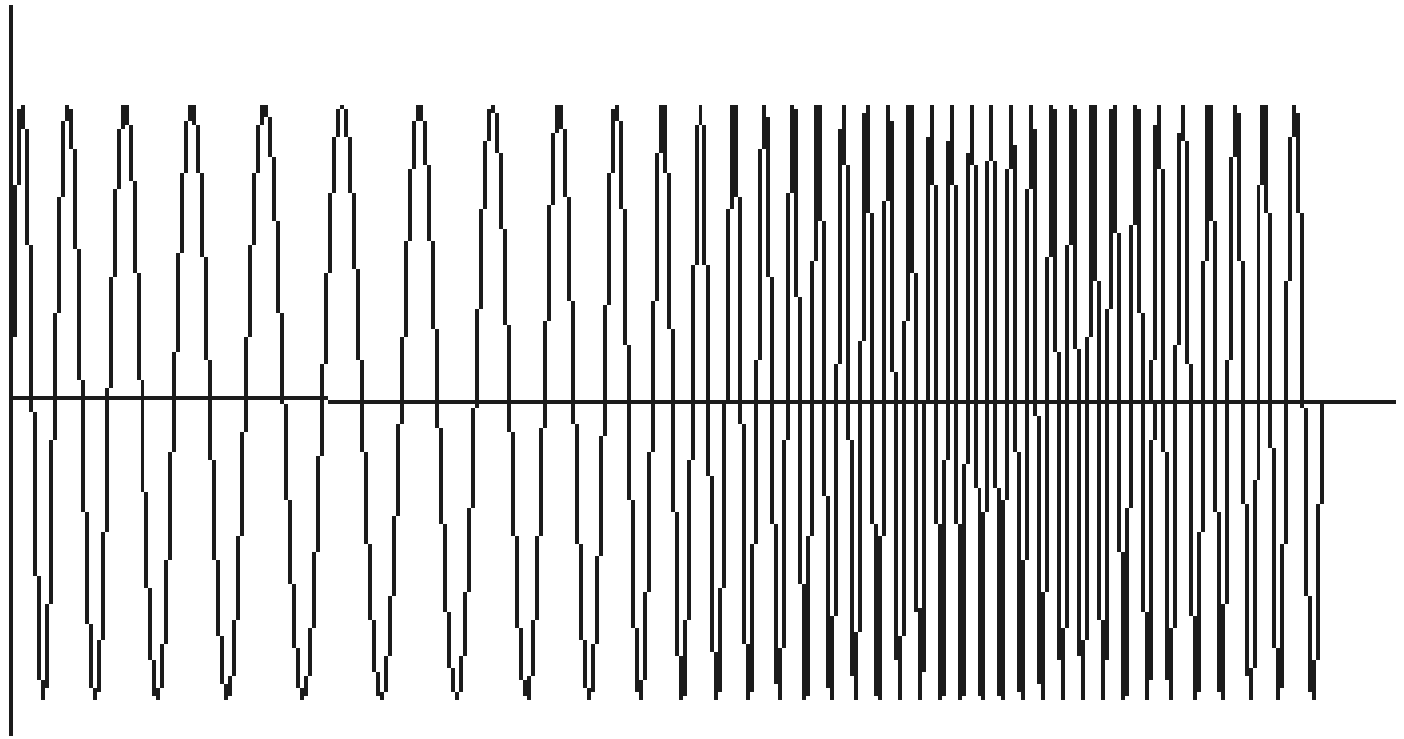
Filtering Applications
&
Edge Detection

Outline

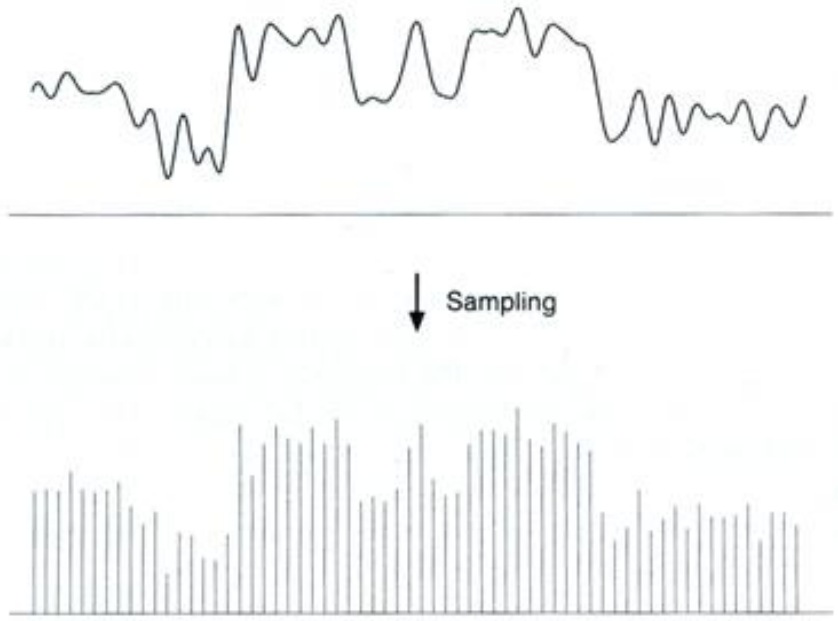
- Sampling & Reconstruction Revisited
  - Anti-Aliasing
- Edges
- Edge detection
- Simple edge detector
- Canny edge detector
- Performance analysis
- Hough Transform

1D Example: Audio



Sampled representations

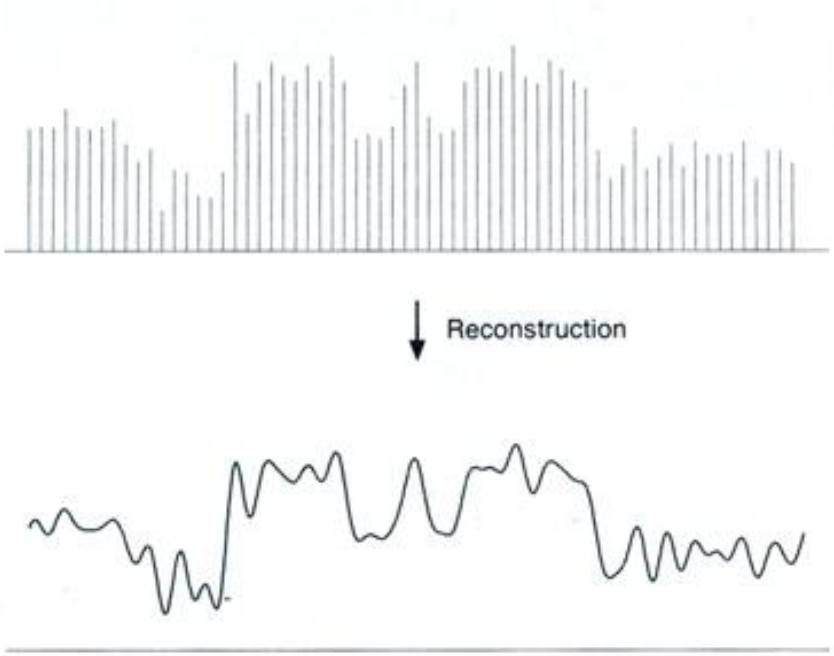
- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function's values at many points



[FvDFH fig.14.14b / Wolberg]

Reconstruction

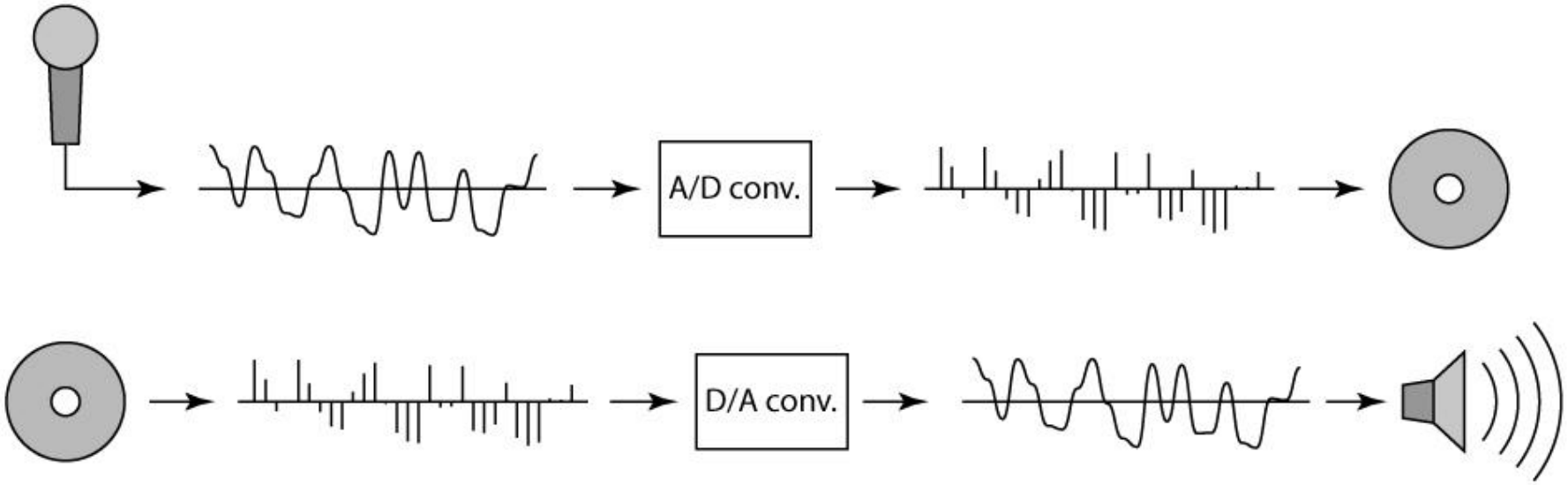
- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)
  - amounts to “guessing” what the function did in between



[FvDFH fig.14.14b / Wolberg]

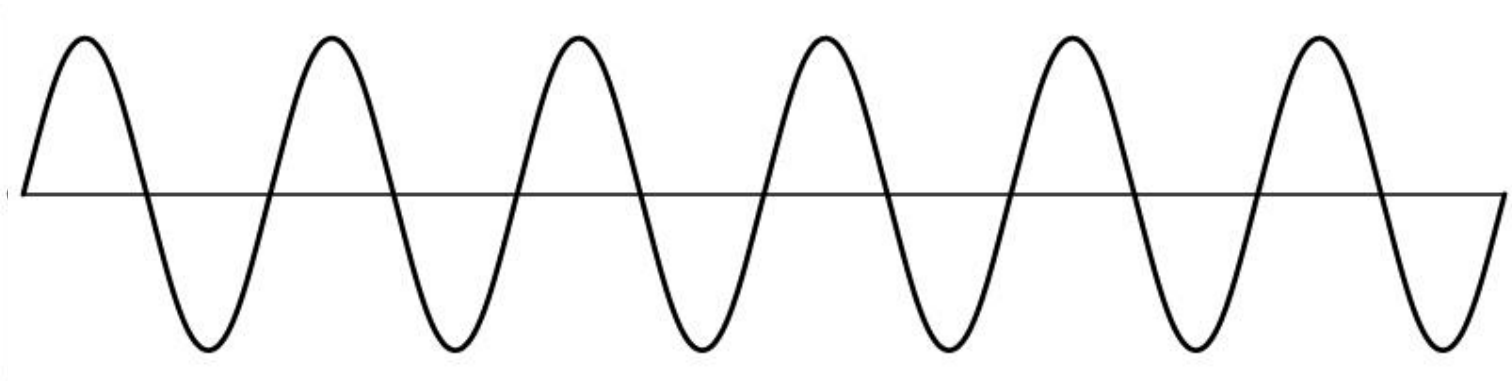
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?



