IMAGE WARPING

Vuong Le
Dept. Of ECE
University of Illinois

ECE 417 – Spring 2013
With some slides from Alexei Efros, Steve Seitz, Jilin Tu, and Hao Tang

Content

- Introduction
- Parametric warping
- Barycentric coordinates
- Interpolation
- MP5

Content

- **Introduction**
- Parametric warping
- Barycentric coordinates
- Interpolation
- MP5

Image Warping

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

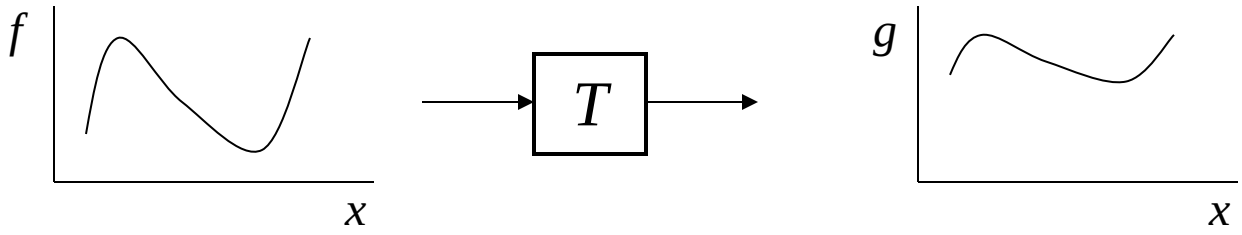


image warping: change ***domain*** of image

$$g(x) = f(T(x))$$

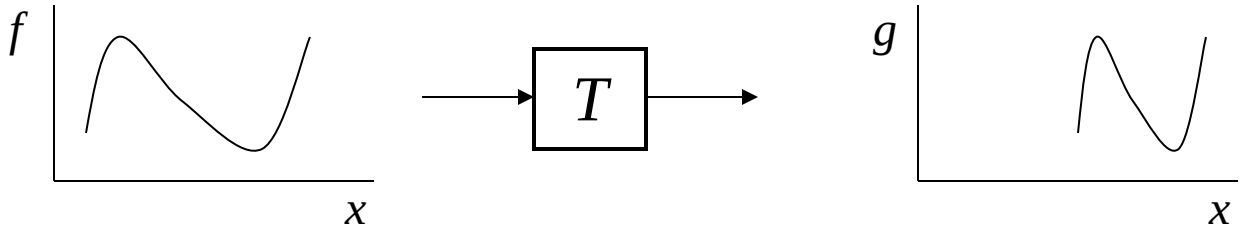


Image Warping

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

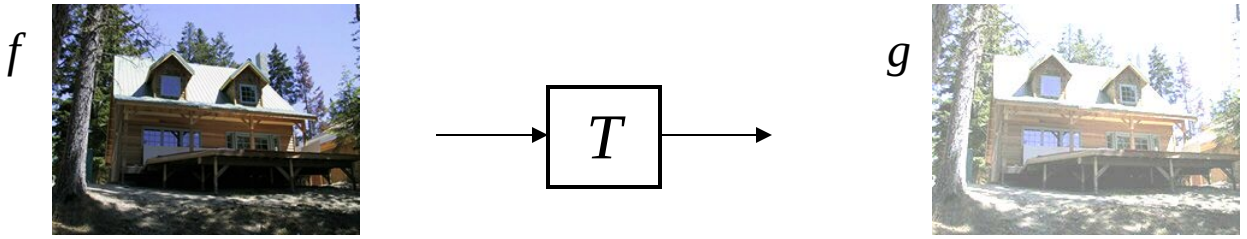
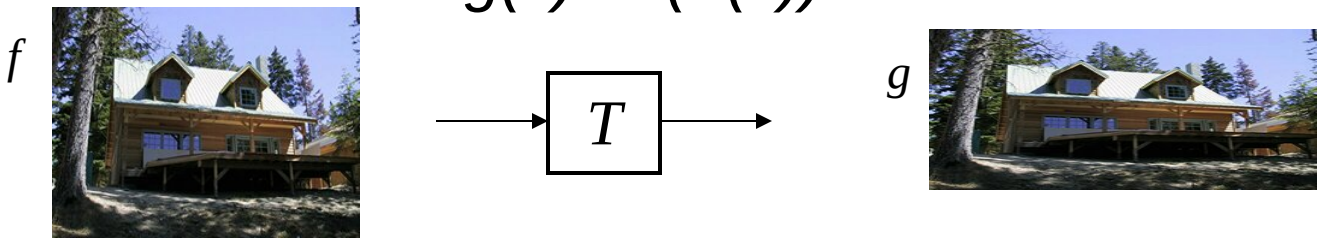


image warping: change ***domain*** of image

$$g(x) = f(T(x))$$



Global parametric image warping

Examples of parametric warps:

translation

rotation

aspect

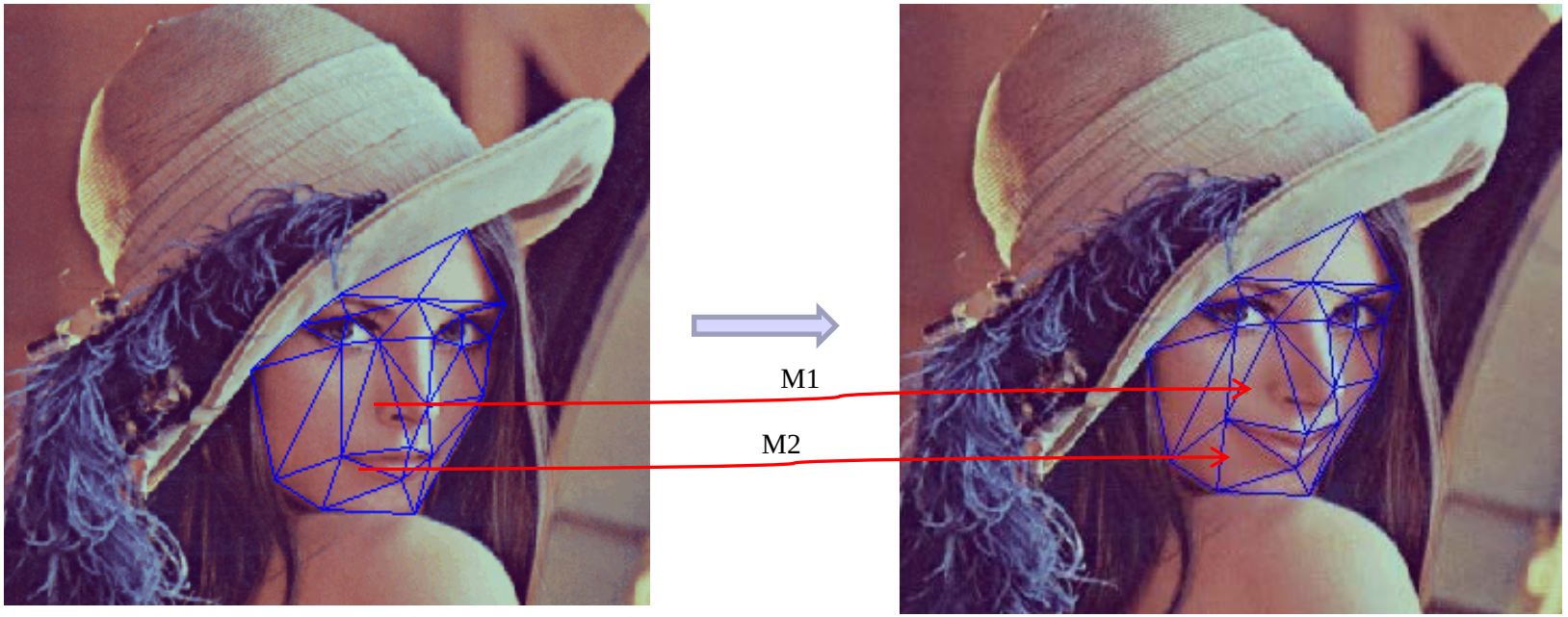
affine

perspective

cylindrical

Piecewise parametric image warping

- Define a mesh of small shape units: triangles/quadrangles
- Apply different transformations for different units



Non-parametric warping

- Move control points of a grid
- Use thin plate splines to produces a smooth vector field

