

# Image Filtering, Warping, and Morphing

Michael Kazhdan

(601.457/657)

### **Outline**



- Image Filtering
- Image Warping
- Image Morphing

# **Image Filtering**



- What about the case when the modification that we would like to make to a pixel depends on the pixels around it?
  - Blurring
  - Edge Detection
  - Etc.

### **Multi-Pixel Operations**



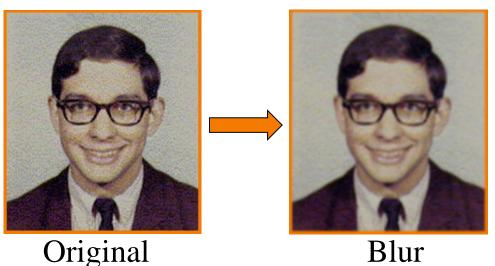
#### Stationary/Local Filtering

#### A general approach is to:

- 1. Define a *mask* of weights telling us how values at adjacent pixels should be combined to generate the new value.
- 2. Apply the (same) mask at every pixel.\*



- To blur across pixels, define a mask:
  - Whose entries sum to one
  - Whose value is larger near the center of the mask
  - Whose values are non-negative



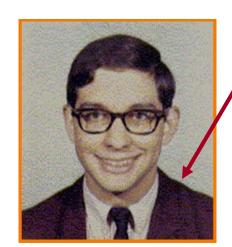
Filter = 
$$\begin{bmatrix} 1/& 2/& 1/\\ /16 & /16 & /16\\ 2/& 4/& 2/\\ /16 & /16 & /16\\ 1/& 2/& 1/\\ /16 & /16 & /16 \end{bmatrix}$$



Pixel(x,y): red = 36

green = 36

blue = 0



Original

Filter = 
$$\begin{bmatrix} 1/& 2/& 1/\\ /16 & /16 & /16\\ 2/& 4/& 2/\\ /16 & /16 & /16\\ 1/& 2/& 1/\\ /16 & /16 & /16 \end{bmatrix}$$



Filter =  $\begin{bmatrix} 1/& 2/& 1/\\ /16 & /16 & /16\\ 2/& 4/& 2/\\ /16 & /16 & /16\\ 1/& 2/& 1/\\ /16 & /16 & /16 \end{bmatrix}$ 

Pixel(x,y): red	= 36
green	= 36
blue	= 0

X - 1 X X + 1

Y - 1	36	109	146
Υ	32	36	109
Y + 1	32	36	73

Pixel(x,y).red and its red neighbors

The second second					
100 CO (100 A) (100 A)					
173000 CALCOLO					
F P50000000		(C) PS(I) (A) (A)	**************************************		
		Contract of the Contract of th	100000		
100000000000000000000000000000000000000	CONTRACTOR OF THE PARTY OF THE	The second second			
0.0000000000000000000000000000000000000	CONTRACTOR OF THE PARTY OF THE				
To a 100 CO	THE SHOOT			200000000000000000000000000000000000000	
0.0000000000000000000000000000000000000	DOCUMENT.			10000000	
F SEA SEE SEE	SUCCESSORY			CO COLUMN	
F (50) TO (120)	SUMP.			-	
A MARCHAEL STATE OF THE SECOND STATE OF THE SE	TOTAL CO.	ALC: UNKNOWN		10000	
0.0000000000000000000000000000000000000		1,000,000	March 1		
C1000000000000000000000000000000000000	2000	THE RESERVE	MODERN		
100000000000000000000000000000000000000		10000000000	DOMESTIC .	100	
100 700 700 700 700	1000	400000000	DOMESTIC STATE		
100 March 100 Ma	2000	The second second	96000000000		
Carlotte Contractor	306			100	
- COLUMN TO THE REAL PROPERTY OF THE PROPERTY OF THE PROPERTY OF THE PROPERTY OF THE PROPERTY	DESCRIPTION OF THE PERSON NAMED IN			396	
100000000000000000000000000000000000000	NCD40 100000				
700000	1000			400	
100 B C 100 B C	100	Contract Con	CONTRACTOR OF THE PARTY.	100	
CHICAGO (1000)	COLUMN TO THE		A STATE OF THE PARTY.	OR THE STREET	
\$156 S01 S001	100 100 100 100				
- 100 march 200 march	100 (0)	ALCOHOL: N		100000	
200 mm 600 fgs	100000000000000000000000000000000000000	SAME OF THE PARTY OF	DESCRIPTION OF	A 100 CO.	
0.0000000000000000000000000000000000000	2010/05/2015	DESCRIPTION AND	VICEOTORICS AN	0.00000	F 128 Co Co.
BODINGS (\$100)	STREET,	DESIGNATION ACCORD	VIII (100 (200 ))	All Bridge	CONTRACTOR OF THE PARTY OF THE
70700 P. Sept. 100	CONTRACTOR	COMMON SERVICES	THE RESIDENCE OF	25 JOSEPH 19	
200000000000000000000000000000000000000	Design Control	-	Contract of the	0.00	
100 March 200 M	DOORS ON	STORY AND VALUE	Marie Control of the	22-410-410	
1 A 18 TO 18 TO 18	CONTRACTOR OF STREET	200			
7 10 20 10 10 10 10 10 10 10 10 10 10 10 10 10	MOST WALL	SOURCE THE R. P. LEWIS CO., LANSING		<b>-037/00</b> 0	
TO SHARE THE PARTY OF THE PARTY	THE RESERVE			1000011000	
STREET, STREET	190400100	ACCRECATION AND ADDRESS OF	ecotores assa	#GOODWICK.	
EXCEPTION 120	STATE OF THE PARTY		ARREST TO SEC.		
2012/09/2012/2012	MANUFACTURE CO.	CORP. TOWN	#E.SB0077		NAME OF TAXABLE PARTY.
600 Sept. 100 Se	THE RESERVE TO SERVE THE PARTY OF THE PARTY	14-130 February	SPECIAL VILLEGE	ALTONOO CA	THE RESERVE OF THE PERSON NAMED IN
DELEMENT OF THE	THE WAY STORY	SACTOR AND ADDRESS.	150 1 100	1000000	4000000
100 mg (100 mg	THE PARTY NAMED IN	MATERIAL PROPERTY.	PERSONAL PROPERTY AND ADDRESS.	1000000	ALC: UNKNOWN
TAX TO SHAP WAS	Company of the last	THE RESERVE AND ADDRESS OF	HERMAN LOSS OF	100	ACCRECATION OF THE PERSON.
F-493236-633	1000000	THE PERSON NAMED IN	ADDRESS OF THE PARTY OF	THE REAL PROPERTY.	
A DOLLARS CHEEKS	SUCCESS OF REAL PROPERTY.	2000000	HE SHADE	St. March	CATALOG CO.
H-3429/709/503		THE RESIDENCE OF THE PERSON NAMED IN COLUMN 1		0.05000000	7000000
ATHER (\$20,000)		TO STREET, SQUARE,	BOARD CF . F .	P COST CANA	
DECEMBER OF THE PARTY.	THE RESERVE OF THE PERSON NAMED IN	Control State of	1000		THE RESERVE OF THE PERSON NAMED IN
100000000000000000000000000000000000000	STATE OF THE REAL PROPERTY.	ASSESSMENT OF THE PARTY OF THE	100.00		
ASSESSED FOR THE PARTY OF THE P	THE RESERVE	THE RESERVE TO SHARE	CONTRACTOR OF THE PARTY OF THE		THE RESERVE
Company of the Compan			TO SECOND SECOND		
The second second		100000	THE REAL PROPERTY.		
A CONTRACTOR OF THE PARTY OF TH	CARL PROPERTY.				
Total Control of the	CONTRACTOR OF THE PERSON	MARKET STATE	10000		
The second second second	A STATE OF THE PARTY OF THE PAR				
TOTAL STATE OF	THE RESERVE OF THE PERSON NAMED IN	Ministration of the Contract o	1000		
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	The second second	COLUMN TO SERVICE	ALCOHOL: U	1000	
The second second	STATE OF THE PARTY	Committee of the Commit	ALC: UNKNOWN		

Original



New	value	for I	Pixel(	x,y	).r	ed =

X + 1

$$(36 * 1/16) + (109 * 2/16) + (146 * 1/16)$$

$$(32 * 2/16) + (36 * 4/16) + (109 * 2/16)$$

$$(32 * 1/16) + (36 * 2/16) + (73 * 1/16)$$



Original

Y - 1	36	109	146
Υ	32	36	109
Y + 1	32	36	73

X - 1

Filter = 
$$\begin{bmatrix} 1/& 2/& 1/\\ /16 & /16 & /16\\ 2/& 4/& 2/\\ /16 & /16 & /16\\ 1/& 2/& 1/\\ /16 & /16 & /16 \end{bmatrix}$$



#### New value for Pixel(x,y).red = 62.69



Original

36	109	146
32	36	109
32	36	73

X X + 1

X - 1

Y - 1

Y + 1

Pixel(x,y).red and its red neighbors

Filter = 
$$\begin{bmatrix} 1/& 2/& 1/\\ /16 & /16 & /16\\ 2/& 4/& 2/\\ /16 & /16 & /16\\ 1/& 2/& 1/\\ /16 & /16 & /16 \end{bmatrix}$$



#### New value for Pixel(x,y).red = 63



Original



Blur

$$\begin{bmatrix} 1/&2/&1/\\/16&/16&/16\\2/&4/&2/\\/16&/16&/16\\1/&2/&1/\\16&/16&/16 \end{bmatrix}$$

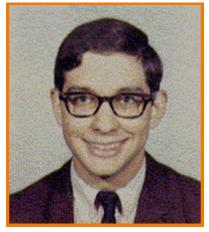


- Repeat for each color channel of each pixel.
- Keep source and destination separate to avoid drift.
- For boundary pixels, not all neighbors are used:
  - » Normalize the mask so the values sum to one, or
  - » Assume that the exterior values are black, or
  - » Assume the exterior values can be obtained by reflecting the image across the boundary, or
  - » Assume...



