קורס גרפיקה ממוחשבת

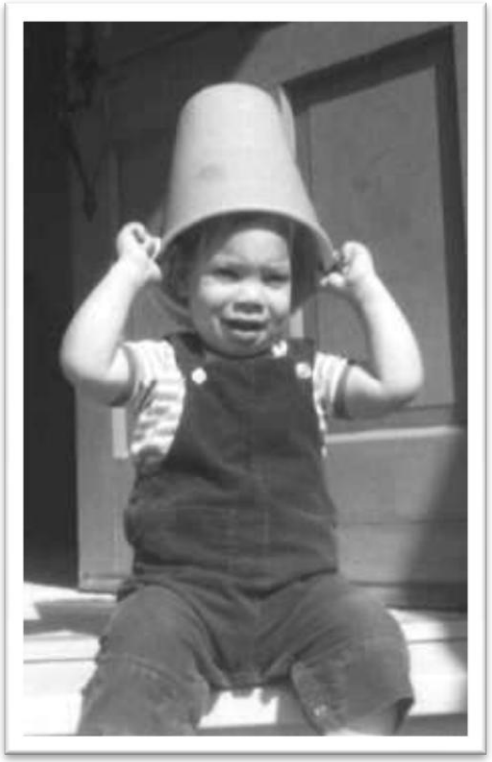
2008 סמסטר ב'

Image Processing

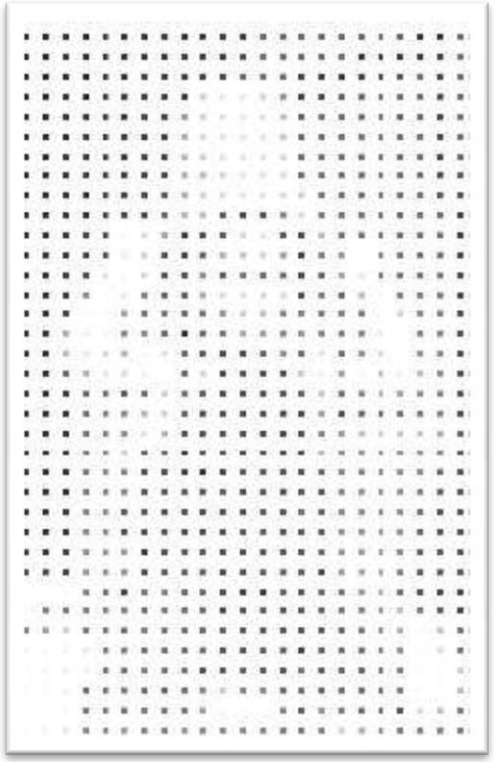
חלק מהשקפים מעובדים משקפים של פרדו דוראנד, טומס פנקהאוסר ודניאל כהן-אור

What is an image?

- An image is a discrete array of samples representing a continuous 2D function



Continuous function

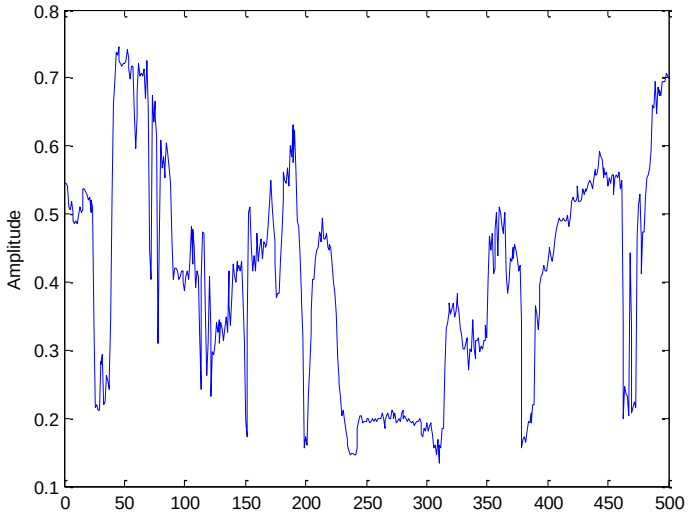


Discrete samples

Converting to digital form

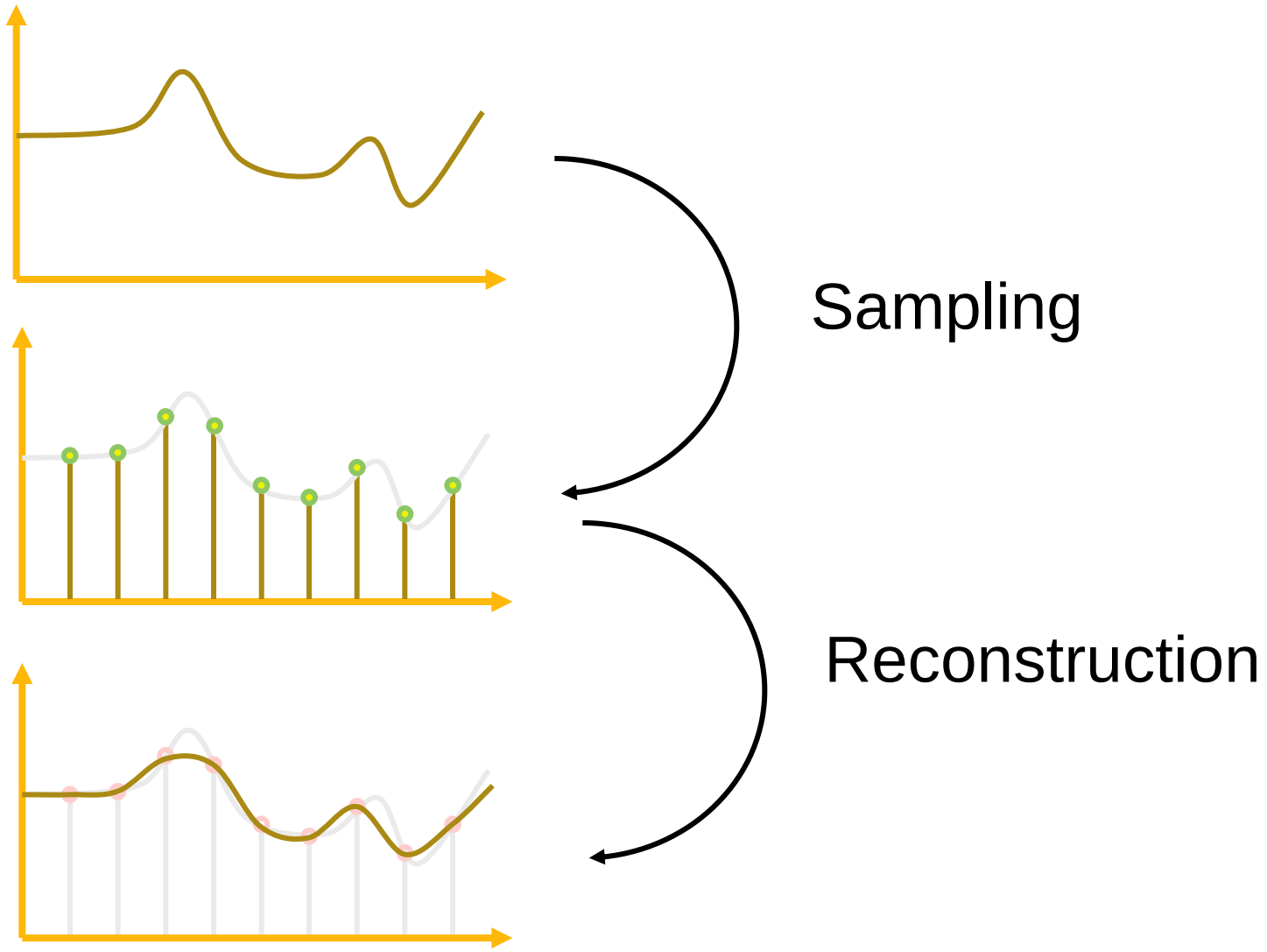
- Convert continuous sensed data into digital form

Quantization



Sampling

Sampling and Reconstruction



Sampling and Reconstruction

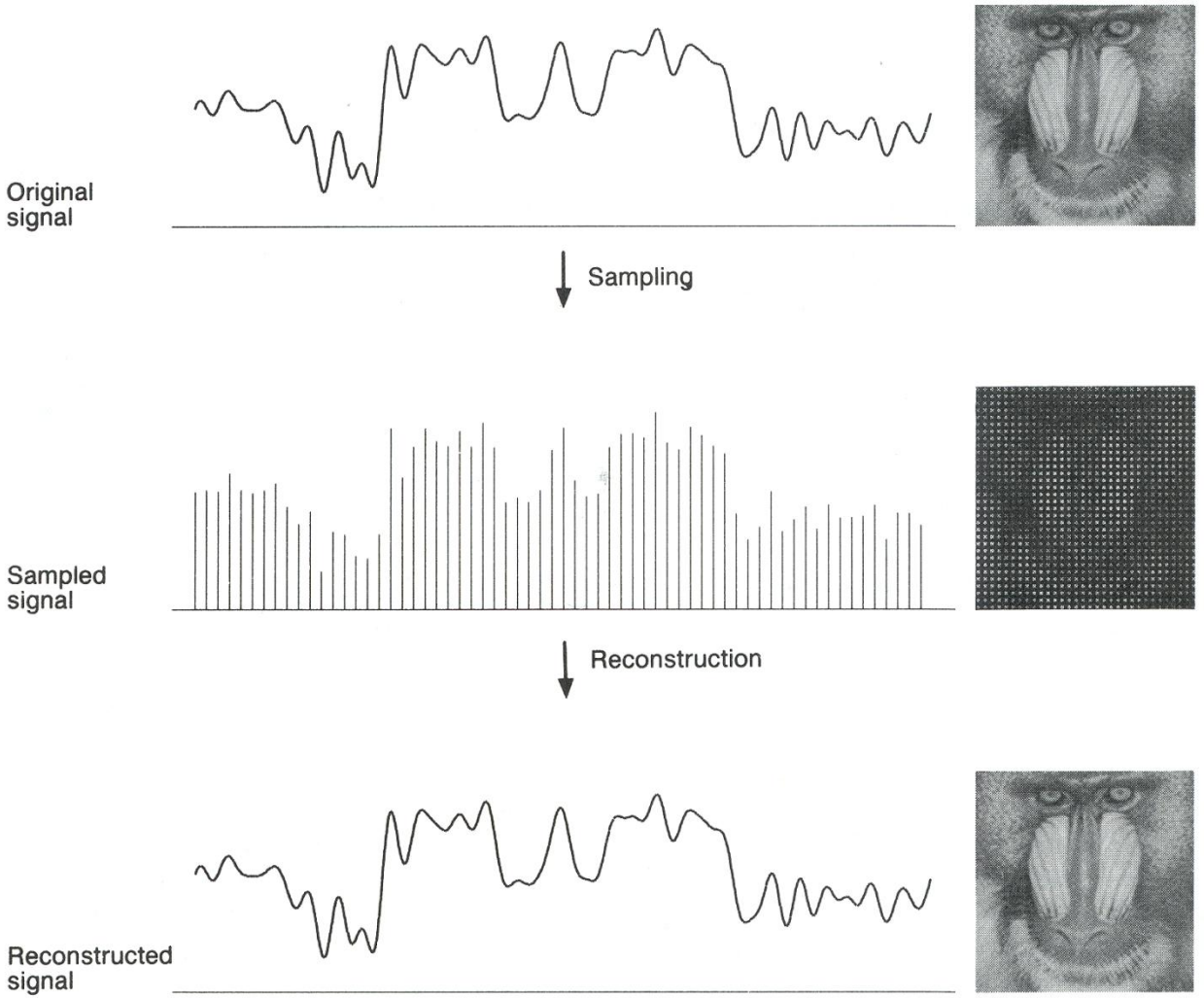
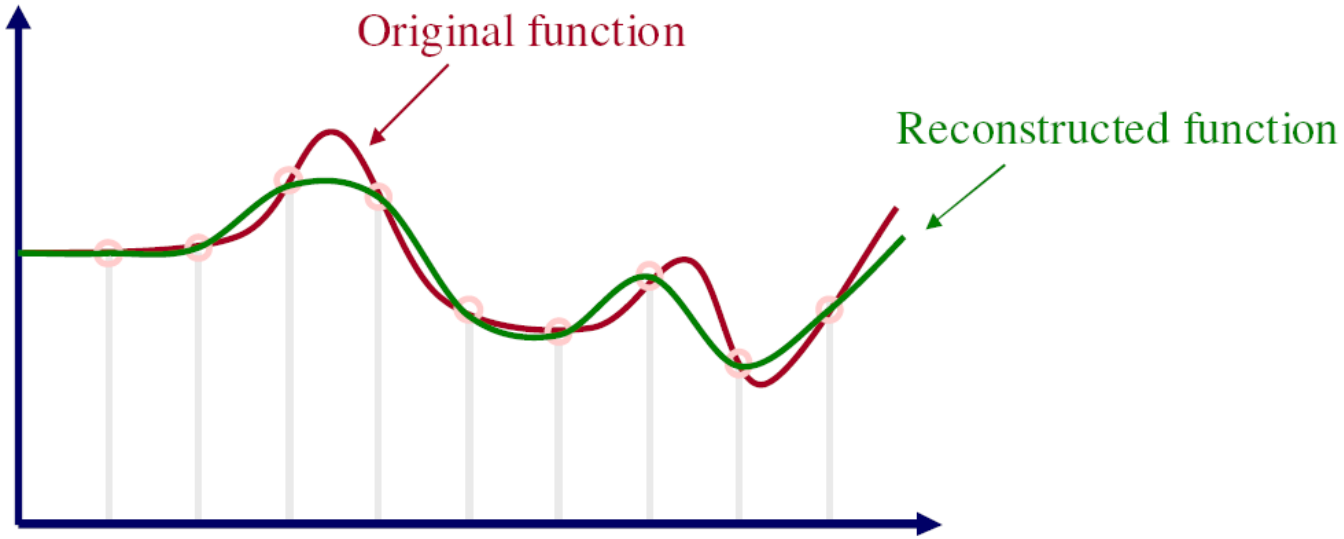


Figure 19.9 FvDFH

Sampling Theory

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?



Aliasing

- What happens when we use too few samples?
  - Aliasing

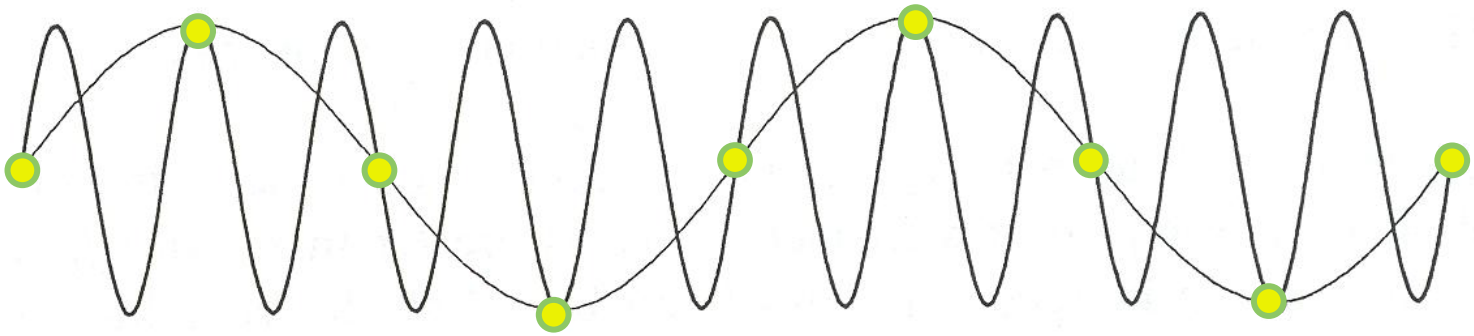


Figure 14.17 FvDFH

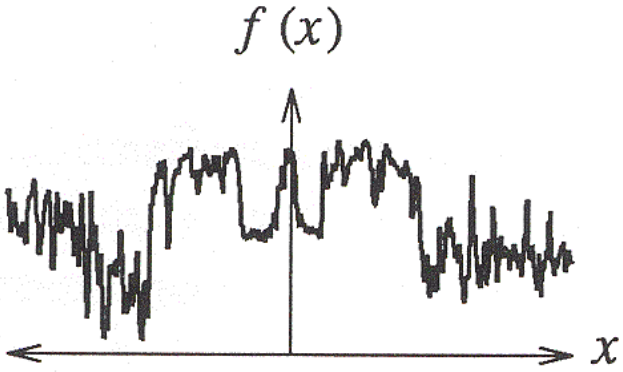
Spectral Analysis

- So our image (function f(x,y)) describes how the signal changes over "time" (x and y axes)
- Aliasing occurs when we use too few samples (what is enough?)