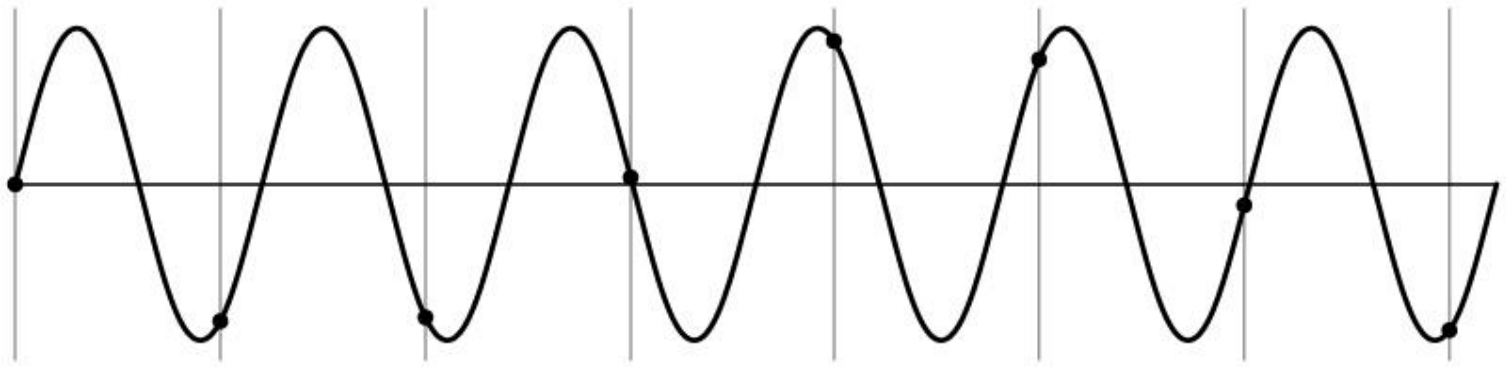
Sampling and Reconstruction

- Simple example: a sine wave



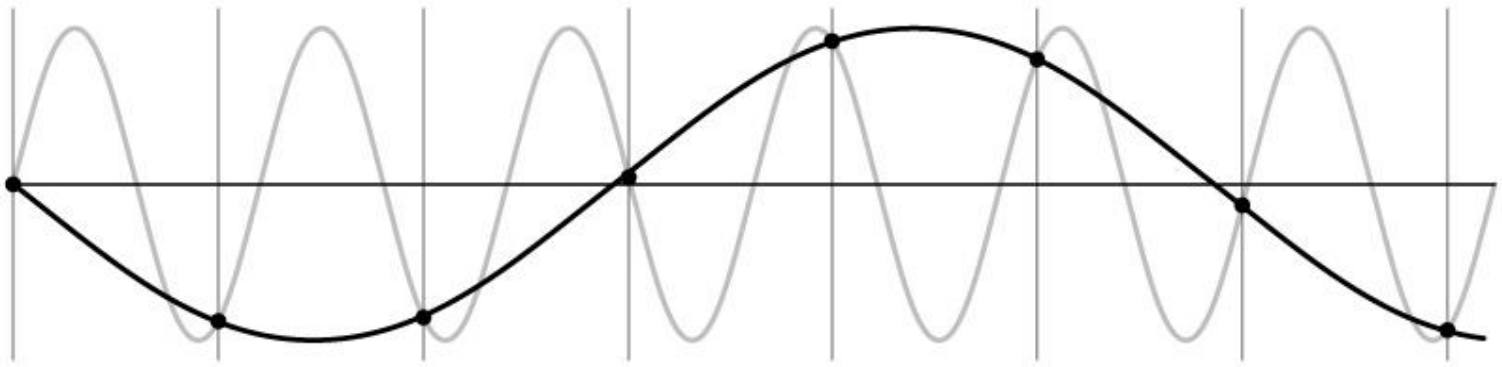
Undersampling

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



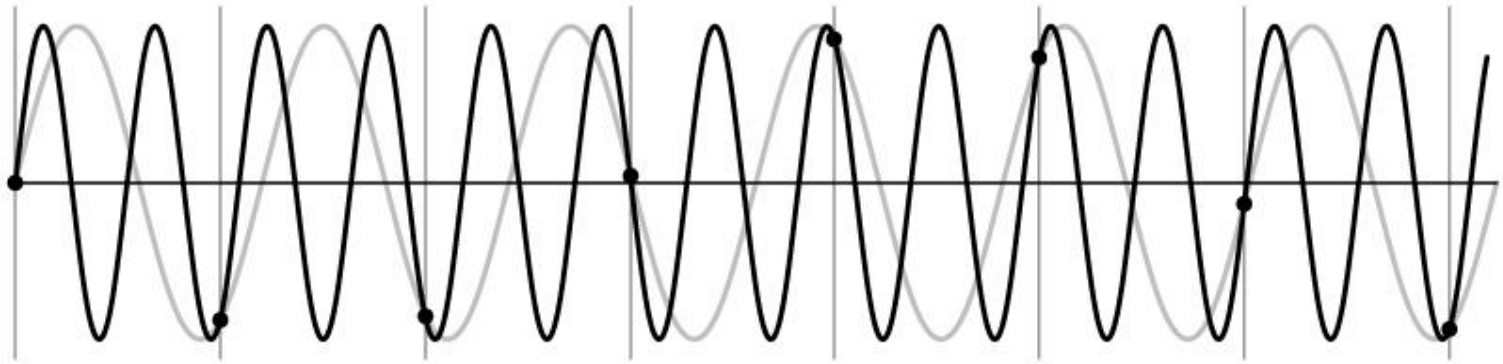
Undersampling

- 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

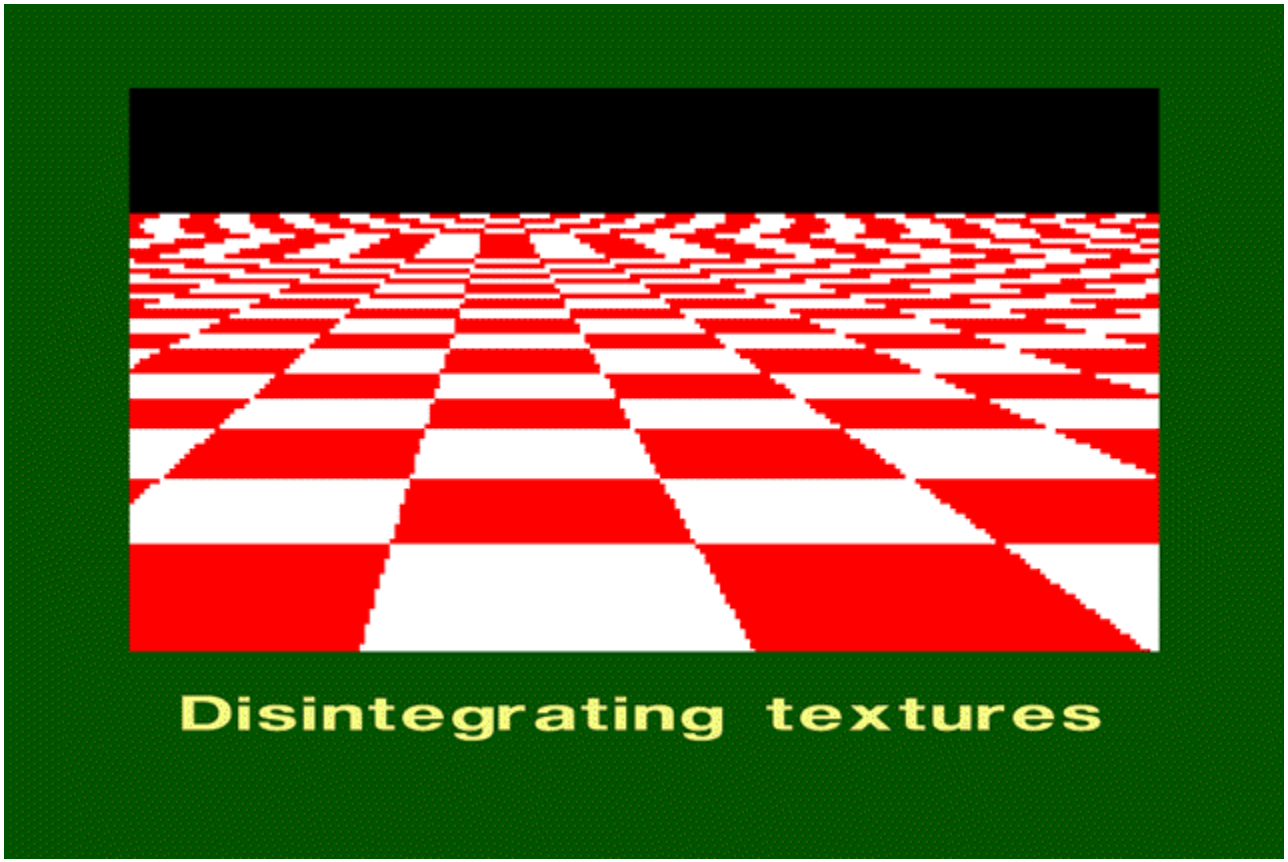


Undersampling

- 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

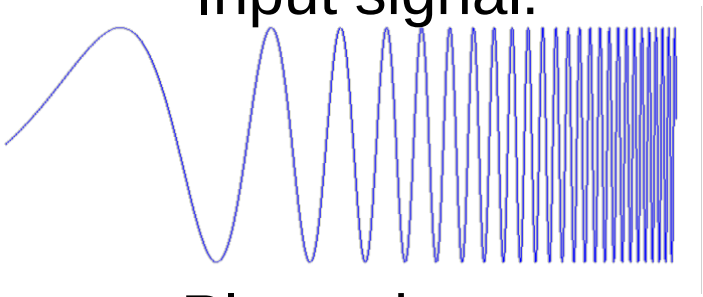


Aliasing in images

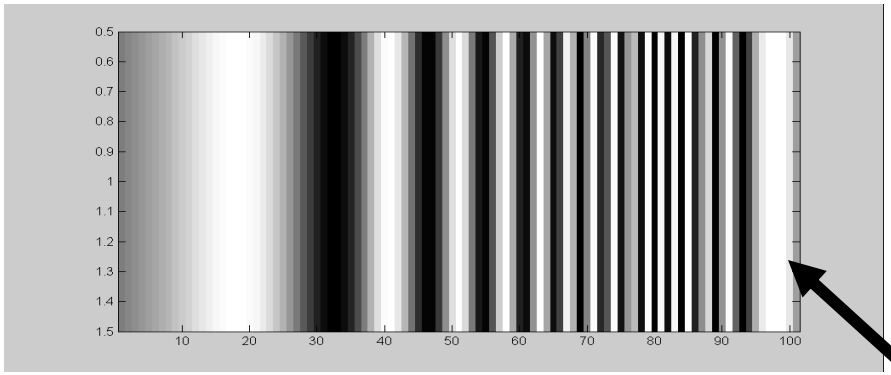


What's happening?

Input signal:



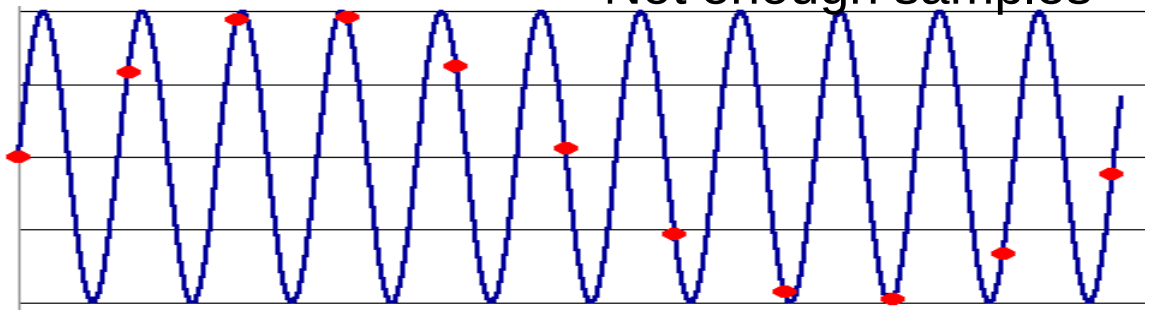
Plot as image:



x = 0:.05:5; imagesc(sin((2.^x).*x))

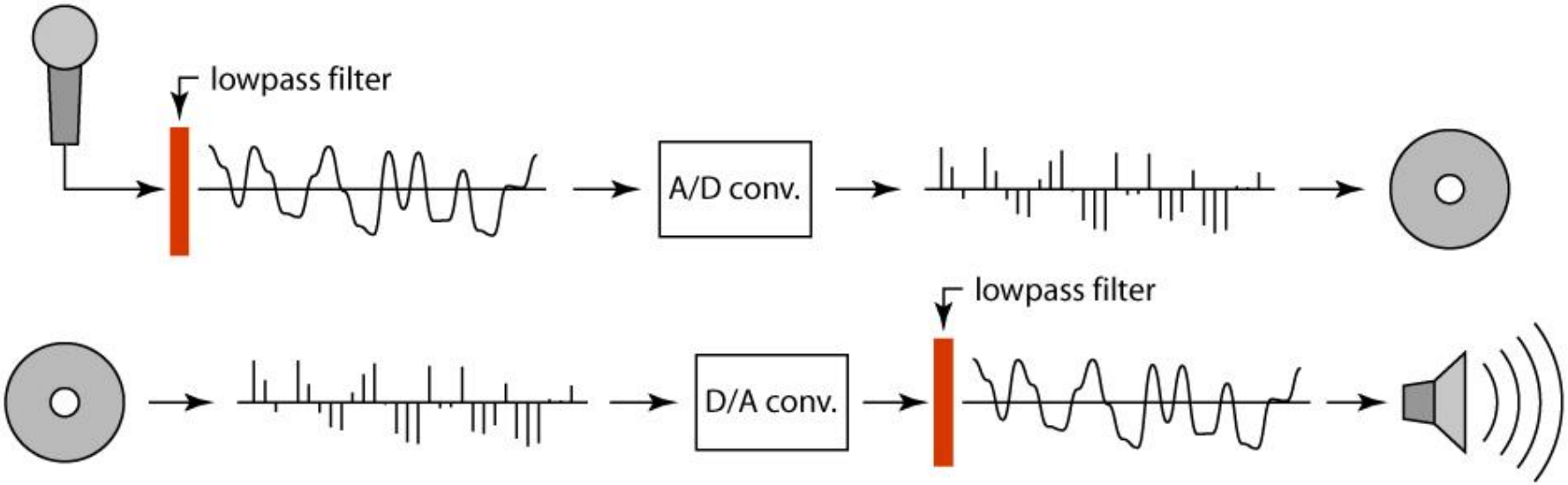
Alias!

Not enough samples



Preventing aliasing

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

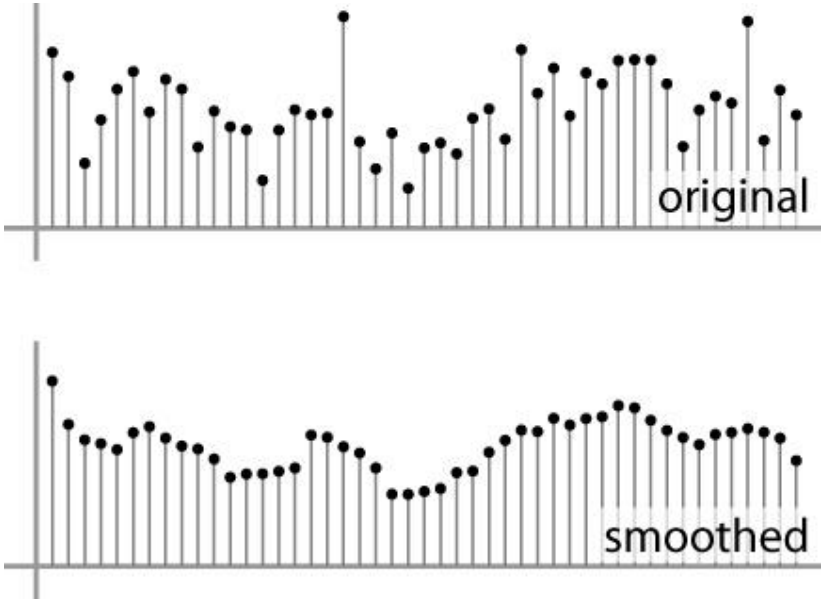


Linear filtering: key points

- Transformations on signals; e.g.:
  - bass/treble controls on stereo
  - blurring/sharpening operations in image editing
  - smoothing/noise reduction in tracking
- Key properties
  - linearity: filter($f + g$) = filter(f) + filter(g)
  - shift invariance: behavior invariant to shifting the input
    - delaying an audio signal
    - sliding an image around
- Can be modeled mathematically by *convolution*

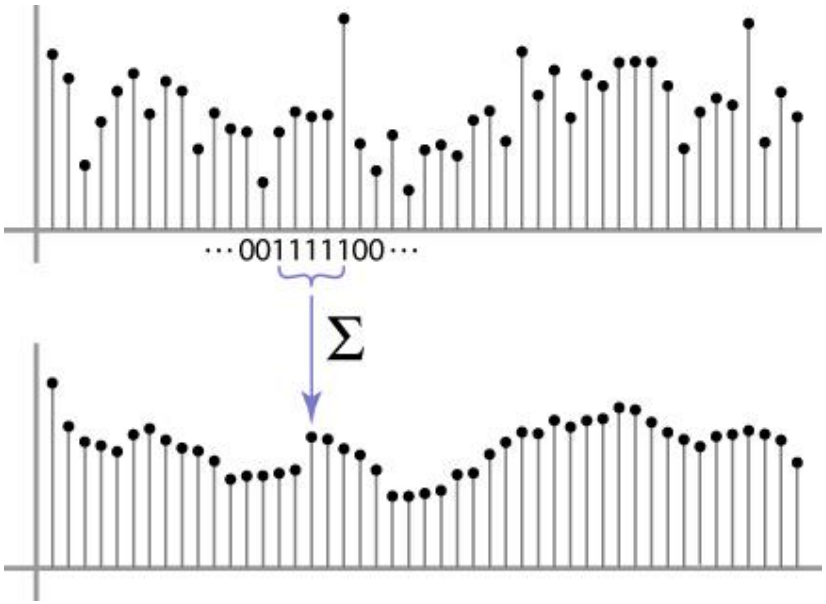
Moving Average

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing



Weighted Moving Average

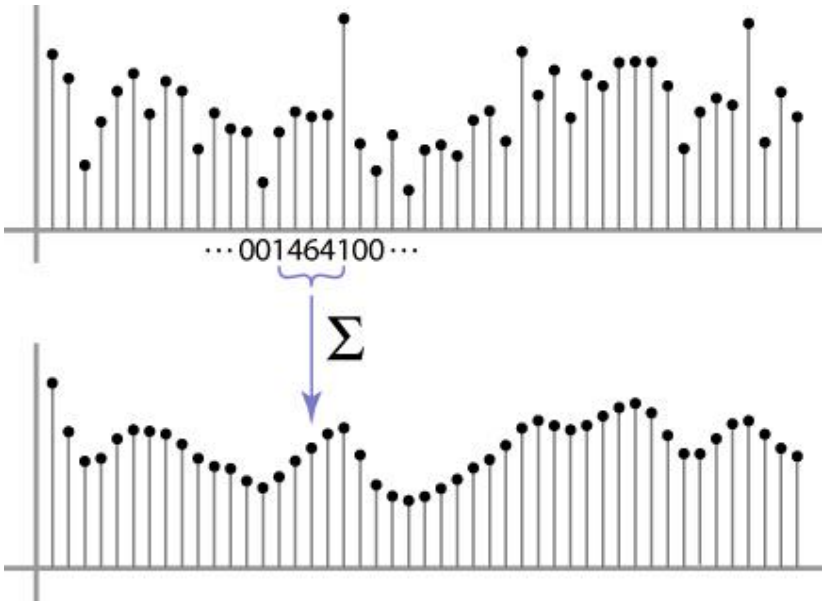
- Can add weights to our moving average
- *Weights* […, 0, 1, 1, 1, 1, 1, 0, …] / 5



Weighted Moving Average

- bell curve (gaussian-like) weights:

  […, 1, 4, 6, 4, 1, …]



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

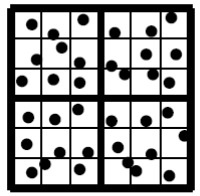
Antialiasing

- What can be done?

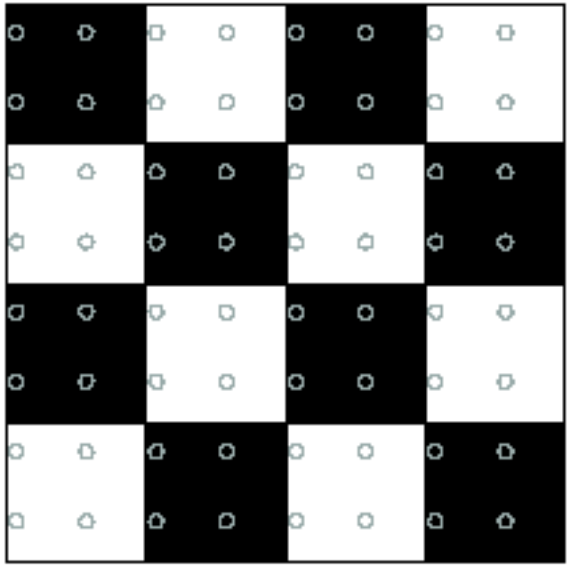
Sampling rate > 2 * max frequency in the image

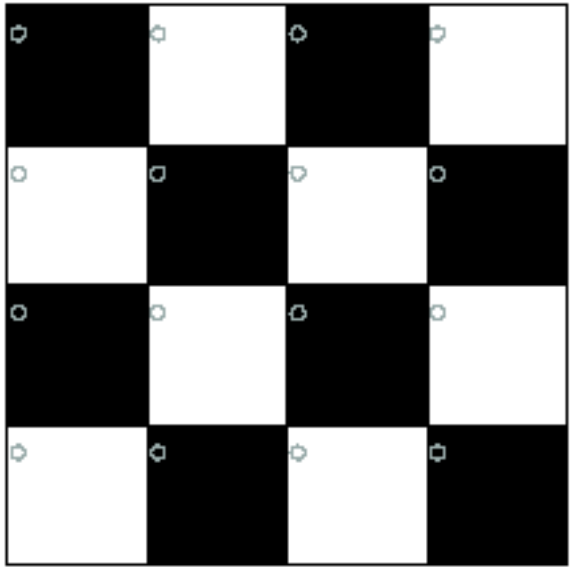
1. Raise sampling rate by *oversampling*
   - *Sample at k times the re*
   - continuous signal: easy
   - discrete signal: need to
2. Lower the max freque
   - Smooth the signal enou
   - Works on discrete signals
3. Improve sampling quality with better sampling
   - Below Nyquist frequency is best 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.