- Masks can have arbitrary size:
  - Can expand smaller masks by zero-padding

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} / 16 \iff \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} / 16$$



Original



Narrow Blur



- Masks can have arbitrary size:
  - Can expand smaller masks by zero-padding
  - Can use more non-zero entries to get a wider blur

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} / 16$$



**Original** 



Narrow Blur



Wide Blur



A general way for defining the entries of an  $n \times n$  mask is to use the values of a Gaussian:

GaussianMask[i][j] 
$$\sim e^{\frac{-(i-r)^2+(j-r)^2}{4r^2}}$$
  
 $i, j \in [0,2r]$ 

- r is the (integer) mask radius
- n = 2r + 1 is the width
- $0 \le i \le 2r$  is the horizontal position in the mask
- $0 \le j \le 2r$  is the vertical position in the mask
- Don't forget to normalize!

#### Note:

- The center of the mask is at index (r,r).
- The Gaussian itself has standard deviation  $\sigma = \sqrt{2} \cdot r$



An edge is where the image is far from constant:

⇒ The difference between the value at the pixel and the average value of neighboring pixels is large (in absolute value)

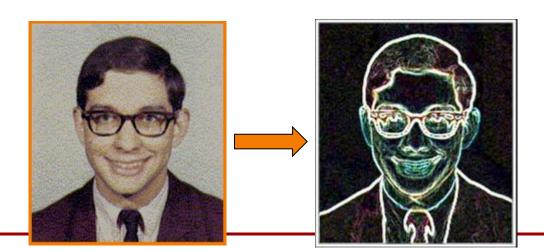


#### An edge is where the image is far from constant:

⇒ The difference between the value at the pixel and the average value of neighboring pixels is large (in absolute value)

#### Define a mask whose:

- Entries sum to zero
- Value is one at the center pixel



Filter = 
$$\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



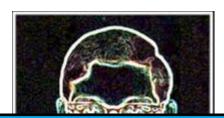
#### An edge is where the image is far from constant:

⇒ The difference between the value at the pixel and the average value of neighboring pixels is large (in absolute value)

#### Define a mask whose:

- Entries sum to zero
- Value is one at the center pixel





 $\begin{bmatrix} -1 & -1 & -1 \end{bmatrix}$ 

Pixels with large absolute values correspond to edges:

- Positive values: "upper" edges
- Negative values: "lower" edges



Pixel(x,y): red = 36

green = 36

blue = 0



Original

Filter = 
$$\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



Filter =  $\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ 

Pixel(x,y): red = 36 = 36 green blue = 0

X - 1 Χ X + 1

36 109 146 36 32 109 Y + 1 32 36 73

> Pixel(x,y).red and its red neighbors

E 0000000					
10 CONTRACTOR					
1,750,000,000		The second second	OR RESIDENCE		
	Secretary and the second		1000		
-702755	220000000000000000000000000000000000000		CARL TO SERVICE		
SCHOOL STATE	HORSON .		The second		
E 100 CO TO	DOCUMENT.			1000000	
1.000.000.00	ALCOHOL:			CA STATE	
1022000	DESCRIPTION OF THE PERSON OF T			1000000	
000000000000000000000000000000000000000	THE RESERVE OF THE PERSON NAMED IN COLUMN TWO IS NOT THE PERSON NAMED IN COLUMN TWO IS NAMED IN COLUMN TW	ACCUPANT OF THE PARTY.	Market Ave.	-	
\$2500 MARCH	ACCUSED TO SECURE	<b>PETTION INC.</b>	ACCOUNT .		
100000000000000000000000000000000000000		CONTRACTOR AND	COMMODIAL.	1920	
100000000000000000000000000000000000000	4000		Approximation in	- C - C - C - C - C - C - C - C - C - C	
10 W W W W W	CORNE CONTRACTOR		こしてかかれている		
622,379	ATTACK MELTING			396	
1000000	2012			2000	
14000	COLUMN TWO IS NOT THE OWNER.			696	
ACCORDING TO SECOND	THE RESERVE	100000000	400000	a 1000	
2 22 20 20 20	100 E. V. C. C. C.	10000		- 200	
	AND DESCRIPTION OF THE PERSON		200	100000	
C0000000000000000000000000000000000000	CONTRACTOR IN CO.	HEREN AND	SECTION AND IN	4 500000	THE REAL PROPERTY.
2000000000	ALC: NUMBER OF	100007 4800	VANUESSI &	ALCOHOL:	
0.0000000000000000000000000000000000000	CHANGE BY SEC.	NAME OF STREET	2000000	28° 5250 to 2	
9200000000	100 CO (CO)	1000500	-	2000	
100 March 1997	DESCRIPTION OF THE PARTY OF THE	TO A STORY OF THE	State of the last	25 of 12" (1015).	
- AUGUSTAN	25000 PR 10000	P20 (400) (500)			
\$1000EA000	COLUMN BOWL		500000	A STATE OF THE PARTY OF THE PAR	
100000000000000000000000000000000000000	STATE OF THE PARTY	CONTRACTOR OF THE PARTY.	6000PA 551	ACCOUNTS TO	
100000000000000000000000000000000000000	SERVICE AND ADDRESS OF	200000000000000000000000000000000000000	ARE THE		100000
200000000000000000000000000000000000000	HONORED TO THE	STATE OF THE PARTY.	HE (2000)	ECONOMICON.	ASSESSMENT OF THE PERSON NAMED IN
100000000000000000000000000000000000000	CTITICO CONSIA	A SECTION OF		TO SECOND	CONTRACTOR OF THE PARTY.
10001100000	COMPANIES OF THE PARTY OF THE P	STATE OF THE STATE OF	6520 NO. 153		
Participation of the	CALL CONTRACTOR AND ADDRESS.	THE HELICISE	1000 LOG		
100000000000000000000000000000000000000	2000000	- COMMERCIAL	BOY (100)		
E175329A005557	DOMEST NO.	0.0000000000000000000000000000000000000	100000	ALC: NO.	100000000000000000000000000000000000000
2005000000000		TOWNS OF	ALC: UNKNOWN A	1000	-
1900000000000		0000Eart (6	200		A COLUMN
1000000000	STATE OF THE PARTY	SECTION 1	200		
(800 PT	ACCUSED TO THE	1000	4000000		
	STATE OF THE PARTY NAMED IN	ALC: N	1000000		
De la Colonia	COMPANIES -	SCHOOL STREET	100000		
100	THE RESERVE OF THE PERSON NAMED IN	AND THE REAL PROPERTY.	10000		
ALC: UNKNOWN	ACCRECATE VALUE OF	STATE OF THE PARTY.	1000		
	AND DESCRIPTION OF THE PARTY.	ALC: UNKNOWN	0.000	THE RESERVE	

Υ

Original



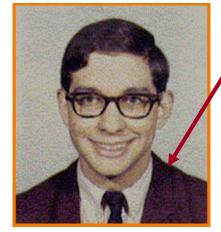
New value	for	Pixel	(x,y)	.red =
-----------	-----	-------	-------	--------

X X + 1

$$(36 * -1/8) + (109 * -1/8) + (146 * -1/8)$$

$$(32 * -1/8) + (36 * 1) + (109 * -1/8)$$

$$(32 * -1/8) + (36 * -1/8) + (73 * -1/8)$$



Original

Y - 1	36	109	146
Y	32	36	109
<b>/</b> + 1	32	36	73

X - 1

Filter = 
$$\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



#### New value for Pixel(x,y).red = -285/8



Original

X - 1	X	X + 1

Y - 1

Y + 1

36	109	146
32	36	109
32	36	73

Pixel(x,y).red and its red neighbors

Filter = 
$$\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



#### New value for Pixel(x,y).red = -35.625



Original



**Detected Edges** 

Filter = 
$$\frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

#### Note:

Output values are not colors, so need to find a way to remap for visualization.