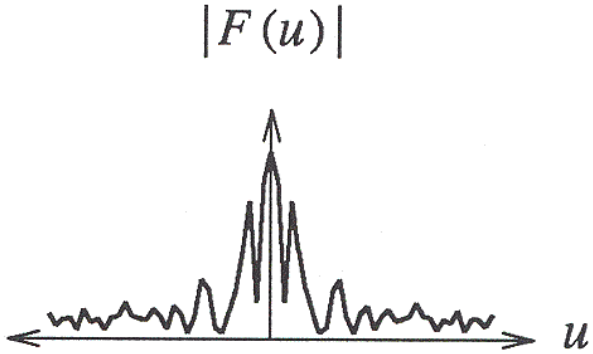
- The more an image changes, the more we need to sample it.
- How do we measure how fast a signal changes?
  - Frequencies

Spectral Analysis

- Spatial domain:
  - Function: f(x)
  - Filtering: convolution



- Frequency domain:
  - Function: F(u)
  - Filtering: multiplication

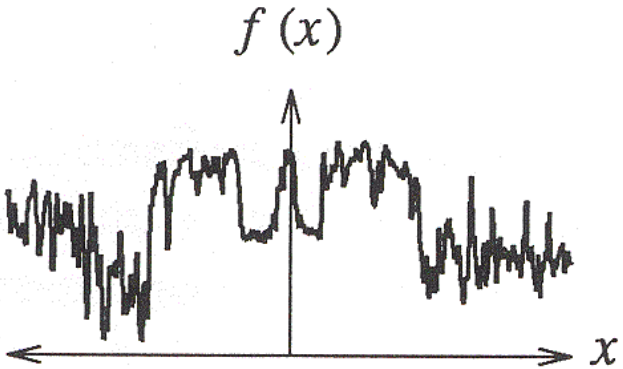


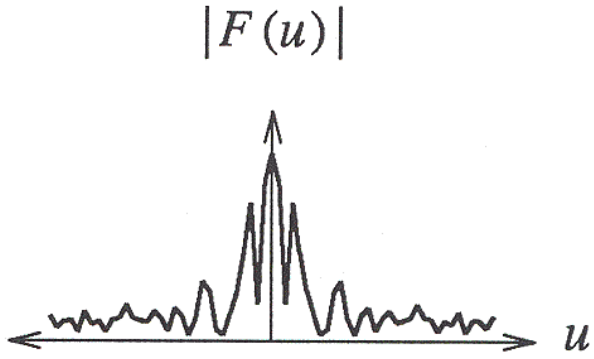
Any signal can be written as a sum of periodic functions.

Fourier

- Joseph Fourier discovered in 1822 that
  - Any periodic function can be expressed as the sum of sines and/or cosines if different frequencies (*Fourier Series*)
  - Even functions that are not periodic can be expressed as the integral of sines and/or cosines (*Fourier Transform*)
  - Initial application was in heat diffusion

Fourier Transform (1D)





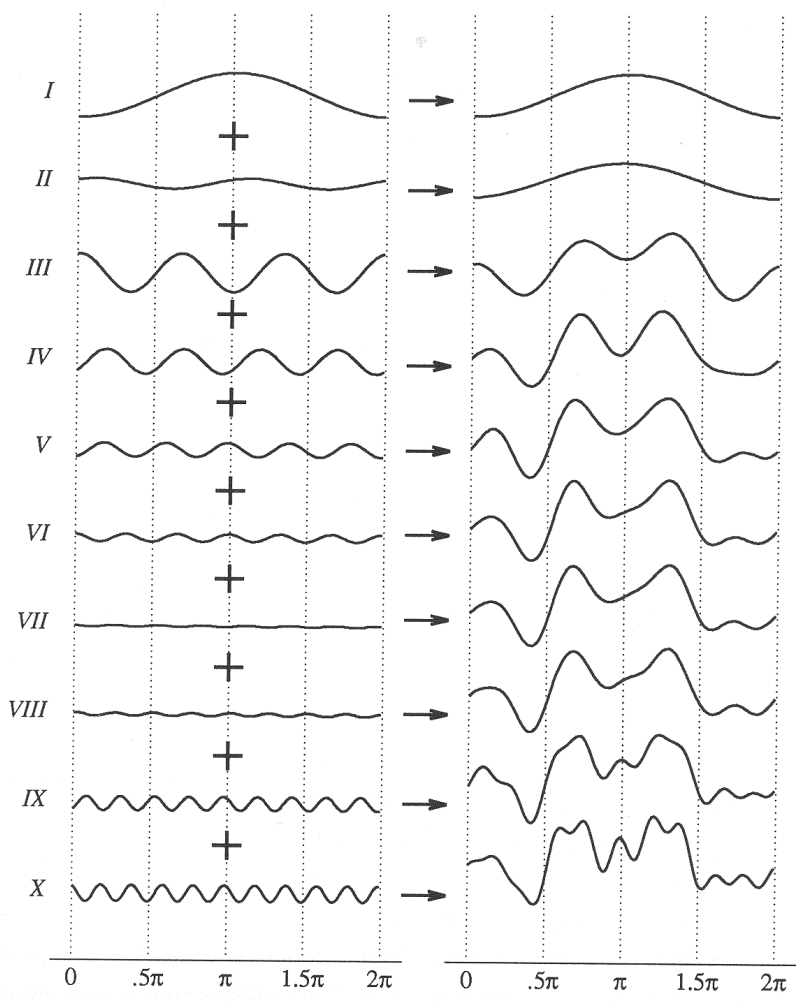


Figure 2.6 Wolberg

Fourier Transform (1D)

- Fourier transform:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i 2\pi x u} dx$$

- Inverse Fourier transform:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{+i 2\pi u x} du$$

Sampling Theorem

- A signal can be reconstructed from its samples, if the original signal has no frequencies above 1/2 the sampling frequency - Shannon
- The minimum sampling rate for bandlimited function is called "Nyquist rate"

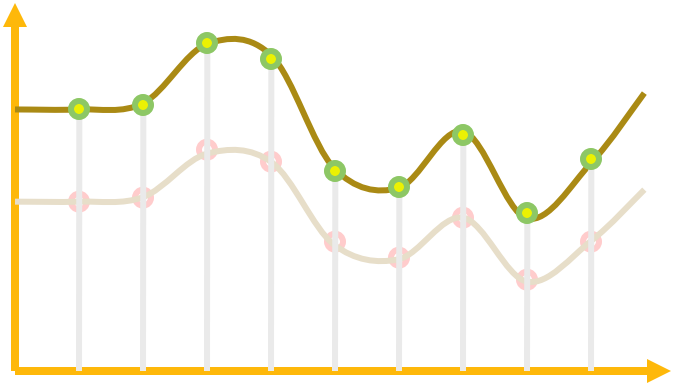
A signal is bandlimited if its highest frequency is bounded. The frequency is called the bandwidth.

Image Processing

- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Quantization
  - Uniform Quantization
  - Floyd-Steinberg dither
- Warping
  - Scale
  - Rotate
  - Warps
- Combining
  - Composite
  - Morph

Adjusting Brightness

- Simply scale pixel components
  - Must clamp to range (e.g., 0 to 1)



Original

Brighter

Adjusting Contrast

- Compute mean luminance L for all pixels
  - luminance = 0.30*r + 0.59*g + 0.11*b
- Scale deviation from L for each pixel component
  - Must clamp to range (e.g., 0 to 1)

Original

More Contrast

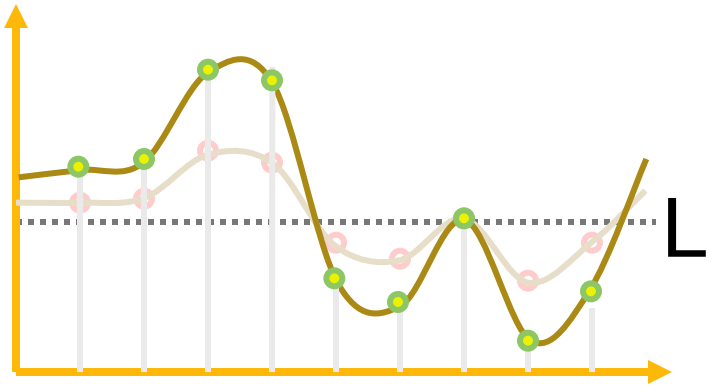
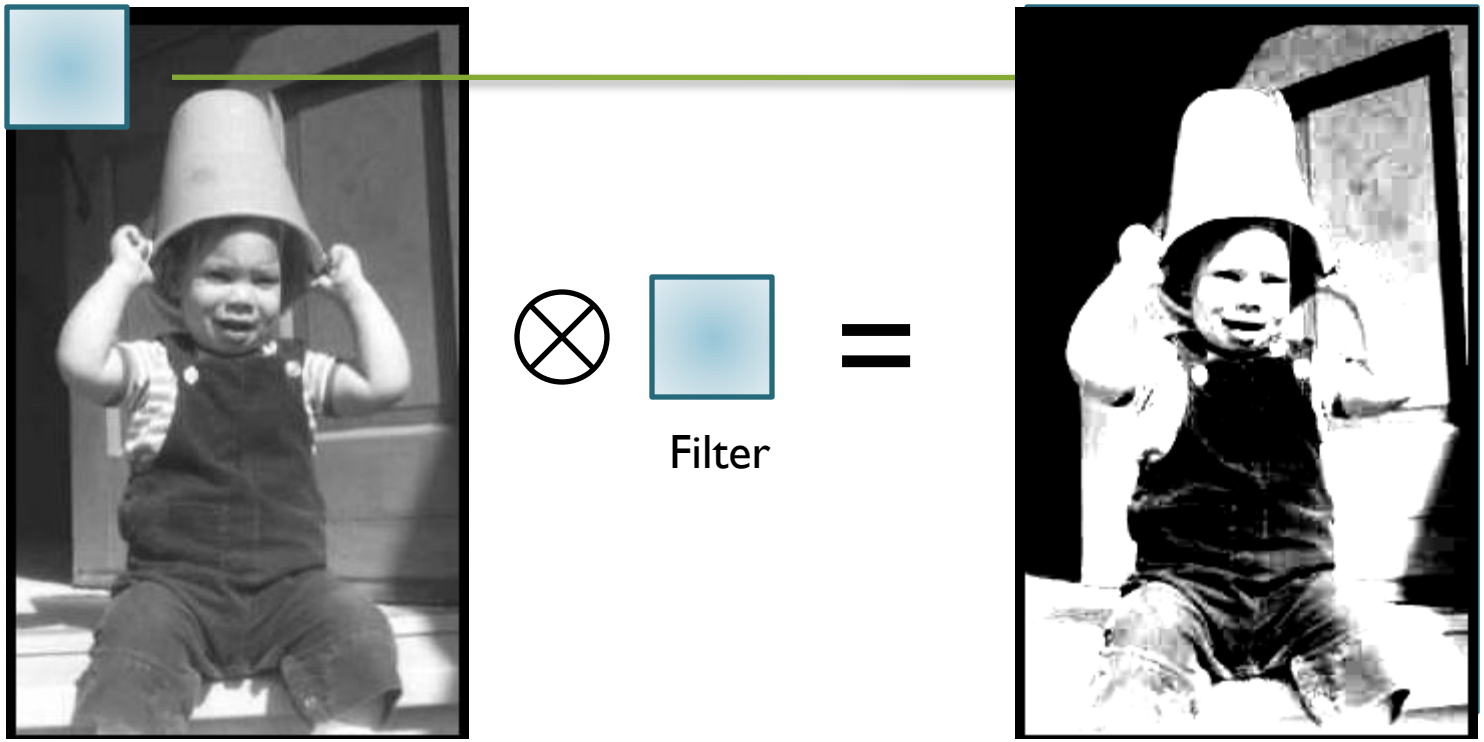


Image Processing

- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Quantization
  - Uniform Quantization
  - Floyd-Steinberg dither
- Warping
  - Scale
  - Rotate
  - Warps
- Combining
  - Composite
  - Morph

Linear Filtering (Spatial Domain)

- Convolution
  - Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by a filter



Adjust Blurriness

- Convolve with a filter whose entries sum to one
  - Each pixel becomes a weighted average of its neighbors

Original

Blur

$$\text{Filter} = \begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$

What do you think happens in the frequency domain?

More on blur (lowpass filters)

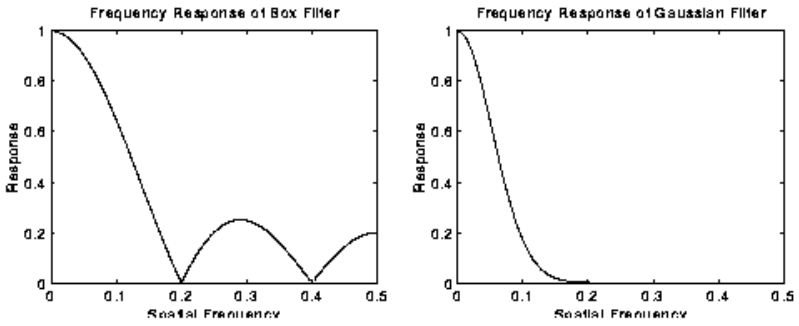
- We can either take a uniform kernel (mean filter)

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

- Or a Gaussian kernel

$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$

- A Gaussian kernel tends to provide gentler smoothing and preserve edges better



Edge Detection

- Convolve with a filter that finds differences between neighbor pixels

Original

Detect edges

$$\text{Filter} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen

- Sum detected edges with original image

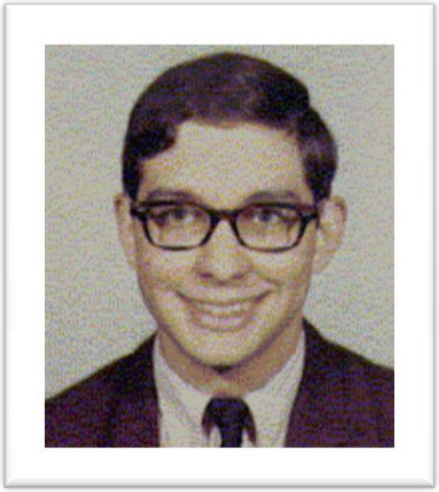
Original

Sharpened

$$\text{Filter} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Emboss

- Convolve with a filter that highlights gradients in particular directions



Original

Embossed

$$\text{Filter} = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Non-linear filtering

- Any operation on a neighborhood around each pixel
- For example: Selecting the **median** value of the neighborhood

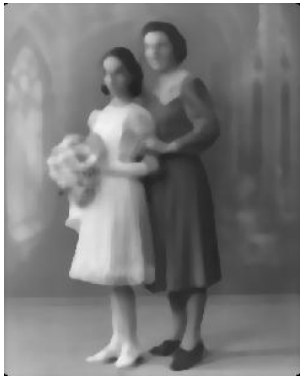
Original

3x3

5x5

7x7

11x11



15x15

Image Processing

- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Quantization
  - Uniform Quantization
  - Floyd-Steinberg dither
- Warping
  - Scale
  - Rotate
  - Warps
- Combining
  - Composite
  - Morph

Quantization

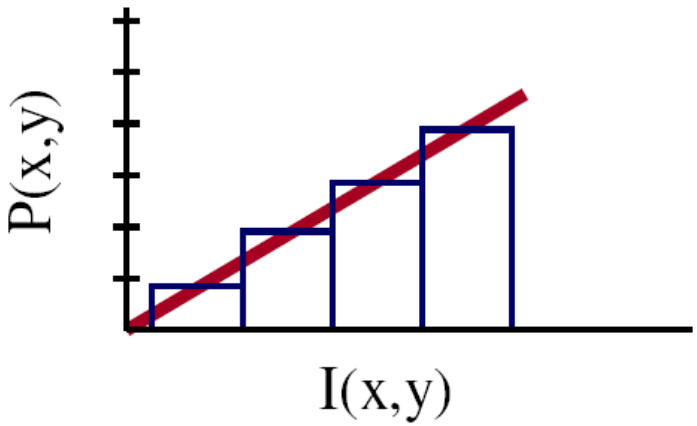
- Reduce intensity resolution
  - Frame buffers have limited number of bits per pixel
  - Physical devices have limited dynamic range

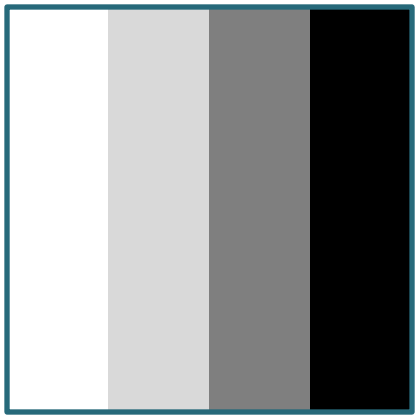
n=0.5

Uniform Quantization

- P(x,y) = round(I(x,y))

I(x,y)



