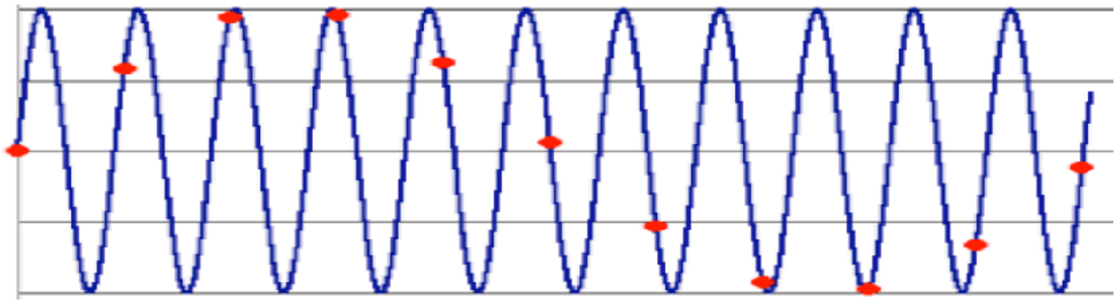
COMP0026: Image Processing

Filtering and Edge Detection

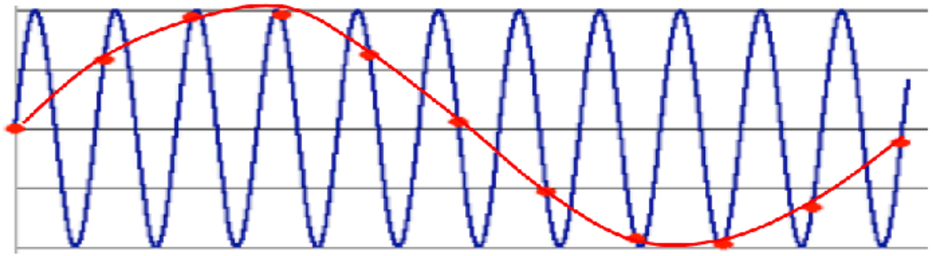
Lectures will be Recorded

Undersampling



- **Aliasing** can arise when you sample a continuous signal or image
  - occurs when your sampling rate is not high enough to capture the amount of detail in your image
  - Can give you the wrong signal/image—an *alias*
  - formally, the image contains structure at different scales
    - called "frequencies" in the Fourier domain
  - the sampling rate must be high enough to capture the highest frequency in the image

Aliasing and Nyquist Rate



- **Aliasing** can arise when you sample a continuous signal or image
  - occurs when your sampling rate is not high enough to capture the amount of detail in your image
  - Can give you the wrong signal/image—an *alias*
  - formally, the image contains structure at different scales
    - called "frequencies" in the Fourier domain
  - the sampling rate must be high enough to capture the highest frequency in the image
- To avoid aliasing:
  - sampling rate ≥ 2 * max frequency in the image
    - said another way: ≥ two samples per cycle
  - This minimum sampling rate is called the **Nyquist rate**

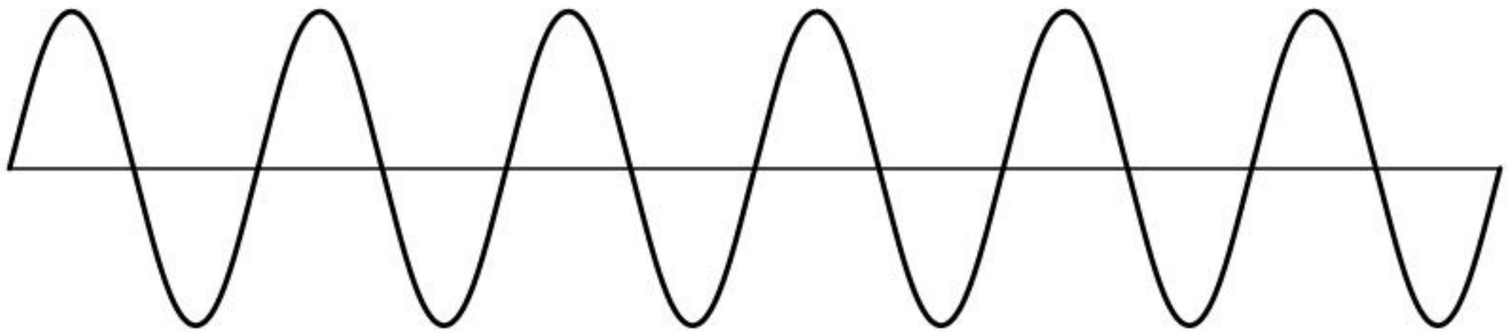
Aliasing

Aliasing artifacts, including“stair-casing”, jaggies and moiré patterns can appear because digital cameras have a limited number of samples/pixels to represent the signal. Therefore they have a maximum spatial frequency, called the Nyquist frequency, above which scene information cannot be correctly captured and will be “aliased” to a lower spatial frequency.

Similar to one-dimensional discrete-time signals, images can also suffer from aliasing if the sampling resolution, or pixel density, is inadequate. For example, a digital photograph of a striped shirt with high frequencies (in other words, the distance between the stripes is small), can cause aliasing of the shirt when it is sampled by the camera's image sensorThe aliasing appears as a moiré pattern. The "solution" to higher sampling in the spatial domain for this case would be to move closer to the shirt, use a higher resolution sensor, or to optically blur the image before acquiring it with the sensor using an optical low-pass filter.

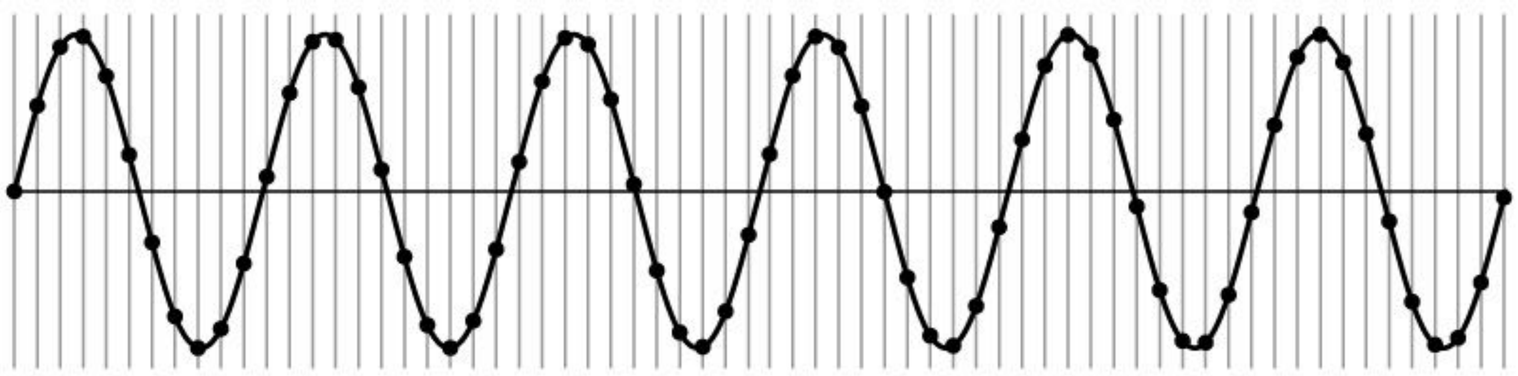
Sampling and Reconstruction

- **Simple example: a sine wave**



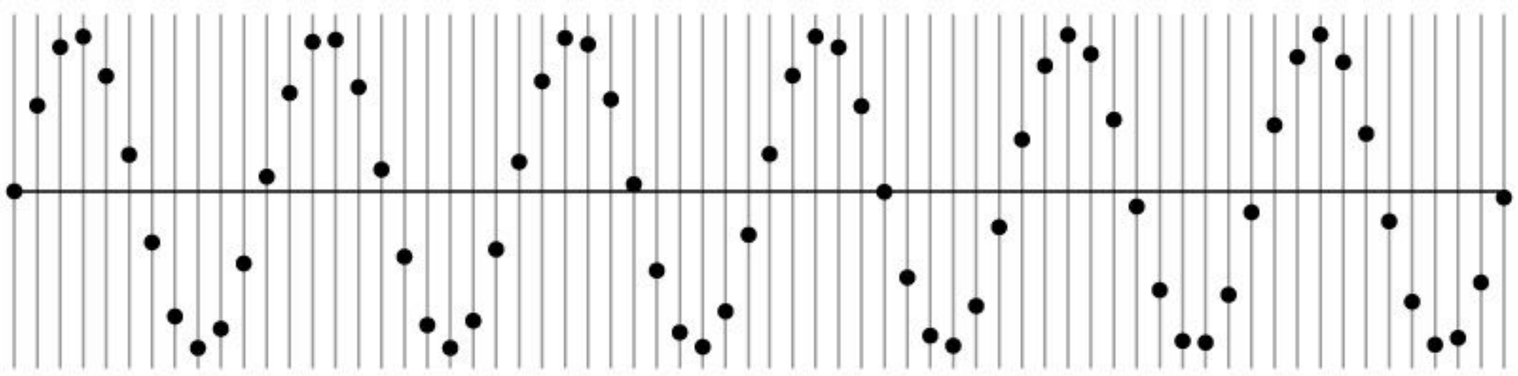
Sampling and Reconstruction

- **Simple example: a sine wave**



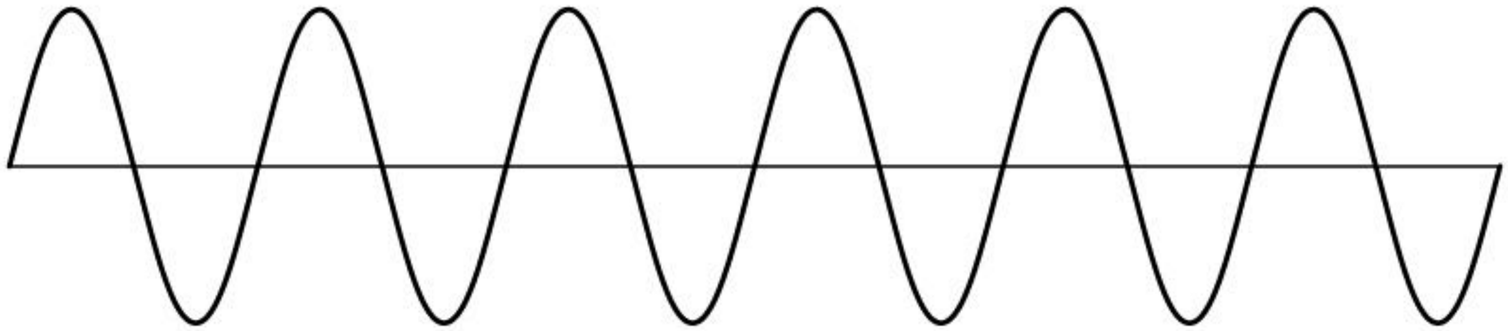
Sampling and Reconstruction

- **Simple example: a sine wave**



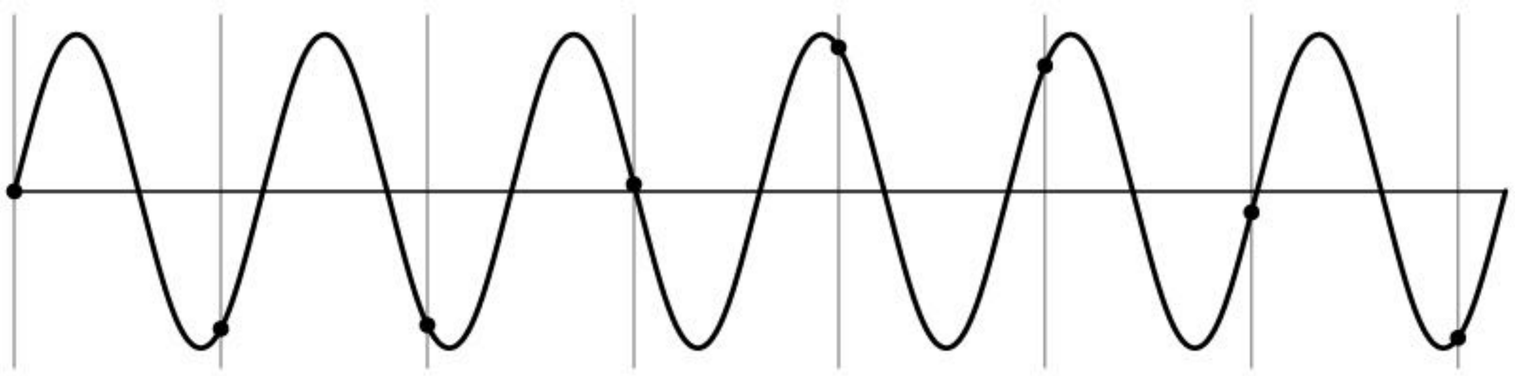
Sampling and Reconstruction

- **Simple example: a sine wave**



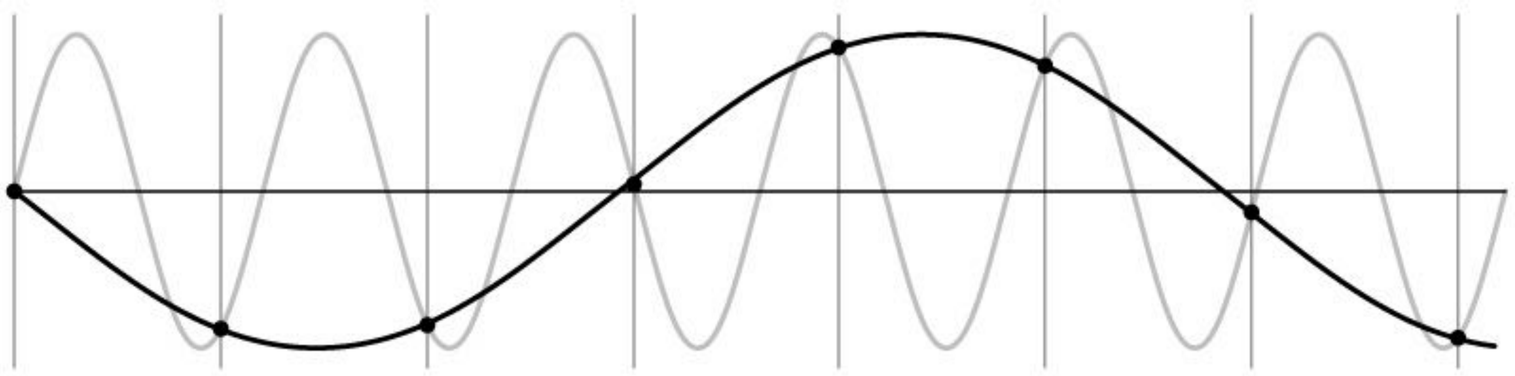
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost



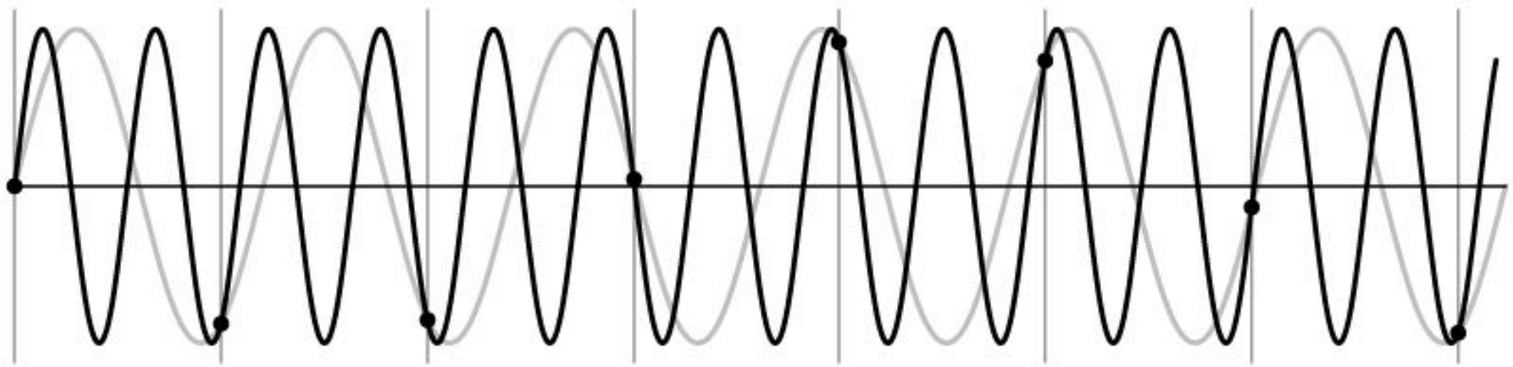
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency

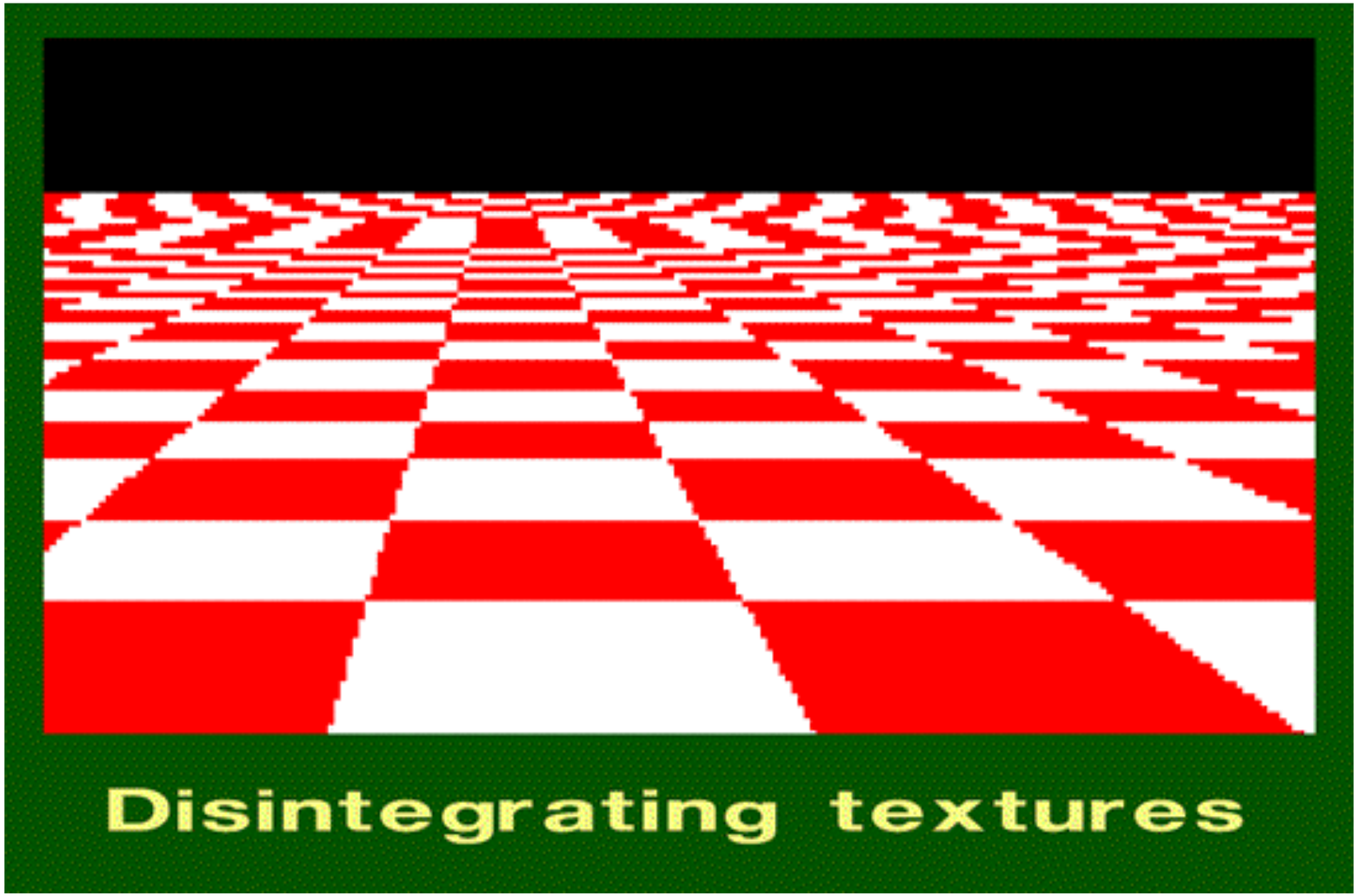


Undersampling

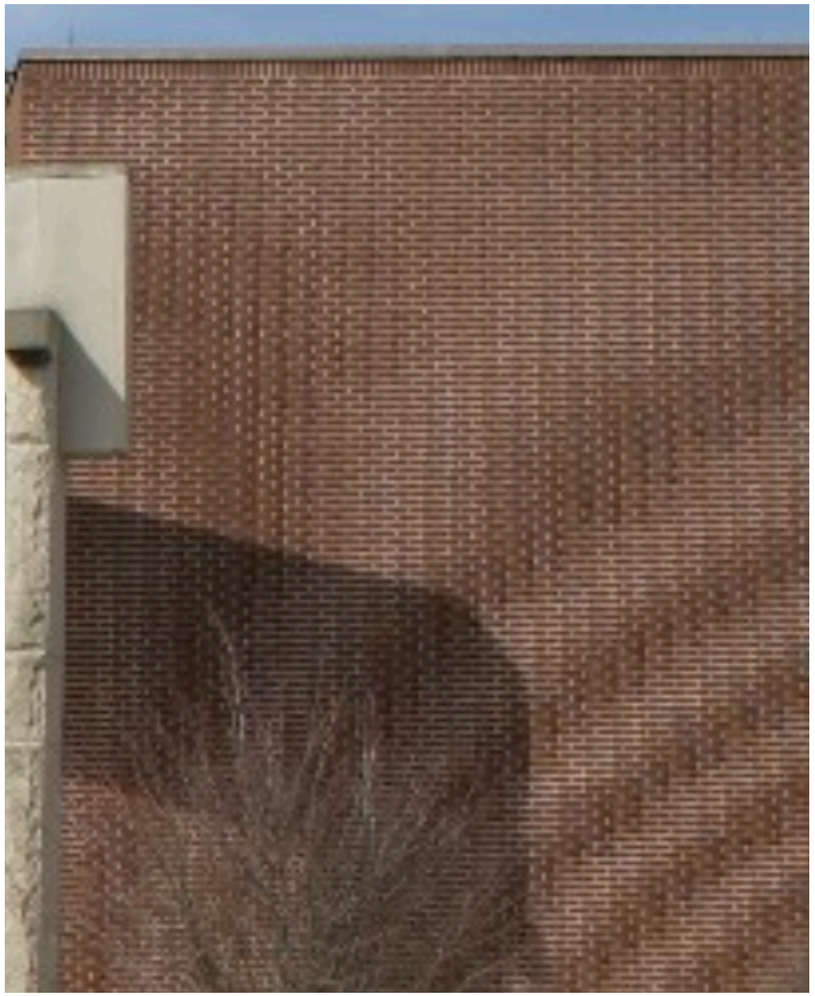
- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals “traveling in disguise” as other frequencies



