated circuit.
(b) Fourier spectrum of (a).
(Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

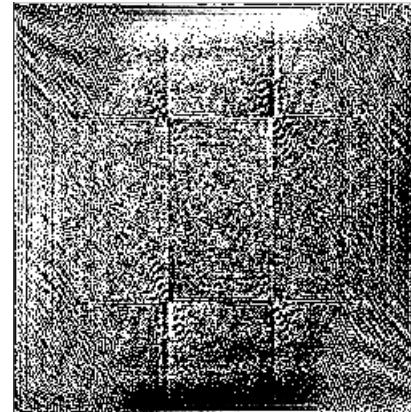
Magnitude and Phase Spectrum



Mad.bw

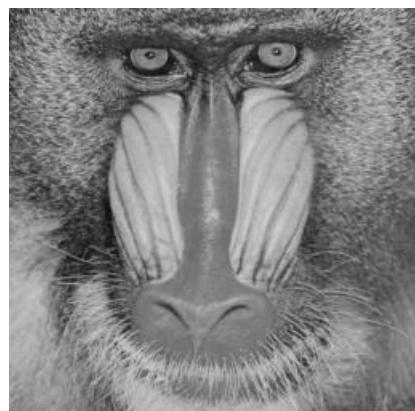


Magnitude

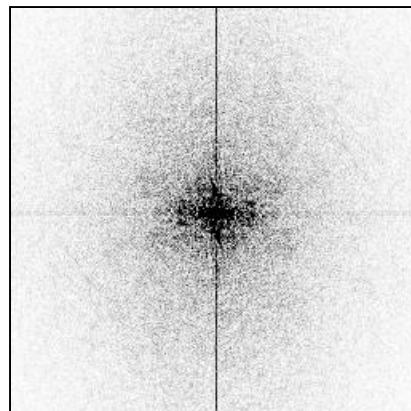


Phase

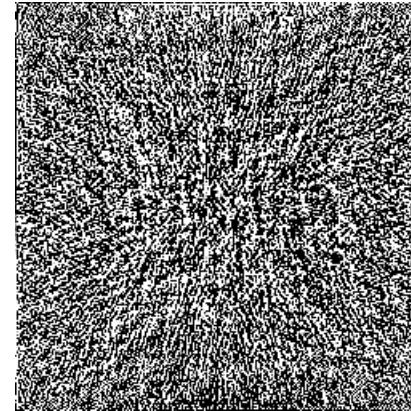
2D-Fourier
transforms
example



Mandrill.bw



Magnitude



Phase

Role of Magnitude vs Phase (1)

Rick



Linda



Pictures reconstructed
using the Fourier phase
of another picture

Mag{Linda}
Phase{Rick}



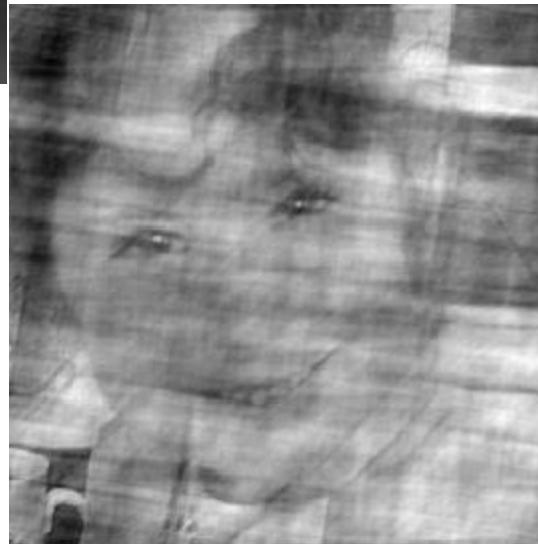
Mag{Rick}
Phase{Linda}



Role of Magnitude vs Phase (2)

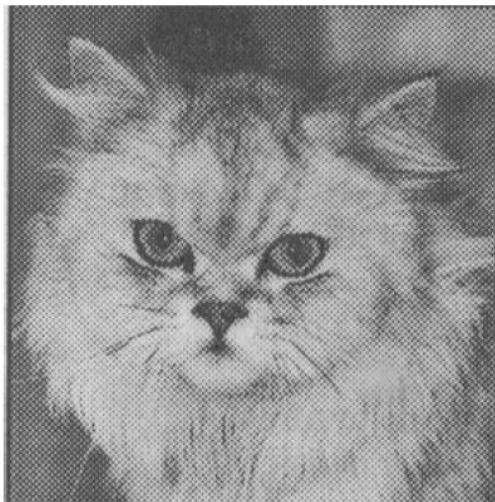


Magnitude

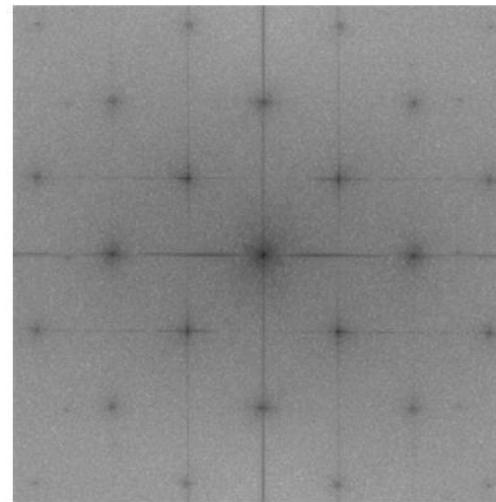


Phase

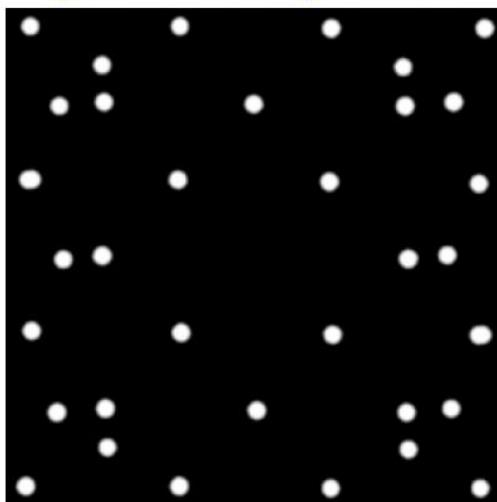
Noise Removal



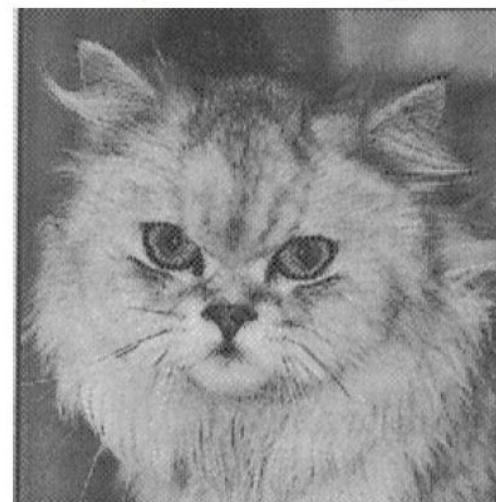
Original with noise patterns



Power spectrum showing noise spikes



Mask to remove periodic noise



Inverse FT with periodic noise removed

Fast Fourier Transform (1)

- The DFT was defined as:

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-i2\pi \frac{ux}{N}} \quad 0 \leq x \leq N-1$$

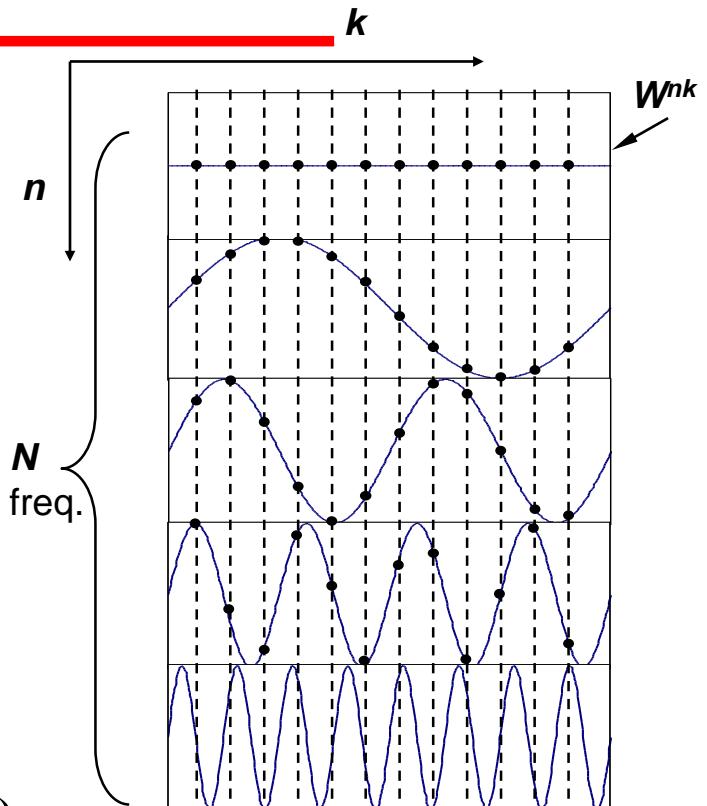
Rewrite:

$$F_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{-i2\pi \frac{nk}{N}} \quad 0 \leq n \leq N-1$$

$$\text{Let } F_n = \sum_{k=0}^{N-1} f_k W^{nk}$$

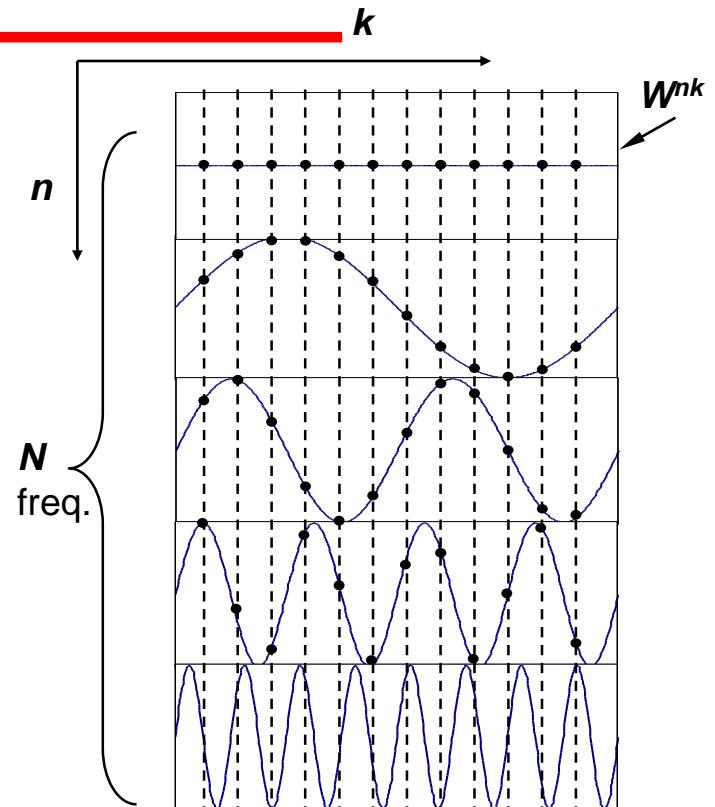
$$\text{where } W = e^{\frac{-i2\pi}{N}} = \cos\left(\frac{-2\pi}{N}\right) + i \sin\left(\frac{-2\pi}{N}\right)$$

Also, Let $N = 2^r$ (N is a power of 2)



Fast Fourier Transform (2)

- W^{nk} can be thought of as a 2D array, indexed by n and k .
- It represents N equispaced values along a sinusoid at each of N frequencies.
- For each frequency n , there are N multiplications (N samples in sine wave of freq. n). Since there are N frequencies, DFT: $O(N^2)$
- With the FFT, we will derive an $O(N \log N)$ process.



Computational Advantage

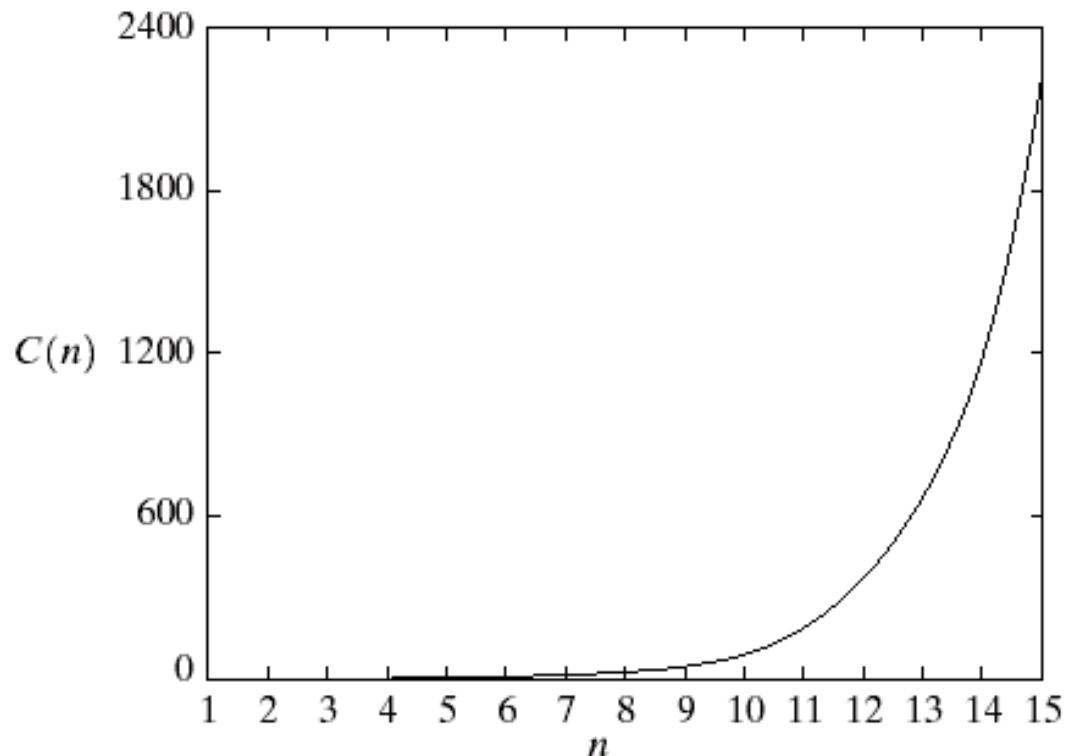


FIGURE 4.42
Computational advantage of the FFT over a direct implementation of the 1-D DFT. Note that the advantage increases rapidly as a function of n .

Danielson–Lanczos Lemma (1)

1942:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi \frac{nk}{N}}$$

$$F_n = \sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{-i2\pi \frac{n(2k)}{N}} + \sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{-i2\pi \frac{n(2k+1)}{N}}$$

Even Numbered Terms

f_0, f_2, f_4, \dots

Odd Numbered Terms

f_1, f_3, f_5, \dots

Danielson–Lanczos Lemma (2)

$$F_n = \sum_{k=0}^{\frac{N}{2}-1} f_{2k} e^{\frac{-i2\pi nk}{N/2}} + W^n \sum_{k=0}^{\frac{N}{2}-1} f_{2k+1} e^{\frac{-i2\pi nk}{N/2}}$$

$$W = e^{\frac{-i2\pi}{N}}$$

$$F_n = F_n^e + W^n F_n^o$$

nth component of F.T. of length N/2 formed from the **even** components of f

nth component of F.T. of length N/2 formed from the **odd** components of f

Divide-and-Conquer solution: Solving a problem (F_n) is reduced to 2 smaller ones.

Potential Problem: n in F_n^e and F_n^o is still made to vary from 0 to N-1. Since each sub-problem is no smaller than original, it appears wasteful.

Solution: Exploit symmetries to reduce computational complexity.

Danielson–Lanczos Lemma (3)

Given : a DFT of length N , $F_{n+N} = F_n$

$$\begin{aligned}\text{Proof : } F_{n+N} &= \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi(n+N)k}{N}} = \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} e^{\frac{-i2\pi Nk}{N}} = \\ &= \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} (\cos 2\pi k - i \sin 2\pi k) = \sum_{k=0}^{N-1} f_k e^{\frac{-i2\pi nk}{N}} = F_n\end{aligned}$$

$$\begin{aligned}W^{n+\frac{N}{2}} &= \cos\left(\frac{-2\pi}{N}\left(n + \frac{N}{2}\right)\right) + i \sin\left(\frac{-2\pi}{N}\left(n + \frac{N}{2}\right)\right) \\ &= \cos\left(\frac{-2\pi n}{N} - \pi\right) + i \sin\left(\frac{-2\pi n}{N} - \pi\right) \\ &= -\cos\left(\frac{-2\pi n}{N}\right) - i \sin\left(\frac{-2\pi n}{N}\right) \\ &= -W^n\end{aligned}$$

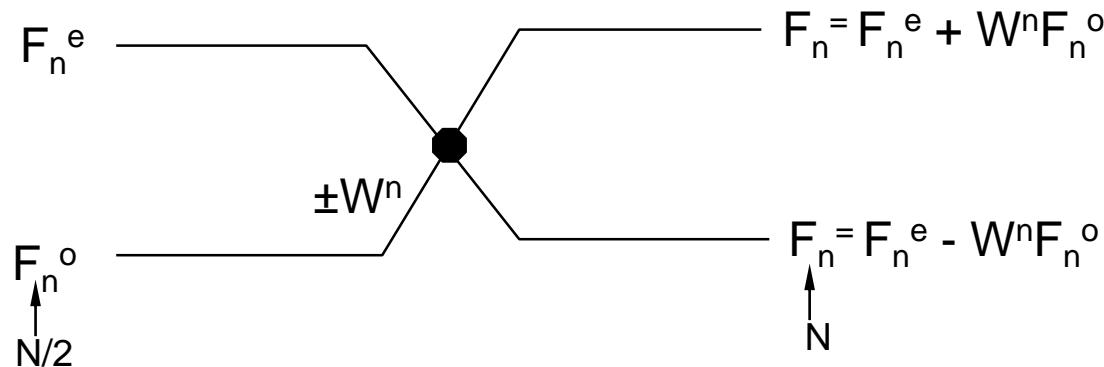
Main Points of FFT

$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi \frac{nk}{N}}$$

$$F_n = F_n^e + W^n F_n^o$$

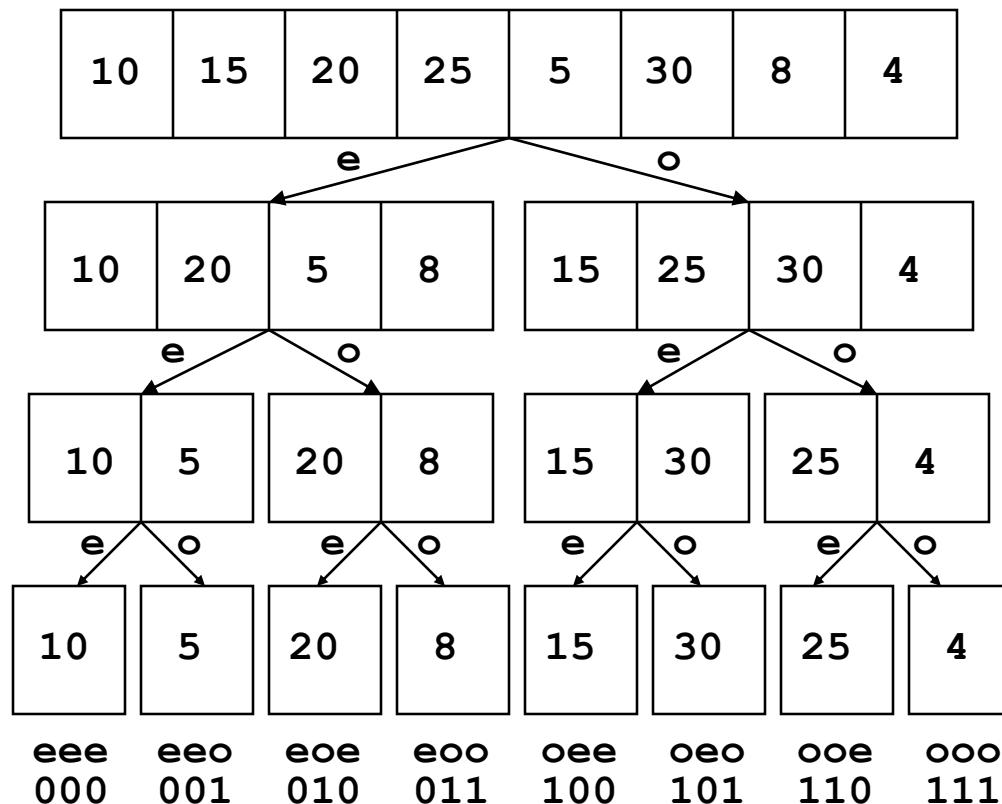
length N length $\frac{N}{2}$ length $\frac{N}{2}$

but $0 \leq n < N$

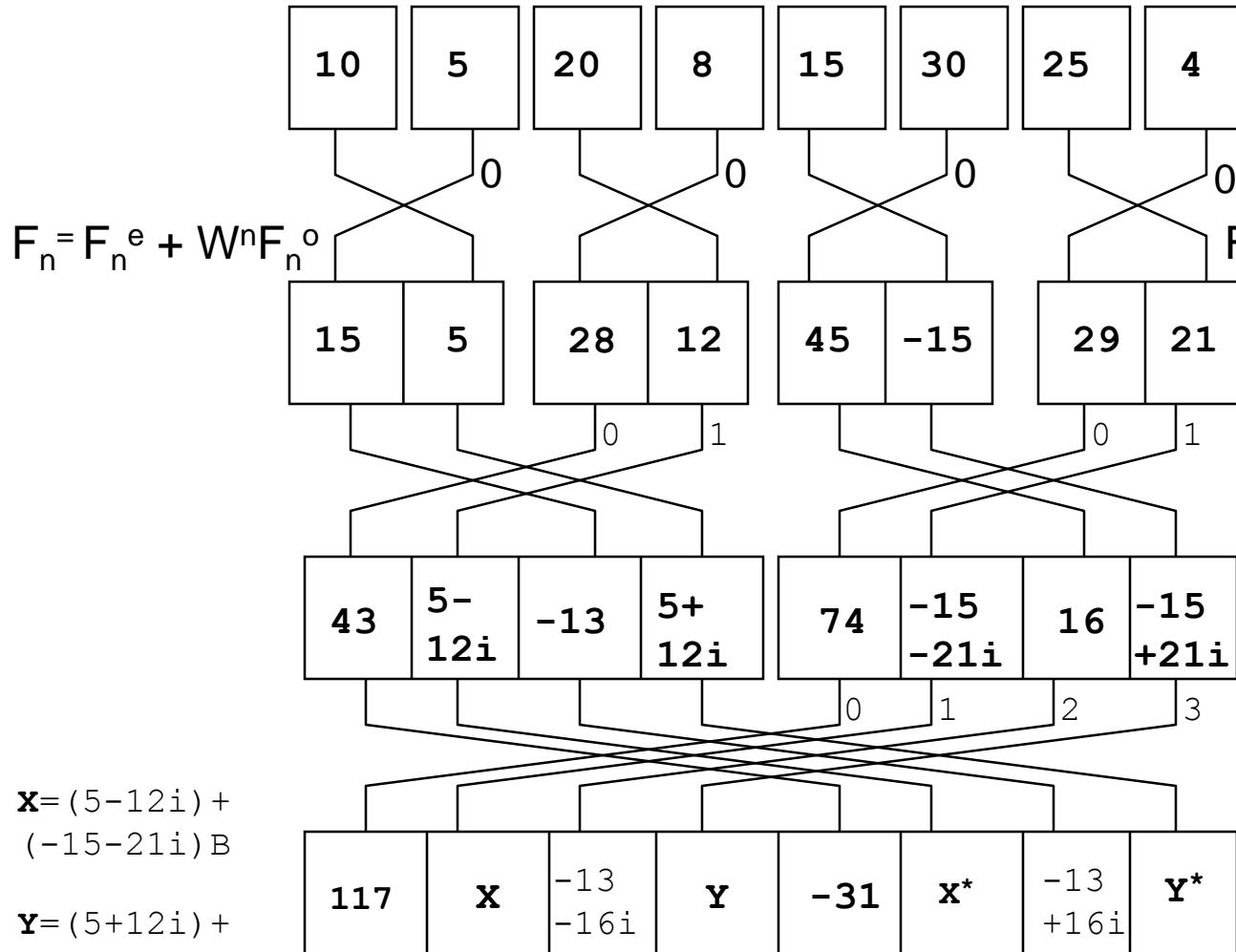


FFT Example (1)

- Input: 10, 15, 20, 25, 5, 30, 8, 4

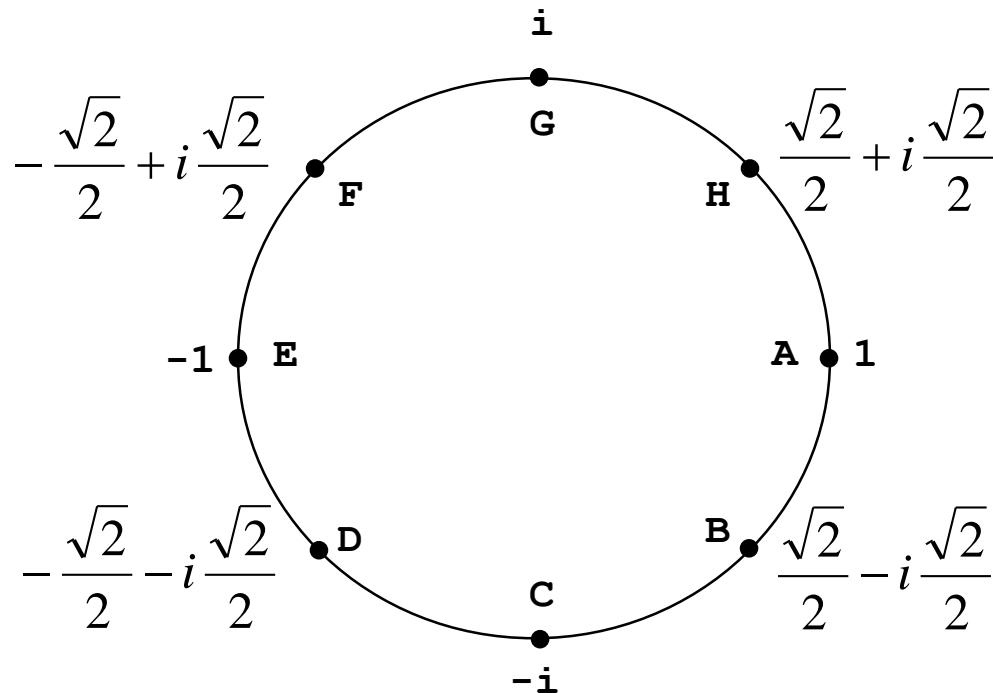


FFT Example (2)



Weights

- DFT is a convolution with kernel values $e^{-i2\pi ux/N}$
- These values are derived from a unit circle.



DFT Example (1)

- Input: 10, 15, 20, 25, 5, 30, 8, 4

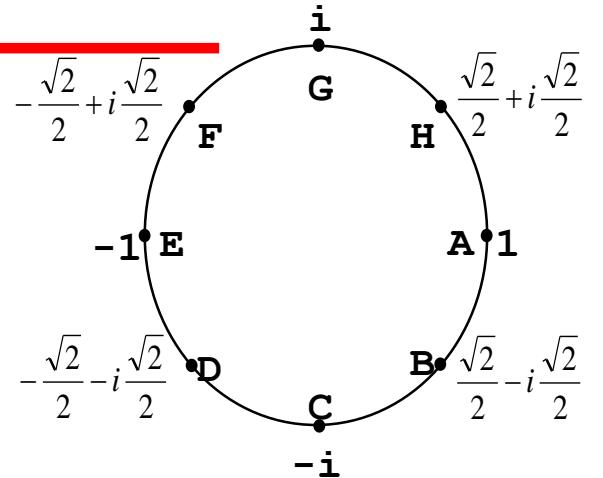
$$F_n = \sum_{k=0}^{N-1} f_k e^{-i2\pi \frac{nk}{N}}$$

$$F_0 = 10 + 15 + 20 + 25 + 5 + 30 + 8 + 4 = 117$$

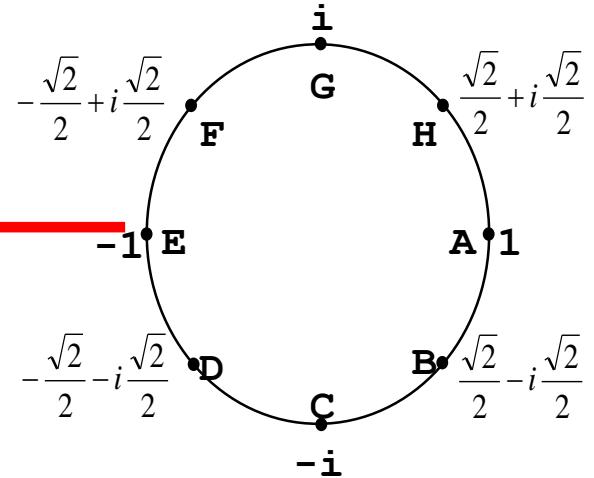
$$\begin{aligned} F_1 &= 10e^{-i2\pi(1)(0)/8} + 15e^{-i2\pi(1)(1)/8} + 20e^{-i2\pi(1)(2)/8} + \dots + 8e^{-i2\pi(1)(6)/8} + 4e^{-i2\pi(1)(7)/8} \\ &= 10A + 15B + 20C + 25D + 5E + 30F + 8G + 4H \end{aligned}$$

$$\begin{aligned} F_2 &= 10e^{-i2\pi(2)(0)/8} + 15e^{-i2\pi(2)(1)/8} + 20e^{-i2\pi(2)(2)/8} + \dots + 8e^{-i2\pi(2)(6)/8} + 4e^{-i2\pi(2)(7)/8} \\ &= 10A + 15C + 20E + 25G + 5A + 30C + 8E + 4G = -13 - 16i \end{aligned}$$

$$\begin{aligned} F_3 &= 10e^{-i2\pi(3)(0)/8} + 15e^{-i2\pi(3)(1)/8} + 20e^{-i2\pi(3)(2)/8} + \dots + 8e^{-i2\pi(3)(6)/8} + 4e^{-i2\pi(3)(7)/8} \\ &= 10A + 15D + 20G + 25B + 5E + 30H + 8C + 4F \end{aligned}$$



DFT Example (2)



$$\begin{aligned}
 F_4 &= 10e^{-i2\pi(4)(0)/8} + 15e^{-i2\pi(4)(1)/8} + 20e^{-i2\pi(4)(2)/8} + \dots + 8e^{-i2\pi(4)(6)/8} + 4e^{-i2\pi(4)(7)/8} \\
 &= 10A + 15E + 20A + 25E + 5A + 30E + 8A + 4E = -31
 \end{aligned}$$

$$\begin{aligned}
 F_5 &= 10e^{-i2\pi(5)(0)/8} + 15e^{-i2\pi(5)(1)/8} + 20e^{-i2\pi(5)(2)/8} + \dots + 8e^{-i2\pi(5)(6)/8} + 4e^{-i2\pi(5)(7)/8} \\
 &= 10A + 15F + 20C + 25H + 5E + 30B + 8G + 4D
 \end{aligned}$$

$$\begin{aligned}
 F_6 &= 10e^{-i2\pi(6)(0)/8} + 15e^{-i2\pi(6)(1)/8} + 20e^{-i2\pi(6)(2)/8} + \dots + 8e^{-i2\pi(6)(6)/8} + 4e^{-i2\pi(6)(7)/8} \\
 &= 10A + 15G + 20E + 25C + 5A + 30G + 8E + 4C = -13 + 16i
 \end{aligned}$$

$$\begin{aligned}
 F_7 &= 10e^{-i2\pi(7)(0)/8} + 15e^{-i2\pi(7)(1)/8} + 20e^{-i2\pi(7)(2)/8} + \dots + 8e^{-i2\pi(7)(6)/8} + 4e^{-i2\pi(7)(7)/8} \\
 &= 10A + 15H + 20G + 25F + 5E + 30D + 8C + 4B
 \end{aligned}$$

TABLE 4.1

Summary of some important properties of the 2-D Fourier transform.

Property	Expression(s)
Fourier transform	$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M+vy/N)}$
Inverse Fourier transform	$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M+vy/N)}$
Polar representation	$F(u, v) = F(u, v) e^{-j\phi(u, v)}$
Spectrum	$ F(u, v) = [R^2(u, v) + I^2(u, v)]^{1/2}, \quad R = \text{Real}(F) \text{ and } I = \text{Imag}(F)$
Phase angle	$\phi(u, v) = \tan^{-1} \left[\frac{I(u, v)}{R(u, v)} \right]$
Power spectrum	$P(u, v) = F(u, v) ^2$
Average value	$\bar{f}(x, y) = F(0, 0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$
Translation	$f(x, y) e^{j2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u - u_0, v - v_0)$ $f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(ux_0/M+vy_0/N)}$ <p>When $x_0 = u_0 = M/2$ and $y_0 = v_0 = N/2$, then</p> $f(x, y)(-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ $f(x - M/2, y - N/2) \Leftrightarrow F(u, v)(-1)^{u+v}$

Conjugate symmetry	$F(u, v) = F^*(-u, -v)$ $ F(u, v) = F(-u, -v) $
Differentiation	$\frac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$ $(-jx)^n f(x, y) \Leftrightarrow \frac{\partial^n F(u, v)}{\partial u^n}$
Laplacian	$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2) F(u, v)$
Distributivity	$\Im[f_1(x, y) + f_2(x, y)] = \Im[f_1(x, y)] + \Im[f_2(x, y)]$ $\Im[f_1(x, y) \cdot f_2(x, y)] \neq \Im[f_1(x, y)] \cdot \Im[f_2(x, y)]$
Scaling	$af(x, y) \Leftrightarrow aF(u, v), f(ax, by) \Leftrightarrow \frac{1}{ ab } F(u/a, v/b)$
Rotation	$x = r \cos \theta \quad y = r \sin \theta \quad u = \omega \cos \varphi \quad v = \omega \sin \varphi$ $f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$
Periodicity	$F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$ $f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$
Separability	See Eqs. (4.6-14) and (4.6-15). Separability implies that we can compute the 2-D transform of an image by first computing 1-D transforms along each row of the image, and then computing a 1-D transform along each column of this intermediate result. The reverse, columns and then rows, yields the same result.

TABLE 4.1
(continued)

Property	Expression(s)
Computation of the inverse Fourier transform using a forward transform algorithm	$\frac{1}{MN} f^*(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M + vy/N)}$ <p>This equation indicates that inputting the function $F^*(u, v)$ into an algorithm designed to compute the forward transform (right side of the preceding equation) yields $f^*(x, y)/MN$. Taking the complex conjugate and multiplying this result by MN gives the desired inverse.</p>
Convolution [†]	$f(x, y) * h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n)$
Correlation [†]	$f(x, y) \circ h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n)h(x + m, y + n)$
Convolution theorem [†]	$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v);$ $f(x, y)h(x, y) \Leftrightarrow F(u, v) * H(u, v)$
Correlation theorem [†]	$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v)H(u, v);$ $f^*(x, y)h(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$

TABLE 4.1
(continued)

Some useful FT pairs:

<i>Impulse</i>	$\delta(x, y) \Leftrightarrow 1$
<i>Gaussian</i>	$A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)} \Leftrightarrow Ae^{-(u^2+v^2)/2\sigma^2}$
<i>Rectangle</i>	$\text{rect}[a, b] \Leftrightarrow ab \frac{\sin(\pi ua)}{(\pi ua)} \frac{\sin(\pi vb)}{(\pi vb)} e^{-j\pi(ua+vb)}$
<i>Cosine</i>	$\cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow \frac{1}{2} [\delta(u + u_0, v + v_0) + \delta(u - u_0, v - v_0)]$
<i>Sine</i>	$\sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow j \frac{1}{2} [\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)]$

[†] Assumes that functions have been extended by zero padding.

TABLE 4.1
(continued)

Filtering in the Frequency Domain

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- This lecture reviews frequency domain filtering.
 - Convolution theorem
 - Frequency bands
 - Lowpass filter
 - Ideal, Butterworth, Gaussian
 - Highpass filter
 - Ideal, Butterworth, Gaussian
 - Notch filter
 - Homomorphic filtering

Basic Steps

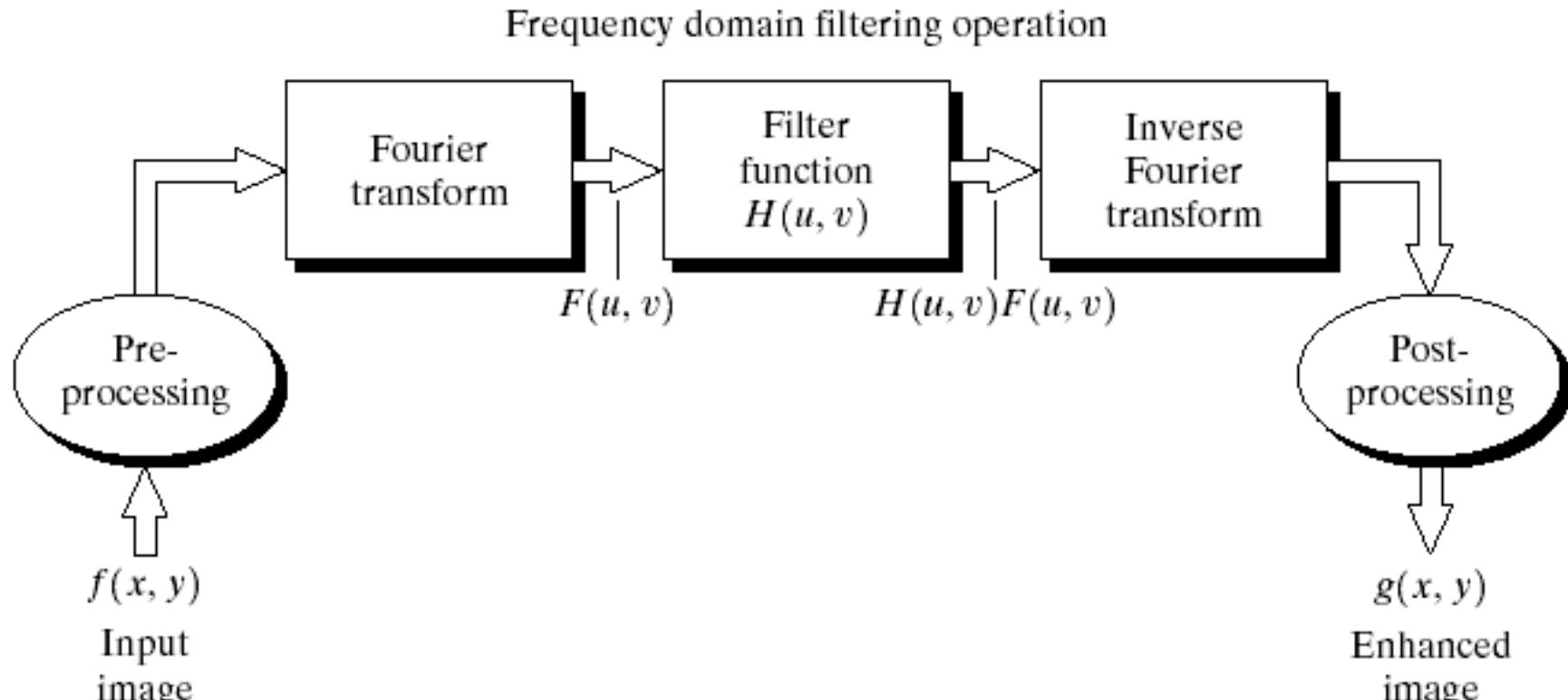


FIGURE 4.5 Basic steps for filtering in the frequency domain.

Basics

1. Multiply input image by $(-1)^{x+y}$ to center the transform to $u = M/2$ and $v = N/2$ (if M and N are even numbers, then shifted coordinates will be integers)
2. Compute $F(u,v)$, the DFT of the image from (1)
3. Multiply $F(u,v)$ by a filter function $H(u,v)$
4. Compute the inverse DFT of the result in (3)
5. Obtain the real part of the result in (4)
6. Multiply the result in (5) by $(-1)^{x+y}$ to cancel the multiplication of the input image.

Convolution Theorem

“Multiply $F(u,v)$ by a filter function $H(u,v)$ ”

This step exploits the convolution theorem:

Convolution in one domain is equivalent to multiplication in the other domain.

$$\begin{aligned} f(x)^*g(x) &\leftrightarrow F(u) G(u) \\ f(x) g(x) &\leftrightarrow F(u)^*G(u) \end{aligned}$$

Periodicity

$$F(u,v) = F(u+M, v) = F(u,v+N) = F(u+M,v+N)$$

a	b
c	d

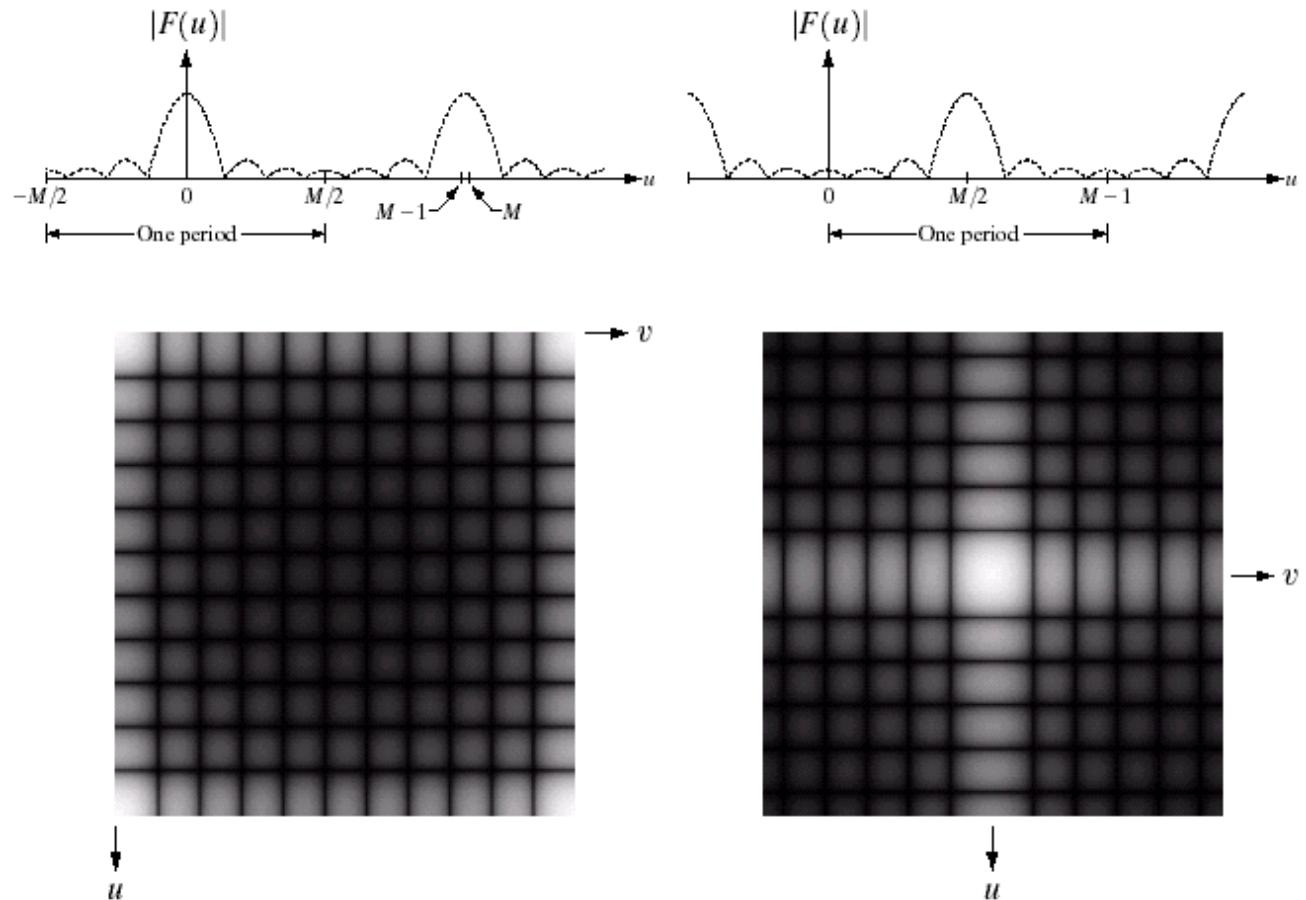
FIGURE 4.34

(a) Fourier spectrum showing back-to-back half periods in the interval $[0, M - 1]$.

(b) Shifted spectrum showing a full period in the same interval.

(c) Fourier spectrum of an image, showing the same back-to-back properties as (a), but in two dimensions.

(d) Centered Fourier spectrum.



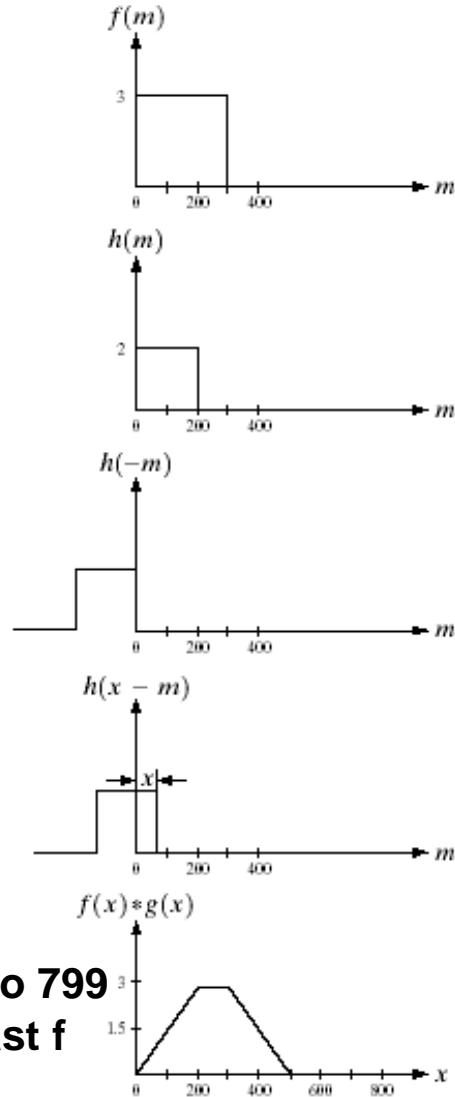
Comments

- The magnitudes from $(M/2)+1$ to $M-1$ are reflections of the values in the half period to the left of the origin.
- DFT is formulated for values of u in the interval $[0, M-1]$.
- This yields two back-to-back half periods in this interval.
- For one full period, move origin to $u=M/2$ by multiplying $f(x)$ by $(-1)^x$.