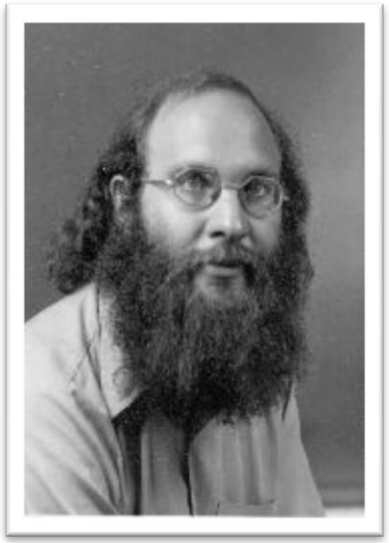


P(x,y) – 2 bits per pixel

Uniform Quantization

- Images with decreasing bits per pixel:

8 bits



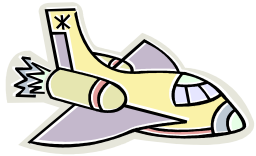
4 bits

2 bits

1 bit

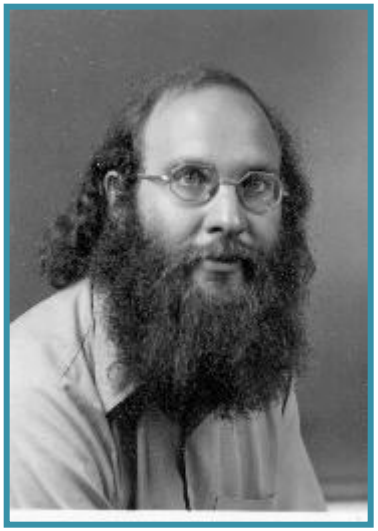
Reducing effects of Quantization

- Dithering
  - Random dither
  - Ordered dither
  - Error diffusion dither
- Halftoning
  - Classical halftoning



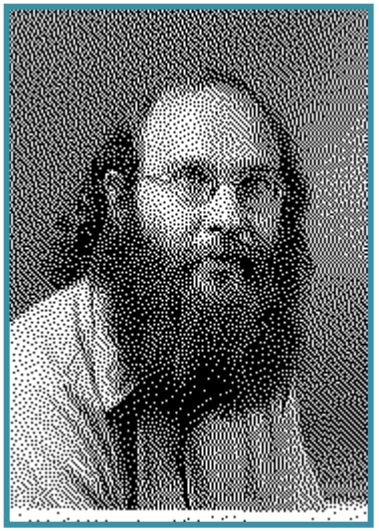
Dithering

- Distribute errors among pixels
  - Exploit spatial integration in our eye
  - Display greater range of perceptible intensities



Original
(8 bits)

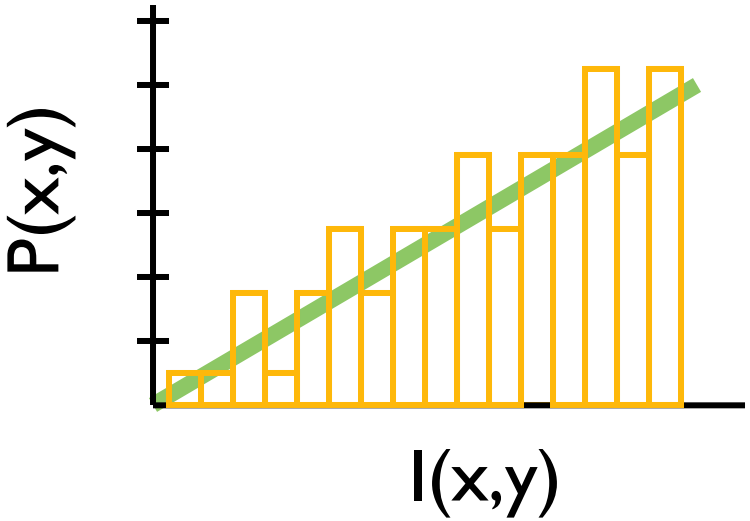
Uniform
Quantization
(1 bit)

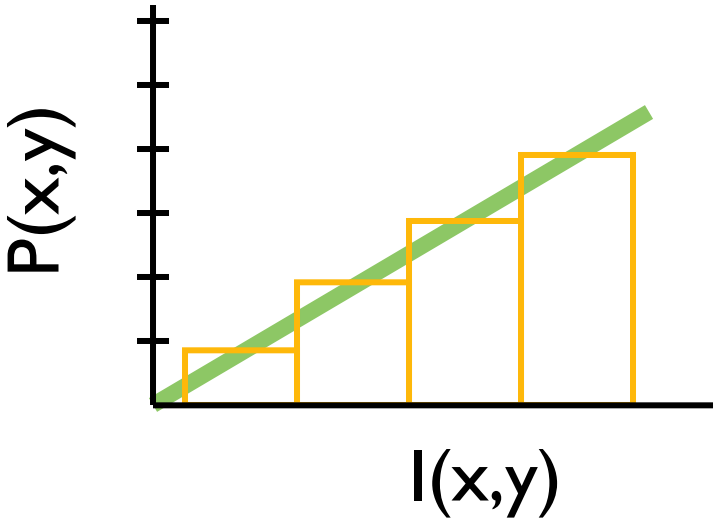


Floyd-Steinberg
Dither
(1 bit)

Random Dither

- Randomize quantization errors
  - Errors appear as noise

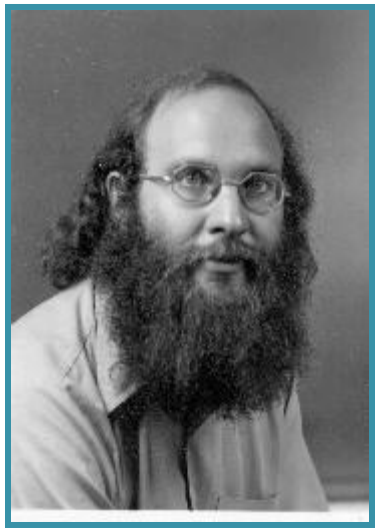




$$P(x, y) = \text{trunc}(I(x, y) + \text{noise}(x,y) + 0.5)$$

1 bit

Random Dither



Original
(8 bits)

Uniform
Quantization
(1 bit)

Random
Dither
(1 bit)

Ordered Dither

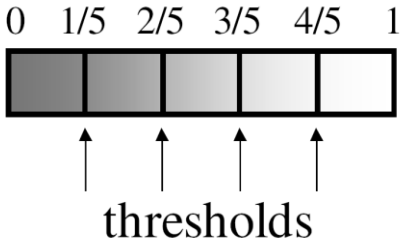
- Pseudo-random quantization errors
  - Matrix stores pattern of threshholds

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

For each pixel (x,y)

oldpixel = I(x,y) +D(x mod n,y mod n)

P(x,y)= find_closest_color(oldpixel)



Ordered Dither

- Bayer's ordered dither matrices

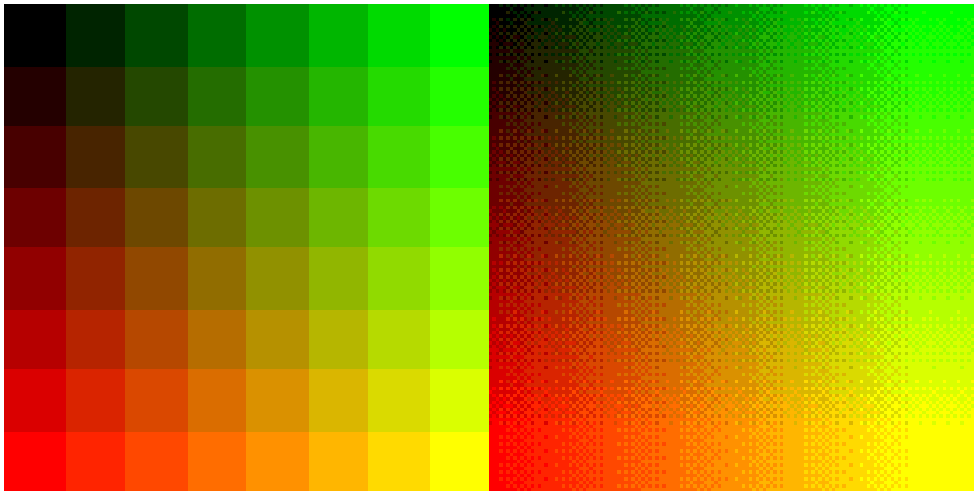
$$D_n = \begin{bmatrix} 4D_{n/2} + D_2(1,1)U_{n/2} & 4D_{n/2} + D_2(1,2)U_{n/2} \\ 4D_{n/2} + D_2(2,1)U_{n/2} & 4D_{n/2} + D_2(2,2)U_{n/2} \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \qquad D_4 = \begin{bmatrix} 15 & 7 & 13 & 5 \\ 3 & 11 & 1 & 9 \\ 12 & 4 & 14 & 6 \\ 0 & 8 & 2 & 10 \end{bmatrix}$$

Basic idea: organize successive integers such that the average distance between two successive numbers in the map is as large as possible

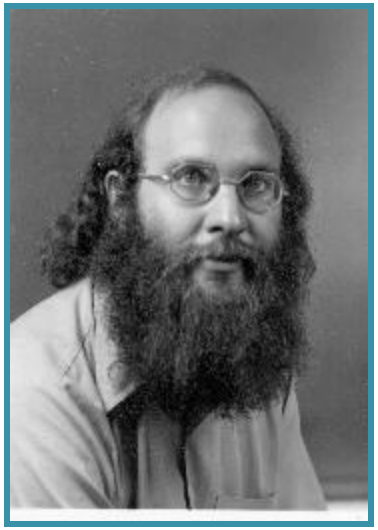
Ordered Dither

- An example
  - Palette consists of 8 red tones, 8 green tones and their combinations (64 colors)
  - Original image had 19600 colors



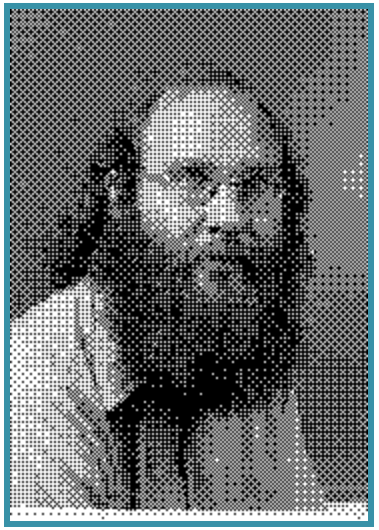
Undithered Dithered

Ordered Dither



Original
(8 bits)

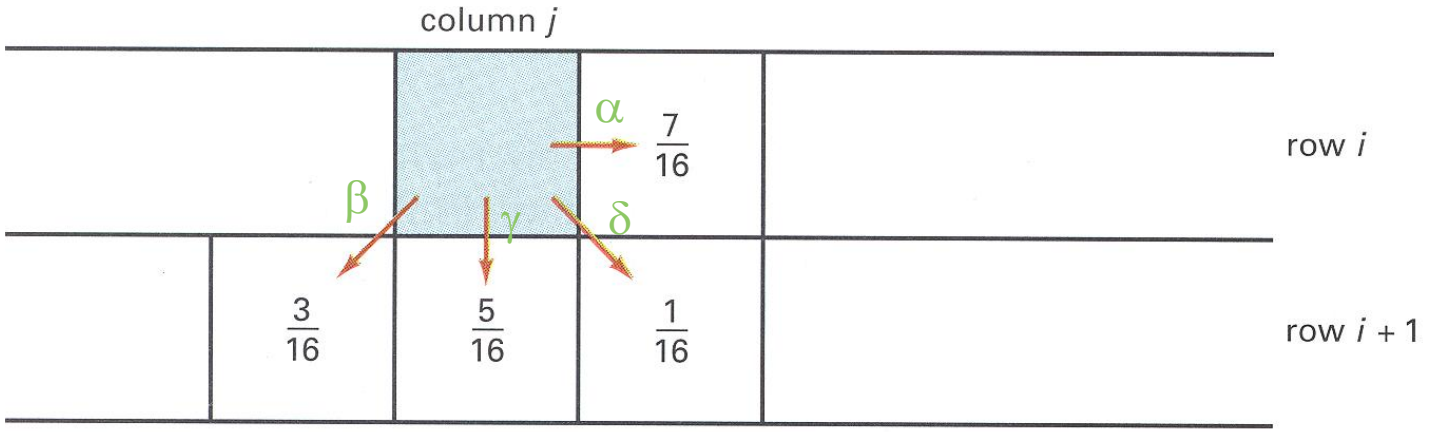
Random
Dither
(1 bit)



Ordered
Dither
(1 bit)

Error Diffusion Dither

- Spread quantization error over neighbor pixels



$$\alpha + \beta + \gamma + \delta = 1.0$$

Figure 14.42 from H&B

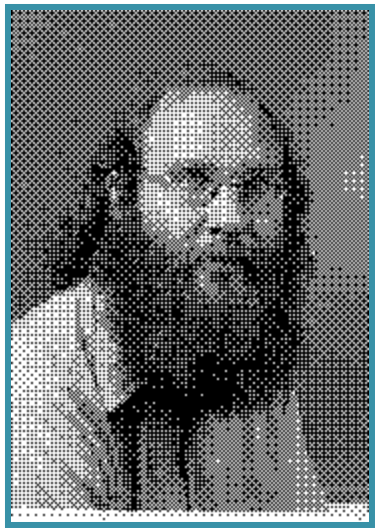
Floyd-Steinberg Algorithm

```
for (x = 0; x < width; x++) {
        for (y = 0; y < height; y++) {
                P(x,y) = trunc(I(x,y) + 0.5)
                e = I(x,y) - P(x,y)
                I(x,y+1) += α*e;
                I(x+1,y-1) += β*e;
                I(x+1,y) += γ*e;
                I(x+1,y+1) += δ *e;
        }
}
```

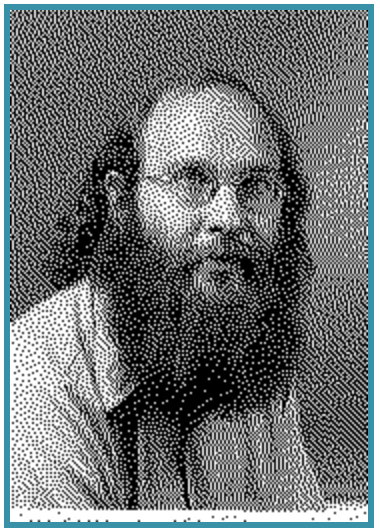
Error Diffusion Dither

Original
(8 bits)

Random
Dither
(1 bit)



Ordered
Dither
(1 bit)



Floyd-Steinberg
Dither
(1 bit)

More examples

Original

Threshold

Random

Bayer

Floyd-Steinberg

Jarvice, Judice & Ninke

Stucki

Burkes

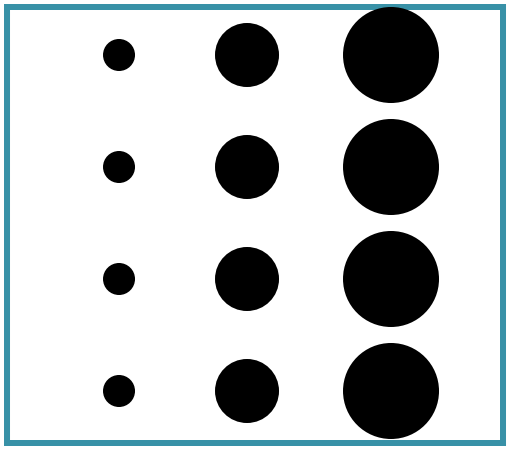
Reducing effects of Quantization

- Dithering
  - Random dither
  - Ordered dither
  - Error diffusion dither
- Halftoning
  - Classical halftoning

Classical Halftoning

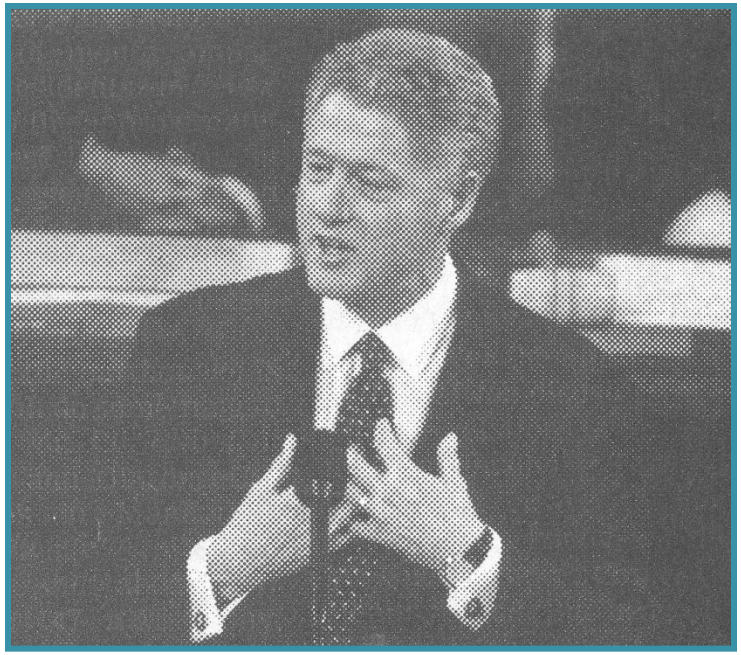
- Use dots of varying size to represent intensities
  - Area of dots proportional to intensity in image