### **Outline**

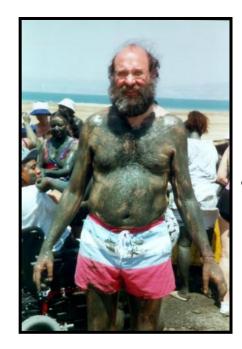


- Image Filtering
- Image Warping
- Image Morphing

# **Image Warping**



- Move pixels of image
  - Mapping
  - Resampling



Source image



Warp

Destination image

#### **Overview**

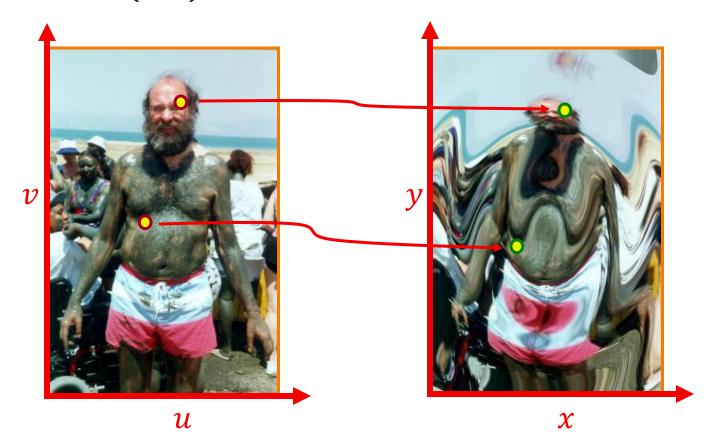


- Mapping
  - Forward
  - Inverse
- Resampling
  - Point sampling
  - Triangle filter
  - Gaussian filter

# **Mapping**



- Define transformation
  - Describe the destination  $(x, y) = \Phi(u, v)$  for every location (u, v) in the source

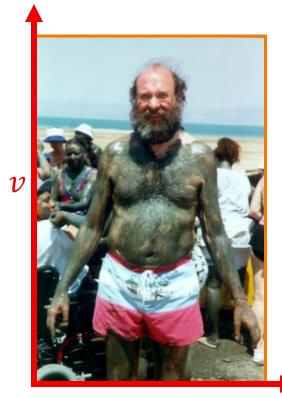


# **Example Mappings**

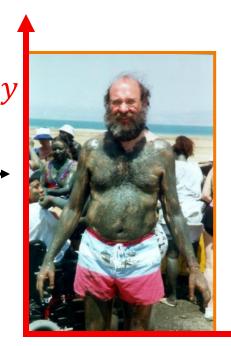


• Scale by  $\sigma$ :

$$\circ \ \Phi(u,v) = (\sigma u, \sigma v)$$



Scale  $\sigma = 0.8$ 

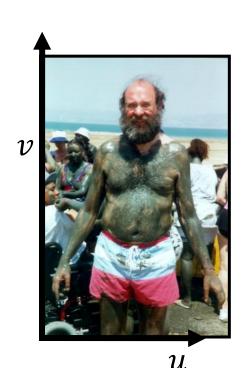


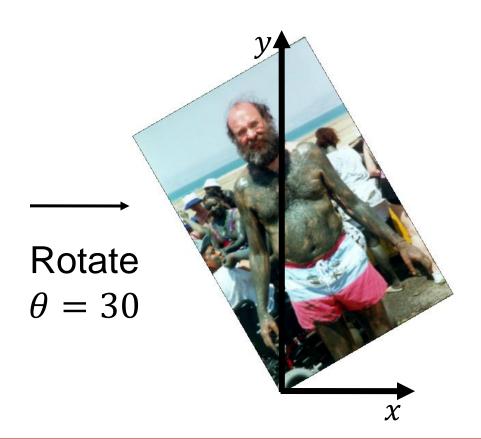
 $\chi$ 

### **Example Mappings**



- Rotate by  $\theta$  degrees:
  - $\Phi(u,v) = (u\cos\theta v\sin\theta, u\sin\theta + v\cos\theta)$

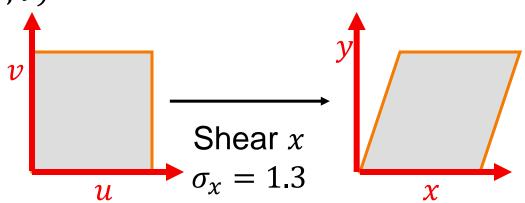




# **Example Mappings**



• Shear in x by  $\sigma_x$ :

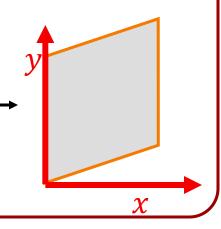


Shear y

• Shear in y by  $\sigma_y$ :

$$\Phi(u,v) = (u,v + \sigma_y \cdot u)$$

u



# **Other Mappings**



• Any function of u and v:

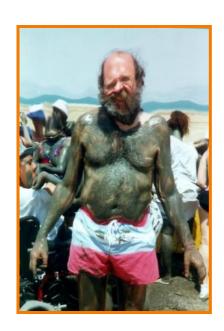
$$\circ \Phi(u,v) = \cdots$$



Fish-eye



"Swirl"



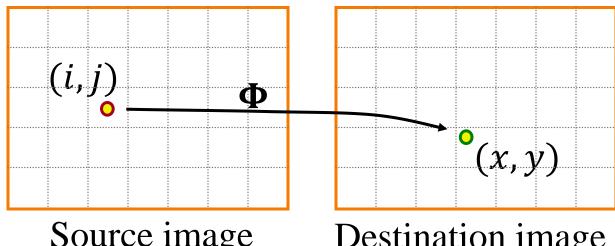
"Rain"

# Image Warping Implementation I



Forward mapping:

```
for( j=0 ; j<srcHeight ; j++ )</pre>
  for( i=0 ; i<srcWidth ; i++ )</pre>
     (x,y) = \Phi(i,j);
    dst(x,y) = src(i,j);
```