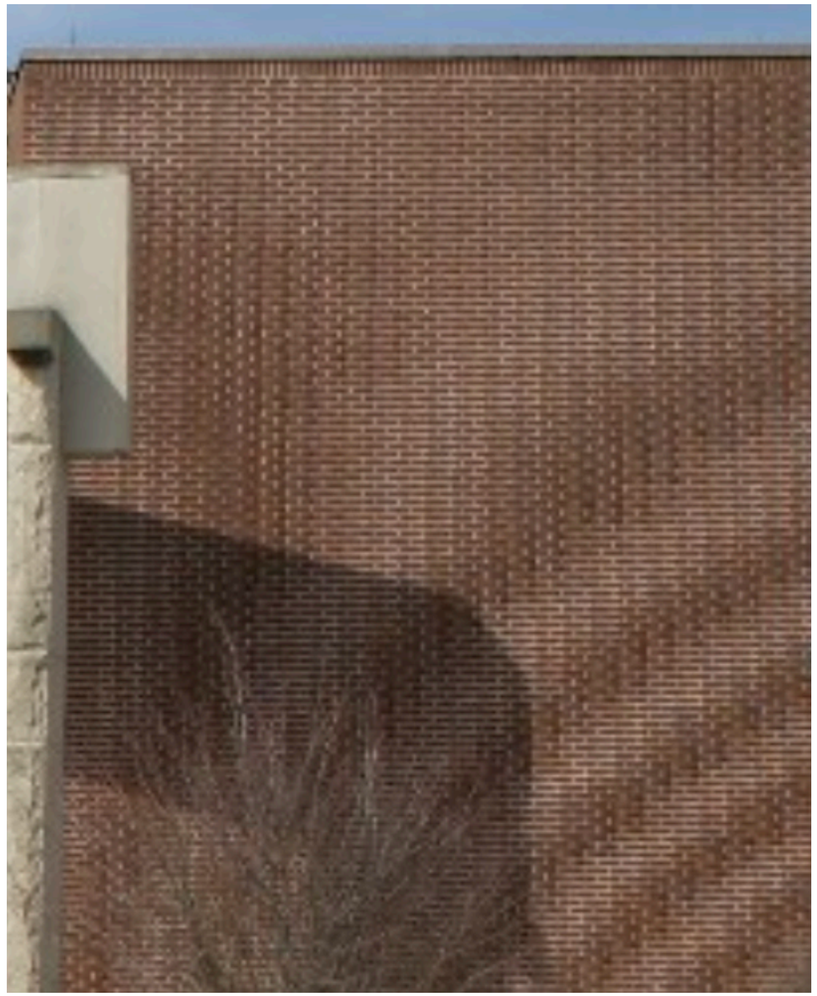
Aliasing in Images

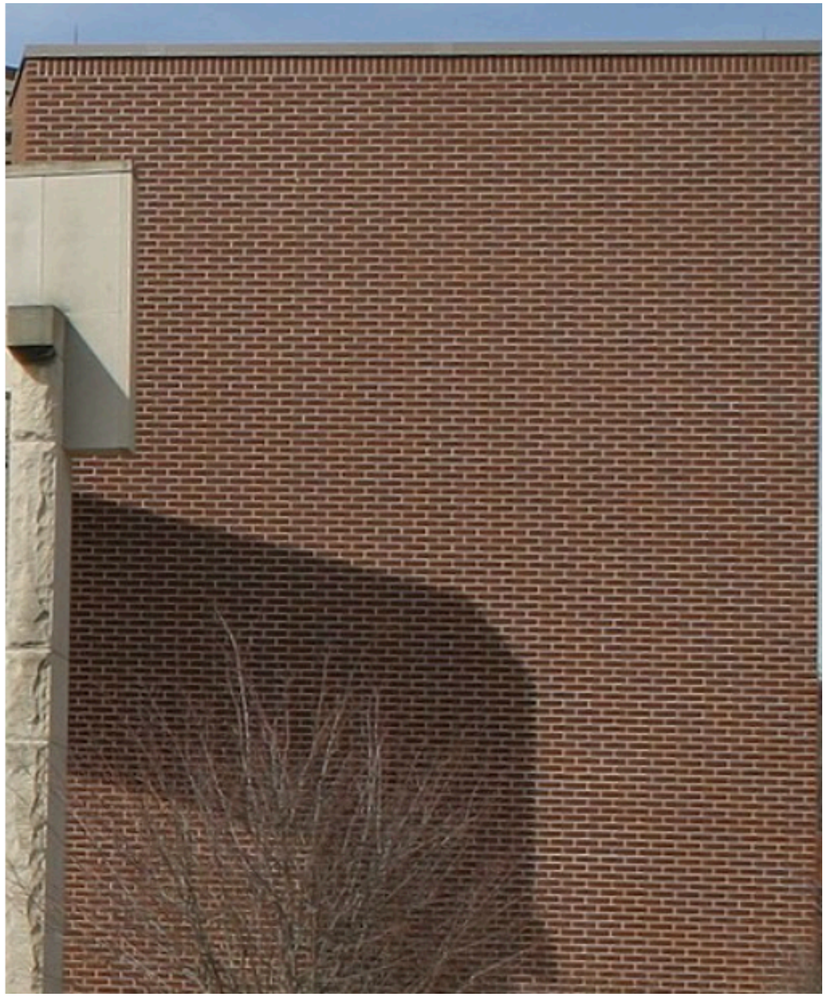


Aliasing in Images



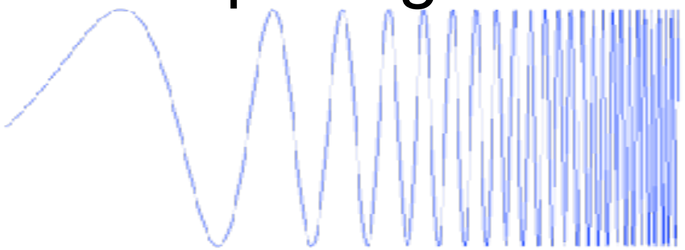
Aliasing in Images





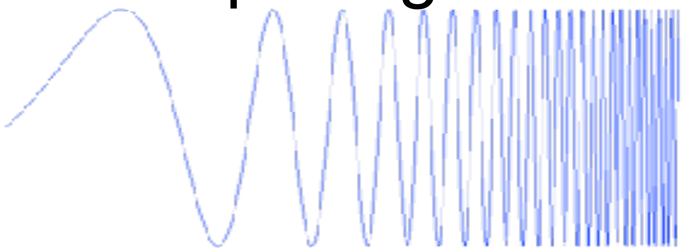
What’s Happening?

Input signal:

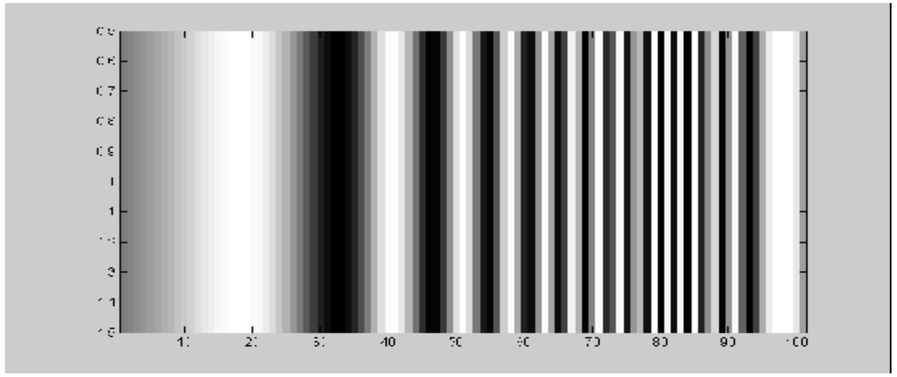


What's Happening?

Input signal:



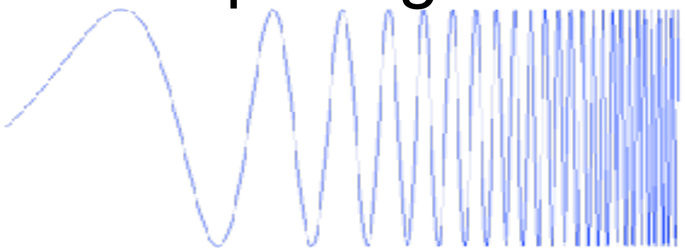
Plot as image:

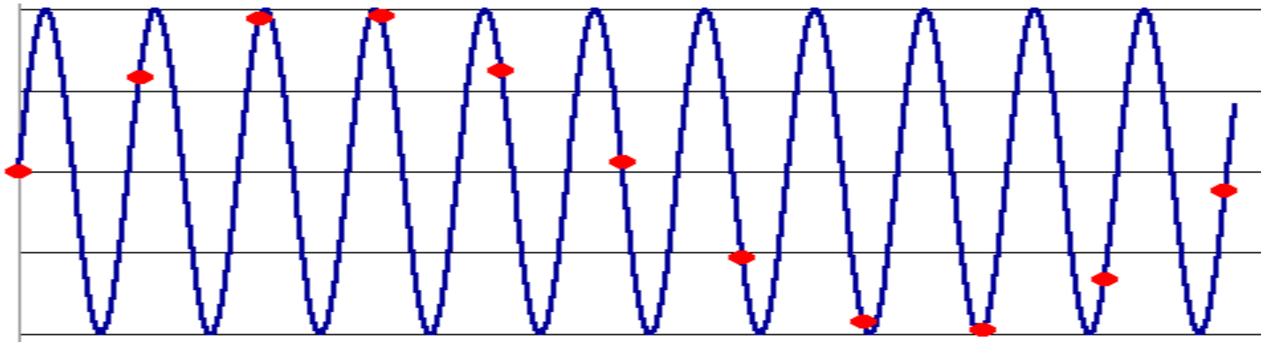


$$x = 0 : 0.05 : 5; \quad \text{imagesc}(\sin(2^x x))$$

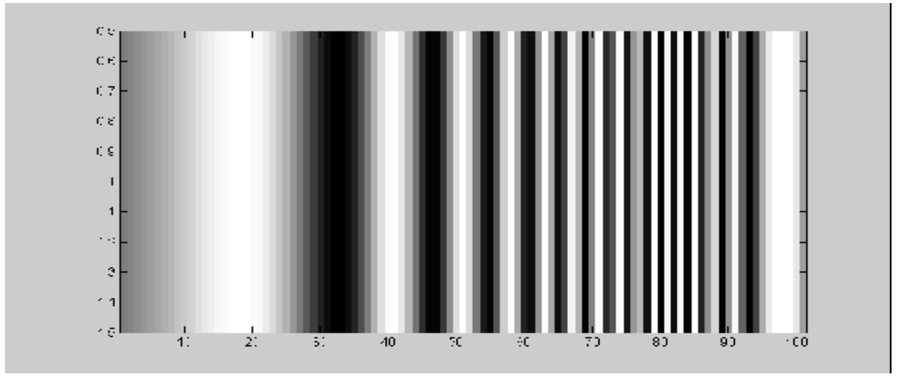
What's Happening?

Input signal:





Plot as image:



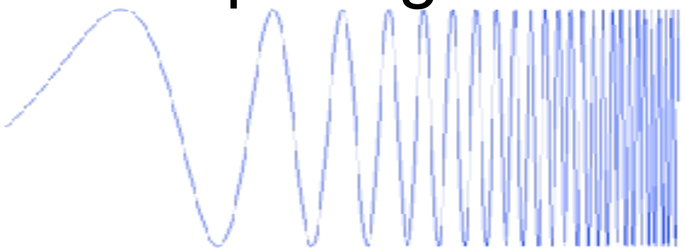
$$x = 0 : 0.05 : 5; \quad \text{imagesc}(\sin(2^x x))$$

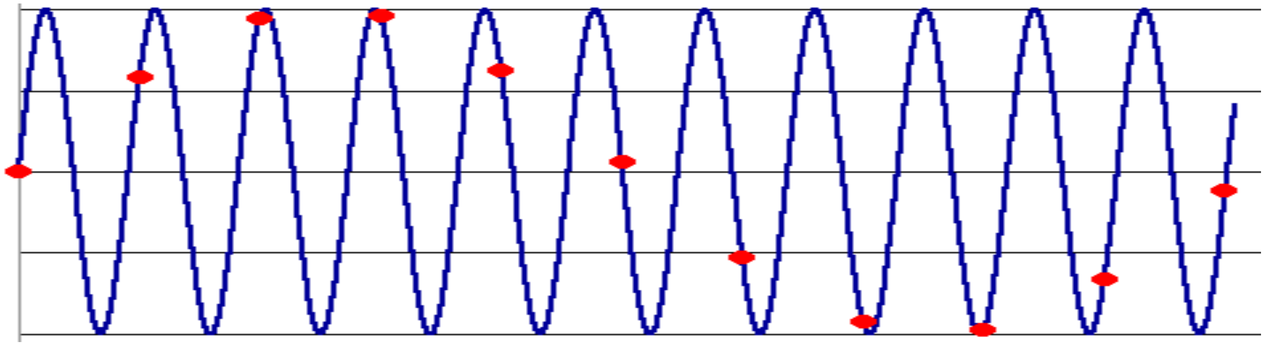
Alias!

Not enough samples

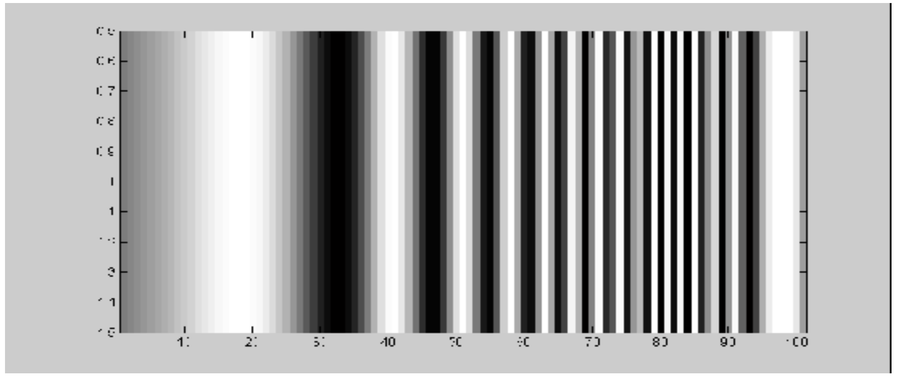
What's Happening?

Input signal:





Plot as image:



$$x = 0 : 0.05 : 5; \quad \text{imagesc}(\sin(2^x x))$$

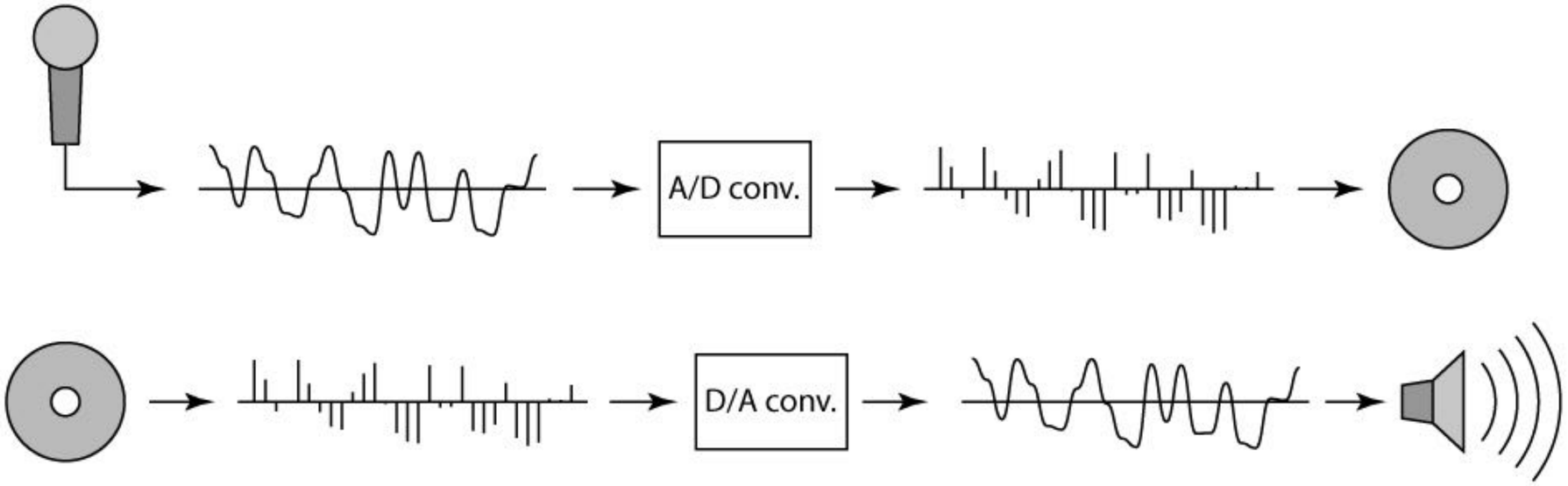
Alias!

Not enough samples

$$\tilde{F}(\mu) = \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F\left(\mu - \frac{n}{\Delta T}\right)$$

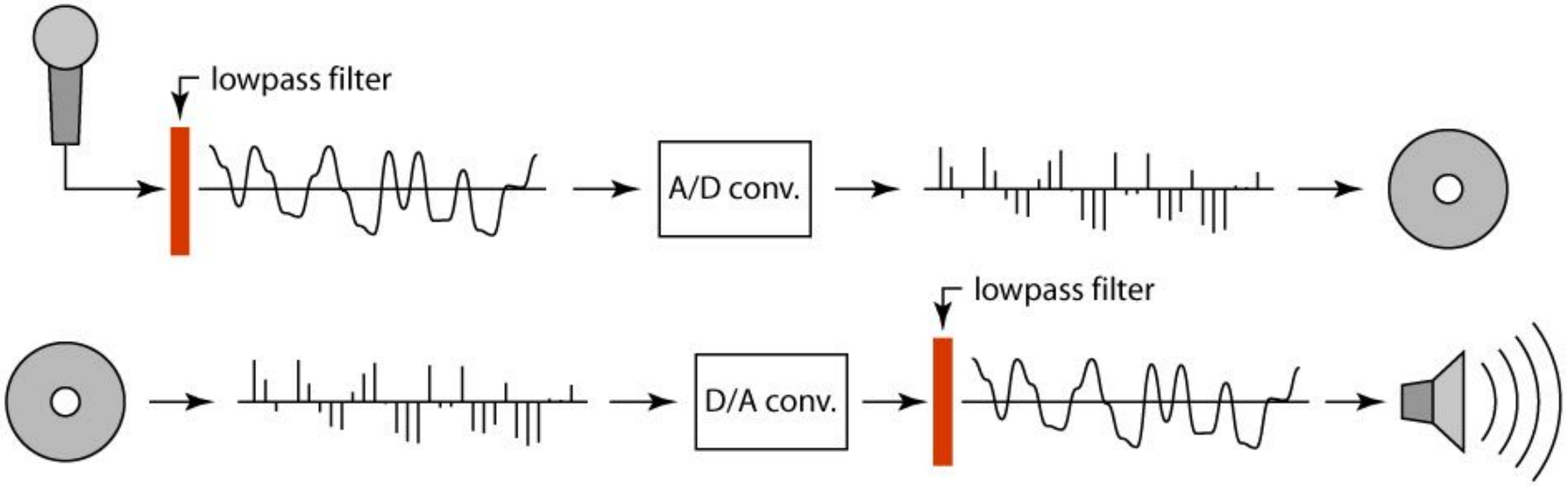
Preventing aliasing

- **Introduce lowpass filters:**
  - **remove high frequencies leaving only safe, low frequencies**
  - **choose lowest frequency in reconstruction (disambiguate)**



Preventing aliasing

- **Introduce lowpass filters:**
  - **remove high frequencies leaving only safe, low frequencies**
  - **choose lowest frequency in reconstruction (disambiguate)**



Antialiasing

Antialiasing

What can be done?

Sampling rate > 2 * max frequency in the image

1. **Raise sampling rate by** *oversampling*

   Sample at k times the resolution

   continuous signal: easy

   discrete signal: need to interpolate

2. **Lower the max frequency by** *prefiltering*

   Smooth the signal enough

   Works on discrete signals

Antialiasing

What can be done?

Sampling rate > 2 * max frequency in the image

1. **Raise sampling rate by** *oversamp*

   Sample at k times the resolution

   continuous signal: easy

   discrete signal: need to interpolate

2. **Lower the max frequency by** *pre*

   Smooth the signal enough

   Works on discrete signals

3. **Improve sampling quality with better sampling**

   Below Nyquist frequency is best case!

