

Image warping/morphing

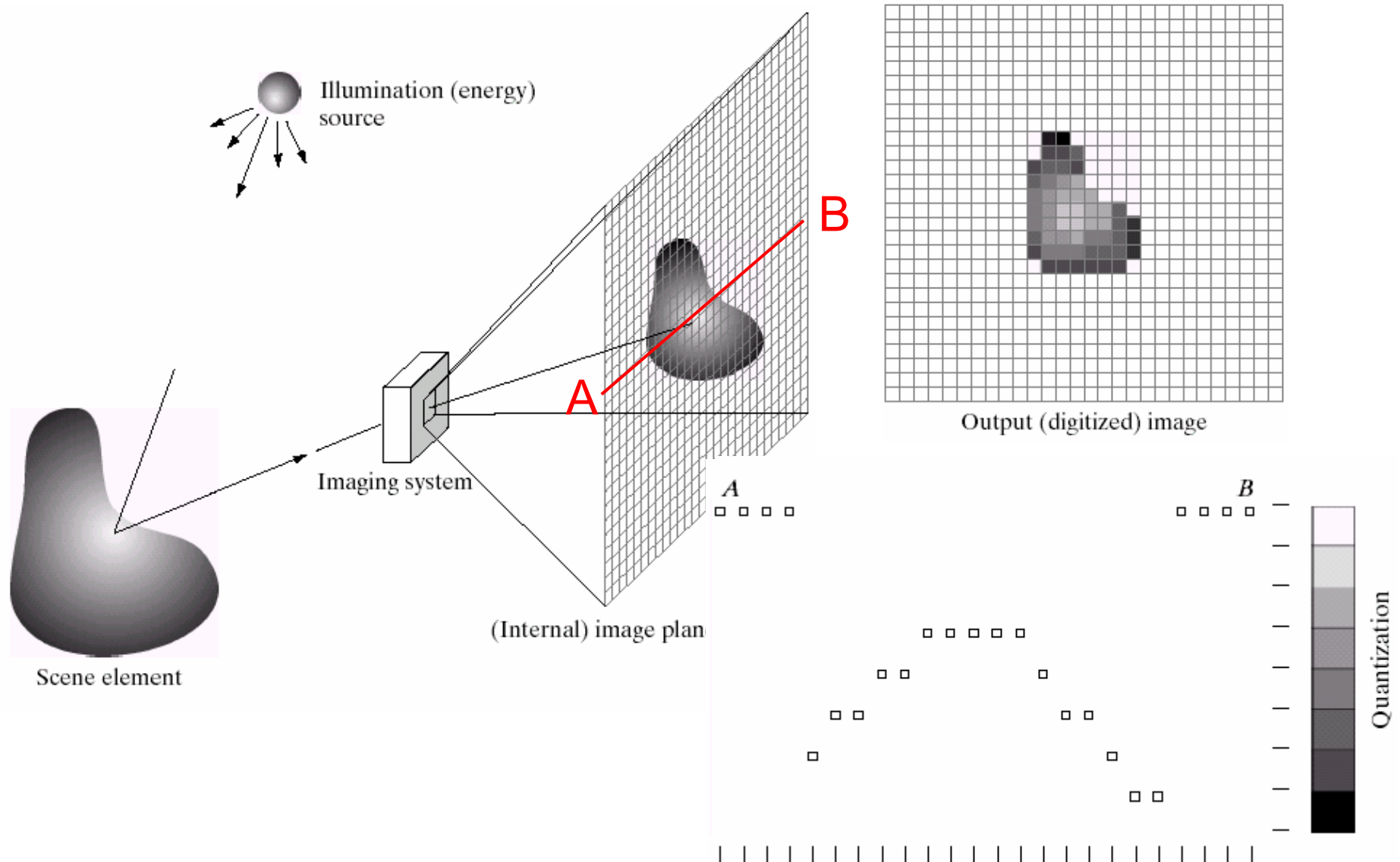
Digital Visual Effects

Yung-Yu Chuang

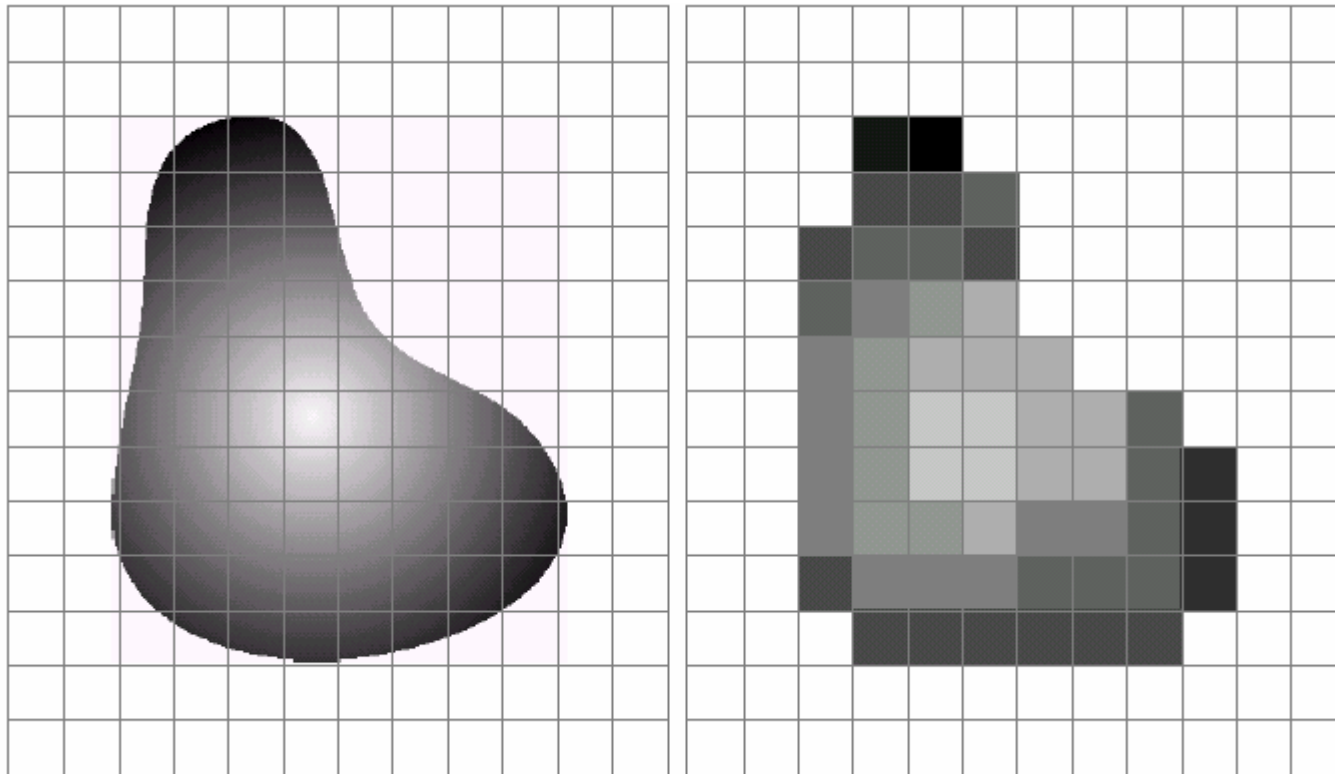
with slides by Richard Szeliski, Steve Seitz, Tom Funkhouser, Jia-Bing Huang and Alexei Efros

Image warping

Image formation



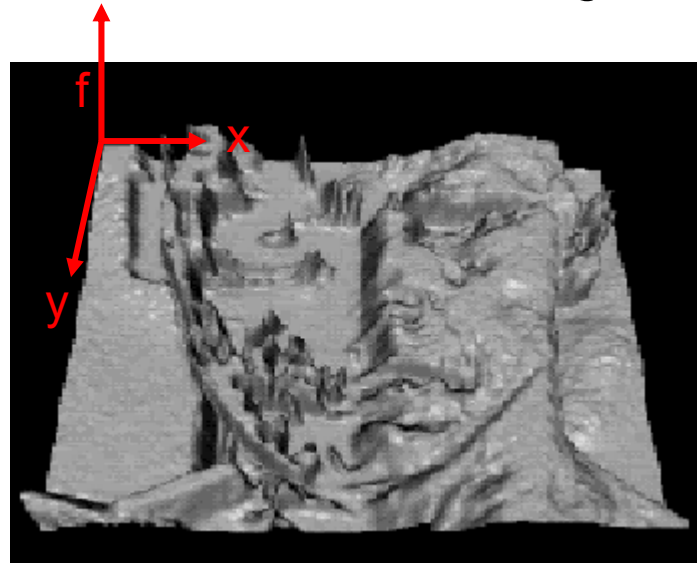
Sampling and quantization



What is an image

- We can think of an **image** as a function, $f: \mathbb{R}^2 \rightarrow \mathbb{R}$:
 - $f(x, y)$ gives the **intensity** at position (x, y)
 - defined over a rectangle, with a finite range:

- $f: [a, b] \times [c, d] \rightarrow [0, 1]$



- A color image

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

A digital image

- We usually operate on **digital (discrete)** images:
 - Sample the 2D space on a regular grid
 - Quantize each sample (round to nearest integer)
- If our samples are D apart, we can write this as:

$$f[i, j] = \text{Quantize}\{ f(i D, j D) \}$$

- The image can now be represented as a matrix of integer values

$j \longrightarrow$

↓	62	79	23	119	120	105	4	0
	10	10	9	62	12	78	34	0
	10	58	197	46	46	0	0	48
	176	135	5	188	191	68	0	49
	2	1	1	29	26	37	0	77
	0	89	144	147	187	102	62	208
	255	252	0	166	123	62	0	31
	166	63	127	17	1	0	99	30

Image warping

image filtering: change *range* of image

$$g(x) = h(f(x))$$

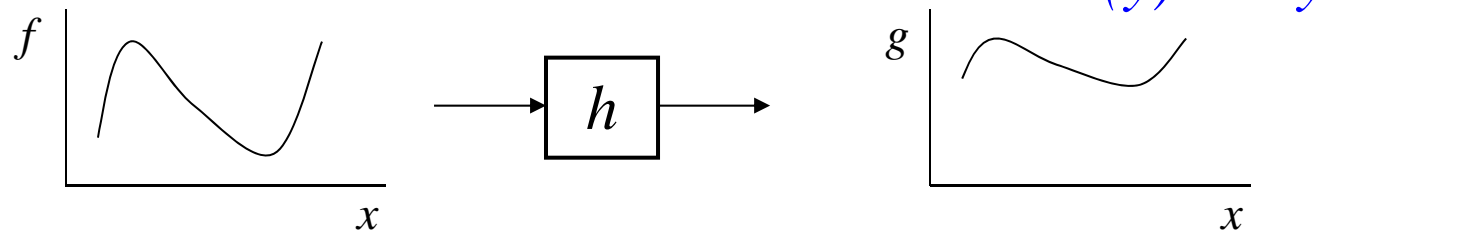


image warping: change *domain* of image

$$g(x) = f(h(x))$$

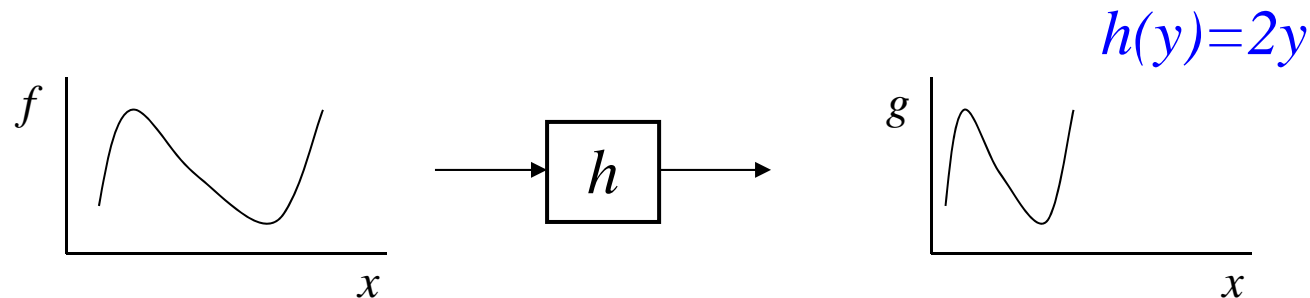


Image warping

image filtering: change *range* of image

$$g(x) = h(f(x))$$

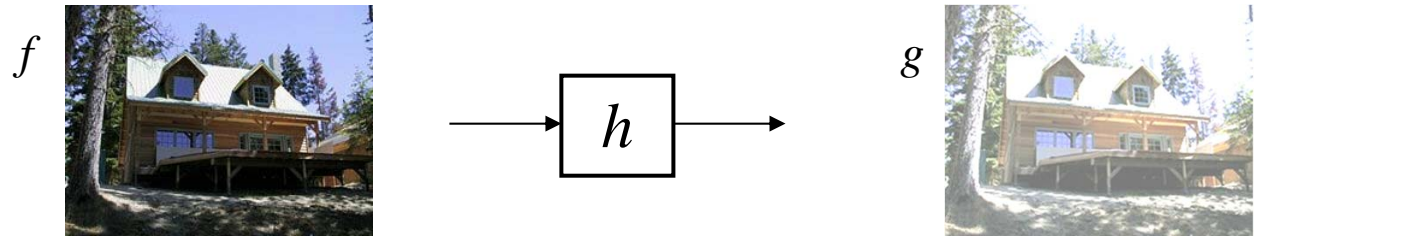
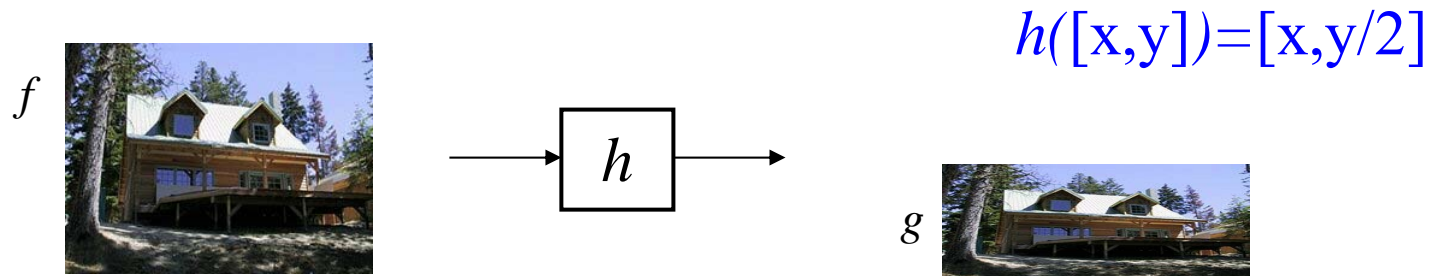


image warping: change *domain* of image

$$g(x) = f(h(x))$$



Parametric (global) warping

Examples of parametric warps:



translation



rotation



aspect



affine



perspective

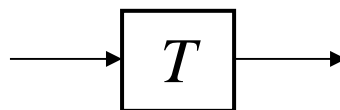


cylindrical

Parametric (global) warping



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

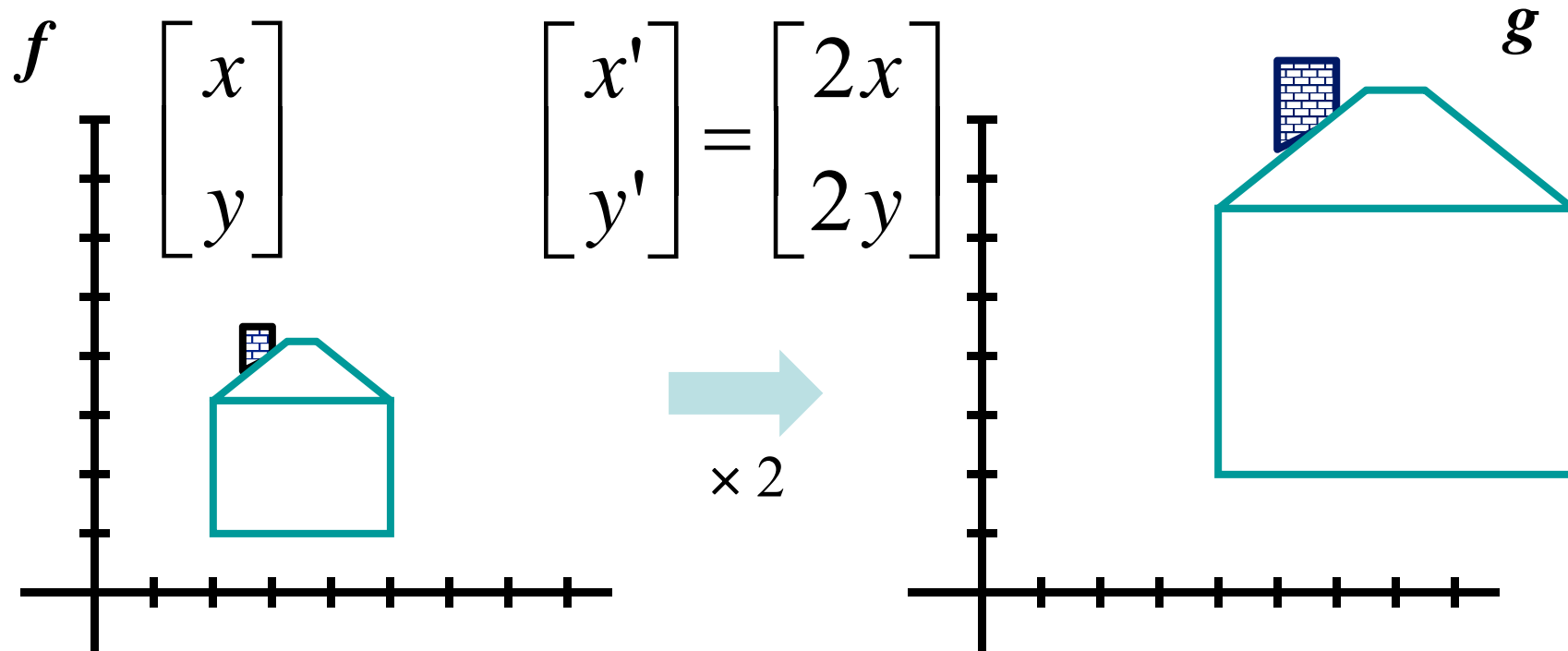


$$\mathbf{p}' = (x', y')$$

- Transformation T is a coordinate-changing machine: $\mathbf{p}' = T(\mathbf{p})$
- What does it mean that T is global?
 - Is the same for any point \mathbf{p}
 - can be described by just a few numbers (parameters)
- 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* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:

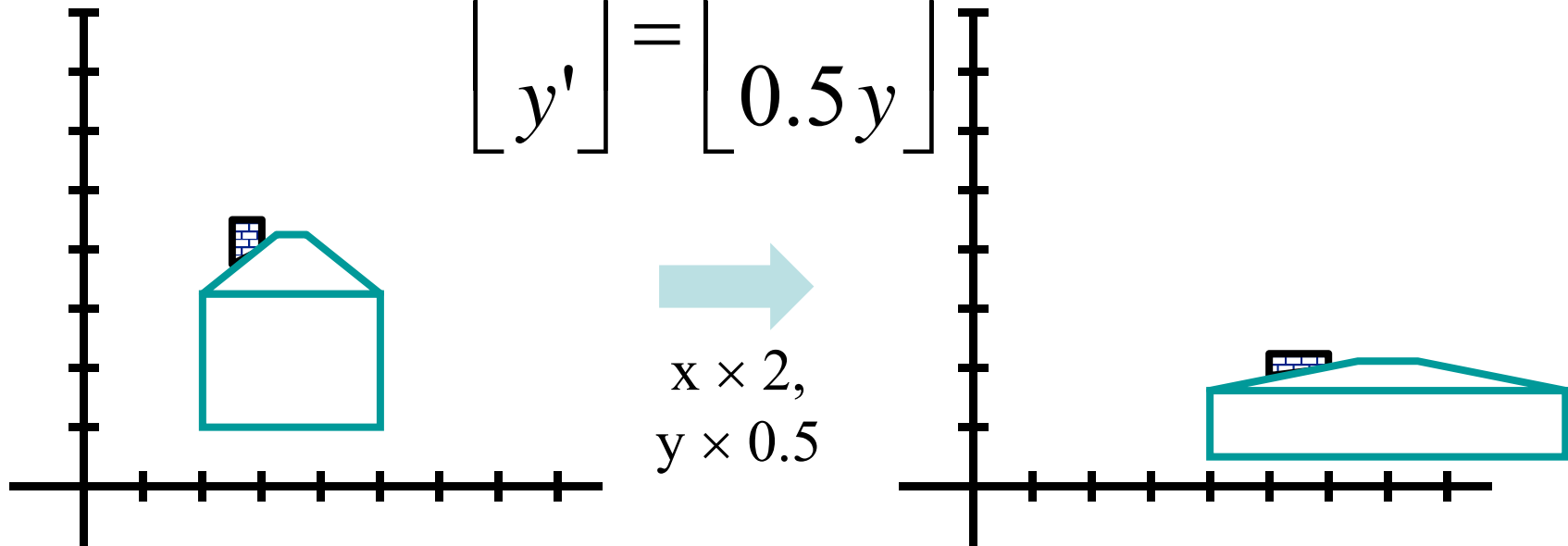


Scaling

- *Non-uniform scaling*: different scalars per component:

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = g\left(\begin{bmatrix} x' \\ y' \end{bmatrix}\right)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2x \\ 0.5y \end{bmatrix}$$



Scaling

- Scaling operation: $x' = ax$

$$y' = by$$

- Or, in matrix form:

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

What's inverse of S?