I(x,y)

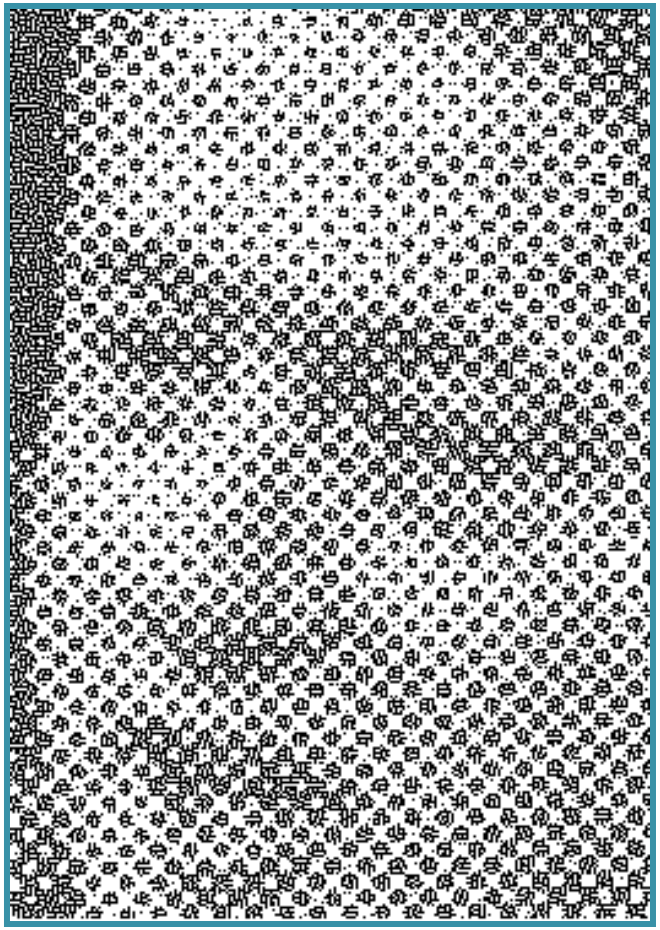


P(x,y)

Classical Halftoning



Newspaper Image



From New York Times, 9/21/99

Halftone patterns

- Use cluster of pixels to represent intensity
  - Trade spatial resolution for intensity resolution

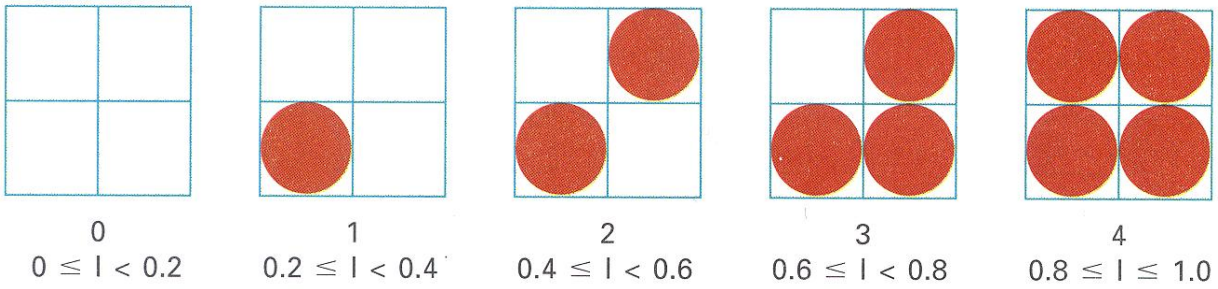


Figure 14.37 from H&B

Halftone patterns

- How many intensities in a n x n cluster?

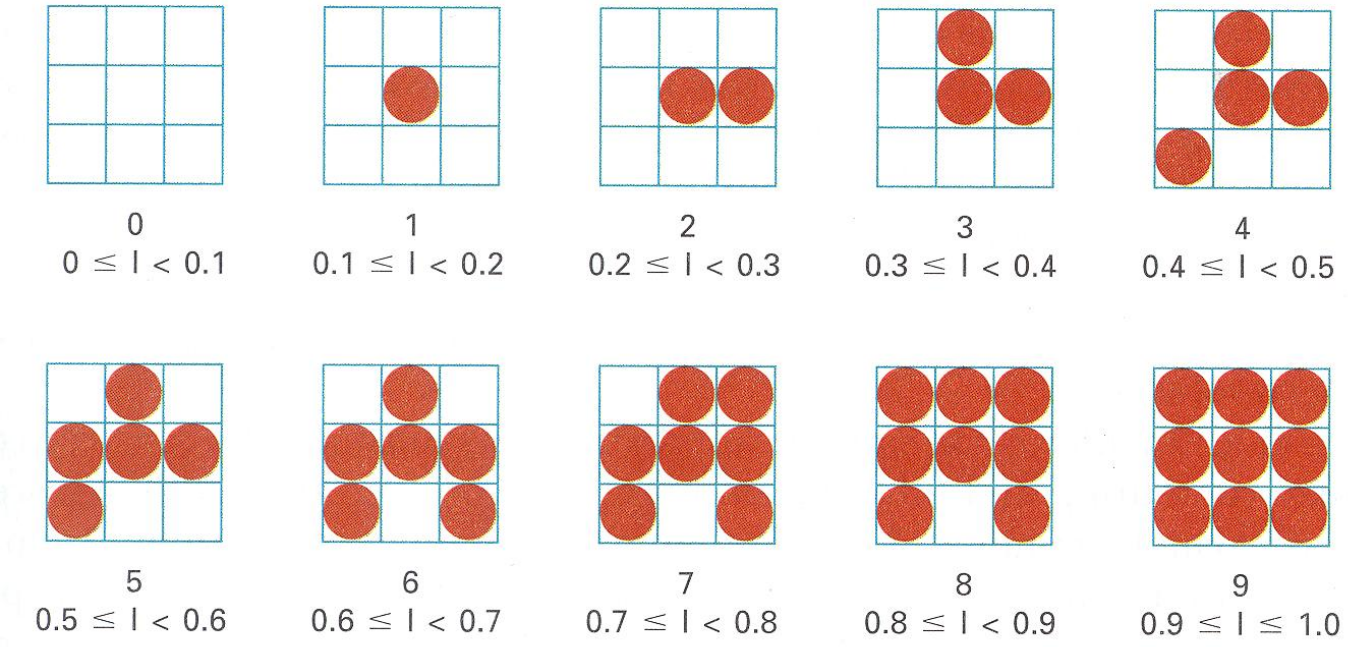


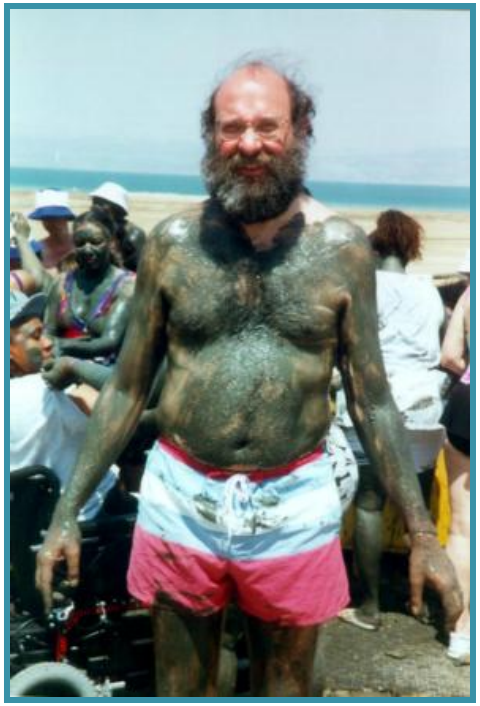
Figure 14.37 from H&B

Image Processing

- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median
- Quantization
  - Uniform Quantization
  - Floyd-Steinberg dither
- Warping
  - Scale
  - Rotate
  - Warps
- Combining
  - Composite
  - Morph

Image Warping

- Move pixels of image



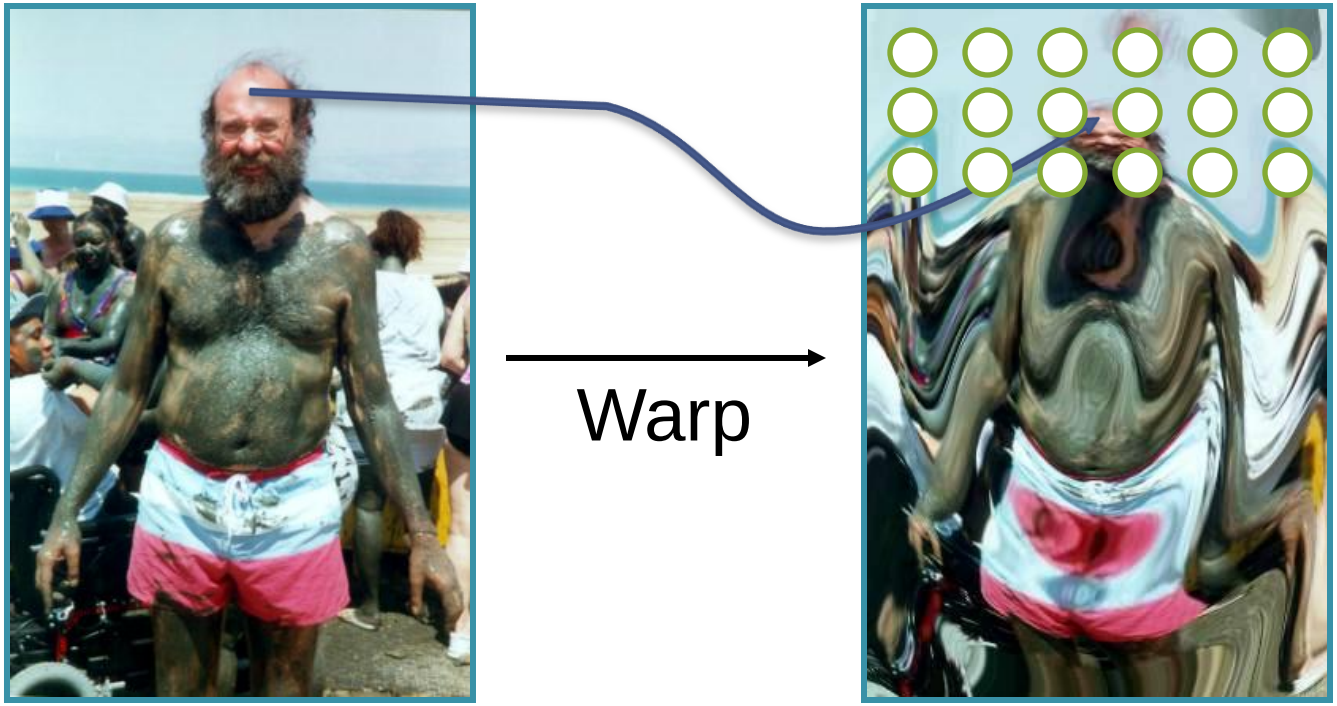
Source image

Warp

Destination image

Image Warping

- Issues
  - How do we specify where every pixel goes? (mapping)
  - How do we compute colors at destination pixels? (resampling)

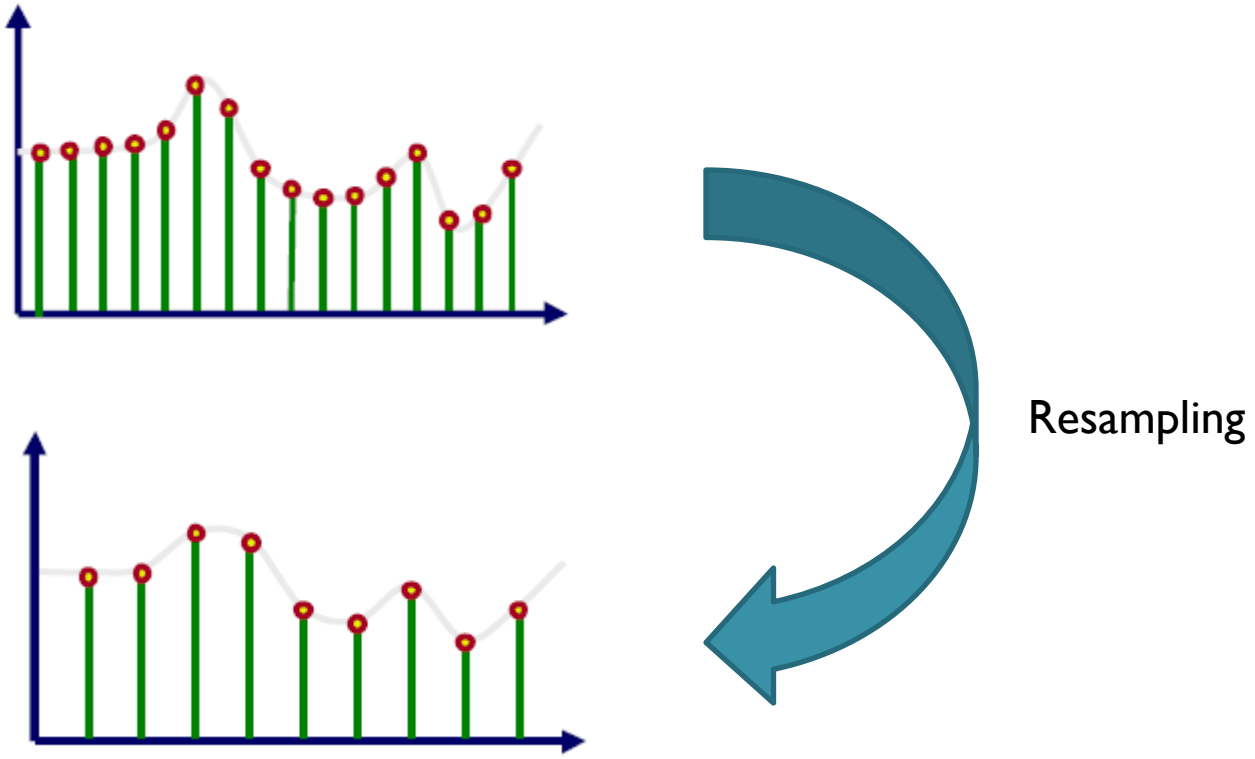


Example

- Image Scaling
  - (x',y') = (sx*x, sy*y);
  - I(x',y') = ?

