Image Warping, Compositing & Morphing

Thomas Funkhouser
Princeton University
COS 426, Spring 2004

Image Processing

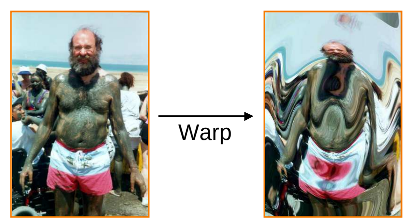
- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite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



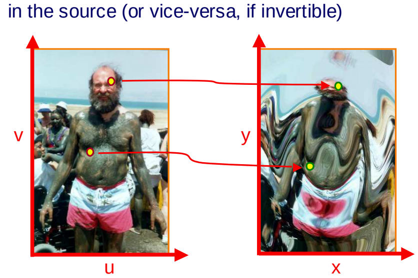
Source image   Destination image

Overview

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

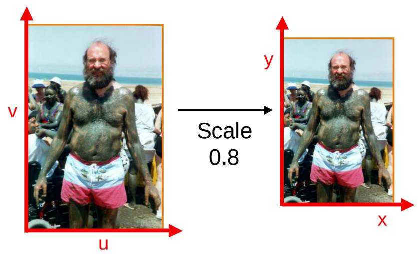
Mapping

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



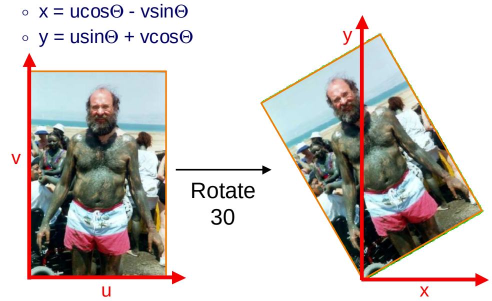
Example Mappings

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



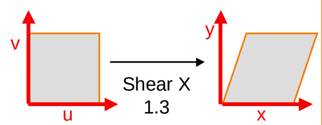
Example Mappings

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

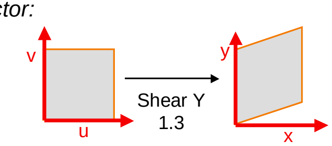


Example Mappings

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

"Swirl"

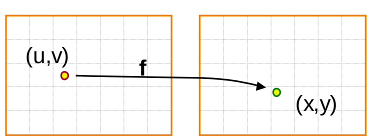
Fish-eye

"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);
  }
}
```

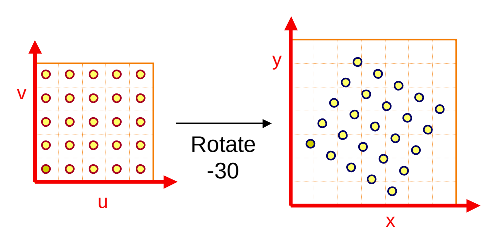


Source image

Destination image

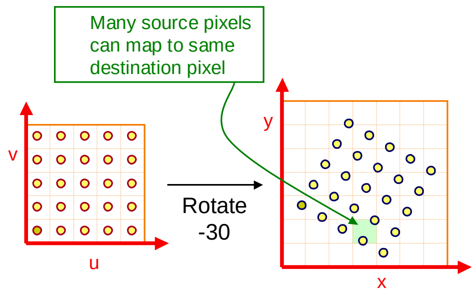
Forward Mapping

- Iterate over source image



Forward Mapping - NOT

- Iterate over source image



Forward Mapping - NOT

- Iterate over source image

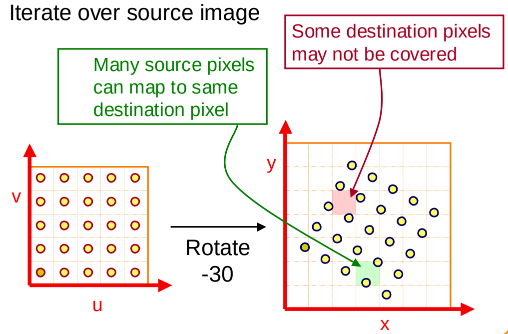
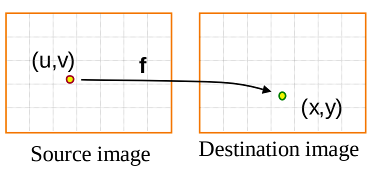


Image Warping Implementation II

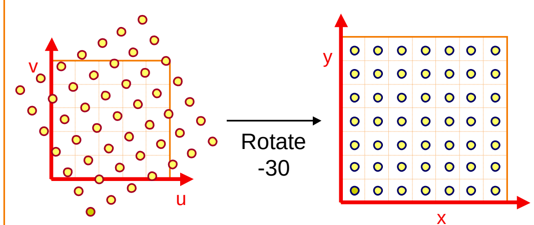
- Reverse mapping:

```
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);
  }
}
```



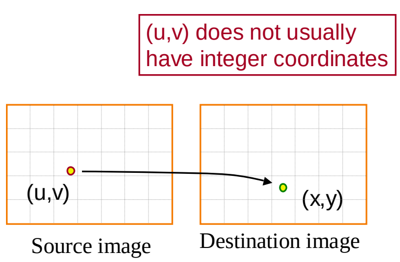
Reverse Mapping

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



Resampling

- Evaluate source image at arbitrary (u,v)



Overview

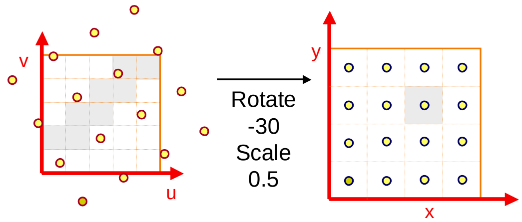
- Mapping
  - Forward
  - Reverse

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

Point Sampling

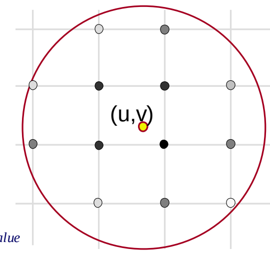
- Take value at closest pixel:
  - int iu = trunc(u+0.5);
  - int iv = trunc(v+0.5);
  - dst(x,y) = src(iu,iv);

This method is simple, but it causes aliasing



Filtering

- Compute weighted sum of pixel neighborhood
  - Weights are normalized values of kernel function
  - Equivalent to convolution at samples



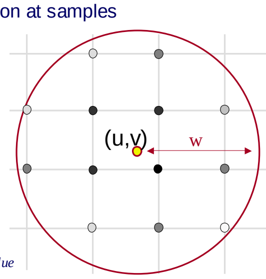
k(i,j) represented by gray value

Filtering

- Compute weighted sum of pixel neighborhood
  - Weights are normalized values of kernel function
  - Equivalent to convolution at samples

```
s = 0;
for (i = -w; i <= w; i++)
  for (j = -w; j <= w; j++)
    s += k(i,j)*I(u+i, v+j);
```

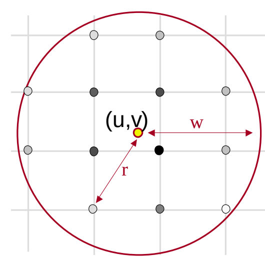
$$\sum k(i,j) = 1$$

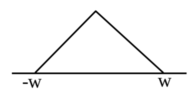


k(i,j) represented by gray value

Triangle Filtering

- Kernel is triangle function



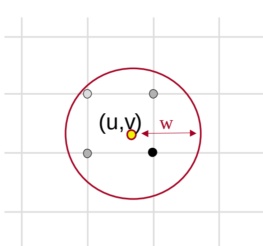


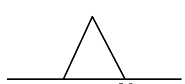
Tent Function

Filter Width = 2

Triangle Filtering

- Kernel is triangle function





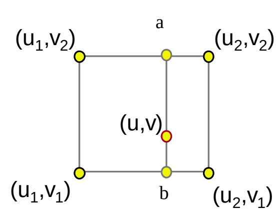
Tent Function

Width of filter affects blurriness

Filter Width = 1

Triangle Filtering (with width = 1)

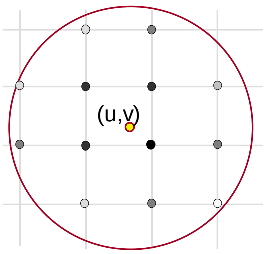
- Bilinearly interpolate four closest pixels
  - a = linear interpolation of src(u_1,v_2) and src(u_2,v_2)
  - b = linear interpolation of src(u_1,v_1) and src(u_2,v_1)
  - dst(x,y) = linear interpolation of “a” and “b”

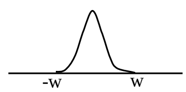


Filter Width = 1

Gaussian Filtering

- Kernel is Gaussian function



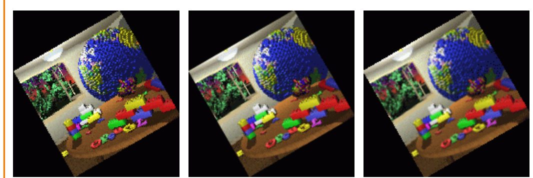


Gaussian Function

Filter Width = 2

Filtering Methods Comparison

- Trade-offs
  - Aliasing versus blurring
  - Computation speed

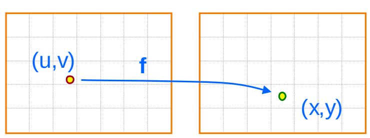


Point | Bilinear | Gaussian

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);
  }
}
```