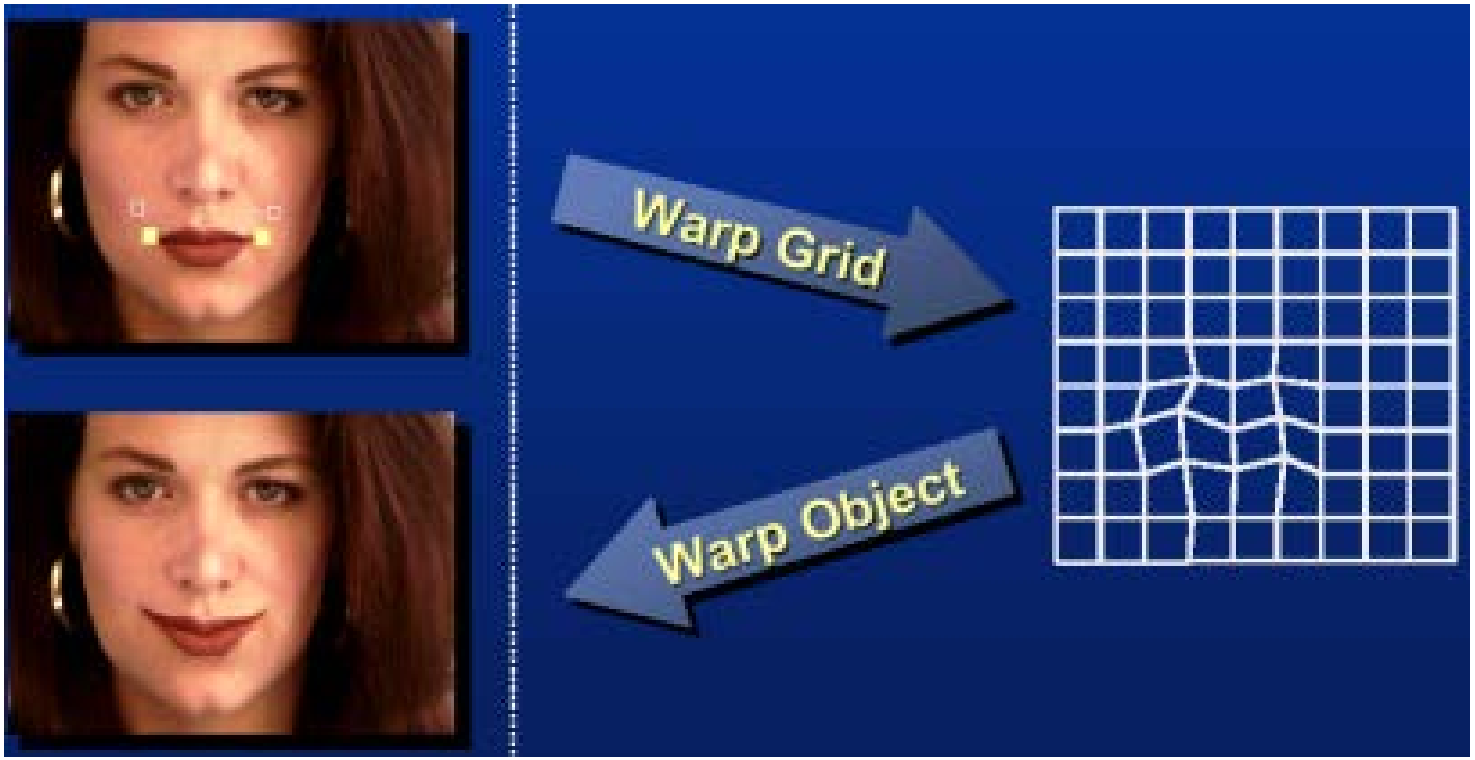
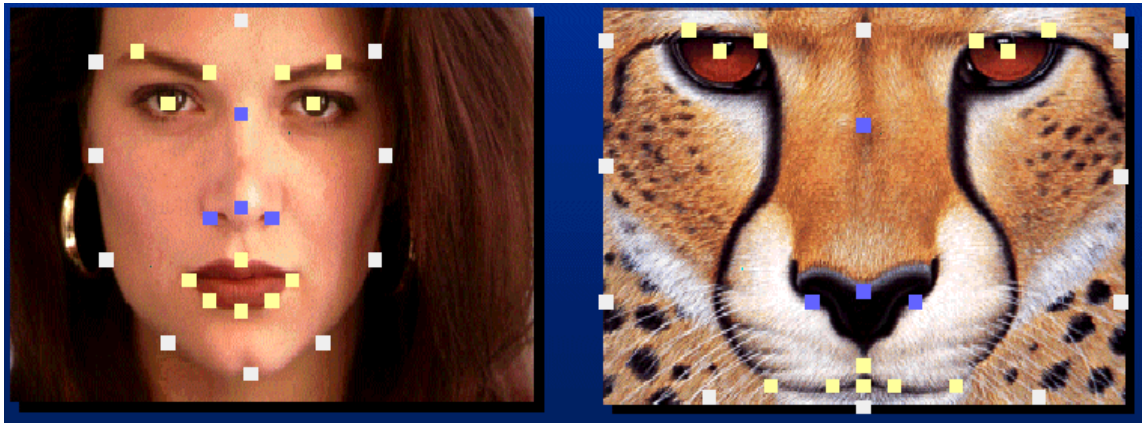
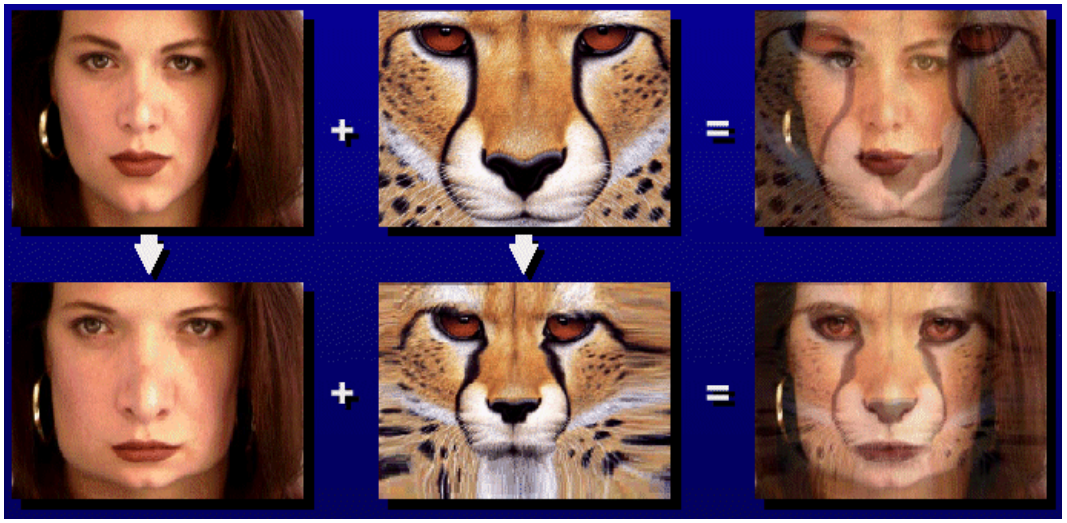
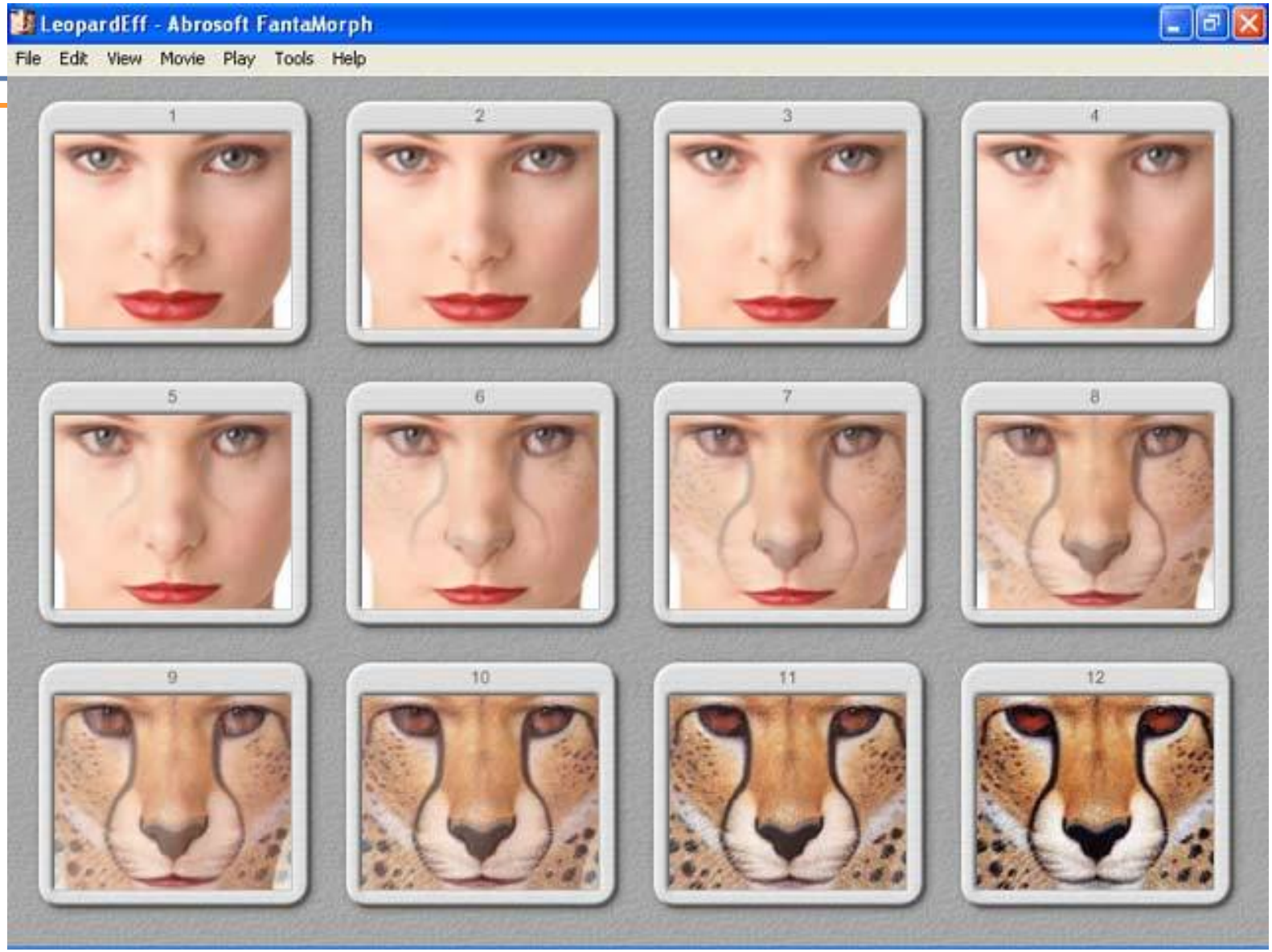