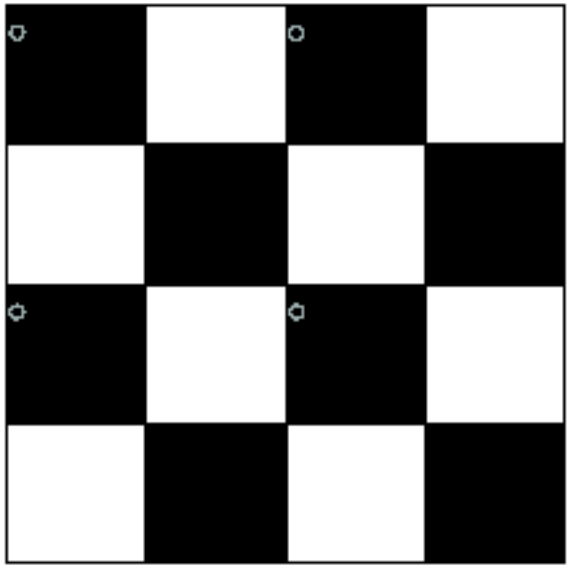
jittered,
9 samples per pixel

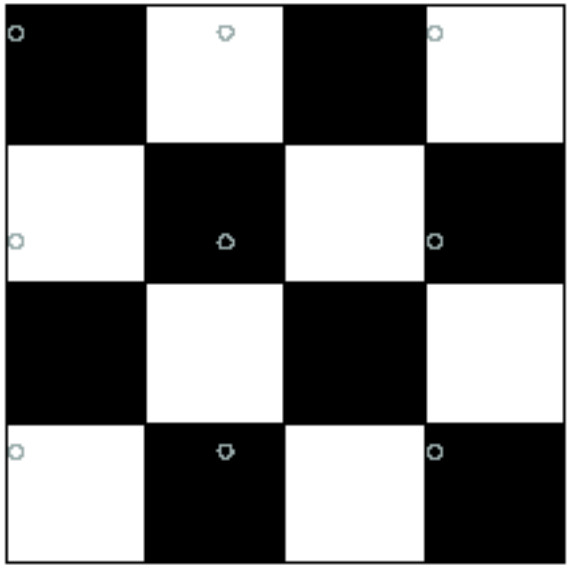




Good sampling:

- 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?

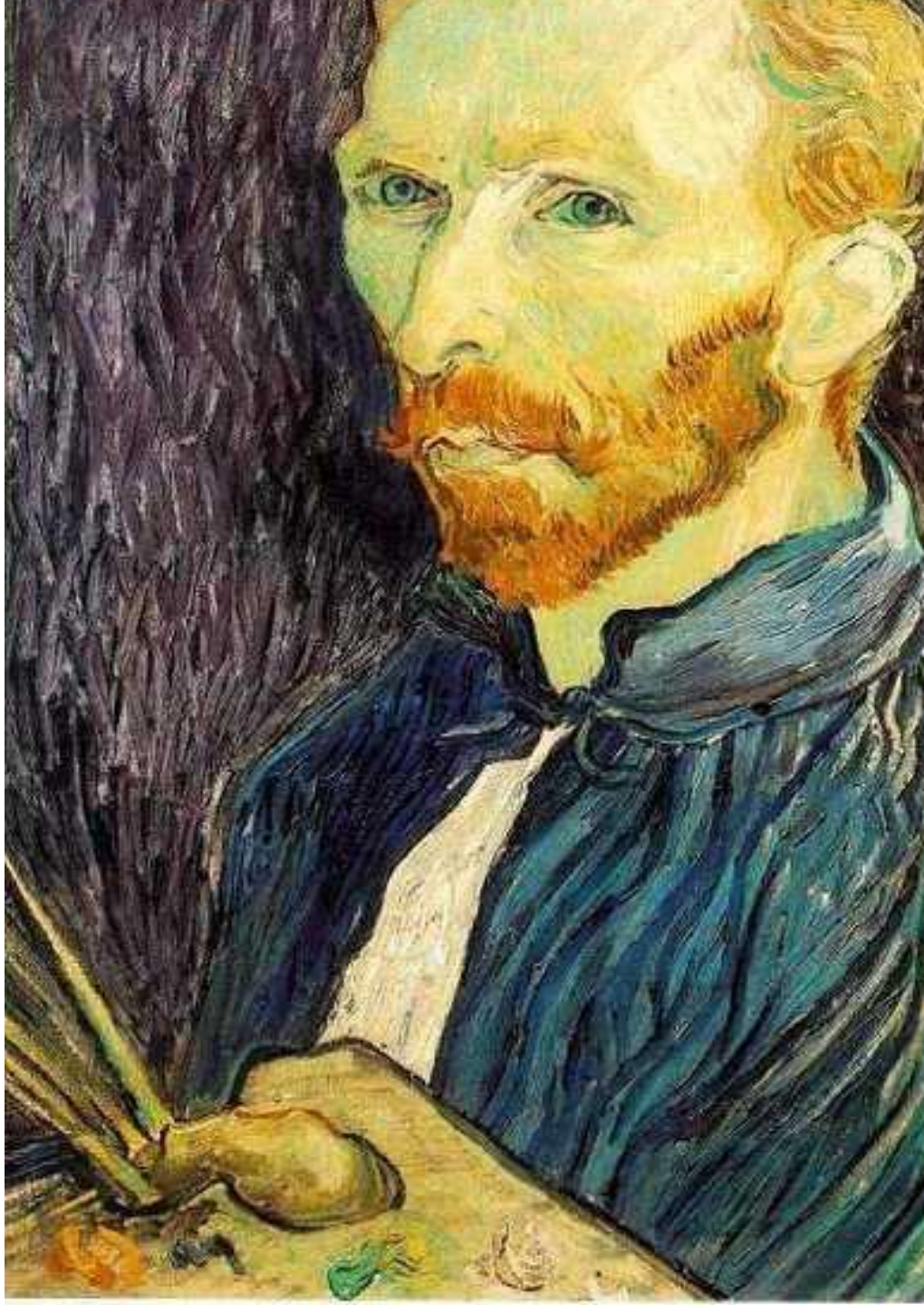
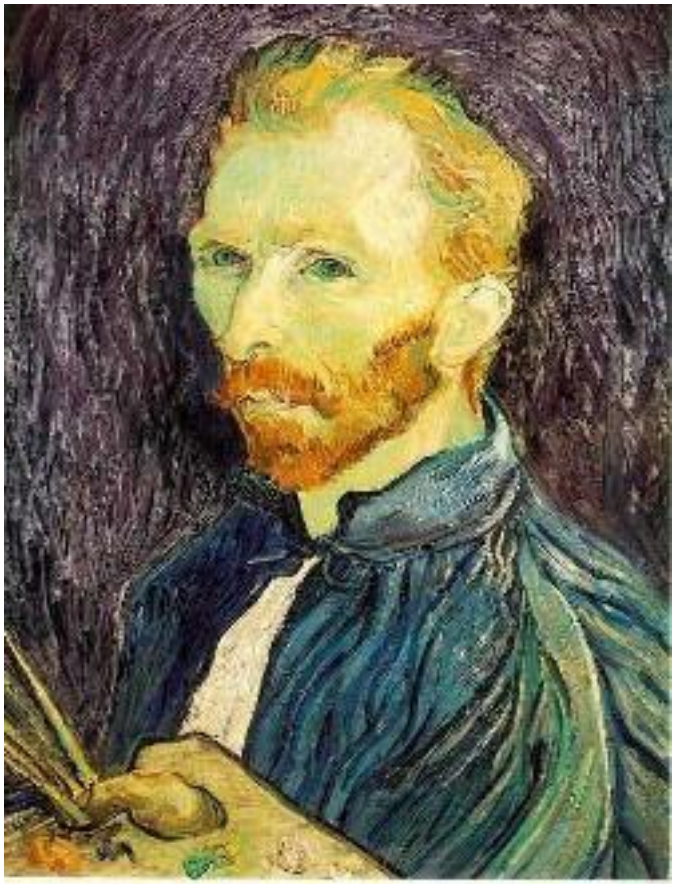


Image sub-sampling



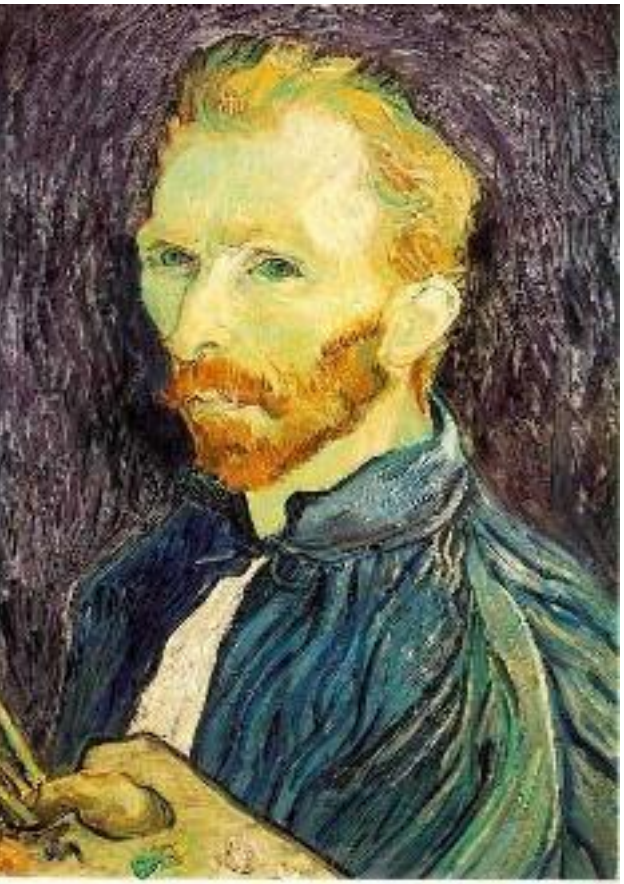
1/4

1/8

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

Slide by Steve Seitz

Image sub-sampling



1/2

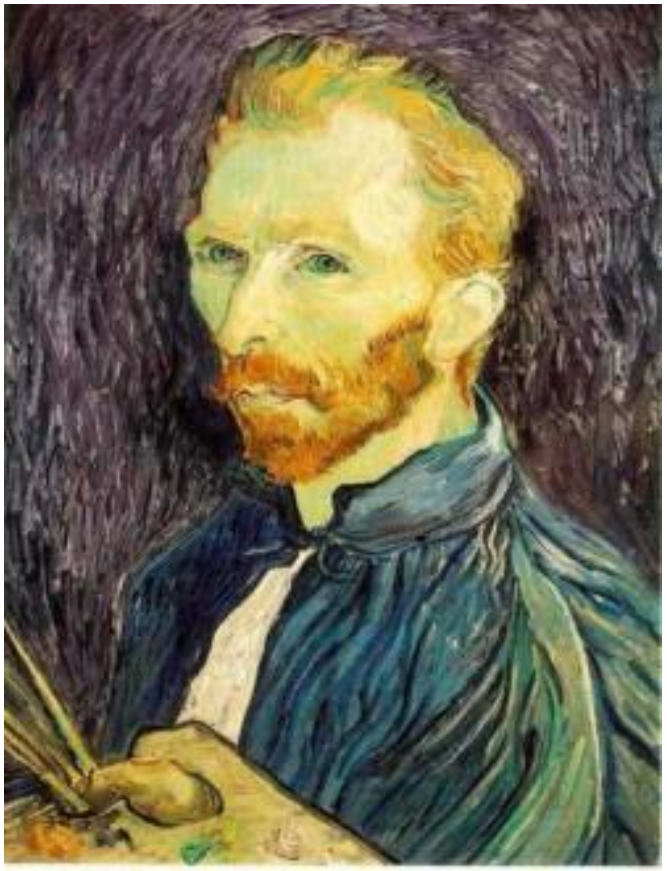
1/4 (2x zoom)

1/8 (4x zoom)

Aliasing! What do we do?

Slide by Steve Seitz

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?

Slide by Steve Seitz

Subsampling with Gaussian Pre-filtering

Gaussian 1/2

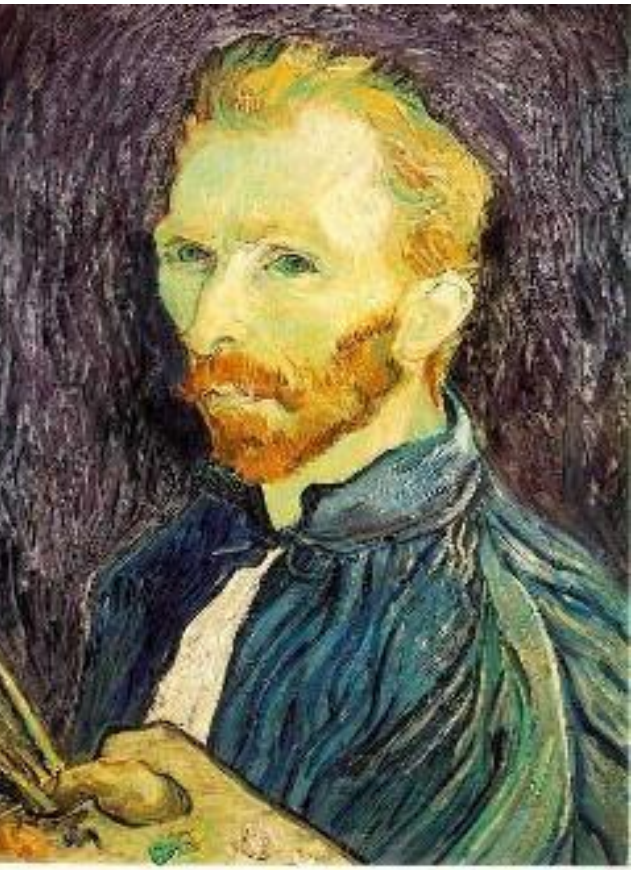
G 1/4

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)

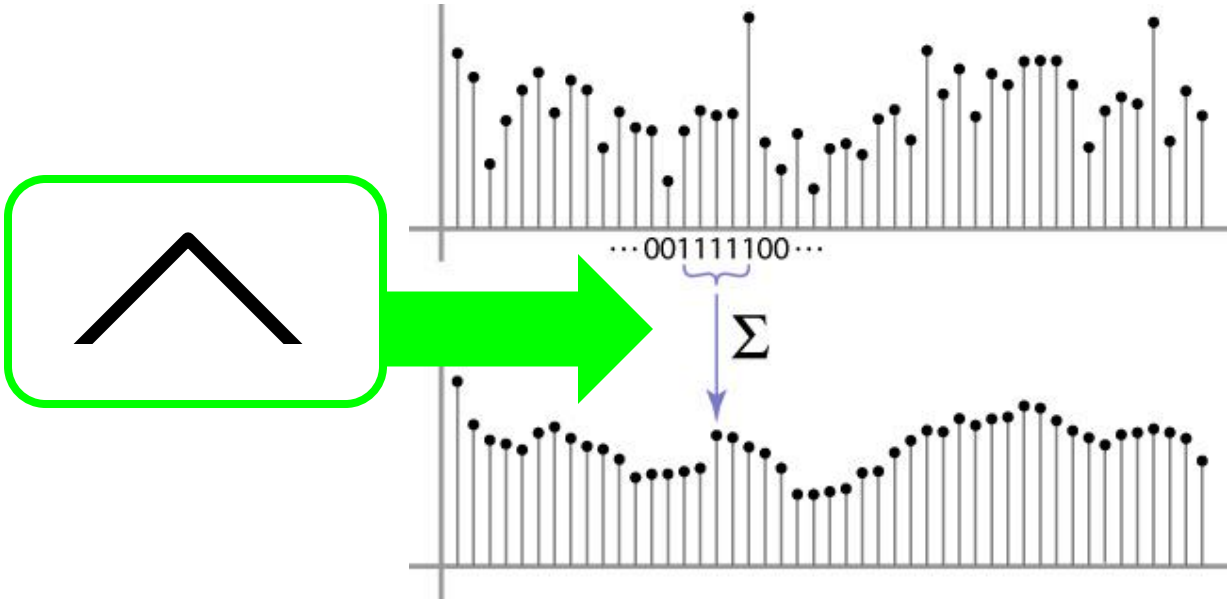
1/8 (4x zoom)

Last Point About Reconstruction

- If we replace box-filter's weights:

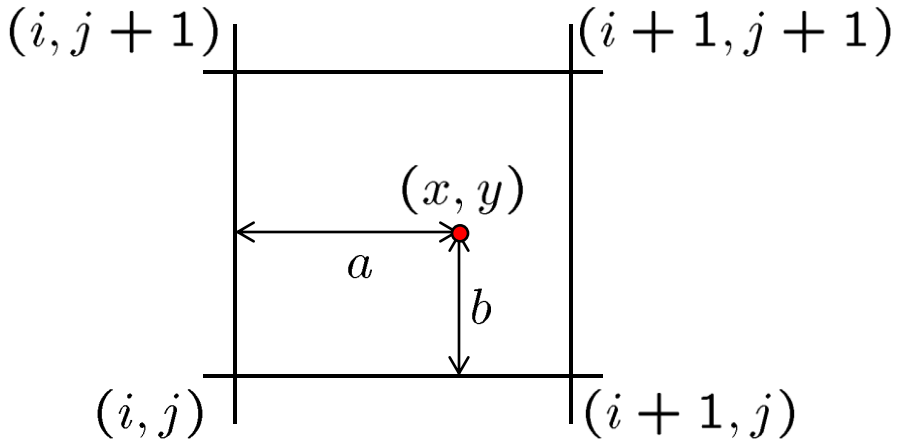
  ([…, 0, 1, 1, 1, 1, 1, 0, …] / 5)

with a triangle, width = 2 x period, then what?



