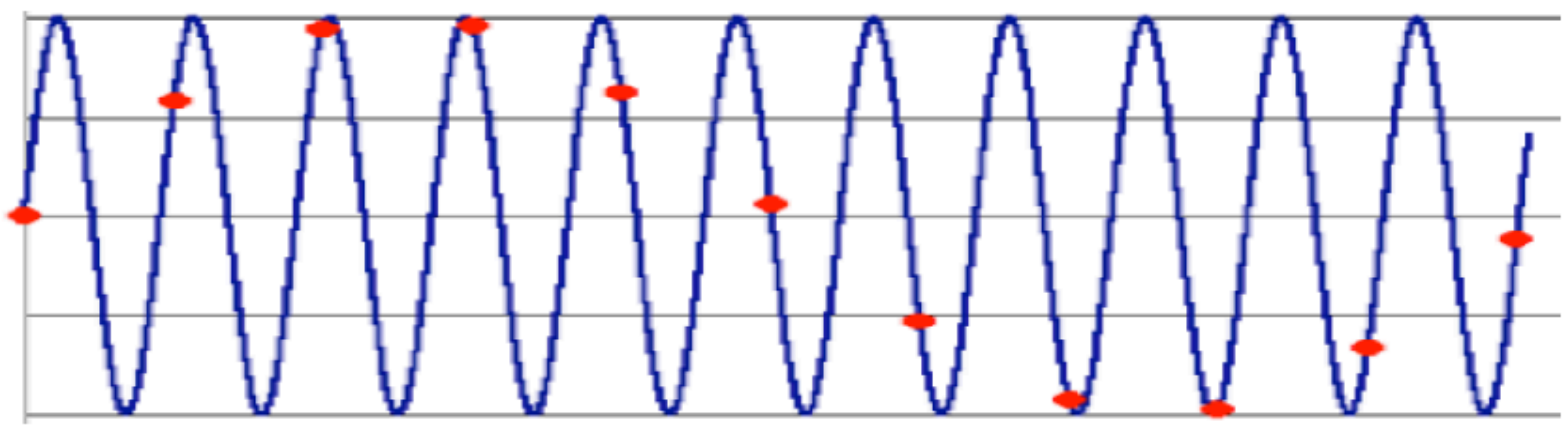
COMP0026: Image Processing

Filtering and Edge Detection

UCL

Lectures will be Recorded

Undersampling

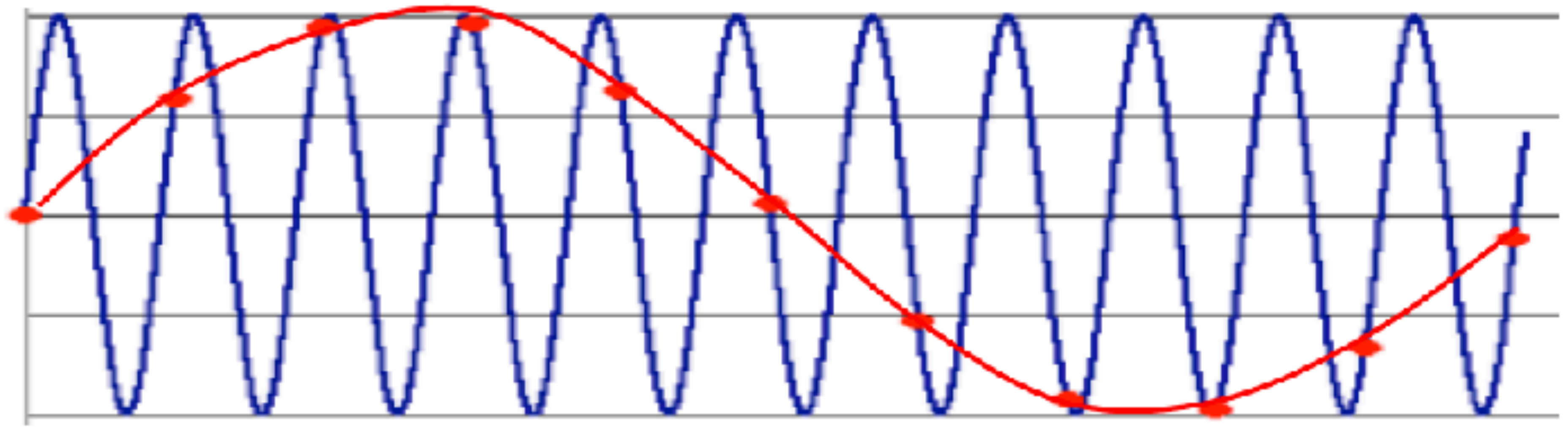


- **Aliasing** can arise when you sample a continuous signal or image
  - occurs when your sampling rate is not high enough to capture the amount of detail in your image
  - Can give you the wrong signal/image—an *alias*
  - formally, the image contains structure at different scales
    - called “frequencies”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

/naiquist/

Aliasing and Nyquist Rate

信号采样频率的下限



- **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

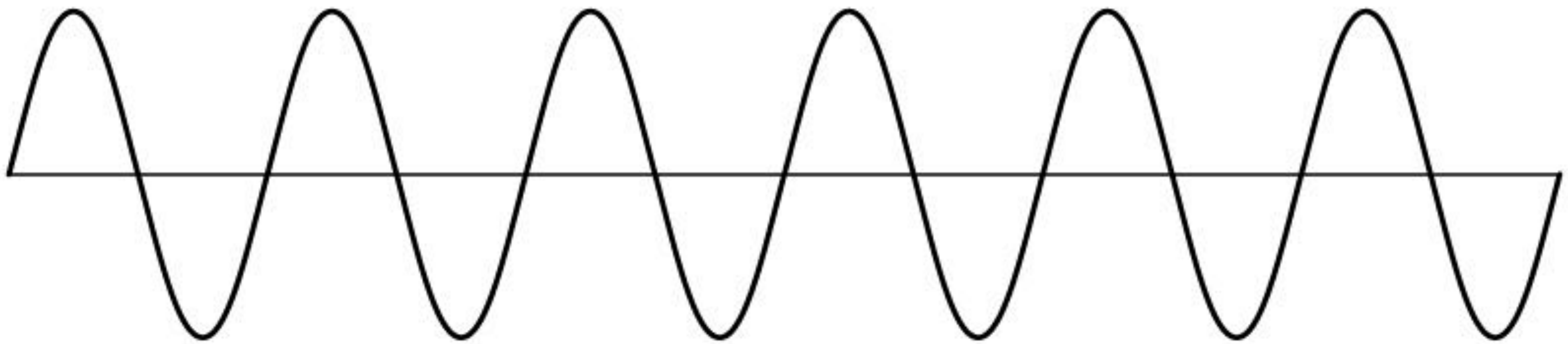
~~(和这个现象反过来~~
~~的下采样，会导致摩尔纹)~~

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.

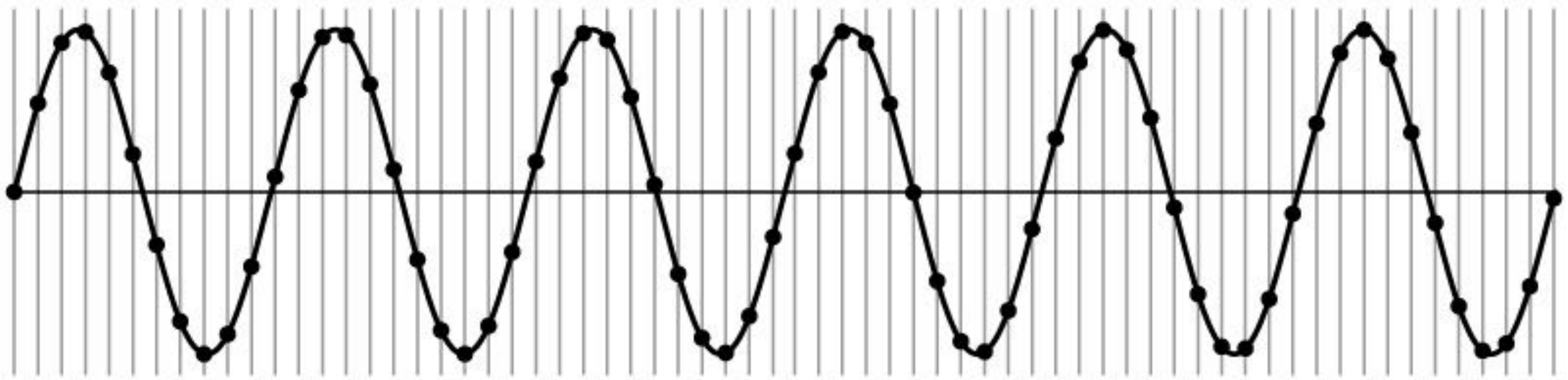
Sampling and Reconstruction

- **Simple example: a sine wave**



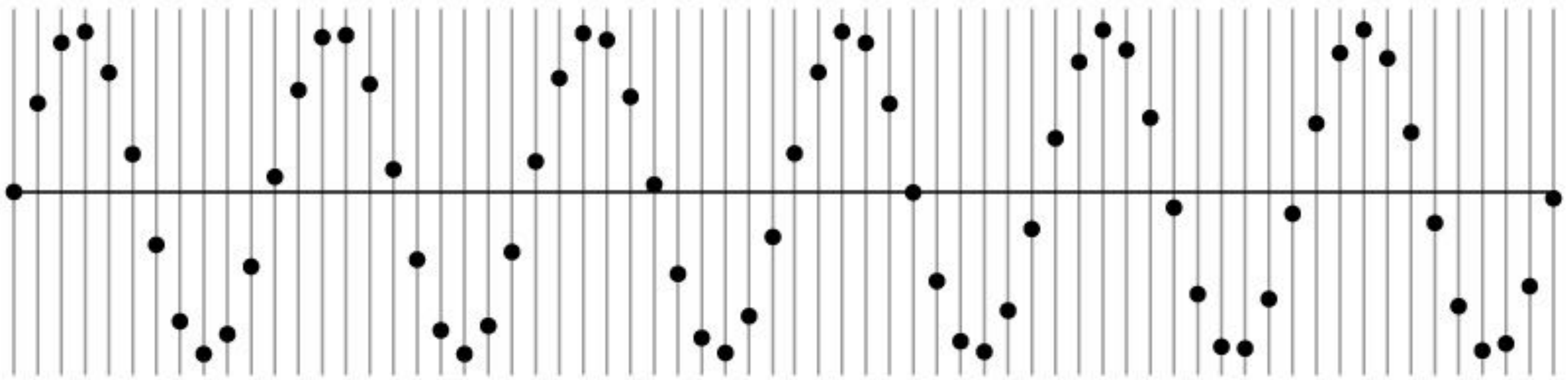
Sampling and Reconstruction

- **Simple example: a sine wave**



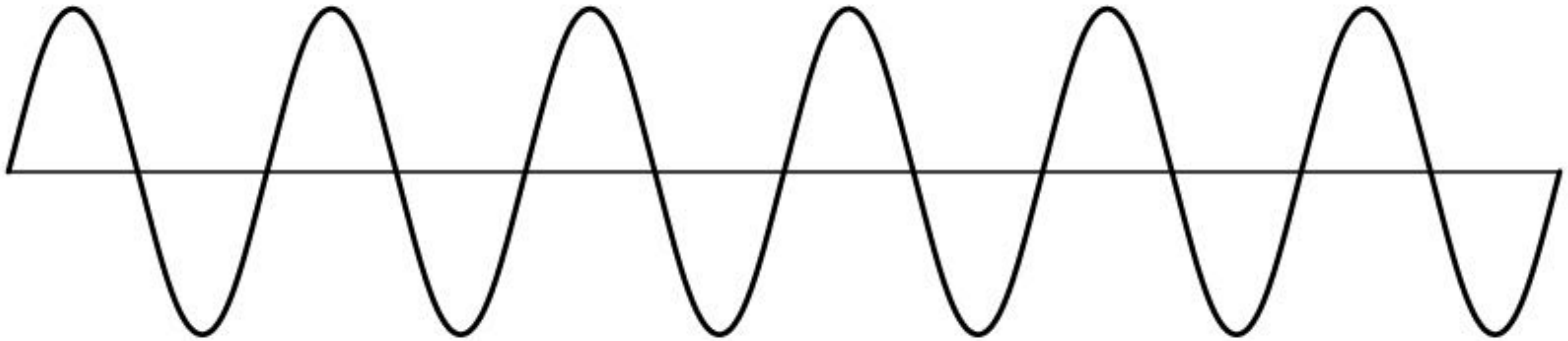
Sampling and Reconstruction

- **Simple example: a sine wave**



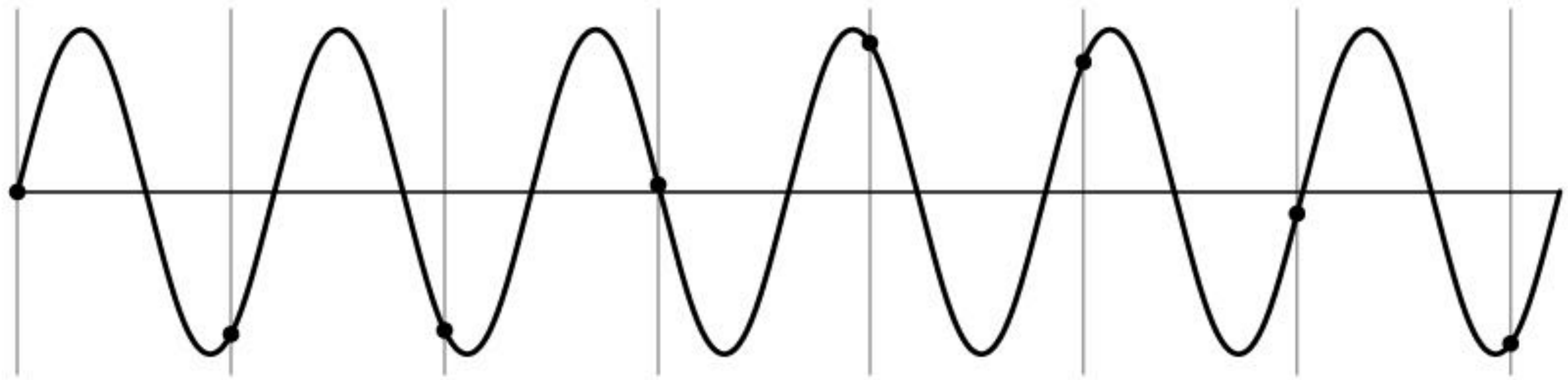
Sampling and Reconstruction

- Simple example: a sine wave



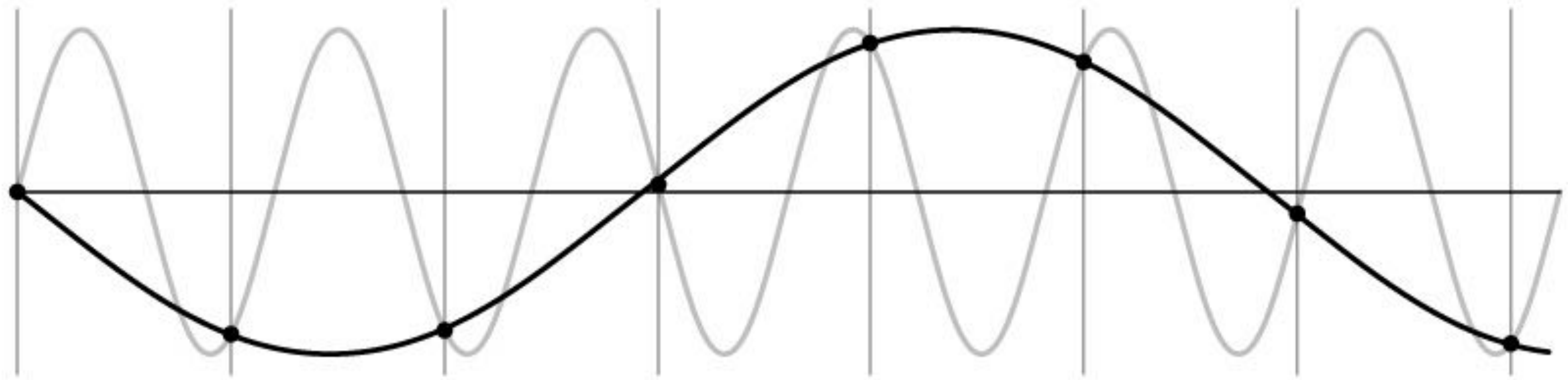
Undersampling

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



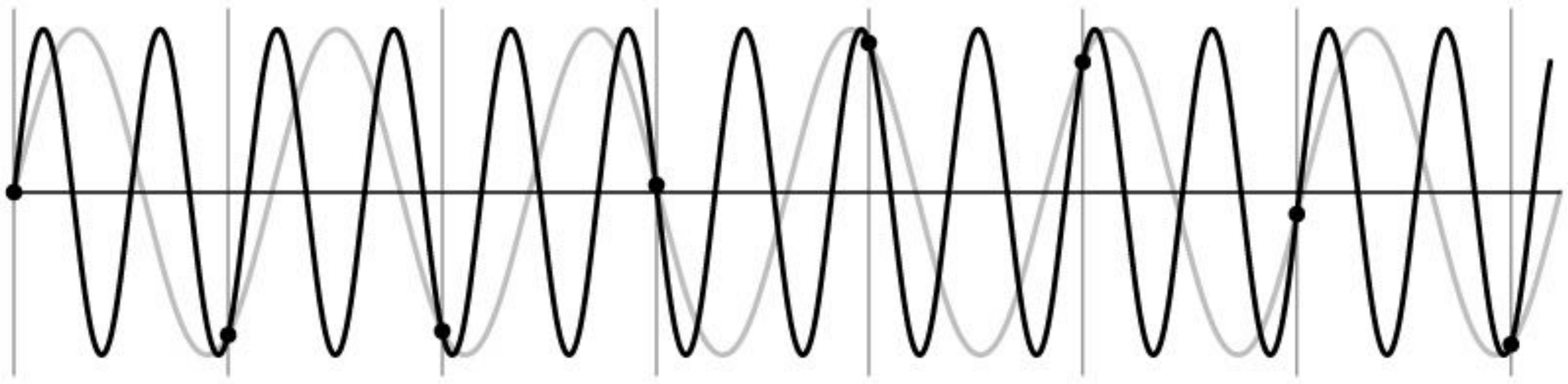
Undersampling

- 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

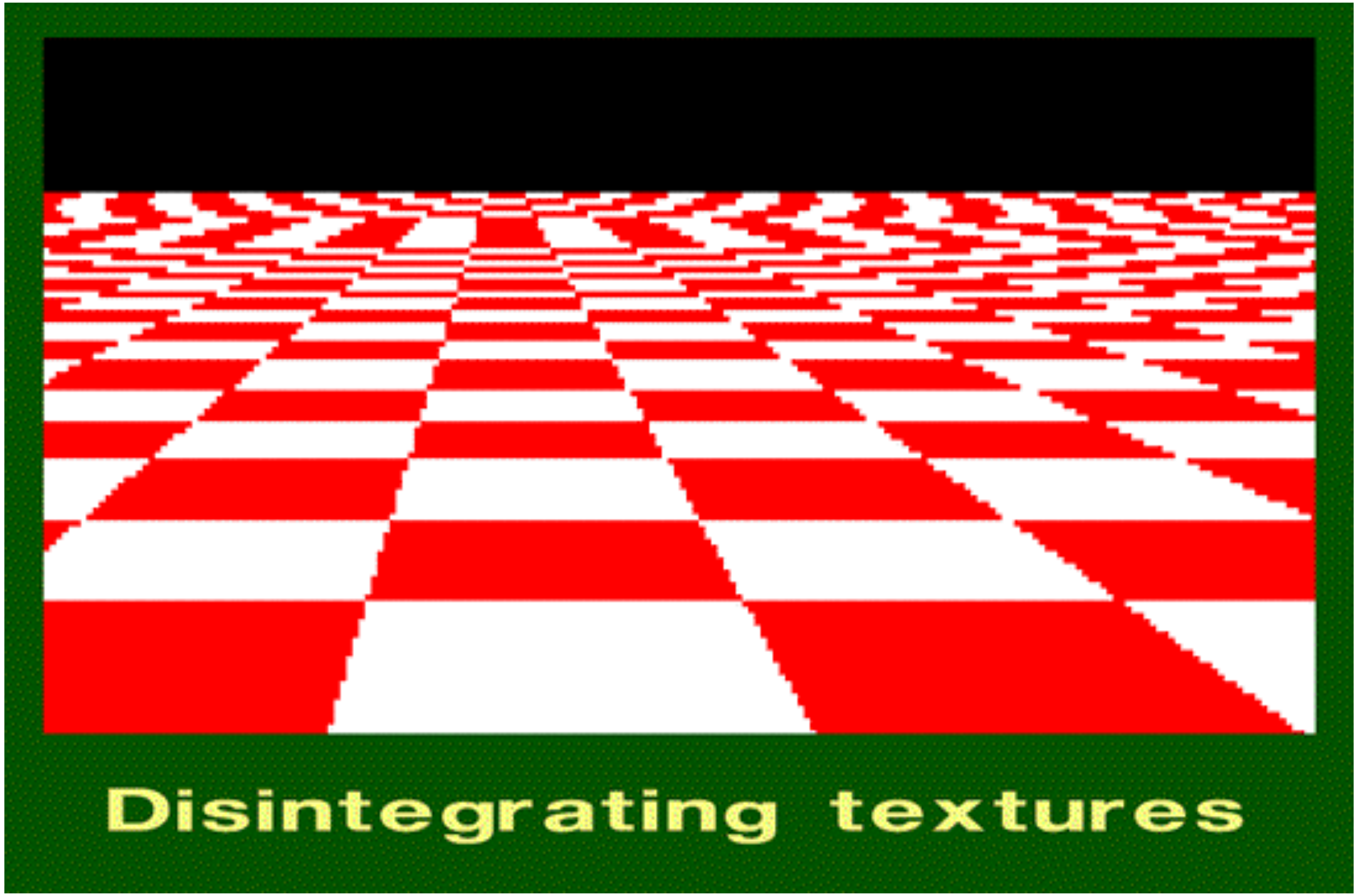


Undersampling

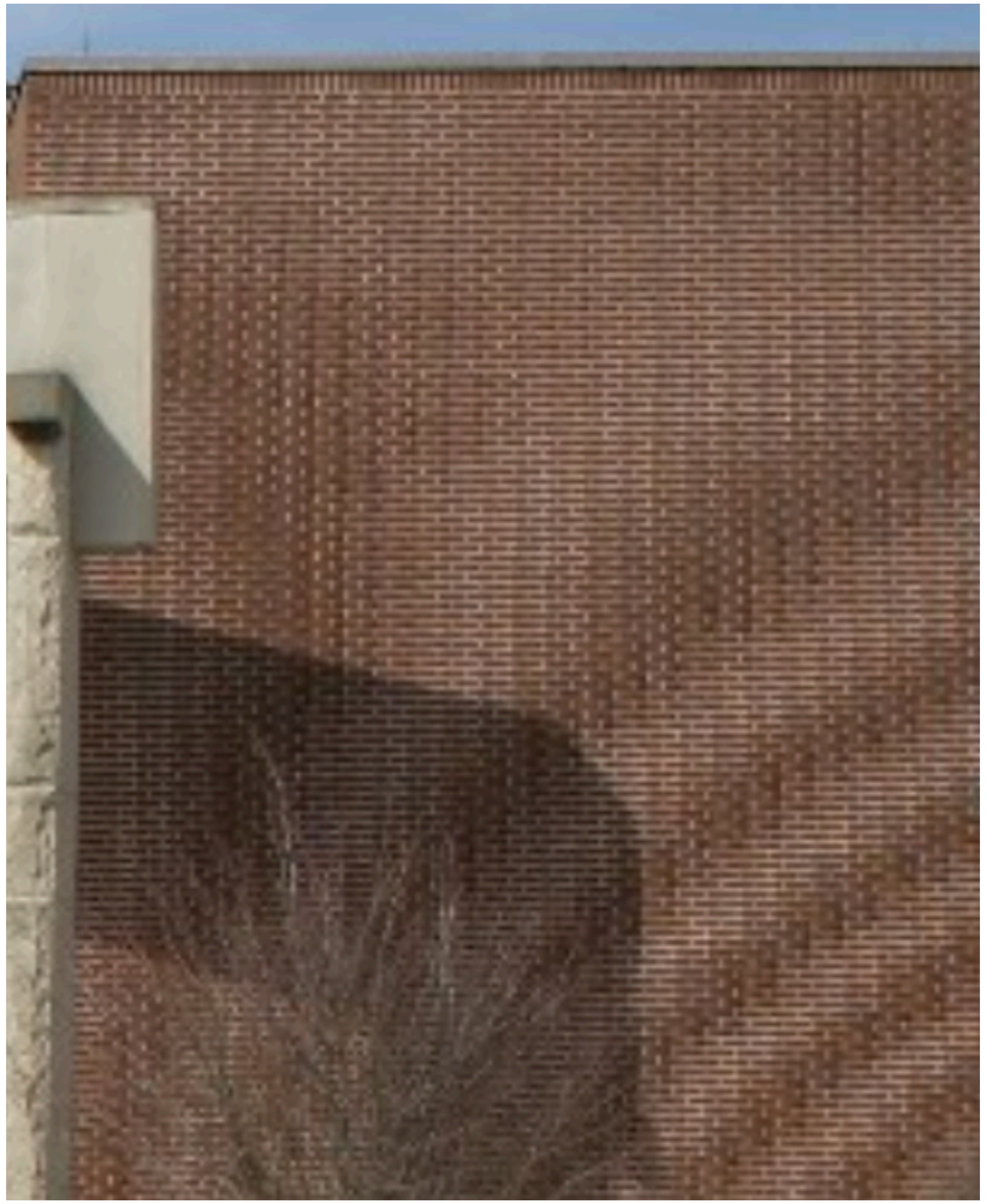
- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals "traveling in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



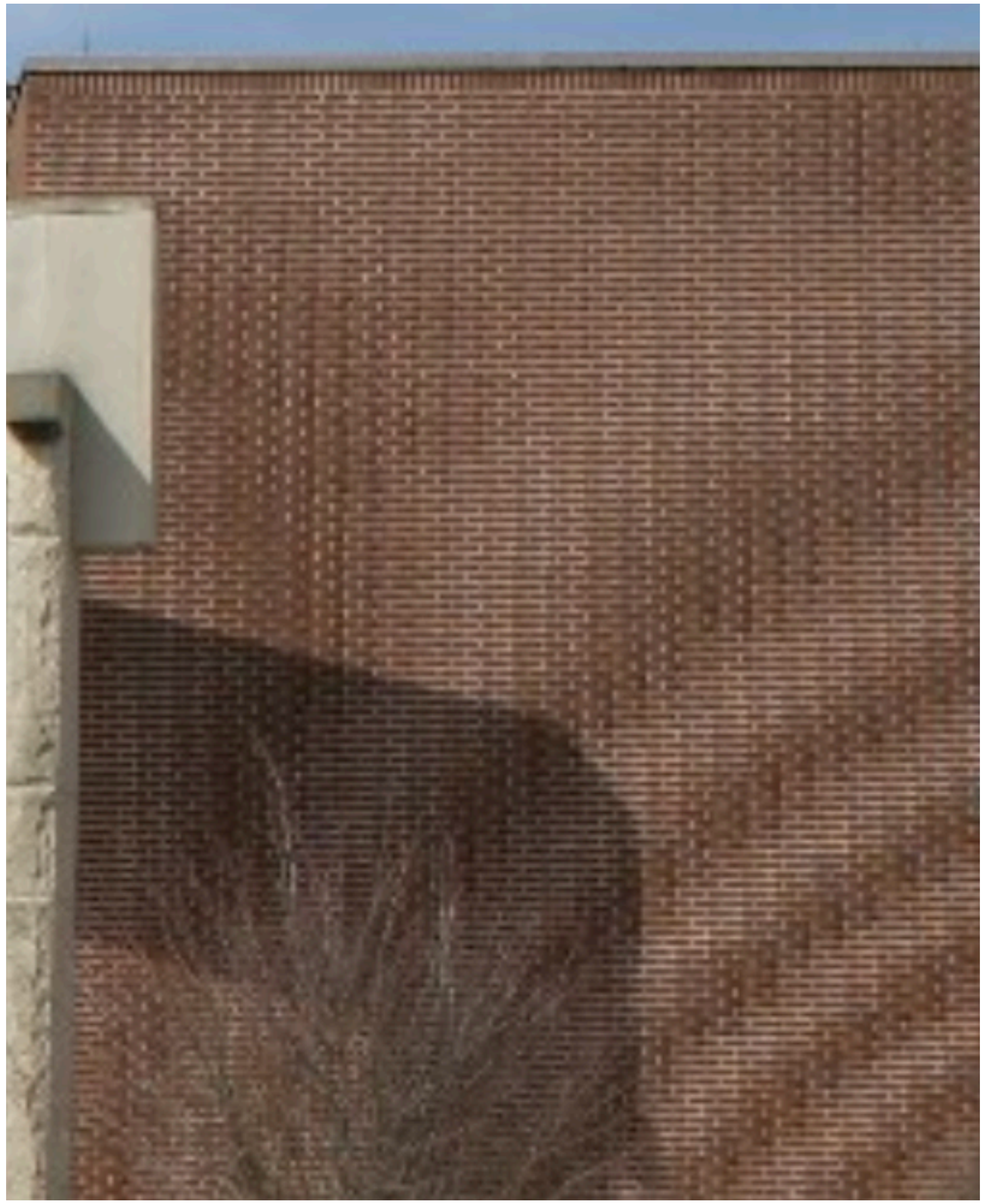
Aliasing in Images

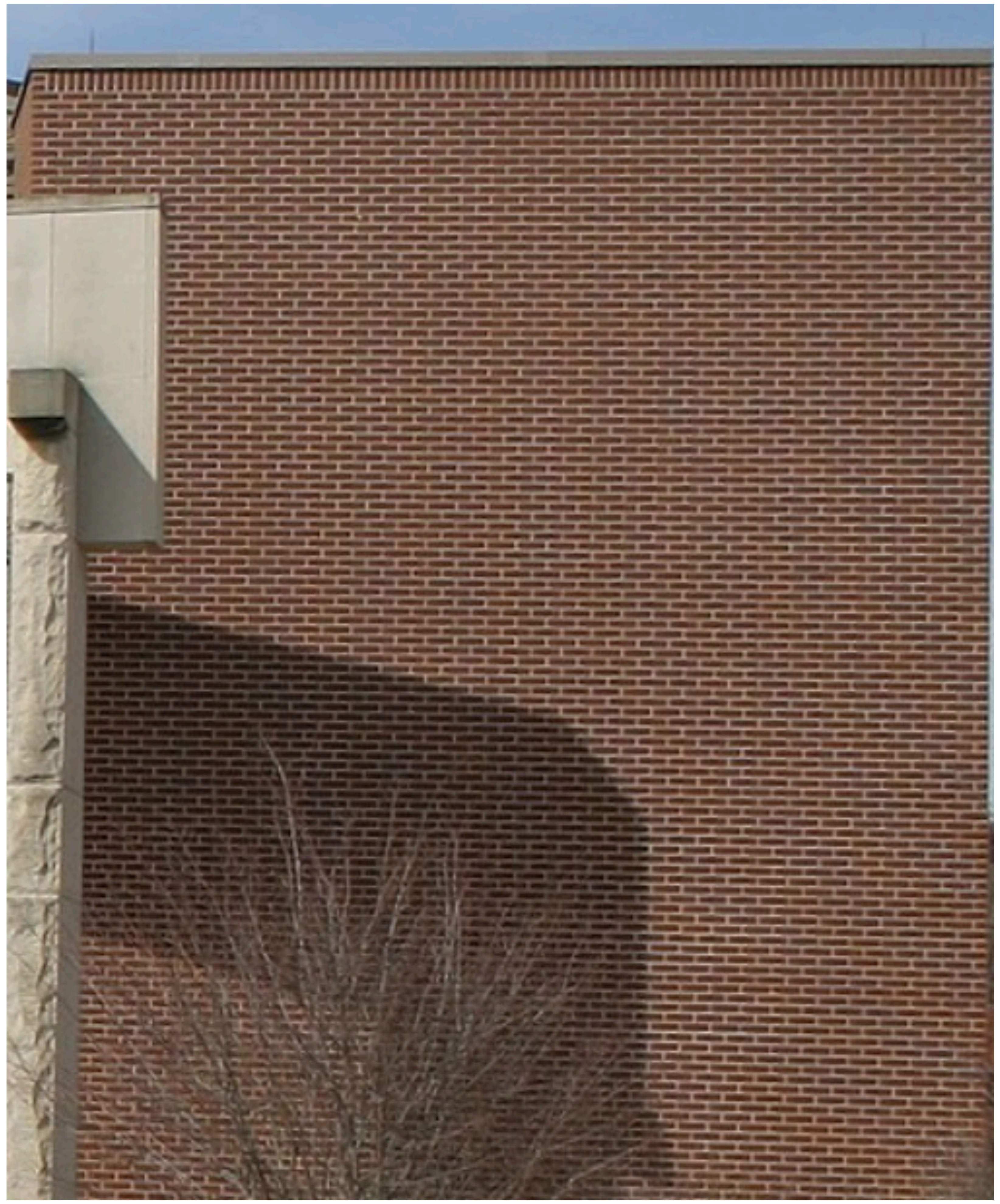


Aliasing in Images



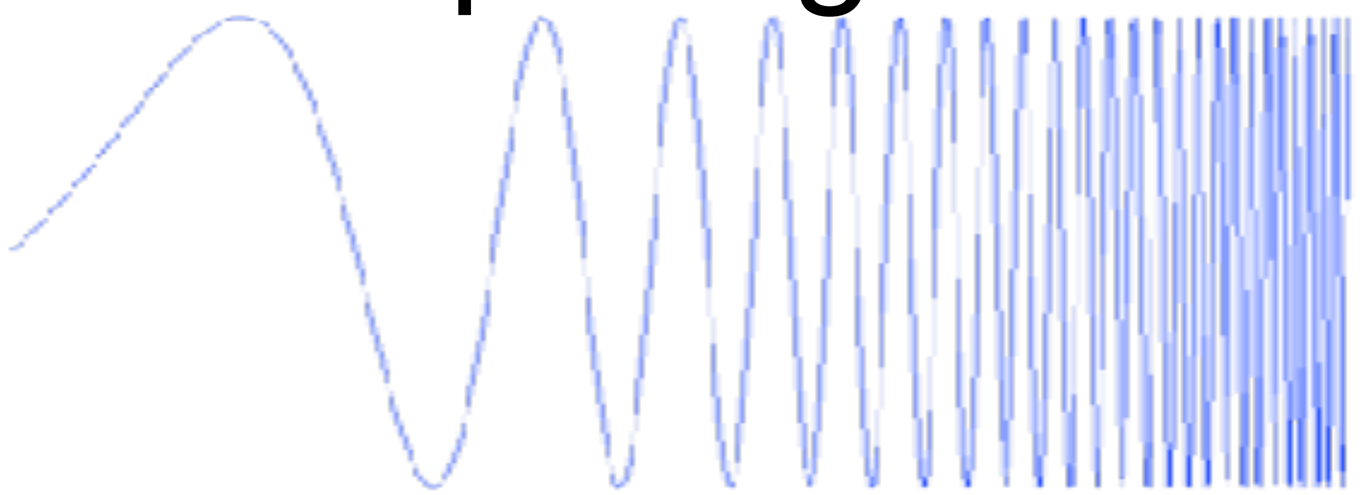
Aliasing in Images





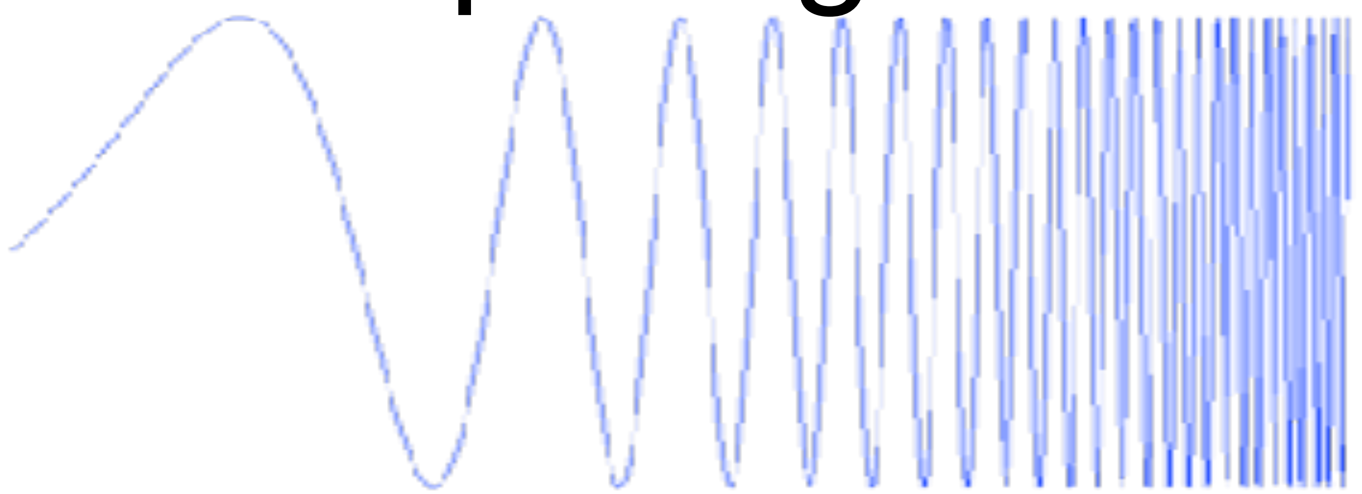
What’s Happening?