Image morphing






LeopardEff - Abrosoft FantaMorph
File Edit View Movie Play Tools Help
1
2
3
4
5
6
7
8
9
10
11
12

Photoshop examples

Photoshop examples

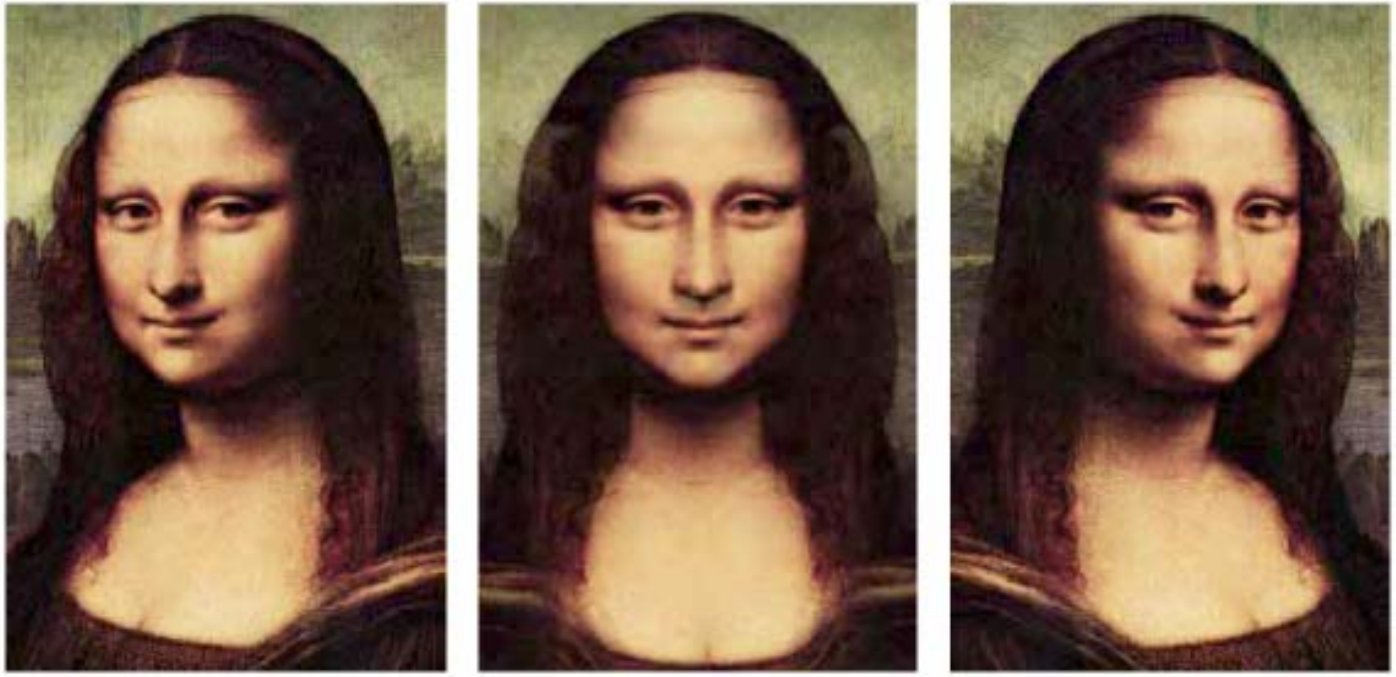
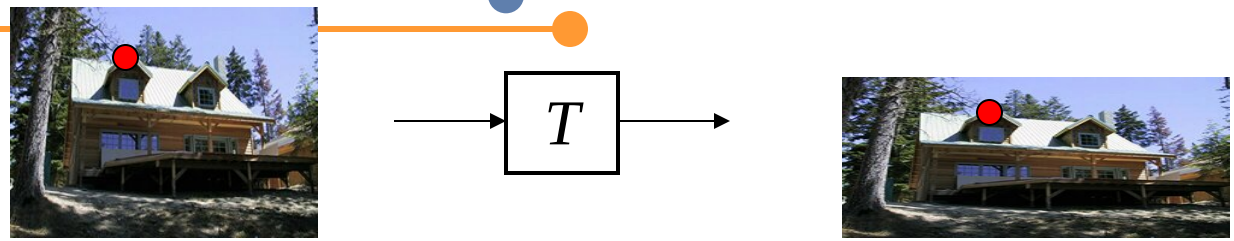


Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and it's reflection (right).

Content

- Introduction
- **Parametric warping**
- Barycentric coordinates
- Interpolation
- MP5

Parametric image warping



$\mathbf{p} = (x,y)$ $\mathbf{p'} = (x',y')$

Transformation T is a coordinate-changing machine:

What does it mean that *T* is parametric?

- can be described by just a few numbers (parameters)

How to represent global and local transforms?

- Global: T Is the same for any point p in the image
- Local: T can be different at different location

Let's try to represent *T* as a matrix: $\mathbf{p'} = \mathbf{M}\mathbf{p}$

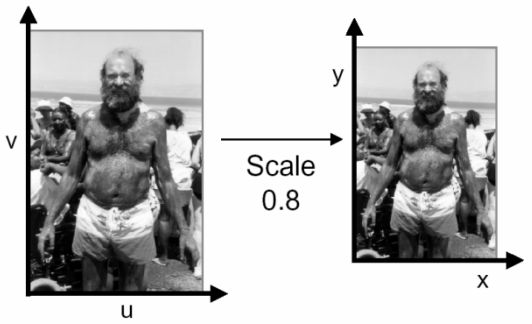
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling

Scaling operation:

$$x' = ax$$

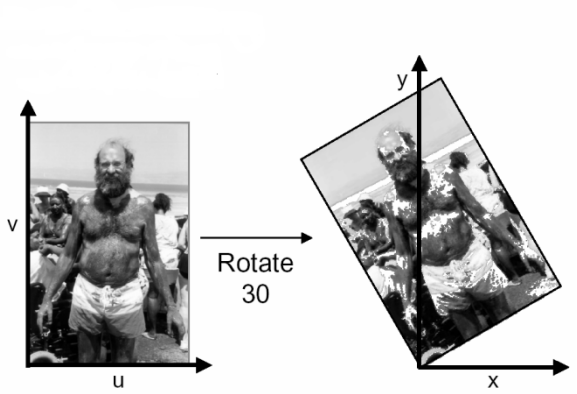
$$y' = by$$

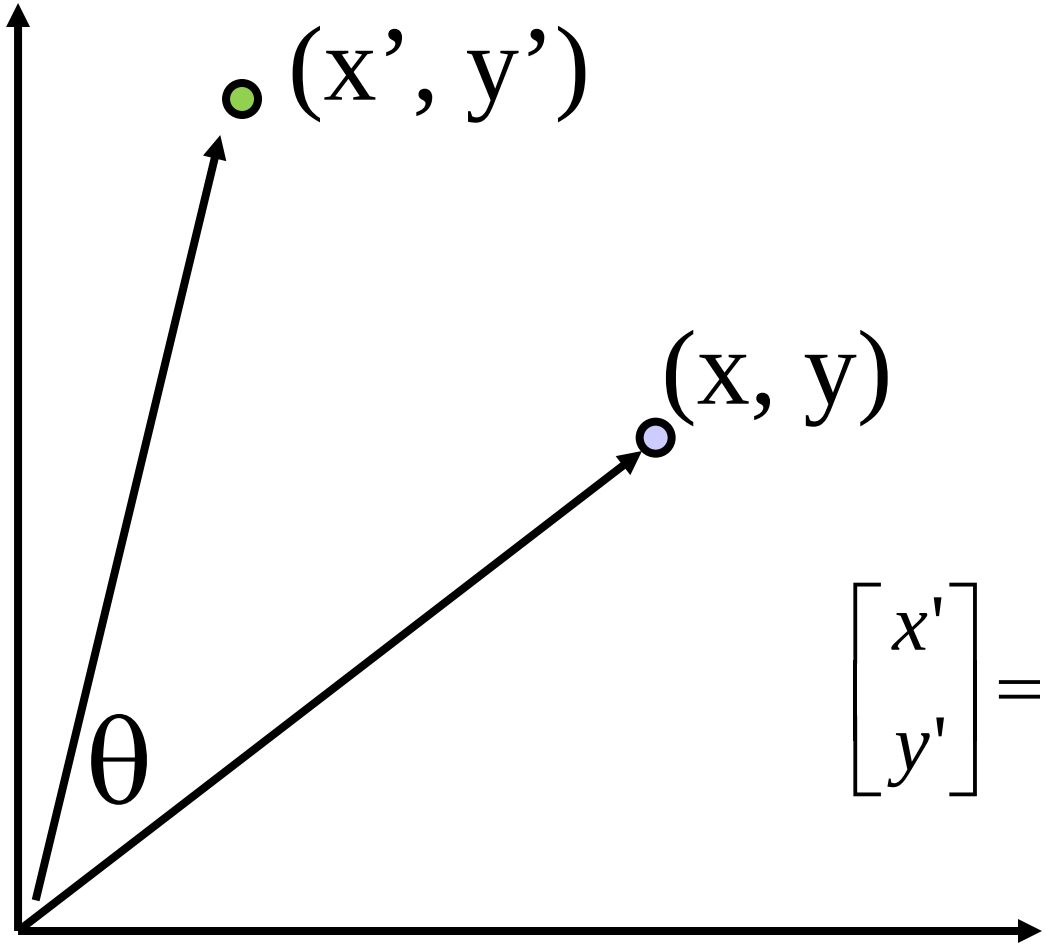


Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

2-D Rotation

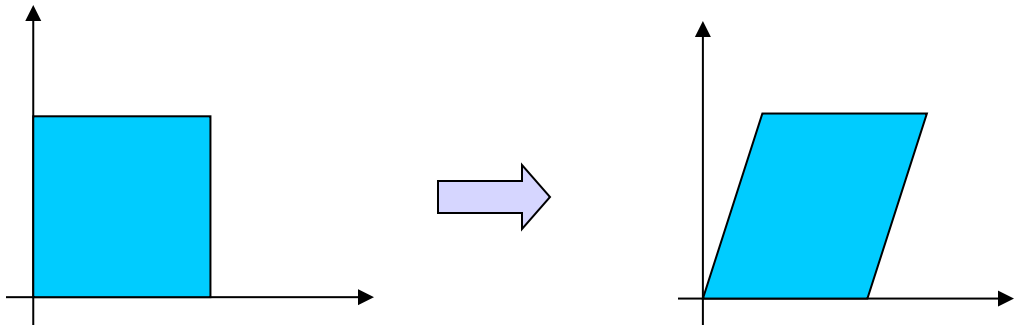




$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shearing in x



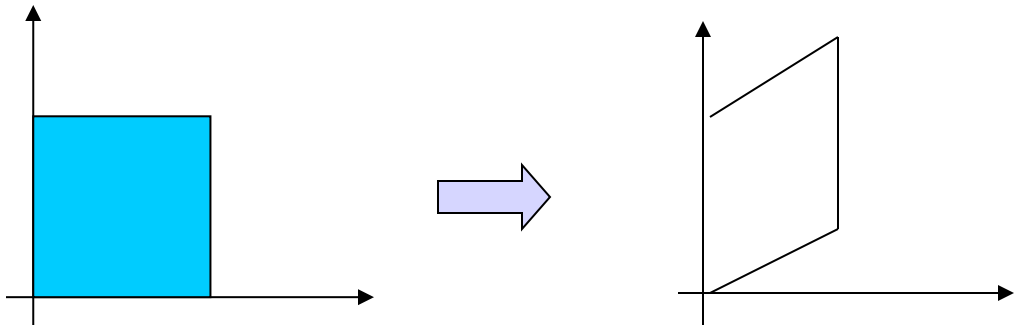
- The y coordinates are unaffected, but the x coordinates are translated linearly with y
- That is

$$x' = x + sh_y * y$$

$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_y \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shearing in y



$$x' = x$$
$$y' = sh_x * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ sh_x & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Shear?

$$x' = x + sh_x * y$$
$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror about Y axis?

$$x' = -x$$
$$y' = y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Mirror over (0,0)?

$$x' = -x$$
$$y' = -y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$
$$y' = y + t_y$$

NO!

Only linear 2D transformations can be represented with a 2x2 matrix

All 2D Linear Transformations

Linear transformations are combinations of …

- Scale,
- Rotation,
- Shear, and
- Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Distance or length ratios are preserved on parallel lines
- Ratios of areas are preserved

Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$

$$y' = y + t_y$$

Homogeneous Coordinates

Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$

$$y' = y + t_y$$

A: Using the rightmost column:

$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

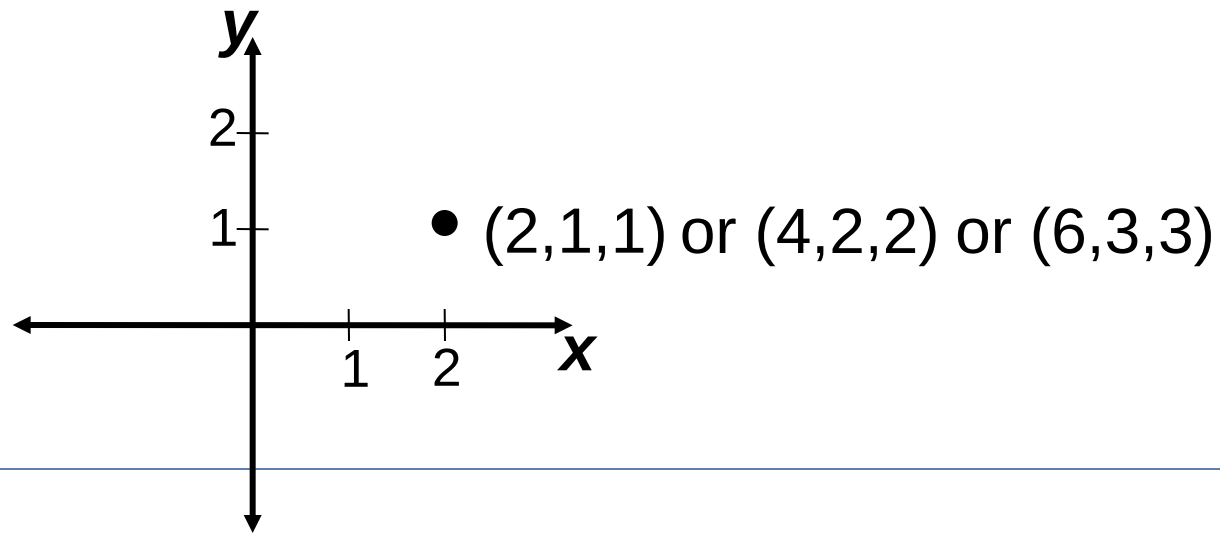
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous Coordinates

Add a 3rd coordinate to every 2D point

- (x, y, w) represents a point at location (x/w, y/w)
- (x, y, 0) represents a *point at infinity*
- (0, 0, 0) is not allowed

- Convenient coordinate system to represent many useful transformations
- (aw, bw, w) represent the same 2D point for any value of *w*



Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Affine Transformations

Affine transformations are combinations of …

- Linear 2D transformations, and
- Translations

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Properties of affine transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines remain parallel
- Length/distance ratios are preserved on parallel lines
- Ratios of areas are preserved

Solution to Affine Warping

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The affine mapping function is represented by

$x(u,v) = a_0 + a_1 u + a_2 v; \quad y(u,v) = b_0 + b_1 u + b_2 v.$

Using the predetermined feature point correspondence, we can set up two sets of equations

$$\begin{bmatrix} 1 & u_1 & v_1 \\ 1 & u_2 & v_2 \\ \dots & \dots & \dots \\ 1 & u_N & v_N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} \quad \text{or Aa = x,} \quad \begin{bmatrix} 1 & u_1 & v_1 \\ 1 & u_2 & v_2 \\ \dots & \dots & \dots \\ 1 & u_N & v_N \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} \quad \text{or Ab = y.}$$

If N = 3, and matrix A is invertible, the solution is

$a = A^{-1}\,\mathrm{x}, b = A^{-1}\,\mathrm{y}.$

If N > 3, we have more equations then unknowns. The least square solution is

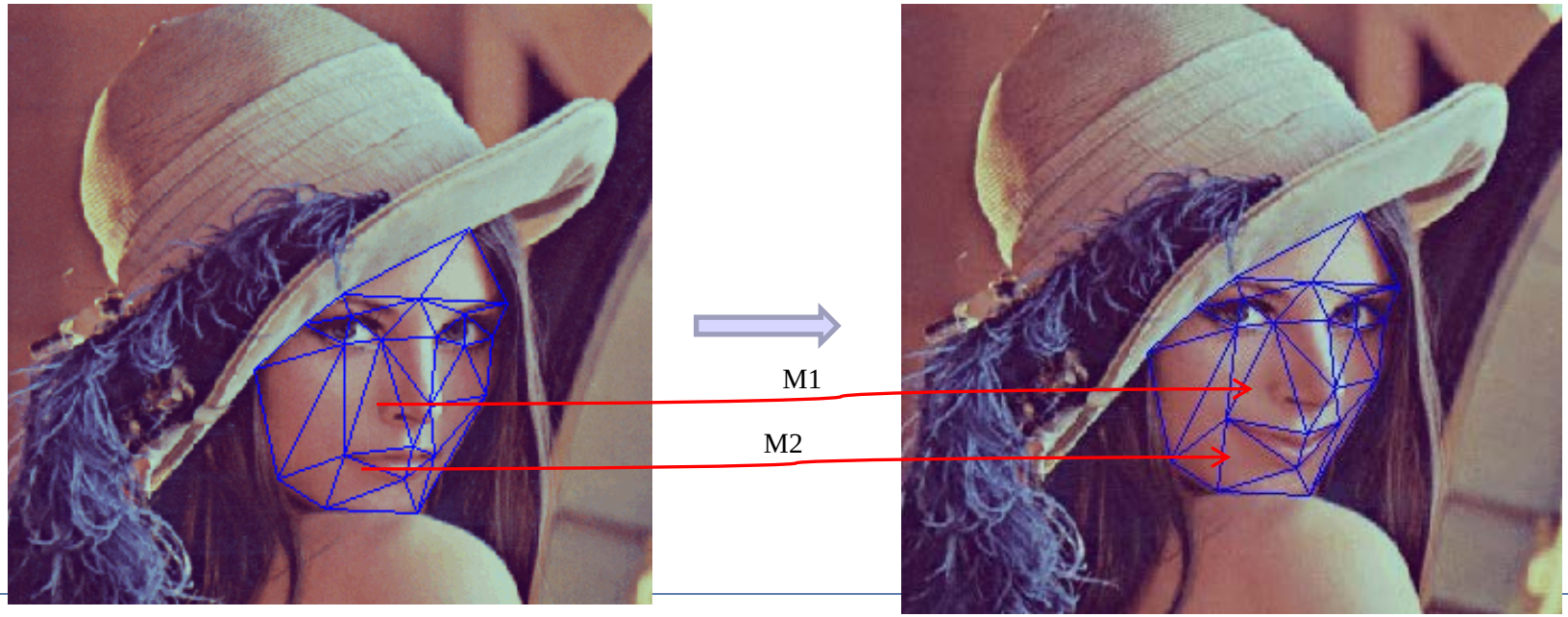
$a = \left(A^T A\right)^{-1} A^T \mathrm{x}, b = \left(A^T A\right)^{-1} A^T \mathrm{y}.$

If N = 3, and matrix A is not invertible, we would need to find more feature points.