Image Warping

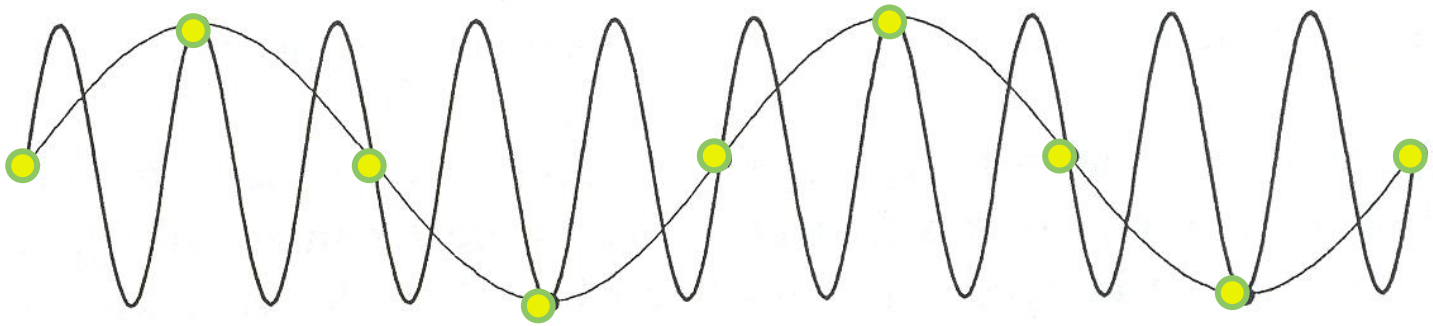
- Image warping requires resampling of image



BACK TO SAMPLING

Aliasing (again)

- In general:
  - Artifacts due to under-sampling or poor reconstruction
- Specifically, in graphics:
  - Spatial aliasing
  - Temporal aliasing

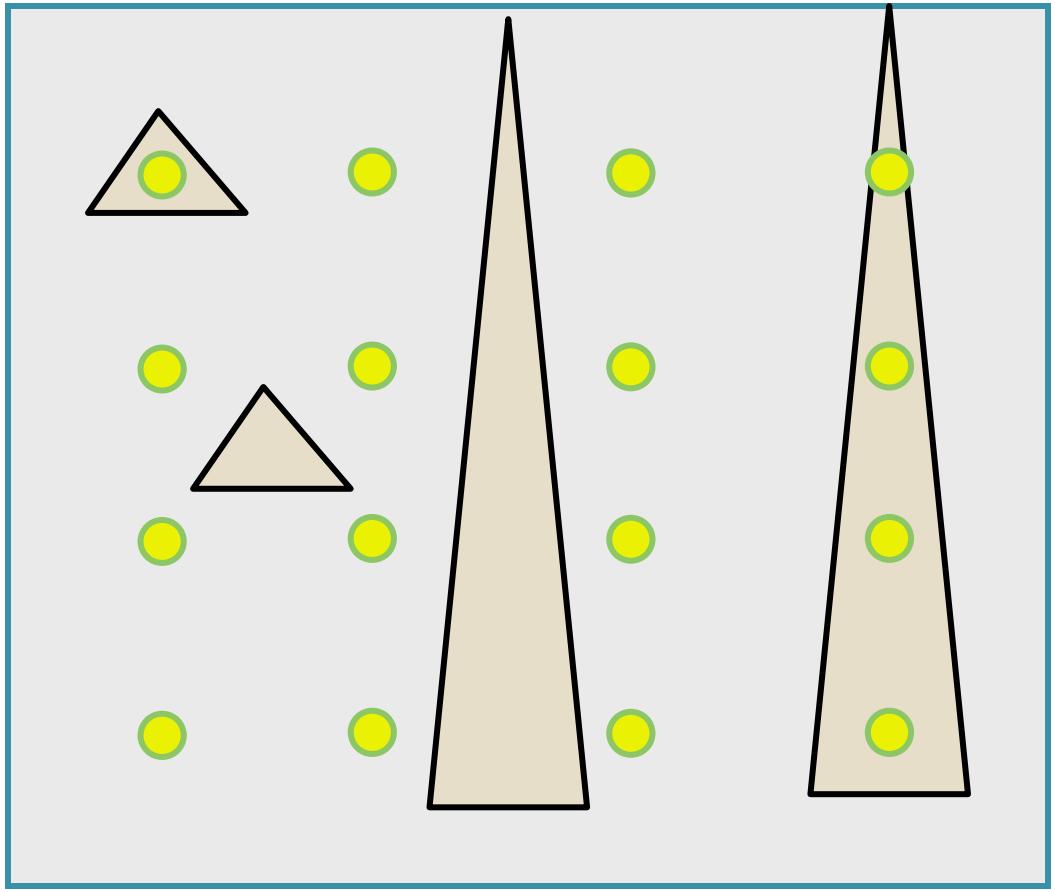


Under-sampling

Figure 14.17 FvDFH

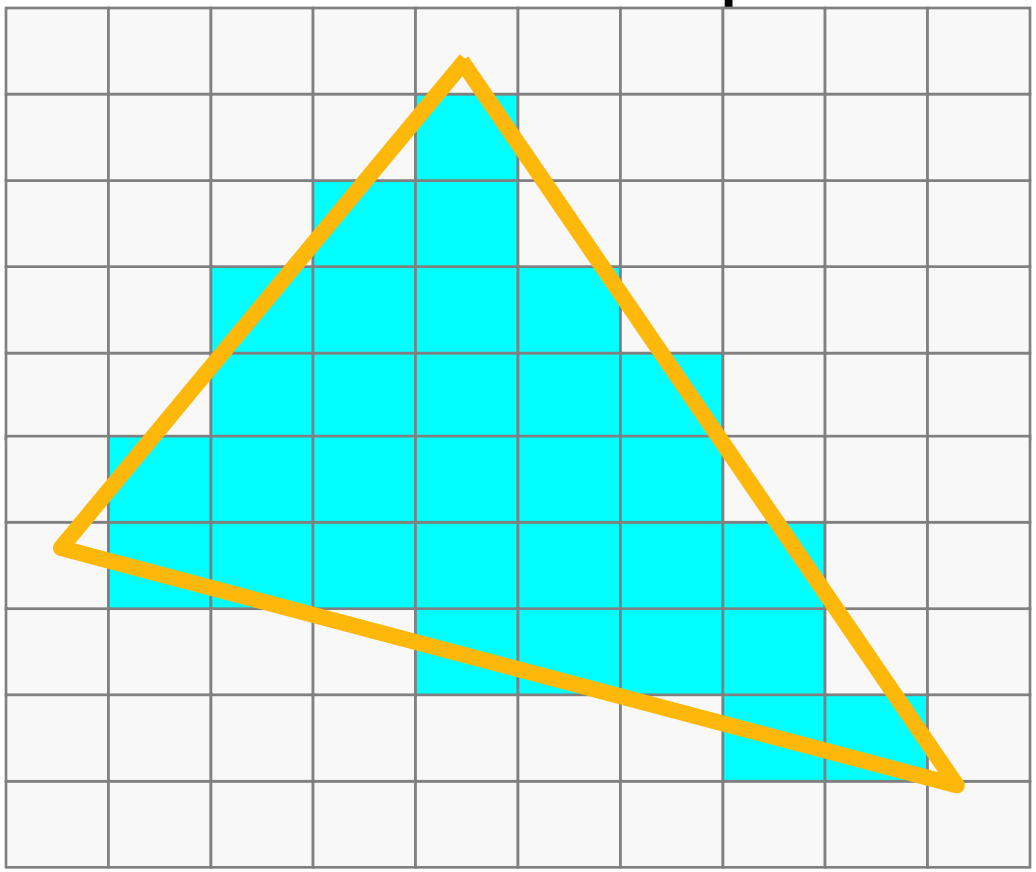
Spatial Aliasing

- Artifacts due to limited spatial resolution



Spatial Aliasing

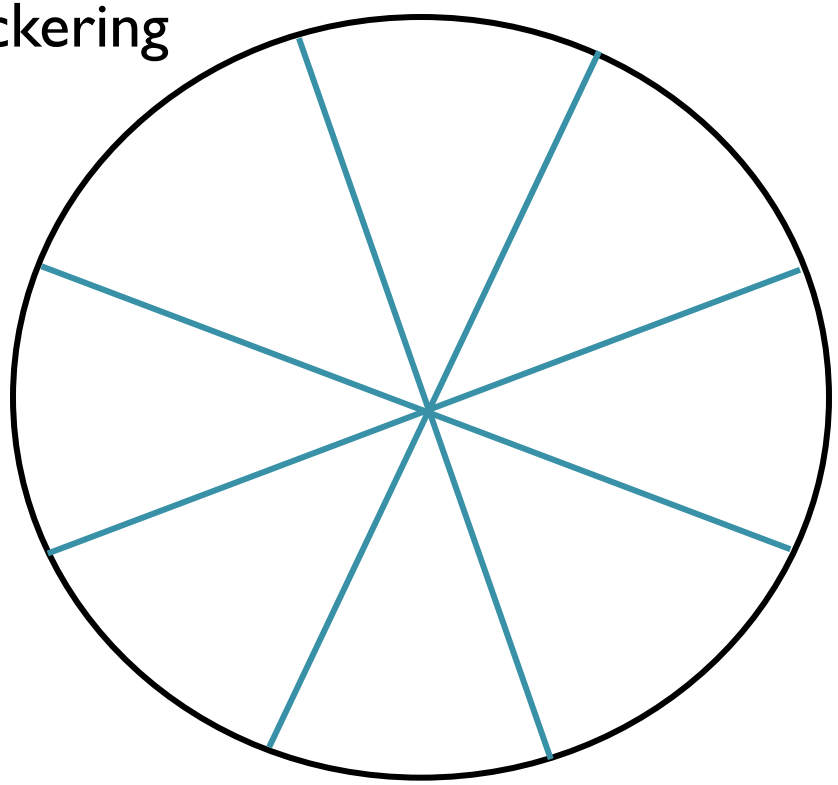
- Artifacts due to limited spatial resolution



"Jaggies"

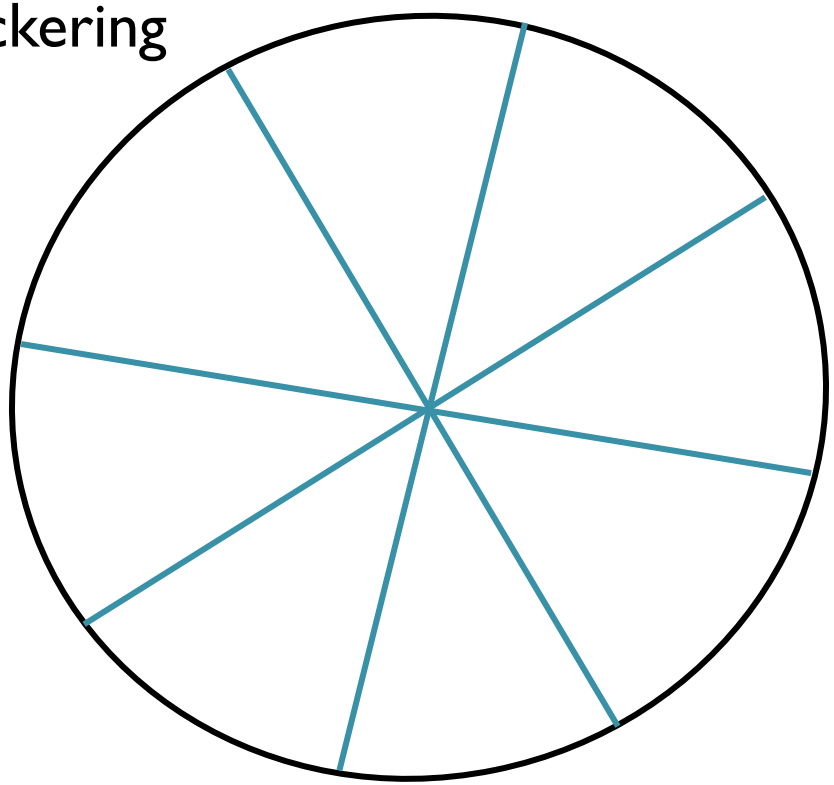
Temporal Aliasing

- Artifacts due to limited temporal resolution
  - Strobing
  - Flickering



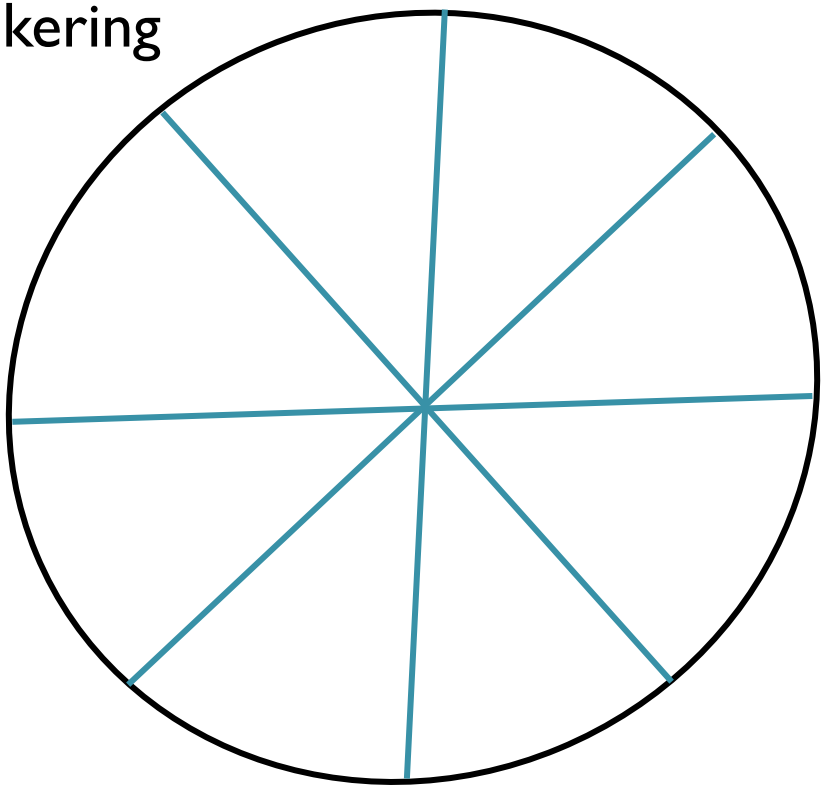
Temporal Aliasing

- Artifacts due to limited temporal resolution
  - Strobing
  - Flickering



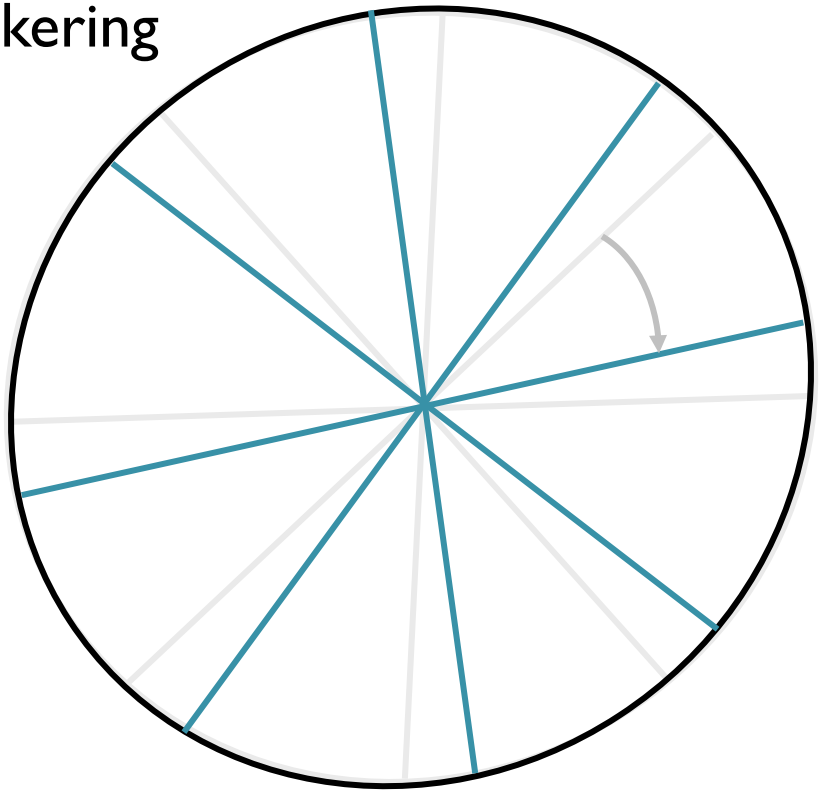
Temporal Aliasing

- Artifacts due to limited temporal resolution
  - Strobing
  - Flickering



Temporal Aliasing

- Artifacts due to limited temporal resolution
  - Strobing
  - Flickering



Antialiasing

- Sample at higher rate
  - Not always possible
  - Doesn't always solve problem
- Pre-filter to form bandlimited signal
  - Form bandlimited function (low-pass filter)
  - Trades aliasing for blurring

Image Processing

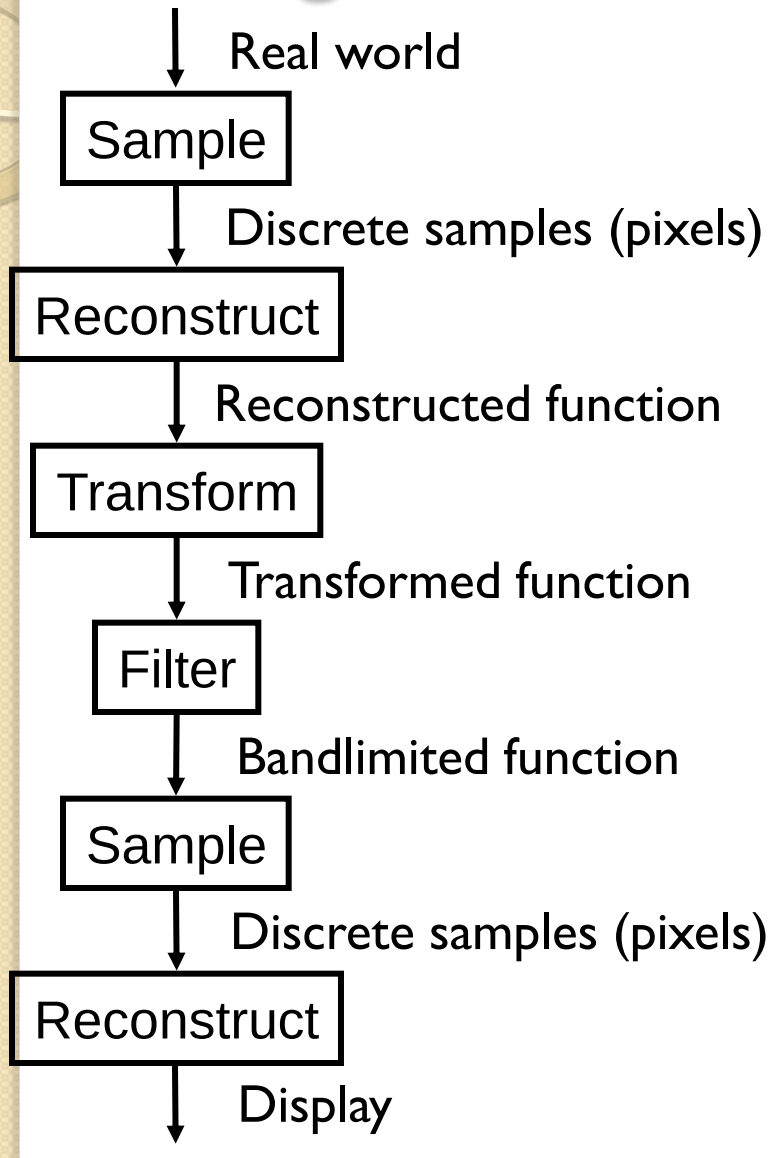


Image Processing

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

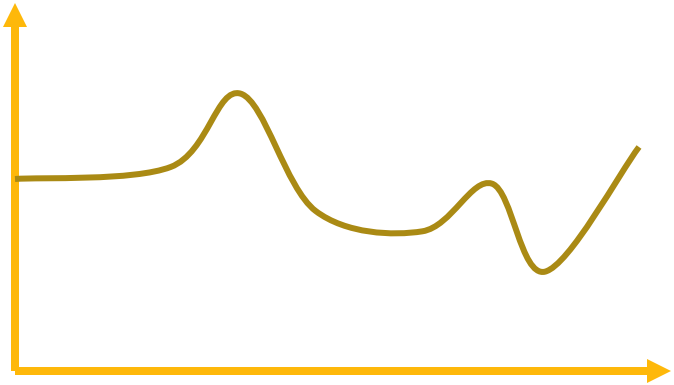
Bandlimited function

Sample

Discrete samples (pixels)