2-D Rotation

- This is easy to capture in matrix form:

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

- Even though $\sin(\theta)$ and $\cos(\theta)$ are nonlinear to θ ,
 - x' is a linear combination of x and y
 - y' is a linear combination of x and y
- What is the inverse transformation?
 - Rotation by $-\theta$
 - For rotation matrices, $\det(\mathbf{R}) = 1$ so $\mathbf{R}^{-1} = \mathbf{R}^T$

2x2 Matrices

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

2D Identity?

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

2D Scale around (0,0)?

$$\begin{aligned} x' &= s_x * x \\ y' &= s_y * y \end{aligned} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

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

2D Rotate around (0,0)?

$$x' = \cos \theta * x - \sin \theta * y$$

$$y' = \sin \theta * x + \cos \theta * y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Shear?

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

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

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

2x2 Matrices

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

2D Mirror about Y axis?

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

2D Mirror over (0,0)?

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

All 2D Linear Transformations

- Linear transformations are combinations of ...
 - Scale,
 - Rotation,
 - Shear, and
 - Mirror
- Properties of linear transformations:
 - Origin maps to origin
 - Lines map to lines
 - Parallel lines remain parallel
 - Ratios are preserved
 - Closed under composition

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

2x2 Matrices

- What types of transformations can **not** be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$

NO!

$$y' = y + t_y$$

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

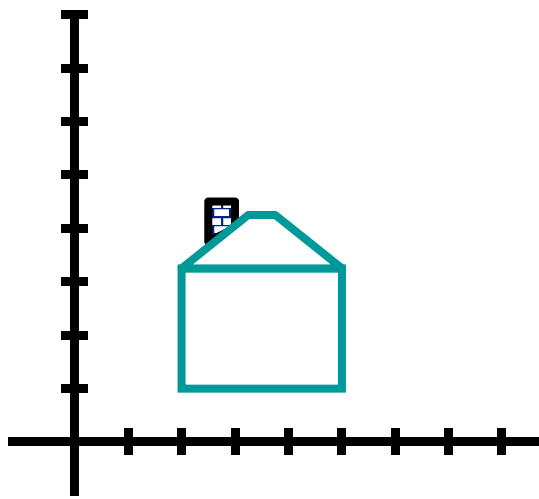
Translation

- Example of translation

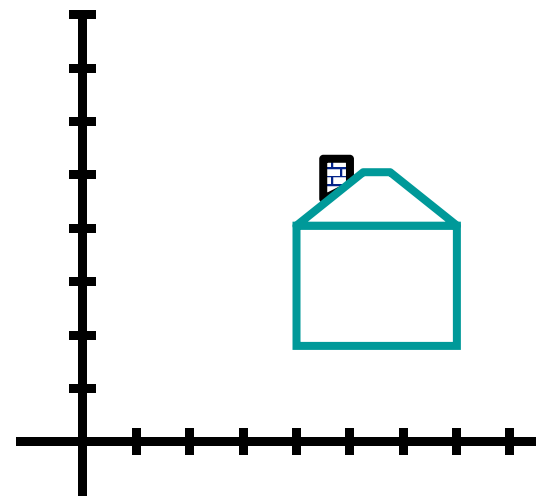
Homogeneous Coordinates



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



$$\begin{aligned} t_x &= 2 \\ t_y &= 1 \end{aligned}$$



Affine Transformations

- Affine transformations are combinations of ...
 - Linear transformations, and
 - Translations
- Properties of affine transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines remain parallel
 - Ratios are preserved
 - Closed under composition
 - Models change of basis

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

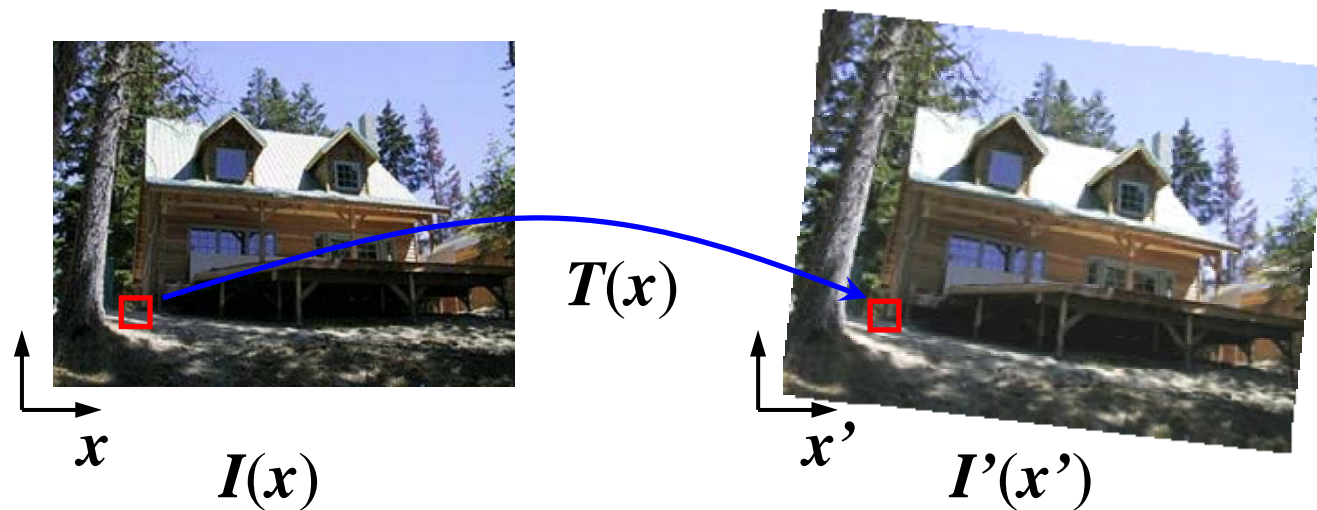
Projective Transformations

- Projective transformations ...
 - Affine transformations, and
 - Projective warps
- Properties of projective transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines do not necessarily remain parallel
 - Ratios are not preserved
 - Closed under composition
 - Models change of basis

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

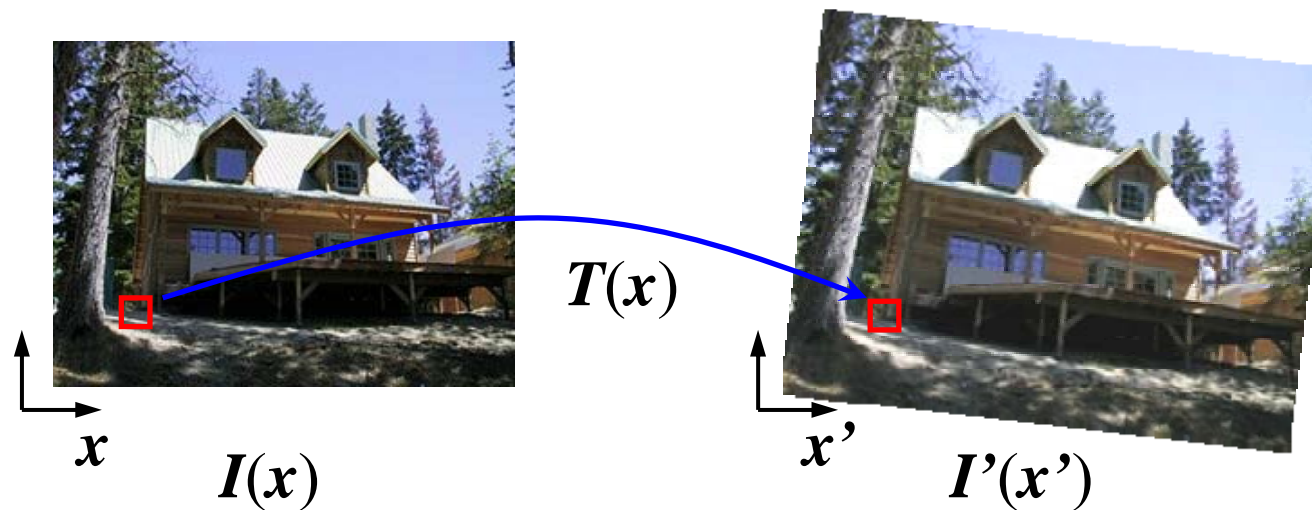
Image warping

- Given a coordinate transform $x' = T(x)$ and a source image $I(x)$, how do we compute a transformed image $I'(x') = I(T(x))$?



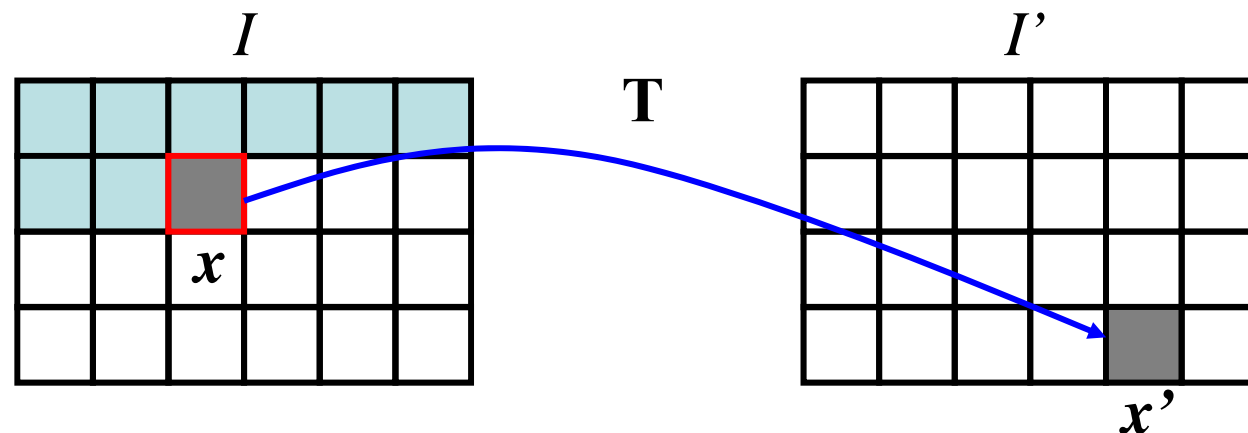
Forward warping

- Send each pixel $I(x)$ to its corresponding location $x' = T(x)$ in $I'(x')$



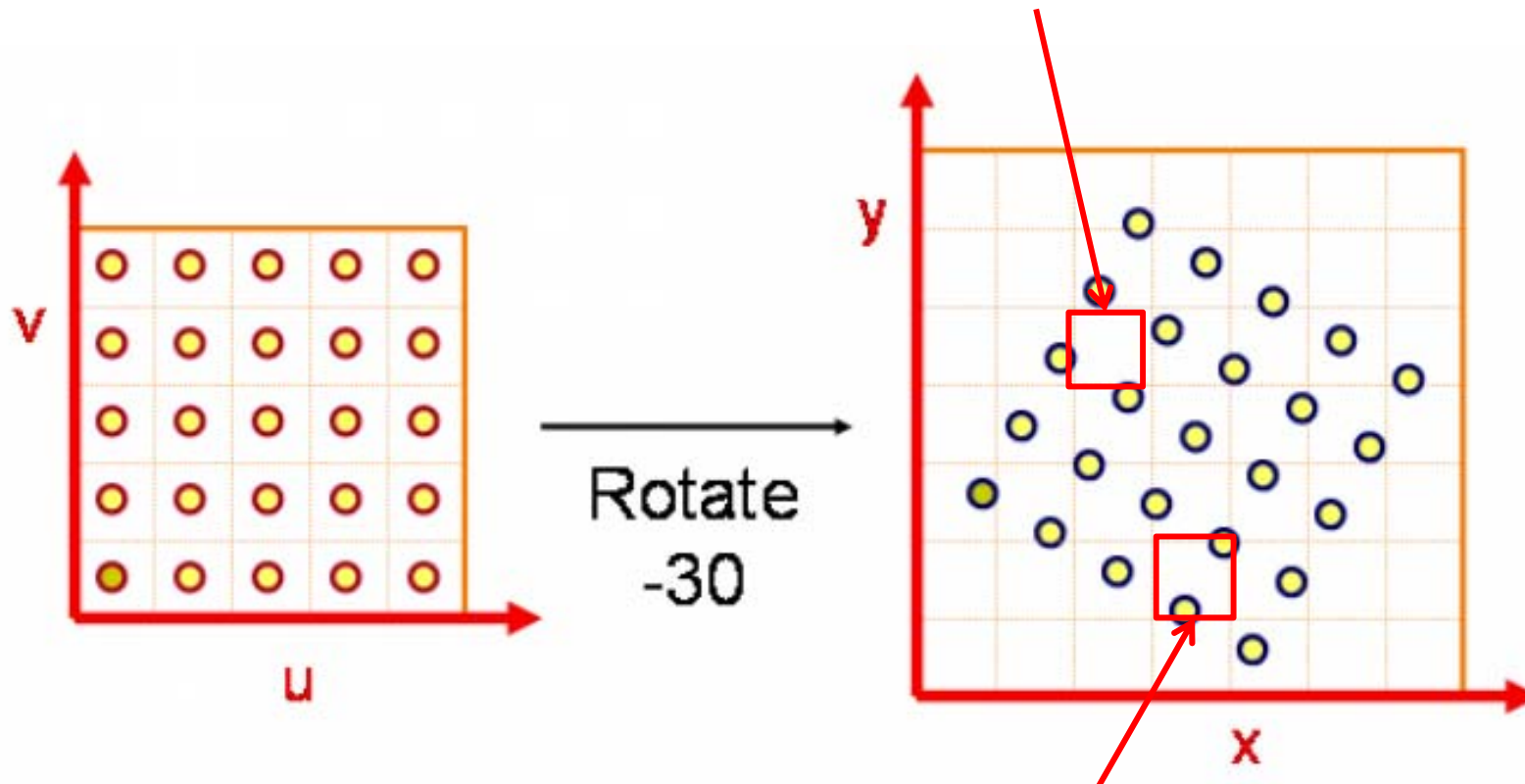
Forward warping

```
fwarp(I, I', T)
{
    for (y=0; y<I.height; y++)
        for (x=0; x<I.width; x++) {
            (x',y')=T(x,y);
            I'(x',y')=I(x,y);
        }
}
```



Forward warping

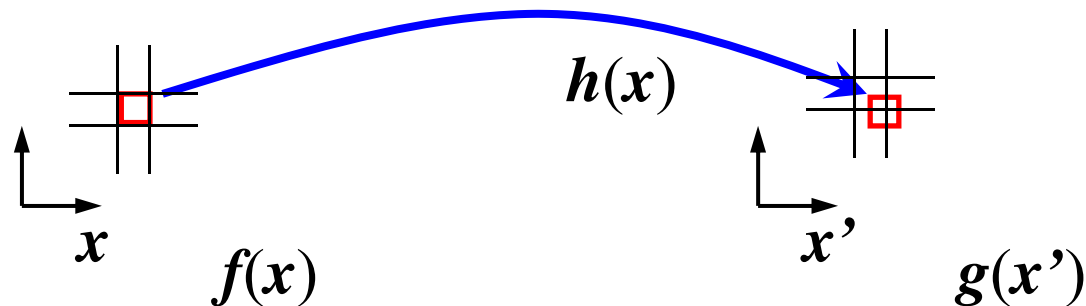
Some destination may
not be covered



Many source pixels could map
to the same destination

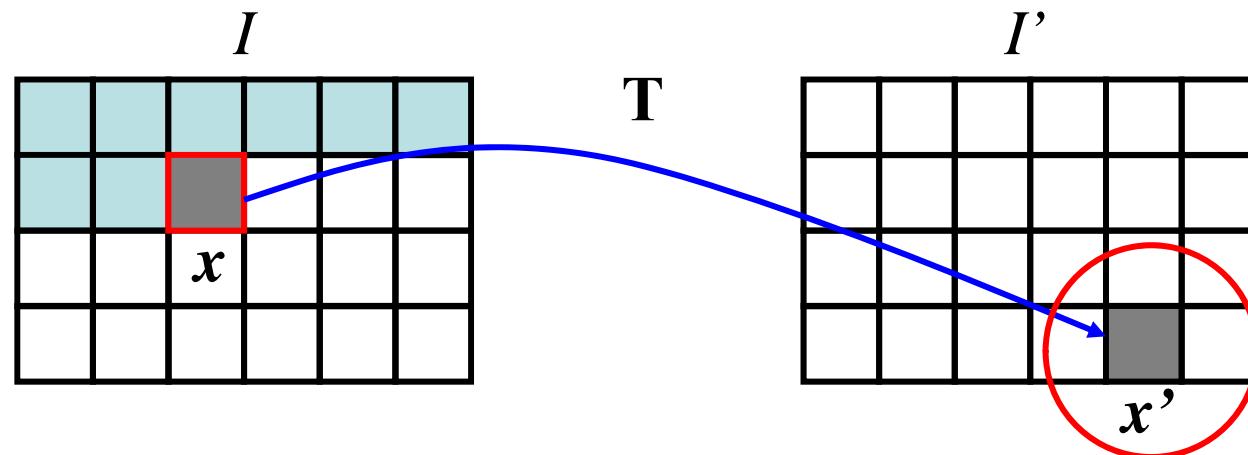
Forward warping

- Send each pixel $I(x)$ to its corresponding location $x' = T(x)$ in $I'(x')$
- What if pixel lands “between” two pixels?
- Will be there holes?
- Answer: add “contribution” to several pixels, normalize later (*splatting*)



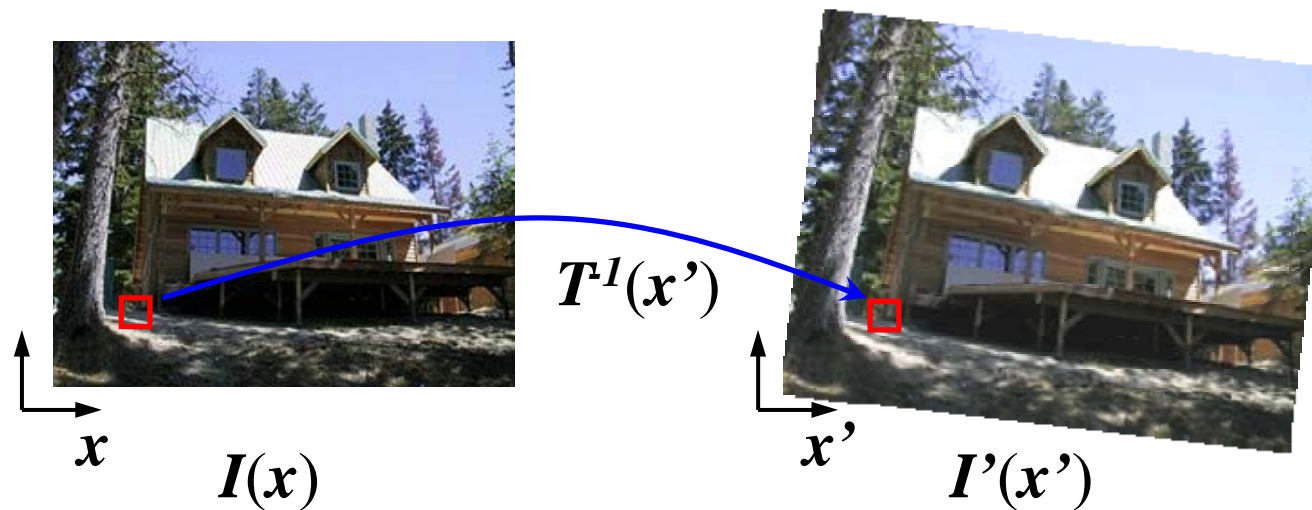
Forward warping

```
fwarp(I, I', T)
{
    for (y=0; y<I.height; y++)
        for (x=0; x<I.width; x++) {
            (x',y')=T(x,y);
            splatting(I',x',y',I(x,y),kernel);
        }
}
```