“Triangle-wise” parametric image warping

- For each transformed image frame
  - Define a mesh of triangles
  - Find a affine transform from each triangle’s vertices, apply it to other points inside the triangle

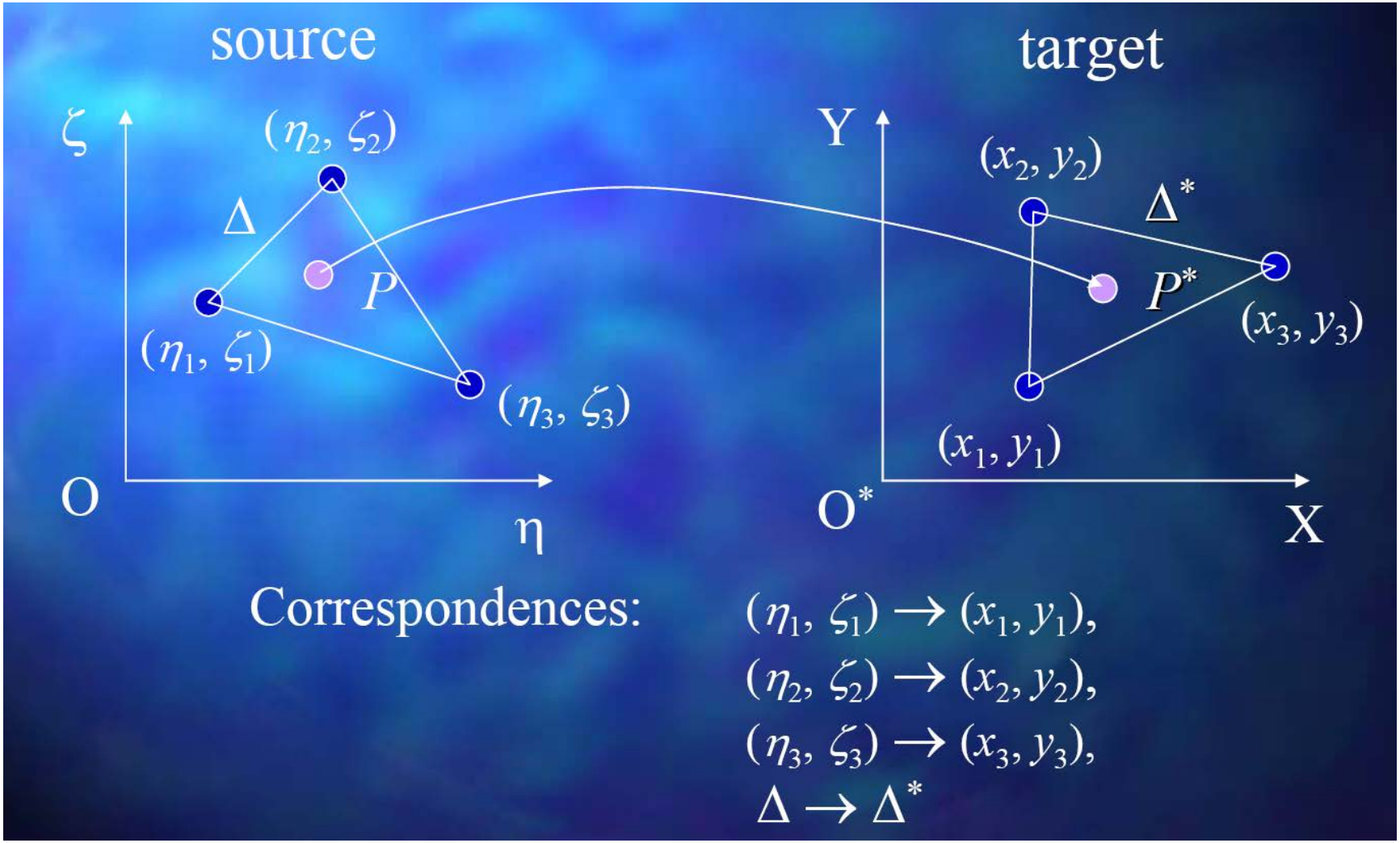
  → Have to solve affine transforms at every frame



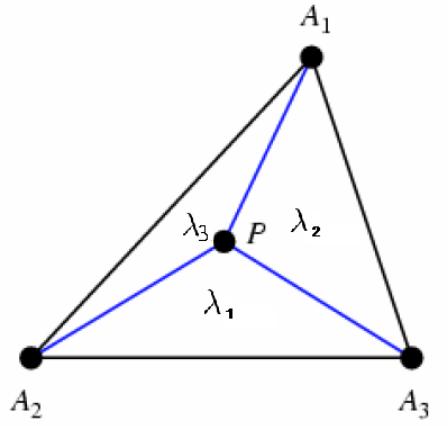
Content

- Introduction
- Parametric warping
- **Barycentric coordinates**
- Interpolation
- MP5

Affine warping for triangles



Barycentric Coordinates



For any (ζ, η) in the triangle, we have

$$\zeta = \lambda_1 \zeta_1 + \lambda_2 \zeta_2 + \lambda_3 \zeta_3$$
$$\eta = \lambda_1 \eta_1 + \lambda_2 \eta_2 + \lambda_3 \eta_3$$
$$1 = \lambda_1 + \lambda_2 + \lambda_3 \qquad (\lambda_i \geq 0,\ i = 1, 2, 3)$$

or

$$\begin{bmatrix} \zeta \\ \eta \\ 1 \end{bmatrix} = \lambda_1 \begin{bmatrix} \zeta_1 \\ \eta_1 \\ 1 \end{bmatrix} + \lambda_2 \begin{bmatrix} \zeta_2 \\ \eta_2 \\ 1 \end{bmatrix} + \lambda_3 \begin{bmatrix} \zeta_3 \\ \eta_3 \\ 1 \end{bmatrix} = \begin{bmatrix} \zeta_1 & \zeta_2 & \zeta_3 \\ \eta_1 & \eta_2 & \eta_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

Barycentric Coordinates

Fact

- ~~Assumption:~~ homogeneous barycentric coordinate is invariant to affine transformation, thus P's corresponding point $P^*(x, y)$ can be represented as

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$$

$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3.$$

with the same λ_i

$$\begin{bmatrix} \zeta \\ \eta \\ 1 \end{bmatrix} = \begin{bmatrix} \zeta_1 & \zeta_2 & \zeta_3 \\ \eta_1 & \eta_2 & \eta_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = M \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

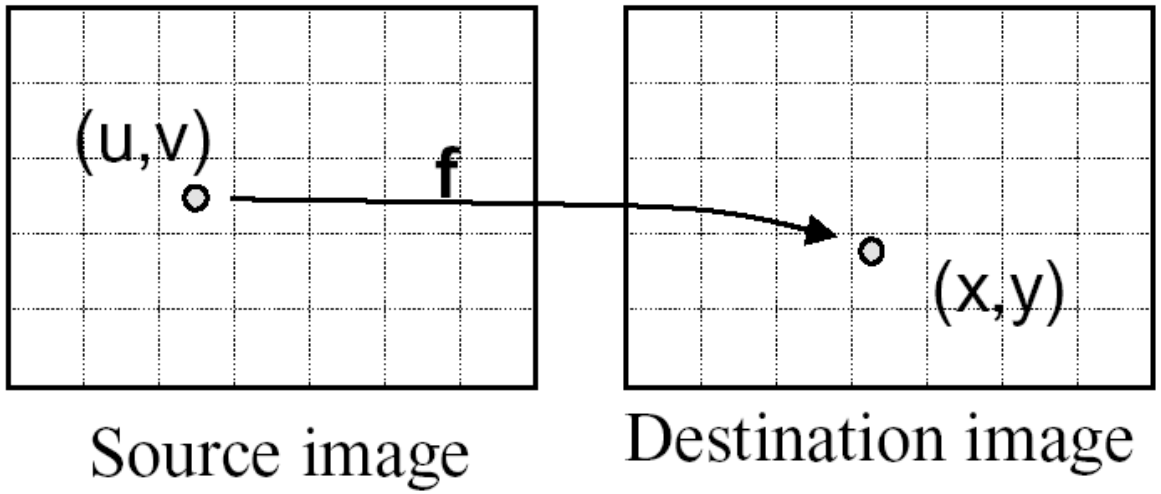
Content

- Introduction
- Parametric warping
- Barycentric coordinates
- **Interpolation**
- MP5

