Image Warping and Morphing

Review: Signal Processing

- **Fourier transform**
- **Filtering and the convolution theorem**
- **Sampling and reconstruction**
- **Nyquist frequency**
- **Aliasing and antialiasing**

Warping transformation: zoom in/out, rotate, warp

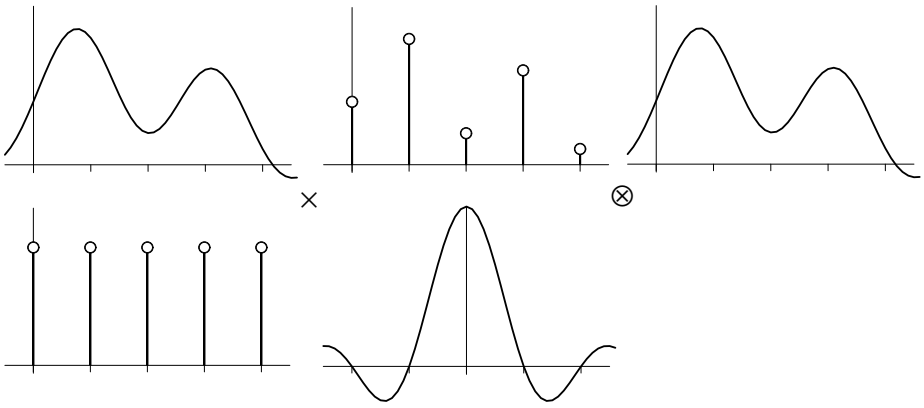
Resampling

Magnification v. minification

Feature-based morphing

Copyright © Pat Hanrahan

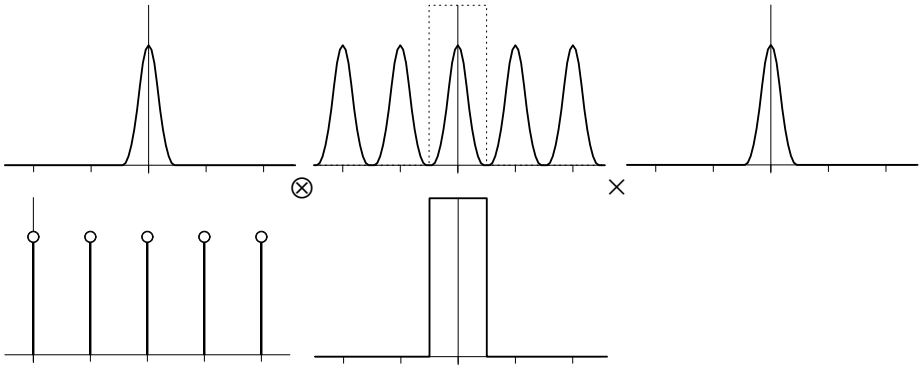
Sampling & Reconstruction: Space



Space Domain

Copyright © Pat Hanrahan

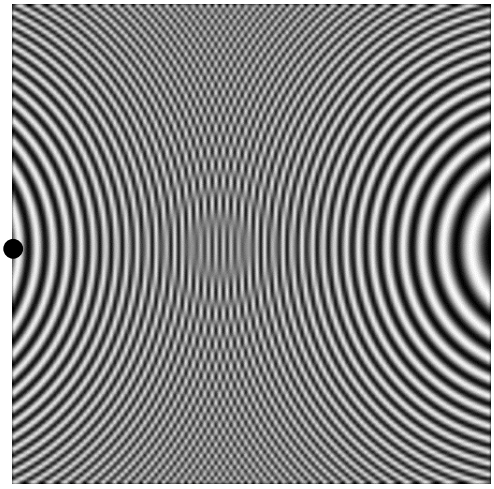
Sampling & Reconstruction: Freq



Frequency Domain

Copyright © Pat Hanrahan

Sampling a "Zone Plate"



