



Sampling, Resampling, and Warping

COS 426, Spring 2016
Adam Finkelstein



Image Processing Operations I

- Luminance
 - Brightness
 - Contrast.
 - Gamma
 - Histogram equalization
- Color
 - Black & white
 - Saturation
 - White balance
- Linear filtering
 - Blur & sharpen
 - Edge detect
 - Convolution
- Non-linear filtering
 - Median
 - Bilateral filter
- Dithering
 - Quantization
 - Ordered dither
 - Floyd-Steinberg

Last
Thursday

Image Processing Operations II



- Transformation

- Scale
- Rotate
- Warp



Today

- Combining images

- Composite
- Morph
- Comp photo

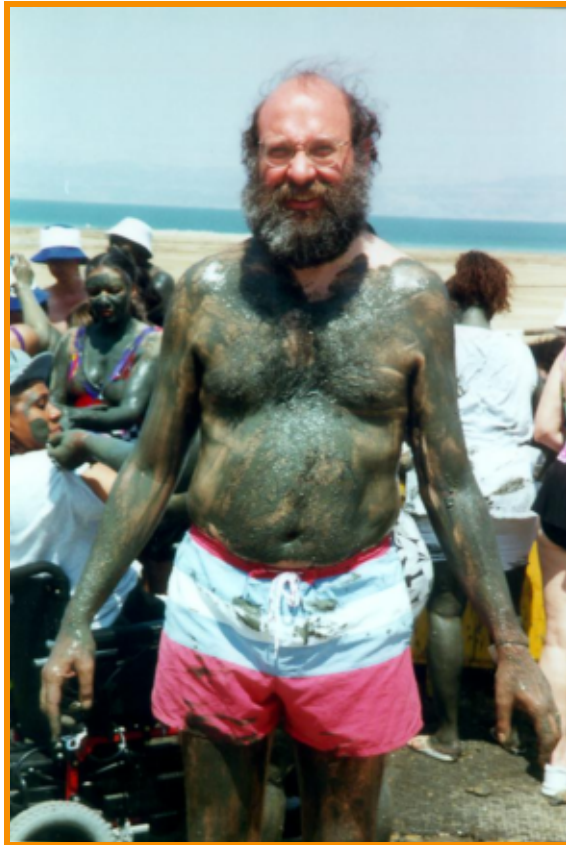


Thursday

Image Transformation



- Move pixels of an image



Source image

→
Warp

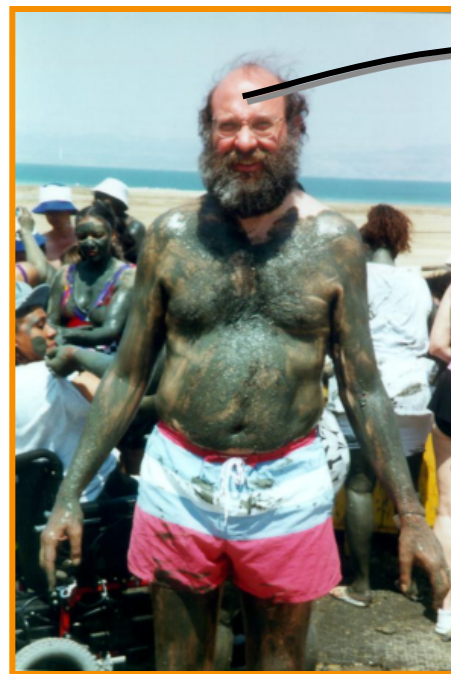


Destination image

Image Transformation



- Issues:
 - 1) Specifying where every pixel goes (mapping)



Source image

Warp



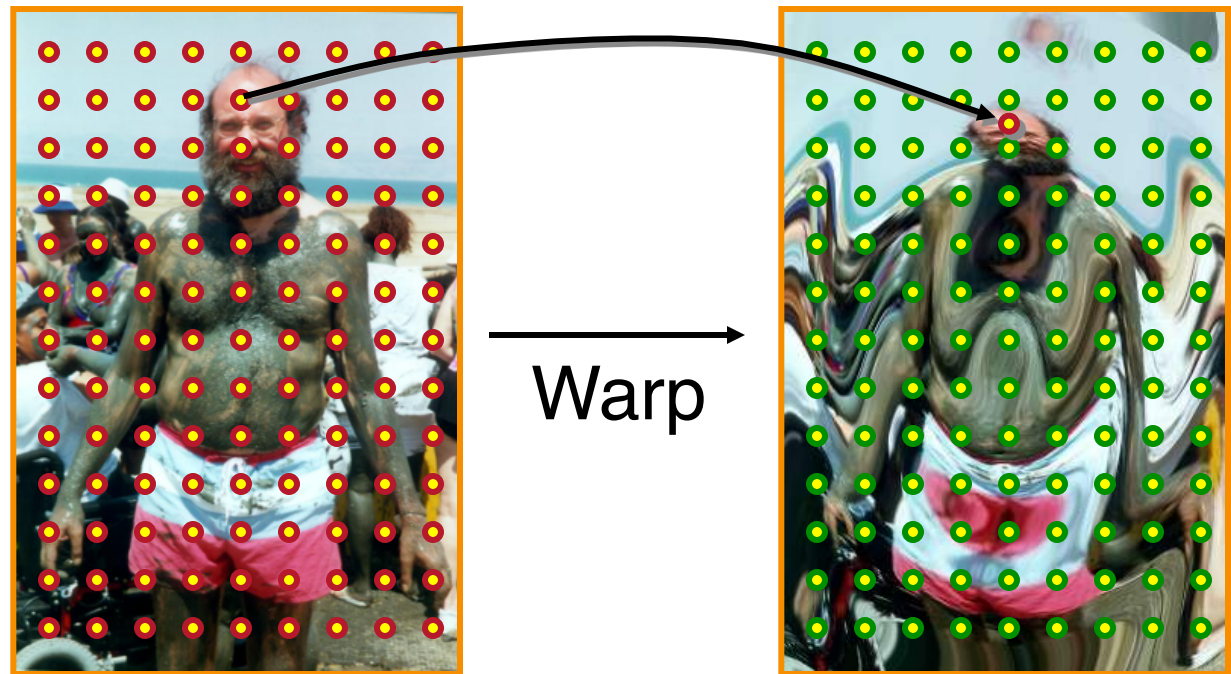
Destination image

Image Transformation



- Issues:

- 1) Specifying where every pixel goes (mapping)
- 2) Computing colors at destination pixels (resampling)



Source image

Destination image

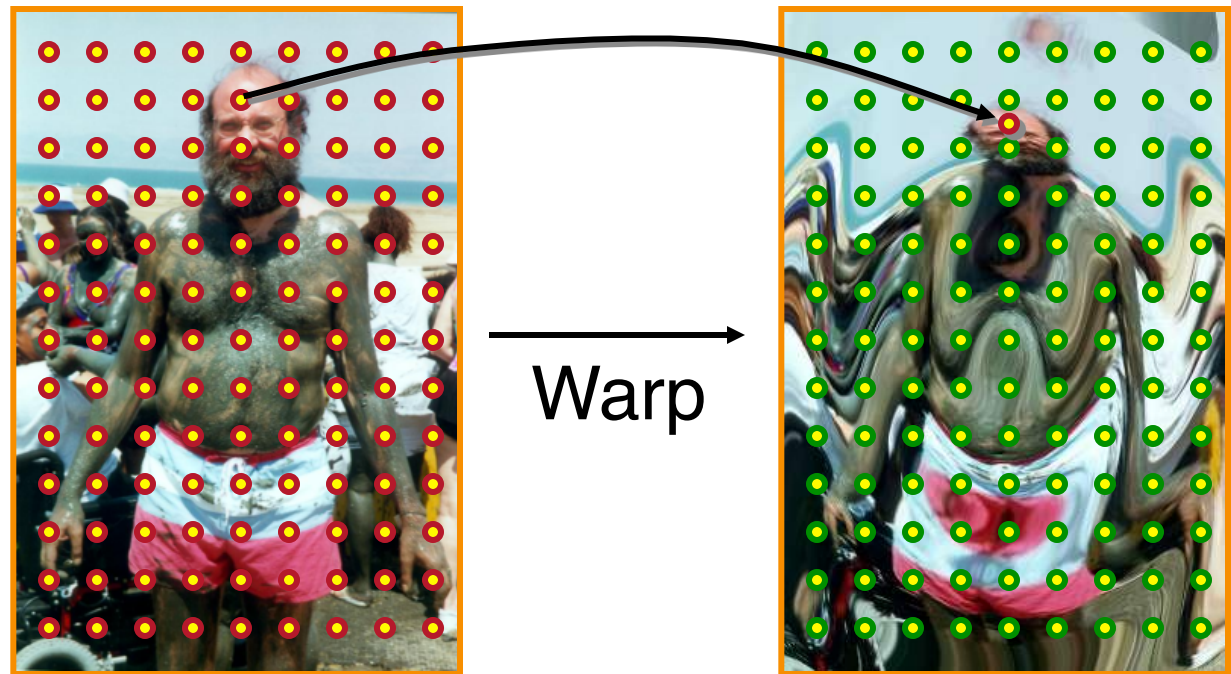
Image Transformation



- Issues:

1) Specifying where every pixel goes (mapping)

2) Computing colors at destination pixels (resampling)



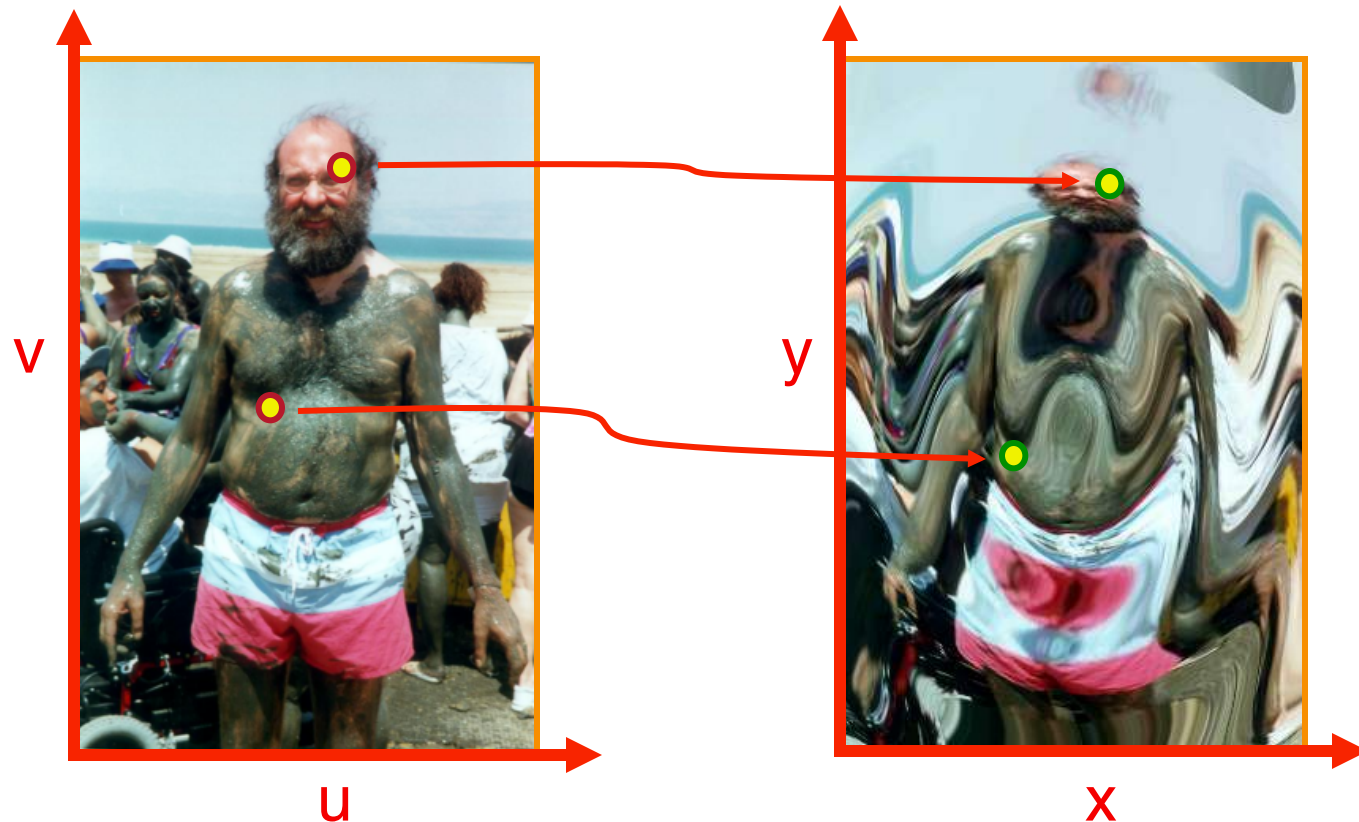
Source image

Destination image

Mapping

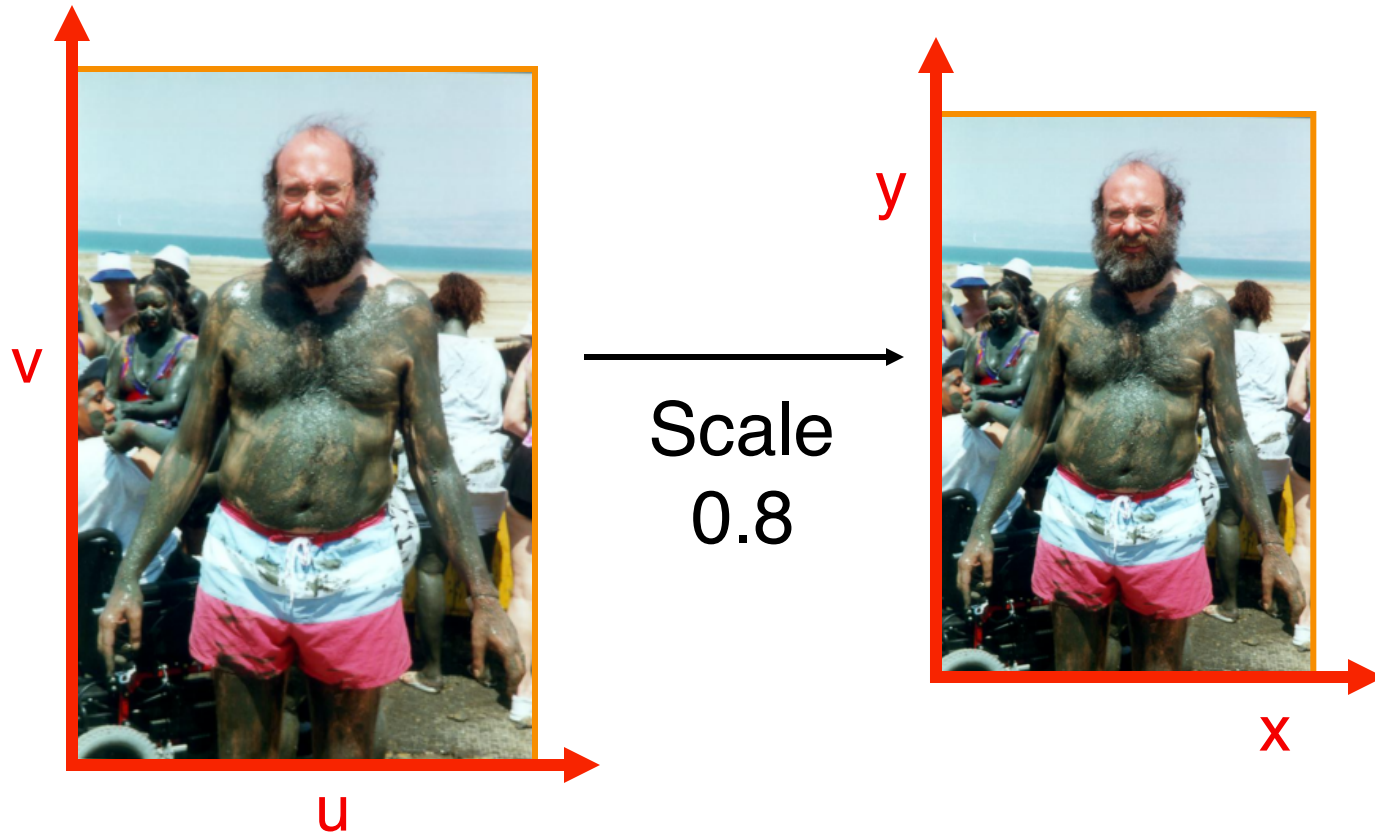


- Define transformation
 - Describe the destination (x,y) for every source (u,v)



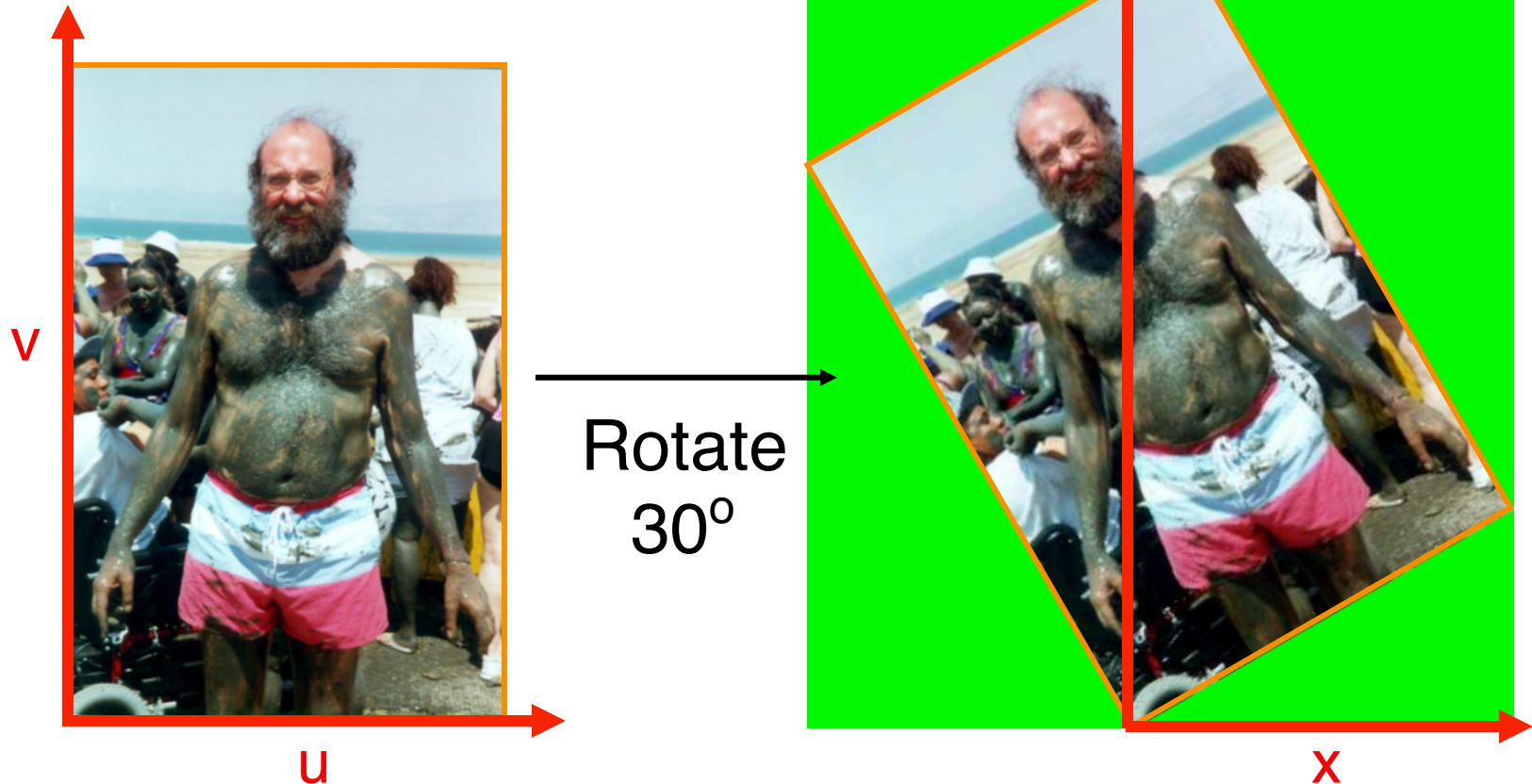
Parametric Mappings

- Scale by *factor*:
 - $x = factor * u$
 - $y = factor * v$



Parametric Mappings

- Rotate by Θ degrees:
 - $x = u \cos \Theta - v \sin \Theta$
 - $y = u \sin \Theta + v \cos \Theta$