   Stratified sampling (jittering)

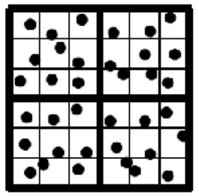
   Importance sampling

   Relies on domain knowledge

N := Nyquist Frequency, in cycles/sec

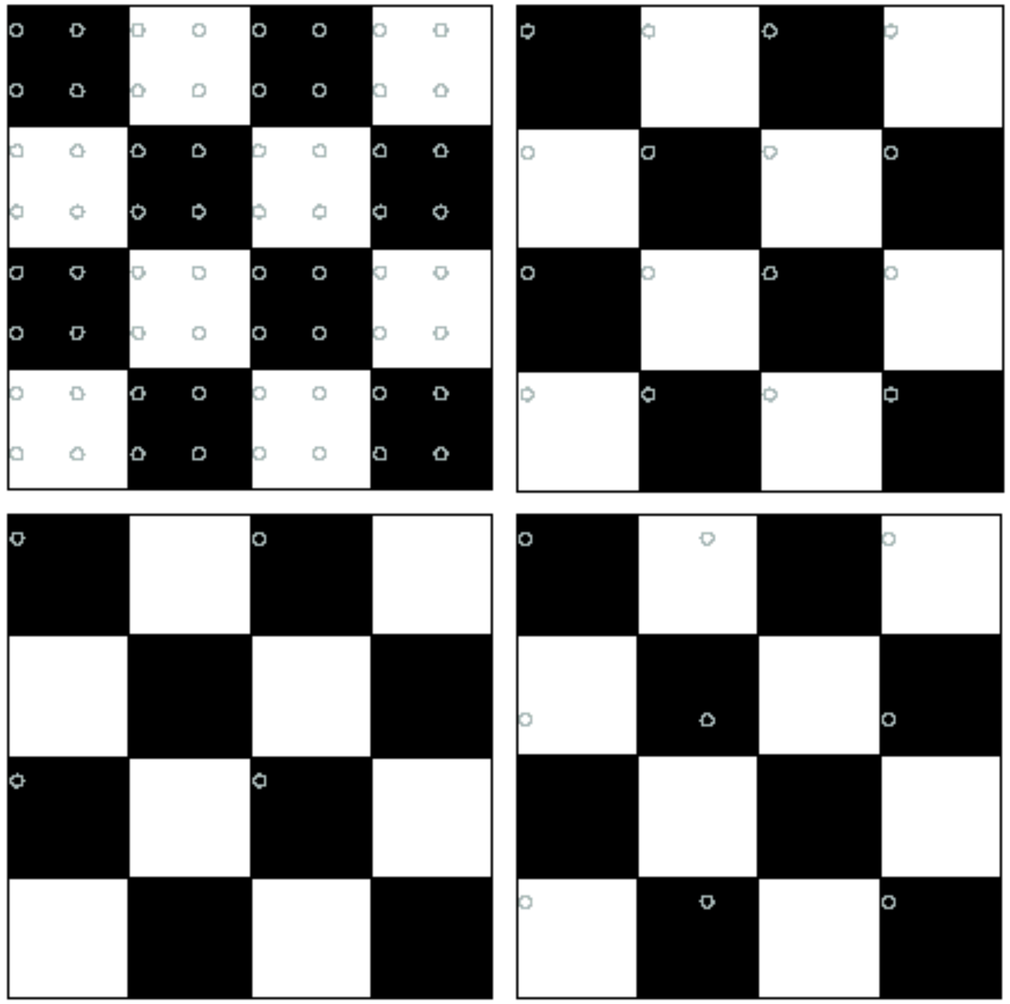
for a system with a fixed sampling rate of 2N samples/sec.

i.e., only frequencies < N will be reconstructed without aliasing.



jittered,
9 samples per pixel

Sampling



Good sampling:

Sample often or, Sample wisely

Bad sampling:

see aliasing in action!

Image Half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

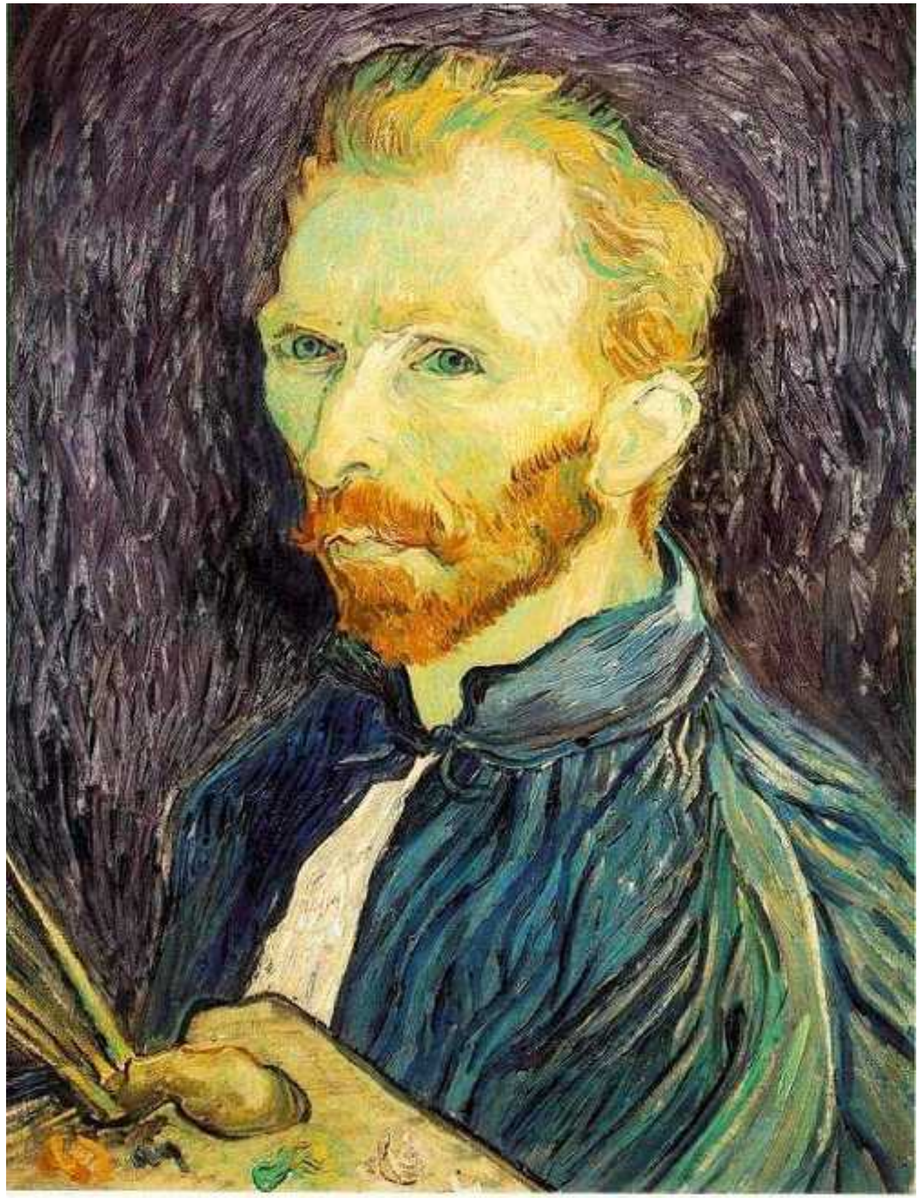
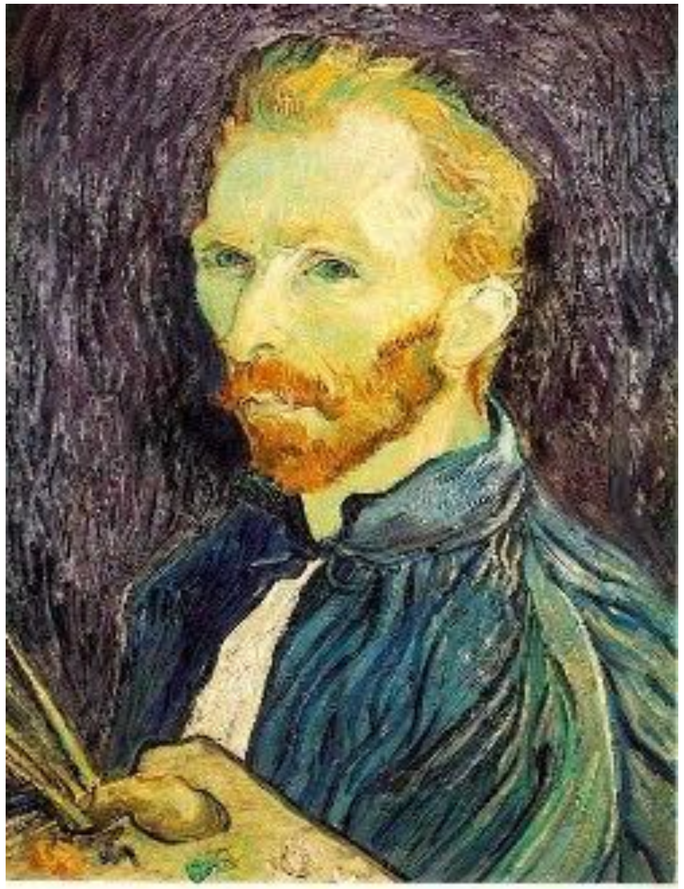
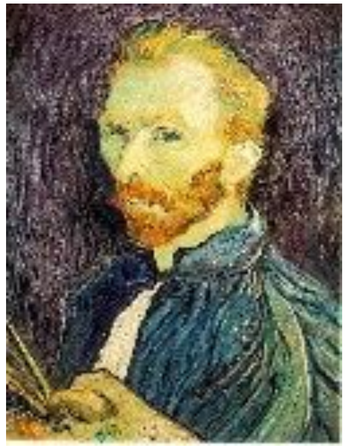


Image sub-sampling

Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*





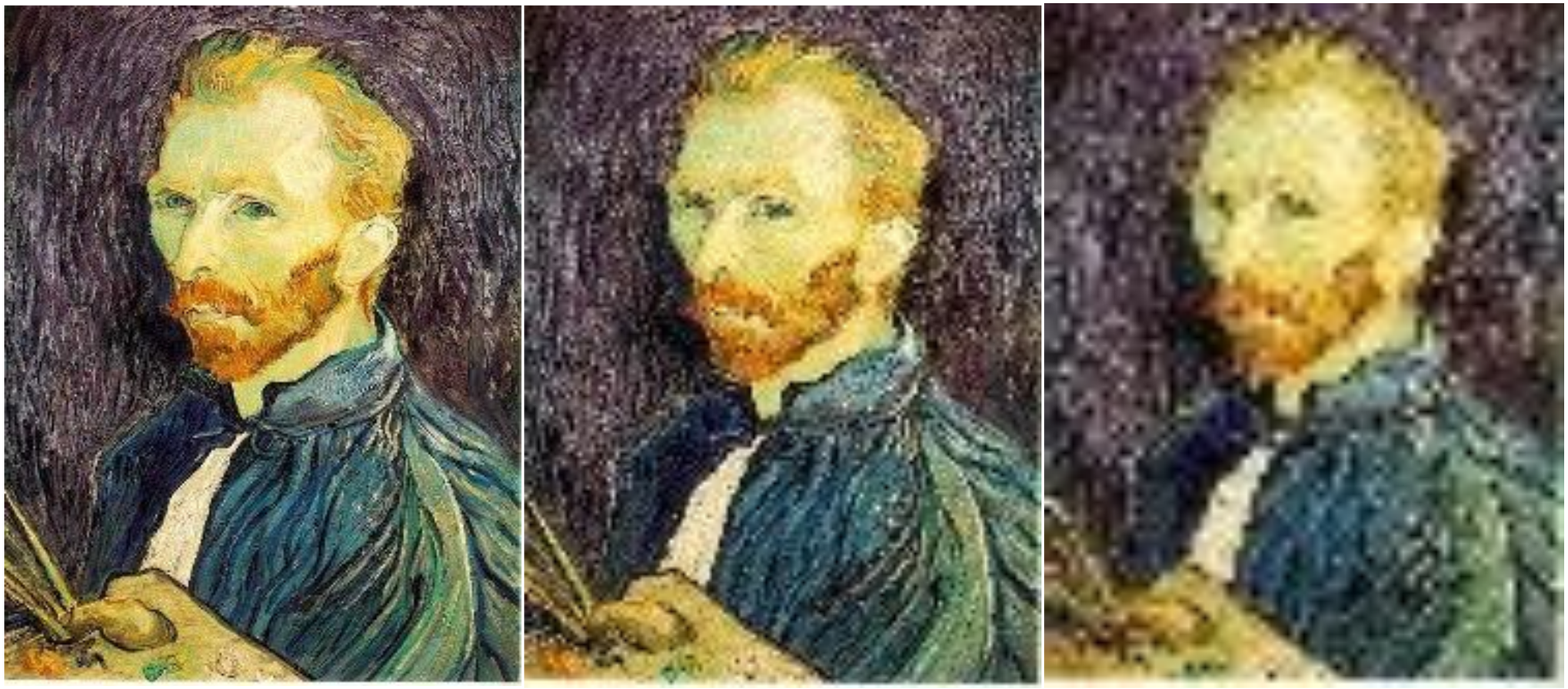
1/4

1/8

Slide by Steve Seitz

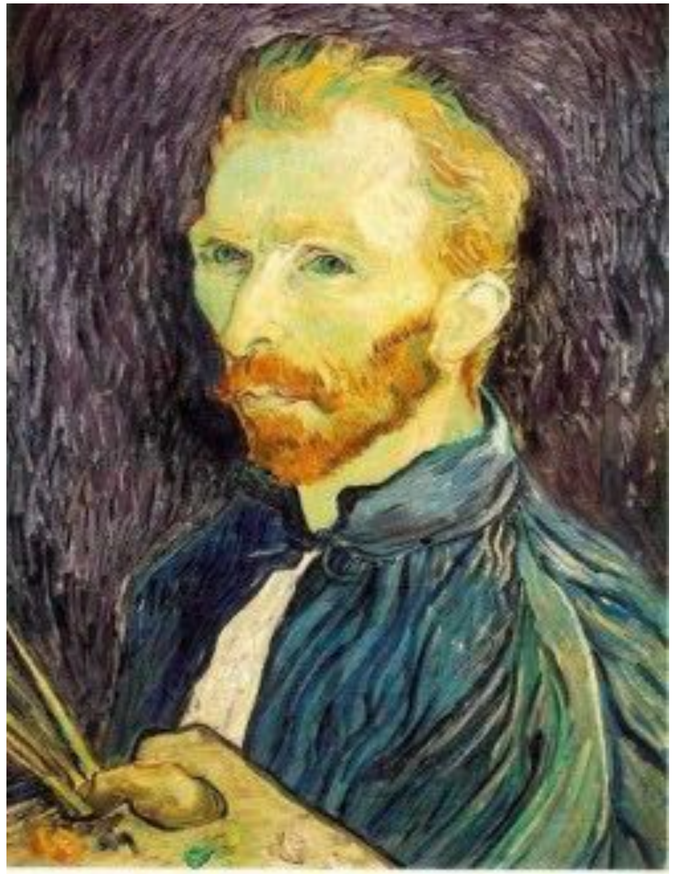
Image Sub-sampling

Aliasing! What do we do?

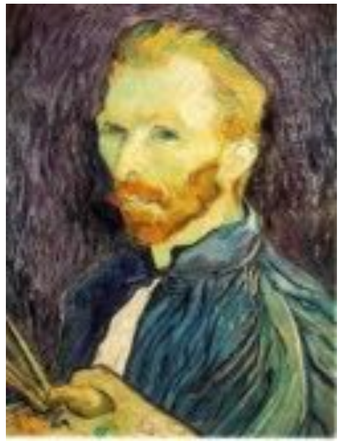


1/2 1/4 (2x zoom) 1/8 (4x zoom)

Gaussian (lowpass) Pre-filtering



Gaussian 1/2



G 1/4

G 1/8

Solution: filter the image, *then* subsample

Filter size should double for each ½ size reduction (why?)

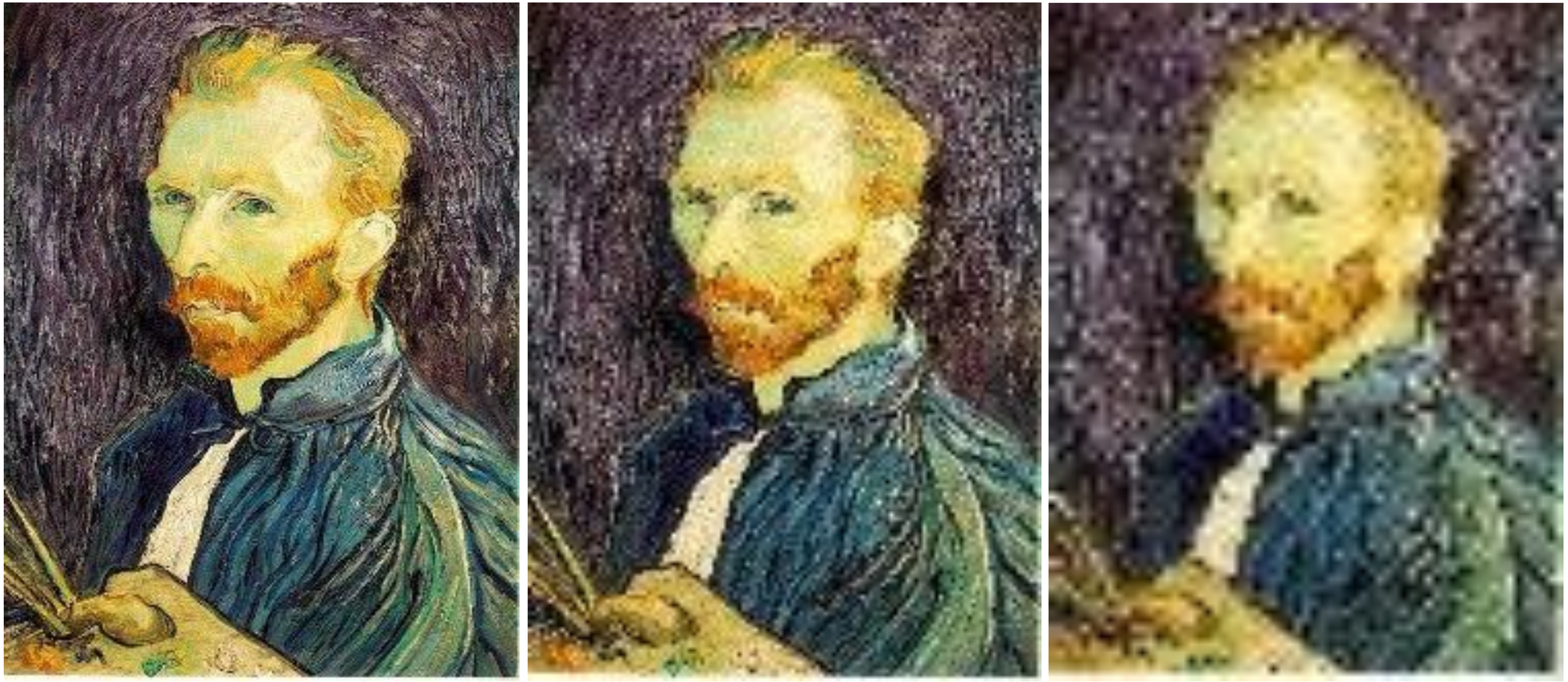
Subsampling with Gaussian Pre-filtering

Gaussian 1/2 G 1/4 G 1/8

Solution: filter the image, *then* subsample

Filter size should double for each ½ size reduction. Why? How can we speed this up?

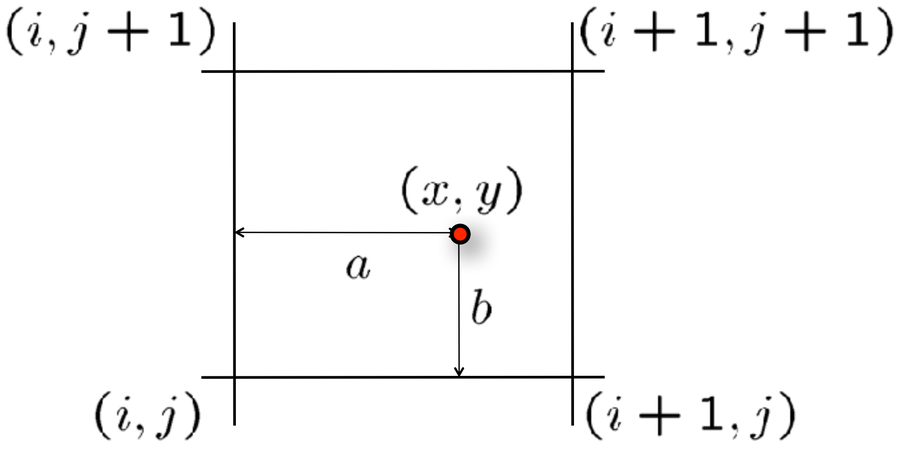
Compare with Just Subsampling



1/2 1/4 (2x zoom) 1/8 (4x zoom)

Bilinear Interpolation

Sampling at *f(x,y)*:



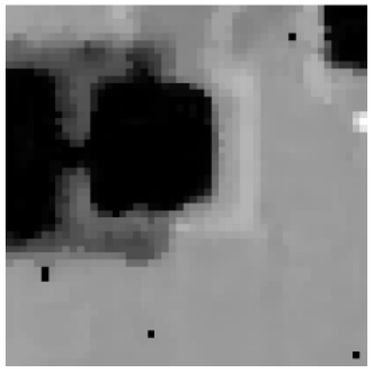
$$
\begin{aligned}
f(x,y) = \; & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Bilinear Interpolation