Image Warping Implementation I

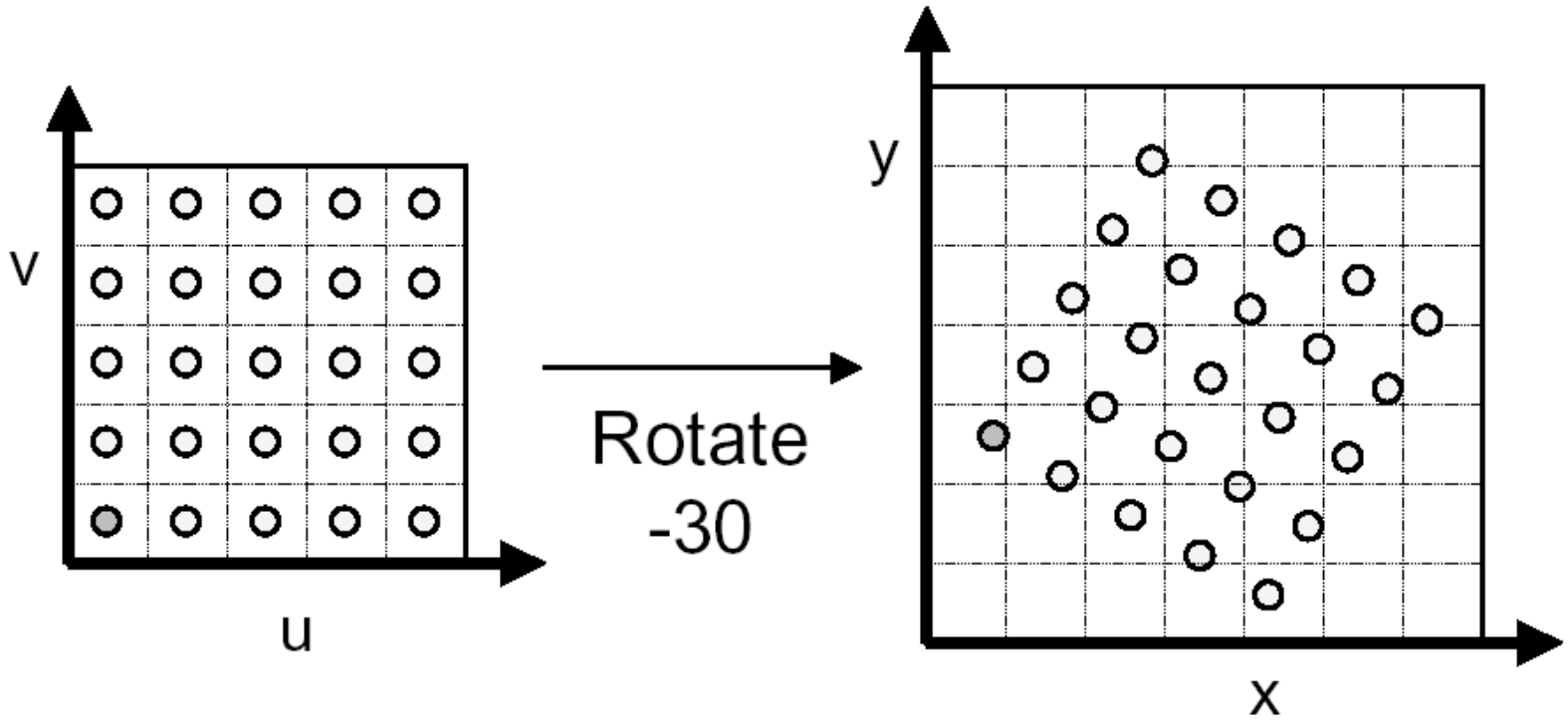
- Forward mapping:

```
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);
  }
}
```