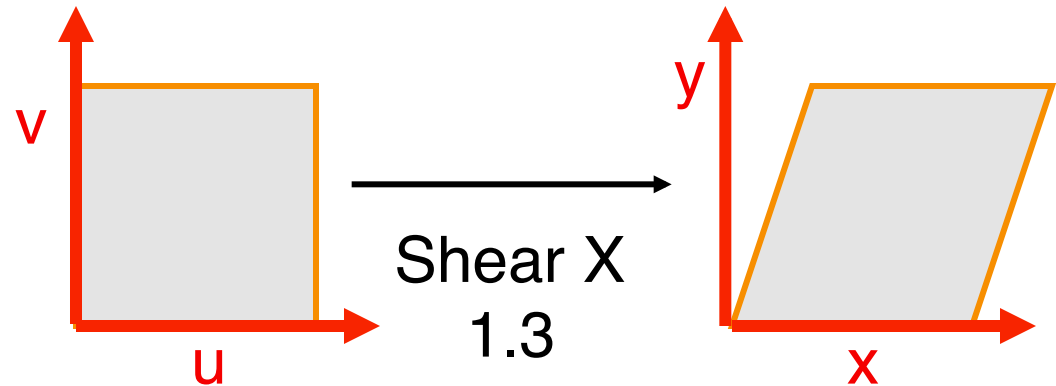




Parametric Mappings

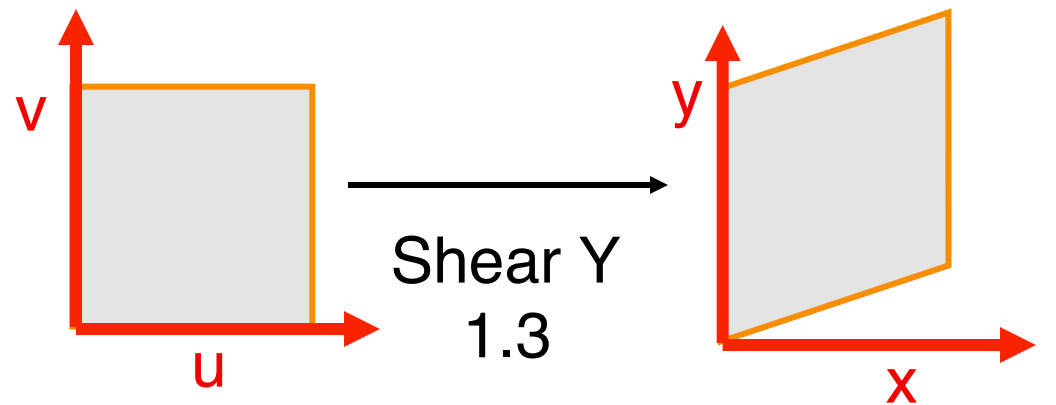
- Shear in X by *factor*:

- $x = u + \textit{factor} * v$
- $y = v$



- Shear in Y by *factor*:

- $x = u$
- $y = v + \textit{factor} * u$

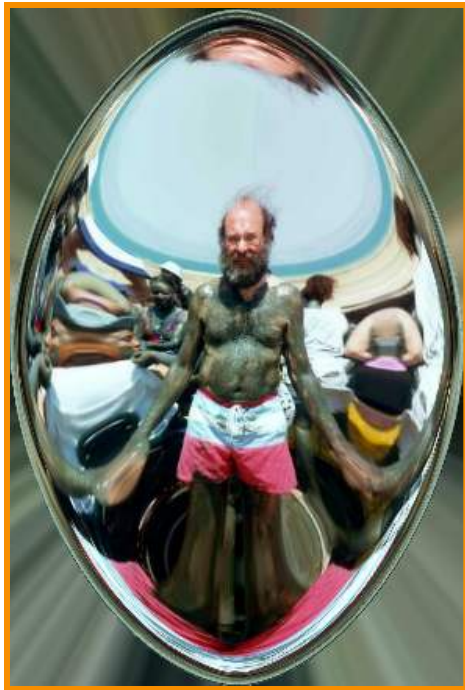


*Non-obvious fact:
You can make rotate
out of three shears.*

Other Parametric Mappings



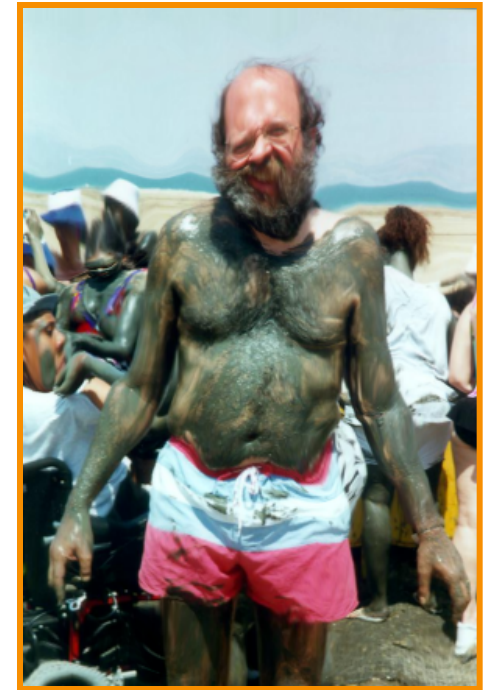
- Any function of u and v :
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$



Fish-eye



“Swirl”



“Rain”

COS426 Examples



Aditya Bhaskara



Wei Xiang

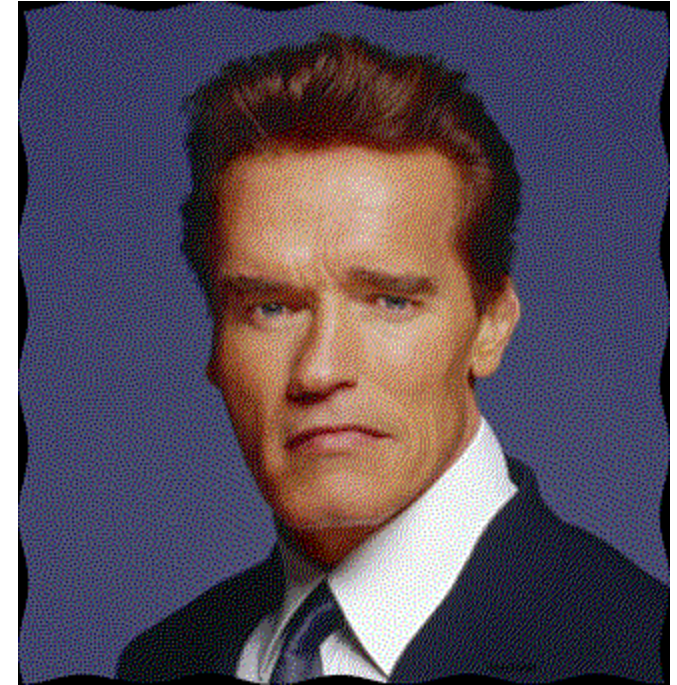
More COS426 Examples



Sid Kapur



Michael Oranato

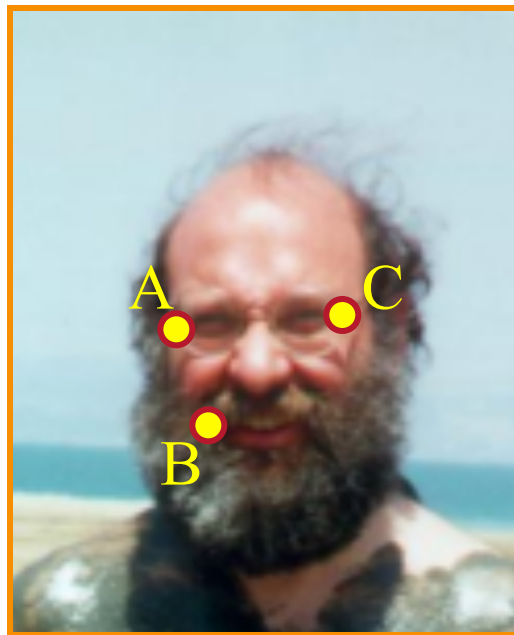


Eirik Bakke

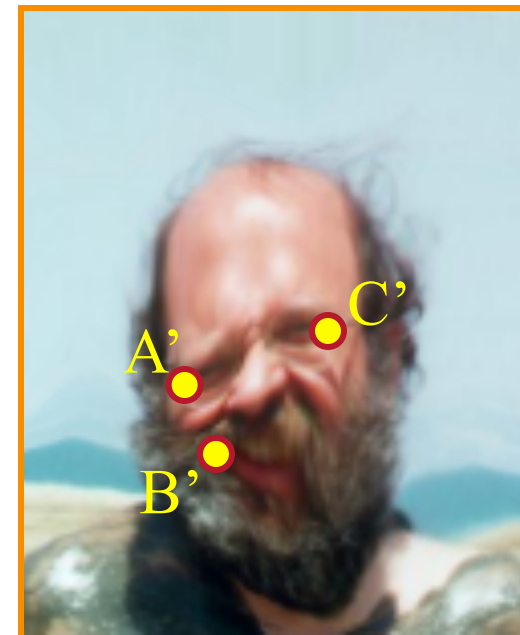
Point Correspondence Mappings



- Mappings implied by correspondences:
 - $A \leftrightarrow A'$
 - $B \leftrightarrow B'$
 - $C \leftrightarrow C'$



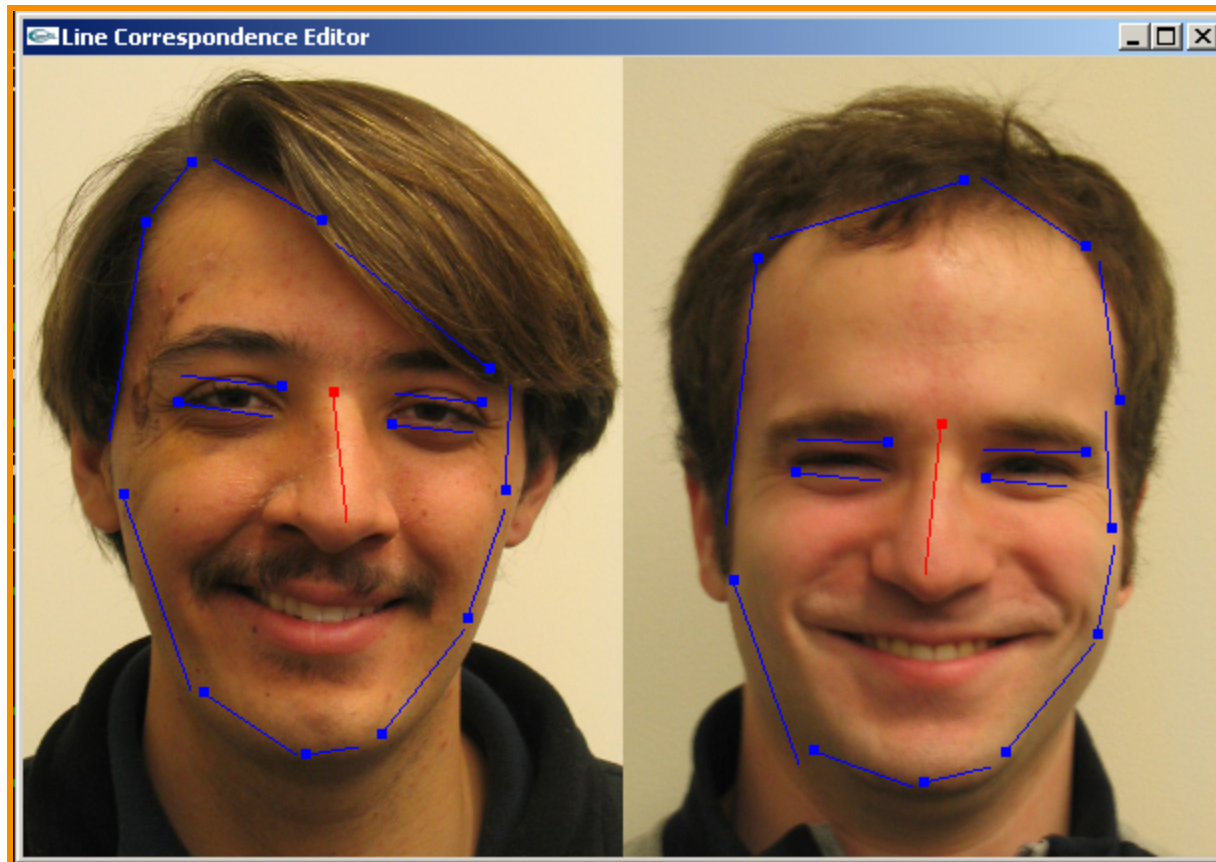
Warp



Line Correspondence Mappings



- Beier & Neeley use pairs of lines to specify warps



Discussed in next lecture....

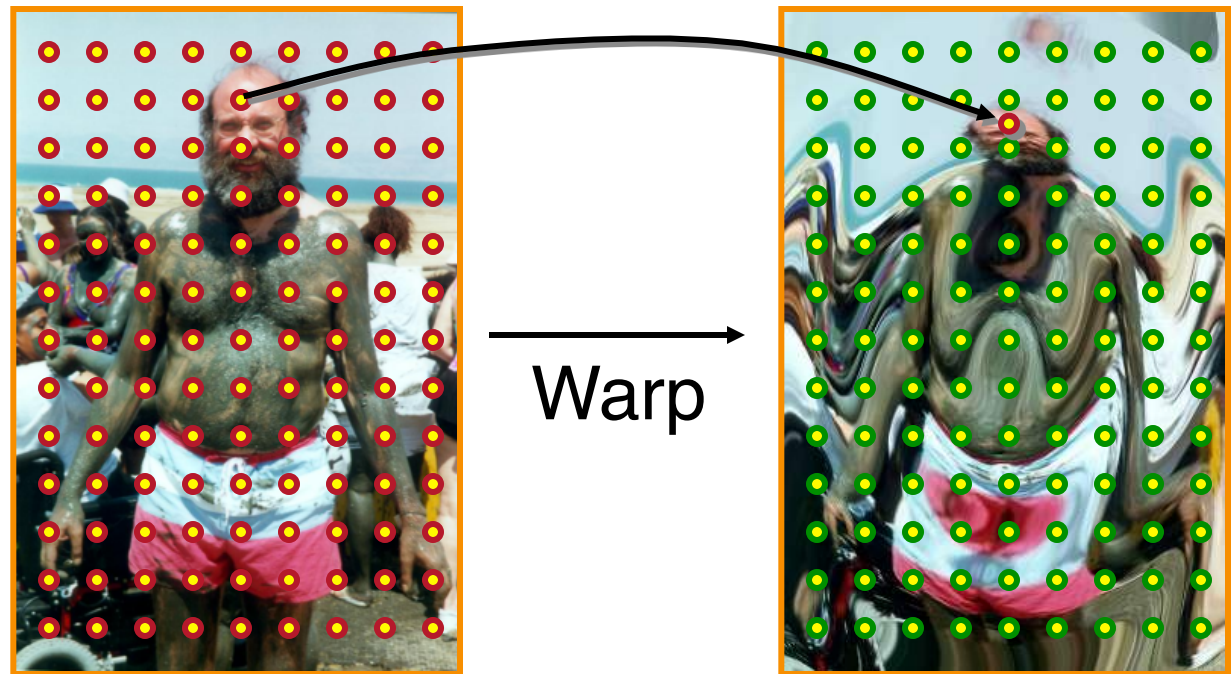
Image Transformation



- Issues:

- 1) Specifying where every pixel goes (mapping)

- 2) Computing colors at destination pixels (resampling)

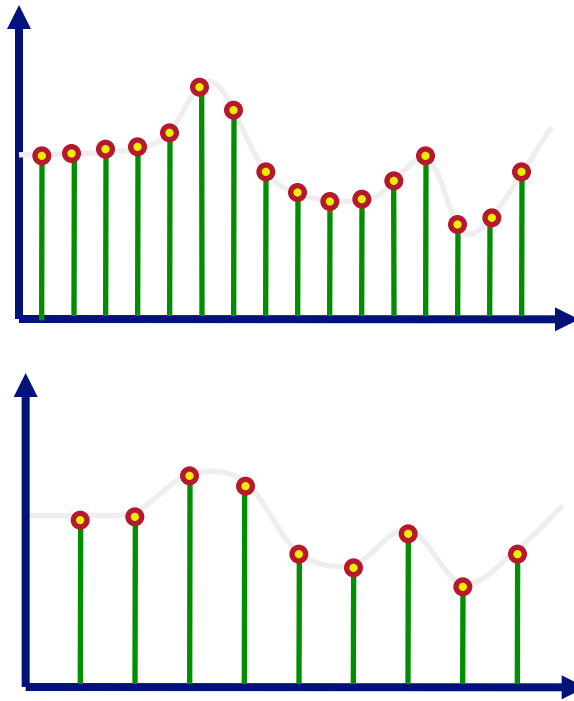


Source image

Destination image

Resampling

Simple example: scaling resolution = resampling



Resampling

Resampling



Example: scaling resolution = resampling



Original



Scaled

Resampling



- Naïve resampling can cause visual artifacts



Original



Scaled



What is the Problem?



Aliasing

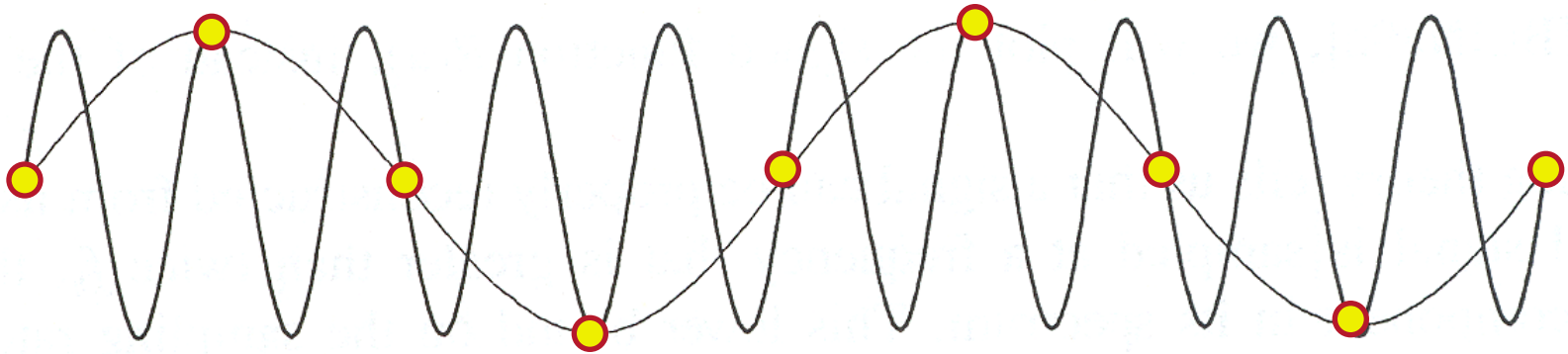


Figure 14.17 FvDFH

Aliasing



Artifacts due to under-sampling

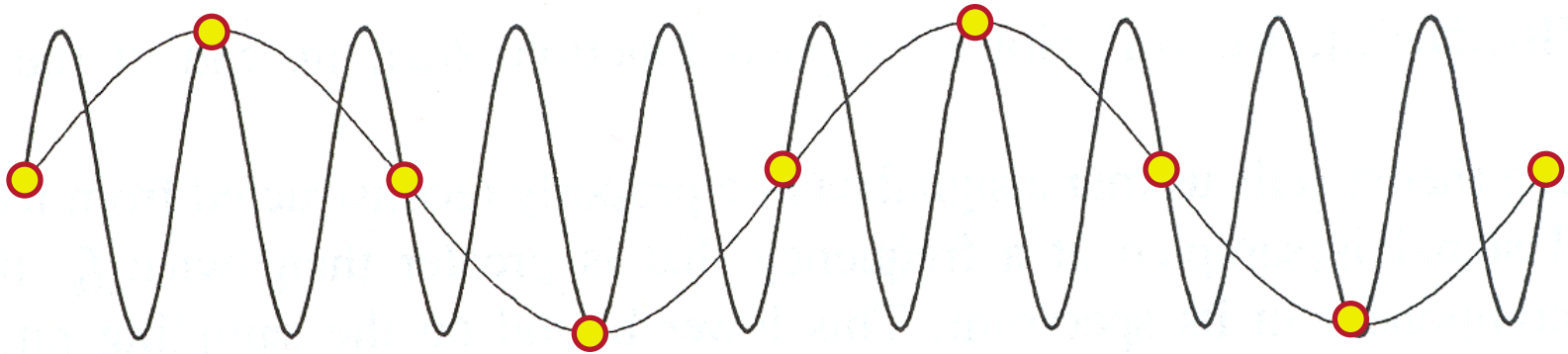
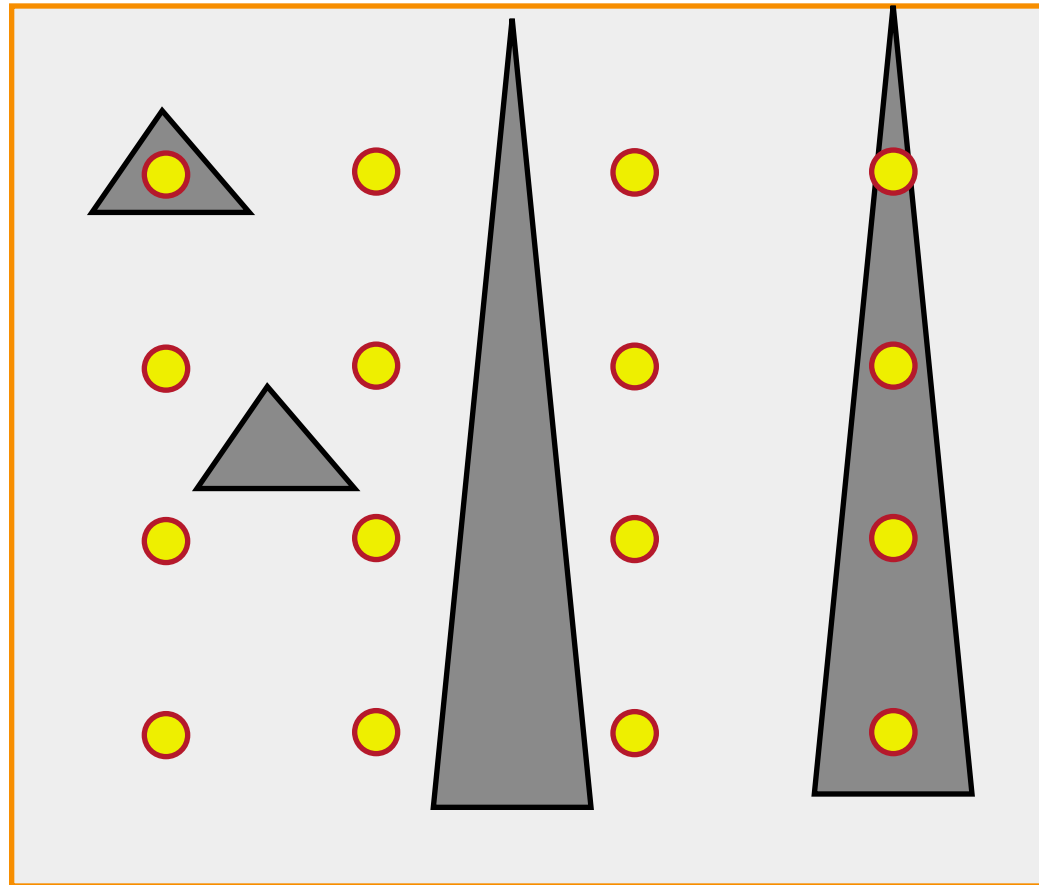