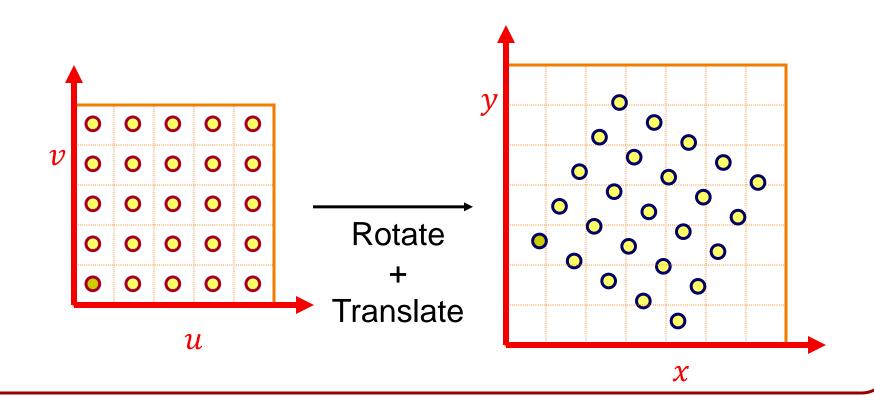
Source image

Destination image

### **Forward Mapping**



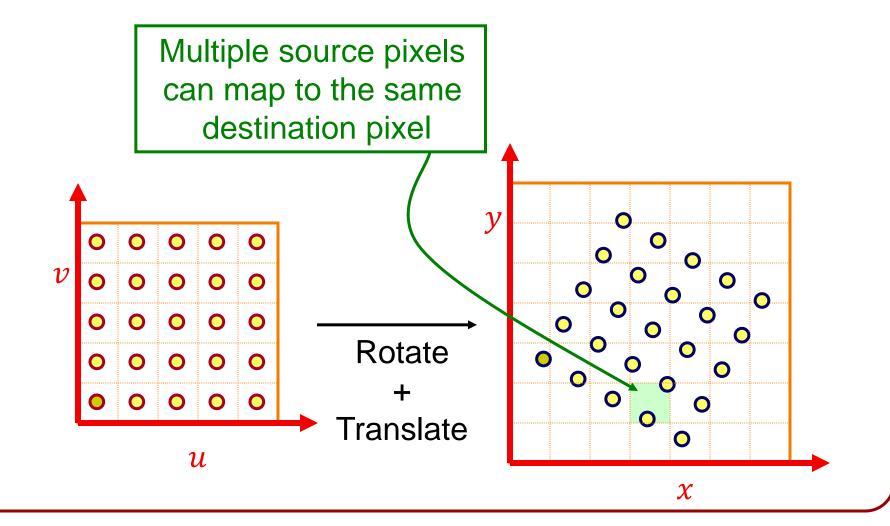
Iterate over source image



# Forward Mapping – BAD!



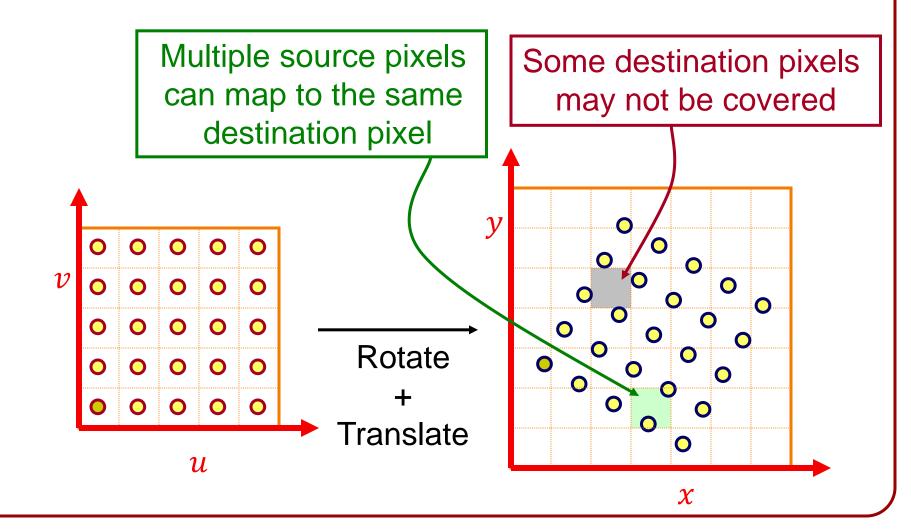
Iterate over source image



# Forward Mapping – BAD!



Iterate over source image

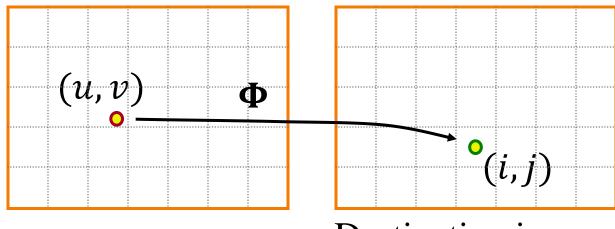


# Image Warping Implementation II



Inverse mapping:

```
for( j=0 ; j<dstHeight ; j++ ) for( i=0 ; i<dstWidth ; i++ )  (u,v) = \Phi^{-1}(i,j);   dst(i,j) = src(u,v);
```



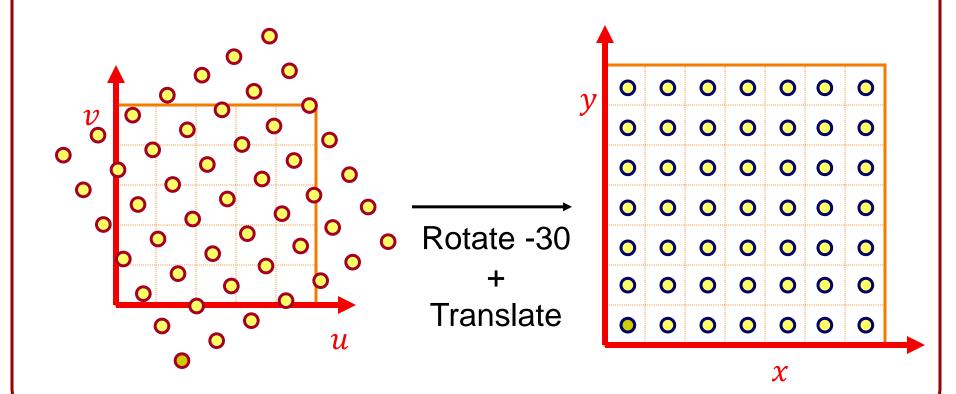
Source image

Destination image

# Reverse Mapping – GOOD!



- Iterate over destination image
  - Must resample source

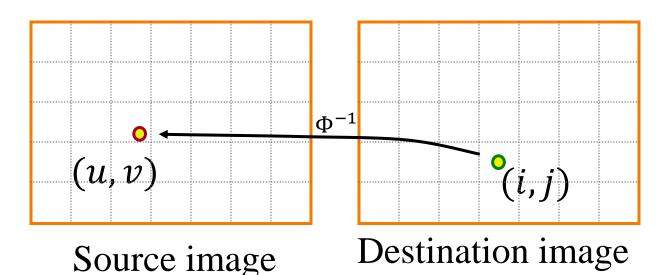


### Resampling



• Evaluate source image at  $(u, v) = \Phi^{-1}(i, j)$ 

(u, v) does not usually have integer coordinates



#### **Overview**



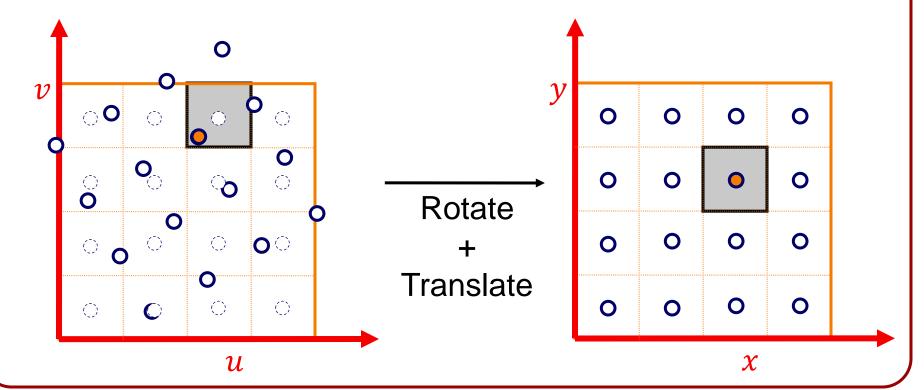
- Mapping
  - Forward
  - Inverse
- Resampling
  - Nearest Point Sampling
  - Bilinear Sampling
  - Gaussian Sampling

## **Nearest Point Sampling**



Take value at closest pixel:

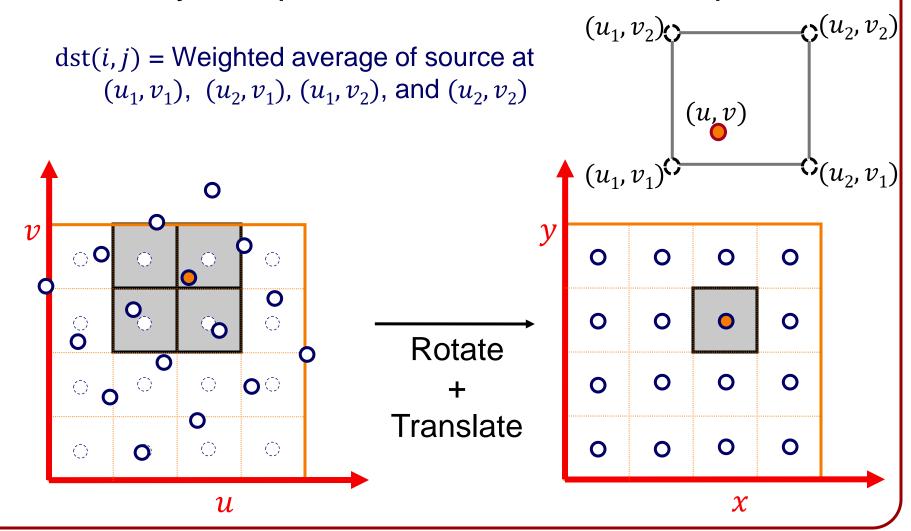
```
int intU = floor(u+0.5);
int intV = floor(v+0.5);
dst(i,j) = src(intU,intV);
```



## **Bilinear Sampling**



Bilinearly interpolate four closest source pixels



### **Linear Sampling**



Linearly interpolate two closest source pixels

```
dst(i) = linear interpolation of u_1 and u_2
```

```
\begin{array}{ccc}
u \\
u_1 & & & \\
0 & \leq du \leq 1
\end{array}
```

```
u = Φ<sup>-1</sup>(i)
u1 = floor(u);
u2 = u1 + 1;
du = u - u1;
dst(i) = src(u1)*(1-du) + src(u2)*du;
```