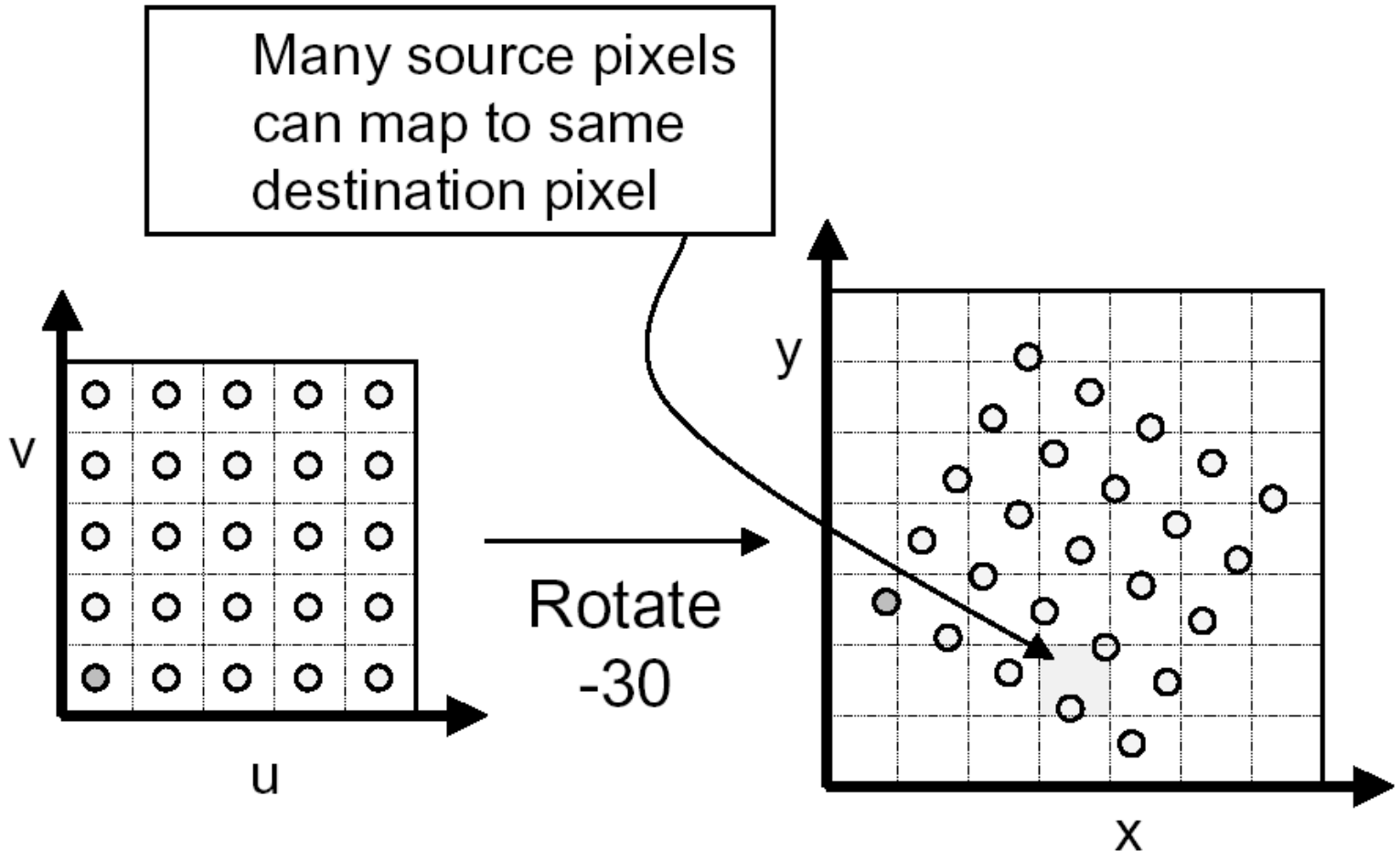
Forward Mapping

- Iterate over source image



Forward Mapping

- Iterate over source image



Forward Mapping

- Iterate over source image

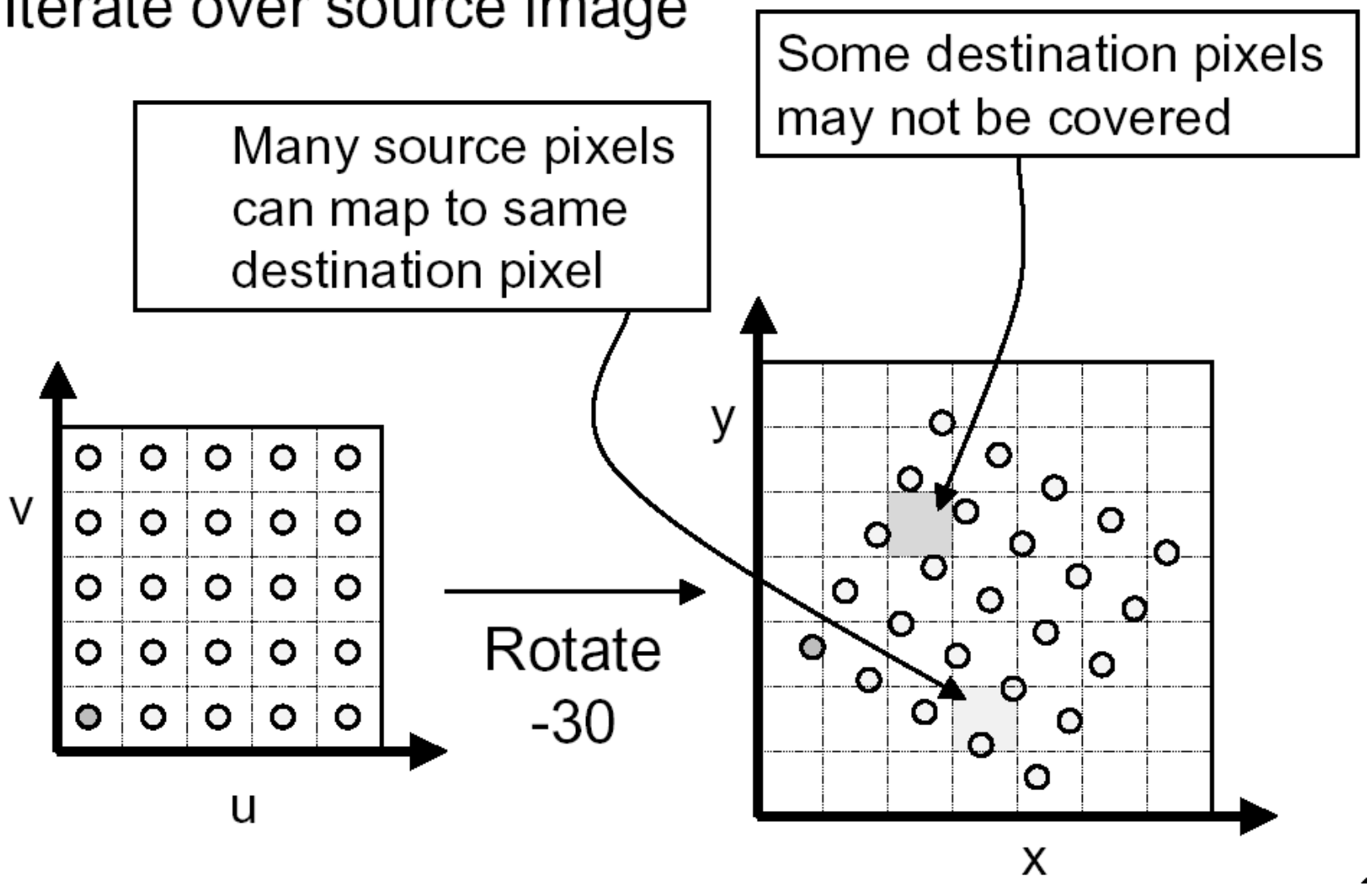
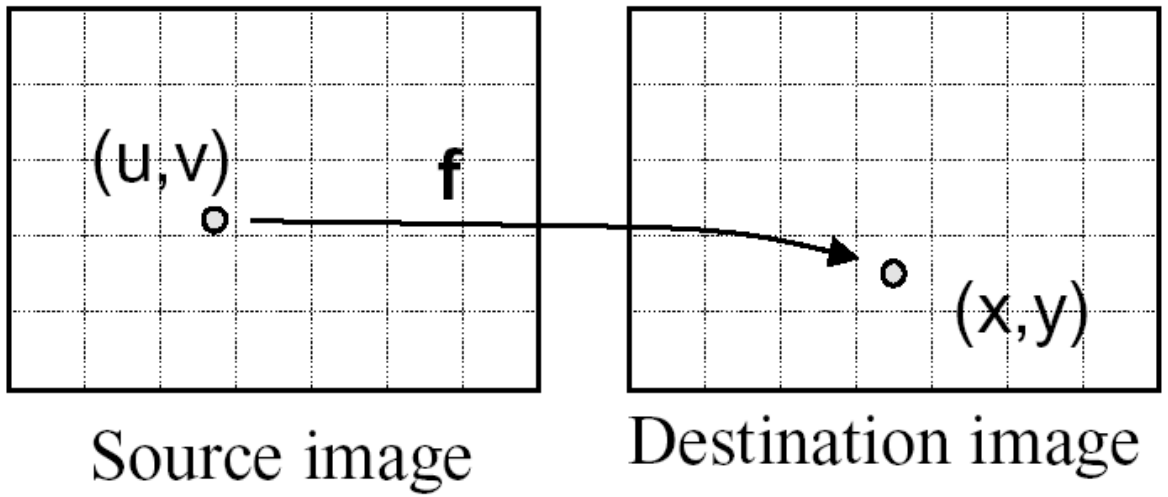


Image Warping Implementation II

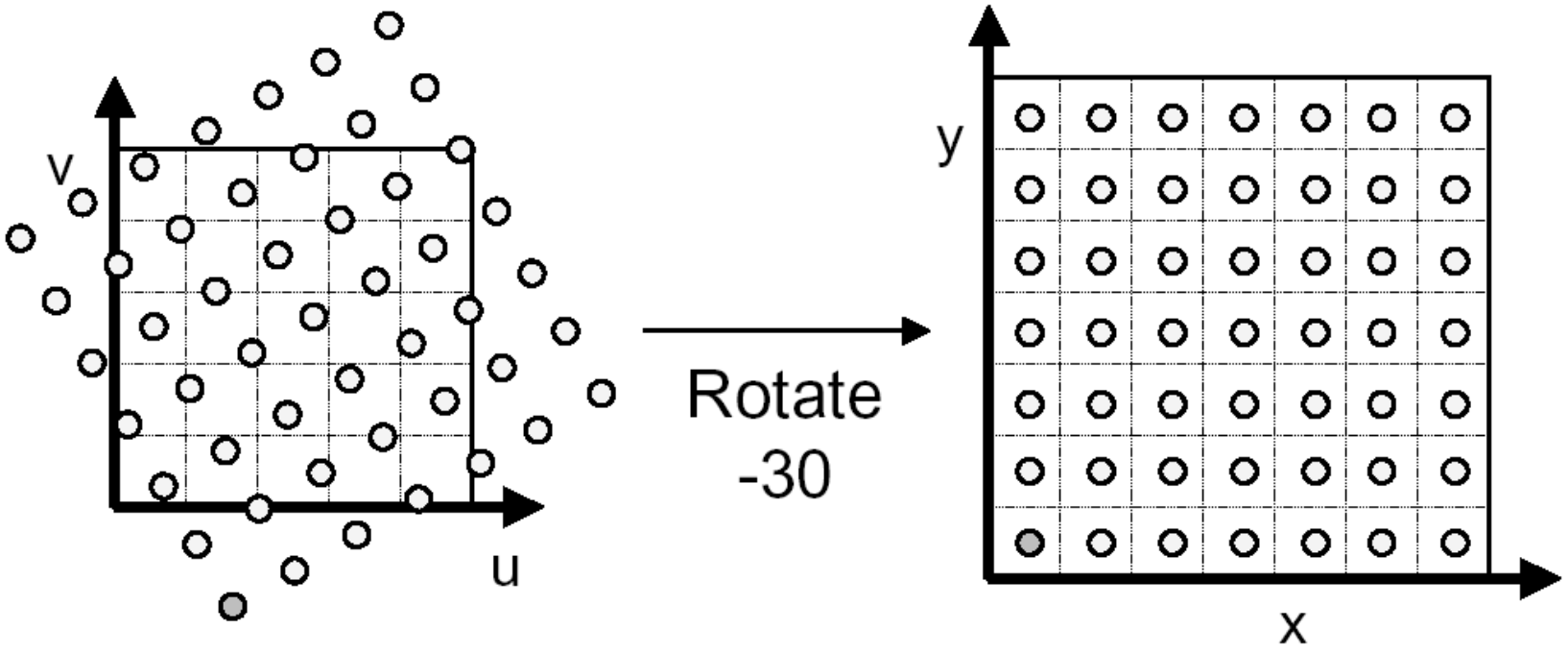
- Reverse mapping:

```
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!



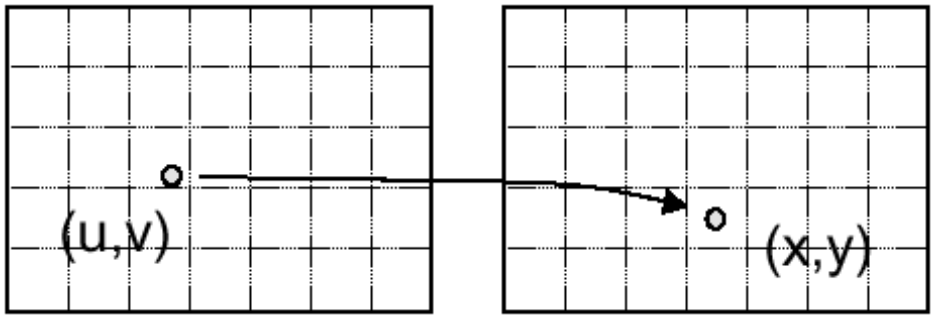
Resampling

Evaluate source image at arbitrary (u, v)

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

Some kinds of resampling

- Nearest neighbor
- Bilinear interpolation
- Gaussian filter

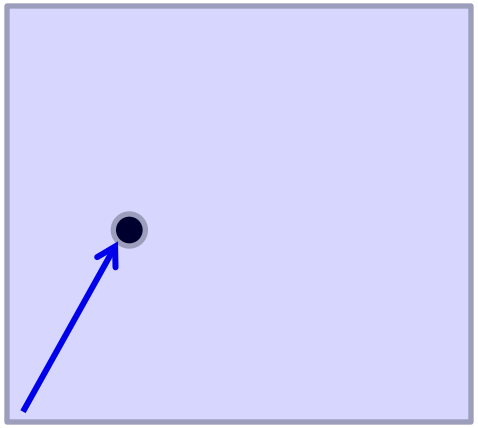


Nearest neighbor

Take value at closest pixel