Figure 14.17 FvDFH

Spatial Aliasing



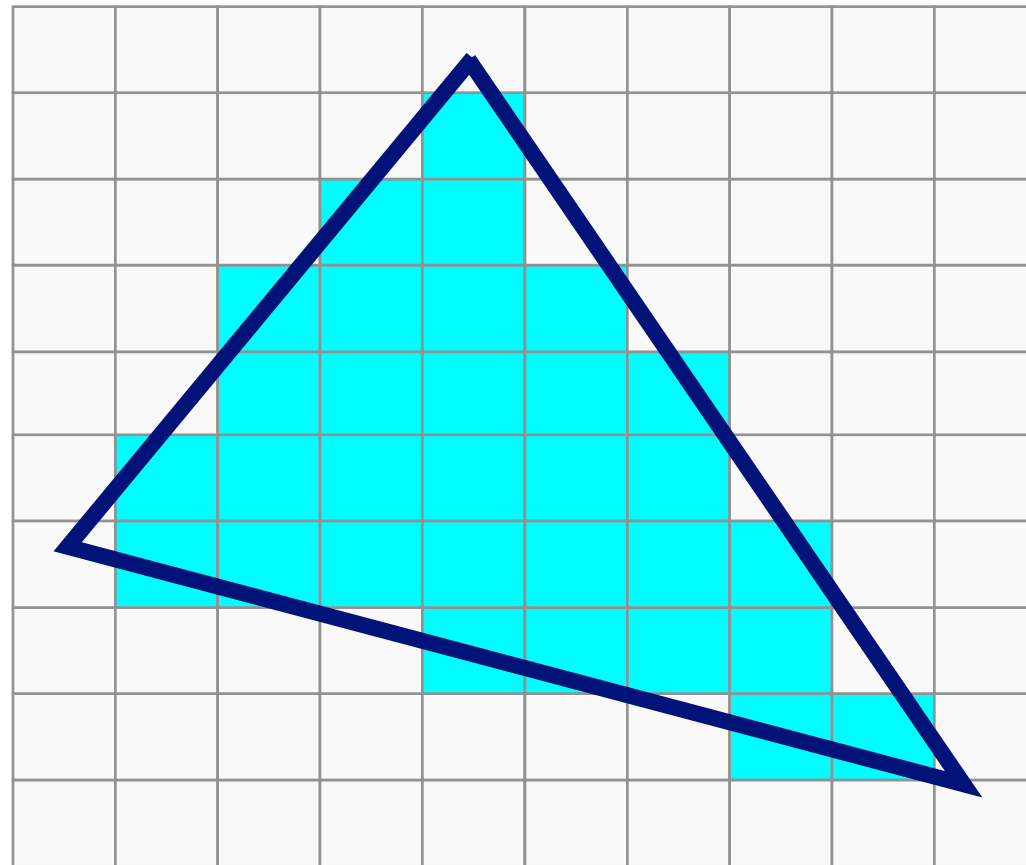
Artifacts due to under-sampling in x,y



Spatial Aliasing



Artifacts due to under-sampling in x,y



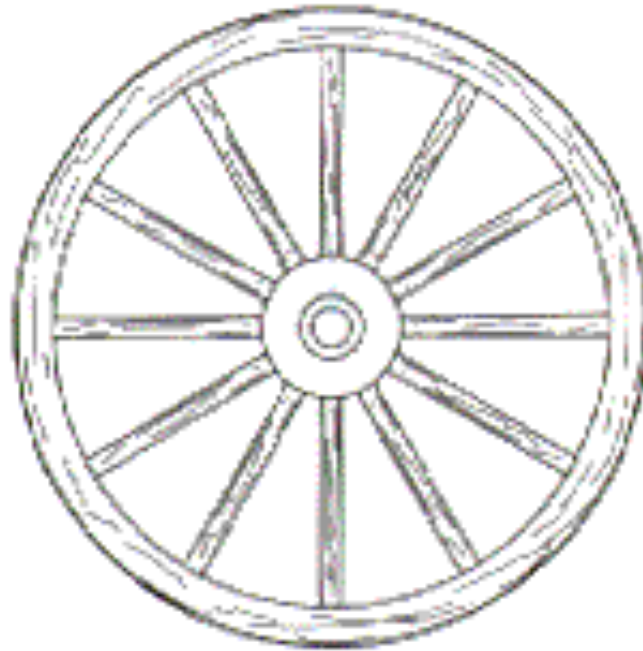
“Jaggies”

Temporal Aliasing



Artifacts due to under-sampling in time

- Strobings
- Flickering

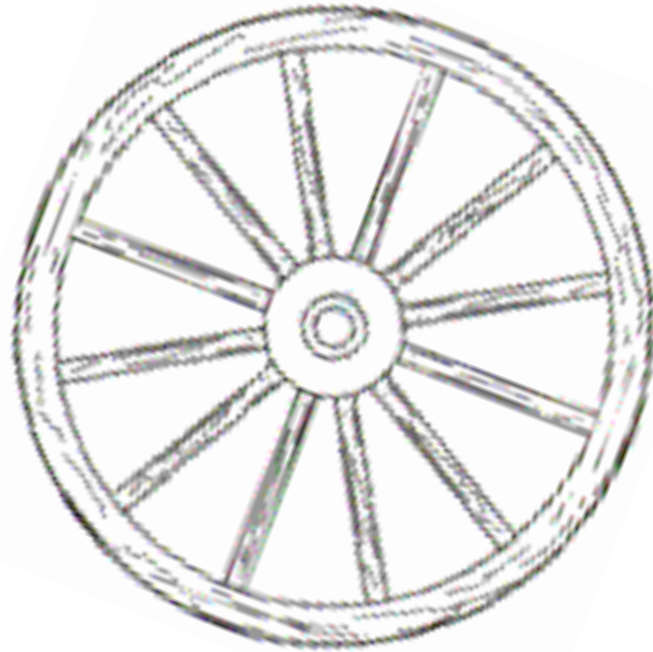


Temporal Aliasing



Artifacts due to under-sampling in time

- Strobings
- Flickering

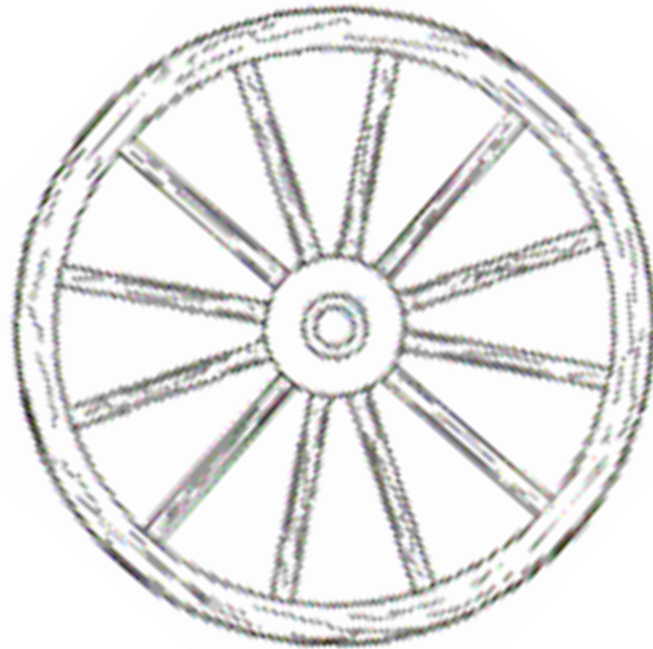


Temporal Aliasing



Artifacts due to under-sampling in time

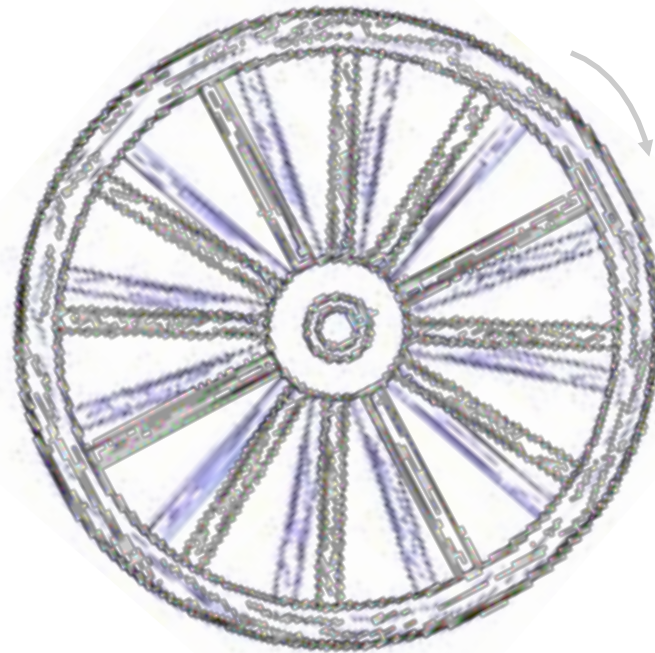
- Strobings
- Flickering



Temporal Aliasing

Artifacts due to under-sampling in time

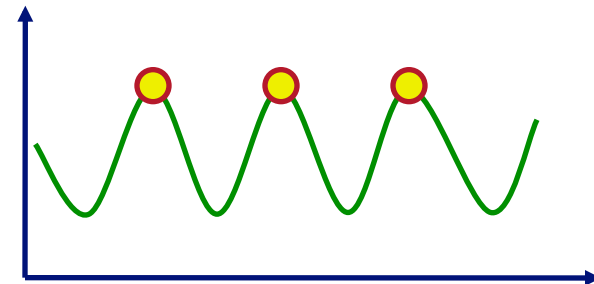
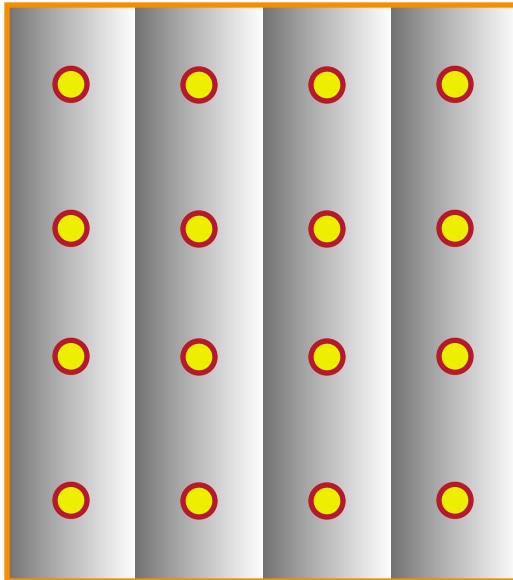
- Strobings
- Flickering



Aliasing



When we under-sample an image, we can create visual artifacts where high frequencies masquerade as low ones

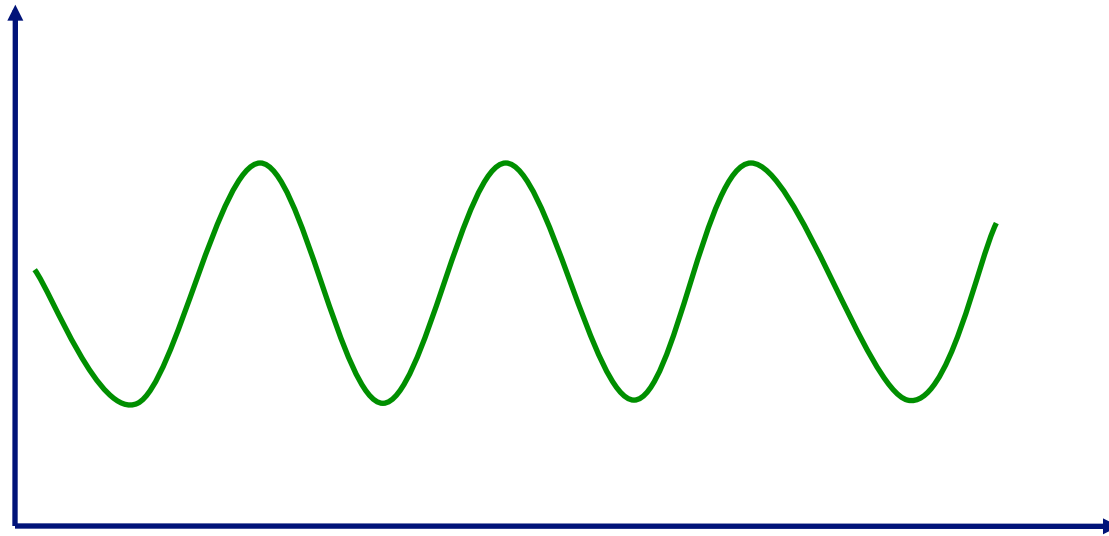


Sampling Theory



How many samples are enough to avoid aliasing?

- 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?

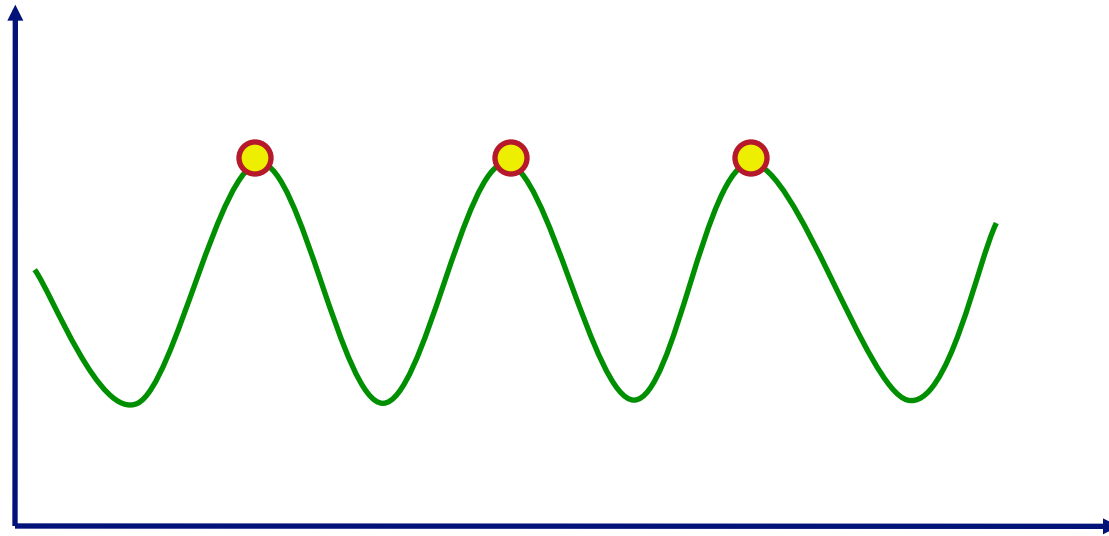


Sampling Theory



How many samples are enough to avoid aliasing?

- 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?

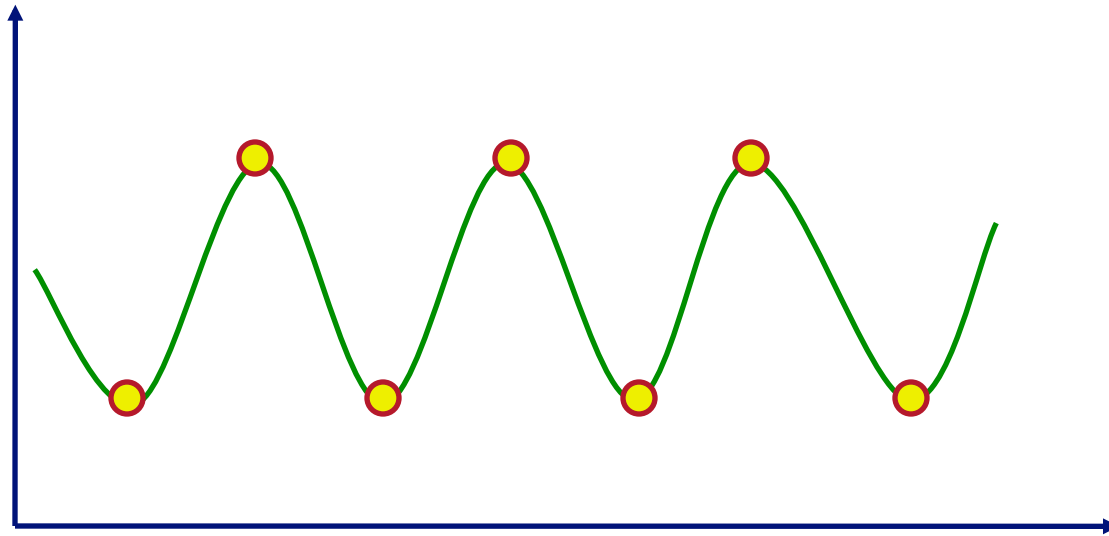


Sampling Theory



How many samples are enough to avoid aliasing?

- 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?

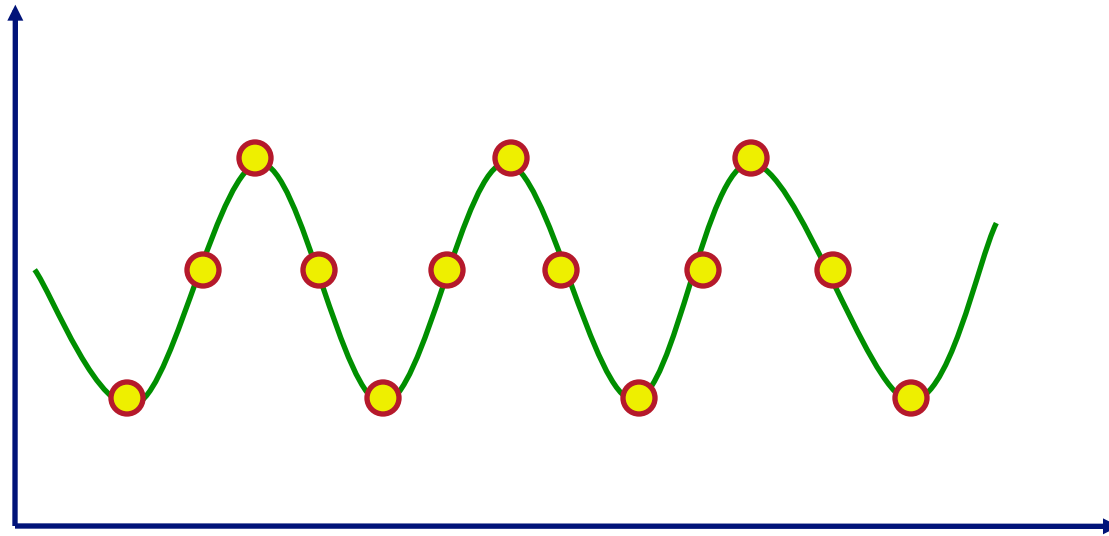


Sampling Theory



How many samples are enough to avoid aliasing?