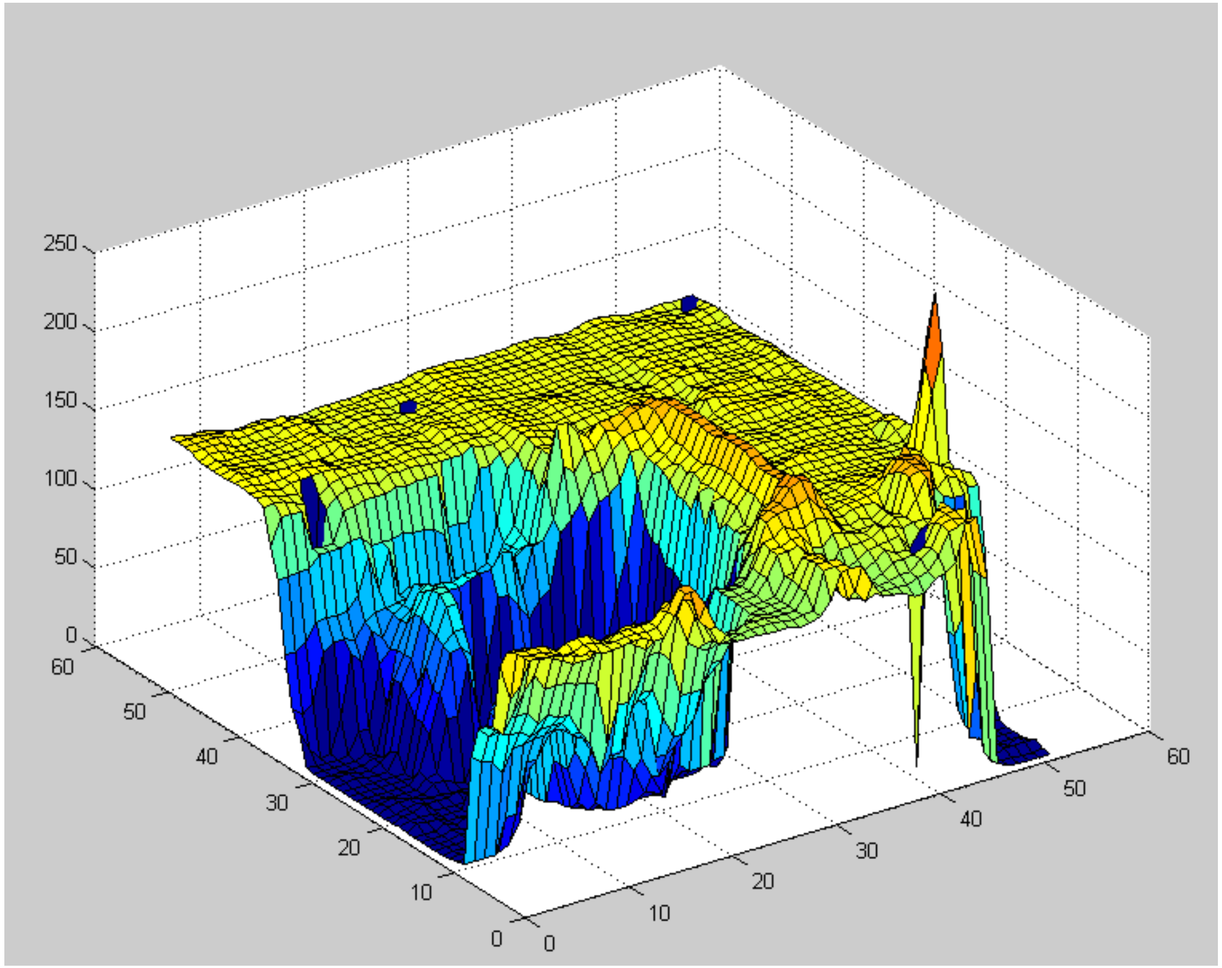
Sampling at *f(x,y):*

Image == Heightfield

I



z = *I*'s intensity



$$F(i) = Hl(i - \hat{i})\, Fl(i) + Hr(i - \hat{i})\, Fr(i)$$

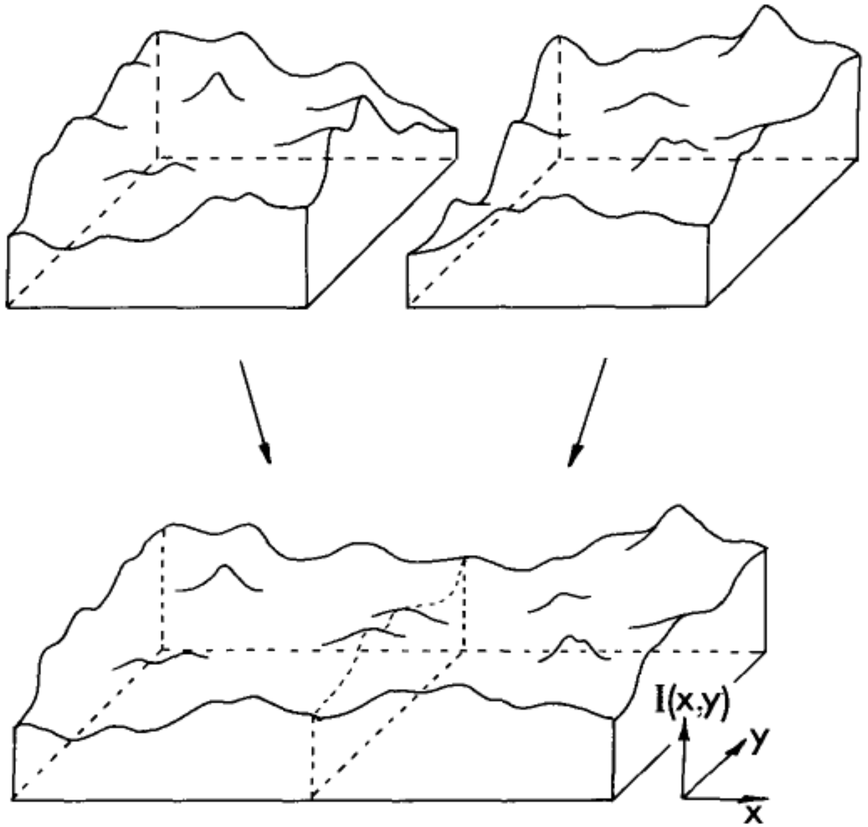


Fig. 1. A pair of images may be represented as a pair of surfaces above the (x, y) plane. The problem of image splining is to join these surfaces with a smooth seam, with as little distortion of each surface as possible.

How does one Normally Blend?

How does one Normally Blend?

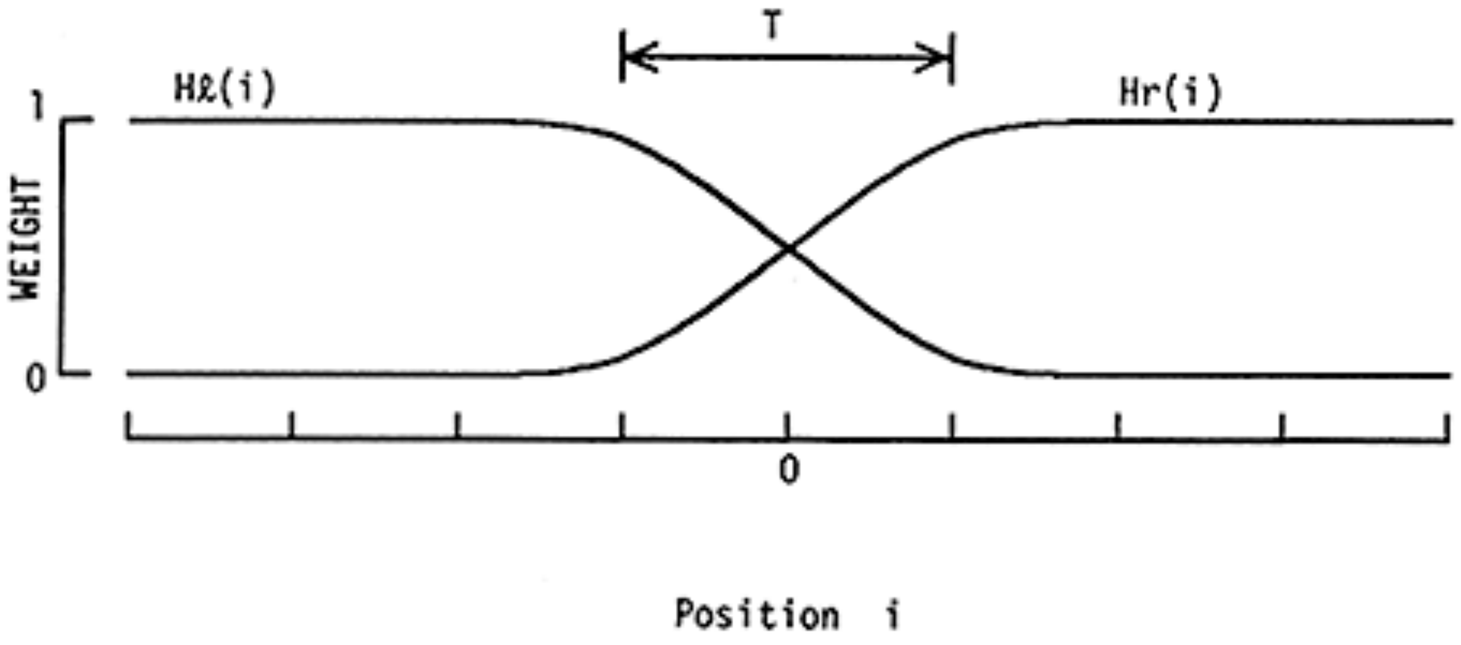
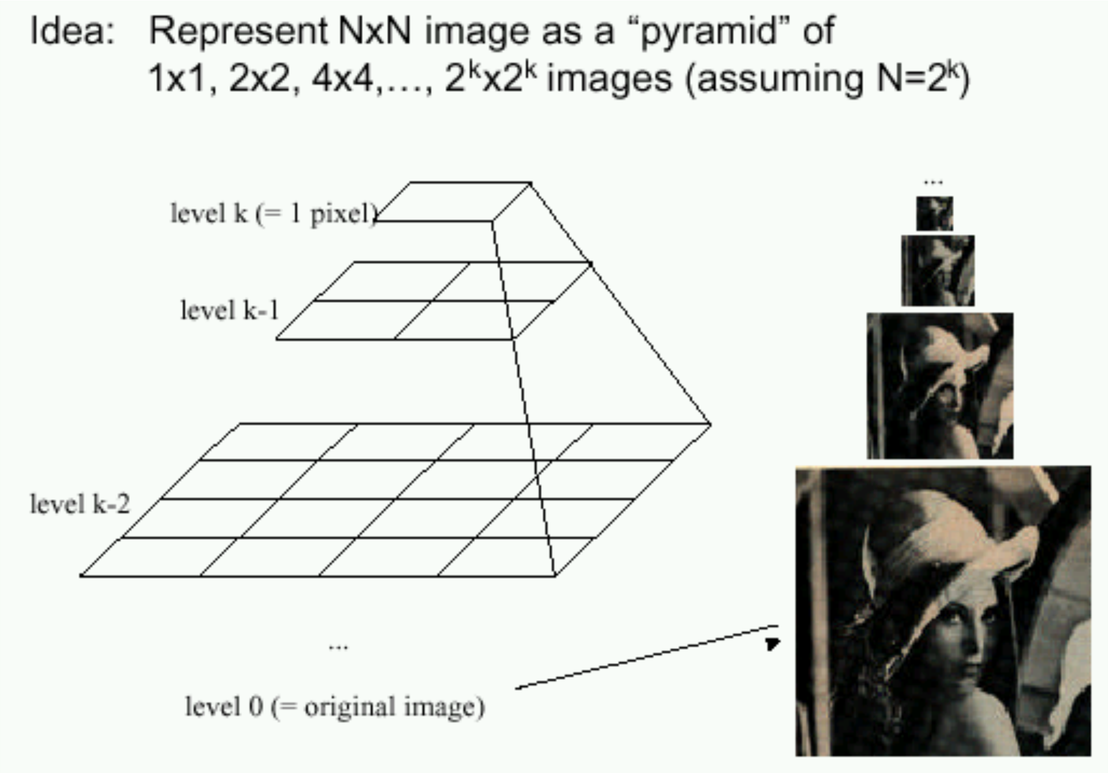


Image Pyramids



Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

In computer graphics, a *mip map* [Williams, 1983]

A precursor to *wavelet transform*

First introduced for compression purposes

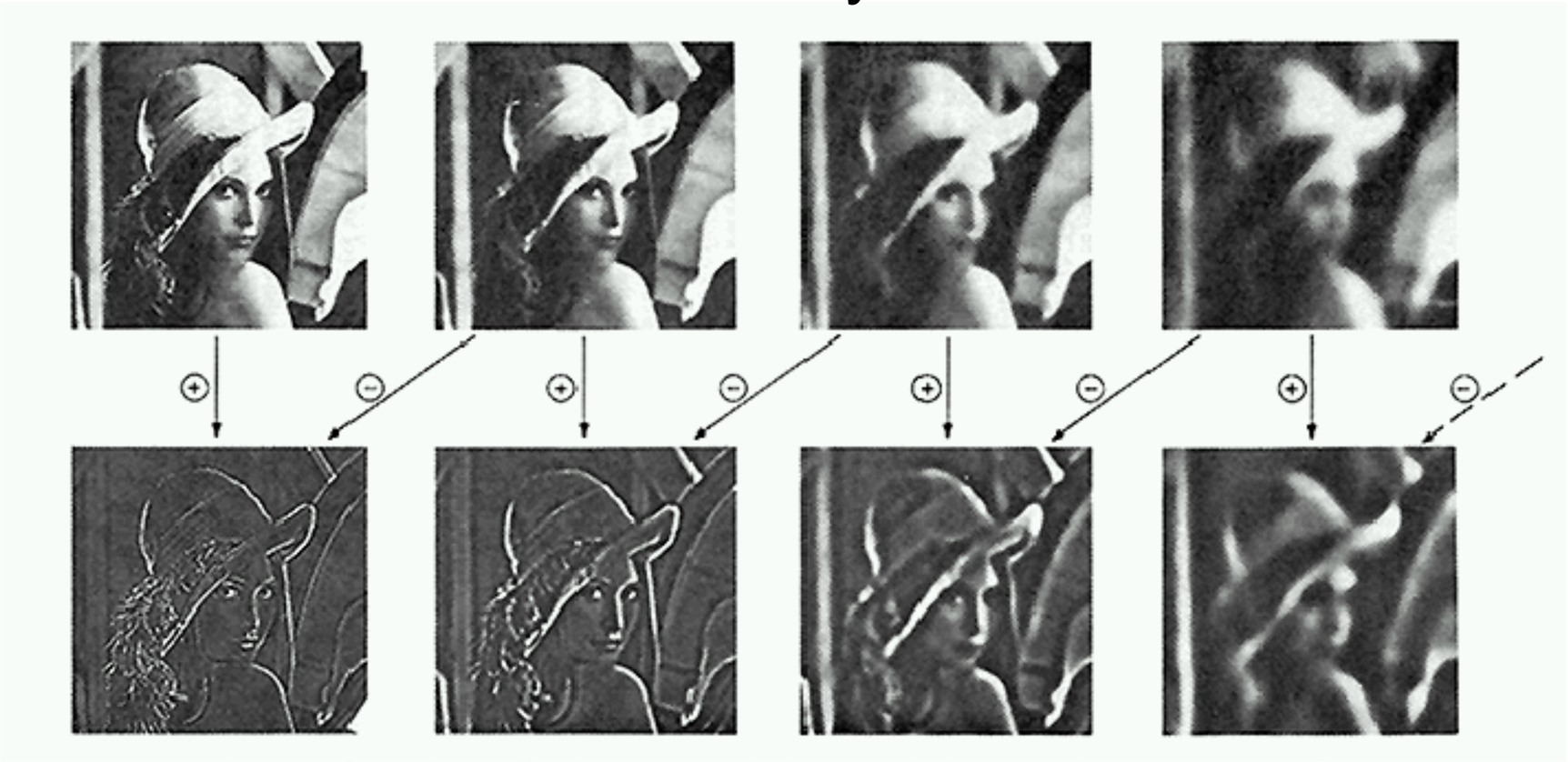
512 256 128 64 32 16 8

A row in the big images is a hair on the zebra's nose; in smaller images, a stripe; in the smallest, the animal's nose

Figure from David Forsyth

Laplacian Pyramid

Gaussian Pyramid



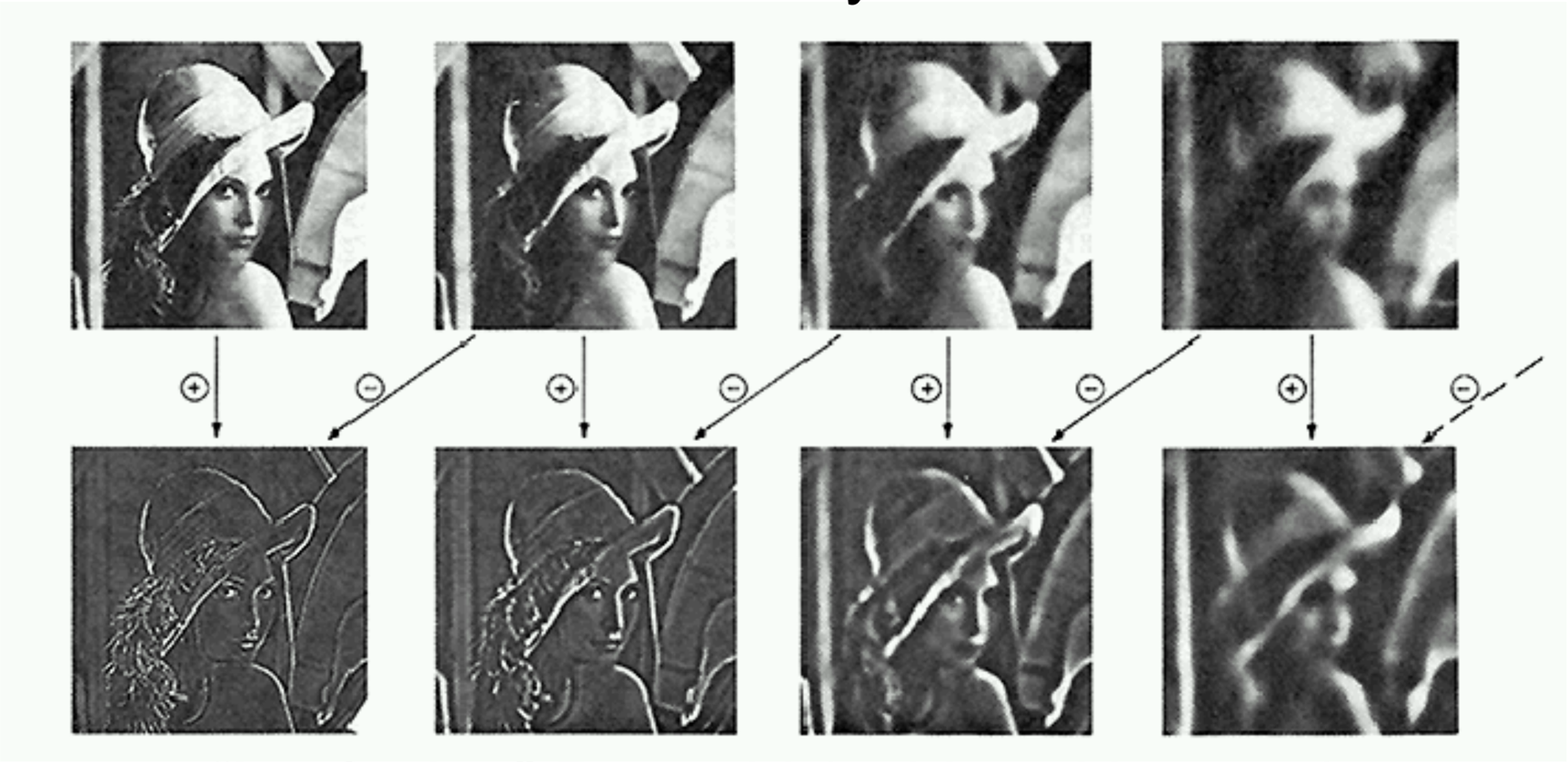
“Laplacian” Pyramid (subband images)

Created from Gaussian pyramid by subtraction

Laplacian Pyramid

Remember Unsharp Mask?

Gaussian Pyramid



“Laplacian” Pyramid (subband images)