Reconstruct

Display



Continuous Function

Image Processing

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

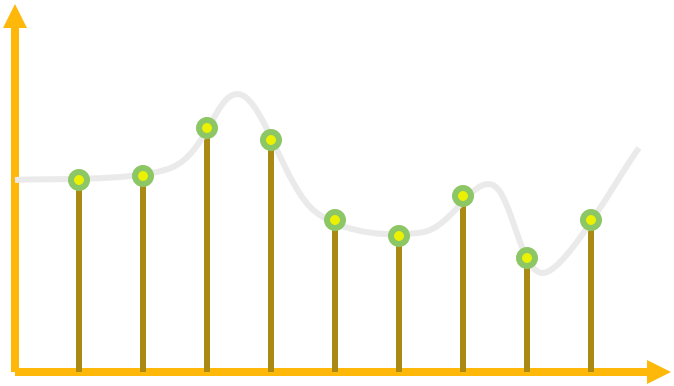
Bandlimited function

Sample

Discrete samples (pixels)

Reconstruct

Display



Discrete Samples

Image Processing

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

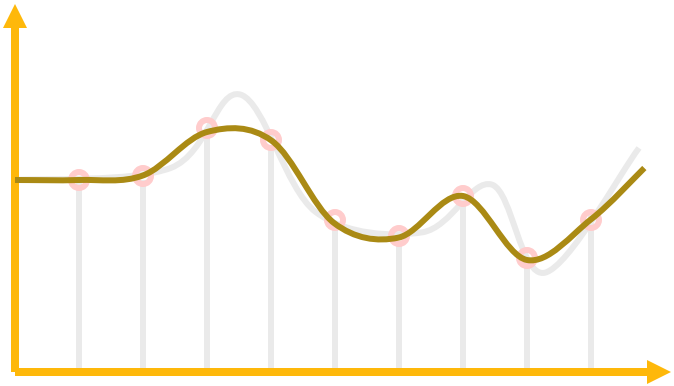
Bandlimited function

Sample

Discrete samples (pixels)