Input signal:

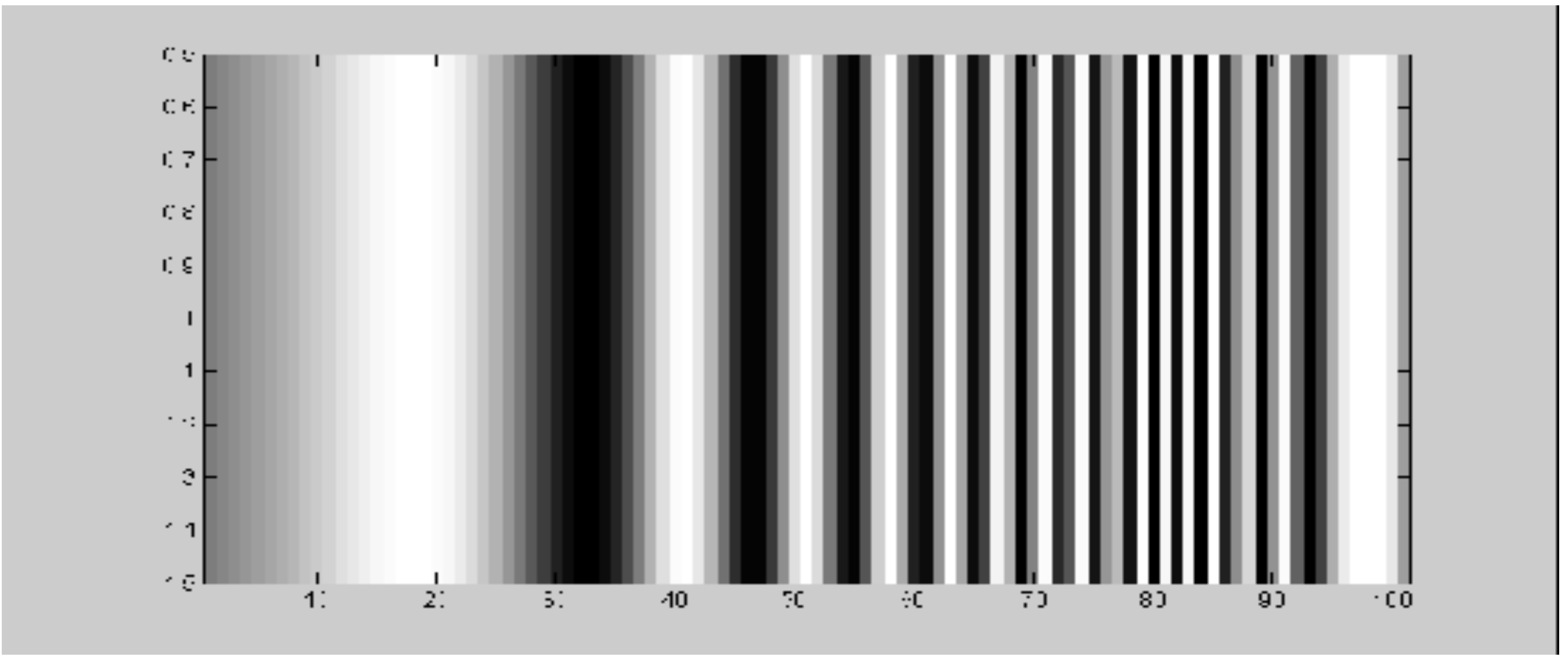


What's Happening?

Input signal:



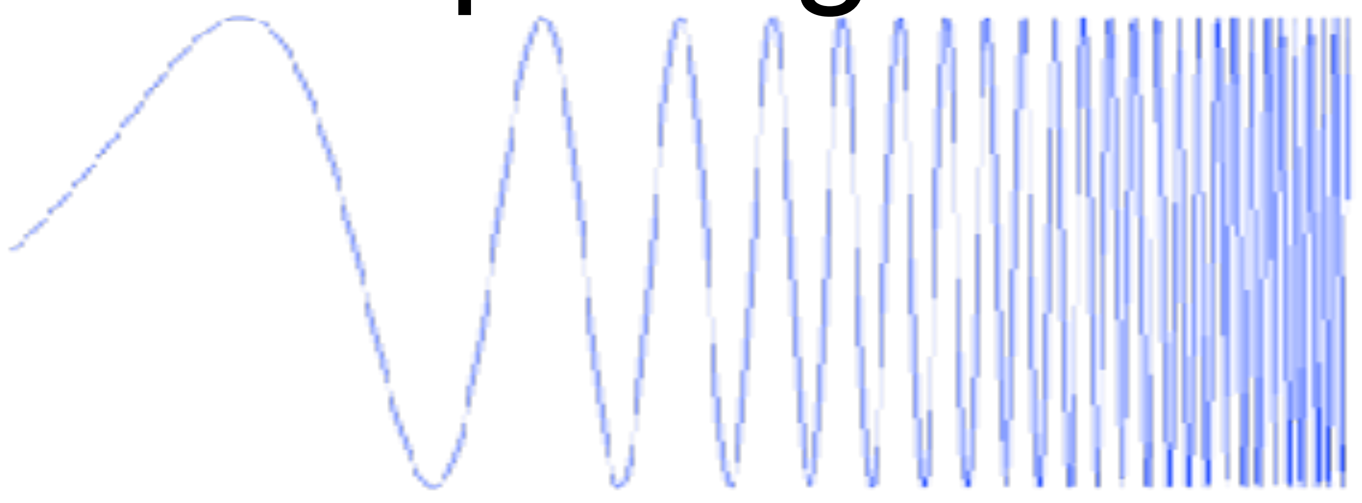
Plot as image:

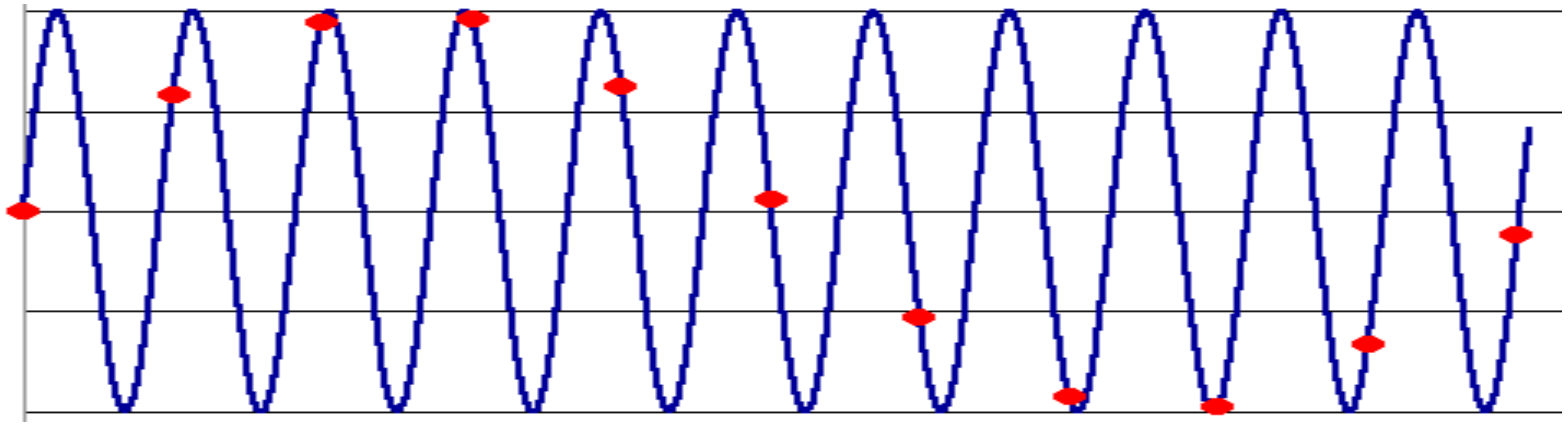


$$x = 0 : 0.05 : 5; \quad \text{imagesc}(\sin(2^x x))$$

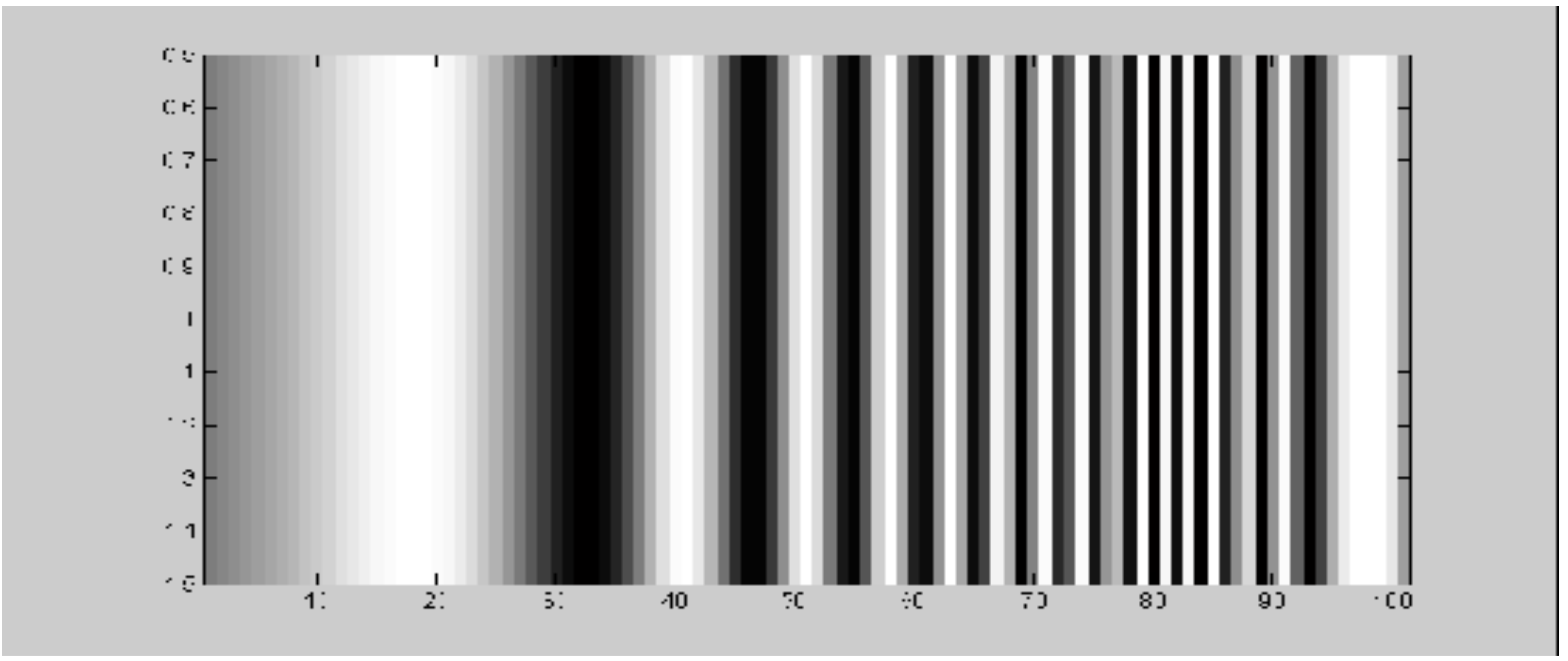
What's Happening?

Input signal:





Plot as image:



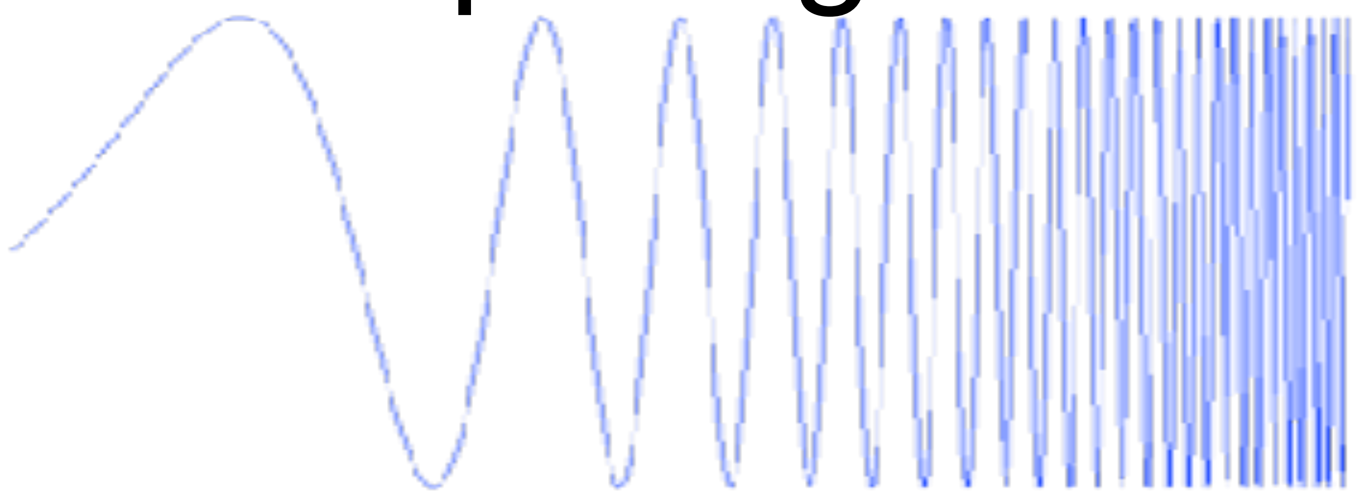
Alias!

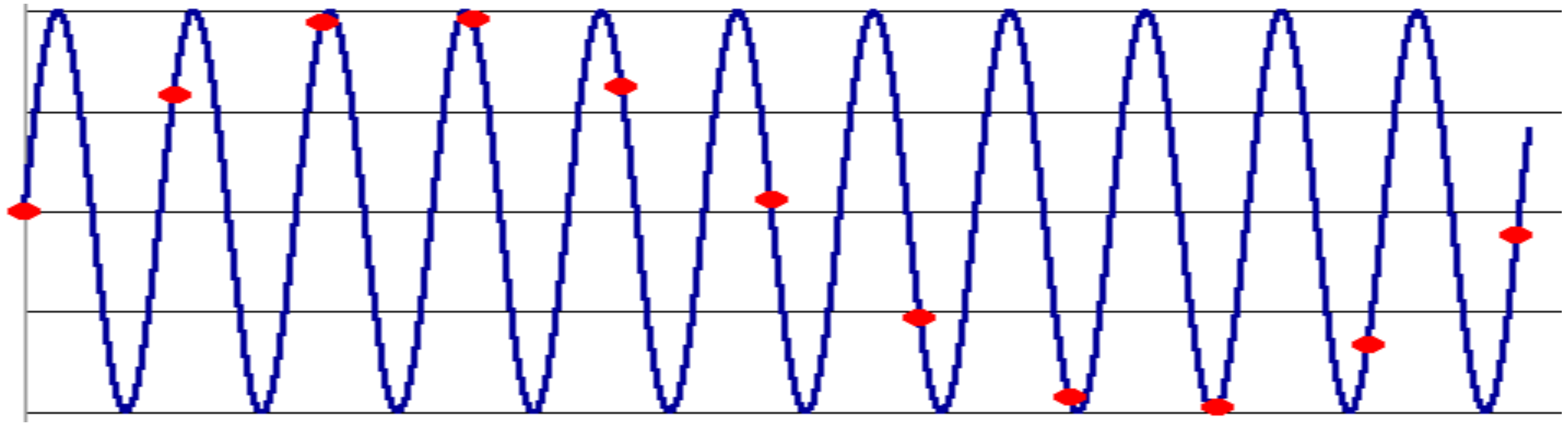
Not enough samples

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

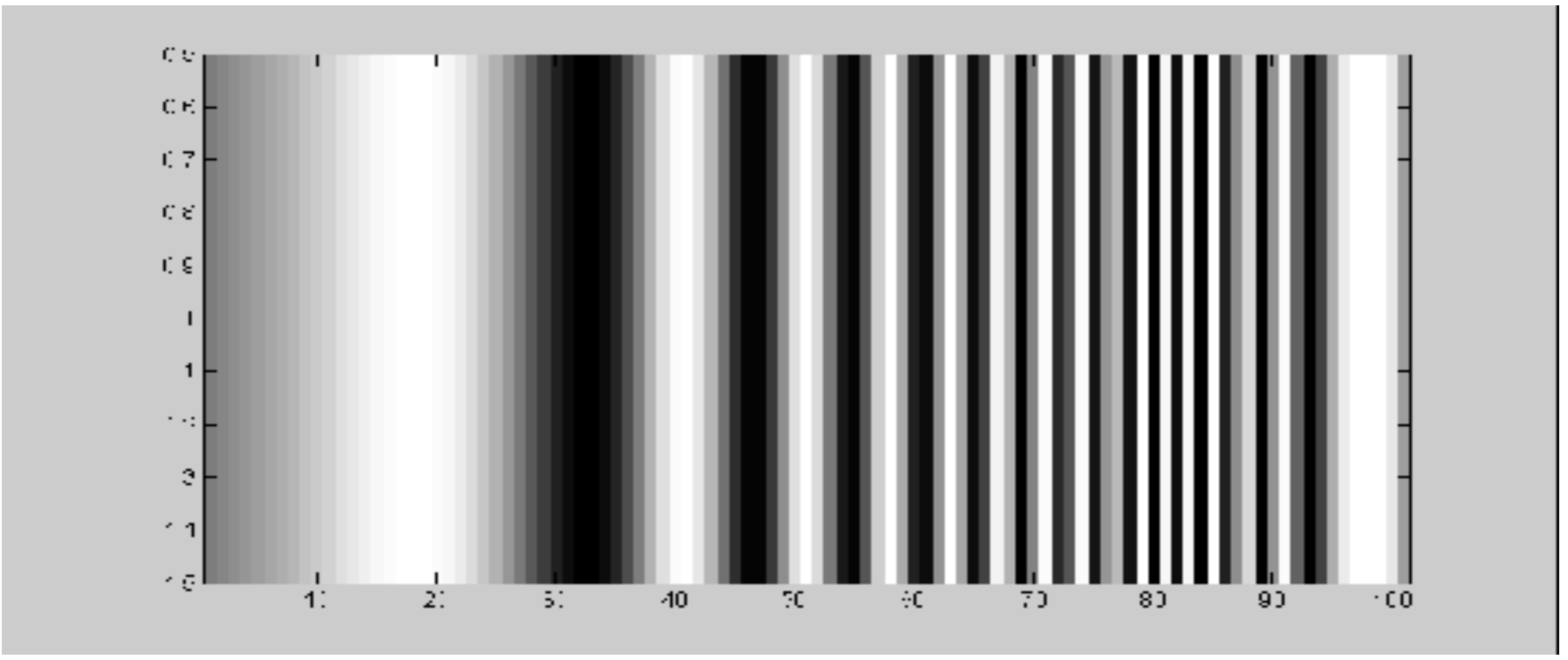
What's Happening?

Input signal:





Plot as image:



Alias!

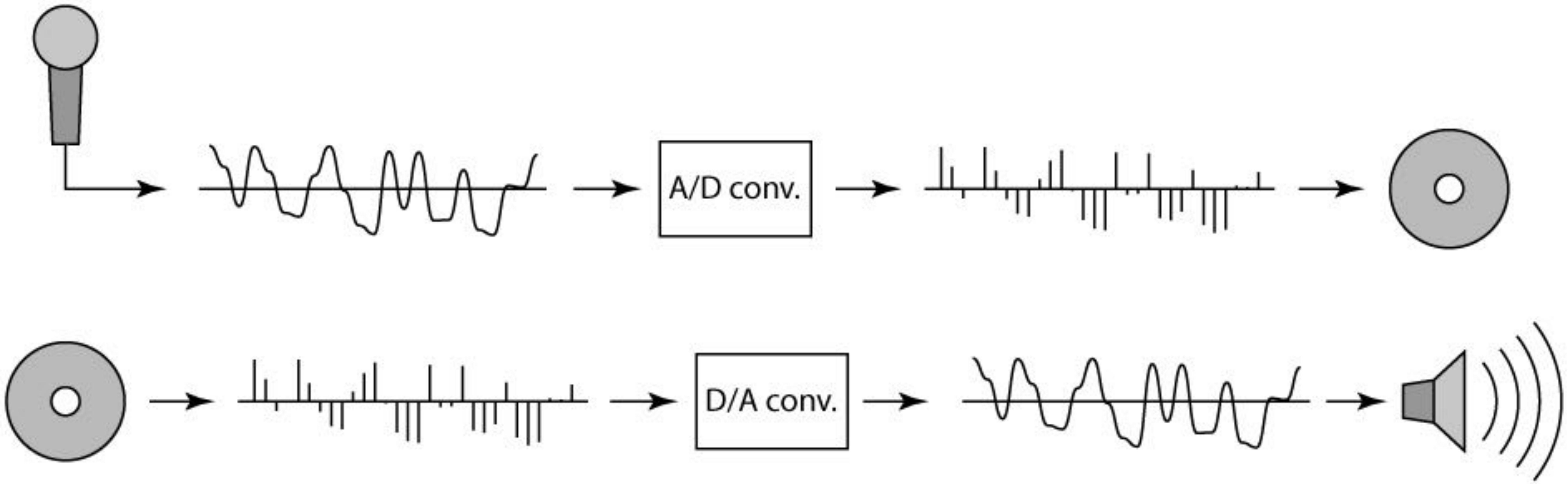
Not enough samples

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

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

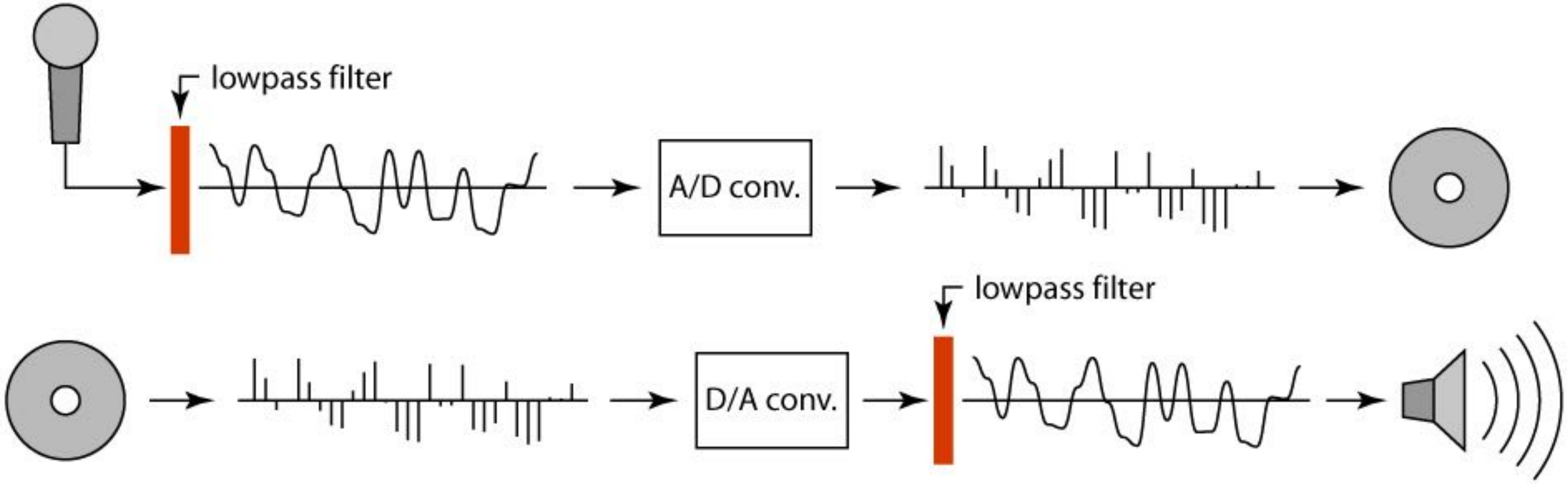
Preventing aliasing

- 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?

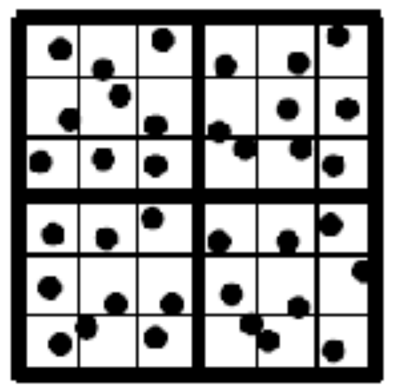
Sampling rate > 2 * max frequency in the image

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

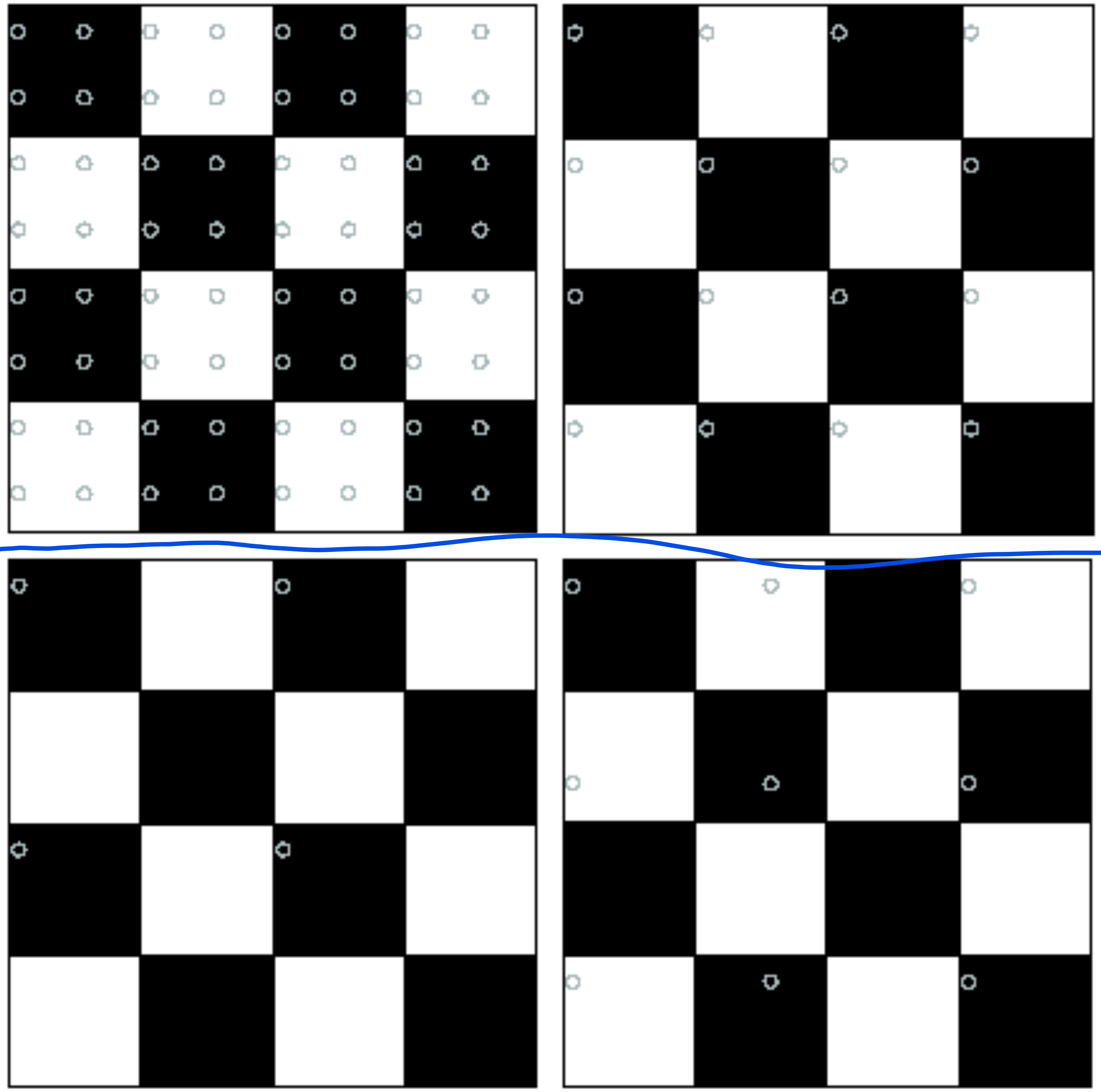
for a system with a fixed sampling rate of 2N samples/sec.

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?