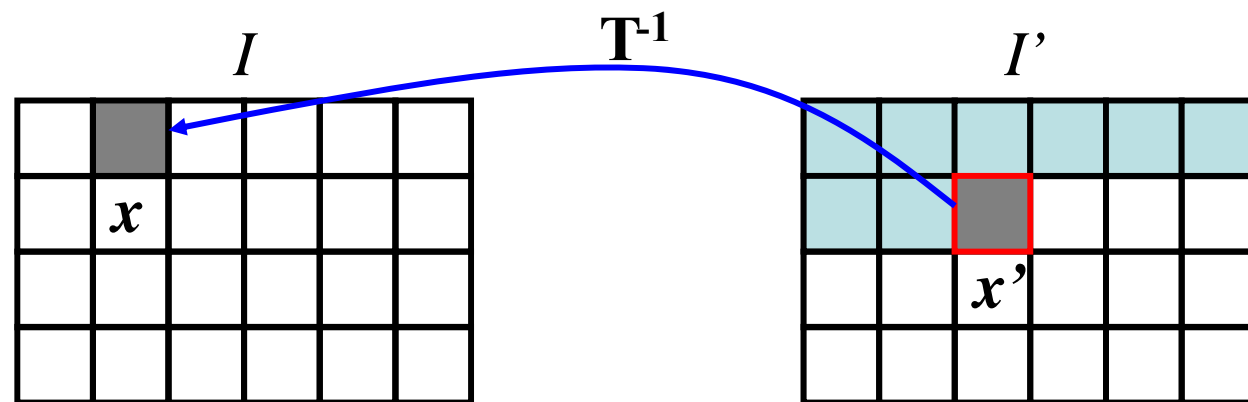
Inverse warping

- Get each pixel $I'(x')$ from its corresponding location $x = T^{-1}(x')$ in $I(x)$



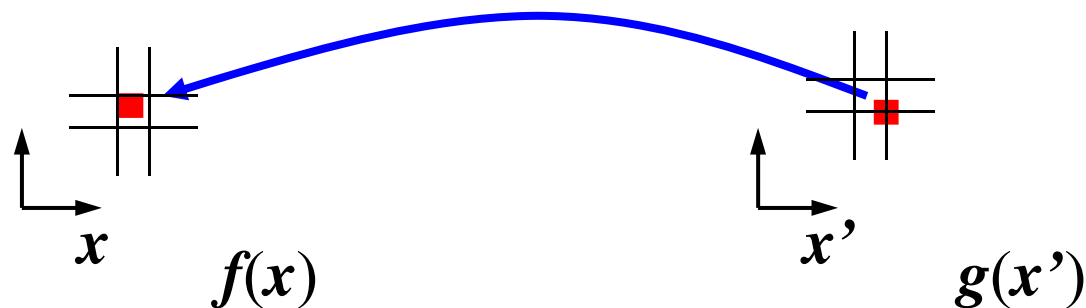
Inverse warping

```
iwarp(I, I', T)
{
    for (y'=0; y'<I'.height; y'++)
        for (x'=0; x'<I'.width; x'++) {
            (x,y)=T-1(x',y');
            I'(x',y')=I(x,y);
        }
}
```



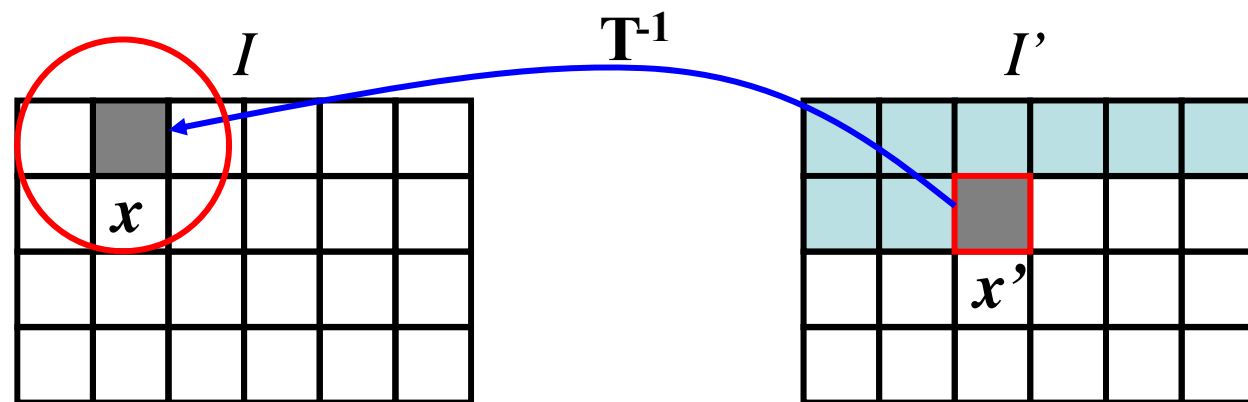
Inverse warping

- Get each pixel $I'(x')$ from its corresponding location $x = T^{-1}(x')$ in $I(x)$
- What if pixel comes from “between” two pixels?
- Answer: *resample* color value from *interpolated (prefiltered)* source image



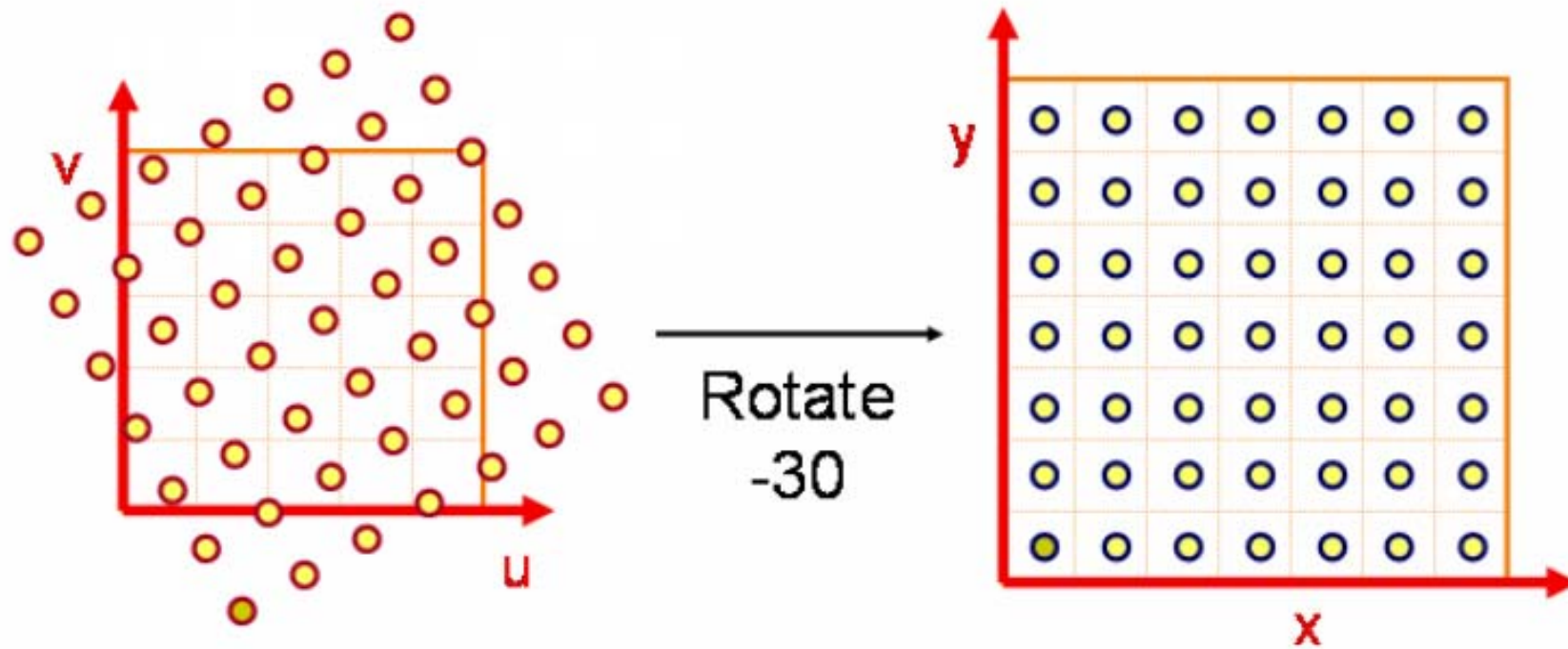
Inverse warping

```
iwarp(I, I', T)
{
    for (y'=0; y'<I'.height; y'++)
        for (x'=0; x'<I'.width; x'++) {
            (x,y)=T-1(x',y');
            I'(x',y')=Reconstruct(I,x,y,kernel);
        }
}
```



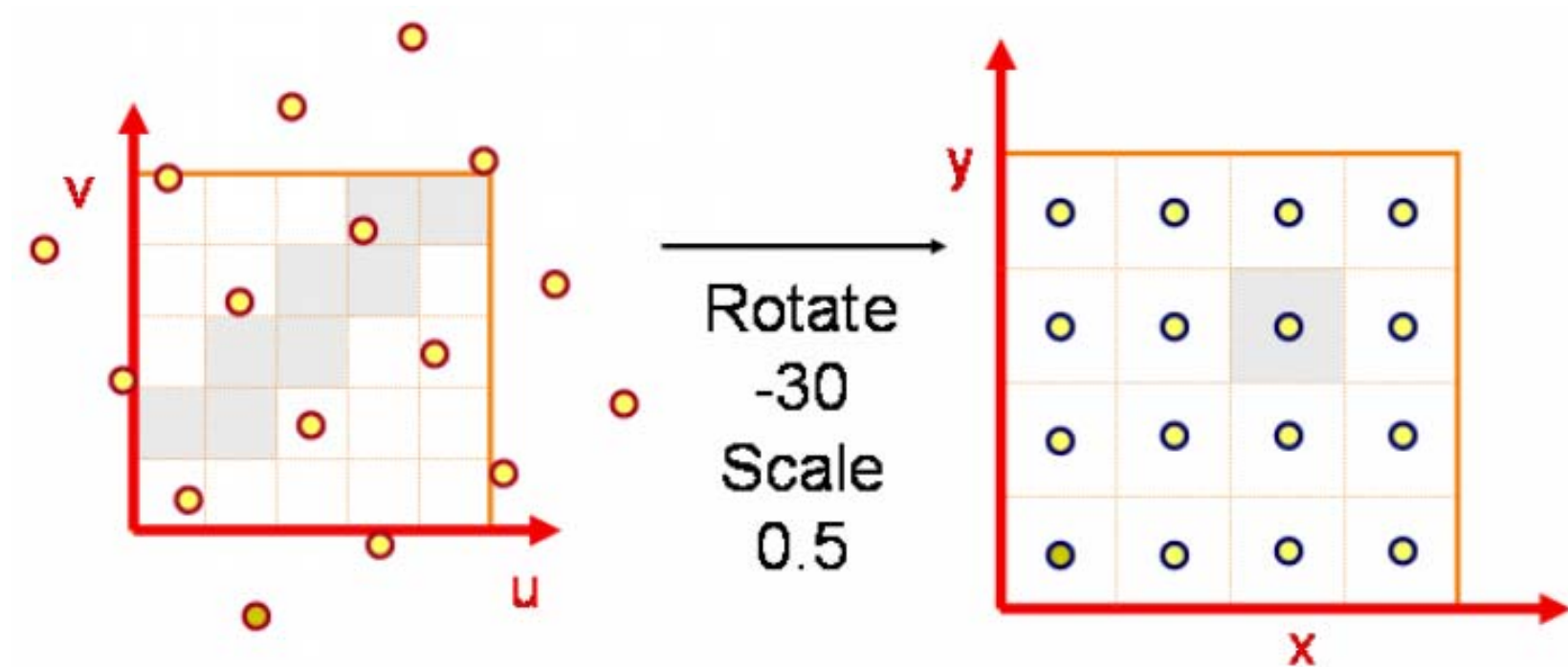
Inverse warping

- No hole, but must resample
- What value should you take for non-integer coordinate? Closest one?



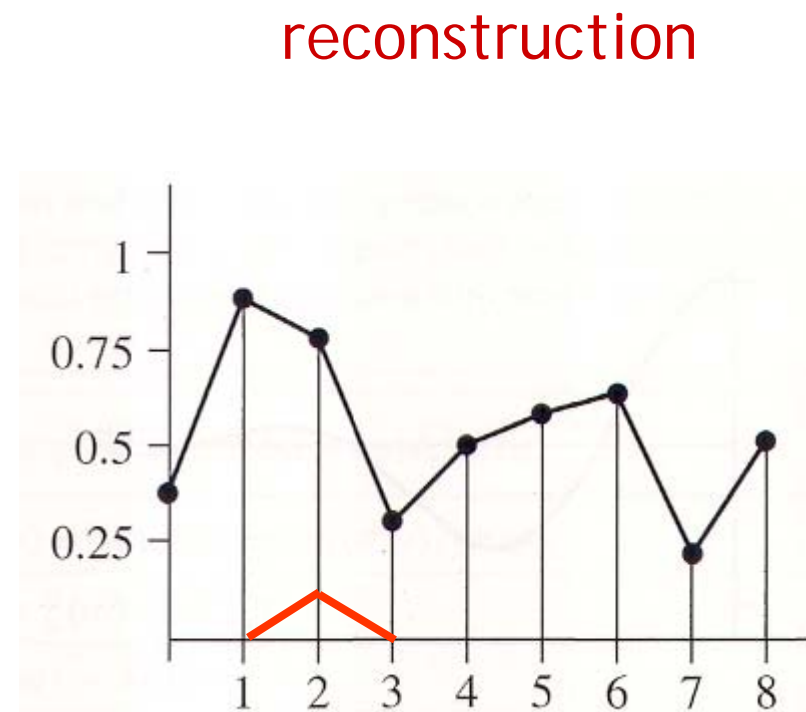
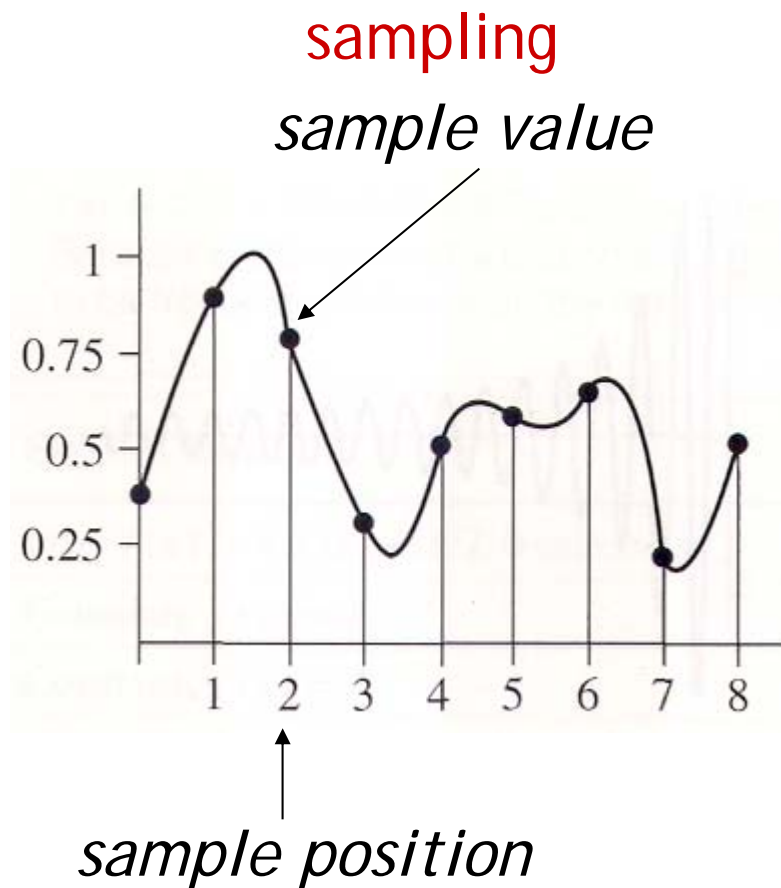
Inverse warping

- It could cause aliasing



Reconstruction

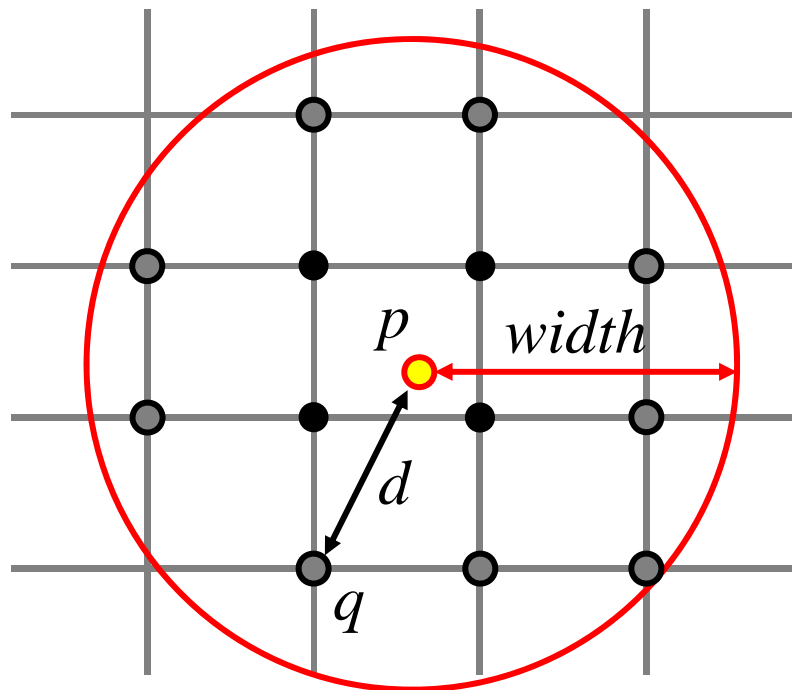
- Reconstruction generates an approximation to the original function. Error is called aliasing.