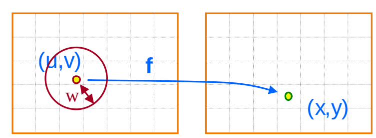
Source image

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);
  }
}
```



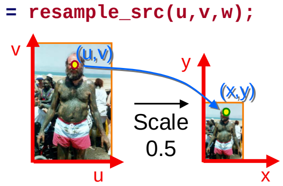
Source image

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);
  }
}
```



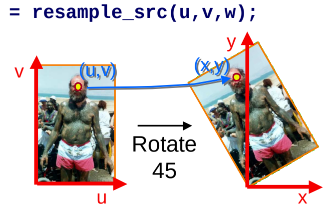
Example: Rotate

- 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);
  }
}
```

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

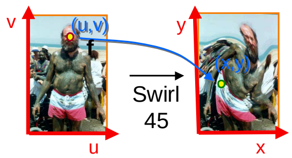
$y = u\sin\Theta + v\cos\Theta$



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);
  }
}
```



Example: Fun

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

Overview: combining images

- Image compositing
  - Blue-screen mattes
  - Alpha channel
  - Porter-Duff compositing algebra
- Image morphing
  - Specifying correspondences
  - Warping
  - Blending

Even CG folks Can Win an Oscar

Smith Duff Catmull Porter

Image Compositing

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

Blue-Screen Matting

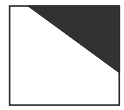
- Composite foreground and background images
  - Create background image
  - Create foreground image with blue background
  - Insert non-blue foreground pixels into background

Problem: no partial coverage!

Alpha Channel

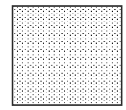
- Encodes pixel coverage information
  - $\alpha = 0$: no coverage (or transparent)
  - $\alpha = 1$: full coverage (or opaque)
  - $0 < \alpha < 1$: partial coverage (or semi-transparent)

- Example: $\alpha = 0.3$



Partial Coverage

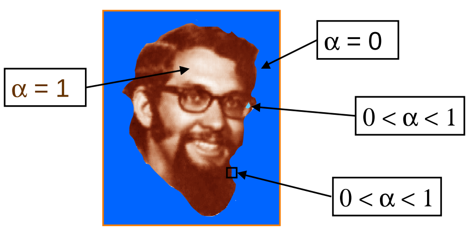
or



Semi-Transparent

Compositing with Alpha

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

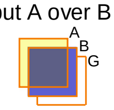


Pixels with Alpha

- Alpha channel convention:
  - (r, g, b, α) represents a pixel that is α covered by the color $C = (r/\alpha, g/\alpha, b/\alpha)$
    - » Color comonents are premultiplied by α
    - » 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

Semi-Transparent Objects

- Suppose we put A over B over background G



  - How much of B is blocked by A?