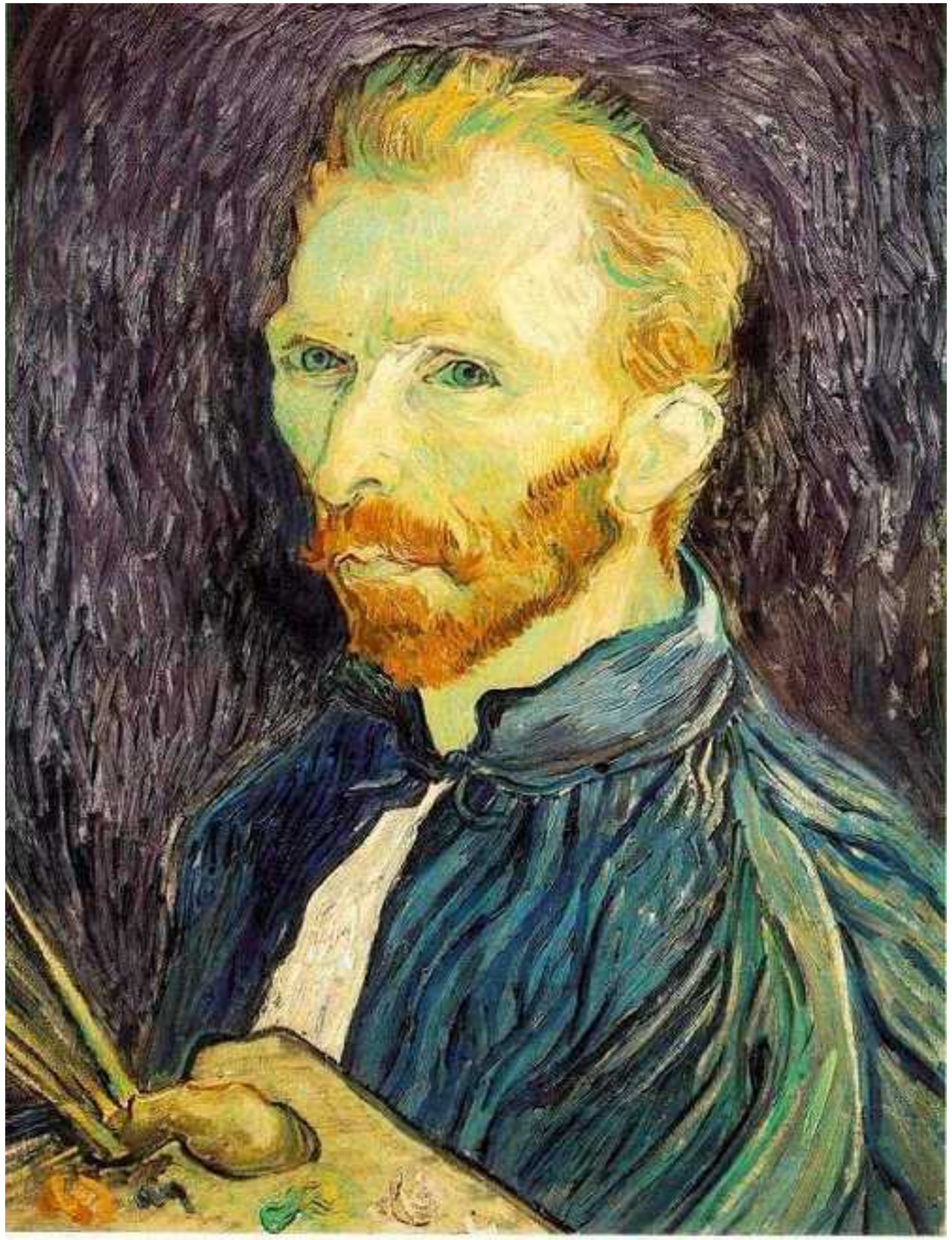
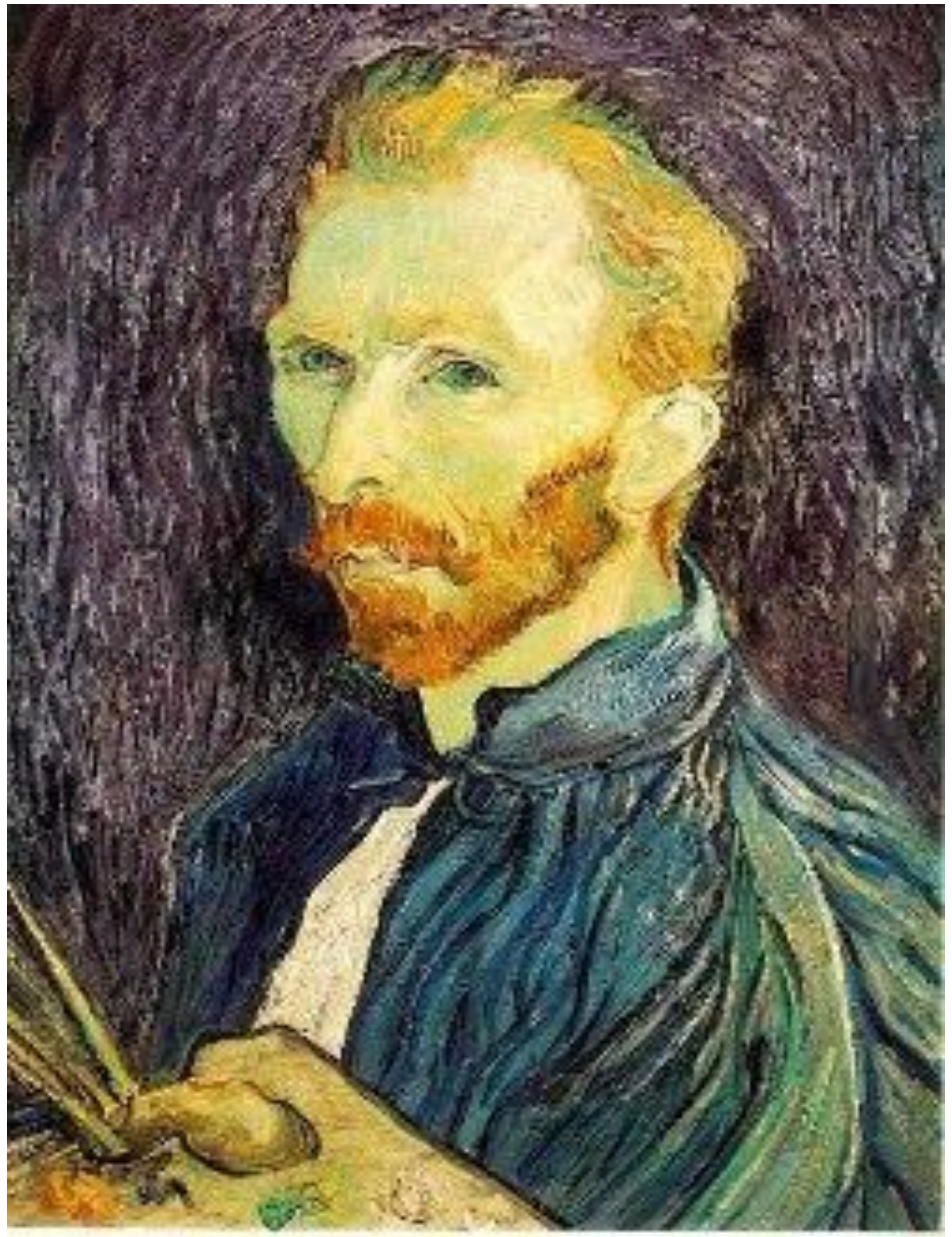


Image sub-sampling

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



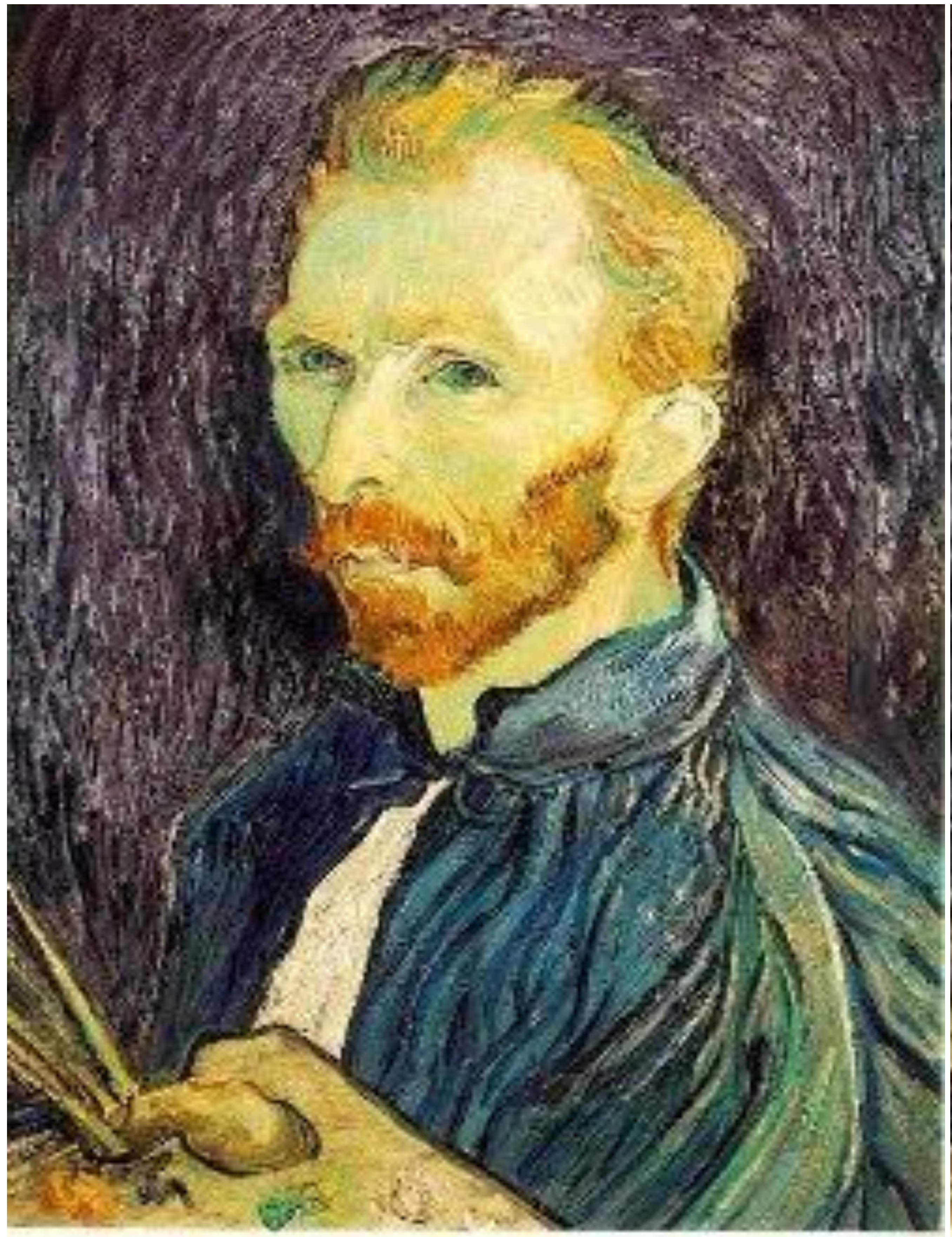
1/4

1/8

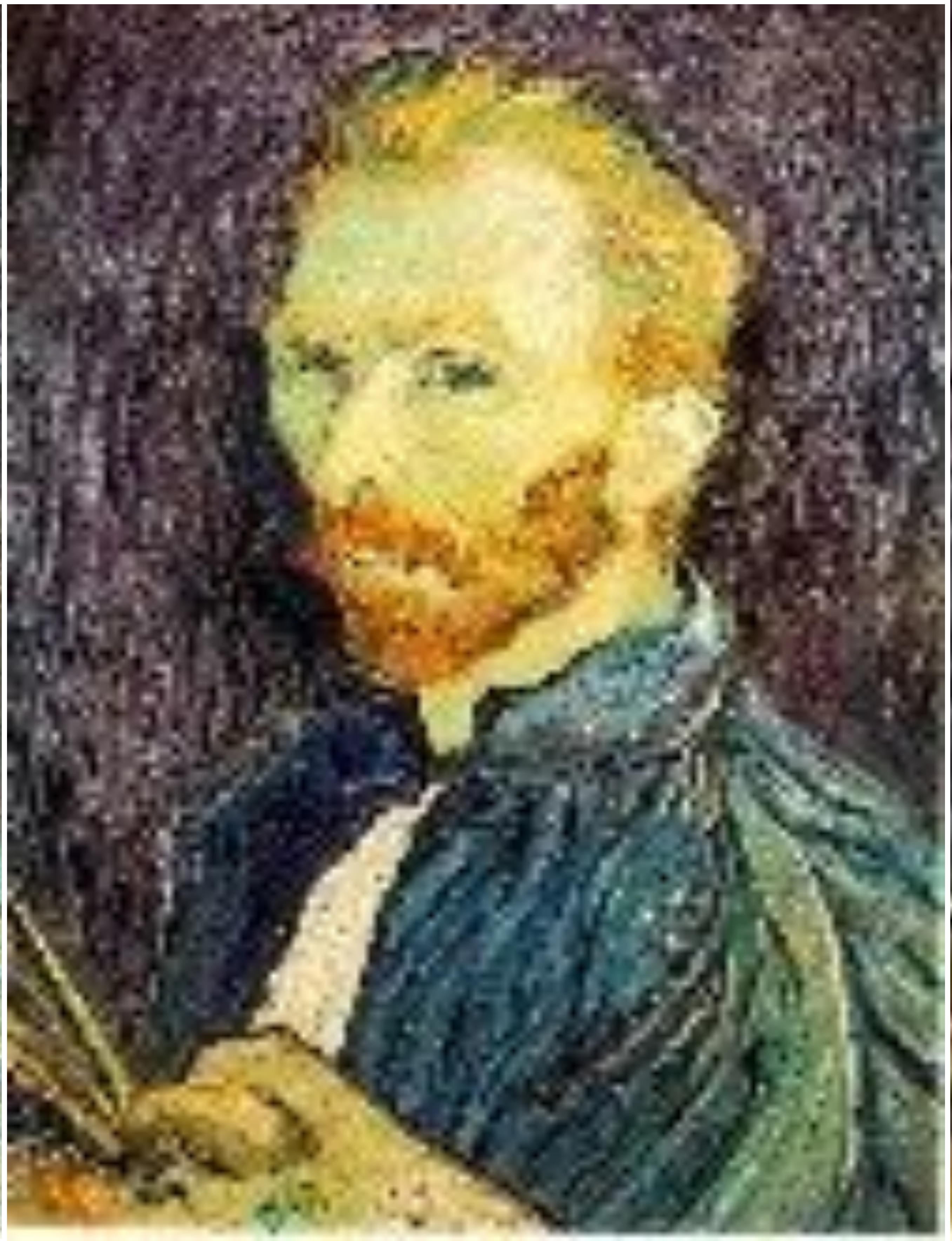
Slide by Steve Seitz

Image Sub-sampling

Aliasing! What do we do?



1/2



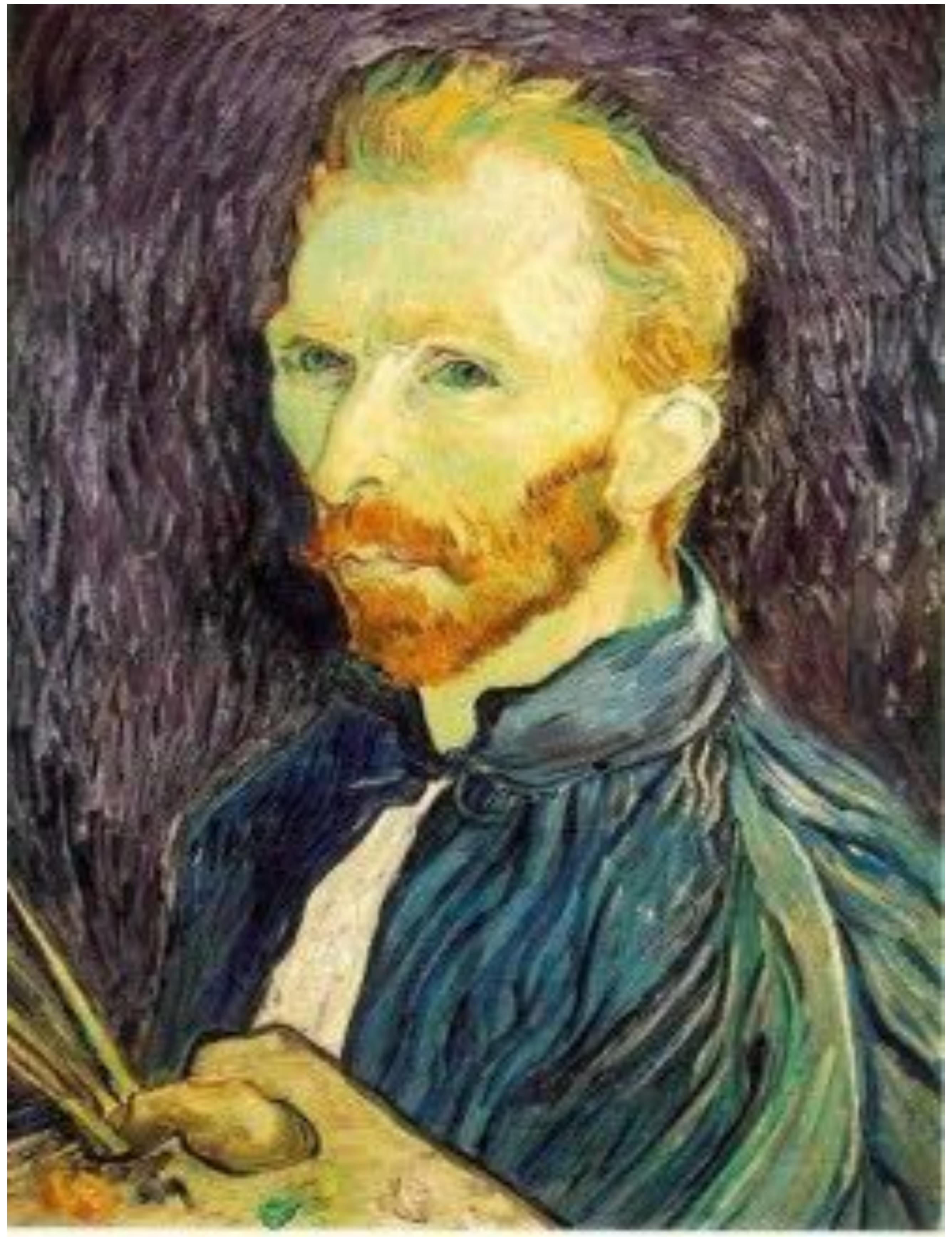
1/4 (2x zoom)

1/8 (4x zoom)

Gaussian (lowpass) Pre-filtering

可以抑掉高的信号 frequency

这里的f不指变化快而是出现的次数



Gaussian 1/2

好多了

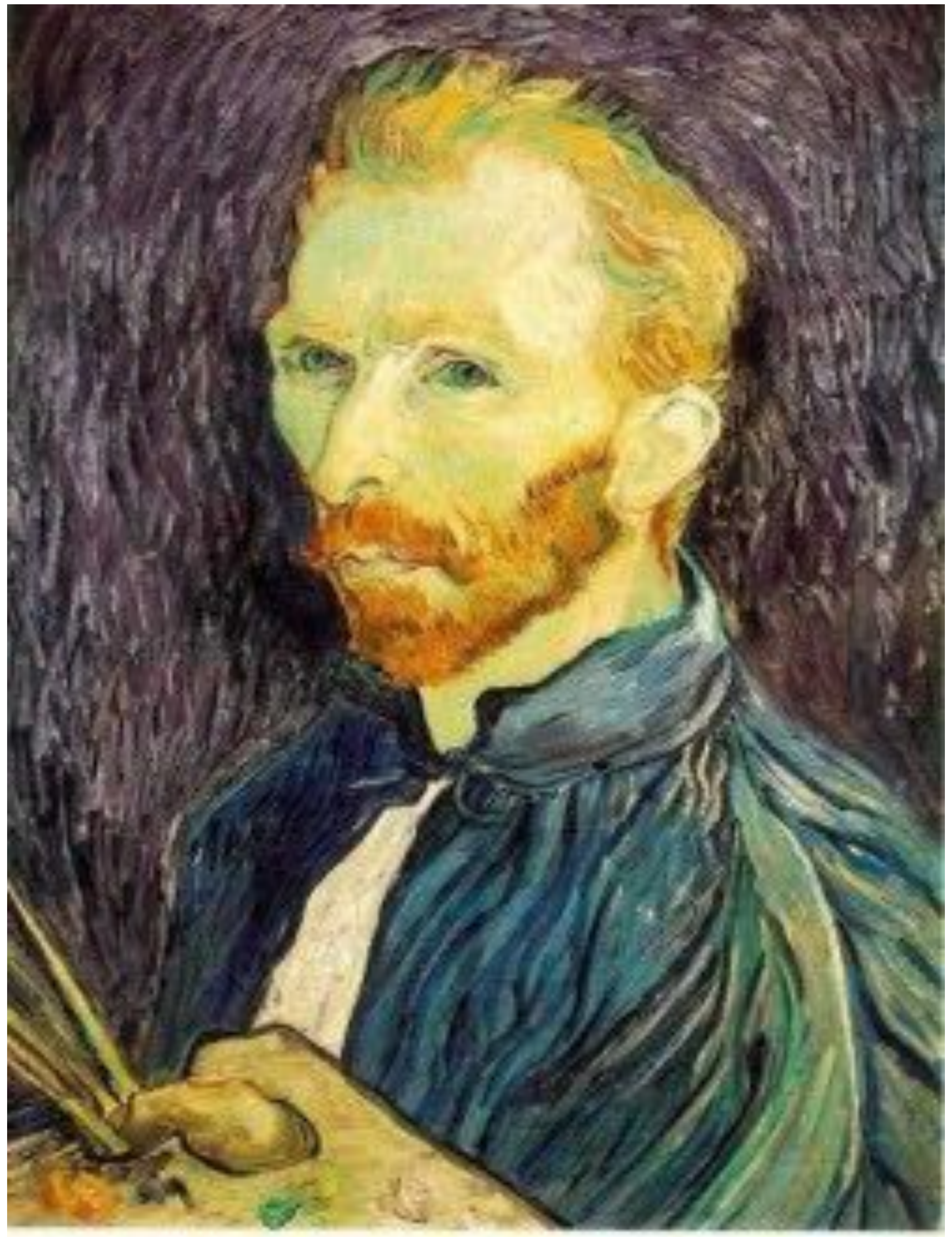
G 1/4

G 1/8

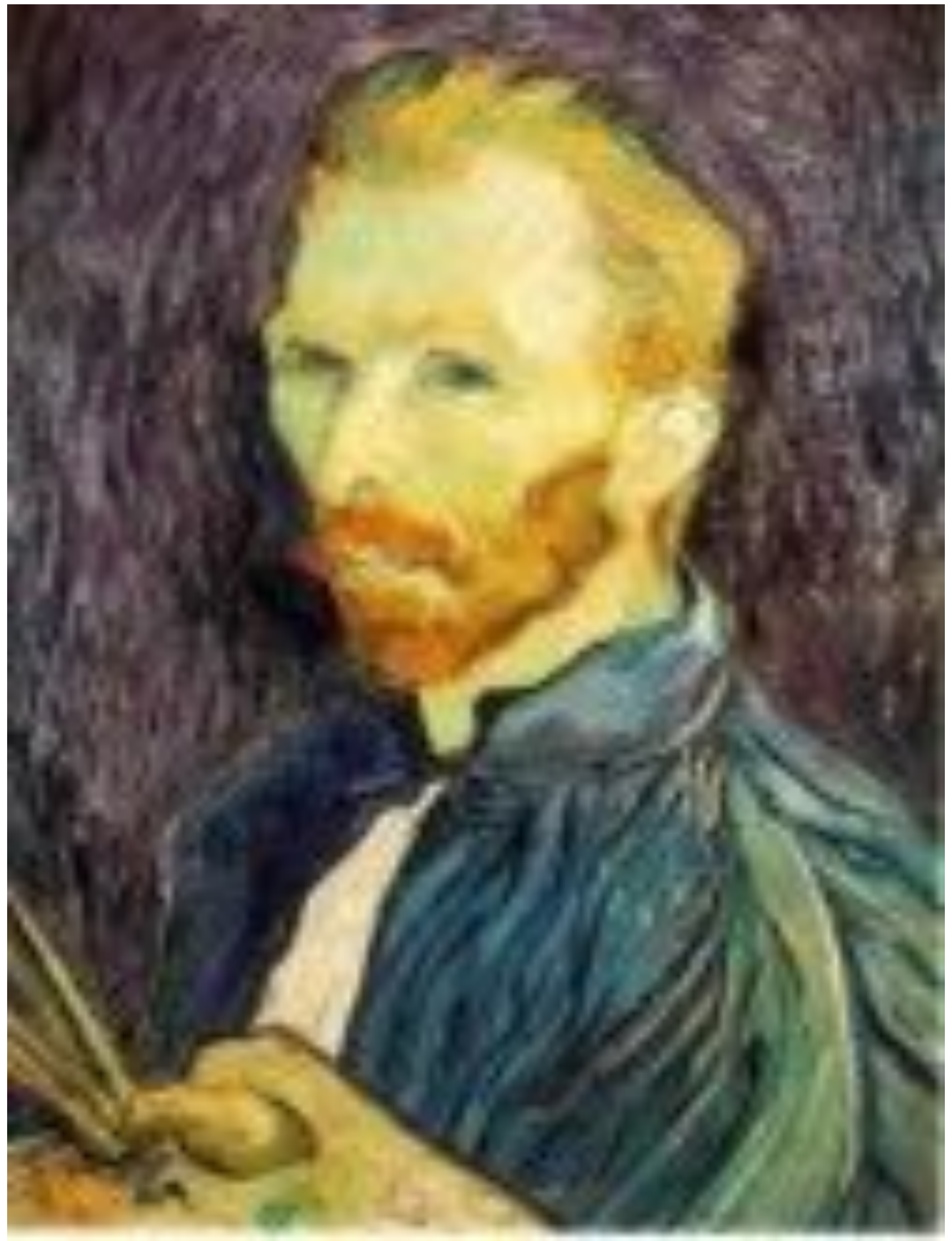
Solution: filter the image, *then* subsample

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

Subsampling with Gaussian Pre-filtering



Gaussian 1/2



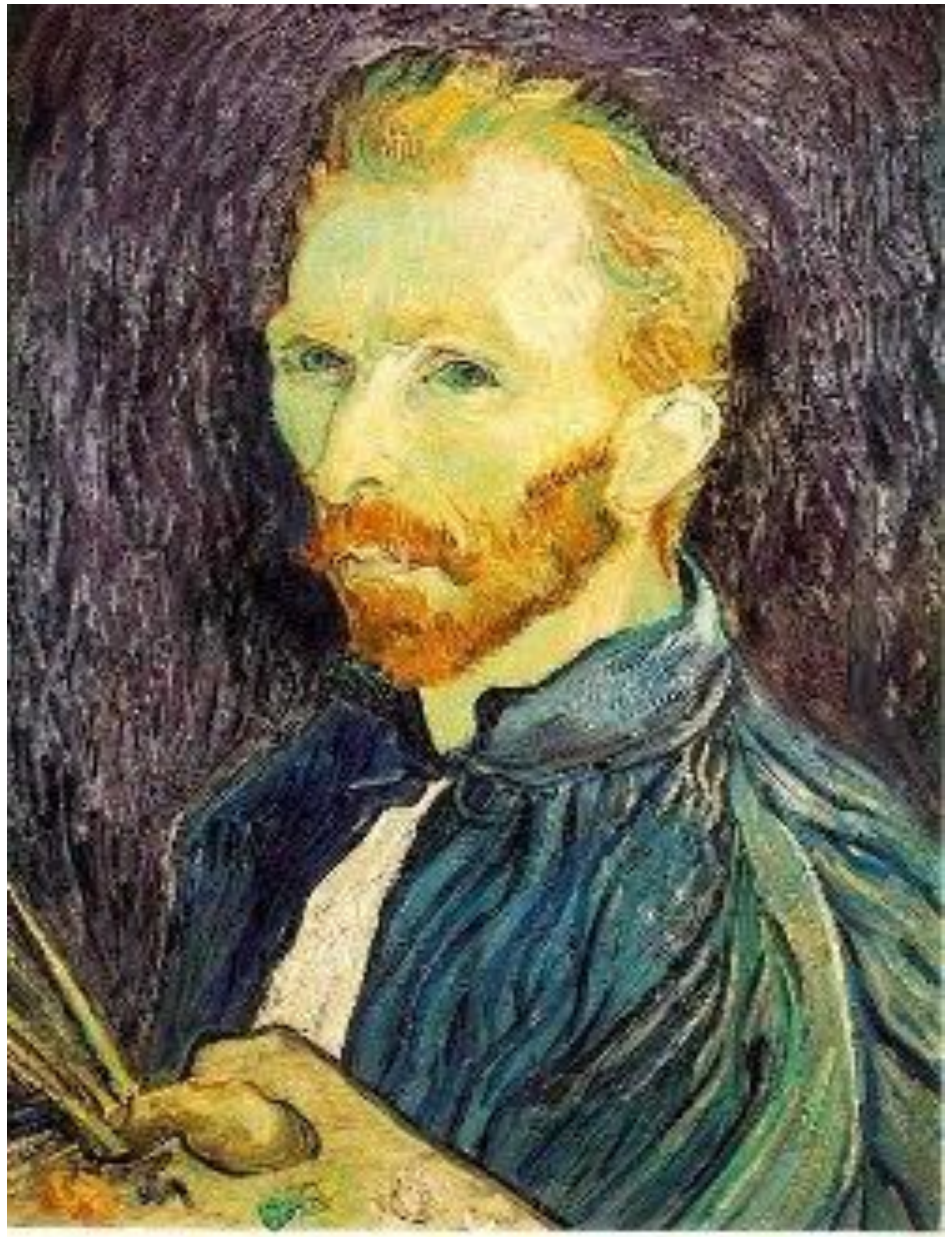
G 1/4

G 1/8

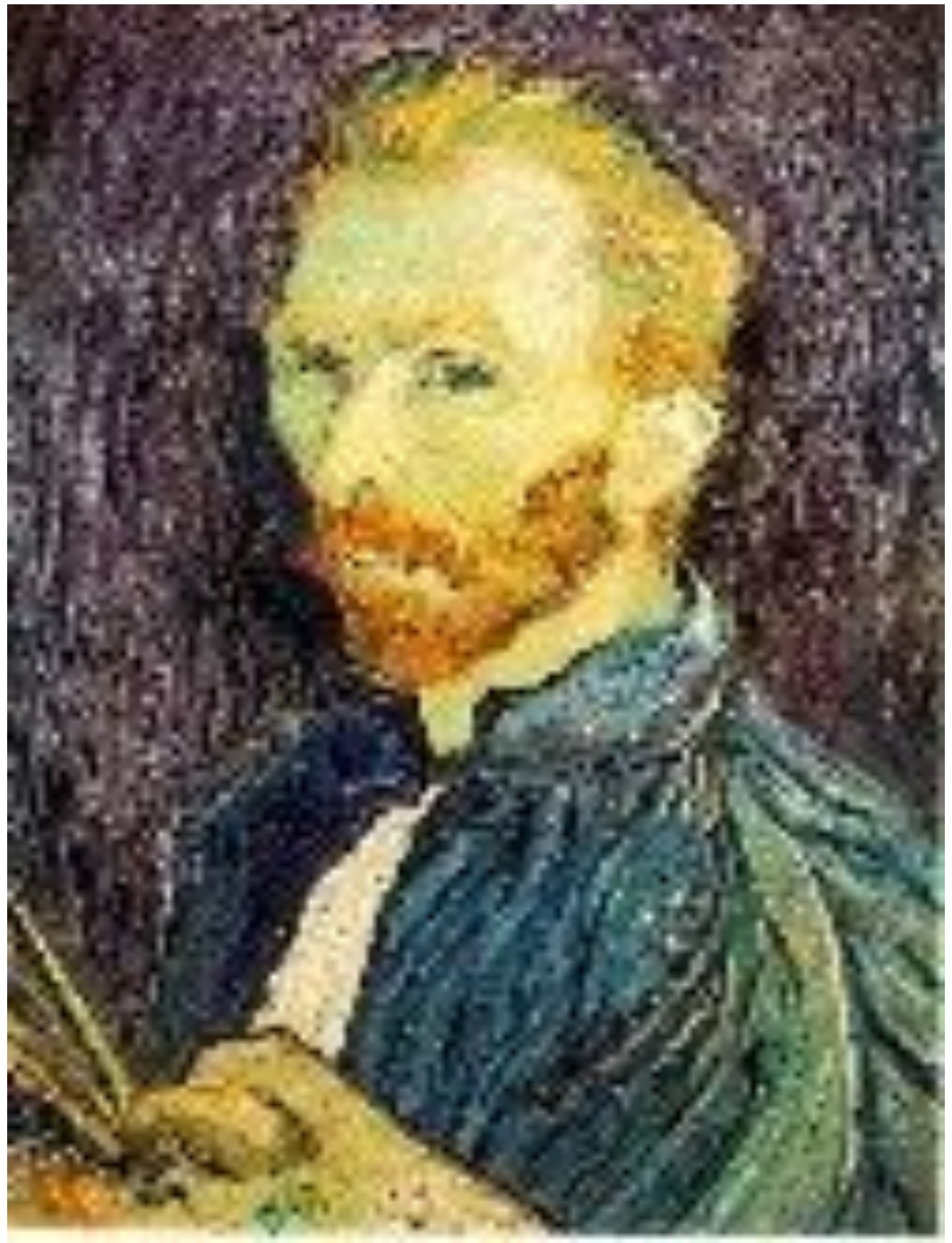
Solution: filter the image, *then* subsample

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

Compare with Just Subsampling



1/2

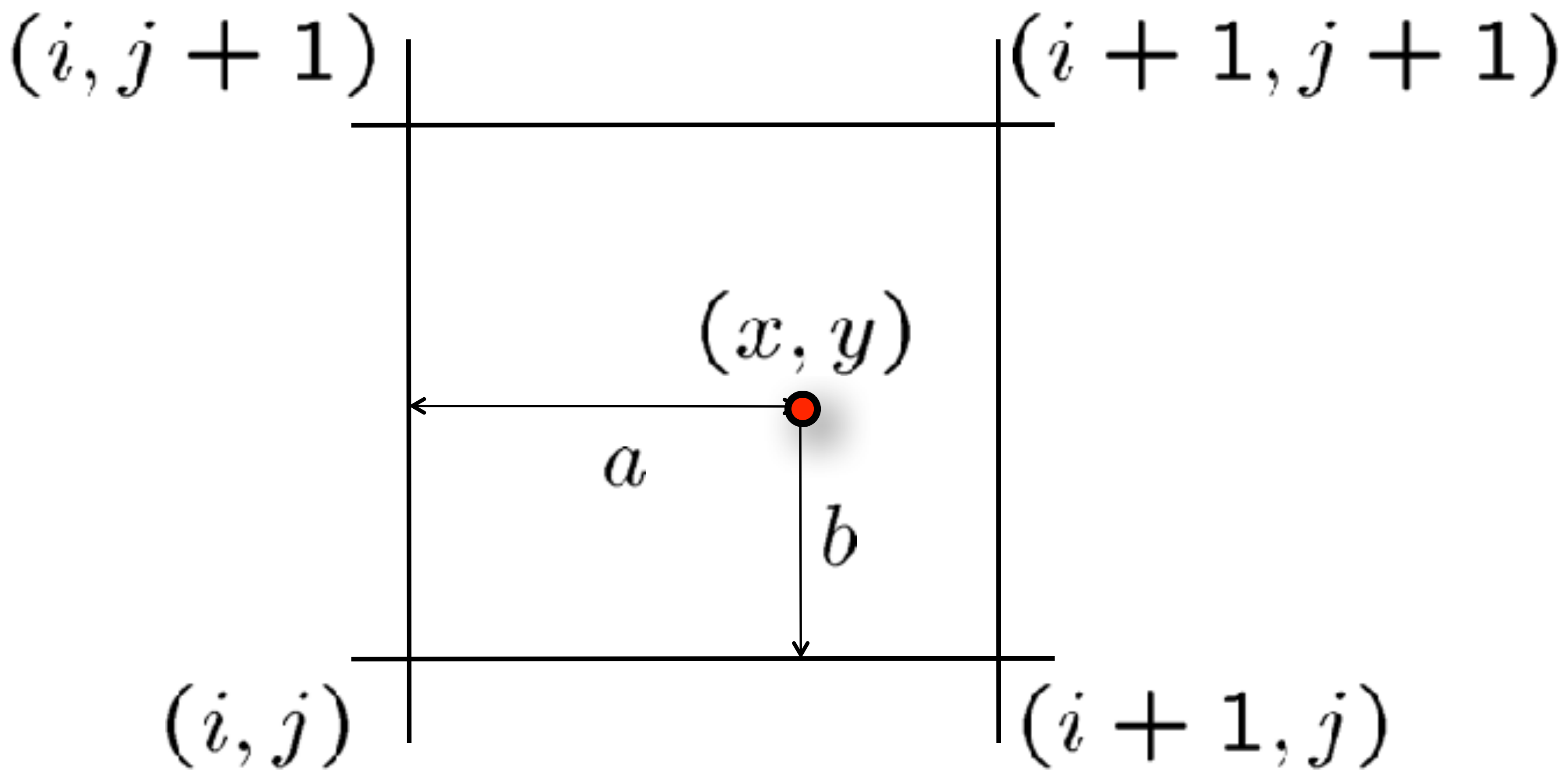


1/4 (2x zoom)