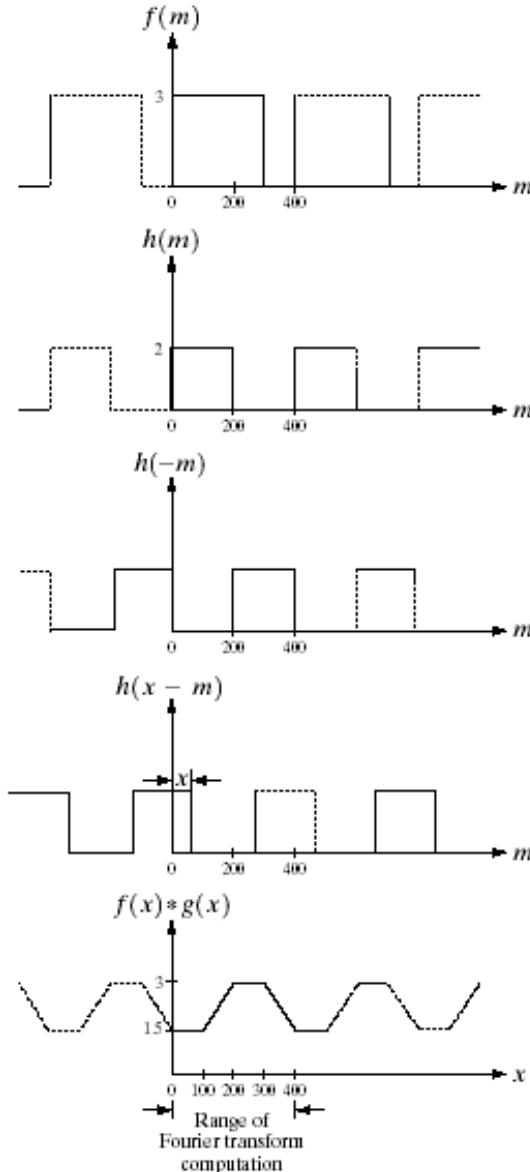
Significance of Periodicity

a	f
b	g
c	h
d	i
e	j

FIGURE 4.36 Left: convolution of two discrete functions. Right: convolution of the same functions, taking into account the implied periodicity of the DFT. Note in (j) how data from adjacent periods corrupt the result of convolution.



**x ranges from 0 to 799
for $h()$ to slide past f**

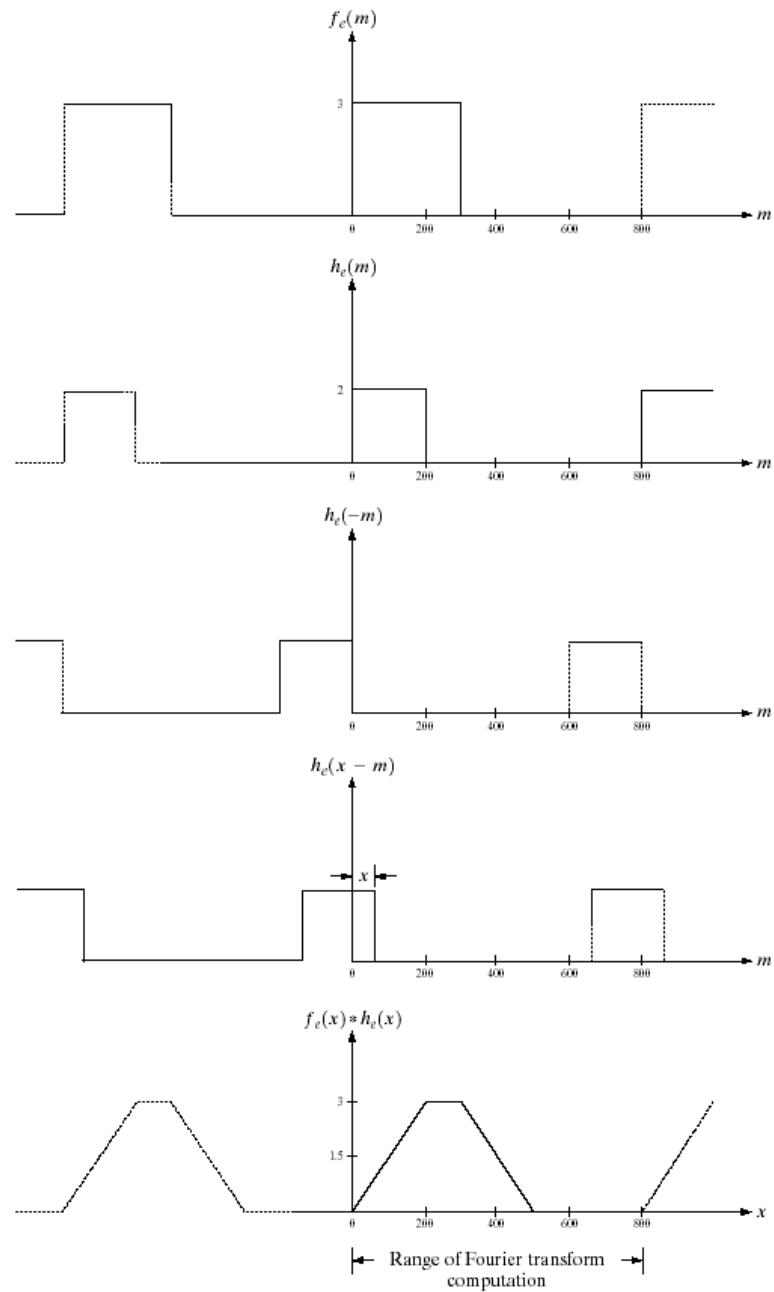


Padding (1)

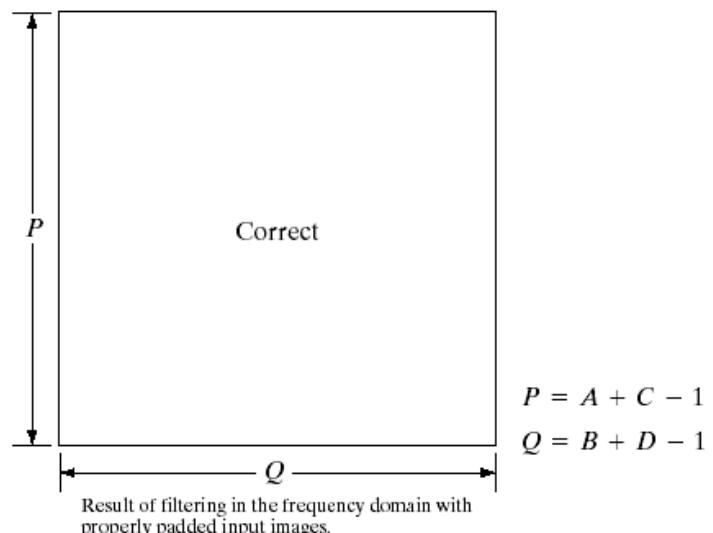
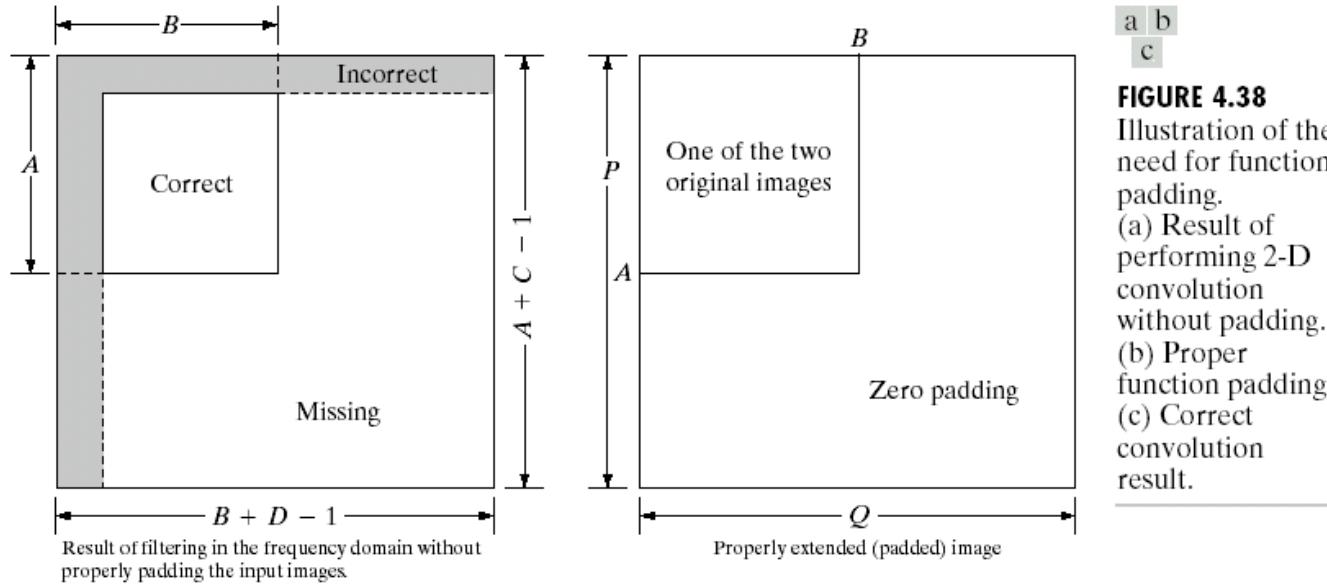
a
b
c
d
e

FIGURE 4.37
Result of performing convolution with extended functions. Compare Figs. 4.37(e) and 4.36(e).

- Corrupted results were due to inadequate period.
- Solution: add padding to increase period.



Padding (2)



Example

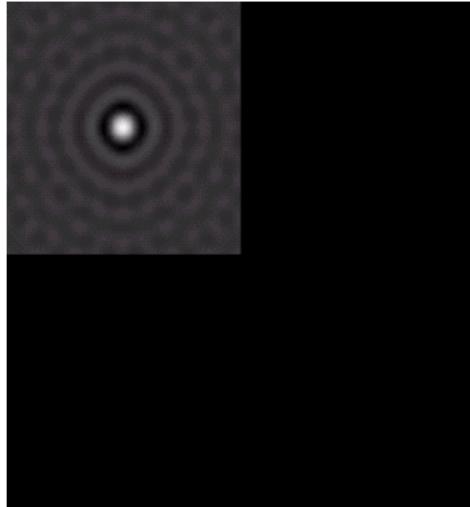


FIGURE 4.39 Padded lowpass filter is the spatial domain (only the real part is shown).

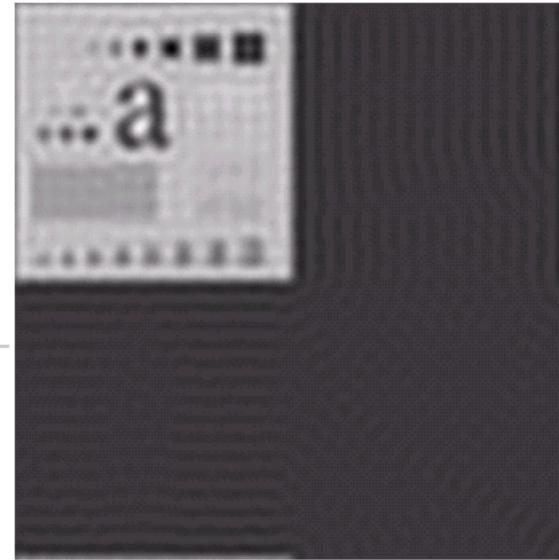


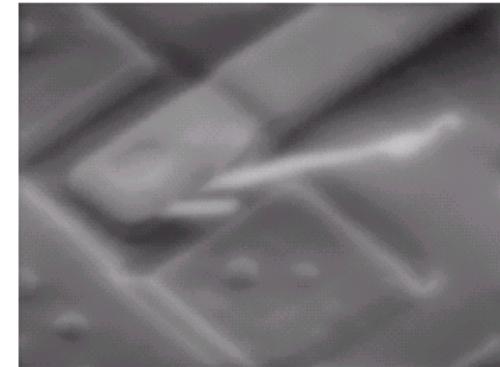
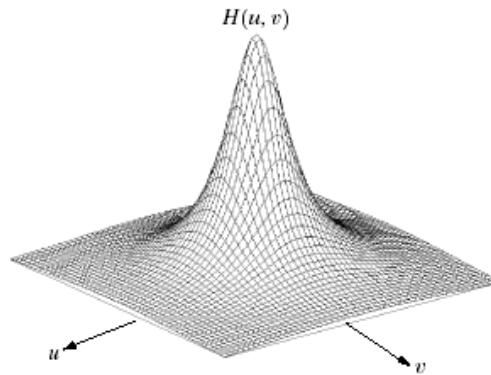
FIGURE 4.40 Result of filtering with padding. The image is usually cropped to its original size since there is little valuable information past the image boundaries.

Frequency Bands

- Low frequencies: general graylevel appearance over smooth areas: slowly varying grayscales.
- High frequencies: responsible for detail, such as edges and noise.
- A filter that attenuates high frequencies while “passing” low frequencies is a *lowpass filter*.
- A filter that attenuates low frequencies while “passing” high frequencies is a *highpass filter*.
- Lowpass filters blur images.
- Highpass filters highlight edges.

Lowpass and Highpass Filters

Lowpass filter



Highpass filter

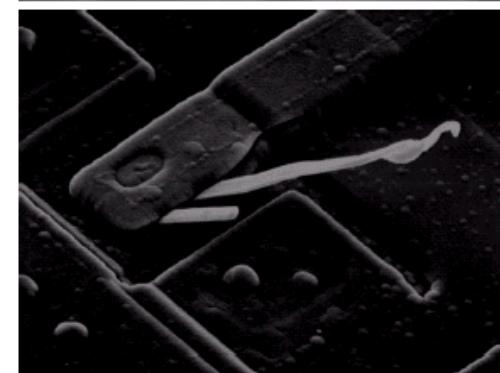
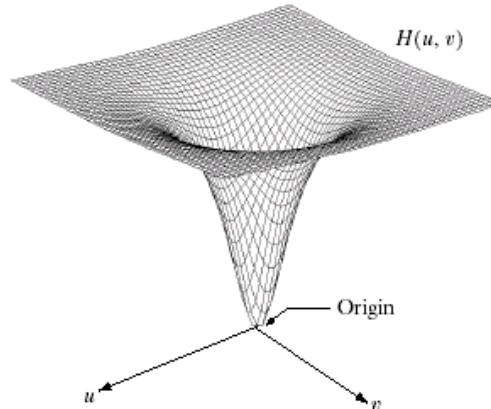


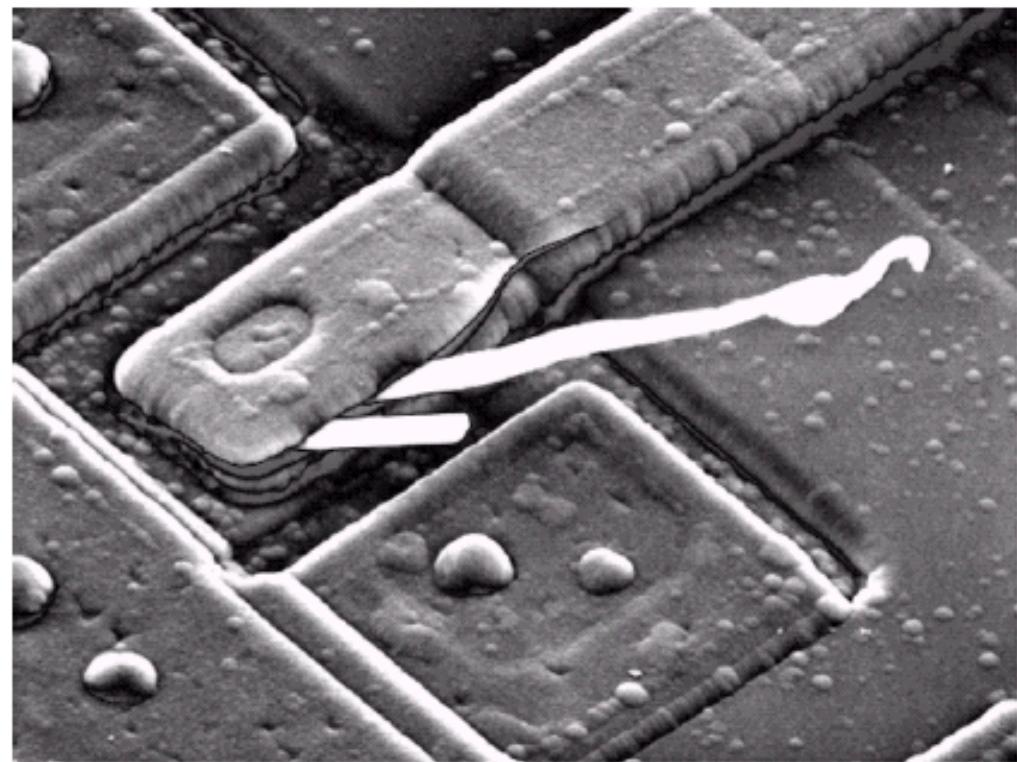
FIGURE 4.7 (a) A two-dimensional lowpass filter function. (b) Result of lowpass filtering the image in Fig. 4.4(a). (c) A two-dimensional highpass filter function. (d) Result of highpass filtering the image in Fig. 4.4(a).

Improved Highpass Output

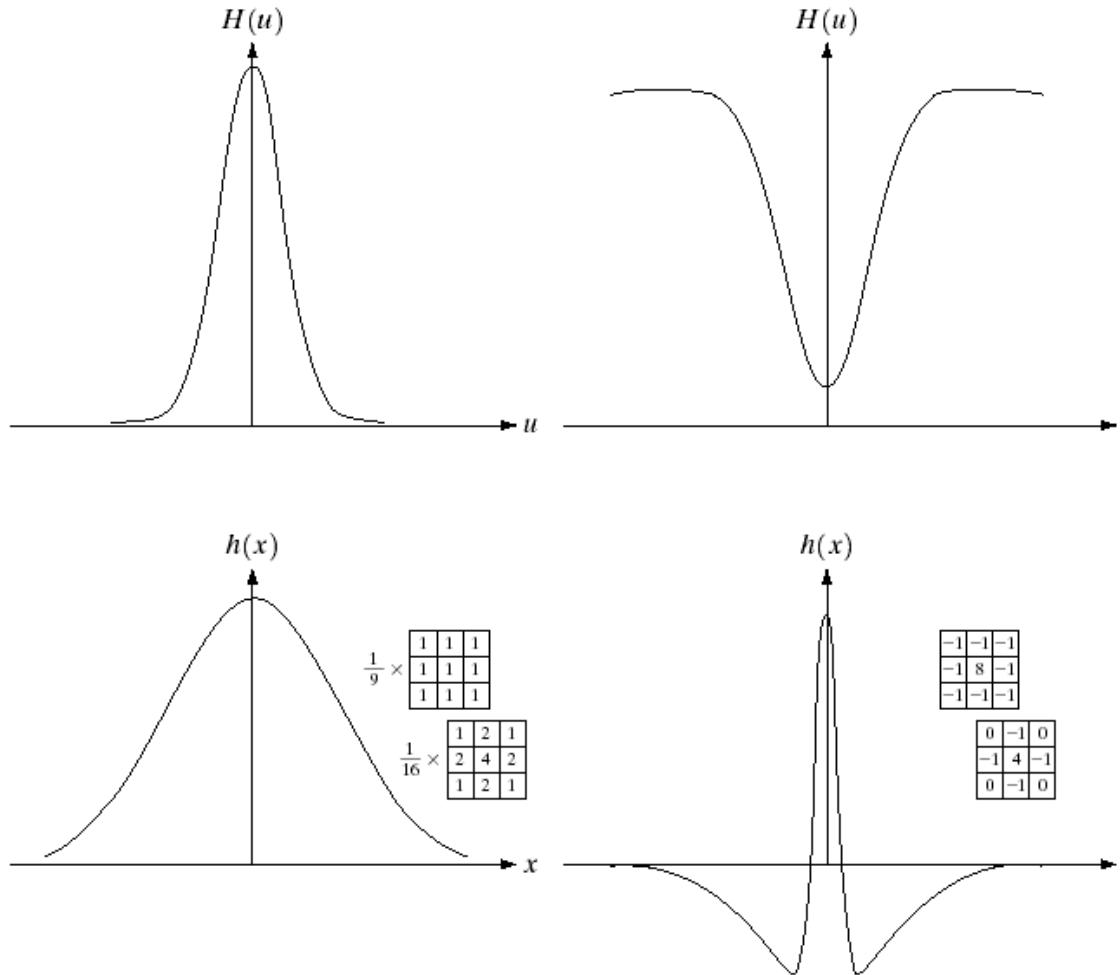
- Add constant to filter so that it will not eliminate $F(0,0)$

FIGURE 4.8

Result of highpass filtering the image in Fig. 4.4(a) with the filter in Fig. 4.7(c), modified by adding a constant of one-half the filter height to the filter function. Compare with Fig. 4.4(a).



Corresponding Filters in the Spatial and Frequency Domains



a	b
c	d

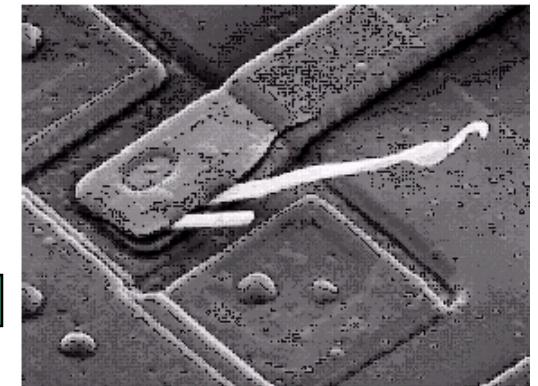
FIGURE 4.9
(a) Gaussian frequency domain lowpass filter.
(b) Gaussian frequency domain highpass filter.
(c) Corresponding lowpass spatial filter.
(d) Corresponding highpass spatial filter. The masks shown are used in Chapter 3 for lowpass and highpass filtering.

Notch Filter

- $F(0,0)$ represents the average value.
- If $F(0,0) = 0$, then average will be 0.
- Achieved by applying a notch filter:

$$H(u,v) = \begin{cases} 0 & \text{if } (u,v) = (M/2, N/2) \\ 1 & \text{otherwise} \end{cases}$$

- In reality the average of the displayed image can't be zero as it needs to have negative gray levels. The output image needs to scale the graylevel.
- Filter has notch (hole) at origin.



Ideal Lowpass Filter

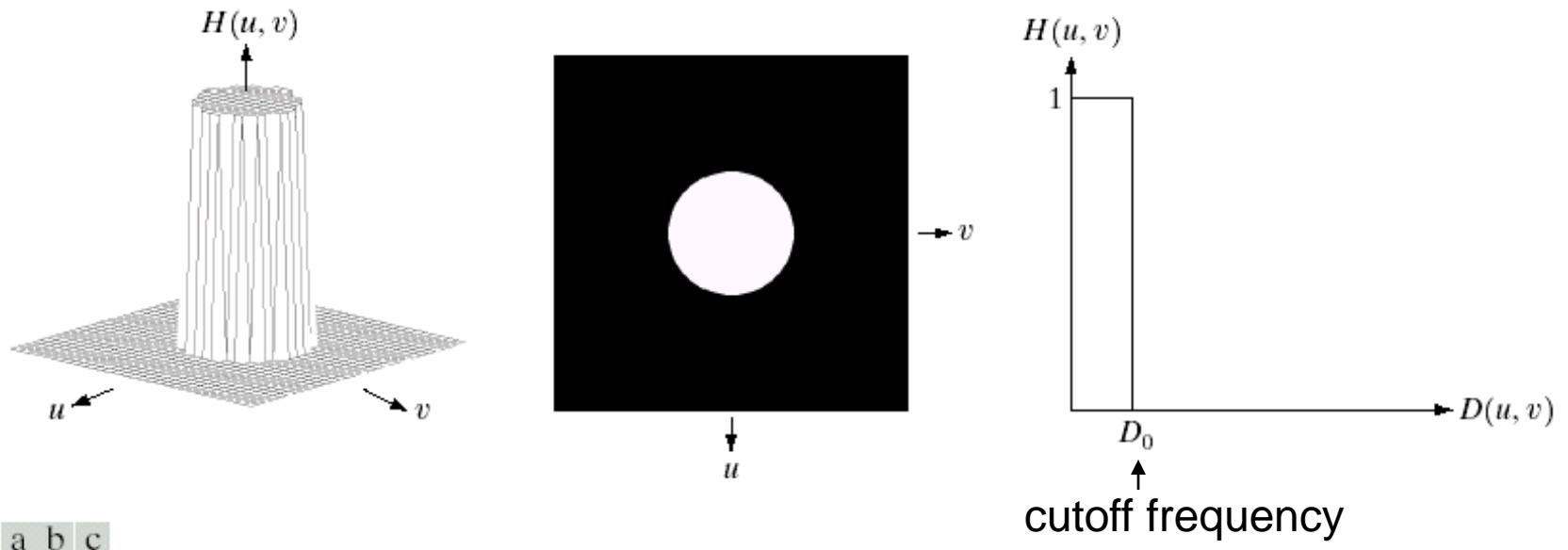
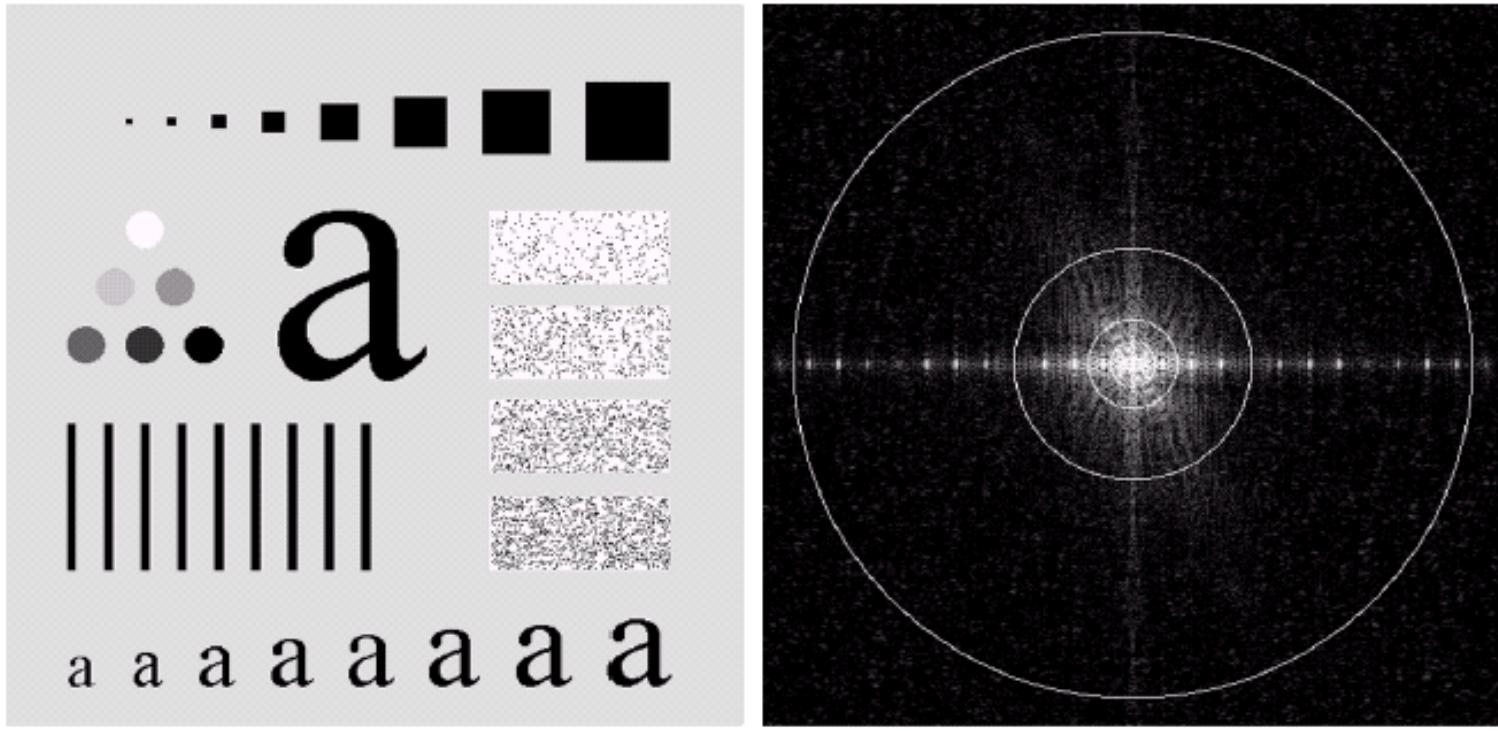


FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Image Power Circles

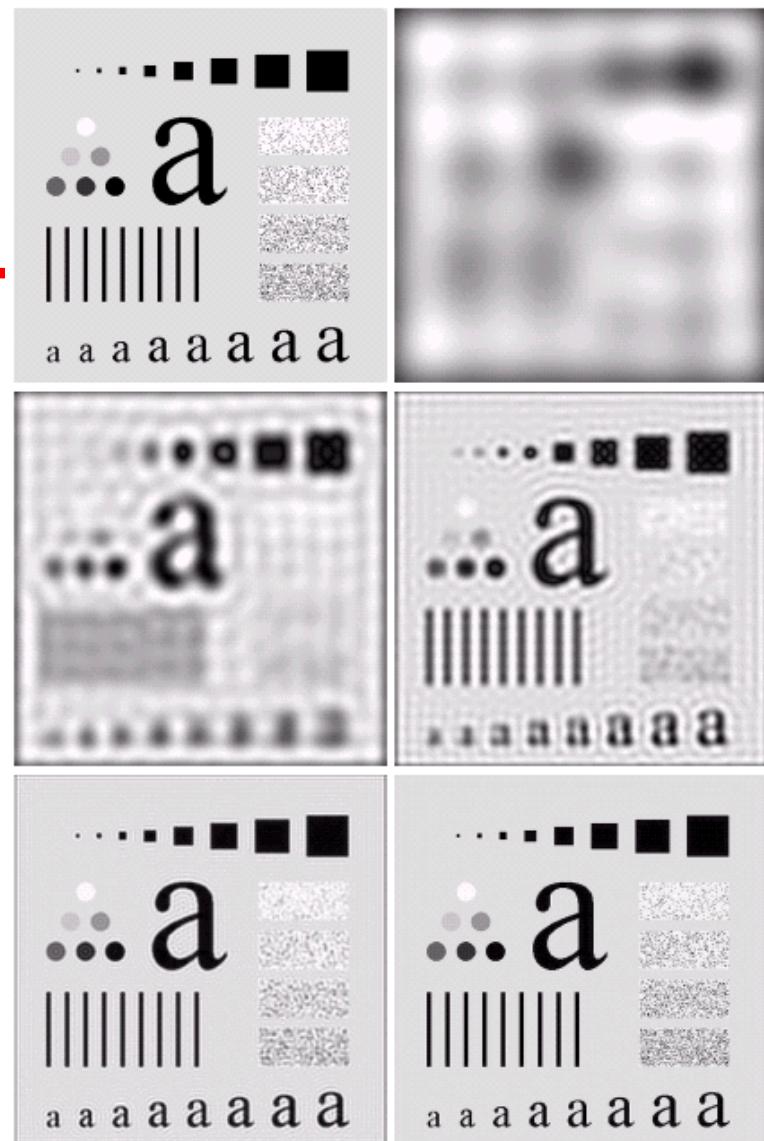


a b

FIGURE 4.11 (a) An image of size 500×500 pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.

Varying ILPF Cutoff Frequencies

- Most sharp detail in this image is contained in the 8% power removed by filter.
- Ringing behavior is a characteristic of ideal filters.
- Little edge info contained in upper 0.5% of spectrum power in this case.



a b
c d
e f

FIGURE 4.12 (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.

ILPF Ringing

- Ringing in spatial filter has two characteristics: dominant component at origin and concentric circular components.
- Center component responsible for blurring.
- Concentric components responsible for ringing.
- Radius of center comp. and #circles/unit distance are inversely proportional to cutoff frequency.

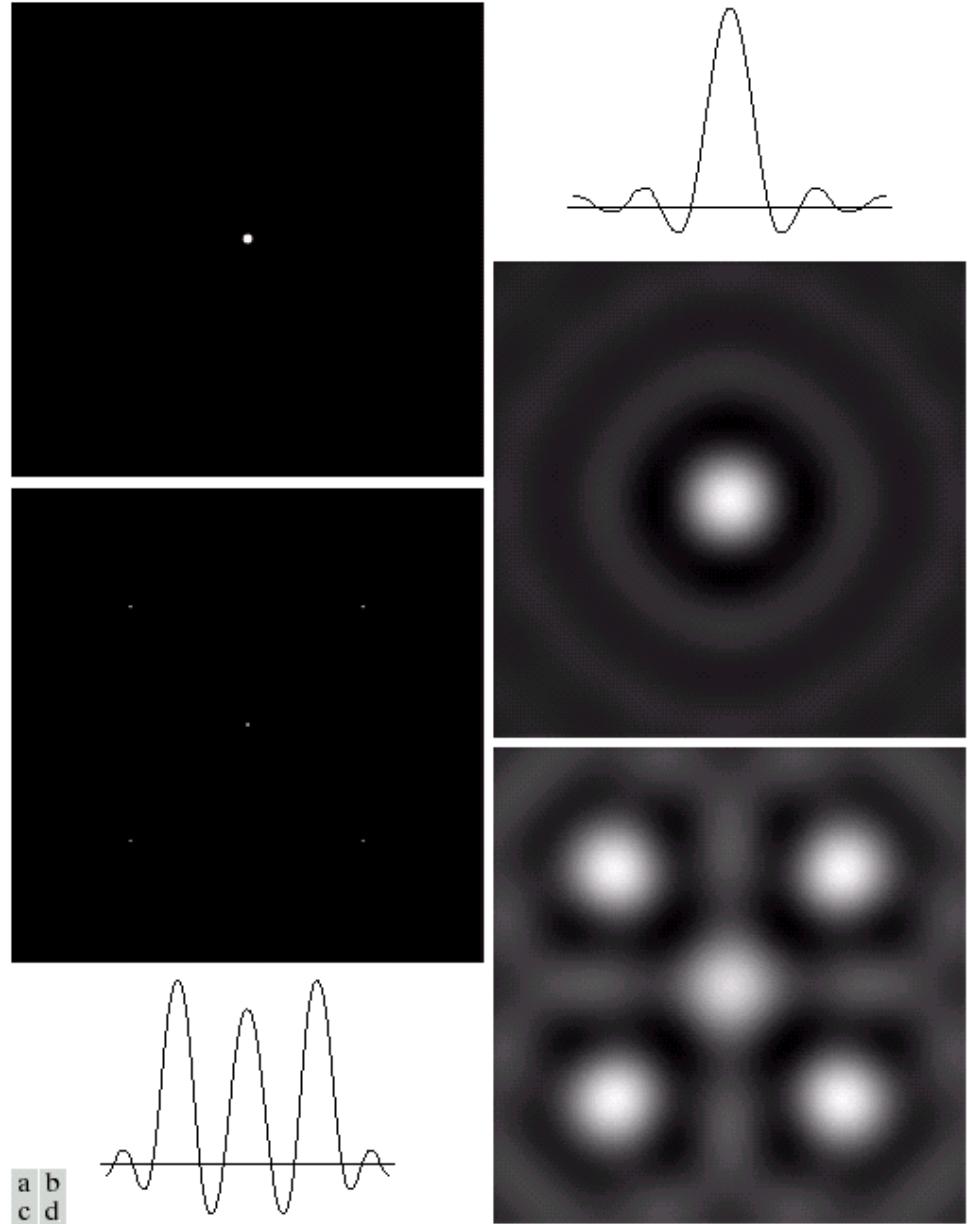
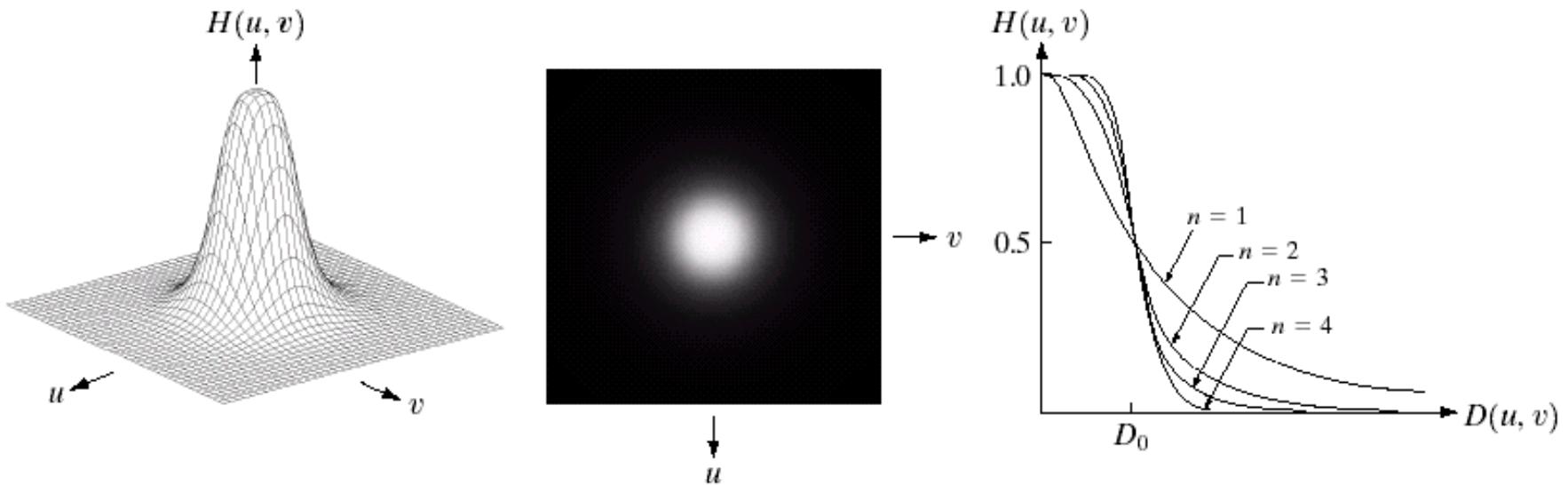


FIGURE 4.13 (a) A frequency-domain ILPF of radius 5. (b) Corresponding spatial filter (note the ringing). (c) Five impulses in the spatial domain, simulating the values of five pixels. (d) Convolution of (b) and (c) in the spatial domain.

Butterworth Lowpass Filter (BLPF)

$$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}} \quad \leftarrow n \text{ is filter order}$$



a b c

FIGURE 4.14 (a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

Varying BLPF Cutoff Frequencies

- Unlike the ILPF, BLPF does not have a sharp discontinuity at the cutoff frequency.
- At D_0 , $H(u,v)$ is down 50% from max value of 1.
- Smooth transition in blurring as cutoff frequency increases.
- No ringing due to filter's smooth transition between low and high frequencies.

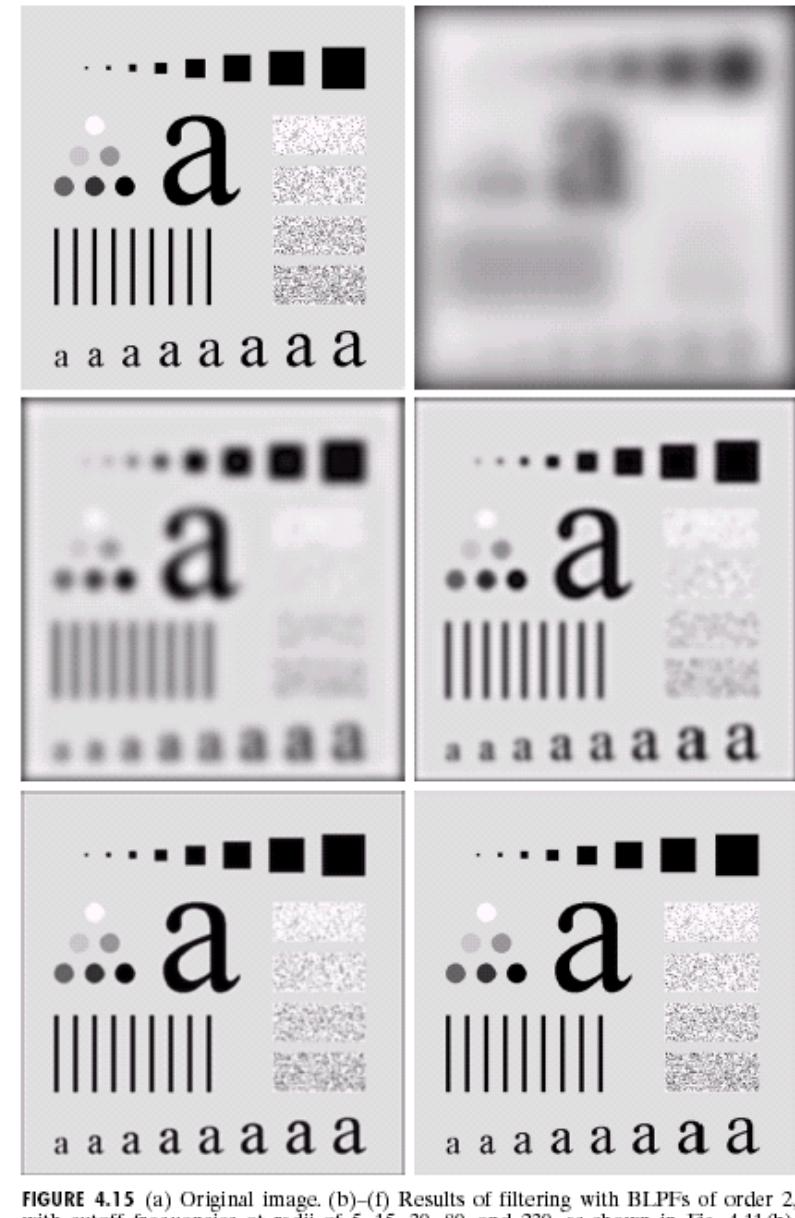


FIGURE 4.15 (a) Original image. (b)-(f) Results of filtering with BLPFs of order 2, with cutoff frequencies at radii of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Fig. 4.12.

Spatial Representation of BLPFs

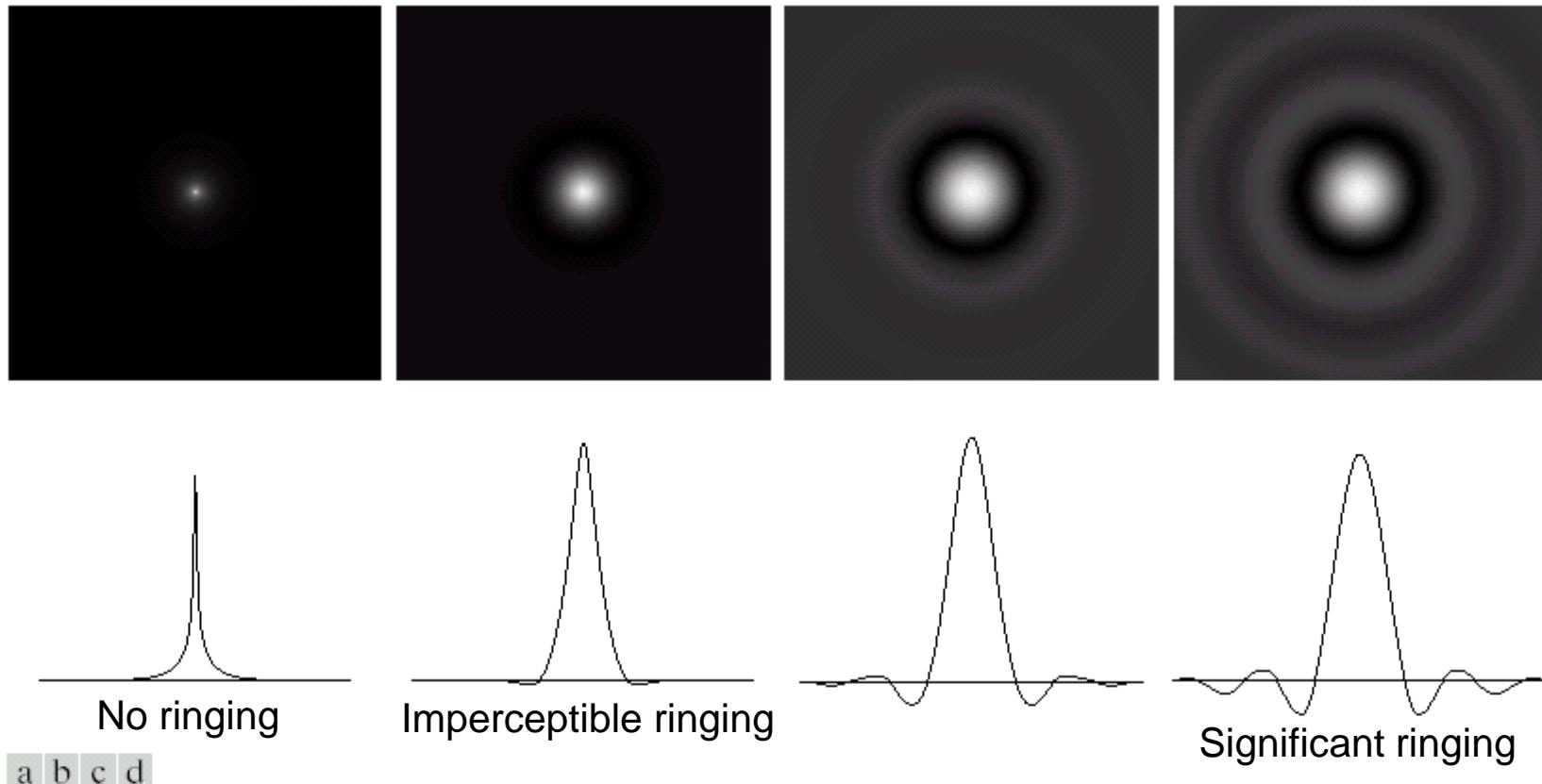
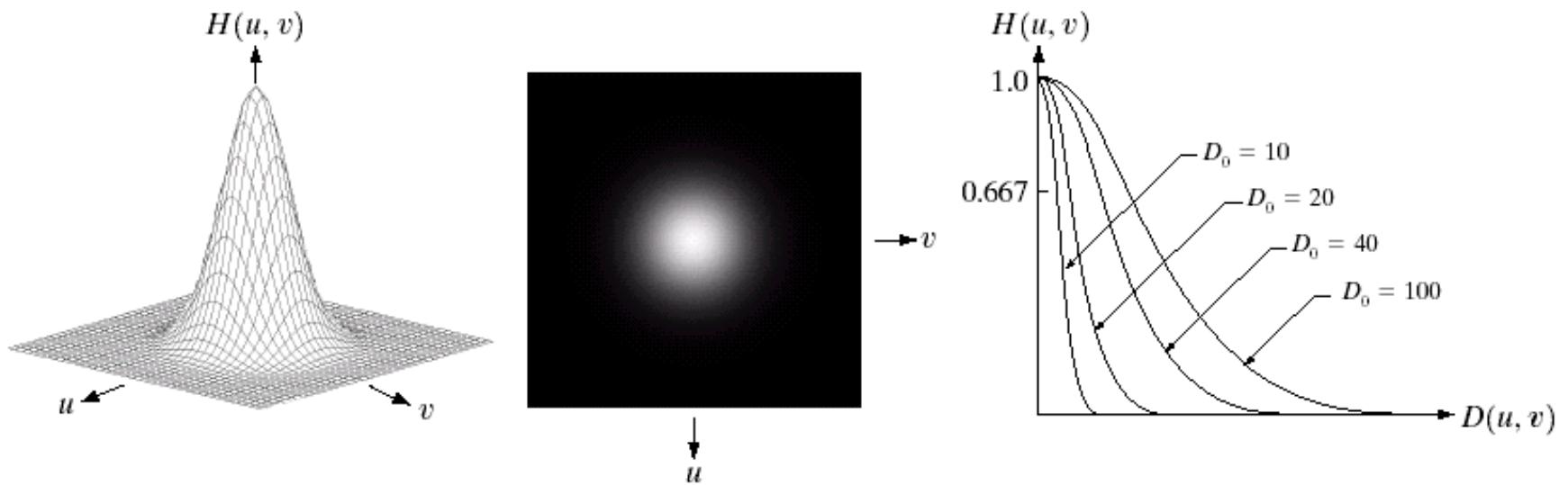


FIGURE 4.16 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.