Bilinear interpolation

- Sampling at *f(x,y):*



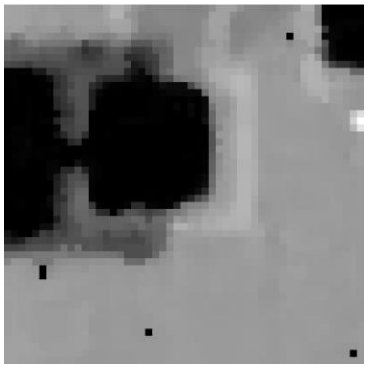
$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad f[i+1,j] \\
& +ab \quad f[i+1,j+1] \\
& +(1-a)b \quad f[i,j+1]
\end{aligned}
$$

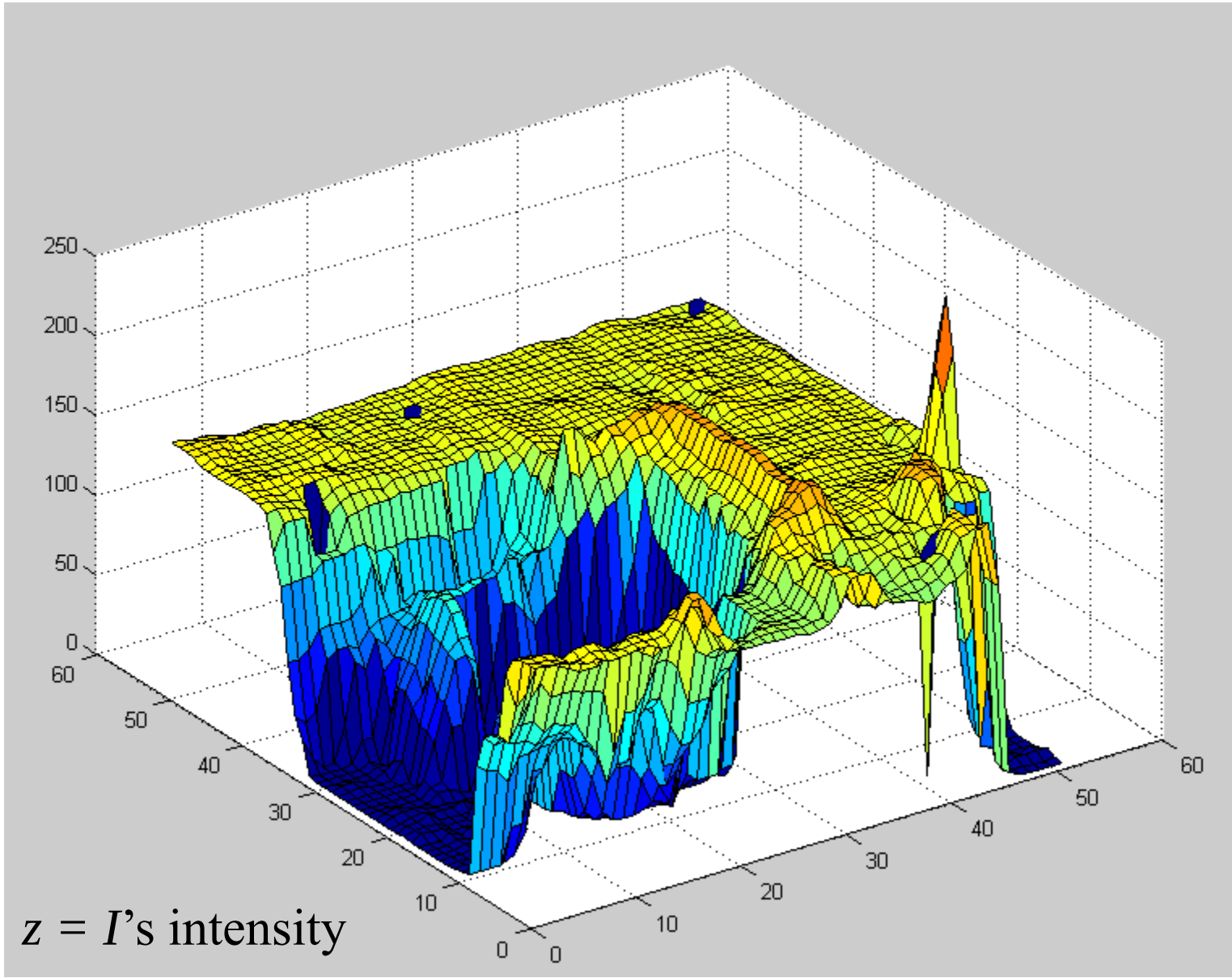
- Time for aliasing Clock-face problem?

Photos + implementation by Rob Orr

Image == Heightfield

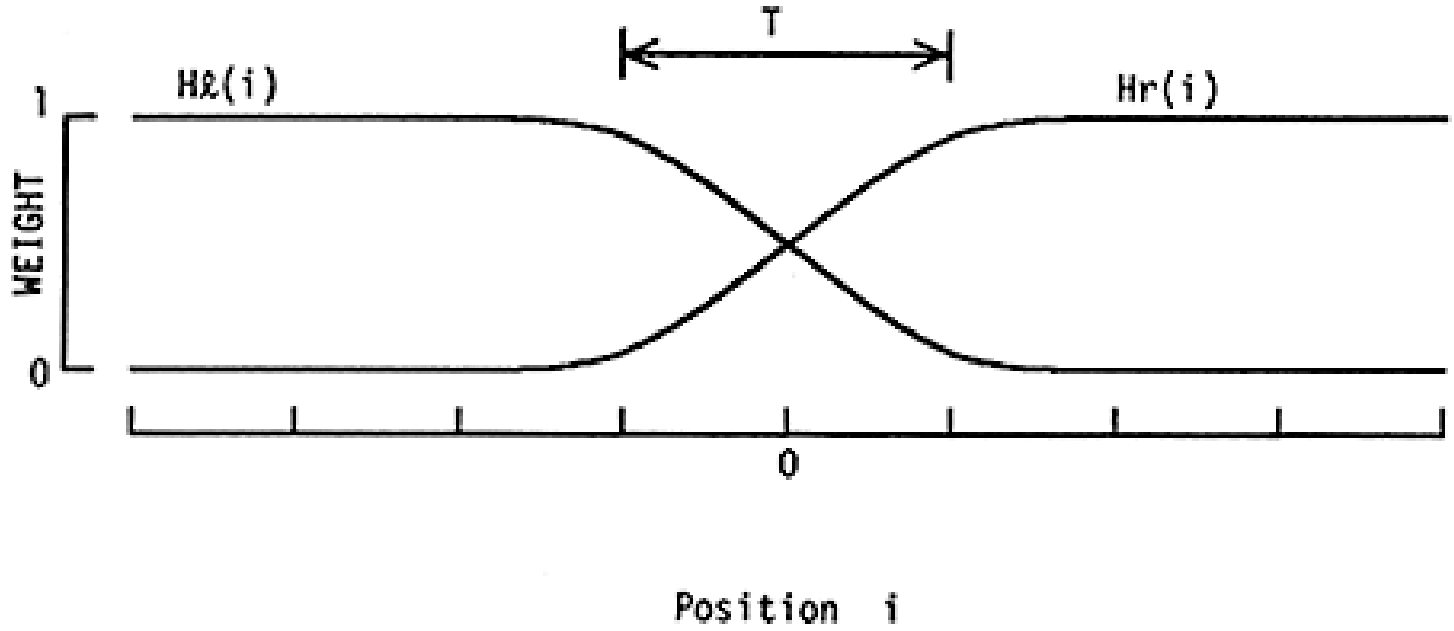
I





z = I's intensity

How does one normally blend?



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

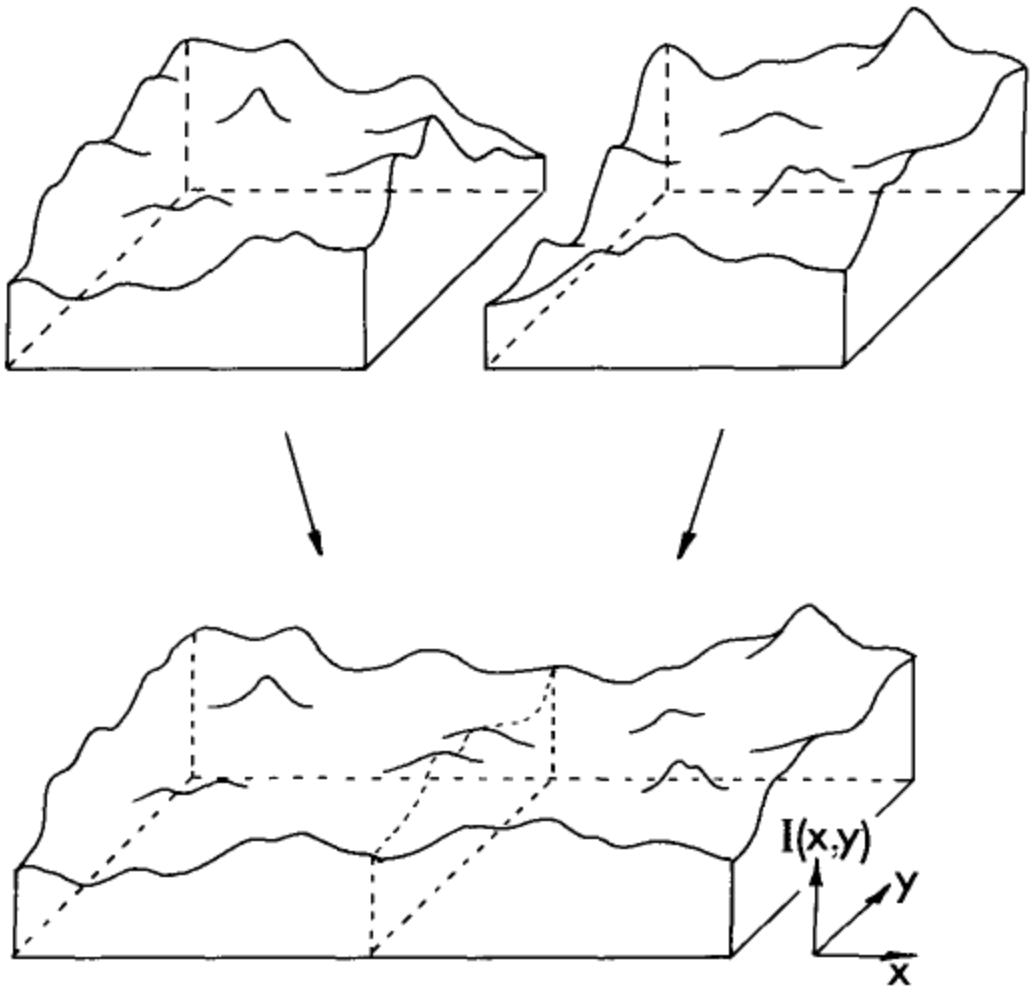
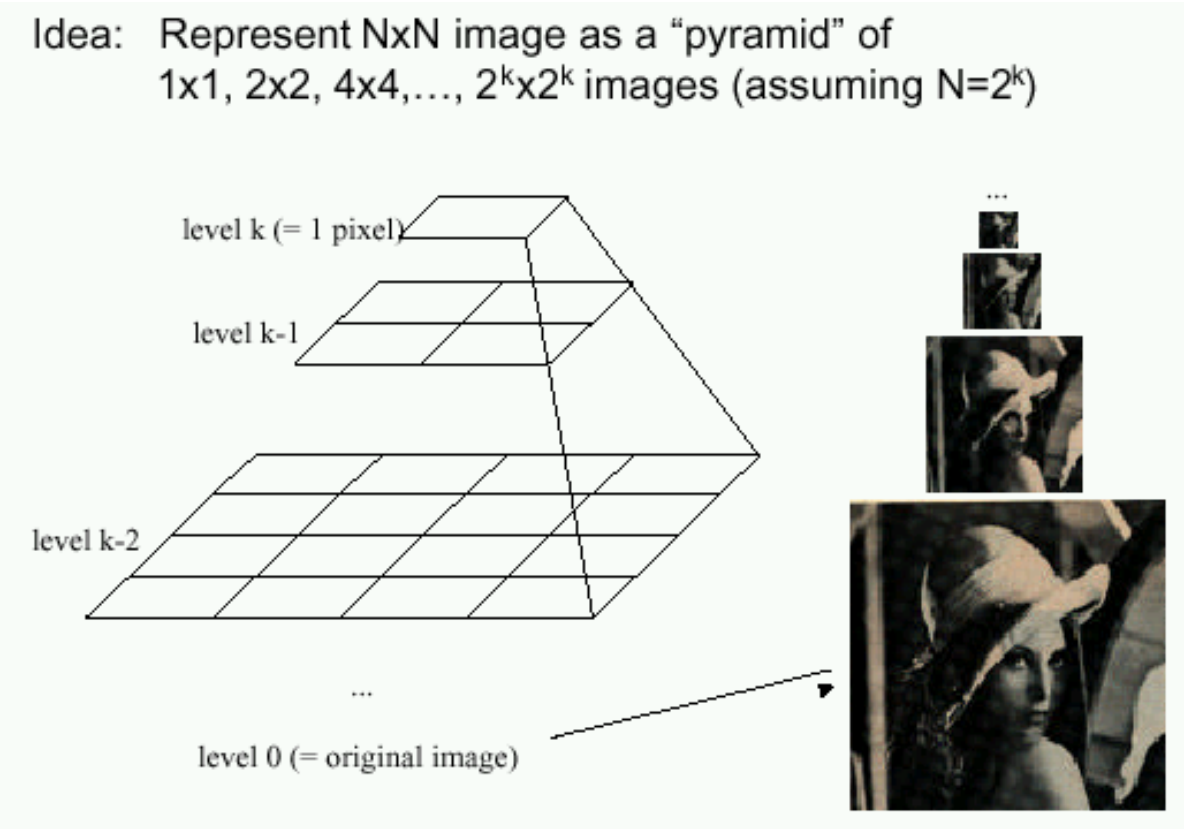


Fig. 1. A pair of images may be represented as a pair of surfaces above the (x, y) plane. The problem of image splining is to join these surfaces with a smooth seam, with as little distortion of each surface as possible.

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

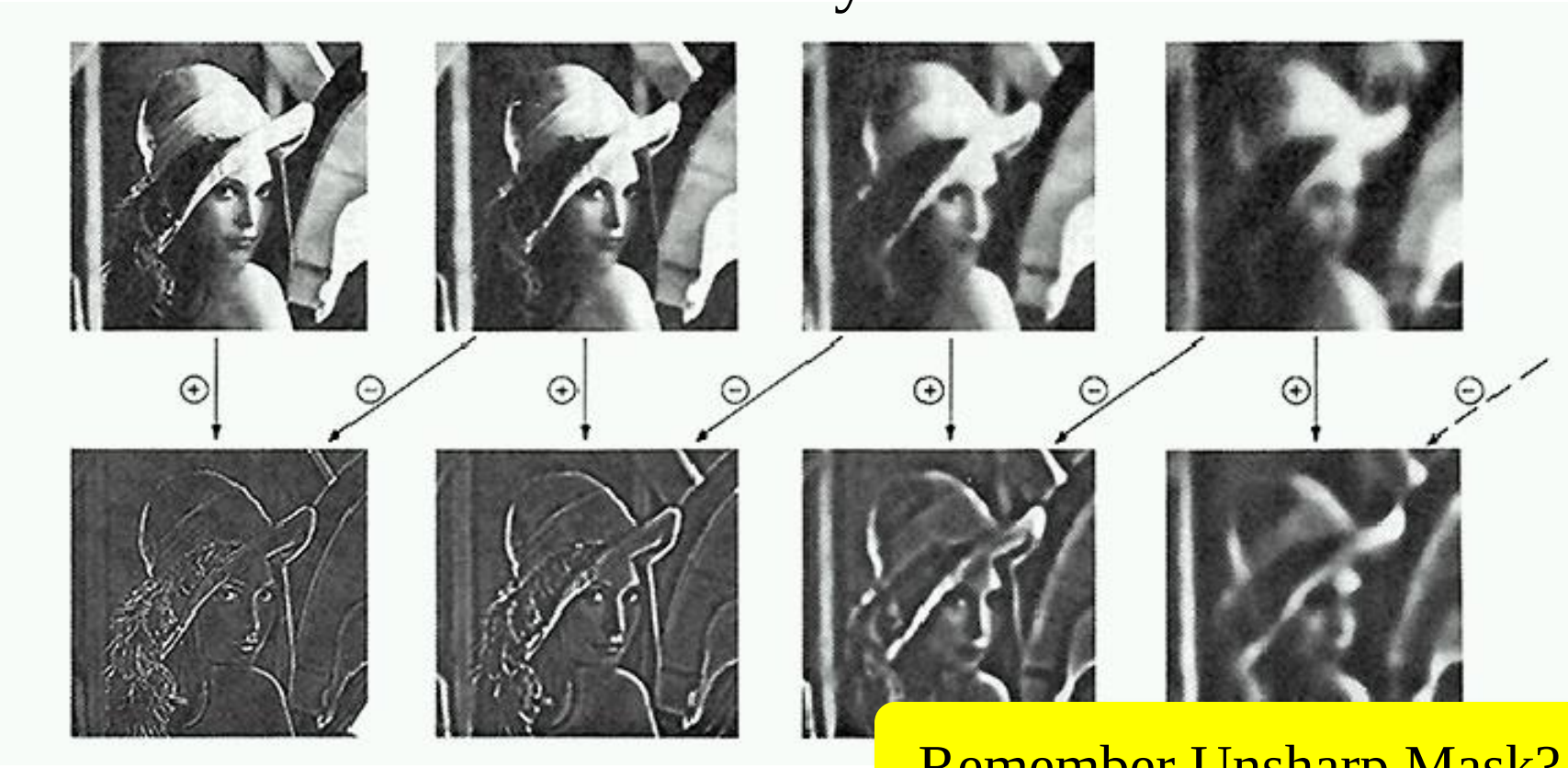
512 256 128 64 32 16 8

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



Remember Unsharp Mask?