Reconstruction

- Computed weighted sum of pixel neighborhood; output is weighted average of input, where weights are normalized values of filter kernel k

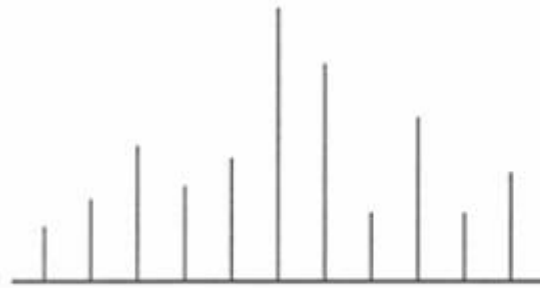
$$p = \frac{\sum_i k(q_i) q_i}{\sum_i k(q_i)}$$



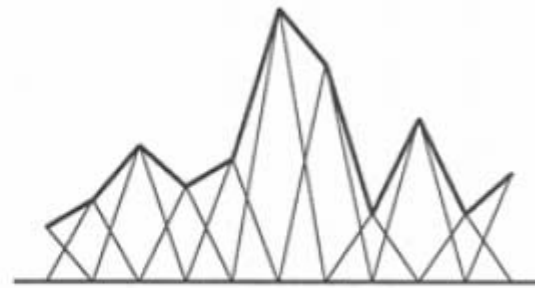
```

color=0;
weights=0;
for all q's dist < width
    d = dist(p, q);
    w = kernel(d);
    color += w*q.color;
    weights += w;
p.Color = color/weights;
  
```

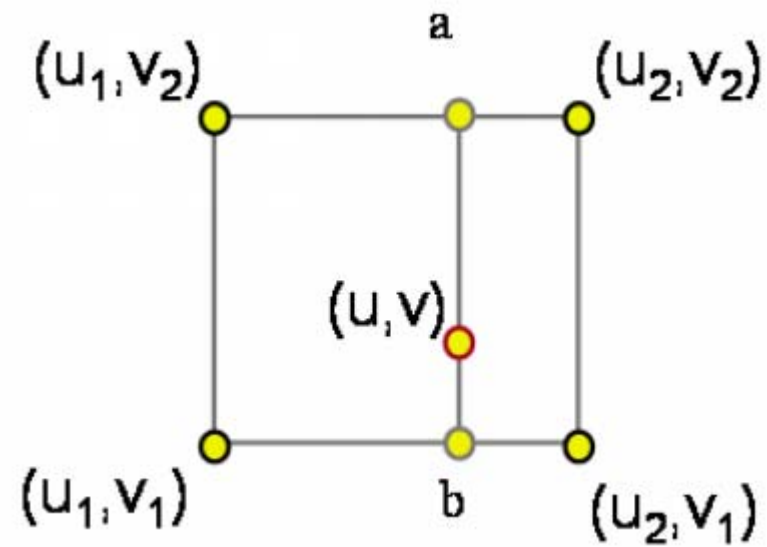
Triangle filter



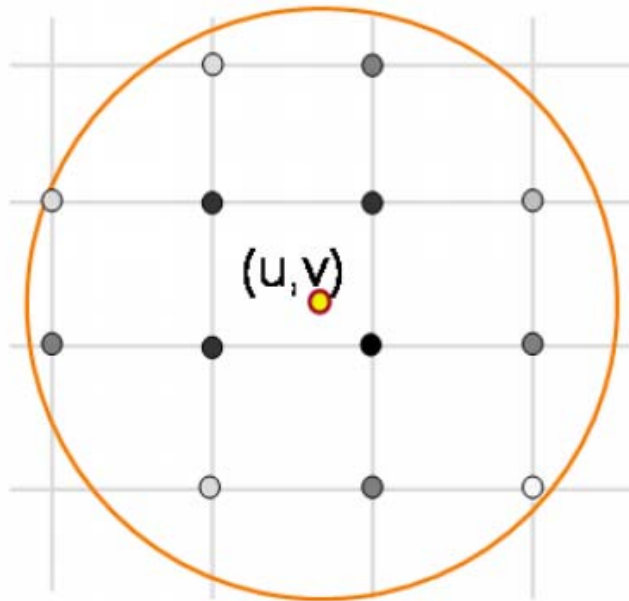
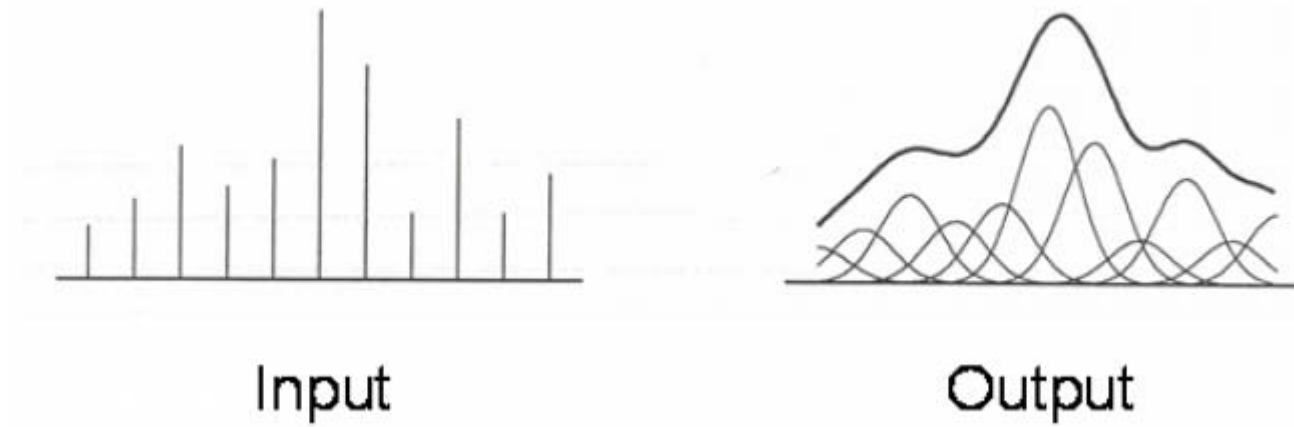
Input



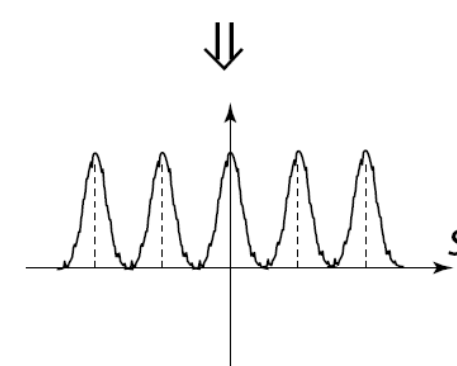
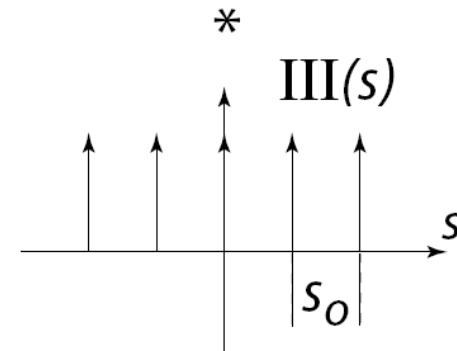
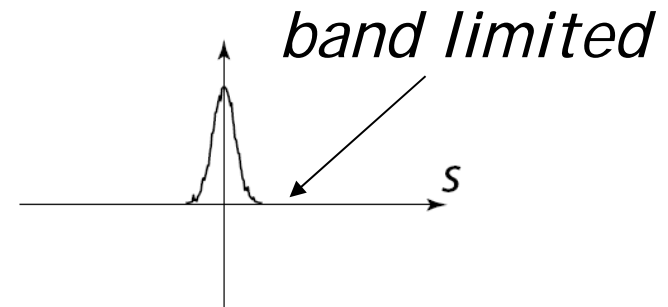
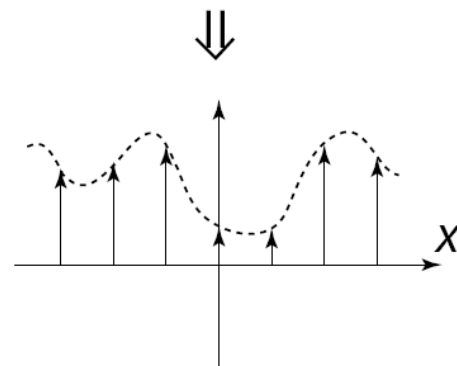
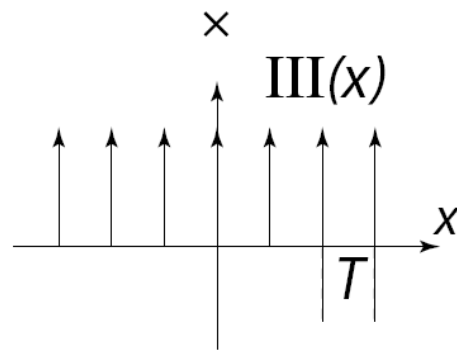
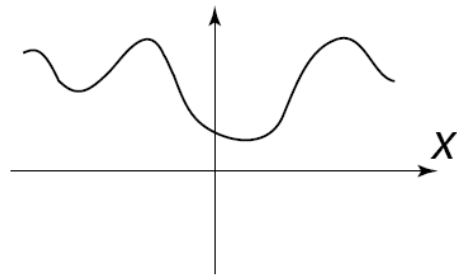
Output



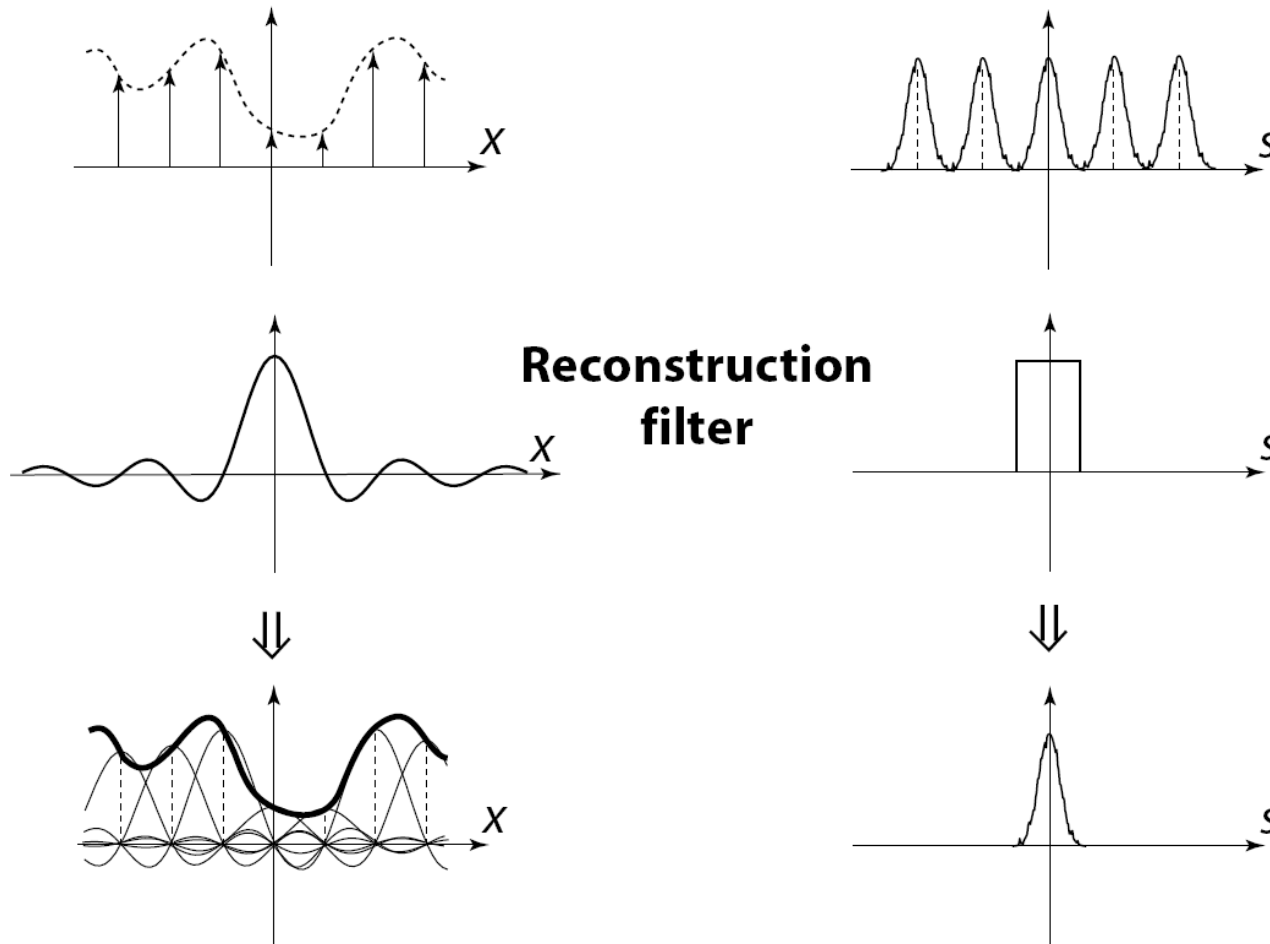
Gaussian filter



Sampling



Reconstruction



The reconstructed function is obtained by interpolating among the samples in some manner

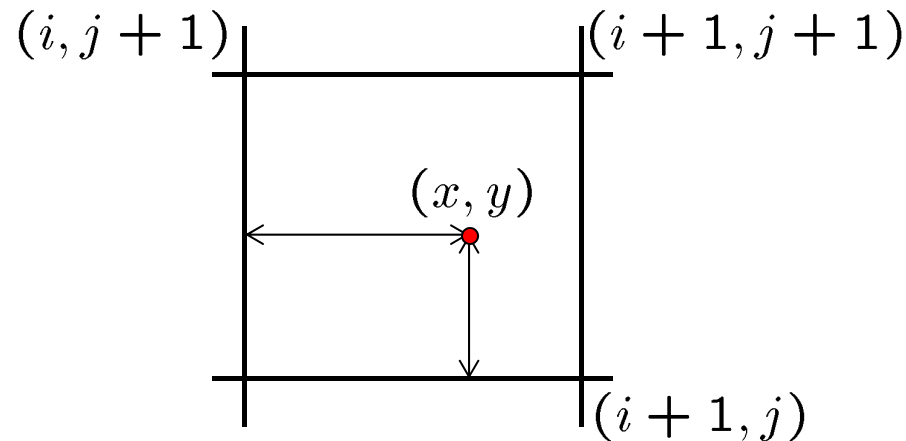
Reconstruction (interpolation)

- Possible reconstruction filters (kernels):
 - nearest neighbor
 - bilinear
 - bicubic
 - sinc (optimal reconstruction)



Bilinear interpolation (triangle filter)

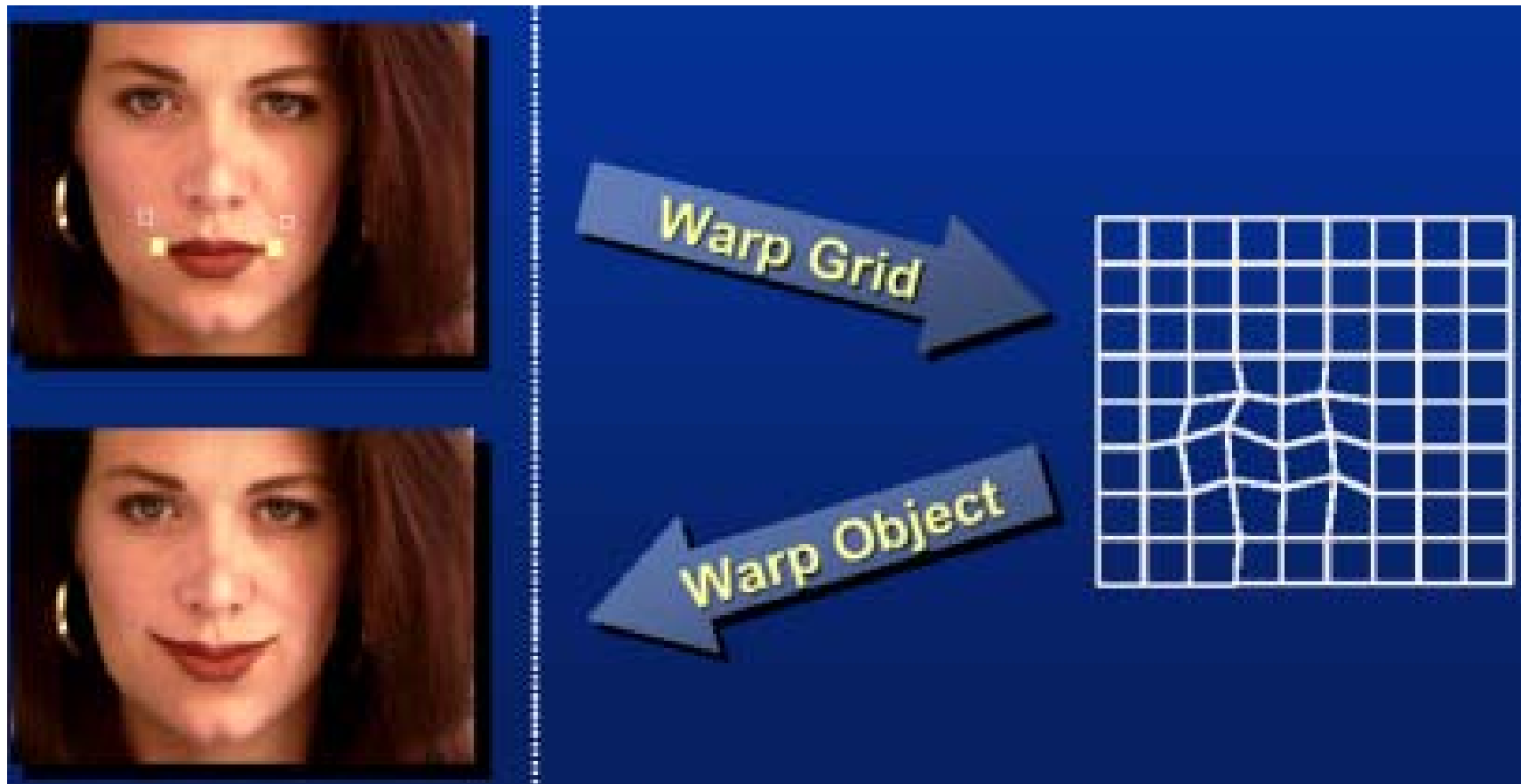
- A simple method for resampling images



$$\begin{aligned} f(x, y) = & (1 - a)(1 - b) f[i, j] \\ & + a(1 - b) f[i + 1, j] \\ & + ab f[i + 1, j + 1] \\ & + (1 - a)b f[i, j + 1] \end{aligned}$$

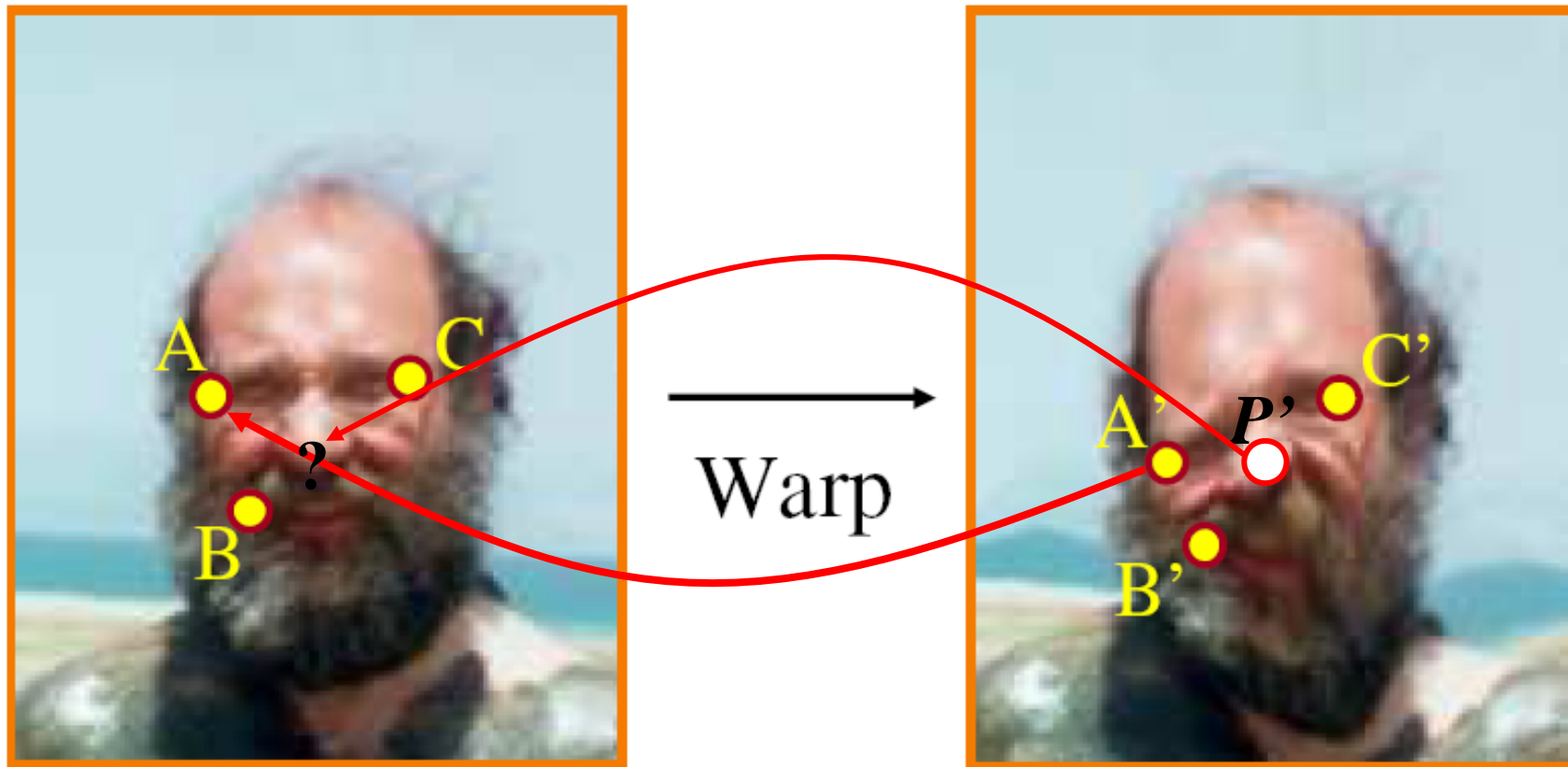
Non-parametric image warping

- Specify a more detailed warp function
- Splines, meshes, optical flow (per-pixel motion)