Gaussian Lowpass Filter (GLPF)

$$H(u, v) = e^{-D^2(u,v)/2D_0^2} \leftarrow \sigma = D_0 = \text{cutoff freq}$$



a b c

FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0 .

Varying GLPF Cutoff Frequencies

- Smooth transition in blurring as cutoff frequency increases.
- GLPF did not achieve as much smoothing as BLPF of order 2 for same cutoff freq.
- GLPF profile is not as tight as profile of the BLPF of order 2.
- No ringing.
- BLPF is more suitable choice if tight control is needed around cutoff frequency. Only drawback: possible ringing.

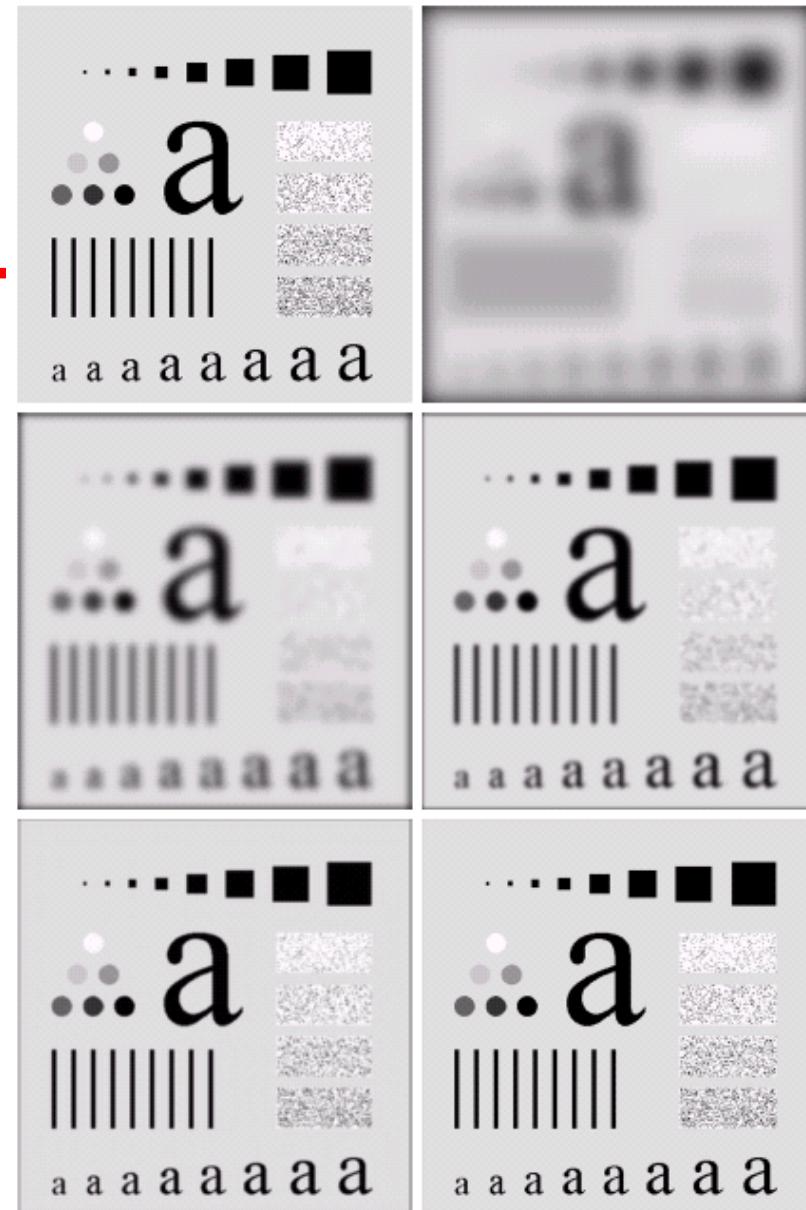


FIGURE 4.18 (a) Original image. (b)–(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.

a b
c d
e f

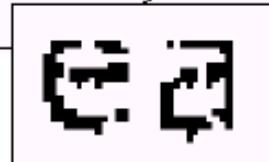
Example (1)

a b

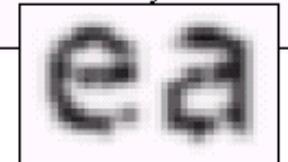
FIGURE 4.19

- (a) Sample text of poor resolution (note broken characters in magnified view).
(b) Result of filtering with a GLPF (broken character segments were joined).

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Example (2)

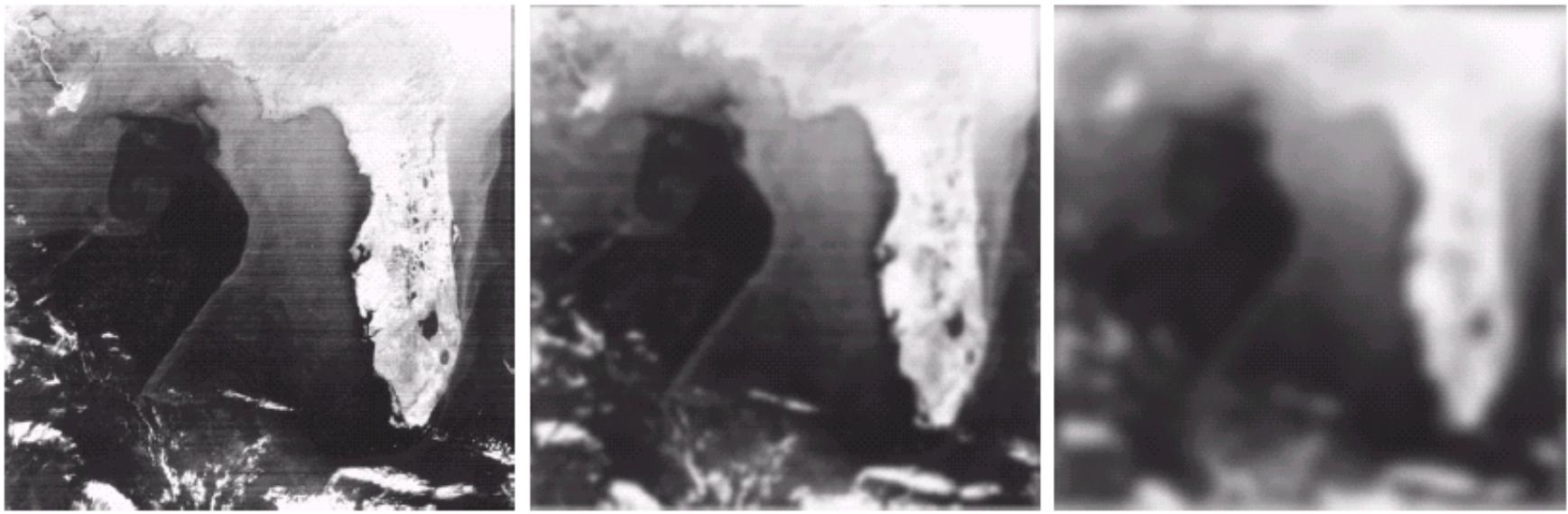


a b c

FIGURE 4.20 (a) Original image (1028×732 pixels). (b) Result of filtering with a GLPF with $D_0 = 100$. (c) Result of filtering with a GLPF with $D_0 = 80$. Note reduction in skin fine lines in the magnified sections of (b) and (c).

Example (3)

- Crude but simple way of removing prominent scan lines.
- Blurs out detail while leaving large features recognizable.
- Averages out features smaller than the ones of interest.



a b c

FIGURE 4.21 (a) Image showing prominent scan lines. (b) Result of using a GLPF with $D_0 = 30$. (c) Result of using a GLPF with $D_0 = 10$. (Original image courtesy of NOAA.)

Highpass Filters: 1-H_{LP}

Ideal highpass filter

$$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$$

Butterworth highpass filter

$$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}}$$

Gaussian highpass filter

$$H(u, v) = 1 - e^{-D^2(u, v)/2D_0^2}$$

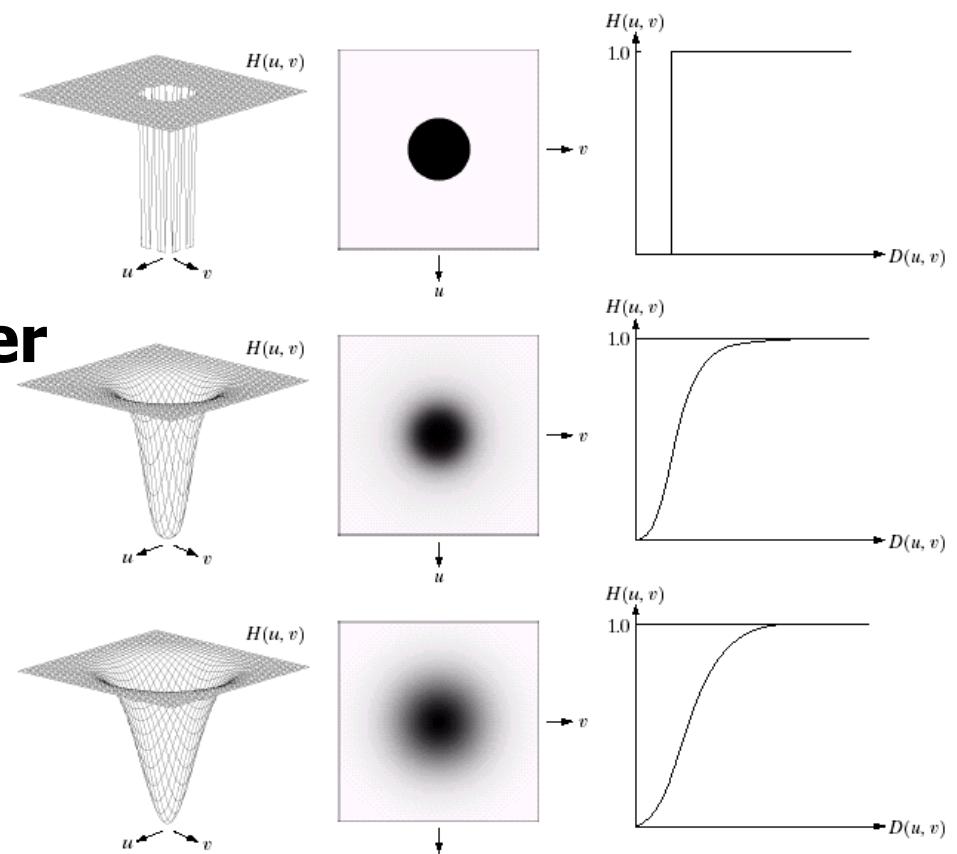


FIGURE 4.22 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Spatial Representation of Filters

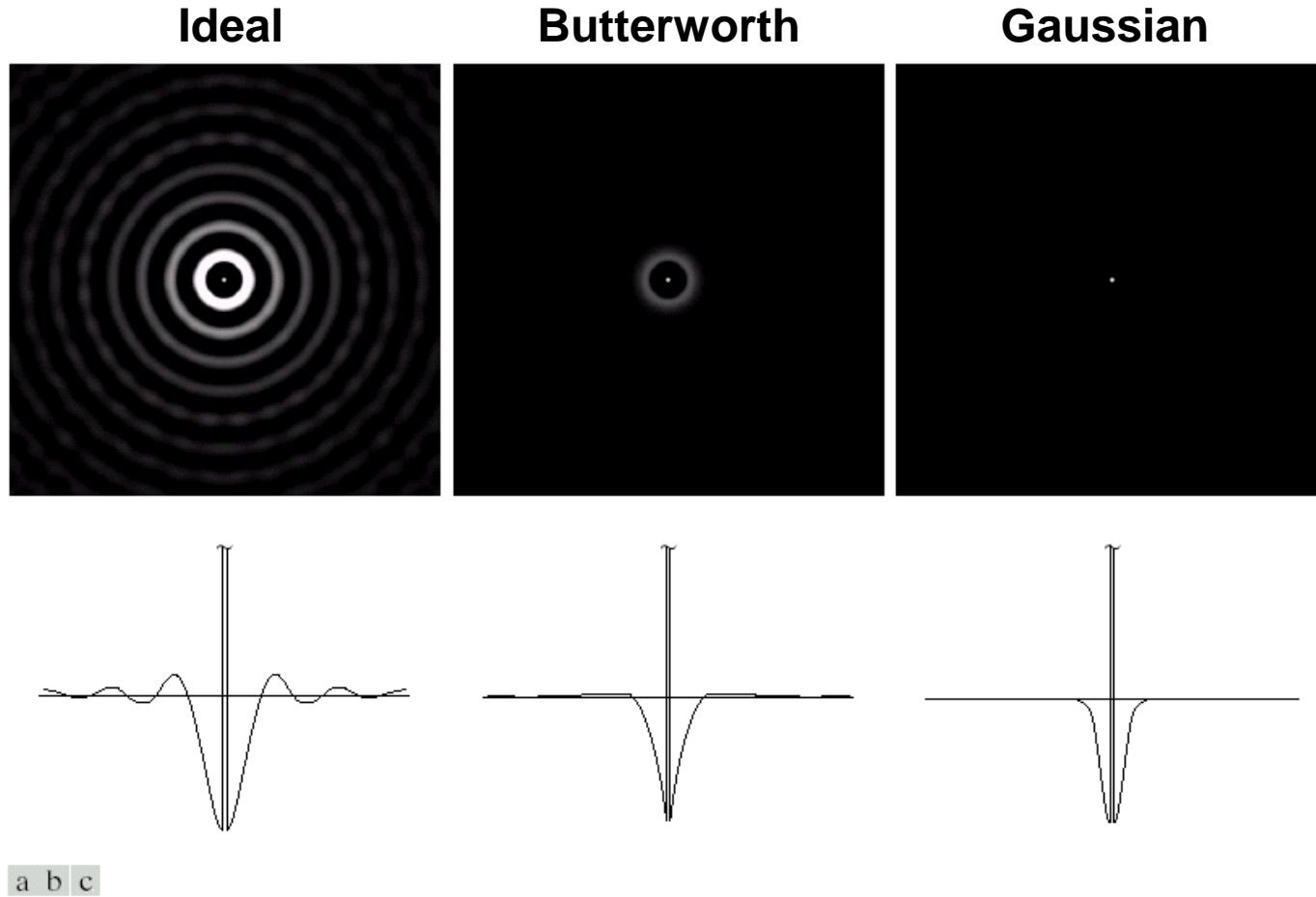
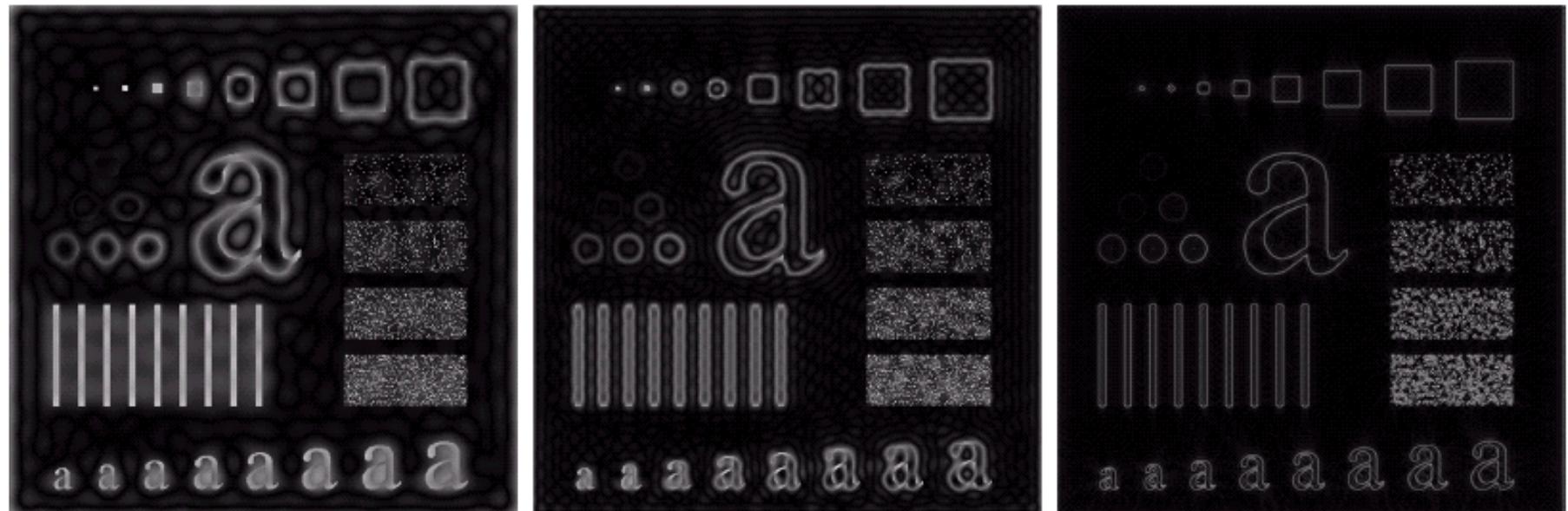


FIGURE 4.23 Spatial representations of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding gray-level profiles.

Varying IHPF Cutoff Frequencies



a b c

FIGURE 4.24 Results of ideal highpass filtering the image in Fig. 4.11(a) with $D_0 = 15, 30$, and 80 , respectively. Problems with ringing are quite evident in (a) and (b).

Varying BHPF Cutoff Frequencies

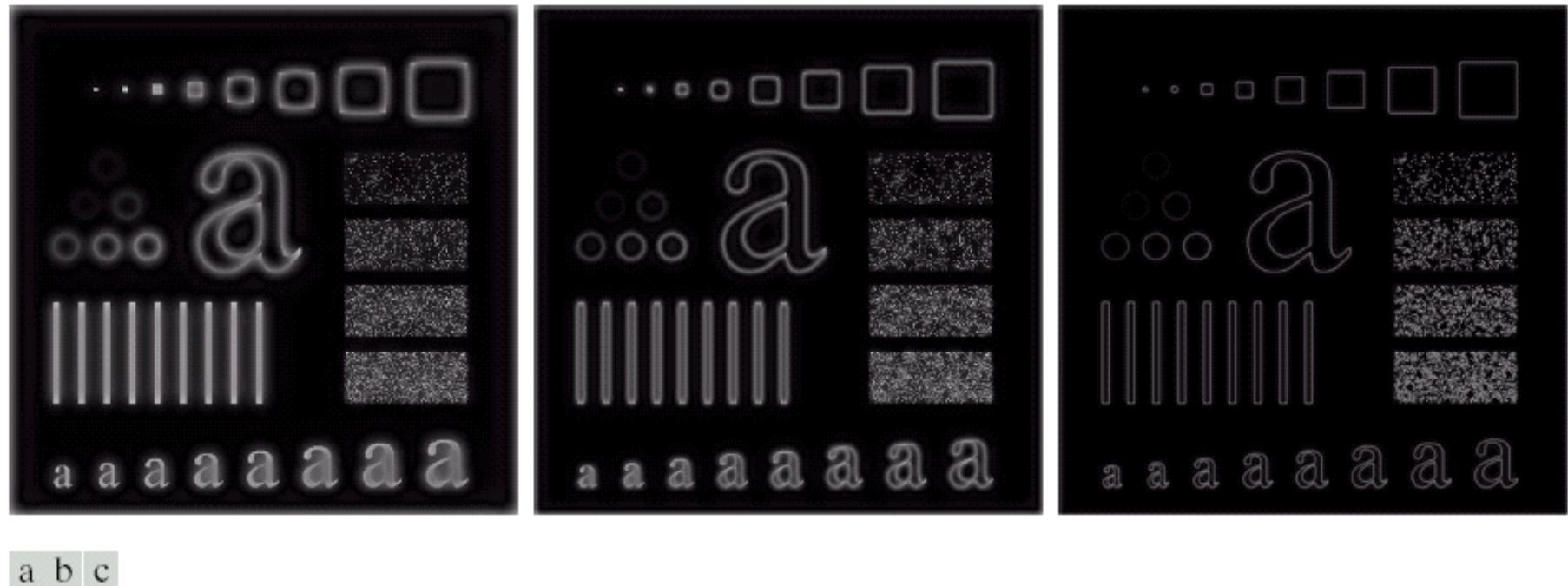


FIGURE 4.25 Results of highpass filtering the image in Fig. 4.11(a) using a BHPF of order 2 with $D_0 = 15$, 30, and 80, respectively. These results are much smoother than those obtained with an ILPF.

Varying GHPF Cutoff Frequencies

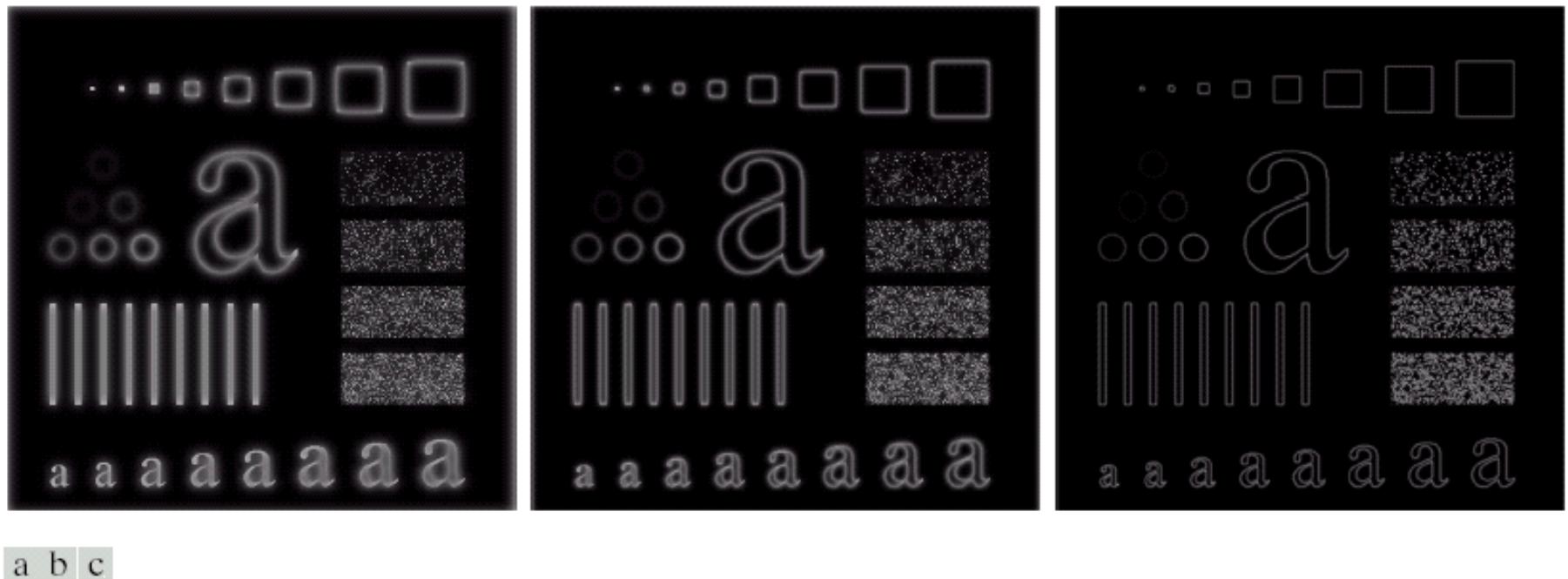
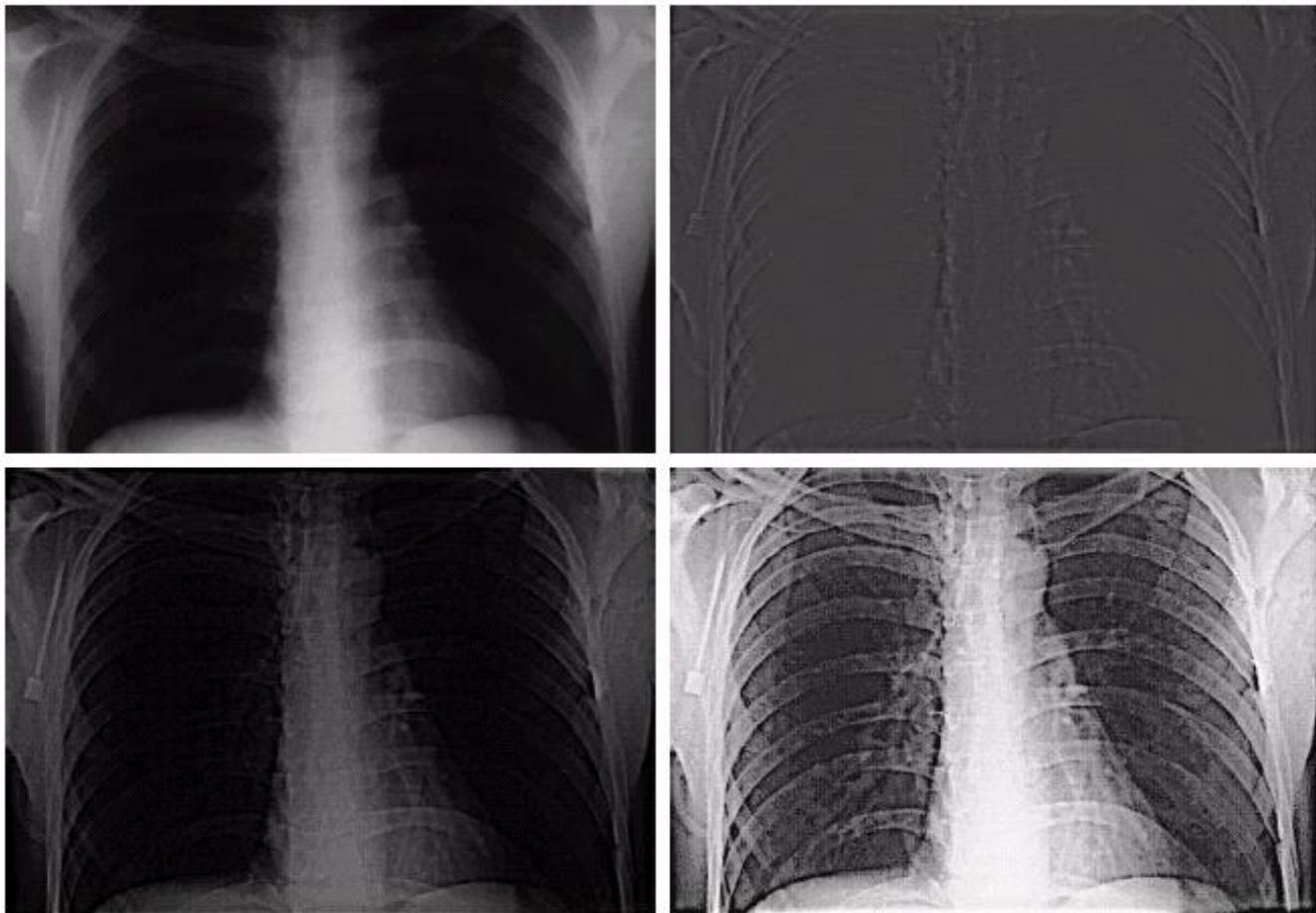


FIGURE 4.26 Results of highpass filtering the image of Fig. 4.11(a) using a GHPF of order 2 with $D_0 = 15$, 30, and 80, respectively. Compare with Figs. 4.24 and 4.25.

Example



a b
c d

FIGURE 4.30

(a) A chest X-ray image. (b) Result of Butterworth highpass filtering. (c) Result of high-frequency emphasis filtering. (d) Result of performing histogram equalization on (c). (Original image courtesy Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

Homomorphic Filtering (1)

- Consider an image to be expressed as:

$$f(x,y) = i(x,y) r(x,y)$$

↑ ↑
illumination reflectance

- Take the logarithm of $f(x,y)$ to separate $i(x,y)$ and $r(x,y)$:

$$\ln f(x,y) = \ln i(x,y) + \ln r(x,y)$$

- Illumination component: slow spatial variations
- Reflectance component: tends to vary abruptly
- Filter out low frequencies to remove illumination effects.

Homomorphic Filtering (2)

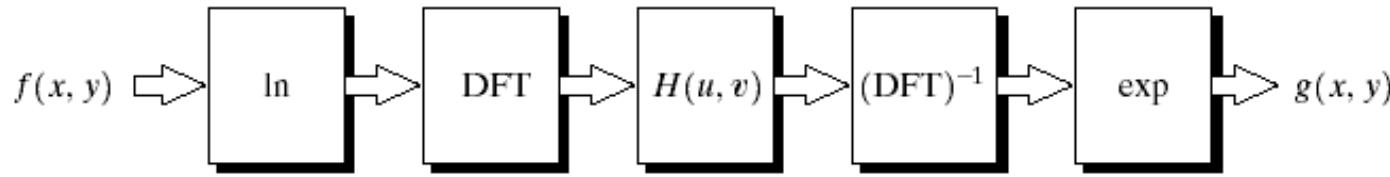


FIGURE 4.31
Homomorphic filtering approach for image enhancement.

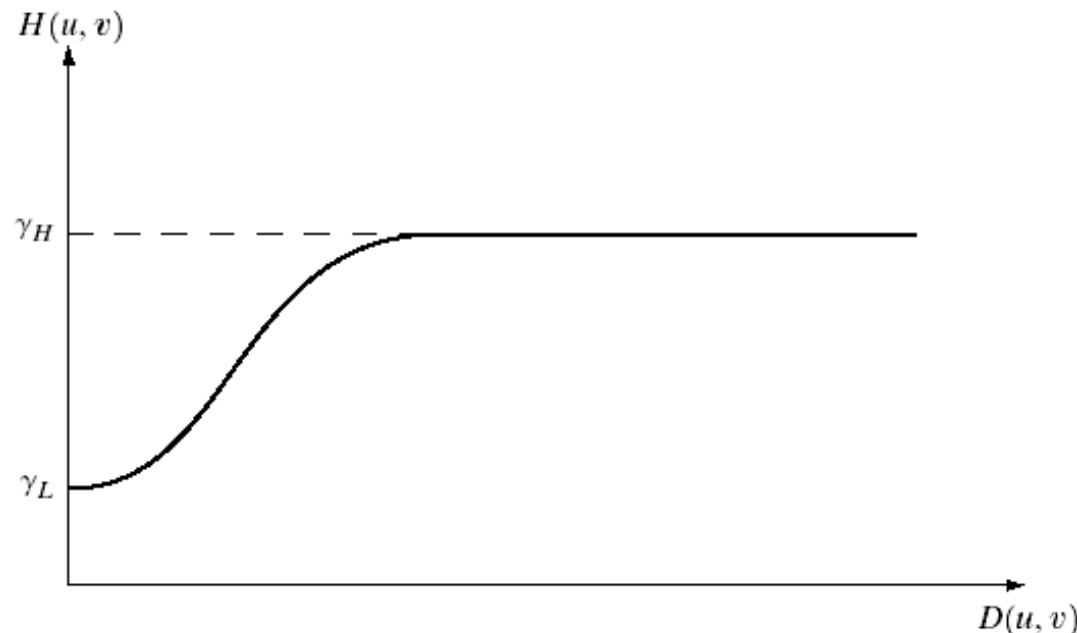


FIGURE 4.32
Cross section of a circularly symmetric filter function. $D(u, v)$ is the distance from the origin of the centered transform.

Example

- Interior details are obscured by glare from outside walls.
- Reduction of dynamic range in brightness, together with an increase in contrast, brought out interior details and balanced the graylevels of the outside wall.

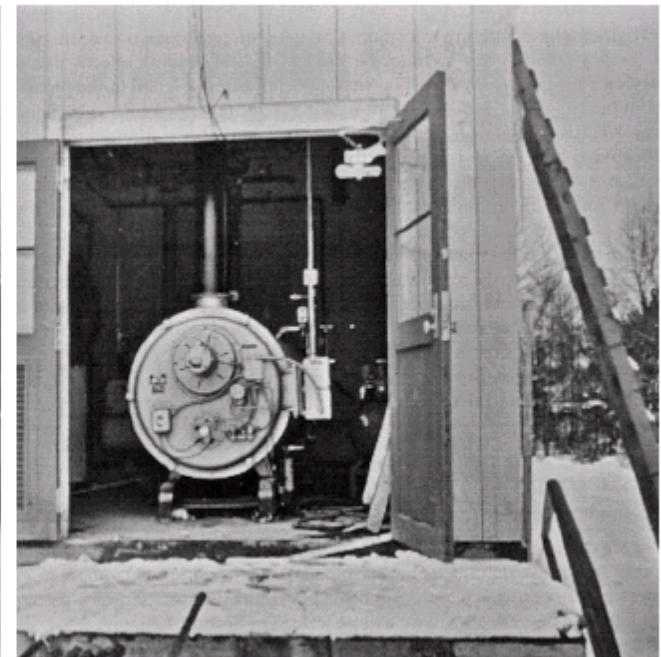
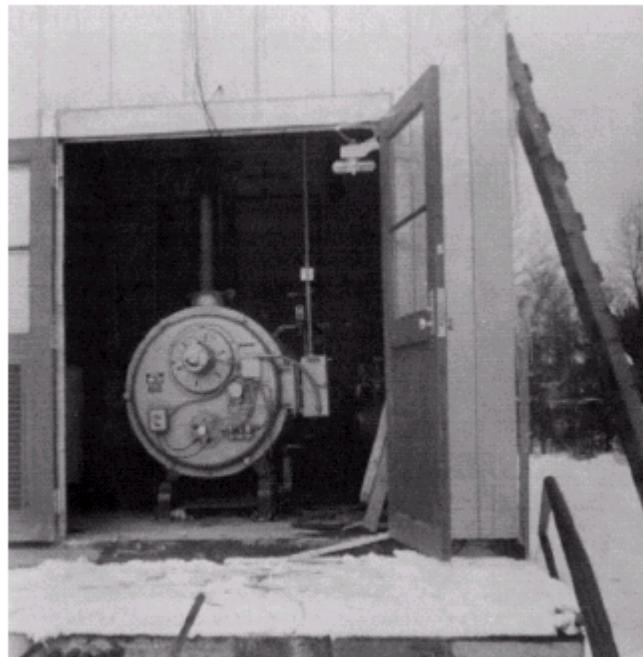
a b

FIGURE 4.33

(a) Original image. (b) Image processed by homomorphic filtering (note details inside shelter).
(Stockham.)

$$\gamma_L = 0.5$$

$$\gamma_H = 2.0$$

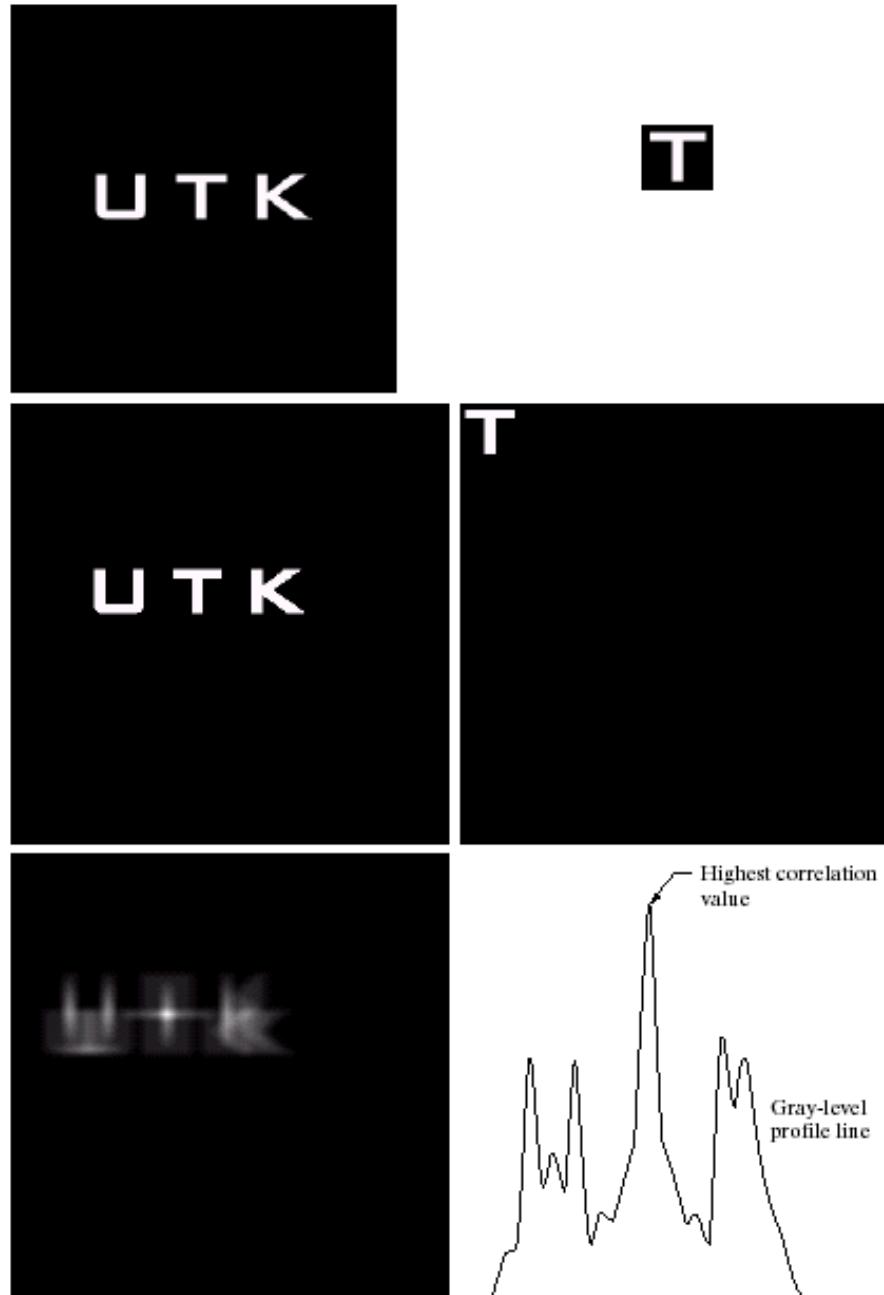


Correlation

- Identical to convolution except that kernel is not flipped.
- Kernel is now referred to as the template.
- It is the pattern that we seek to find in the larger image.

$$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v)H(u, v)$$

Example (1)



a b
c d
e f

FIGURE 4.41
(a) Image.
(b) Template.
(c) and
(d) Padded
images.
(e) Correlation
function displayed
as an image.
(f) Horizontal
profile line
through the
highest value in
(e), showing the
point at which the
best match took
place.

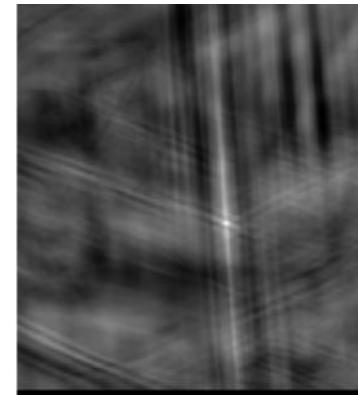
Example (2)



Reference image



Template



Correlation image

Peak corresponds to best match location

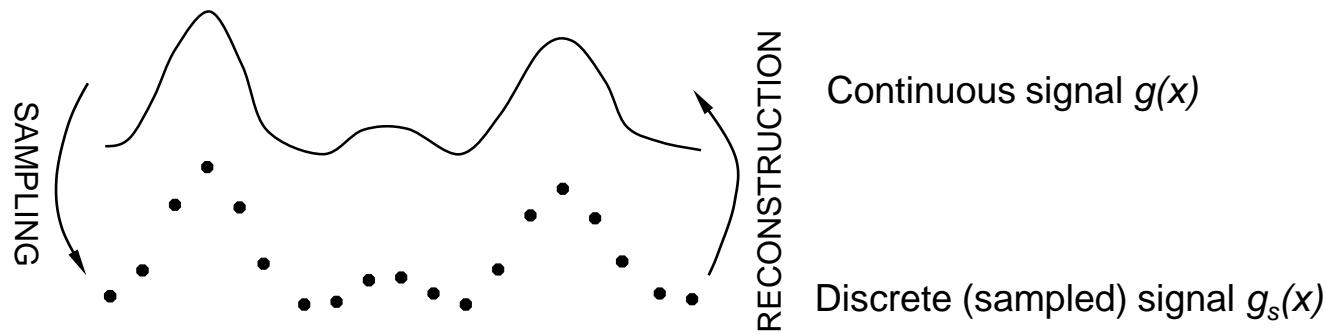
Sampling Theory

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we describe sampling theory:
 - Motivation
 - Analysis in the spatial and frequency domains
 - Ideal and nonideal solutions
 - Aliasing and antialiasing
 - Example: oversampling CD players

Sampling Theory



Sampling theory addresses the two following problems:

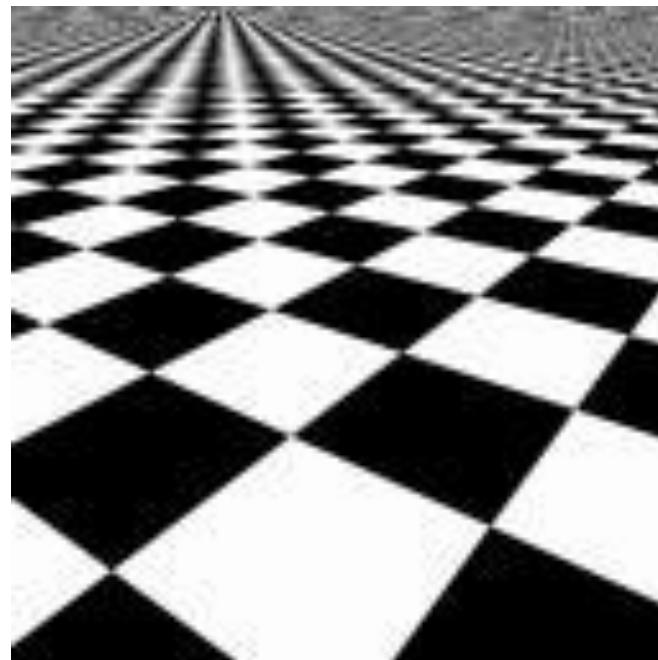
1. Are the samples of $g_s(x)$ sufficient to exactly describe $g(x)$?
2. If so, how can $g(x)$ be reconstructed from $g_s(x)$?

Motivation

- Improper consideration leads to jagged edges in magnification and Moire effects in minification.

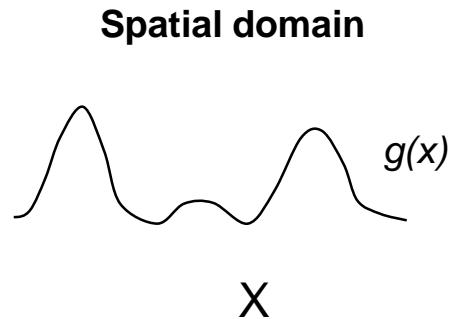


Unfiltered image

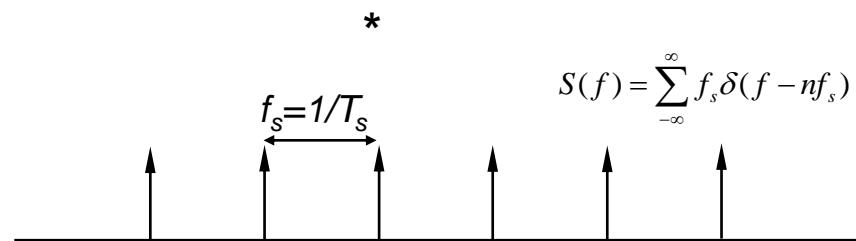
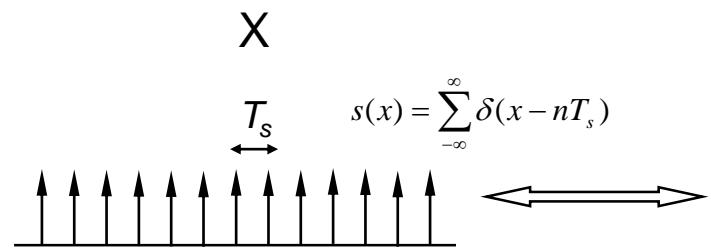
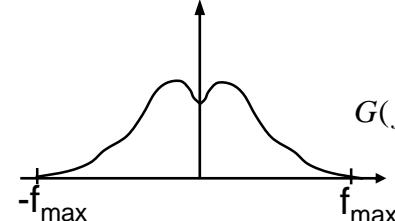


Filtered image

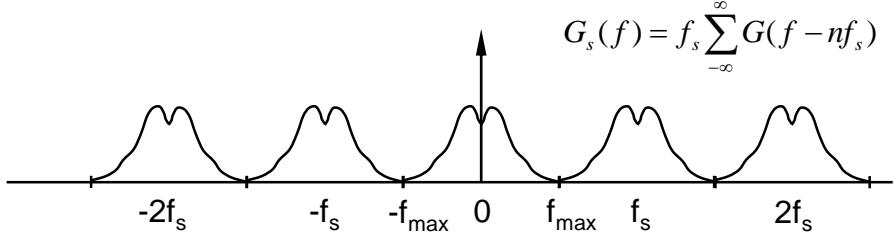
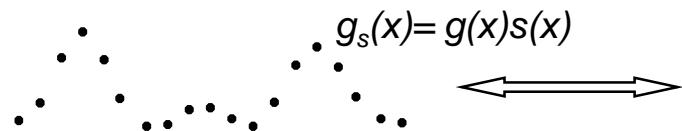
Insight: Analysis in Frequency Domain



Frequency domain

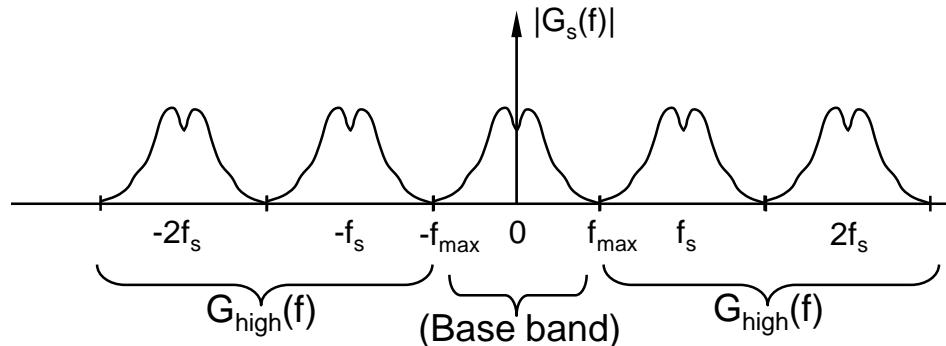


=



Exploit Well-Known Properties of Fourier Transforms

1. Multiplication in spatial domain \leftrightarrow convolution in frequency domain.
2. Fourier transform of an impulse train is itself an impulse train.
3. $G_s(f)$ is the original spectrum $G(f)$ replicated in the frequency domain with period f_s .



$$G_s(f) = G(f) + G_{high}(f) \rightarrow \text{introduced by sampling}$$

Solution to Reconstruction

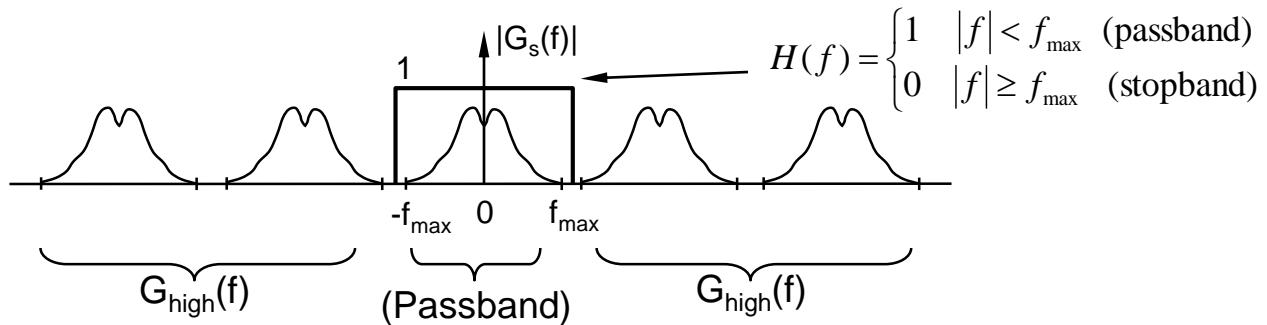
Discard replicas $G_{high}(f)$, leaving baseband $G(f)$ intact.