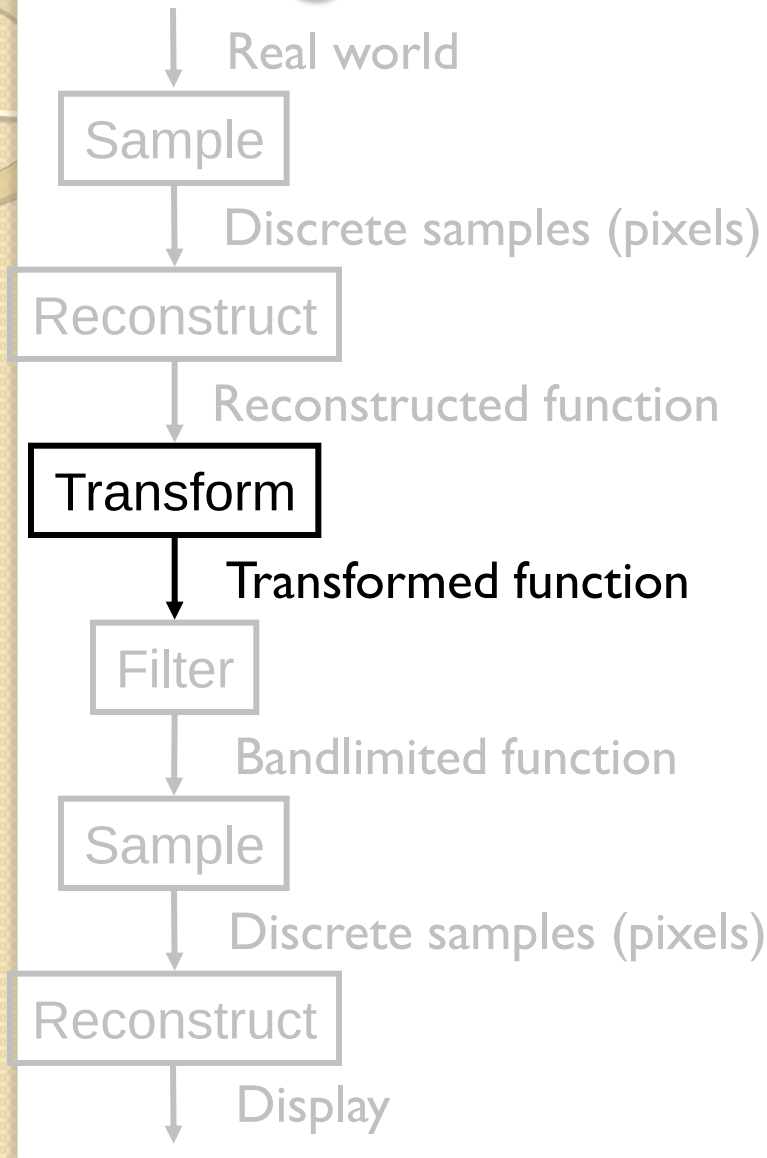
Reconstruct

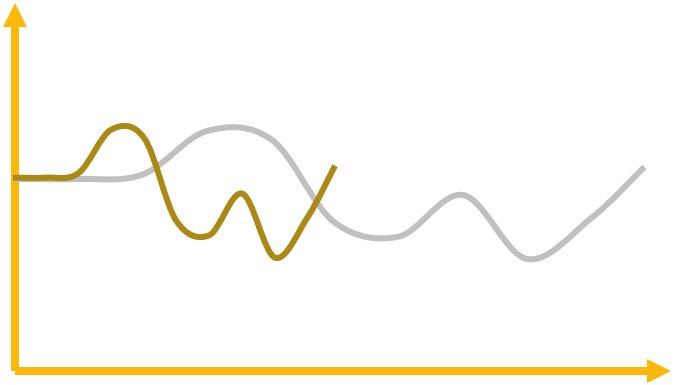
Display



Reconstructed Function

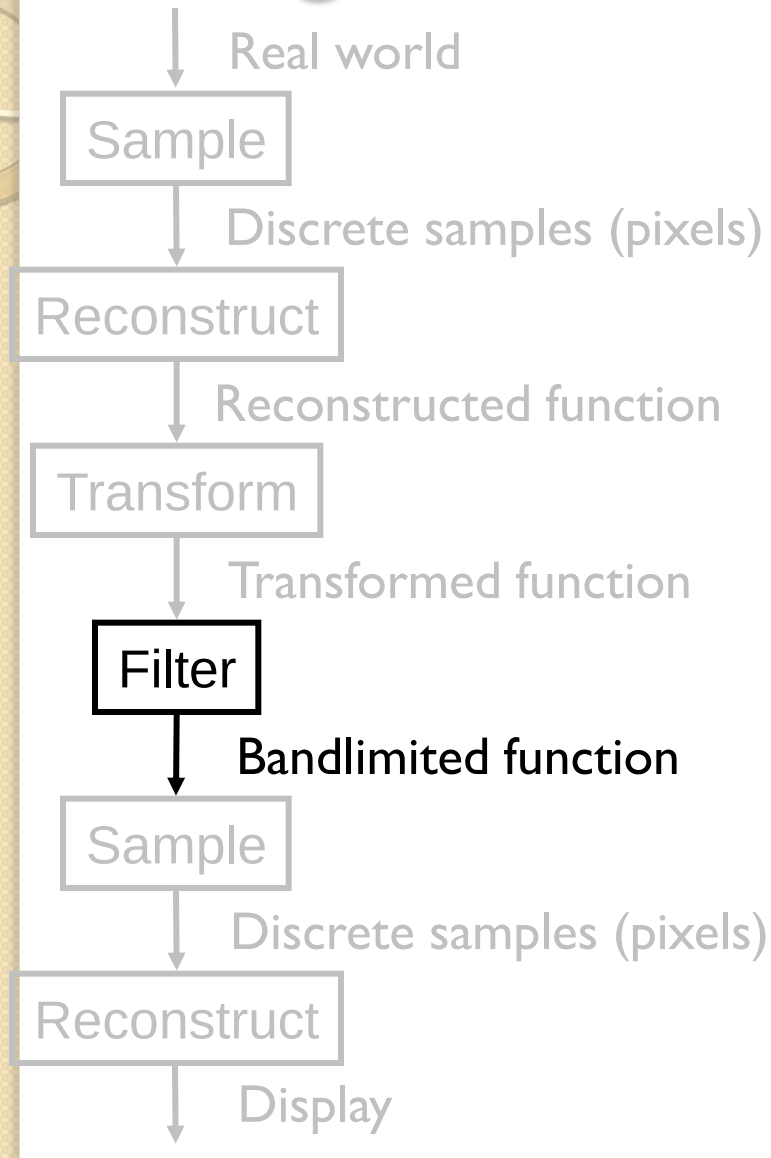
Image Processing

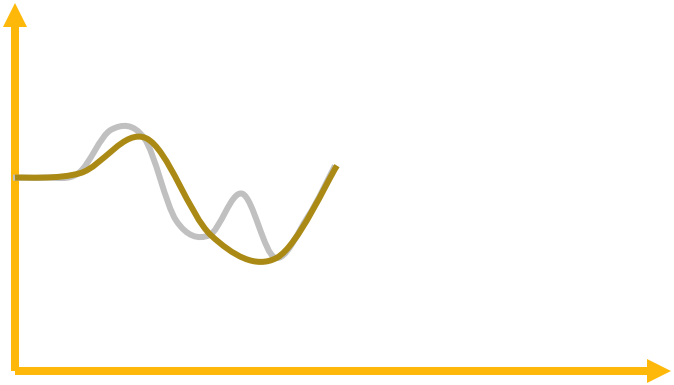




Transformed Function

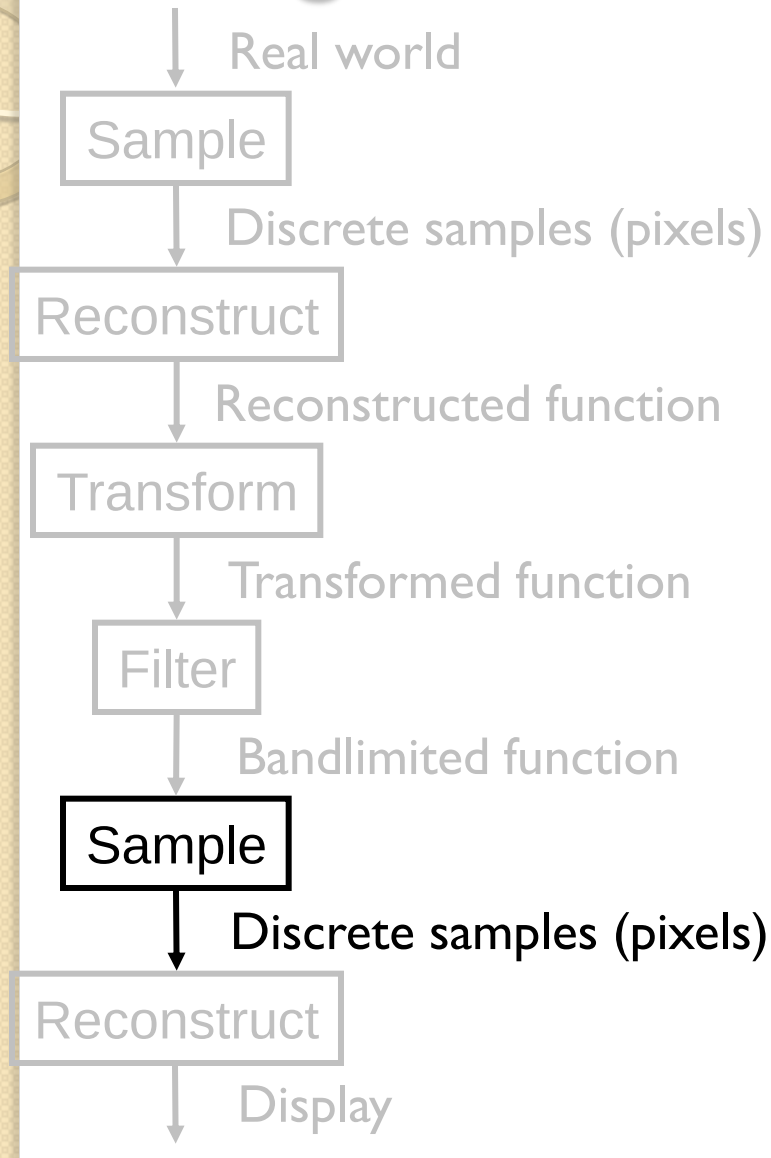
Image Processing

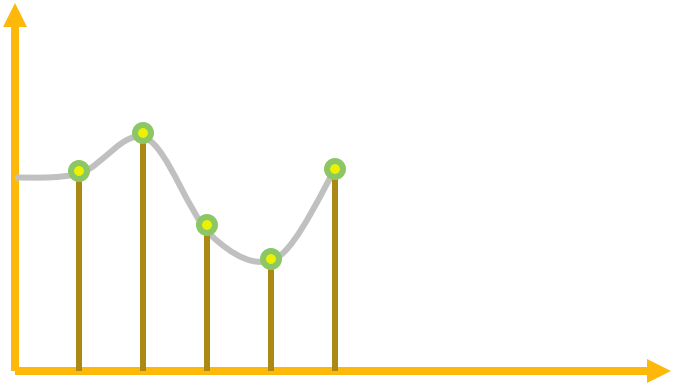




Bandlimited Function

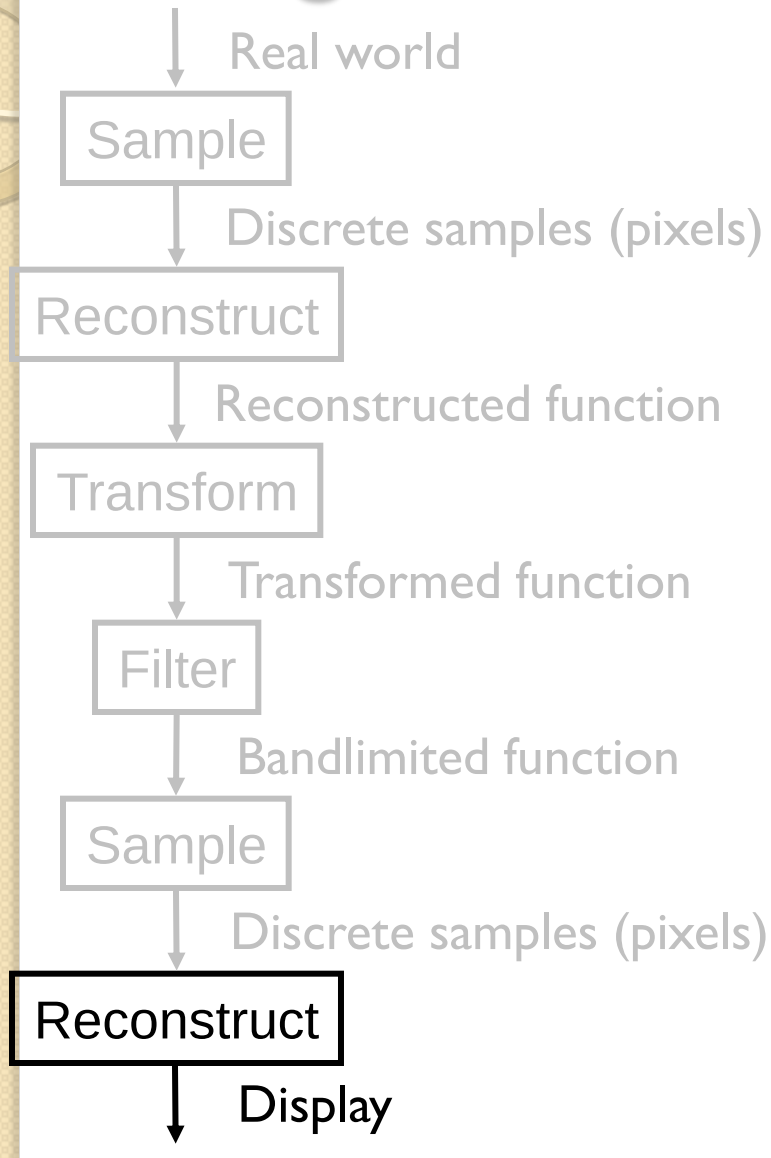
Image Processing

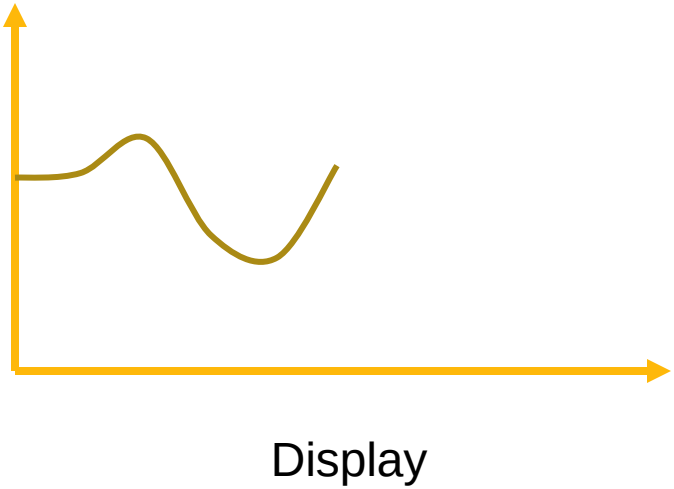




Discrete samples

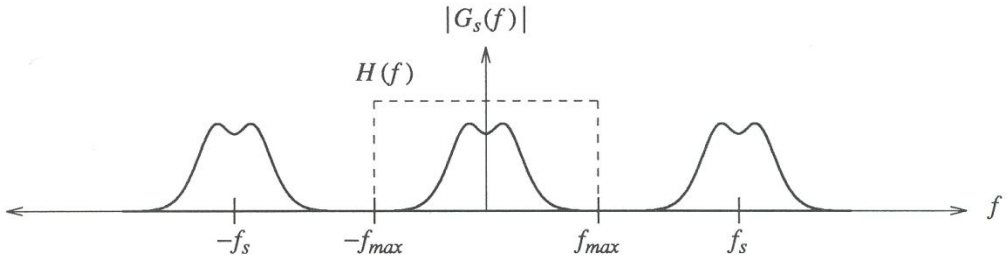
Image Processing



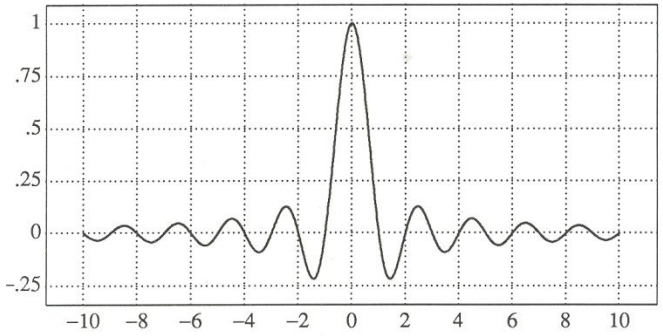


Ideal Bandlimiting Filter

- Frequency domain



- Spatial domain

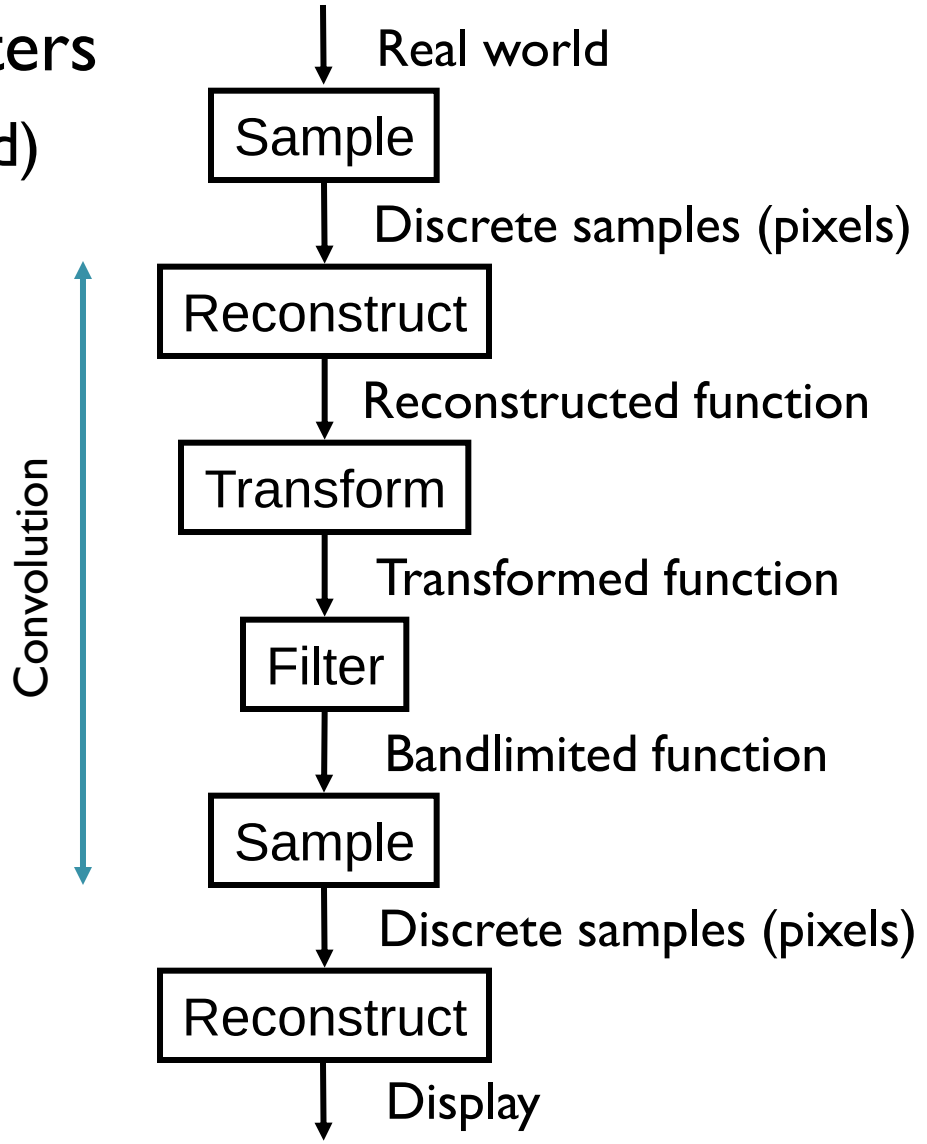


$$Sinc(x) = \frac{\sin \pi x}{\pi x}$$

Figure 4.5 Wolberg

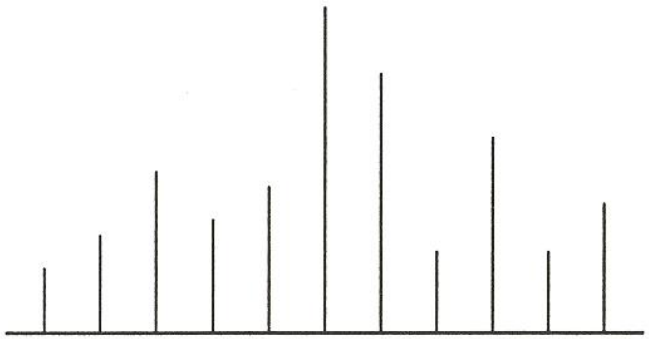
Practical Image Processing

- Finite low-pass filters
  - Point sampling (bad)
  - Triangle filter
  - Gaussian filter

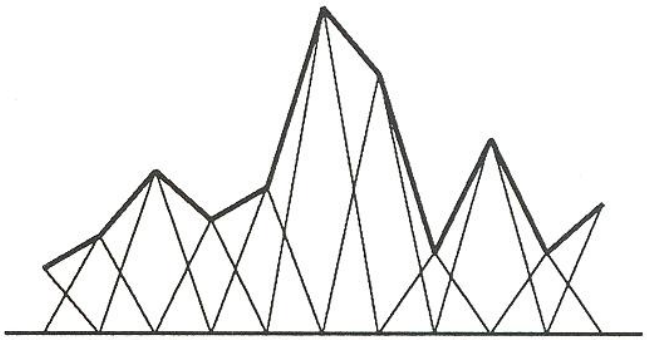


Triangle Filter

- Convolution with triangle filter



Input

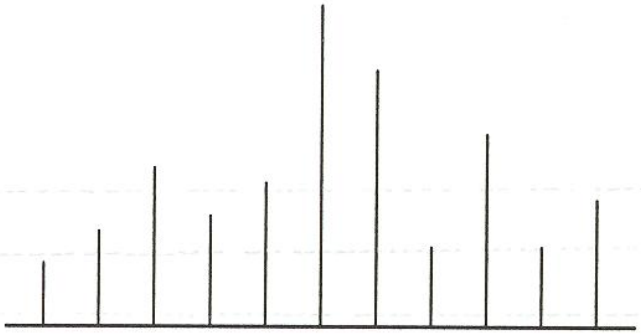


Output

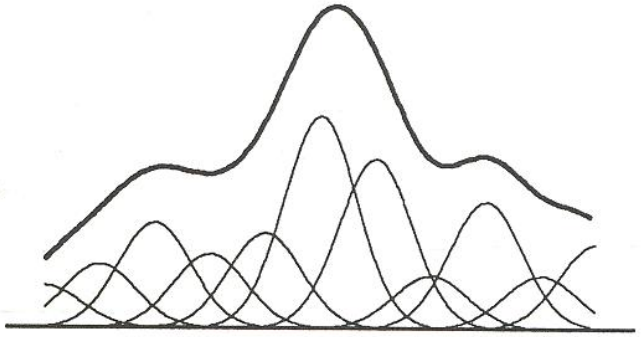
Figure 2.4 Wolberg

Gaussian Filter

- Convolution with Gaussian filter



Input



Output

Figure 2.4 Wolberg

AND BACK TO WARPING

Image Resampling

- What if we are resampling a 2D image?

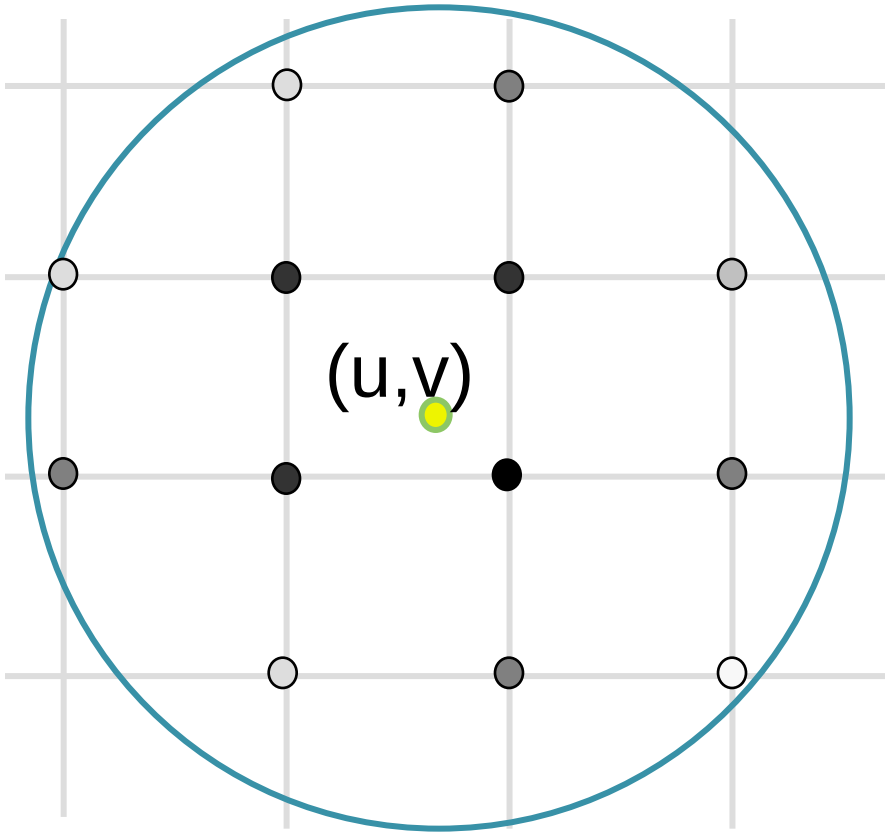


Image Resampling

- Compute weighted sum of pixel neighborhood
  - Output is weighted average

```
dst(u,v)=0;
for(ix=u-w;ix<=u+w;ix++)
  for(iy=v-w;iy<=v+w;iy++)
    d=dist between (ix,iy) and (u,v)
    dst(u,v) += k(ix,iy) * src(ix,iy)
```

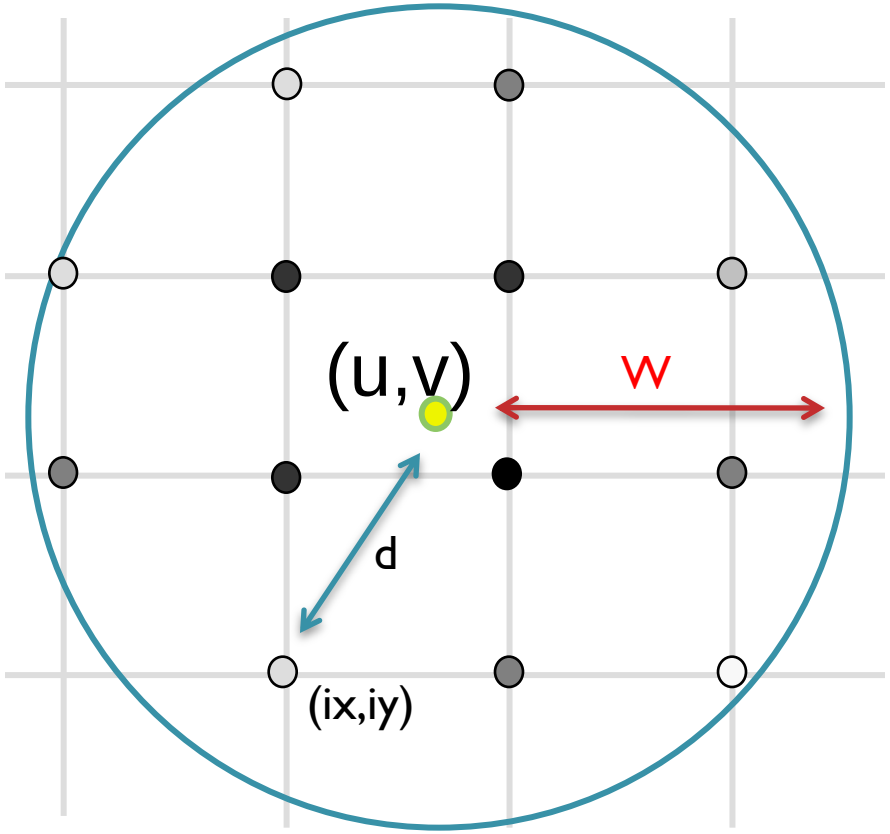
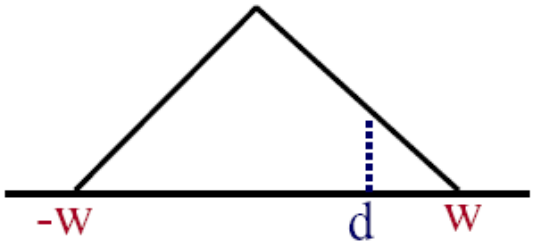


