Image Processing Primer

Image Processing

- From image generation to image processing
- We will look at techniques from *image processing* for transforming (grayscale) images
- We'll see how image processing techniques can automatically extract important characteristics of images, e.g., by detecting edges and removing noise (errors/defects in the images)
- Image *intensity* will be the primary image attribute we will examine (for grayscale images)
- Techniques can be generalized to work for color images too

Image Processing

- Operations performed over images (2D or 3D)
- Purpose:
  - Enhance certain features of the image
  - De-emphasize other features of the image
- Implemented as *filters* or transformations:
  - Some operate on the entire set of pixels at once (global operations)
  - Examples: brightness and contrast enhancement

Image Processing

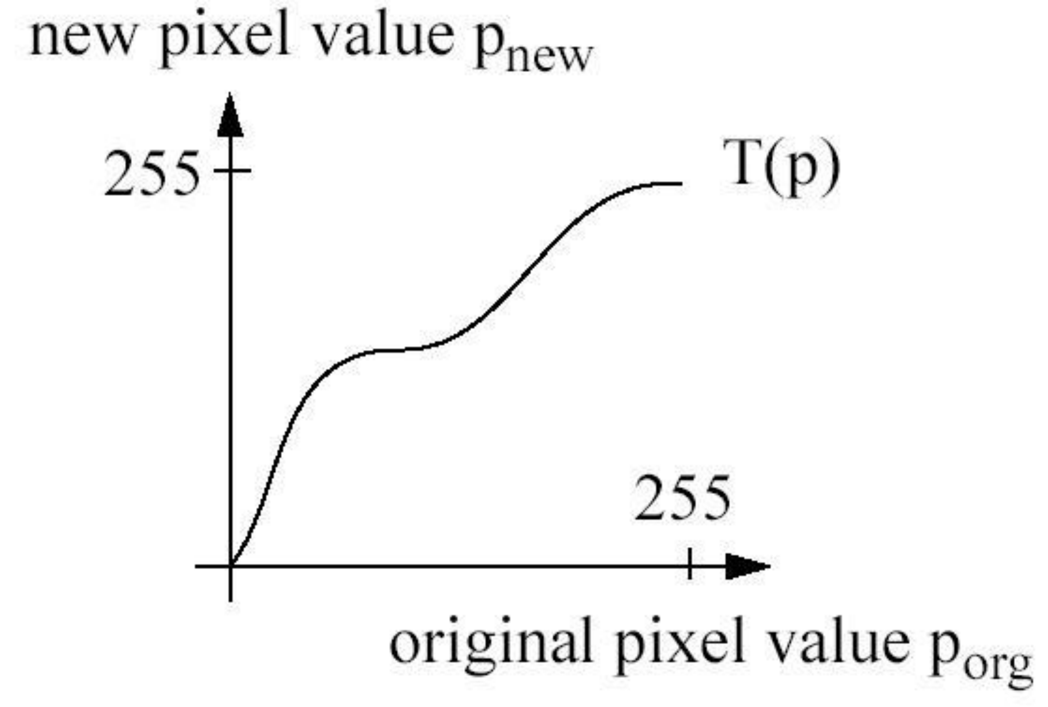
- Some operate only on a subset of pixels (local operations in a pixel neighborhood)
- Examples: edge detection, contouring, image sharpening, blurring, “noise” reduction

Intensity Transformations

- Intensity transformations are one the more basic needs for manipulating or processing images
- Modify distribution of gray levels in an image
- Example: sometimes the goal is to reduce the number of grayscale levels used to represent images
- Reasons: memory, display/printing limitations, cost, etc.
- In practice, we need to reduce the number of bpp (bits per pixel) (e.g., 24 → 8 bits) which is used to represent each pixel / dot
- This process is sometimes called **quantization – we replace one set of possible values with a smaller set that introduces a little error as possible (image compression)**
- Usually intensity transformations used for **image enhancement**

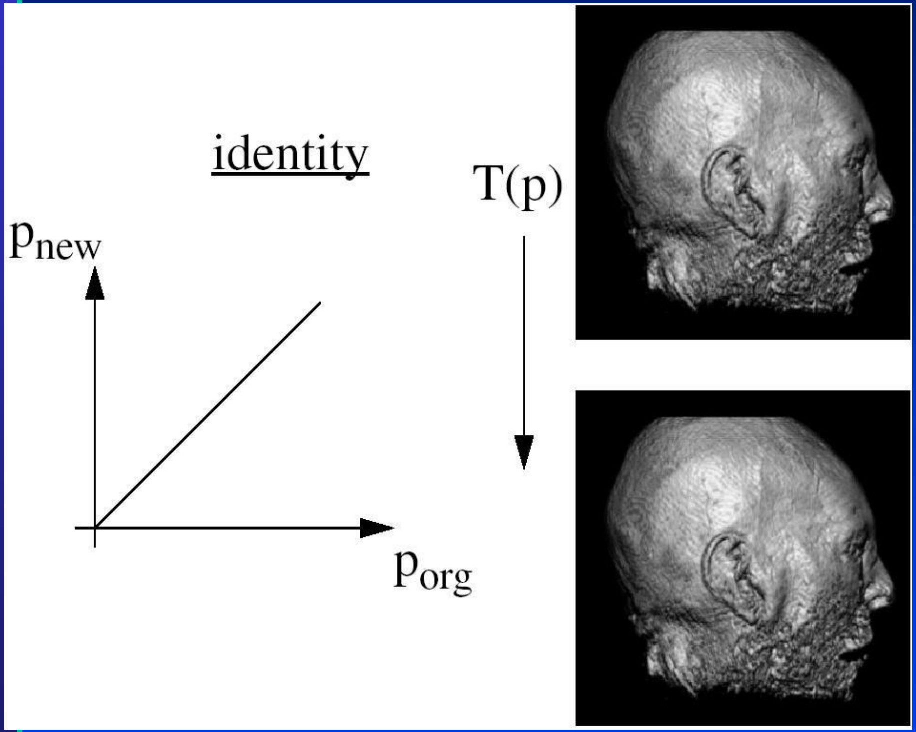
Intensity Transformations

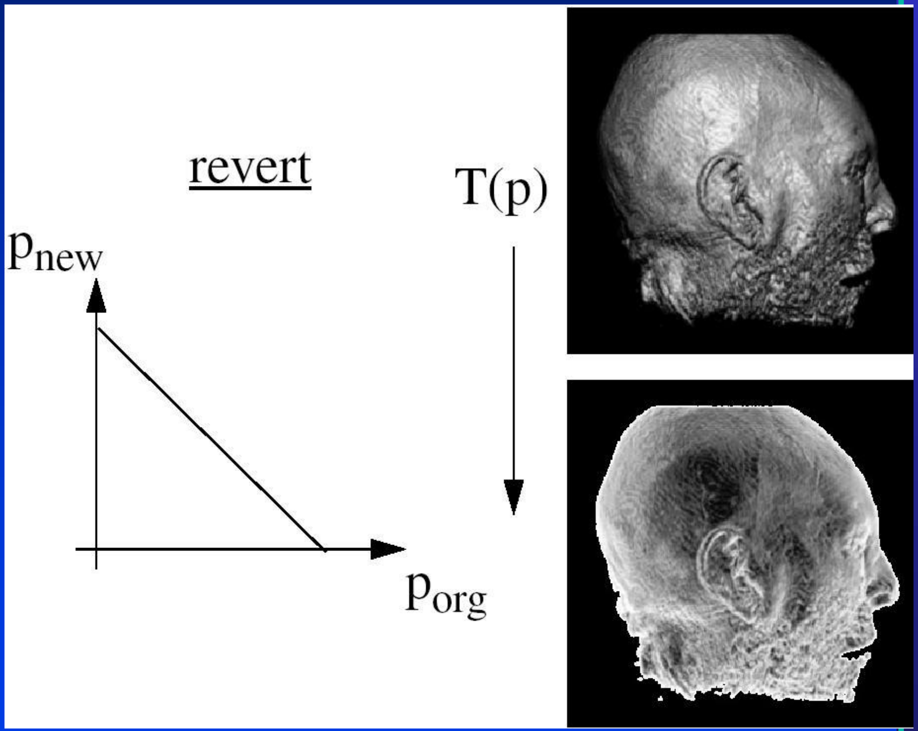
- An intensity transformation most easily expressed as function T(p) over domain of possible pixel intensities
- The new pixel intensity is given as the height of the function if we assume a uniform scaling along both axes
- How might we discretize an intensity transformations and present it in a computer?



Intensity Transformation Examples

- What would happen to the image in each case?





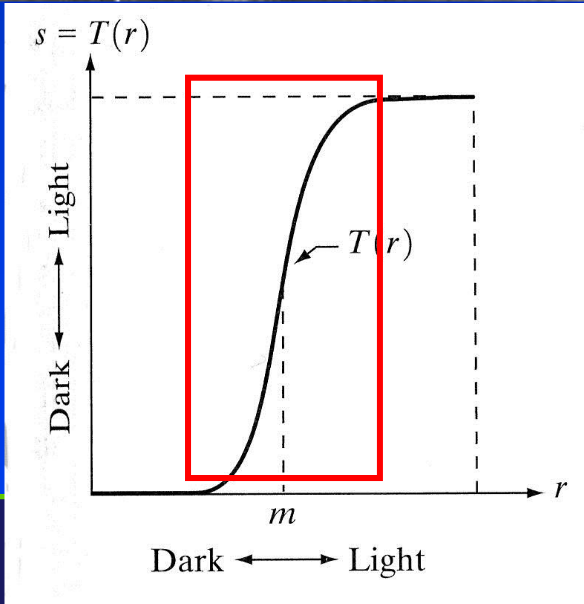
- What does the bottom-right image look like?

Contrast Enhancement

- Oftentimes one is given an image with poor contrast
- The image seems washed out and features are hard to see
- Need to enhance the contrast somehow

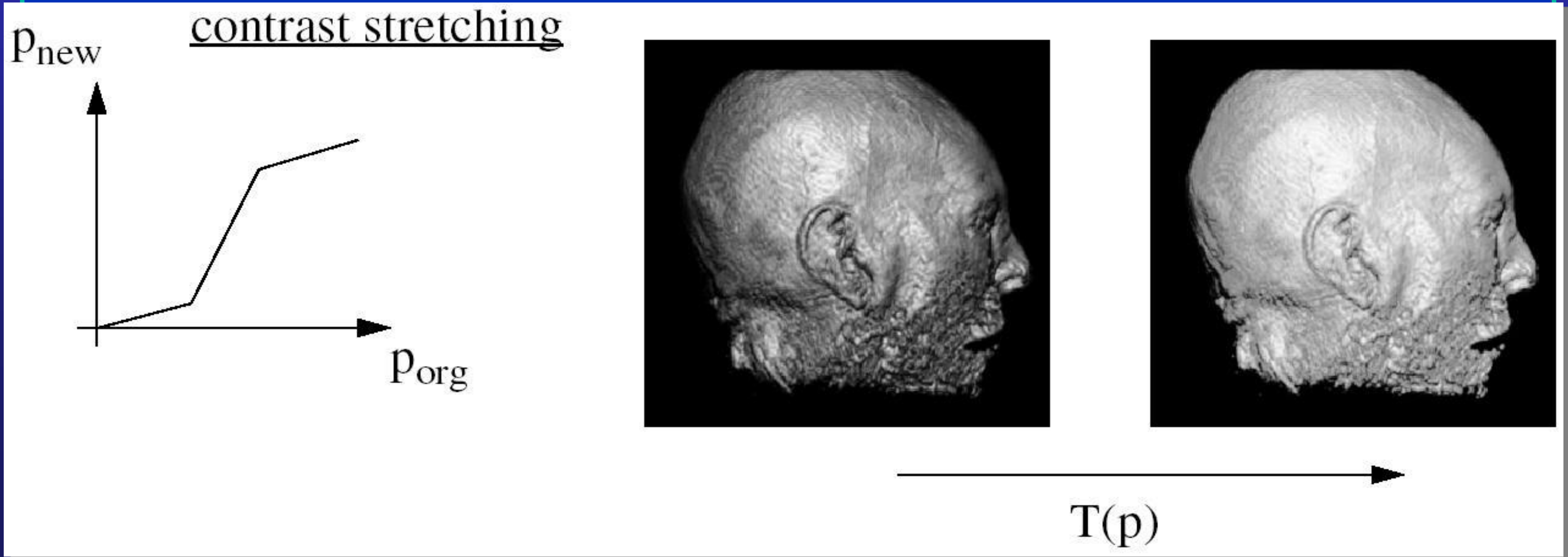
Contrast Enhancement

- One technique for fixing such images is process called *contrast stretching*
- Basic idea: perform an intensity transformation to cause darker shades to become darker, and lighter shades to become lighter



Contrast Enhancement

- Piecewise linear functions are typically used to specify contrast stretching instead of continuous ones
- Give the user more freedom and greater control
- Also easier to implement in software (how?)



Thresholding

- Another way of manipulating contrast is called thresholding
- What's going to happen?
- How many bits are required now?

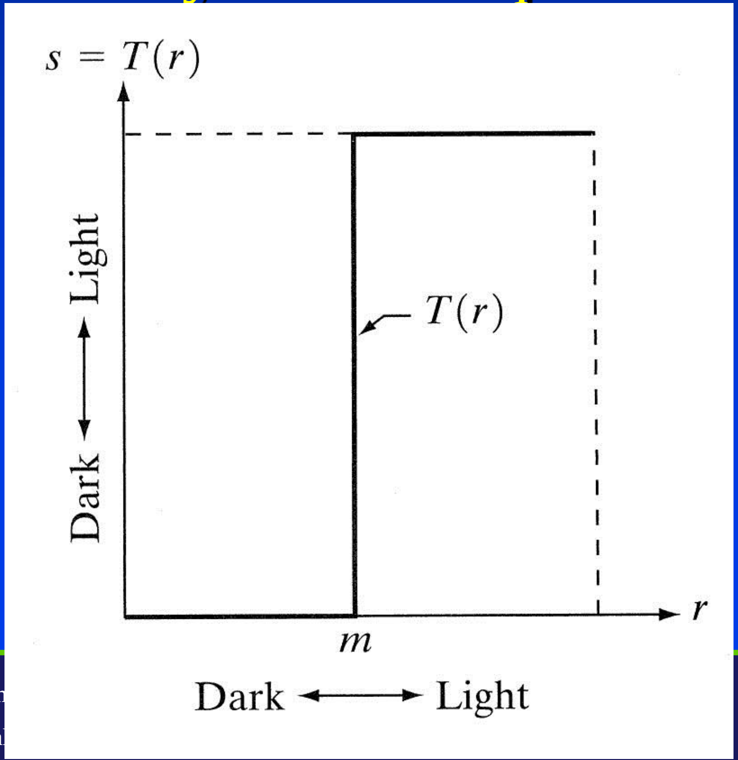


Image Transformations

- Another example of thresholding using a linear ramp

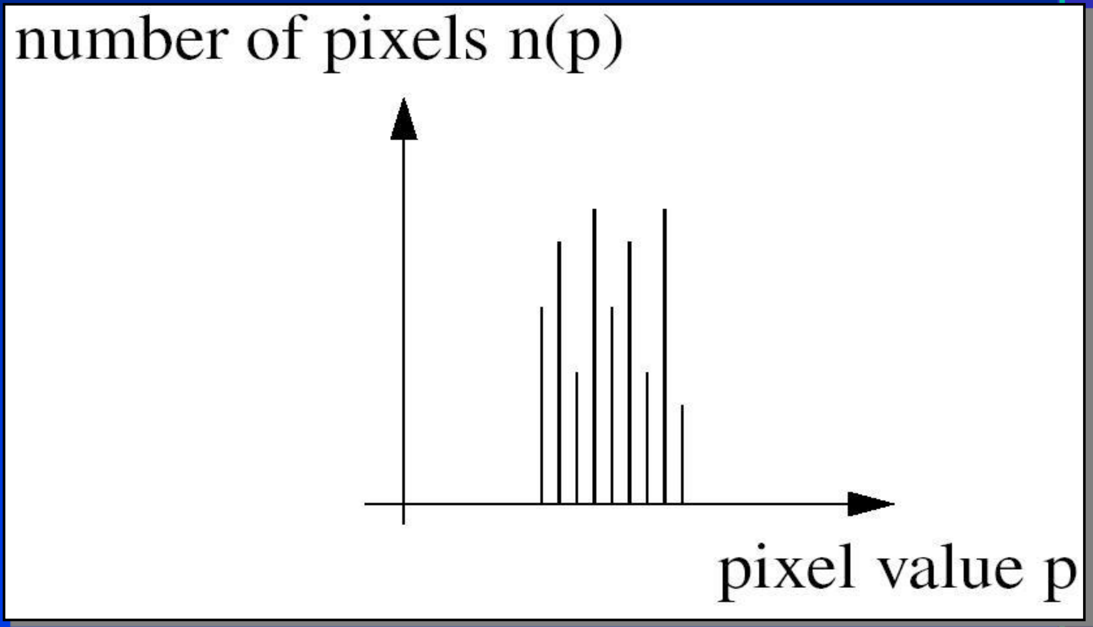
- Why were some of the graylevels preserved?
- Compare with the cat image

Histograms

- An important concept in image processing (and probability & statistics) is the **histogram**
- Suppose we can display 256 discrete gray level intensities, ranging from 0 to 255 (8-bit image)
- To generate a histogram of the image, we would first count the number of pixels having each intensity:
  - p_0: $n_0 = n(p_0)$
  - p_i: $n_i = n(p_i)$
  - etc.