"Laplacian" Pyramid (subband images)

- Created from Gaussian pyramid by subtraction

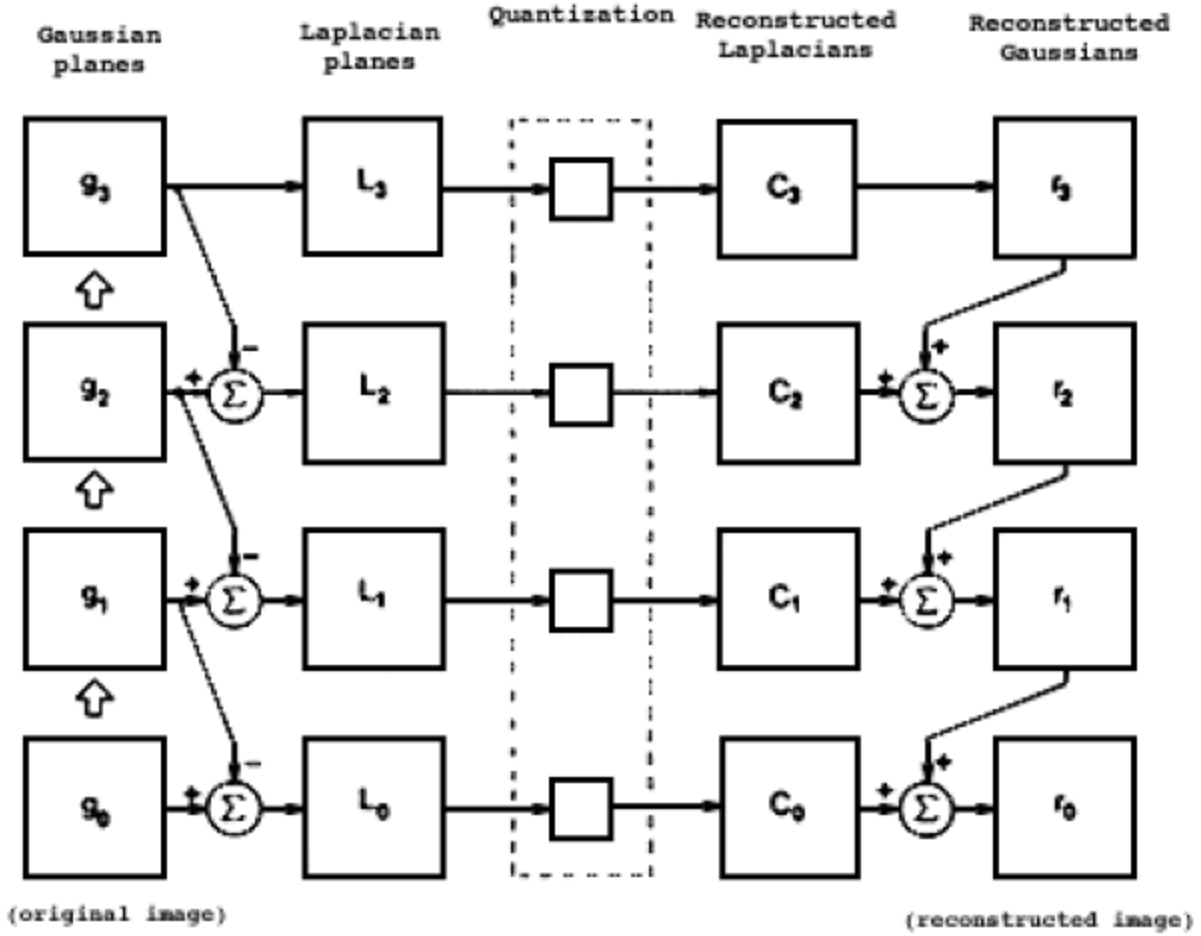
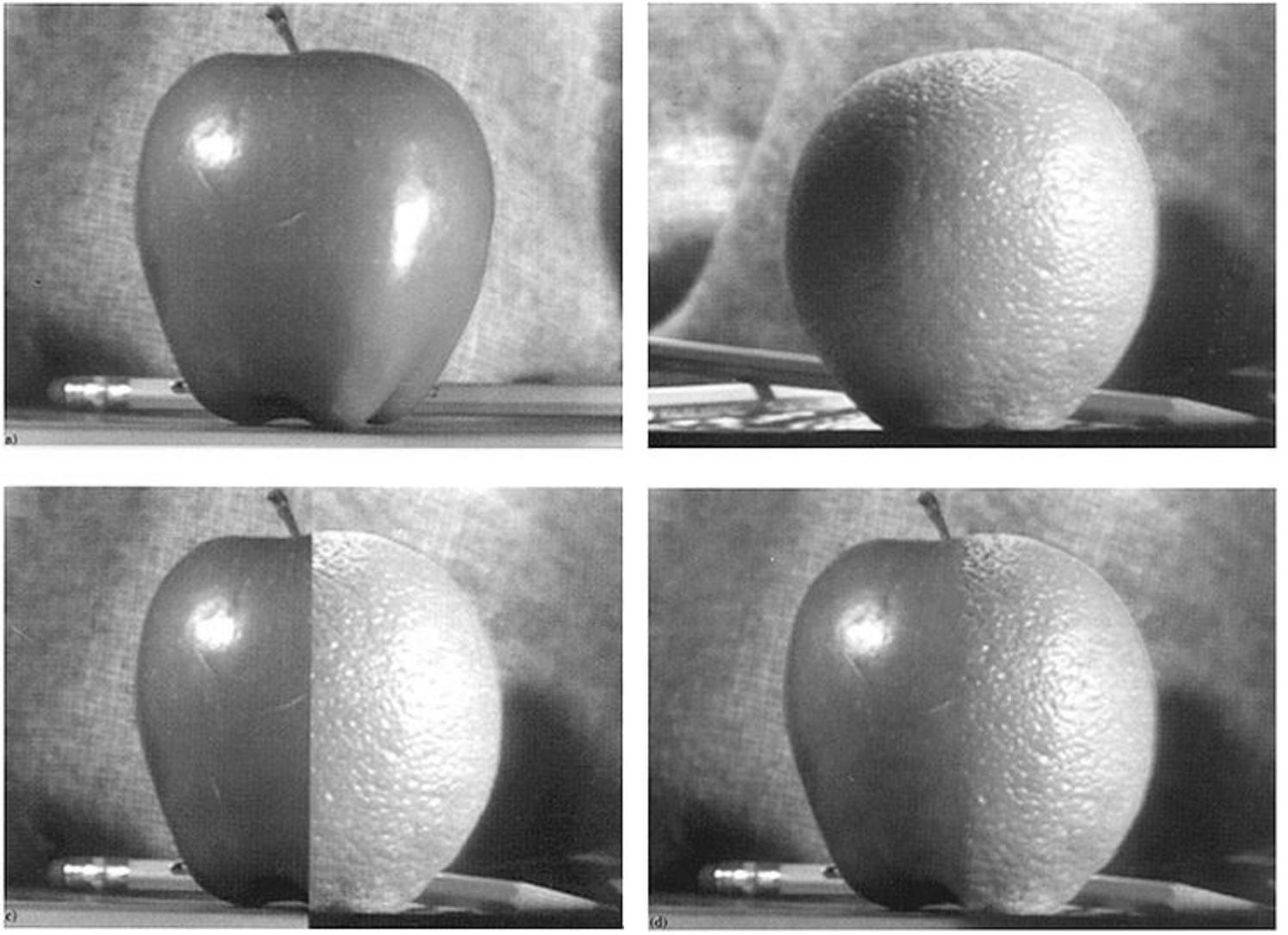


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels g_1, g_2, … through repeated local averaging. Levels of the Laplacian pyramid L_0, L_1, … are then computed as the differences between adjacent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