```
int iu = trunc(u + 0.5);
int iv = trunc(v + 0.5);
dst(x, y) = src(iu, iv);
```

Simple, but causes aliasing



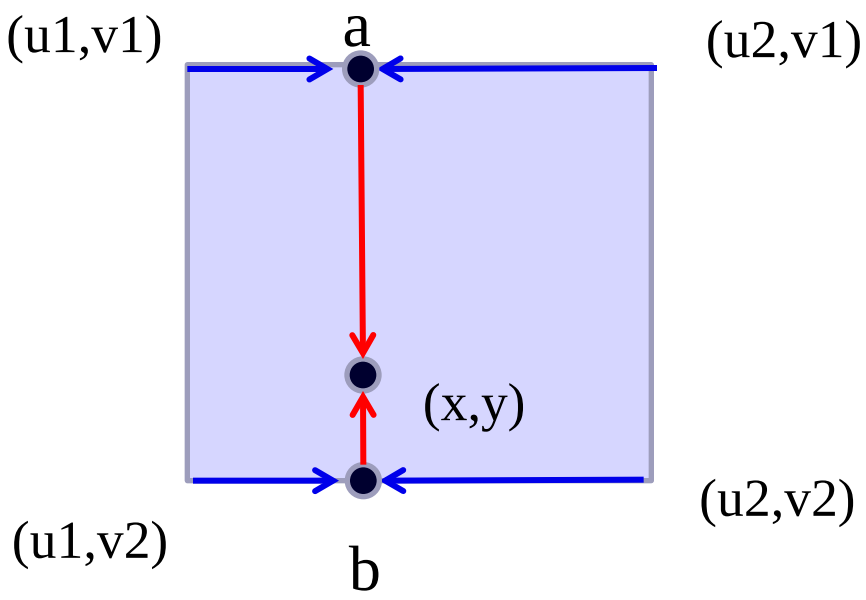
Bilinear interpolation

Bilinearly interpolate four surrounding pixels

a = linear interpolation of src(u_1, v_1) and src(u_2, v_1)

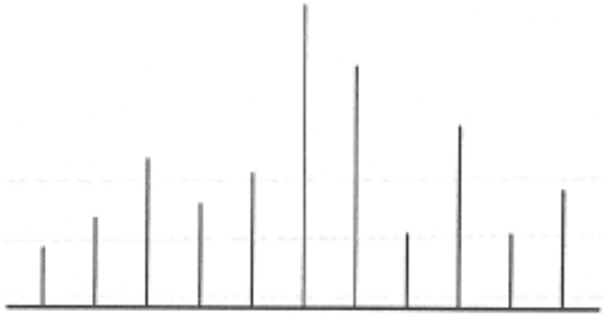
b = linear interpolation of src(u_1, v_2) and src(u_2, v_2)

dst(x, y) = linear interpolation of ‘a’ and ‘b’

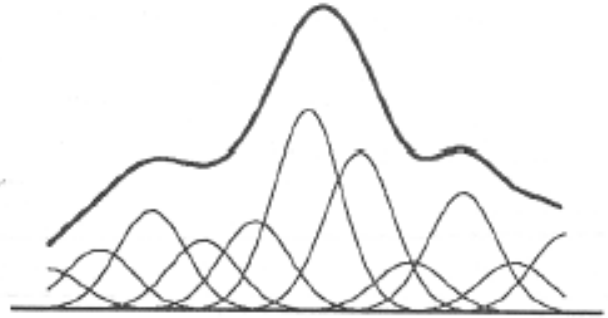


Gaussian Filter

Convolve with Gaussian filter



Input



Output

Width of Gaussian kernel affects bluriness

Image Warping Implementation II – with resampling

- 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);
  }
}
```

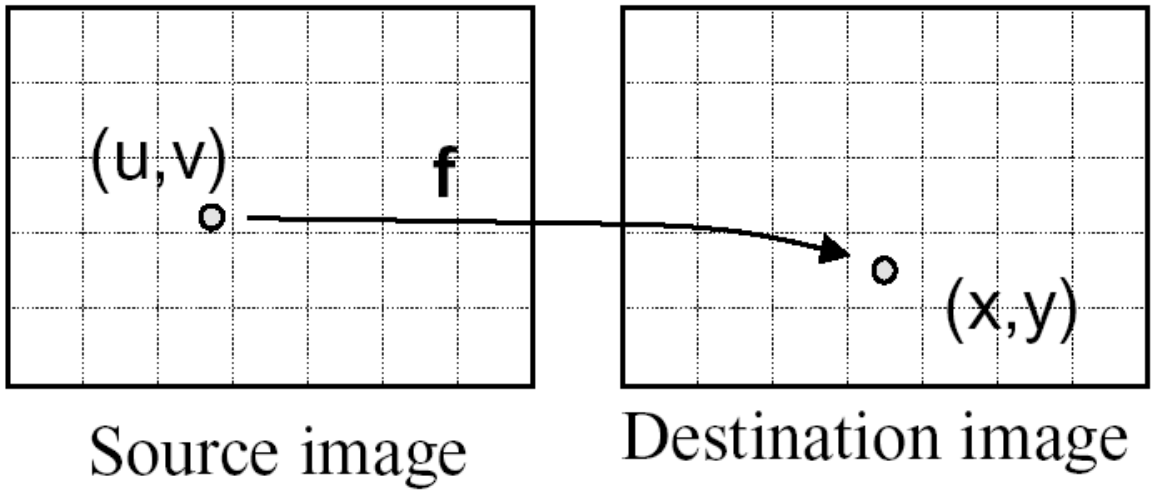
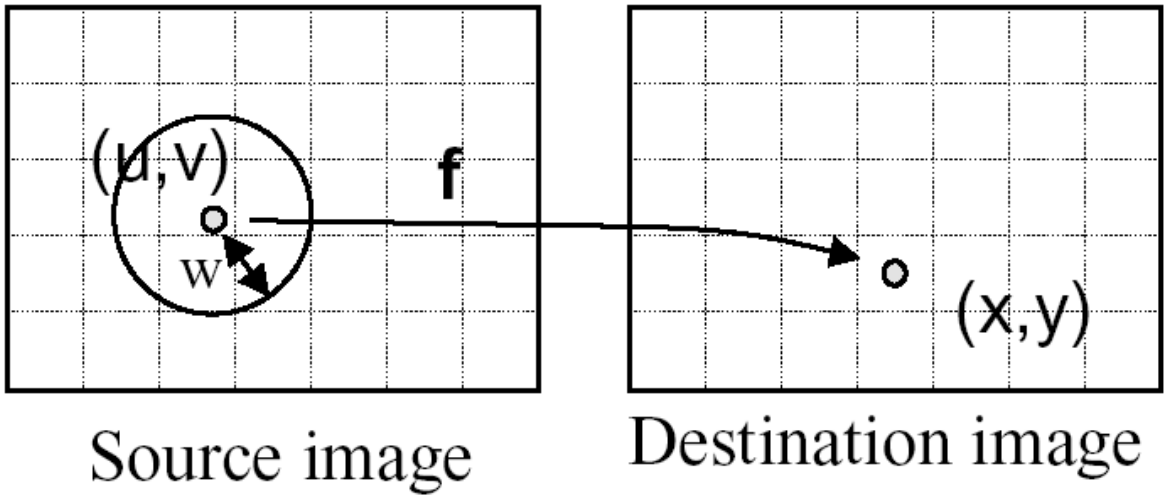


Image Warping Implementation II – with Gaussian resampling

- Reverse mapping:

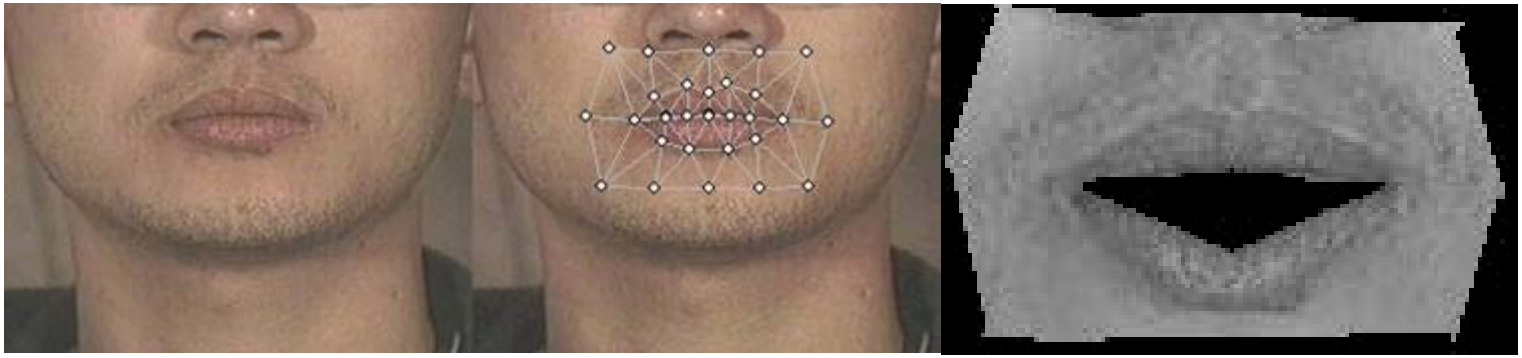
```
for (int x = 0; x < xmax; x++) {
  for (int y = 0; y < ymax; y++) {
    float u = f_x^-1(x,y);
    float v = f_y^-1(x,y);
    dst(x,y) = resample_src(u,v,w);
  }
}
```



Content

- Introduction
- Parametric warping
- Barycentric coordinates
- Interpolation
- **MP5**

Warping in MP5



- Warping
  - For each triangle find the affine transform
  - Apply it to the points inside
  - Assign image value to the triangle points
- Suggestions
  - Barycentric coordinate
  - Reverse mapping

Preview of MP5