Two Provisions for exact reconstruction:

- The signal must be bandlimited. Otherwise, the replicas would overlap with the baseband spectrum.
- $f_s > 2f_{\max}$ (Nyquist frequency)

Ideal Filter in the Frequency Domain: Box Filter

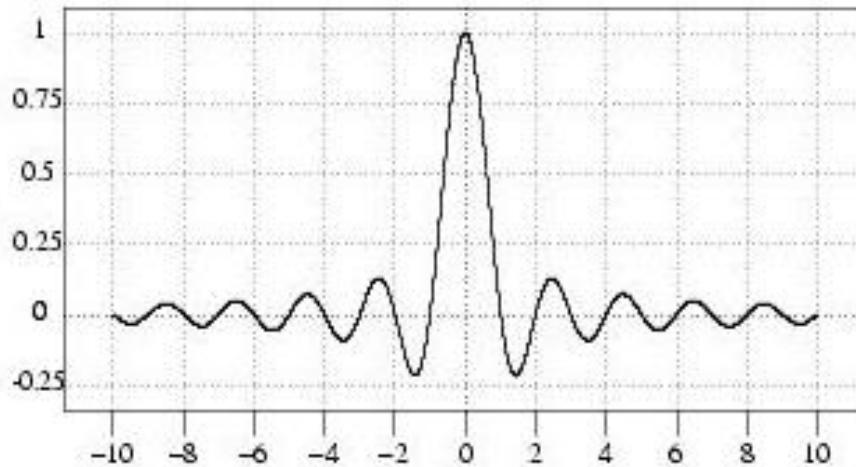
- Assuming provisions hold, we can discard G_{high} by multiplying $G_s(f)$ with a box filter in the frequency domain.



Ideal Filter in the Spatial Domain: Sinc Function

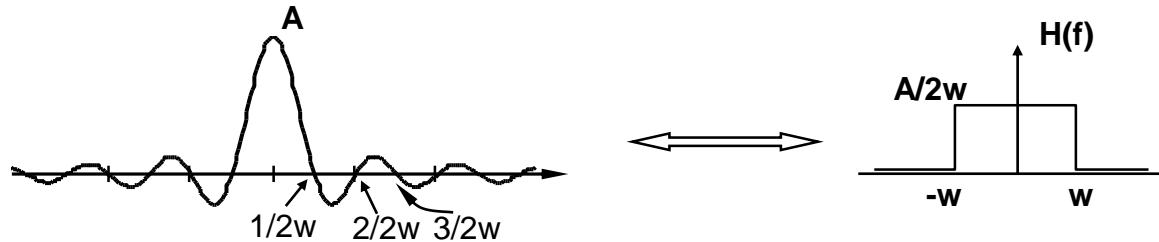
The ideal lowpass filter in the spatial domain is the inverse Fourier transform of the ideal box:

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$



Reciprocal Relationship

- Reciprocal relationship between spatial and frequency domains:



- For interpolation, A must equal 1 (unity gain when sinc is centered at samples; pass through 0 at all other samples)

$W=f_{\max}=0.5$ cycle / pixel for all digital images

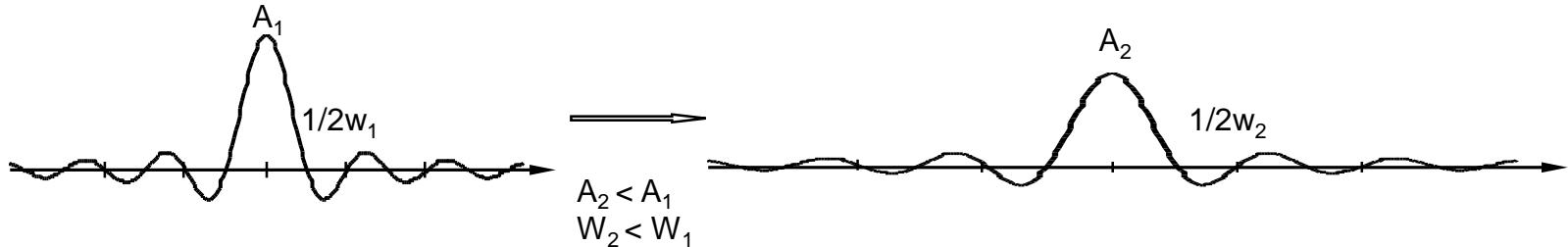


Highest freq: on-off sequence

1 cycle is 2 pixels (black, white, black, white), therefore $\frac{1}{2}$ cycle per pixel is f_{\max} .

Blurring

- To blur an image, $W \downarrow$ and $A \downarrow$ so that $A/2W$ remains at 1.



- To interpolate among sparser samples, $A=1$ and $W \downarrow$ which means that $(A/2W) \uparrow$. Since sampling decreases the amplitude of the spectrum, $(A/2W) \uparrow$ serves to restore it.

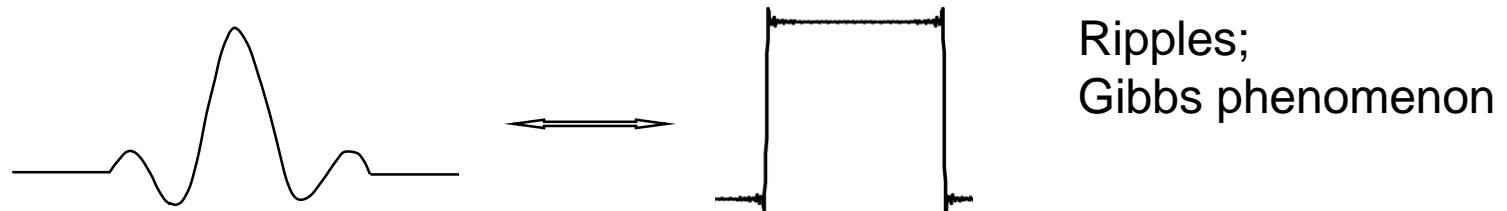
$$\begin{aligned}g(x) &= \text{sinc}(x) * g_s(x) \\&= \int_{-\infty}^{\infty} \text{sinc}(\lambda) g_s(x - \lambda) d\lambda\end{aligned}$$

Note : sinc requires infinite number of neighbours :impossible .

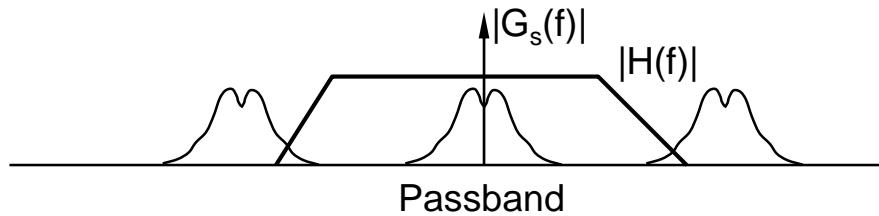
- Ideal low-pass filtering is impossible in the spatial domain. It is not impossible in the frequency domain for finite data streams.

Nonideal Reconstruction

- Possible solution to reconstruction in the spatial domain: truncated sinc.



- Alternate solution: nonideal reconstruction



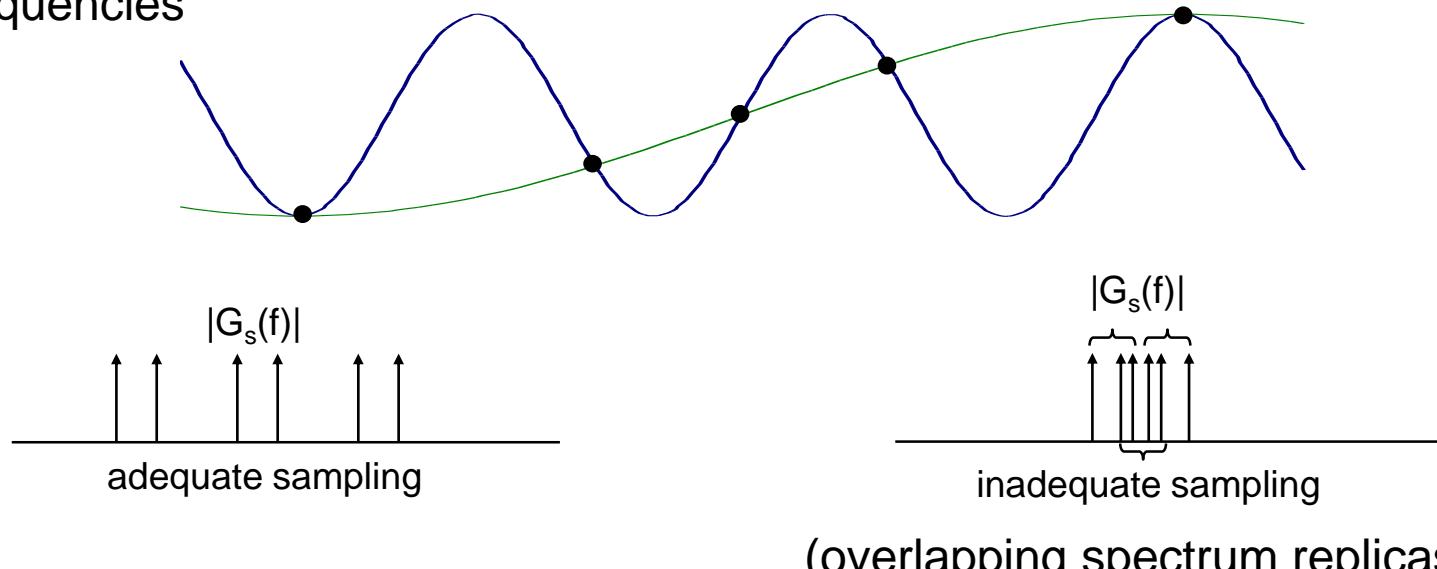
- Nonideal because it attenuates the higher frequencies in the passband and doesn't fully suppress the stopband.

Aliasing

If $f_s < 2f_{\max}$ our signal is undersampled.

If f_{\max} is infinite, our signal is not bandlimited.

Either way, we have aliasing: high frequencies masquerade, or alias as, low frequencies

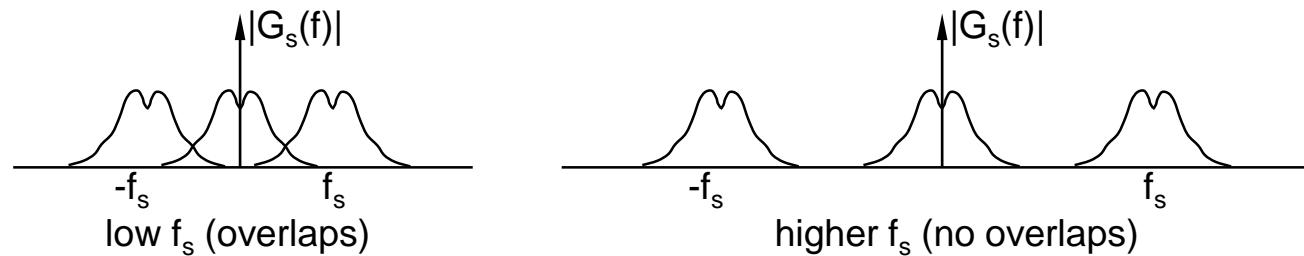


Temporal aliasing: stagecoach wheels go backwards in films
(sampled at 24 frames/sec).

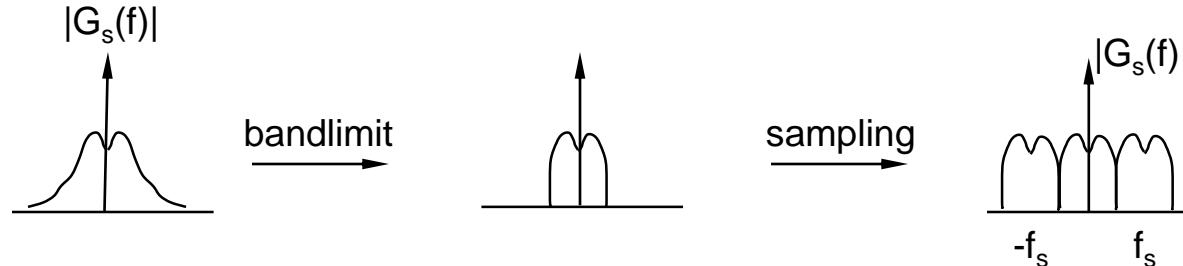
Antialiasing

Two approaches to antialiasing for combating aliasing:

1. Sample input at higher rate: increase f_s



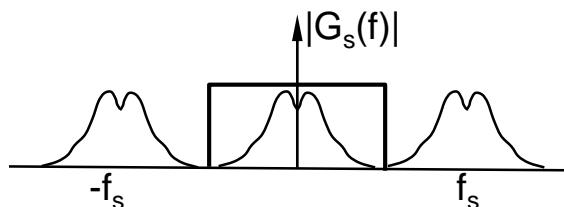
2. bandlimit input before sampling at low f_s



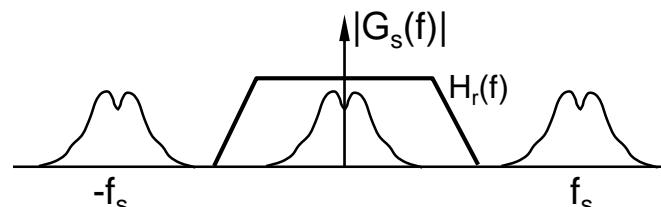
Antialiasing

Intuition for increasing f_s :

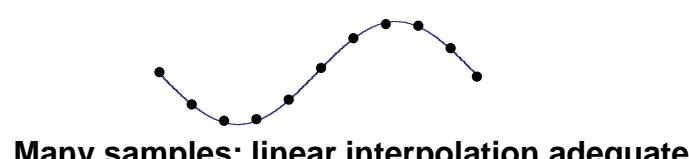
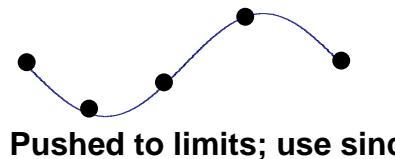
Higher sampling rates permit sloppier reconstruction filters.



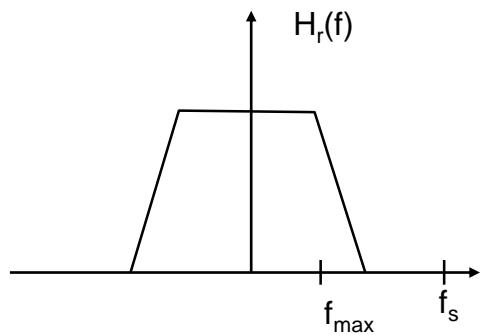
Marginally acceptable f_s
Requires ideal lowpass filter



high f_s allows nonideal reconstruction filter.
This adequate because $H_r(f)$ doesn't
taper off until $|F| > |F_{\max}|$



Nonideal Reconstruction Filter



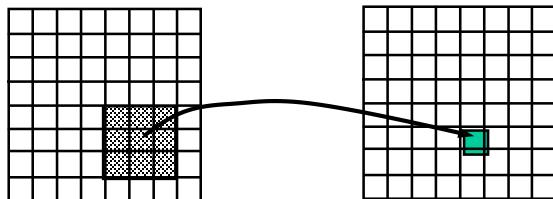
$$H_r(f) = \begin{cases} 1 & 0 \leq |f| \leq f_{\max} \\ \neq 0 & f_{\max} < |f| < f_s - f_{\max} \\ 0 & |f| \geq f_s - f_{\max} \end{cases}$$

Nonideal Reconstruction Filter

Intuition for bandlimiting input:

Inability to change f_s : e.g., restricted by display resolution.

For example, an image in perspective may require infinite pixels to capture all visual details. However, only finite # pixels available, e.g. 512×512

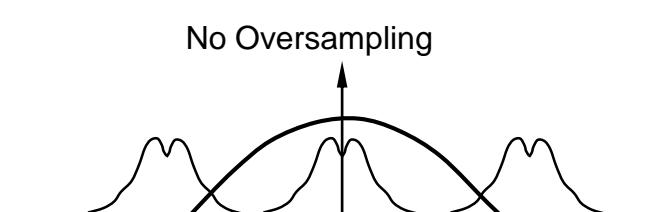


Many pixels in oblique image map to one output pixel. Must blur neighborhood before returning a single value to the output image. In general, dull signals don't need bandlimiting, rich ones do.

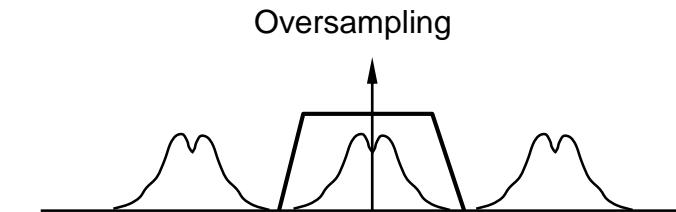
Example: Oversampling CD players

$$f_s = 44.1 \text{ kHz (sample/second)}$$

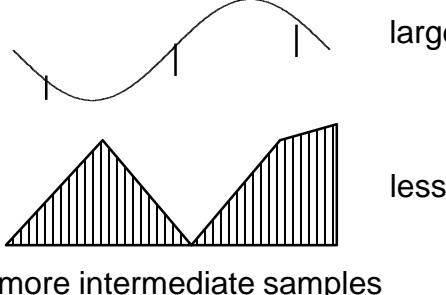
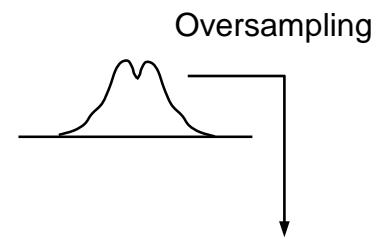
$$= f_{\max} \text{ Nyq rate} * 2 + 4.1 \text{ kHz cushion} = 44.1 \text{ KHz}$$



Sloppy reconstruction by
Analog output filter
(speaker)

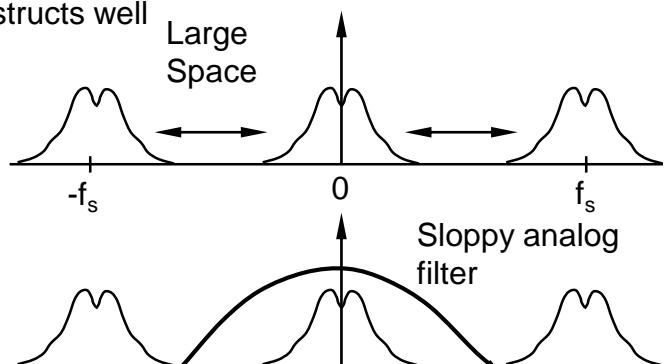


Accurate digital
Filter in CD
Player reconstructs well

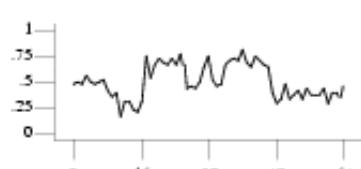
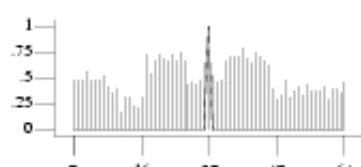
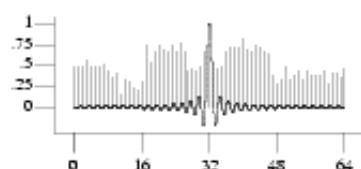
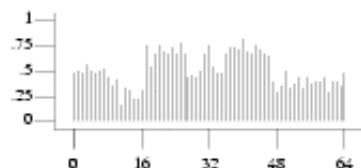
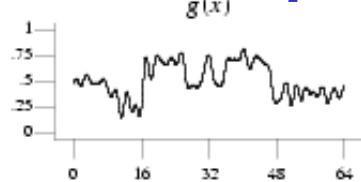


large overshoot/undershoots

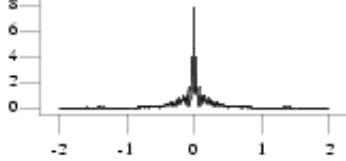
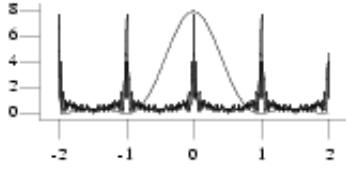
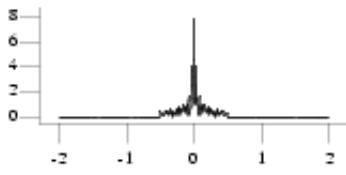
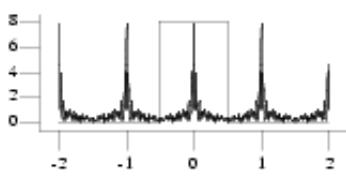
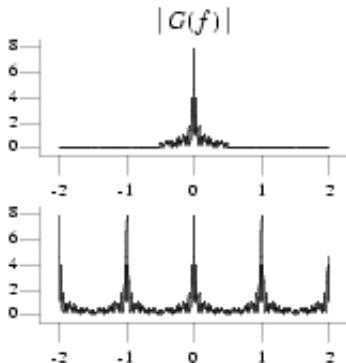
less



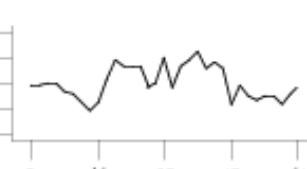
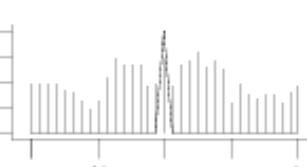
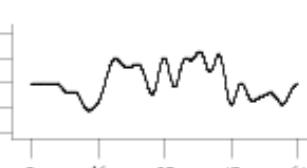
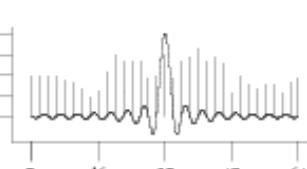
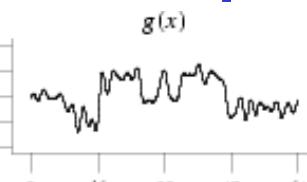
Adequate Sampling



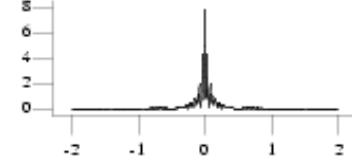
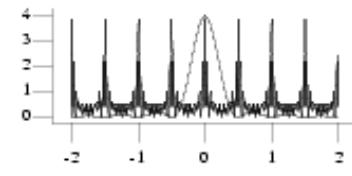
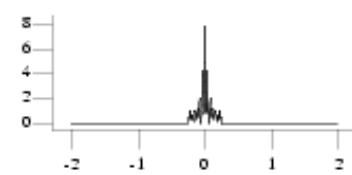
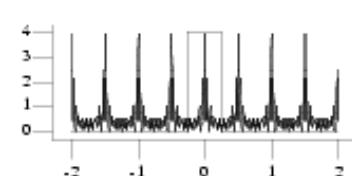
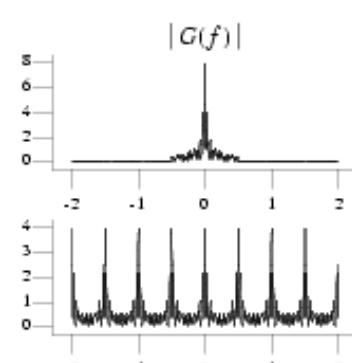
$|G(f)|$



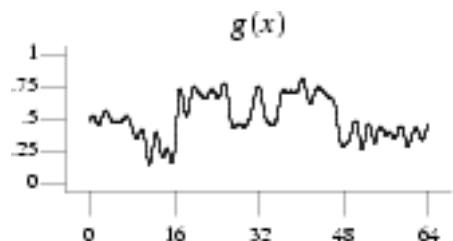
Inadequate Sampling



$|G(f)|$



Antialiasing



$$|G(f)|$$

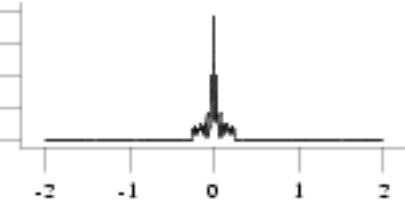
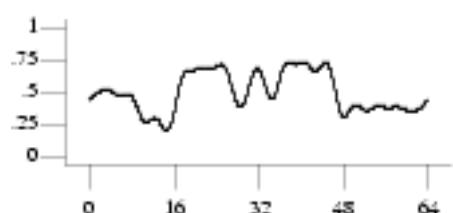
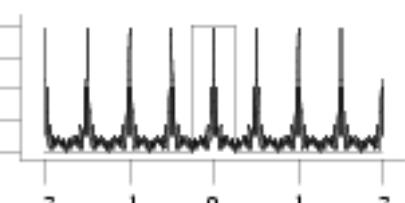
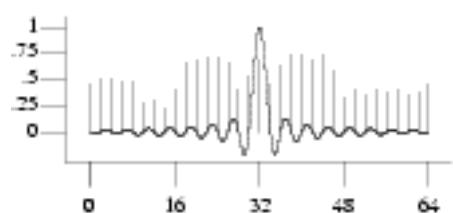
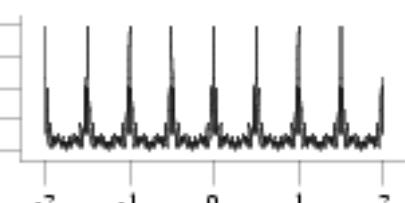
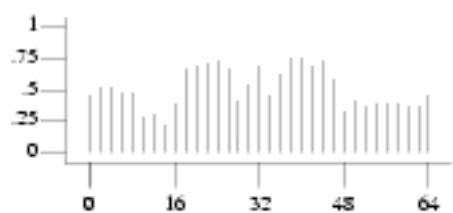
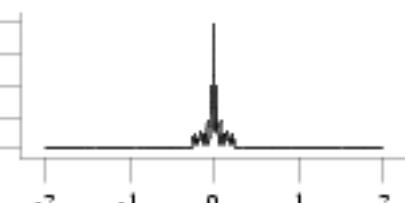
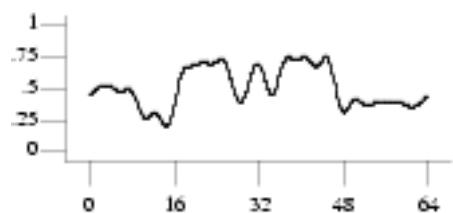
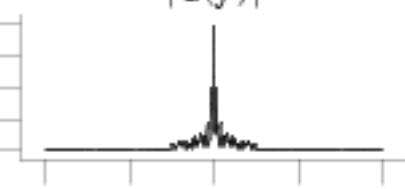


Image Resampling

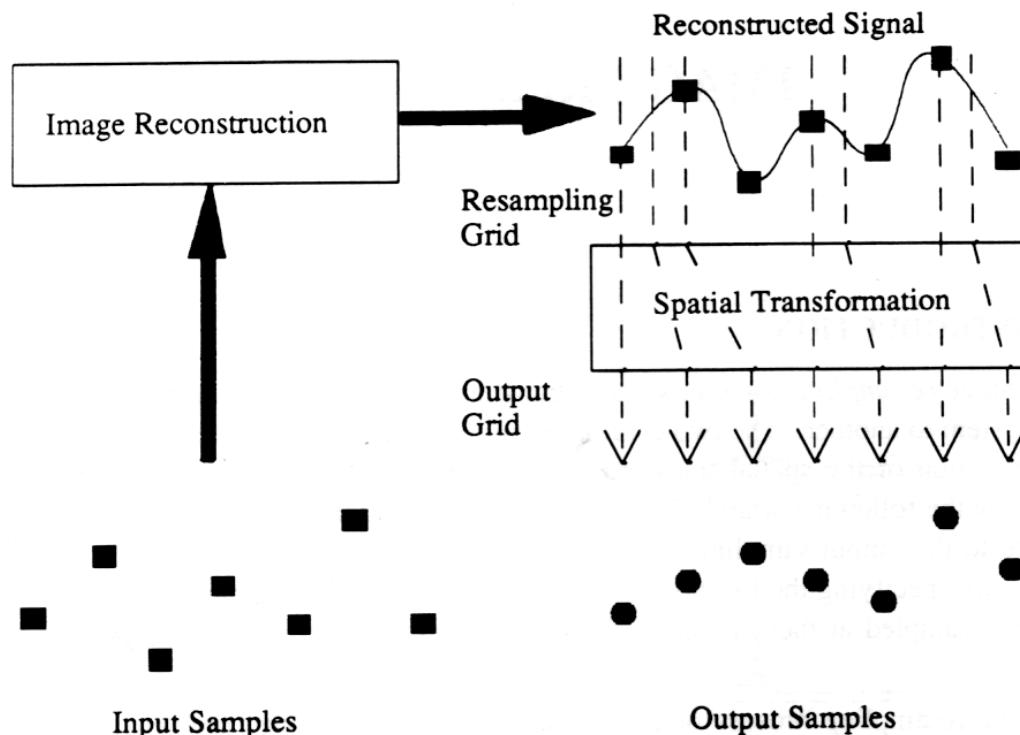
Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we review image resampling:
 - Ideal resampling
 - Mathematical formulation
 - Resampling filter
 - Tradeoffs between accuracy and complexity
 - Software implementation

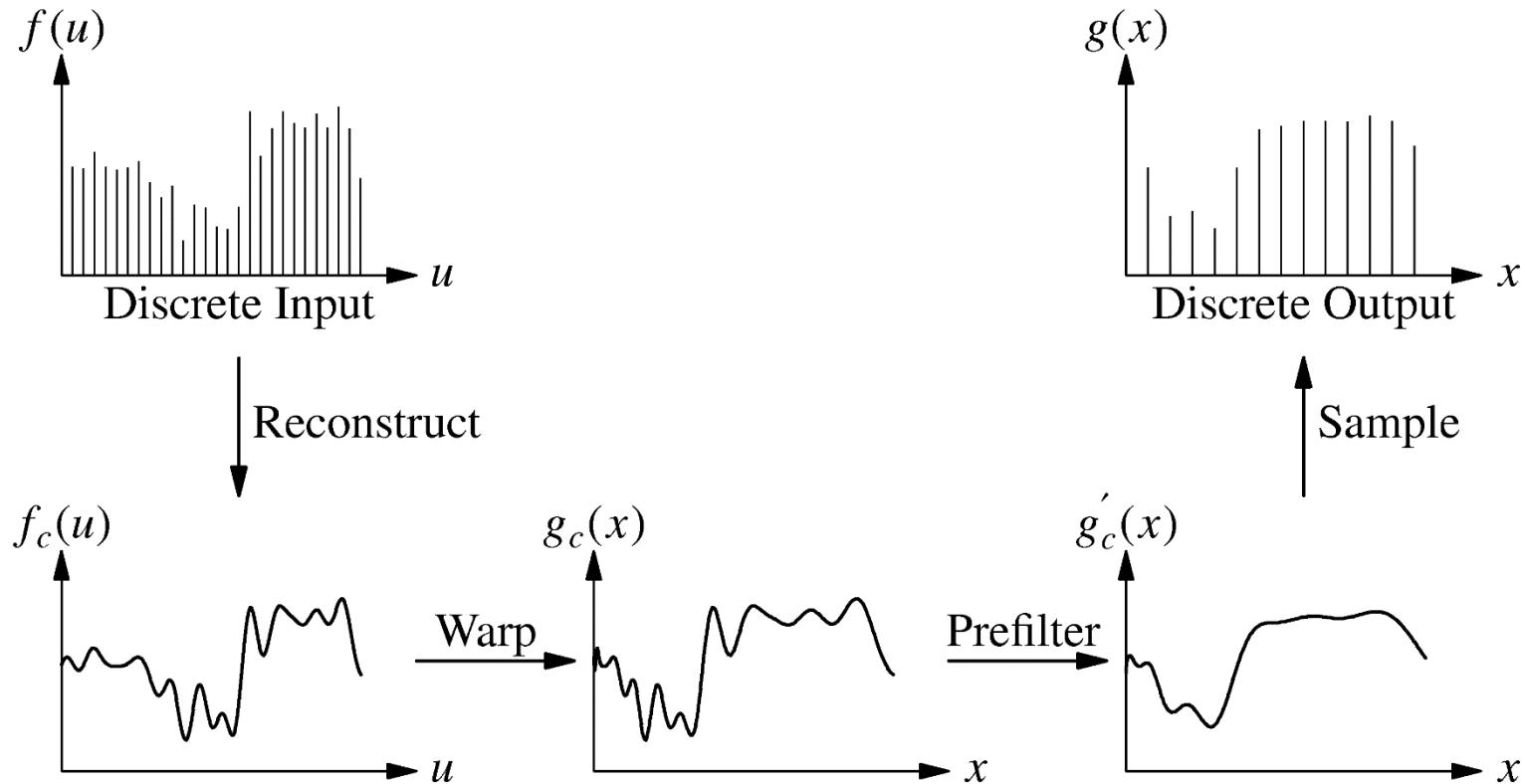
Definition

- Image Resampling: Process of transformation a sampled image from one coordinate system to another.



Ideal Resampling

- Ideal Resampling: reconstruction, warping, prefiltering, sampling



Mathematical Formulation (1)

Stages :

Discrete Input

Math Definition :

$$f(u), u \in z$$

Reconstruction Input

$$f_c = f(u) * r(u) = \sum_{k \in z} f(k)r(u - k)$$

Warped Signal

$$g_c(x) = f_c(m^{-1}(x))$$

Continuous Output

$$\dot{g_c}(x) = g_c(x) * h(x) = \int g_c(t)h(x - t)dt$$

Discrete Output

$$g(x) = \dot{g_c}(x)S(x)$$

Two filtering components : reconstruction and prefiltering (bandlimiting warped signal before sampling). Cascade them into one by working backwards from $g(x)$ to $f(u)$:

$$g(x) = \dot{g_c}(x) \quad \text{for } x \in z$$

$$g(x) = \int f_c(m^{-1}(t))h(x - t)dt = \int \left[\sum_{k \in z} f(k)r(m^{-1}(t) - k) \right] h(x - t)dt$$

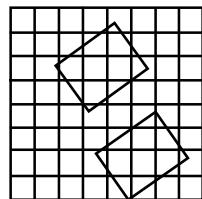
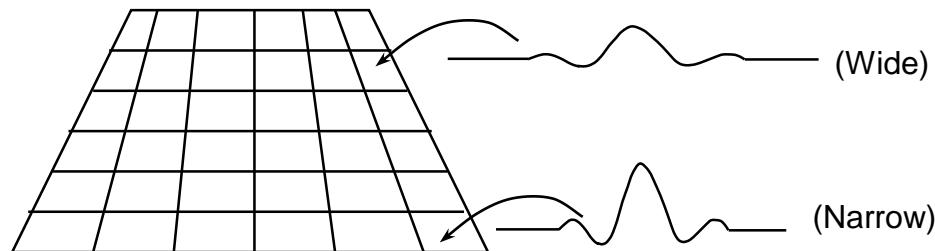
$$g(x) = \sum_{k \in z} f(k)\rho(x, k) \quad \text{where } \rho(x, k) = \int r(m^{-1}(t) - k)h(x - t)dt$$

Mathematical Formulation (2)

$$\rho(x, k) = \int r(m^{-1}(t) - k) h(x - t) dt$$

Selects filter response
used to index filter
coefficients

Spatially varying resampling
filter expressed in terms
of output space



Mathematical Formulation (3)

- We can express $\rho(x, k)$ in terms of input space:

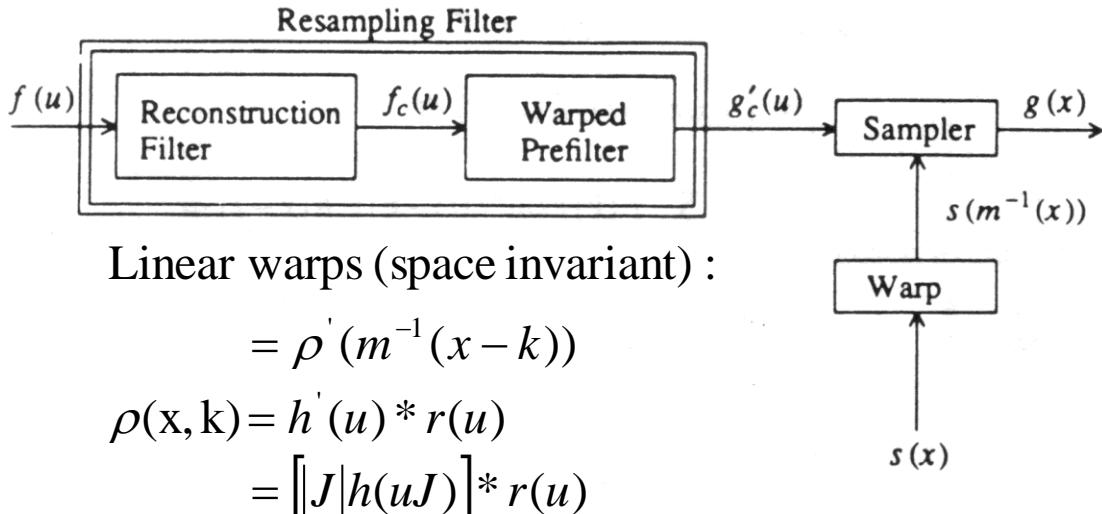
Let $t = m(u)$, we have

$$\rho(x, k) = \int r(u - k) h(x - m(u)) \left| \frac{\partial m}{\partial u} \right| dt$$

where $\left| \frac{\partial m}{\partial u} \right|$ is the determinant of Jacobian matrix :

$$1D: \left| \frac{\partial m}{\partial u} \right| = \frac{dm}{du} \quad 2D: \left| \frac{\partial m}{\partial u} \right| = \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix} \text{ where } x_u = \frac{\partial x}{\partial u}$$

Resampling Filter



$\rho_{mag}(x, k) = r(u)$: Shape of $r(u)$ remains the same, independently of $m(u)$, (independently of scale factor)

$\rho_{min}(x, k) = |J|h(uJ)|$: Shape of prefilter is based on desired frequency response characteristic (performance in passband and stopband). Unlike $r(u)$, though, the prefilter must be scaled proportional to the minification factor (broader and shorter for more minification)

Fourier Transform Pairs

- The shape of ρ_{min} is a direct consequence of the reciprocal relation between the spatial and frequency domains.

\leftrightarrow denotes Fourier Transform pair

$$H(u) = \int h(u)e^{-i2\pi fu} du$$

$$H(m(u)) = \int h(m(u))e^{-i2\pi fu} du$$

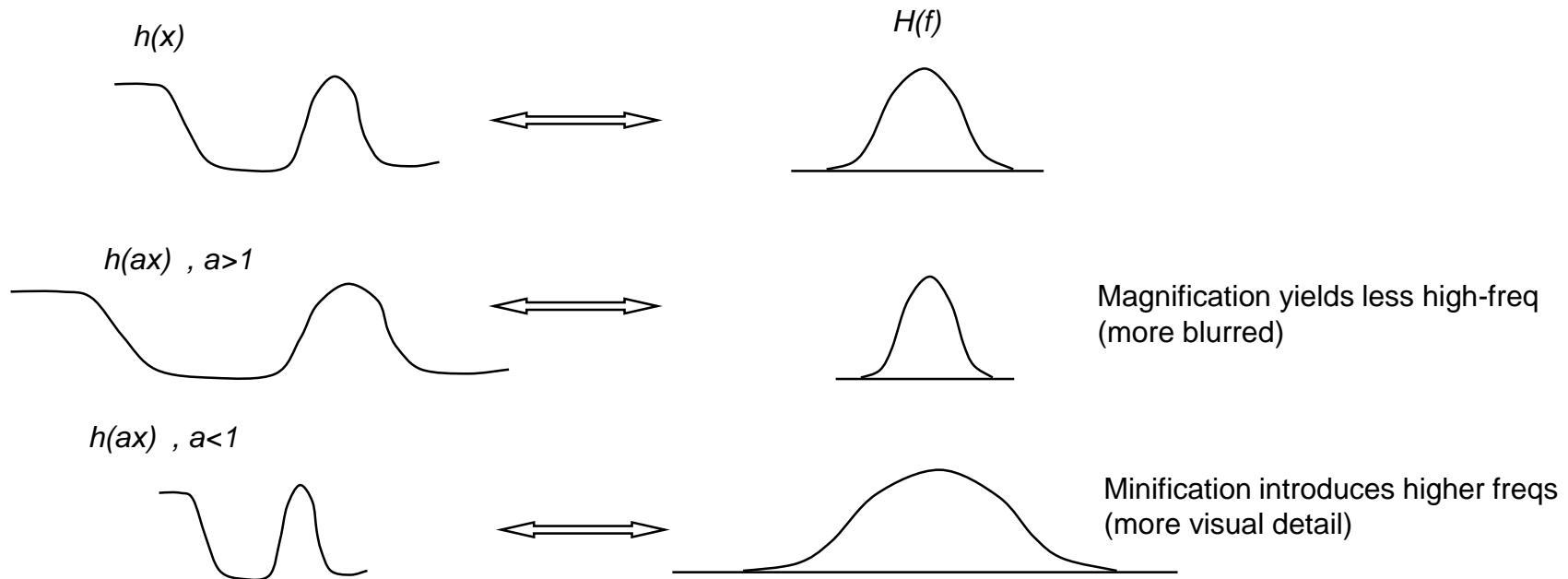
$$\text{Let } x = au = m(u) \text{ and } dx = \left| \frac{\partial m}{\partial u} \right| du$$

$$H(m(u)) = \int h(x)e^{-i2\pi fm^{-1}(x)} \frac{dx}{\left| \frac{\partial m}{\partial u} \right|} \quad \text{where } m^{-1}(x) = \frac{x}{a}; \quad \left| \frac{\partial m}{\partial u} \right| = |J| = a$$

$$h(au) \leftrightarrow \frac{1}{a} \int h(x)e^{-i2\pi \frac{fx}{a}} dx$$

$$h(au) \leftrightarrow \frac{1}{a} H\left(\frac{f}{a}\right)$$

Reciprocal Relationship



Intuition: $f = 1 / T$

Consequences: narrow filters in spatial domain (desirable) yield wide frequency spectrums (undesirable). Tradeoff between accuracy and complexity.

Software (1)

```
resample1D(IN, OUT, INlen, OUTlen, filtertype, offset)
unsigned char *IN, *OUT;
int INlen, OUTlen, filtertype, offset;
{
    int i;
    int left, right;      // kernel extent in input
    int pixel;            // input pixel value
    double u, x;          // input (u) , output (x)
    double scale;          // resampling scale factor
    double (*filter)();   // pointer to filter fct
    double fwidth;         // filter width (support)
    double fscale;         // filter amplitude scale
    double weight;         // kernel weight
    double acc;             // convolution accumulator

    scale = (double) OUTlen / INlen;
```

Software (2)

```
switch(filtertype) {
    case 0: filter = boxFilter; // box filter (nearest nbr)
              fwidth = .5;
              break;
    case 1: filter = triFilter; // triangle filter (lin intrp)
              fwidth = 1;
              break;
    case 2: filter = cubicConv; // cubic convolution filter
              fwidth = 2;
              break;
    case 3: filter = lanczos3; // Lanczos3 windowed sinc fct
              fwidth = 3;
              break;
    case 4: filter = hann4;      // Hann windowed sinc function
              fwidth = 4;          // 8-point kernel
              break;
}
```

Software (3)

```
if(scale < 1.0) {    // minification: h(x) -> h(x*scale)*scale
    fwidth = fwidth / scale;           // broaden filter
    fscale = scale;                  // lower amplitude

    /* roundoff fwidth to int to avoid intensity modulation */
    if(filtertype == 0) {
        fwidth = CEILING(fwidth);
        fscale = 1.0 / (2*fwidth);
    }
} else fscale = 1.0;

// project each output pixel to input, center kernel, and convolve
for(x=0; x<OUTlen; x++) {
    /* map output x to input u: inverse mapping */
    u = x / scale;

    /* left and right extent of kernel centered at u */
    if(u - fwidth < 0) {
        left = FLOOR(u - fwidth);
    } else    left = CEILING(u - fwidth);
    right = FLOOR(u + fwidth);
```

Software (4)

```
/* reset acc for collecting convolution products */
acc = 0;

/* weigh input pixels around u with kernel */
for(i=left; i <= right; i++) {
    pixel = IN[ CLAMP(i, 0, INlen-1)*offset];
    weight = (*filter)((u - i) * fscale);
    acc += (pixel * weight);
}

/* assign weighted accumulator to OUT */
OUT[x*offset] = acc * fscale;
}
```

Software (5)

```
double boxFilter(double t)
{
    if((t > -.5) && (t <= .5)) return(1.0);
    return(0.0);
}
```

```
double triFilter(double t)
{
    if(t < 0) t = -t;
    if(t < 1.0) return(1.0 - t);
    return(0.0);
}
```

Software (6)

```
double cubicConv(double t)
{
    double A, t2, t3;

    if(t < 0) t = -t;
    t2 = t * t;
    t3 = t2 * t;

    A = -1.0; // user-specified free parameter
    if(t < 1.0) return((A+2)*t3 - (A+3)*t2 + 1);
    if(t < 2.0) return(A*(t3 - 5*t2 + 8*t - 4));
    return(0.0);
}
```

Software (7)

```
double sinc(double t)
{
    t *= PI;
    if(t != 0) return(sin(t) / t);
    return(1.0);
}

double lanczos3(double t)
{
    if(t < 0) t = -t;
    if(t < 3.0) return(sinc(t) * sinc(t/3.0));
    return(0.0);
}
```

Software (8)

```
double hann4(double t)
{
    int N = 4;                  // fixed filter width
    if(t < 0) t = -t;
    if(t < N)
        return(sinc(t) * (.5 + .5*cos(PI*t / N)));
    return(0.0);
}
```

Image Reconstruction

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we describe image reconstruction:
 - Interpolation as convolution
 - Interpolation kernels for:
 - Nearest neighbor
 - Triangle filter
 - Cubic convolution
 - B-Spline interpolation
 - Windowed sinc functions

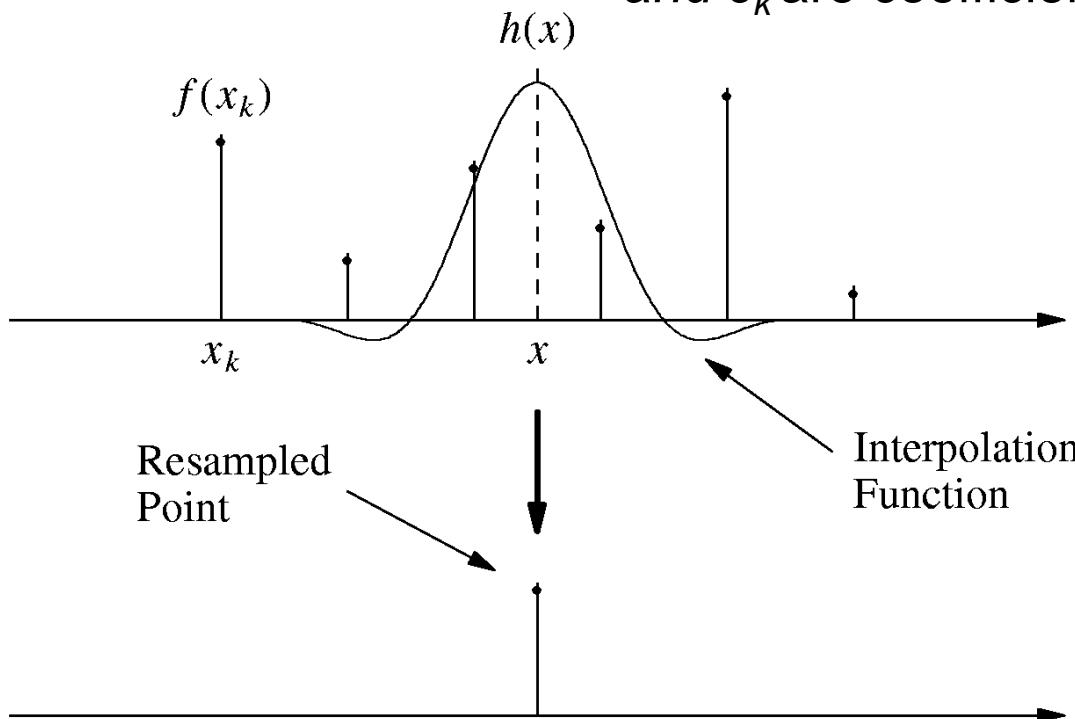
Introduction

- Reconstruction is synonymous with interpolation.
- Determine value at position lying between samples.
- Strategy: fit a continuous function through the discrete input samples and evaluate at any desired set of points.
- Sampling generates infinite bandwidth signal.
- Interpolation reconstructs signal by smoothing samples with an interpolation function (kernel).

