Image Warping

Thomas Funkhouser
Princeton University
C0S 426, Fall 2000

Image Processing

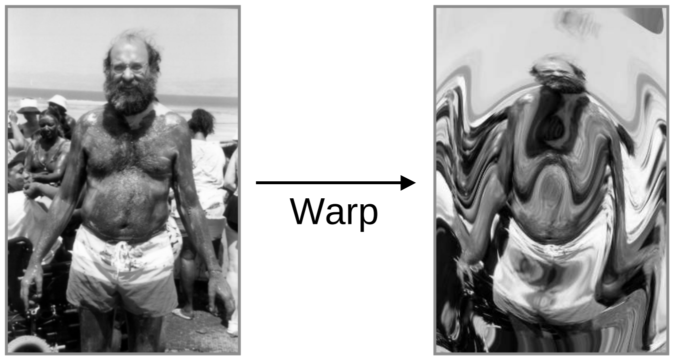
- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite
  - Morph

Image Processing

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite
  - Morph

Image Warping

- Move pixels of image
  - Mapping
  - Resampling



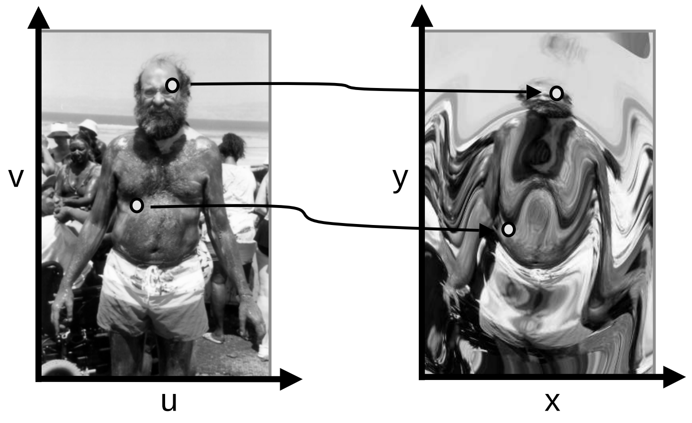
Source image Destination image

Overview

- Mapping
  - Forward
  - Reverse
- Resampling
  - Point sampling
  - Triangle filter
  - Gaussian filter

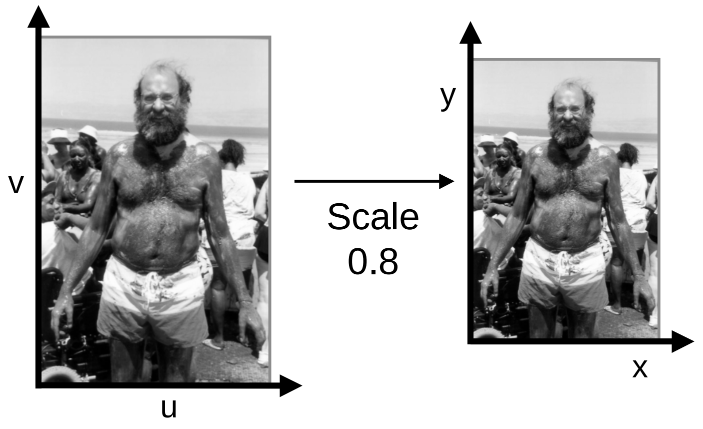
Mapping

- Define transformation
  - Describe the destination (x,y) for every location (u,v) in the source (or vice-versa, if invertible)



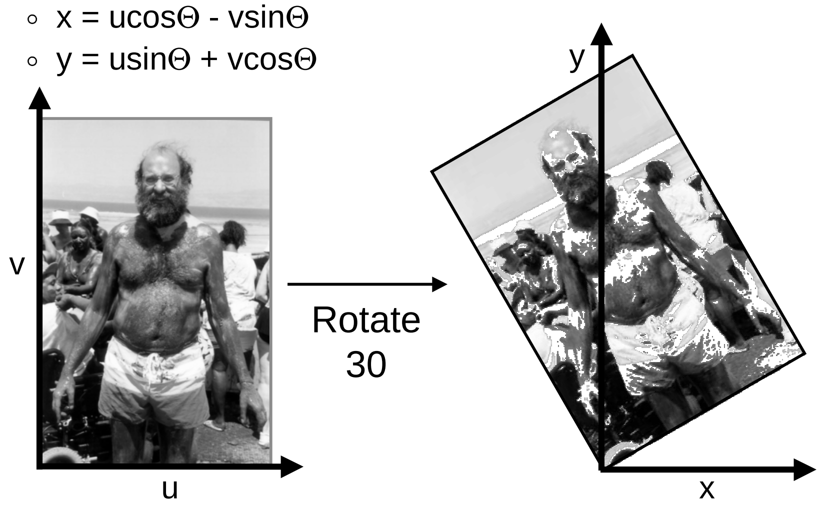
Example Mappings

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



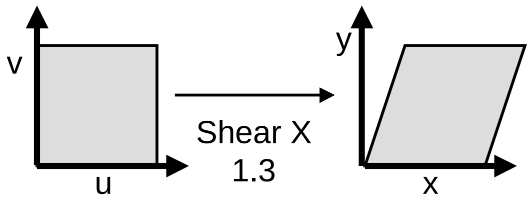
Example Mappings

- Rotate by Θ degrees:
  - x = ucosΘ - vsinΘ
  - y = usinΘ + vcosΘ

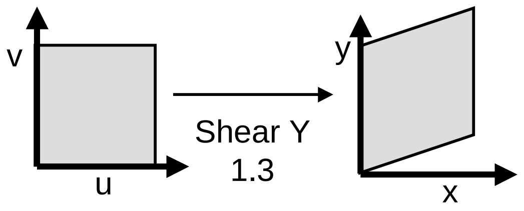


Example Mappings

- Shear in X by *factor:*
  - x = u + *factor* * v
  - y = v



- Shear in Y by *factor:*
  - x = u
  - y = v + *factor* * u



Other Mappings

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

Fish-eye

"Swirl"

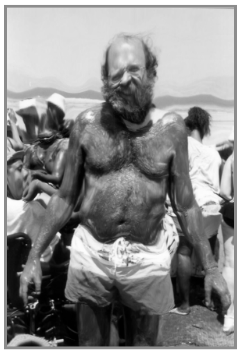
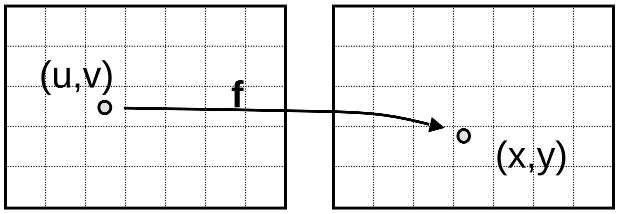

"Rain"

Image Warping Implementation I

- Forward mapping:

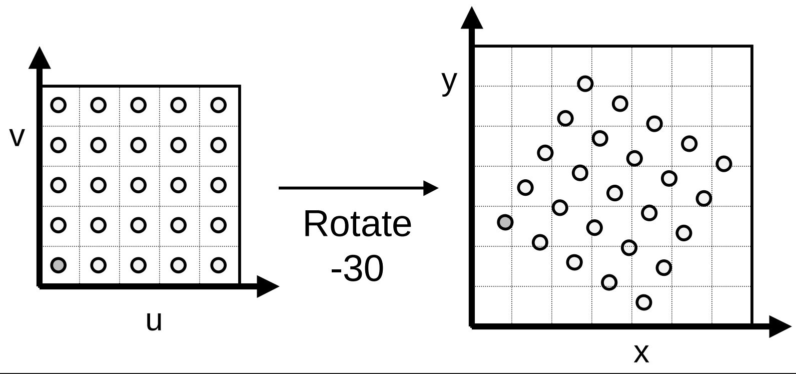
```
for (int u = 0; u < umax; u++) {
  for (int v = 0; v < vmax; v++) {
    float x = f_x(u,v);
    float y = f_y(u,v);
    dst(x,y) = src(u,v);
  }
}
```



Source image Destination image

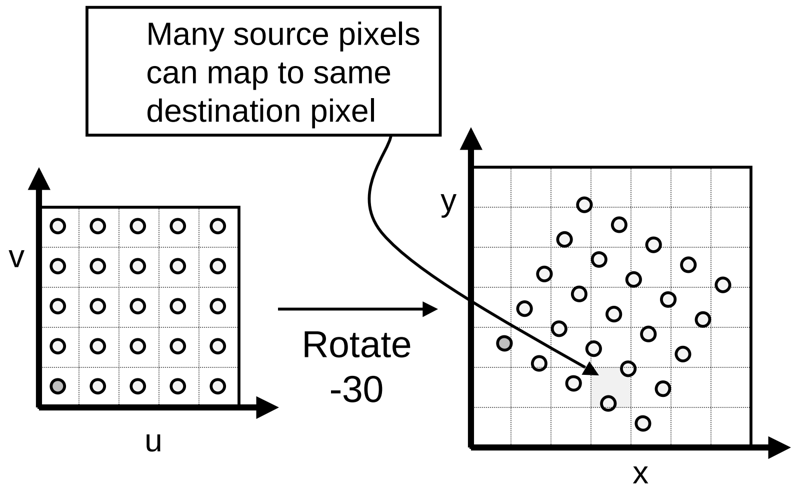
Forward Mapping

- Iterate over source image



Forward Mapping - NOT

- Iterate over source image



Forward Mapping - NOT

- Iterate over source image

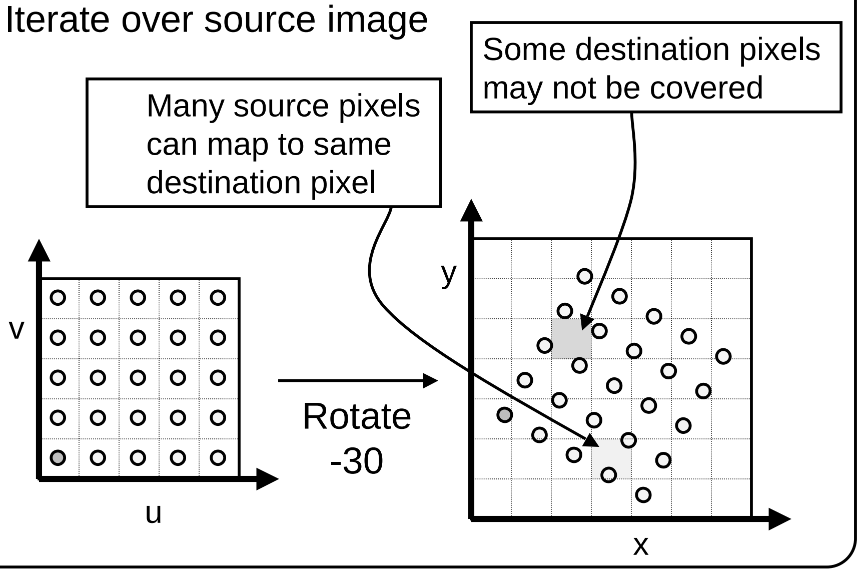
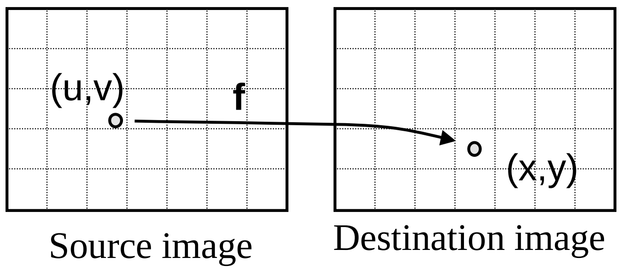

Some destination pixels may not be covered
Many source pixels can map to same destination pixel
v
u
Rotate -30
y
x

Image Warping Implementation II

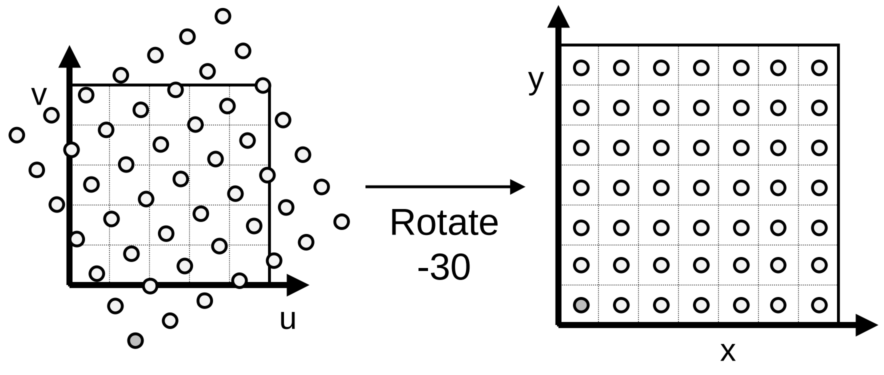
- Reverse mapping:

```
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = fx-1(x,y);
    float v = fy-1(x,y);
    dst(x,y) = src(u,v);
  }
}
```



Reverse Mapping

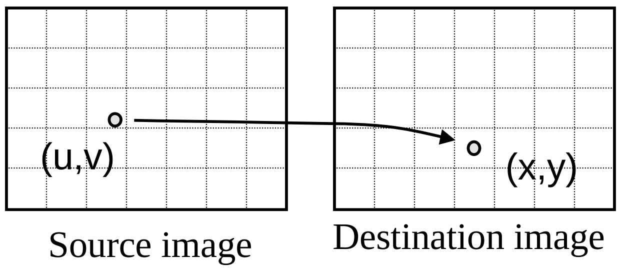
- Iterate over destination image
  - Must resample source
  - May oversample, but much simpler!



Resampling

- Evaluate source image at arbitrary (u,v)

(u,v) does not usually have integer coordinates



Overview

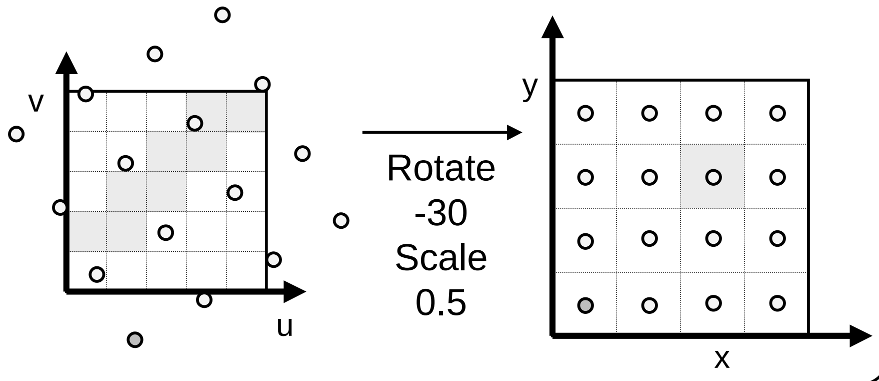
- Mapping
  - Forward
  - Reverse

» **Resampling**
  - Point sampling
  - Triangle filter
  - Gaussian filter

Point Sampling

- Take value at closest pixel:
  - int iu = trunc(u+0.5);
  - int iv = trunc(v+0.5);
  - dst(x,y) = src(iu,iv);

This method is simple, but it causes aliasing



Triangle Filtering

- Convolve with triangle filter

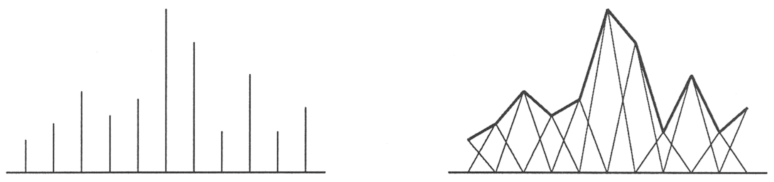
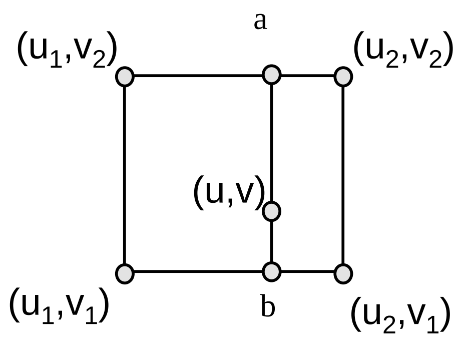


Figure 2.4 Wolberg

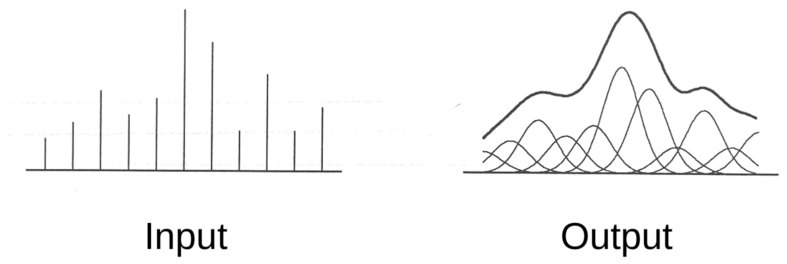
Triangle Filtering

- 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

- Convolve with Gaussian filter

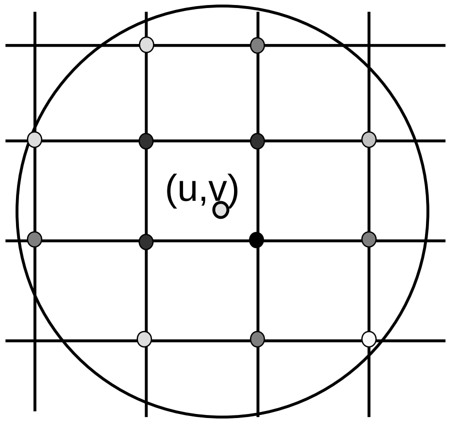


Width of Gaussian kernel affects bluriness

Figure 2.4 Wolberg

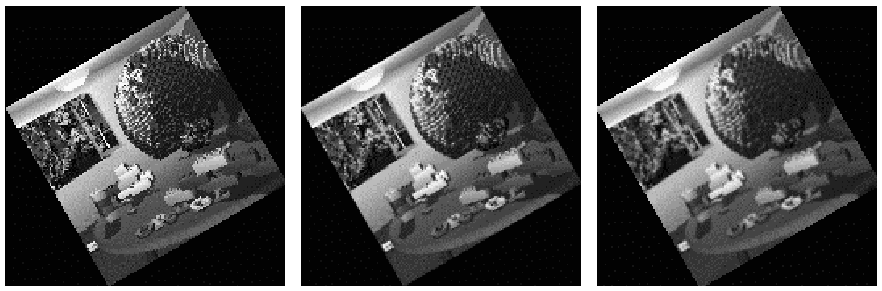
Gaussian Filtering

- Compute weighted sum of pixel neighborhood:
  - Weights are normalized values of Gaussian function



Filtering Methods Comparison

- Trade-offs
  - Aliasing versus blurring
  - Computation speed

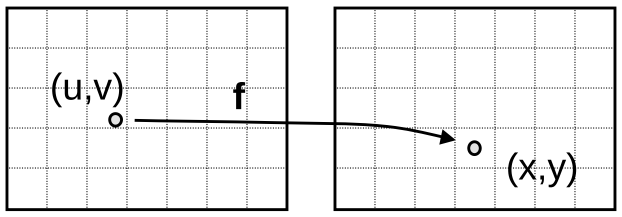


Point Bilinear Gaussian

Image Warping Implementation

- Reverse mapping:

```
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = $f_x^{-1}$(x,y);
    float v = $f_y^{-1}$(x,y);
    dst(x,y) = resample_src(u,v,w);
  }
}
```

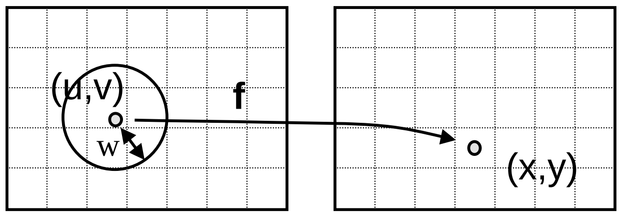


Source image   Destination image

Image Warping Implementation

- Reverse mapping:

```
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = $f_x^{-1}$(x,y);
    float v = $f_y^{-1}$(x,y);
    dst(x,y) = resample_src(u,v,w);
  }
}
```

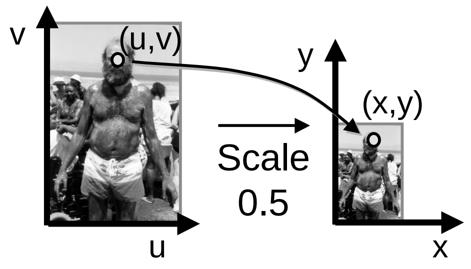


Source image   Destination image

Example: Scale

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

```
float w ≅ max(1.0/sx,1.0/sy);
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = x / sx;
    float v = y / sy;
    dst(x,y) = resample_src(u,v,w);
  }
}
```

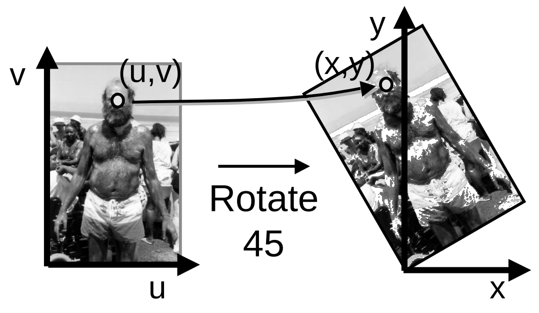


Example: Rotate

- Rotate (src, dst, theta):

```
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = x*cos(-Θ) - y*sin(-Θ);
    float u = x*sin(-Θ) + y*cos(-Θ);
    dst(x,y) = resample_src(u,v,w);
  }
}
```

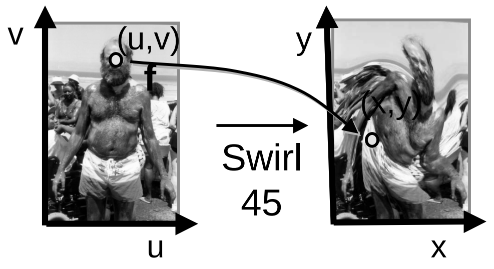
$x = u\cos\Theta - v\sin\Theta$
$y = u\sin\Theta + v\cos\Theta$



Example: Fun

- Swirl (src, dst, theta):

```
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = rot(dist(x,xcenter)*theta);
    float v = rot(dist(y,ycenter)*theta);
    dst(x,y) = resample_src(u,v,w);
  }
}
```



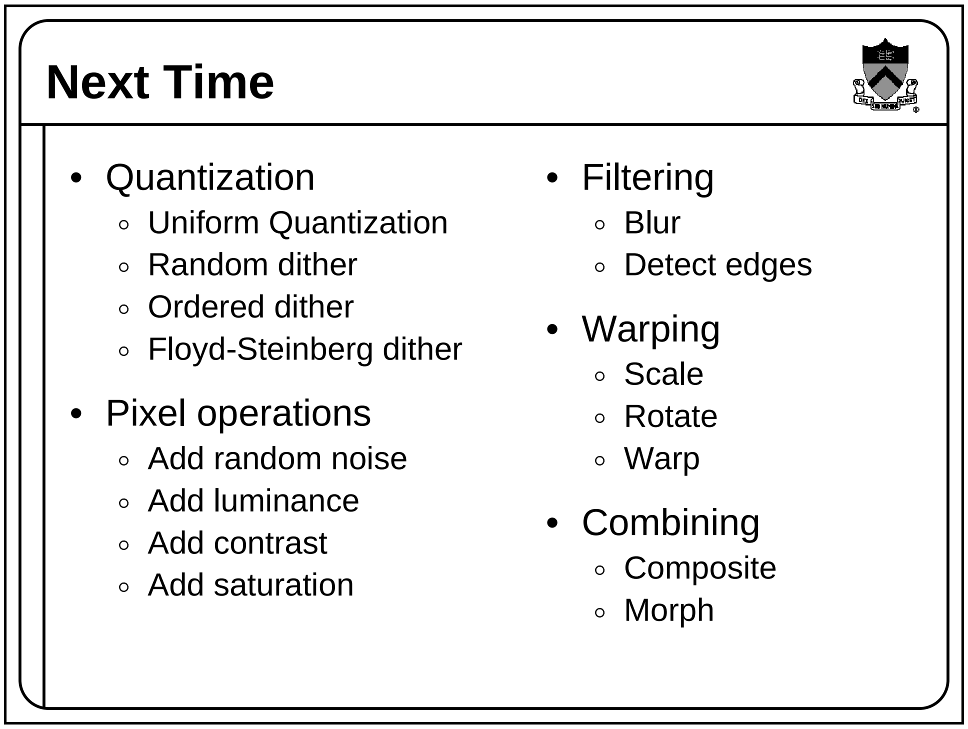
Summary

- Mapping
  - Forward
  - Reverse
- Resampling
  - Point sampling
  - Triangle filter
  - Gaussian filter

Reverse mapping is simpler to implement

Different filters trade-off speed and aliasing/blurring

Fun and creative warps are easy to implement!



Next Time

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite
  - Morph