1/8 (4x zoom)

Bilinear Interpolation

Sampling at *f(x,y)*:



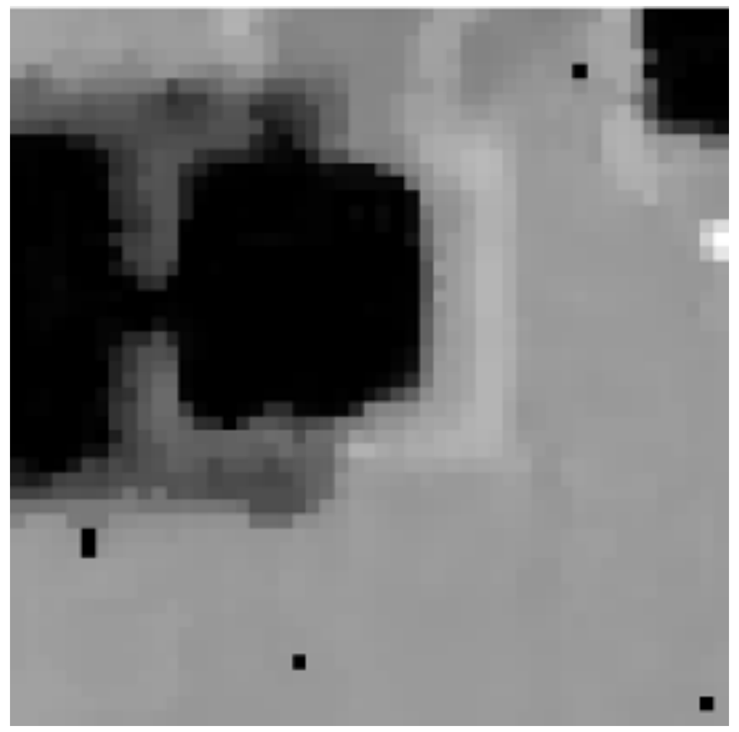
$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

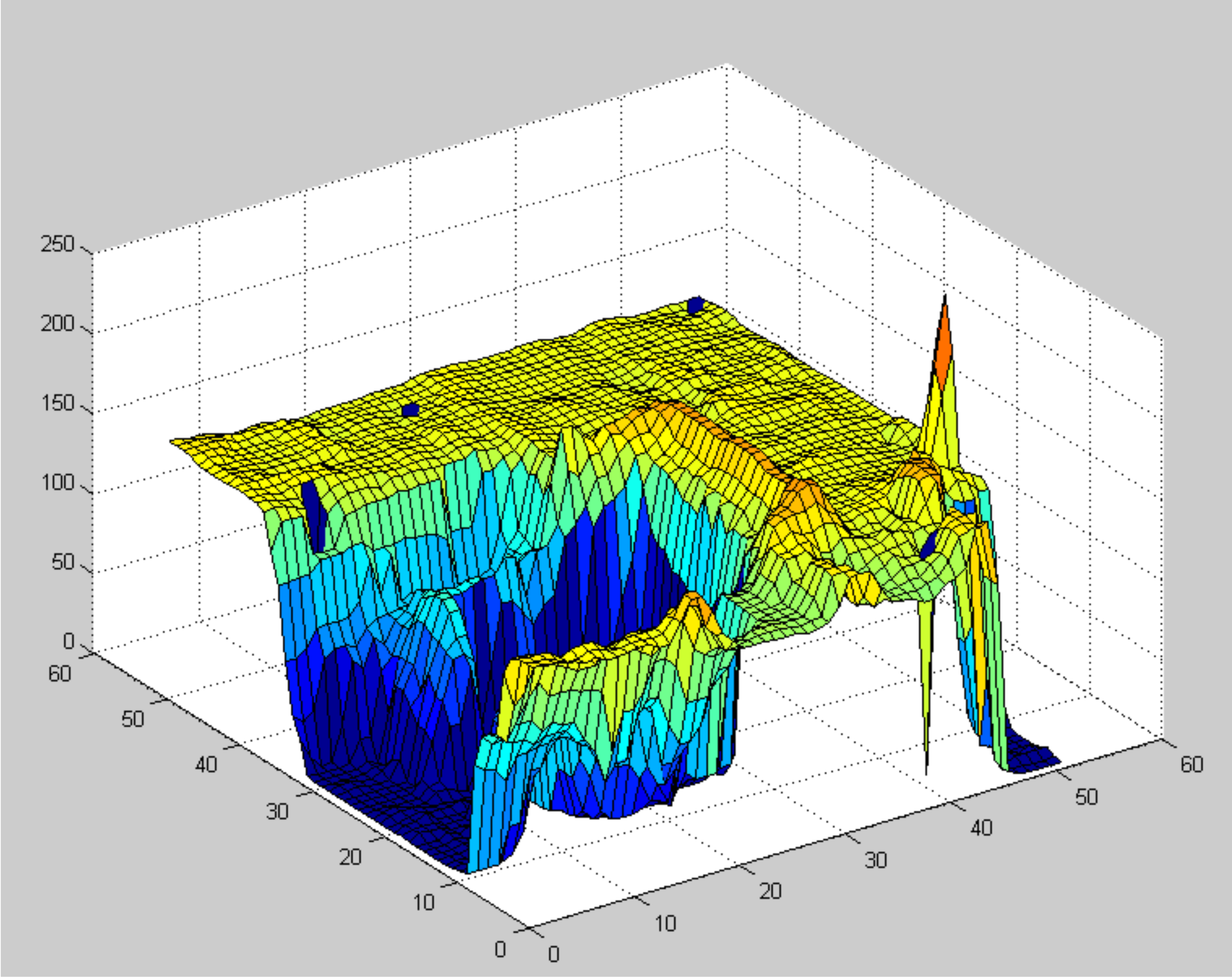
Bilinear Interpolation

Sampling at *f(x,y)*:

Image == Heightfield

I





z = *I*'s intensity

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

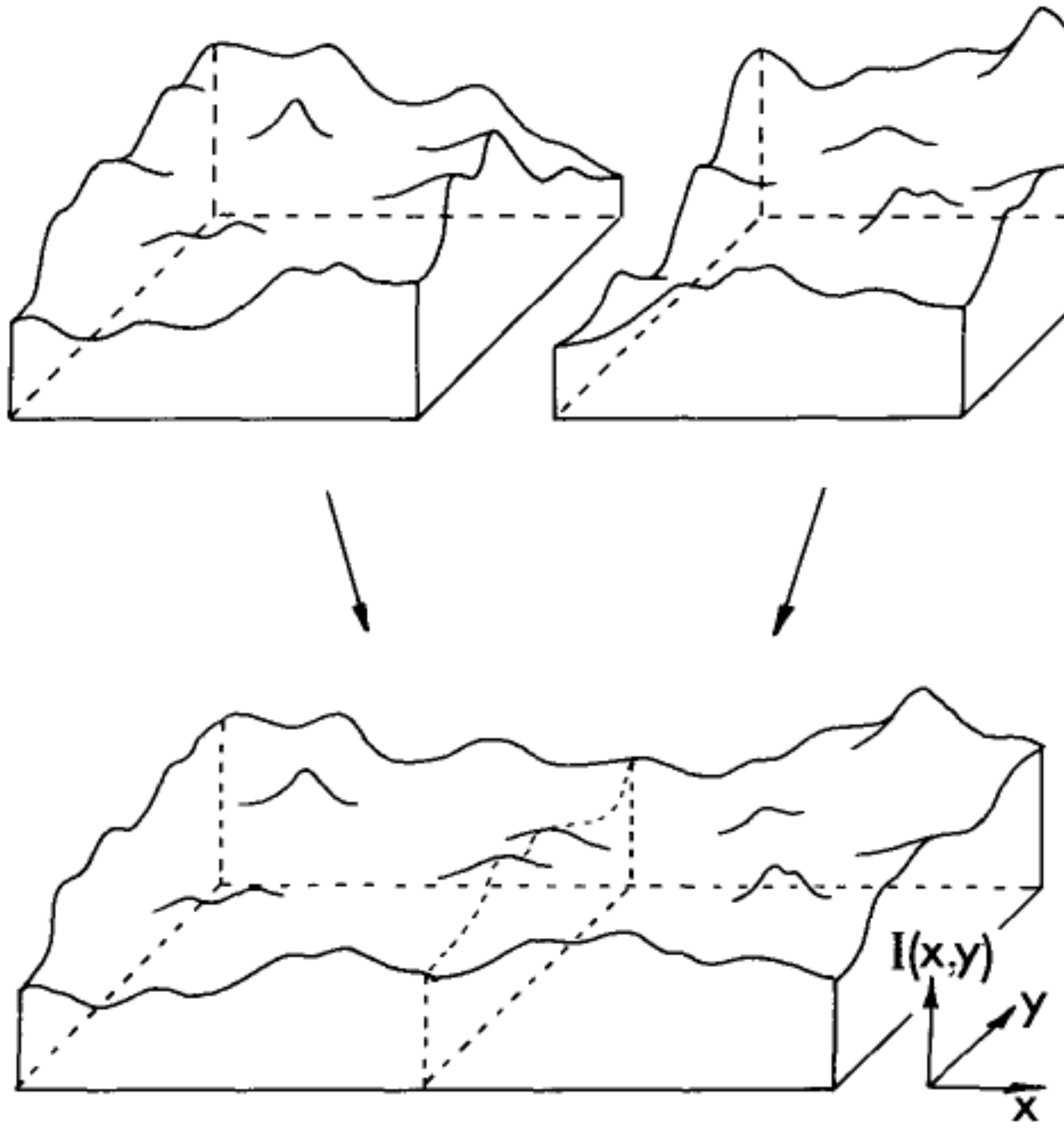


Fig. 1. A pair of images may be represented as a pair of surfaces above the (x, y) plane. The problem of image splining is to join these surfaces with a smooth seam, with as little distortion of each surface as possible.

How does one Normally Blend?

How does one Normally Blend?

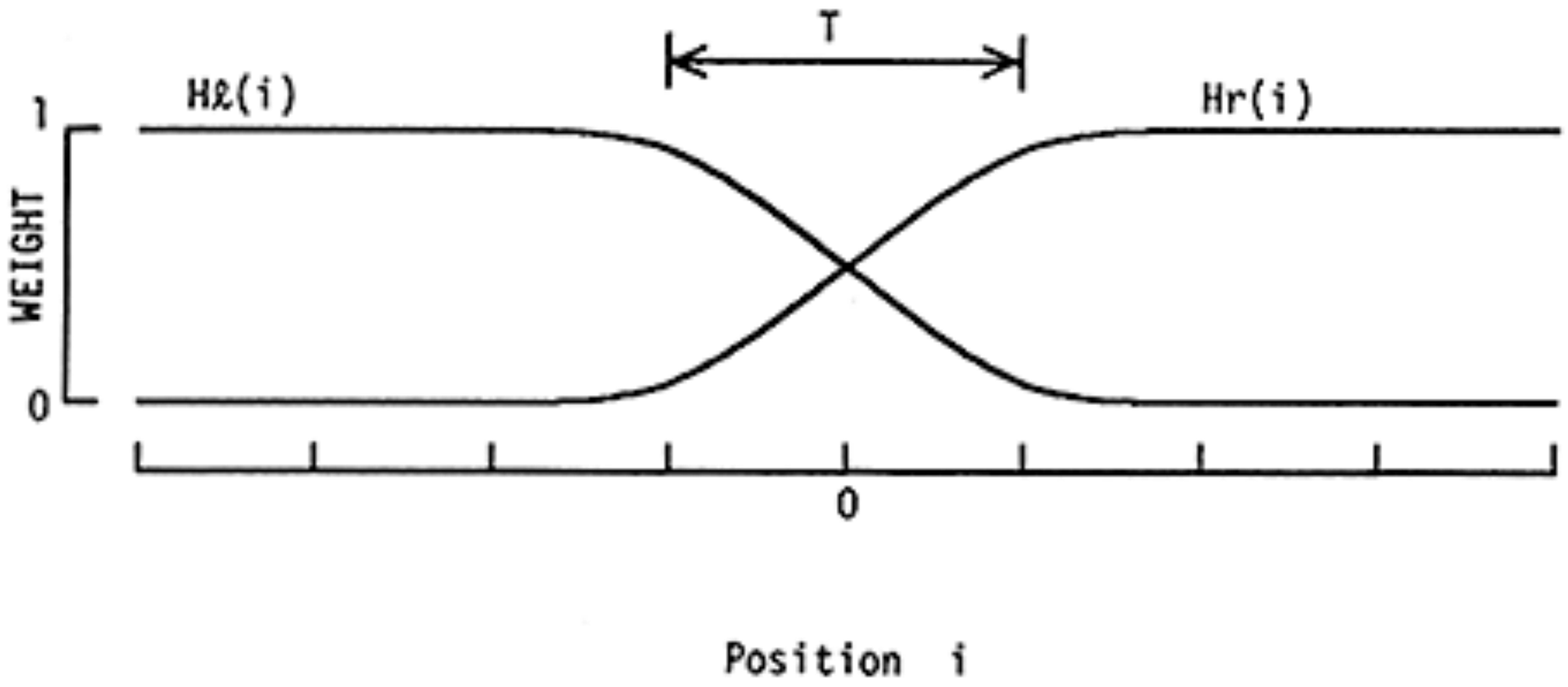
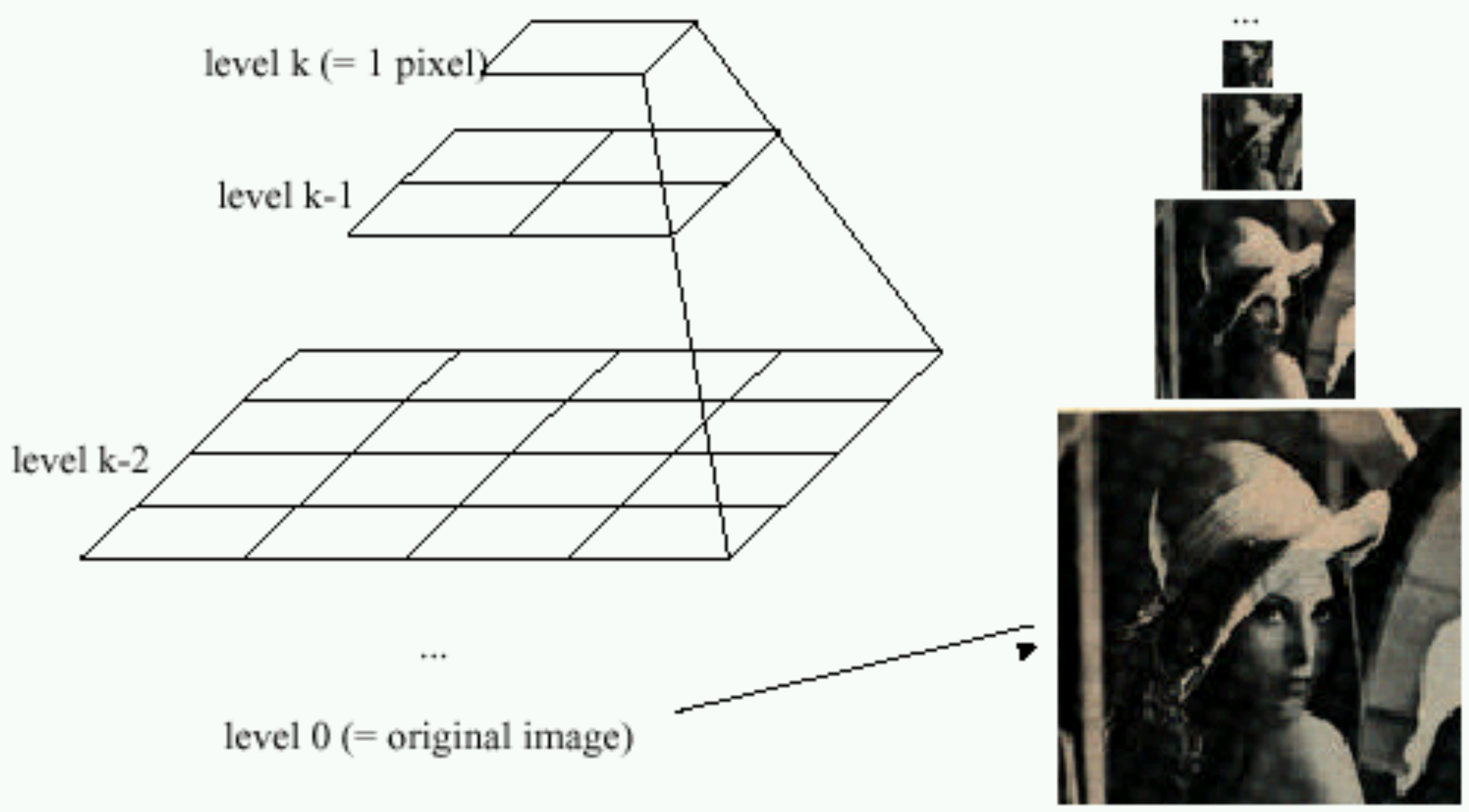


Image Pyramids

Idea: Represent NxN image as a “pyramid” of $1\text{x}1, 2\text{x}2, 4\text{x}4, \ldots, 2^k\text{x}2^k$ images (assuming $N=2^k$)

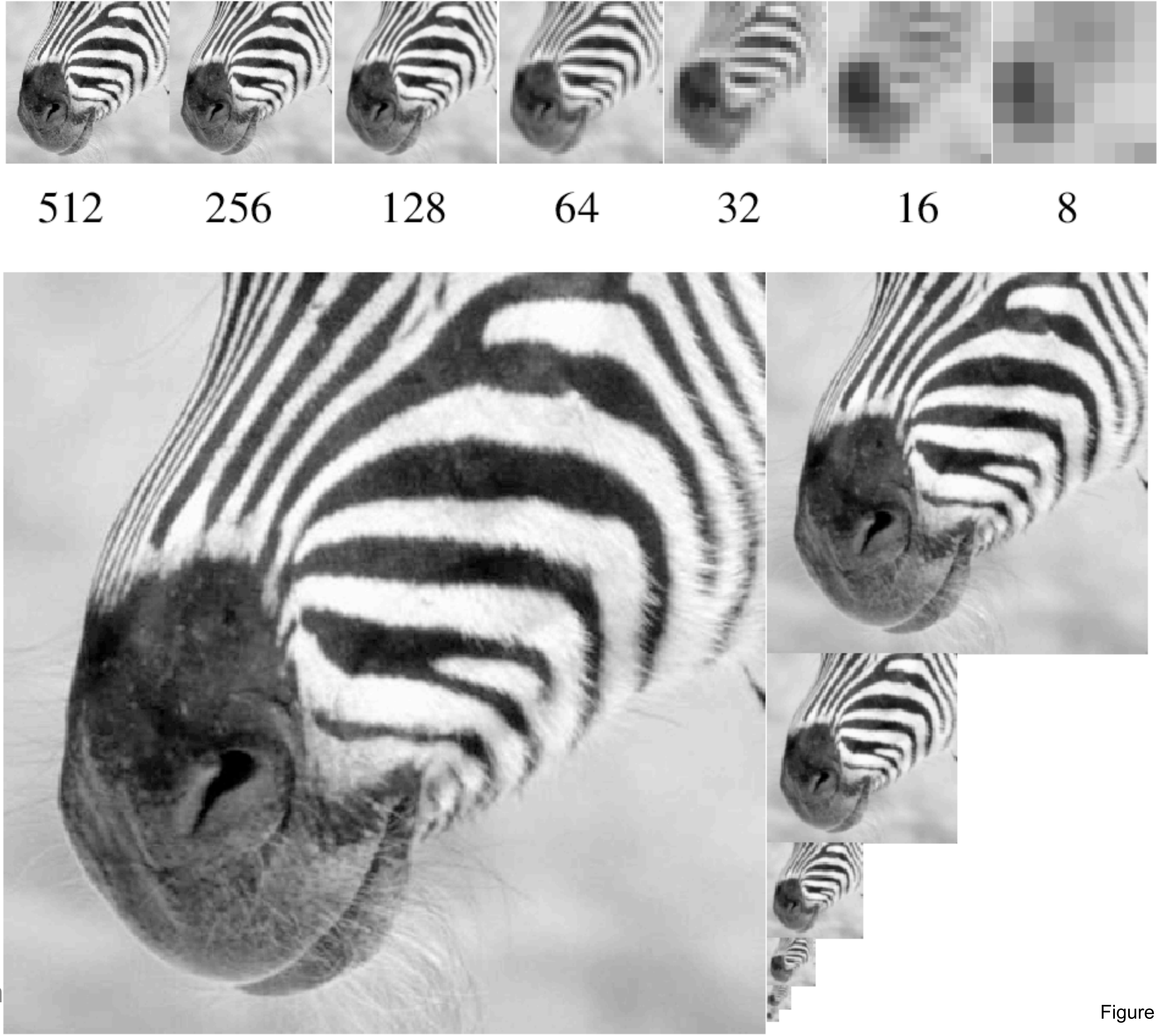


Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

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

A precursor to *wavelet transform*

First introduced for compression purposes



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

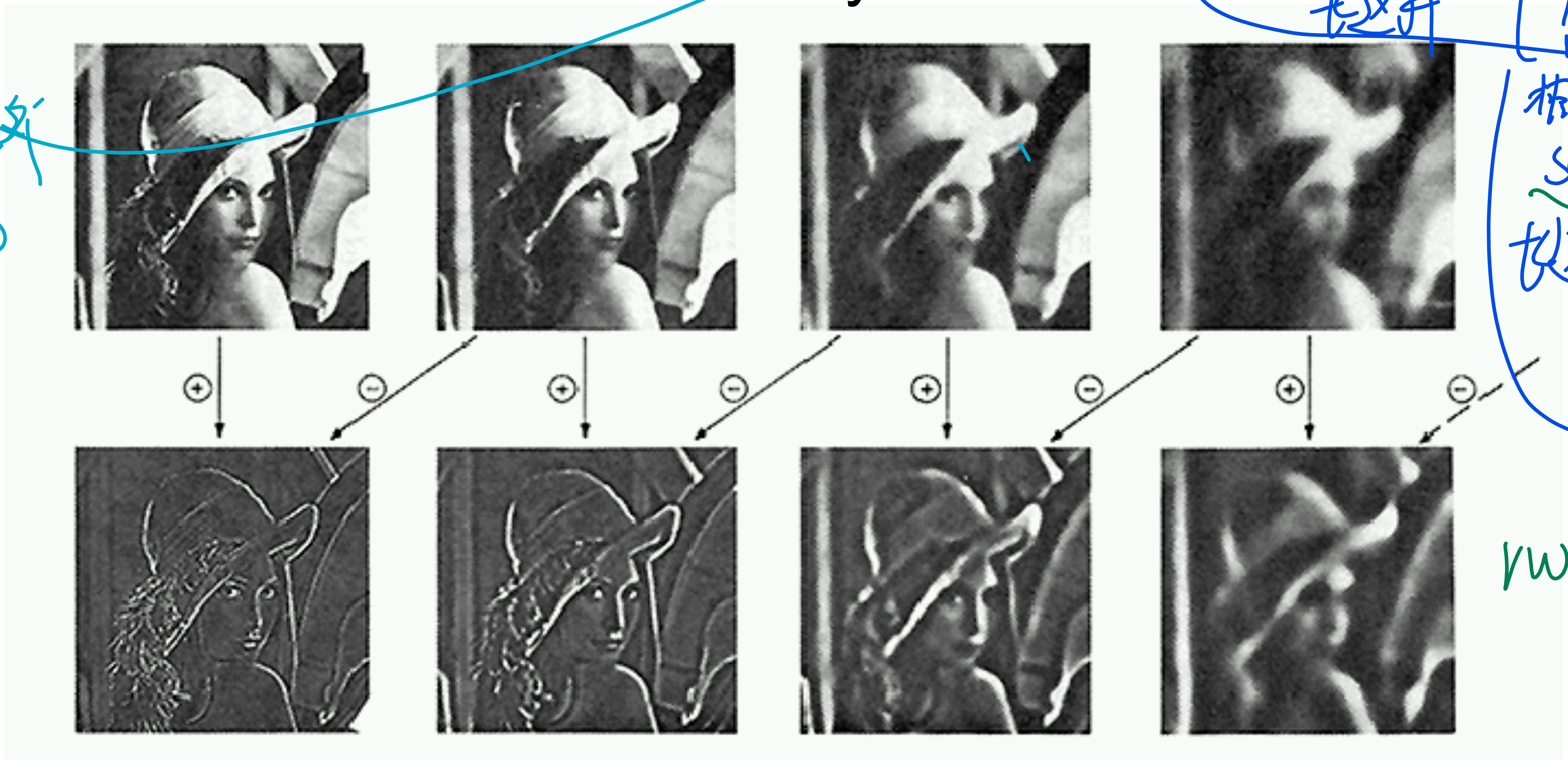
Laplacian Pyramid

这里 laplace 名字只是相同

laplacian 复习:
二阶导边缘检测
(用图像梯度的二阶导)
卷积 $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \Leftrightarrow \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial x^2}$

相对应
sobel 算子是一阶
卷积 $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
$\Rightarrow$ 以前一节课

Gaussian Pyramid



“Laplacian” Pyramid (subband images)

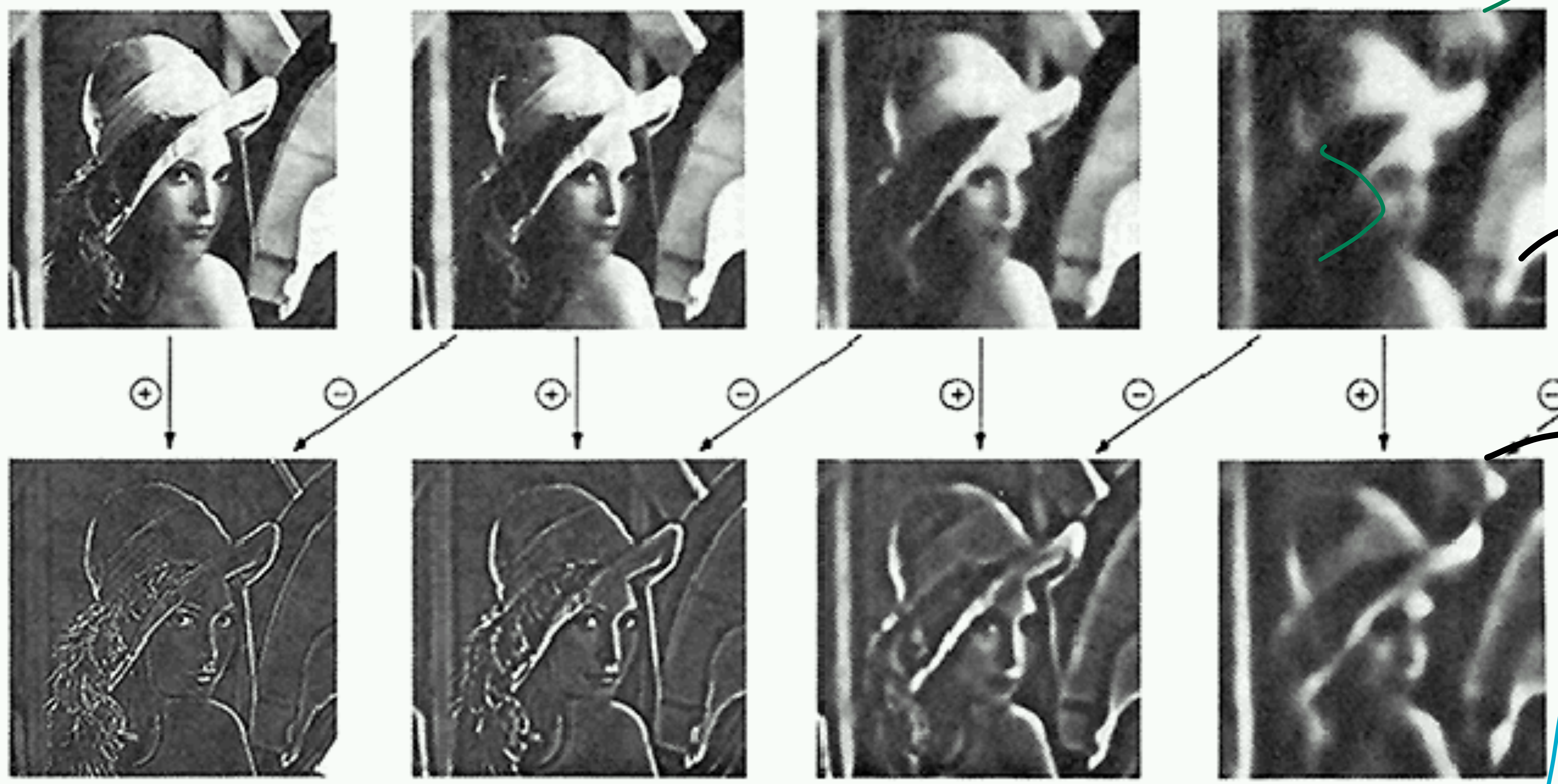
Created from Gaussian pyramid by subtraction

Laplacian Pyramid

Remember Unsharp Mask?