Interpolation

For equi-spaced data, interpolation can be expressed as a convolution:

$$f(x) = \sum_{k=0}^{K-1} c_k h(x - x_r) \quad \text{where } K \text{ is the number of neighborhood pixels and } c_k \text{ are coefficients for kernel } h$$



Kernel Samples: $h(-d), h(-1-d), h(1-d), h(2-d)$

If h is symmetric,
 $h(d), h(1+d), h(1-d), h(2-d)$

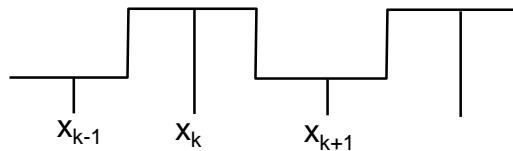
Interpolation Kernel

- Set of weights applied to neighborhood pixels
- Often defined analytically
- Usually symmetric: $h(x) = h(-x)$
- Commonly used kernels:
 - Nearest neighbor (pixel replication)
 - Triangle filter (linear interpolation)
 - Cubic convolution (smooth; used in digital cameras)
 - Windowed sinc functions (highest quality, more costly)

Nearest Neighbor

$$\text{Interpolating Polynomial : } f(x) = f(x_k) \quad \frac{x_{k-1} + x_k}{2} < x \leq \frac{x_k + x_{k+1}}{2}$$

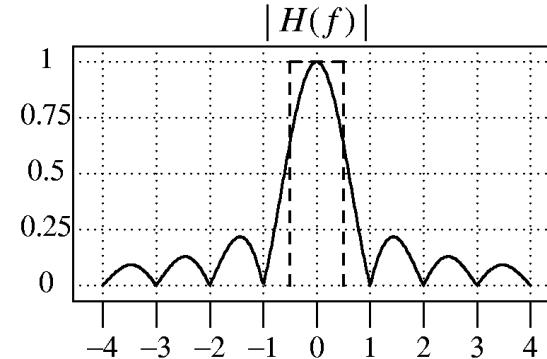
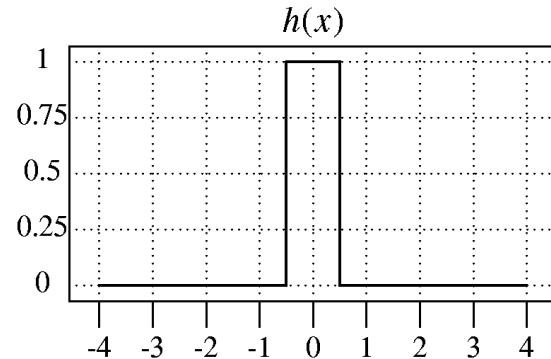
$$\text{Interpolating Kernel : } h(x) = \begin{cases} 1 & 0 \leq |x| < 0.5 \\ 0 & 0.5 \leq |x| \end{cases}$$



Other names: box filter, sample-and hold function, and Fourier window.

Poor stopband. NN achieves magnification by pixel replication. Very blocky.

Shift errors of up to 1/2 pixel are possible. Common in hardware zooms.



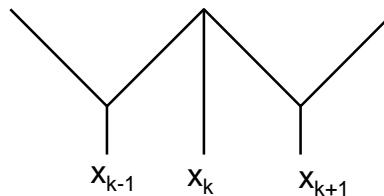
Triangle Filter

Interpolating Polynomial : $f(x) = a_1x + a_0$

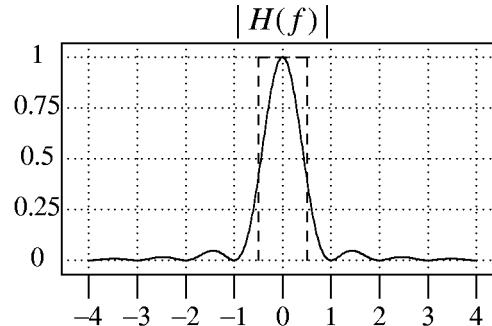
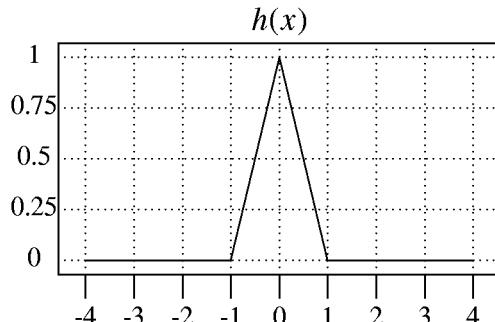
$$[f_0 \ f_1] = [a_1 \ a_0] \begin{bmatrix} x_0 & x_1 \\ 1 & 1 \end{bmatrix} \quad \text{Solve for } a_1, a_0$$

$$f(x) = f_0 + \left[\frac{x - x_0}{x_1 - x_0} \right] (f_1 - f_0)$$

Interpolation Kernel : $h(x) = \begin{cases} 1 - |x| & 0 \leq |x| < 1 \\ 0 & 1 \leq |x| \end{cases}$



Other names for h : triangle filter, tent filter, roof function, chateau function, and Bartlett window.

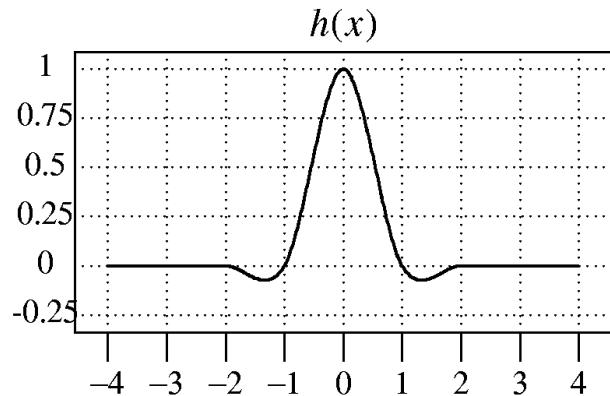


In the frequency domain:

$$\text{Sinc} * \text{Sinc} = \text{Sinc}^2$$

Cubic Convolution (1)

Third degree approximation to sinc. Its kernel is derived from constraints imposed on the general cubic spline interpolation formula.



$$h(x) = \begin{cases} a_{30}|x|^3 + a_{20}|x|^2 + a_{10}|x| + a_{00} & 0 \leq |x| < 1 \\ a_{31}|x|^3 + a_{21}|x|^2 + a_{11}|x| + a_{01} & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}$$

Determine coefficients by applying following constraints:

1. $h(0) = 1$ and $h(x) = 0$ for $|x| = 1, 2$
2. h must be continuous at $|x| = 0, 1, 2$
3. h must have a continuous first derivative at $|x| = 0, 1, 2$

Cubic Convolution (2)

Constraint (1) states that when h is centered on an input sample, the interpolation function is independent of neighboring samples.

First 2 constraints give 4 equations:

$$1 = h(0) = a_{00}$$

$$0 = h(1^-) = a_{30} + a_{20} + a_{10} + a_{00}$$

$$0 = h(1^+) = a_{31} + a_{21} + a_{11} + a_{01}$$

$$0 = h(2^-) = 8a_{31} + 4a_{21} + 2a_{11} + a_{01}$$

3 more equations are obtained from constraint (3):

$$-a_{10} = h'(0^-) = h'(0^+) = a_{10}$$

$$3a_{30} + 2a_{20} + a_{10} = h'(1^-) = h'(1^+) = 3a_{31} + 2a_{21} + a_{11}$$

$$12a_{31} + 4a_{21} + a_{11} = h'(2^-) = h'(2^+) = 0$$

Total 7 equations, 8 unknowns \rightarrow free variable ($a = a_{31}$)

$$h(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \leq |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}$$

Cubic Convolution (3)

How to pick a? Add some heuristics (make it resemble Sinc function):

$$h''(0) = -2(a+3) < 0 \rightarrow a > -3 \quad \text{Concave downward at } x = 0$$

$$h''(1) = -4a > 0 \quad \text{Concave upward at } x = 1$$

This bounds a to the [-3, 0] range.

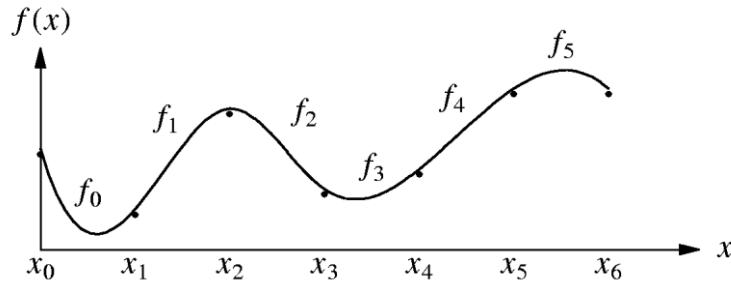
Common choices:

$a = -1$ matches the slope of sinc at $x=1$ (sharpens image)

$a = -0.5$ makes the Taylor series approximately agree in as many terms as possible with the original signal

$a = -.75$ sets the second derivative of the 2 cubic polynomials in h to 1
(continuous 2nd derivative at $x = 1$)

Cubic Splines (1)



$$f_k(x) = a_3(x - x_k)^3 + a_2(x - x_k)^2 + a_1(x - x_k) + a_0$$

6 polynomial segments, each of 3rd degree.

f_k 's are joined at x_k (for $k=1, \dots, n-2$) such that f_k , f'_k , and f''_k are continuous.

$$a_0 = y_k$$

$$a_1 = y'_k \quad \text{where} \quad \Delta y_k = y_{k+1} - y_k$$

$$a_2 = 3\Delta y_k - 2y'_k - y'_{k+1}$$

$$a_3 = -2\Delta y_k + y'_k + y'_{k+1}$$

(proof in App. 2)

Cubic Splines (2)

- The derivatives may be determined by solving the following system of linear equations:

$$\begin{bmatrix} 2 & 4 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & 1 & 4 & 1 \\ & & & \ddots & \\ & & & & 1 & 4 & 1 \\ & & & & & 4 & 2 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} -5y_0 + 4y_1 + y_2 \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ 3(y_4 - y_2) \\ \vdots \\ 3(y_{n-1} - y_n - 3) \\ -y_{n-3} - 4y_{n-2} + 5y_{n-1} \end{bmatrix}$$

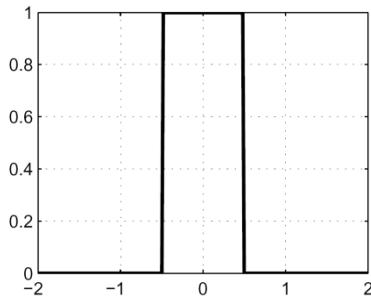
↑

global dependencies

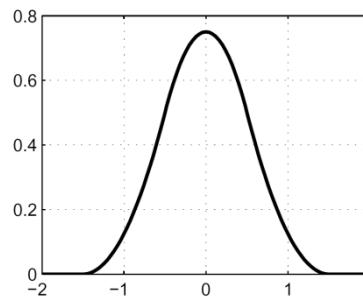
- Introduced by the constraints for continuity in the first and second derivatives at the knots.

B-Splines

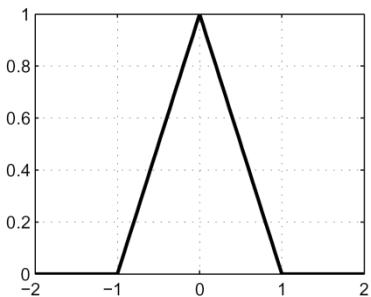
To analyze cubic splines, introduce cubic B-Spline interpolation kernel:



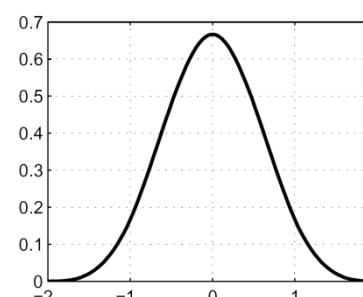
B₀ box filter



B₂ = B₀ * B₀ * B₀



B₁ = B₀ * B₀



B₃ = B₀ * B₀ * B₀ * B₀

$$h(x) = \frac{1}{6} \begin{cases} 3|x|^3 - 6|x|^2 + 4 & 0 \leq |x| < 1 \\ -|x|^3 + 6|x|^2 - 12|x| + 8 & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}$$

Parzen Window: Not interpolatory because it does not satisfy $h(0) = 1$, and $h(1) = h(2) = 0$. Indeed, it approximates the data.

Interpolatory B-Splines

$$f(x_j) = \sum_{k=j-2}^{j+2} c_k h(x_j - x_k)$$

Since $h(0) = \frac{4}{6}$, $h(-1) = h(1) = \frac{1}{6}$, $h(-2) = h(2) = 0$

we have $f(x_j) = \frac{1}{6}(c_{j-1} + 4c_j + c_{j+1})$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \ddots & & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix}$$

$$\begin{array}{lll} F & = & K \\ C & = & K^{-1}F \end{array}$$

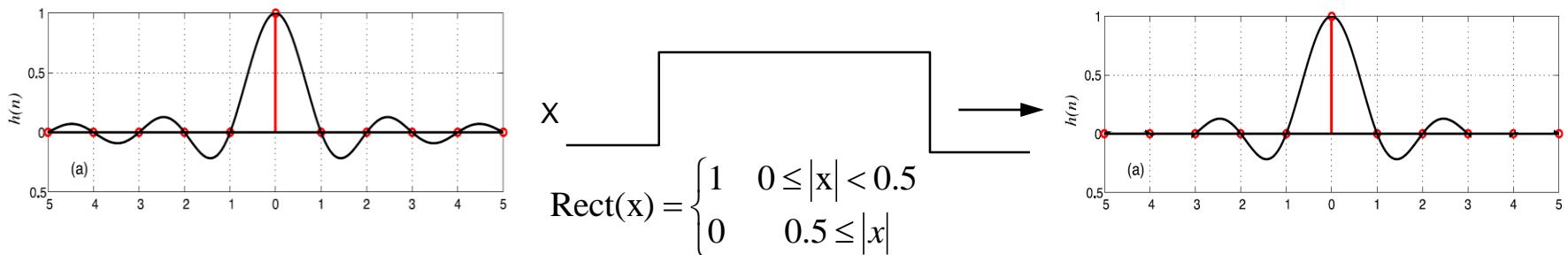
K^{-1} ← inverse of tridiagonal matrix; Computation is $O(n)$

All previous methods used data values for c_k from $C = K^{-1}F$.

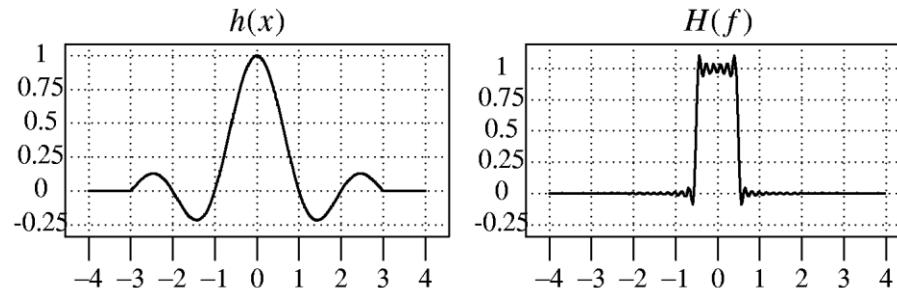
Truncated Sinc Function

- Alternative to previous kernels: use windowed sinc function.

Truncated Sinc



Truncating in spatial domain = convolving spectrum (box) with a Sinc function.



Ringing can be mitigated by using a smoothly tapering windowing function.

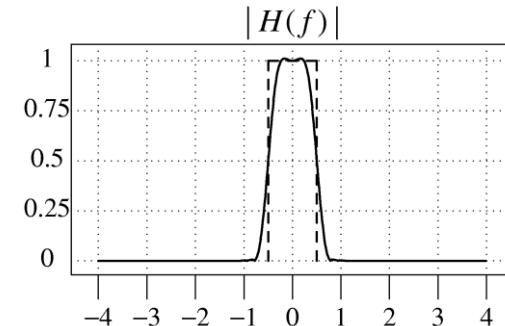
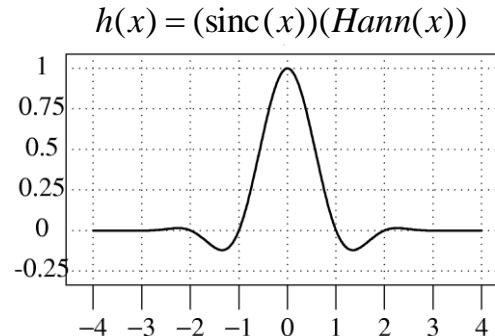
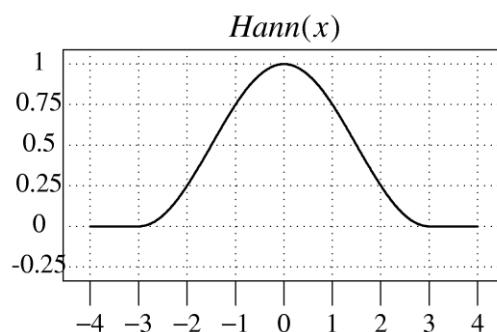
Popular window functions: Hann, Hamming, Blackman, Kaiser, and Lanczos.

Hann/Hamming Window

$$\text{Hann/Hamming}(x) = \begin{cases} \alpha + (1-\alpha)\cos\frac{2\pi x}{N-1} & |x| < \frac{N-1}{2} \\ 0 & o/w \end{cases}$$

N = number of samples in windowing function.

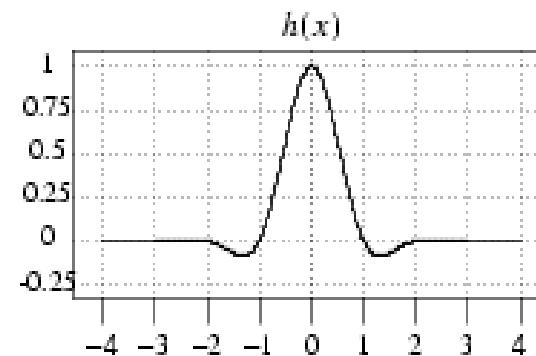
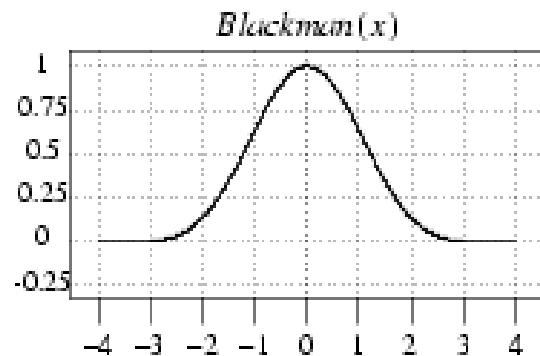
Hann: $\alpha = 0.5$; Hamming: $\alpha = 0.54$. Also known as raised cosine window.



$|H(f)|$ is $\text{sinc}+2$ shifted sincs. These cancel the right and left side lobes of $\text{Rect}(x)$.

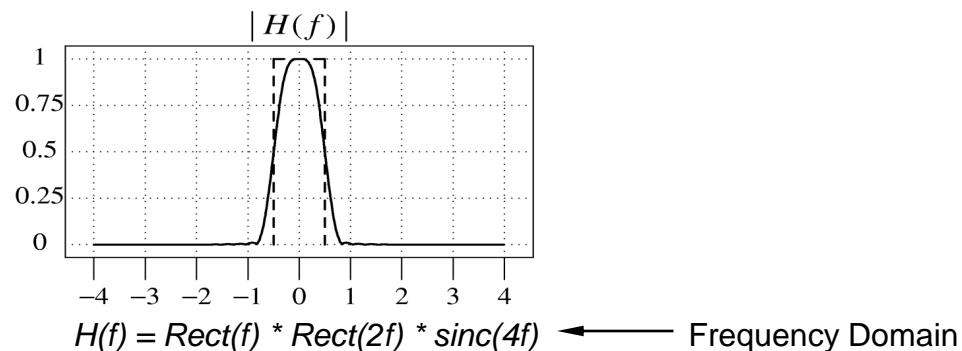
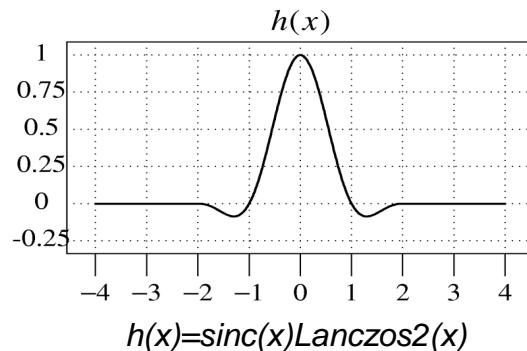
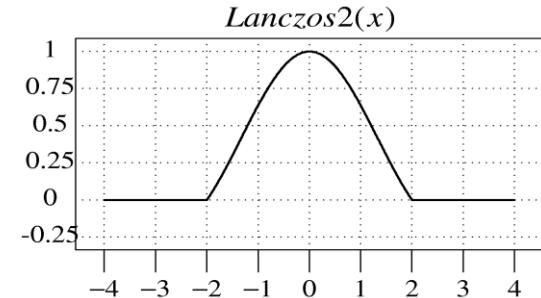
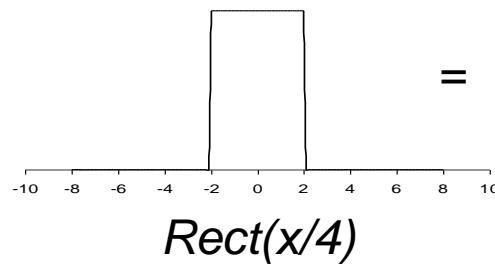
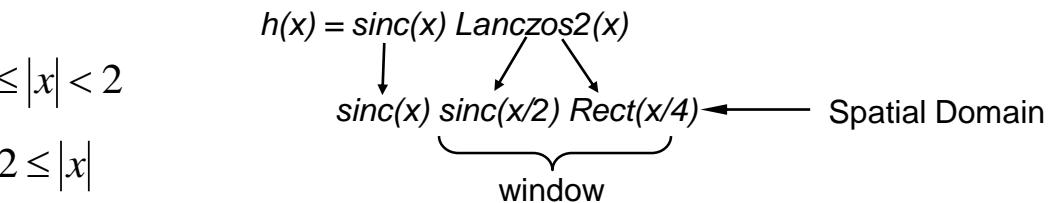
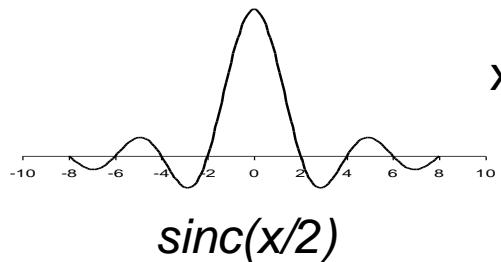
Blackman Window

$$Blackman(x) = \begin{cases} .42 + .5\cos\frac{2\pi x}{N-1} + .08\cos\frac{4\pi x}{N-1} & |x| < \frac{N-1}{2} \\ 0 & o/w \end{cases}$$



Lanczos Window (1)

$$Lanczos2(x) = \begin{cases} \text{sinc}\left(\frac{\pi x}{2}\right) & 0 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}$$

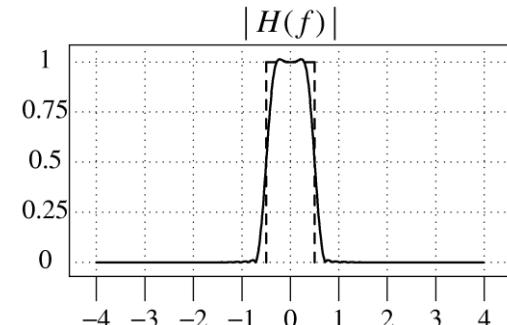
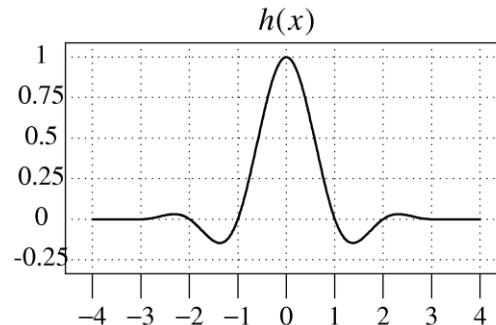
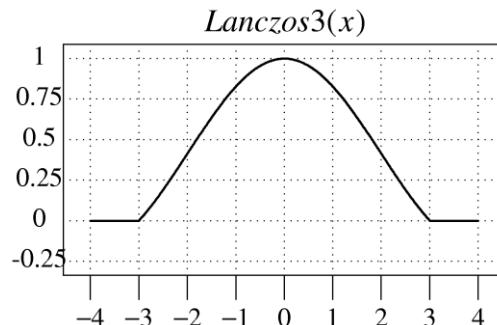


Lanczos Window (2)

- Generalization to N lobes:

$$LanczosN(x) = \begin{cases} \text{sinc}\left(\frac{\pi x}{N}\right) & 0 \leq |x| < N \\ 0 & N \leq |x| \end{cases}$$

- Let $N = 3$, this lets 3 lobes pass under the Lanczos window.



- Better passband and stopband response

Comparison of Interpolation Methods

NN, linear, cubic convolution, windowed sinc, sinc
poor > ideal
(blocky, blurred, ringing, no artifacts)

Convolution Implementation

1. Position (center) kernel in input.
2. Evaluate kernel values at positions coinciding with neighbors.
3. Compute products of kernel values and neighbors.
4. Add products; init output pixel.

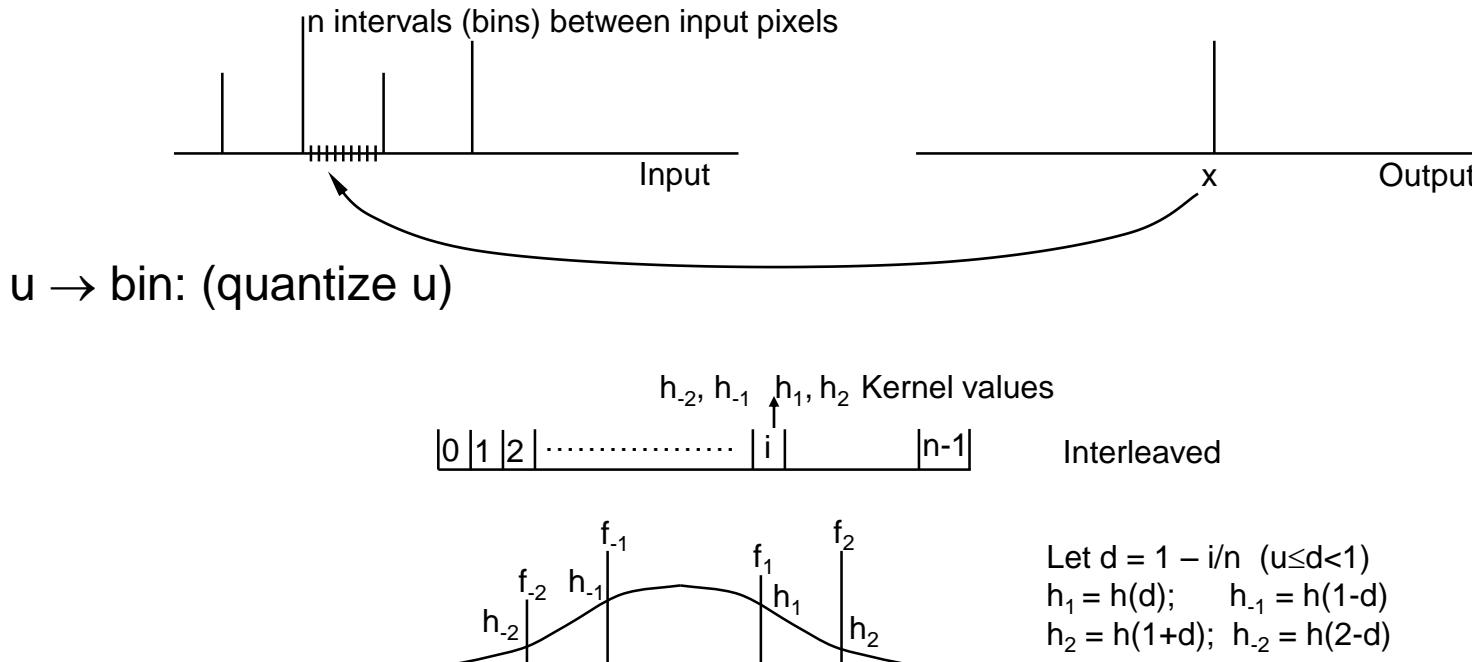
Step (1) can be simplified by incremental computation for space-invariant warps. ($\text{newpos} = \text{oldpos} + \text{inc}$).

Step (2) can be simplified by LUT.

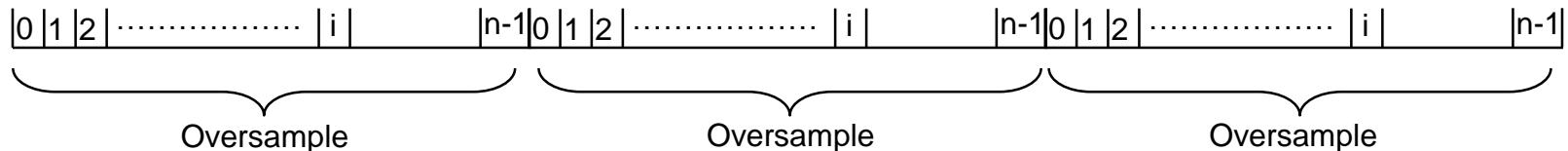
Interpolation with Coefficient Bins

Implementation #1: Interp. with Coefficient Bins (for space-invariant warps)

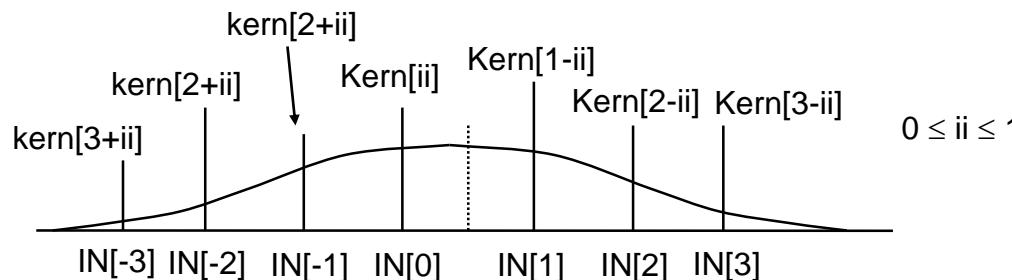
- **Strategy:** accelerate resampling by precomputing the input weights and storing them in LUT for fast access during convolution.



Uninterleaved Coefficient Bins



```
ii = bin #
0 ≤ ii < oversample
val =
    IN[-2] * kern[2 * oversample + ii]
    +
    IN[-1] * kern[1 * oversample + ii]
    +
    IN[ 0] * kern[ii]                               Refer to code on p. 151
    +
    IN[ 1] * kern[1 * oversample - ii]
    +
    IN[ 2] * kern[2 * oversample - ii]
    +
    IN[ 3] * kern[3 * oversample - ii];
if(ii == 0)val +=IN[-3] * kern[3 * oversample - ii];
```

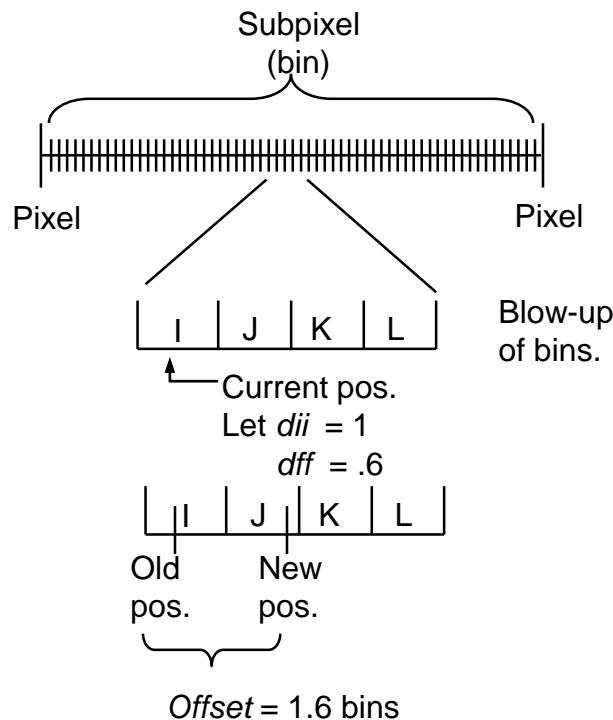


Kernel Position

- Since we are assuming space invariance, the new position for the kernel = oldpos + offset.

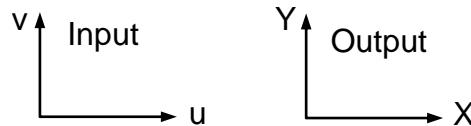
$$\text{offset} = dii + dff; \quad dii = \# \text{ whole bins} \left(\frac{\text{INlen} * \text{oversample}}{\text{OUTlen}} \right)$$

$$dff = \text{partial bin } (\text{INlen} * \text{oversample}) \% \text{OUTlen}$$



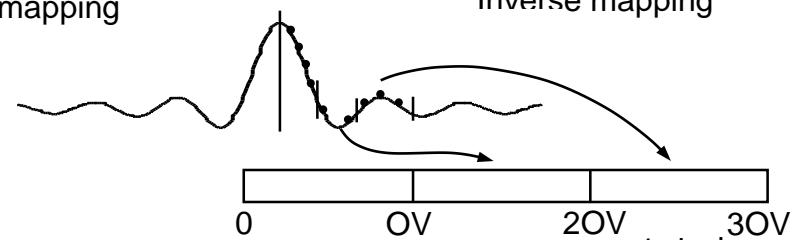
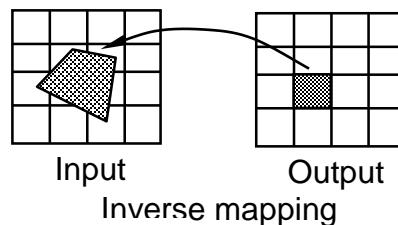
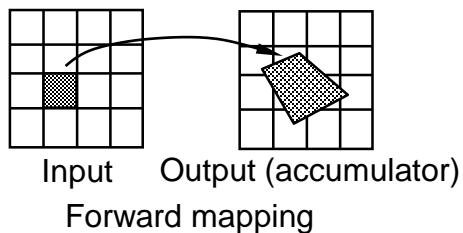
Offset must be accurate to avoid accrual of error in the incremental repositioning of the kernel.

Forward vs. Inverse Mapping

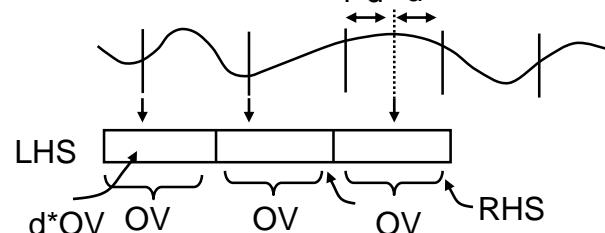


- Forward mapping: $x = X(u, v); y = Y(u, v)$
- Inverse mapping: $u = U(x, y); v = V(x, y)$

Ch. 3, Sec. 1



Coefficient Bins for kernel eval for fast Convolution for image reconstruction.



Fant's Algorithm

Implementation #2: Fant's Resampling Algorithm (for space-var. warps)



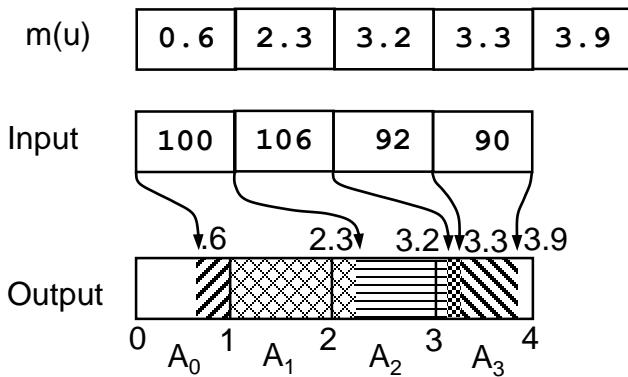
Input and output are streams of pixels that are consumed and generated at rate determined by the spatial mapping.

Three conditions per stream:

- 1) Current input pixel is entirely consumed without completing an output pixel.
- 2) The input is entirely consumed while completing the output pixel.
- 3) Output pixel computed without entirely consuming the current input pixel.

Algorithm uses linear interpolation for image reconstruction and box filtering (unweighted averaging) for antialiasing. Code on p.156.

Example

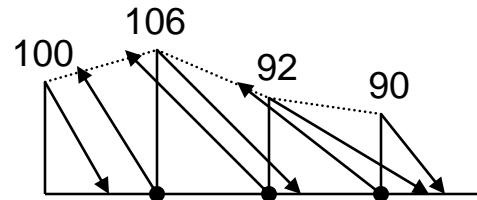


$$A_0 = (100)(.4) = 40$$

$$A_1 = \left[(100)\left(1 - \frac{.4}{1.7}\right) + (106)\left(\frac{.4}{1.7}\right) \right](1) = 101$$

$$A_2 = \left[(100)\left(1 - \frac{1.4}{1.7}\right) + (106)\left(\frac{1.4}{1.7}\right) \right](.3) + (106)(.7) = 106$$

$$A_3 = \left[(106)\left(1 - \frac{.7}{.9}\right) + (92)\left(\frac{.7}{.9}\right) \right](.2) + (92)(.1) + (90)(.6) = 82$$



Antialiasing

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

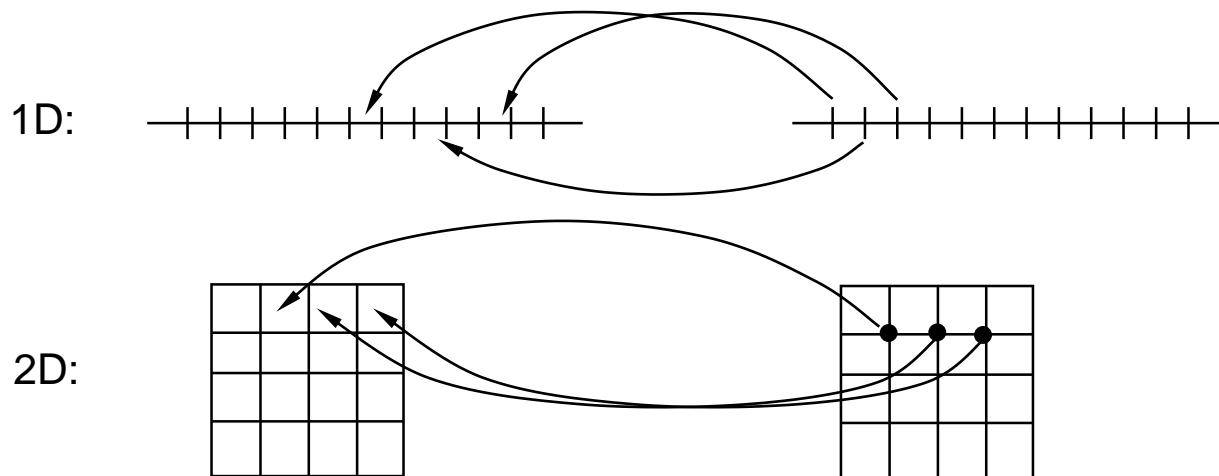
- In this lecture we review antialiasing:
 - Supersampling (uniform, adaptive, irregular)
 - Direct convolution
 - Feibusch-Levoy-Cook
 - Gangnet-Perry-Coveignoux
 - Greene-Heckbert (Elliptical Weighted Average)
 - Prefiltering
 - Mip-maps / pyramids
 - Summed area tables

Antialiasing

- Aliasing is due to undersampling the input signal.
- It results in false artifacts, e.g., moire effects.
- Antialiasing combats aliasing. Solutions:
 - Raise sampling rate
 - Bandlimit input
 - In practice, do both in inverse mapping formulation
- Implementations:
 - Supersampling (uniform, adaptive, irregular)
 - Direct convolution
 - Prefiltering

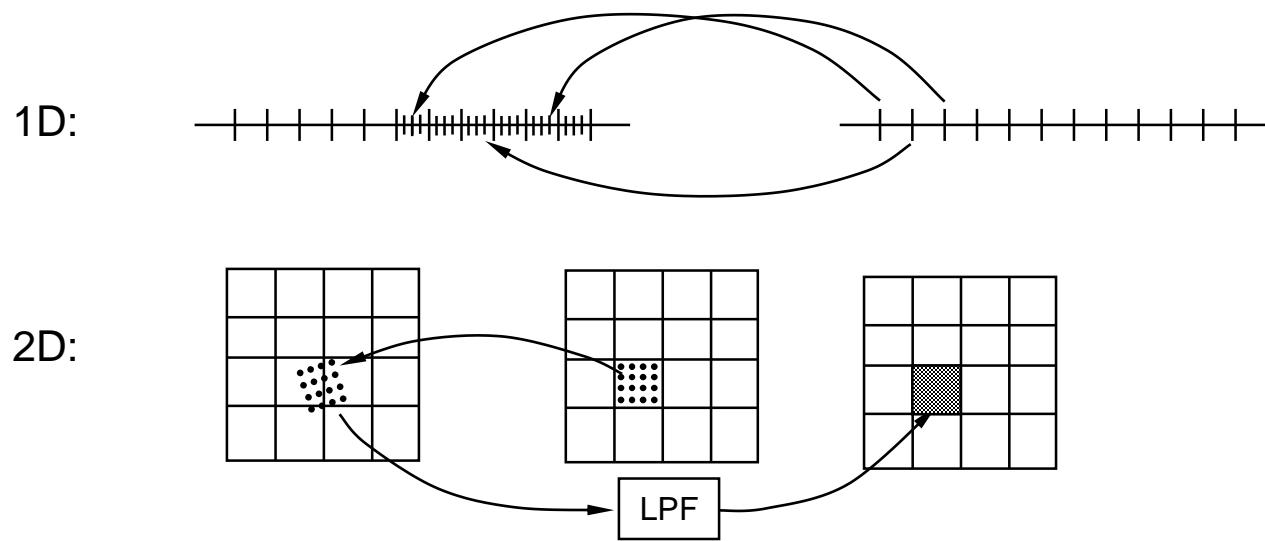
Point Sampling

- One input sample per output pixel
- Problem: information is lost between samples
- Solution: sample more densely and average results



Supersampling (1)

- Multiple input samples per output pixel



Supersampling (2)



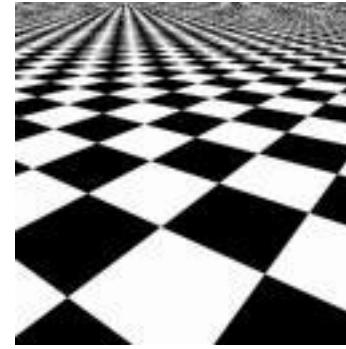
1 sample/pixel



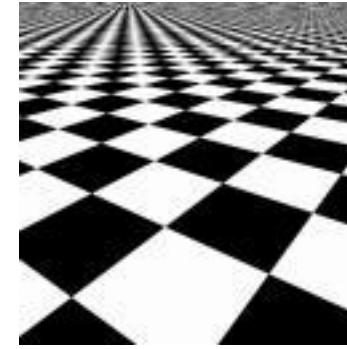
4 samples/pixel



16 samples/pixel



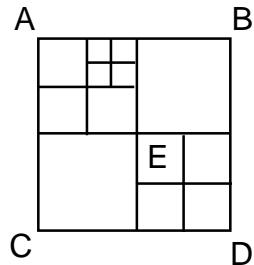
64 samples/pixel



256 samples/pixel

Adaptive Supersampling

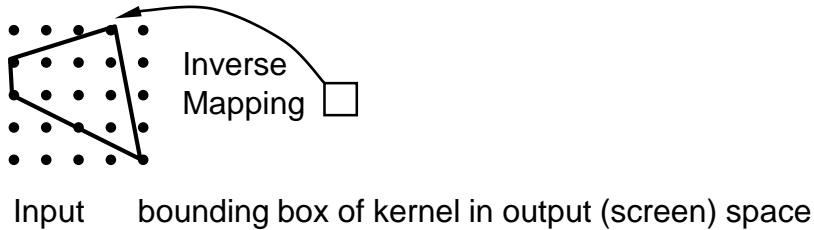
- Collect more samples in areas of high intensity variance or contrast.



```
IF (f(A,B,C,D,E)>thr) then subdivide /* quad tree */
```

- To avoid artifacts due to regular sampling pattern (along rectilinear grid), use irregular sampling.
- Three common forms of stochastic sampling:
 - Poisson
 - Jittered
 - Point-diffusion sampling

Direct Convolution: Feibush-Levoy-Cook (1980)



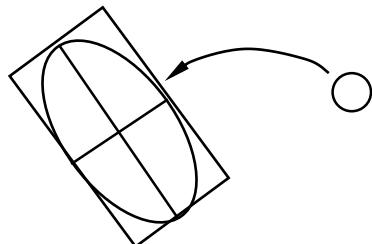
1. All input pixels contained within the bounding rect. of this quadrilateral are mapped onto output.
2. The extra selected points are eliminated by clipping them against the bounding rect. of kernel.
3. Init output pixel with weighted average of the remaining samples. Note that filter weights are stored in a LUT (e.g., a Gaussian filter.)