Histograms

- Then we can plot the counts in a graph to view distribution of intensities across image
- Q: Given an array histogram[], AND array of pixels with associated intensities (pixels[i].intensity), how would you build the histogram?

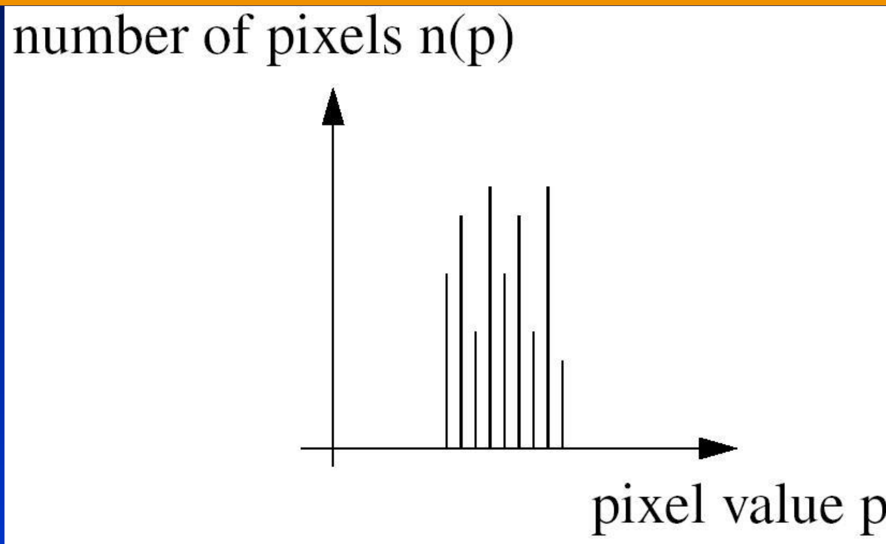


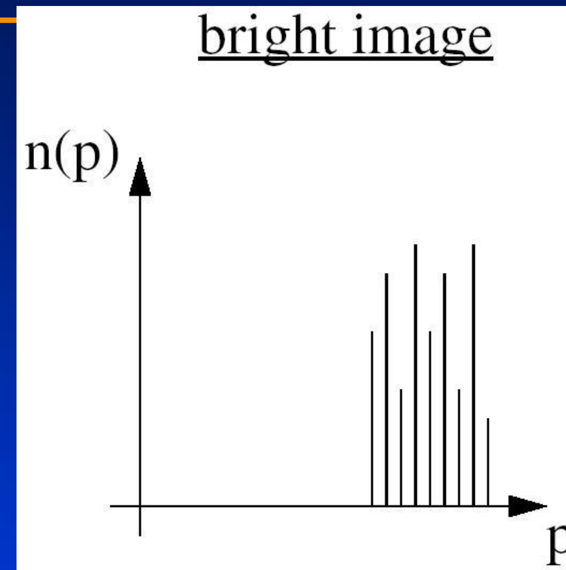
- A: histogram[pixels[i].intensity]++ in a **for** loop over pixels[]

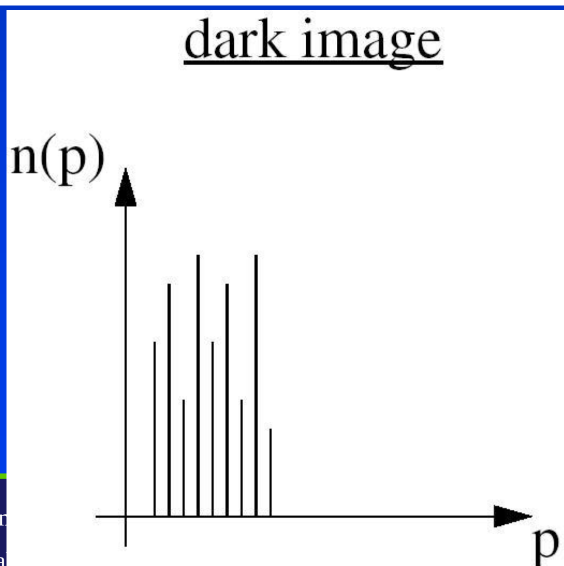
Histograms

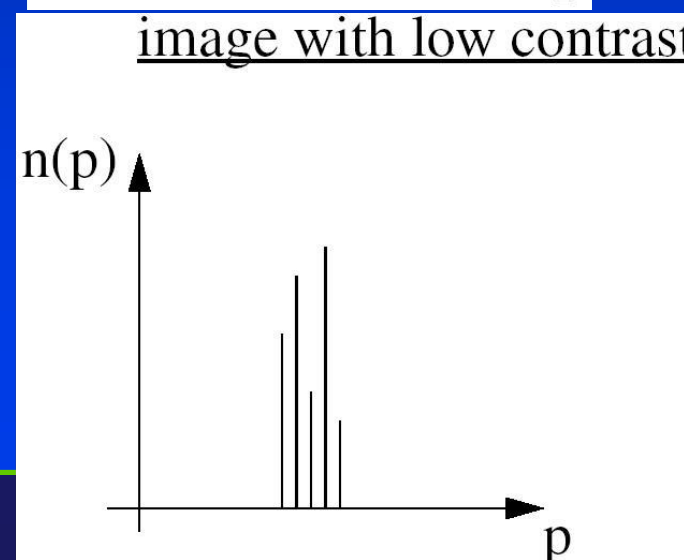
- If we were to divide each count by the total number of pixels, this would produce something akin to a *probability distribution function* (pdf), which one finds in probability

Example Histograms

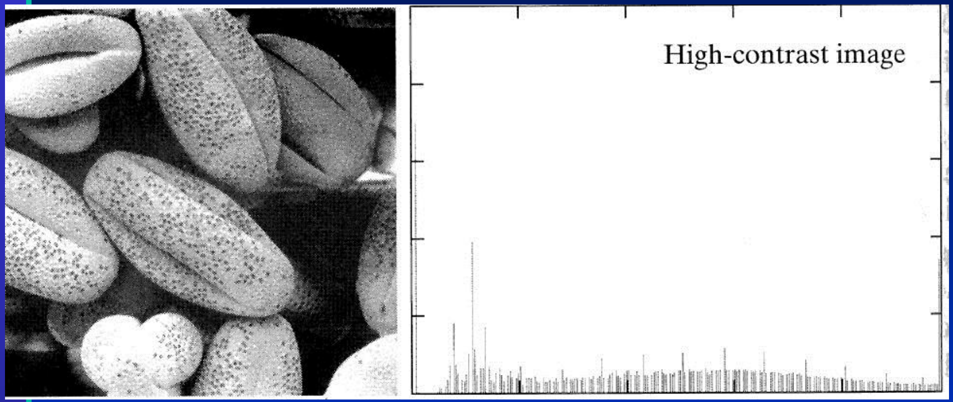


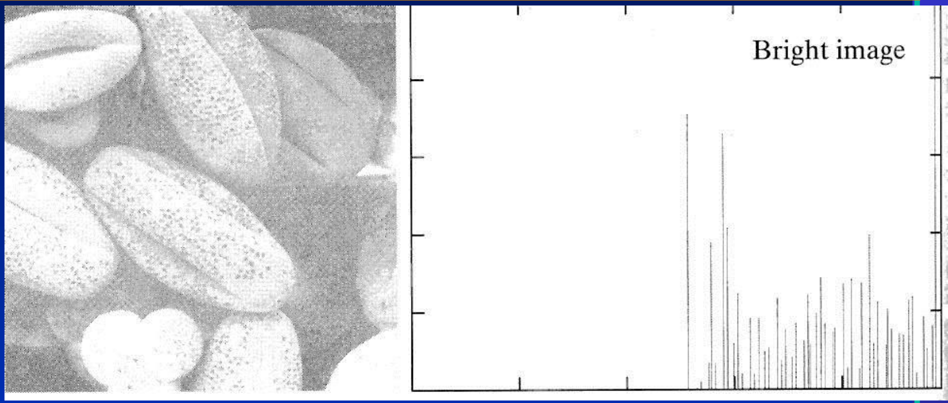


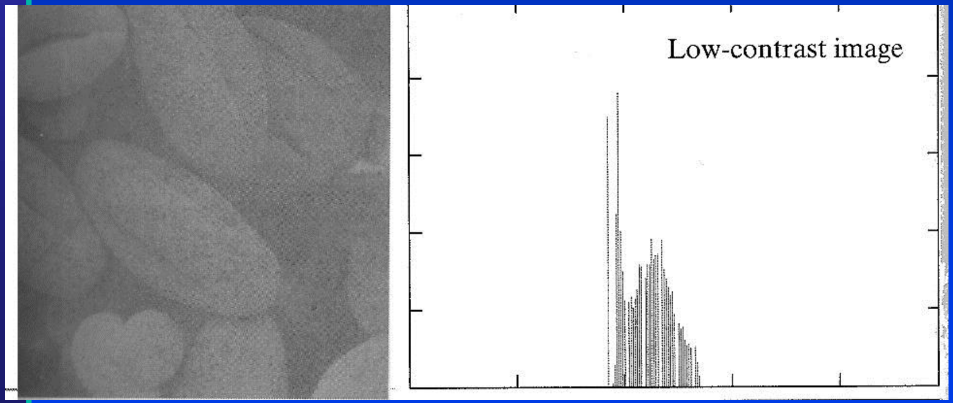


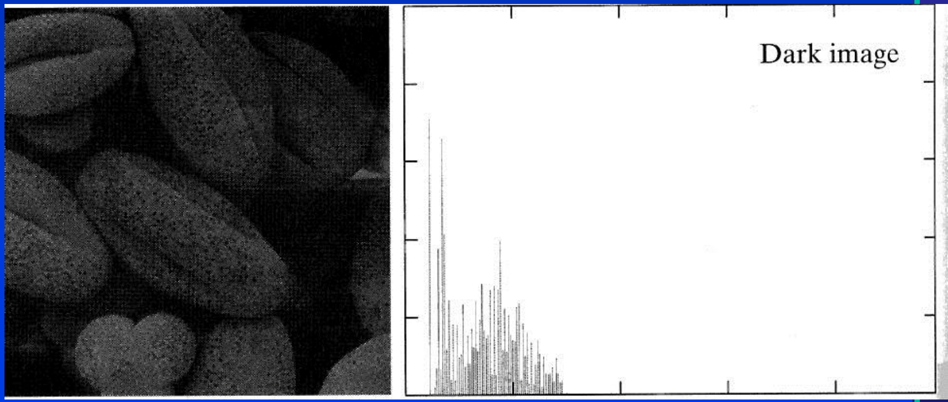


Example Histograms



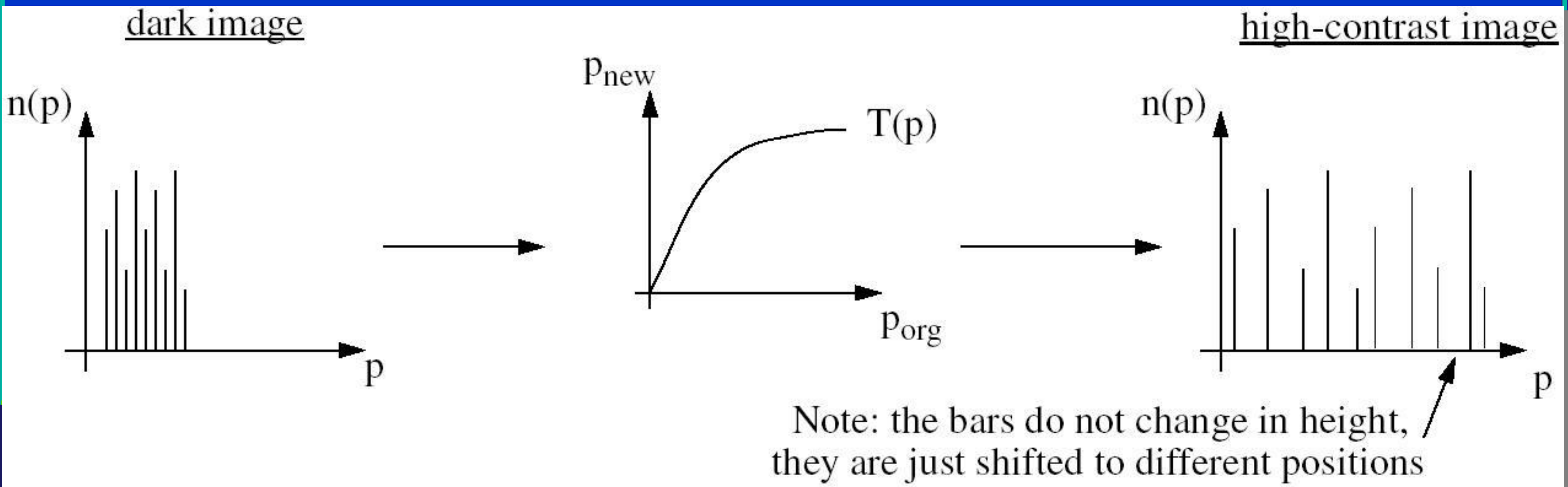






Histogram Equalization

- One automated (i.e., algorithmic) technique for improving contrast is *histogram equalization*
- Basic idea: increase range of intensities displayed in an image by “stretching” the histogram (similar to contrast stretching)
- In such way, range of displayed intensities becomes more uniform



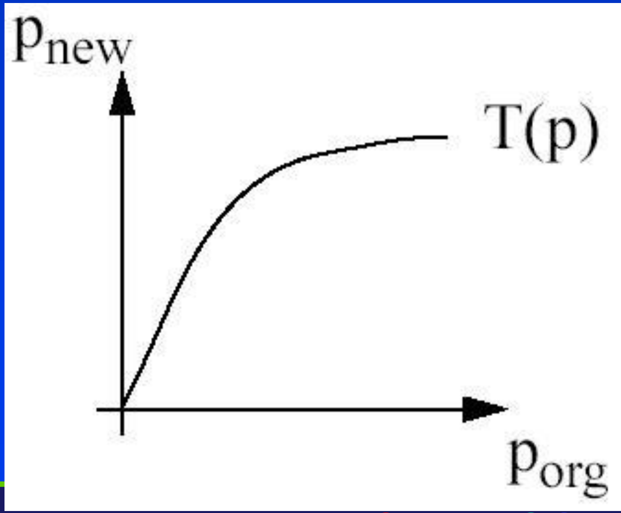
Histogram Equalization

- The discrete histogram equalization equation is

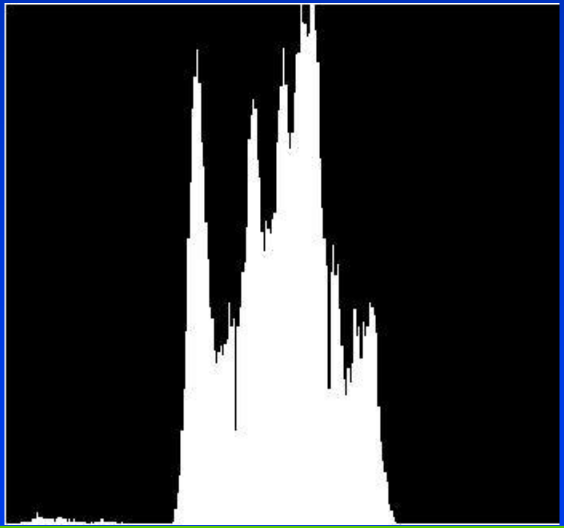
$$p_{new}(k) = \sum_{j=0}^{k} \frac{n(j)}{n_{total}} p_{\max}$$