- 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?

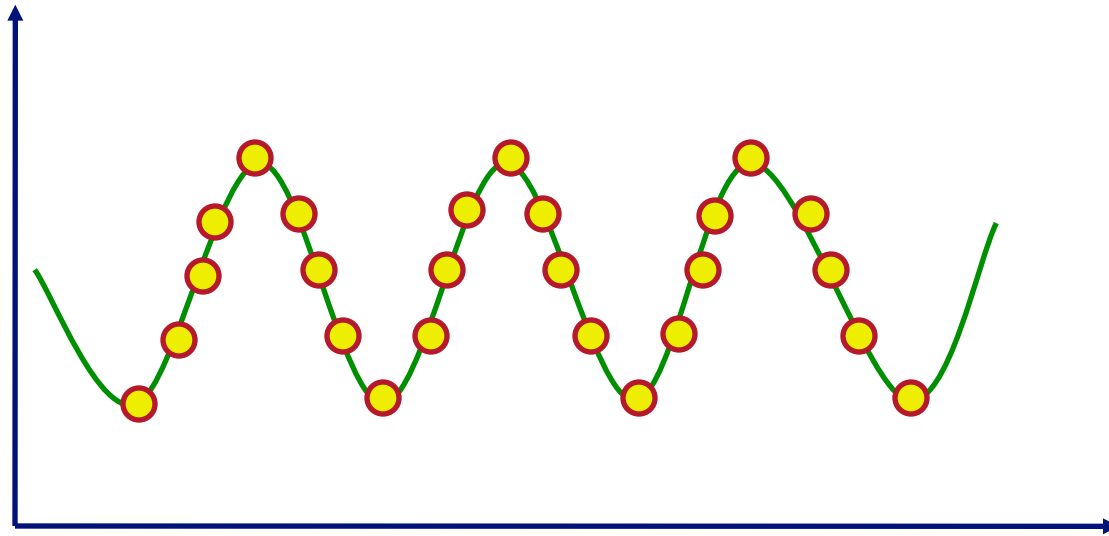


Sampling Theory



How many samples are enough to avoid aliasing?

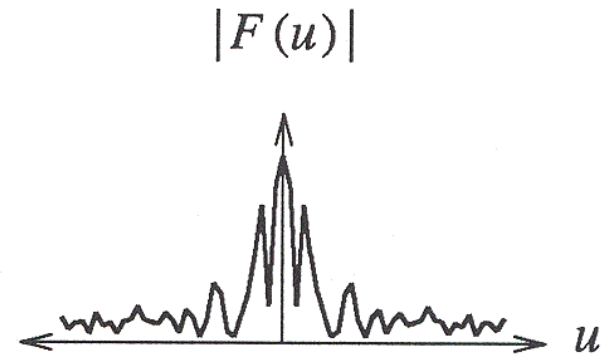
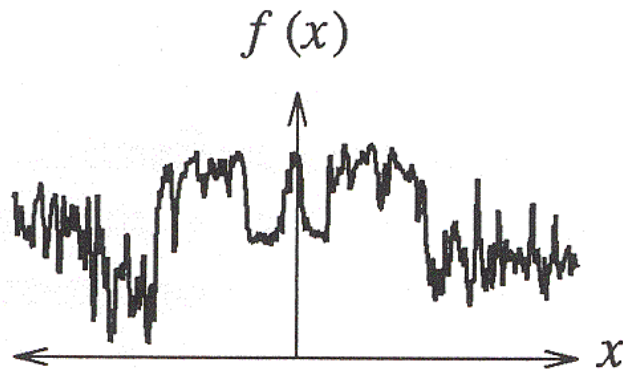
- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?



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 Transform

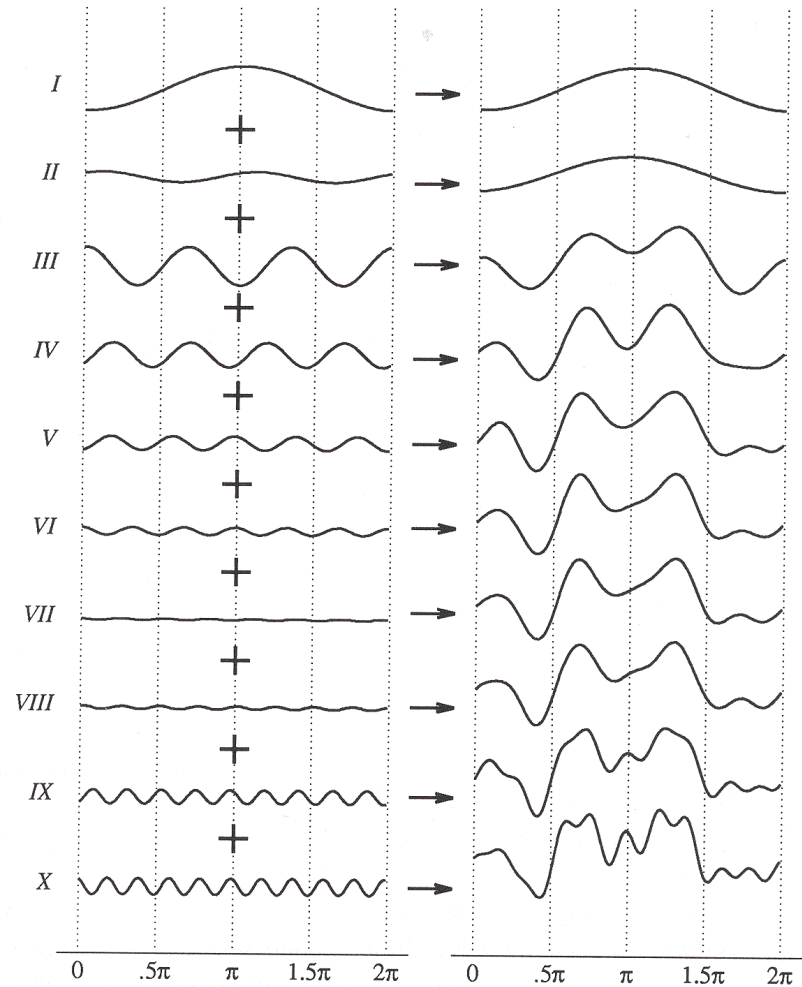
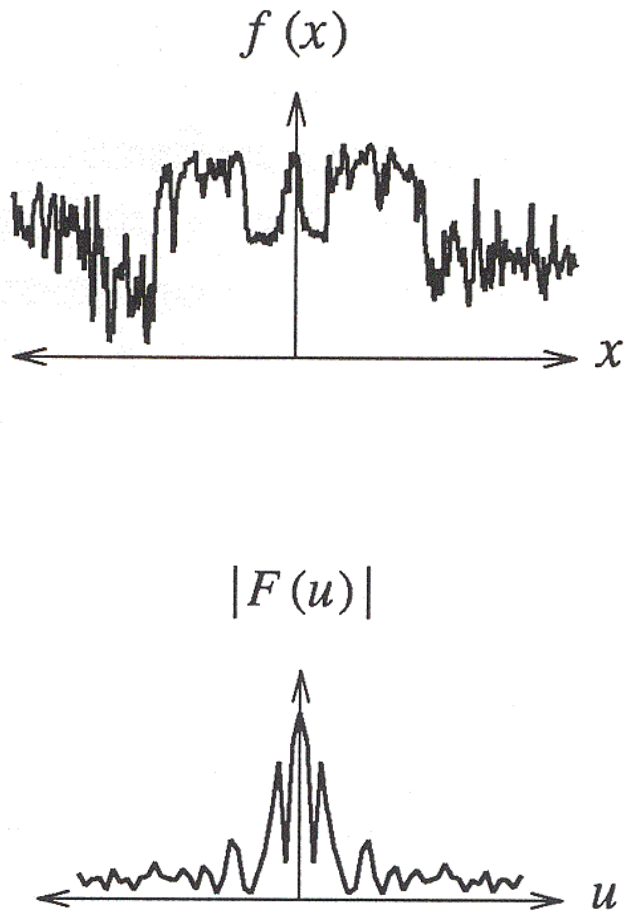


Figure 2.6 Wolberg



Fourier Transform

- Fourier transform:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xu} dx$$

- Inverse Fourier transform:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{+i2\pi ux} du$$

Sampling Theorem



- A signal can be reconstructed from its samples, iff the original signal has no content $\geq 1/2$ the sampling frequency - Shannon
- The minimum sampling rate for a “bandlimited” function is called the “Nyquist rate”

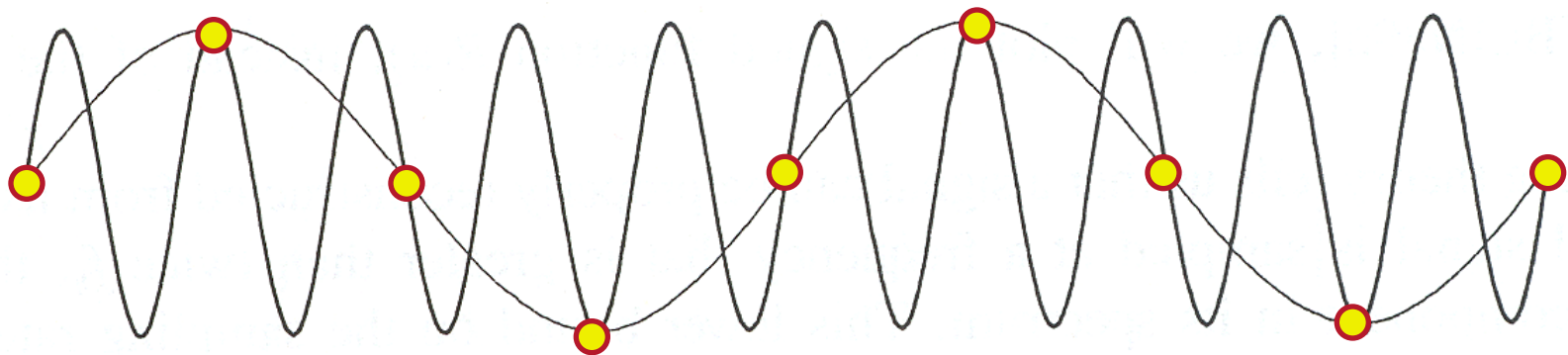
A signal is bandlimited if its highest frequency is bounded. The frequency is called the bandwidth.



Sampling Theorem

- A signal can be reconstructed from its samples, iff the original signal has no content $\geq 1/2$ the sampling frequency - Shannon

Aliasing will occur if the signal is under-sampled



Under-sampling

Figure 14.17 FvDFH

Sampling and Reconstruction

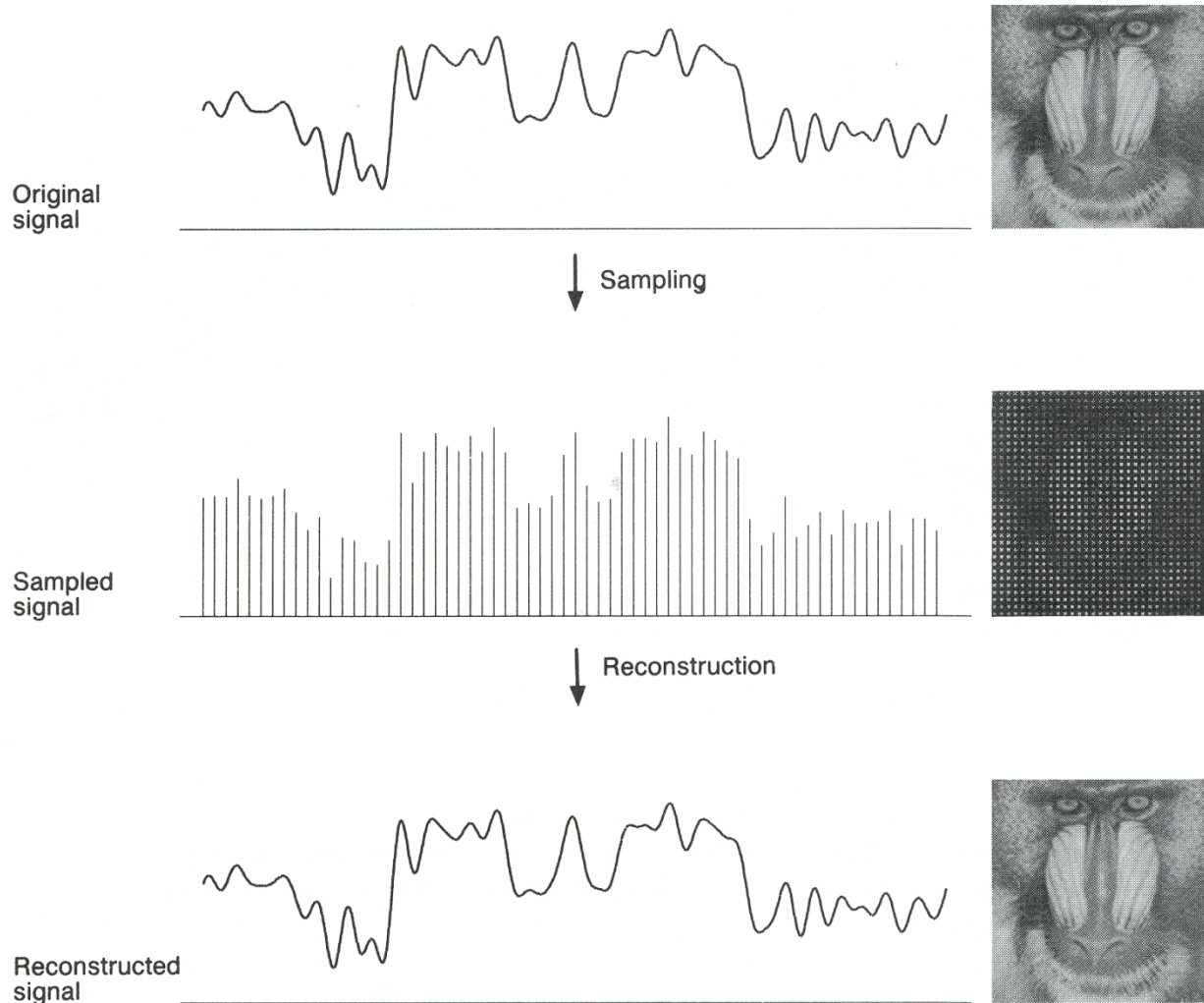
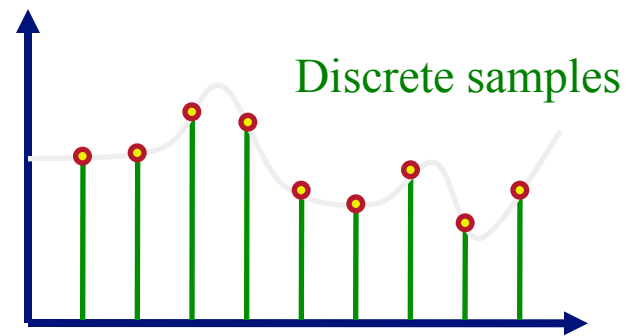
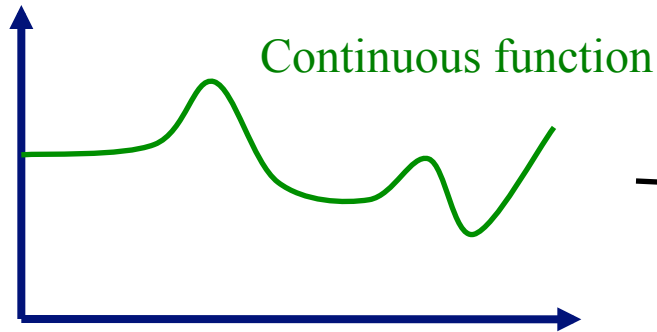


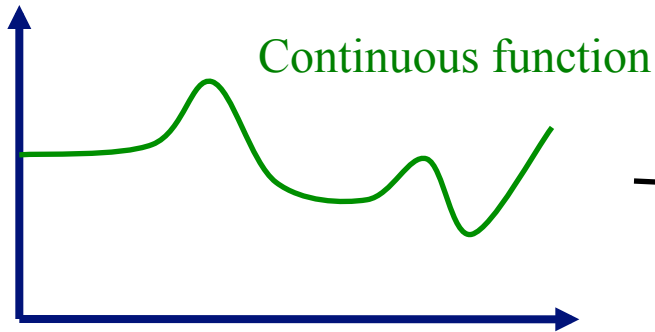
Figure 19.9 FvDFH

Sampling and Reconstruction

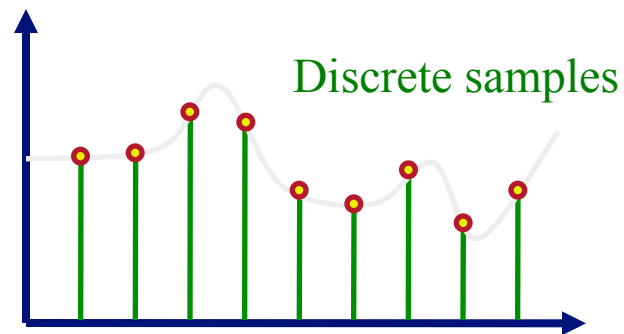


Sampling

Sampling and Reconstruction



Sampling



Reconstruction

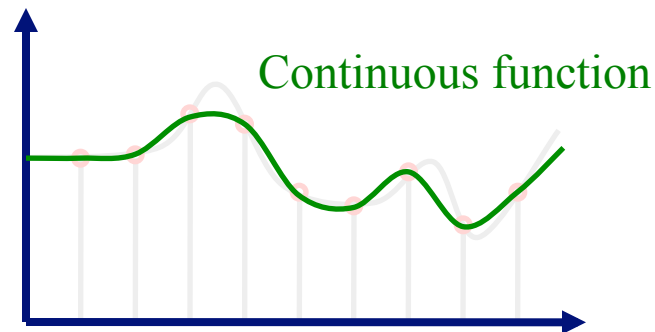
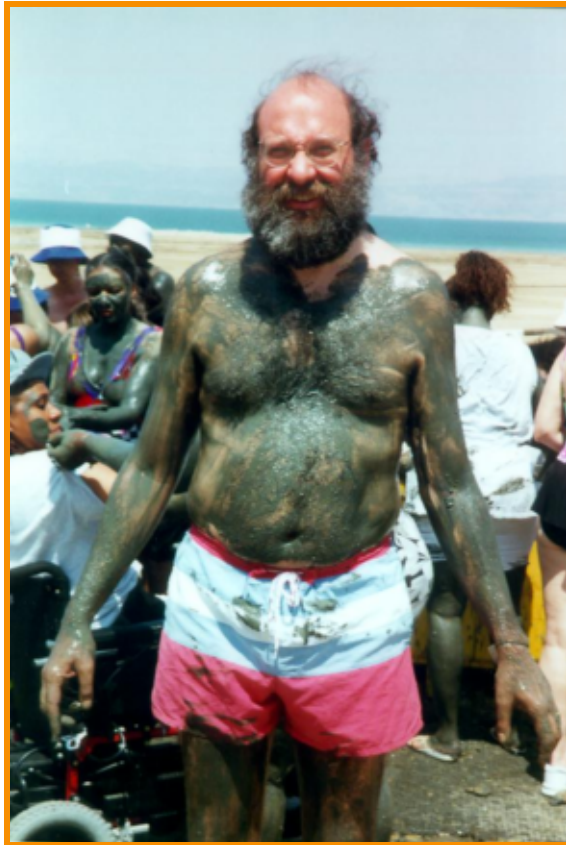


Image Processing



OK ... but how does that affect image processing?