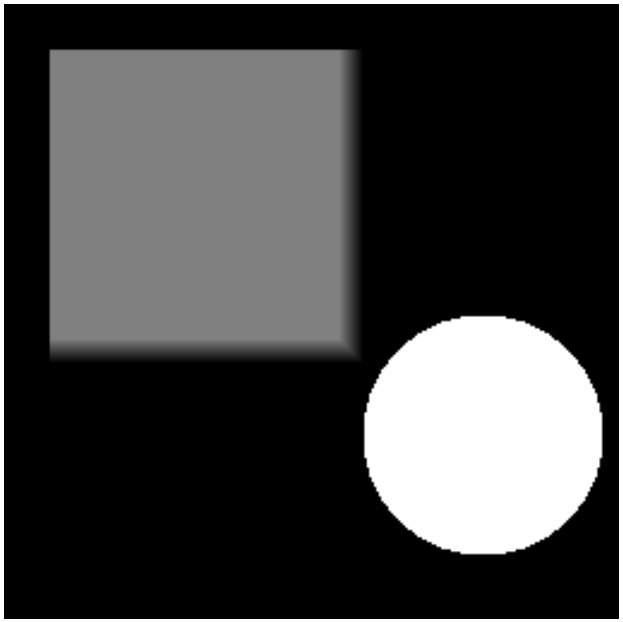


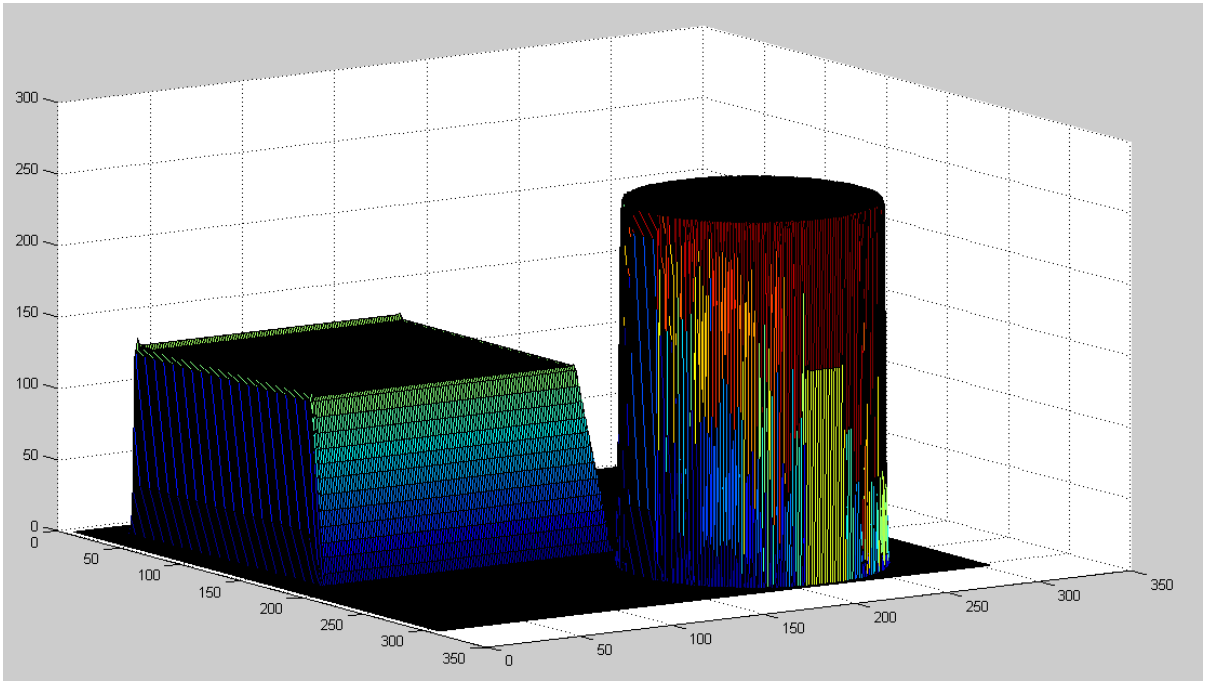
Photos + implementation by Rob Orr

Photos + implementation by Rob Orr

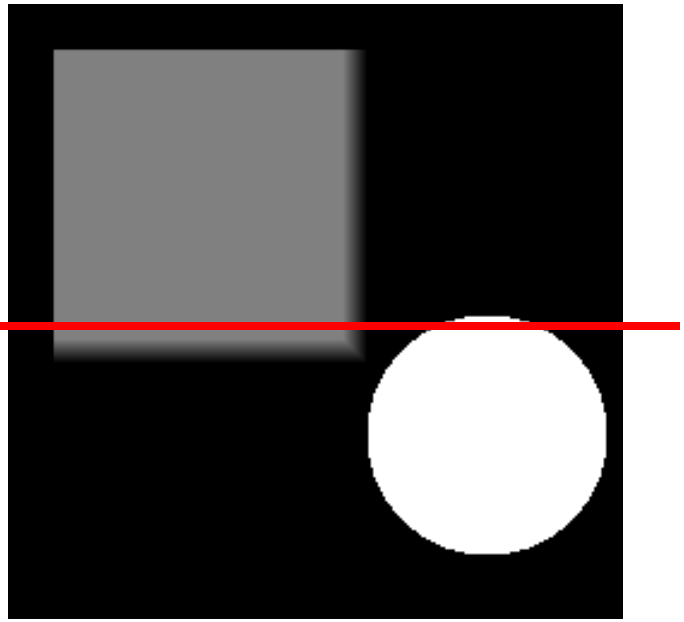
Photos + implementation by Rob Orr

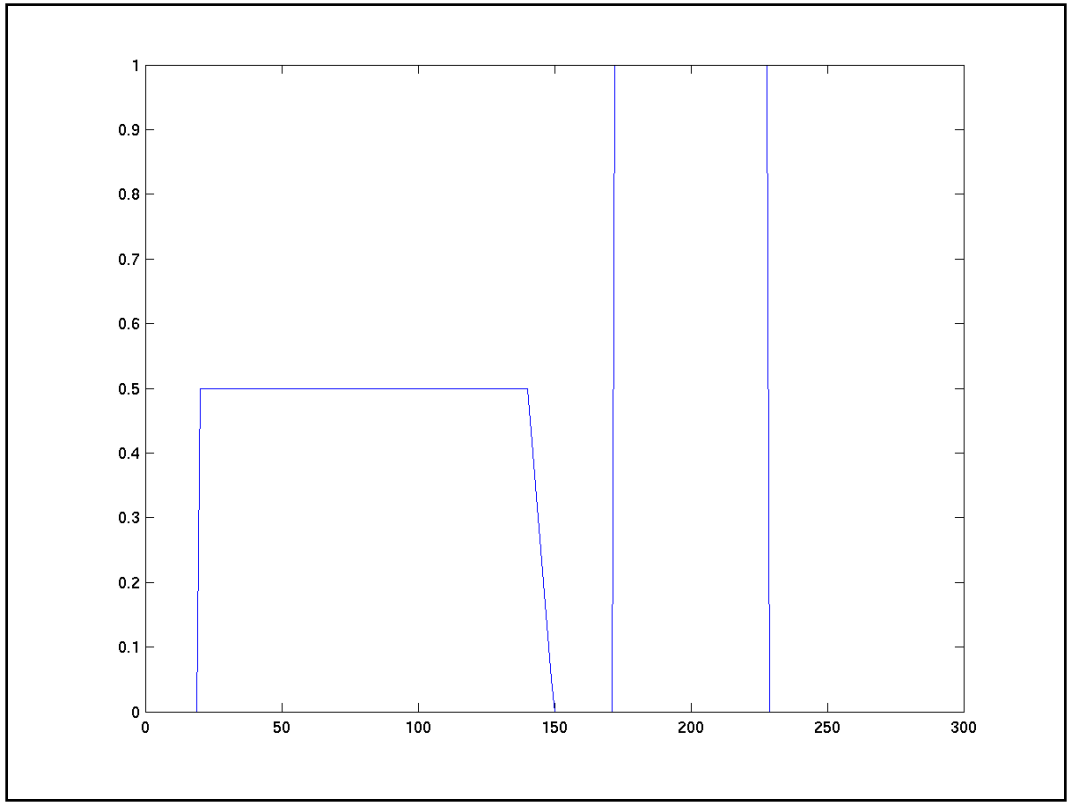
Step and Ramp Edges





Step and Ramp Edges

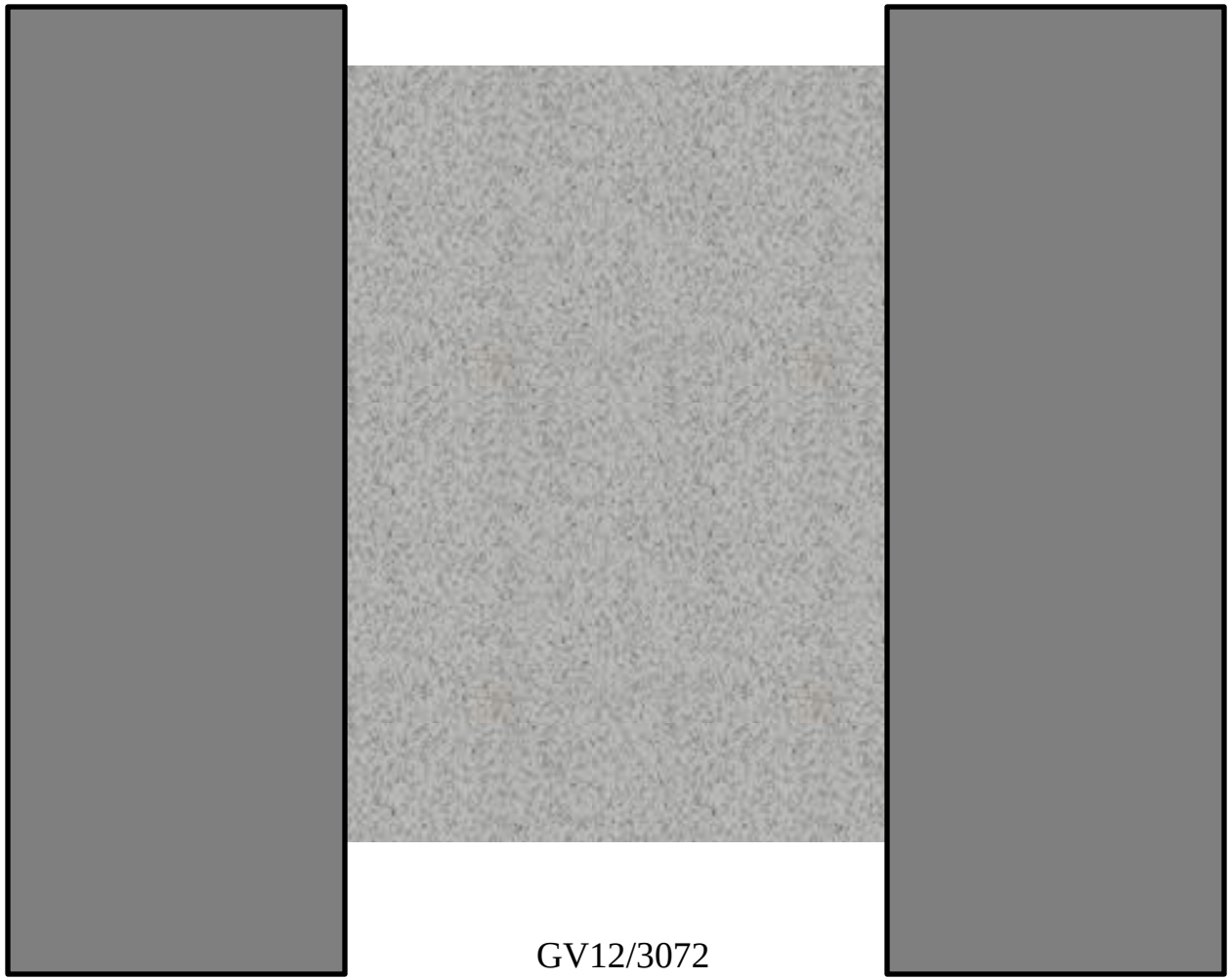




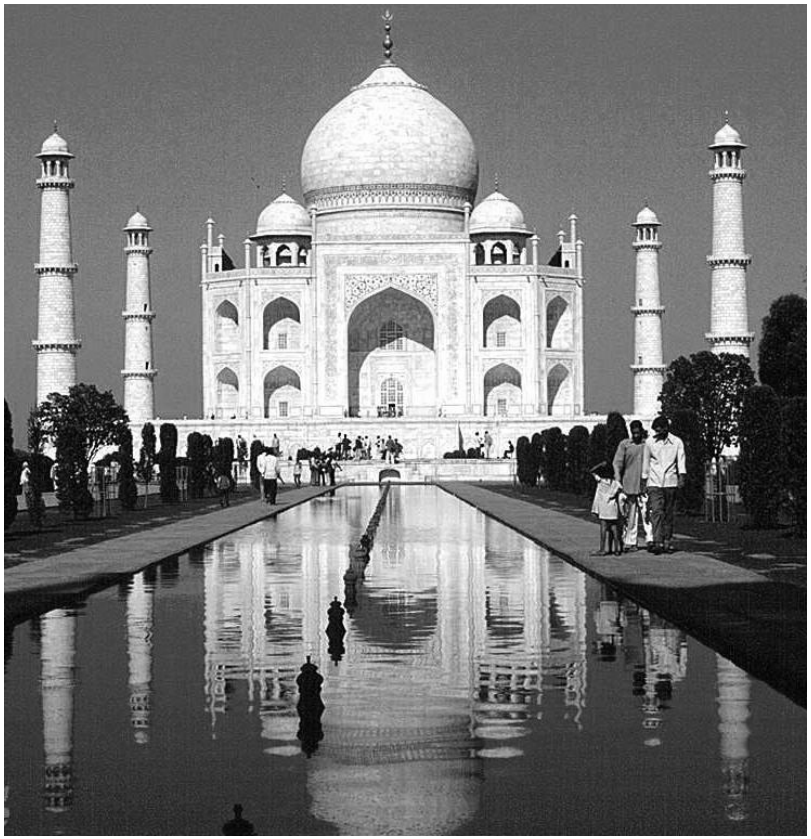
What looks like an edge?

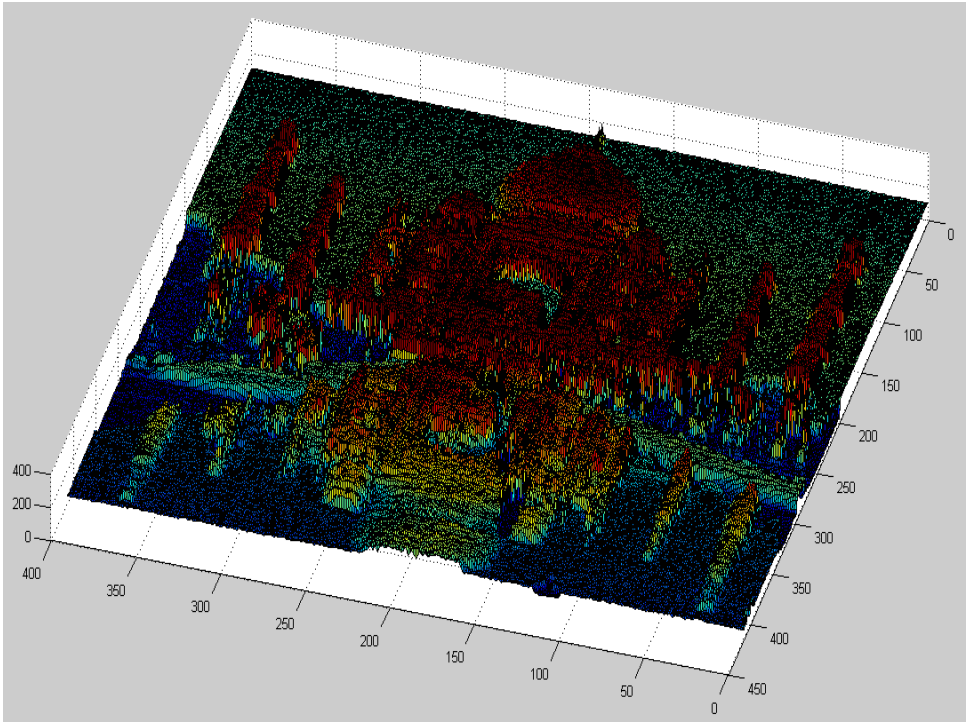
- Boundaries between regions in images:
  - Material change
  - Occlusion boundary
  - Crease boundaries
  - Shadow boundaries
- Sharp changes of gray level: Texture
- (Motion boundaries)

Motion Boundaries

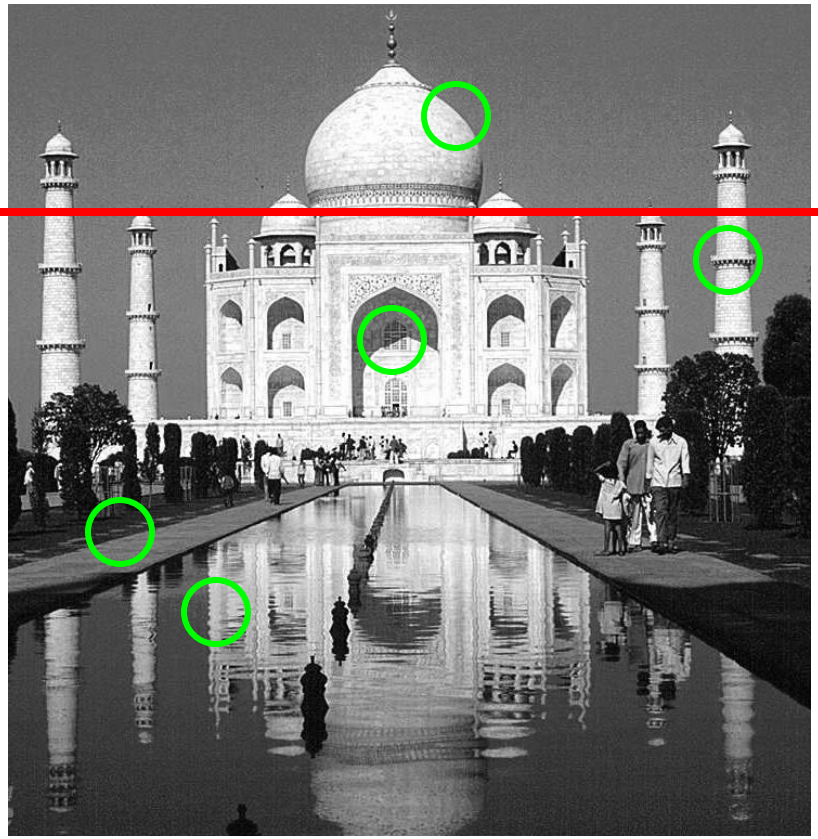


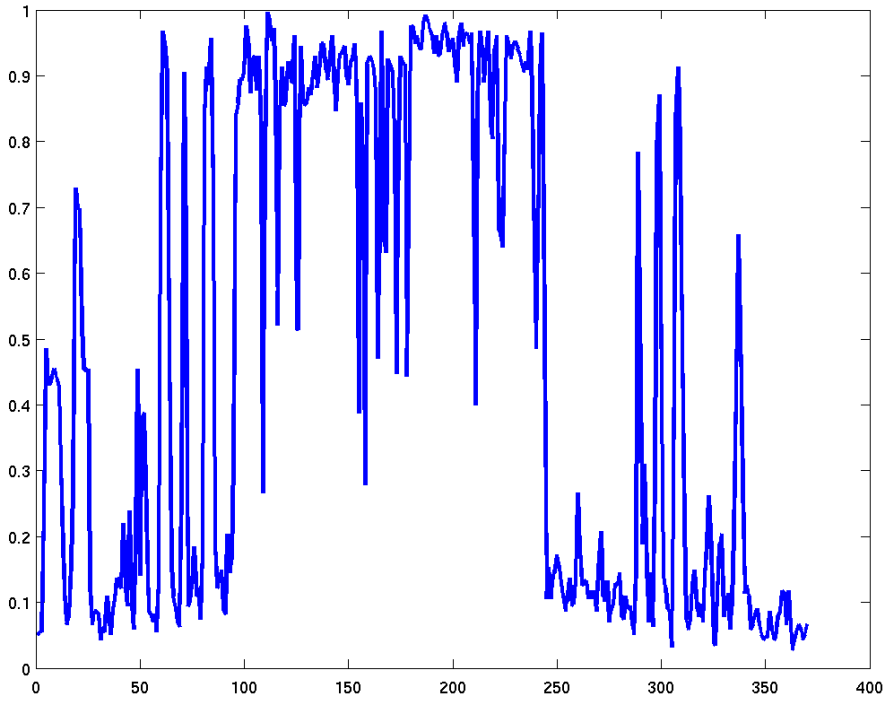
Real Edges




400
200
0
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300
350
400
450

Real Edges





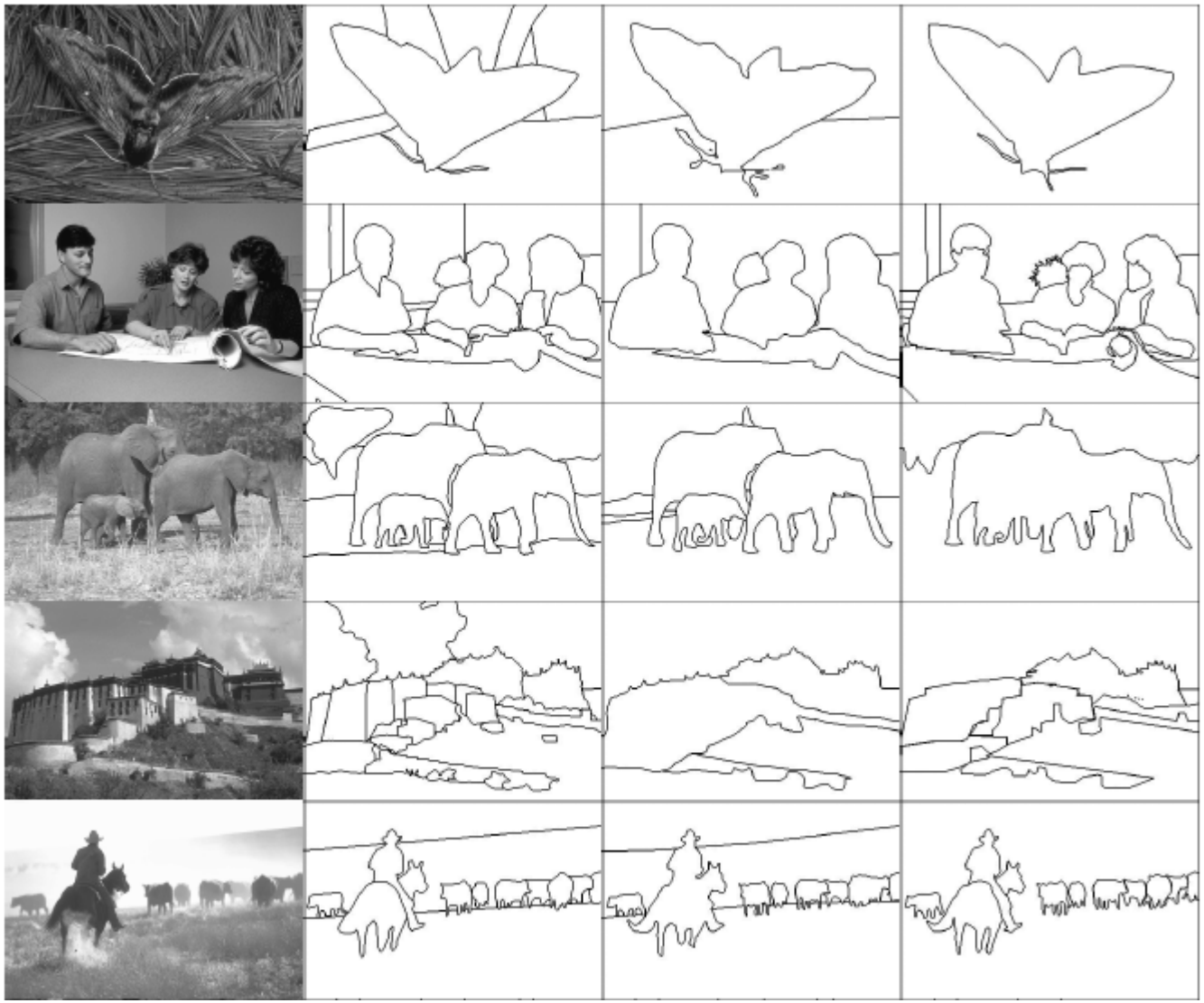
Applications

- 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

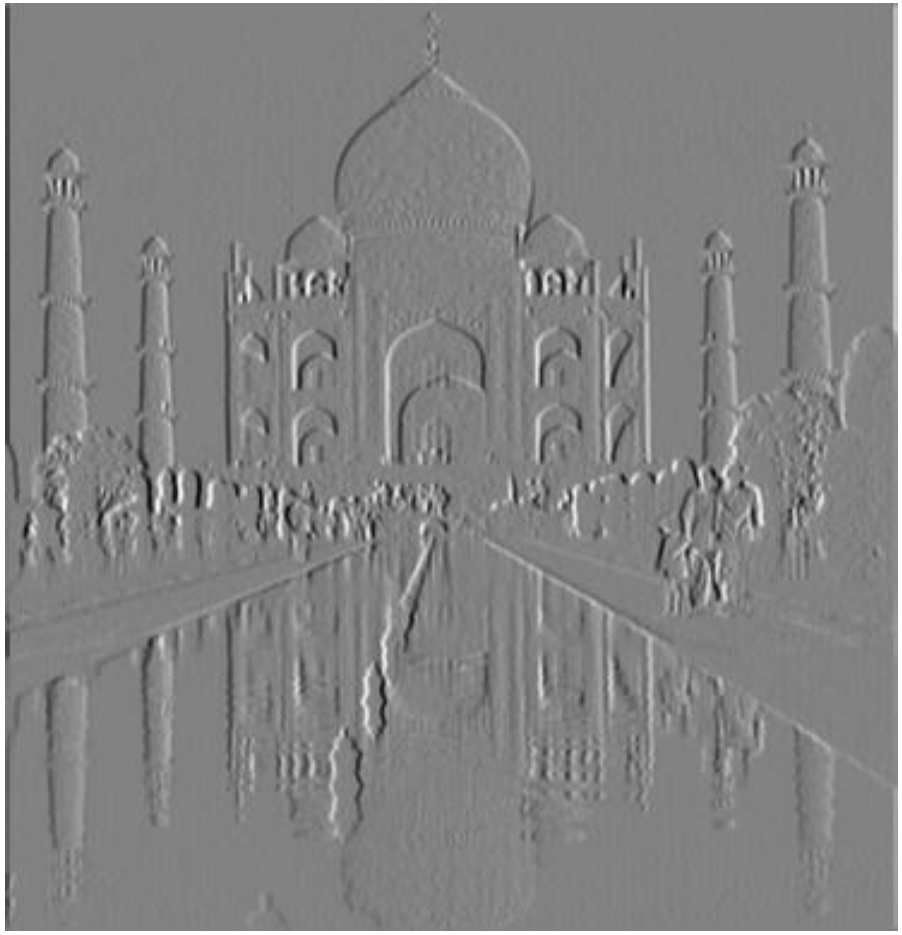
- Prewitt kernels

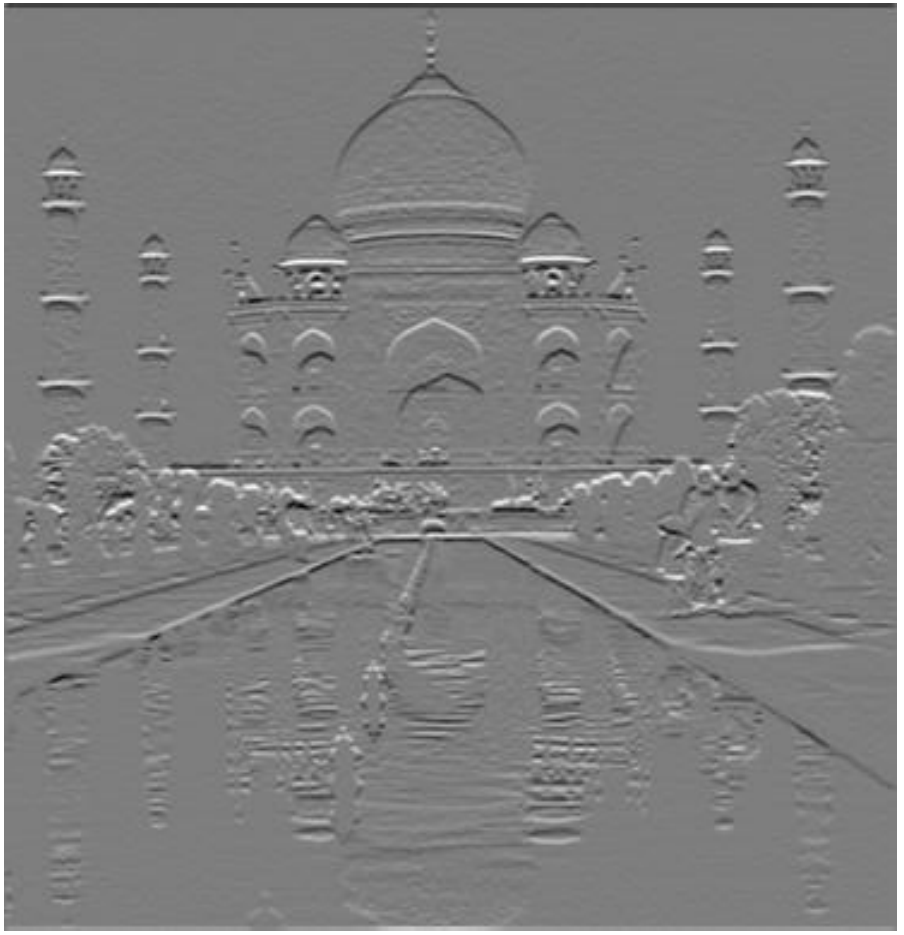
$$k = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- 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