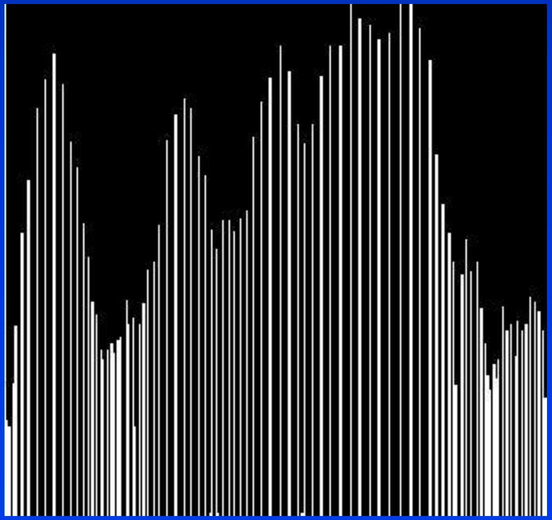
$$p_{new}(k) = \sum_{j=0}^{k} \frac{n(p_{org}(j))}{n_{total}} p_{\max}$$

- p_{max} is maximum possible intensity (not necessarily maximum intensity that happens to appear in the image)
- We accumulate a running total
- This accumulation explains shape of function, which resembles a *cumulative distribution function*



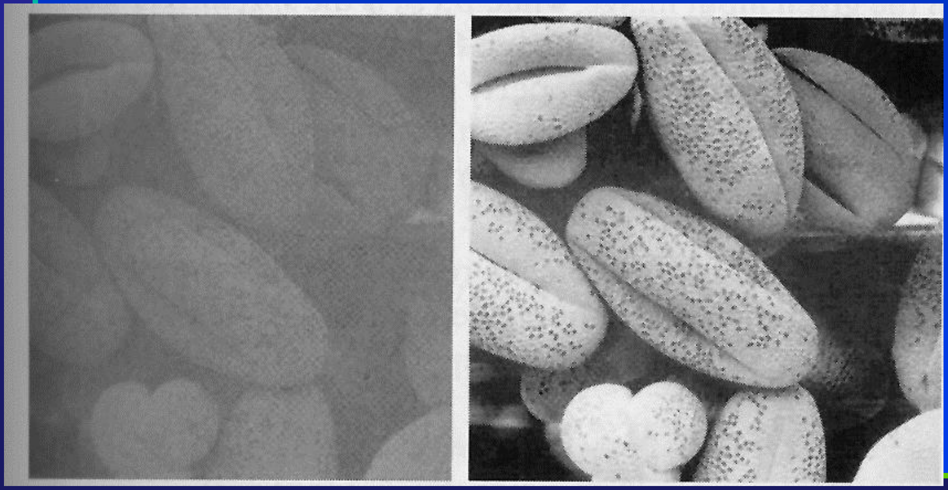
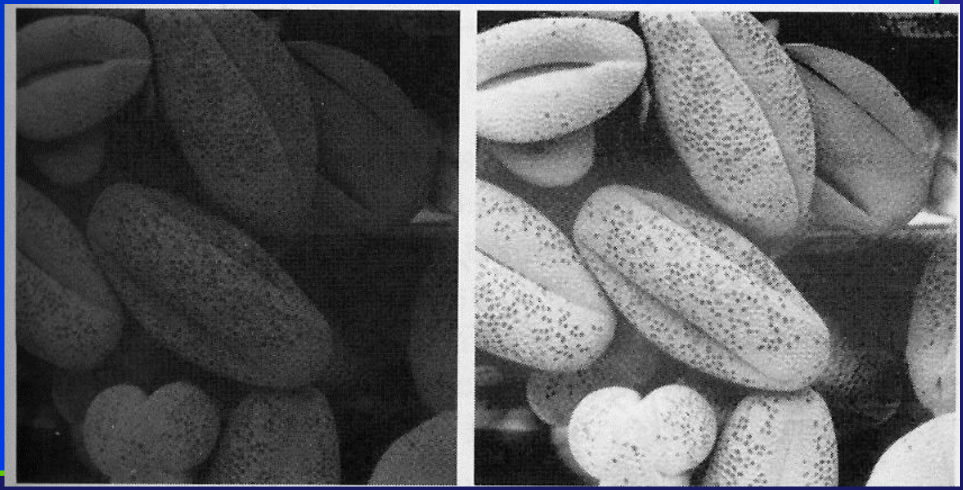
Histogram Equalization Example





Histogram Equalization Examples

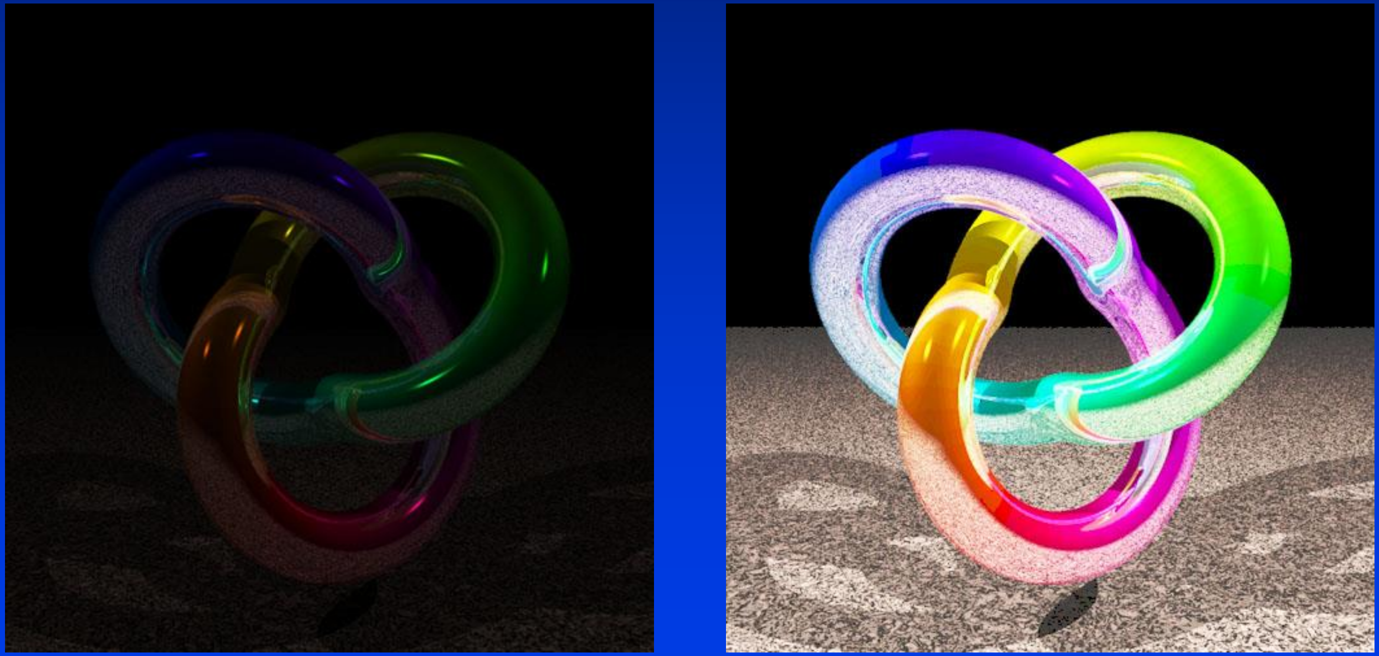


Histogram Equalization Example

Can This Work for Color Images?

- How do we apply histogram equalization to color image?
- Convert RGB → HSV, then equalize histogram of V



- Could we equalize the H (Hue) or S (Saturation) channels?

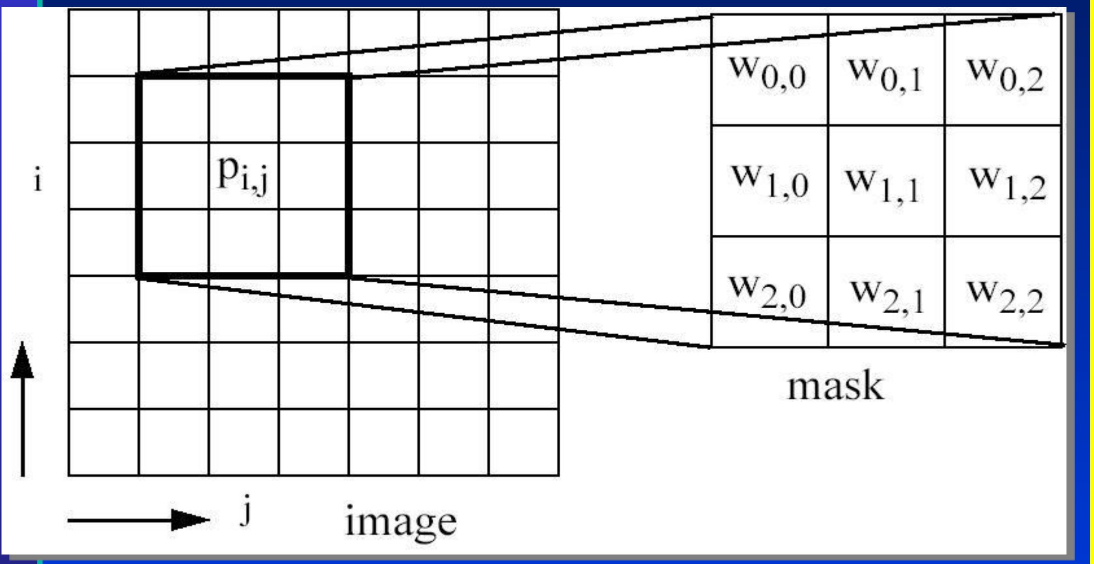
Histograms Summary

- Histograms are a useful tool for studying images
- We can manipulate images to improve contrast
  - contrast stretching and thresholding (manually)
  - histogram equalization (automatically)
- These are all *global* processes
- Suppose we localize computations and use only local information when processing an image?
- This brings us *discrete convolution* or *filtering*

Discrete Convolution (Filtering)

- Examples of image processing based on local information include
  - smoothing (noise removal, image compression), and
  - edge enhancement (di-blurring, sharpening, feature extraction
- We use discrete convolution for these operations
  - place a square matrix of weights called a *mask* over each pixel
  - mask takes a weighted sum of neighboring pixels according to weights in mask
  - the resulting intensity is the new output pixel
  - when done for all pixels, a new image is produced of same resolution as original

Discrete Convolution (Filtering)



for each i, j

$temp = 0$

for each k, l

$$temp += p^{org}_{(i-1+k,\, j-1+l)} \cdot w_{(k,l)}$$

$$p^{new}_{i,j} = temp$$

$$p^{new}_{i,j} = \sum_{k=0}^{2} \sum_{l=0}^{2} p^{org}_{(i-1+k,\, j-1+l)} \cdot w_{(k,l)}$$

- **Very important note**: do not replace computed values into the original image, but write to an output image
- You need a second memory buffer (array) for this

Image Smoothing

- A smoothing mask averages local pixel neighborhood
- Each pixel's value is replaced by its local average in the output image
- Can be used to remove noise, like speckling

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

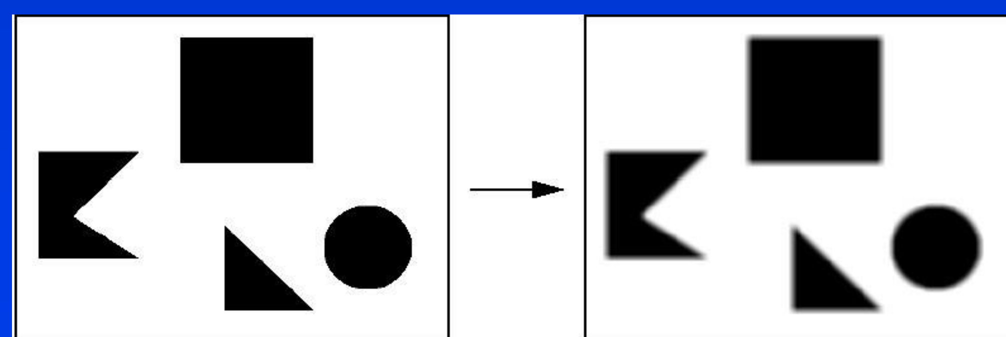


Image Smoothing

- By “high frequency” we mean abrupt changes in the intensities, as can be seen in the images to the right
- “High frequency” is a term related to signal processing theory (Fourier analysis), from which discrete convolution is derived

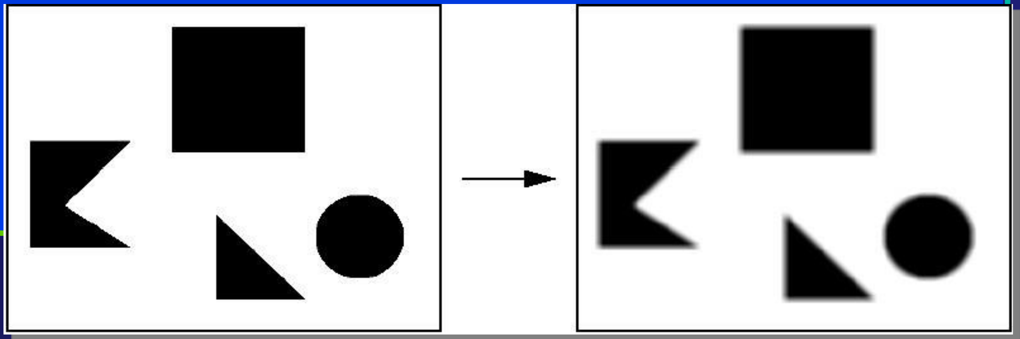


Image Smoothing

- Larger masks smooth more and cut more noise
- Always make sure that sum of all mask elements equals 1.0
- What would happen if the sum weren't 1.0?
- Image brightness would increase or decrease
- Smoothing the image blurs it – larger masks blur more
- Jagged edges are replaced by blur

Image Smoothing

- Smoothing is often used in graphical applications
- Why diagonal lines (and fonts) on a screen look smooth, even though they are comprised of a sequence of pixels
- This kind of blurring is a special application of image smoothing known as *anti-aliasing*
- Eye is tricked into seeing a “continuous” line segment

Image Smoothing Example

- Results of smoothing top-left image with masks of size 3, 5, 9, 15, and 25
- Notice how some of circles completely disappear
- Also notice how smoothing lessens or even eliminates noise in rectangles

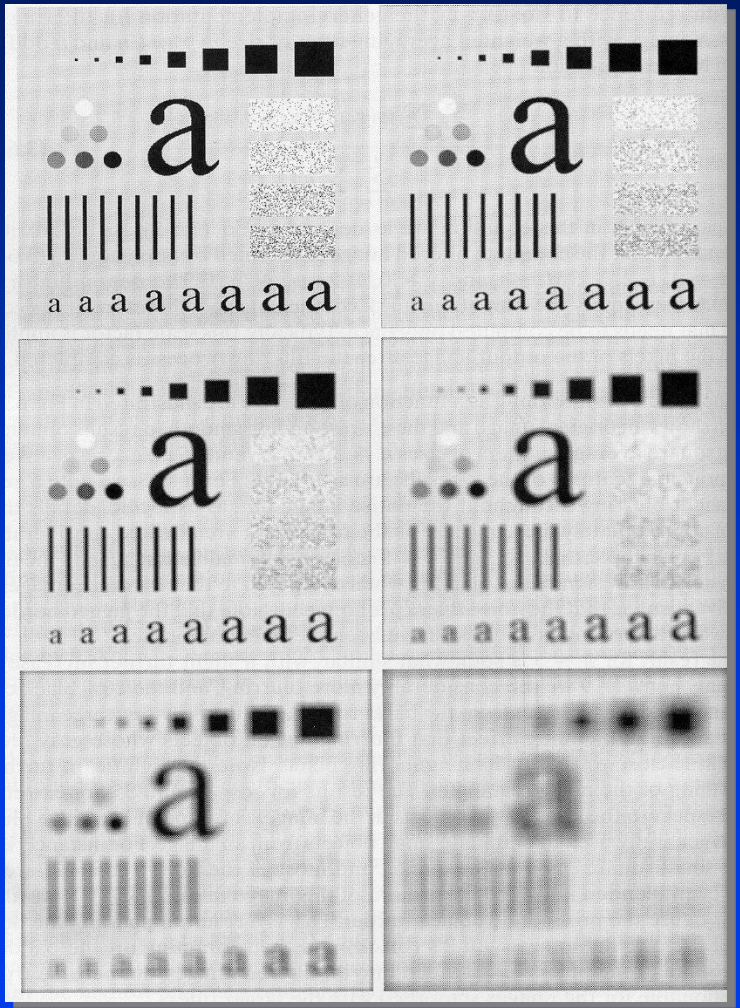
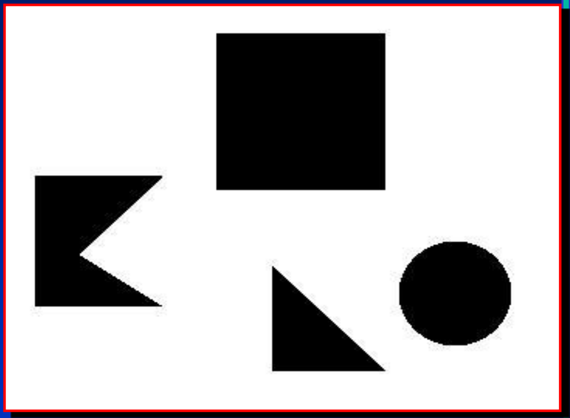