Source image

→
Warp



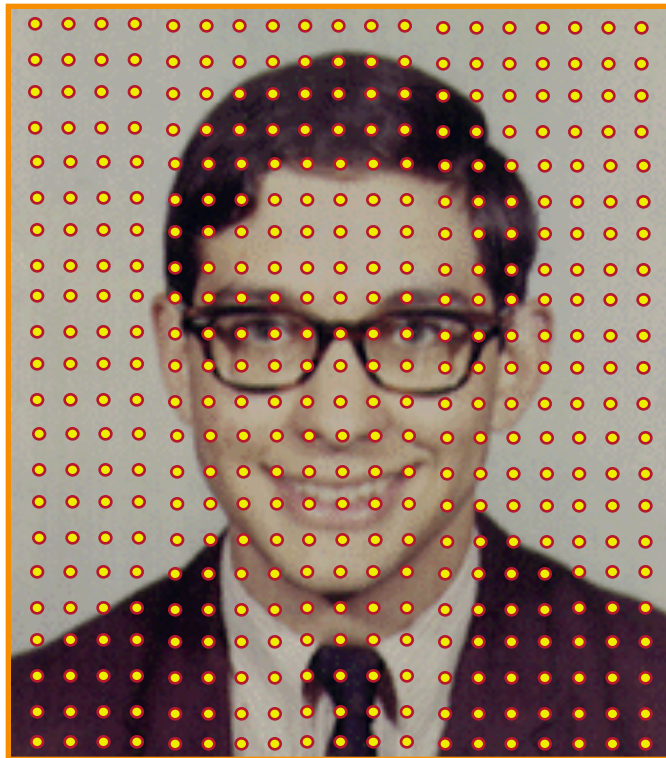
Destination image

Image Processing

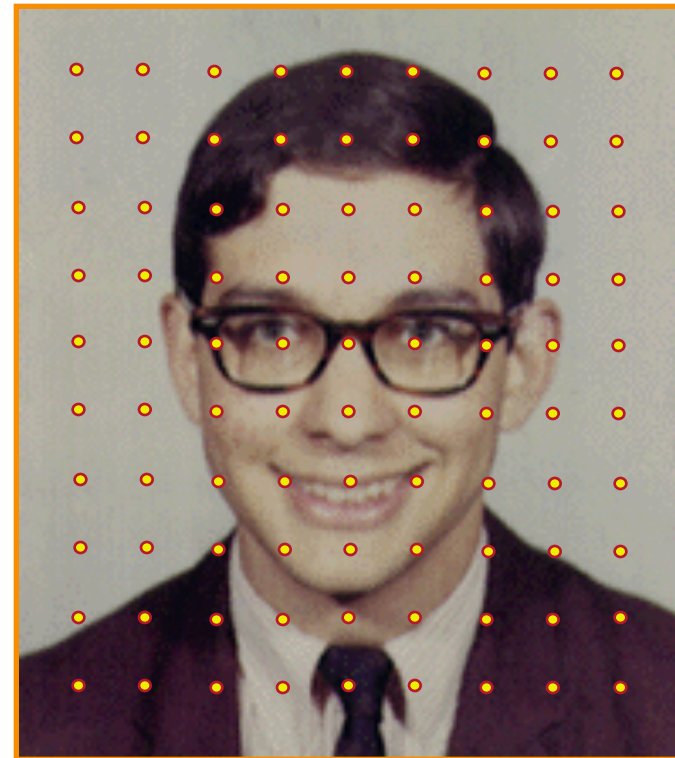


Image processing often requires resampling

- Must band-limit before resampling to avoid aliasing

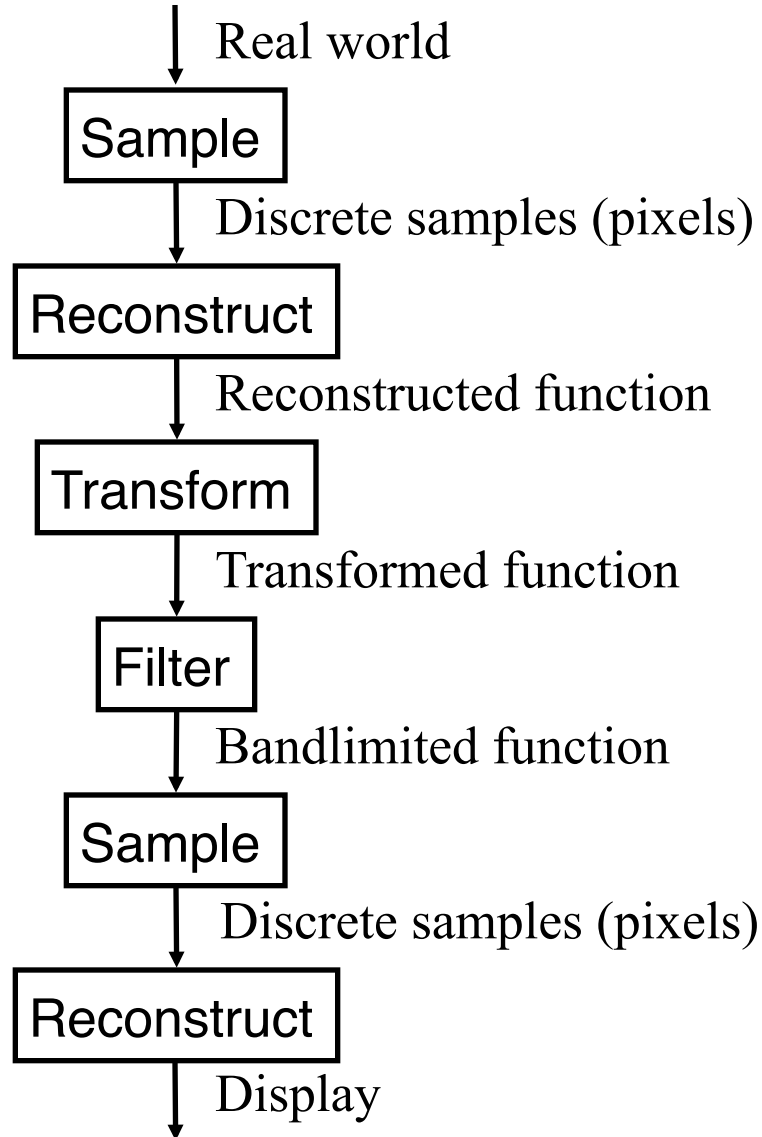


Original image

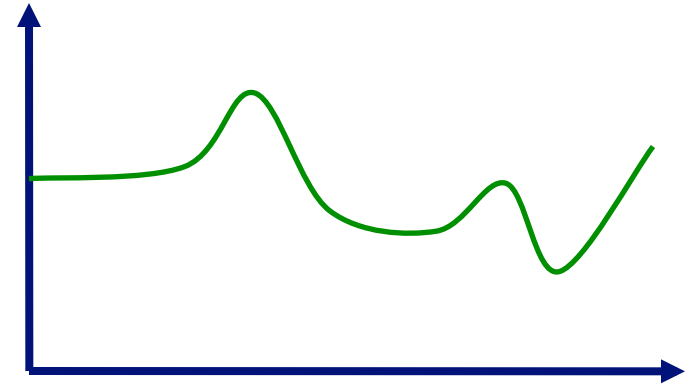
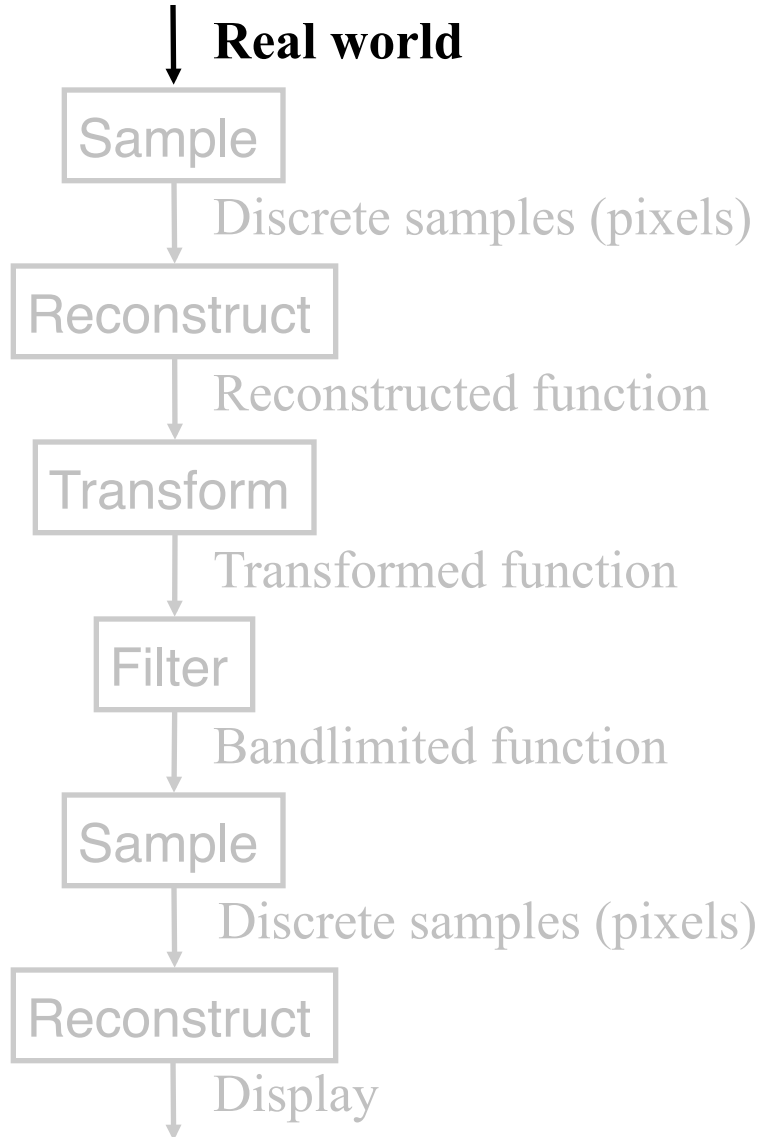


1/4 resolution

Ideal Image Processing

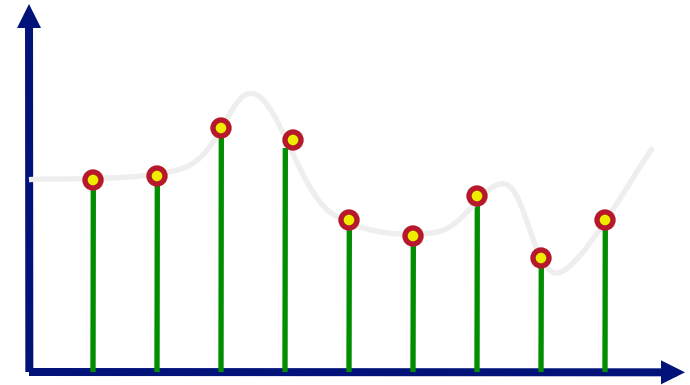
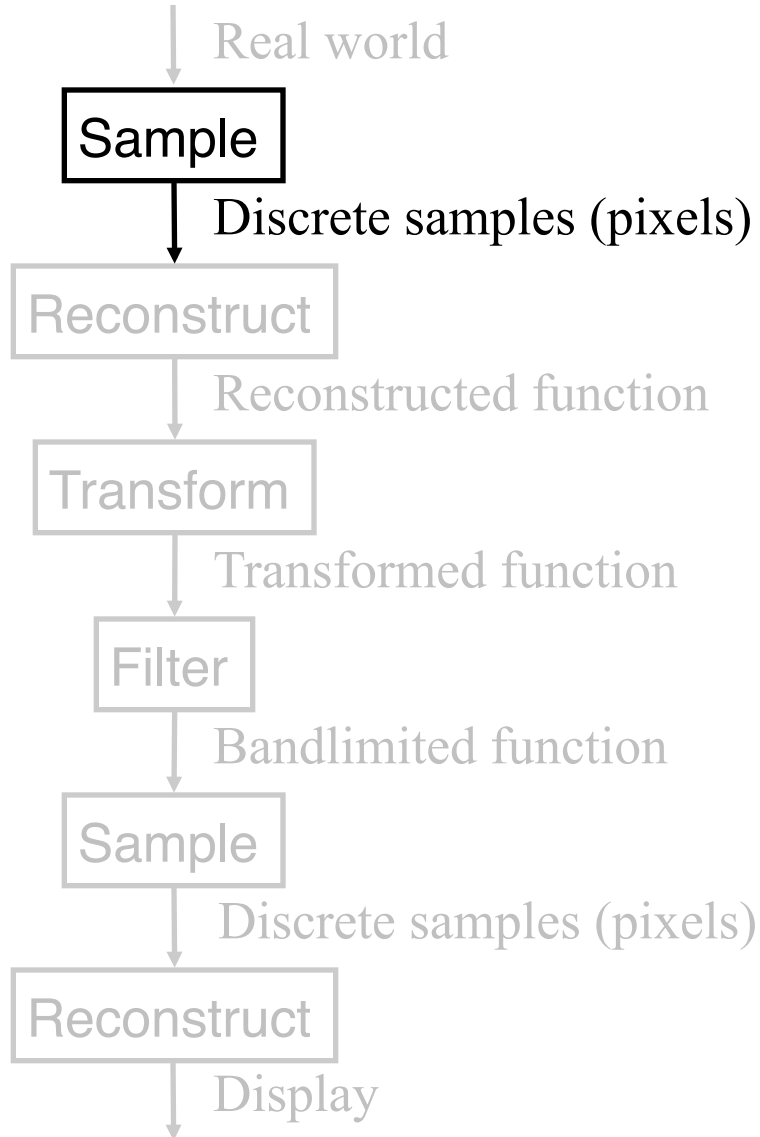


Ideal Image Processing



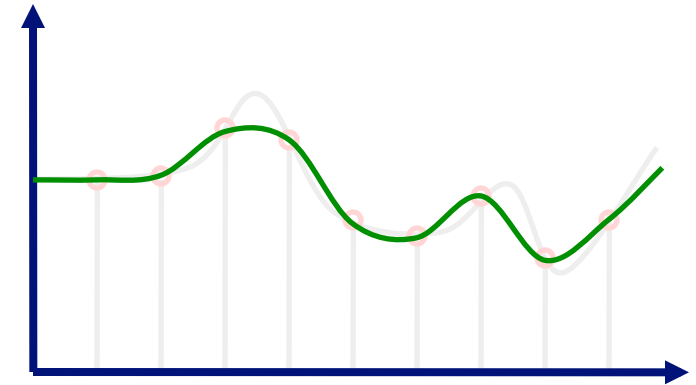
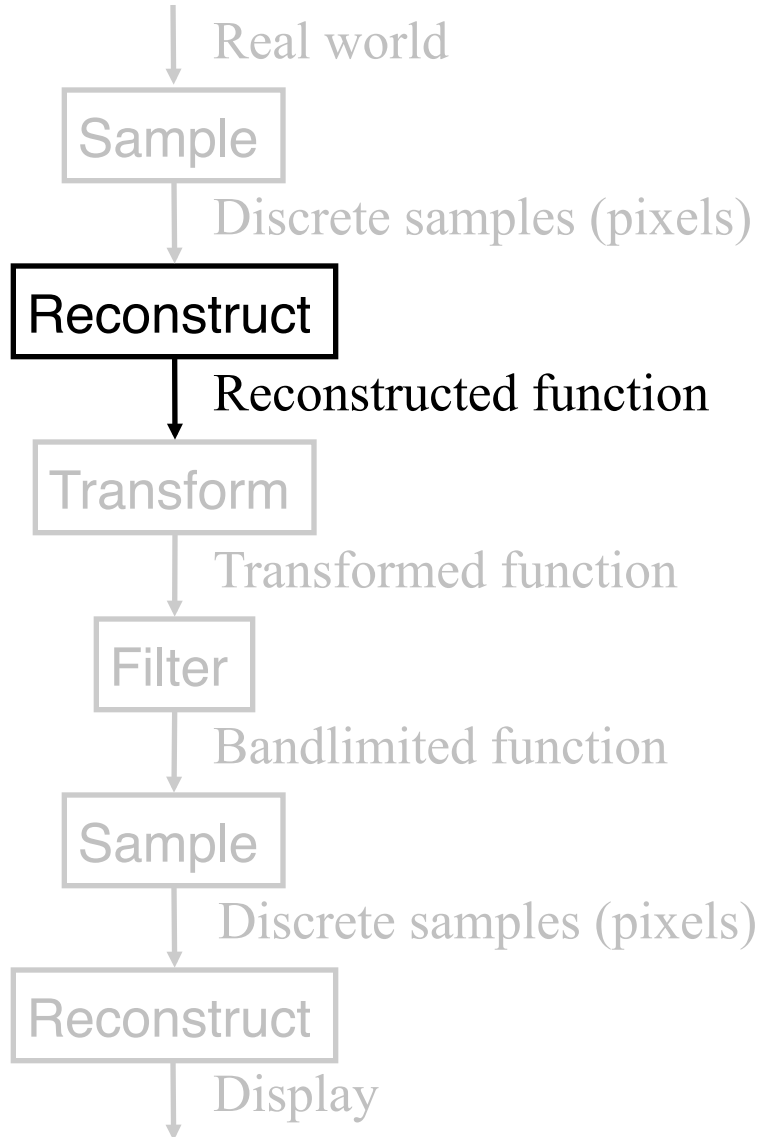
Continuous Function

Ideal Image Processing



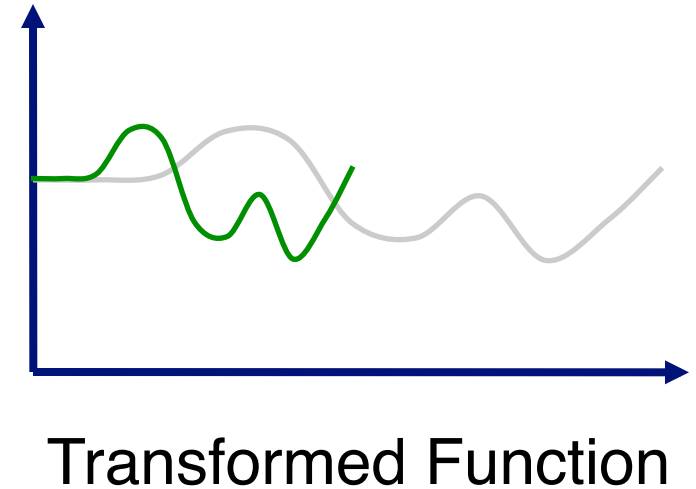
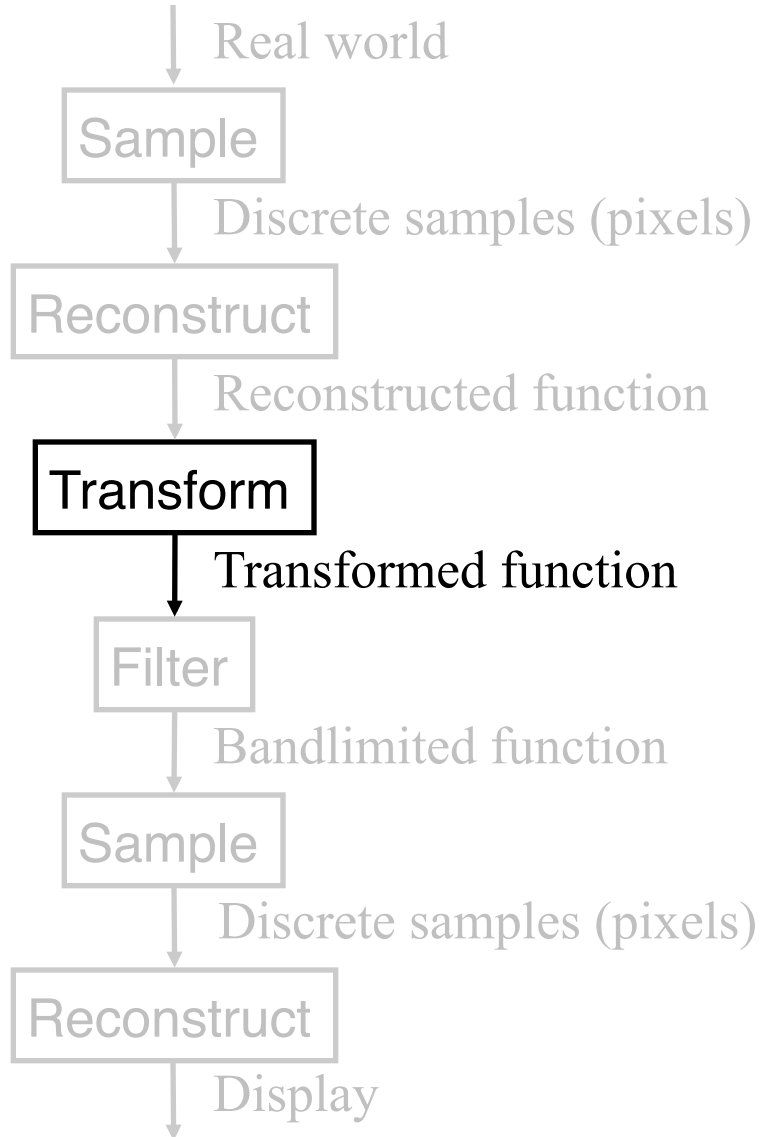
Discrete Samples