Non-parametric image warping

- Mappings implied by correspondences
- Inverse warping

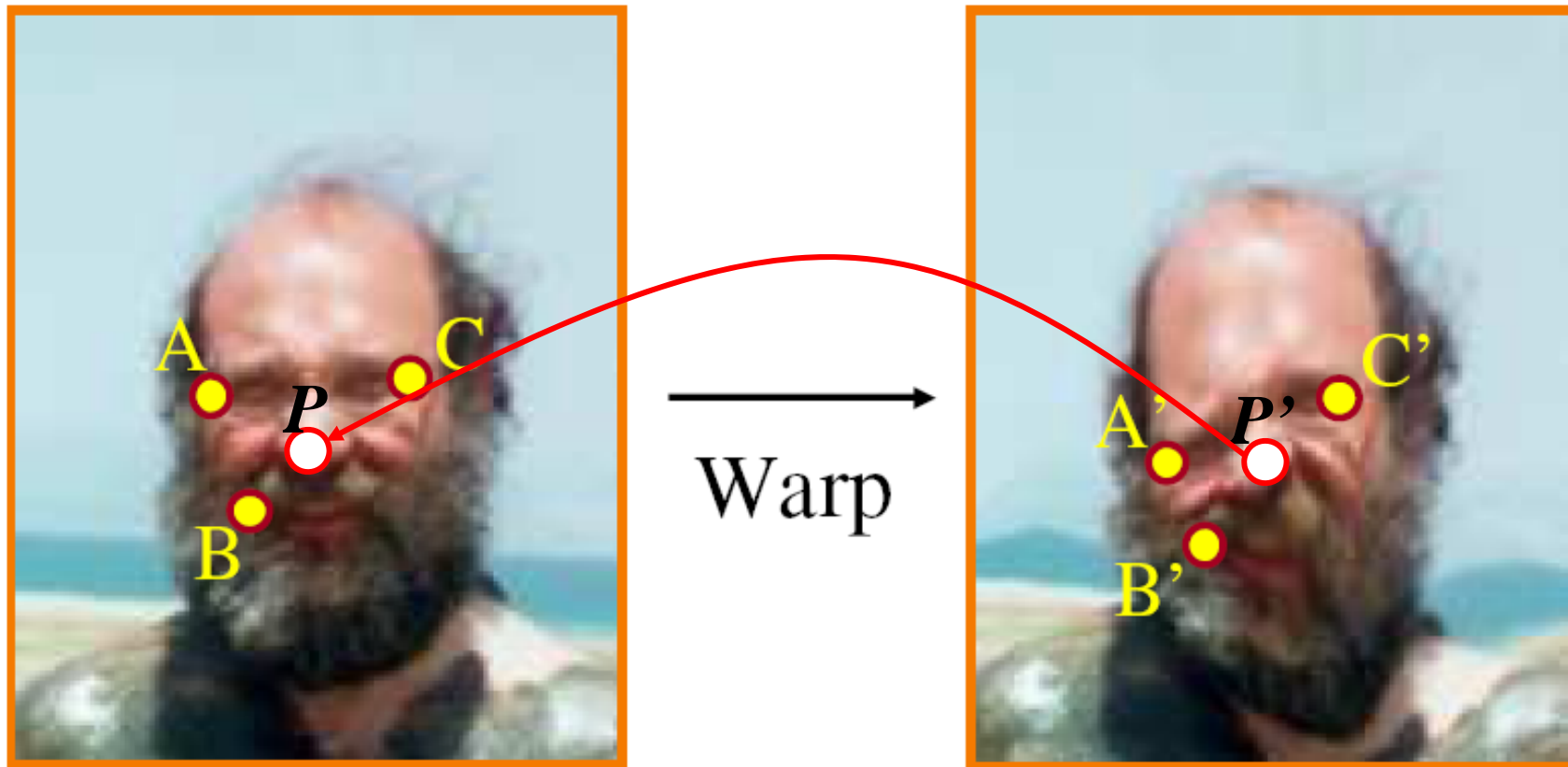


Non-parametric image warping

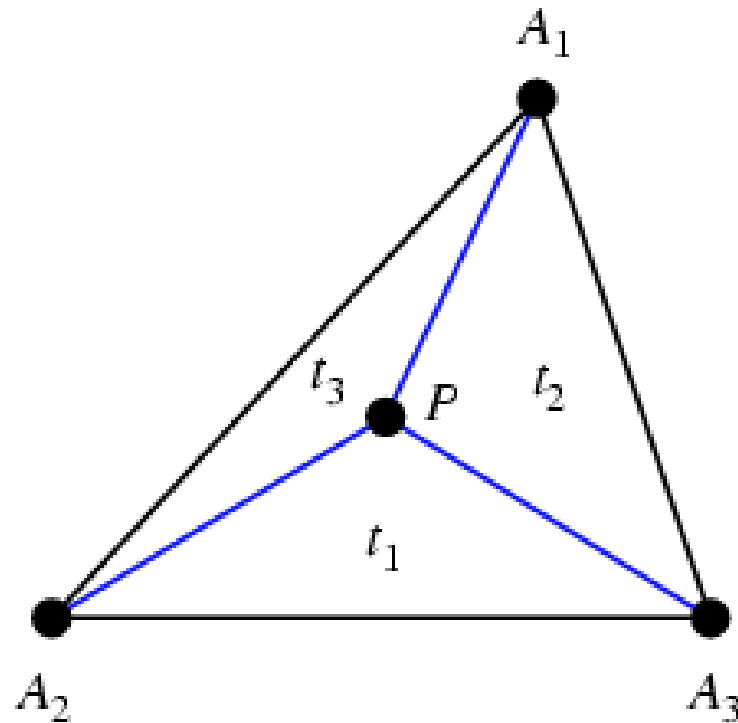
$$P = w_A A + w_B B + w_C C$$

$$P' = w_A A' + w_B B' + w_C C'$$

Barycentric coordinate



Barycentric coordinates



$$P = t_1 A_1 + t_2 A_2 + t_3 A_3$$

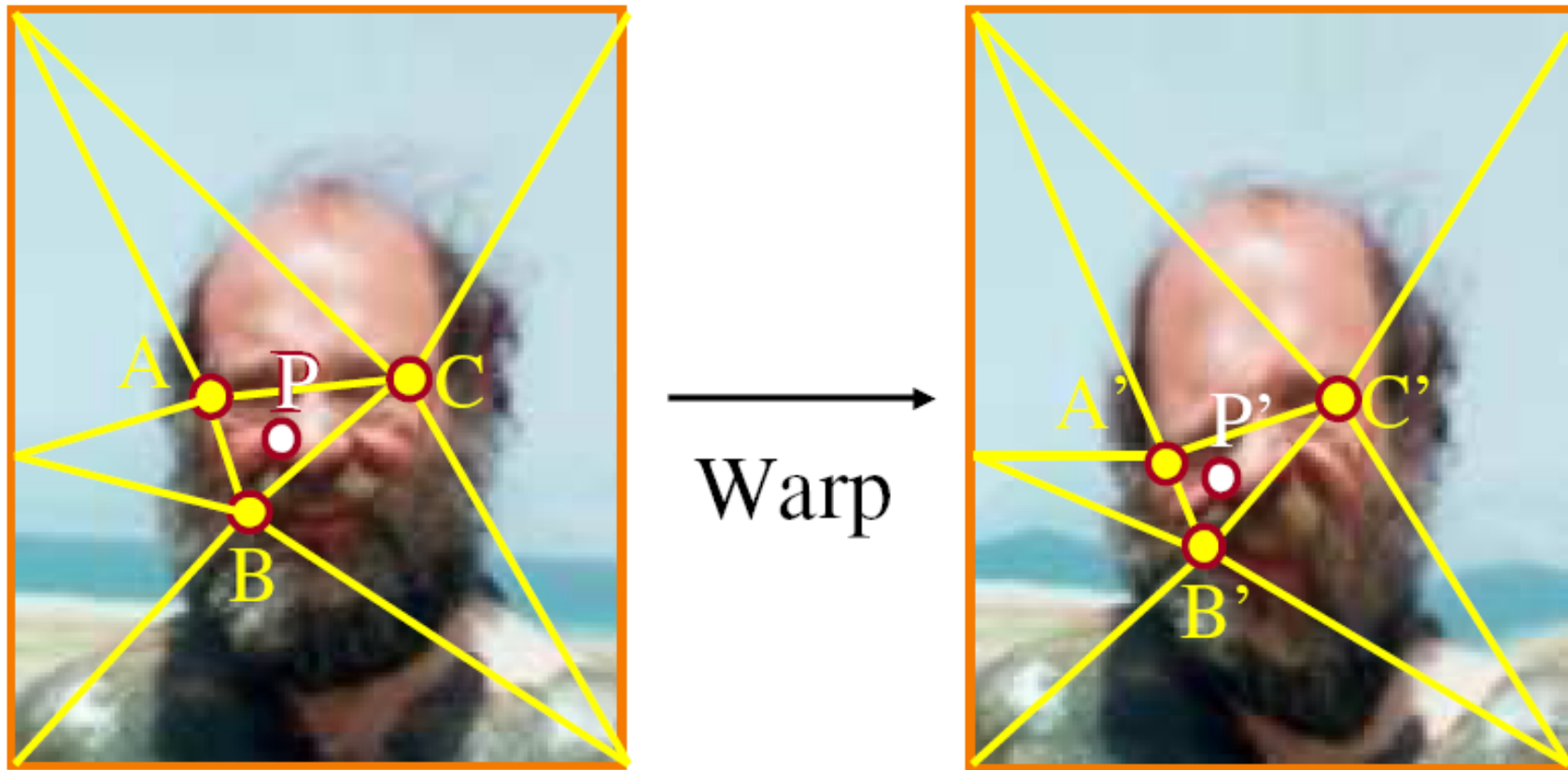
$$t_1 + t_2 + t_3 = 1$$

Non-parametric image warping

$$P = w_A A + w_B B + w_C C$$

$$P' = w_A A' + w_B B' + w_C C'$$

Barycentric coordinate



Non-parametric image warping

Gaussian $\rho(r) = e^{-\beta r^2}$
 thin plate
 spline $\rho(r) = r^2 \log(r)$

$$\Delta P = \frac{1}{K} \sum_i k_{x_i}(P') \Delta X_i$$

radial basis function



→
Warp



Image warping

- Warping is a useful operation for mosaics, retargeting, video matching, view interpolation and so on.

An application of image warping: face beautification

Data-driven facial beautification



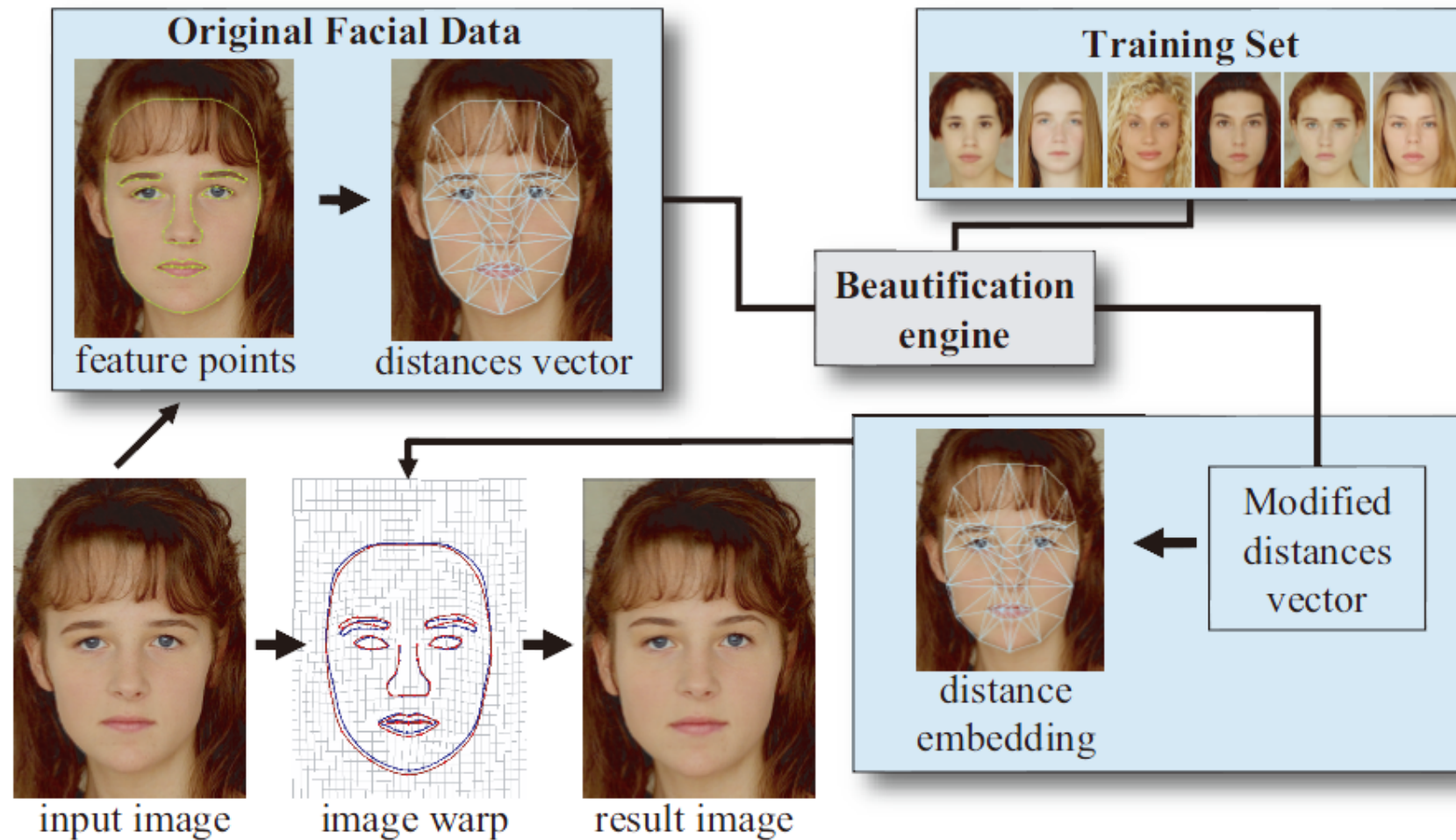
Facial beautification



Facial beautification



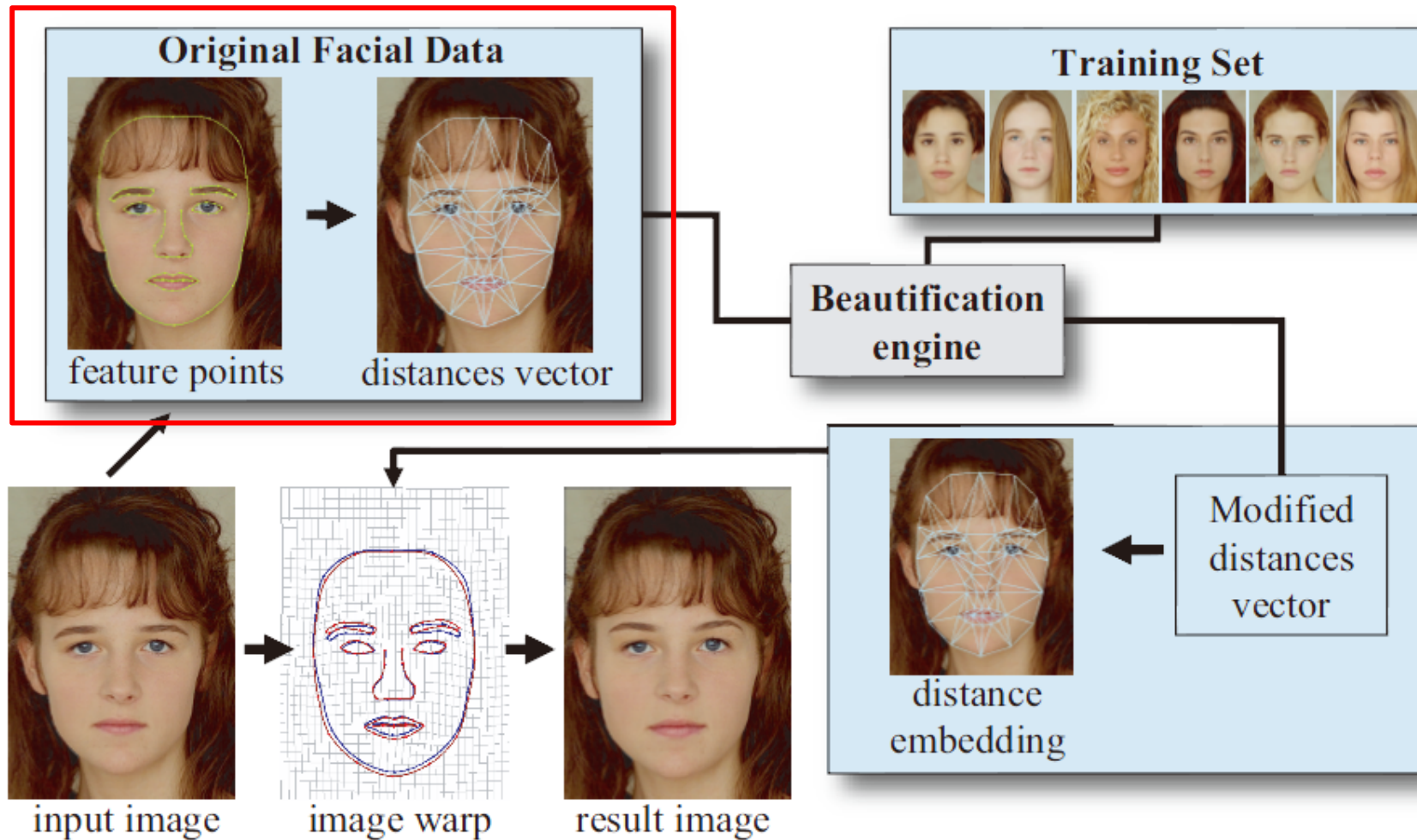
Facial beautification



Training set

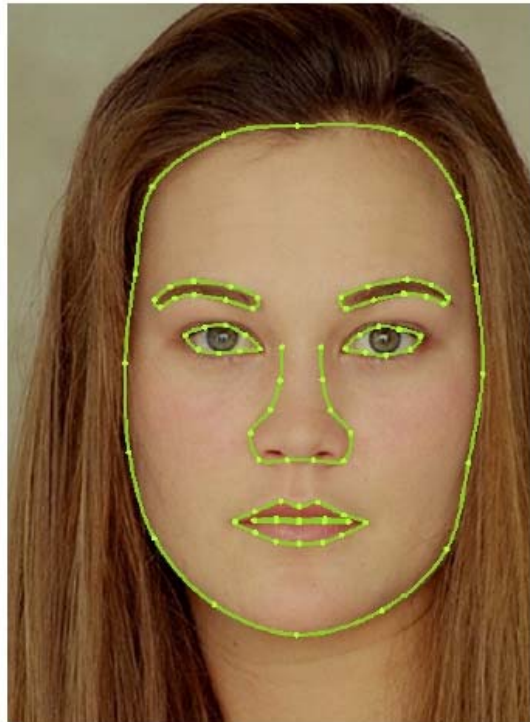
- Face images
 - 92 young Caucasian female
 - 33 young Caucasian male

Feature extraction

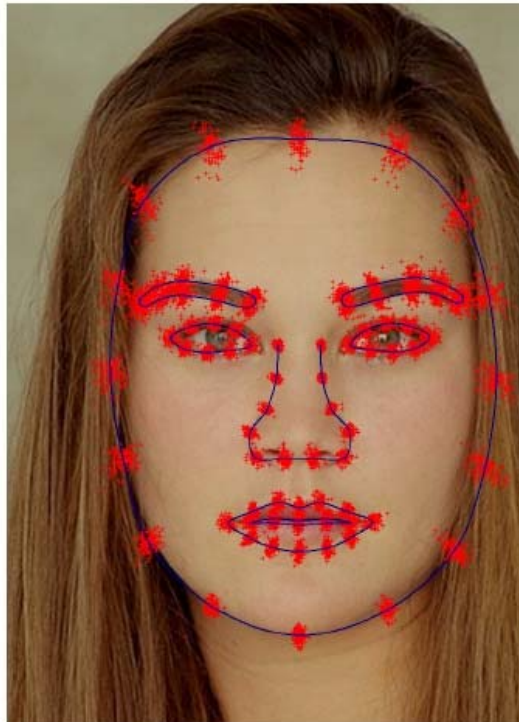


Feature extraction

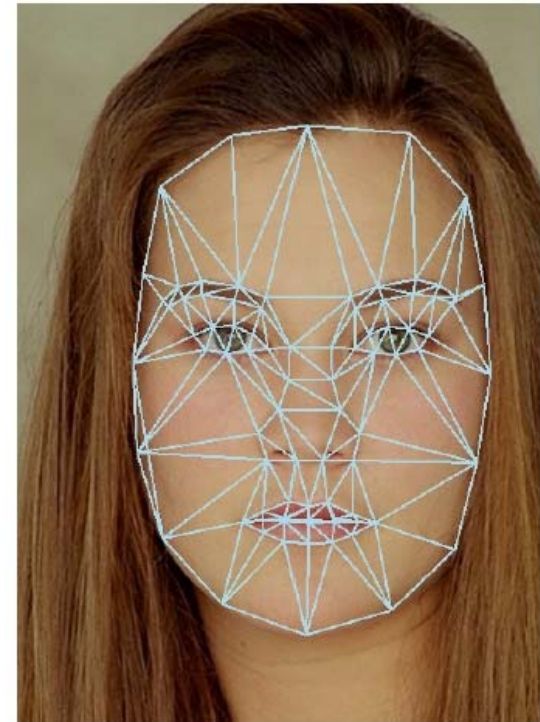
- Extract 84 feature points by BTSM
- Delaunay triangulation -> 234D distance vector (normalized by the square root of face area)