Advantages:

- 1) Easy to collect input pixels in rect. input region
- 2) Easy to clip output pixels in rect. output region.
- 3) Arbitrary filter weights can be stored in LUT.

Direct Convolution: Gangnet-Perry-Coveignoux (1982)

Improve results by supersampling.

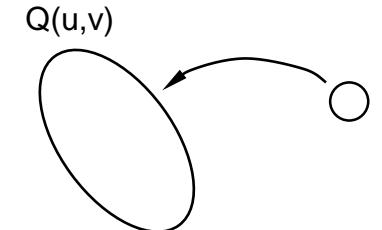


Major axis determines supersampling rate.

They used bilinear interpolation for image reconstruction and a truncated sinc (2 pixels on each side) as kernel.

Elliptical weighted average (EWA): Greene-Heckbert (1986)

EWA distorts the circular kernel into an ellipse in the input where the weighting can be computed directly.



No mapping back to output space.

$$Q(u, v) = Au^2 + Buv + Cv^2 \quad u = v = 0 \text{ is the center of the ellipse}$$

$$A = V_x^2 + V_y^2$$

$$B = -2(U_x V_x + U_y V_y)$$

$$C = U_x^2 + U_y^2$$

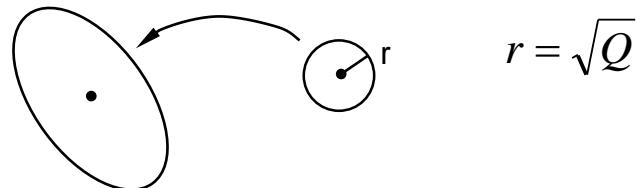
where

$$(U_x, V_x) = \left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x} \right) \quad (U_y, V_y) = \left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y} \right)$$

Point - inclusion test : $Q(u, v) < F$ for $F = (U_x V_y - U_y V_x)^2$

EWA Advantages

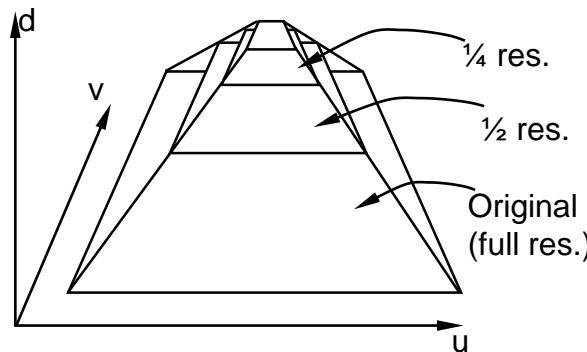
- Only 1 function evaluated; not 4 needed for quadrilateral.
- If $Q < F$, then sample is weighted with appropriate LUT entry.
- In output space, LUT is indexed by r
- Instead of determining which concentric circle the point lies in the output, we determine which concentric ellipse the point lies in the input and use it to index the LUT.



Comparisons

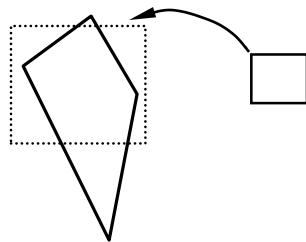
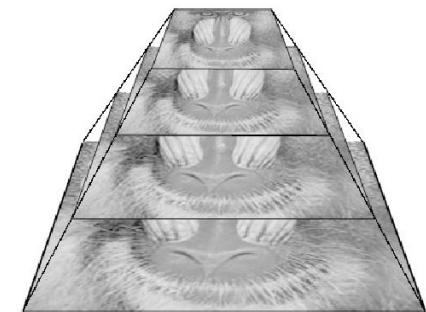
- All direct convolution methods are $O(N)$ where $N = \#$ input pixels accessed. In Feibush and Gangnet, these samples must be mapped into output space, EWA avoids this costly step.
- Direct convolution imposes minimal constraints on filter area (quad, ellipse) and filter kernel (precomputed LUT). Additional speedups: prefiltering with pyramids and preintegrated tables -> approx convolution integral w/ a constant # of accesses (indep. of # input pixels in preimage.)
- Drawback of pyramids and preintegrated tables: filter area must be square or rectangular; kernel must be a box filter.

Pyramids



$$\text{memory cost} = 1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots = \frac{4}{3}$$

only 33% more memory



Size of square is used to determine pyramid level to sample.
Aliasing vs. Blurring tradeoff.

$$d^2 = \max \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2, \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right]$$

d is proportional to span of the preimage area.

Summed-Area Table

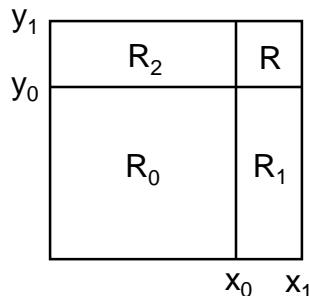


Table stores running sum

$$sum_R = T[x_1, y_1] - T[x_1, y_0] - T[x_0, y_1] + T[x_0, y_0]$$

Restricted to rectangular regions and box filtering

To compute current entry $T[x_1, y_1]$ let sum R be current pixel value v :

$$T[x_1, y_1] = v + T[x_1, y_0] + T[x_0, y_1] - T[x_0, y_0]$$

Example

90	10	20	30
50	60	70	80
25	75	200	180
100	50	70	80

Input

265	460	820	1190
175	360	700	1040
125	250	520	780
100	150	220	300

Summed-area table

$$sum_R = T[x_1, y_1] - T[x_1, y_0] - T[x_0, y_1] + T[x_0, y_0]$$

$$T[x_1, y_1] = v + T[x_1, y_0] + T[x_0, y_1] - T[x_0, y_0]$$

Spatial Transformations

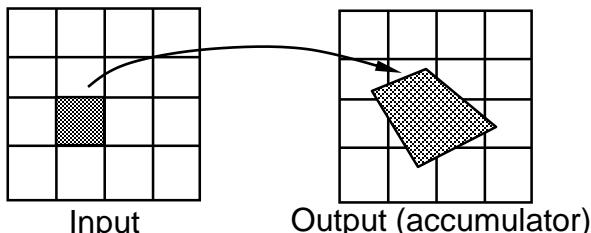
Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

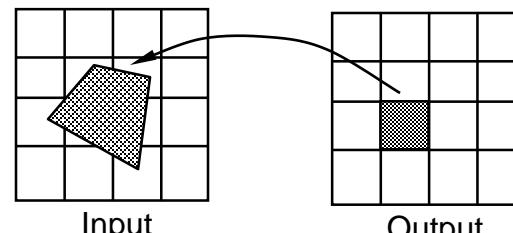
- In this lecture we review spatial transformations:
 - Forward and inverse mappings
 - Transformations
 - Linear
 - Affine
 - Perspective
 - Bilinear
 - Inferring affine and perspective transformations

Forward and Inverse Mappings

- A spatial transformation defines a geometric relationship between each point in the input and output images.
- Forward mapping: $[x, y] = [X(u, v), Y(u, v)]$
- Inverse mapping : $[u, v] = [U(x, y), V(x, y)]$
- Forward mapping specifies output coordinates (x,y) for each input point (u,v) .
- Inverse mapping specifies input point (u,v) for each output point (x,y) .



Forward mapping



Inverse mapping

Linear Transformations

$$[x, y] = [u, v] \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$x = a_{11}u + a_{21}v$$

$$y = a_{12}u + a_{22}v$$

- Above equations are linear because they satisfy the following two conditions necessary for any linear function L(x):
 - 1) $L(x+y) = L(x) + L(y)$
 - 2) $L(cx) = cL(x)$ for any scalar c and position vectors x and y.
- Note that linear transformation are a sum of scaled input coordinate: they do not account for simple translation.

We want:

$$x = a_{11}u + a_{21}v + a_{31}$$

$$y = a_{12}u + a_{22}v + a_{32}$$

$$[x, y] = [u, v, 1] \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

Homogeneous Coordinates

- To compute inverse, the transformation matrix must be square.
- Therefore,

$$[x, y, 1] = [u, v, 1] \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

- All 2-D position vectors are now represented with three components: homogeneous notation.
- Third component (w) refers to the plane upon which the transformation operates.
- $[u, v, 1] = [u, v]$ position vector lying on $w=1$ plane.
- The representation of a point in the homogeneous notation is no longer unique: $[8, 16, 2] = [4, 8, 1] = [16, 32, 4]$.
- To recover any 2-D position vector $p[x, y]$ from $p_h = [x', y', w']$, divide by the homogeneous coordinate w' .

$$[x, y] = \left[\frac{x'}{w'} \quad \frac{y'}{w'} \right]$$

Affine Transformations (1)

$$[x, y, 1] = [u, v, 1] \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

$$\text{Translation} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_u & T_v & 1 \end{bmatrix}$$

$$\text{Rotation} \rightarrow \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Scale} \rightarrow \begin{bmatrix} S_u & 0 & 0 \\ 0 & S_v & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Shear rows} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ H_u & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Shear columns} \rightarrow \begin{bmatrix} 1 & H_u & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

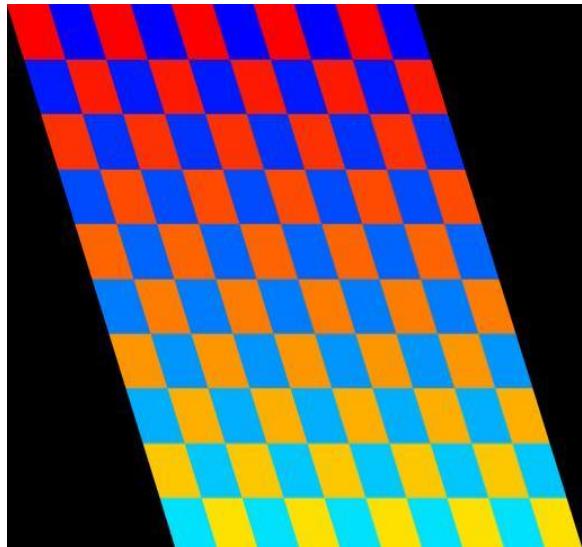
Affine Transformations (2)

- Affine transformation have 6 degrees of freedom: $a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{23}$.
- They can be inferred by giving the correspondence of three 2-D points between the input and output images. That is,

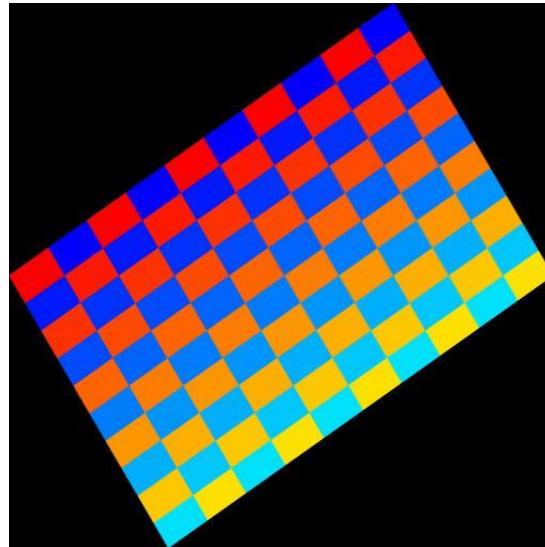
$$\begin{matrix} (u_1, v_1) & \rightarrow & (x_1, y_1) \\ (u_2, v_2) & \rightarrow & (x_2, y_2) \\ (u_3, v_3) & \rightarrow & (x_3, y_3) \end{matrix} \left. \right\} \text{6 constraints : (3 for } u \rightarrow x, \text{ 3 for } v \rightarrow y\text{)}$$

- All points lie on the same plane.
- Affine transformations map triangles onto triangles.

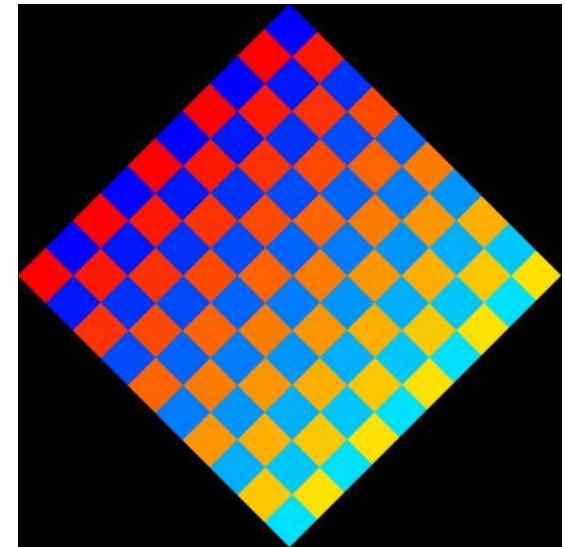
Affine Transformations (3)



Skew (shear)



Rotation/Scale



Rotation

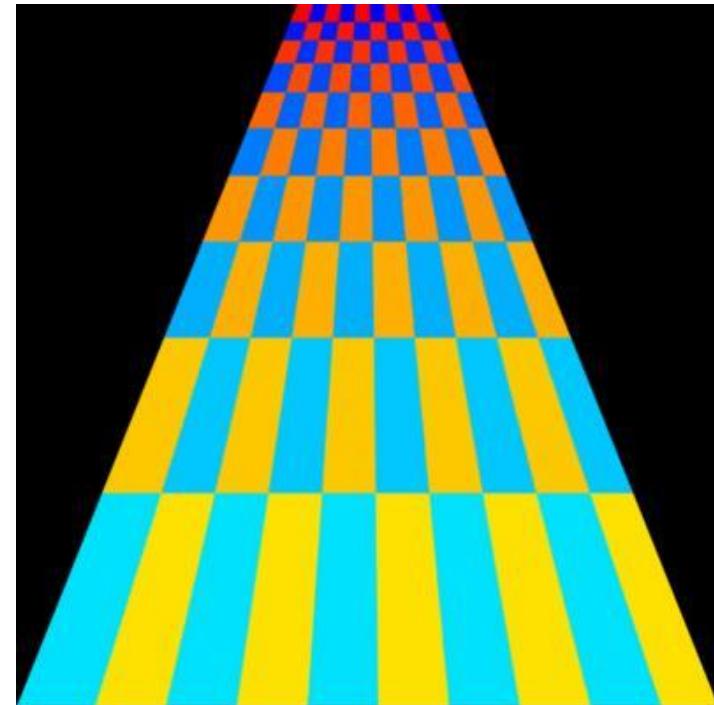
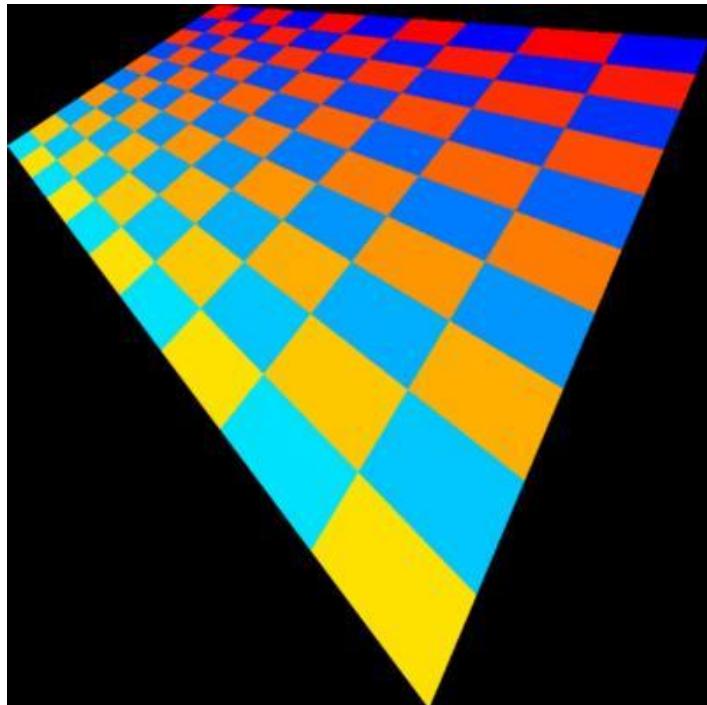
Perspective Transformations (1)

$$[x', y', w'] = [u, v, w] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$
$$x = \frac{x'}{w} = \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}}$$
$$y = \frac{y'}{w} = \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}}$$

$\begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$ not necessarily $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ as in affine transformations

- Without loss of generality, we set $a_{33}=1$.
- This yields 8 degrees of freedom and allows us to map planar quadrilaterals to planar quadrilaterals (correspondence among 4 sets of 2-D points yields 8 coordinates).
- Perspective transformations introduce foreshortening effects.
- Straight lines are preserved.

Perspective Transformations (2)

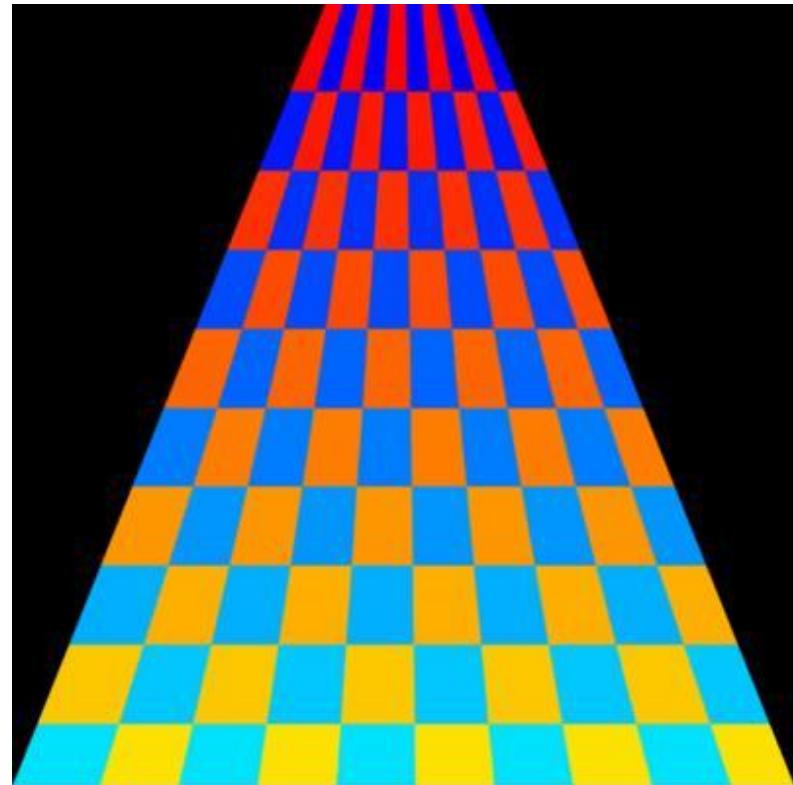
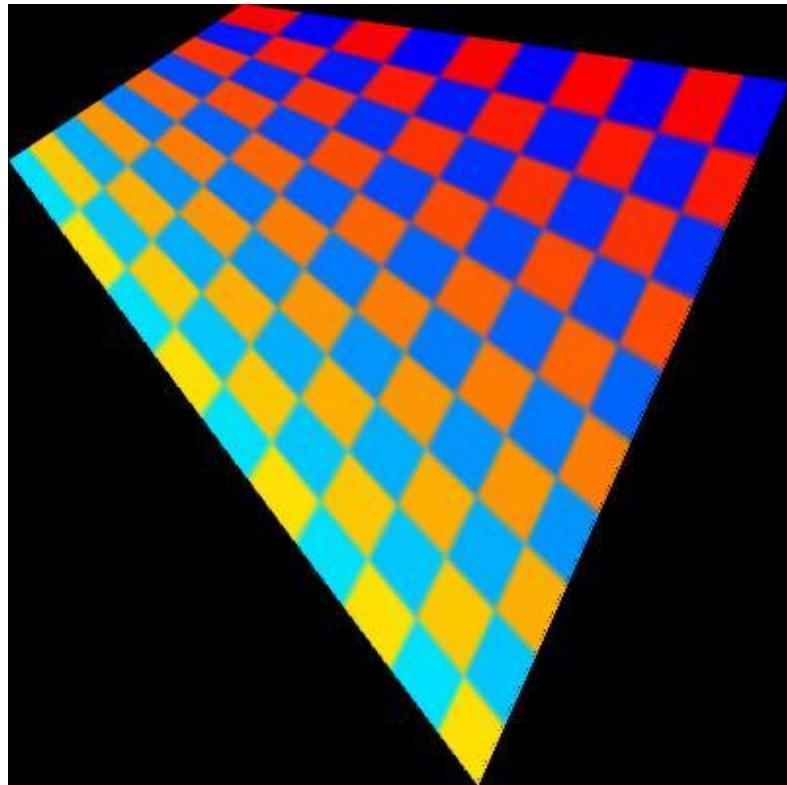


Bilinear Transforms (1)

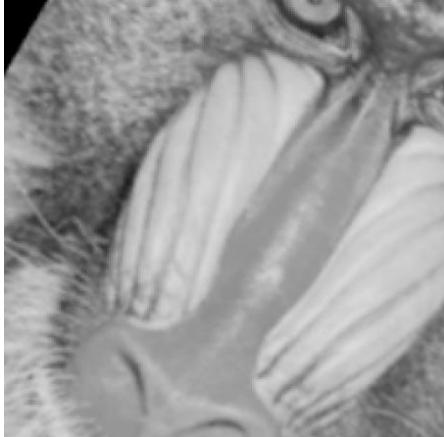
$$[x, y] = [uv, u, v, 1] \begin{bmatrix} a_3 & b_3 \\ a_2 & b_2 \\ a_1 & b_1 \\ a_0 & b_0 \end{bmatrix}$$

- 4-corner mapping among nonplanar quadrilaterals (2^{nd} degree due to uv factors).
- Conveniently computed using separability (see p. 57-60).
- Preserves spacing along edges.
- Straight lines in the interior no longer remain straight.

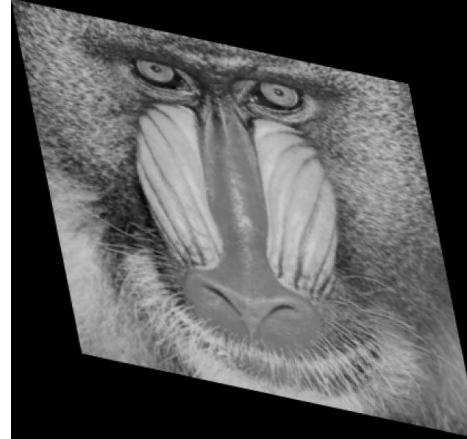
Bilinear Transforms (2)



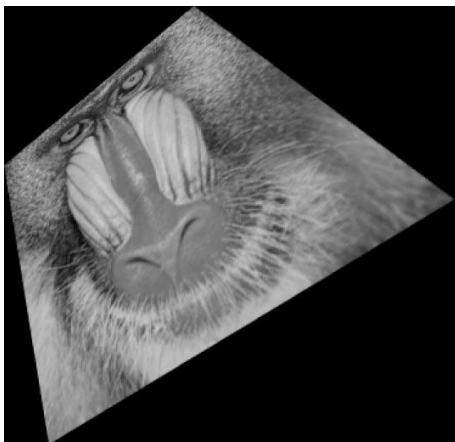
Examples



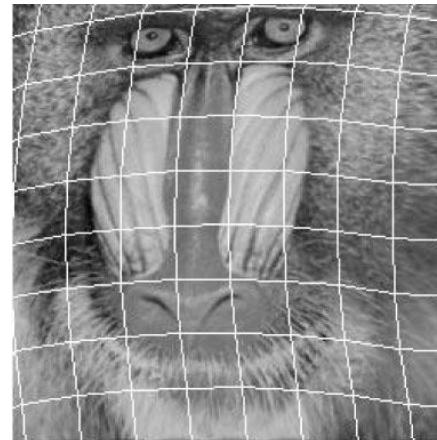
Similarity transformation (RST)



Affine transformation



Perspective transformation



Polynomial transformation

Scanline Algorithms

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we review scanline algorithms:
 - Incremental texture mapping
 - 2-pass Catmull-Smith algorithm
 - Rotation
 - Perspective
 - 3-pass shear transformation for rotation
 - Morphing

Catmull-Smith Algorithm

- Two-pass transform
- First pass resamples all rows: $[u, v] \rightarrow [x, v]$
 $[x, v] = [F_v(u), v]$ where $F_v(u) = X(u, v)$ is the forward mapping fct
- Second pass resamples all columns: $[x, v] \rightarrow [x, y]$
 $[x, y] = [x, G_x(v)]$ where $G_x(v) = Y(H_x(v), v)$
- $H_x(v)$ is the inverse projection of x' , the column we wish to resample.
- It brings us back from $[x, v]$ to $[u, v]$ so that we can directly index into Y to get the destination y coordinates.

Example: Rotation (1)

$$[x, y] = [u, v] \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

Pass 1: $[x, v] = [u \cos \theta - v \sin \theta, v]$

Pass 2: a) Compute $H_x(v)$. Recall that $x = u \cos \theta - v \sin \theta$

$$u = \frac{x + v \sin \theta}{\cos \theta}$$

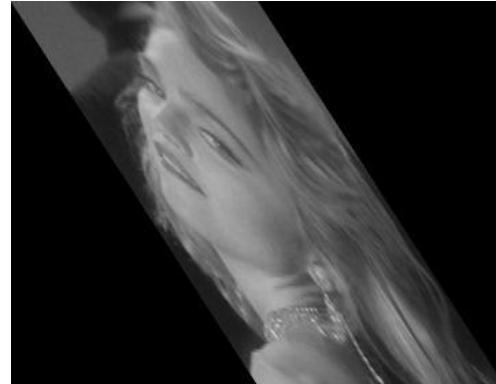
b) Compute $G_x(v)$. Substitute $H_x(v)$ into $y = u \sin \theta + v \cos \theta$

$$y = \frac{x \sin \theta + v}{\cos \theta}$$

2-Pass Rotation



scale/shear
rows



scale/shear
columns



3-Pass Rotation Algorithm

- Rotation can be decomposed into two scale/shear matrices.

$$R = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 \\ -\sin \theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \tan \theta \\ 0 & \frac{1}{\cos \theta} \end{bmatrix}$$

- Three pass transform uses on shear matrices.

$$R = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\tan\left(\frac{\theta}{2}\right) & 1 \end{bmatrix} \begin{bmatrix} 1 & \sin \theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\tan\left(\frac{\theta}{2}\right) & 1 \end{bmatrix}$$

- Advantage of 3-pass: no scaling necessary in any pass.

3-Pass Rotation



shear
rows



shear columns



shear
rows



Software Implementation

- The following slides contain code for `initMatrix.c` to produce a 3x3 perspective transformation matrix from a list of four corresponding points (e.g. image corners).
- That matrix is then used in `perspective.c` to resample the image.
- The code in `resample.c` performs the actual scanline resample.

initMatrix.c (1)

```
/*
 * initMatrix:
 *
 * Given Icorr, a list of the 4 correspondence points for the corners
 * of image I, compute the 3x3 perspective matrix in Imatrix.
 */
void initMatrix(imageP I, imageP Icorr, imageP Imatrix)
{
    int      w,    h;
    float   *p,   *a, a13, a23;
    float   x0,  x1, x2, x3;
    float   y0,  y1, y2, y3;
    float   dx1, dx2, dx3, dy1, dy2, dy3;

    /* init pointers */
    a = (float *) Imatrix->buf[0];
    p = (float *) Icorr->buf[0];

    /* init u,v,x,y vars and print them */
    x0 = *p++;      y0 = *p++;
    x1 = *p++;      y1 = *p++;
    x2 = *p++;      y2 = *p++;
    x3 = *p++;      y3 = *p++;
```

initMatrix.c (2)

```
w = I->width;
h = I->height;
UI_printf("\nCorrespondence points:\n");
UI_printf("%4d %4d %6.1f %6.1f\n", 0, 0, x0, y0);
UI_printf("%4d %4d %6.1f %6.1f\n", w, 0, x1, y1);
UI_printf("%4d %4d %6.1f %6.1f\n", w, h, x2, y2);
UI_printf("%4d %4d %6.1f %6.1f\n", 0, h, x3, y3);

/* compute auxiliary vars */
dx1 = x1 - x2;
dx2 = x3 - x2;
dx3 = x0 - x1 + x2 - x3;
dy1 = y1 - y2;
dy2 = y3 - y2;
dy3 = y0 - y1 + y2 - y3;
```

initMatrix.c (3)

```
/* compute 3x3 transformation matrix:  
 * a0 a1 a2  
 * a3 a4 a5  
 * a6 a7 a8  
 */  
a13 = (dx3*dy2 - dx2*dy3) / (dx1*dy2 - dx2*dy1);  
a23 = (dx1*dy3 - dx3*dy1) / (dx1*dy2 - dx2*dy1);  
a[0] = (x1-x0+a13*x1) / w;  
a[1] = (y1-y0+a13*y1) / w;  
a[2] = a13 / w;  
a[3] = (x3-x0+a23*x3) / h;  
a[4] = (y3-y0+a23*y3) / h;  
a[5] = a23 / h;  
a[6] = x0;  
a[7] = y0;  
a[8] = 1;  
}
```

perspective.c (1)

```
#define X(A, U, V)          ((A[0]*U + A[3]*V + A[6]) / (A[2]*U + A[5]*V + A[8]))
#define Y(A, U, V)          ((A[1]*U + A[4]*V + A[7]) / (A[2]*U + A[5]*V + A[8]))
#define H(A, X, V)          (((- (A[5]*V+A[8])*X+ A[3]*V + A[6]) / (A[2]*X - A[0])))

/*
 * perspective:
 *
 * Apply a perspective image transformation on input I1.
 * The 3x3 perspective matrix is given in Imatrix.
 * The output is stored in I2.
 */
void perspective(imageP I1, imageP Imatrix, imageP I2)
{
    int    i, w, h, ww, hh;
    uchar *p1, *p2;
    float u, v, x, y, xmin, xmax, ymin, ymax, *a, *F;
    imageP II;

    w = I1->width;
    h = I1->height;
    a = (float *) Imatrix->buf[0];
```

perspective.c (2)

```
xmin = xmax = X(a, 0, 0);
x = X(a, w, 0); xmin = MIN(xmin, x);
x = X(a, w, h); xmin = MIN(xmin, x);
x = X(a, 0, h); xmin = MIN(xmin, x);

xmax = MAX(xmax, x);
xmax = MAX(xmax, x);
xmax = MAX(xmax, x);

ymin = ymax = Y(a, 0, 0);
y = Y(a, w, 0); ymin = MIN(ymin, y);
y = Y(a, w, h); ymin = MIN(ymin, y);
y = Y(a, 0, h); ymin = MIN(ymin, y);

ymax = MAX(ymax, y);
ymax = MAX(ymax, y);
ymax = MAX(ymax, y);

ww = CEILING(xmax) - FLOOR(xmin);
hh = CEILING(ymax) - FLOOR(ymin);

/* allocate mapping fct buffer */
x = MAX(MAX(w, h), MAX(ww, hh));
F = (float *) malloc(x * sizeof(float));
if(F == NULL) IP_bailout("perspective: No memory");

/* allocate intermediate image */
II = IP_allocImage(ww, h, BW_TYPE);
IP_clearImage(II);
p1 = (uchar *) II->buf[0];
p2 = (uchar *) II->buf[0];
```

perspective.c (3)

```
/* first pass: resample rows */
for(v=0; v<h; v++) {
    /* init forward mapping function F; map xmin to 0 */
    for(u=0; u< w; u++) F[(int) u] = X(a, u, v) - xmin;

    resample(p1, w, 1, F, p2);
    p1 += w;
    p2 += ww;
}

/* display intermediate image */
IP_copyImage(II, NextImageP);
IP_displayImage();

/* init final image */
IP_copyImageHeader(I1, I2);
I2->width  = ww;
I2->height = hh;
IP_initChannels(I2, BW_TYPE);
IP_clearImage(I2);
```

perspective.c (4)

```
/* second pass: resample columns */
for(x=0; x<ww; x++) {
    p1 = (uchar *) II->buf[0] + (int) x;
    p2 = (uchar *) I2->buf[0] + (int) x;

    /* skip past padding */
    for(v=0; v<h; v++,p1+=ww) {
        if(*p1) break;                  /* check for nonzero pixel */
        u = H(a, (x+xmin), v);        /* else, if pixel is black */
        if(u>=0 && u<w) break;       /* then check for valid u */
    }

    /* init forward mapping function F; map ymin to 0 */
    for(i=0; v<h; v++) {
        u = H(a, (x+xmin), v);
        u = CLIP(u, 0, w-1);
        F[i++] = Y(a, u, v) - ymin;
    }
    resample(p1, i, ww, F, p2);
}
IP_freeImage(II);
}
```

resample.c (1)

```
/*
 * Resample the len elements of src (with stride offst) into dst according
 * to the monotonic spatial mapping given in F (len entries).
 * The scanline is assumed to have been cleared earlier.
 */
void resample(uchar *src, int len, int offst, float *F, uchar *dst)
{
    int      u, uu, x, xx, ix0, ix1, I0, I1, pos;
    double  x0, x1, dI;

    if(F[0] < F[len-1])
        pos = 1;                  /* positive output stride */
    else    pos = 0;                /* negative output stride */

    for(u=0; u<len-1; u++) {
        /* index into src */
        uu = u * offst;

        /* output interval (real and int) for input pixel u */
        if(pos) { /* positive stride */
            ix0 = x0 = F[u];
            ix1 = x1 = F[u+1];
            I0  = src[uu];
            I1  = src[uu+offst];
        }
    }
}
```

Wolberg: Image Processing Course Notes

resample.c (2)

```
else { /* flip interval to enforce positive stride */
    ix0 = x0 = F[u+1];
    ix1 = x1 = F[u];
    I0  = src[uu+offst];
    I1  = src[uu];
}

/* index into dst */
xx = ix0 * offst;

/* check if interval is embedded in one output pixel */
if(ix0 == ix1) {
    dst[xx] += I0 * (x1-x0);
    continue;
}
/* else, input straddles more than one output pixel */

/* left straddle */
dst[xx] += I0 * (ix0+1-x0);
```

resample.c (3)

```
/* central interval */
xx += offst;
dI = (I1-I0) / (x1-x0);
for(x=ix0+1; x<ix1; x++,xx+=offst)
    dst[xx] = I0 + dI*(x-x0);

/* right straddle */
if(x1 != ix1)
    dst[xx] += (I0 + dI*(ix1-x0)) * (x1-ix1);

}
```

Image Warping: A Review

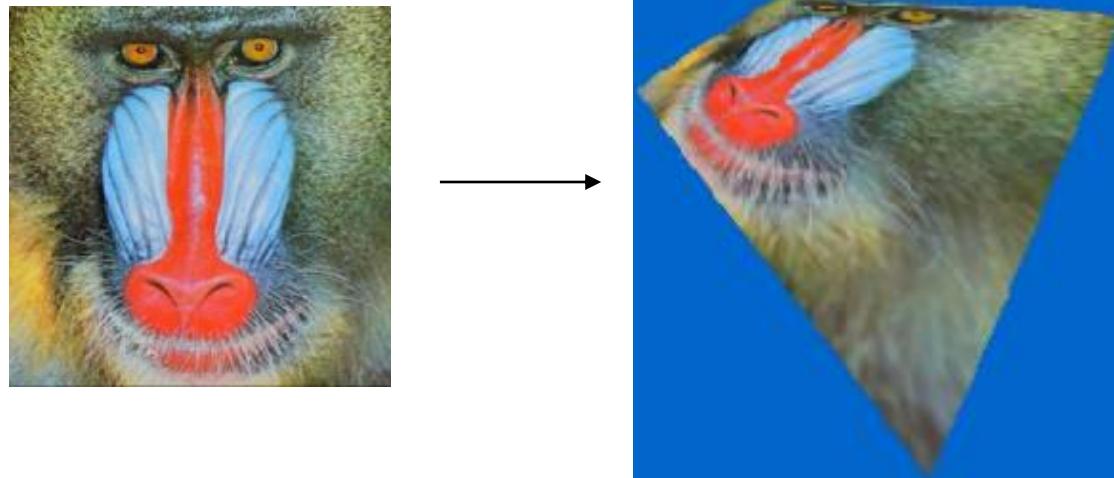
Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we review digital image warping:
 - Geometric transformations
 - Forward inverse mapping
 - Sampling
 - Image reconstruction
 - Interpolation kernels
 - Separable transforms
 - Fant's resampling algorithm

Definition

- Image warping deals with the geometric transformation of digital images.



Geometric Transformations

- Affine
- Perspective
- Bilinear
- Polynomial
- Splines
- Elastic (local deformations)

Spatial Transformations

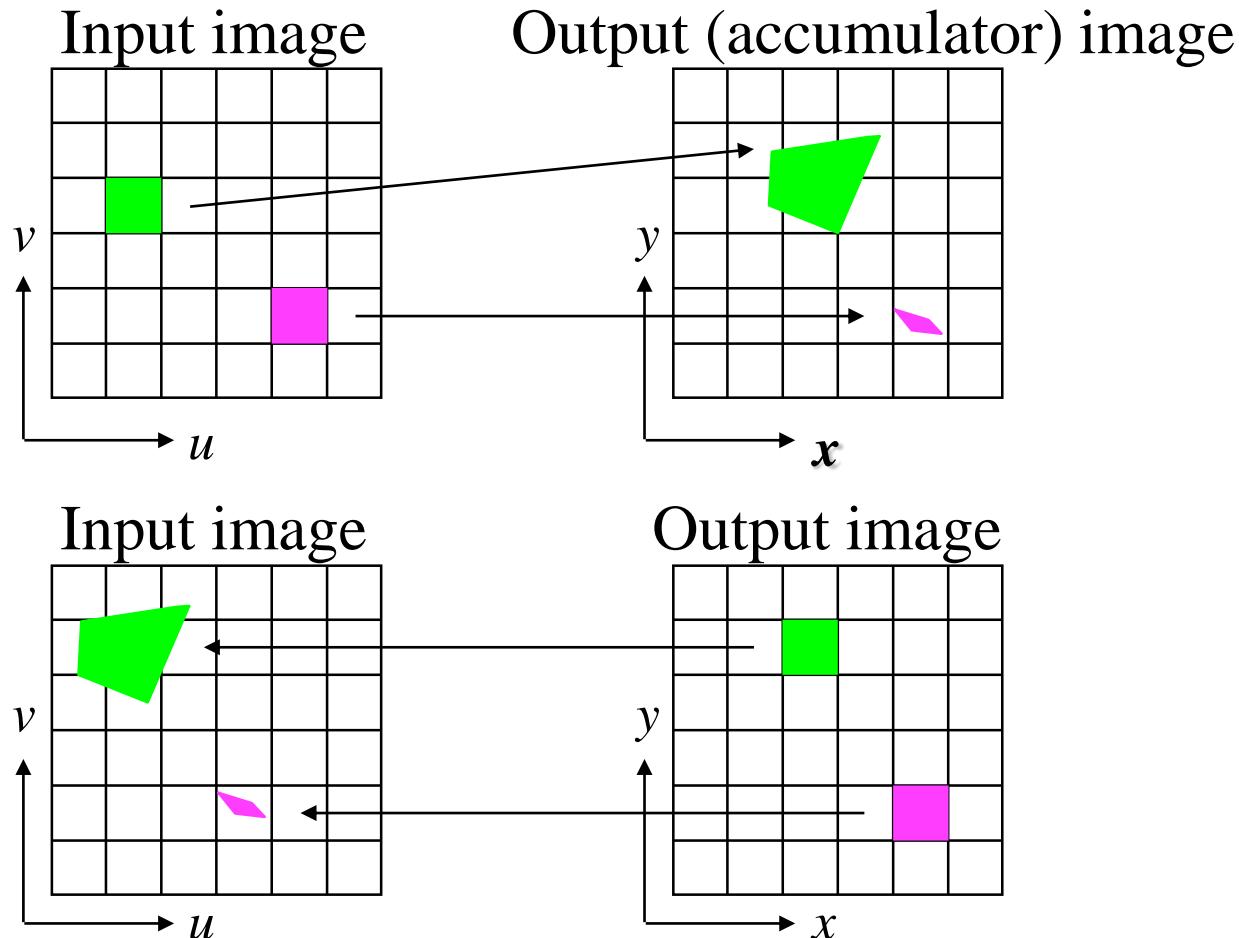
- Forward Mapping

$$[x, y] = [X(u, v), Y(u, v)]$$

- Inverse Mapping

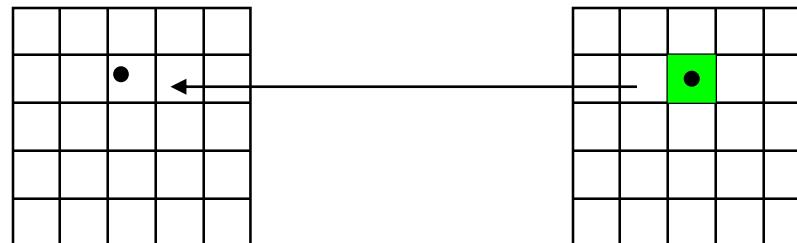
$$[u, v] = [U(x, y), V(x, y)]$$

Forward / Inverse Mapping

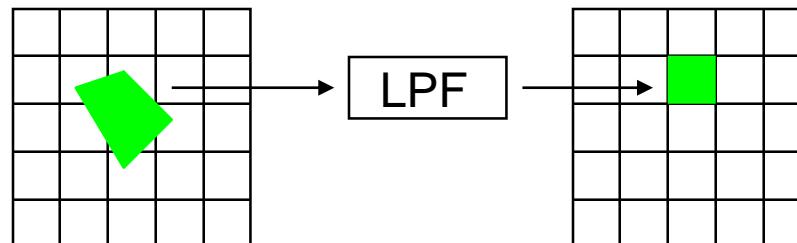


Sampling

- Point Sampling



- Area Sampling



Area Sampling

- Treats pixels as finite areas
- Avoids aliasing (undersampling) artifacts
- Approximated by supersampling

Supersampling

- Average of projected subpixels

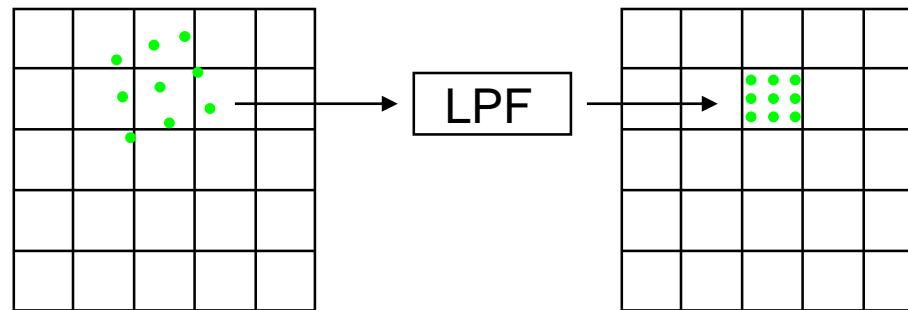
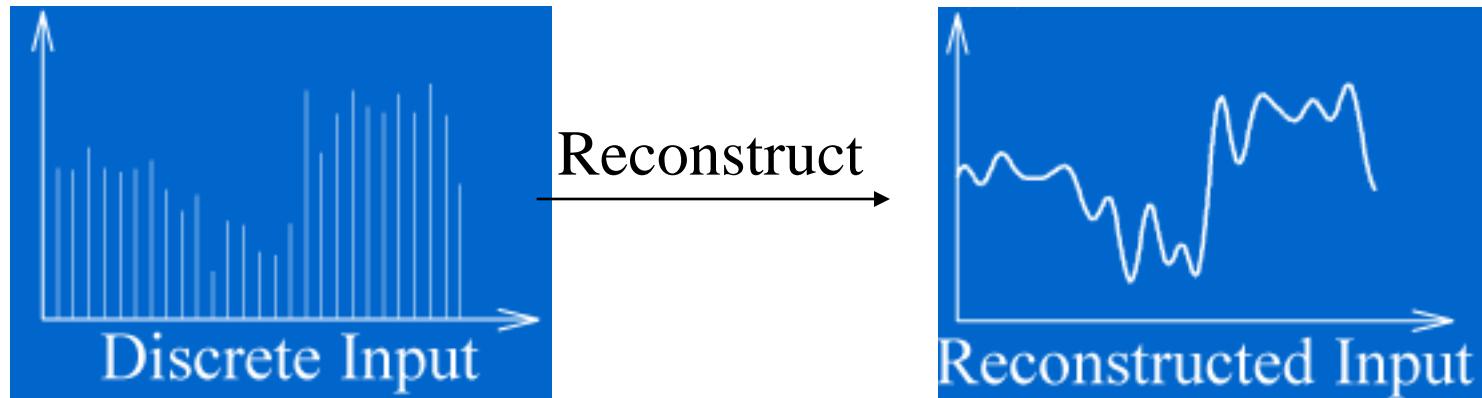


Image Reconstruction

- Pixel values are known at integer positions
- Samples can project to real-valued positions
- How do we evaluate the image values at these real-valued positions? Reconstruction