Created from Gaussian pyramid by subtraction

Laplacian Pyramid

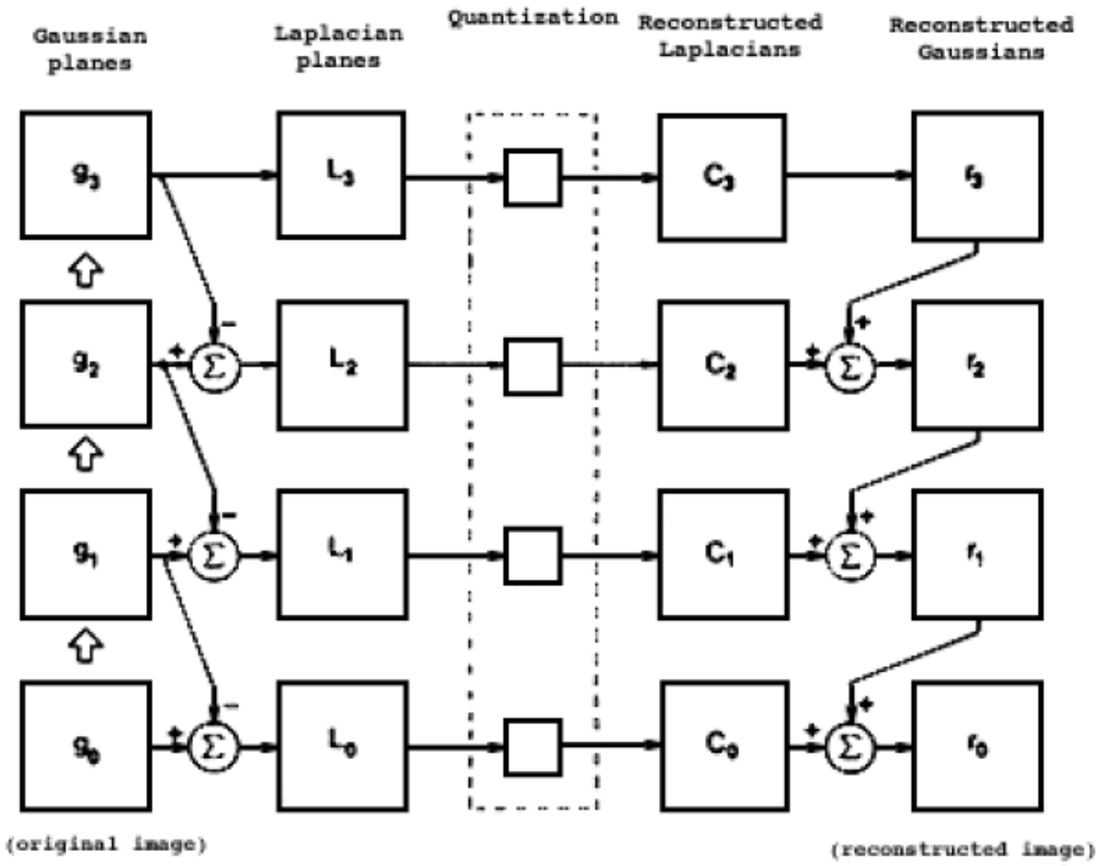
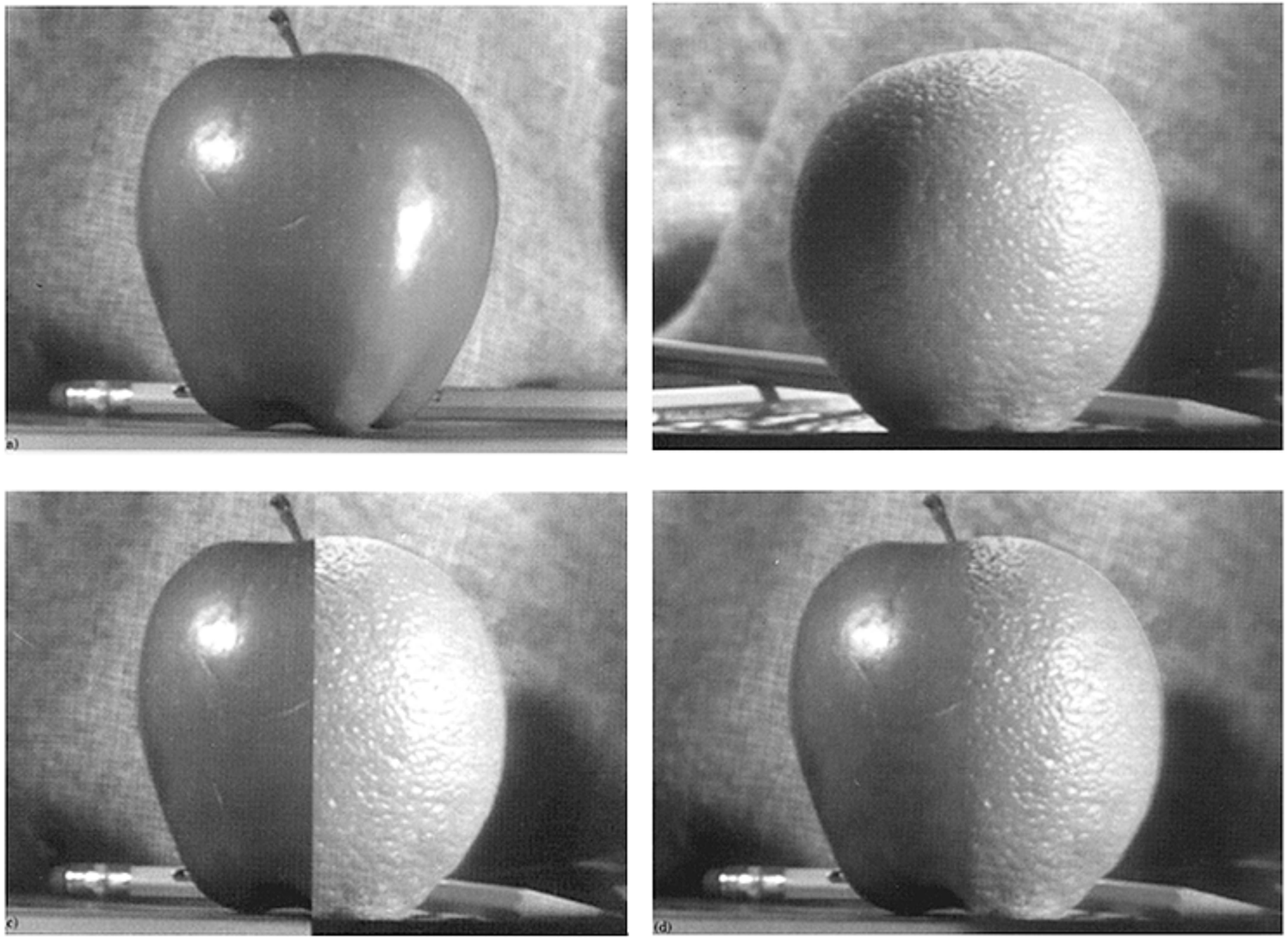


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels $g_1, g_2, \ldots$ through repeated local averaging. Levels of the Laplacian pyramid $L_0, L_1, \ldots$ are then computed as the differences between adjacent Gaussian levels. Laplacian pyramid elements are quantized to yield the Laplacian pyramid code C_0, $C_1, C_2, \ldots$. Finally, a reconstructed image r_0 is generated by summing levels of the code pyramid.

Fun with Image Pyramids: Blending

A Multiresolution Spline With Application to Image Mosaics
Burt & Adelson '83



Photos + implementation by Rob Orr

Photos + implementation by Rob Orr

Photos + implementation by Rob Orr

Edge “Detection”

Edge “Detection”

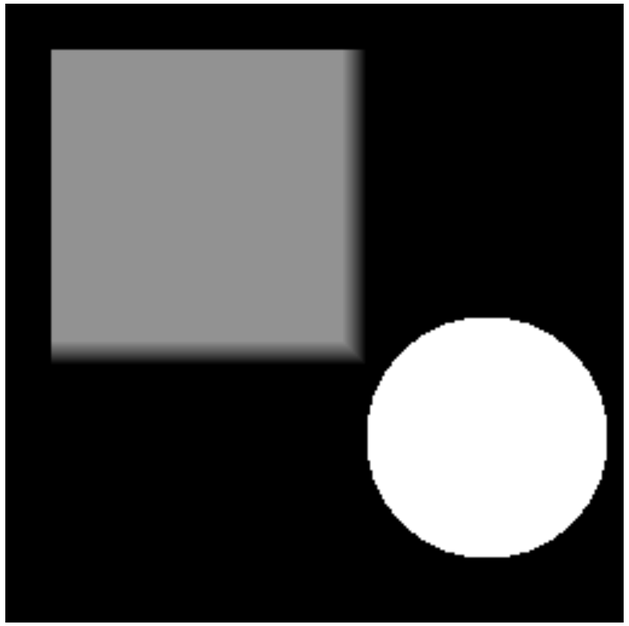
Key Steps

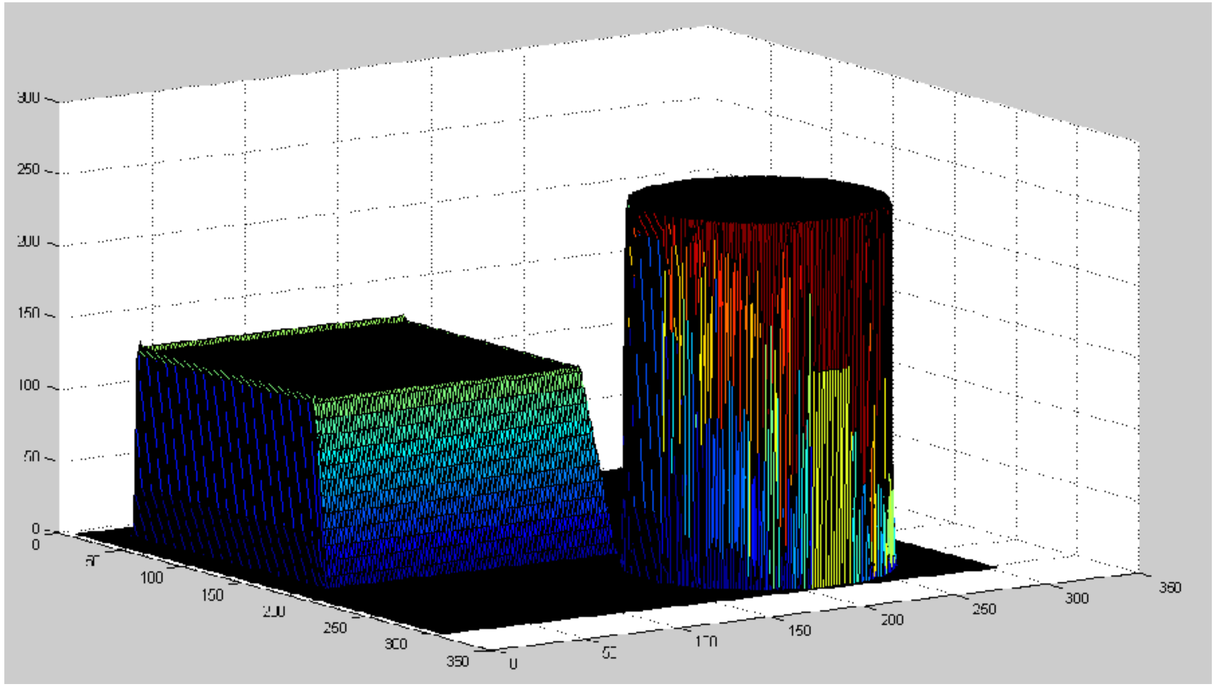
- Image smoothing

- Detect edge points (candidates)

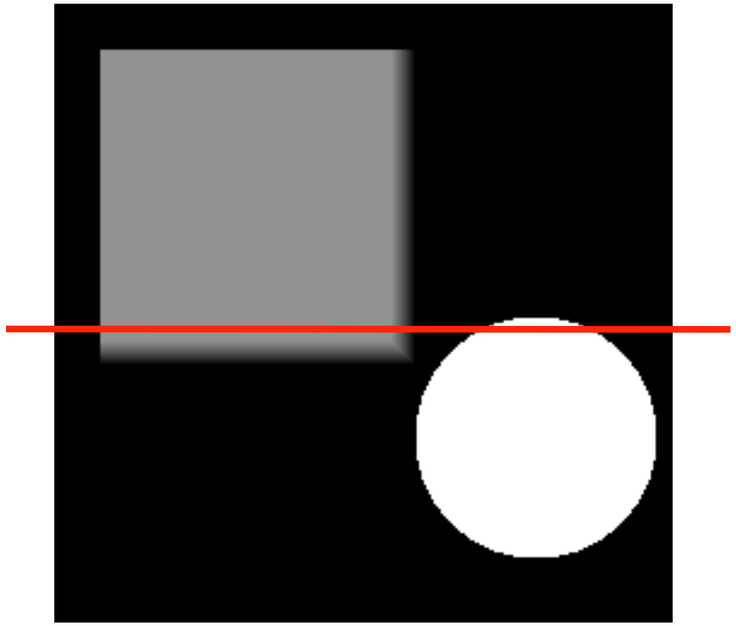
- Edge localization
  - non-mamixa suppression
  - double thresholding

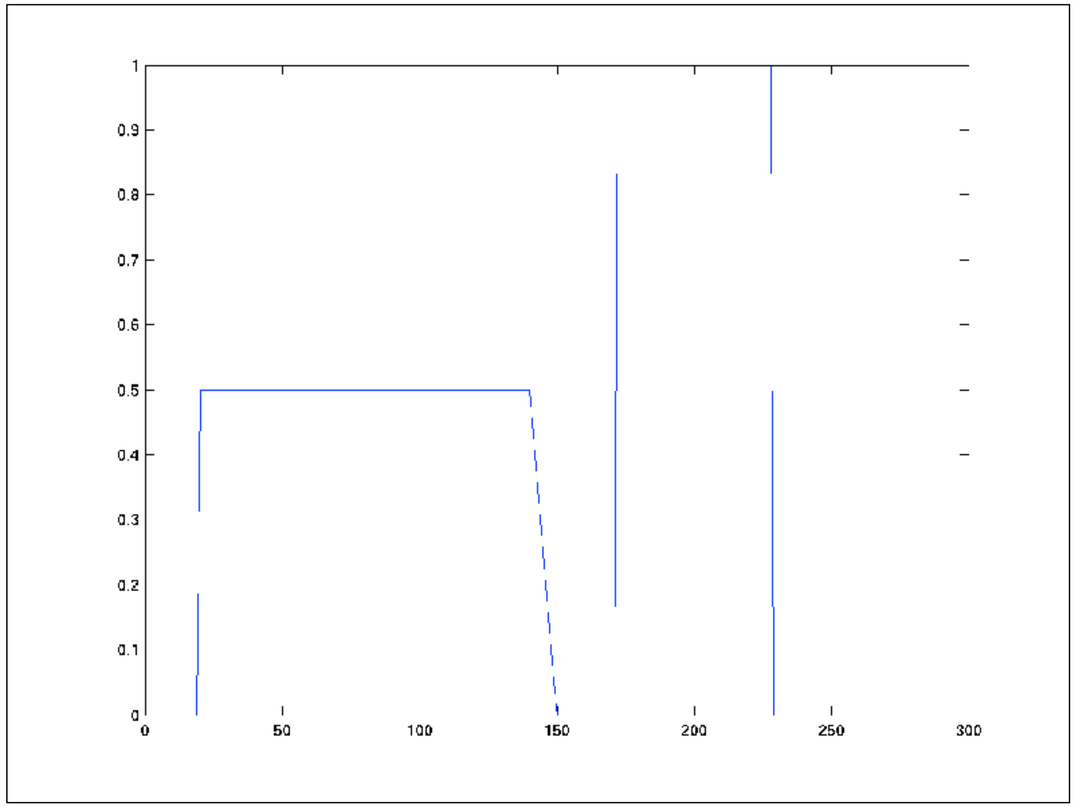
Step and Ramp Edges





Step and Ramp Edges





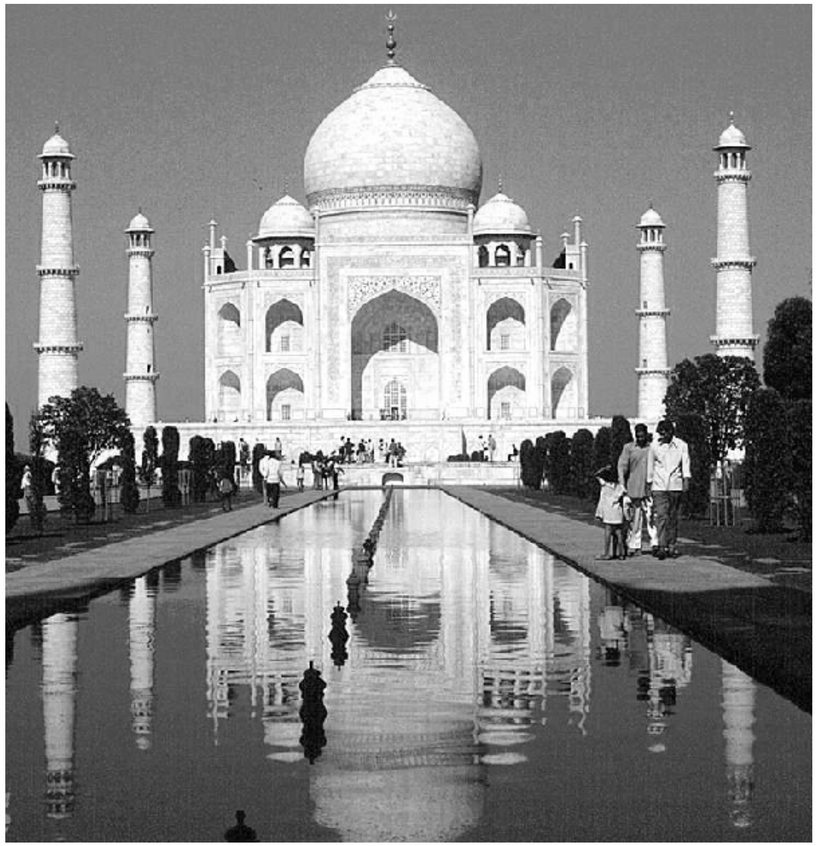
What is An Edge?

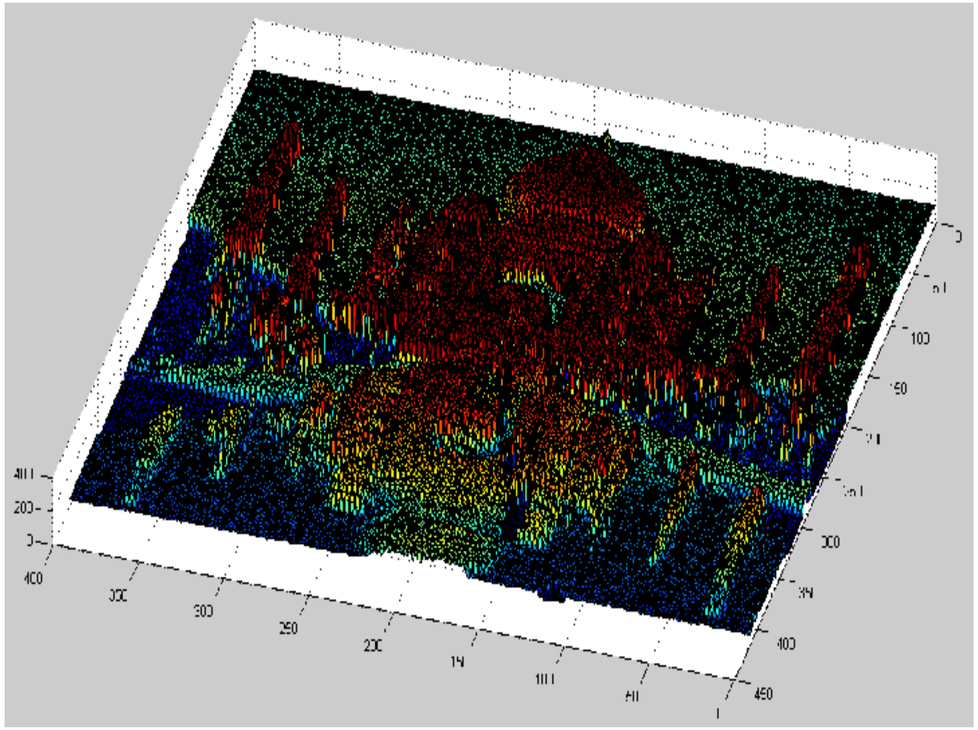
- Boundaries between regions in images:
  - Material change
  - Occlusion boundary
  - Crease boundaries
  - Shadow boundaries

- Sharp changes of gray level: Texture

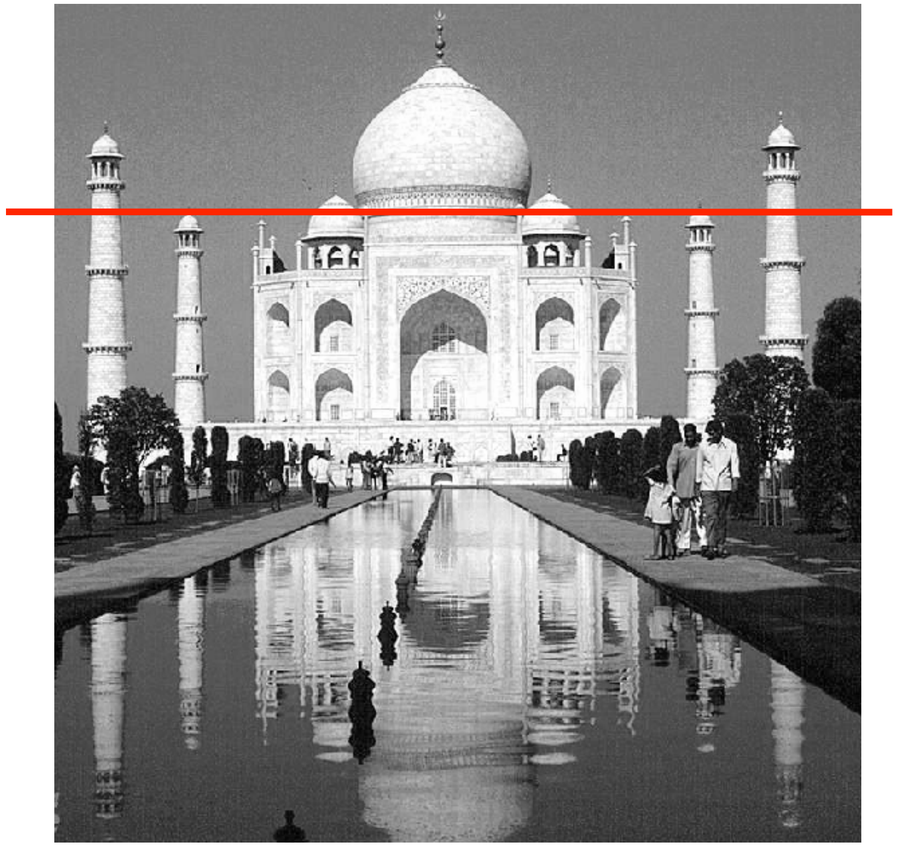
- (Motion boundaries)