BTSM

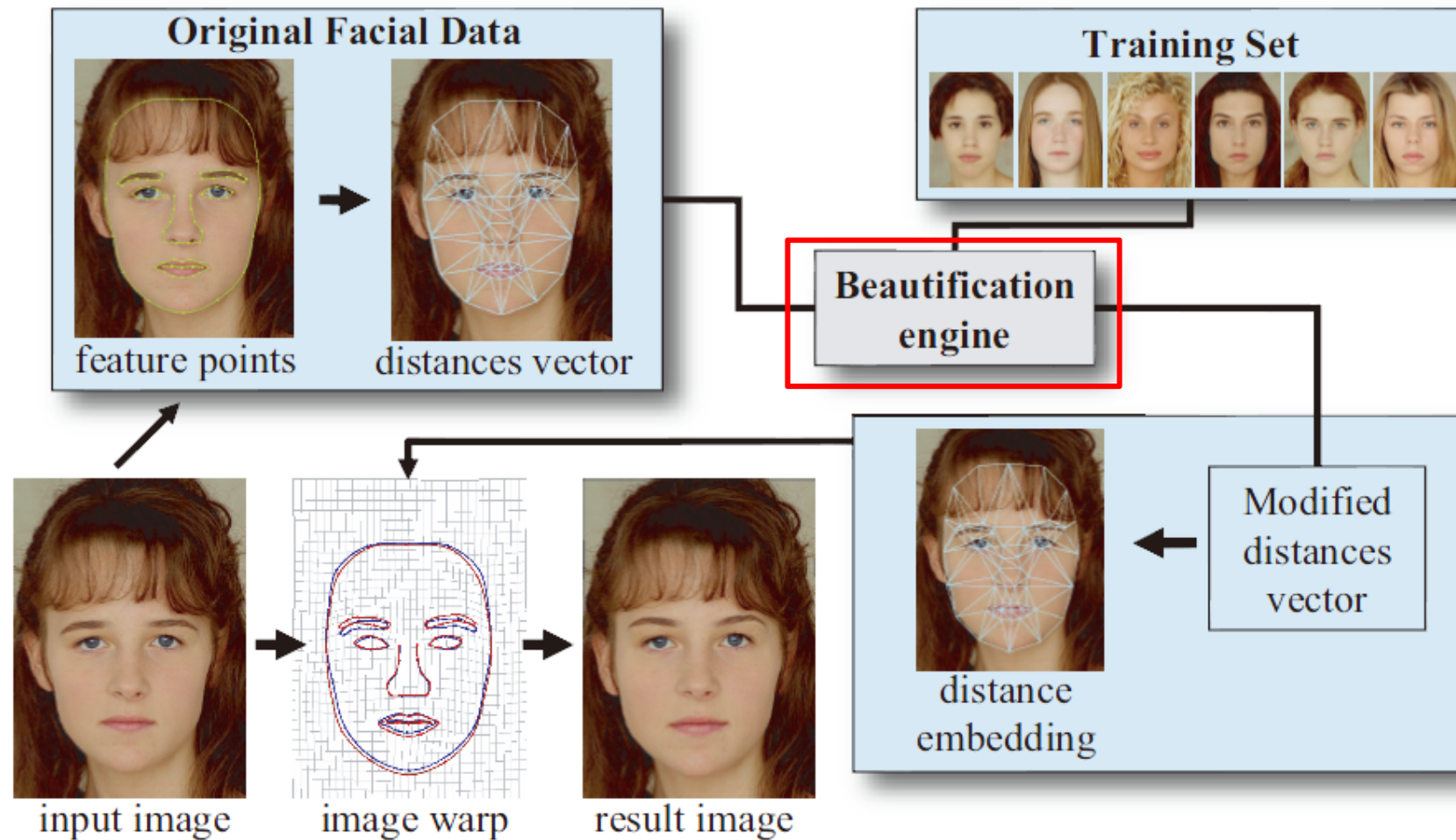


scatter plot for
all training faces



234D vector

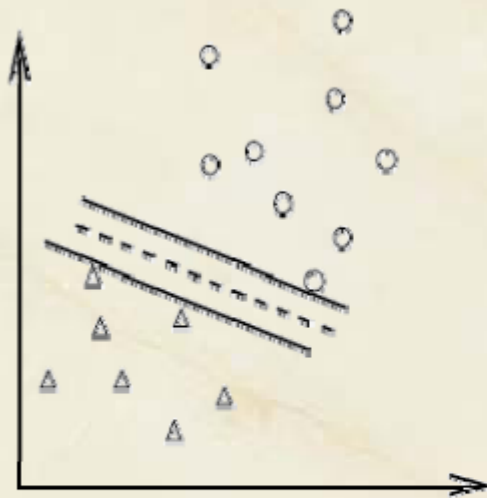
Beautification engine



Support vector regression (SVR)

- Similar concept to SVM, but for regression
- RBF kernels
- $f_b(v)$

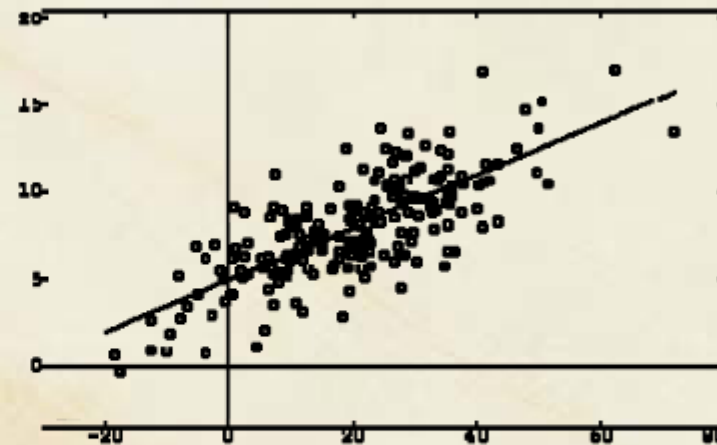
SVM $f(x) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$



$$\text{minimize } \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) \geq 1$$

SVR $f(x) = \mathbf{w}^T \mathbf{x} + b$



$$\text{minimize } \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } \|y_i - (\mathbf{w}^T \cdot \mathbf{x}_i + b)\| \leq \varepsilon$$

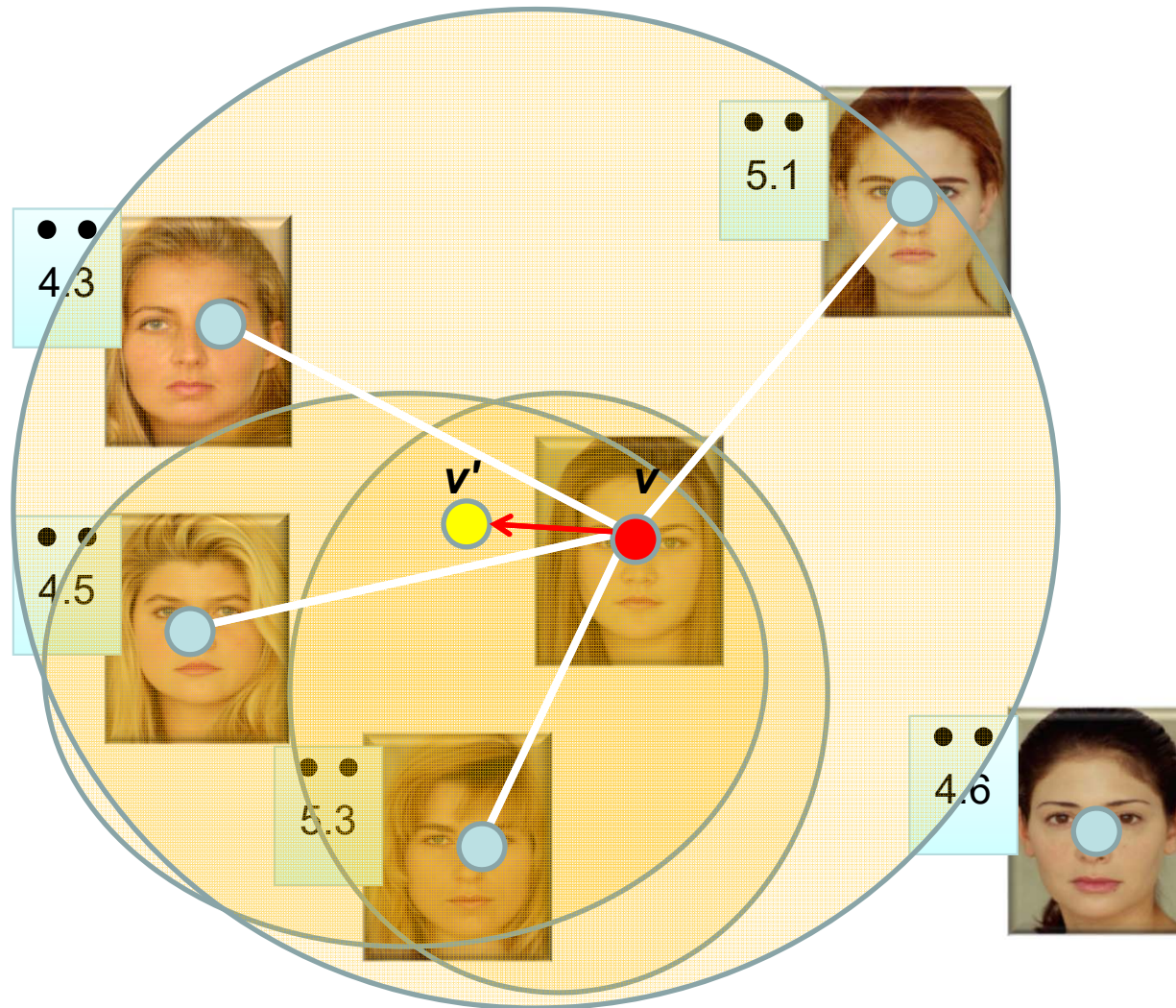
Beautification process

- Given the normalized distance vector v , generate a nearby vector v' so that

$$f_b(v') > f_b(v)$$

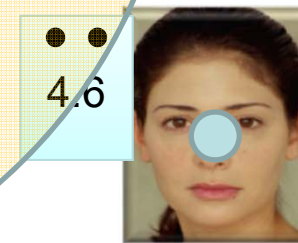
- Two options
 - KNN-based
 - SVR-based

KNN-based beautification



$$w_i = \frac{b_i}{\|\mathbf{v} - \mathbf{v}_i\|}$$

$$\mathbf{v}' = \frac{\sum_{i=1}^K w_i \mathbf{v}_i}{\sum_{i=1}^K w_i}$$



SVR-based beautification

- Directly use f_b to seek v'

$$\mathbf{v}' = \underset{\mathbf{u}}{\operatorname{argmin}} E(\mathbf{u}), \text{ where } E(\mathbf{u}) = -f_b(\mathbf{u})$$

- Use standard no-derivative direction set method for minimization
- Features were reduced to 35D by PCA

SVR-based beautification

- Problems: it sometimes yields distance vectors corresponding to invalid human face
- Solution: add log-likelihood term (LP)

$$E(\mathbf{u}) = (\alpha - 1)f_b(\mathbf{u}) - \alpha LP(\mathbf{u})$$

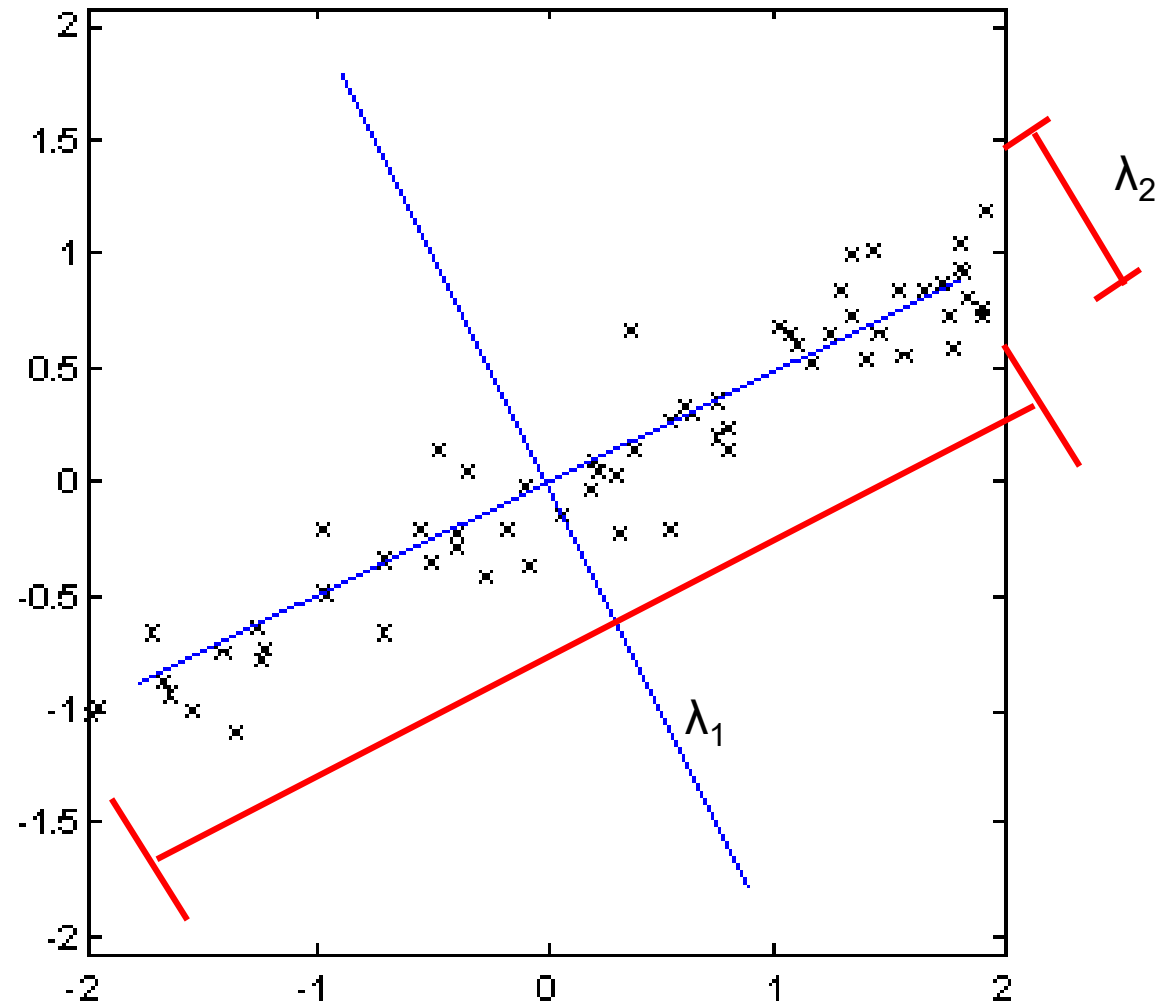
- LP is approximated by modeling face space as a multivariate Gaussian distribution

$$P(\hat{\mathbf{u}}) = \frac{1}{(2\pi)^{N/2} \sqrt{\prod_i \lambda_i}} \prod_i \exp\left(\frac{-\beta_i^2}{2\lambda_i}\right)$$

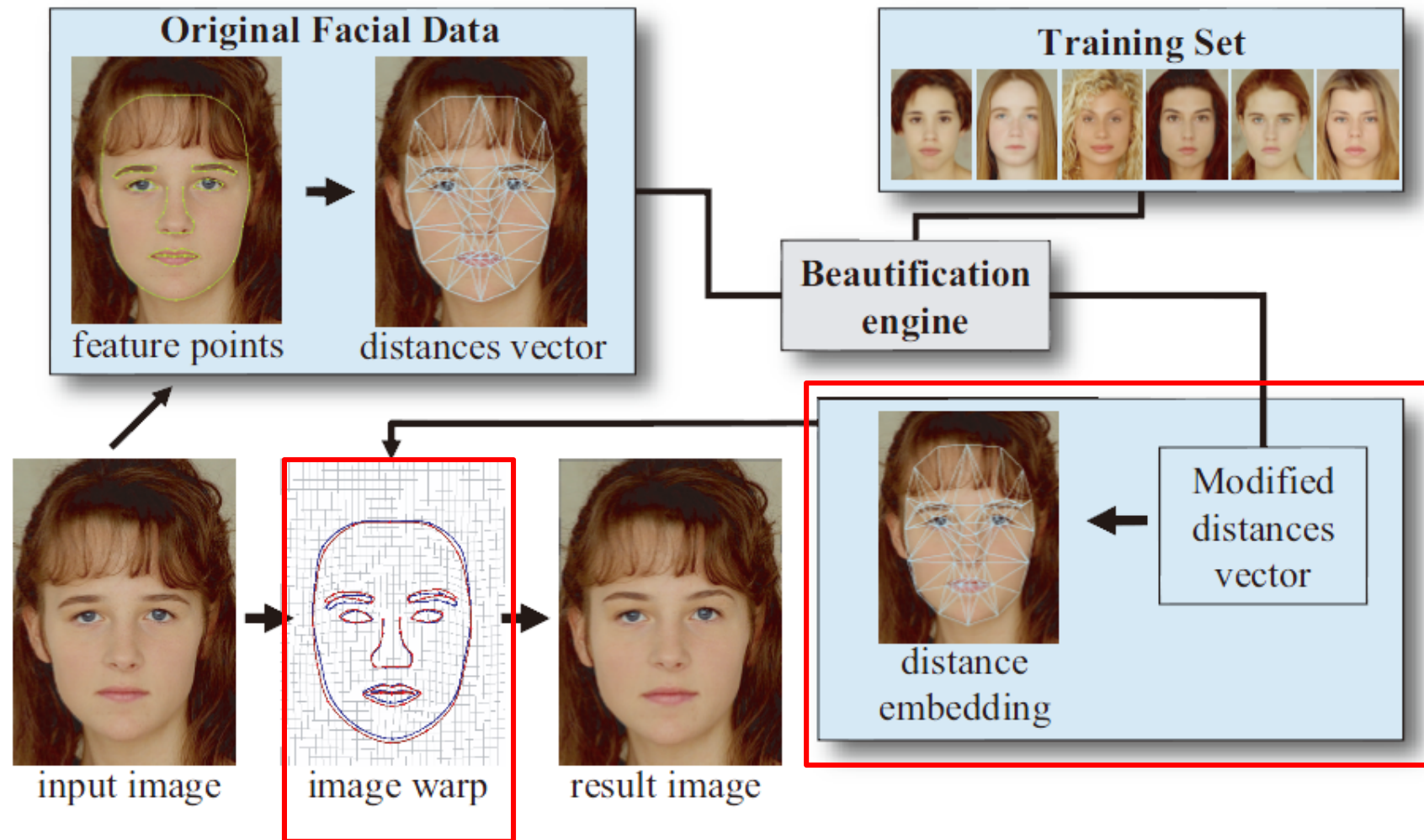
$\hat{\mathbf{u}}$'s i-th component
 \mathbf{u} 's projection in PCA space
 i-th eigenvalue

$$LP(\hat{\mathbf{u}}) = \sum \frac{-\beta_i^2}{2\lambda_i} + \text{const}$$

PCA




Embedding and warping



Distance embedding

- Convert modified distance vector v' to a new face landmark

$$E(q_1, \dots, q_N) = \sum_{e_{ij}} \alpha_{ij} \left(\|q_i - q_j\|^2 - d_{ij}^2 \right)^2$$