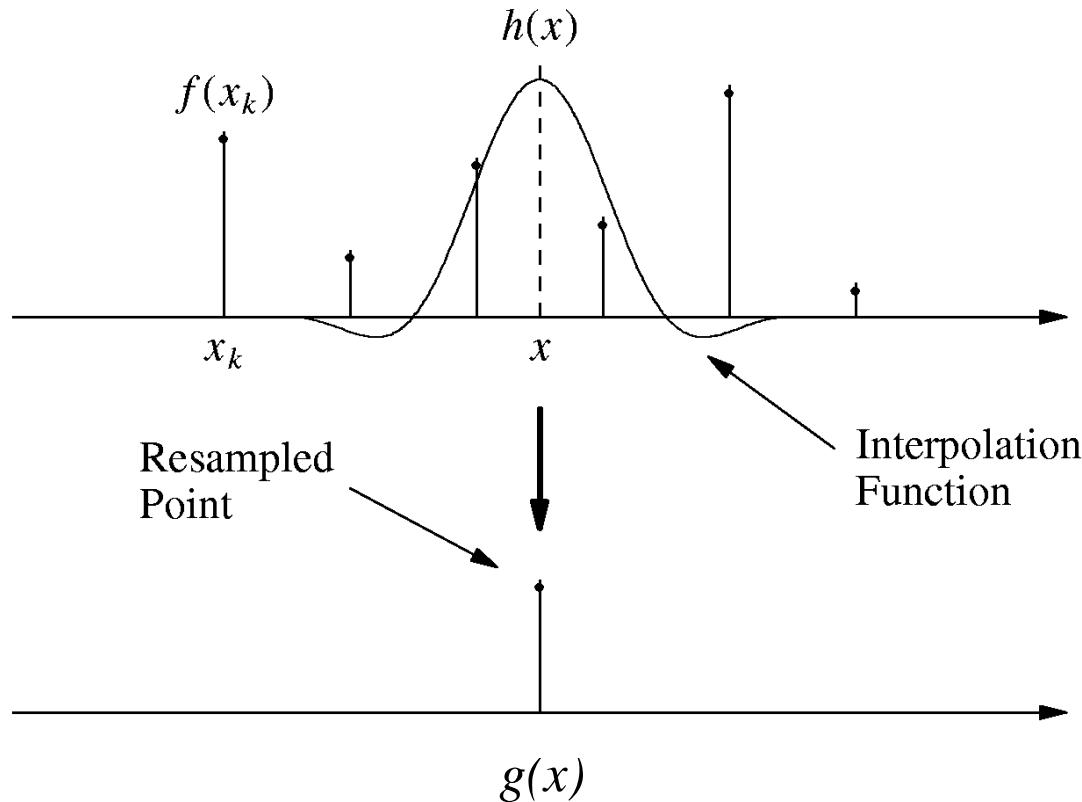


Interpolation

- Reconstruction interpolates the input
- In practice, interpolation is performed at points of interest only, not entire function
- Interpolation is achieved by convolution

Convolution

$$g(x) = \sum_{k=0}^N f(x_k) h(x - x_k)$$



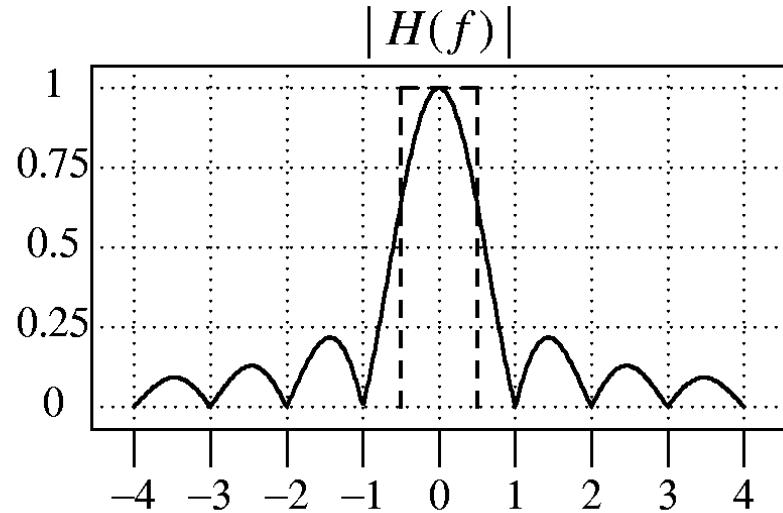
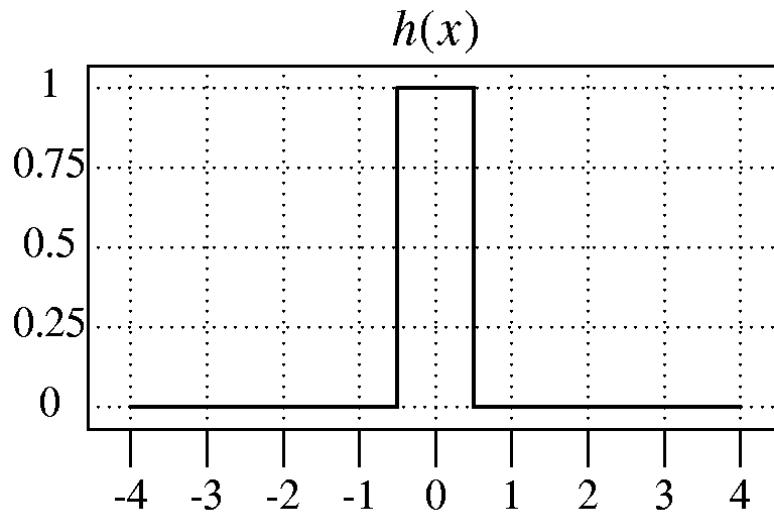
Interpolation Functions

Interpolation functions/kernels include:

- Box filter
- Triangle filter
- Cubic convolution
- Windowed sinc functions

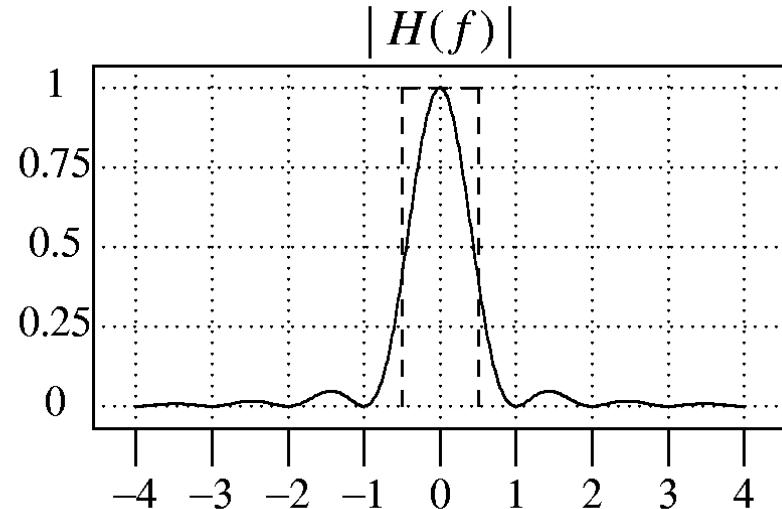
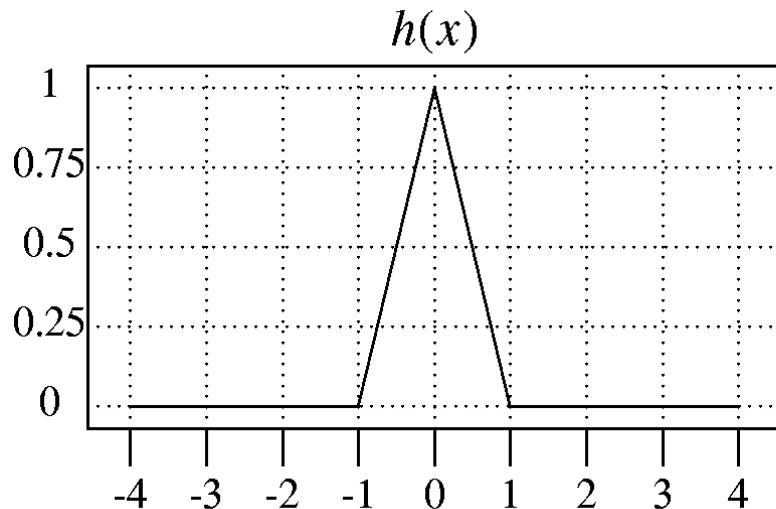
Box Filter

- Nearest neighbor interpolation
- Blocky artifacts may occur



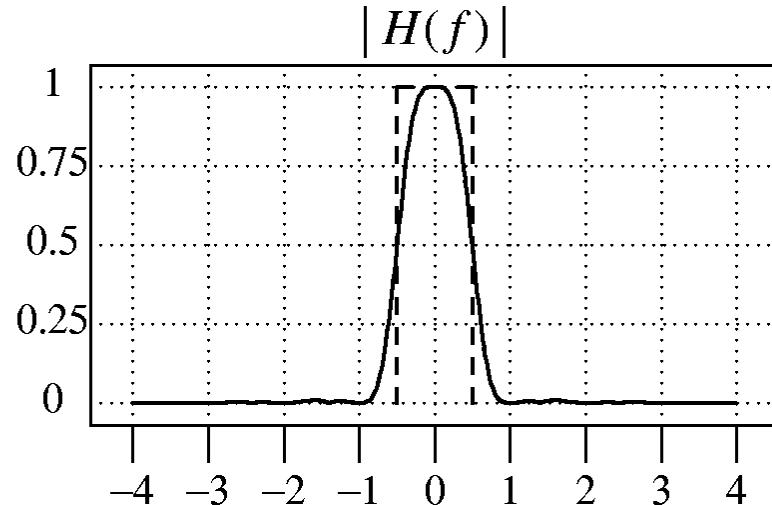
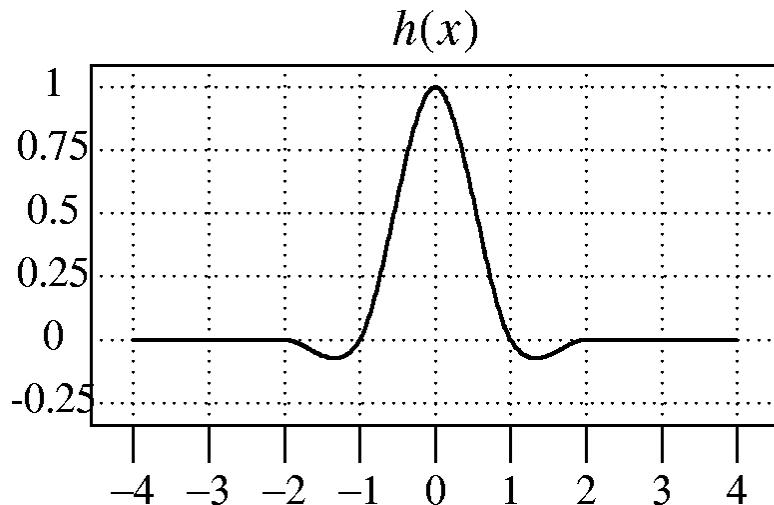
Triangle Filter

- Linear interpolation
- Popular for use with small deformations



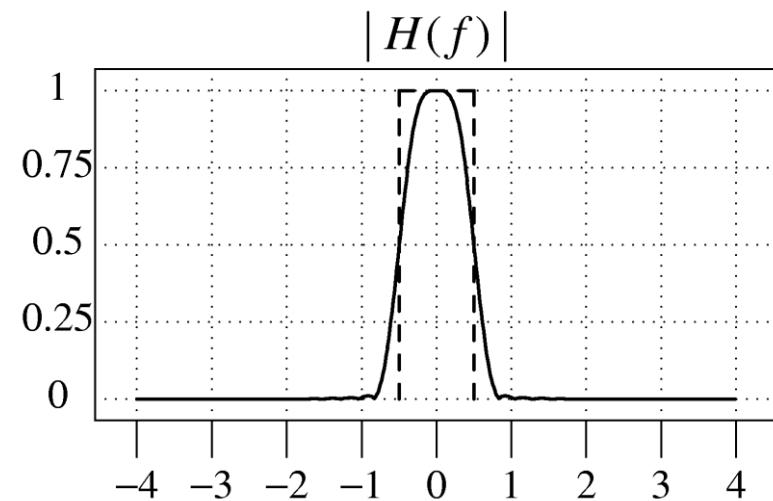
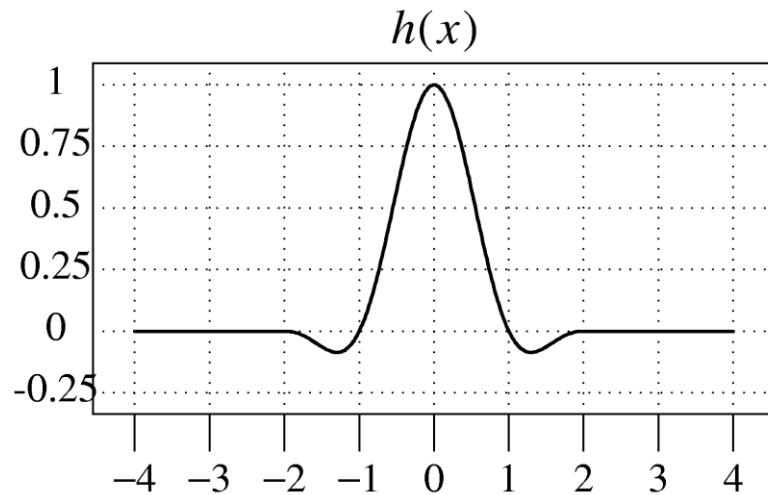
Cubic Convolution

- Local cubic interpolation algorithm
- Advanced feature in digital cameras

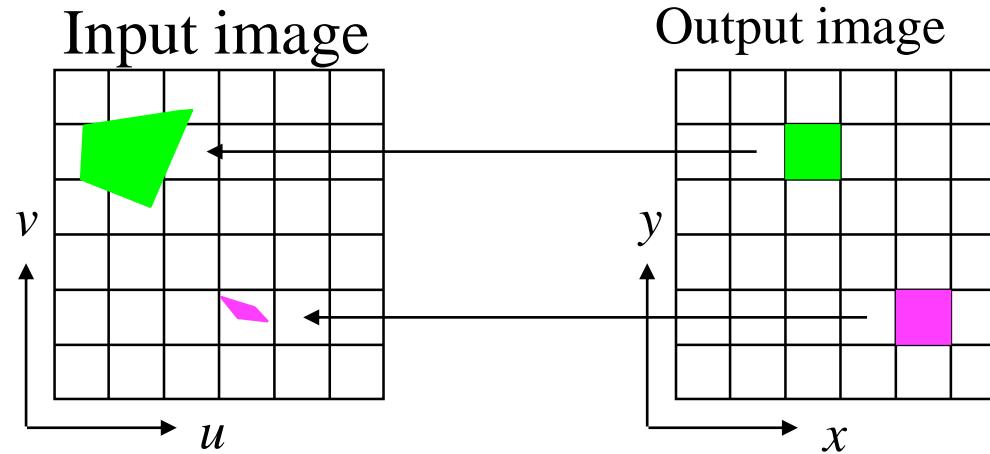


Windowed Sinc Function

- Smoothly tapered ideal sinc function

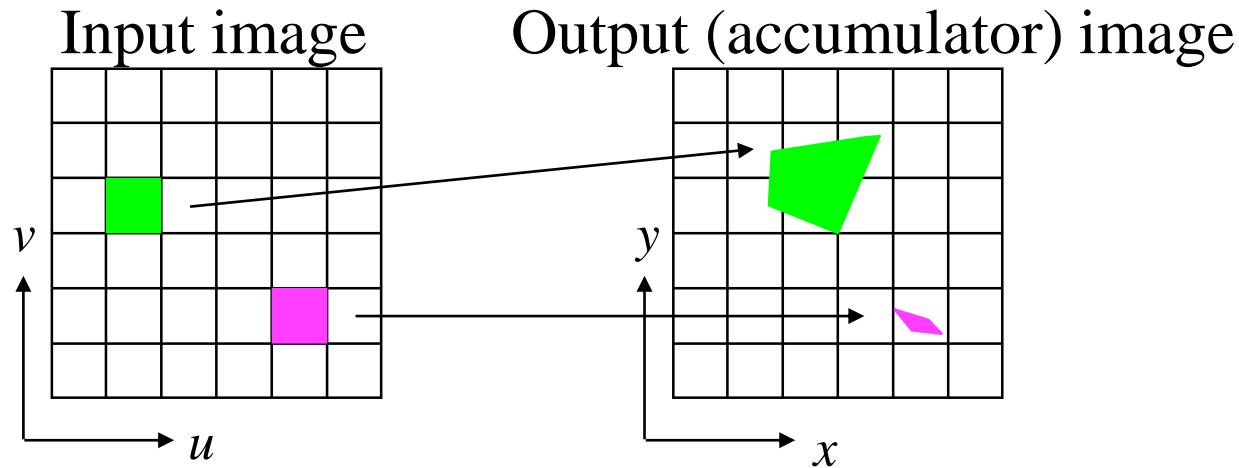


Inverse Mapping



- Visit output in scanline order
- Supersampling approximates area sampling
- Popular in computer graphics

Forward Mapping



- Visit input in scanline order
- Use output accumulator array
- 2D antialiasing is difficult
- Separable transforms facilitate efficient solution

Separable Transforms

$$[X(u, v), Y(u, v)] = F(u, v) \circ G(x, v)$$

- $F(u, v)$ is a row-preserving transformation that maps all input points to their final column positions, i.e., $[x, v]$.
- $G(x, v)$ is a column-preserving transformation that maps the $[x, v]$ points to their final row positions, i.e., $[x, y]$.

Catmull-Smith Algorithm

- **First pass**

Maps image $S(u,v)$ into intermediate image $I(x,v)$

$$I(x,v) = S(X(u,v), v)$$

- **Second pass**

Maps $I(x,v)$ into target image $T(x,y)$

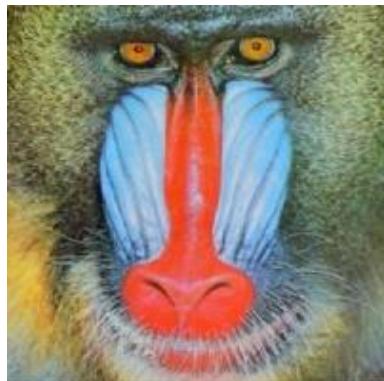
$$T(x,y) = I(x, Y(H_x(v), v))$$

where H_x is the solution to $x=X(u,v)$ for u

2-Pass Rotation

 F  X, Y G 

2-Pass Perspective



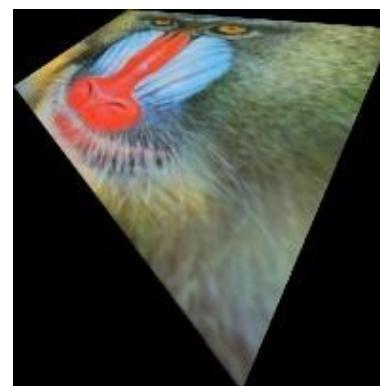
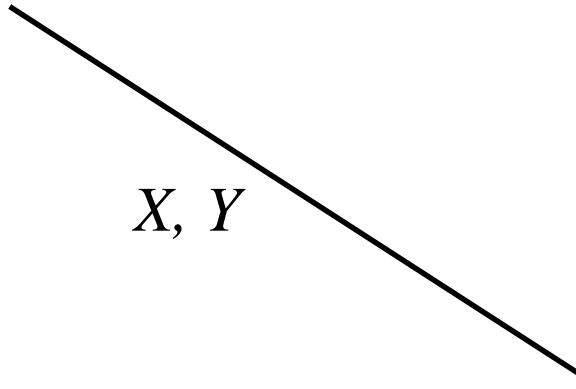
F



G



X, Y



Fant's Algorithm

- Forward mapping intensity resampling
- Scanline order in input *and* output
- Amenable to hardware implementation

Fant's algorithm: Example (1)

$XLUT$	<table border="1"><tr><td>.6</td><td>2.3</td><td>3.2</td><td>3.3</td><td>3.9</td></tr></table>	.6	2.3	3.2	3.3	3.9
.6	2.3	3.2	3.3	3.9		
$YLUT$	<table border="1"><tr><td>100</td><td>106</td><td>115</td><td>120</td></tr></table>	100	106	115	120	
100	106	115	120			
I	<table border="1"><tr><td>100</td><td>106</td><td>92</td><td>90</td></tr></table>	100	106	92	90	
100	106	92	90			
$YLUT_x$	<table border="1"><tr><td>100</td><td>101</td><td>105</td><td>113</td></tr></table>	100	101	105	113	
100	101	105	113			
I_x	<table border="1"><tr><td>40</td><td>101</td><td>106</td><td>82</td></tr></table>	40	101	106	82	
40	101	106	82			

Fant's algorithm: Example (2)

$$I_x(0) = (100)((.4)) = 40$$

$$I_x(1) = \left[(100) \left[1 - \frac{.4}{1.7} \right] + (106) \left[\frac{.4}{1.7} \right] \right] ((1)) = 101$$

$$I_x(2) = \left[(100) \left[1 - \frac{1.4}{1.7} \right] + (106) \left[\frac{1.4}{1.7} \right] \right] ((.3)) + (106)((.7)) = 106$$

$$I_x(3) = \left[(106) \left[1 - \frac{.7}{.9} \right] + (92) \left[\frac{.7}{.9} \right] \right] ((.2)) + (92)((.1)) + (90)((.6)) = 82$$

Bibliography

- Catmull, E. and A.R. Smith, “*3-D Transformations of Images in Scanline Order*,” Proc. Siggraph ‘80, pp. 279-285, 1980.
- Fant, Karl M., “*A Nonaliasing, Real-Time Spatial Transform Technique*,” IEEE Computer Graphics and Applications, vol. 6, no. 3, pp. 71-80, 1986.
- Wolberg, George, *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA 1990.

Image Morphing

Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we review digital image morphing, a powerful visual effect for creating a fluid transformation from one image to another.
 - Background
 - Morphing Algorithms
 - Mesh (parametric grid)
 - Line pairs
 - Scattered points
 - Examples

Inbetween Image Generation

Cross-dissolve vs. Image Morphing

cross-dissolve

**source
image**



**dest.
image**

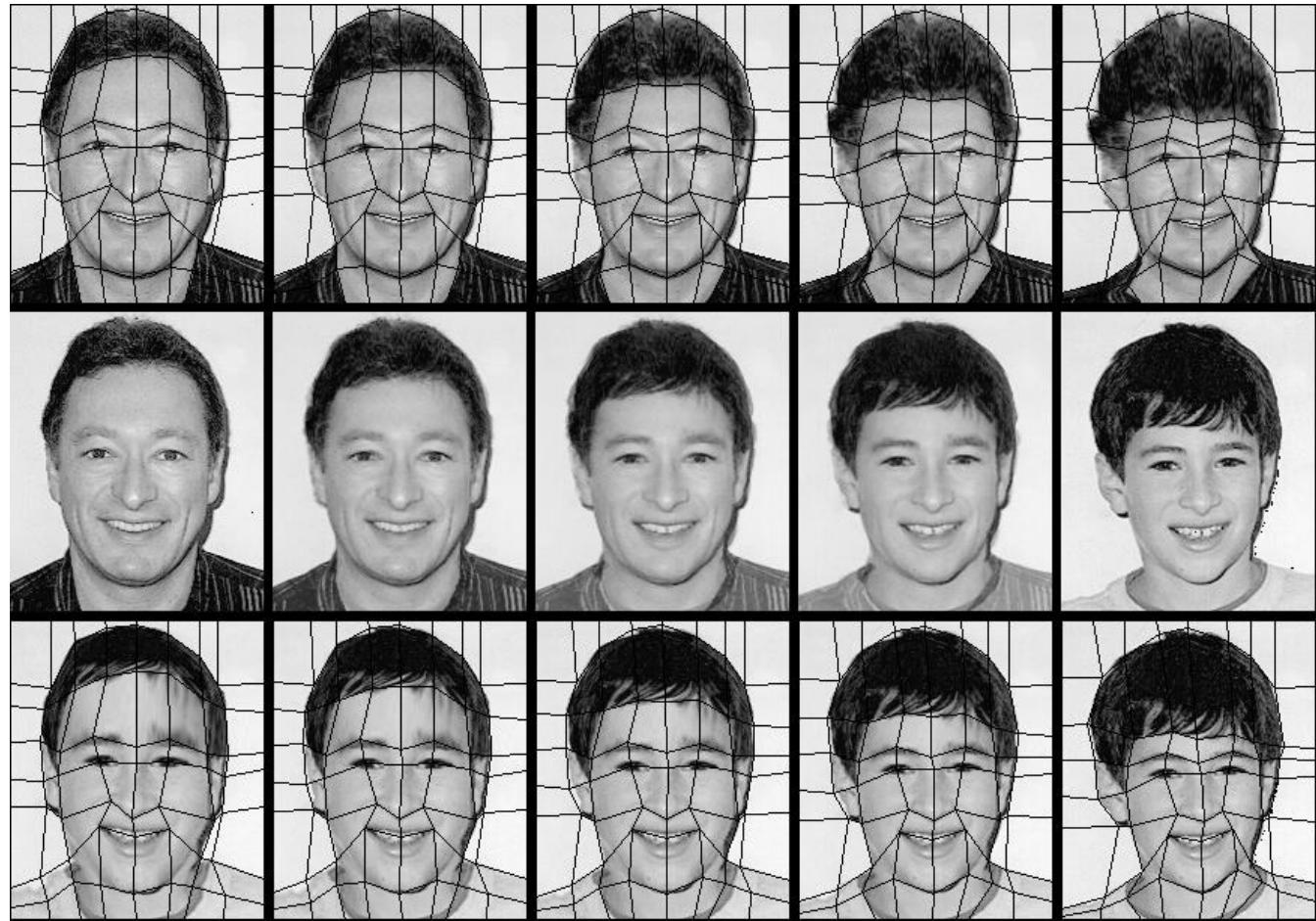
**warped
source
image**



**warped
dest.
image**

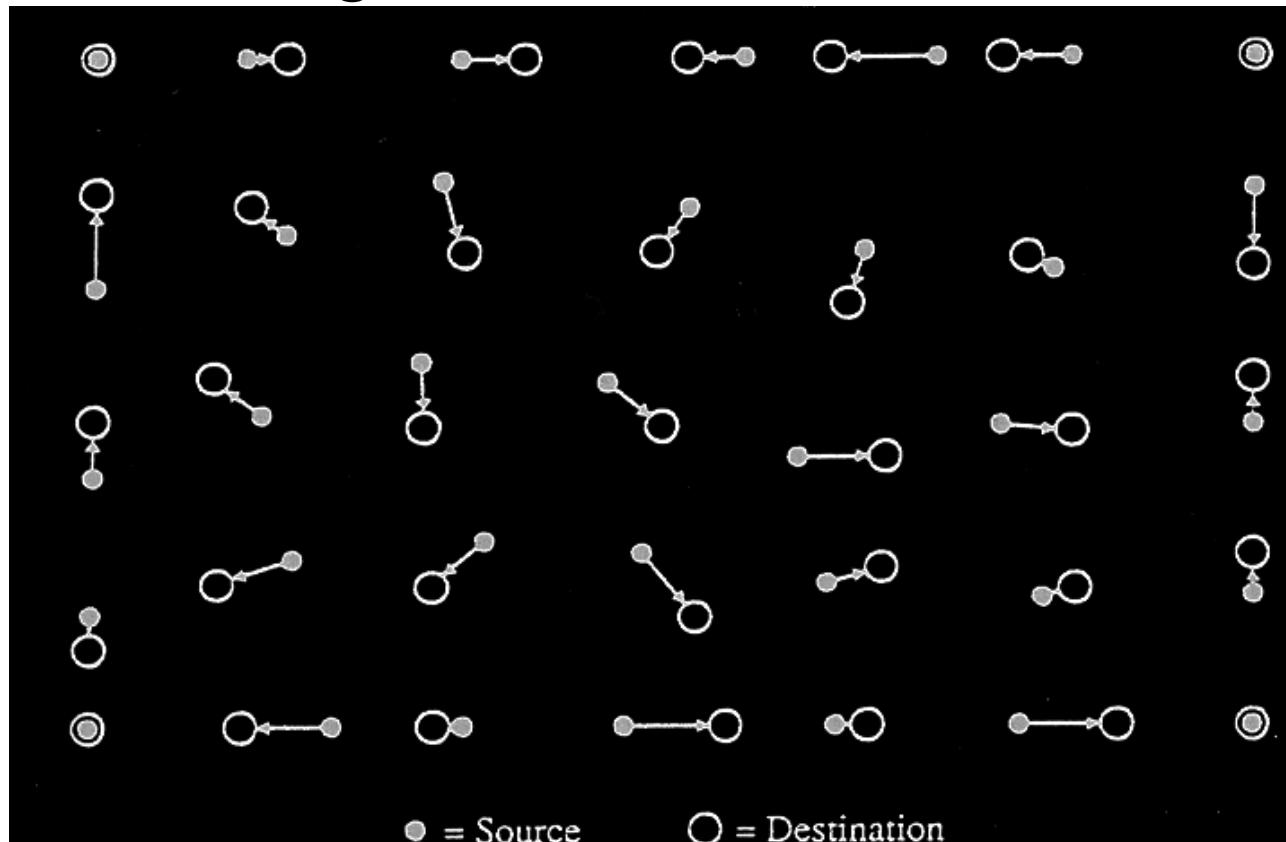
image morphing

Mesh Warp (1)



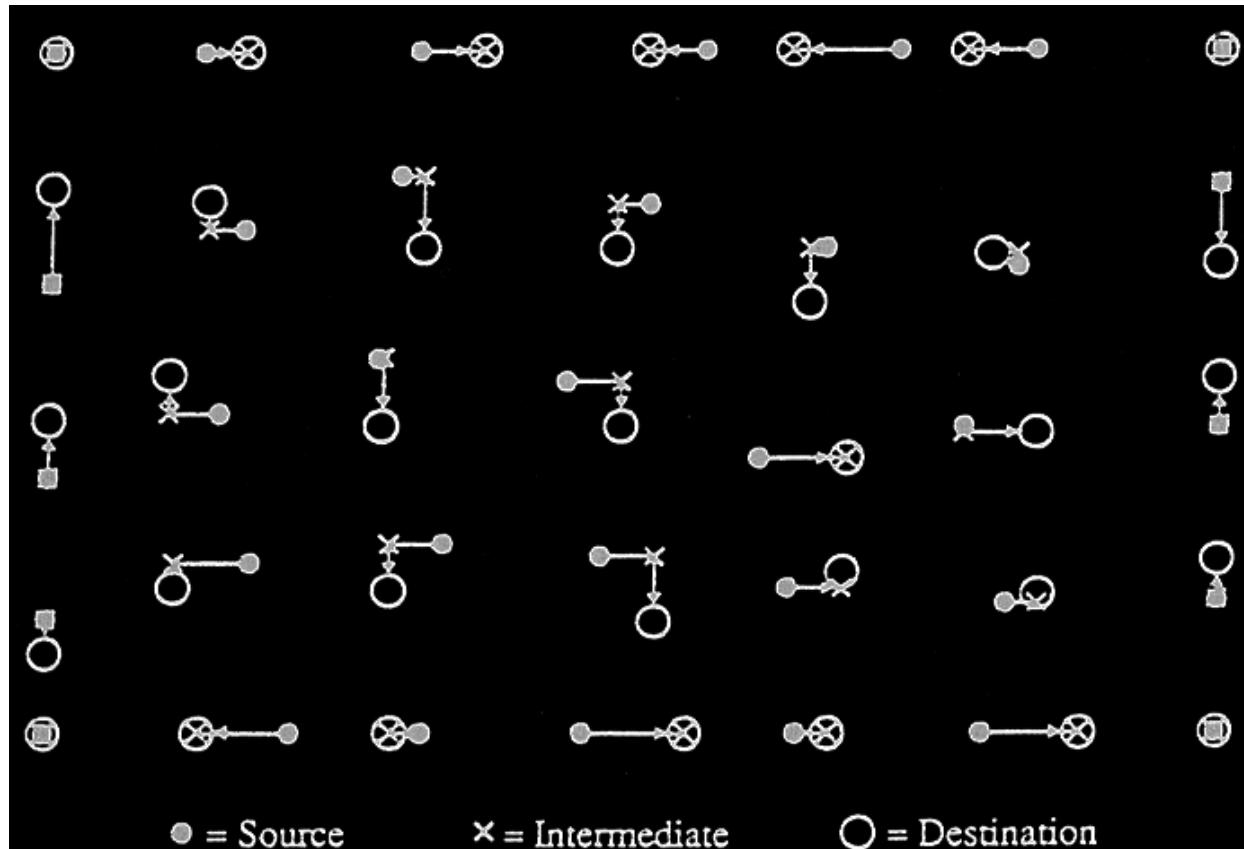
Mesh Warp (2)

Source / Target Meshes



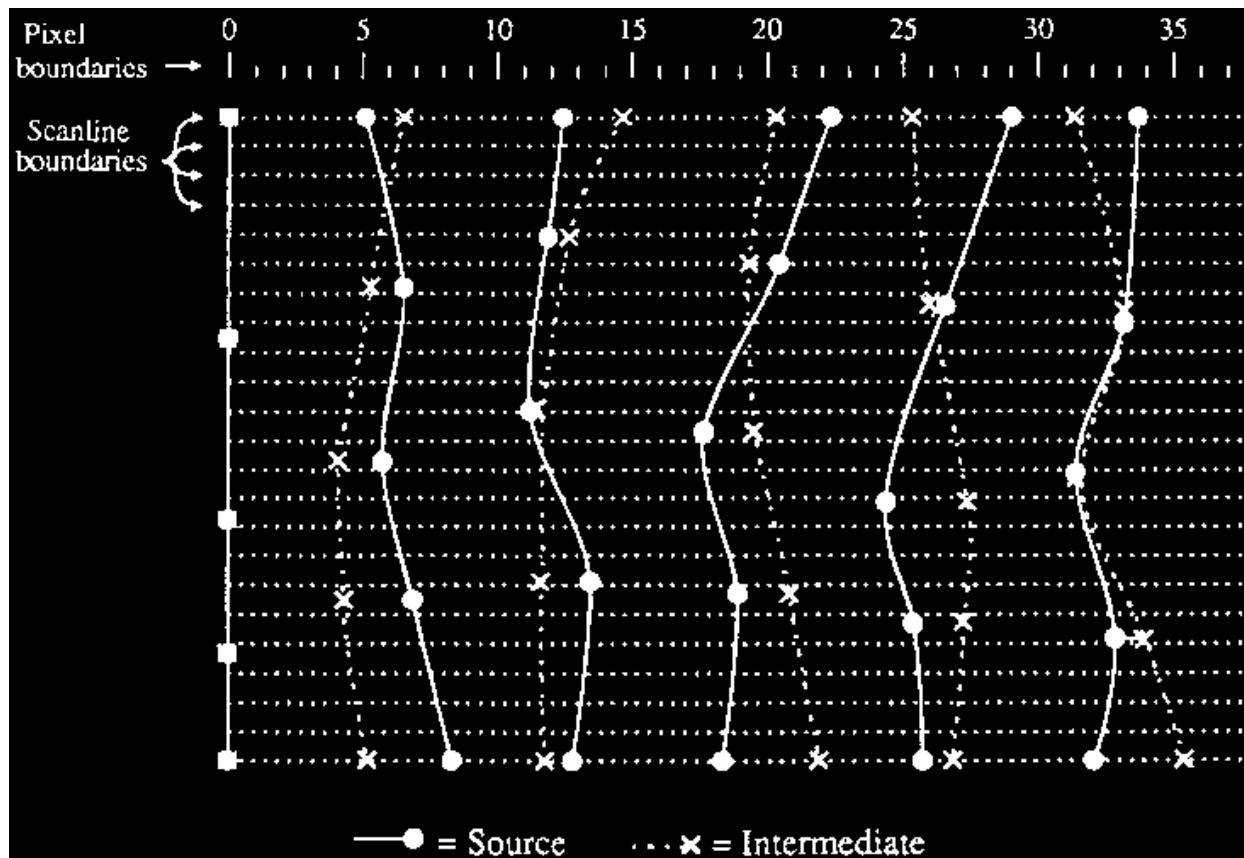
Mesh Warp (3)

Intermediate Mesh



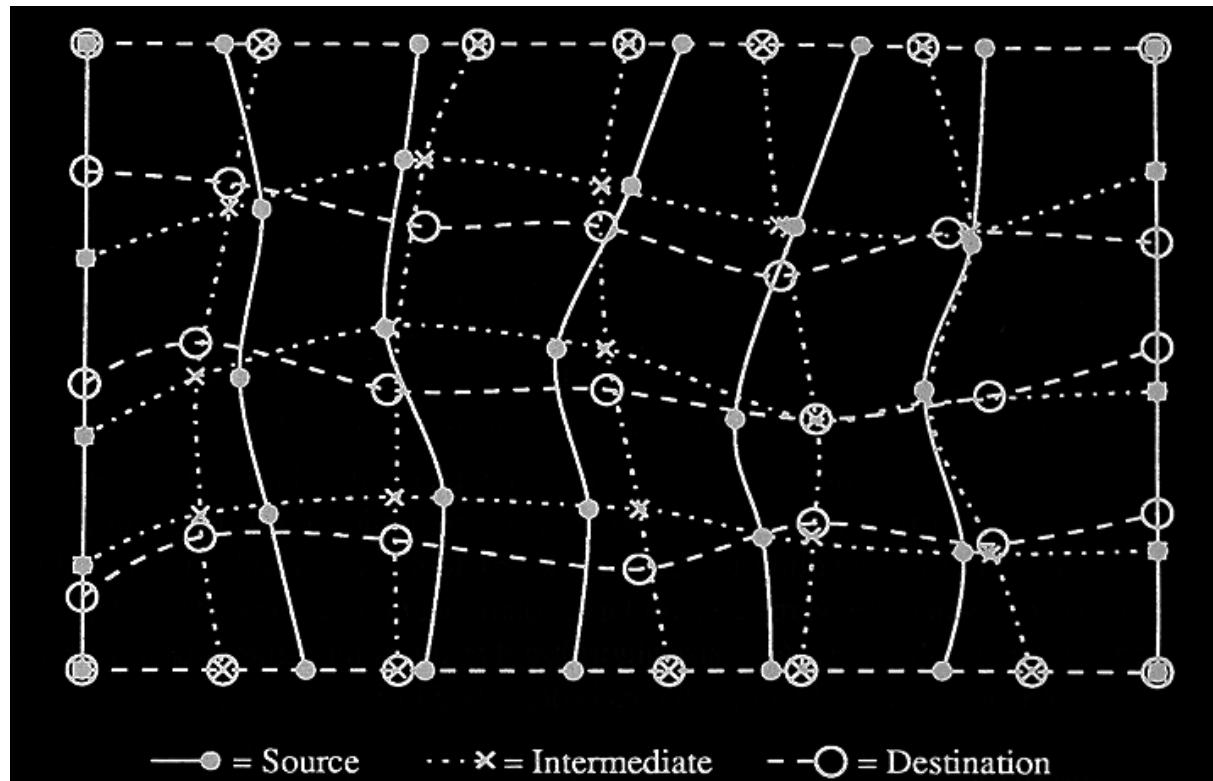
Mesh Warp (4)

Horizontal Resampling

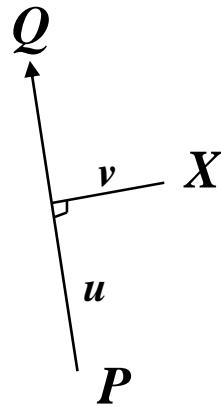


Mesh Warp (5)

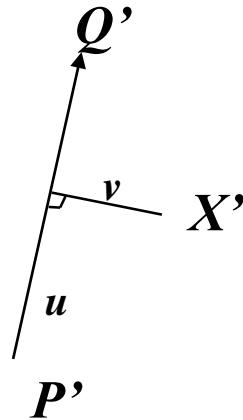
Vertical Resampling



Line Warp (1)



Destination Image

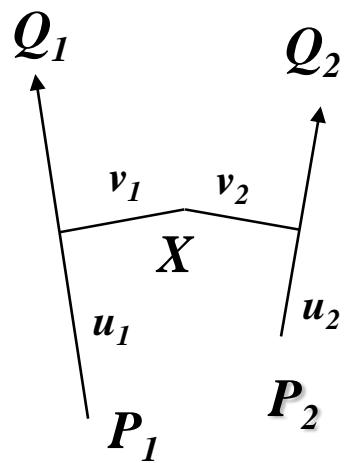


Source Image

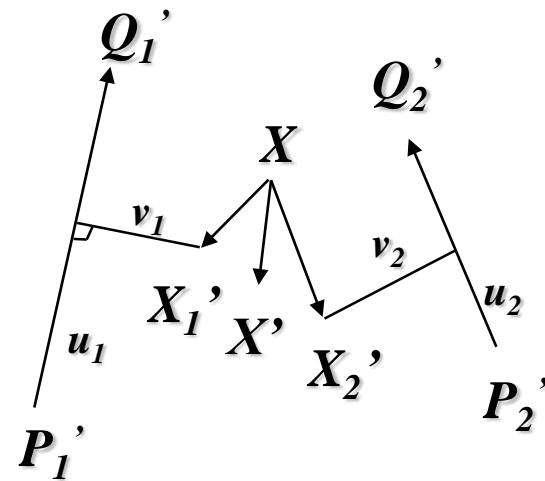
$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2} \quad v = \frac{(X - P) \cdot \text{Perp}(Q - P)}{\|Q - P\|}$$

$$X' = P' + u \cdot (Q' - P') + \frac{v \cdot \text{Perp}(Q' - P')}{\|Q' - P'\|}$$

Line Warp (2)



Destination Image

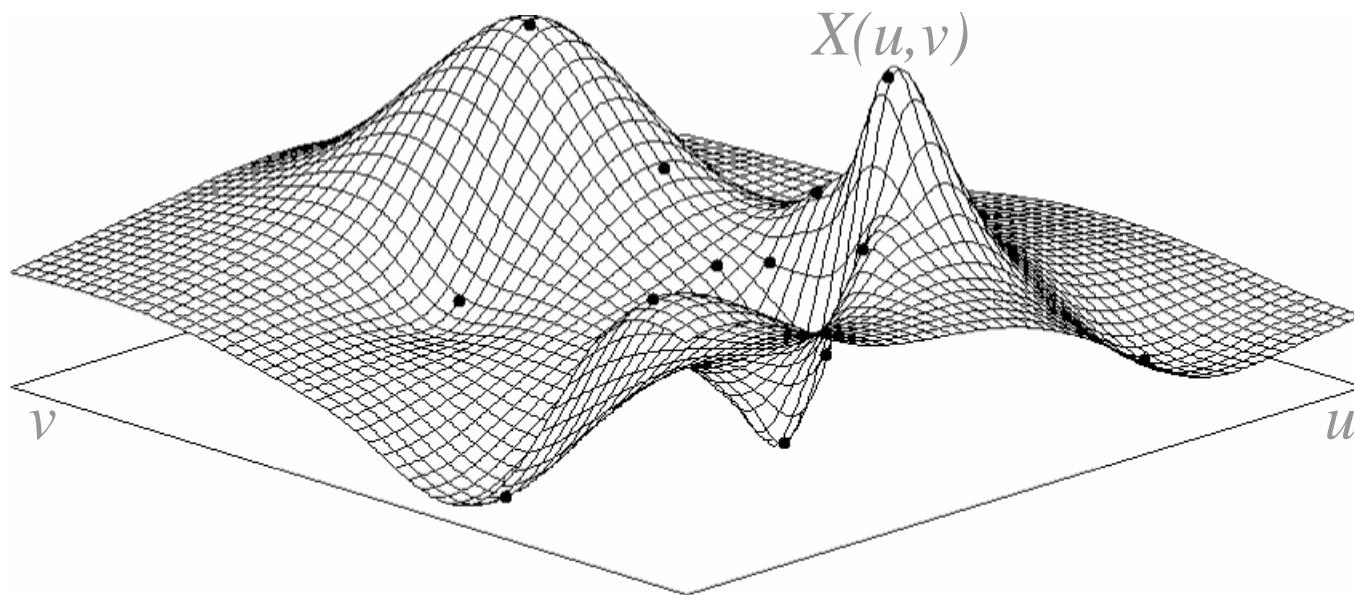


Source Image

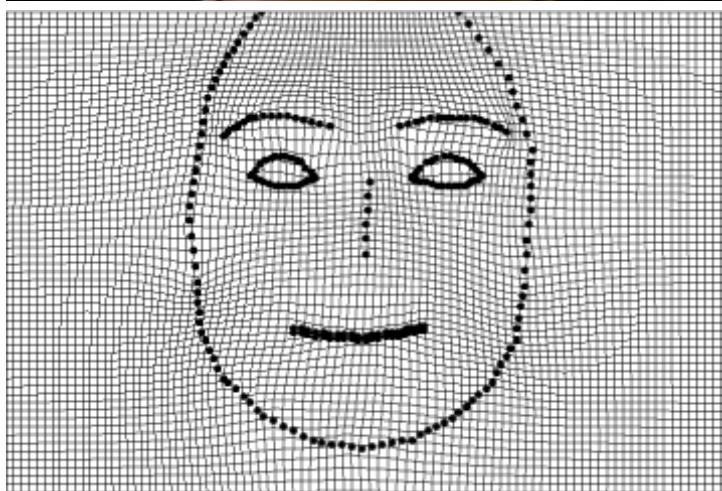
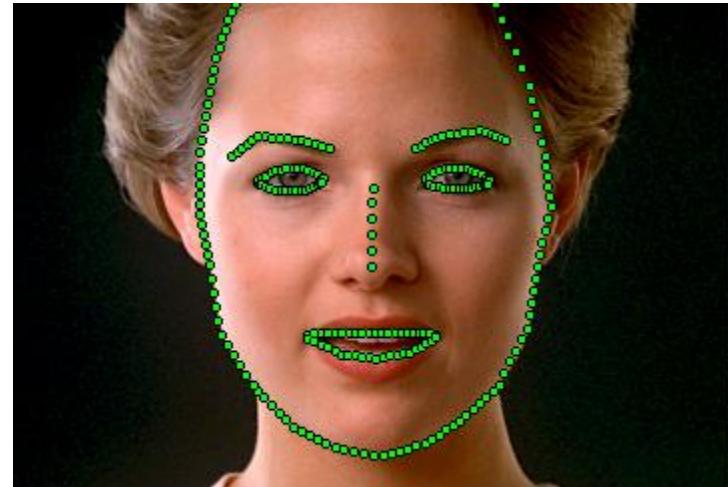
Scattered Point Constraints

Treat correspondence data as scattered data:

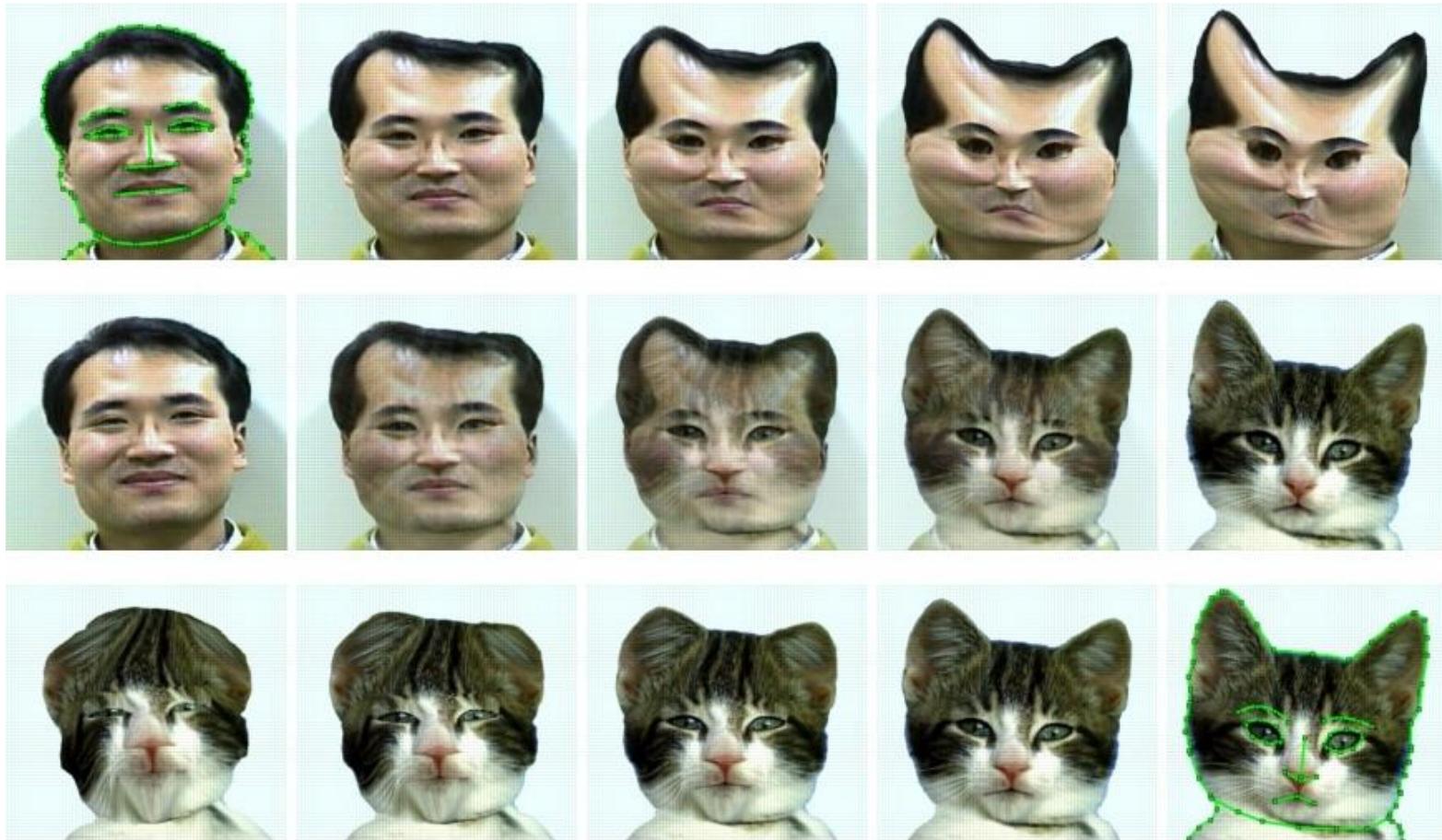
$$[x, y] = [X(u, v), Y(u, v)]$$



Example



Uniform Transition



Nonuniform Transition



Morph Sequence



Morph Sequence



Bibliography

- Beier, T. and S. Neely, "Feature-Based Image Metamorphosis," Proc. Siggraph '92, pp. 35-42, 1992.
- Smythe, D., "A Two-Pass Mesh Warping Algorithm for Object Transformation and Image Interpolation," ILM Technical Memo #1030, Computer Graphics Dept., Lucasfilm Ltd., 1990.
- Lee, S., K.Y. Chwa, S.Y. Shin, and G. Wolberg, "Image Metamorphosis Using Snakes and Free-Form Deformations," Proc. Siggraph '95, pp. 439-448, 1992.
- Lee, S., G. Wolberg, K.Y. Chwa, and S.Y. Shin, "Image Metamorphosis with Scattered Feature Constraints," IEEE Trans. Visualization and Computer Graphics, vol. 2, no. 4, pp. 337-354, 1996.
- Wolberg, G., "Image Morphing: A Survey", Visual Computer, vol. 14, no. 8/9, pp. 360-372, 1998.
- Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, CA 1990.