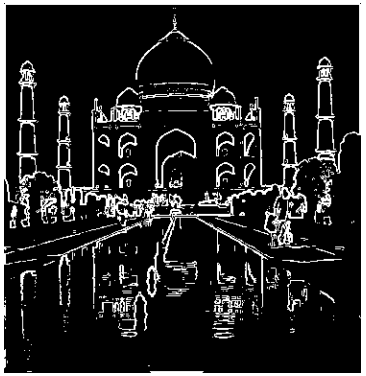
$$\mathbf{g}(x,y) = \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} (k_x * f)(x,y) \\ (k_y * f)(x,y) \end{bmatrix}$$

- Gradient magnitude and direction are:

$$|\mathbf{g}| = \left(g_x^2 + g_y^2\right)^{1/2}$$

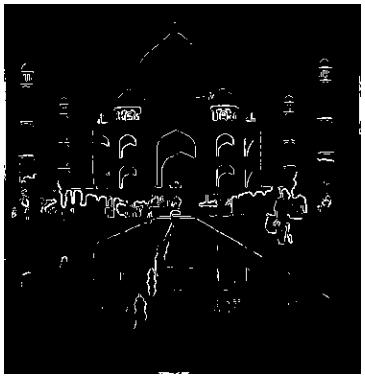
$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Simple: Edge Map



T=0.25

T=0.5



T=0.75

One more pair of kernels

- Robert's Cross Operator:

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

Canny Edge Detector

- Combine noise reduction and edge enhancement.

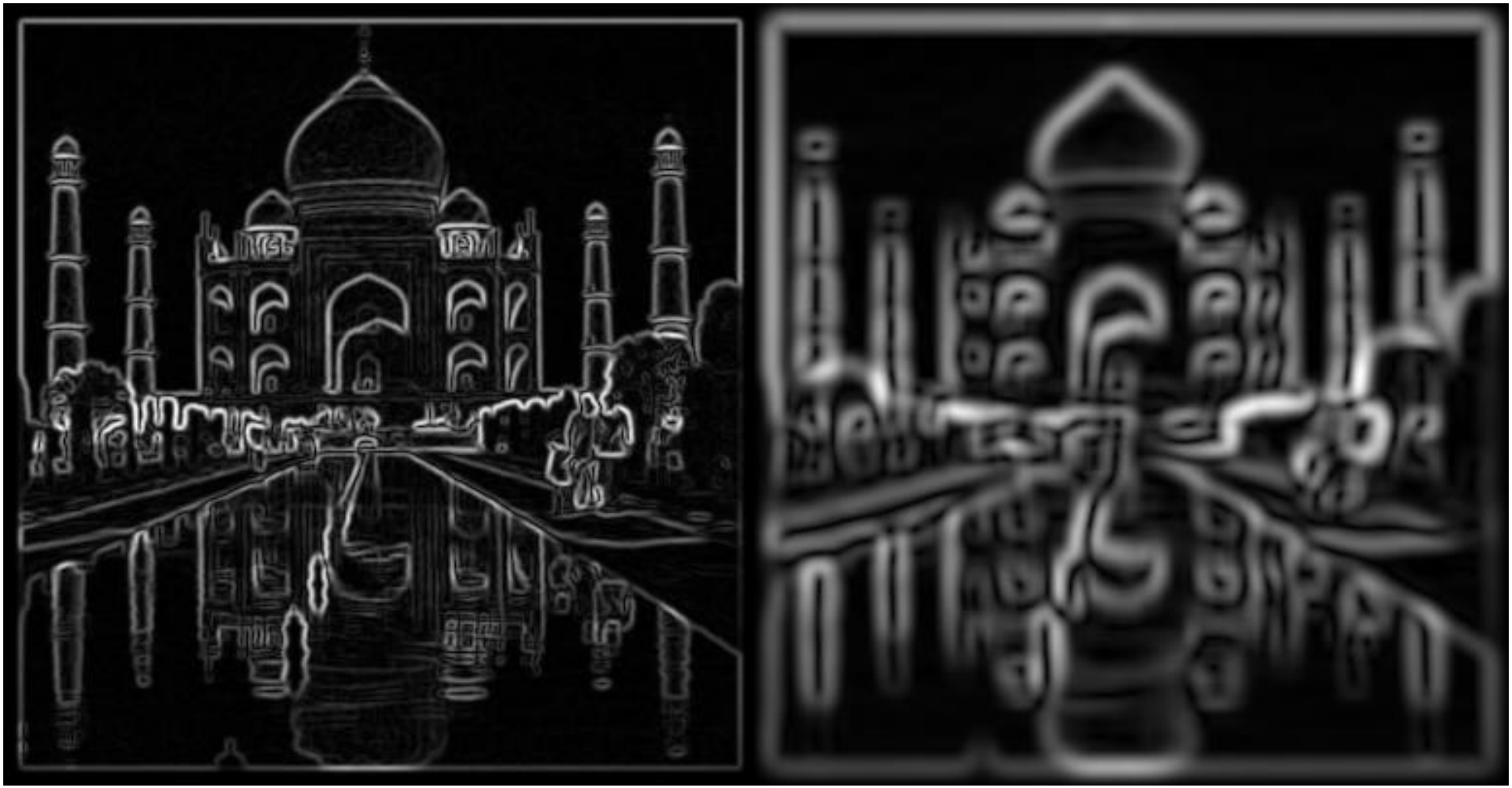
- Two-step edge localization
  - Non-maximal suppression
  - Hysteresis thresholding

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



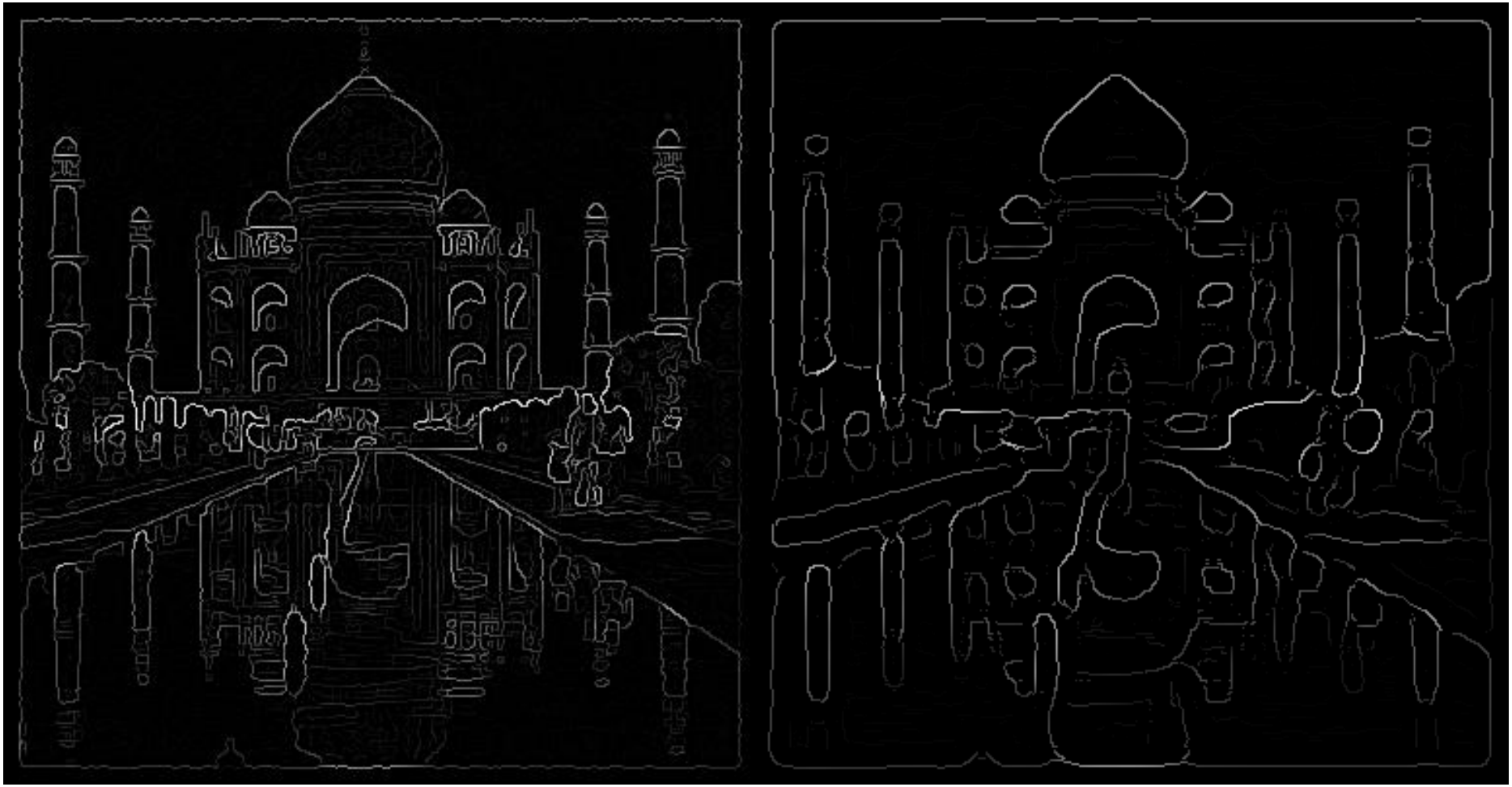
σ=1

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

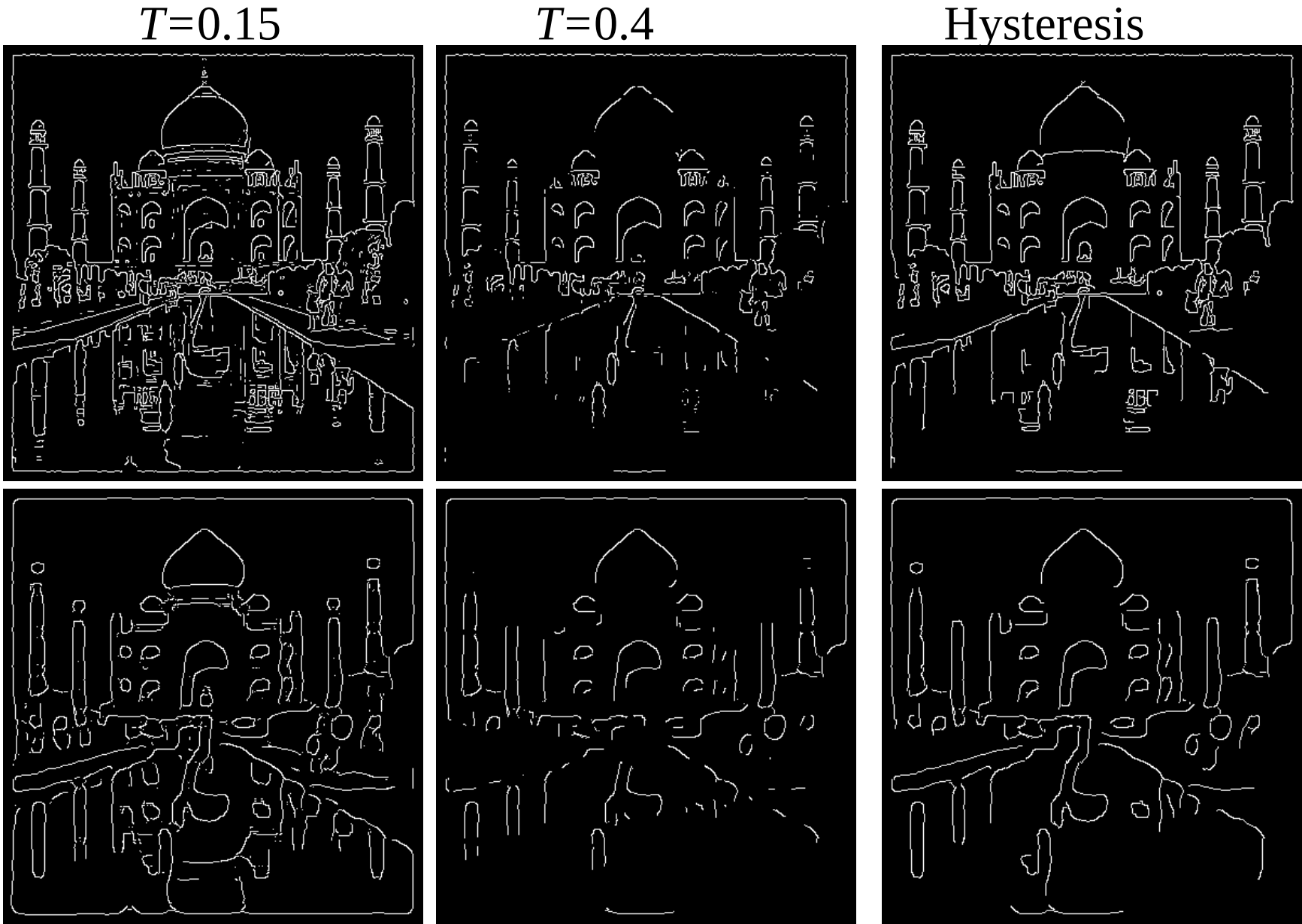


σ=1 σ=5

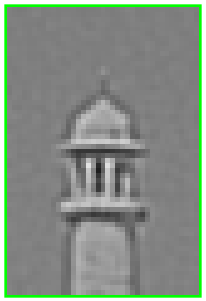
Canny: Hysteresis Tresholding

- 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

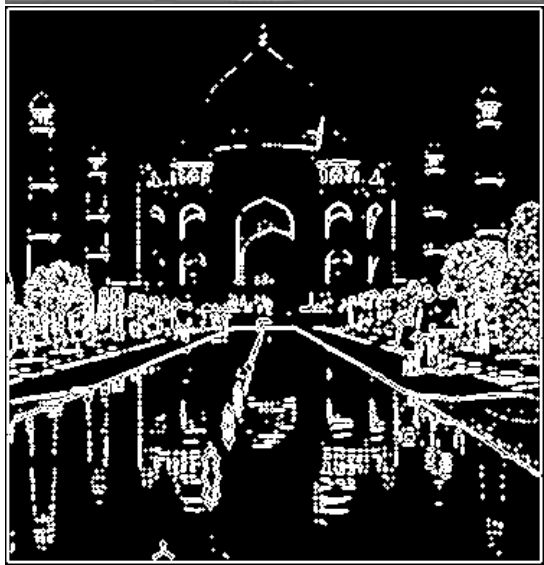


Marr-Hildreth Edge Detector



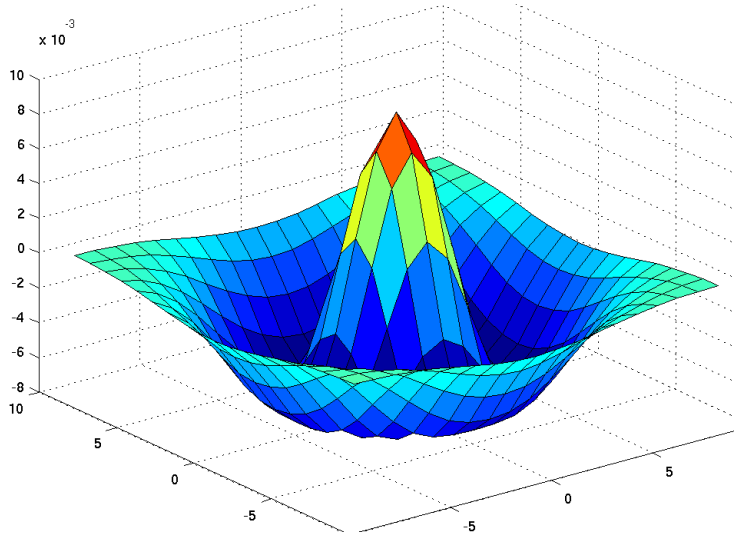
```
k=LoG(3,1);

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

- Convolve with second derivative operator
  - Laplacian of Gaussian
  - $k(x,y) = \nabla^2 G(x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) G(x,y)$
- Find zero crossings
- Sensitive to noise.

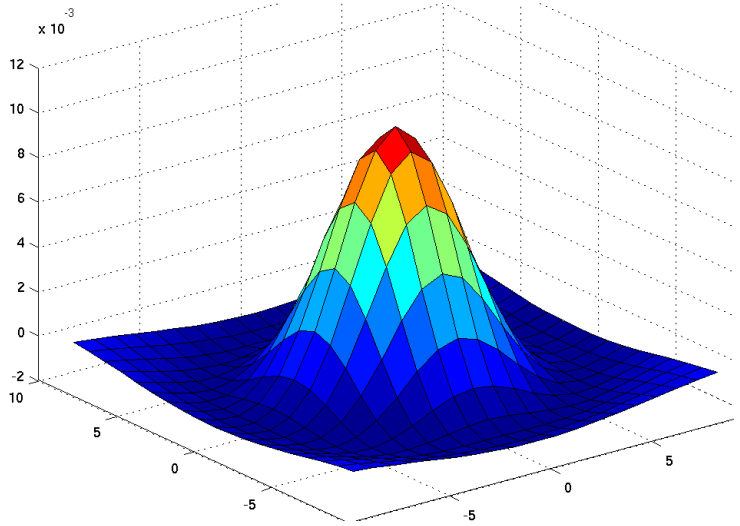


LoG vs DoG

- Laplacian sometimes approximated by difference of two Gaussian filters:




```
k=LoG(9,3);

surf(x,y,-k);
```




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

surf(x,y,-k);
```

Model Fitting

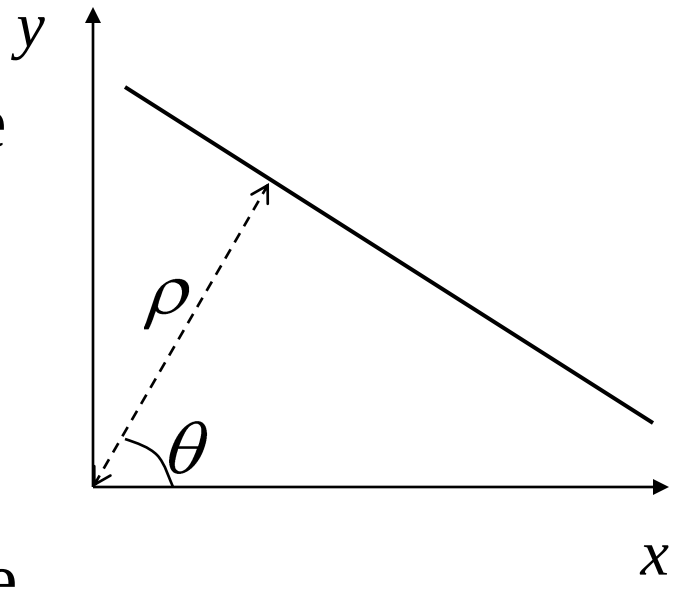
- Locate edges by fitting a surface.
- Create a model edge, eg step edge with parameters:
  - Orientation
  - Position
  - Intensities either side of the edge
- Find least-squares fit in each small window.
- Accept if fit is above a threshold.