Ideal Image Processing

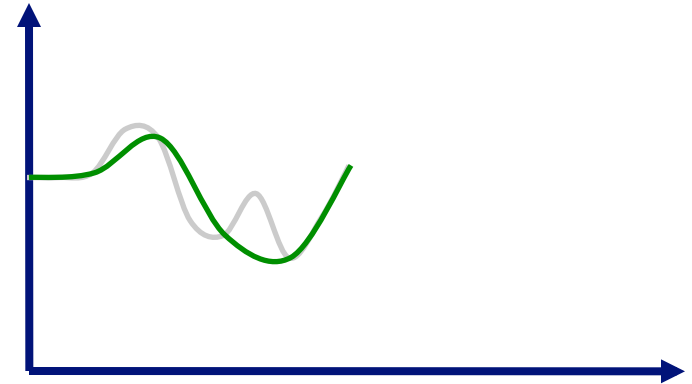
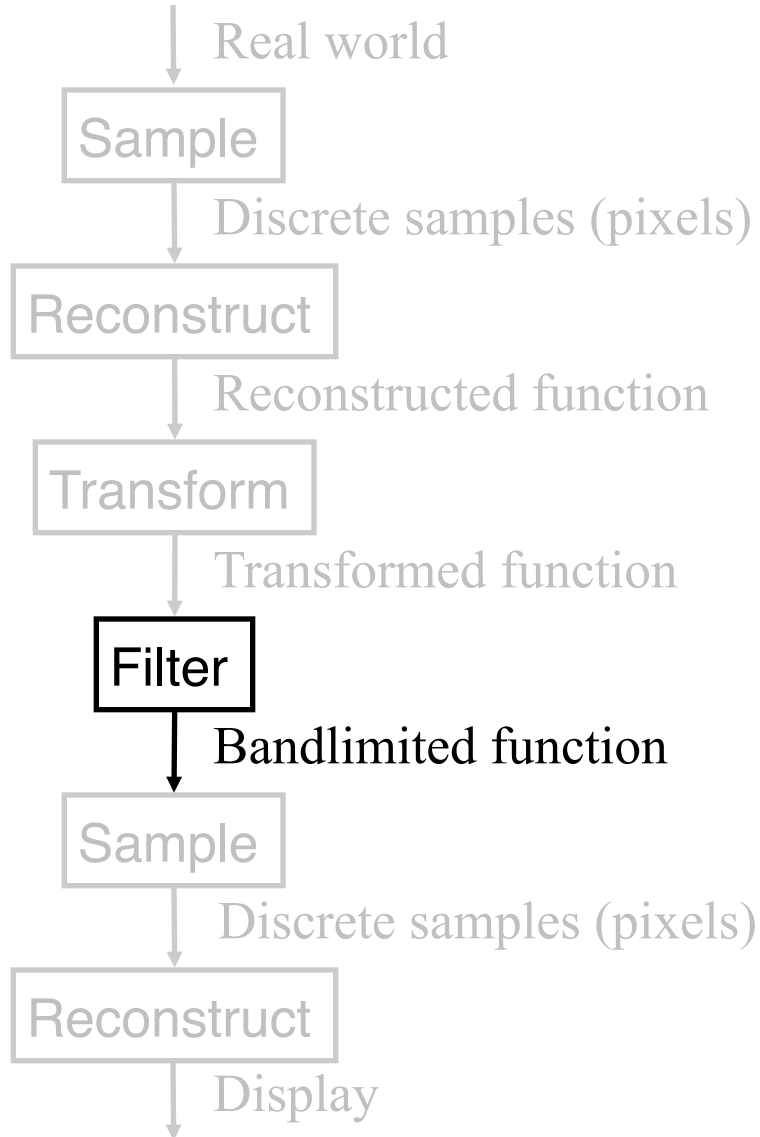


Reconstructed Function

Ideal Image Processing

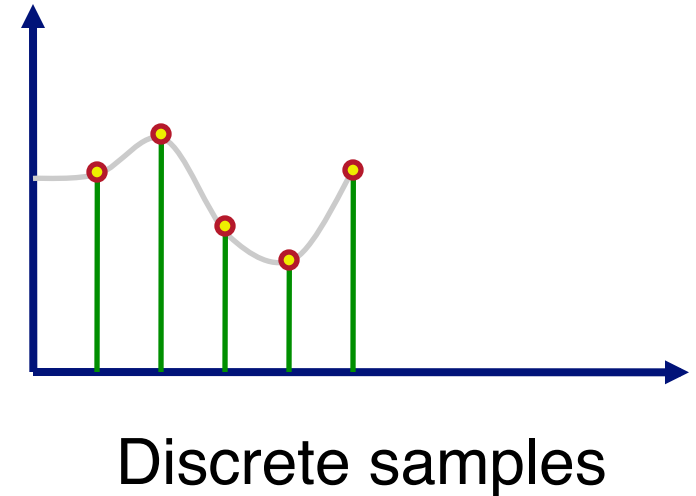
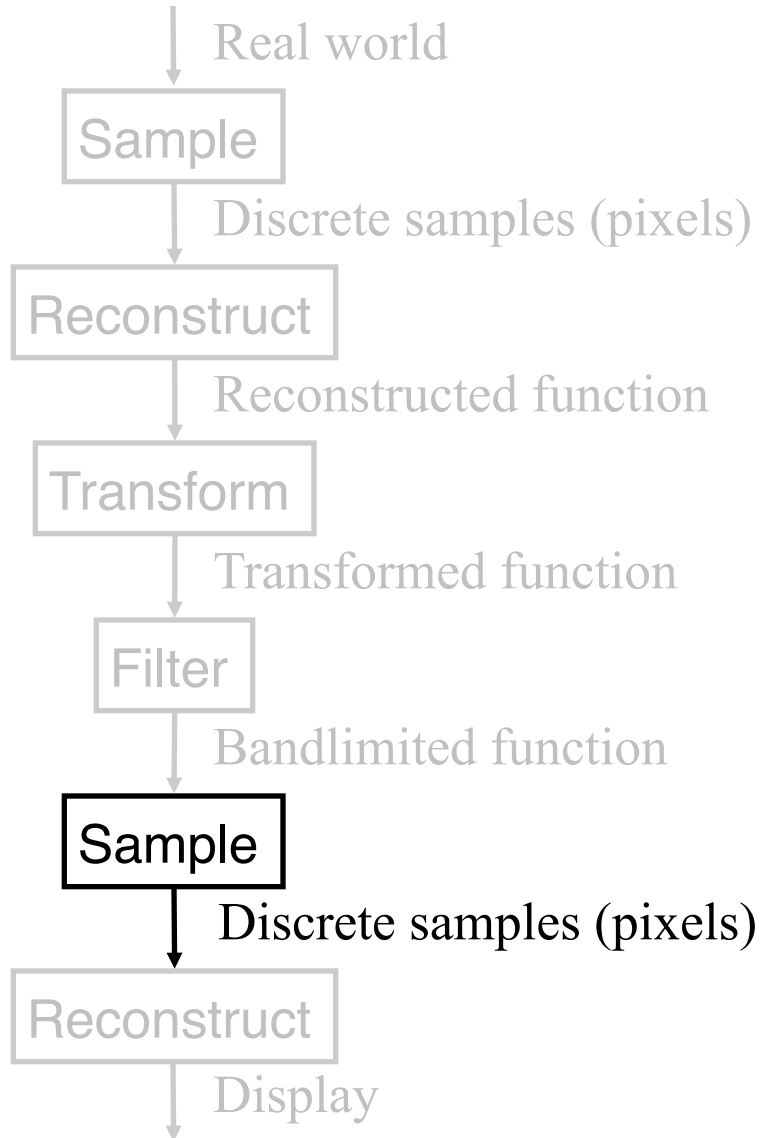


Ideal Image Processing

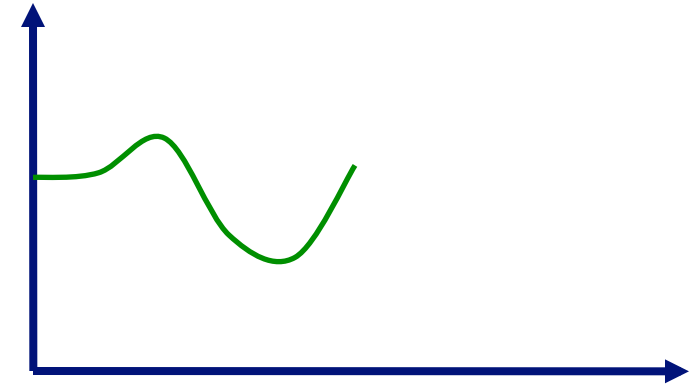
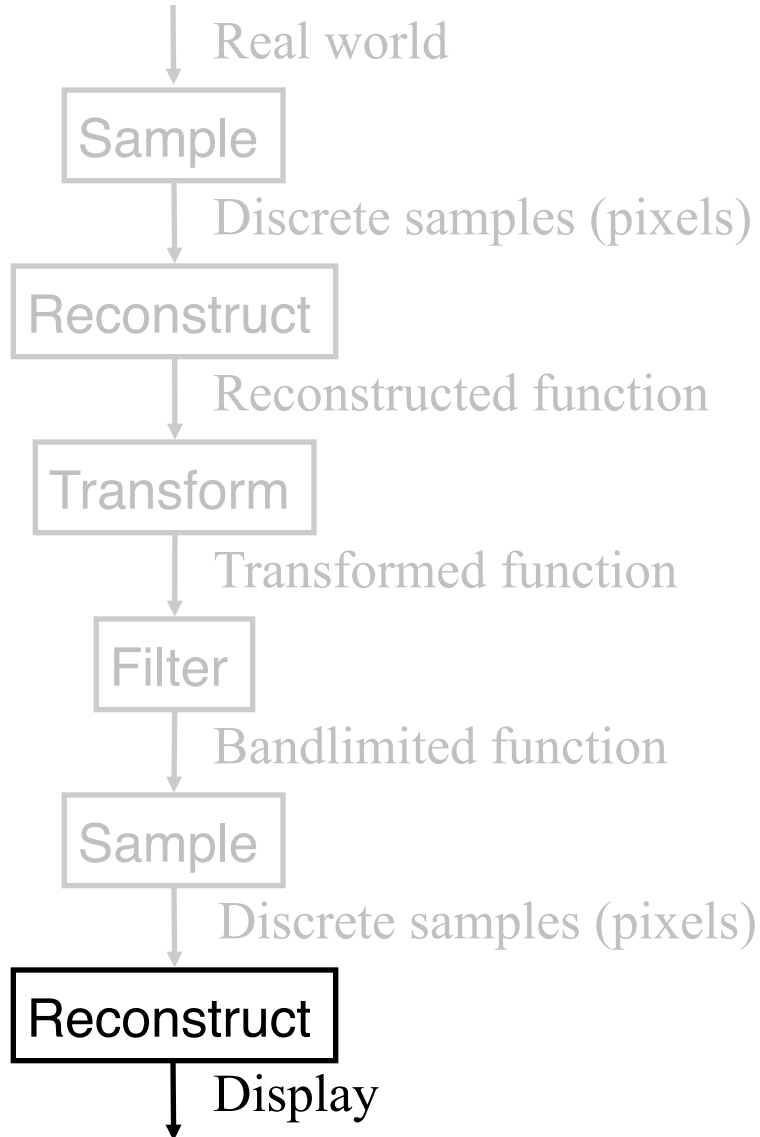


Bandlimited Function

Ideal Image Processing



Ideal Image Processing

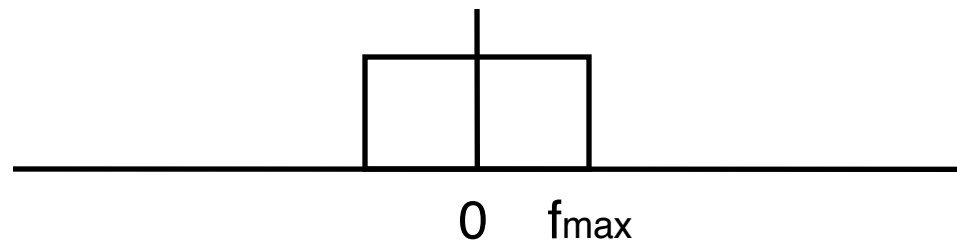


Display

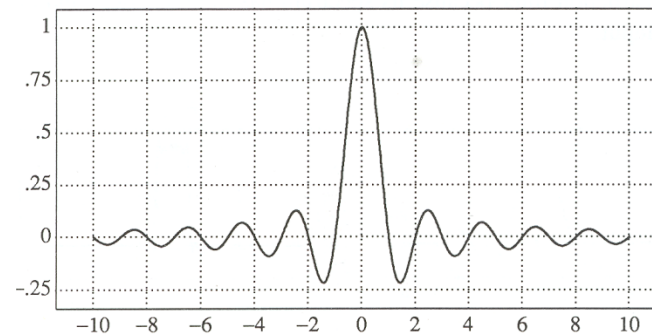


Ideal Bandlimiting Filter

- Frequency domain



- Spatial domain



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

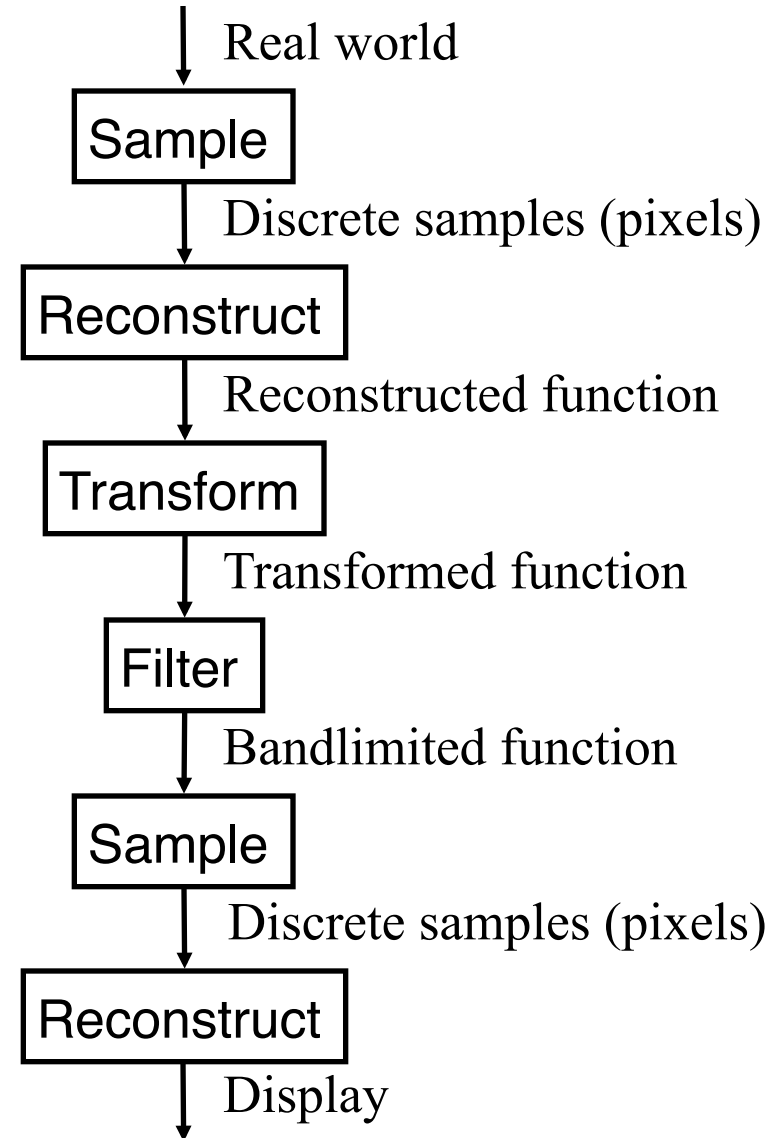
Figure 4.5 Wolberg

Practical Image Processing



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

Low-Pass Filter

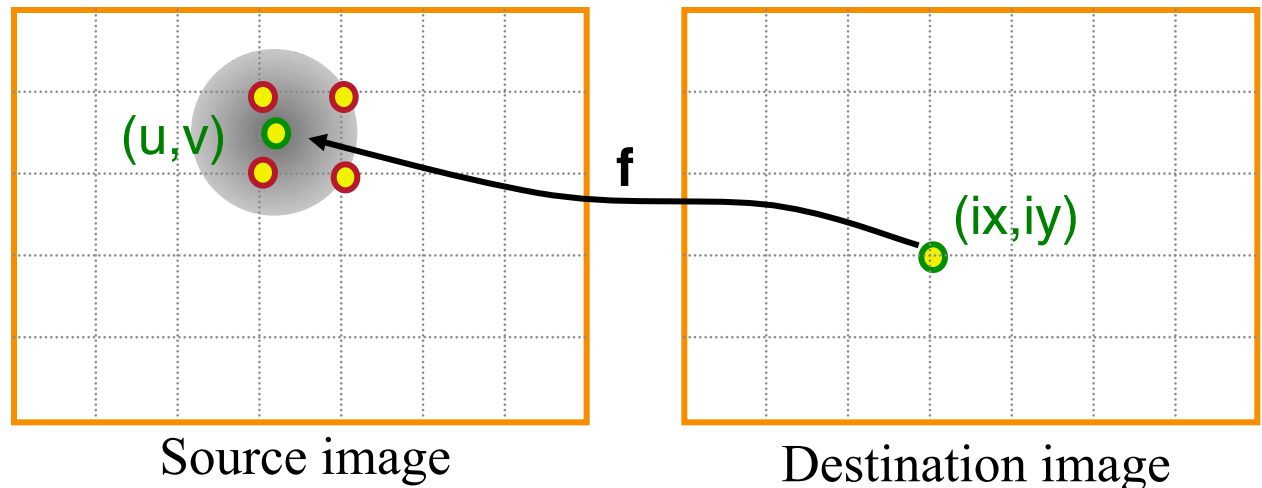


Practical Image Processing



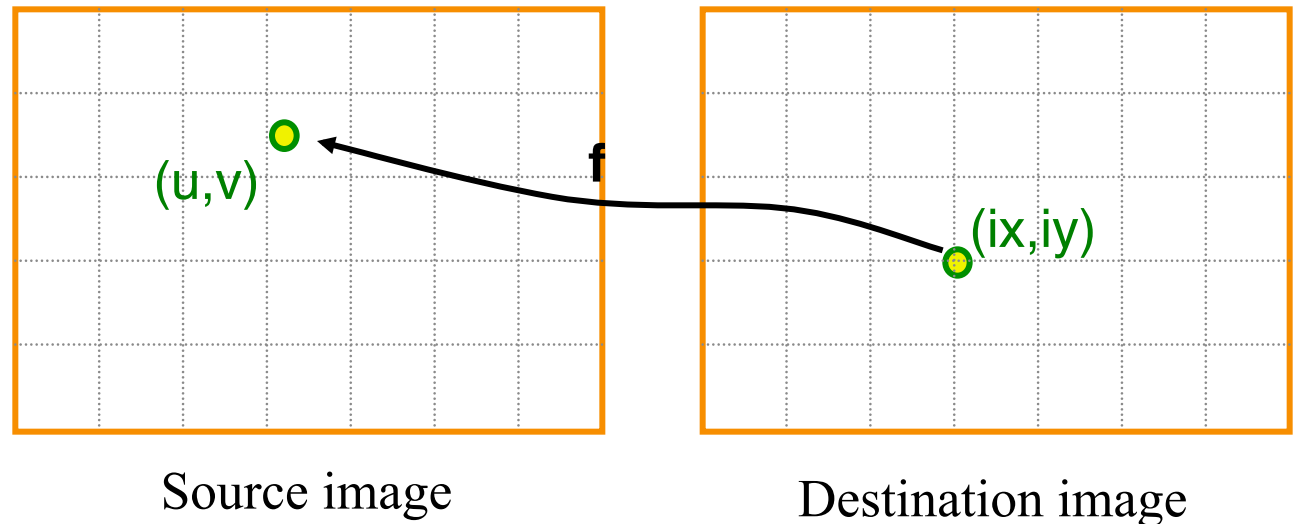
- Reverse mapping:

```
Warp(src, dst) {  
  for (int ix = 0; ix < xmax; ix++) {  
    for (int iy = 0; iy < ymax; iy++) {  
      float w  $\approx$  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, k, w);  
    }  
  }  
}
```