α_A

  - How much of B shows through A

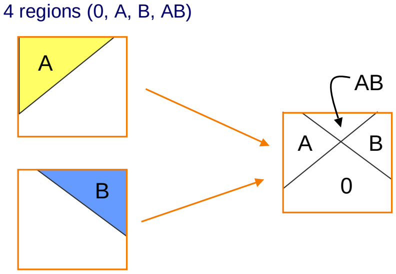
$(1-\alpha_A)$

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

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

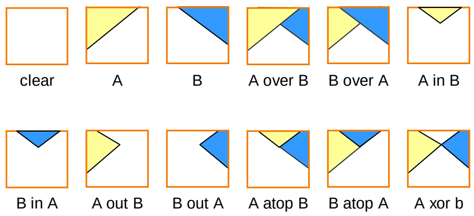
Opaque Objects

- How do we combine 2 partially covered pixels?
  - 3 possible colors (0, A, B)
  - 4 regions (0, A, B, AB)



Composition Algebra

- 12 reasonable combinations



Porter & Duff `84

Example: C = A Over B

- For colors that are <u>not</u> premultiplied:
  - $C = \alpha_A A + (1-\alpha_A)\ \alpha_B B$
  - $\alpha = \alpha_A + (1-\alpha_A)\ \alpha_B$
- For colors that <u>are</u> premultiplied:
  - $C' = A' + (1-\alpha_A)\ B'$
  - $\alpha = \alpha_A + (1-\alpha_A)\ \alpha_B$

A over B

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

Image Composition Example

Jurassic Park

Overview

- Image compositing
  - Blue-screen mattes
  - Alpha channel
  - Porter-Duff compositing algebra
- Image morphing
  - Specifying correspondences
  - Warping
  - Blending

Image Morphing

- Animate transition between two images

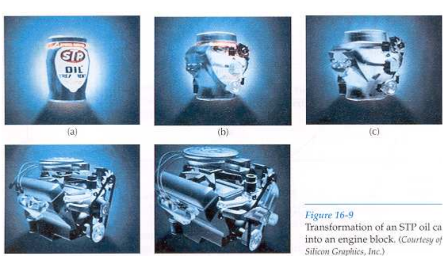


Figure 16-9
Transformation of an STP oil ca[n] into an engine block. *(Courtesy of Silicon Graphics, Inc.)*

H&B Figure 16.9

Cross-Dissolving

- Blend images with "over" operator
  - alpha of bottom image is 1.0
  - alpha of top image varies from 0.0 to 1.0

$$\text{blend}(i,j) = (1-t)\ \text{src}(i,j) + t\ \text{dst}(i,j) \quad (0 \le t \le 1)$$

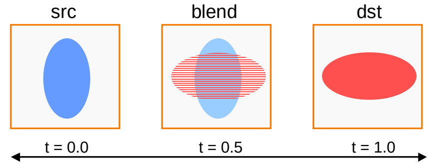


Image Morphing

- Combines warping and cross-dissolving

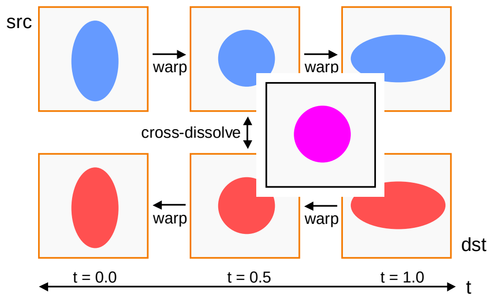
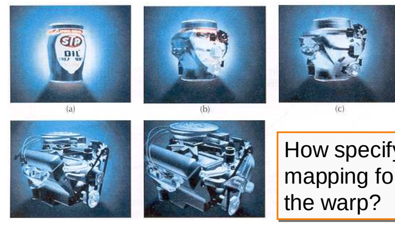


Image Morphing

- The warping step is the hard one
  - Aim to align features in images

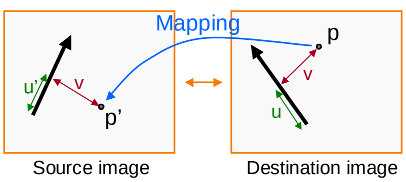


How specify mapping for the warp?

H&B Figure 16.9

Feature-Based Warping

- Beier & Neeley use pairs of lines to specify warp
  - Given p in dst image, where is p' in source image?



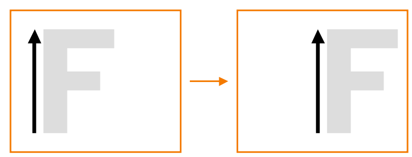
u is a fraction

v is a length (in pixels)

Beier & Neeley
SIGGRAPH 92

Warping with One Line Pair

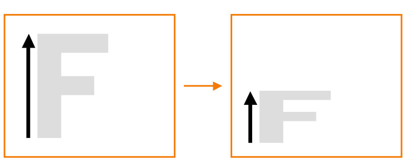
- What happens to the “F”?