Pyramid Blending

Gaussian Pyramid



“Laplacian” Pyramid (subband images)

Created from Gaussian pyramid by subtraction

step 1. 先用 Gaussian 得 Pyramid

step 2 得 laplacian

step 3 blend

image 1

image 2

step 4 add up

Laplacian Pyramid

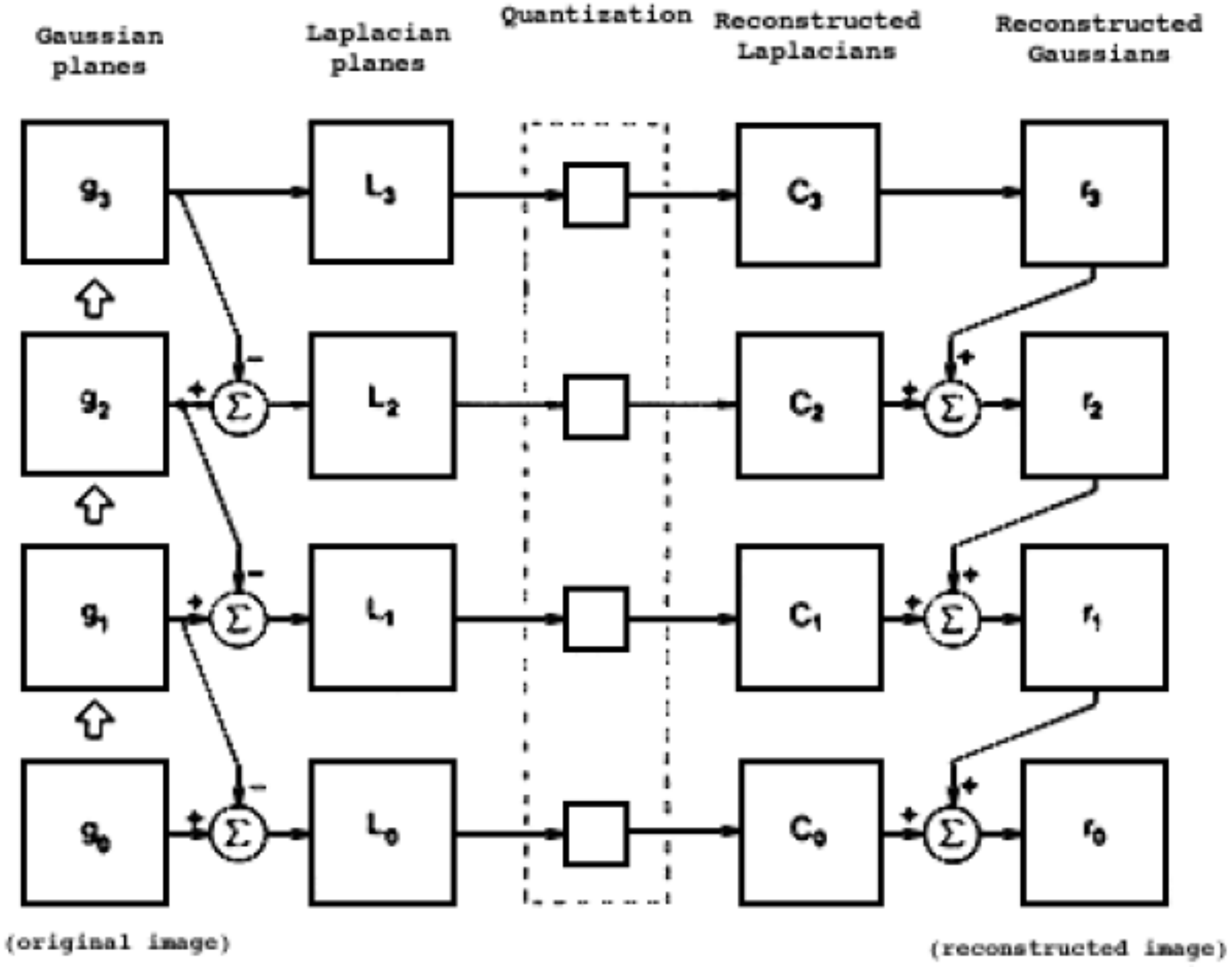
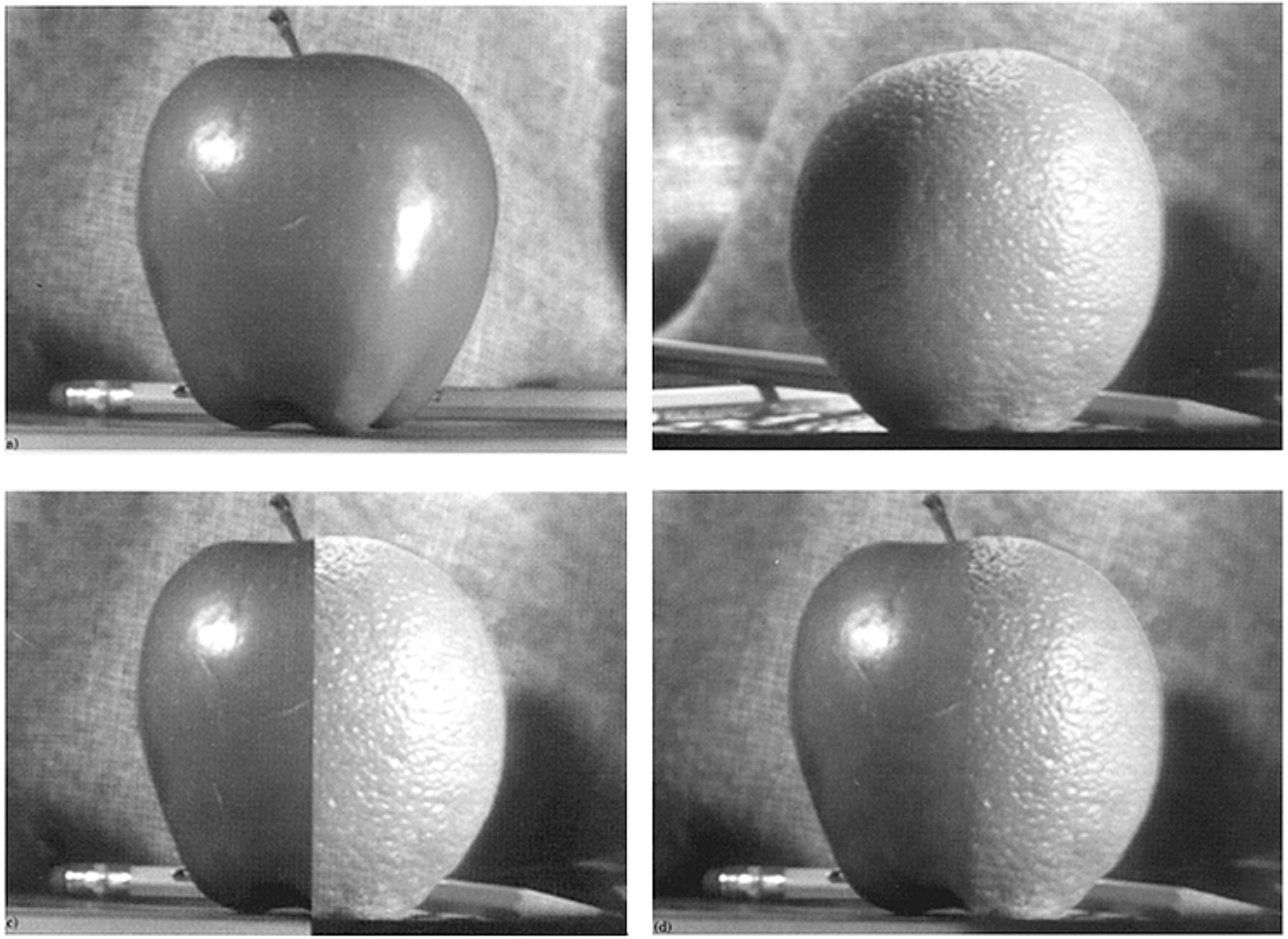


Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image g_0 (lower left) is used to generate Gaussian pyramid levels $g_1, g_2, \ldots$ through repeated local averaging. Levels of the Laplacian pyramid $L_0, L_1, \ldots$ are then computed as the differences 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

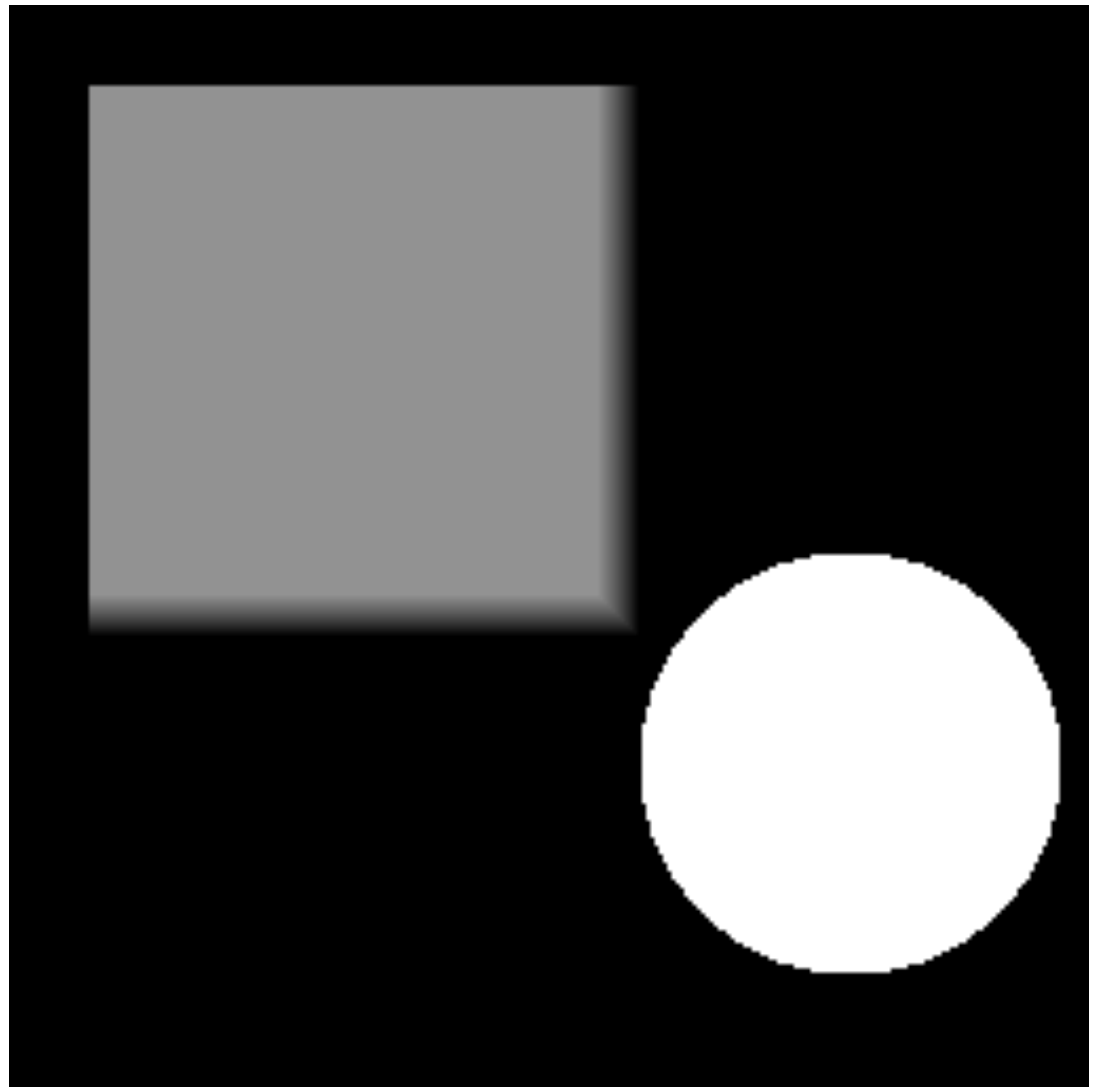
Edge “Detection”

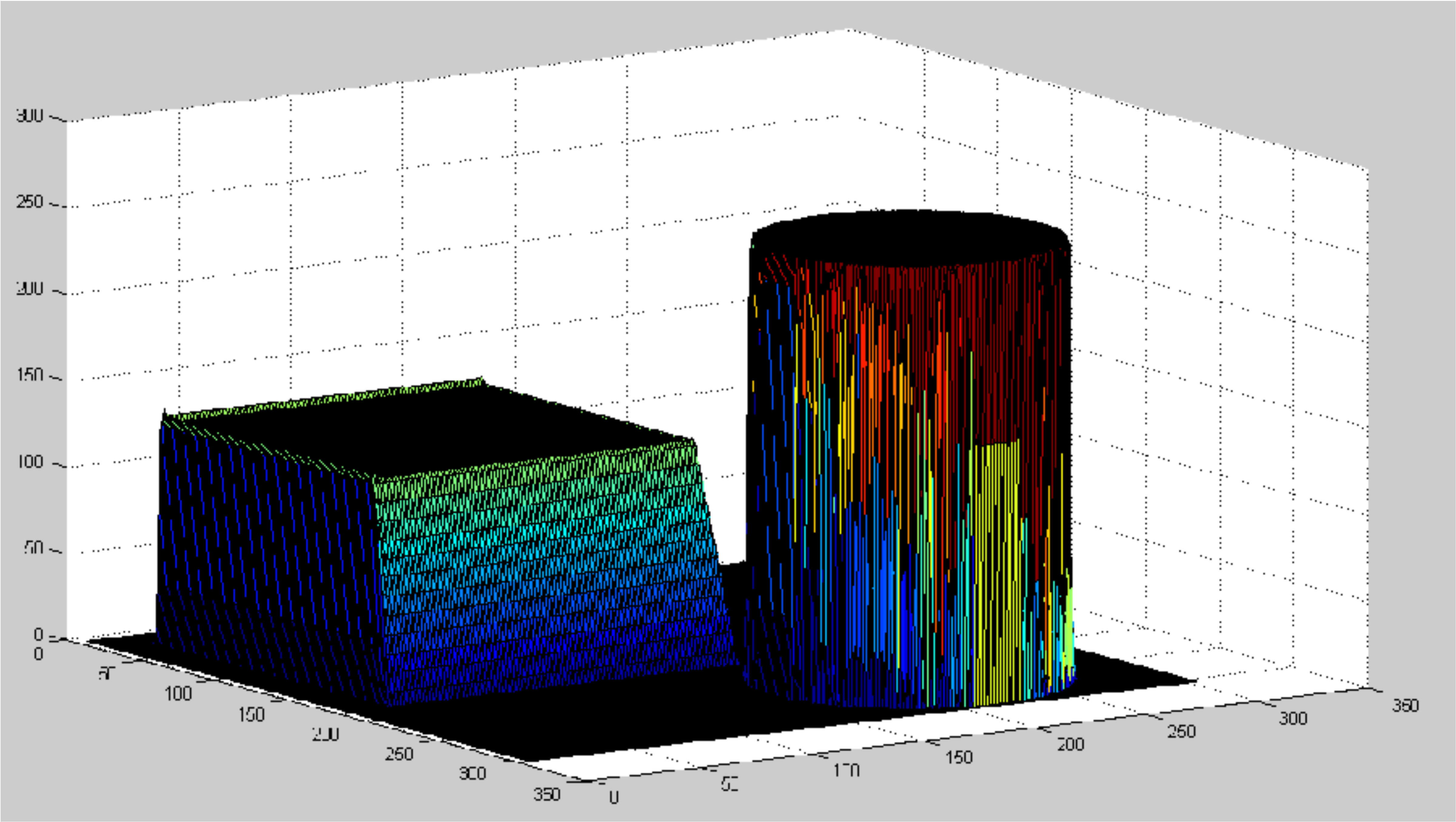
Edge “Detection”

Key Steps

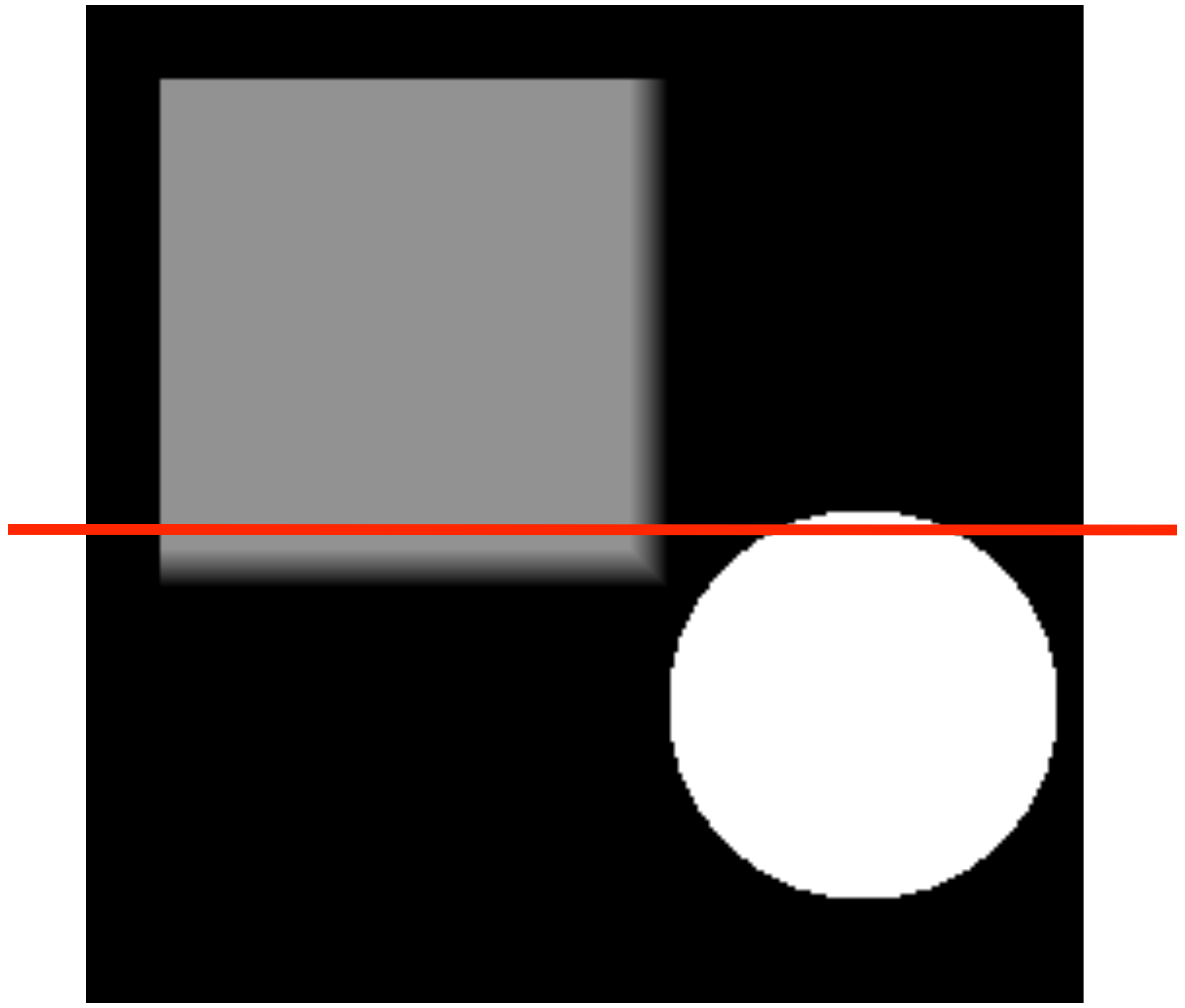
- Image smoothing Gaussian (防噪声，否则 noise 在下一步也会变大)
- Detect edge points (candidates) laplacian
- Edge localization
  - non-mamixa suppression 确保每个 edge 只被标记一次
    非极大值抑制
  - double thresholding

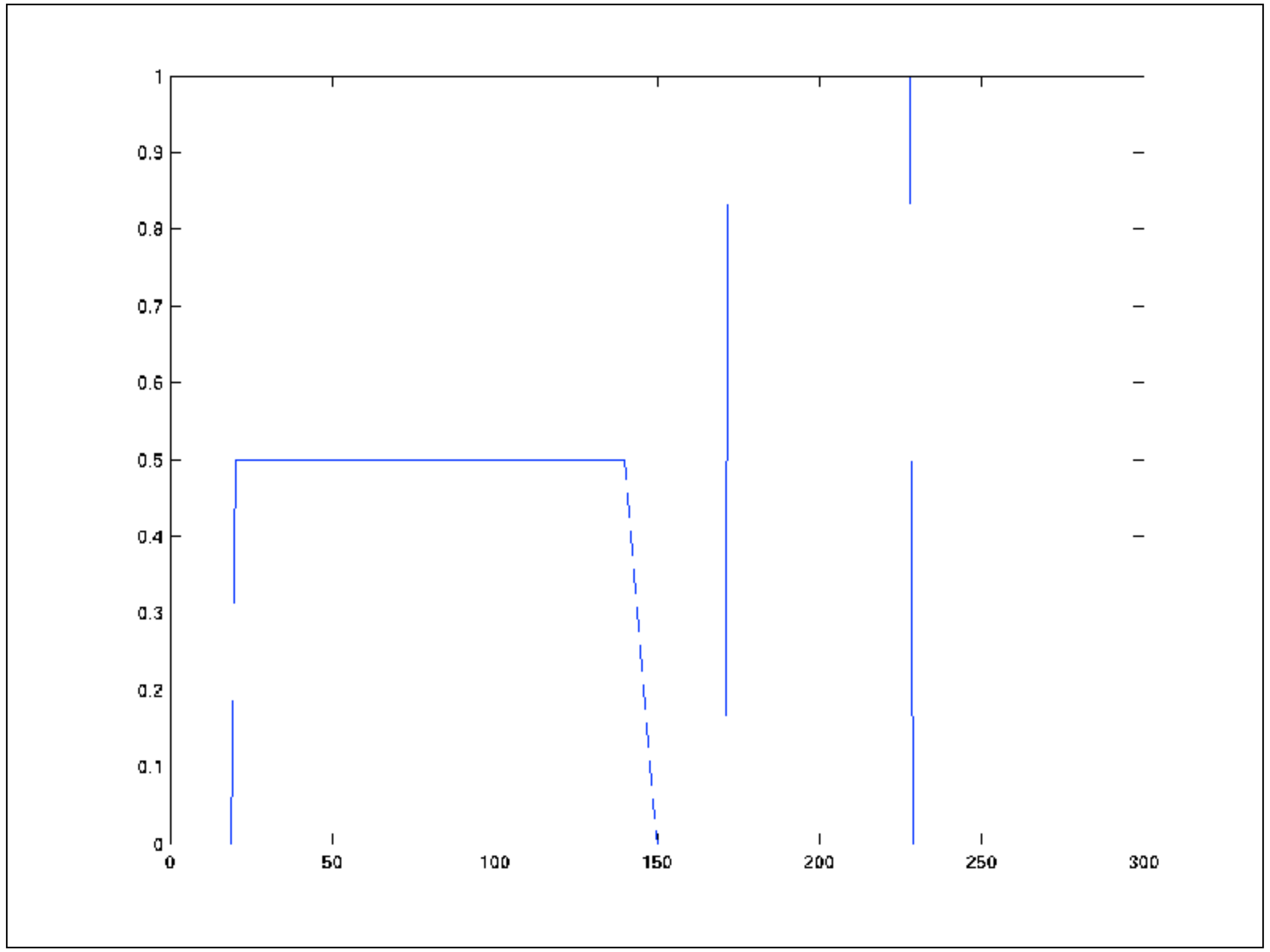
Step and Ramp Edges





Step and Ramp Edges





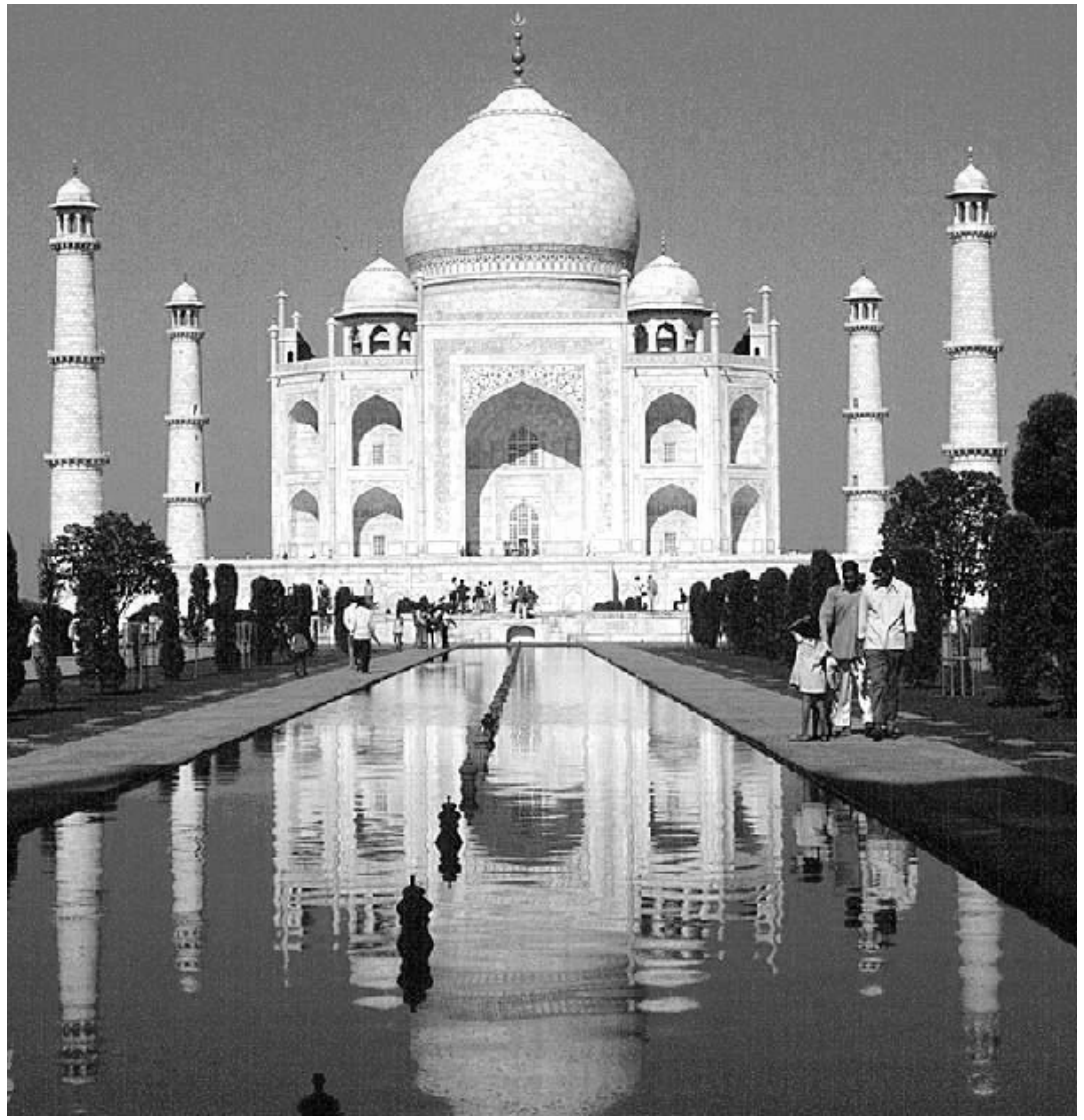
What is An Edge?

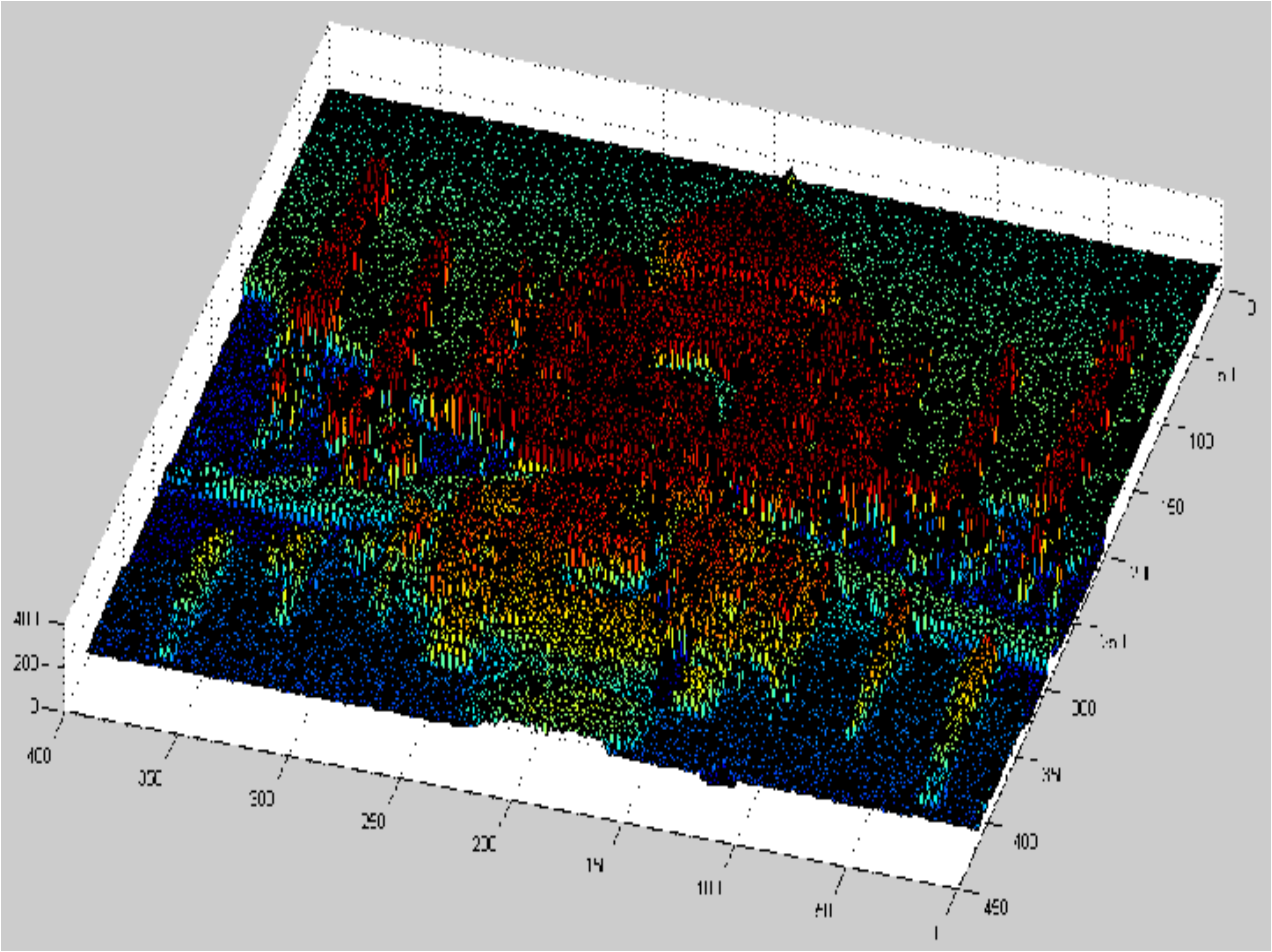
- Boundaries between regions in images:
  - Material change
  - Occlusion boundary
  - Crease boundaries
  - Shadow boundaries

- Sharp changes of gray level: Texture

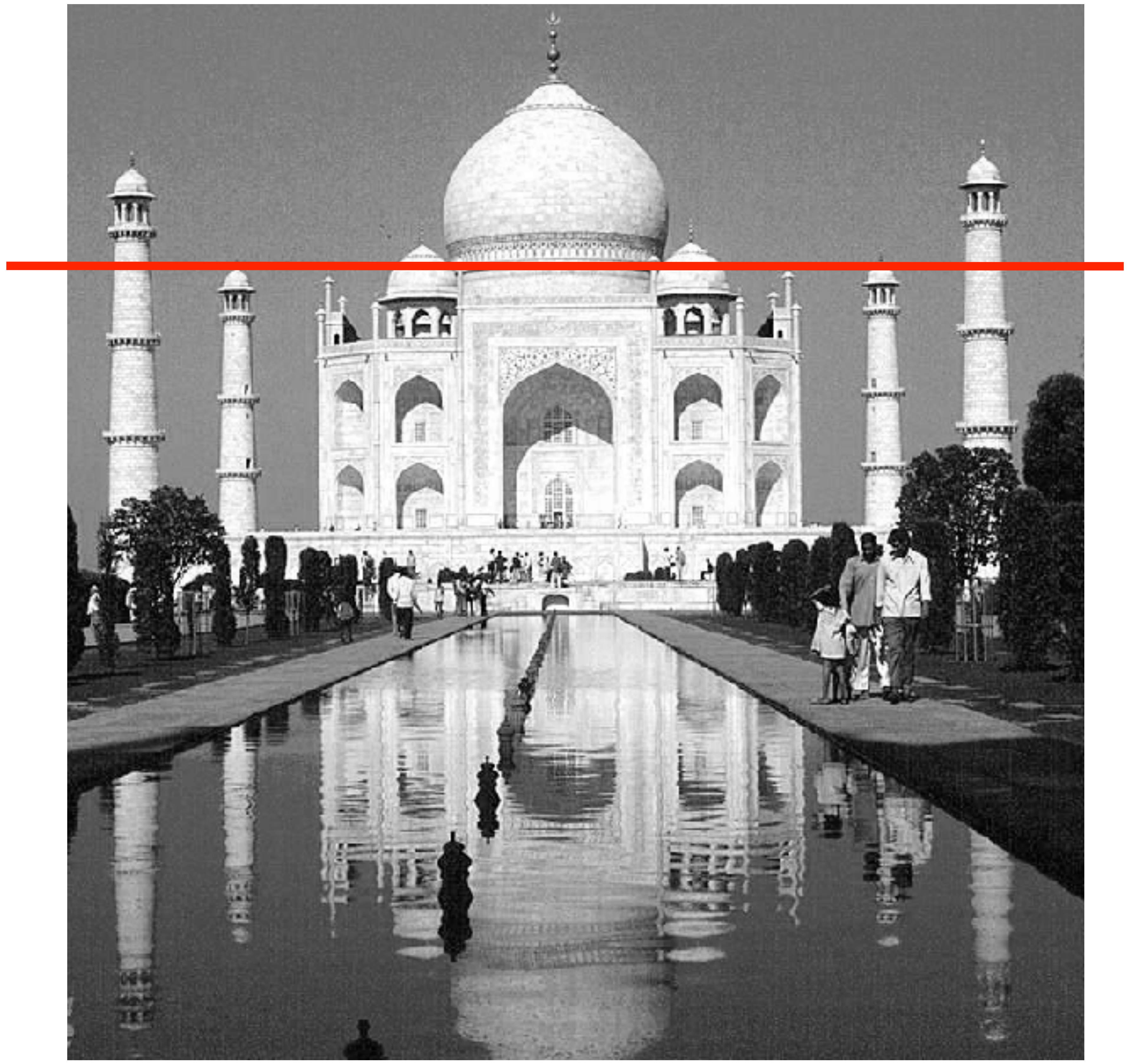
- (Motion boundaries)