Resampling

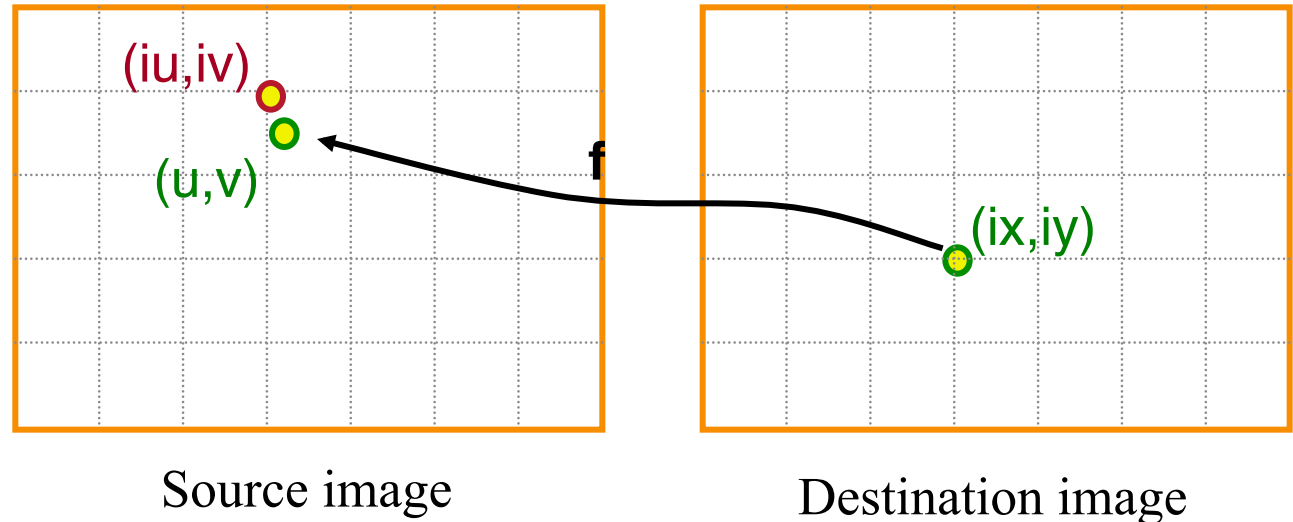
- Compute value of 2D function at arbitrary location from given set of samples



Point Sampling

- Possible (poor) resampling implementation:

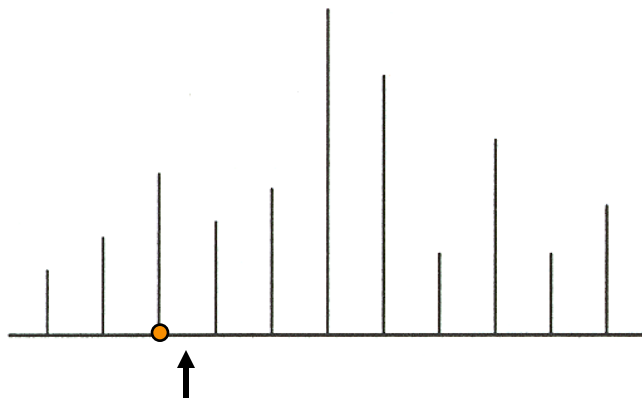
```
float Resample(src, u, v, k, w) {  
    int iu = round(u);  
    int iv = round(v);  
    return src(iu, iv);  
}
```



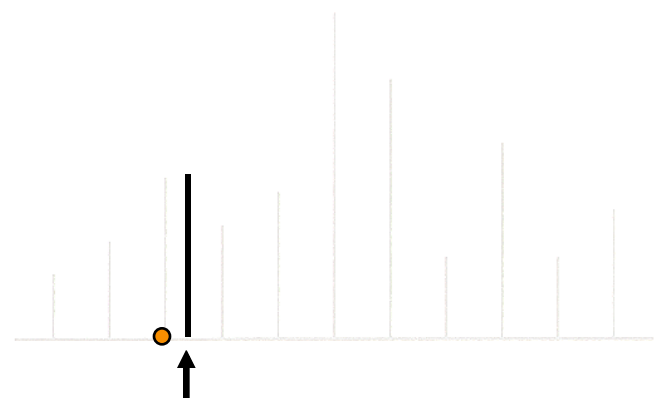


Point Sampling

- Use nearest sample



Input



Output

Point Sampling



Point Sampled: Aliasing!

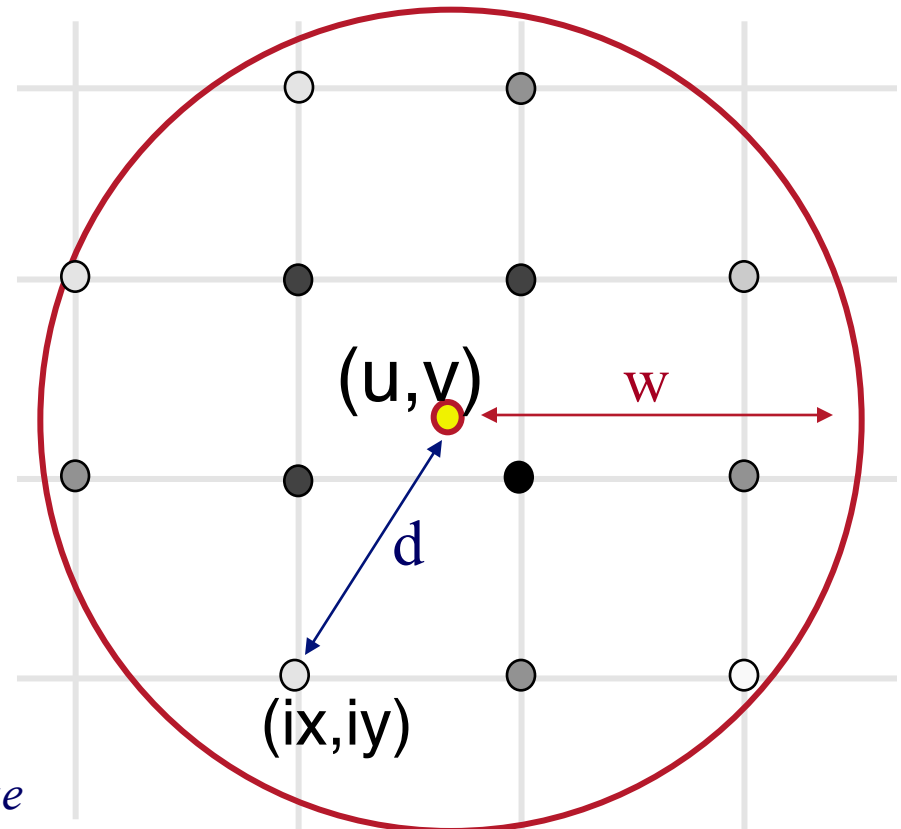


Correctly Bandlimited

Resampling with Low-Pass Filter



- Output is weighted average of input samples, where weights are normalized values of filter (k)



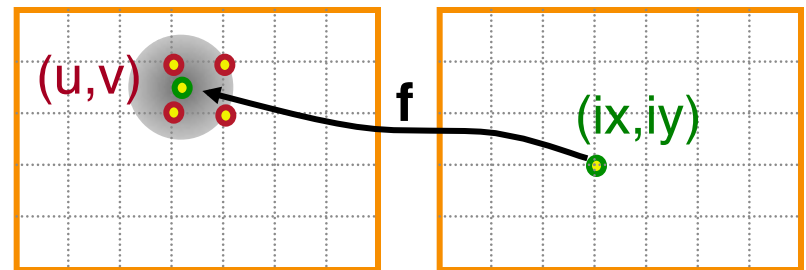
$k(ix, iy)$ represented by gray value



Resampling with Low-Pass Filter

- Possible implementation:

```
float Resample(src, u, v, k, w)
{
    float dst = 0;
    float ksum = 0;
    int ulo = u - w; etc.
    for (int iu = ulo; iu < uhi; iu++) {
        for (int iv = vlo; iv < vhi; iv++) {
            dst += k(u,v,iu,iv,w) * src(iu,iv)
            ksum += k(u,v,iu,iv,w);
        }
    }
    return dst / ksum;
}
```



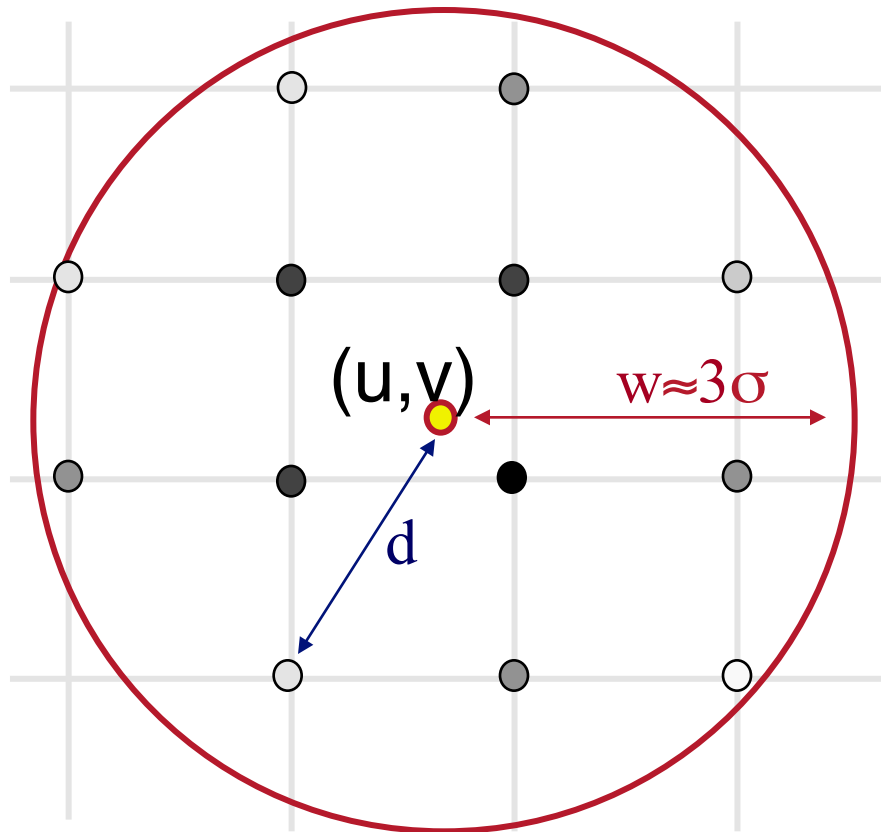
Source image

Destination image

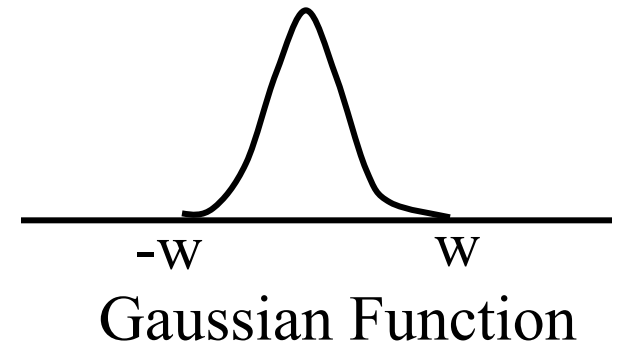
Resampling with Gaussian Filter



- Kernel is Gaussian function



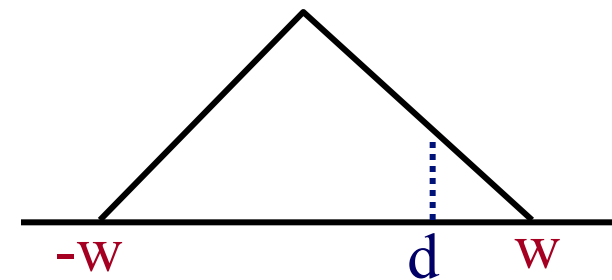
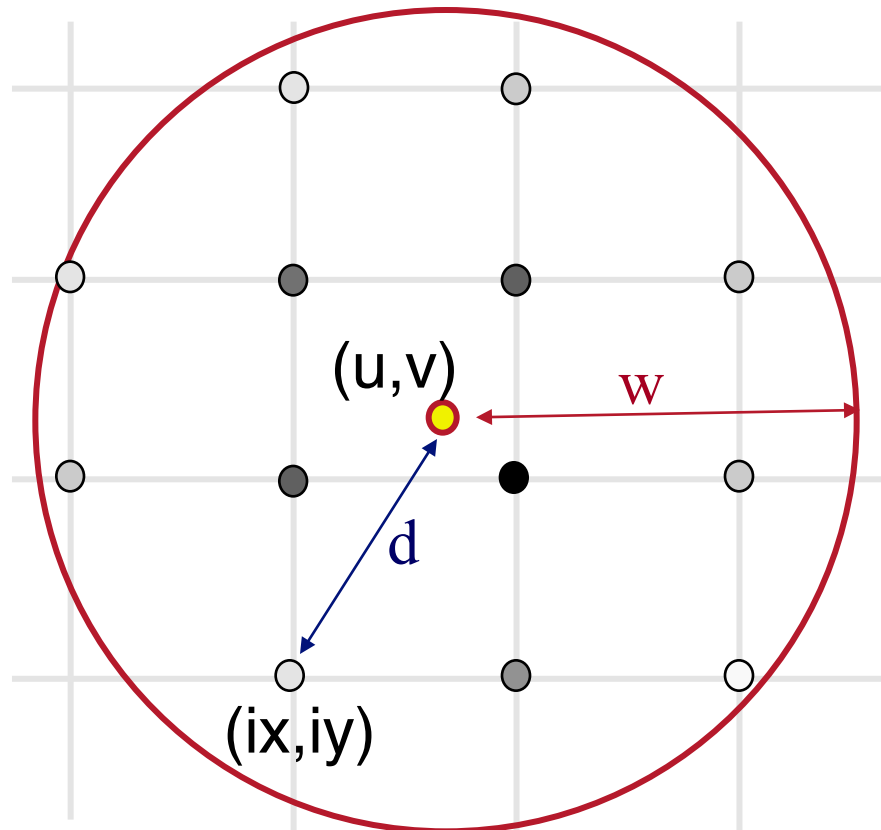
$$G(d, \sigma) = e^{-d^2 / (2\sigma^2)}$$



- Drops off quickly, but never gets to exactly 0
- In practice: compute out to $w \sim 2.5\sigma$ or 3σ

Resampling with Triangle Filter

- For isotropic Triangle filter, $k(ix, iy)$ is function of d and w



Triangle filter

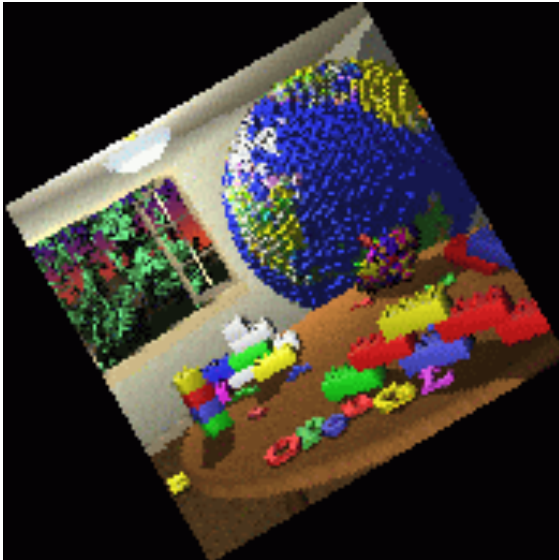
$$k(i,j) = \max(1 - d/w, 0)$$

Filter Width = 2

Sampling Method Comparison



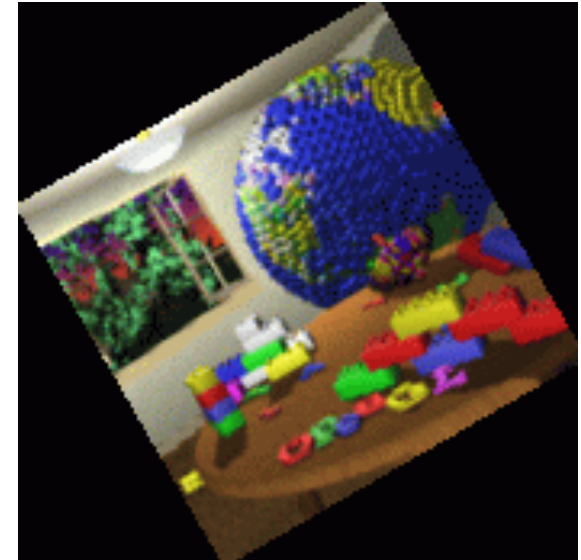
- Trade-offs
 - Aliasing versus blurring
 - Computation speed



Point



Triangle

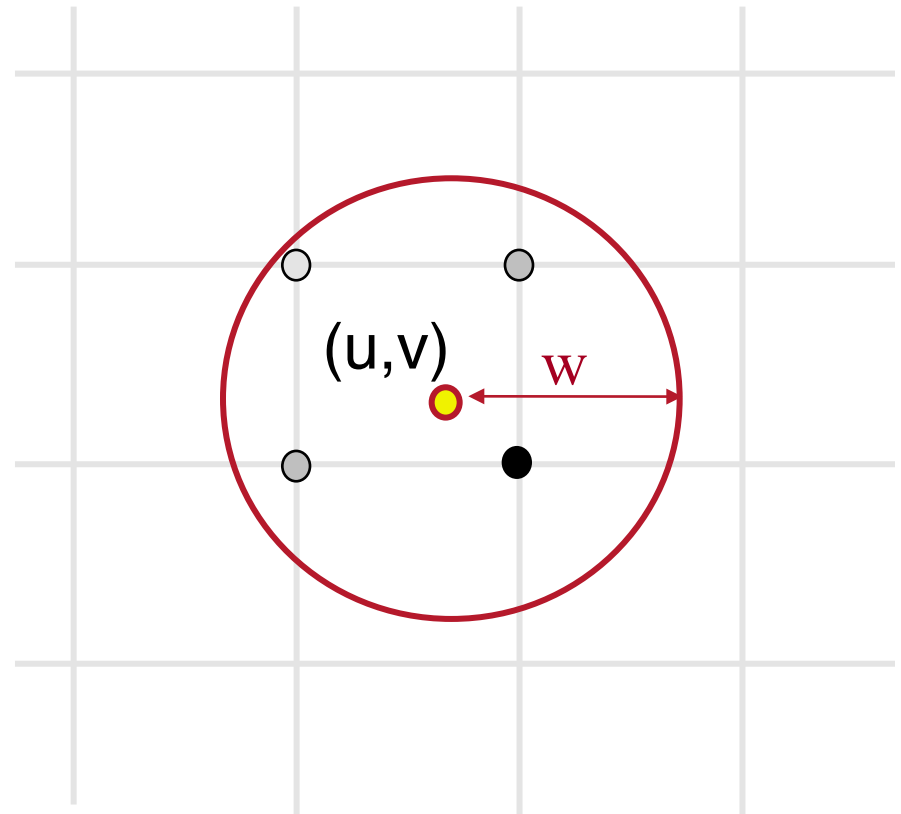


Gaussian

Resampling Details

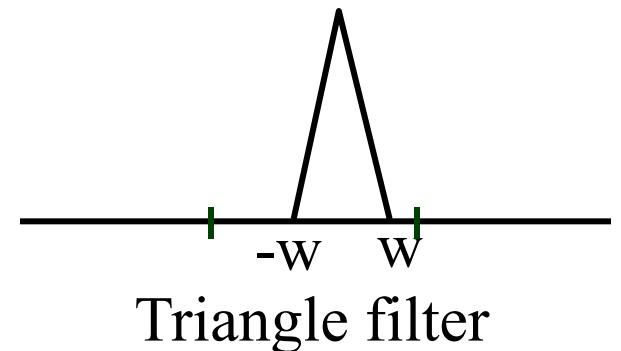
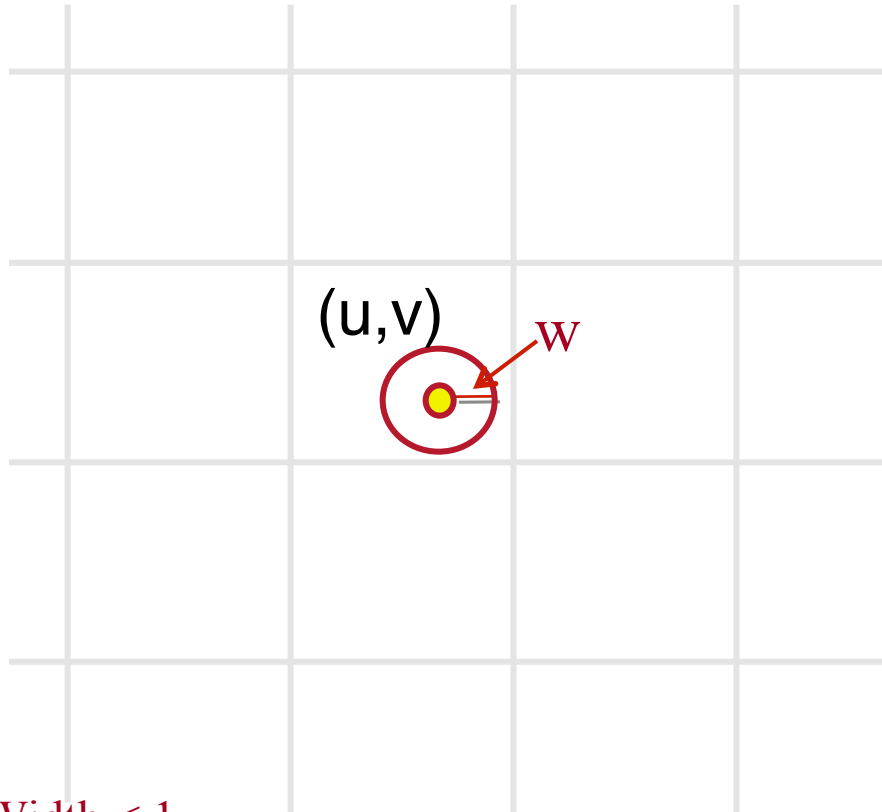
- Filter width chosen based on scale factor of map

Filter must be wide enough to avoid aliasing



Resampling Details

What if width (w) is smaller than sample spacing
For example, when scaling an image larger.