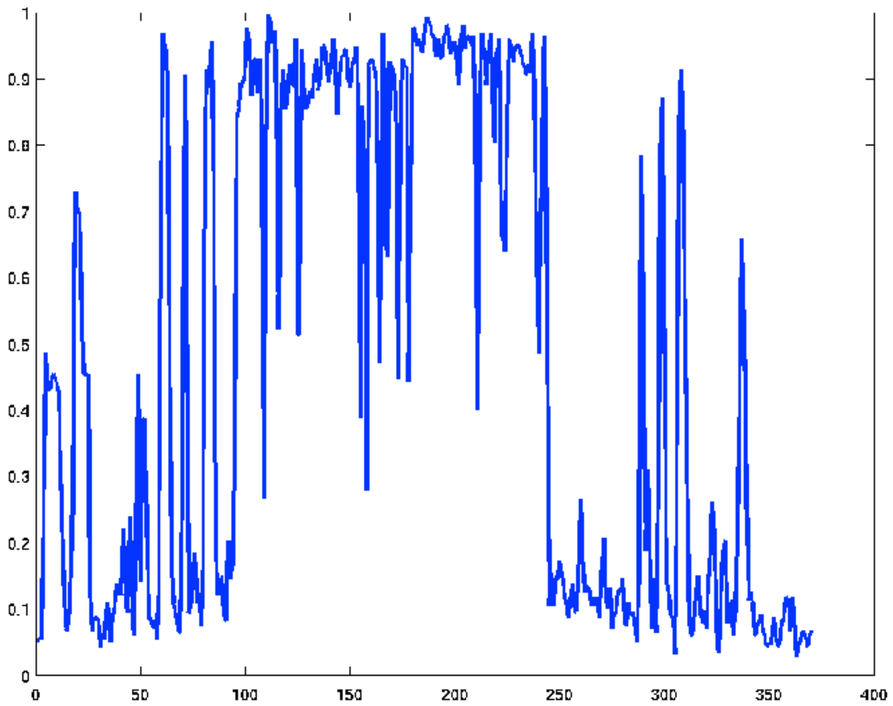
Real Edges



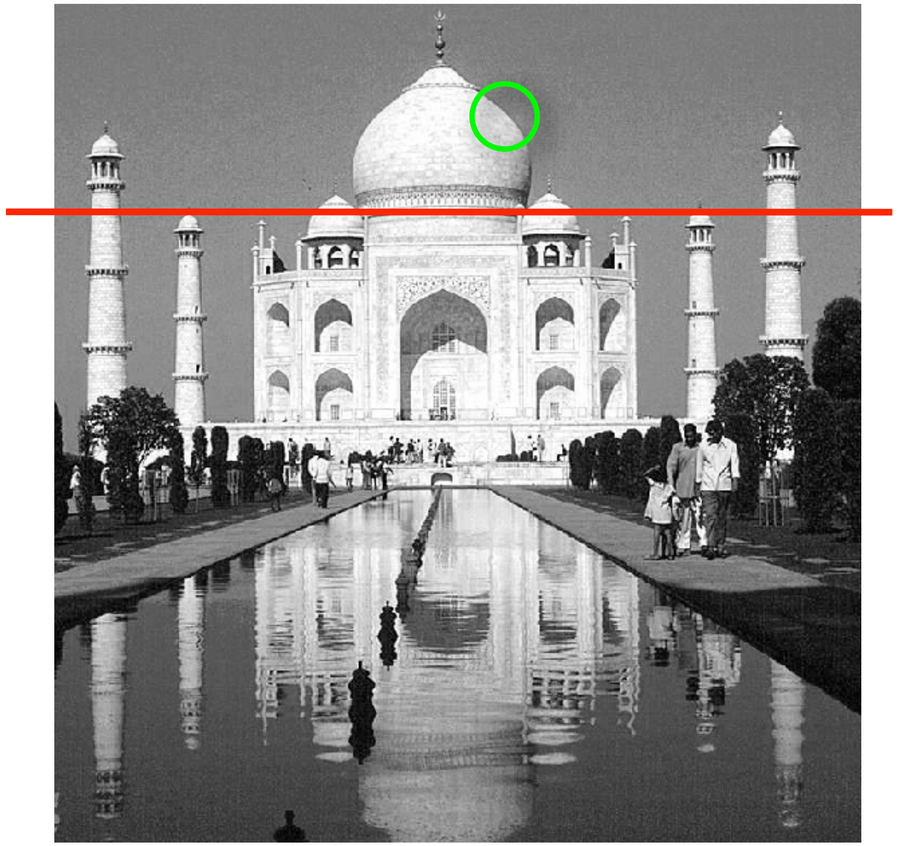


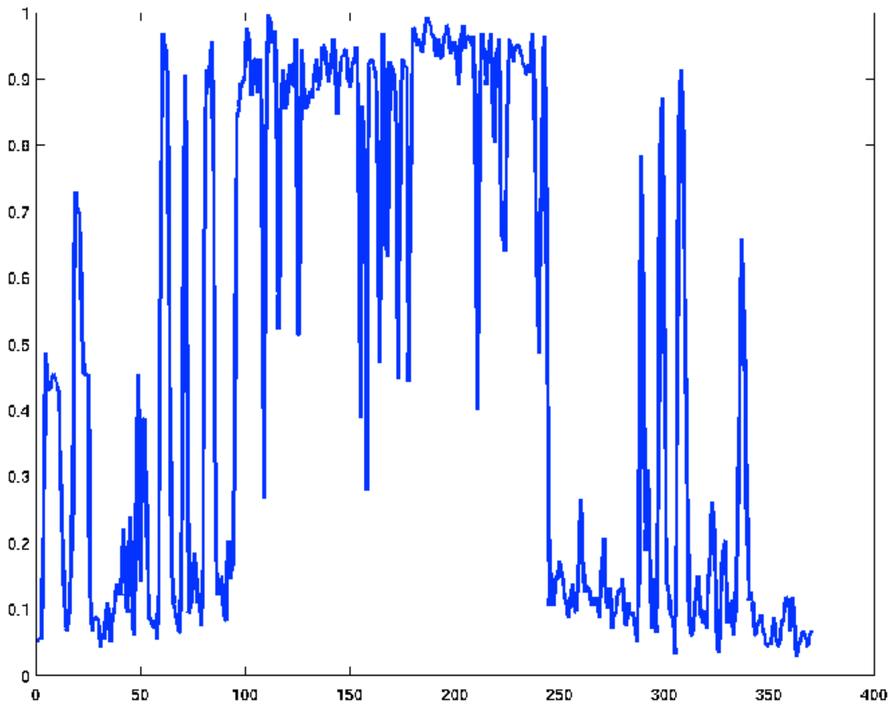
Real Edges



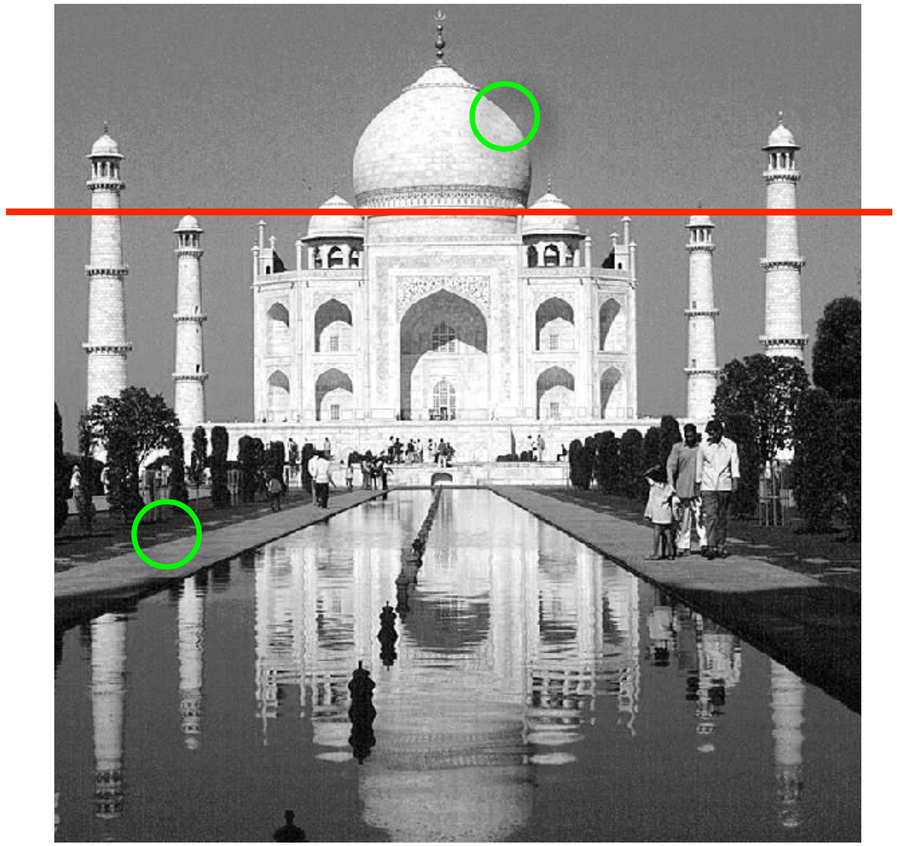


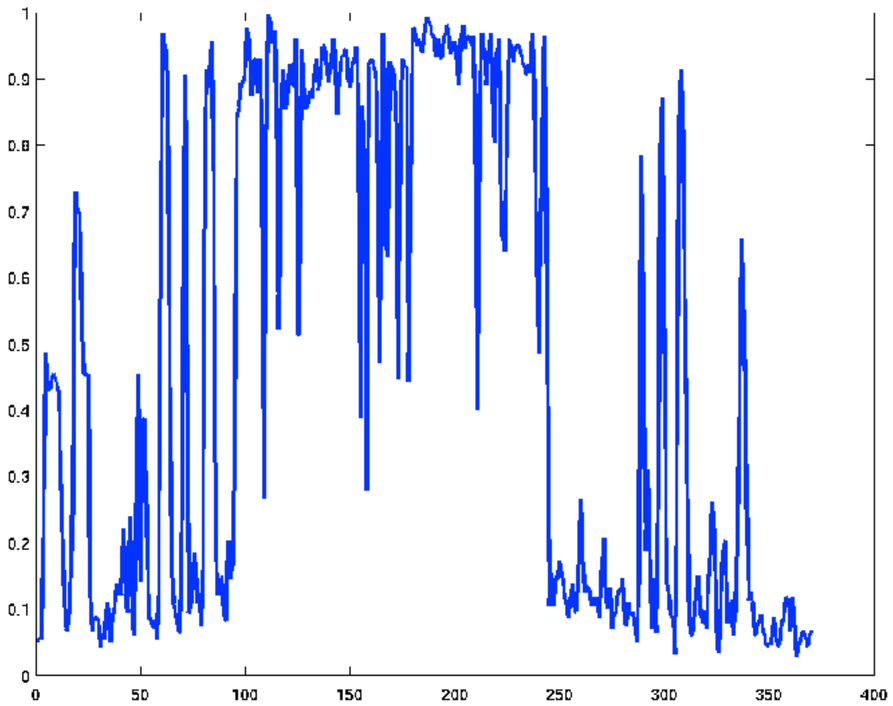
Real Edges



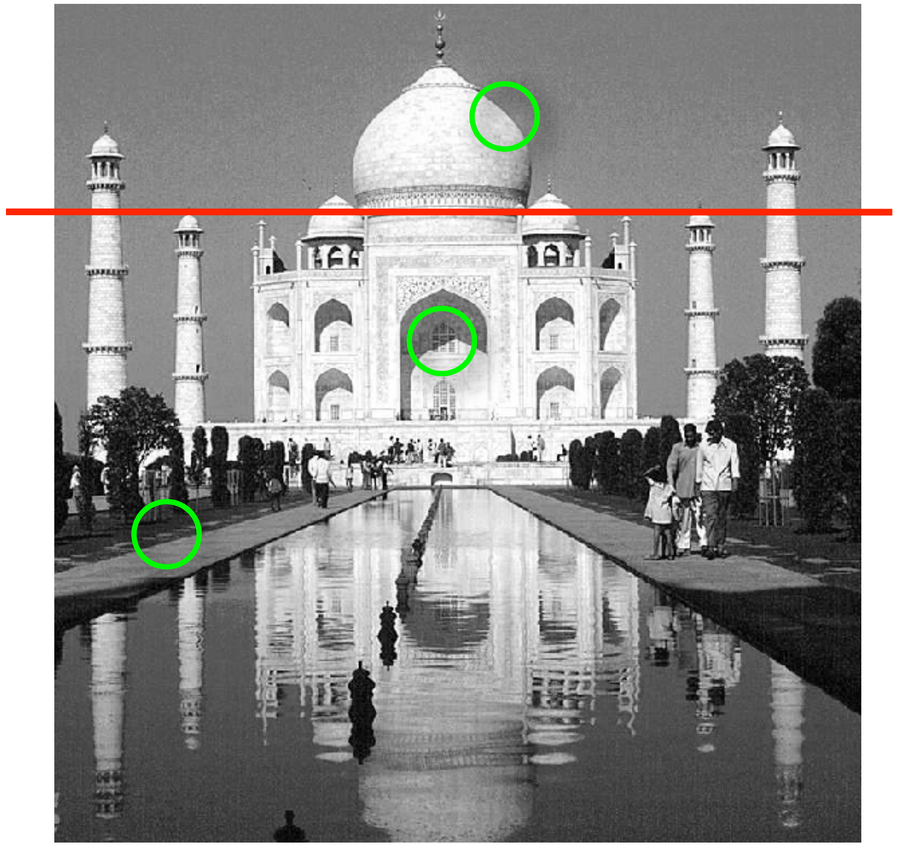


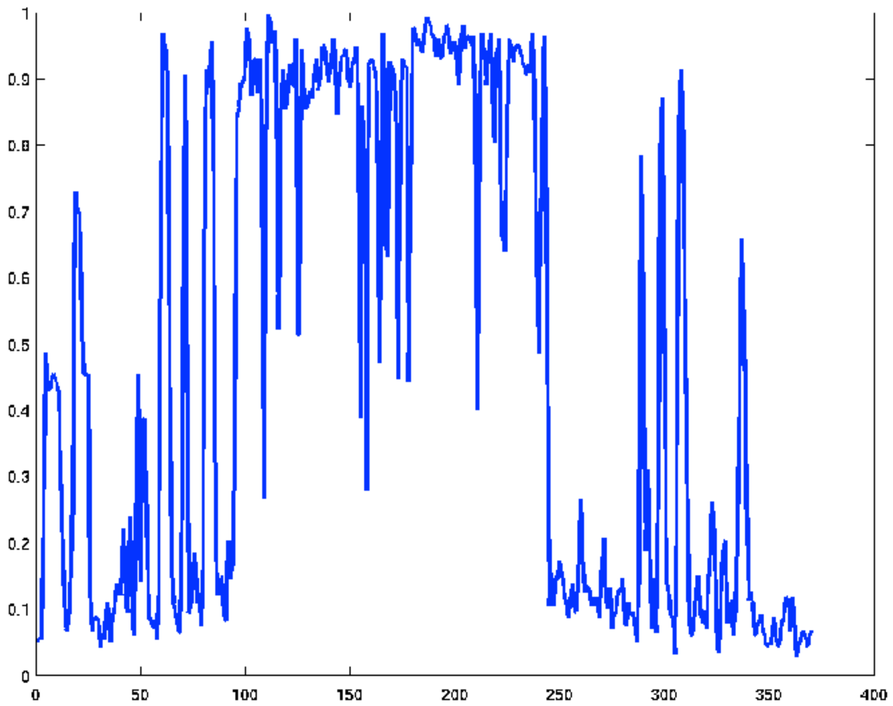
Real Edges



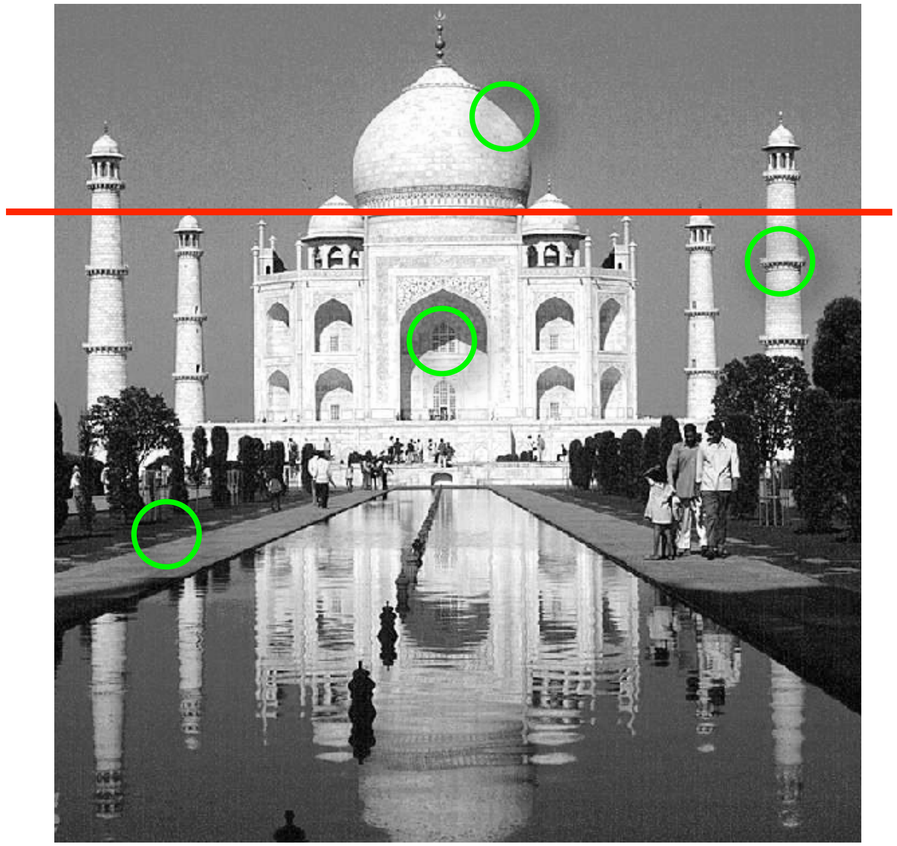


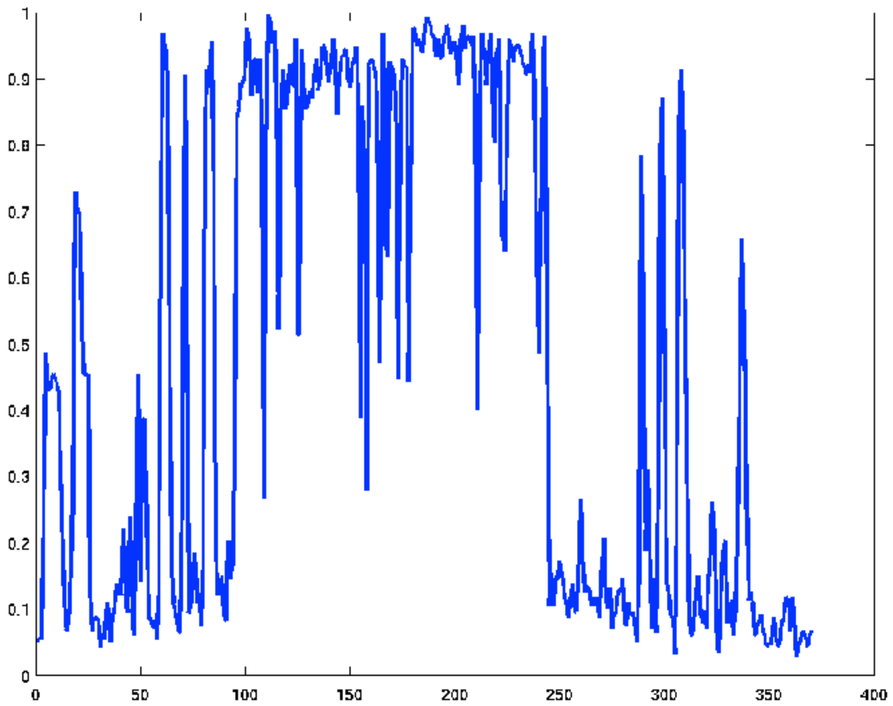
Real Edges



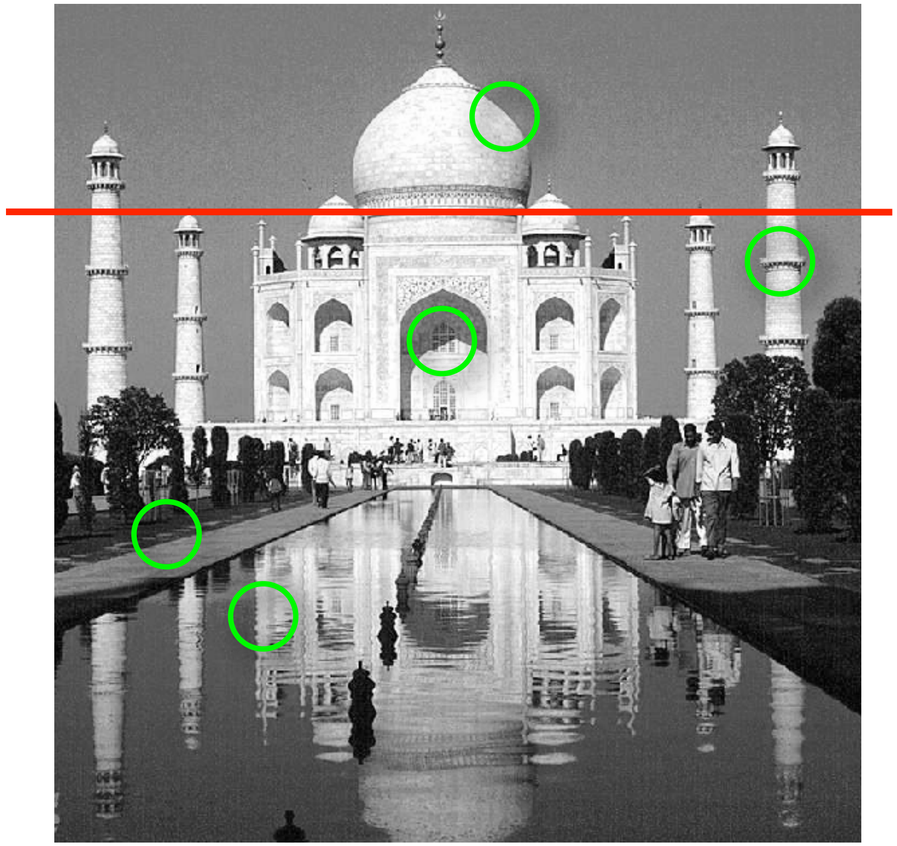


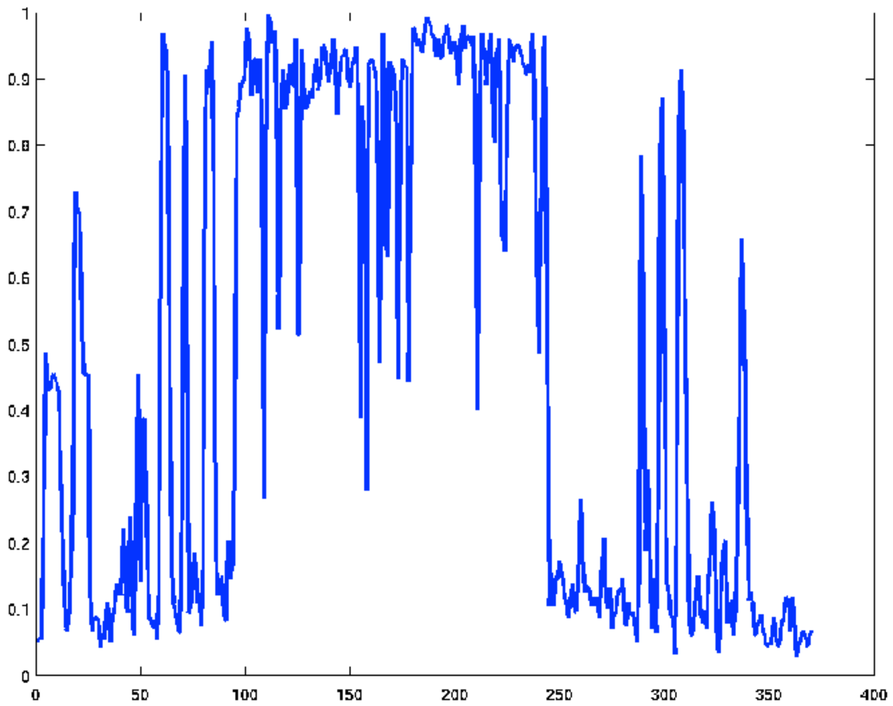
Real Edges





Real Edges



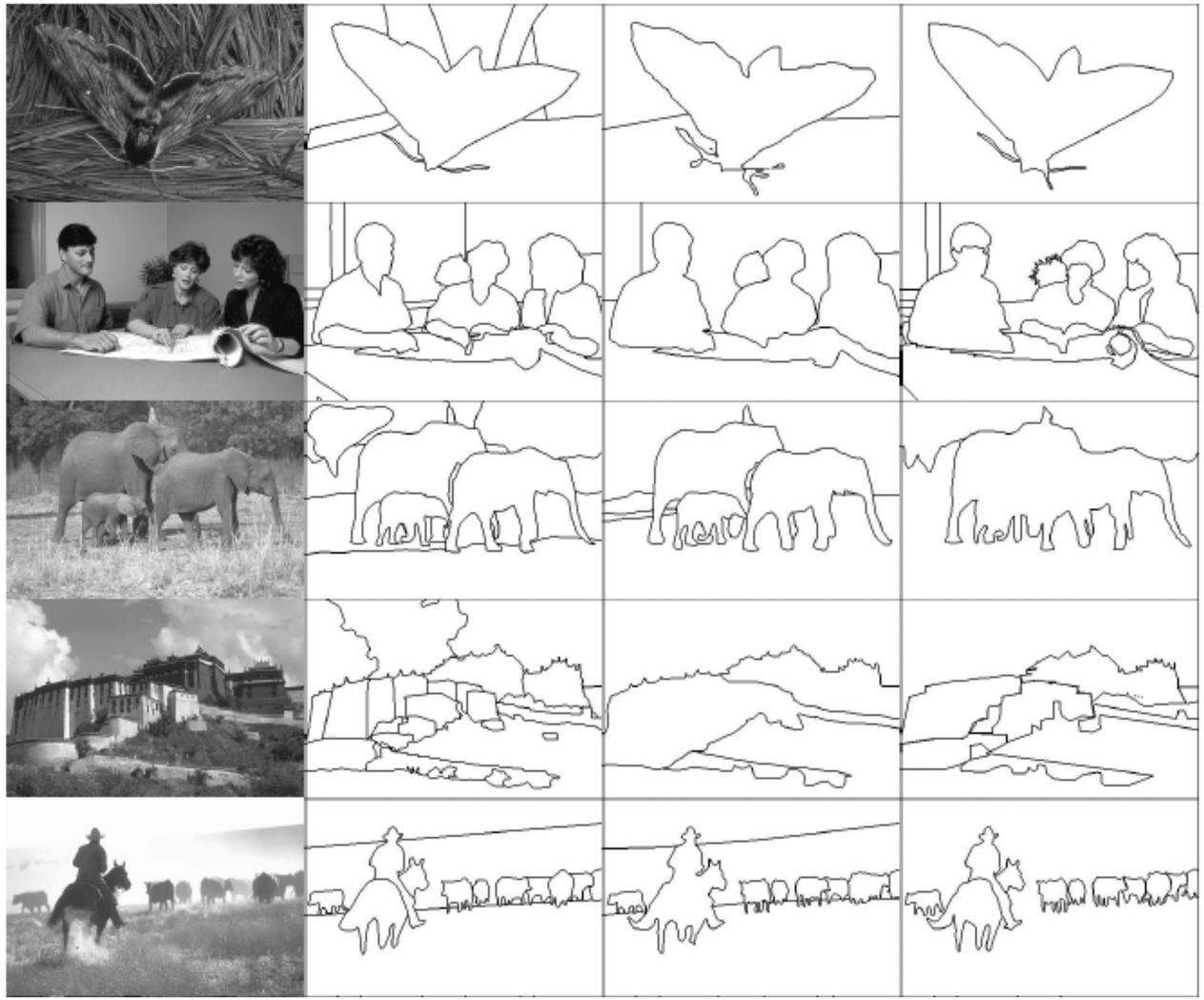


Applications

- Segmentation
- Stereo matching
- Theory underlies many more sophisticated image processing algorithms

Humans Disagree…

A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics
Martin Fowlkes Tal Malik, ICCV01



Edge Detection

- A wide range of techniques.

- Three steps to perform
  - Noise reduction
  - Edge enhancement
  - Edge localization

- For today's purposes:
  output a binary image with edge pixels marked

Simple Edge Detector

- **Minimal noise reduction**
- **Crude localization**
- **Compute image gradients**

$$g_x(x, y) = f(x+1, y) - f(x-1, y)$$
$$g_y(x, y) = f(x, y+1) - f(x, y-1)$$

Simple: Gradient Kernels

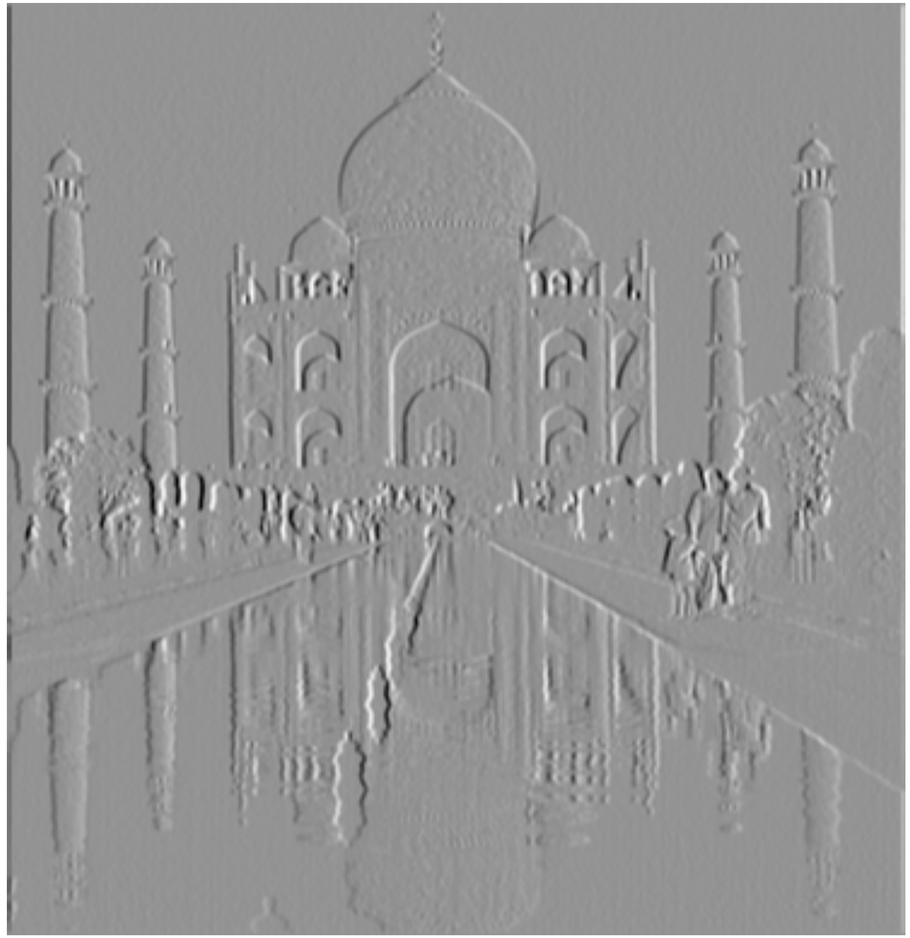
- **Prewitt kernels**

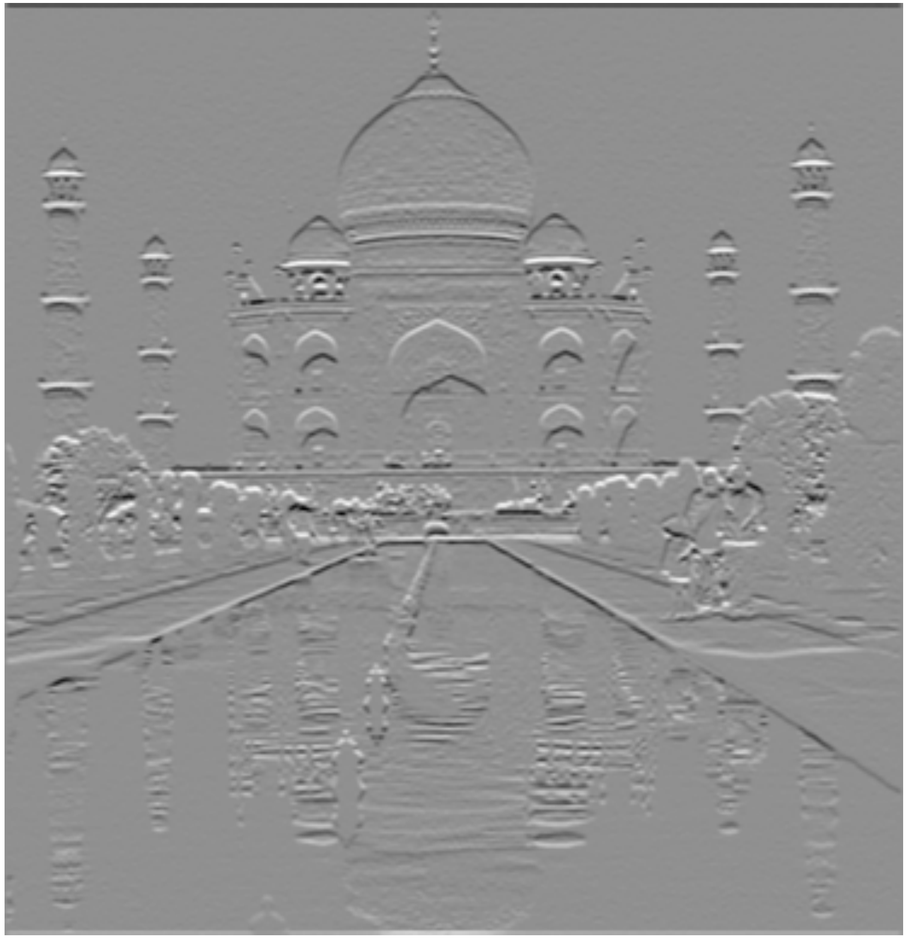
$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Sobel kernels**

$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Image Gradients (Sobel)





Simple: Gradient Vector

- **Gradient vector**

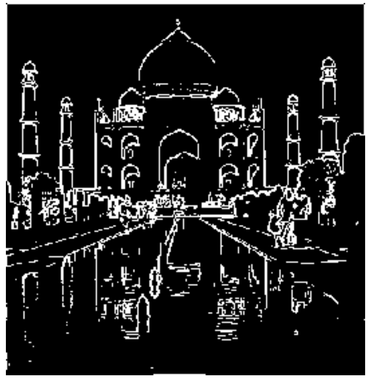
$$\mathbf{g}(x,y) = \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} (k_x \star f)(x,y) \\ (k_y \star f)(x,y) \end{bmatrix}$$