### **Bilinear Sampling**



Bilinearly interpolate four closest source pixels

```
a = \text{linear interpolation of } \operatorname{src}(u_1, v_1) \text{ and } \operatorname{src}(u_2, v_1)
    b = \text{linear interpolation of } \operatorname{src}(u_1, v_2) \text{ and } \operatorname{src}(u_2, v_2)
    dst(i, j) = linear interpolation of a and b
(\mathbf{u},\mathbf{v}) = \Phi^{-1}(\mathbf{i},\mathbf{j})
u1 = floor(u), u2 = u1 + 1;
v1 = floor(v), v2 = v1 + 1;
du = u - u1;
a = src(u1,v1)*(1-du)
   + src(u2,v1)*(du);
b = src(u1, v2) * (1-du)
   + src(u2, v2) *du;
dv = v - v1;
dst(i,j) = a*(1-dv) + b*dv;
```

```
(u_1, v_2) (u, v_2) (u_2, v_2) (u, v_1) (u, v_1) (u_2, v_1)
```

### **Bilinear Sampling**



Bilinearly interpolate four closest source pixels

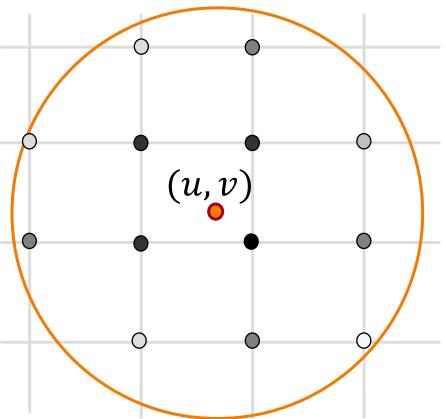
```
a = \text{linear interpolation of } \operatorname{src}(u_1, v_1) \text{ and } \operatorname{src}(u_2, v_1)
    b = \text{linear interpolation of } \operatorname{src}(u_1, v_2) \text{ and } \operatorname{src}(u_2, v_2)
    dst(i, j) = linear interpolation of a and b
(\mathbf{u},\mathbf{v}) = \Phi^{-1}(\mathbf{i},\mathbf{j})
                                                   (u_1, v_2), (u, v_2), (u_2, v_2)
u1 =
             Make sure to test that the pixels
       (u_1, v_1), (u_2, v_2), (u_1, v_2), \text{ and } (u_2, v_1)
                        are within the image.
                                                    \overline{(u_1, v_1)} (u, v_1) (u_2, v_1)
   + src(u2,v1)*( du);
b = src(u1, v2) * (1-du)
   + src(u2, v2) *du;
dv = v - v1;
dst(i,j) = a*(1-dv) + b*dv;
```

## **Gaussian Sampling**



Compute weighted sum of pixel neighborhood:

 The blending weights are the <u>normalized</u> values of a Gaussian function.

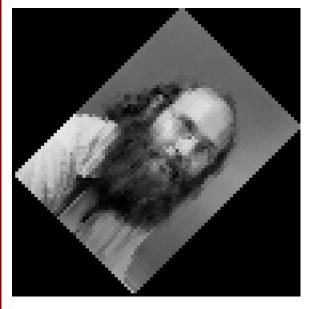


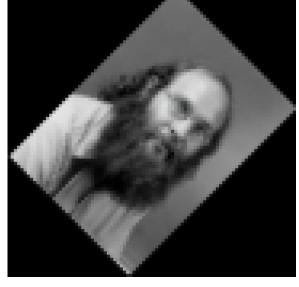
## Filtering Methods Comparison



- Trade-offs
  - Jagged edges versus blurring
  - Computation speed

We'll talk more about trade-offs next time.







**Nearest** 

Bilinear

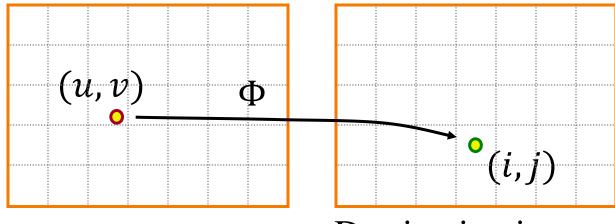
Gaussian

#### **Image Warping Implementation**



Inverse mapping:

```
for( j=0 ; j<dstHeight ; j++ ) for( i=0 ; i<dstWidth ; i++ )  (u,v) = \Phi^{-1}(i,j);  dst(i,j) = resample_src(u,v,r);
```



Source image

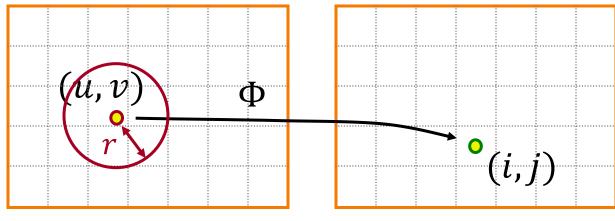
Destination image

### **Image Warping Implementation**



Inverse mapping:

```
for( j=0 ; j<dstHeight ; j++ )
  for( i=0 ; i<dstWidth ; i++ )
    (u,v) = Φ<sup>-1</sup>(i,j);
    dst(i,j) = resample_src(u,v,r);
```



Source image

Destination image

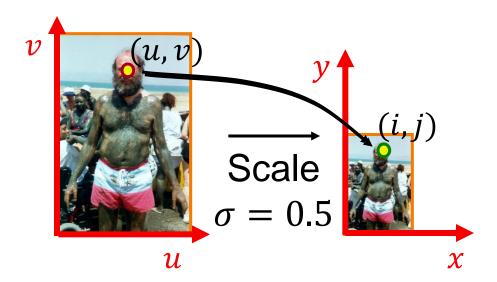
#### **Example: Scale**



```
Scale( src , dst , \sigma ):
```

```
r \cong ?;
for(j=0; j<dstHeight; j++)
for(i=0; i<dstWidth; i++)
(u,v) = (i,j) / \sigma;
dst(i,j) = resample_src(u,v,r);
```

$$r = \frac{1}{\sigma}$$



#### **Example: Rotate**



```
Rotate( src , dst , \theta ):
 r \cong ?;
 for( j=0 ; j<dstHeight ; j++ )</pre>
   for( i=0 ; i<dstWidth ; i++ )</pre>
      (u,v) = (i*cos(-\theta) - j*sin(-\theta)),
                   i*sin(-\theta) + j*cos(-\theta));
      dst(x,y) = resample src(u,v,r); y
                              (u, v)
       r = 1
```

 $\frac{(u,v)}{\text{Rotate}}$   $\theta = 30$ 

IJ.

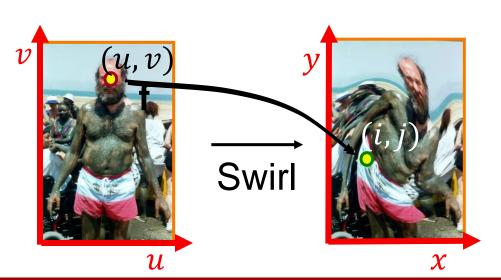
#### **Example:**



#### General( src, dst, $\Phi$ ):

```
 \begin{array}{l} r \cong ?; \\ \\ for(\ j=0\ ;\ j< dst Height\ ;\ j++\ ) \\ \\ for(\ i=0\ ;\ i< dst Width\ ;\ i++\ ) \\ \\ (u,v) = \Phi^{-1}(i,j); \\ \\ dst(i,j) = resample\_src(u,v,r); \\ \end{array}
```

r = ?



#### **Example:**



#### General( src, dst, $\Phi$ ):

```
 \begin{array}{l} r \cong ?; \\ \\ for(\ j=0\ ;\ j< dst Height\ ;\ j++\ ) \\ \\ for(\ i=0\ ;\ i< dst Width\ ;\ i++\ ) \\ \\ (u,v) = \Phi^{-1}(i,j); \\ \\ dst(i,j) = resample\_src(u,v,r); \\ \end{array}
```

Instead of using a fixed radius circle to sample the source, we can:

- 1. Have the radius changes
- 2. Use an ellipse.

For example, the parameters can be determined by looking at the derivative/Jacobean of  $\Phi$ .