Image Sharpening

- This operation enhances the edges, rather than blurring the image
- **Edge enhancement**
- It has little effect in smoothly varying areas that have no edges
- Why do this?
- Extract boundaries of regions, perhaps



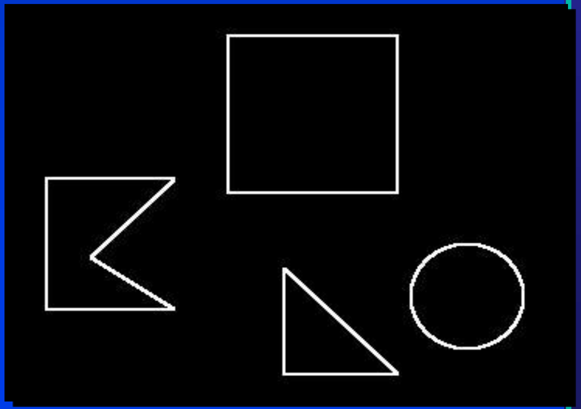
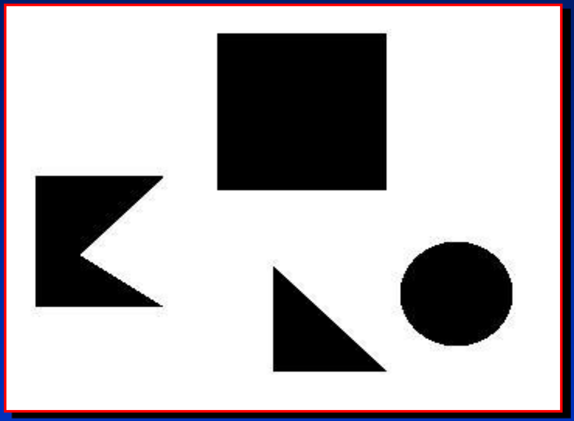
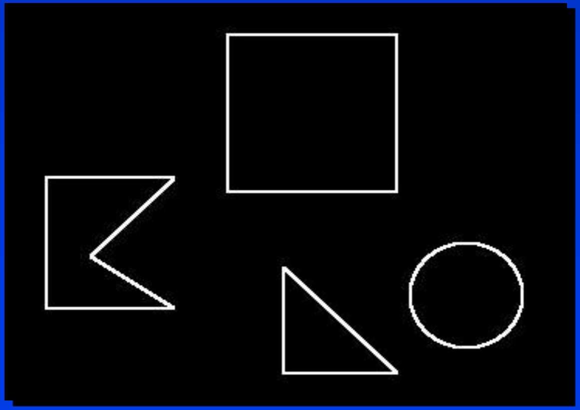


Image Sharpening





- An edge in image indicates that there is a high local first derivative or *gradient* at the given pixels
- Sharpening masks therefore implement some sort of differentiation
- Usually we are only interested in **gradient magnitude**

Image Sharpening

- Image gradient computation
- Usually, we are only interested in the **gradient magnitude**

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

$$|\nabla f| = mag(\nabla f) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}}$$

Image Sharpening Mask Example: The Sobel Mask

- The Sobel filter comes in a pair of masks
- Each mask computes an image for x-derivative (dx), other for y-derivative (dy)
- Note that the dy-masks do some smoothing in x-direction (dx-mask smoothes in y)
- This decreases sensitivity to noise in one direction

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

dy

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

dx

Sobel

Image Sharpening Mask Example: The Sobel Mask

- But increases the sensitivity in the other direction, which is exactly what we want
- Pixel values below zero will occur at edges with negative gradients

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

dy

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

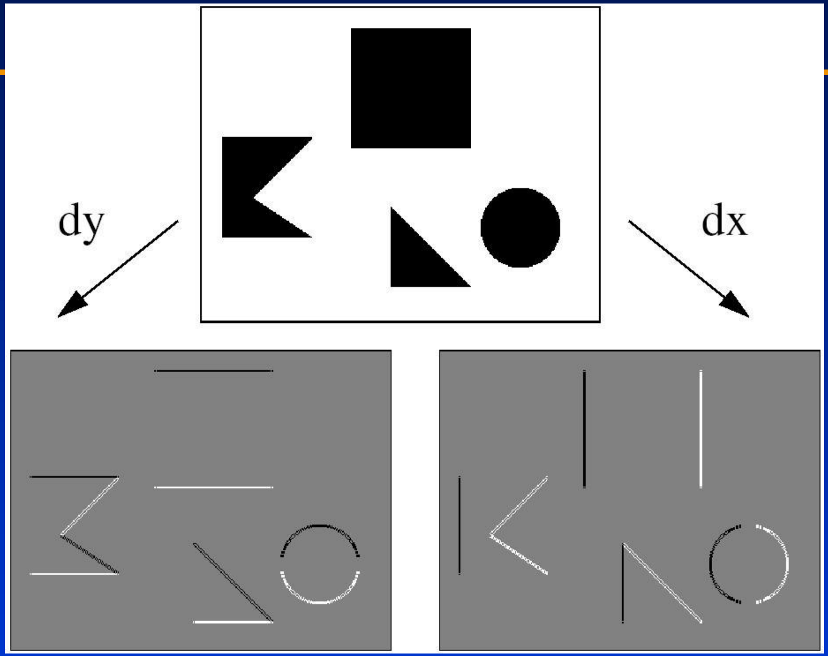
dx

Sobel

- But this is OK because we are actually only interested in the magnitude, not the sign…why only the magnitude?
- High magnitude (positive or negative) indicates an edge!

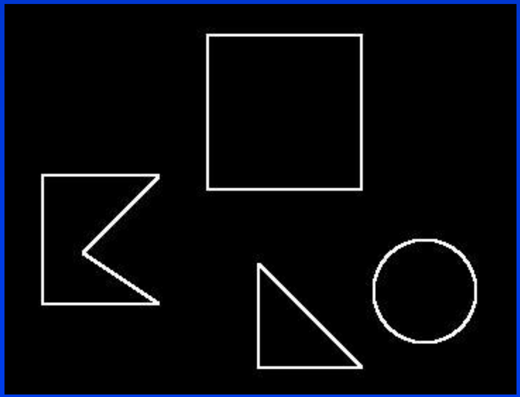
Sobel Mask

- We use the Sobel mask by
  applying the two masks separately, thereby generating two images, img_{dx} and img_{dy}



- Their pixels are combined
  by

$$img_{new} = \left(img_{dx}^{2} + img_{dx}^{2}\right)^{\frac{1}{2}}$$



Sobel Mask

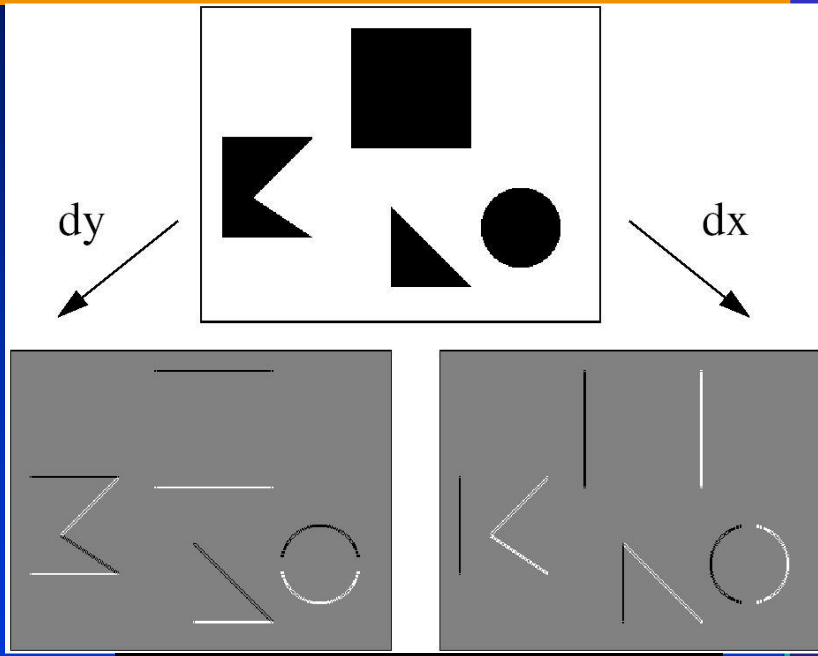
$$img_{new} = \left(img_{dx}{}^2 + img_{dy}{}^2\right)^{\frac{1}{2}}$$

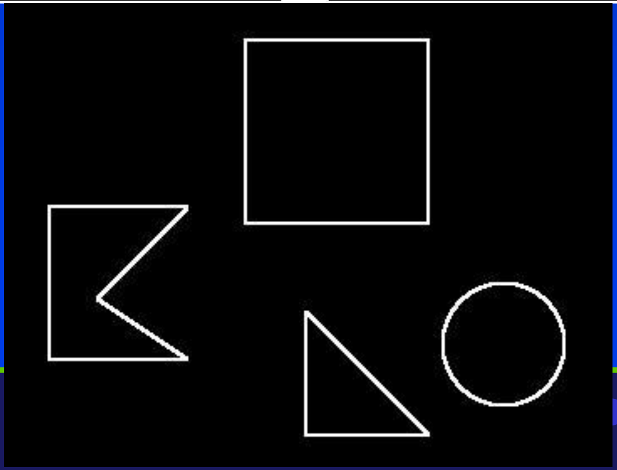
$$img_{new} = |img_{dx}| + |img_{dy}|$$

- Since this formulus is very computationally expensive, typically the following approximation is used instead:

$$img_{new} = |img_{dx}| + |img_{dy}|$$

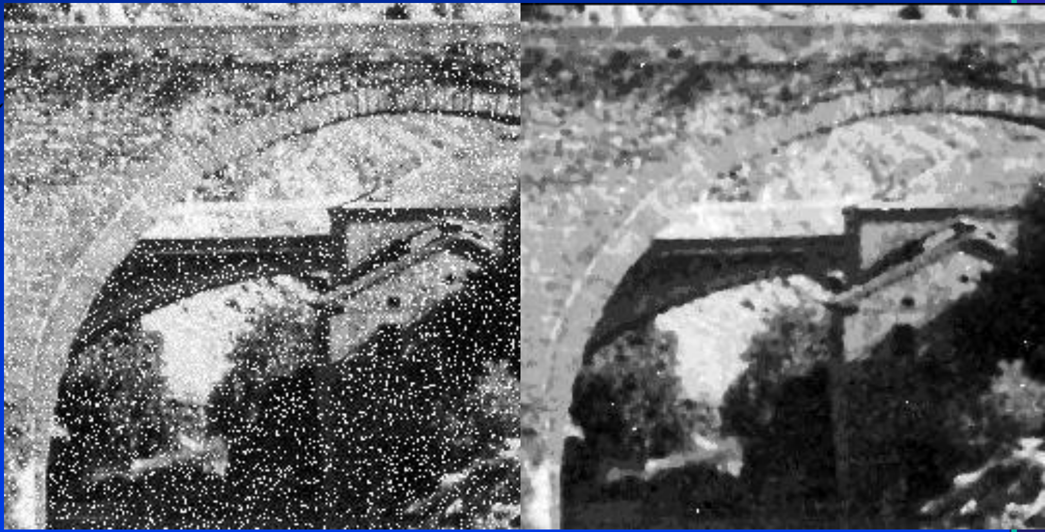
- Again, gradient magnitude is what we want, not direction





Median Filter

- The median filter is example of an *order-statistics filter*
- Employs local statistical information about pixels to produce output pixel
- Note that we don't use a fixed mask for all pixels
- With a median filter, look at local neighbor-hood and take median value



- Naturally, this requires some kind of sorting algorithm

Median Filter

- Median filters are effective for removing **impulse noise**, also called **salt-and-pepper noise**
- Suppose we took the mean instead of the median?
- That's just image smoothing!
- Since median filters perform less blurring than smoothing masks, they end to preserve features like lines and edges

Image Enhancement via Image Masking/Subtraction

- Many other techniques for image enhance exist
- Say we want to visualize blood vessels in brain
- First, we take an image of brain (e.g., MRI)
  – this will be called the *mask*
- Then we inject a contrast agent and take another image
- Then we subtract first image from second
- The resulting image shows changes introduced by contrast agent

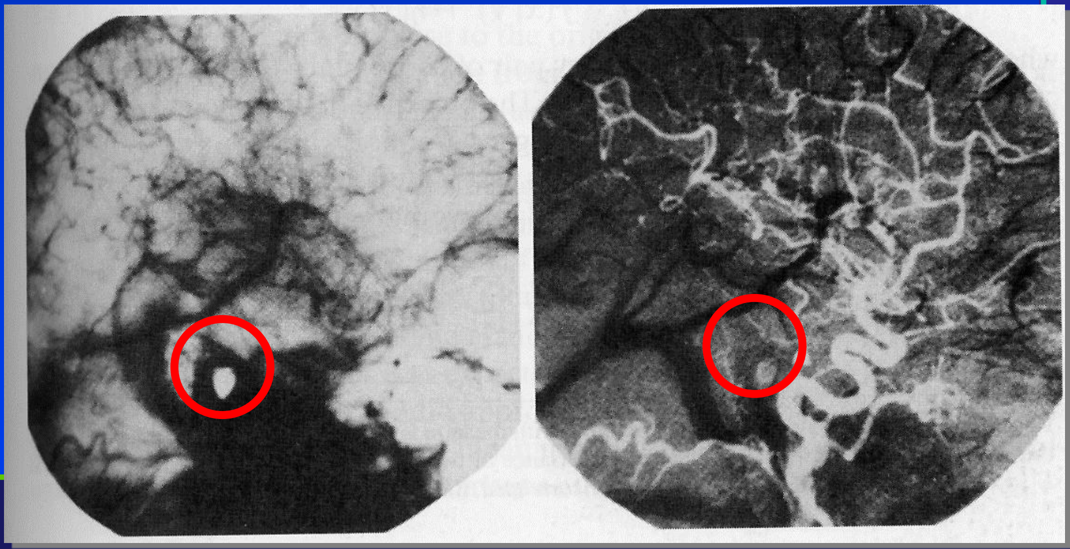
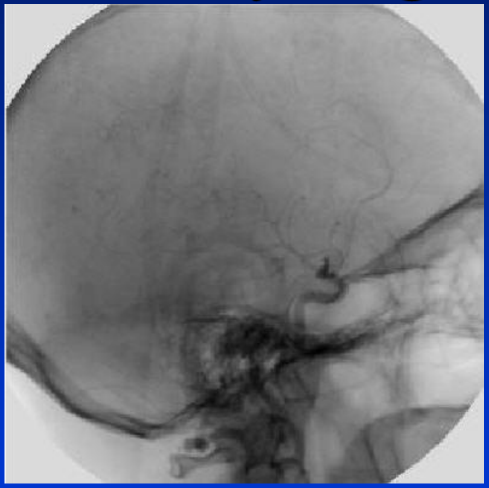