- **Gradient magnitude and direction are:**

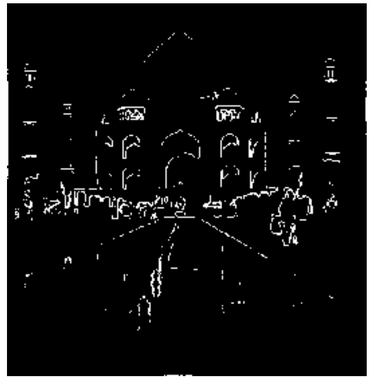
$$\|\mathbf{g}\| = \sqrt{g_x^2 + g_y^2}$$

$$\arg(\mathbf{g}) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Simple: Edge Map



T=0.25



T=0.5

T=0.75

One more pair of kernels

Robert's Cross Operator:

$$k_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad k_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

more for historic importance

Canny Edge Detector

- Combine noise reduction and edge enhancement.

- Two-step edge localization
  - Non-maximal suppression
  - Hysteresis thresholding

Canny Edge Detector

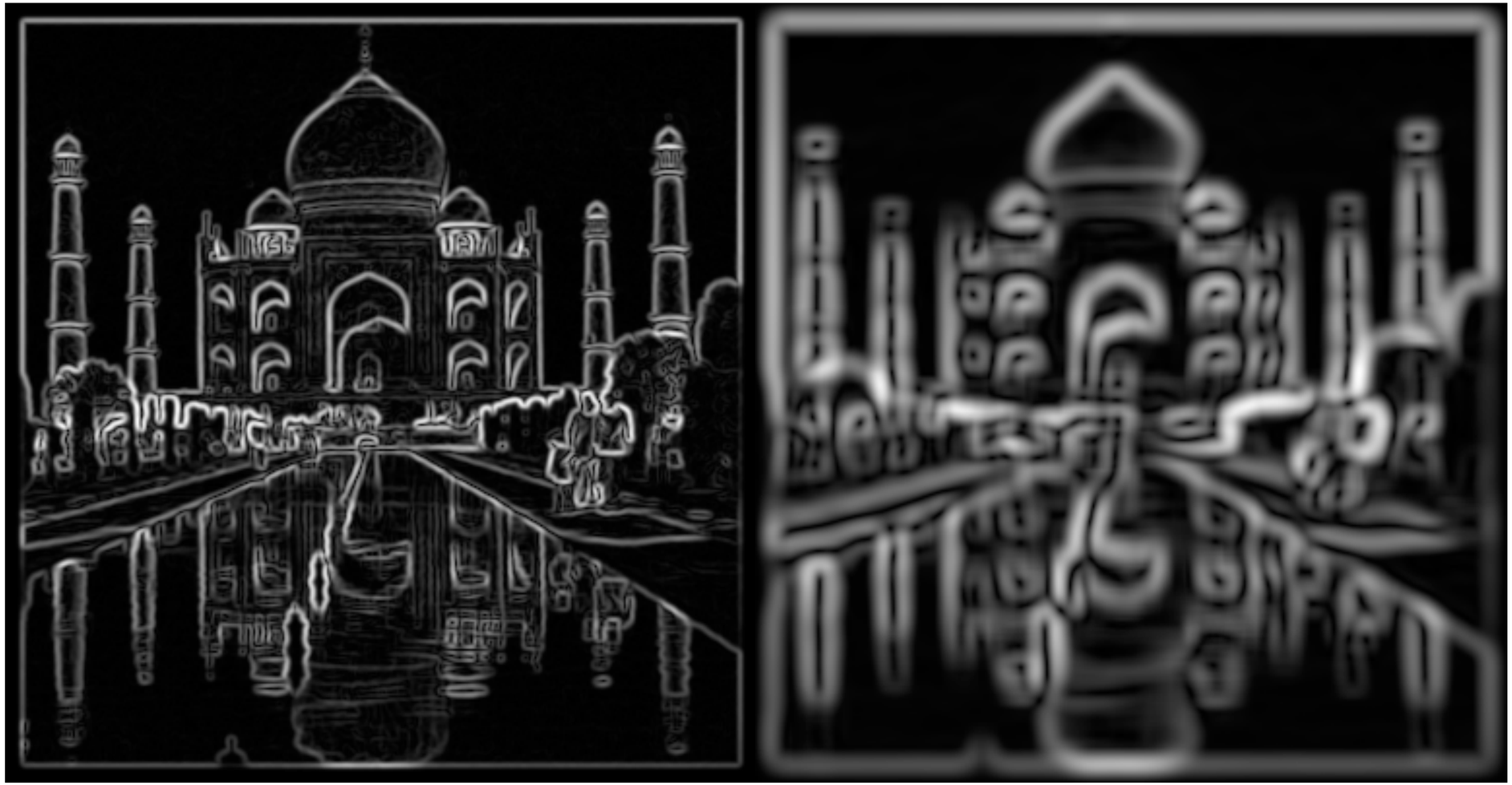
- Smooth input image with Gaussian
- Compute gradient image
- Apply non-maximal suppression
- Double thresholding

Canny: Smoothing and Edge Enhancement

1. **Smooth with a Gaussian kernel and differentiate**
2. **Equivalently, convolve with derivative of Gaussian**

- **Exploits separability of Gaussian kernel for convolution**
- **Balances localization and noise sensitivity**

Canny: Derivative of Gaussian

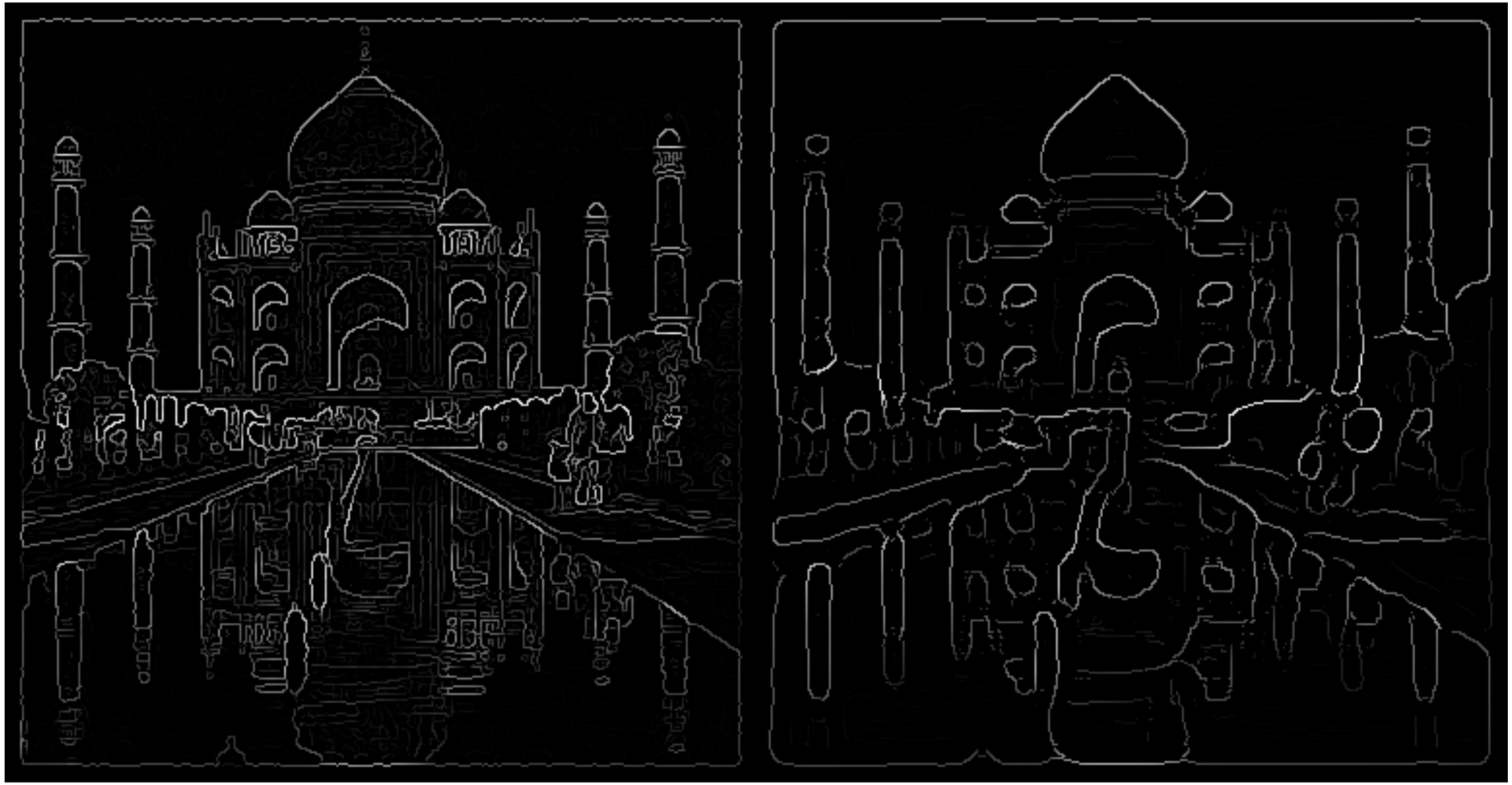


σ=1 σ=5

Canny: Non-maximal Suppression

- For each pixel
  - If the two pixels normal to edge direction have lower gradient magnitude
    - Keep the gradient magnitude
  - Otherwise
    - Set the gradient magnitude to zero

Canny: Non-max Output

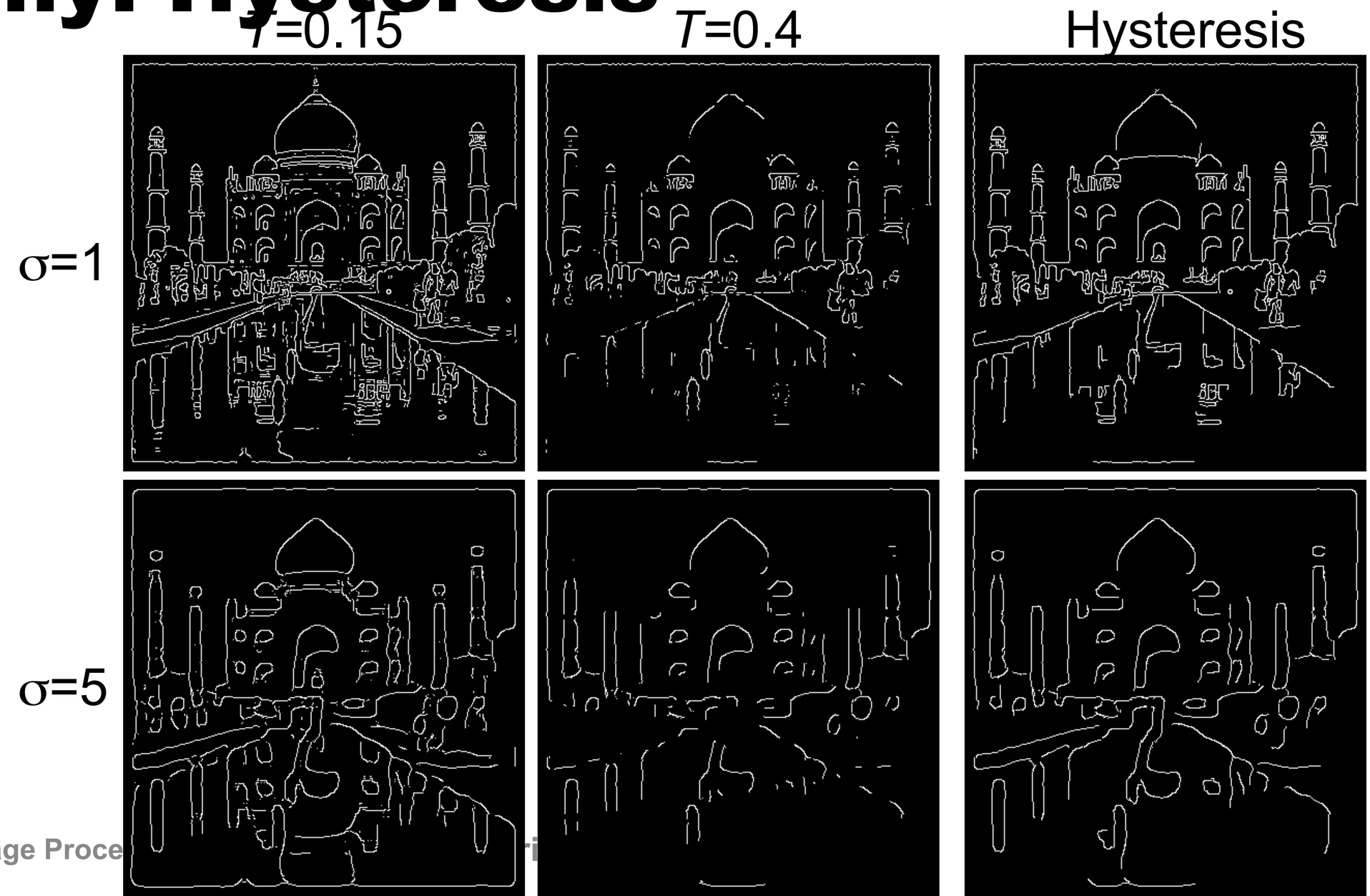


σ=1 σ=5

Canny: Hysteresis Threshold

- Threshold with high and low thresholds.
- Initialize edge map from high threshold.
- Iteratively add pixels in low threshold map with 8-neighbors already in the edge map.
- Repeat until convergence.