1 if i and j belong to different facial features
10 otherwise

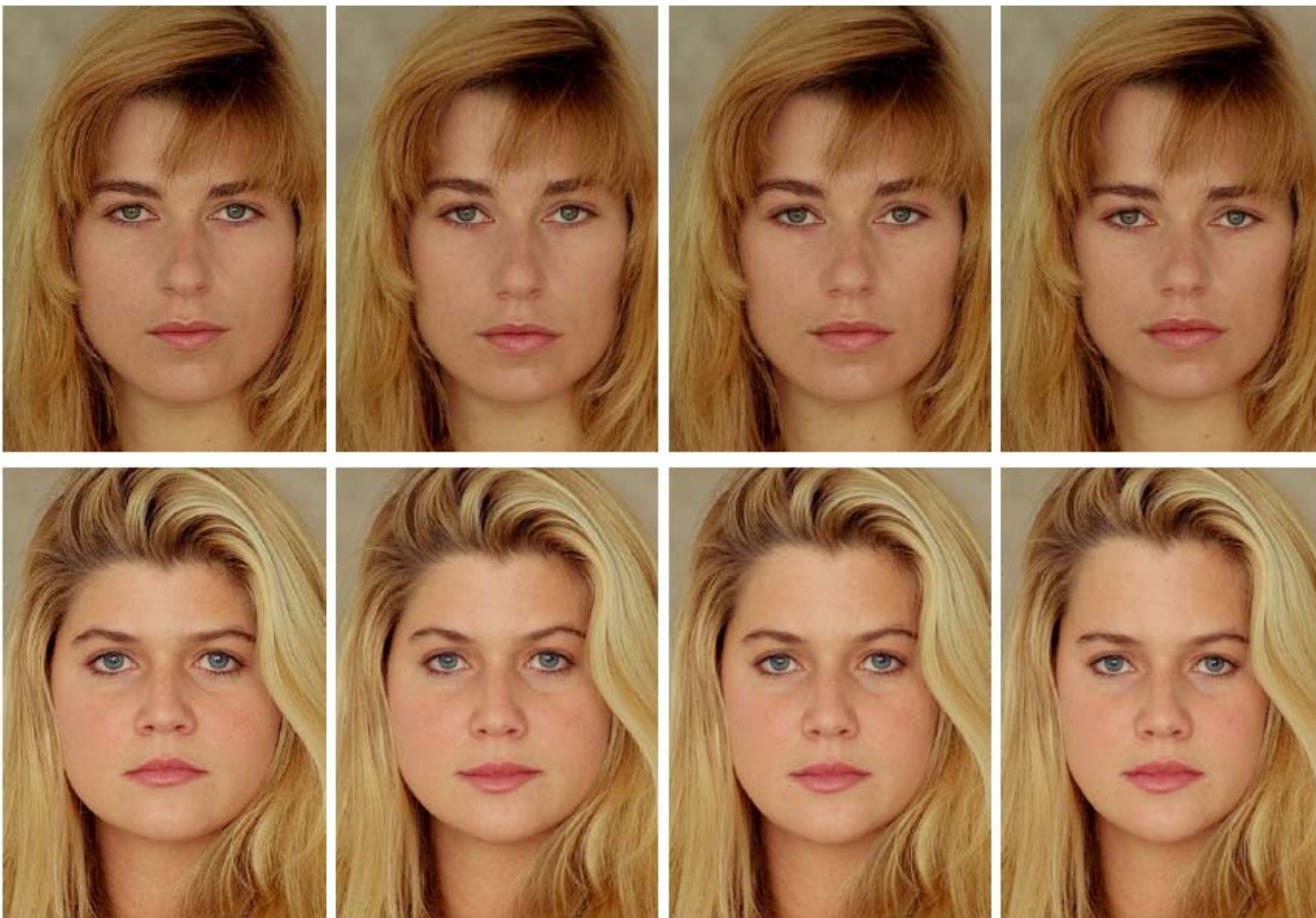
- A graph drawing problem referred to as a stress minimization problem, solved by LM algorithm for non-linear minimization

Distance embedding

- Post processing to enforce similarity transform for features on eyes by minimizing

$$\sum \|Sp_i - q_i\|^2$$

$$S = \begin{pmatrix} a & b & t_x \\ -b & a & t_y \\ 0 & 0 & 1 \end{pmatrix}$$



Original

K=3

K=5

SVR

Results (in training set)



User study

Original portrait	3.37 (0.49)
Warped to mean	3.75 (0.49)
KNN-beautified (best)	4.14 (0.51)
SVR-beautified	4.51 (0.49)

Results (not in training set)



By parts



(a)



(b)



(c)

eyes



(d)



(e) full



(f)

mouth

Different degrees



50%



100%

Facial collage



Results

- [video](#)

Image morphing

Image morphing

- The goal is to synthesize a fluid transformation from one image to another.
- Cross dissolving is a common transition between cuts, but it is not good for morphing because of the ghosting effects.



image #1



dissolving



image #2

Artifacts of cross-dissolving



<http://www.salavon.com/>

Image morphing

- Why ghosting?
- Morphing = warping + cross-dissolving

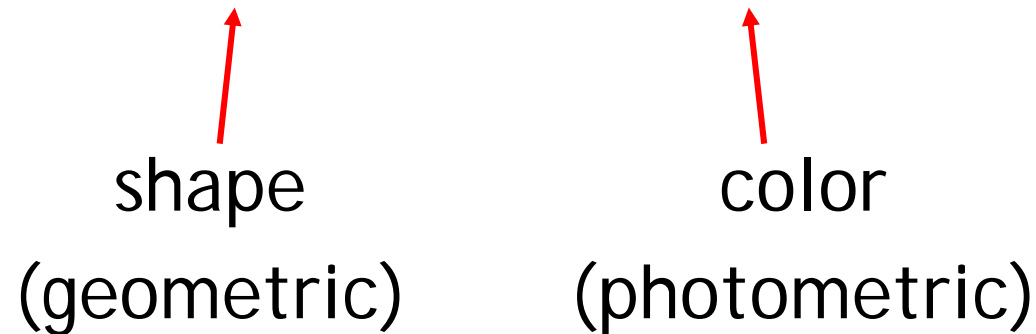
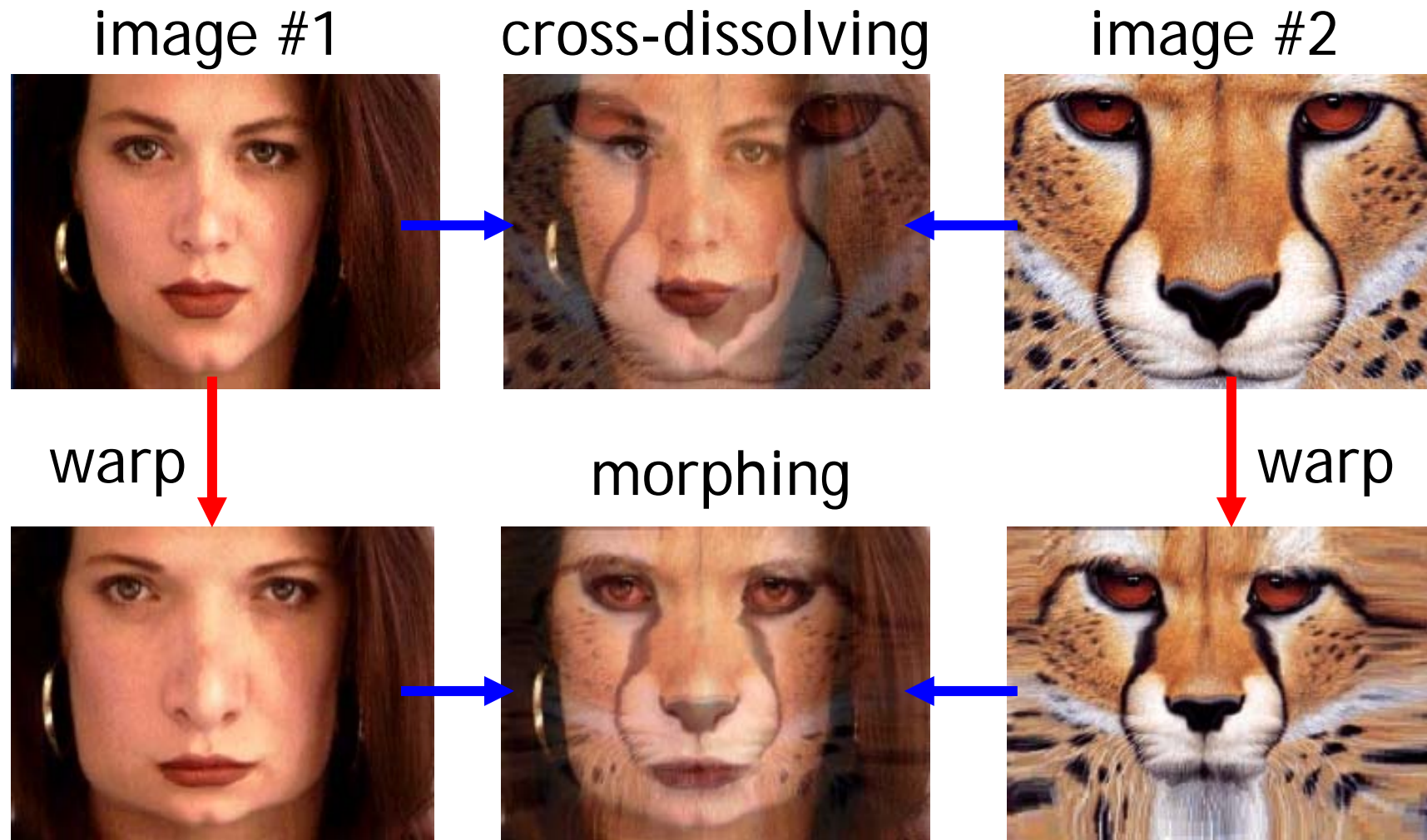


Image morphing



Morphing sequence



Face averaging by morphing



average faces

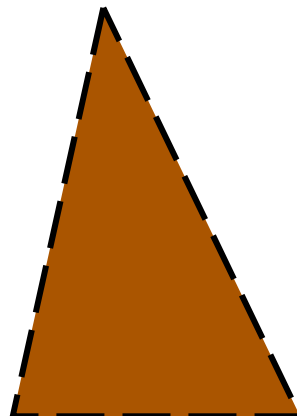
Image morphing

create a morphing sequence: for each time t

1. Create an intermediate warping field (by interpolation)
2. Warp both images towards it
3. Cross-dissolve the colors in the newly warped images



$t=0$



$t=0.33$



$t=1$

An ideal example (in 2004)



$t=0$



$t=0.75$



$t=1$

An ideal example (in 2004)



t=0



morphing



t=1

An ideal example



t=0



middle face (t=0.5)



t=1

Warp specification (mesh warping)

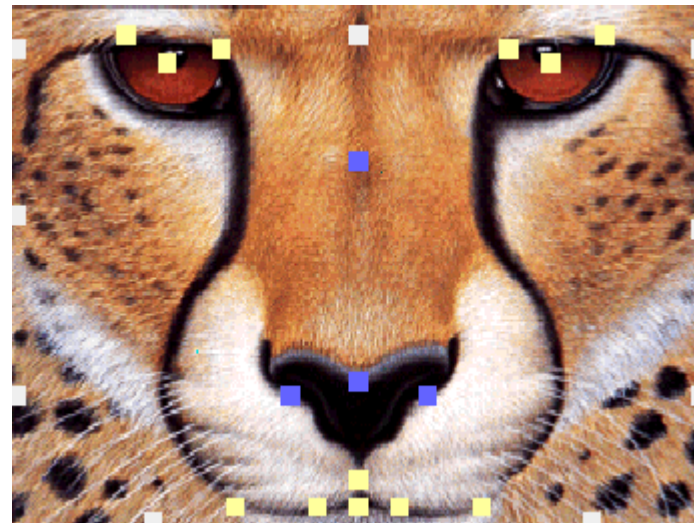
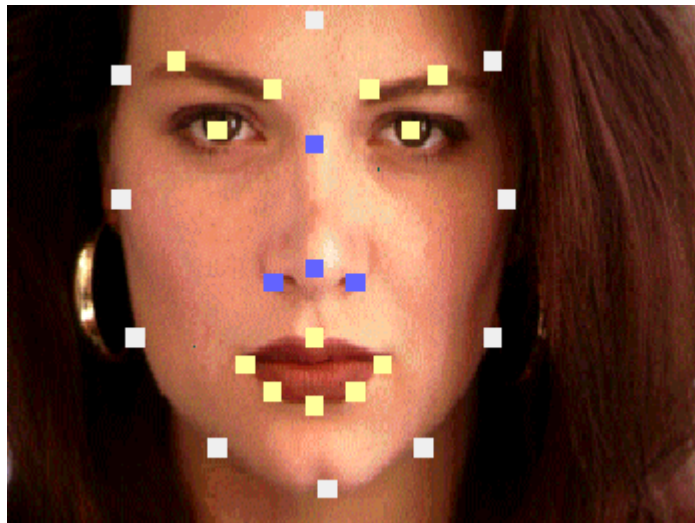
- How can we specify the warp?
 1. Specify corresponding *spline control points*
interpolate to a complete warping function



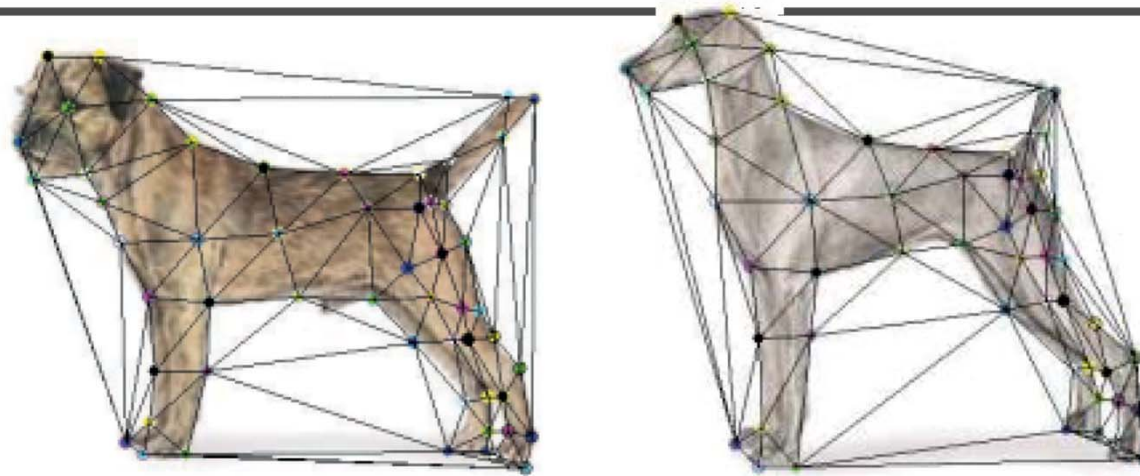
easy to implement, but less expressive

Warp specification

- How can we specify the warp
 1. Specify *landmarks*
 - *interpolate* to a complete warping function
 2. Specify corresponding *points*
 - *interpolate* to a complete warping function



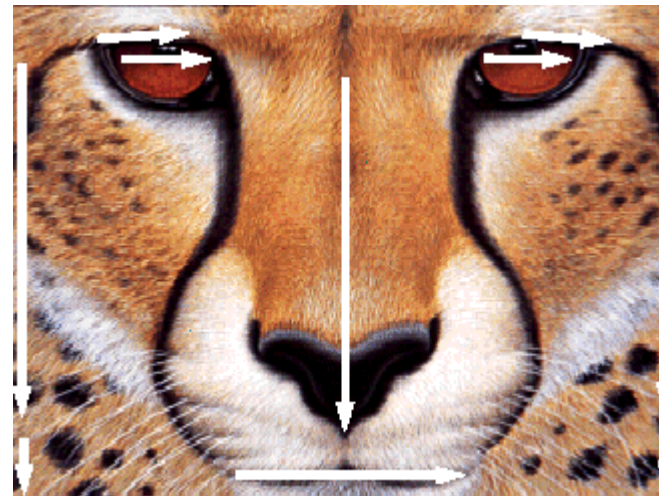
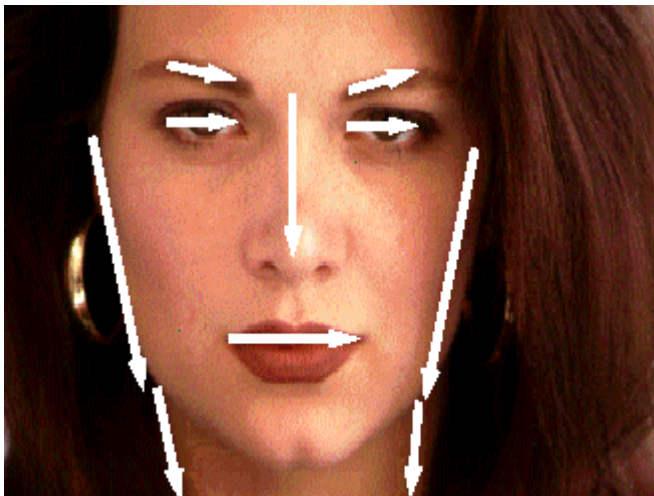
Solution: convert to mesh warping



1. Define a triangular mesh over the points
 - Same mesh in both images!
 - Now we have triangle-to-triangle correspondences
2. Warp each triangle separately from source to destination
 - How do we warp a triangle?
 - 3 points = affine warp!
 - Just like texture mapping

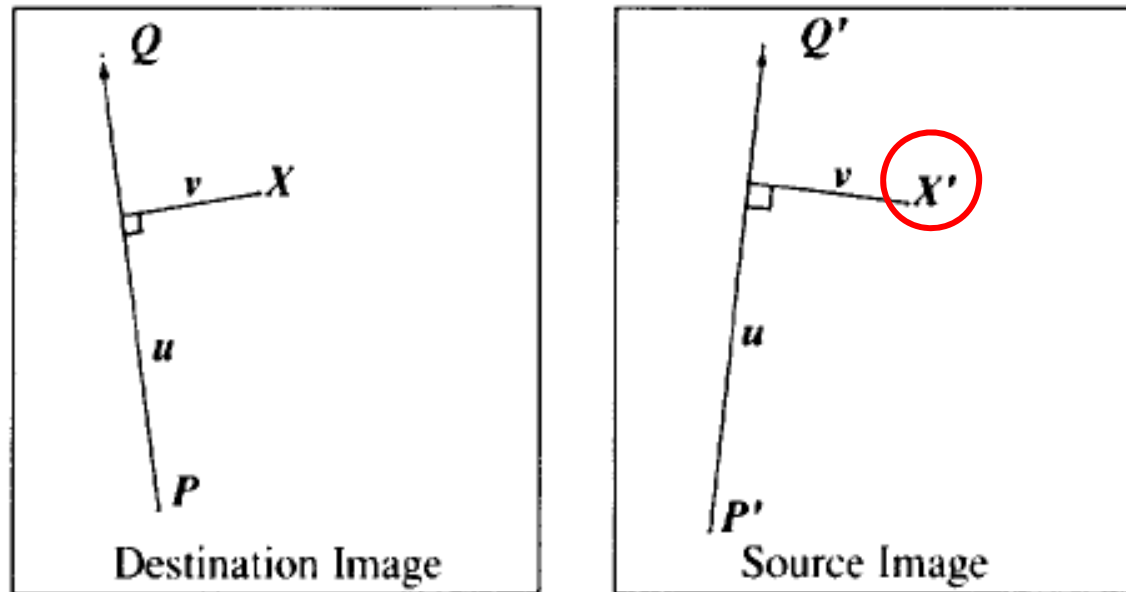
Warp specification (field warping)

- How can we specify the warp?
 3. Specify corresponding *vectors*
 - *interpolate* to a complete warping function
 - The Beier & Neely Algorithm



Beier&Neely (SIGGRAPH 1992)

- Single line-pair PQ to P'Q':