Real Edges

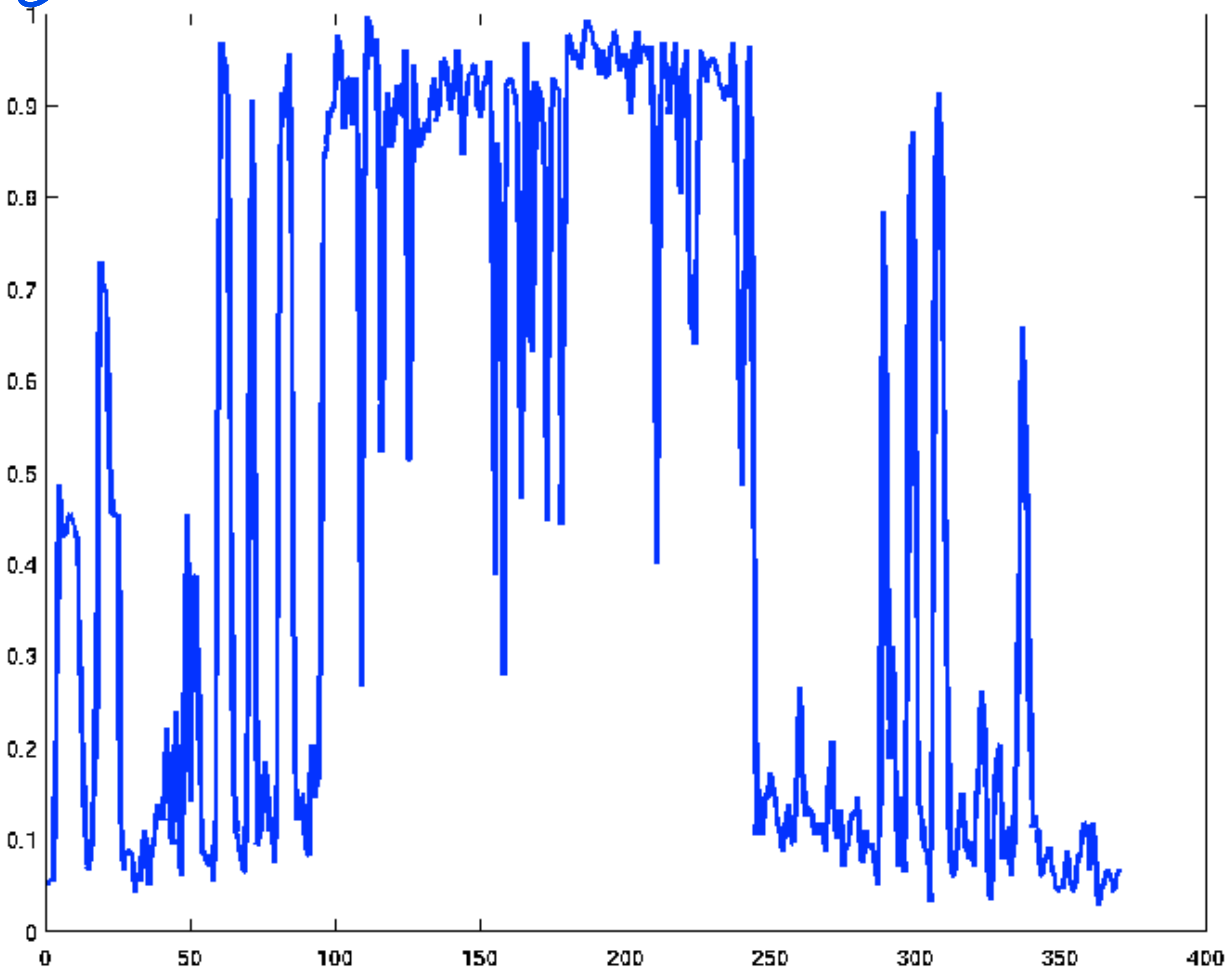




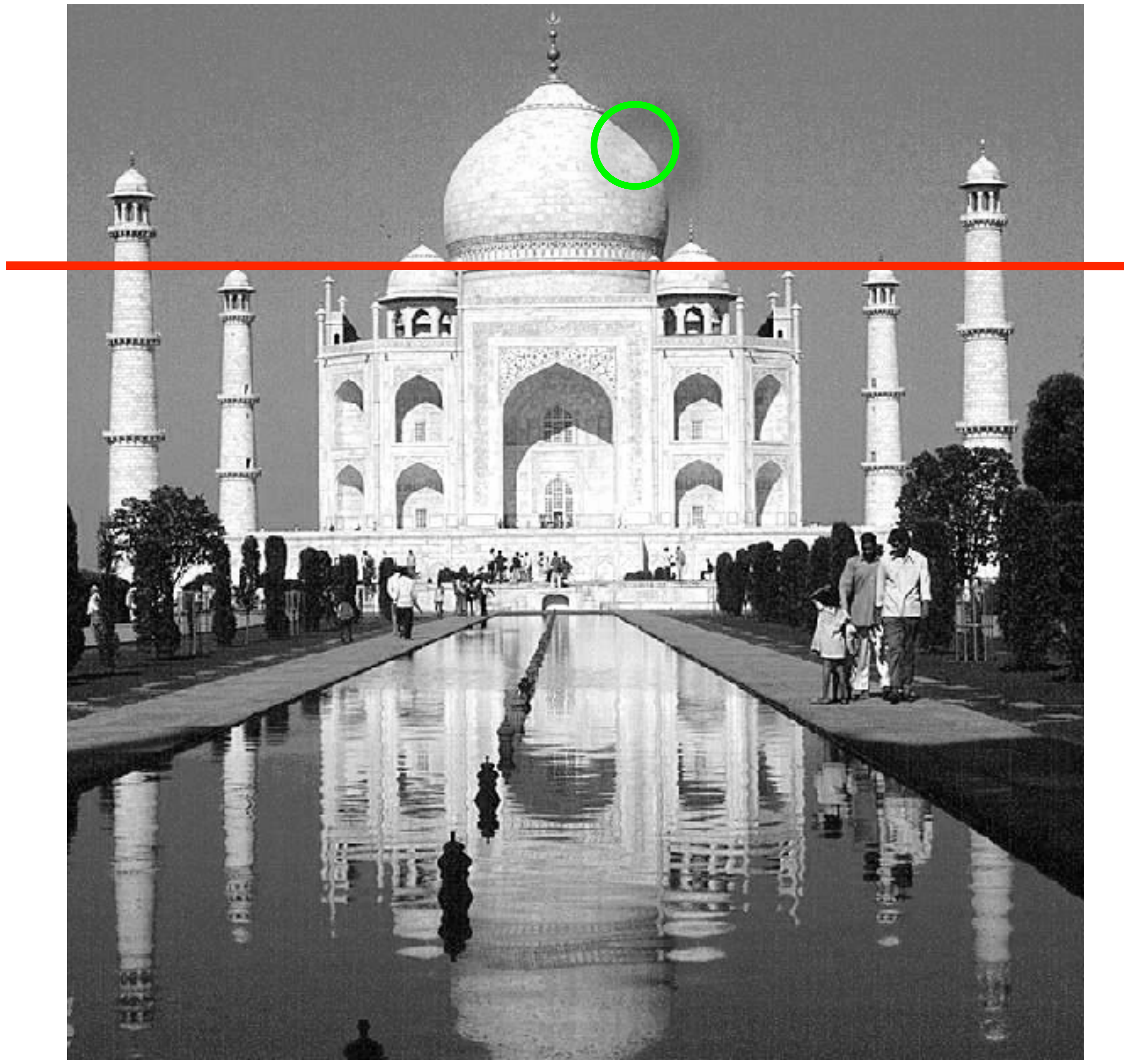
Real Edges

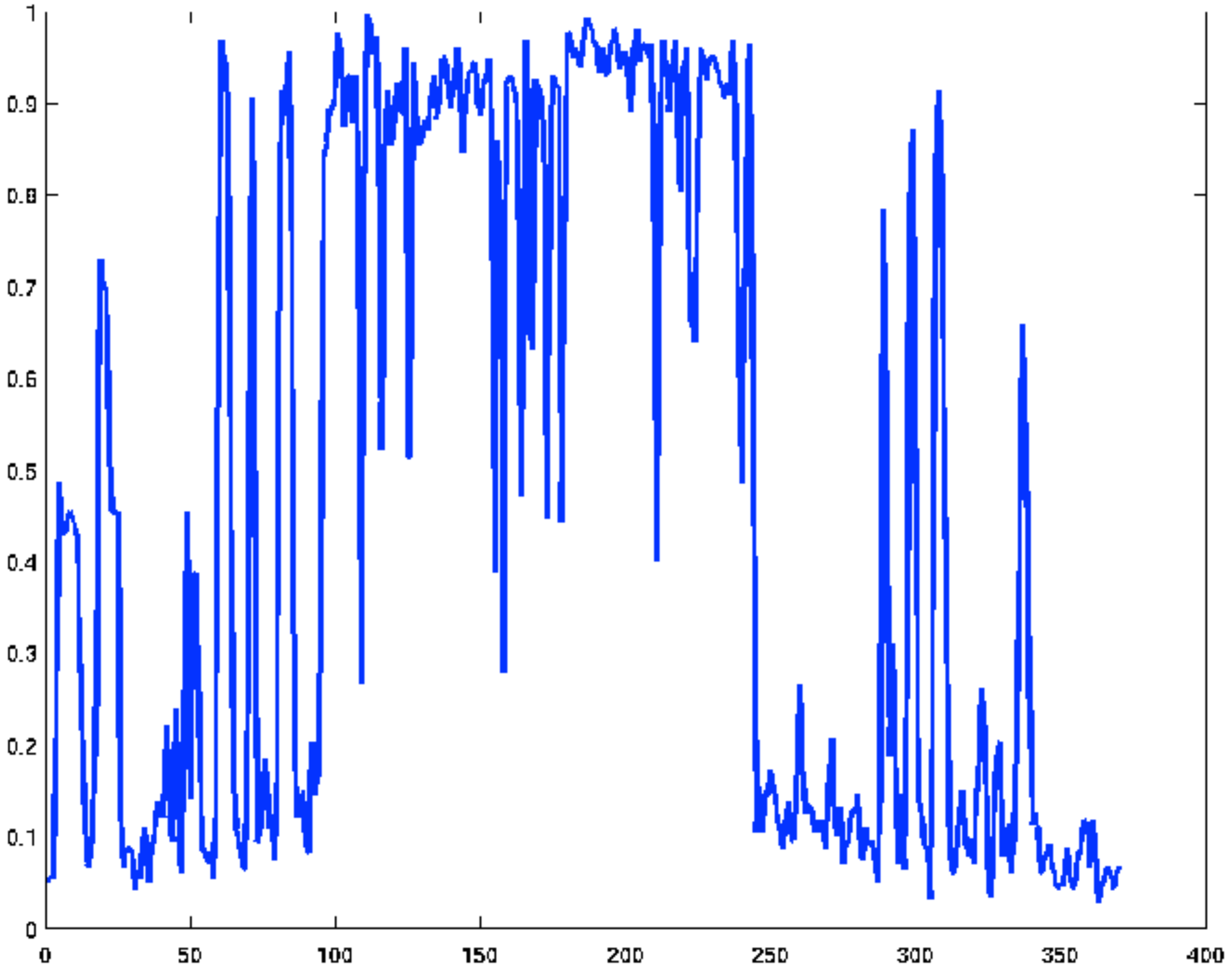


灰度值 (Intensity)