Image Resampling

- For isotropic Triangle and Gaussian filters, k(ix,iy) is a function of d and w



Triangle filter

$k(i,j)=1 - d/w$

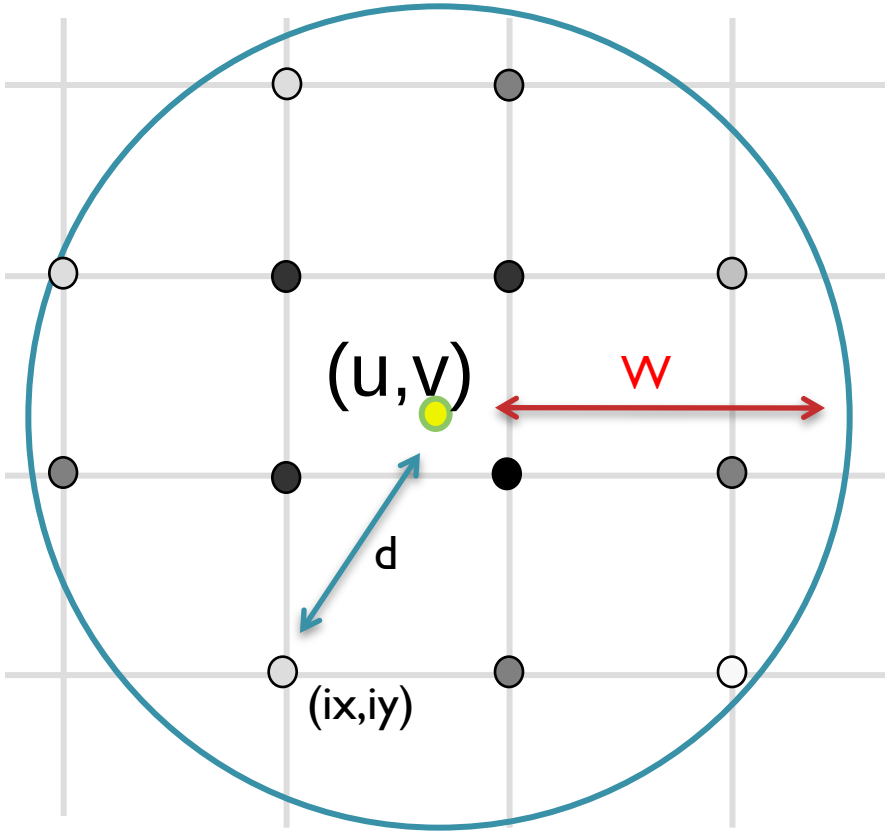
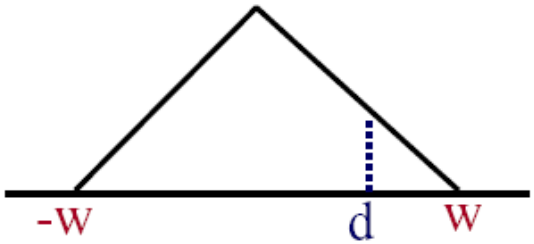


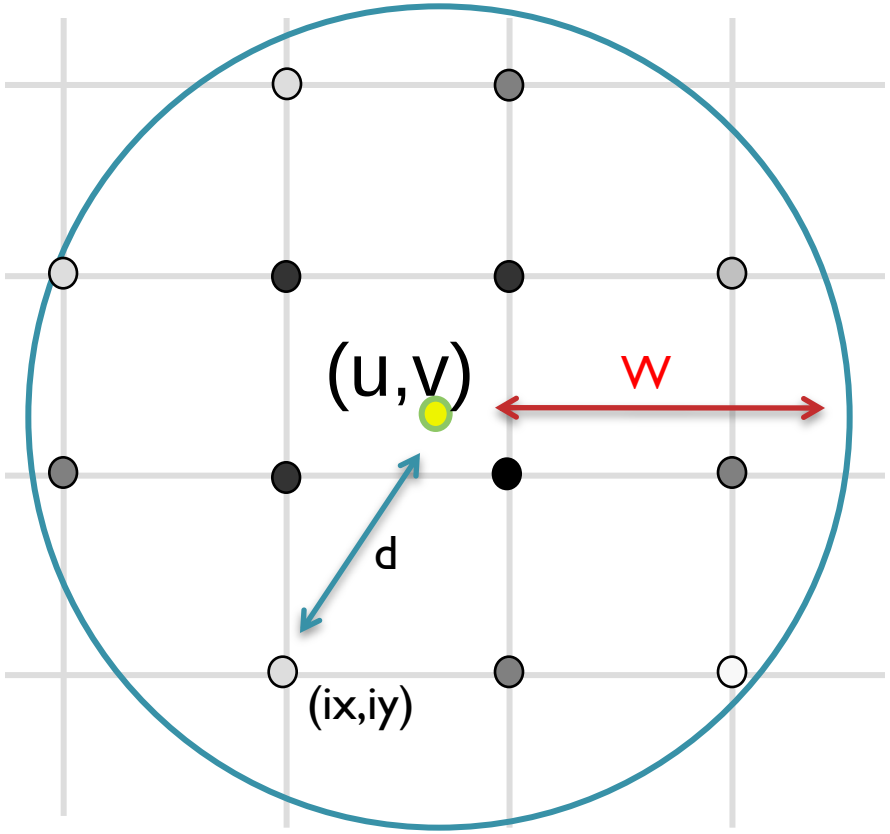
Image Resampling

- For isotropic Triangle and Gaussian filters, k(ix,iy) is a function of d and w



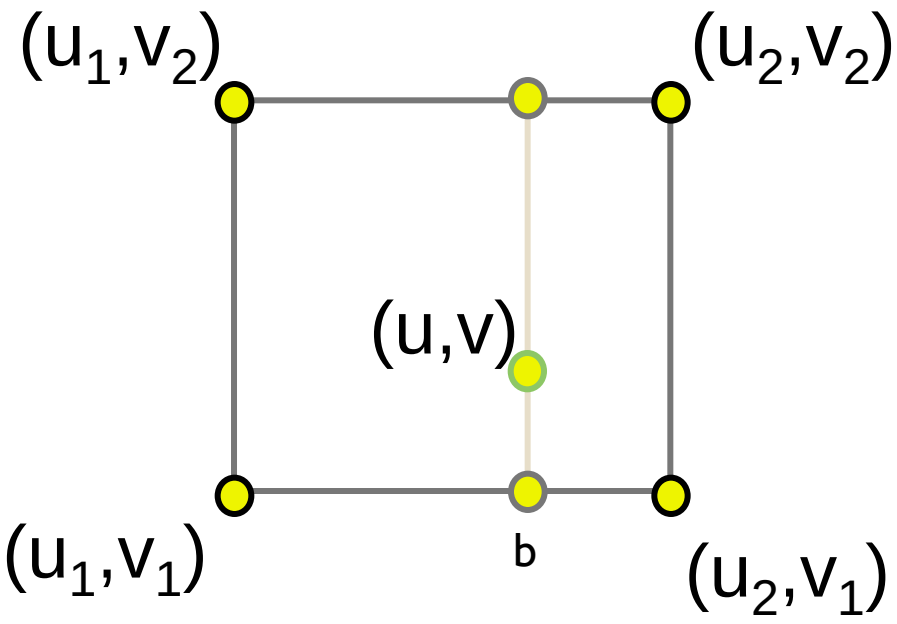
Triangle filter

$k(i,j)=1 - d/w$



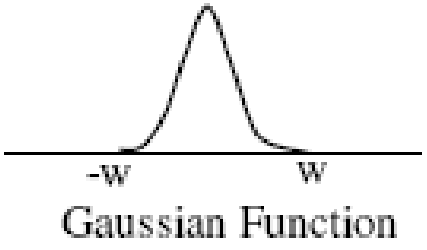
Triangle Filtering (width <= 1)

- Bilinearly interpolate four closest pixels
  - a = linear interpolation of src(u_1,v_2) and src(u_2,v_2)
  - b = linear interpolation of src(u_1,v_1) and src(u_2,v_1)
  - dst(x,y) = linear interpolation of "a" and "b"



Gaussian Filtering

- Kernel is a Guassian function



$$G_\sigma(d) = 2^{-(d/\sigma)^2}$$

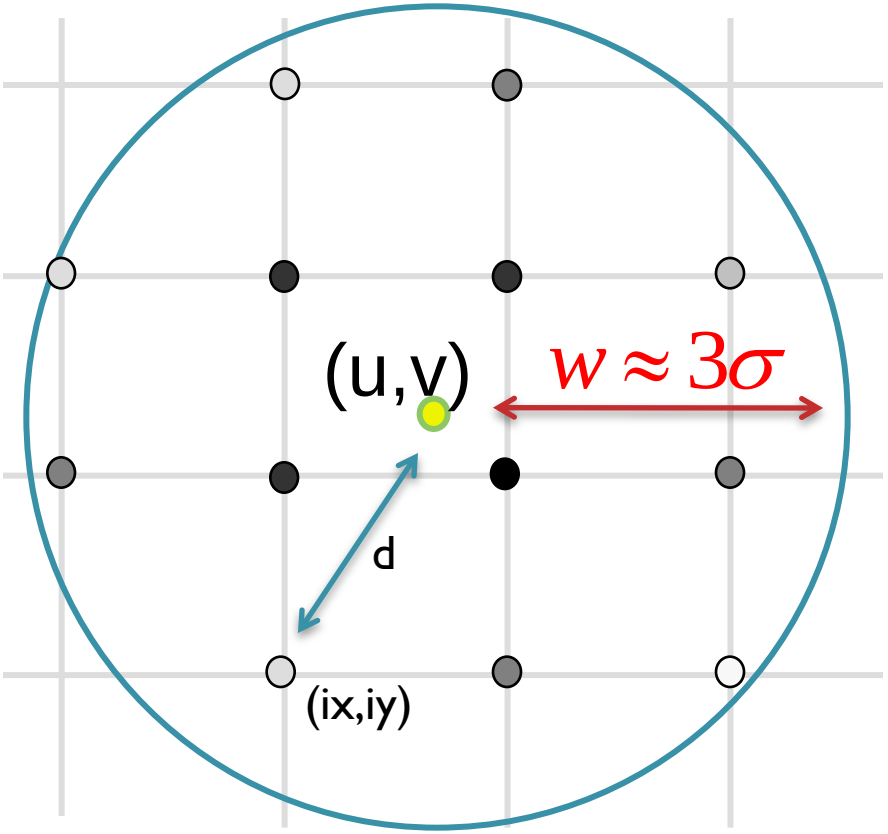
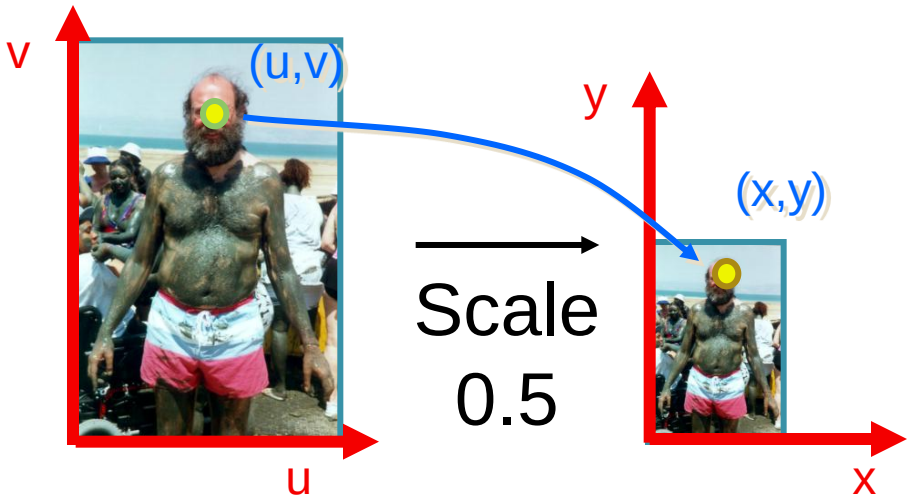


Image Scale

- Scale (src, dst, sx, sy):

```
w ≅ max(1/sx,1/sy);
for (int ix = 0; ix < xmax; ix++) {
  for (int iy = 0; iy < ymax; iy++) {
    float u = ix / sx;
    float v = iy / sy;
    dst(ix,iy) = resample(src,u,v,k,w);
  }
}
```



How do we resample?

- Point sampling
  - Simple but causes aliasing

```
Float resample(src,u,v,w) {
   int iu = round(u);
   int iv = round(v);
   return src(iu,iv);
}
```

- Triangle and Gaussian
  - Algorithm as we saw earlier

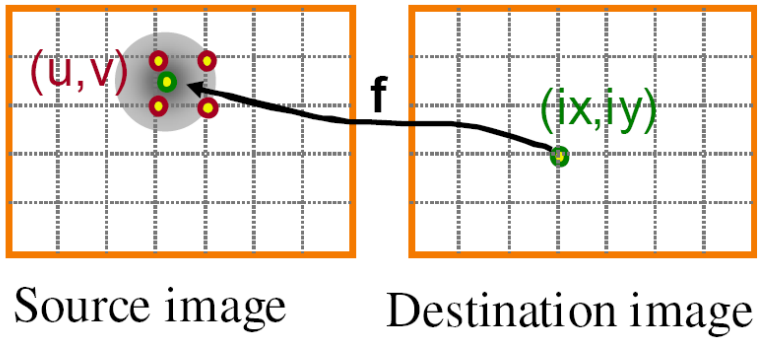


Image Warping (in General)

- Reverse Mapping

```
Warp(src, dst) {
  for (int ix = 0; ix < xmax; ix++) {
    for (int iy = 0; iy < ymax; iy++) {
      float w ≈ 1 / scale(ix, iy);
      float u = f_x^-1(ix,iy);
      float v = f_y^-1(ix,iy);
      dst(ix,iy) = Resample(src,u,v,w);
    }
  }
}
```

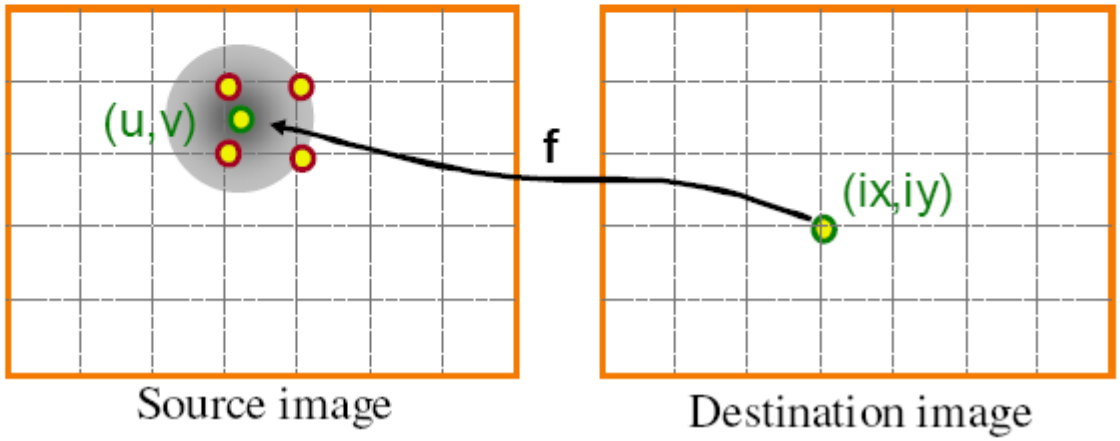
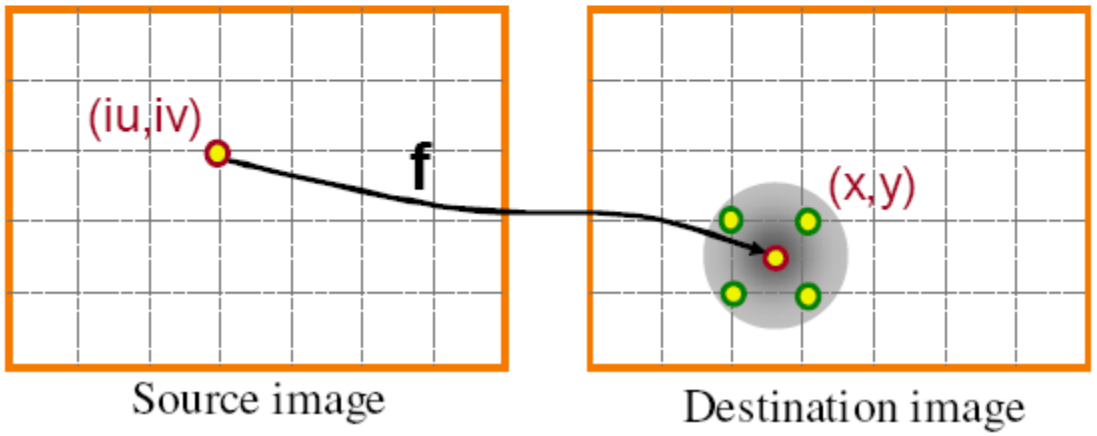


Image Warping (in General)

- Alternative (forward)

```
Warp(src, dst) {
  for (int iu = 0; iu < umax; iu++) {
    for (int iv = 0; iv < vmax; iv++) {
      float x = f_x(iu,iv);
      float y = f_y(iu,iv);
      float w ≈ 1 / scale(x, y);
      Splat(src(iu,iv), x, y, w);      weighting ???
    }
  }
}
```



That's it for today

- Next time?
  - Finishing corners on image processing
  - Transformations and Projections
  - Rendering