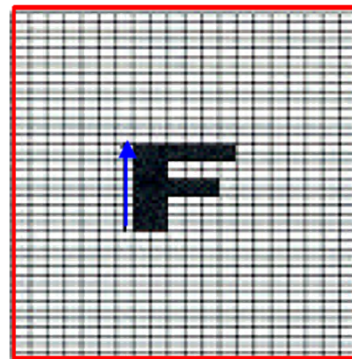
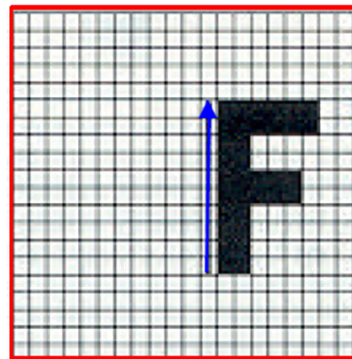
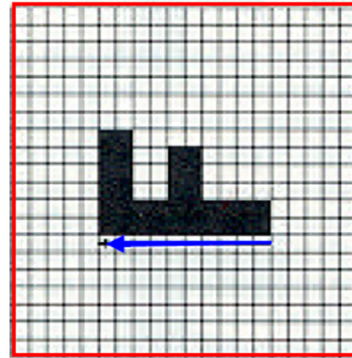
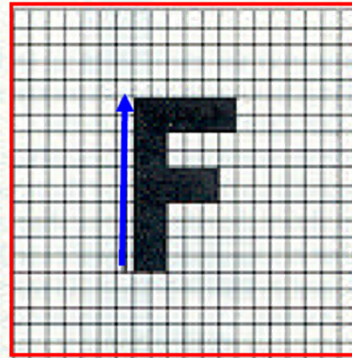
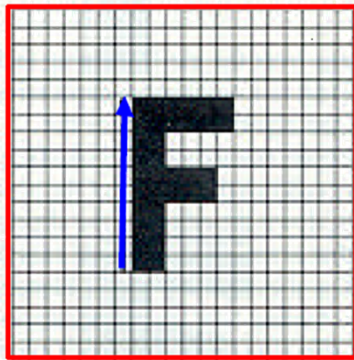
$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2} \quad (1)$$

$$v = \frac{(X - P) \cdot \text{Perpendicular}(Q - P)}{\|Q - P\|} \quad (2)$$

$$\textcircled{X'} = P' + u \cdot (Q' - P') + \frac{v \cdot \text{Perpendicular}(Q' - P')}{\|Q' - P'\|} \quad (3)$$

Algorithm (single line-pair)

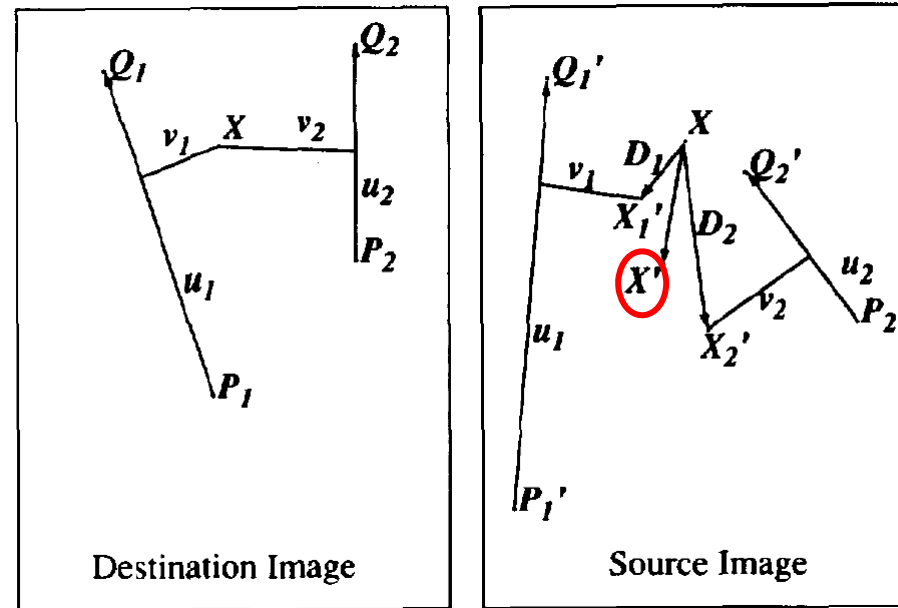
- For each X in the destination image:
 1. Find the corresponding u, v
 2. Find X' in the source image for that u, v
 3. $\text{destinationImage}(X) = \text{sourceImage}(X')$
- Examples:



Affine transformation

Multiple Lines

$$D_i = X'_i - X_i$$



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

length = length of the line segment,

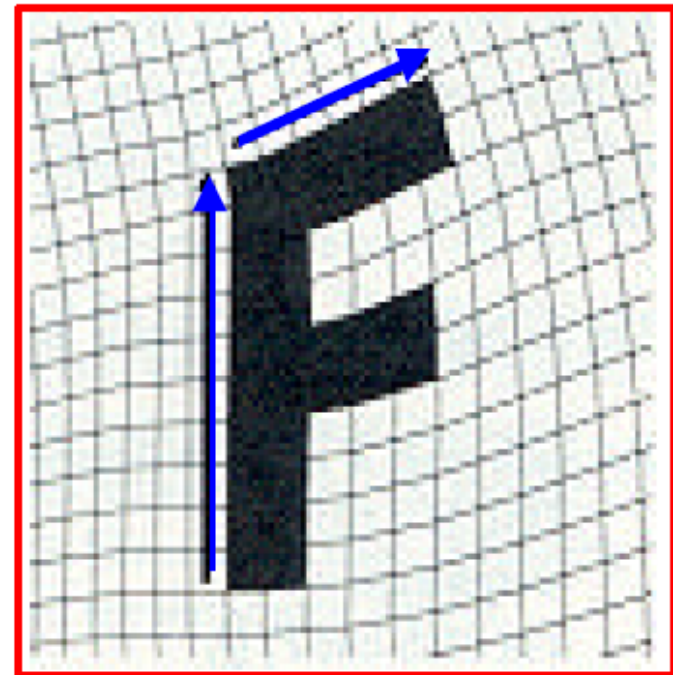
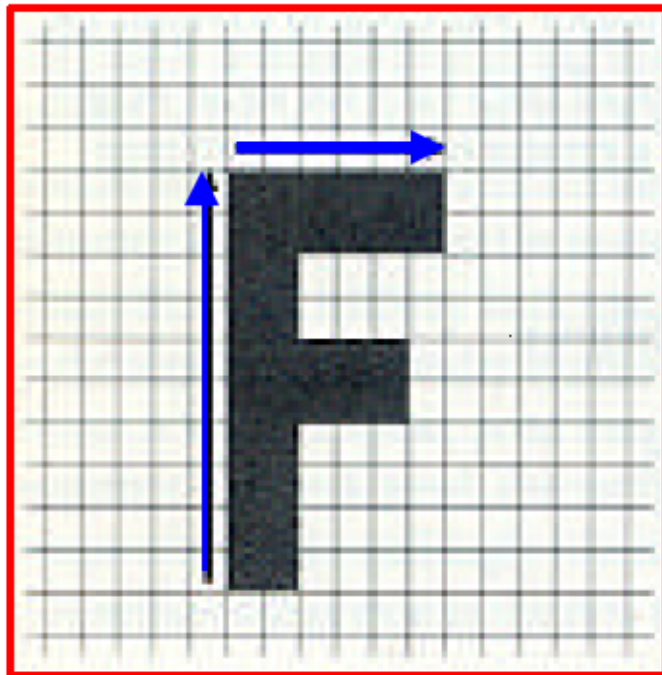
dist = distance to line segment

The influence of a , p , b . The same as the average of X'_i

Full Algorithm

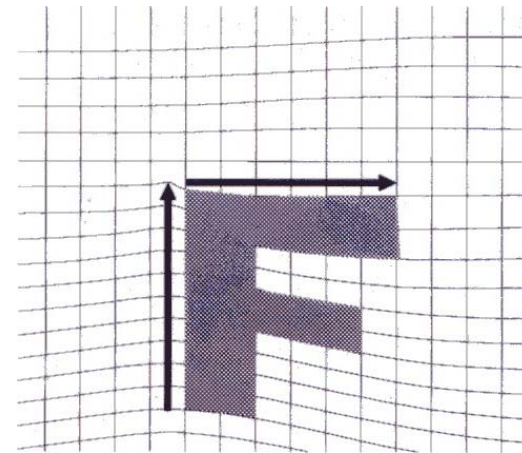
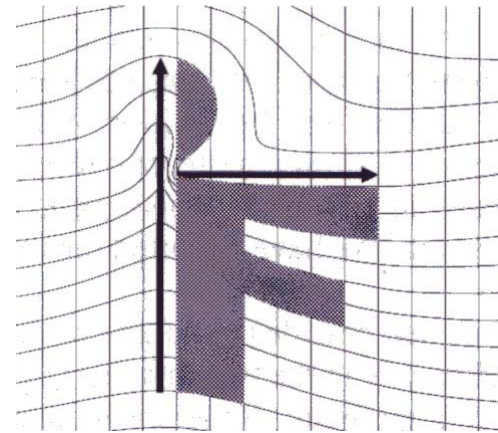
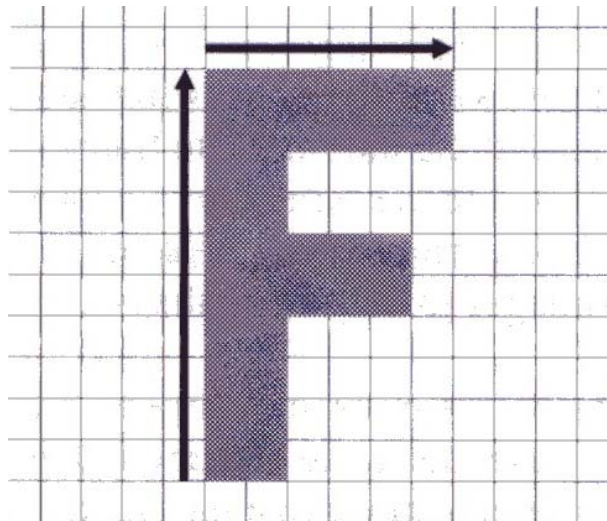
```
WarpImage(SourceImage, L'[...], L[...])
begin
    foreach destination pixel X do
        XSum = (0,0)
        WeightSum = 0
        foreach line L[i] in destination do
            X'[i] = X transformed by (L[i], L'[i])
            weight[i] = weight assigned to X'[i]
            XSum = Xsum + X'[i] * weight[i]
            WeightSum += weight[i]
        end
        X' = XSum/WeightSum
        DestinationImage(X) = SourceImage(X')
    end
    return Destination
end
```


Resulting warp



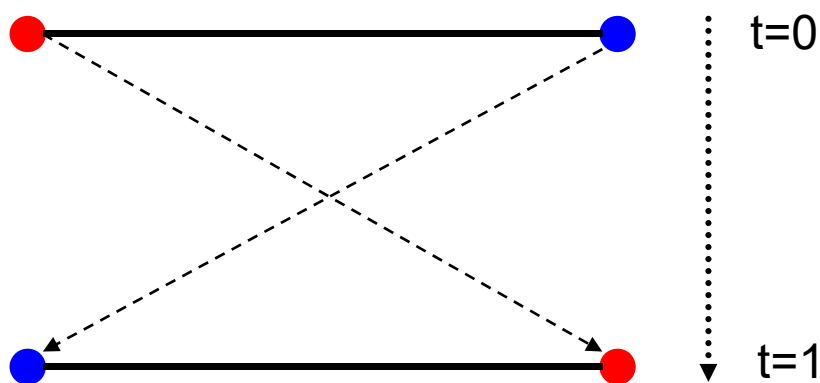
Comparison to mesh morphing

- Pros: more expressive
- Cons: speed and control



Warp interpolation

- How do we create an intermediate warp at time t ?
 - linear interpolation for line end-points
 - But, a line rotating 180 degrees will become 0 length in the middle
 - One solution is to interpolate line mid-point and orientation angle



Animation

GenerateAnimation(Image₀, L₀[...], Image₁, L₁[...])

begin

foreach intermediate frame time t **do**

for $i=1$ to number of line-pairs **do**

$L[i] = \text{line } t\text{-th of the way from } L_0[i] \text{ to } L_1[i].$

end

$\text{Warp}_0 = \text{WarpImage}(\text{Image}_0, L_0[...], L[...])$

$\text{Warp}_1 = \text{WarpImage}(\text{Image}_1, L_1[...], L[...])$

foreach pixel p in FinalImage **do**

$\text{FinalImage}(p) = (1-t) \text{Warp}_0(p) + t \text{Warp}_1(p)$

end

end

end

Animated sequences

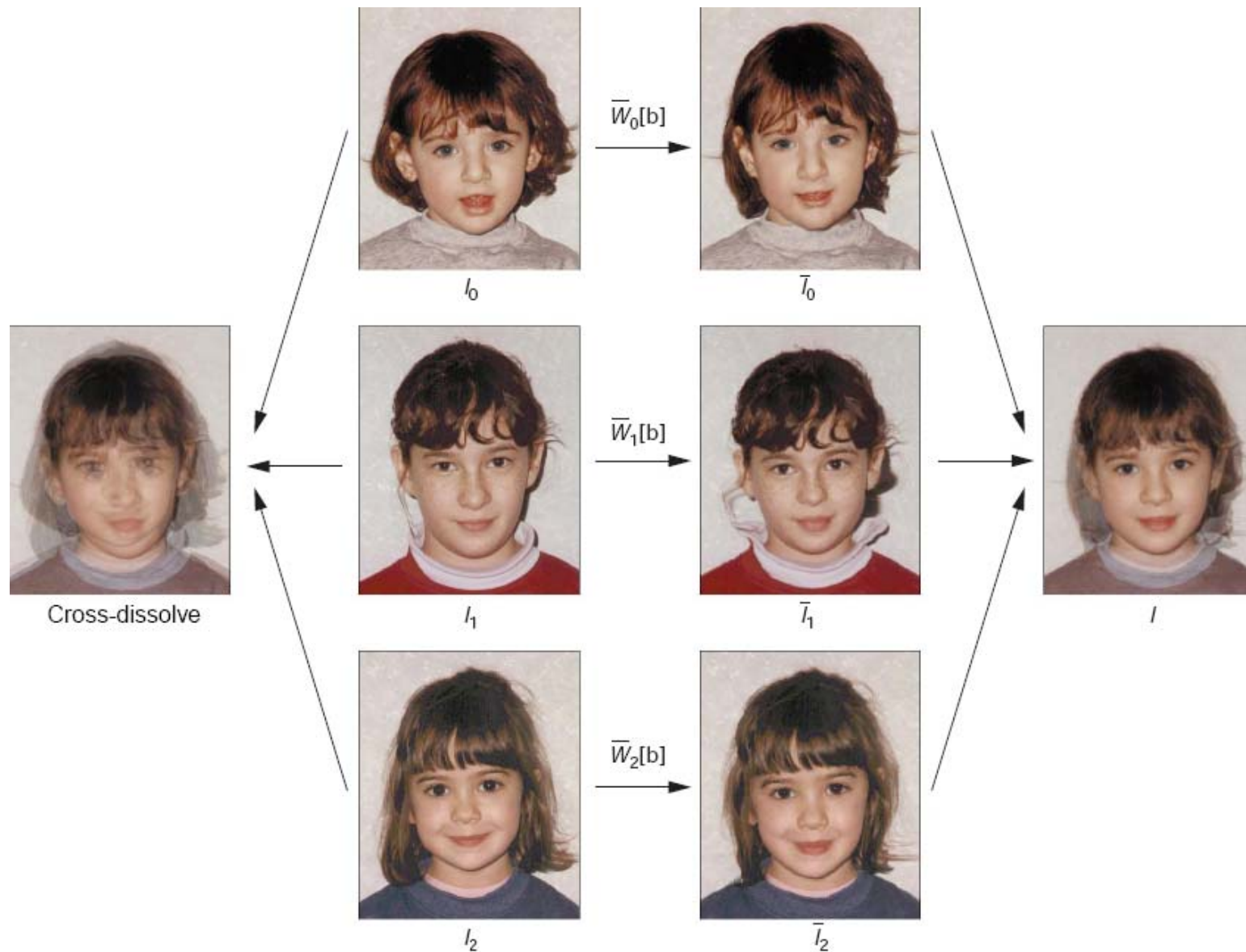
- Specify keyframes and interpolate the lines for the inbetween frames
- Require a lot of tweaking

Results



Michael Jackson's MTV "Black or White"

Multi-source morphing



Multi-source morphing



(a)



(b)



(c)



(d)



(e)



(f)



(g)



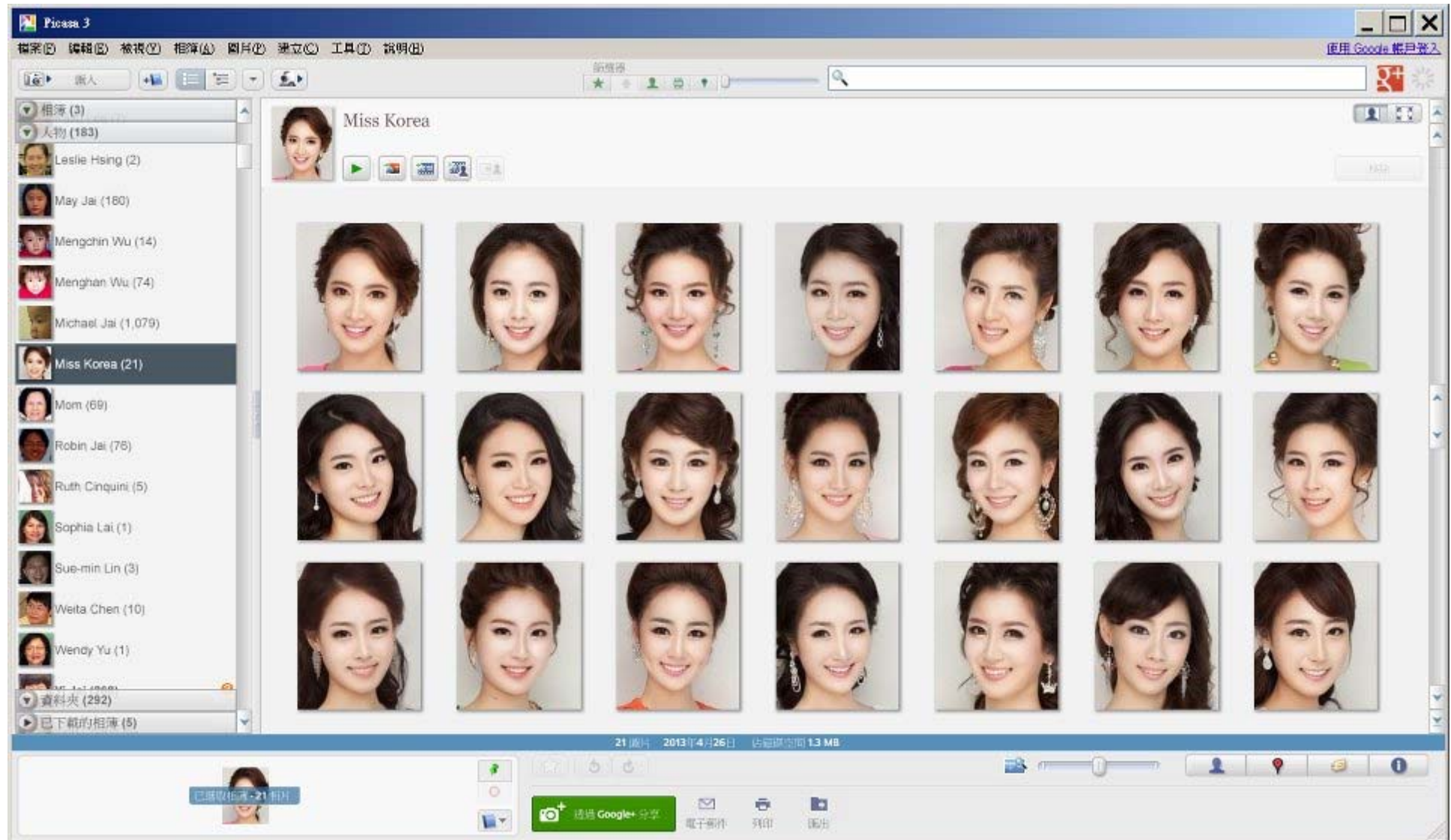
(h)

Miss Korea



Picasa recognizes them as the same person

DigiVFX



Align and mean face



Morphing sequence

- [video](#)

Woman in arts



References

- Thaddeus Beier, Shawn Neely, [Feature-Based Image Metamorphosis](#), SIGGRAPH 1992, pp35-42.
- Detlef Ruprecht, Heinrich Muller, [Image Warping with Scattered Data Interpolation](#), IEEE Computer Graphics and Applications, March 1995, pp37-43.
- Seung-Yong Lee, Kyung-Yong Chwa, Sung Yong Shin, [Image Metamorphosis Using Snakes and Free-Form Deformations](#), SIGGRAPH 1995.
- Seungyong Lee, Wolberg, G., Sung Yong Shin, [Polymorph: morphing among multiple images](#), IEEE Computer Graphics and Applications, Vol. 18, No. 1, 1998, pp58-71.
- Peinsheng Gao, Thomas Sederberg, [A work minimization approach to image morphing](#), The Visual Computer, 1998, pp390-400.
- George Wolberg, [Image morphing: a survey](#), The Visual Computer, 1998, pp360-372.
- [Data-Driven Enhancement of Facial Attractiveness](#), SIGGRAPH 2008