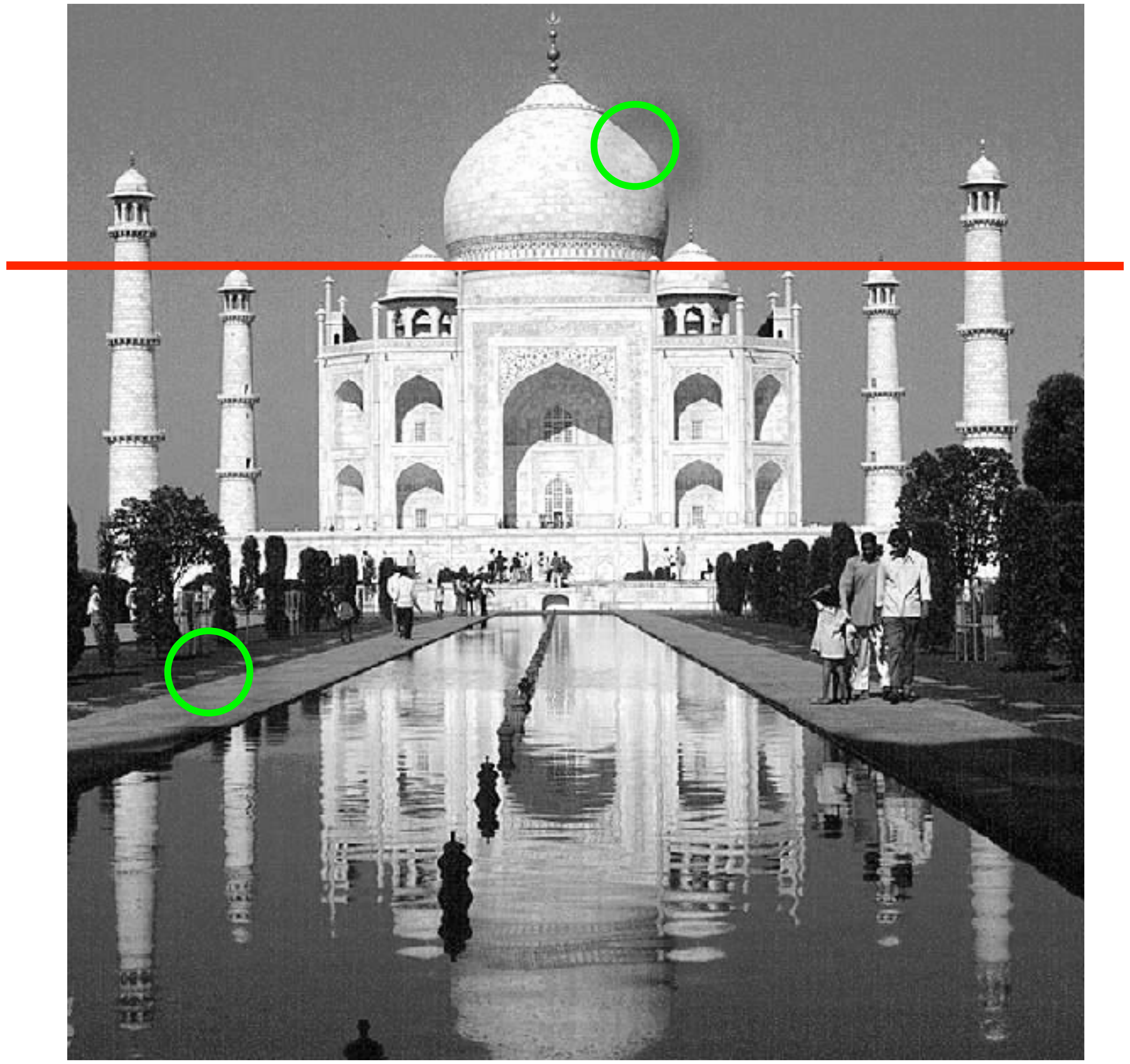


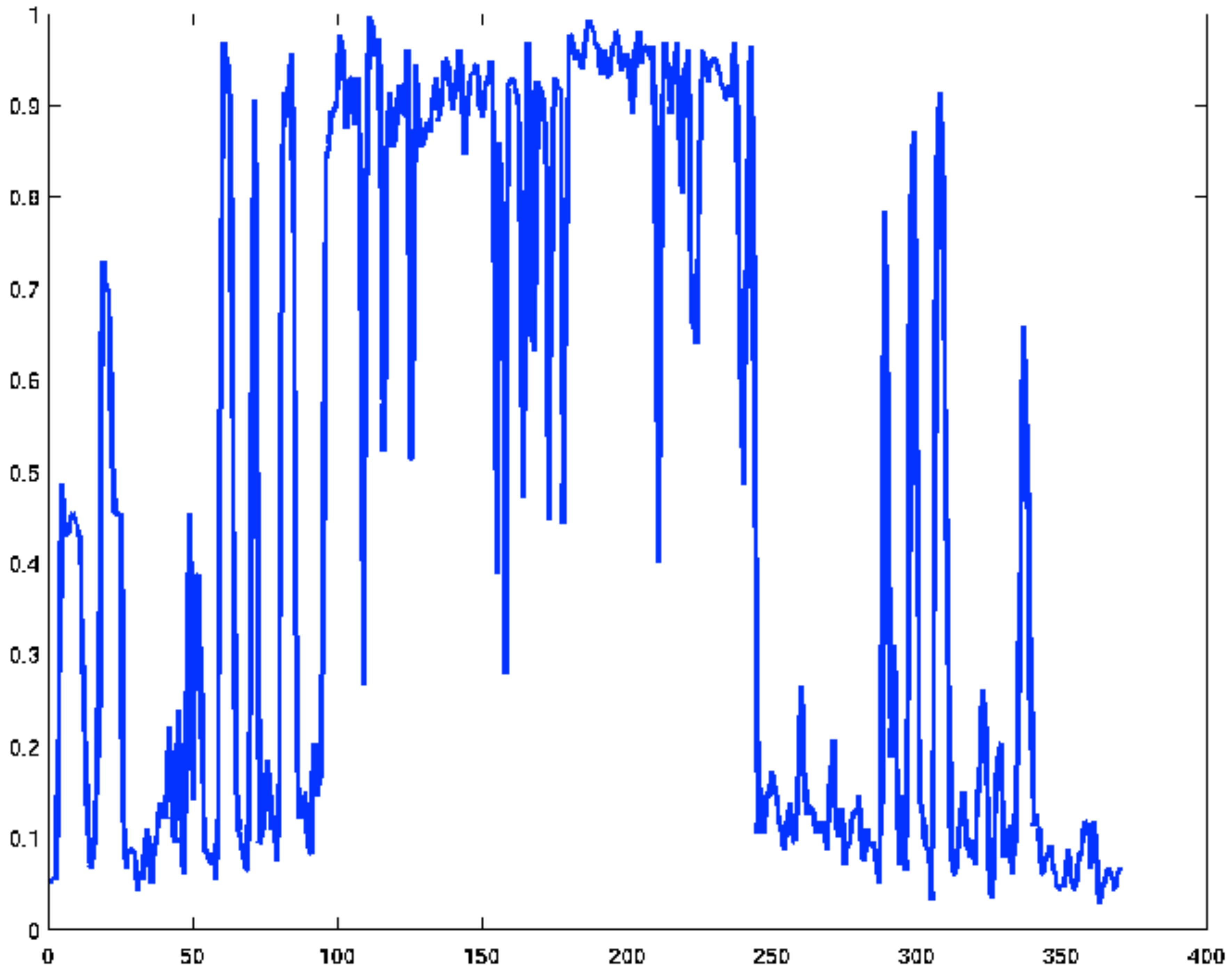
Real Edges



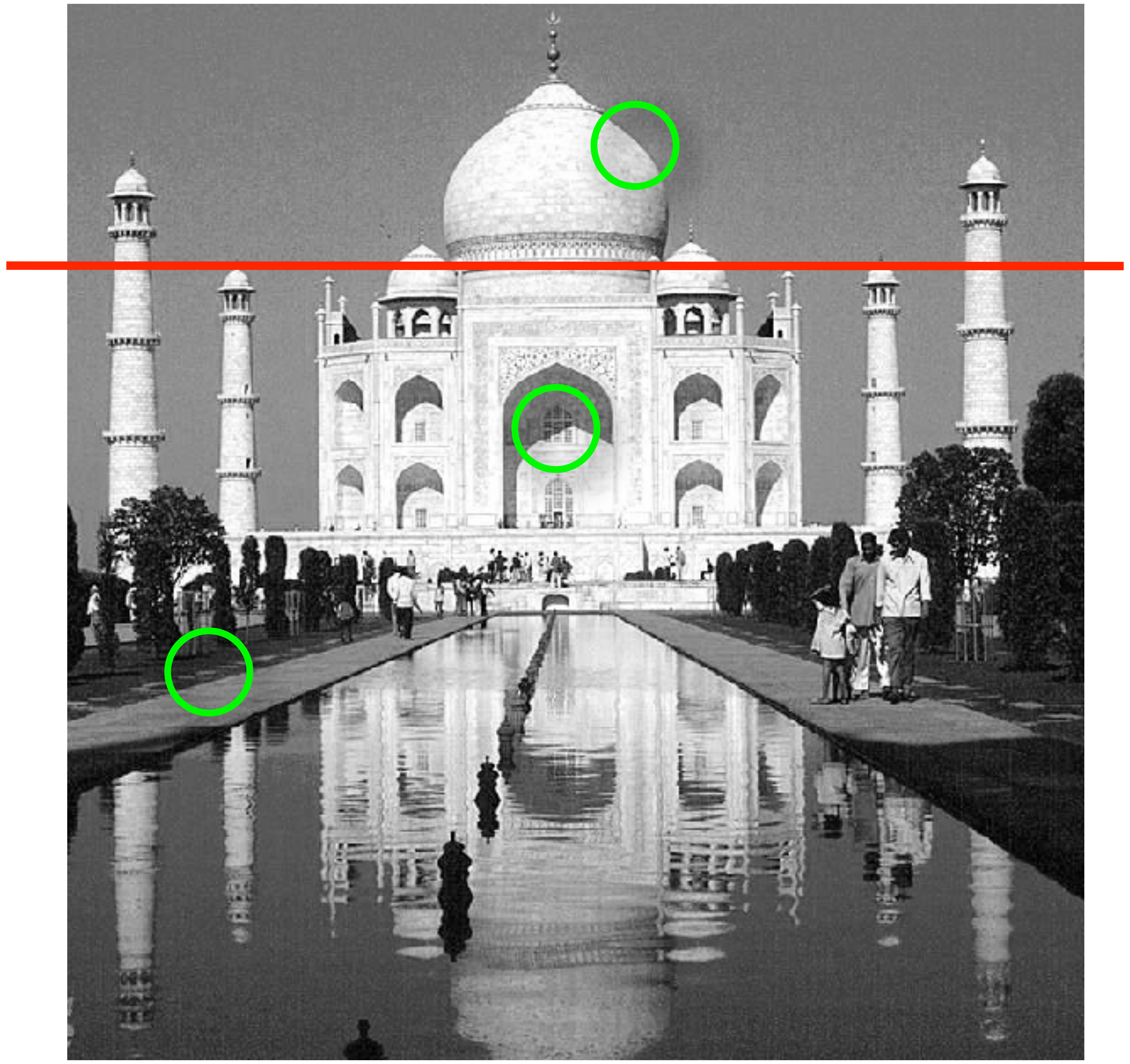


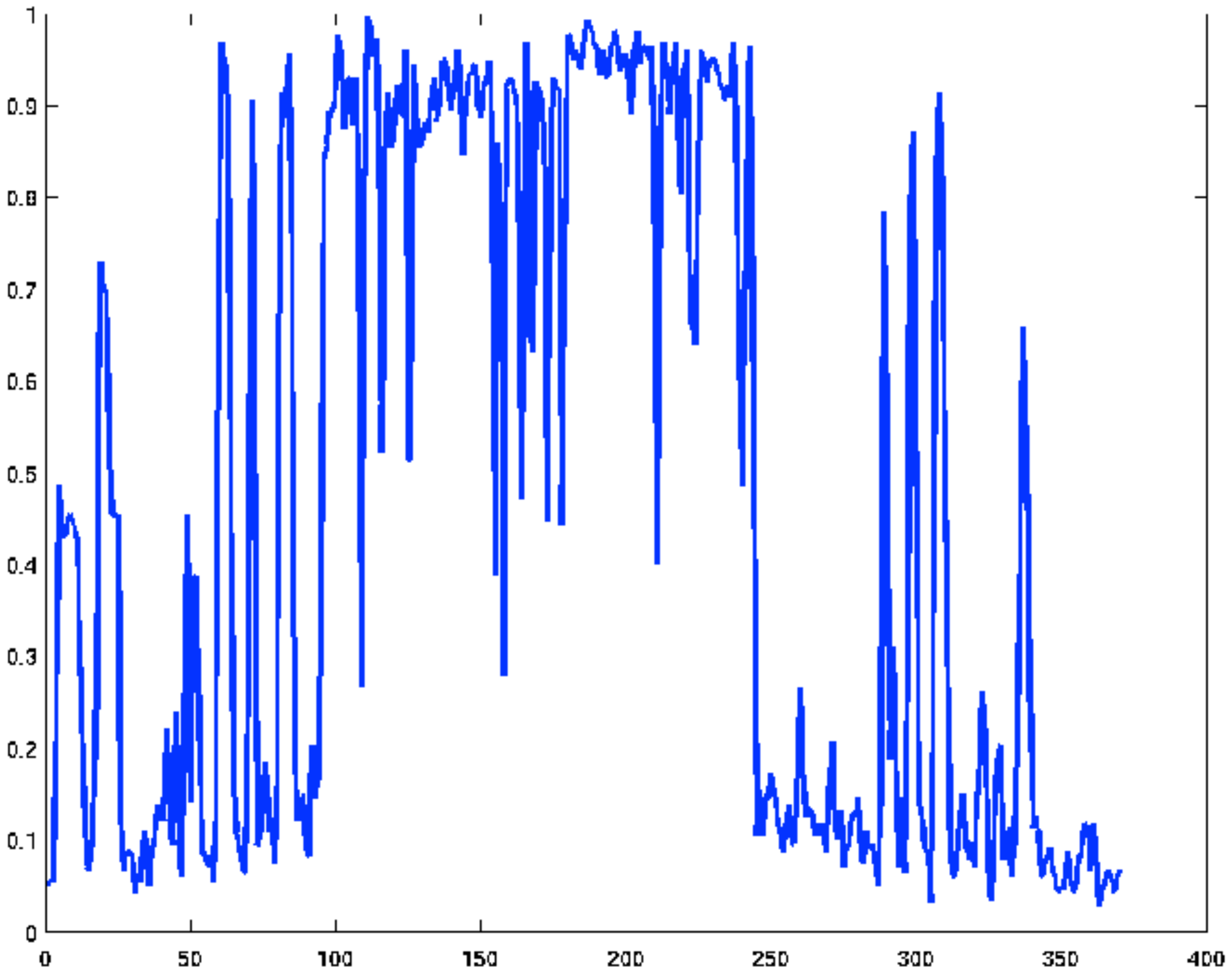
Real Edges



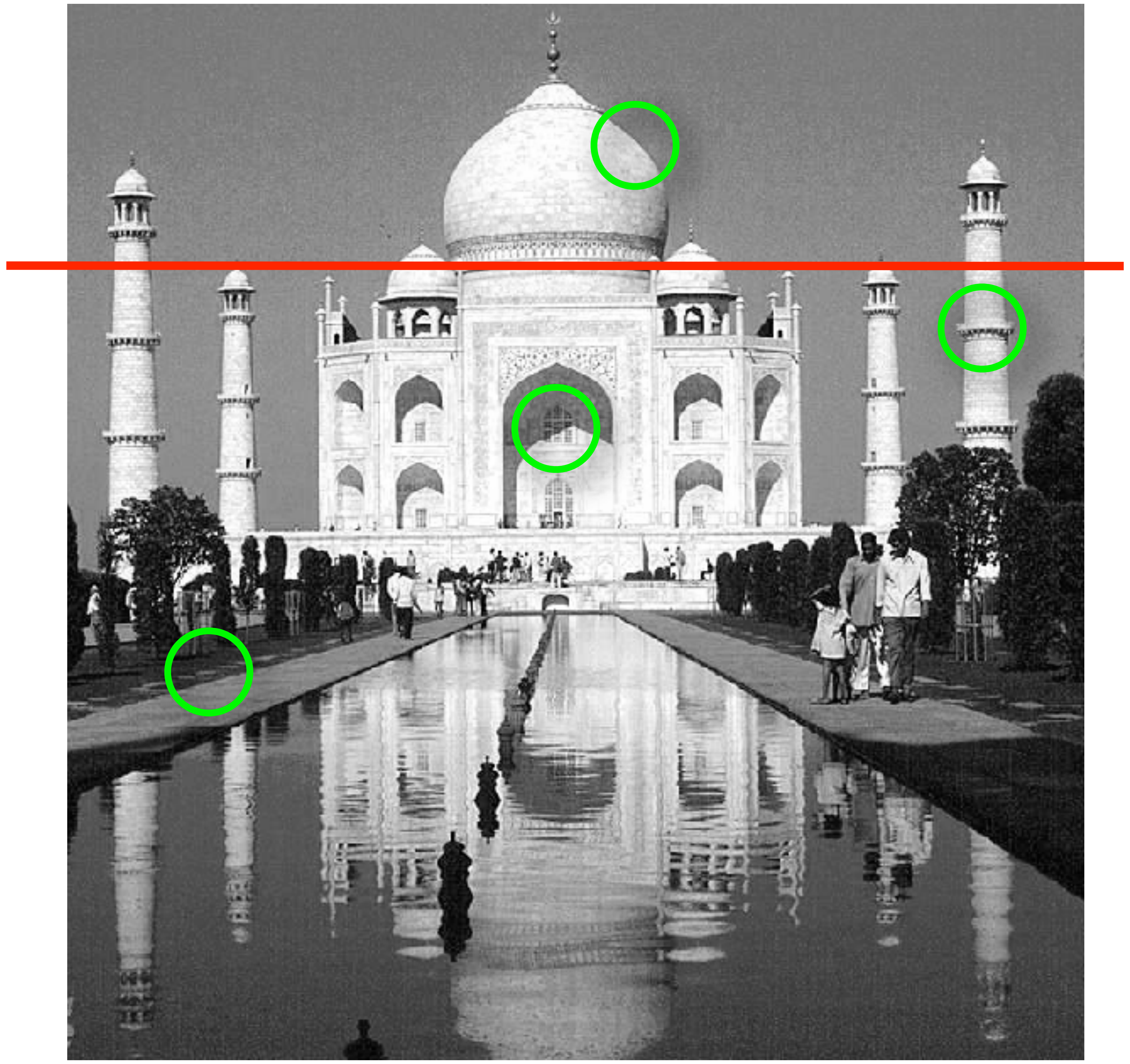


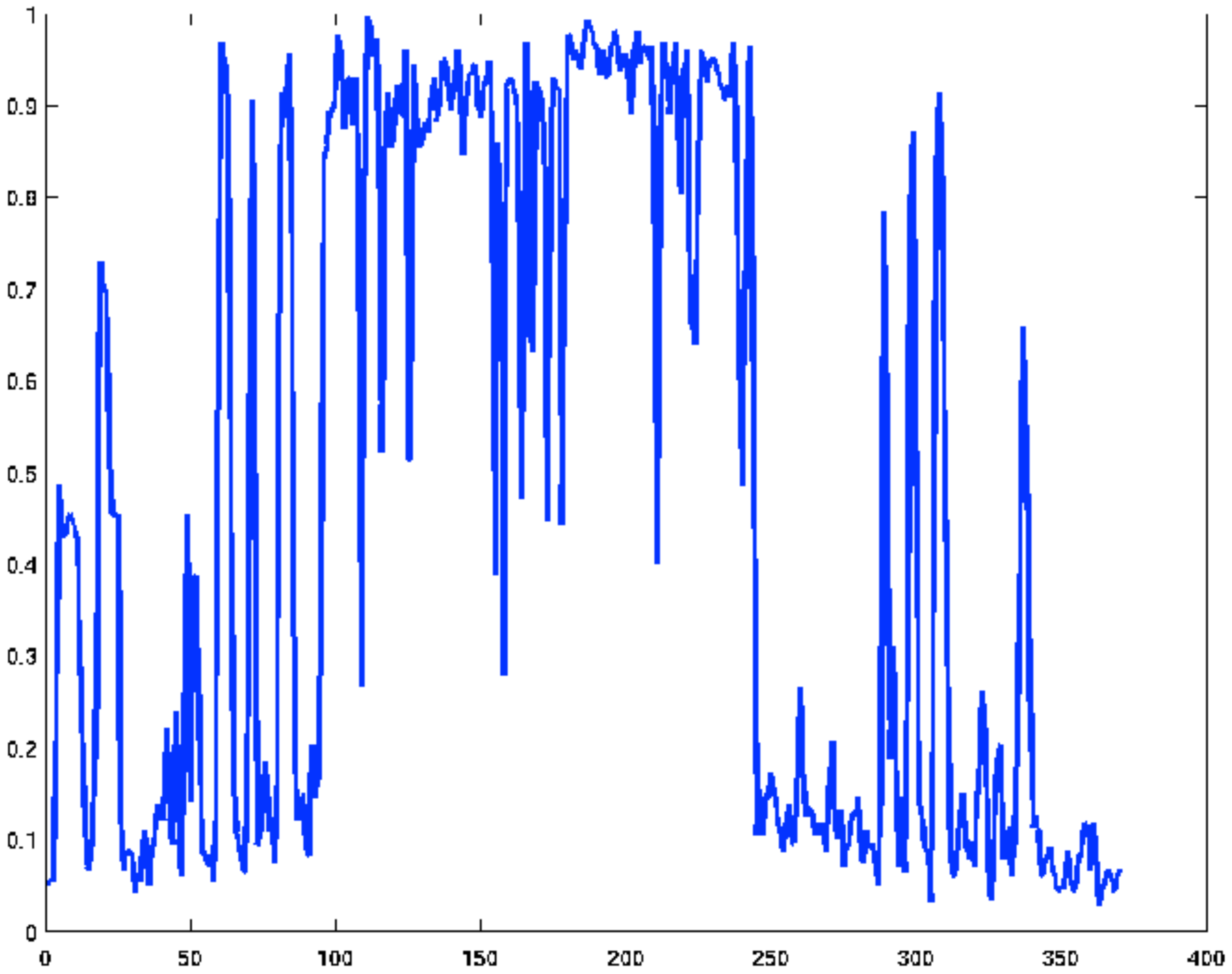
Real Edges



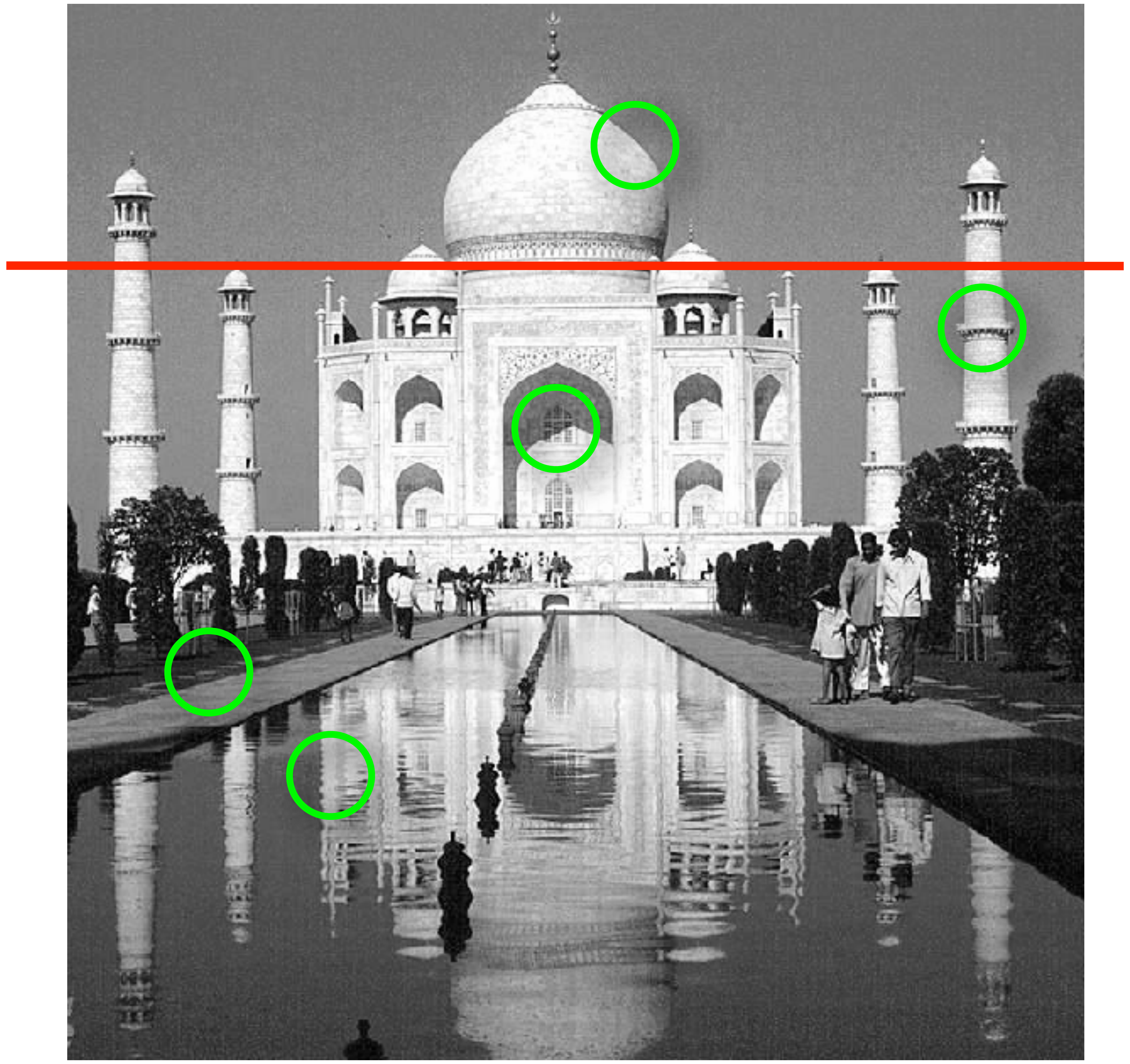


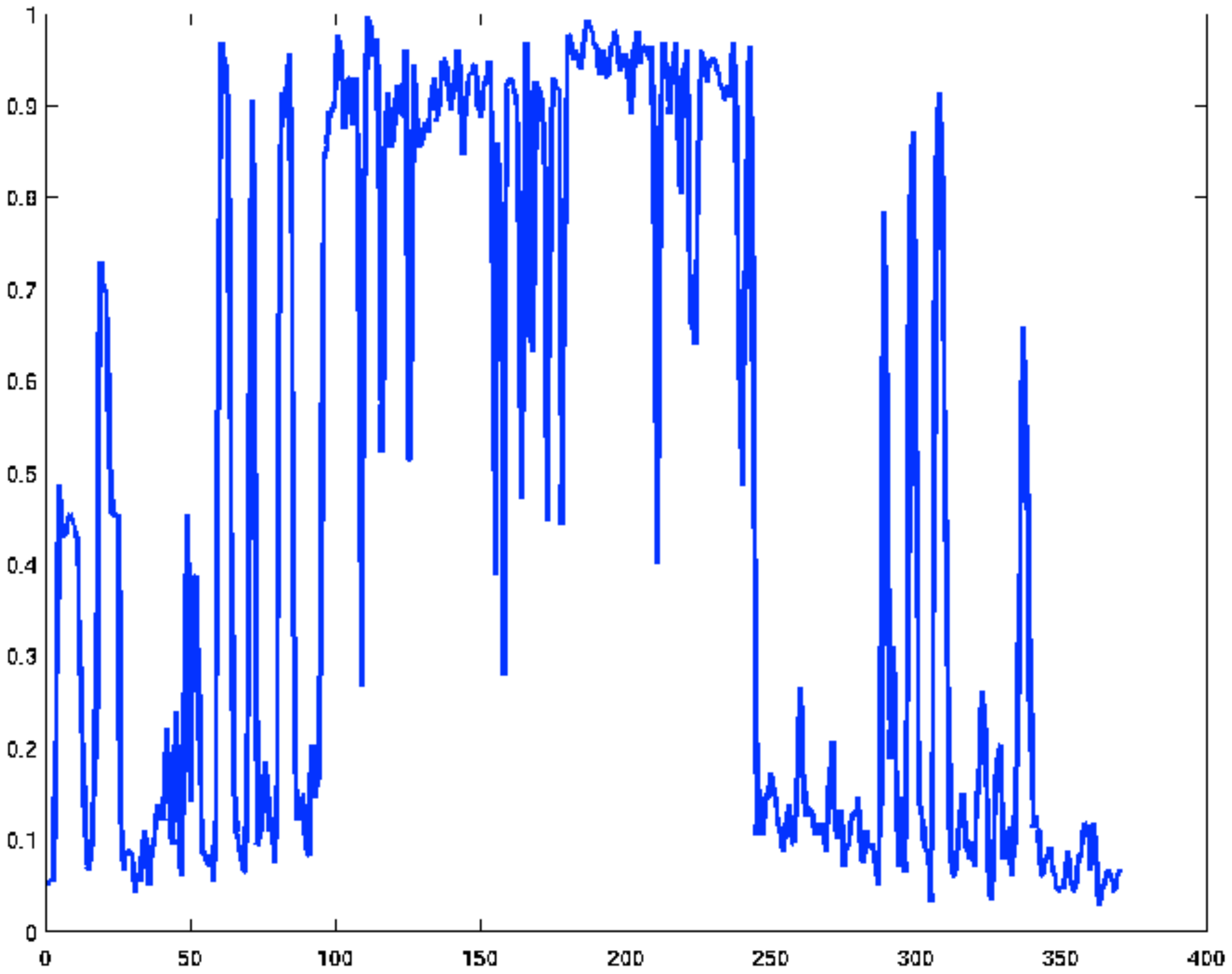
Real Edges





Real Edges





Applications

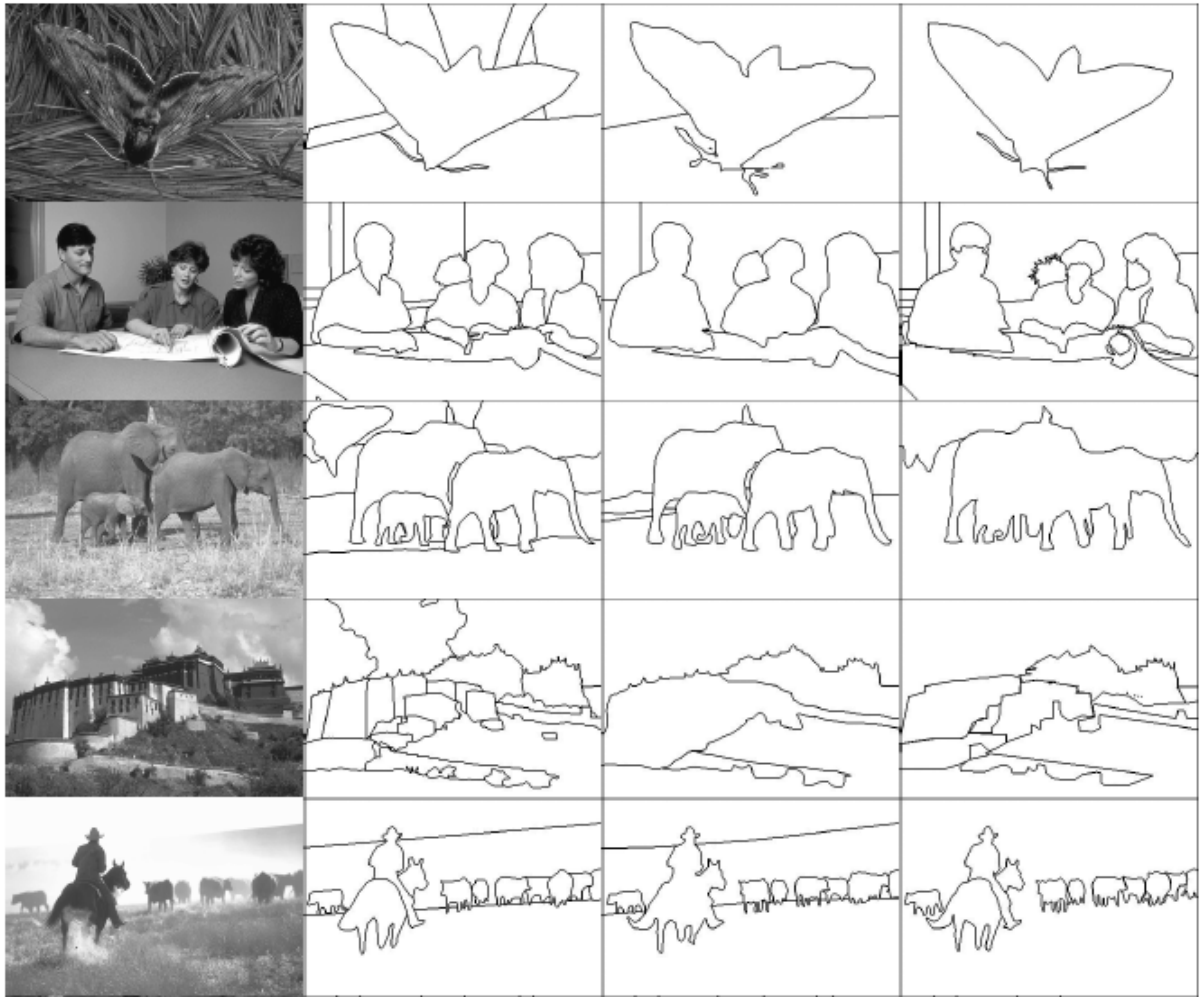
- Segmentation

- 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

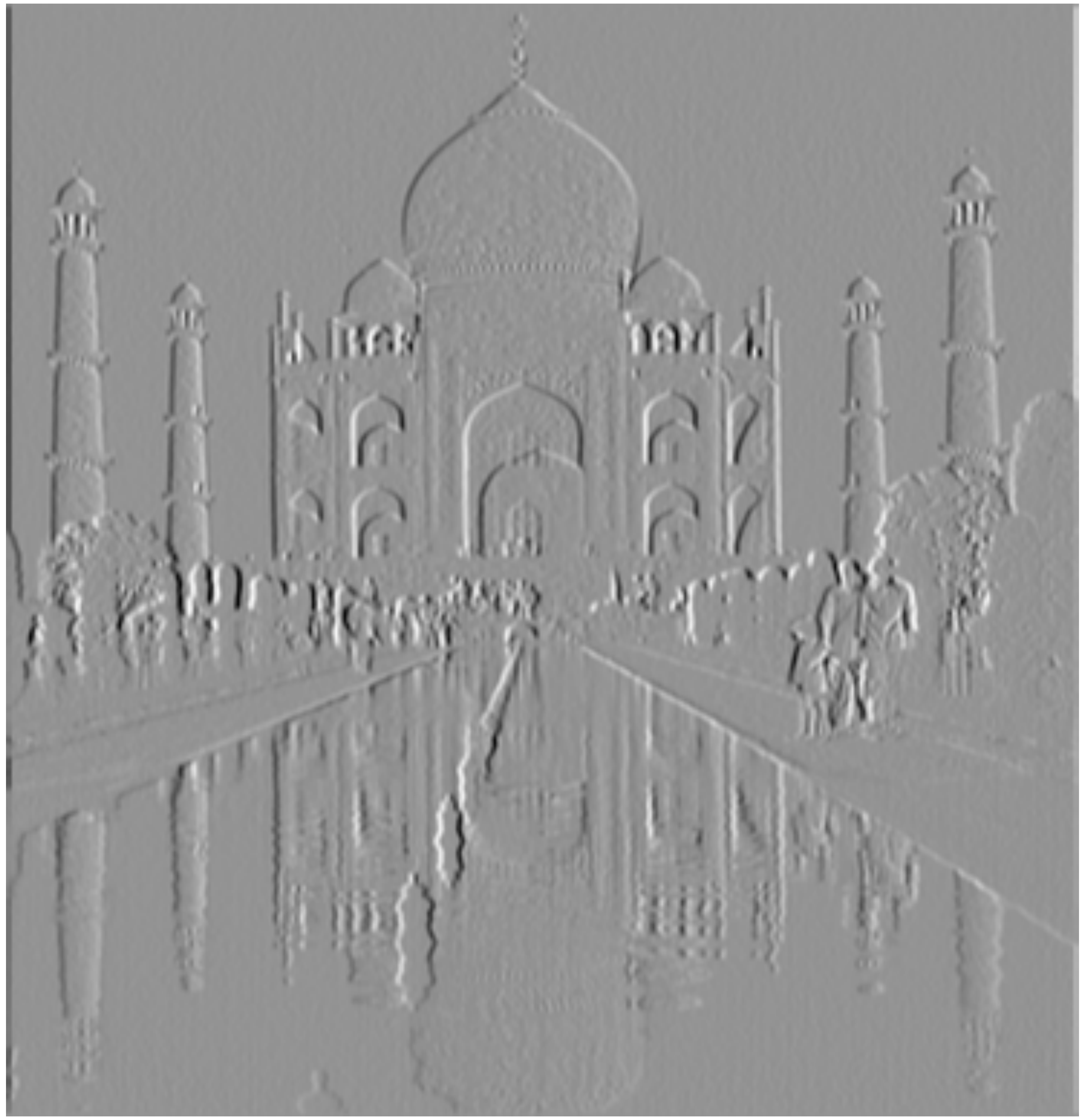
- **Prewitt kernels**

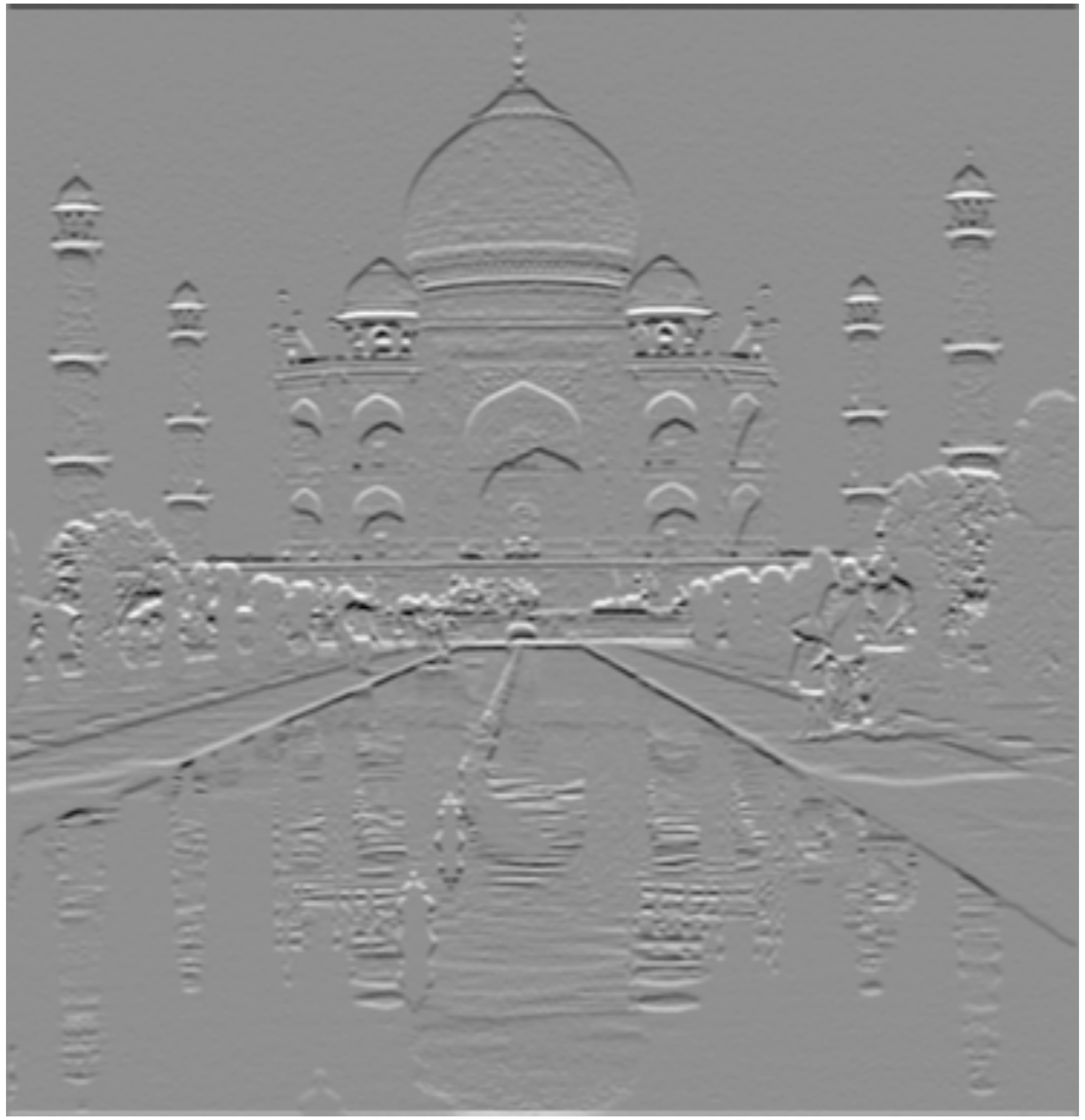
$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **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

Intensity

- **Gradient vector**

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

$$\frac{f(x+dx) - f(x)}{dx}$$

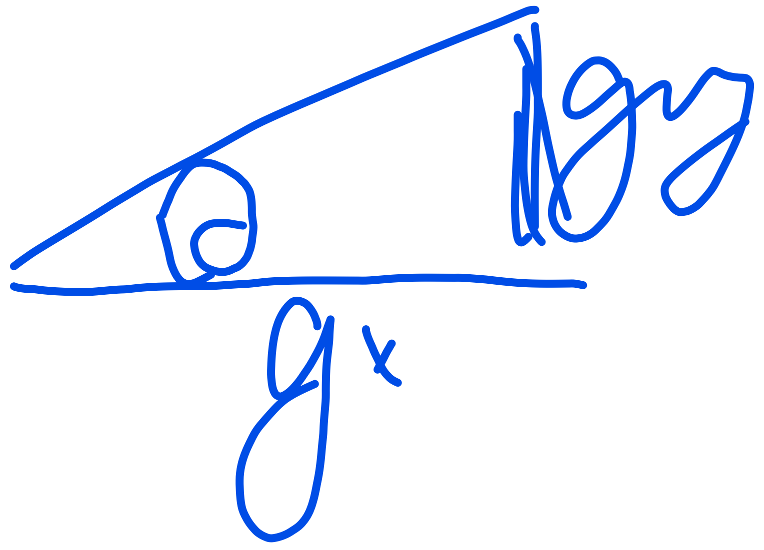
偏导数

- **Gradient magnitude and direction are:**

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

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

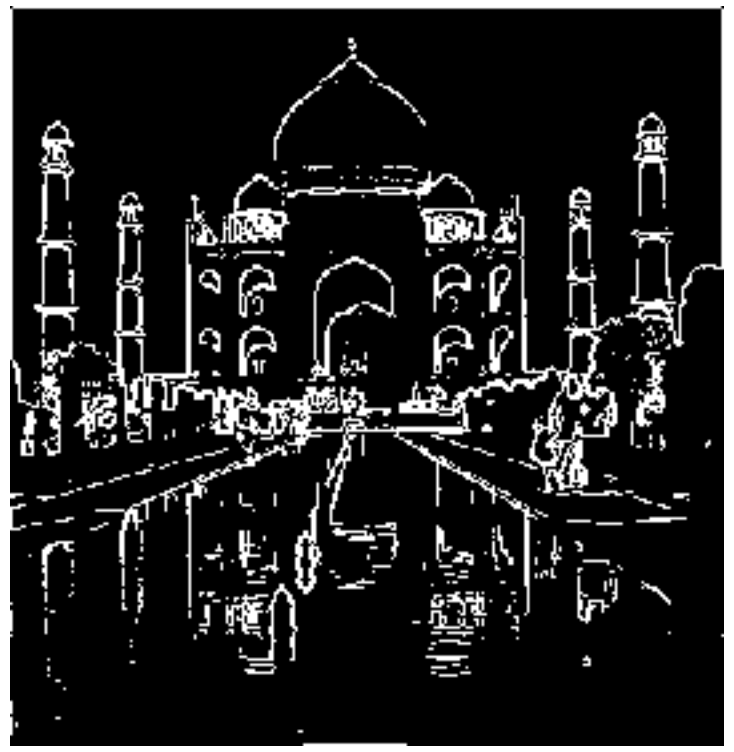
角度



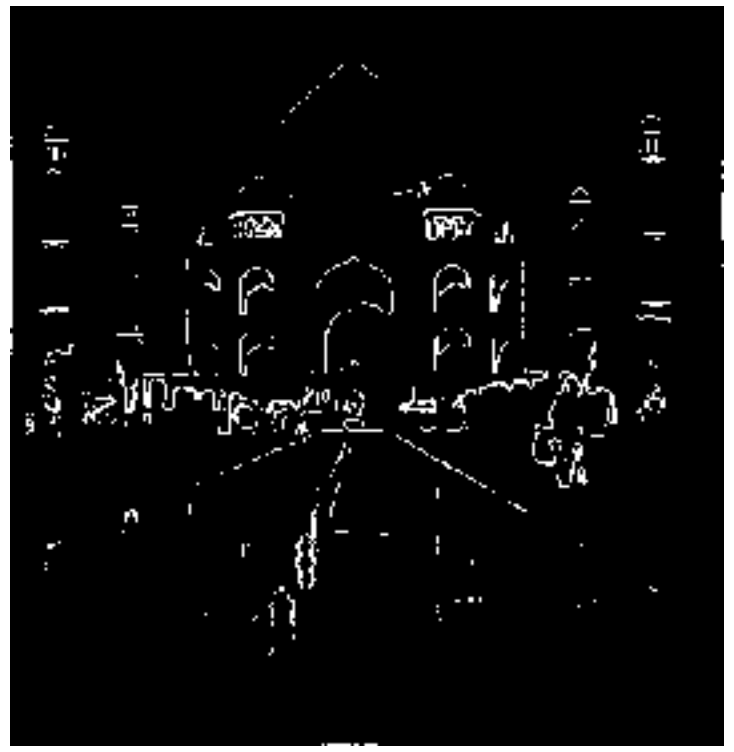
其实的相当于Sobel算子

Simple: Edge Map

threshold



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**

在上传的图像中，展示了使用Canny边缘检测算法处理后的两幅图像，这两幅图像通过高斯函数的不同标准差（σ）来展示边缘检测的结果差异。

Canny边缘检测算法是一种流行的边缘检测算法，它包括几个步骤：

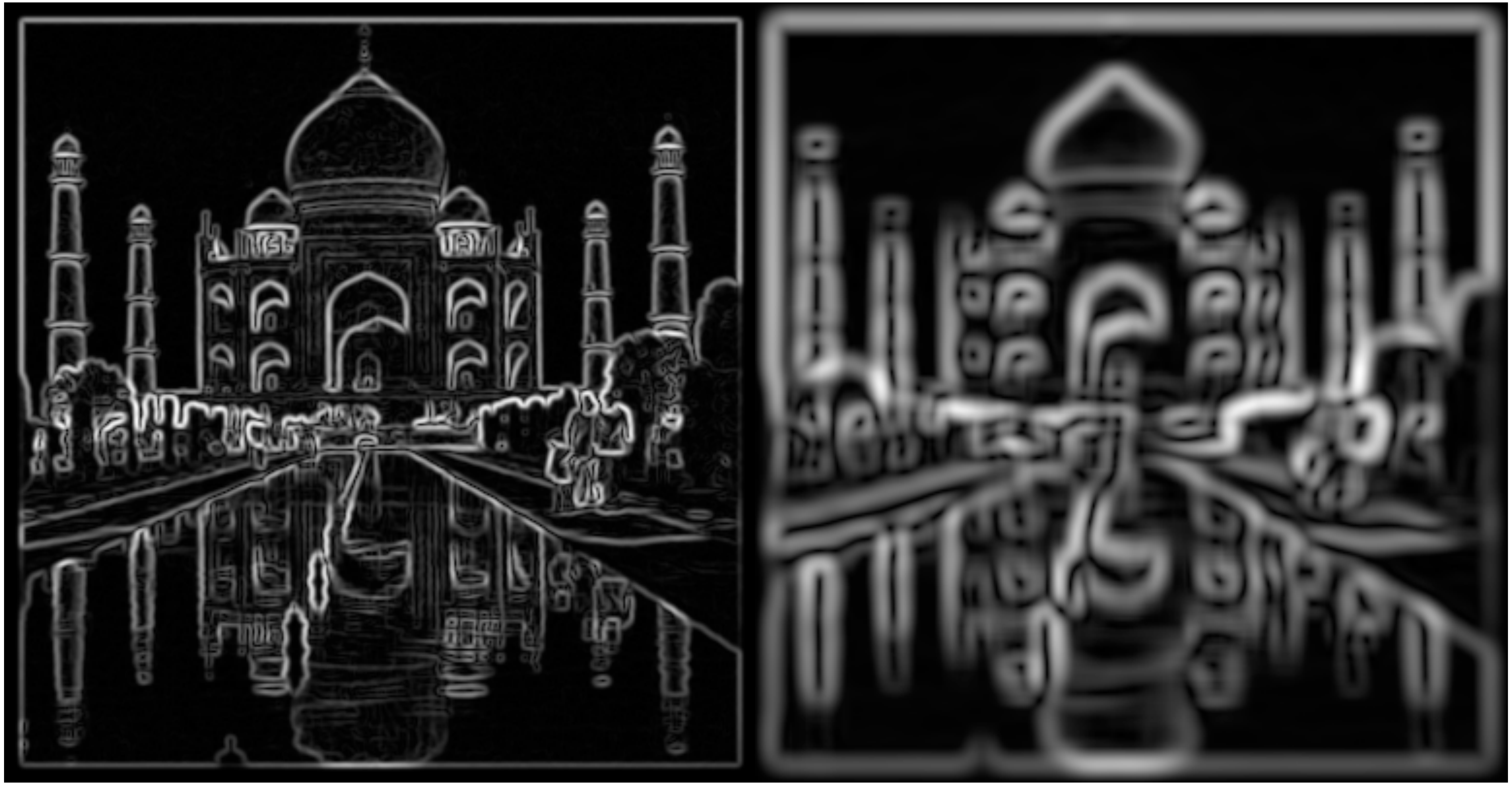
1. **噪声降低**：首先使用高斯滤波器平滑图像，以降低图像噪声。高斯滤波器的标准差 σ 决定了滤波的程度，较小的 σ 值保留更多的细节，而较大的 σ 值则会产生更模糊的效果。
2. **计算梯度**：接下来计算平滑后图像的梯度强度和方向，通常使用Sobel算子来实现。
3. **非极大值抑制**：算法会沿着梯度方向，检查梯度的局部最大值，并抑制（即置为零）非最大值，以确保边缘尽可能细。
4. **双阈值**：确定潜在边缘的强度，通常使用两个阈值：一个高阈值用来标识强边缘，一个低阈值用来标识弱边缘。
5. **边缘跟踪**：通过滞后阈值来确定真正的边缘，即从强边缘开始，追踪所有与强边缘相连的弱边缘。

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

不是 max 就抑制

1. 对图像中的每一个像素点，确定该点的梯度方向。
2. 确定与梯度方向正交的方向上相邻的两个像素点。换句话说，就是在边缘的法线方向上考虑两侧的像素点。
3. 比较当前像素点的梯度强度与这两个相邻像素点的梯度强度。
4. 如果当前像素点的梯度强度不是最大的，即它不是邻域内梯度强度的局部最大值，则认为该点不在边缘上，将其梯度强度设置为零。
5. 如果当前像素点的梯度强度是最大的，那么它可能位于边缘上，保留其梯度强度。

You

如何确定一个点的梯度方向呢

ChatGPT

确定图像中一个点的梯度方向通常涉及以下步骤：

1. **计算梯度分量**：首先，你需要计算该点在水平和垂直方向上的梯度分量，通常表示为 g_x 和 g_y。这些可以通过应用Sobel、Prewitt或Roberts等梯度算子来获得。
2. **梯度方向计算**：有了这两个分量，梯度方向 θ 可以通过反三角函数atan2来计算，公式为：
   $\theta = \mathrm{atan2}(g_y, g_x)$
   atan2函数考虑了梯度分量的符号，能够提供正确的角度值，即使是当 g_x 为0的时候。梯度方向 θ 的范围通常是从 -π 到 π 或者从 -180度到180度。
3. **方向离散化**：在边缘检测算法（如Canny算法）中，通常需要将梯度方向离散化，也就是将连续的梯度方向量化为几个特定的方向。例如，可以将梯度方向量化为水平、垂直和两个对角方向（0°，45°，90°，135°），这样做可以简化后续的非极大值抑制步骤。

梯度方向告诉我们图像中亮度变化最大的方向，这个信息在边缘检测中是至关重要的，因为边缘通常是沿着亮度变化最大的方向。在Canny边缘检测算法中，非极大值抑制将使用这个方向来检查一

请再详细说一下第一步，用一个像素点举例子

ChatGPT

在第一步中，为了计算图像中一个点的梯度分量，我们使用差分方法来近似导数。这里以一个3x3的像素区域为例，其中中心像素点的坐标为 (x, y)，周围的点分别对应其左右、上下的像素值。我们假设有如下的像素矩阵：

$$\begin{matrix} I(x-1,y-1) & I(x,y-1) & I(x+1,y-1) \\ I(x-1,y) & I(x,y) & I(x+1,y) \\ I(x-1,y+1) & I(x,y+1) & I(x+1,y+1) \end{matrix}$$

这里，$I(x, y)$ 表示在坐标 (x, y) 处的像素亮度值。

为了计算梯度分量 g_x 和 g_y，我们可以使用如下公式：

- 水平梯度分量 g_x 可以近似为：
  $g_x \approx I(x+1, y) - I(x-1, y)$
  这是因为 g_x 表示在 x 方向（水平方向）上亮度的变化率。
- 垂直梯度分量 g_y 可以近似为：
  $g_y \approx I(x, y+1) - I(x, y-1)$
  这是因为 g_y 表示在 y 方向（垂直方向）上亮度的变化率。

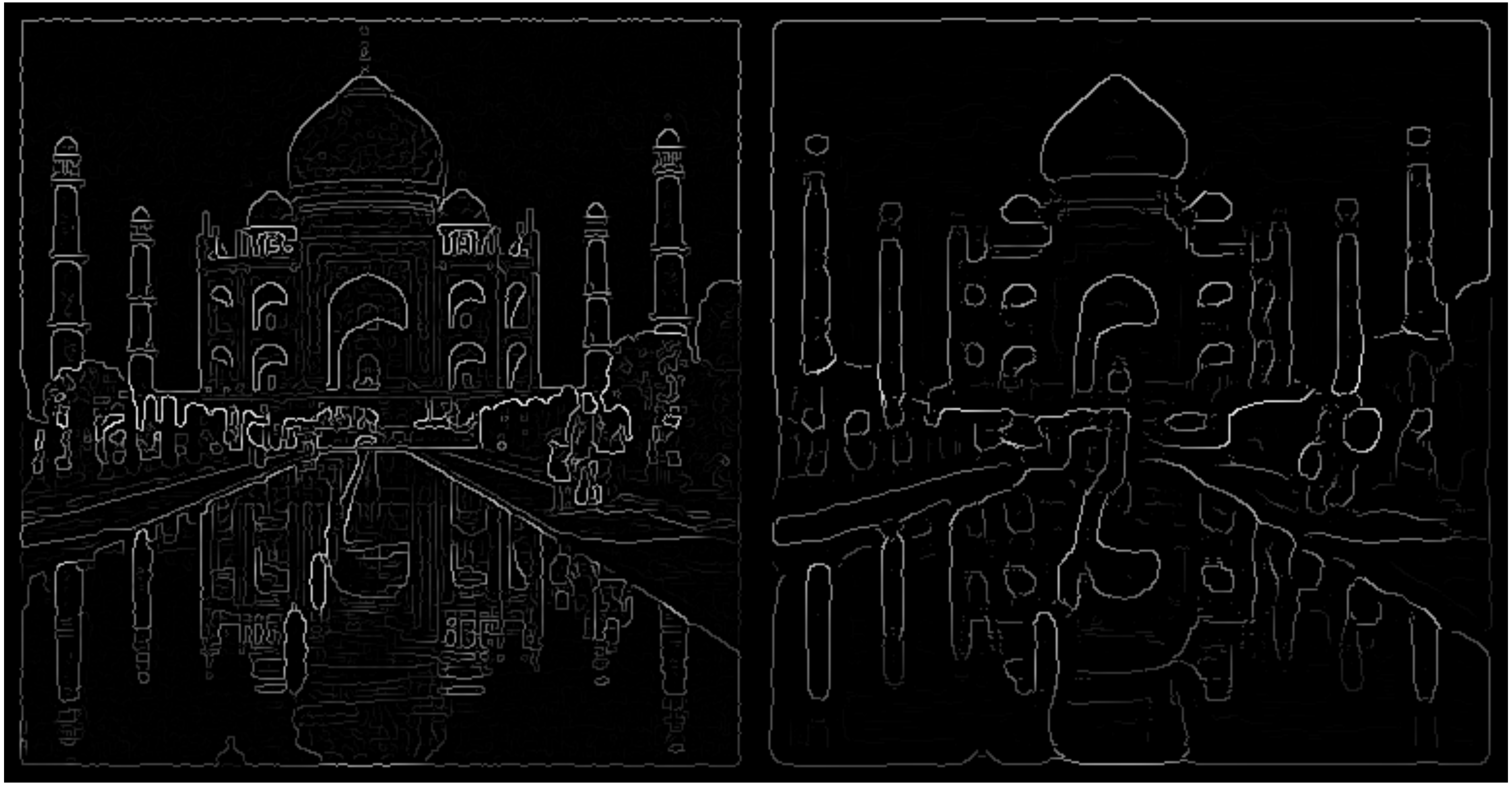
在实际应用中，为了提高梯度计算的鲁棒性，我们通常使用像素差的加权和来计算 g_x 和 g_y。例如，Sobel算子使用以下卷积核来计算 g_x 和 g_y：

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I$$

其中，* 表示卷积操作。通过这种方式，不仅考虑了相邻像素点，而且还赋予了不同位置的像素不同的权重，从而增强了边缘的响应。使用这些算子可以有效地计算出中心像素点 (x, y) 的梯度分