Robust Log-Polar Registration

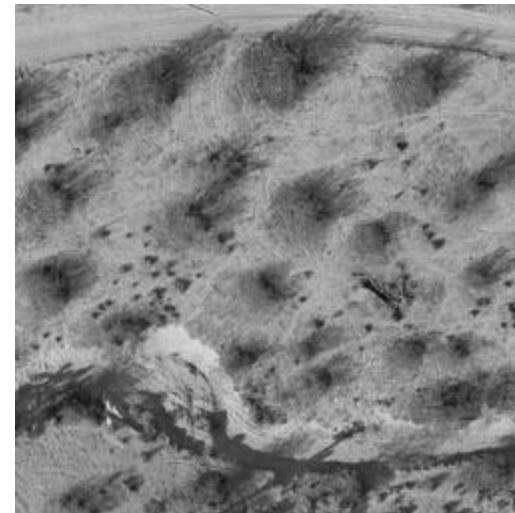
Prof. George Wolberg
Dept. of Computer Science
City College of New York

Objectives

- In this lecture we review image registration, a process to align multiple images together.
 - Affine registration
 - Perspective registration
 - Polar coordinates
 - Log-polar algorithm
 - Examples

Introduction

- Image registration refers to the problem of aligning a set of images.
- The images may be taken at different times, by different sensors, or from different viewpoints.



Applications

- Multisensor data fusion
 - integrating information taken from different sensors
- Image analysis / surveillance / change detection
 - for images taken at different times/conditions
- Image mosaics
 - generating large panoramas from overlapping images
- Super-resolution images / noise removal
 - integrating multiple registered images

Geometric Transformations

Geometric transformation models for registration:

- Rigid transformation
 - translation, rotation (3 parameters)
- Affine transformation
 - translation, rotation, scale, shear (6 parameters)
- Perspective transformation
 - affine, perspective foreshortening (8 parameters)
- Local transformation
 - terrain relief, nonlinear, nonrigid

Parameter Estimation (1)

Moderate perspective
 $|\beta|, |\gamma| < 40^\circ$



Severe perspective
 $40^\circ < |\beta|, |\gamma| < 90^\circ$



Parameter Estimation (2)

The objective function (similarity measure):

$$\chi^2(\mathbf{a}) = \sum_{i=1}^N (I_1(x, y) - I_2(Q_a\{x', y'\}))^2$$

$Q\{\}$ is the affine or perspective transform applied to image I_2 . We use the Levenberg-Marquardt method to minimize the above objective function for each pyramid level.

Affine Registration

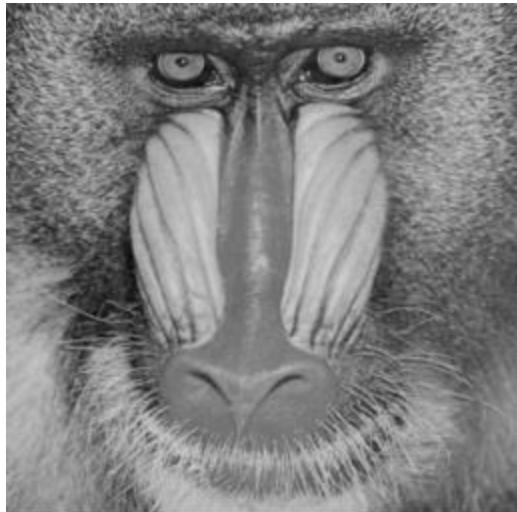


Image I_1

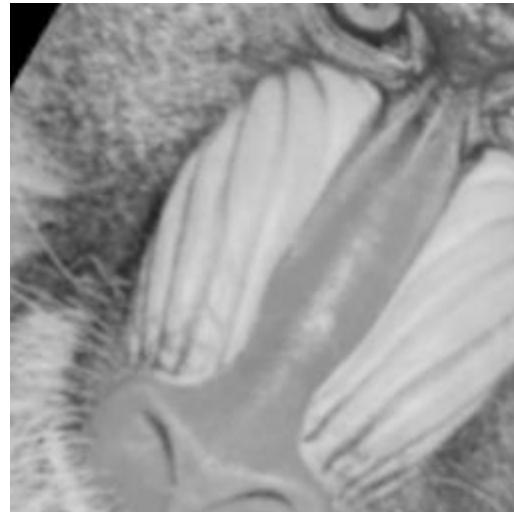
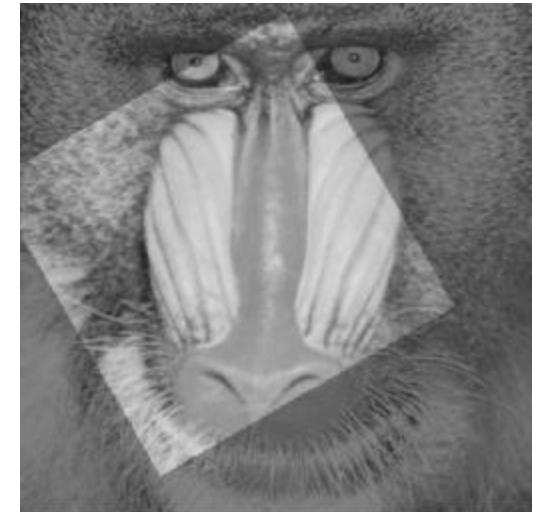


Image I_2



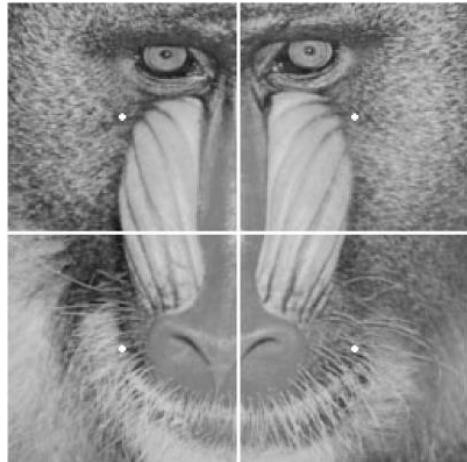
I_2 registered to I_1

Actual parameters:

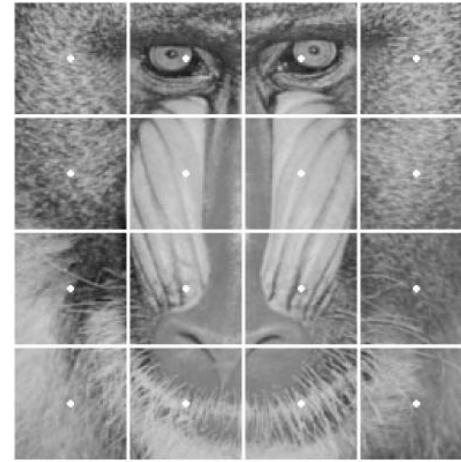
$$RST = (30^\circ, 1.5, 30, 30)$$

Estimated parameters: $RST = (29.99^\circ, 1.51, 29.51, 30.66)$

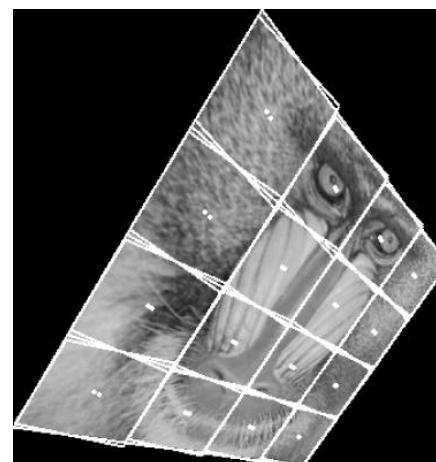
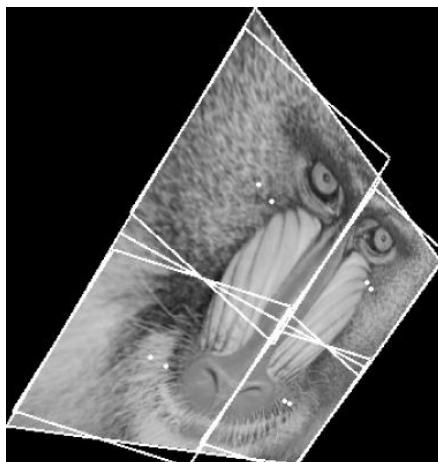
Local Affine Approximation



2 X 2 tiles



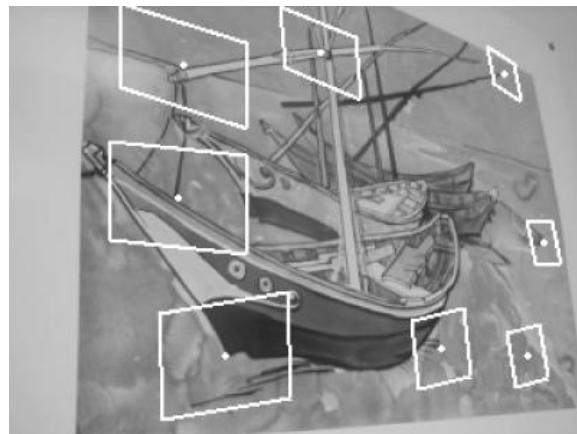
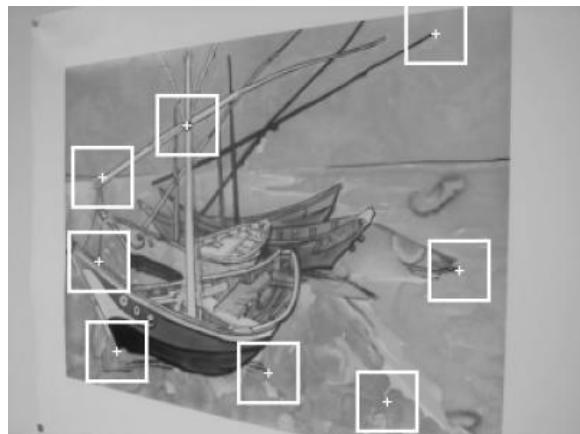
4 X 4 tiles



Piecewise Affine Approximation

- Consider I_2 subdivided into a regular grid of tiles
- For each tile $T_{i_2}^i$ search for similar tile $T_{i_1}^i$ in I_1 by performing affine registration using LMA
- Get collection of affine parameters and tile centers
- Plug into system of equations
- Solve for 8 persp. parameters using the pseudoinverse

Example (1)



Example (1)



Perspective Registration



Glare Removal

- The camera's flash introduces a highlight.
- Each highlight will appear at a different location on each image when captured from different viewpoints.
- If we register these images onto a common frame, we can render the final image by fusing the registered images.



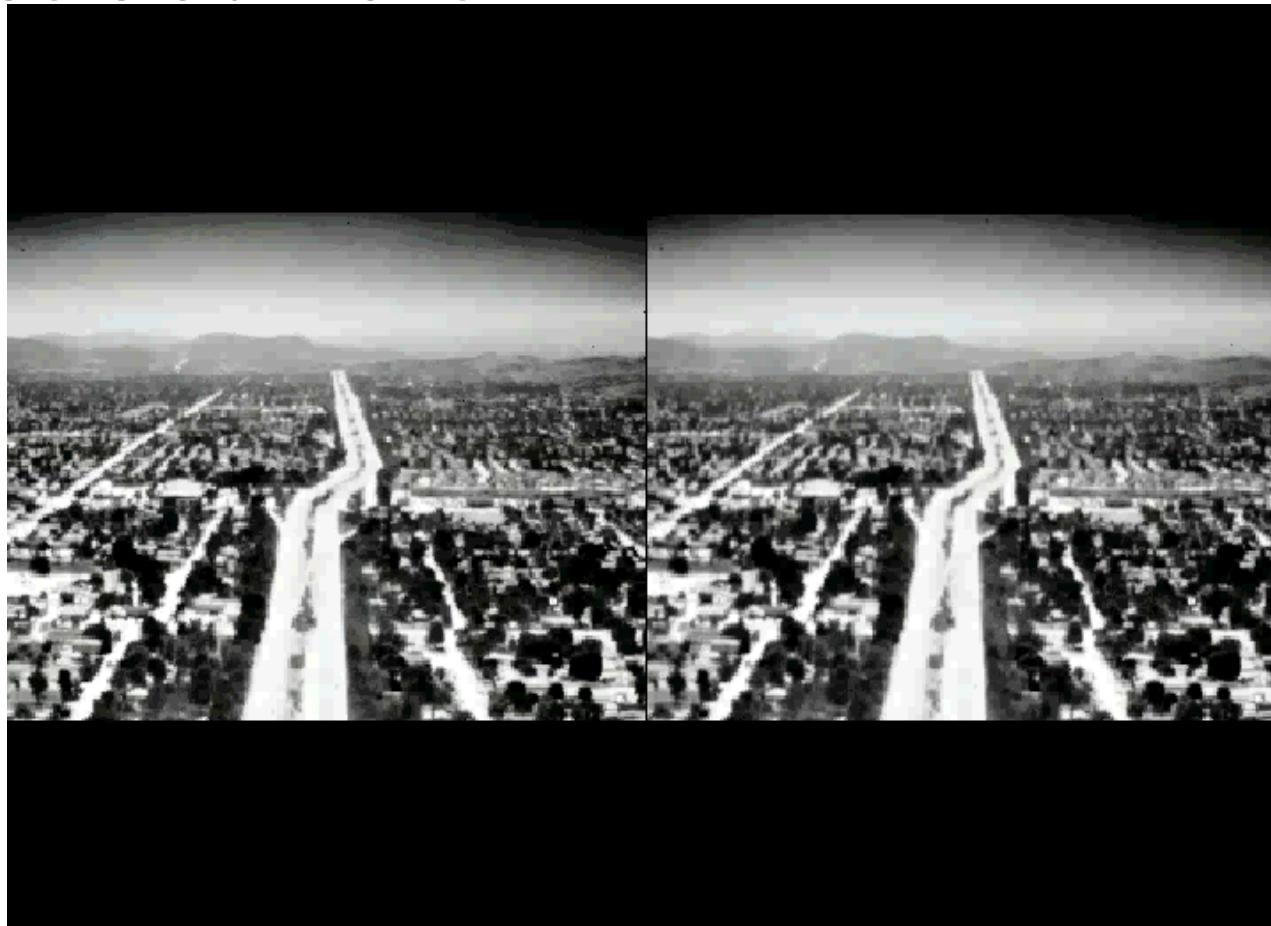
Example (2)

- Video mosaic



Example (3)

- Video stabilization

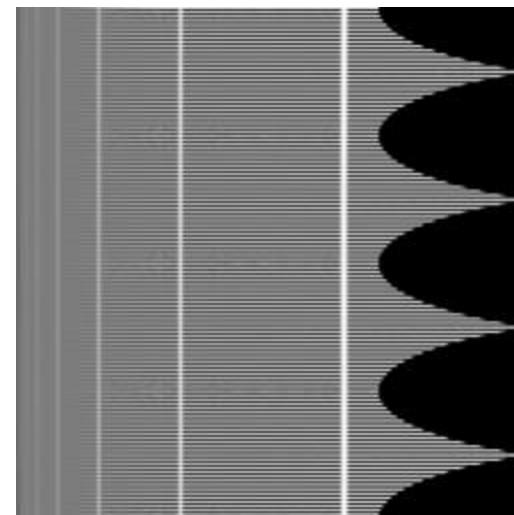
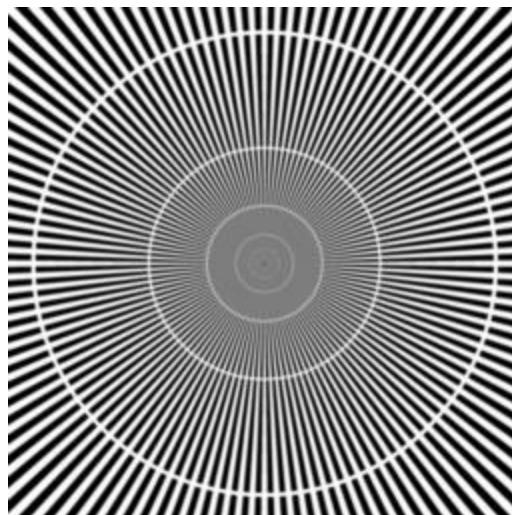


Discussion

- Method: Levenberg-Marquadt Alg. (nonlinear least squares)
- Benefit: estimation of real-valued parameters
- Drawback: the registered images must be fairly close in scale (<1.5), rotation ($<45^\circ$), and translation
- Solution: log-polar registration

Polar Coordinates

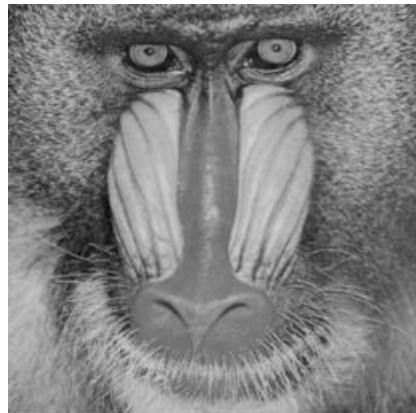
Radial lines in (x,y) Cartesian space map to horizontal lines in (r,θ) polar coordinate space.



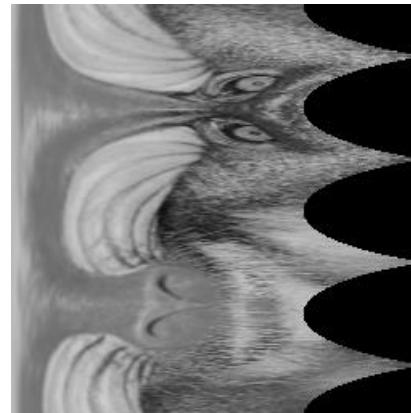
$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

$$\theta = \tan^{-1} \frac{y - y_c}{x - x_c}$$

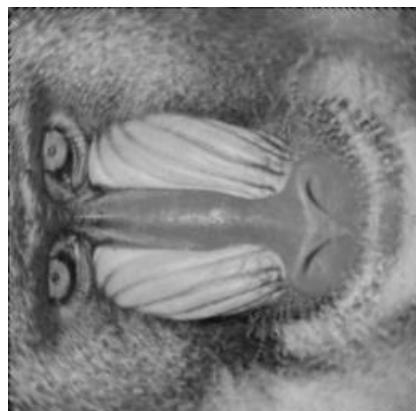
Polar Coordinates: Example



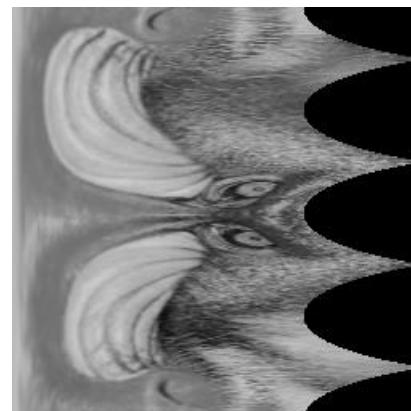
Input image



Polar transformation



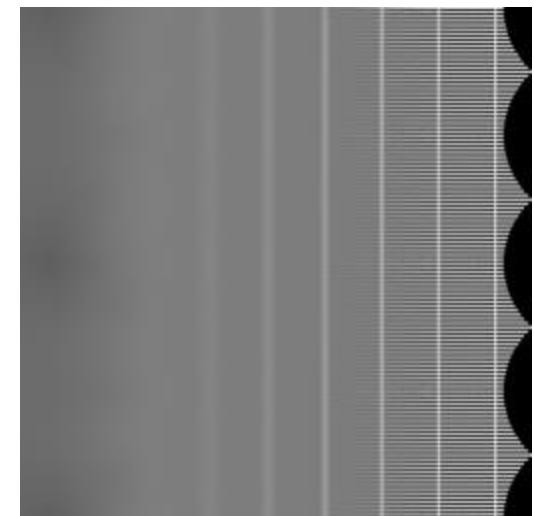
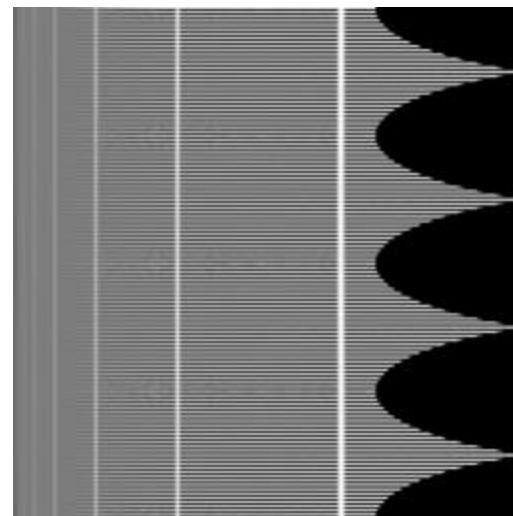
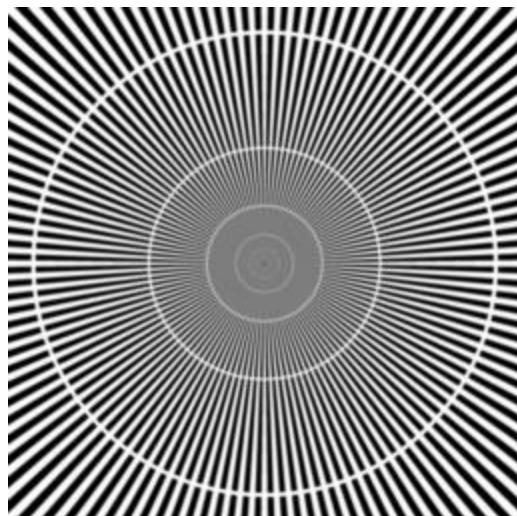
Output in Cartesian space



Circular shift in polar space

Log-Polar Coordinates

Radial lines in (x,y) map to horizontal lines in $(\log r, \theta)$



Example

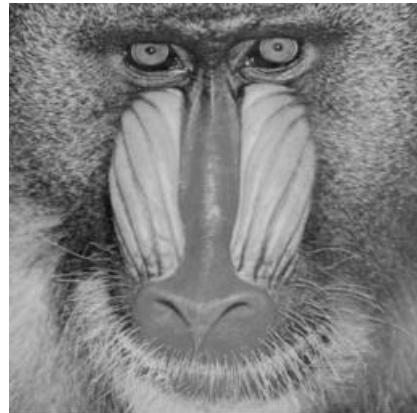
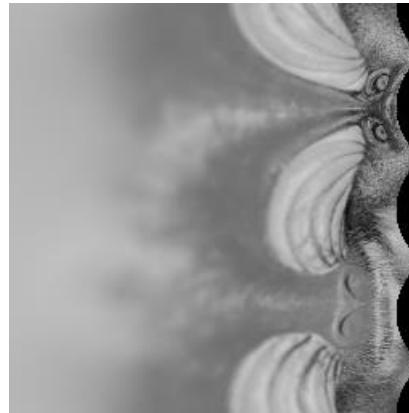


Image I_1



Log-polar(I_1)

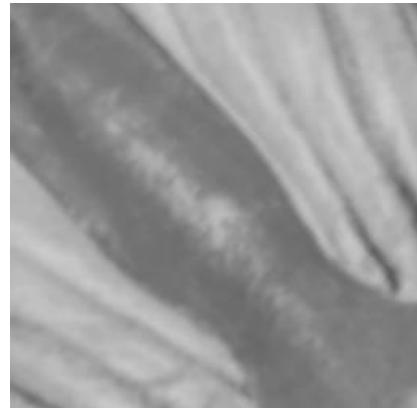


Image I_2 : 3x; 45°

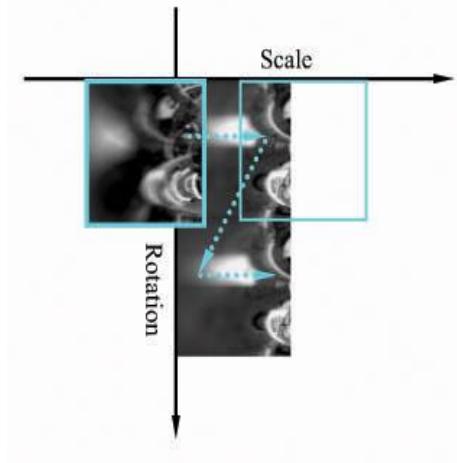
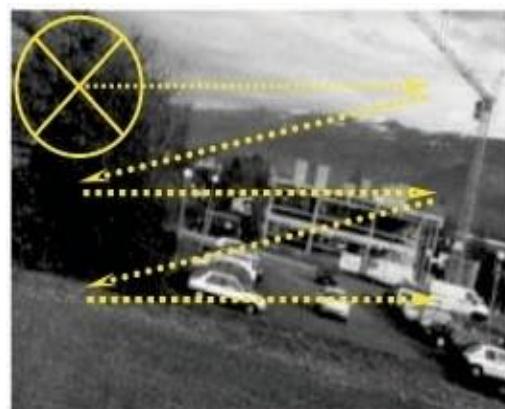
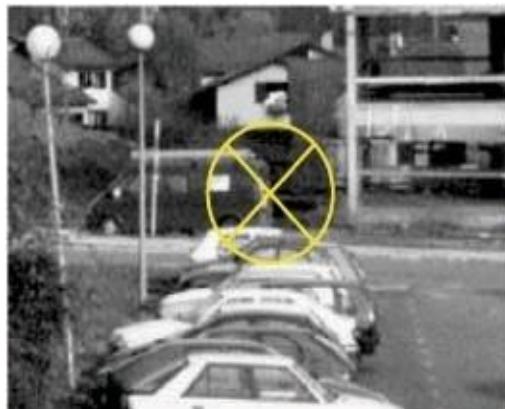


Log-polar(I_2)

Log-Polar Registration

- Benefit: Correlation recovers rotation/scale (fast)
- Drawback: Image origins must be known
- Solution:
 1. Crop central region I_1' from I_1
 2. Compute I_{1p}' , the log - polar transformation of I_1'
 3. For all positions (x, y) in I_2 :
 - Crop region I_2'
 - Compute I_{2p}'
 - Cross - correlate I_{1p}' and $I_{2p}' \rightarrow (dx, dy)$
 - if maximum correlation, save (x, y) and (dx, dy)
 4. Scale $\leftarrow dx$
 5. Rotation $\leftarrow dy$
 6. Translation $\leftarrow (x, y)$

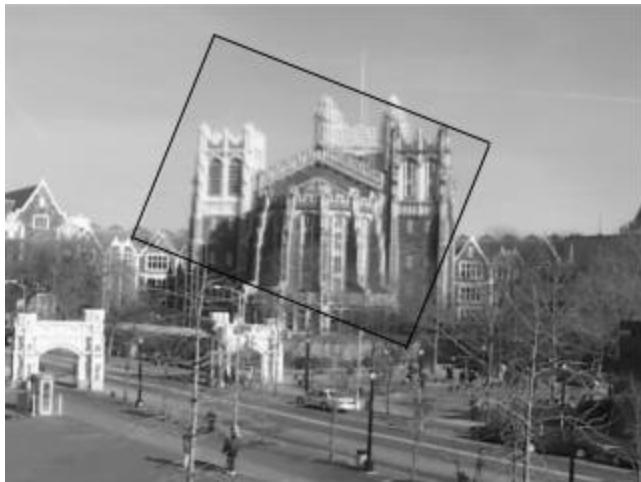
Scanning Window



Example



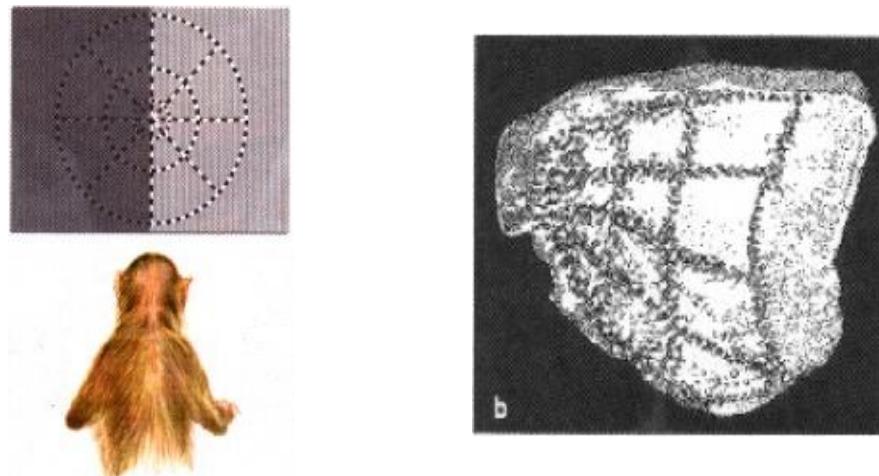
Example



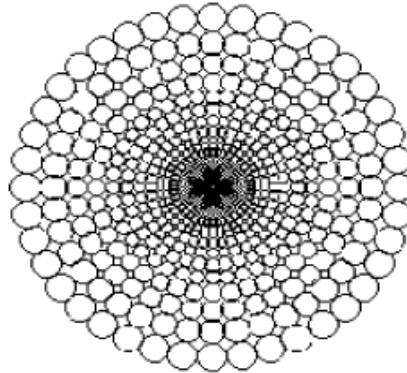
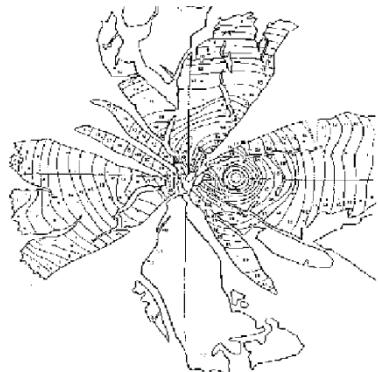
Rotation = -21°
Scale = 0.469
Translation = (-11, -51)

Biological Motivations (1)

- Despite the difficulties of nonlinear processing, the log-polar transform has received considerable attention. [Marshal41] et. al. and [Hubel74] discovered a log-polar mapping in the primate visual system and this conformal map is an accepted model of the representation of the retina in the primary visual cortex in primates [Schwartz79] [Weinshall87].



Biological Motivations (2)



The left figure shows contours of cone cell density. In fact, the density of the ganglion cells which transmit information out of the retina is distributed logarithmically. From Osterberg, G. (1935)

Benefits:

1. Reduce the amount of information traversing the optical nerve while maintaining high resolution in the fovea and capturing a wide field of view in the periphery.
2. Invariant to scale and rotation.

Biological Motivations (3)

Fovea based vision



Related Work

- The Fourier-Mellin transform uses log-polar mapping to align images related by scale, rotation, and translation
{Casasent76, Decastro87, Chen94, Reddy96, Lucchese96, Chang97, Lucchese97a, Lucchese97b, Stone99, Stone03, Keller03}.
- Detect lines in log-Hough space
{Weiman79, Giesler98, Weiman90, Young00}.
- Recognizing 2D objects that were arbitrarily rotated or scaled
{Sandini92, Ferrari95}.
- Tracking a moving object {Capurro97}.
- Estimation of time-to-impact from optical flow {Sandini91}
- Finding disparity map in stereo images {Sandini01}.
- Using log-polar transform to calculate time-to-crash for mobile vehicles {Pardo02}.
- A foveated binocular stereo system using log polar transforms
{Bernardino02}.

Previous work: Fourier-Mellin

$$I_1(x, y) = I_2(s(x \cos(\theta_0) + y \sin(\theta_0) - x_0), s(-x \sin(\theta_0) + y \cos(\theta_0)) - y_0)$$

$$F_1(\omega_x, \omega_y) = \frac{1}{s} F_2\left(\frac{\omega_x \cos(\theta_0) + \omega_y \sin(\theta_0)}{s}, \frac{-\omega_x \sin(\theta_0) + \omega_y \cos(\theta_0)}{s}\right) e^{J(x_0 \omega_x + y_0 \omega_y)}$$

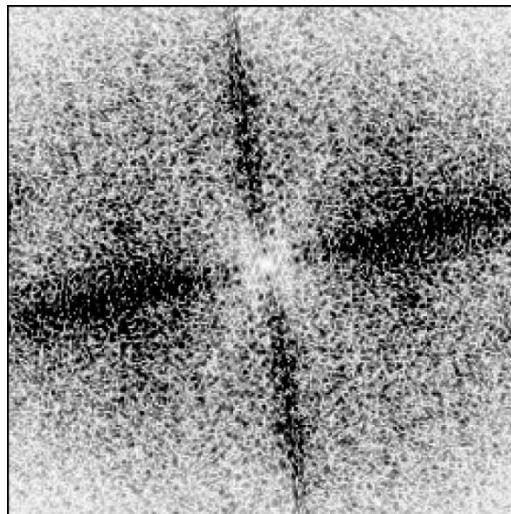
- The magnitude of spectra $|F_1|$ is a rotated and scaled replica of $|F_2|$. We can recover this rotation and scale by representing the spectra $|F_1|$ and $|F_2|$ in log-polar coordinates:

$$|F_1(\log r, \theta)| = \frac{1}{s} |F_2(\log r + \log s, \theta - \theta_0)|$$

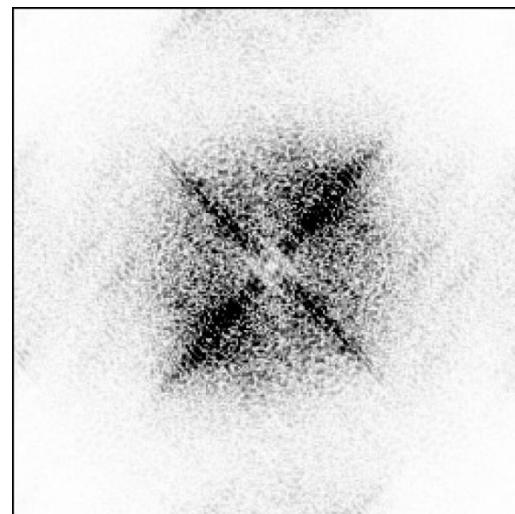
- Phase-correlation can be used to recover the amount shift

Previous work: Fourier-Mellin

- Example: $s = 1.5, \theta = 36^\circ$



se Notes



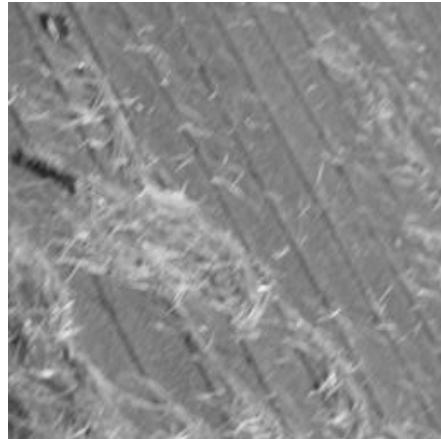
Previous work: Fourier-Mellin

- Benefit: We can search for the scale and rotation independent of the translations.
- Drawbacks: Large translation, scale, or mild perspective will alter the coefficients of the finite and discrete Fourier transform.

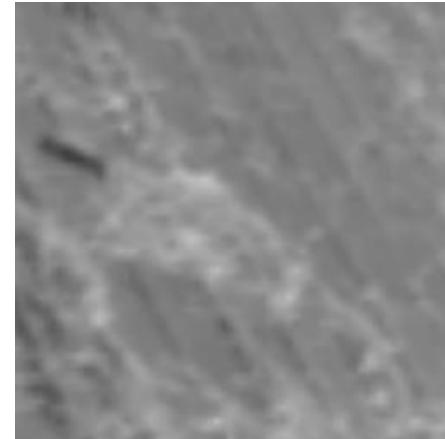
Previous work: Fourier-Mellin



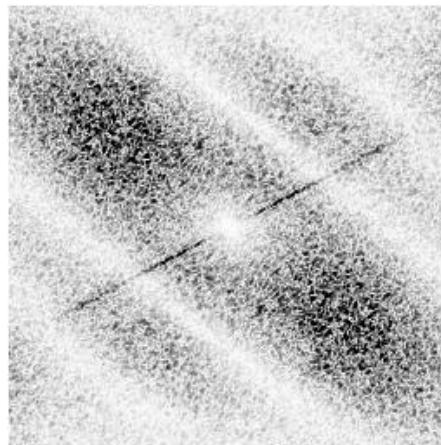
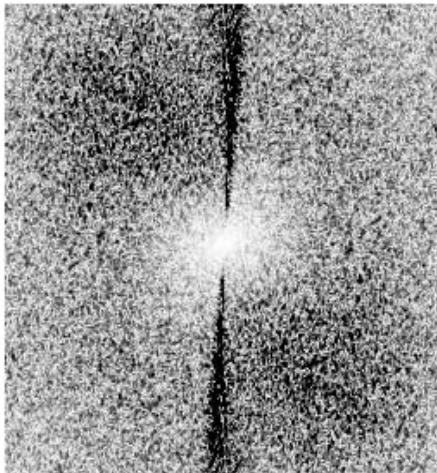
(a) Reference image



(b) Target image (real)



(c) Target image (synthetic)



Power spectra of (a), (b), and (c)

Example from INRIA



Example (1)



Example (2)



Example (3)

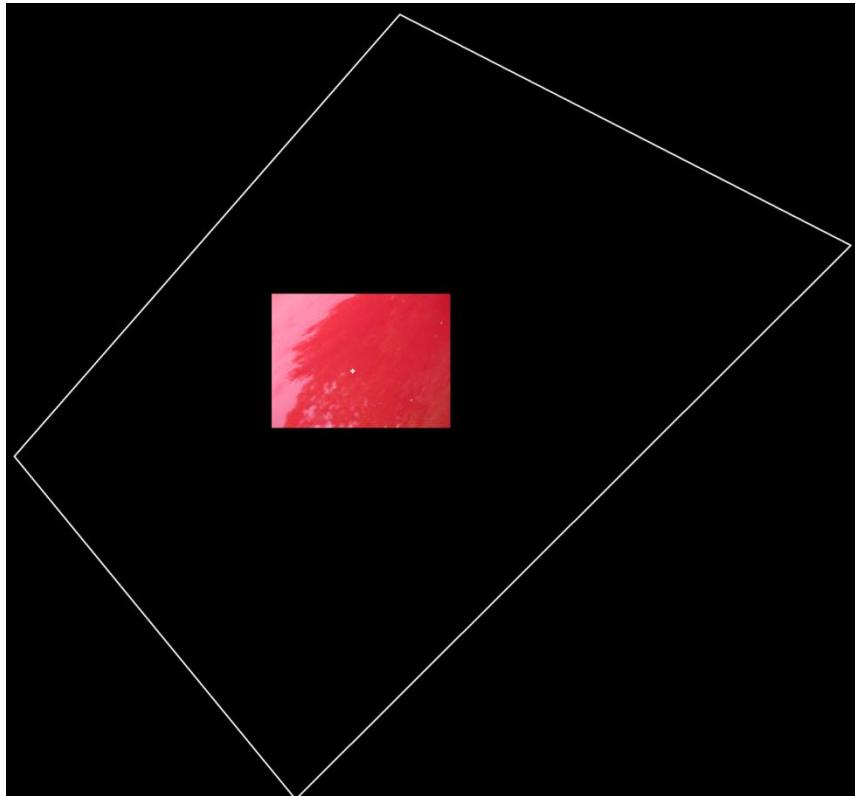
Reference image



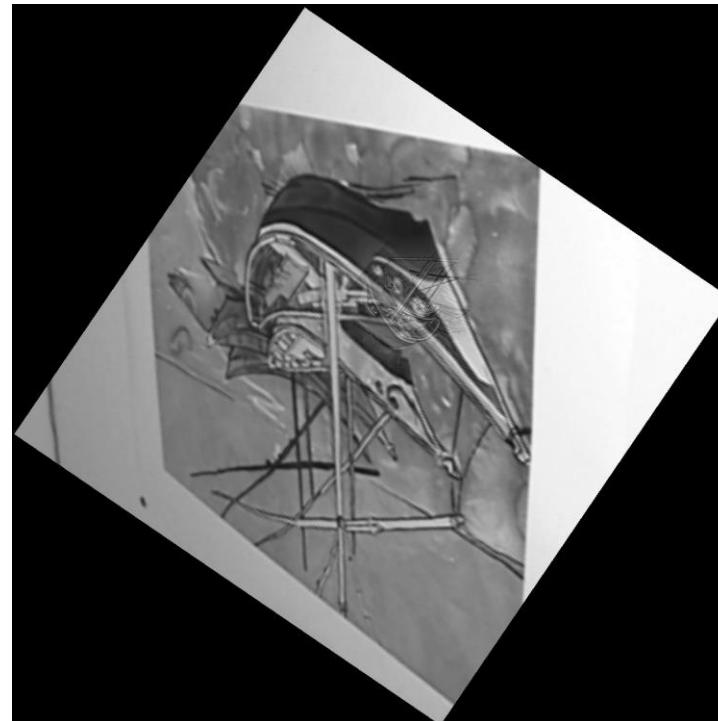
Target image



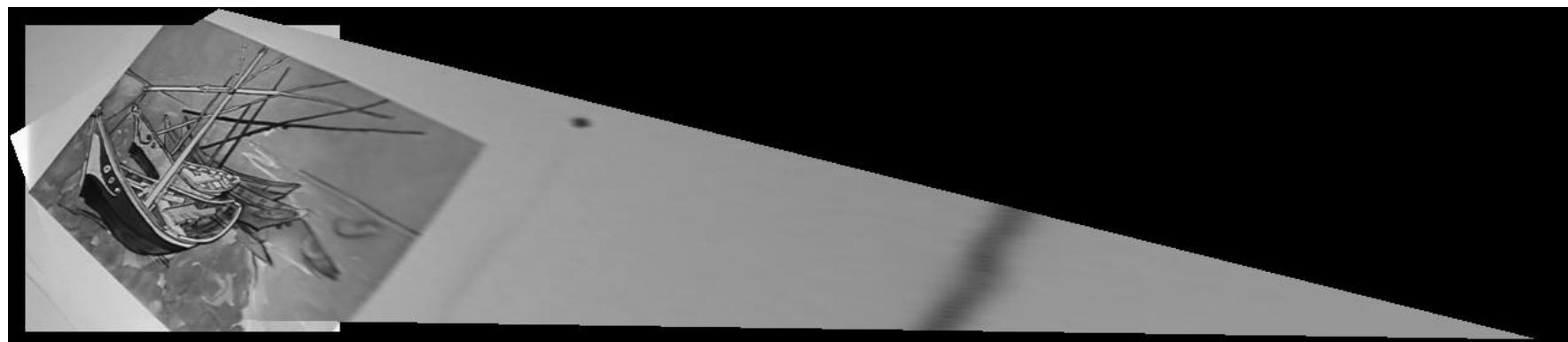
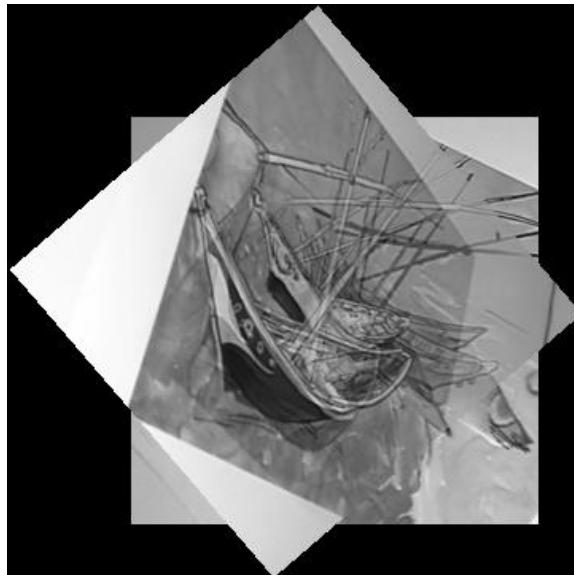
Recovered parameters:
 $\theta = 228.17^\circ$, $S = 3.528217$
 $T_x = -32$ $T_y = 35$



Example: Large Perspective (1)



Example: Large Perspective (2)



Summary

- Log-polar registration:
 - recovers large-scale rotation/scale/translation
 - applies fast correlation in log-polar space
 - exploits fast multiresolution processing
 - initial estimate for affine/perspective registration
- Perspective registration:
 - handles small distortions with subpixel accuracy
 - applies Levenberg-Marquardt algorithm
 - nonlinear least squares optimization
 - exploits fast multiresolution processing