Hough Transform

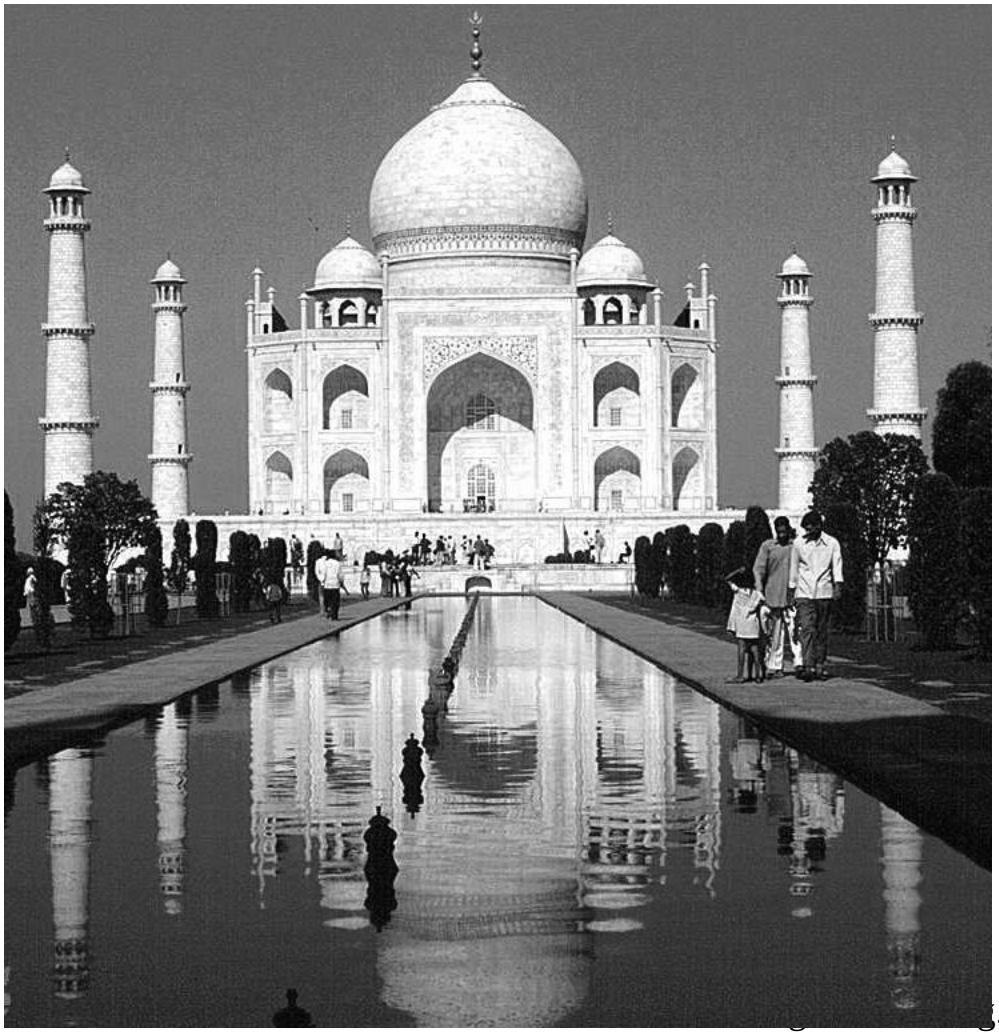
- Finds the most likely lines in an image.
- At every edge pixel, compute the local equation of the edge line:

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

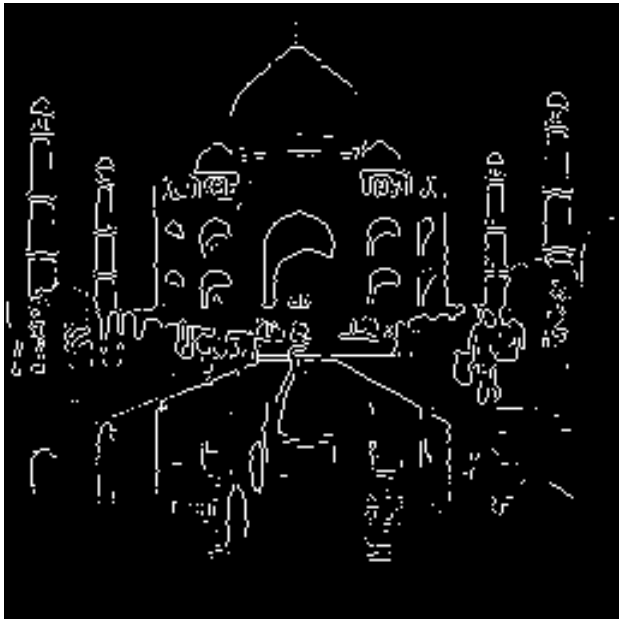
- Store a histogram of the line parameters θ and ρ.
- The fullest histogram bins are the dominant image lines.



Example

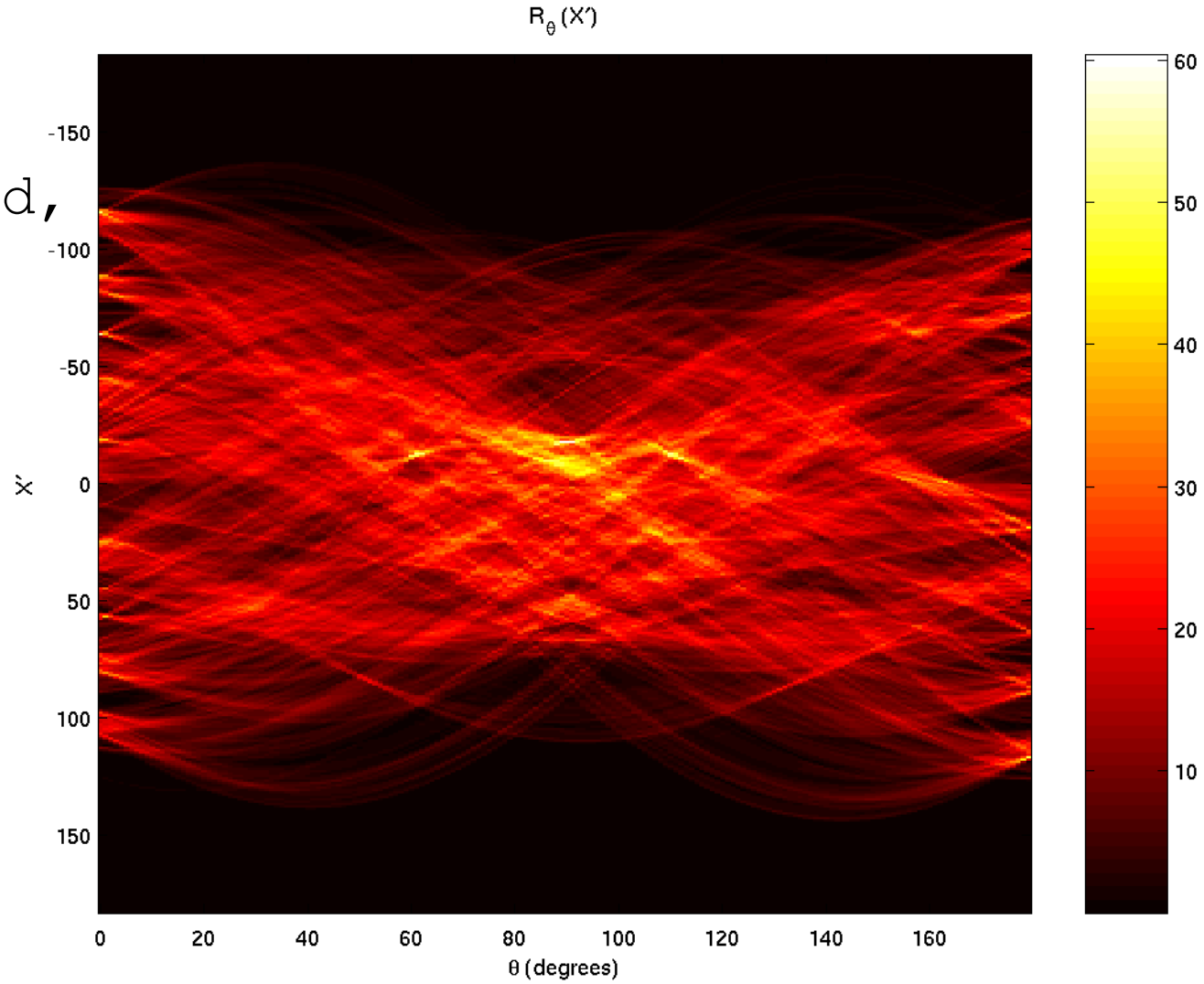


```
ed=edge(im);
```



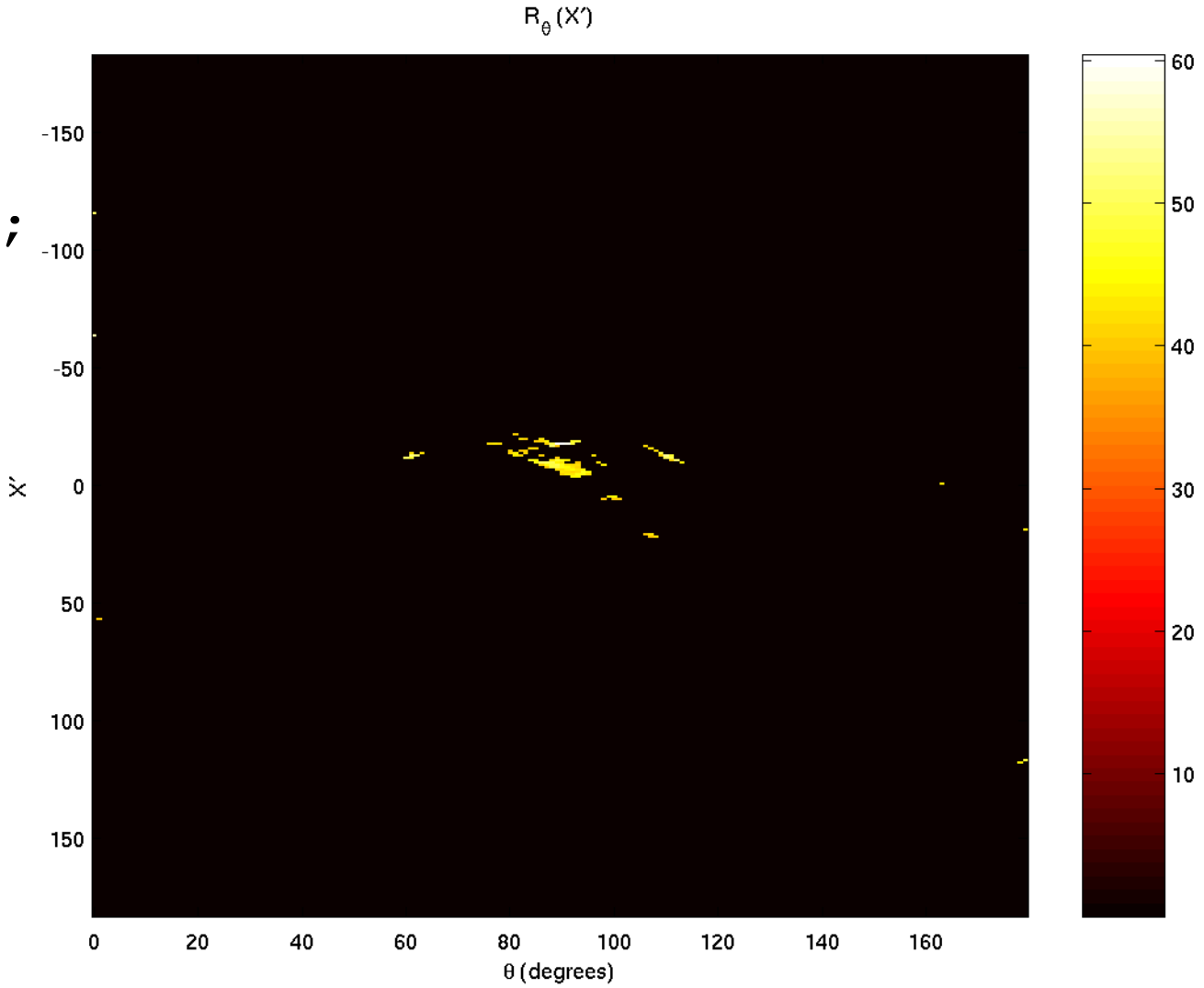
Hough Transform

```
theta=0:179;

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



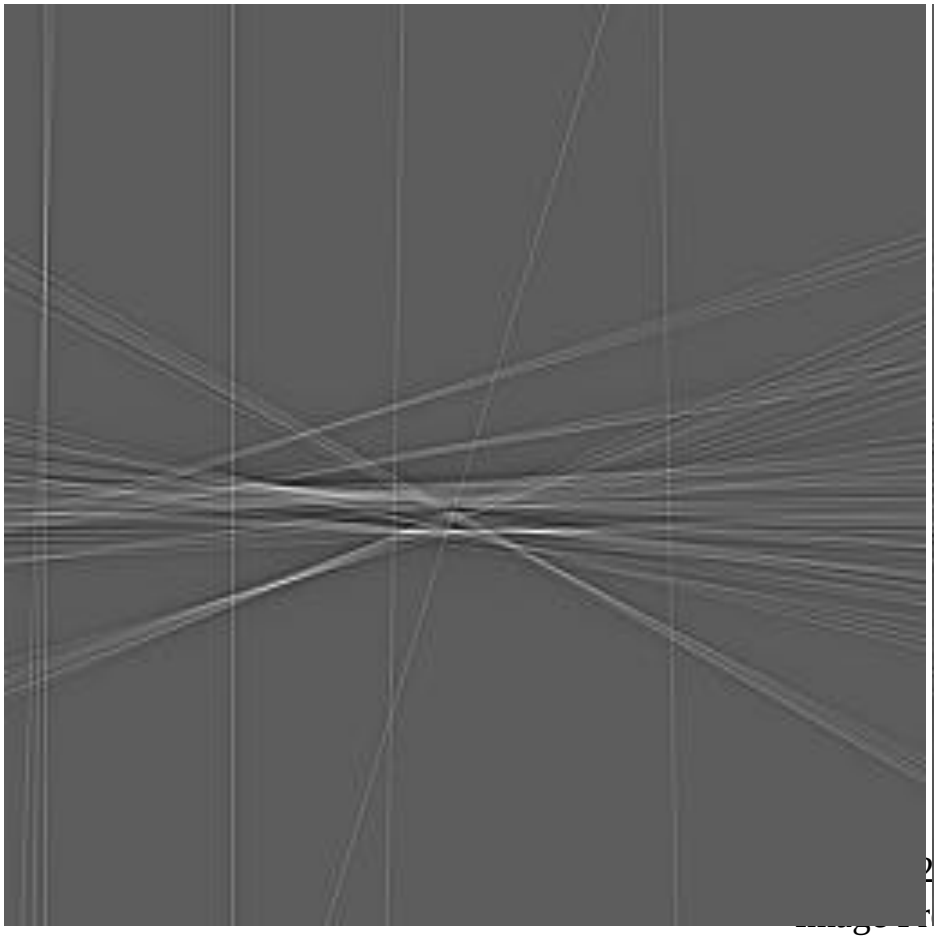
Thresholded

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



Inverted and Superimposed

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



Generalization

- The general Hough transform works with any parametric shape.
- E.g., circles: $(x - x_0)^2 + (y - y_0)^2 = r^2$
- Make a 3D histogram of x_0, y_0 and r.
- 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