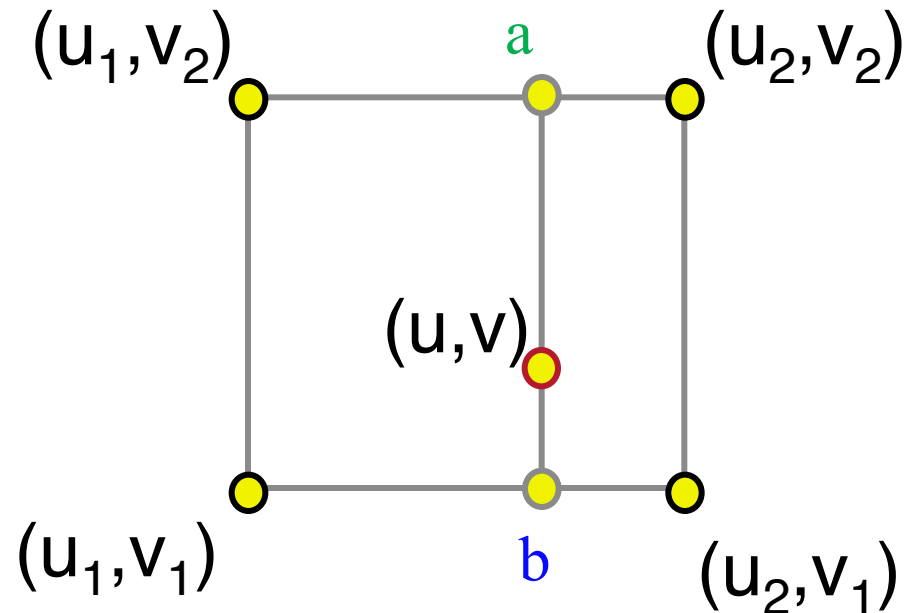


Filter Width < 1



Resampling Details

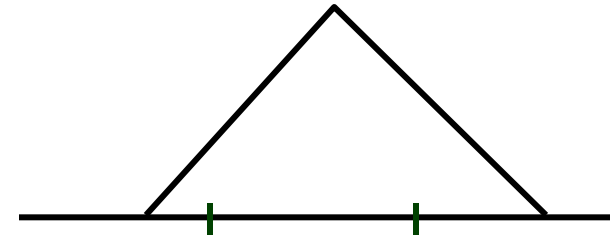
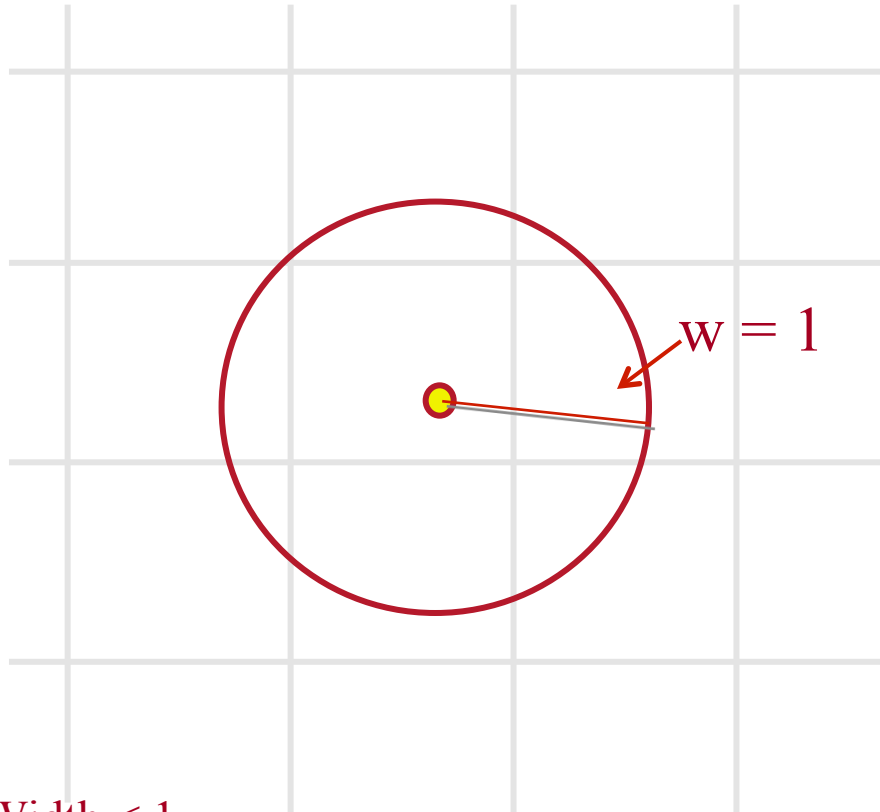
- Alternative 1: Bilinear interpolation of closest pixels
 - **a** = linear interpolation of $\text{src}(u_1, v_2)$ and $\text{src}(u_2, v_2)$
 - **b** = linear interpolation of $\text{src}(u_1, v_1)$ and $\text{src}(u_2, v_1)$
 - **dst**(x, y) = linear interpolation of “**a**” and “**b**”



Filter Width < 1

Resampling Details

- Alternative 2: force width to be at least 1

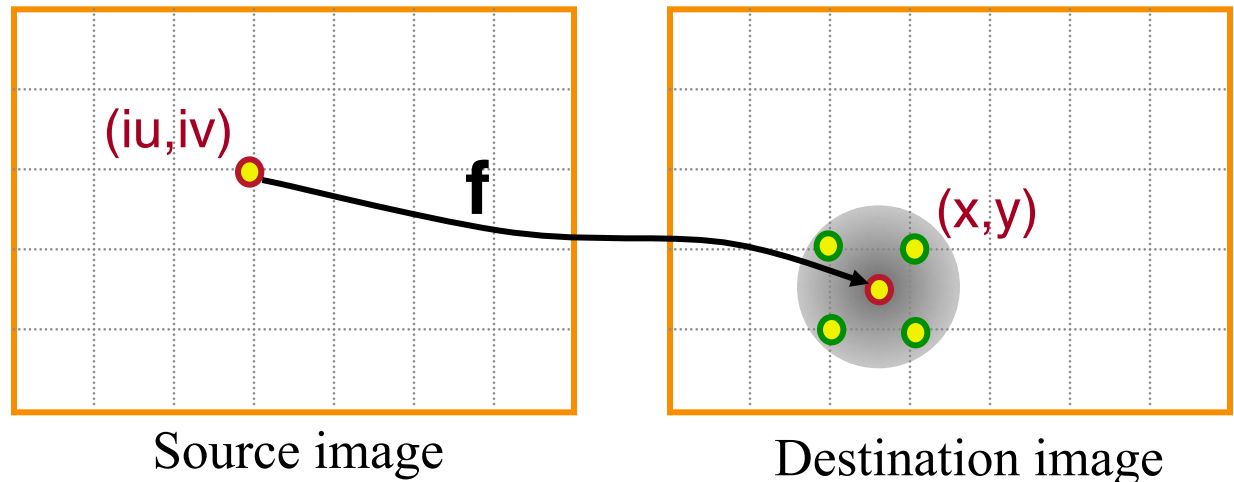


Filter Width < 1

Alternative Algorithm

- Forward mapping:

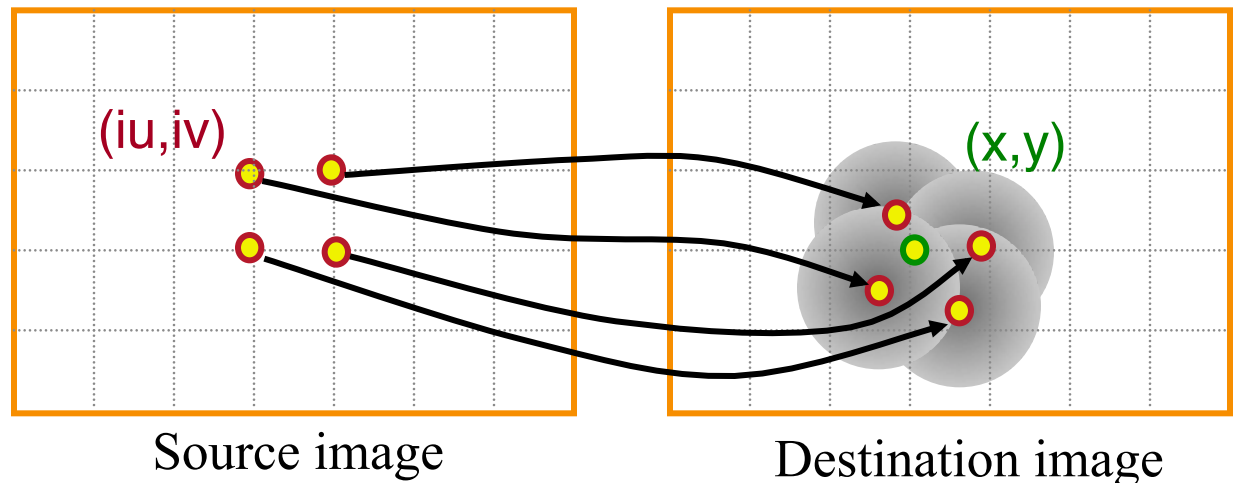
```
Warp(src, dst) {  
  for (int iu = 0; iu < umax; iu++) {  
    for (int iv = 0; iv < vmax; iv++) {  
      float x = fx(iu, iv);  
      float y = fy(iu, iv);  
      float w ≈ 1 / scale(x, y);  
      Splat(src(iu, iv), x, y, k, w);  
    }  
  }  
}
```



Alternative Algorithm

- Forward mapping:

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

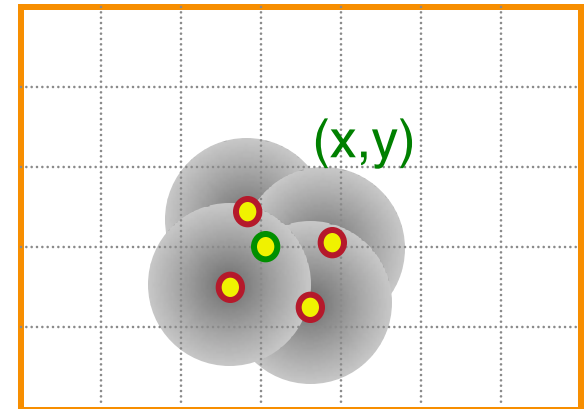


Alternative Algorithm

- Forward mapping:

```
for (int iu = 0; iu < umax; iu++) {  
    for (int iv = 0; iv < vmax; iv++) {  
        float x = fx(iu,iv);  
        float y = fy(iu,iv);  
        float w ≈ 1 / scale(x, y);  
        for (int ix = xlo; ix <= xhi; ix++) {  
            for (int iy = ylo; iy <= yhi; iy++) {  
                dst(ix,iy) += k(x,y,ix,iy,w) * src(iu,iv);  
            }  
        }  
    }  
}
```

Problem?

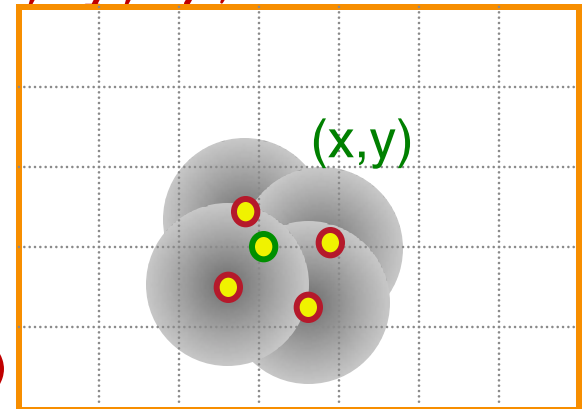


Destination image

Alternative Algorithm

- Forward mapping:

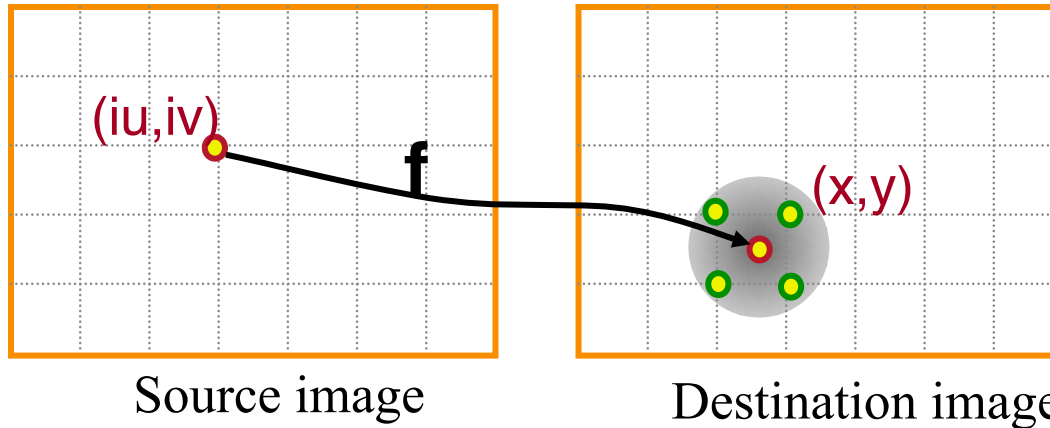
```
for (int iu = 0; iu < umax; iu++) {  
    for (int iv = 0; iv < vmax; iv++) {  
        float x = fx(iu,iv);  
        float y = fy(iu,iv);  
        float w ≈ 1 / scale(x, y);  
        for (int ix = xlo; ix <= xhi; ix++) {  
            for (int iy = ylo; iy <= yhi; iy++) {  
                dst(ix,iy) += k(x,y,ix,iy,w) * src(iu,iv);  
                ksum(ix,iy) += k(x,y,ix,iy,w);  
            }  
        }  
    }  
}  
  
for (ix = 0; ix < xmax; ix++)  
    for (iy = 0; iy < ymax; iy++)  
        dst(ix,iy) /= ksum(ix,iy)
```



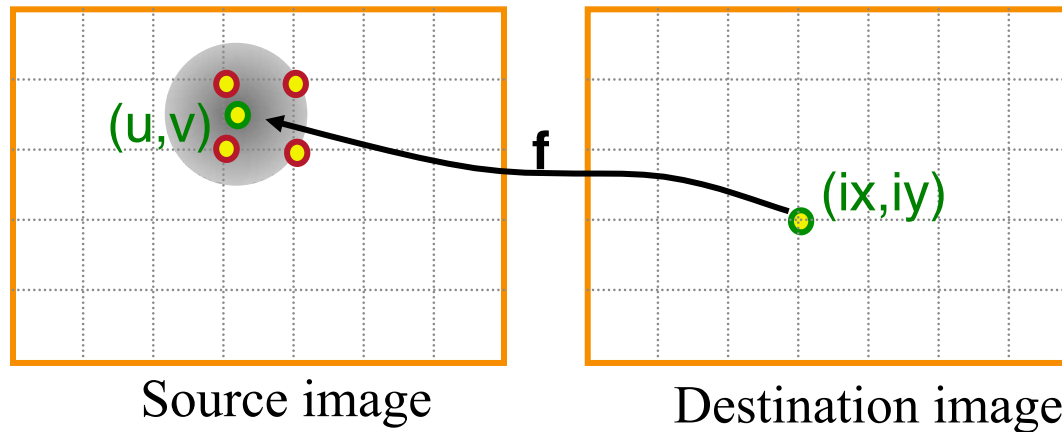
Destination image

Forward vs. Reverse Mapping?

- Forward mapping



- Reverse mapping





Forward vs. Reverse Mapping

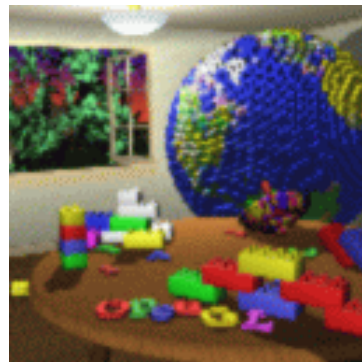
- Tradeoffs:
 - Forward mapping:
 - Requires separate buffer to store weights
 - Reverse mapping:
 - Requires inverse of mapping function, random access to original image

Reverse mapping is usually preferable

Putting it All Together

- Possible implementation of image blur:

```
Blur(src, dst, sigma) {  
    w  $\approx$  3*sigma;  
    for (int ix = 0; ix < xmax; ix++) {  
        for (int iy = 0; iy < ymax; iy++) {  
            float u = ix;  
            float v = iy;  
            dst(ix,iy) = Resample(src,u,v,k,w) ;  
        }  
    }  
}
```



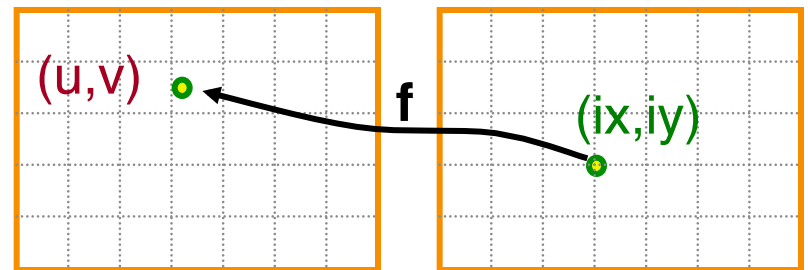
Increasing sigma →



Putting it All Together

- Possible implementation of image scale:

```
Scale(src, dst, sx, sy) {  
    w  $\approx$  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);  
        }  
    }  
}
```



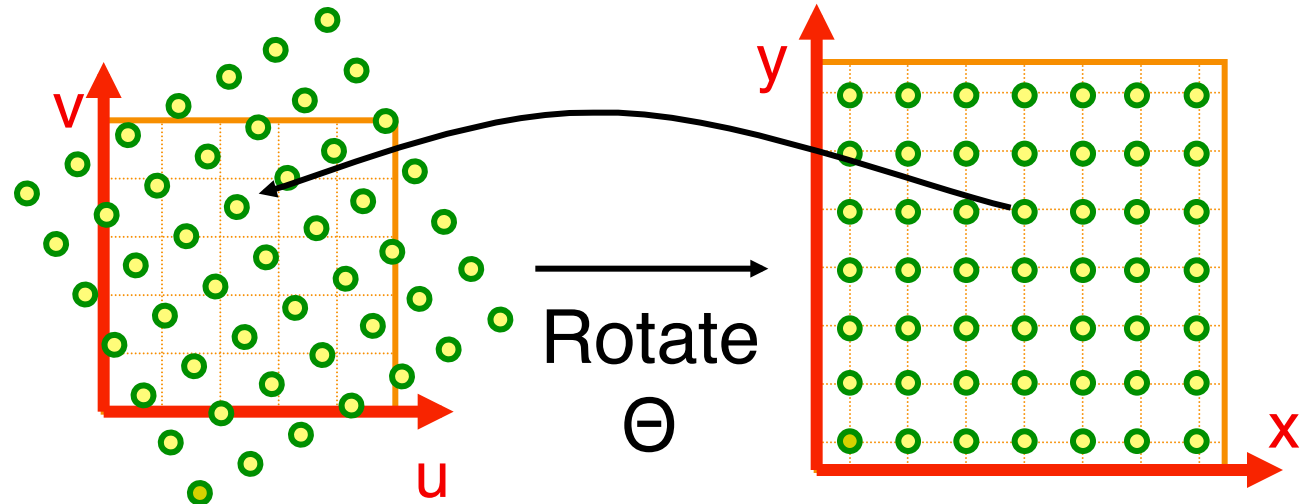
Source image

Destination image

Putting it All Together

- Possible implementation of image rotation:

```
Rotate(src, dst,  $\Theta$ ) {  
    w  $\approx$  1  
    for (int ix = 0; ix < xmax; ix++) {  
        for (int iy = 0; iy < ymax; iy++) {  
            float u = ix*cos(- $\Theta$ ) - iy*sin(- $\Theta$ );  
            float v = ix*sin(- $\Theta$ ) + iy*cos(- $\Theta$ );  
            dst(ix,iy) = Resample(src,u,v,k,w);  
        }  
    }  
}
```



Summary



- Mapping
 - Parametric
 - Correspondences
- Sampling, reconstruction, resampling
 - Frequency analysis of signal content
 - Filter to avoid aliasing
 - Reduce visual artifacts due to aliasing
 - » **Blurring is better than aliasing**
- Image processing
 - Forward vs. reverse mapping