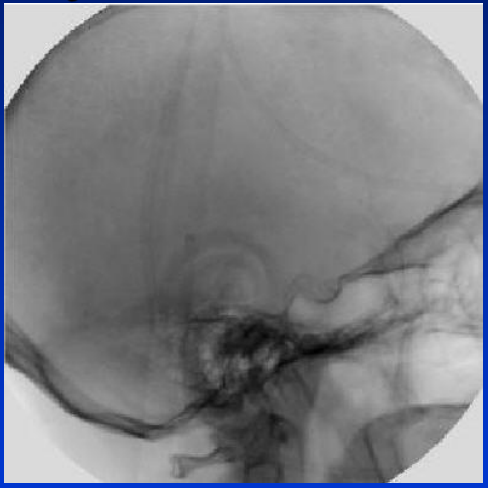
Image Subtraction Example

- X-ray angiography to enhance *perfused* vessels



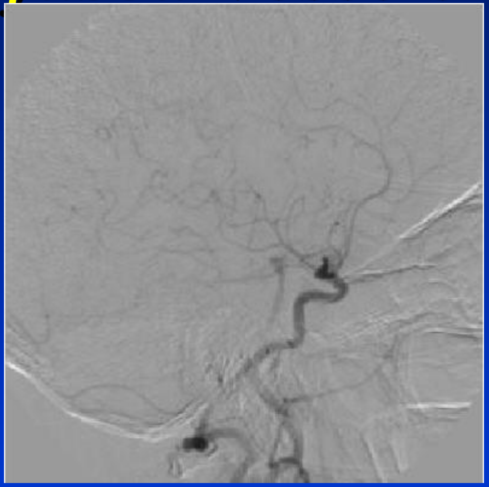
perfused

–



non-perfused (mask)

=

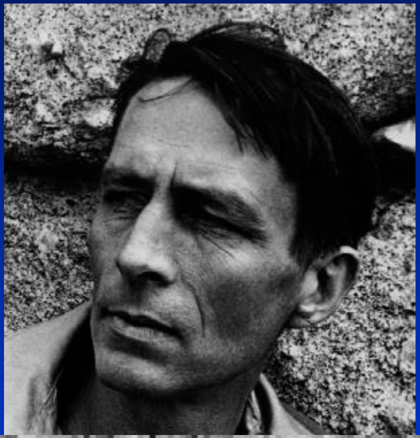


contrast-enhanced

- Perfuse = to force fluid through something

Subsampling

- Sometimes we need to change image resolution
- Subsampling used to decrease resolution
- Supersampling used to increase resolution
- How can we improve image quality in both cases?
- Interpolation



Quantization

- Very common technique in all of computer science
- Basic idea: represent broad range of values using a much smaller set
- In image processing: reduce numb of graylevels (bits) represented
- For normal vectors: store only a subset of the infinite number of possibilities (unit sphere)

Color Transformations

- Image transformations not limited to intensity trans.
- We can also transform the H and S channels using transfer functions

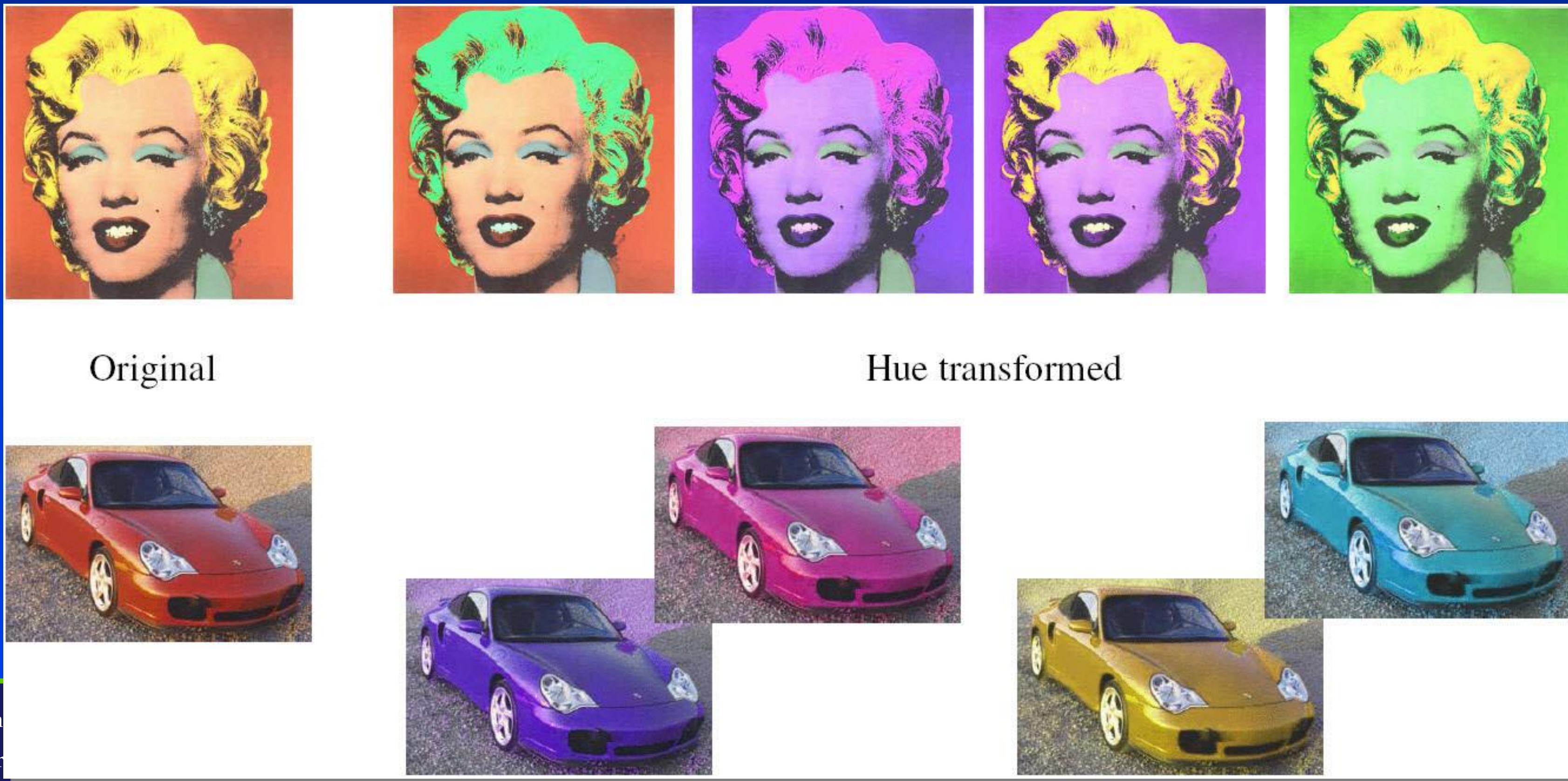


Image Sampling

Sampling

- Image acquisition: sampling a continuous object or scene into a discrete image grid
  - digital camera
  - flatbed/desktop scanner
  - medical scanner, such as MRI, CT

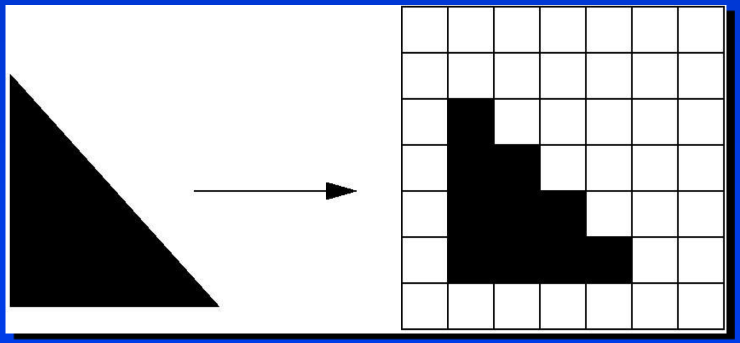
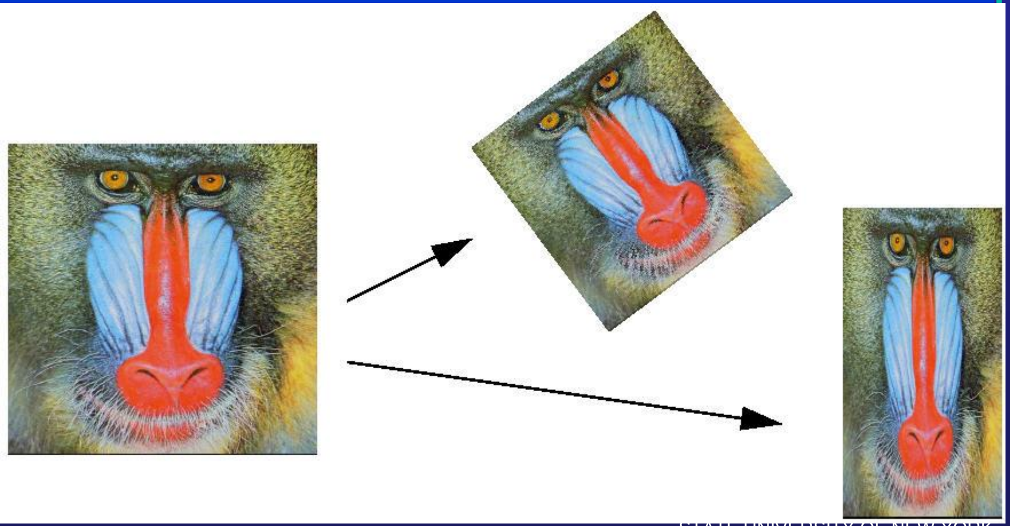


Image Manipulation

- Image manipulation: resampling an already discrete image
  - reduce or enlarge the image
  - increase or decrease resolution possibly
  - rotate, shear, squeeze



Sampling and Reconstruction

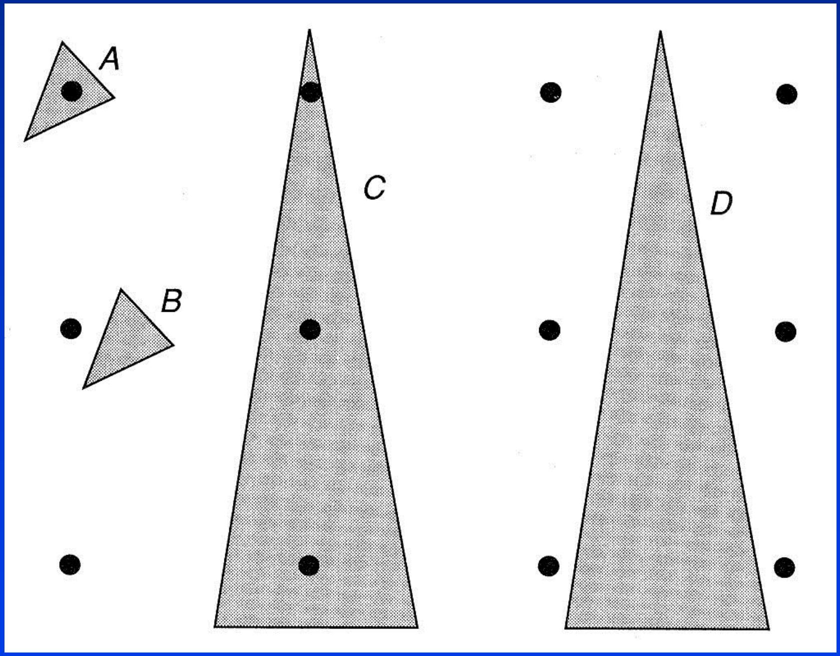
- Consider a photo taken by a digital camera
- The camera *samples* a continuous *signal*, the scene, onto a regular 2D grid (*raster*) to generate the pixels
- When we look at the resulting photo, our eye *reconstructs* the original continuous scene if the image resolution is high enough
- Same thing happens when we look at printed material (e.g., laser printer 600+ dpi)
- If the image is of low resolution, it seems blocky – our eye is not given enough data to reconstruct the original scene
- We see defects or artifacts in the photo because the original data-set (the scene) was *under-sampled*
- This phenomenon is called *aliasing*

Aliasing

- Aliasing includes a class of artifacts that arise due to undersampling, i.e., when an insufficient number of samples is taken from an in put *signal*
- Think of the signal as a generic function in 1D, 2D, 3D, etc.
- Aliasing often occurs in:
  - image size reduction (subsampling)
  - discretization, causing jagged edges or ‘jaggies’ (see the triangle in first slide); also called *staircasing artifacts*
- Intuitively, aliasing is caused by a lack of information
- Not enough information is sampled from the input to represent (approximate) the original data-set fairly or accurately
- Let’s look at how sampling is typically done

Point Sampling

- Simplest way to select each pixel's value is just to sample the original signal at some location, and take that value as the pixel's value
- We will assume, as usual, that the pixels are arranged on a regular, uniform, rectangular grid

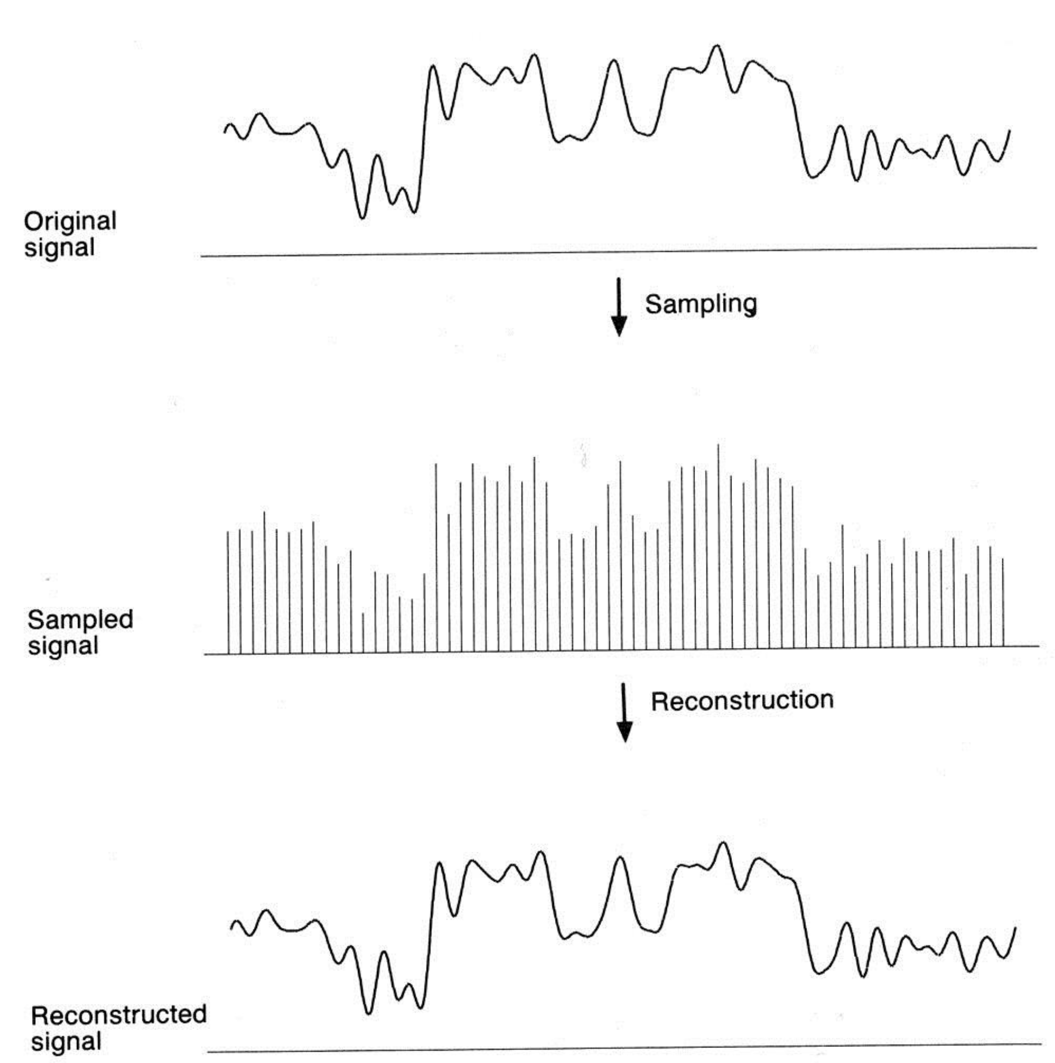


- Can you see any problems with sampling the input using this particular point sampling?
- Triangles B and D are totally missed by this particular point sampling; other point samplings would catch them

Reconstruction from Point Sampling

A successful example

- Interpolation fills in the gaps between adjacent samples
- Different interpolators (functions) will generate different reconstructions
- We will study exactly how interpolation works in a couple weeks