Canny: Non-max Output

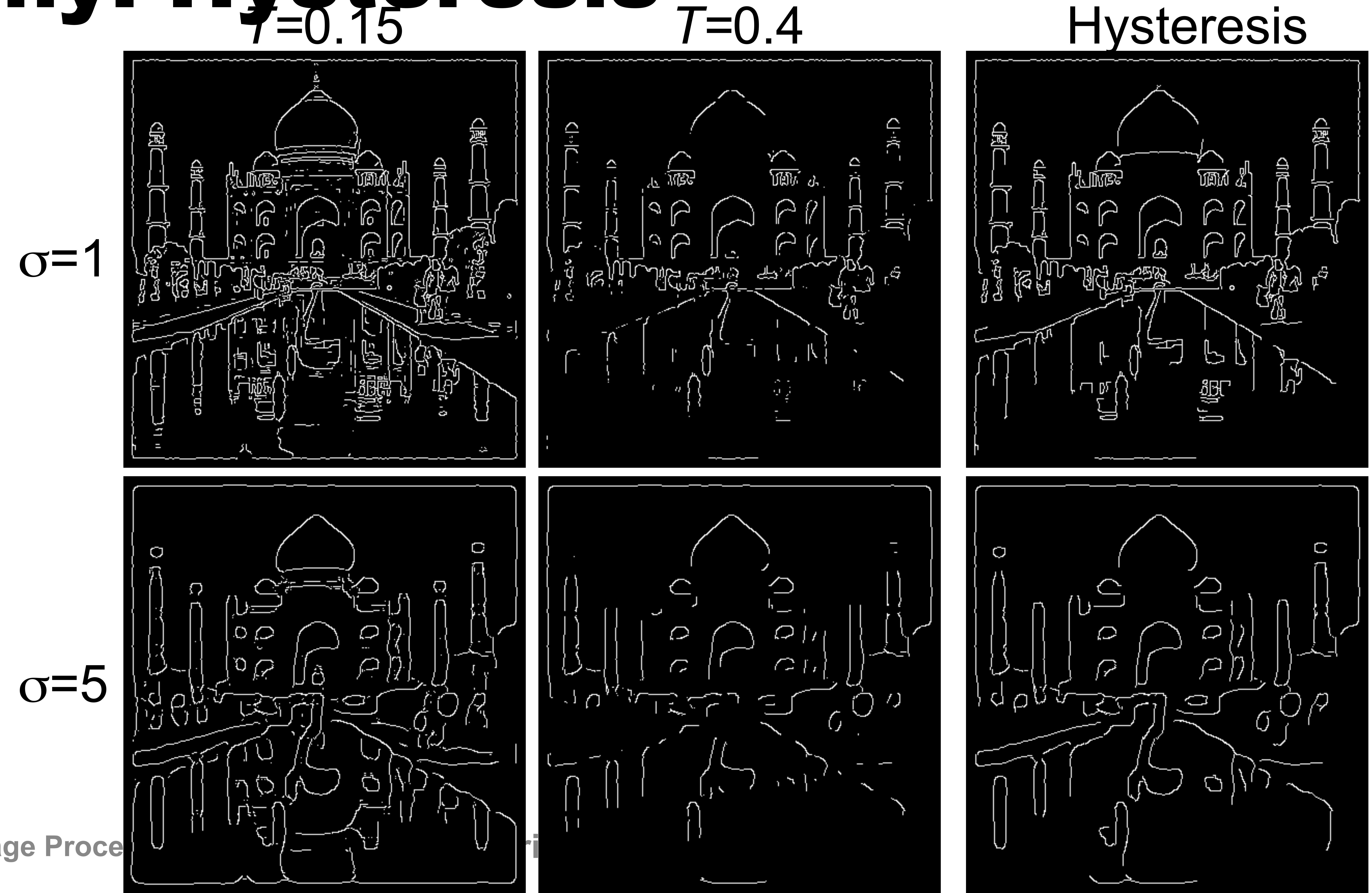


σ=1 σ=5

Canny: Hysteresis Threshold

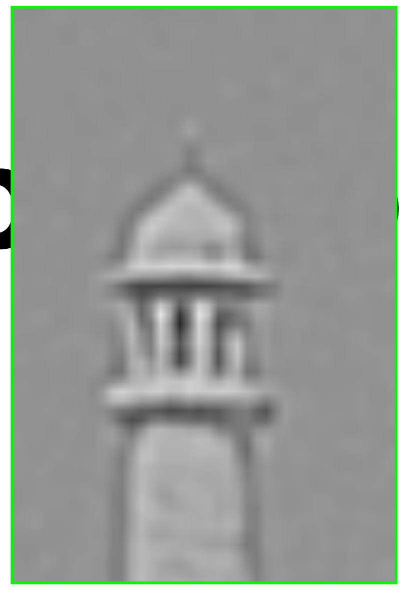
- **Threshold with high and low thresholds.**
- **Initialize edge map from high threshold.**
- **Iteratively add pixels in low threshold map with 8-neighbors already in the edge map.**
- **Repeat until convergence.**

Canny: Hysteresis



Marr-Hildreth Edge Detector

- **Convolve with second derivative op r**
  - **Laplacian of Gaussian**



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

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

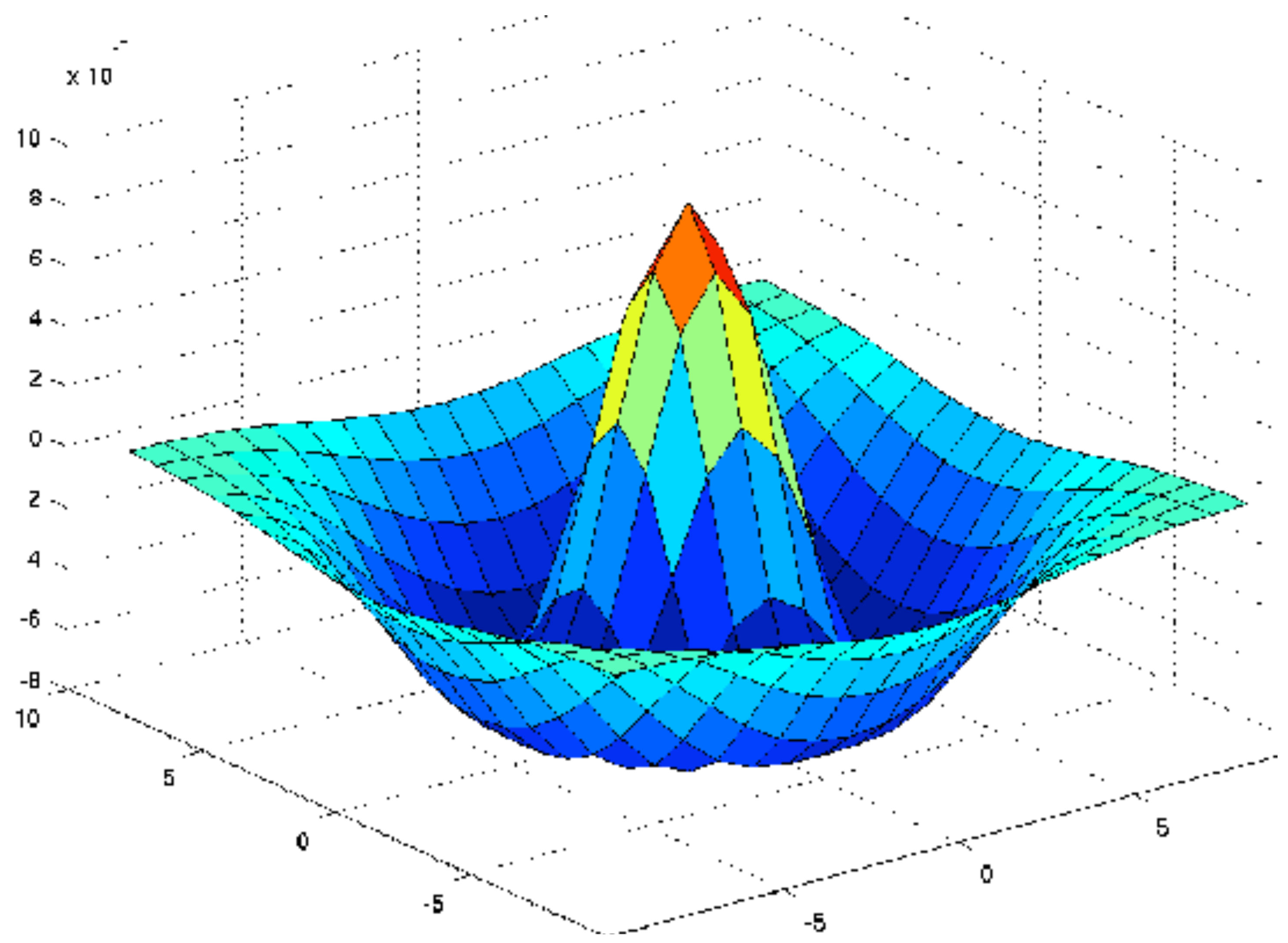
$$k(x,y) = \nabla^2 G(x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) G(x,y)$$

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

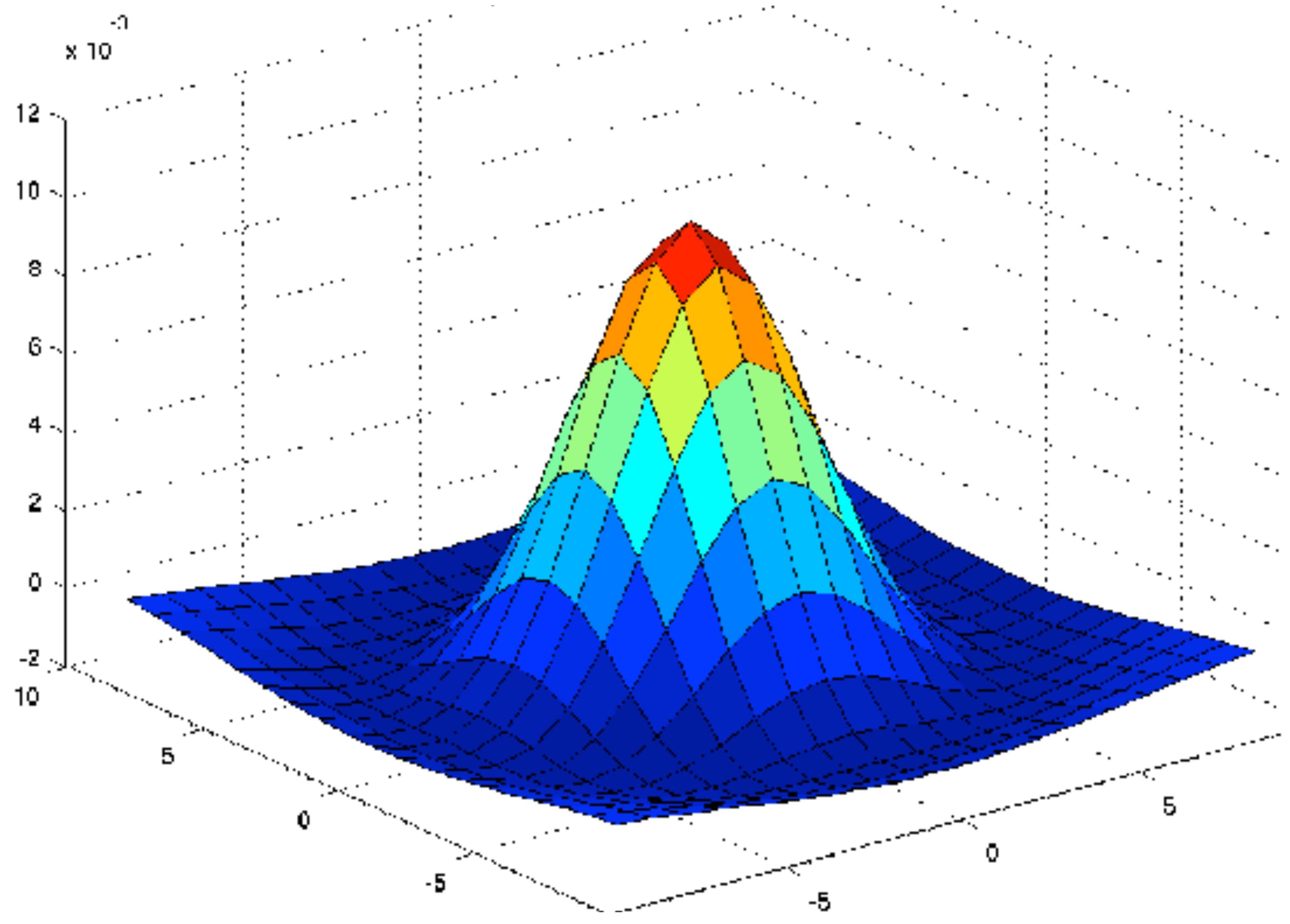
- **Find zero crossings**
- **Sensitive to noise.**

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

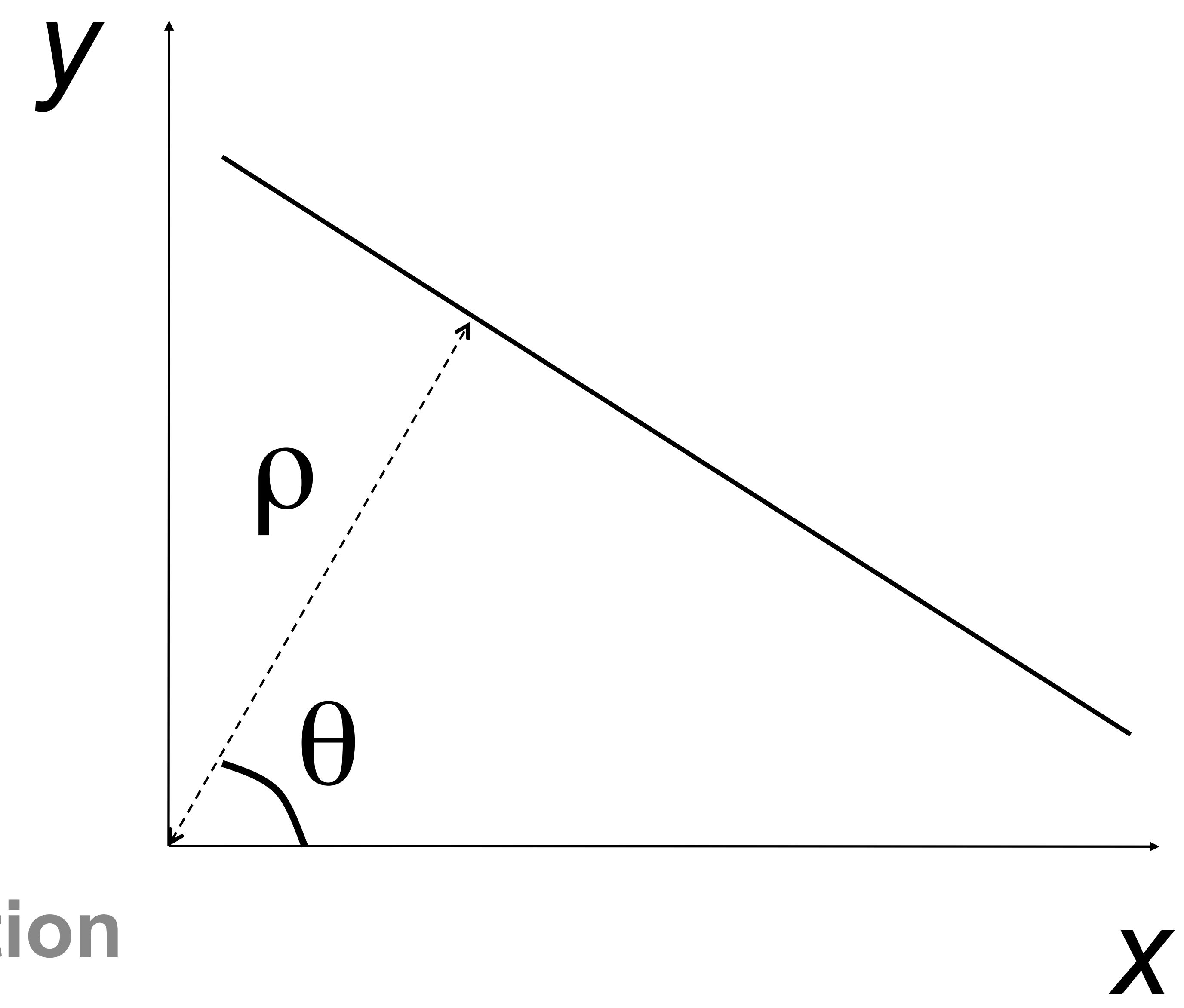
- Locate edges by fitting a surface.
- Create a model edge, e.g., step edge with parameters:
  - Orientation
  - Position
  - Intensities either side of the edge
- Find least-squares fit in each small window.
- Accept if fit is above a threshold.

Hough Transform

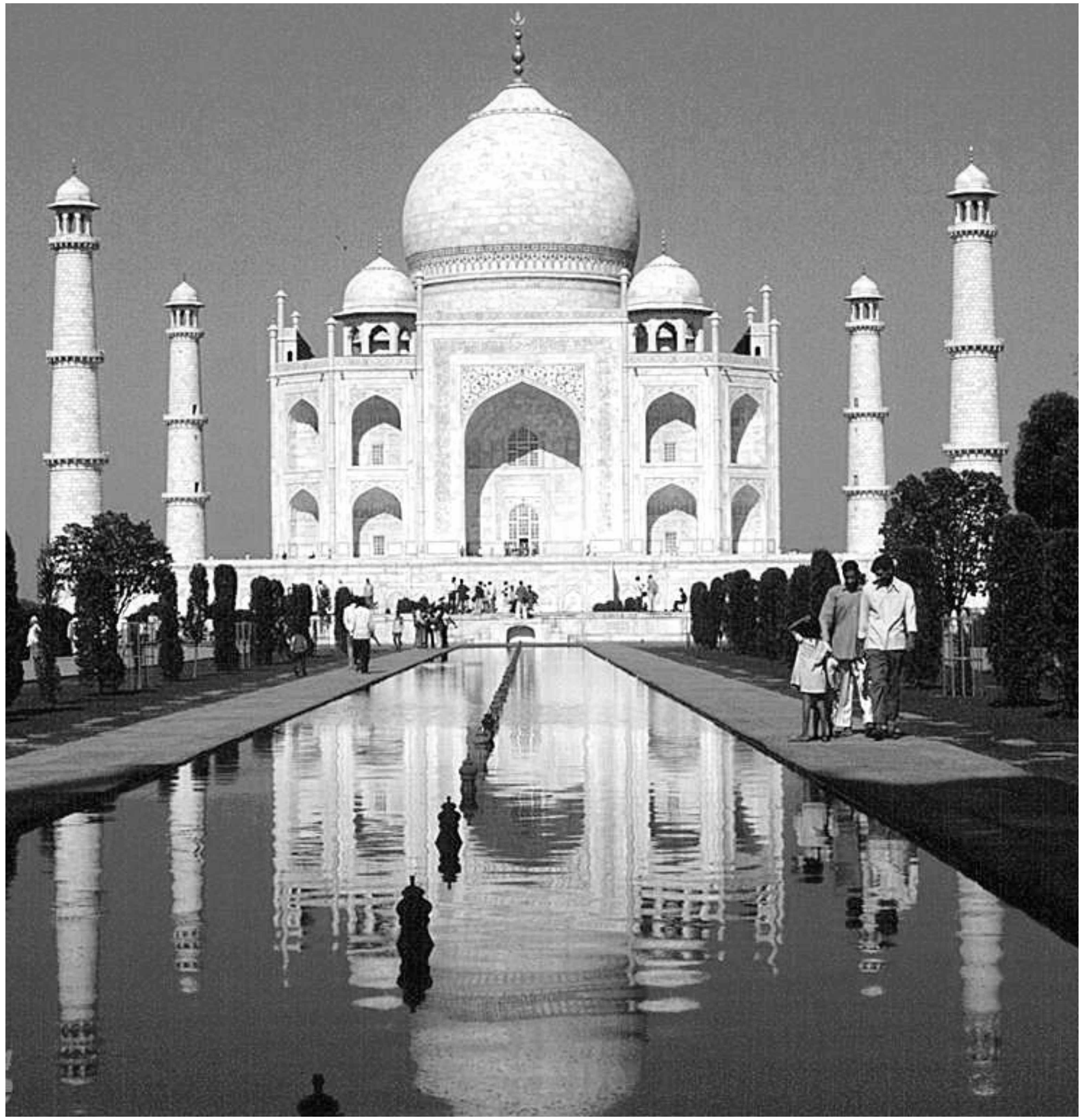
- **Finds the most likely lines in an image.**
- **At every edge pixel, compute the local equation of the edge line:**

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

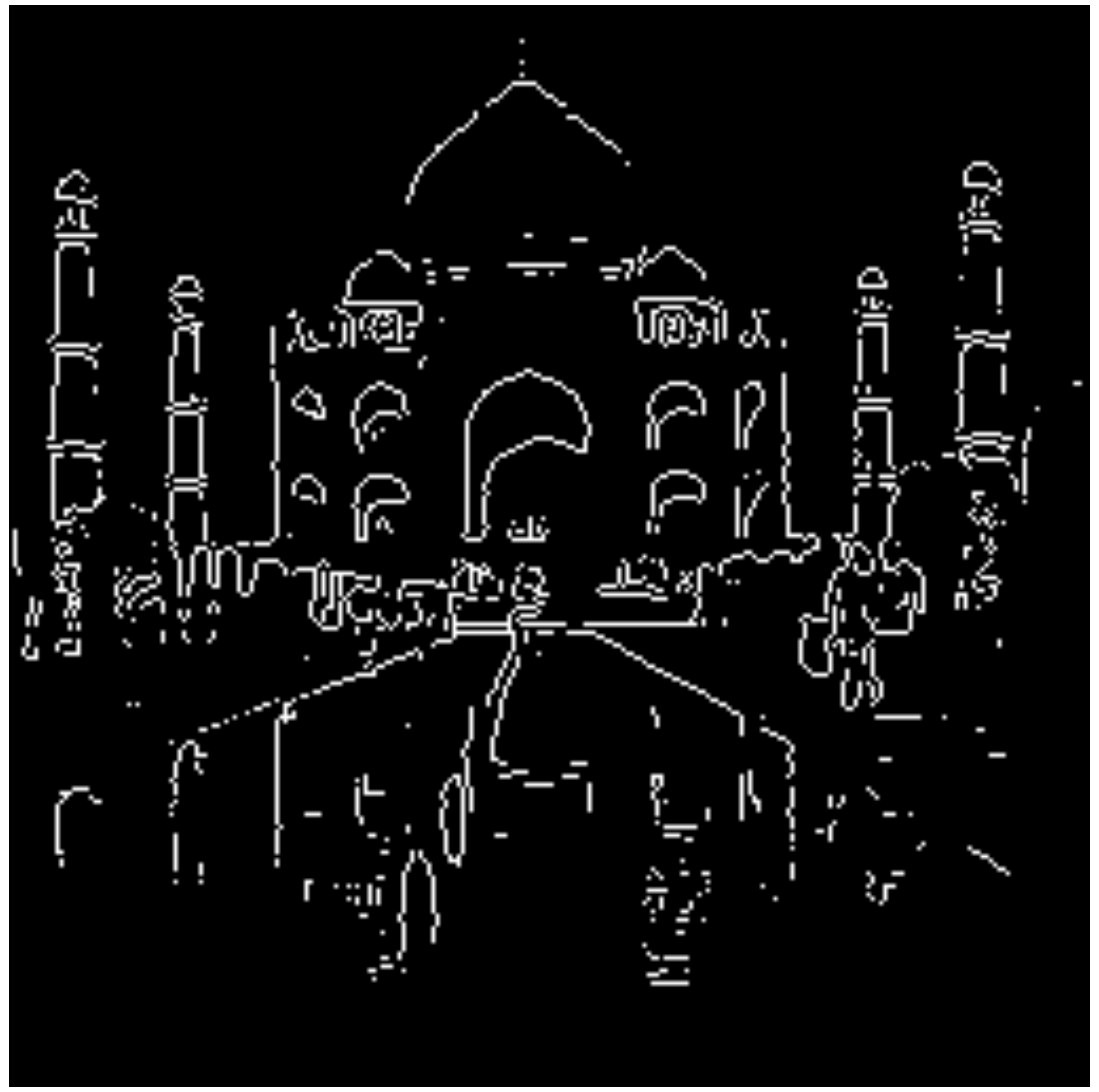
$$x \cos \theta + y \sin \theta = \rho$$



Example

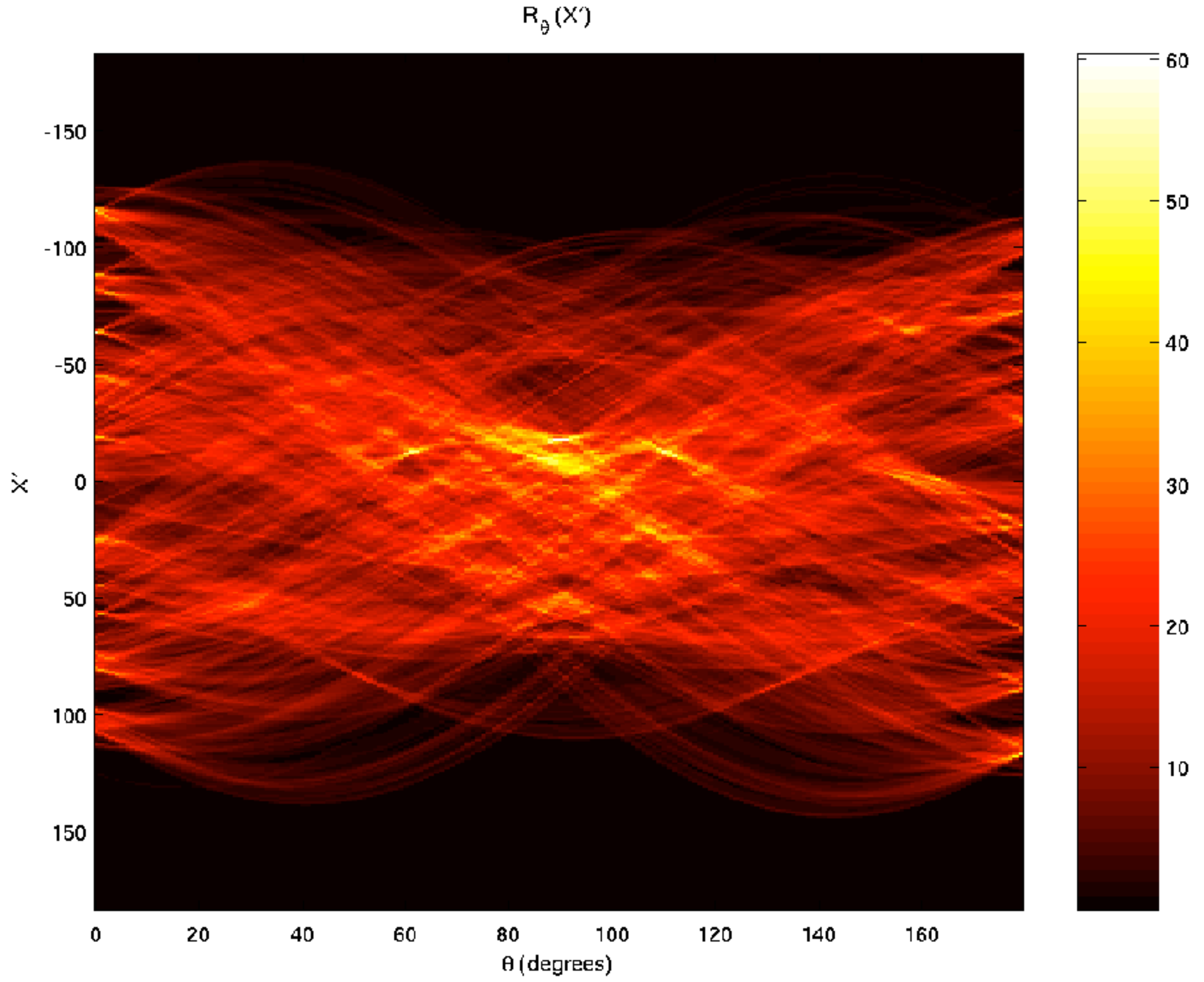


`ed=edge(im);`



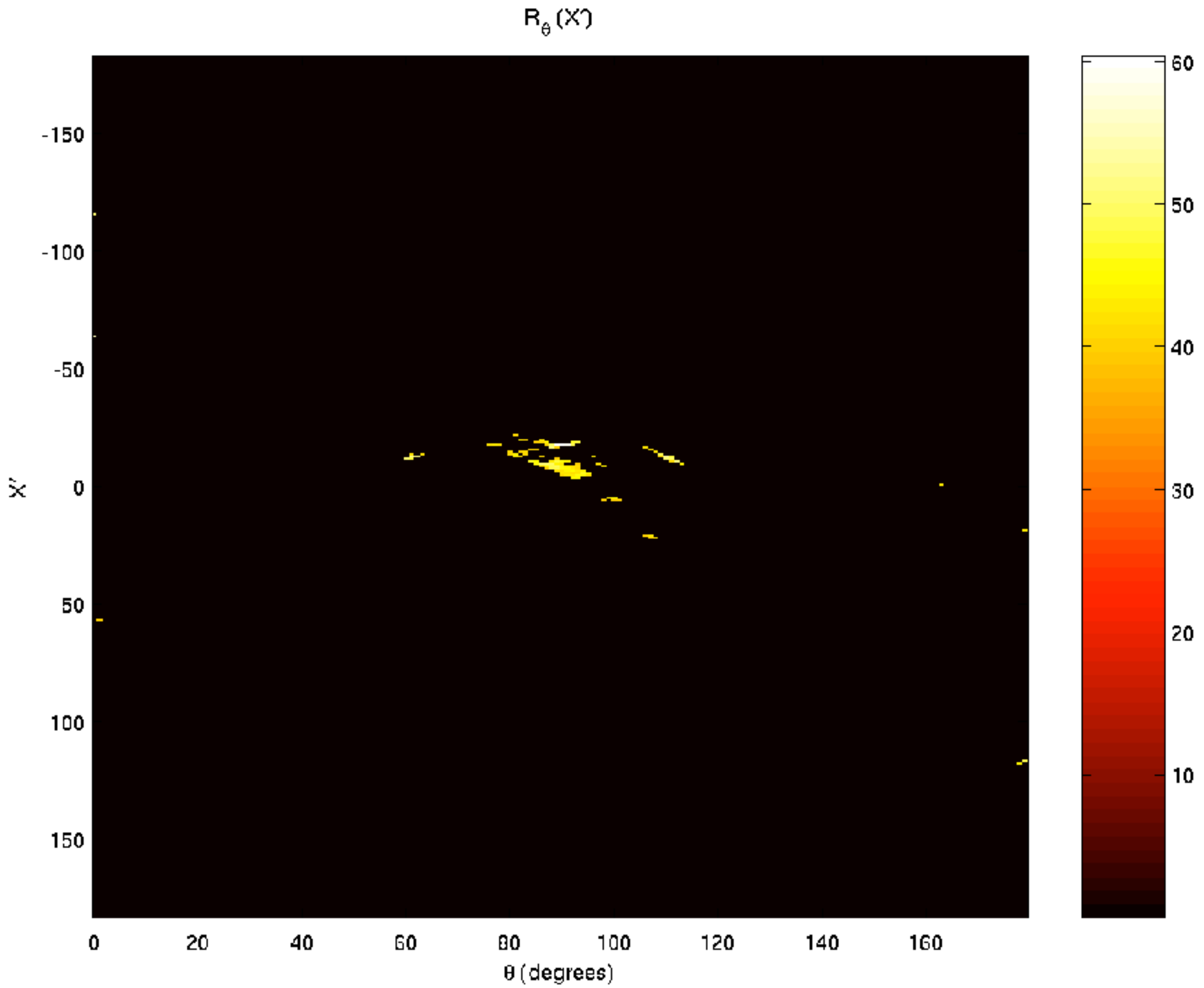
Hough Transform

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



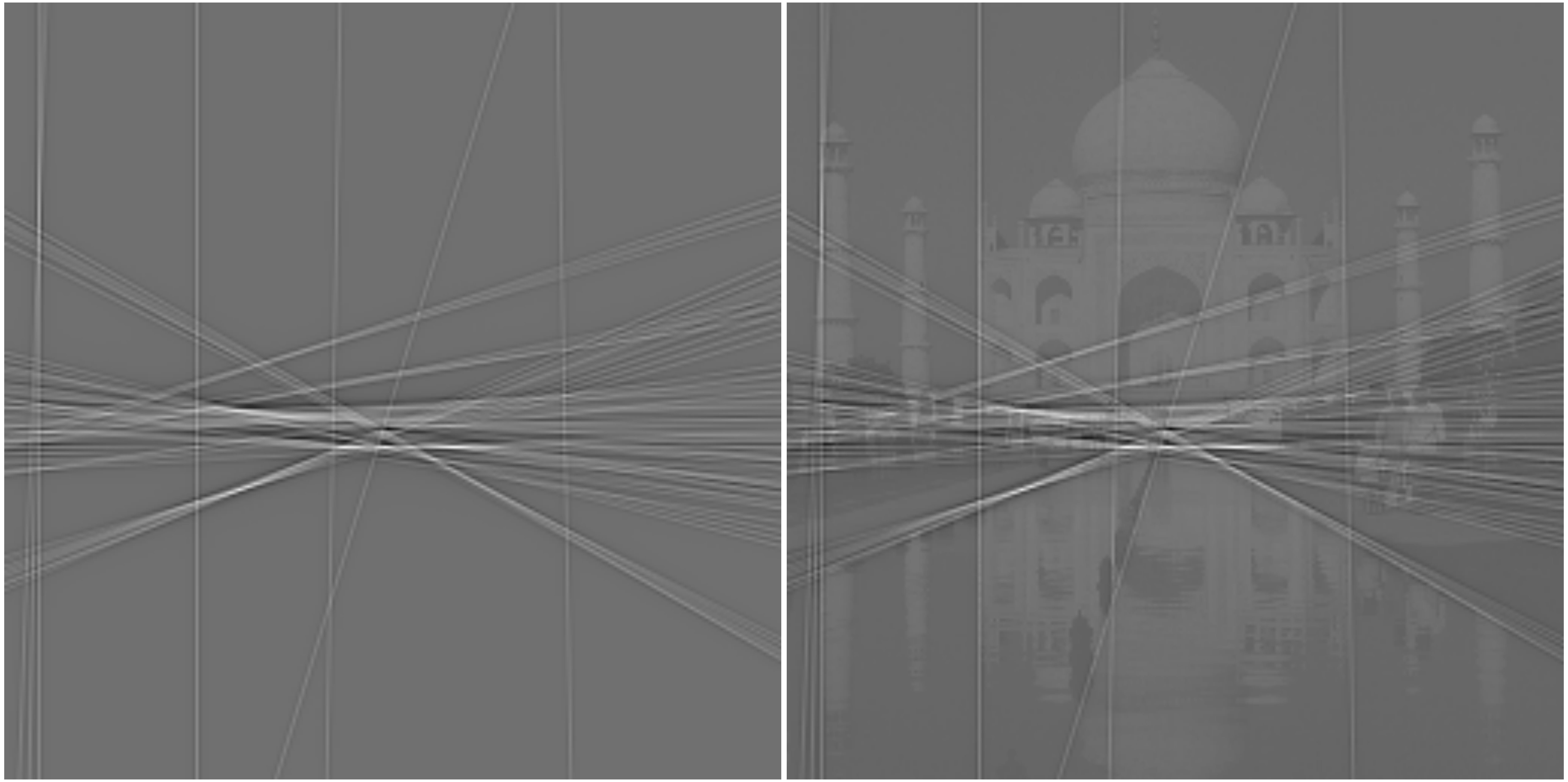
Thresholded

```
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