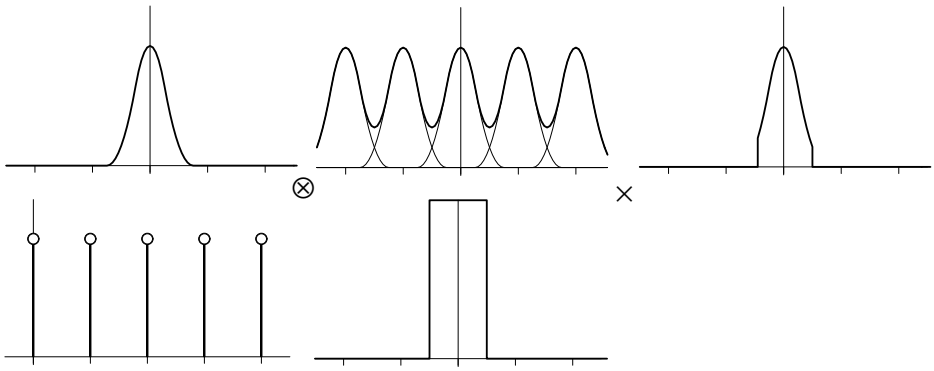
0,0

Note Moire Patterns

$F(x,y) = \sin r^2$

Copyright © Pat Hanrahan

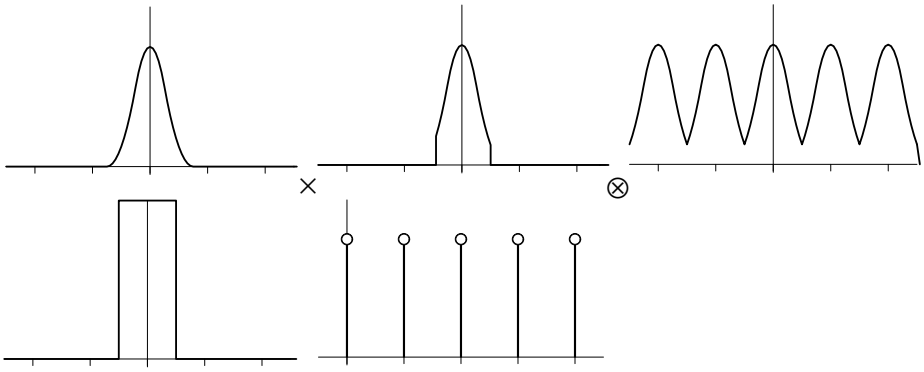
Undersampling: Aliasing



Frequency Domain

Copyright © Pat Hanrahan

Antialiasing by Prefiltering



Frequency Domain

Copyright © Pat Hanrahan

Antialiasing by Prefiltering

Ideally: perfectly low-pass filter (convolve with a sinc function) to bandlimit the function to the Nyquist sampling rate.

Typically: convolve with a box filter (square pixel area average) or circular filter (circular pixel average) before sampling

Practically:

- **Constant colored lines and polygonal fragments doable**
- **Complex environments not doable**

Copyright © Pat Hanrahan

Image Warping and Morphing

- **Image warping**

  Transform an image:

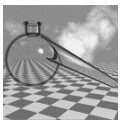
  Zoom up, down

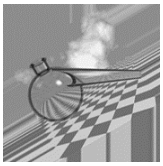
  Rotate, shear

  Warp: control mesh

  Later in the course: texture mapping

  Resampling

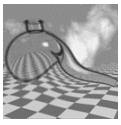




- **Image morphing**

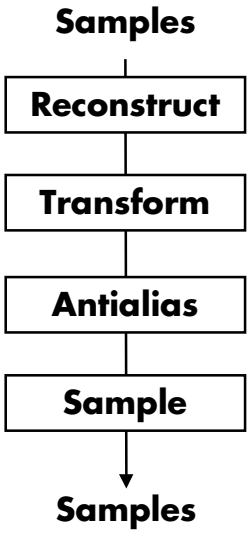
  Simultaneous warp and cross-dissolve

  Feature-based warping



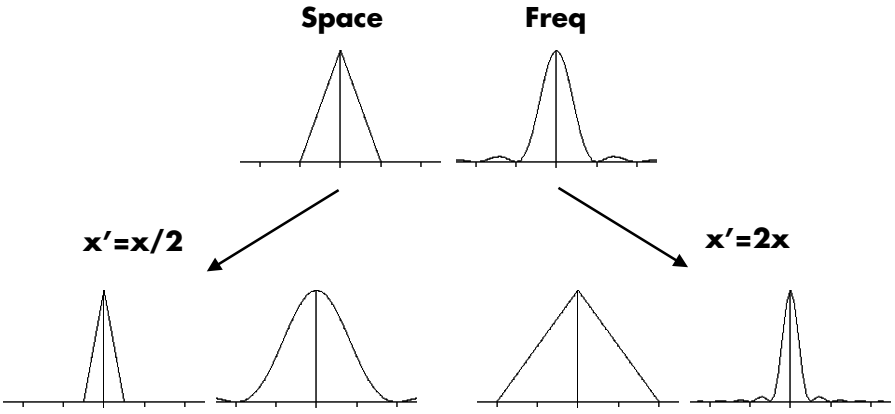
Copyright © Pat Hanrahan

Idealized Resampling



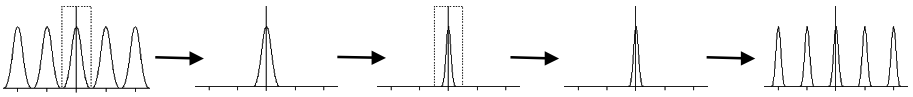
Copyright © Pat Hanrahan

Scaling



Copyright © Pat Hanrahan

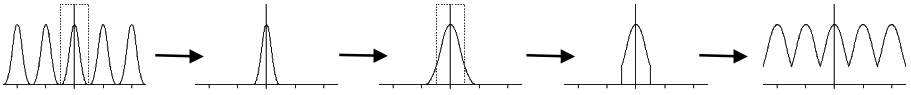
Magnification



Reconstruct **Transform** **Antialias** **Sample**
x′=2x

Copyright © Pat Hanrahan

Minification



Reconstruct **Transform** **Antialias** **Sample**
x′=x/2

Copyright © Pat Hanrahan

Magnification

One to many ...

Just reconstruct, no need to antialias

Magnification

1. **Reconstruct**
2. **Transform**
3. **Sample**

Interpolation

1. **Transform**
2. **Reconstruct (interpolate)**

Copyright © Pat Hanrahan

Minification

Many to one ...

Just antialias, no need to reconstruct

Minification

1. **Transform**
2. **Antialias**
3. **Sample**

Decimate

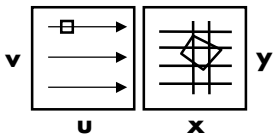
1. **Transform**
2. **Filter (averaging)**

Copyright © Pat Hanrahan

Mapping Algorithms

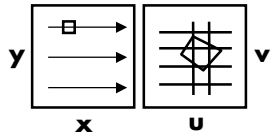
Forward mapping (video effects)

```
for all (u,v)
  (x,y) = f(u,v)
  image[x][y] = texture[u][v]
```



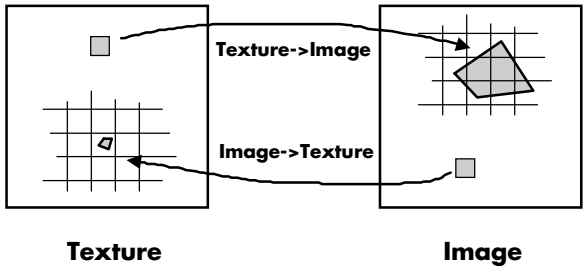
Backward mapping (workstations)

```
for all (x,y)
  (u,v) = f^-1(x,y)
  image[x][y] = texture[u][v]
```



Copyright © Pat Hanrahan

Filtering



Two cases:

1. **Magnification / Interpolation**
2. **Minification / Decimation / Averaging**

Copyright © Pat Hanrahan

Bilinear Interpolation

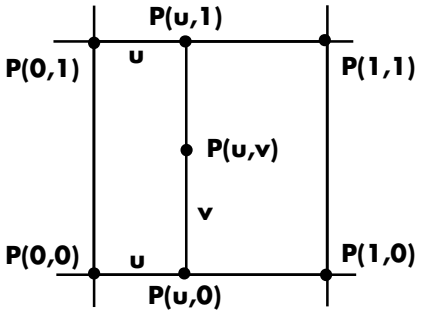
Linear interpolation (lerp)

```
v' = lerp(v0,v1,t) = (1-t)*v0 + t*v1;
```