Canny: Hysteresis



Marr-Hildreth Edge Detector

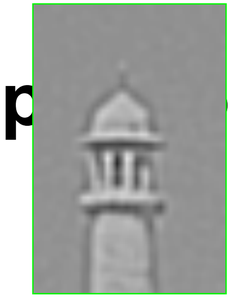
- Convolve with second derivative op r
  - Laplacian of Gaussian

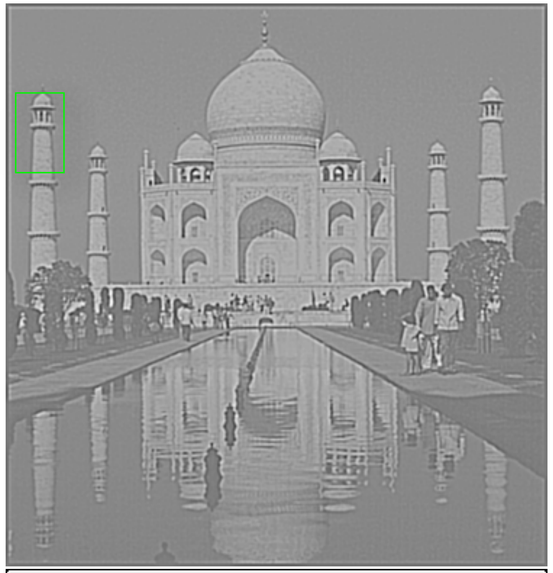
```
k=LoG(3,1);

tc=conv2(im,k);
```

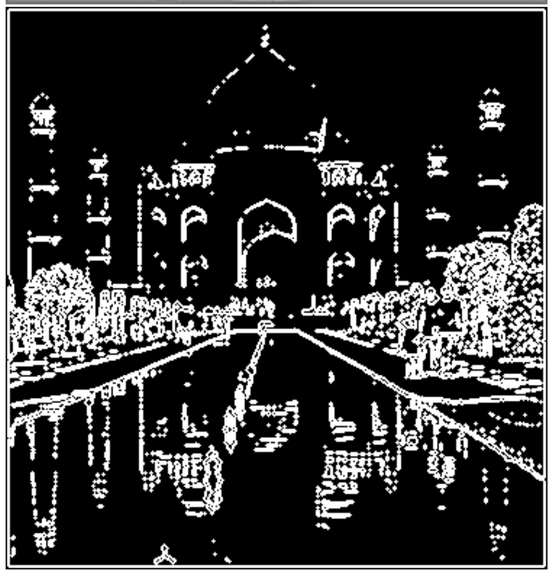
$$k(x,y) = \nabla^2 G(x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) G(x,y)$$

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad \nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Find zero crossings
- Sensitive to noise.

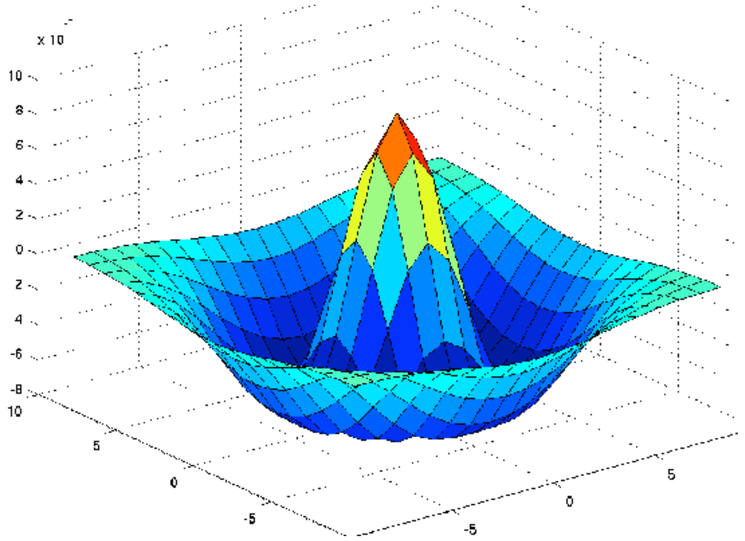




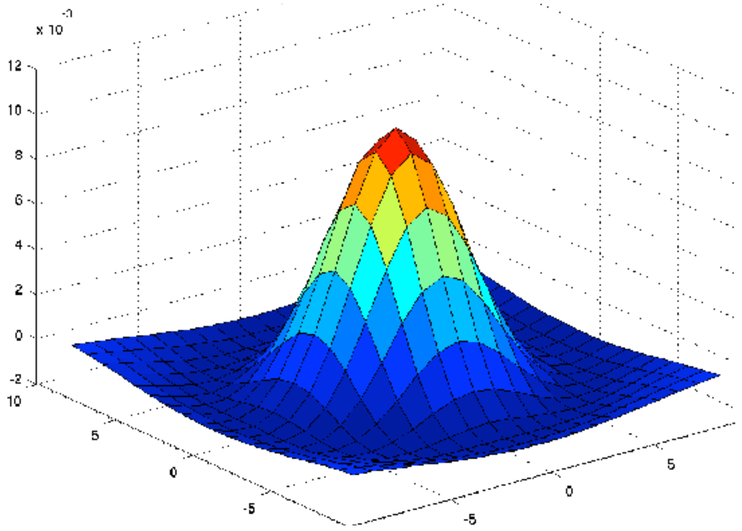


LoG vs DoG

- **Laplacian sometimes approximated by difference of two Gaussian filters (ratio of about 1.6):**



```
k=LoG(9,3);
surf(x,y,-k);
```



```
k= GausKern(9,3)-GausKern(9,4.8);
surf(x,y,-k);
```

Model Fitting

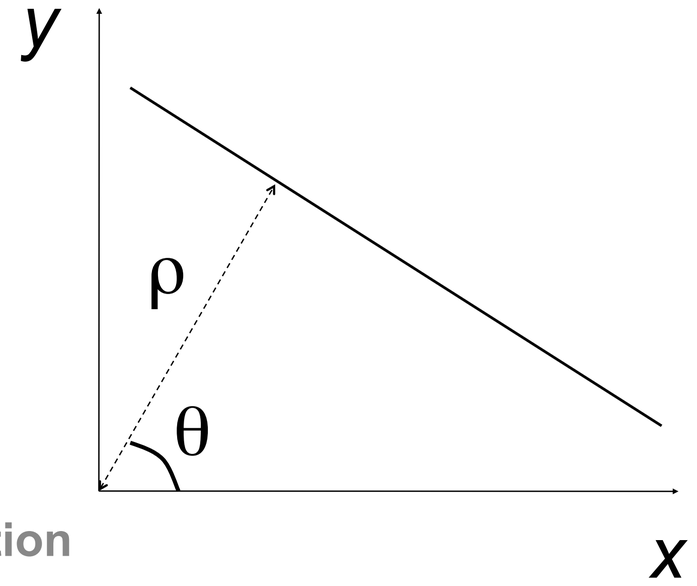
- Locate edges by fitting a surface.
- Create a model edge, e.g., step edge with parameters:
  - Orientation
  - Position
  - Intensities either side of the edge
- Find least-squares fit in each small window.
- Accept if fit is above a threshold.

Hough Transform

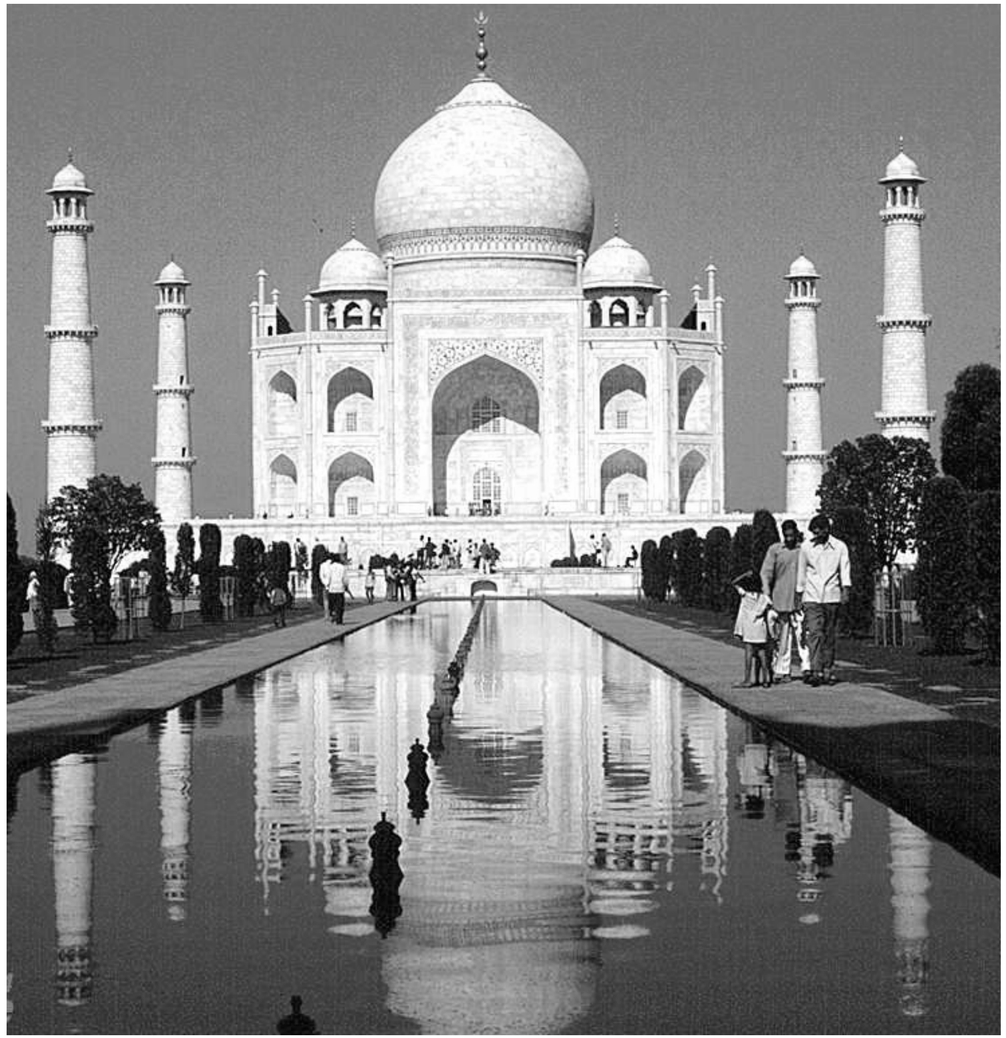
- **Finds the most likely lines in an image.**
- **At every edge pixel, compute the local equation of the edge line:**

- **Store a histogram of the line parameters θ and ρ.**
- **The most-full histogram bins are the dominant image lines.**

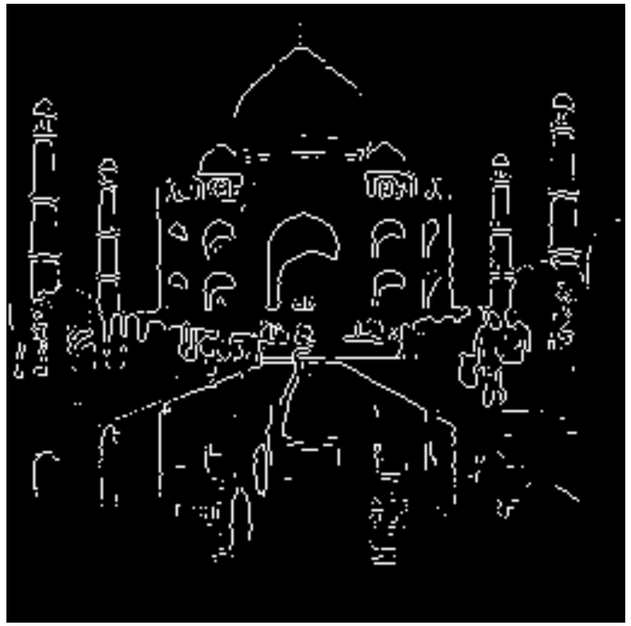
$$x \cos\theta + y \sin\theta = \rho$$



Example

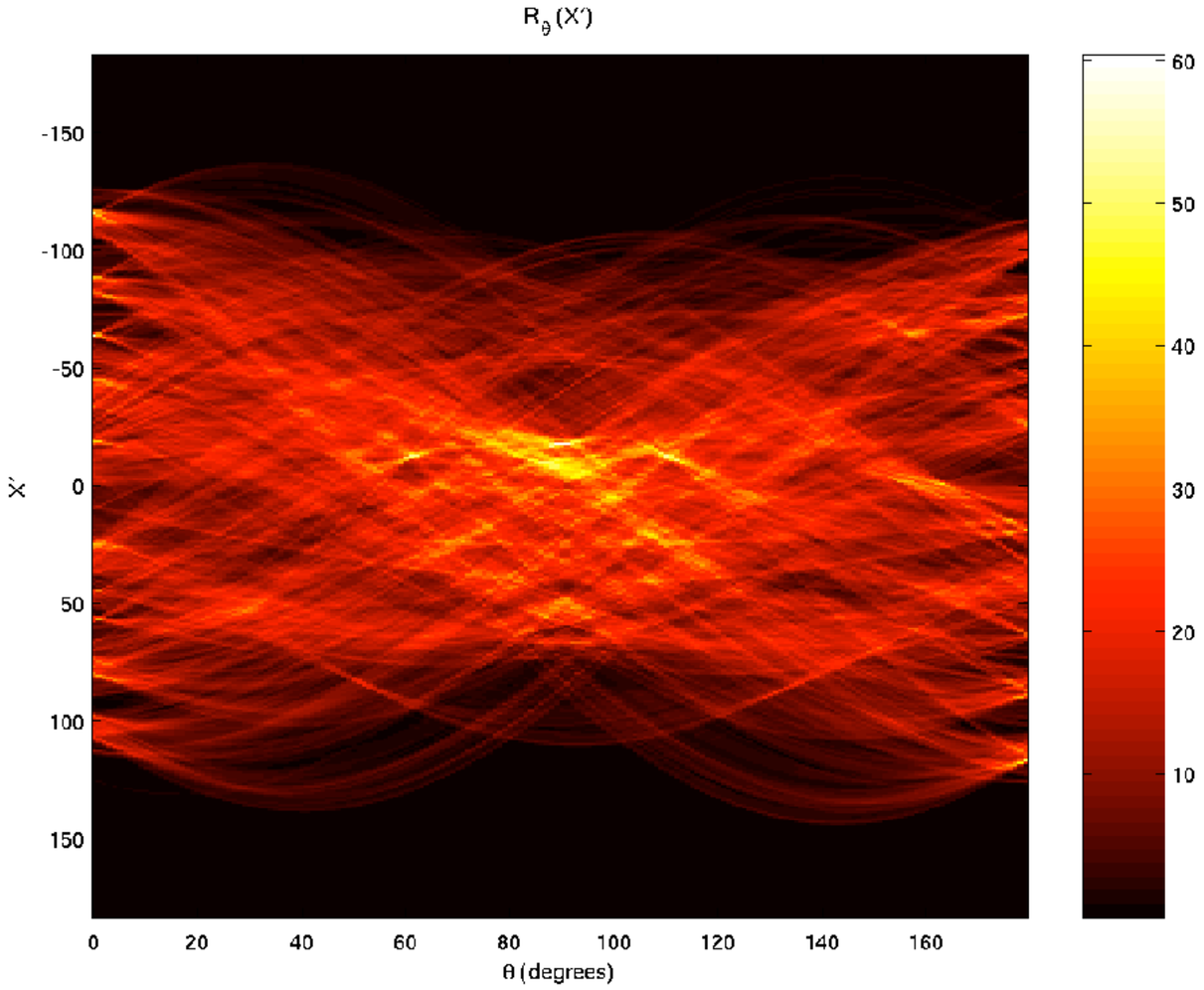


```
ed=edge(im);
```



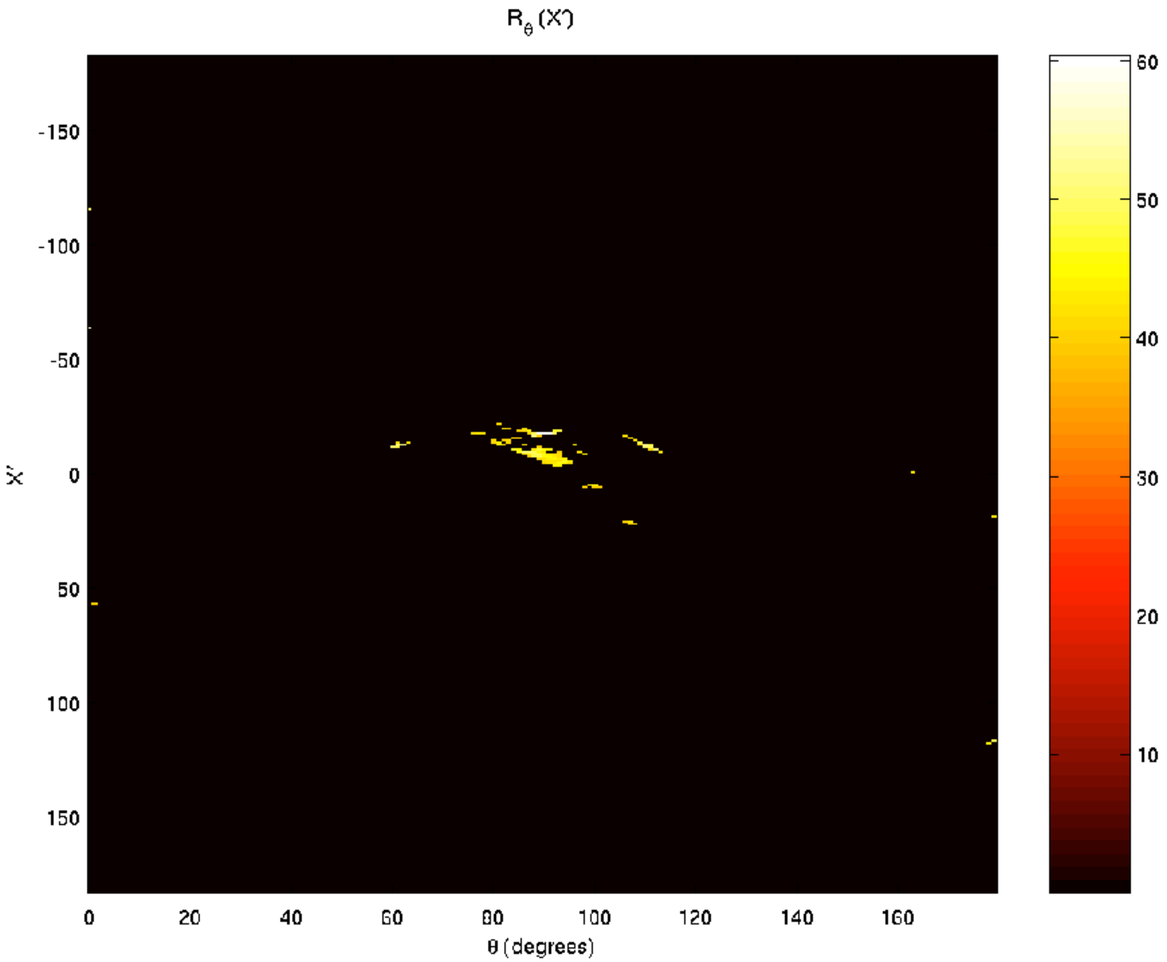
Hough Transform

```
theta=0:179;
[R,xp]=
radon(ed,theta);
```



Thresholded

```
RT=R.*(R>40);
```



Inverted and Superimposed

```
lines=iradon(RT,theta);
```

Generalization

- **The general Hough transform works with any parametric shape.**
- **E.g., circles:**
  - **Make a 3D histogram of x_0, y_0 and r.**

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

  - **Threshold and back project in the same way.**

Pros and Cons

- **First-derivative approach**
  - Fast, simple to implement and understand.
  - Noise sensitive, misses corners, lots of thresholds to select.
- **Second-derivative approach**
  - Few thresholds to choose, fast.
  - Very sensitive to noise.
- **Model fitting**
  - Slow.
  - Less sensitive to noise.

Performance

- How can we evaluate edge-detector performance?
  - Probability of false edges
  - Probability of missing edges
  - Error in edge angle
  - Mean squared distance from true edge

Summary

- Resize by re-sampling (must pre-filter!!)
- Image pyramids applications
- Edge detection
  - Simple
  - Canny
- Hough Transform