#### **Outline**



- Image Filtering
- Image Warping
- Image Morphing



#### Recall:

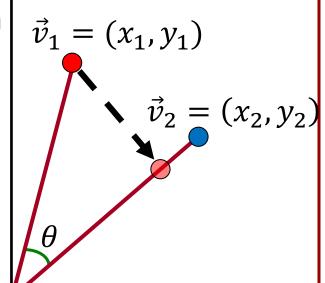
1. For a vector  $\vec{v} = (x, y)$ , its 90° (ccw) rotation is given by:  $\vec{v}^{\perp} = (-y, x)$ 



#### Recall:

- 1. For a vector  $\vec{v} = (x, y)$ , its 90° (ccw) rotation is given by:  $\vec{v}^{\perp} = (-y, x)$
- 2. For two vectors  $\vec{v}_1 = (x_1, y_1)$  and  $\vec{v}_2 = (x_2, y_2)$   $\langle \vec{v}_1, \vec{v}_2 \rangle \equiv x_1 \cdot x_2 + y_1 \cdot y_2 = ||\vec{v}_1|| \cdot ||\vec{v}_2|| \cdot \cos \theta$
- $\Rightarrow$  The signed length of the projection of  $v_1$  onto the line through  $v_2$  is:

$$||v_1|| \cdot \cos \theta = \frac{\langle v_1, v_2 \rangle}{||v_2||}$$





#### Recall:

- 1. For a vector  $\vec{v} = (x, y)$ , its 90° (ccw) rotation is given by:  $\vec{v}^{\perp} = (-y, x)$
- 2. For two vectors  $\vec{v}_1 = (x_1, y_1)$  and  $\vec{v}_2 = (x_2, y_2)$   $\langle \vec{v}_1, \vec{v}_2 \rangle \equiv x_1 \cdot x_2 + y_1 \cdot y_2 = ||\vec{v}_1|| \cdot ||\vec{v}_2|| \cdot \cos \theta$
- 3. The inner-product of two vectors is *isometry-invariant*. If  $\mathbf{R} \in \mathbb{R}^{2 \times 2}$  is a rotation, then:

$$\langle \mathbf{R}(\vec{v}_1), \mathbf{R}(\vec{v}_2) \rangle = \langle \vec{v}_1, \vec{v}_2 \rangle$$

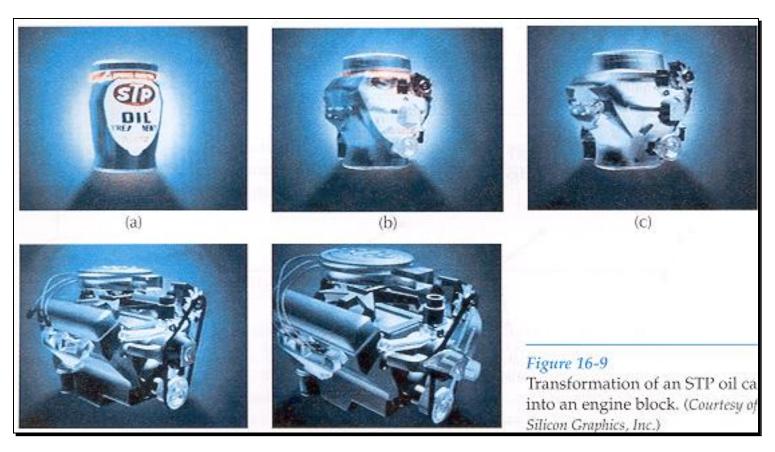
⇒ In particular:

$$\left\langle \vec{v}_1^{\perp}, \vec{v}_2^{\perp} \right\rangle = \left\langle \vec{v}_1, \vec{v}_2 \right\rangle$$

### **Image Morphing**



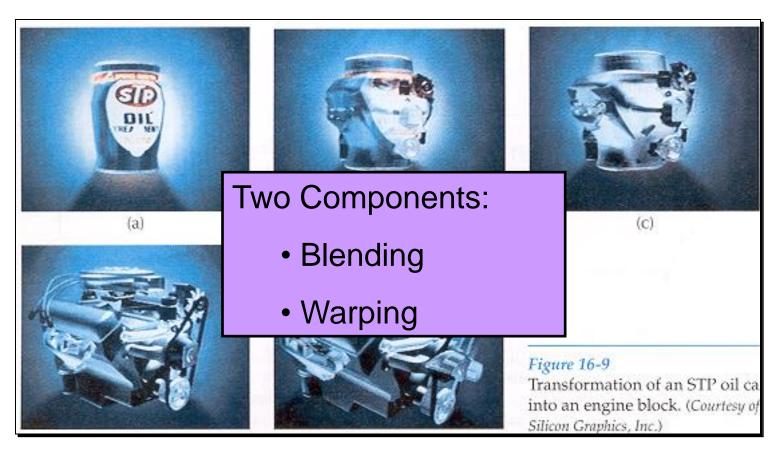
Animate transition between two images



### **Image Morphing**



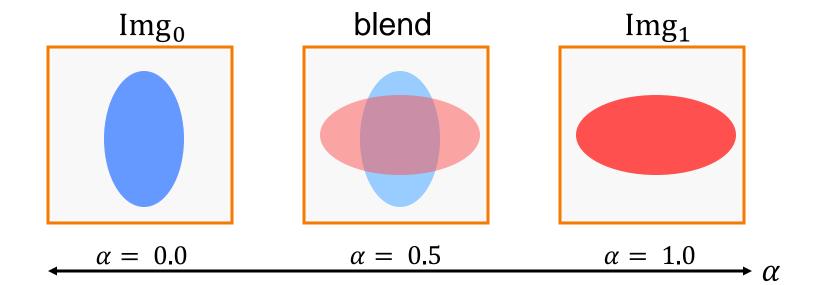
Animate transition between two images



# **Blending**

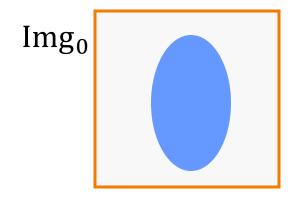


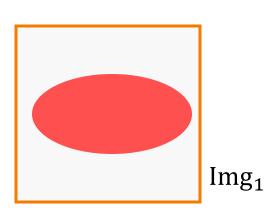
Blend **colors** using an  $\alpha$ -blend ( $\alpha \in [0,1]$ ): blend $(i,j,\alpha) = (1-\alpha) \cdot \operatorname{Img}_0(i,j) + \alpha \cdot \operatorname{Img}_1(i,j)$ 





Deform Img<sub>0</sub> so its **shape** matches that of Img<sub>1</sub>...





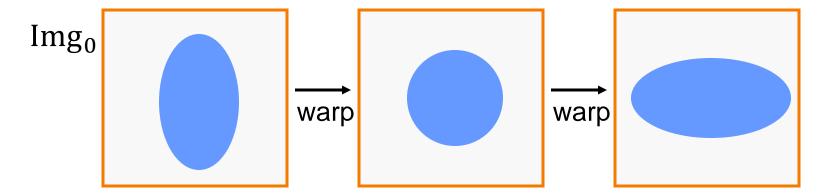
$$\alpha = 0.0$$

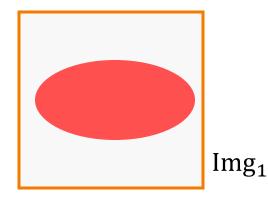
$$\alpha = 0.5$$

$$\alpha = 1.0$$



Deform Img<sub>0</sub> so its **shape** matches that of Img<sub>1</sub>...





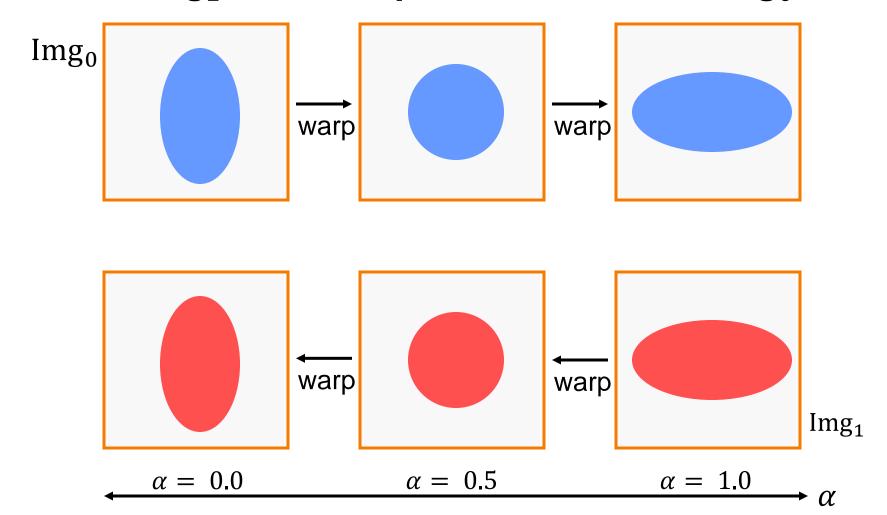
$$\alpha = 0.0$$

$$\alpha = 0.5$$

$$\alpha = 1.0$$



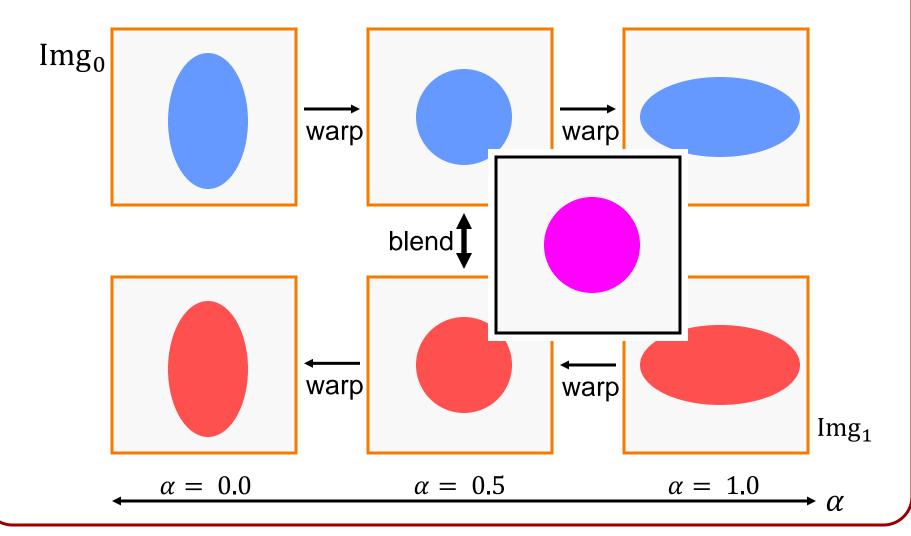
Deform Img<sub>1</sub> so its **shape** matches that of Img<sub>0</sub>...