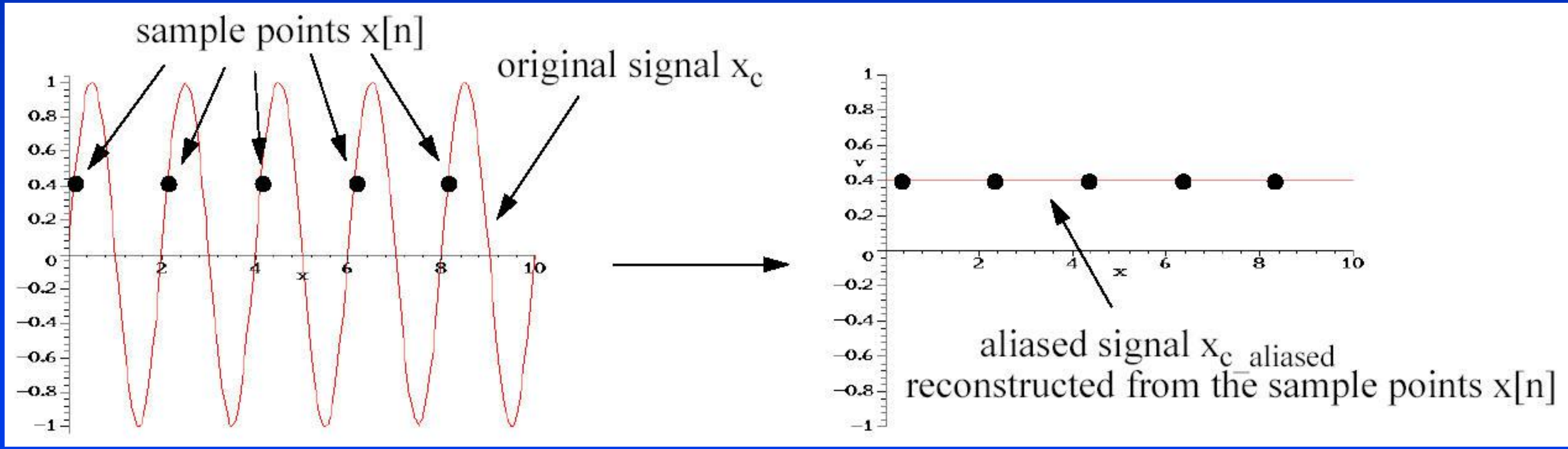


Point Sampling Pitfalls

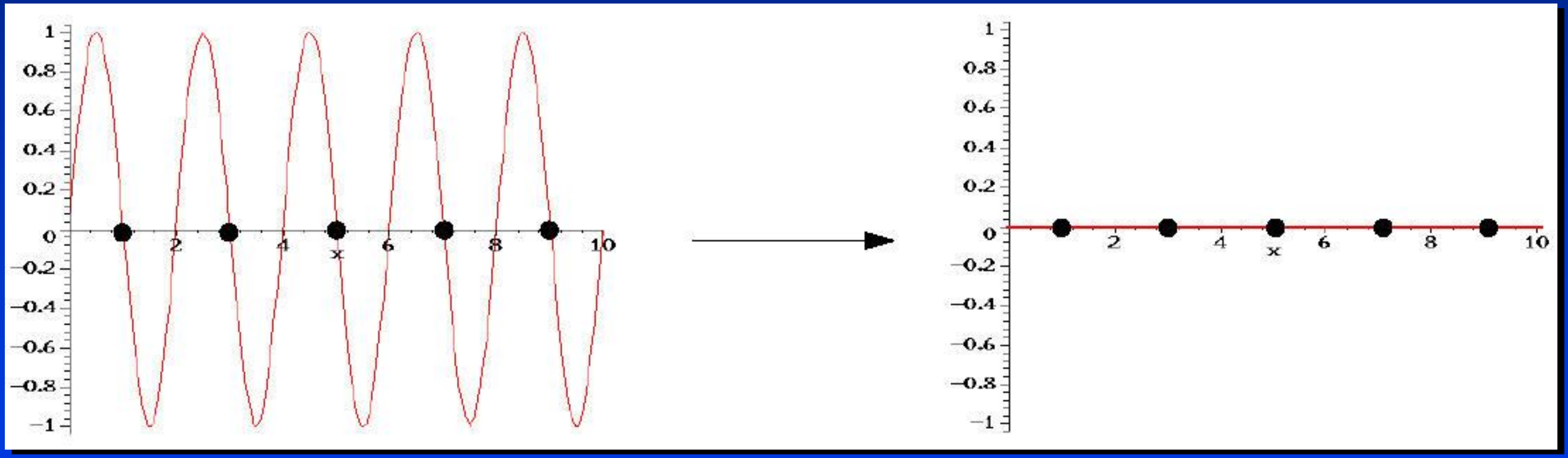
- In the previous example, we took enough samples to recover the original signal
- That is, the *sampling frequency* was high enough in order to record enough information to reconstruct the input
- But, as we saw with the triangles, this is not always possible:
  - fixed resolution
  - memory budget
  - computation time
- Let's look at some examples and try to determine a scheme for deciding what the minimum sampling frequency needs to be in order to be able to reconstruct the original signal faithfully with minimum error and minimum aliasing

Sine Wave Aliasing – Example 1

- Frequency of original signal: 0.5 (oscillations per time unit)
- If the sampling frequency is also 0.5 (samples per time unit), the original signal can not be recovered

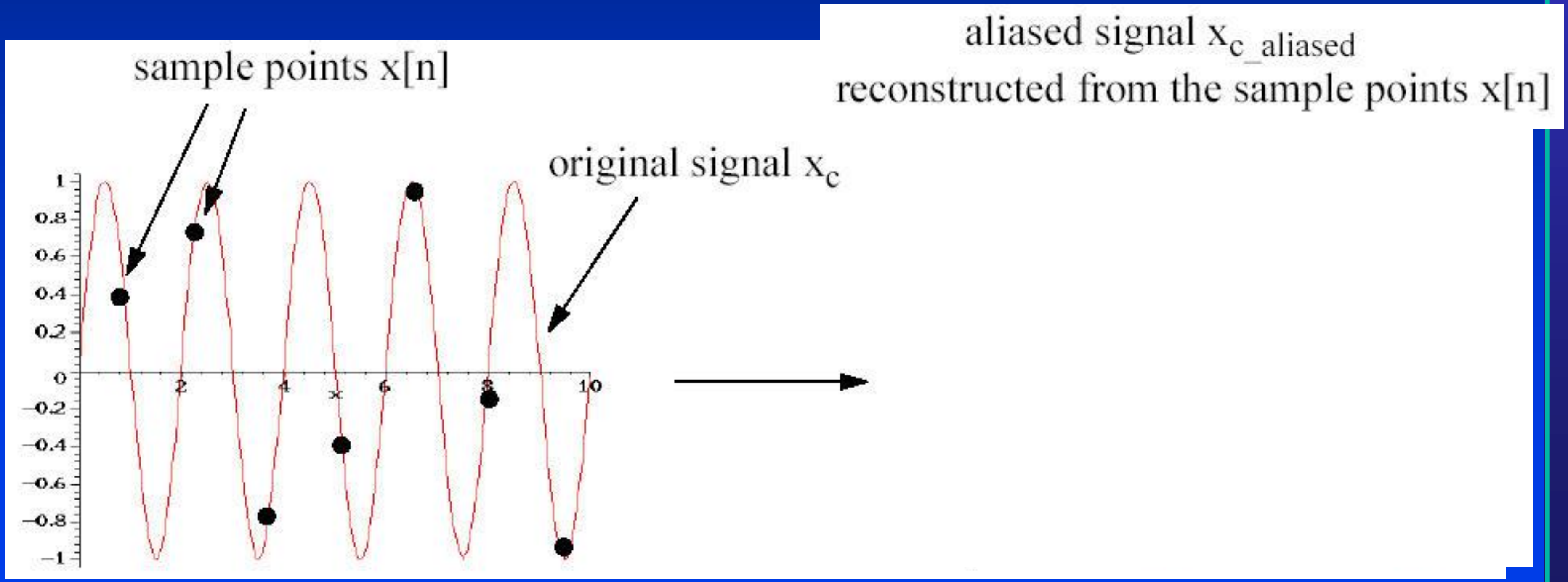


Sine Wave Aliasing – Example 1



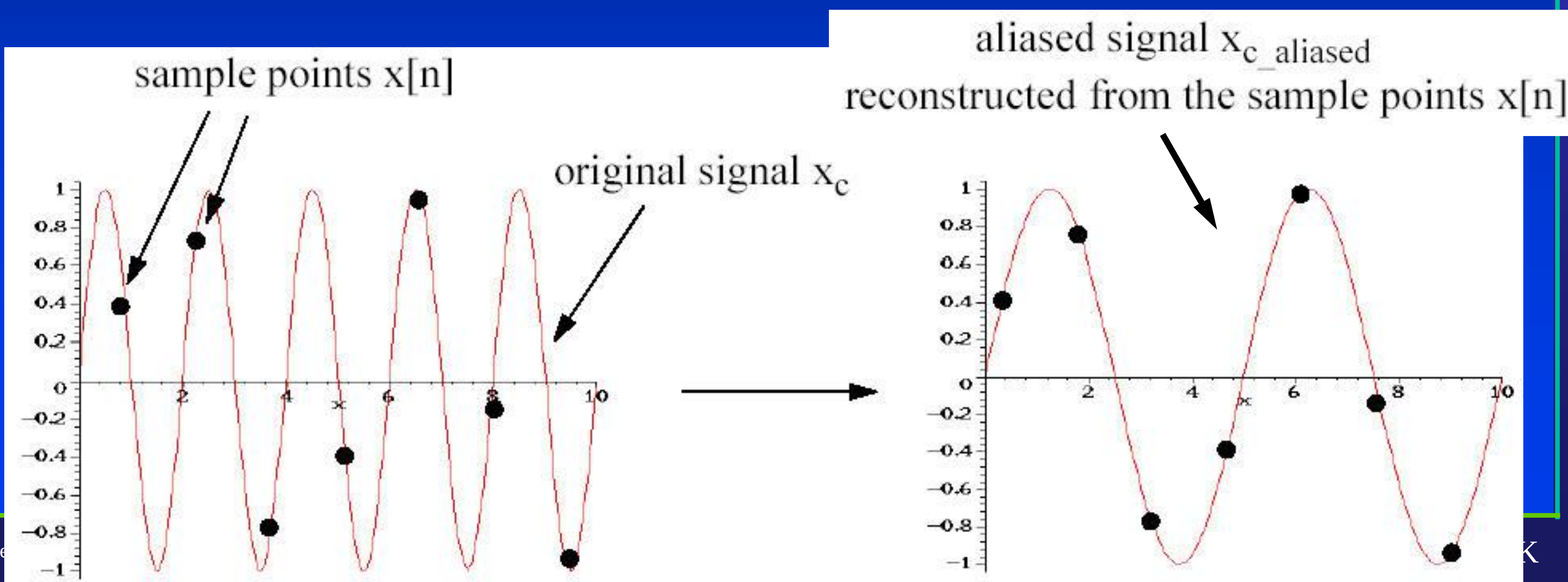
Sine Wave Aliasing – Example 2

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 0.7 (samples per time unit)



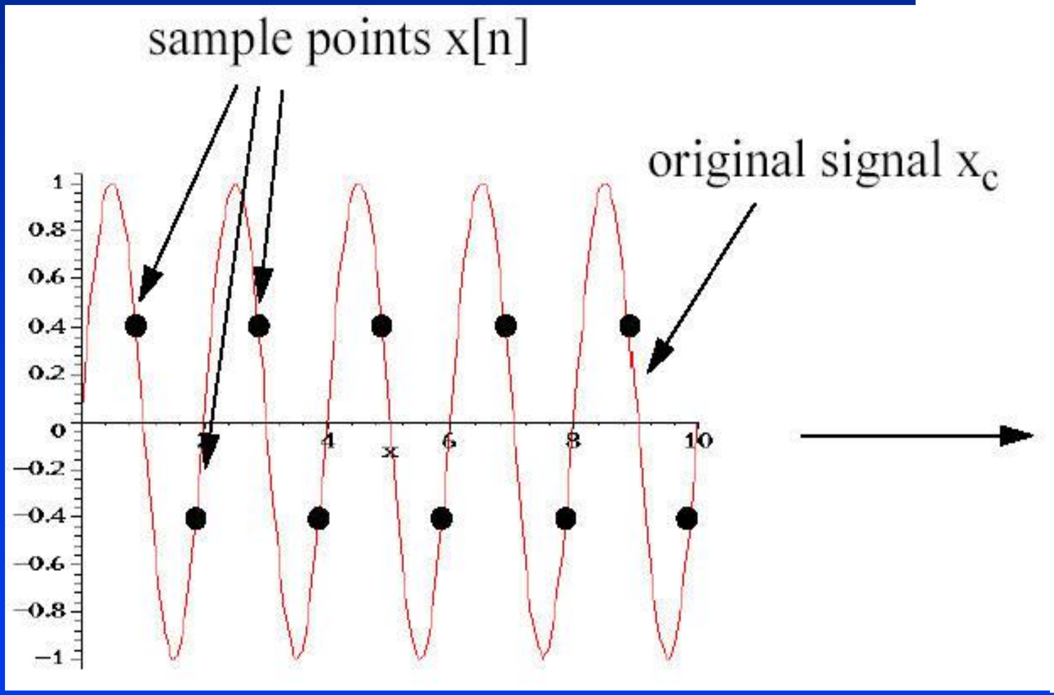
Sine Wave Aliasing – Example 2

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 0.7 (samples per time unit)
- Looking at the sample points x[n], they appear to originate from a sine wave $x_{c\_aliased}$ of much lower frequency
- Again, the original sine wave (the input signal) is lost and can not be recovered



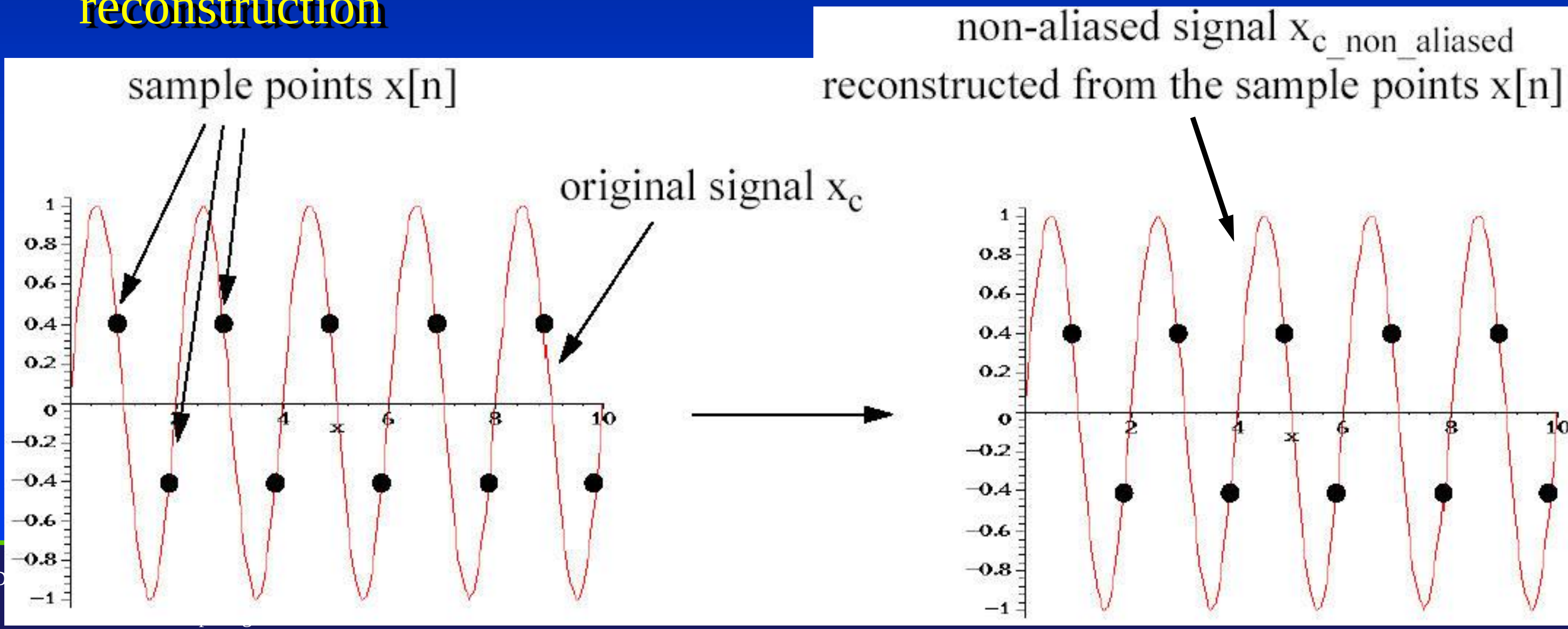
Sine Wave Aliasing – Example 3

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 1.0 (sample per time unit)