Equivalent to reconstructing with a triangle filter

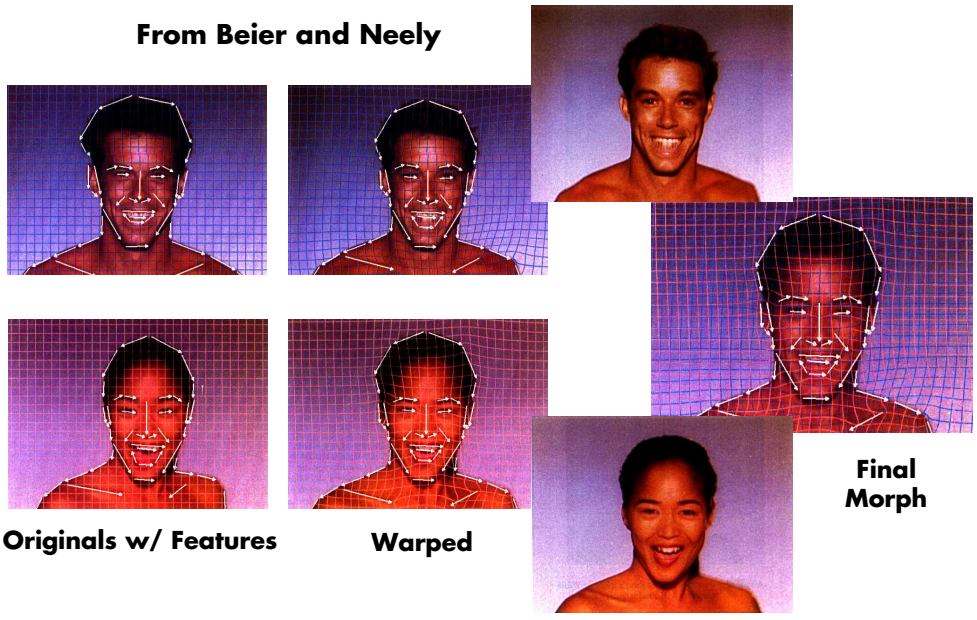
Bilinear interpolation

```
P(u,0) = lerp(P(0,0),P(1,0),u);
P(u,1) = lerp(P(0,1),P(1,1),u);
P(u,v) = lerp(P(u,0),P(u,1),v);
```



Copyright © Pat Hanrahan

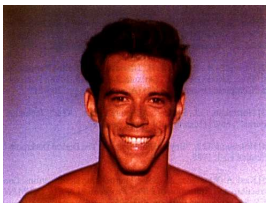
From Beier and Neely



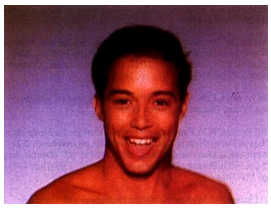
Originals w/ Features

Warped

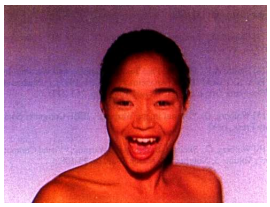
Final Morph



First Frame



Intermediate Frame



Last Frame

Creating Correspondences

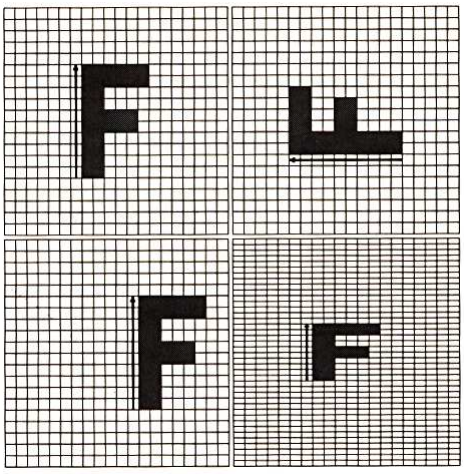
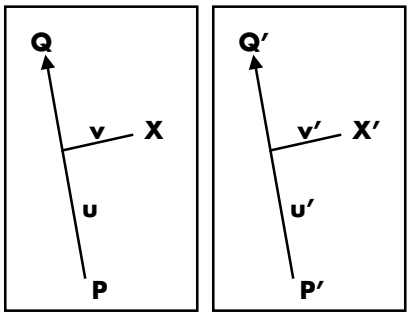


Figure 2, Beier and Neely



Source Destination

Copyright © Pat Hanrahan

Multiple Line Algorithm

```
For each pixel X in the destination
    DSUM = 0;
    weightsum = 0;
    for each line (Pi,Qi)
        calculate (u,v) based on (Pi,Qi)
        calculate (u',v') based on (Pi',Qi')
        weight = length^p/(a+distance)^b
        DSUM += weight * (X'-X)
        weightsum += weight
    X' = X + DSUM/weightsum
    destination[X] = source[X'];
```


Copyright © Pat Hanrahan