# **Image Morphing**



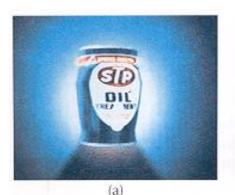
... then blend colors



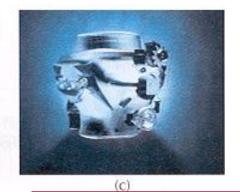
### **Image Morphing**



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











How do we specify the mapping for the warp?

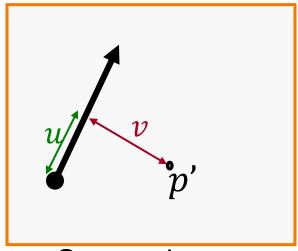
into an engine block. (Courtesy of Silicon Graphics, Inc.)

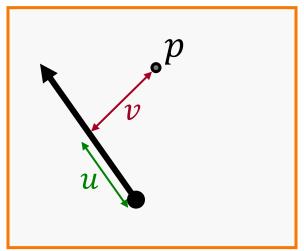
### **Feature-Based Warping**



#### [Beier & Neeley, 1992] use a pair of lines:

- Given p in the destination image, where is p' in the source?
- Describe p relative to the destination line
- Map the description to the source





Source image

Destination image

u is a signed <u>fraction</u> v is a signed <u>length</u> (in pixels)

Beier & Neeley SIGGRAPH 92

# **Feature-Based Warping**

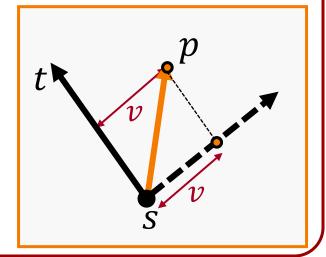


How do we calculate v (perp. pixel distance)?

#### Recall:

 $\frac{\langle \vec{v}_1, \vec{v}_2 \rangle}{\|\vec{v}_2\|}$  is the signed length of the projection of  $\vec{v}_1$  on the line through  $\vec{v}_2$ .  $\vec{v}_1$   $\vec{v}_2$ 

$$v = \frac{\langle p - s, (t - s)^{\perp} \rangle}{\|(t - s)^{\perp}\|}$$



# **Feature-Based Warping**

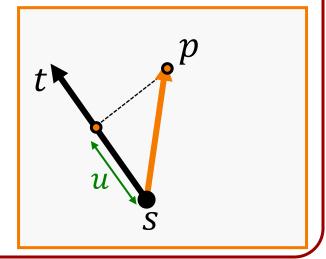


How do we calculate u (parallel fractional distance)?

#### Recall:

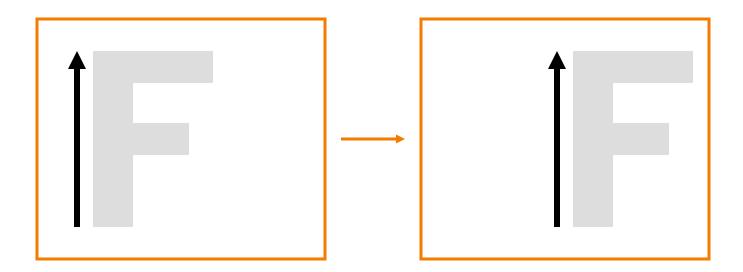
 $\frac{\langle \vec{v}_1, \vec{v}_2 \rangle}{\|\vec{v}_2\|}$  is the signed length of the projection of  $\vec{v}_1$  on the line through  $\vec{v}_2$ .

through 
$$\vec{v}_2$$
.
$$u = \frac{\langle \vec{p} - \vec{s}, \vec{t} - \vec{s} \rangle}{\|t - \vec{s}\|} \cdot \frac{1}{\|t - \vec{s}\|}$$





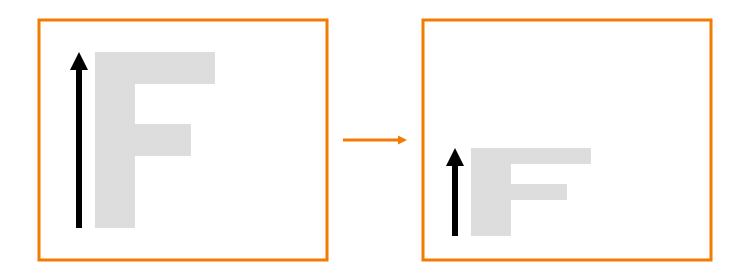
What happens to the "F"?



Translation!



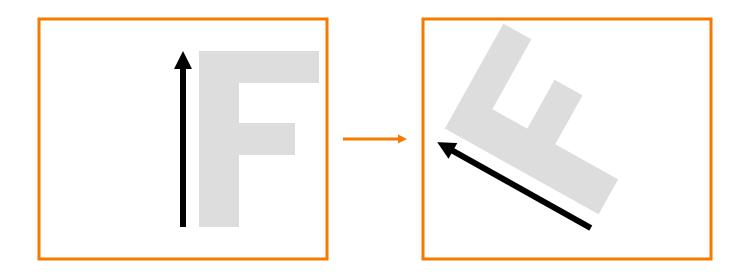
What happens to the "F"?



Non-uniform scale!



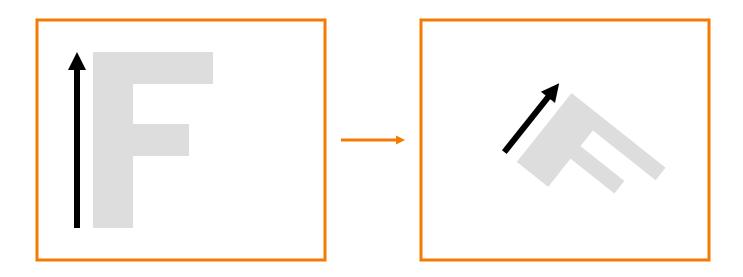
What happens to the "F"?



Rotation!



What happens to the "F"?

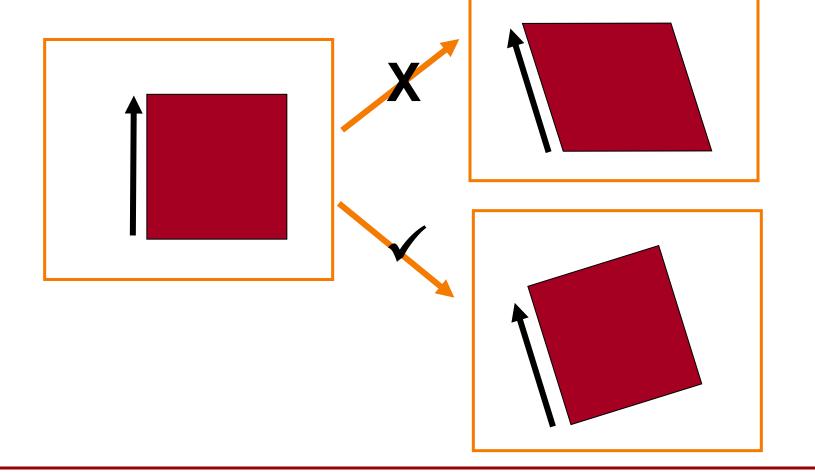


What types of <u>affine</u> transformations can't be specified?



Can't specify arbitrary scales, skews, mirrors,

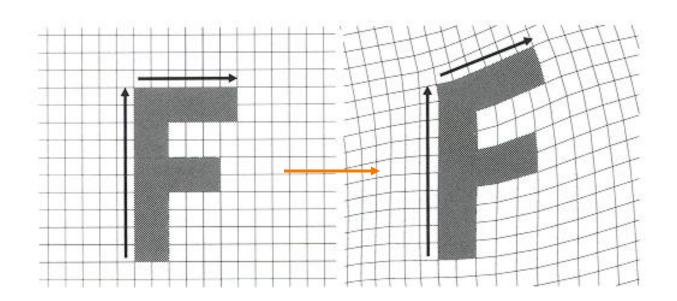
angular changes...