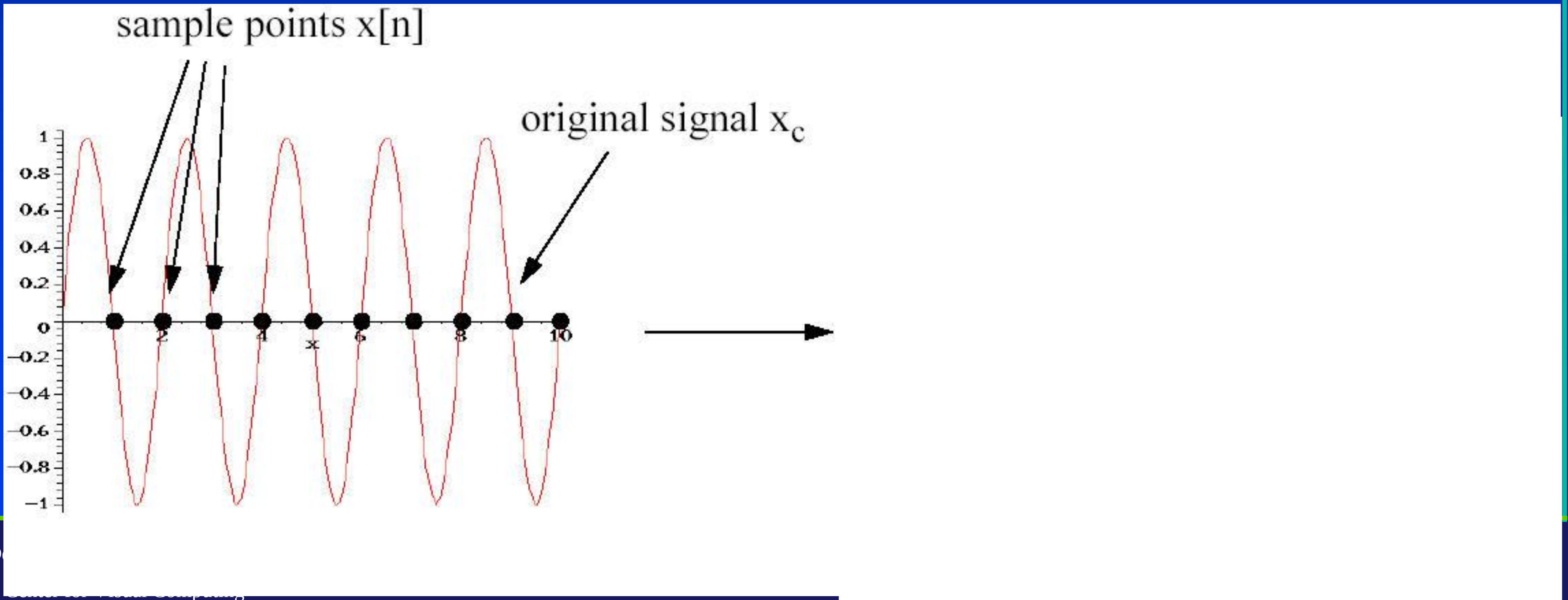
Sine Wave Aliasing – Example 3

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 1.0 (sample per time unit)
- Now the original signal can be recovered
- We learn that we need to sample each oscillation period twice for good reconstruction



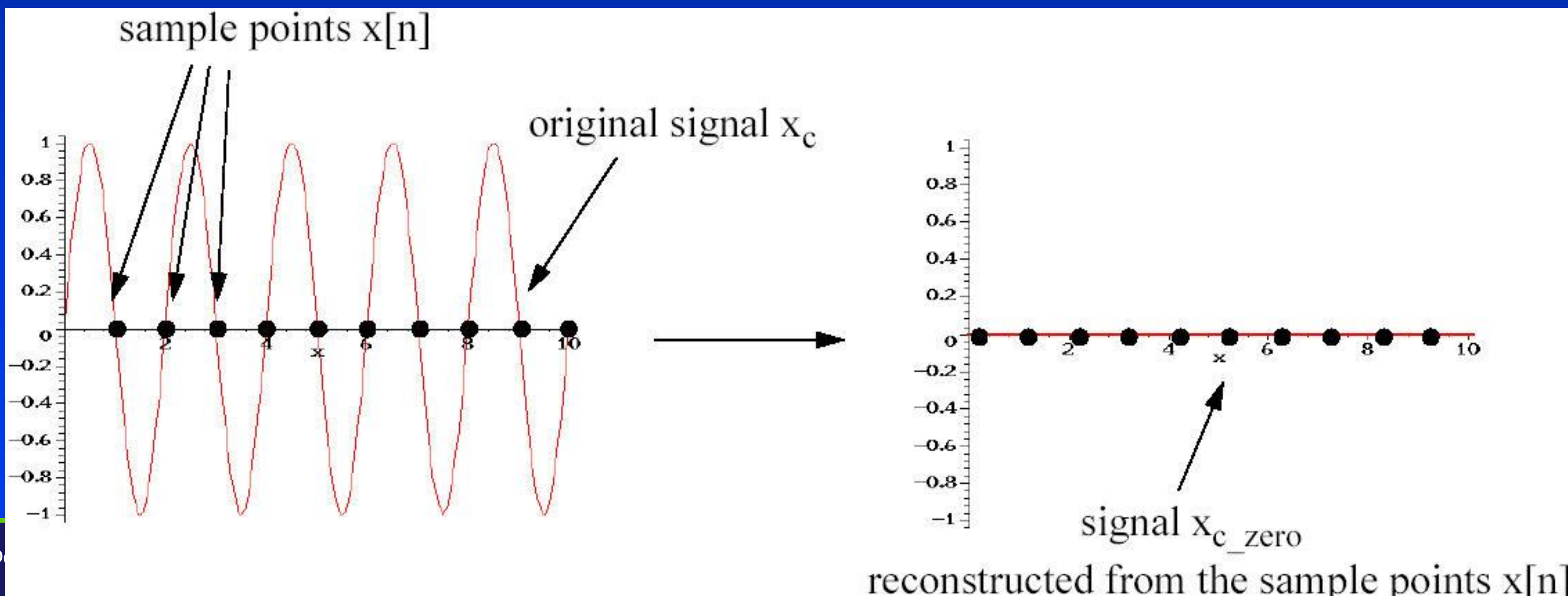
Sine Wave Aliasing – Example 4

- In practice, it is best to use more than two samples per oscillation period
- One may get wrong reconstructions for some special sample alignments using exactly two samples per oscillation
- Thus, to be on the safe side, sample each oscillation period more than twice



Sine Wave Aliasing – Example 4

- In practice, it is best to use more than two samples per oscillation period
- One may get wrong reconstructions for some special sample alignments using exactly two samples per oscillation
- Thus, to be on the safe side, sample each oscillation period more than twice



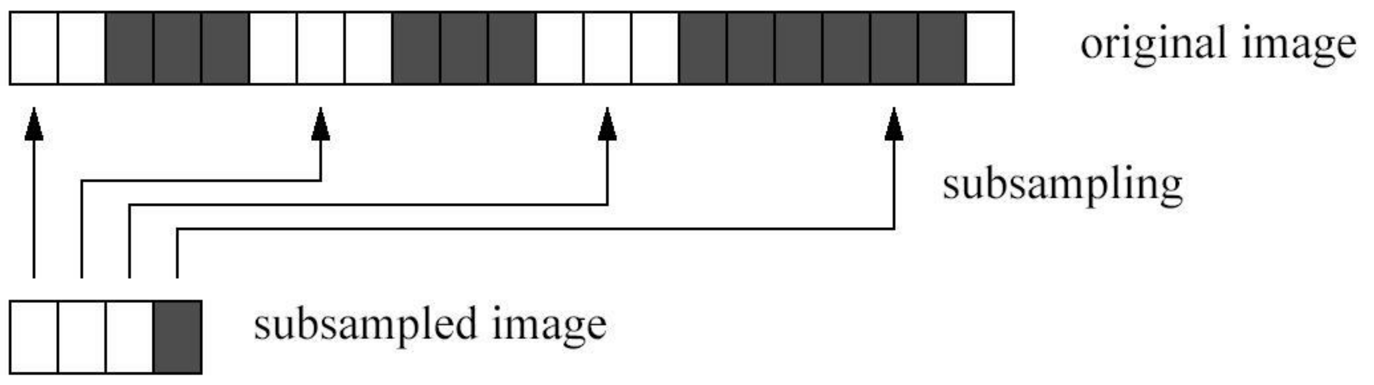
Aliasing

- In a nutshell, aliasing manifests itself as frequencies that appear in the sampled signal that do not appear in the original input
- Once this happens, you're basically doomed
- Nothing you can do will bring back the original
- The same thing can happen with images
- Let's take a look

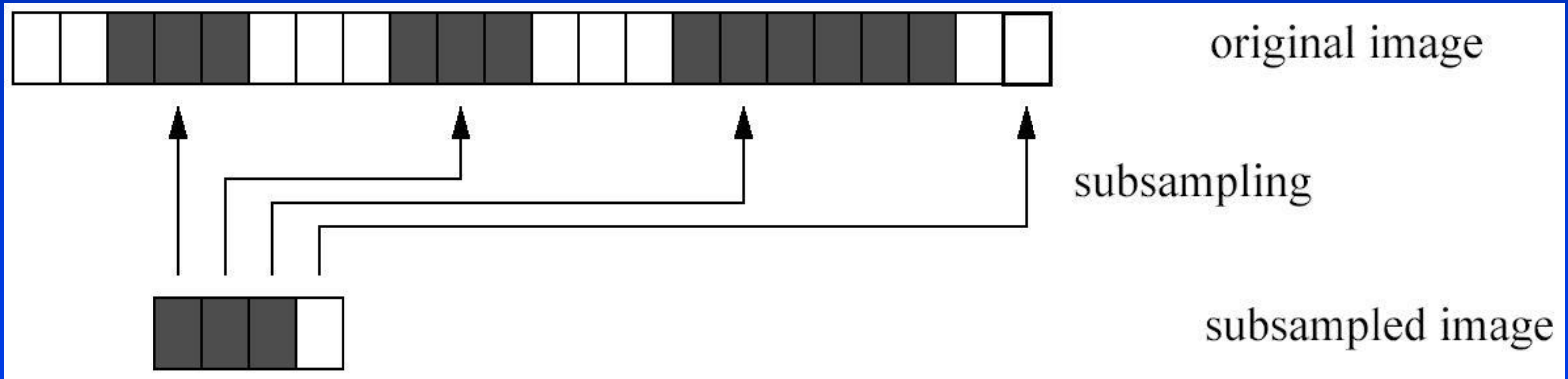
Point Sampling: Discrete Example

- Image reduction (i.e., resolution reduction) involves a type of sampling: *resampling* a given 2D image to generate a new 2D image
- This process is called *subsampling* when we shrink an image
- When we enlarge an image, we need to *supersample* it to generate the output image
- Subsampling can lead to aliasing just as can the sampling of a continuous signal
- Let’s try using point sampling to subsample a 1D image to reduce its size and see how aliasing might arise

Point Sampling: Aliasing



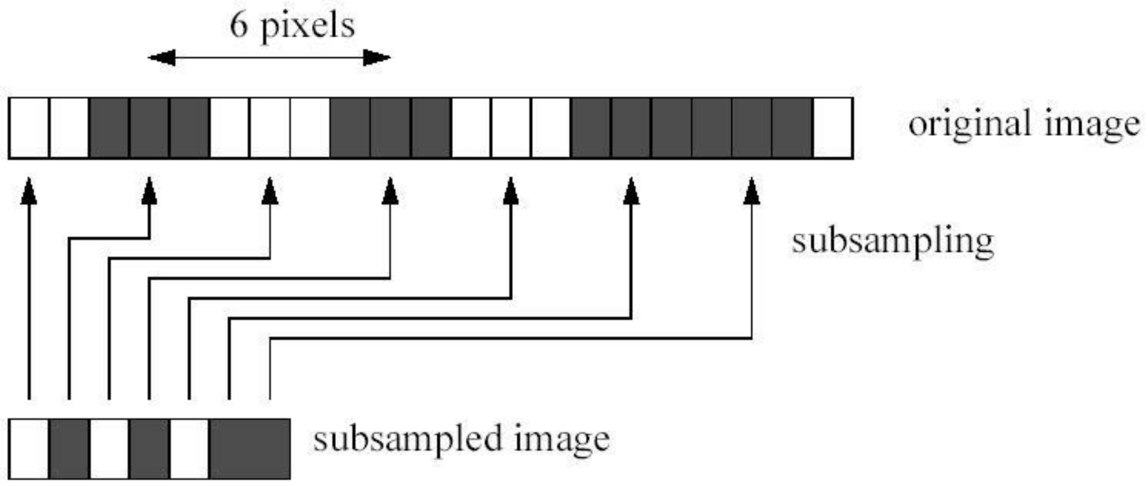
- This subsampling missed two of the three stripes…



- ….whereas this subsampling captured them, but the result isn't striped

The Nyquist Theorem

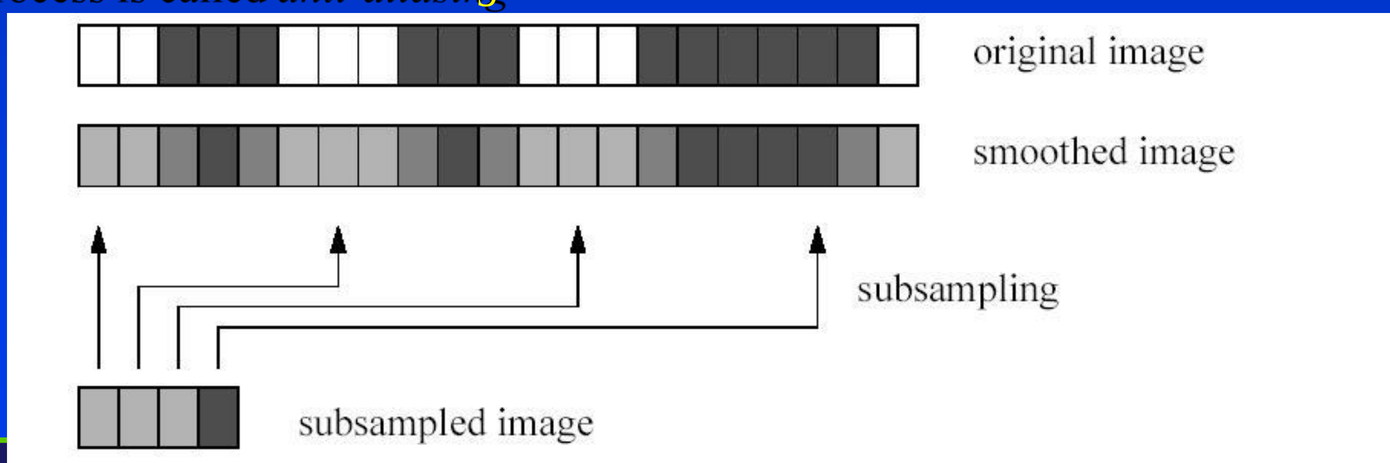
- The duration (period) of each of the first two stripes is 6 pixels
- Since frequency = 1/period, the frequency of the stripes is 1/6
- If we want to catch each stripe in the subsampled image we need to sample every third pixel
- That is, we need to sample the stripes at a frequency = 2·(1/6) = 1/3
- Note: all stripes are present (no matter at what offset we sample)



- This is formalized by the *Nyquist theorem*, which states: to avoid aliasing, one must always sample at least at twice the highest frequency in the image