Translation!

Warping with One Line Pair

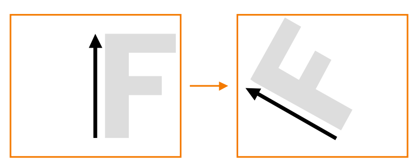
- What happens to the “F”?



Scale!

Warping with One Line Pair

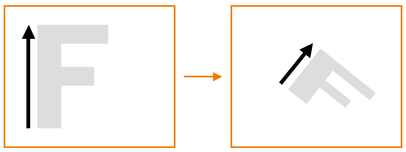
- What happens to the “F”?



Rotation!

Warping with One Line Pair

- What happens to the “F”?

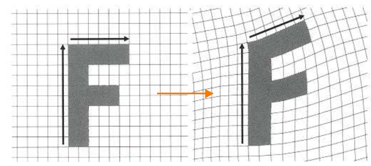


In general, similarity transformations

What types of transformations can't be specified?

Warping with Multiple Line Pairs

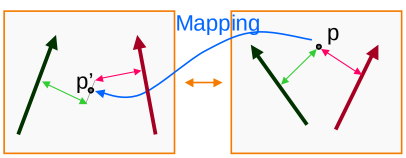
- Use weighted combination of points defined by each pair of corresponding lines



Beier & Neeley, Figure 4

Warping with Multiple Line Pairs

- Use weighted combination of points defined by each pair of corresponding lines



Source image   Destination image

p’ is a weighted average

Weighting Effect of Each Line Pair

- To weight the contribution of each line pair, Beier & Neeley use:

$$weight[i] = \left(\frac{length[i]^p}{a + dist[i]}\right)^b$$

Where:

- *length[i]* is the length of L[i]
- *dist[i]* is the distance from X to L[i]
- *a, b, p* are constants that control the warp

Warping Pseudocode

```
WarpImage(Image, L'[…], L[…])
begin
    foreach destination pixel p do
        psum = (0,0)
        wsum = 0
        foreach line L[i] in destination do
            p'[i] = p transformed by (L[i],L'[i])
            psum = psum + p'[i] * weight[i]
            wsum += weight[i]
        end
        p' = psum / wsum
        Result(p) = Image(p')
    end
end
```

Morphing Pseudocode

```
GenerateAnimation(Image_0, L_0[…], Image_1, L_1[…])
begin
    foreach intermediate frame time t do
        for i = 1 to number of line pairs do
            L[i] = line t-th of the way from L_0 [i] to L_1 [i]
        end
        Warp_0 = WarpImage(Image_0, L_0, L)
        Warp_1 = WarpImage(Image_1, L_1, L)
        foreach pixel p in FinalImage do
            Result(p) = (1-t) Warp_0 + t Warp_1

    end
end
```

Beier & Neeley Example

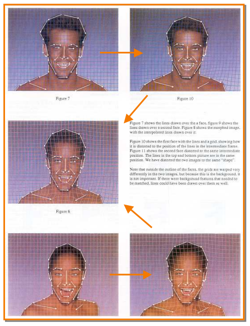
Image$_0$

Warp$_0$

Result

Image$_1$

Warp$_1$



Beier & Neeley Example

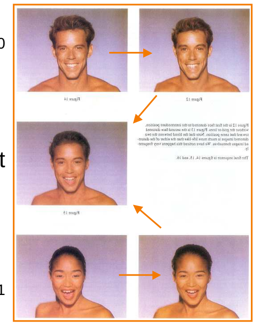
Image$_0$

Warp$_0$

Result

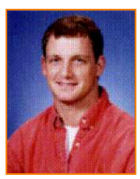
Image$_1$

Warp$_1$



CS426 Examples

CS426 Class, Fall98

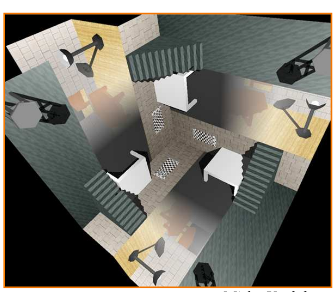


Eric Schmidt, Fall00

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

Next Time: 3D Rendering



Misha Kazhdan,
CS426, Fall99