## Warping with Multiple Line Pairs



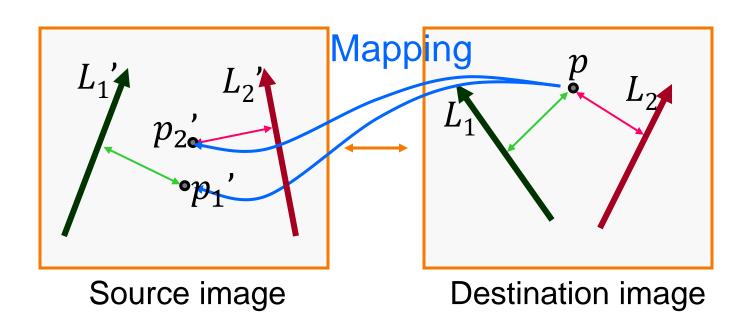
 Use weighted combination of points defined by each pair of corresponding lines



#### Warping with Multiple Line Pairs



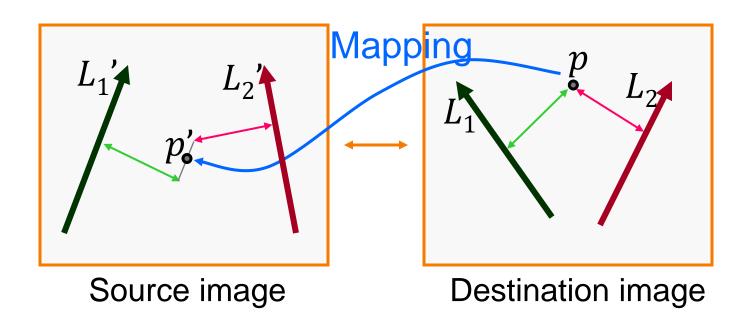
 Use weighted combination of points defined by each pair of corresponding lines



#### Warping with Multiple Line Pairs



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



p' is a weighted average

### Weighting Effect of Each Line Pair



• Given a set of line pairs  $\{L_{in}[0], ..., L_{in}[N]\}$  and  $\{L_{out}[0], ..., L_{out}[N]\}$ , to weight the contribution of each line pair, [Beier & Neely, 1992] use:

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

#### where:

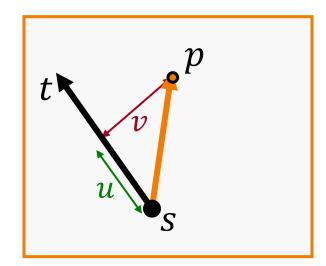
- length[i] is the length of line  $L_{out}[i]$
- dist[i](p) is the distance from p to  $L_{out}[i]$
- a (small),  $b \in [0.5,2.0]$ ,  $c \in [0.0,1.0]$  are constants that control the warp

#### **Feature-Based Warping**



How do we calculate the unsigned distance from a point p to the line segment from s to t?

$$\operatorname{dist}(p) = \begin{cases} |v| & \text{if } u \in [0,1] \\ ||p-s|| & \text{if } u < 0 \\ ||p-t|| & \text{if } u > 1 \end{cases}$$



#### Warping



Given a source image and set of corresponding line segment pairs, warp the source to the target by:

- Iterating over each pixel in the target
  - For each pair of line segments
    - » Compute the corresponding position in the source
    - » Compute the weights
  - Average to get the final source position
  - Sample the source (at the source position) to get the color at the target pixel

#### Warping Pseudocode

```
Warp(Img_{src}, L_{src}[N], L_{tqt}[N])
    foreach destination pixel p_{trat}:
        p_{src} = (0,0)
        sum = 0
        for i = 0 to N:
            q_{src} = p_{trgt} transformed by (L_{src}[i], L_{trgt}[i])
            p_{src} += q_{src} * weight[i]( p_{trat})
            sum += weight[i](p_{trat})
        p_{src} /= sum
        Img_{trqt}(p_{trqt}) = Img_{src}(p_{src})
    return Img<sub>trat</sub>
```

### Morphing at $\alpha \in [0, 1]$



Given two images, given a set of corresponding line segment pairs, and an interpolation time  $\alpha \in [0,1]$ :

- Compute the  $\alpha$ -blend of the of line segments (by blending the end-points)
- Warp the first image using the first set of line segments and the blended line segments
- Warp the second image using the second set of line segments and the blended line segments
- $\circ$  Compute the  $\alpha$ -blend of the warped images

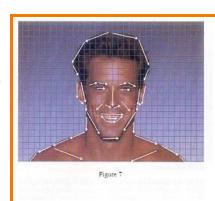
### Morphing at $\alpha \in [0, 1]$ Pseudocode



```
Morph(Img_0, L_0[N], Img_1, L_1[N], \alpha)
    foreach i \in \{1,...,N\}:
          L_{\alpha}[i] = line \alpha-th of the way from L_{\Omega}[i] to L_{1}[i]
     Warp_0 = Warp(Img_0, L_0[], L_\alpha[])
                                                             -warp
     Warp_1 = Warp(Img_1, L_1[], L_{\alpha}[])
    return (1-\alpha)^* Warp<sub>0</sub> + \alpha^* Warp<sub>1</sub>
                                                             -\alpha-blend
```

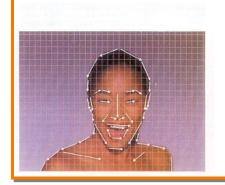


 $Img_0$ 



Warp<sub>0</sub>

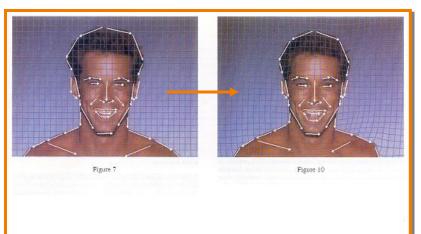
Img<sub>1</sub>



Warp<sub>1</sub>

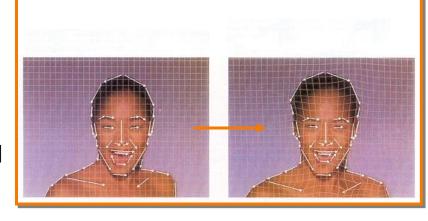


 $Img_0$ 



Warp<sub>0</sub>

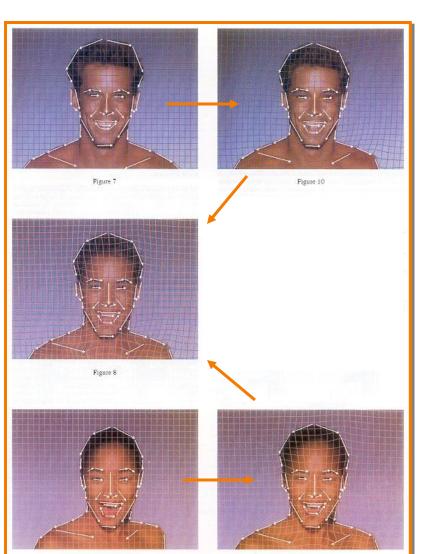
Img<sub>1</sub>



Warp<sub>1</sub>







Warp<sub>0</sub>

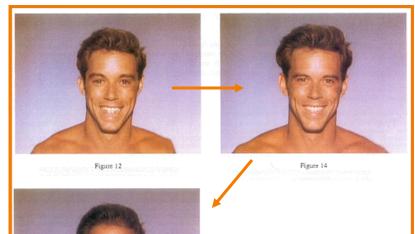
Result

Img<sub>1</sub>

Warp<sub>1</sub>



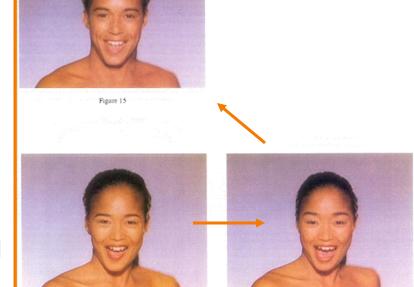




Warp<sub>0</sub>

Result





Warp<sub>1</sub>

#### **Animation Pseudocode**



```
Animate( Img_0 , L_0[N] , Img_1 , L_1[N] , Imgs_{out}[T+1] ) { for each t \in \{0,...,T\}: Imgs_{out}[t] = Morph(Img_0 , L_0[N] , Img_1 , L_1[N] , t/T ) }
```

#### Morphing



Check out Michael Jackson's "Black or White" video at:

https://www.youtube.com/watch?v=pTFE8cirkdQ



Or the earlier Plymouth Voyager commercial at: <a href="https://www.youtube.com/watch?v=0b939O7dGqQ">https://www.youtube.com/watch?v=0b939O7dGqQ</a>