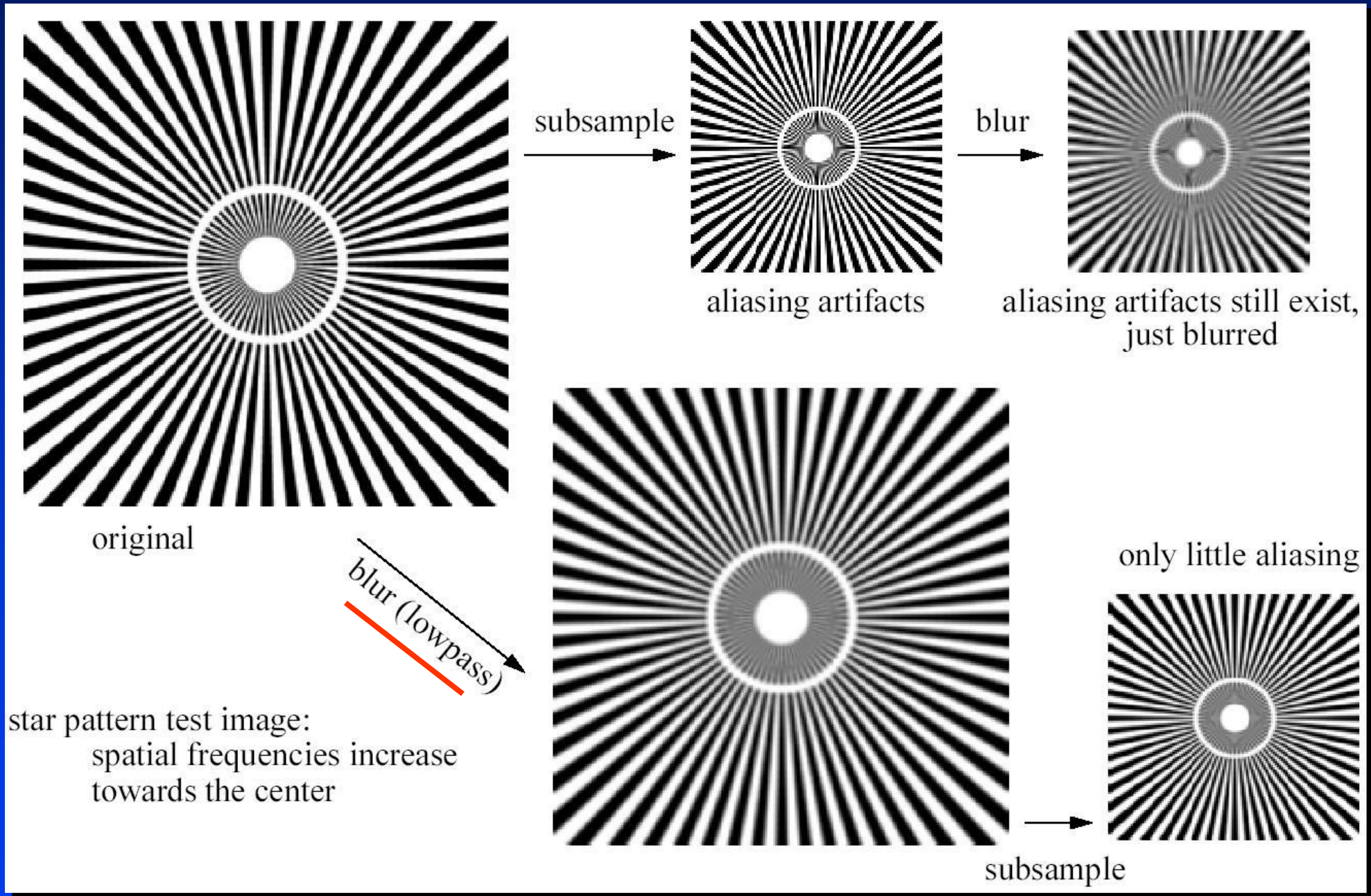
Anti-Aliasing

- But what do we do if we cannot afford to sample at or above this Nyquist frequency?
- This can occur when:
  - we would like to reduce the size of the image below its allowed Nyquist limit
  - the scene to be discretized is very busy and our digital camera does not have enough resolution
  - "busy" here means that the scene (signal) has many high frequencies
- In these cases we must reduce the frequency content, *before* sampling takes place
- This is done by smoothing (blurring) the image or scene (prior to sampling)
- This process is called *anti-aliasing*



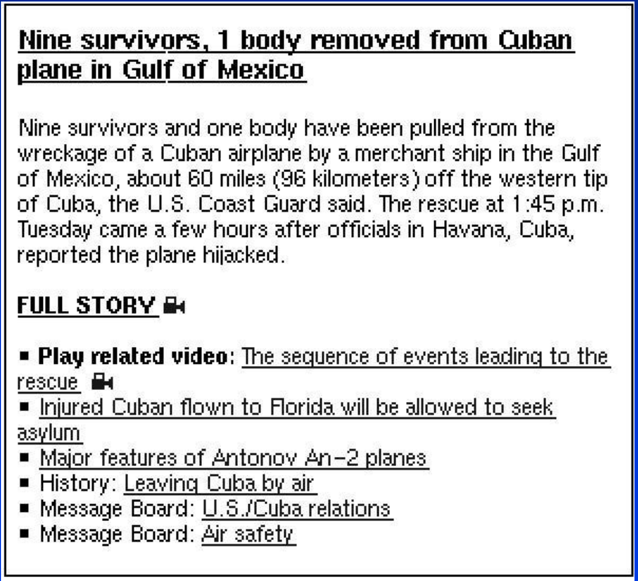
although some stripes appear blurred, all stripes contribute to the image (at all offsets)

Anti-Aliasing Example - 1



Anti-Aliasing Example - 2

We observe: anti-aliasing (i.e., blurring, lowpassing) must be applied before sampling

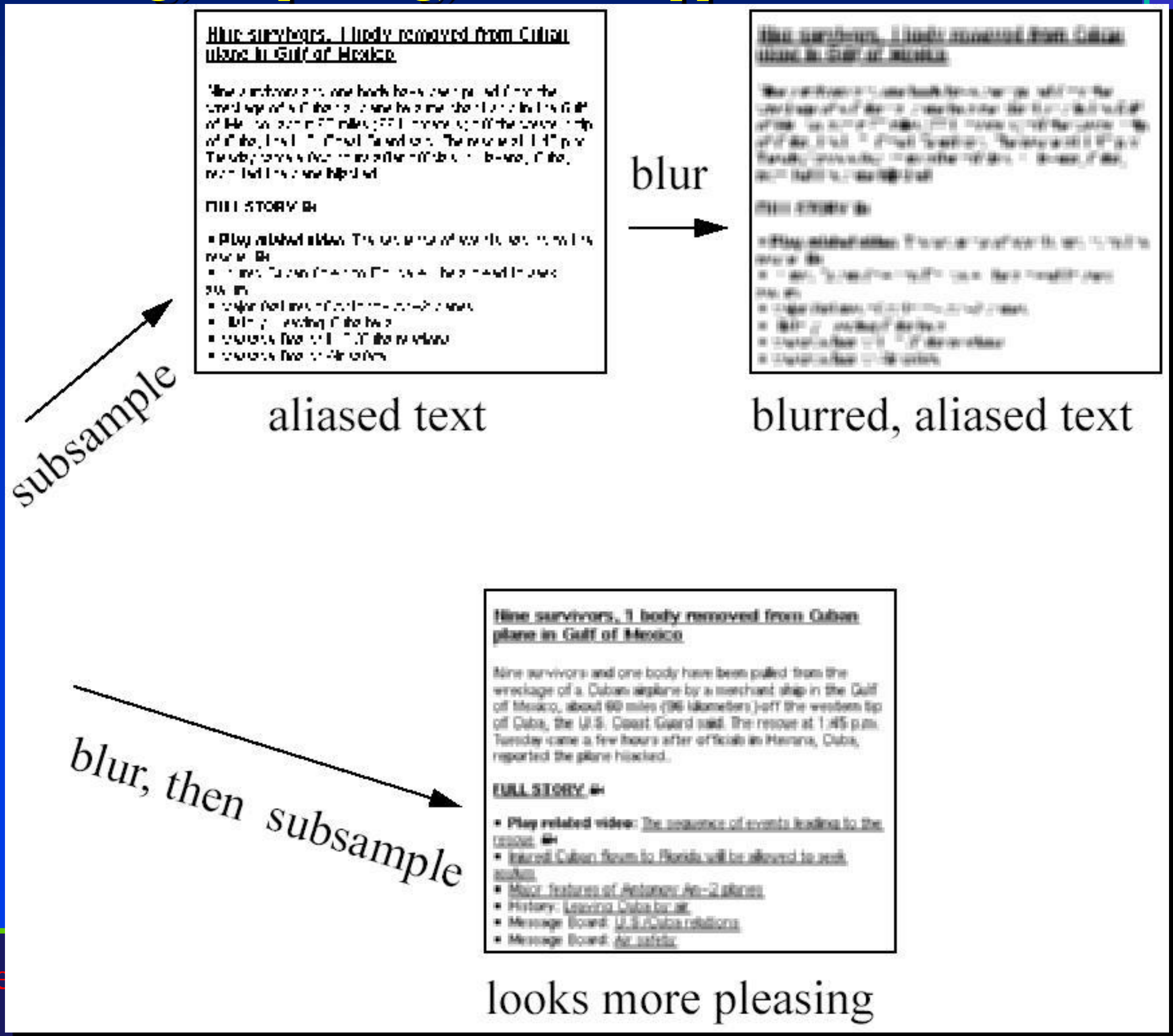


Nine survivors, 1 body removed from Cuban plane in Gulf of Mexico

Nine survivors and one body have been pulled from the wreckage of a Cuban airplane by a merchant ship in the Gulf of Mexico, about 60 miles (96 kilometers) off the western tip of Cuba, the U.S. Coast Guard said. The rescue at 1:45 p.m. Tuesday came a few hours after officials in Havana, Cuba, reported the plane hijacked.

FULL STORY

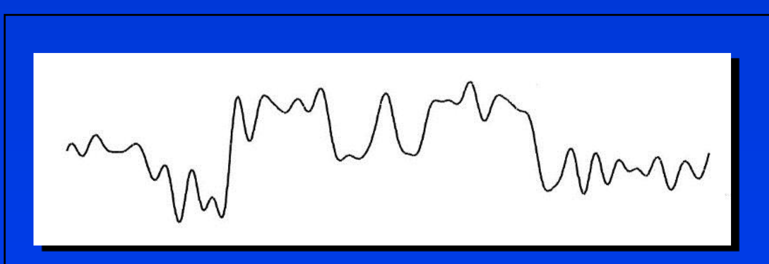
- **Play related video:** The sequence of events leading to the rescue
- Injured Cuban flown to Florida will be allowed to seek asylum
- Major features of Antonov An-2 planes
- History: Leaving Cuba by air
- Message Board: U.S./Cuba relations
- Message Board: Air safety

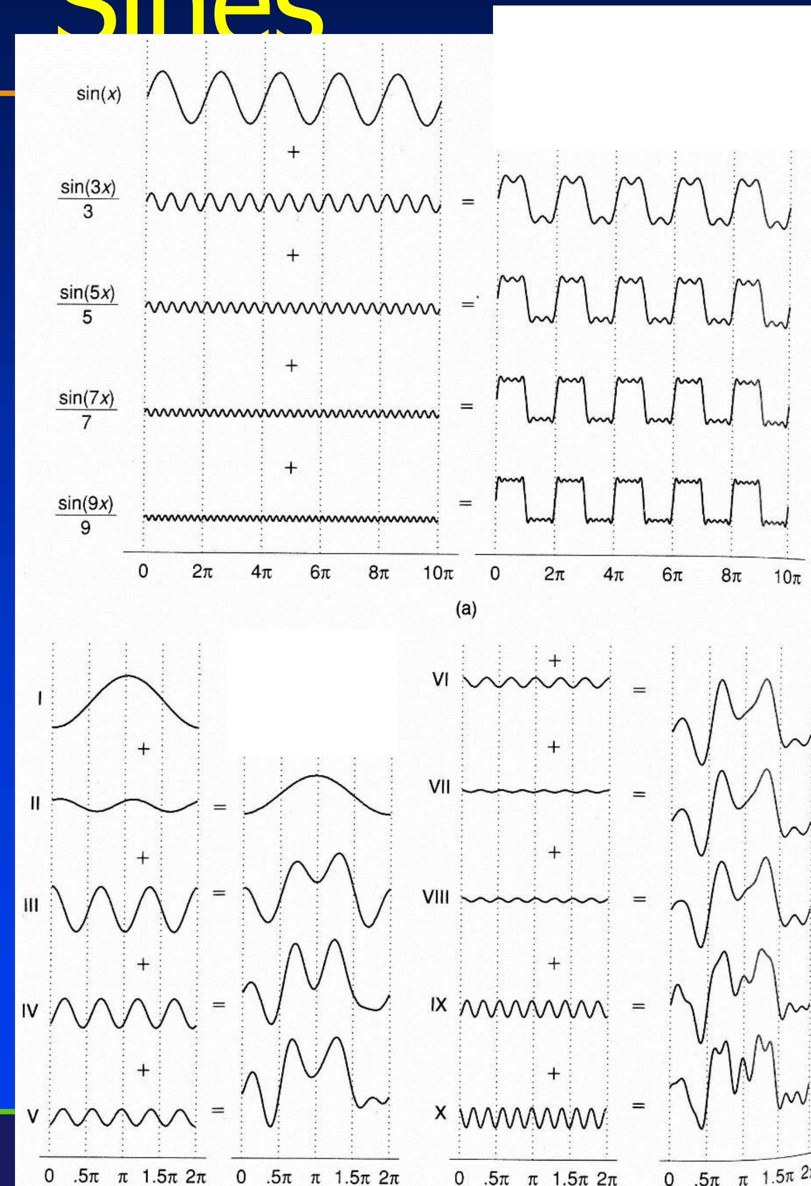


Sampling Theory

- So far we have been considering signals in the *spatial domain*, i.e., a signal as a plot of amplitude (intensity for pixels) against spatial position (we looked at 1D and 2D)
- A signal may also be considered in the *frequency domain*, i.e., as a sum of sine waves, possibly offset from each other (called *phase shift*) and each having different frequencies and amplitudes
- Each sine wave in the (possibly infinite) sum represents a component of the signal's *frequency spectrum*
- Periodic signals can each be represented as the sum of phase-shifted sine waves whose frequencies are integral multiples (*harmonics*) of the signal's *fundamental frequency* (frequency component that has the greatest wavelength)
- Let's look at some examples

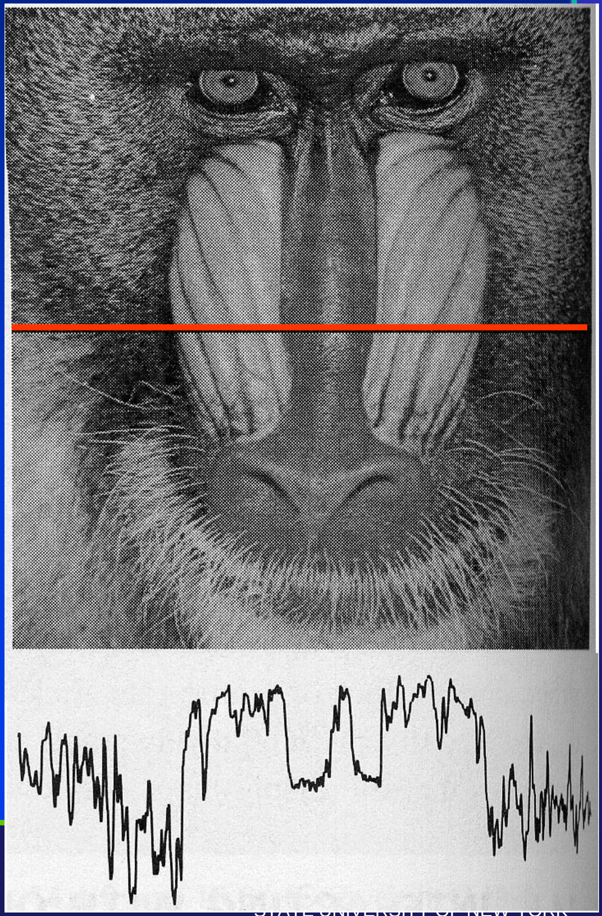
Signals as Sums of Sines





Images as Signals

- Can we apply this idea to images, which are non-periodic?
- Yes, with some important distinctions
- Finite domain, zero outside the image
- An image's frequency spectrum, however, will not consist of integer multiples of some fundamental frequency
- Original signal cannot be represented as a sum of countably many sine waves, but as an integral over a continuum of frequencies
- *Fourier analysis* is the process by which we determine which sine waves must be used to represent a particular signal
- Very important subject in EE



Temporal Aliasing

- Aliasing can manifest itself over time as well as space (i.e., images)
- Wagon wheel in old Western movies: