



# Image Warping and Compositing

Thomas Funkhouser  
Princeton University  
COS 426, Fall 1999



## Overview

- 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
  - Warps
  - Morphs
- Compositing
  - Matting

## Overview

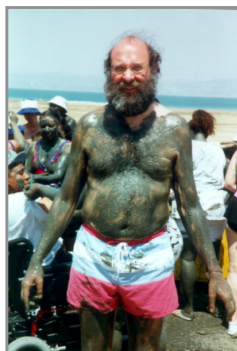


- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warps
  - Morphs
- Compositing
  - Matting

## Image Warping



- Move pixels of image
  - Mapping
  - Resampling



Source image

→ Warp

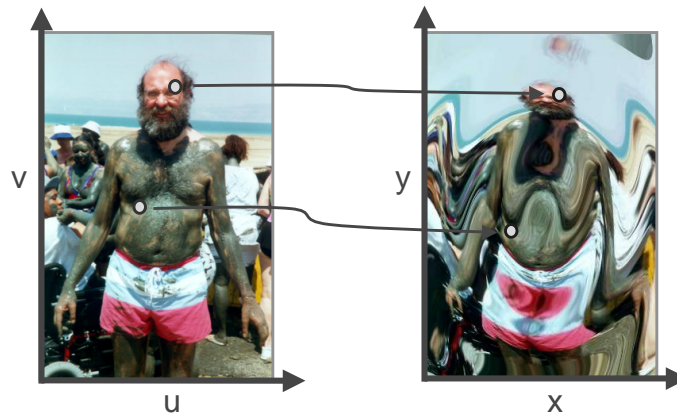


Destination image

## Mapping



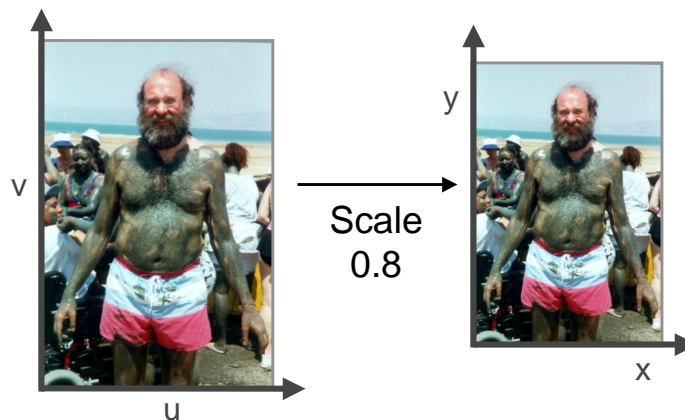
- Define transformation
  - Describe the destination  $(x,y)$  for every location  $(u,v)$  in the source (or vice-versa, if invertible)



## Example Mappings



- Scale by *factor*.
  - $x = \text{factor} * u$
  - $y = \text{factor} * v$

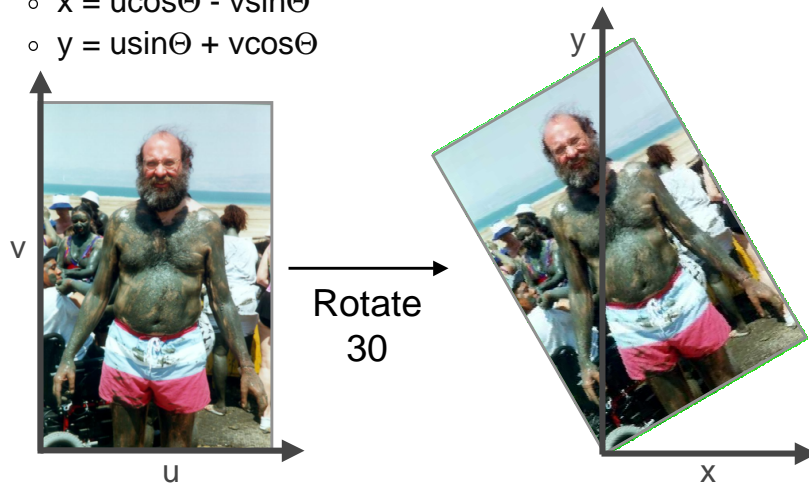


## Example Mappings



- Rotate by  $\Theta$  degrees:

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

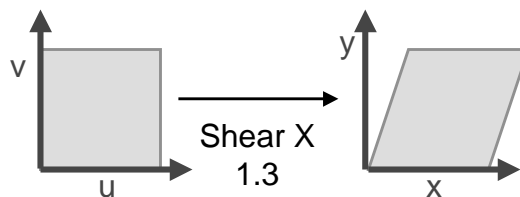


## Example Mappings



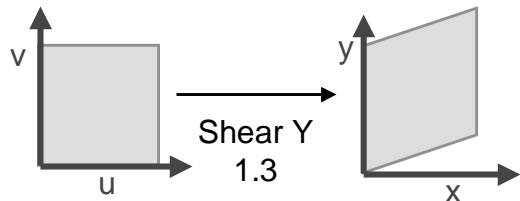
- Shear in X by *factor*:

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



- Shear in Y by *factor*:

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

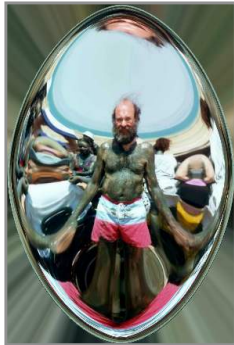


## Other Mappings



- Any function of  $u$  and  $v$ :

- $x = f_x(u, v)$
- $y = f_y(u, v)$



Fish-eye



“Swirl”



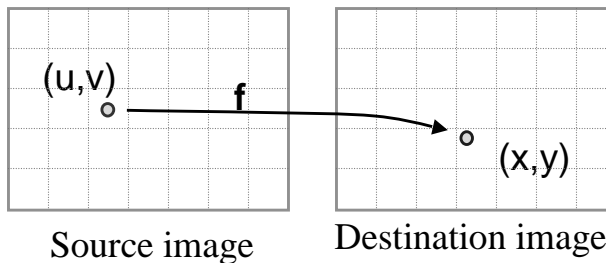
“Rain”

## Image Warping Implementation I



- Forward mapping:

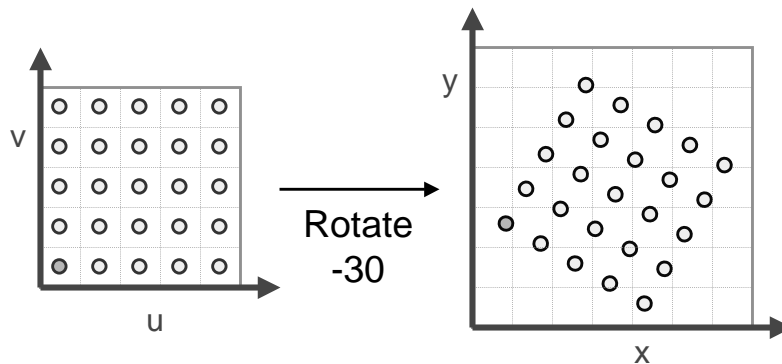
```
for (int u = 0; u < umax; u++) {  
    for (int v = 0; v < vmax; v++) {  
        float x = fx(u, v);  
        float y = fy(u, v);  
        dst(x, y) = src(u, v);  
    }  
}
```



## Forward Mapping



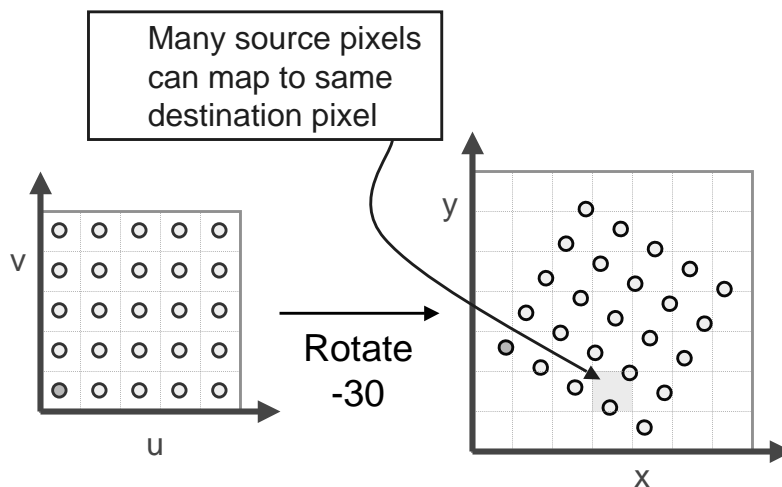
- Iterate over source image



## Forward Mapping - NOT



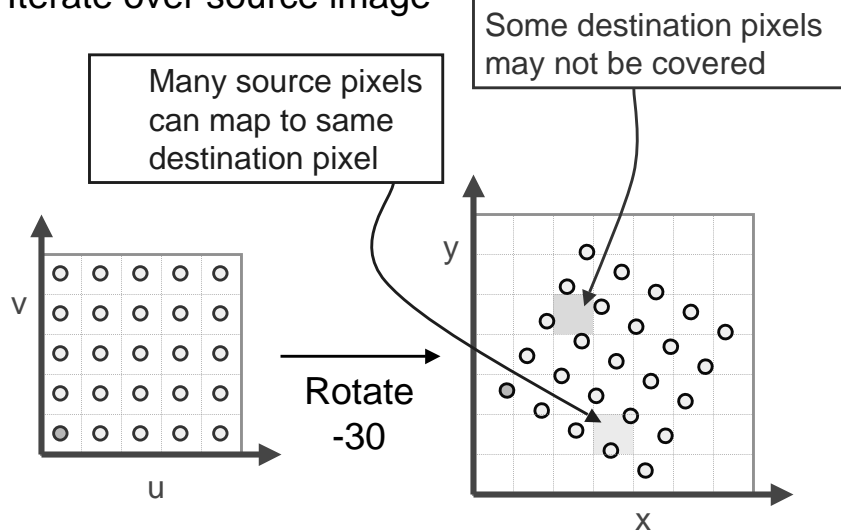
- Iterate over source image



## Forward Mapping - NOT



- Iterate over source image

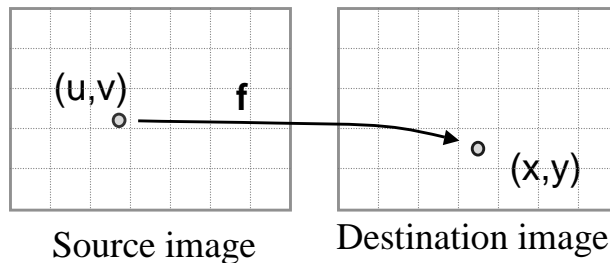


## Image Warping Implementation II



- Reverse mapping:

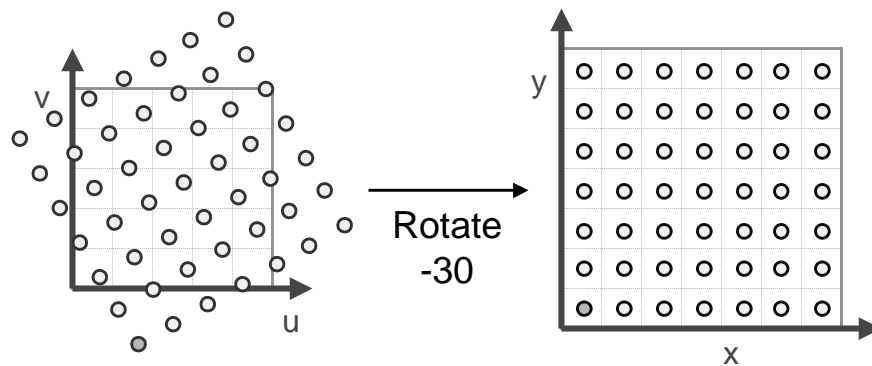
```
for (int x = 0; x < xmax; x++) {  
    for (int y = 0; y < ymax; y++) {  
        float u =  $f_x^{-1}(x,y)$ ;  
        float v =  $f_y^{-1}(x,y)$ ;  
        dst(x,y) = src(u,v);  
    }  
}
```



## Reverse Mapping



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

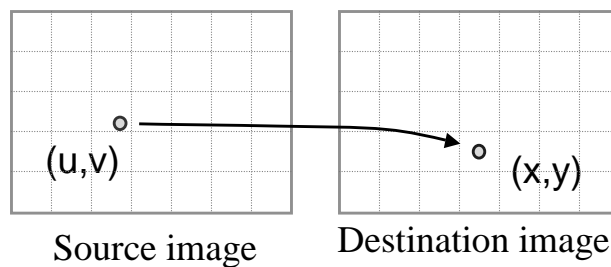


## Resampling



- Evaluate source image at arbitrary  $(u,v)$

$(u,v)$  does not usually have integer coordinates





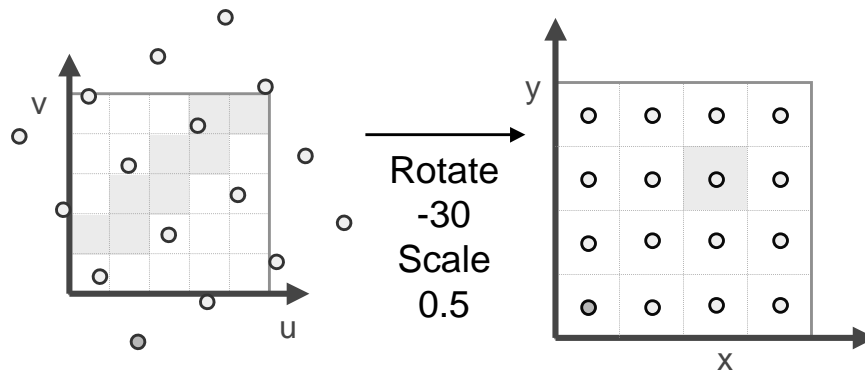
## Point Sampling



- Take value at closest pixel:

- $\text{int } iu = \text{trunc}(u+0.5);$
- $\text{int } iv = \text{trunc}(v+0.5);$
- $\text{dst}(x,y) = \text{src}(iu,iv);$

This method is simple,  
but it causes aliasing

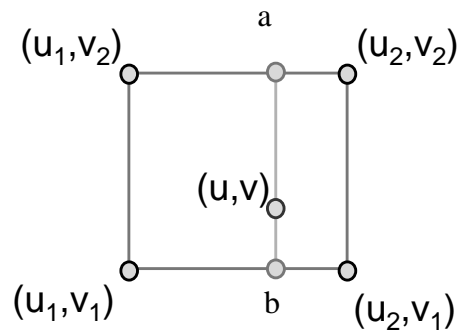


## Triangle Filtering



- Bilinear interpolation of four closest pixels

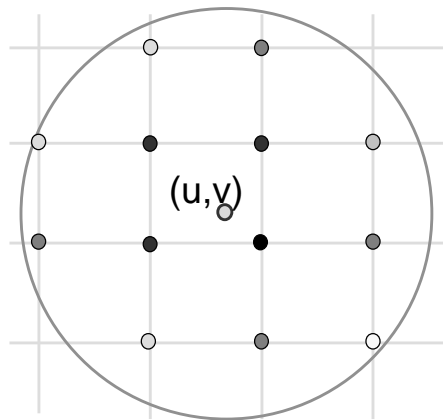
- $a = \text{linear interpolation of } \text{src}(u_1, v_2) \text{ and } \text{src}(u_2, v_2)$
- $b = \text{linear interpolation of } \text{src}(u_1, v_1) \text{ and } \text{src}(u_2, v_1)$
- $\text{dst}(x,y) = \text{linear interpolation of "a" and "b"}$



## Gaussian Filtering



- Weighted combination of pixel neighborhood:
  - Weights are normalized values of gaussian function



## Gaussian Filtering



- Width of gaussian kernel affects blurriness

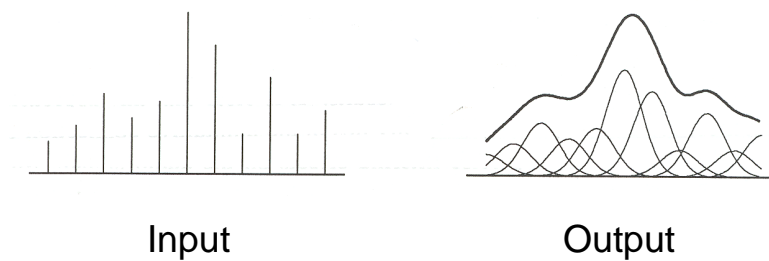


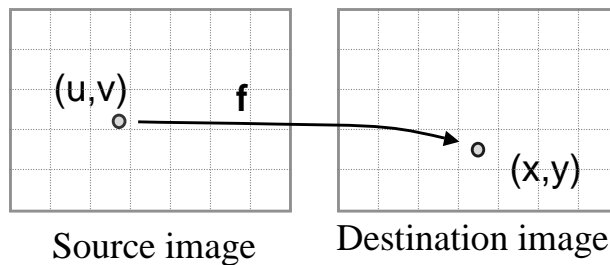
Figure 2.4 Wolberg

## 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);  
    }  
}
```



## Overview



- Image warping
  - Mapping
  - Resampling
- Image compositing
  - Blue-screen mattes
  - Alpha channel
  - Porter-Duff compositing algebra

## Image Compositing



- Separate an image into “elements”
  - Render independently
  - Composite together
- Applications
  - Cel animation
  - Chroma-keying
  - Blue-screen matting



Dobkin meets Elvis

## Blue-Screen Matting



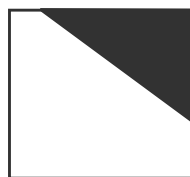
- Composite foreground and background images
  - Create background image
  - Create foreground image with blue background
  - Insert non-blue foreground pixels into background



## Alpha Channel

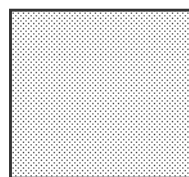


- Encodes pixel coverage information
  - $\alpha = 0$ : no coverage (or transparent)
  - $\alpha = 1$ : full coverage (or opaque)
  - $0 < \alpha < 1$ : partial coverage (or semi-transparent)
- Example:  $\alpha = 0.3$



Partial  
Coverage

or



Semi-  
Transparent

## Pixels with Alpha

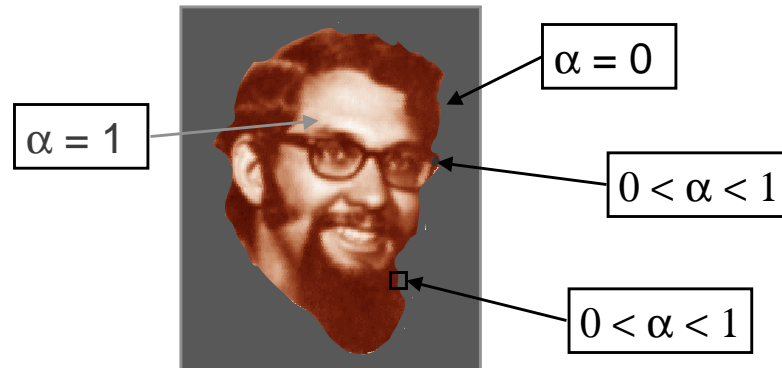


- Alpha channel convention:
  - $(r, g, b, \alpha)$  represents a pixel that is  $\alpha$  covered by the color  $C = (r/\alpha, g/\alpha, b/\alpha)$ 
    - » Color components are premultiplied by  $\alpha$
    - » Can display  $(r, g, b)$  values directly
    - » Closure in composition algebra
- What is the meaning of the following?
  - $(0, 1, 0, 1)$  = Full green, full coverage
  - $(0, 1/2, 0, 1)$  = Half green, full coverage
  - $(0, 1/2, 0, 1/2)$  = Full green, half coverage
  - $(0, 1/2, 0, 0)$  = No coverage

## Compositing with Alpha



- Controls the linear interpolation of foreground and background pixels when elements are composited



## Semi-Transparent Objects



- Suppose we put A over B over background G

- How much of B is blocked by A?

$$\alpha_A$$

- How much of B shows through A?

$$(1 - \alpha_A)$$

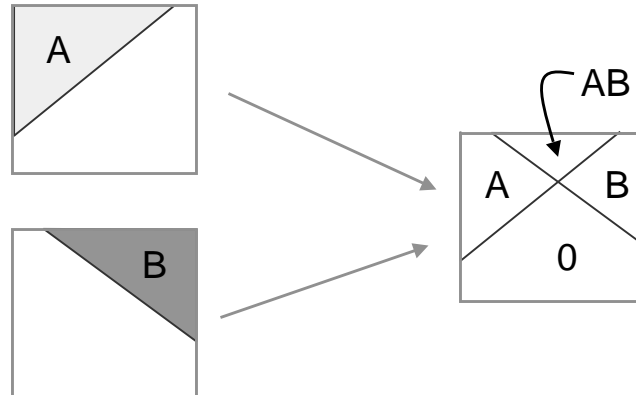
- How much of G shows through both A and B?

$$(1 - \alpha_A) * (1 - \alpha_B)$$

## Opaque Objects



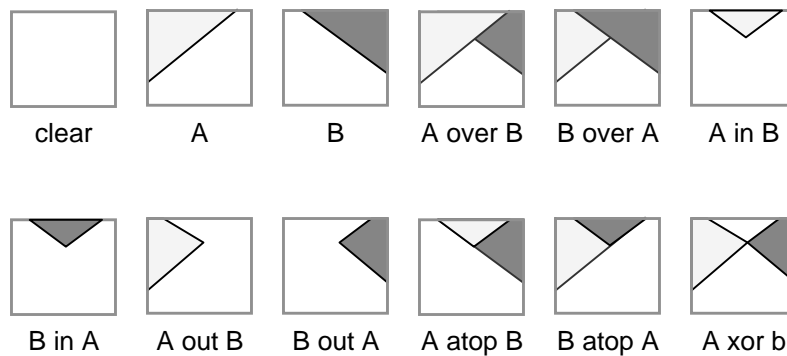
- How do we combine two partially covered pixels?
  - 3 possible colors (0, A, B)
  - 4 regions (0, A, B, AB)



## Composition Algebra



- 12 reasonable combinations



Porter & Duff '84

## Over Operator



- For  $C_B$  and  $C_F$ , which are not premultiplied:
  - $C' = \alpha_B(1-\alpha_F)*C_B + \alpha_F*C_F$
  - $\alpha = \alpha_B(1-\alpha_F) + \alpha_F$
- For  $C'_B$  and  $C'_F$ , which are premultiplied:
  - $C' = (1-\alpha_B)*C'_F + C'_B$
  - $\alpha = \alpha_B(1-\alpha_F) + \alpha_F$

Assumption:  
coverages of B and F  
are uncorrelated  
for each pixel

## Image Composition Example





## Summary



- Image Warping
  - Mapping
  - Resampling
  - Reconstruction
- Image Composition
  - Alpha channel
  - Composition algebra
  - Blue-screen matting