



>>> IMAGE PROCESSING AND COMPUTATIONAL PHOTOGRAPHY

SESSION 9: WARPING AND MORPHING

Oriol Pujol & Simone Balocco

Outline

- Image warping and morphing
 - Global coordinate transformations
 - Meshes and triangulation
 - Texture mapping
 - Interpolation
- Applications
 - Morphing and transitions

Image Transformations

image filtering: change **range** of image

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

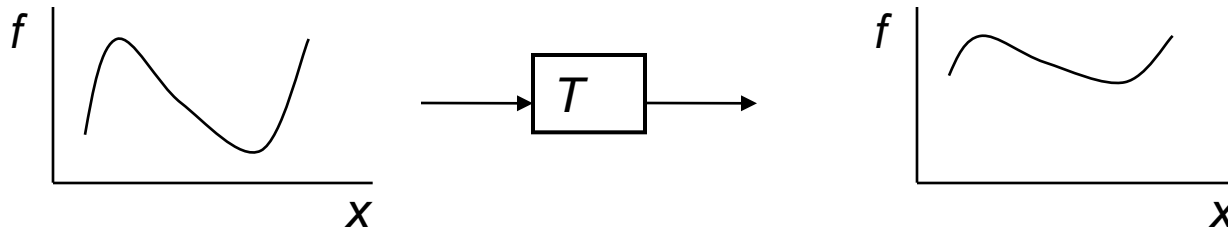


image warping: change **domain** of image

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

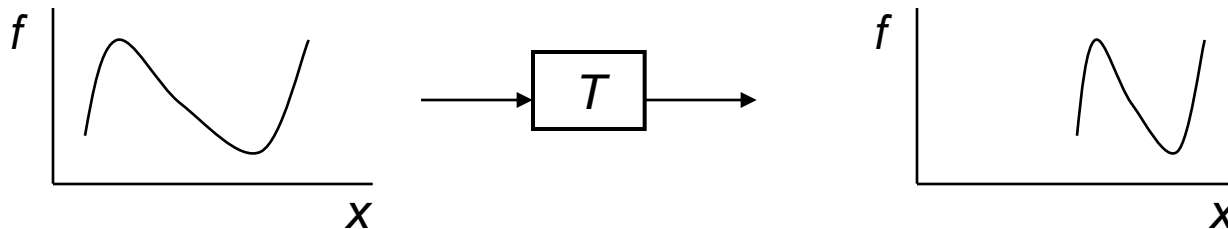


Image Transformations

image filtering: change **range** of image

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

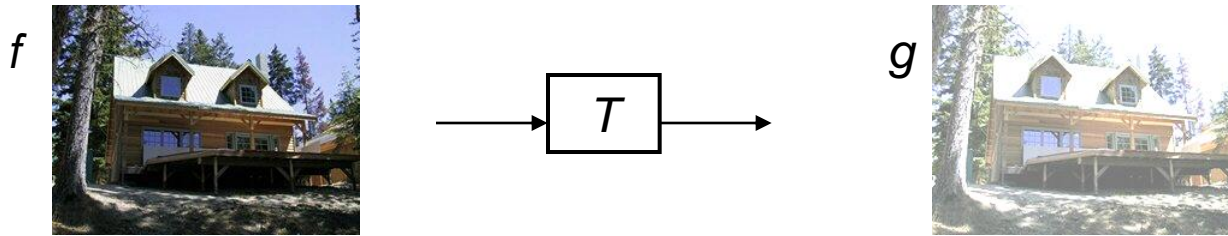
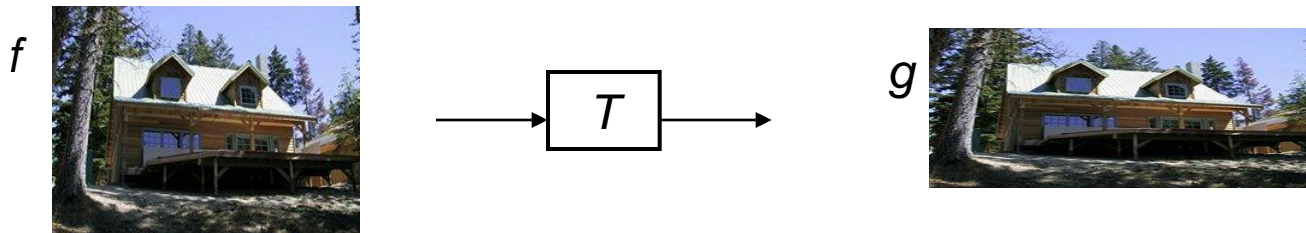


image warping: change **domain** of image

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



Parametric (global) warping

Examples of parametric warps:



translation



rotation



aspect



affine



perspective

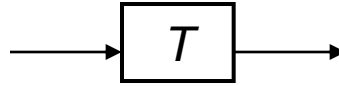


cylindrical

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)

Let's represent T as a matrix:

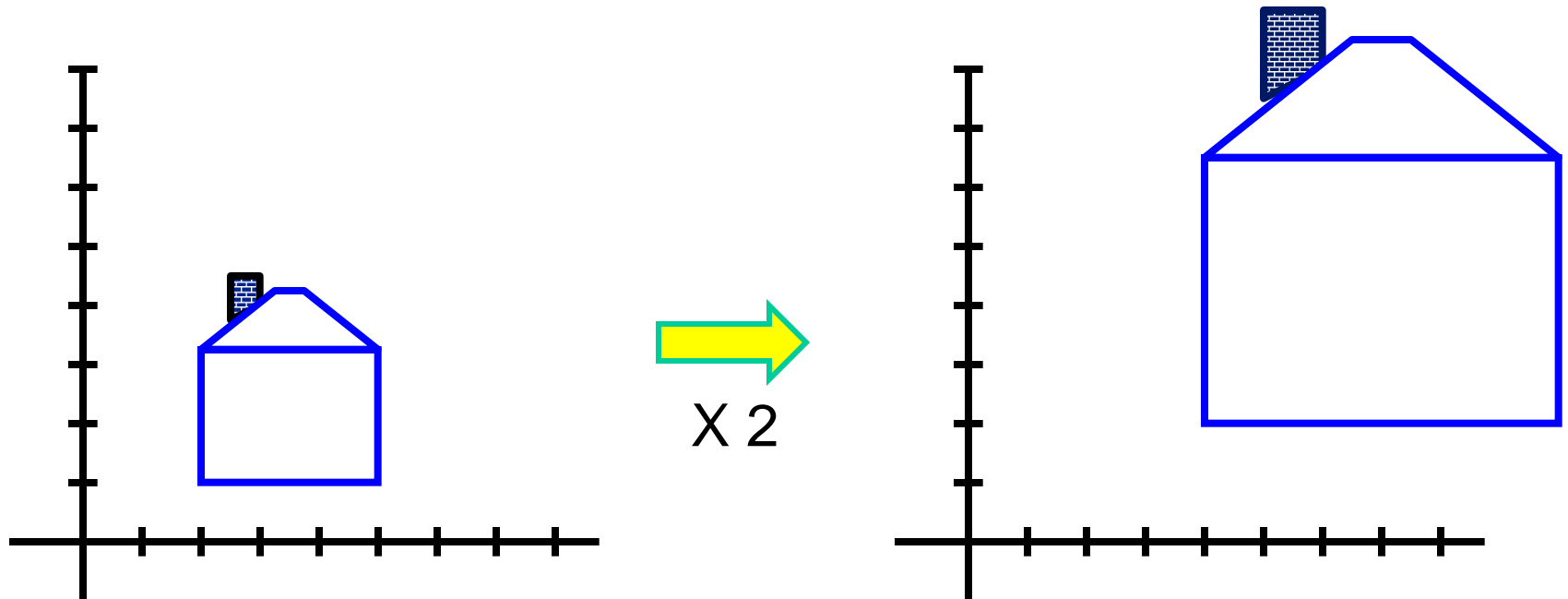
$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} x \\ y \end{pmatrix}$$

Scaling

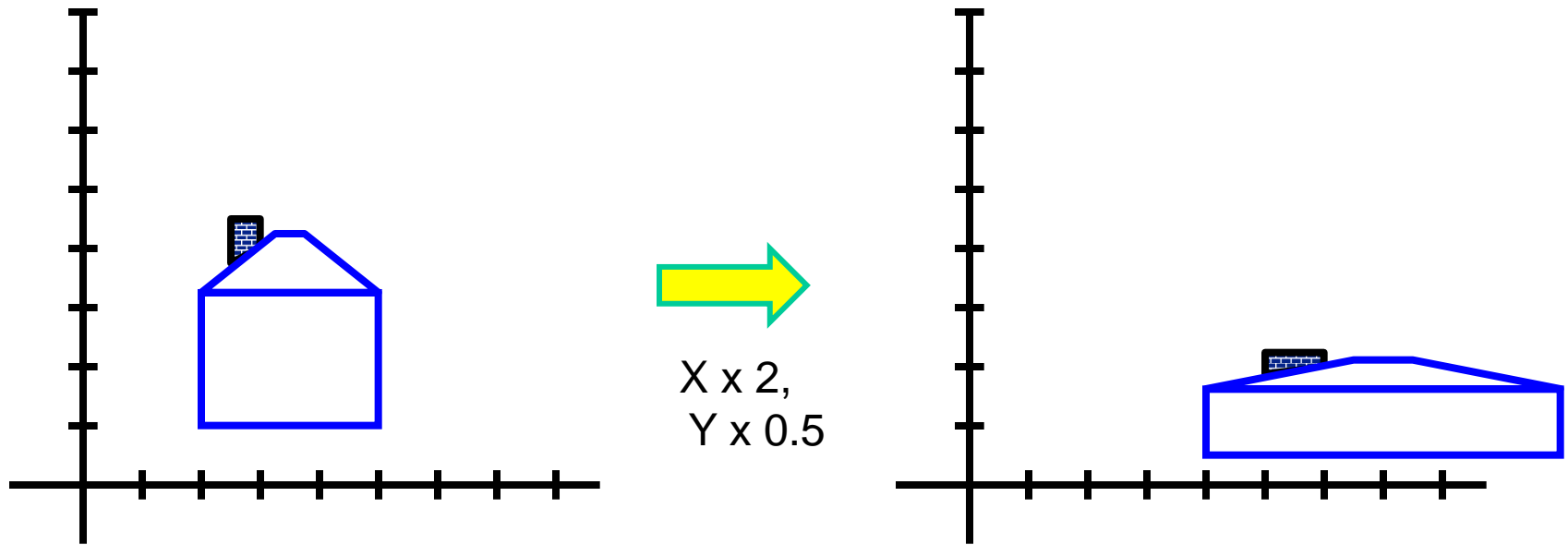
Scaling a coordinate means multiplying each of its components by a scalar

Uniform scaling means this scalar is the same for all components:



Scaling

Non-uniform scaling: different scalars per component:



Scaling

Scaling operation:

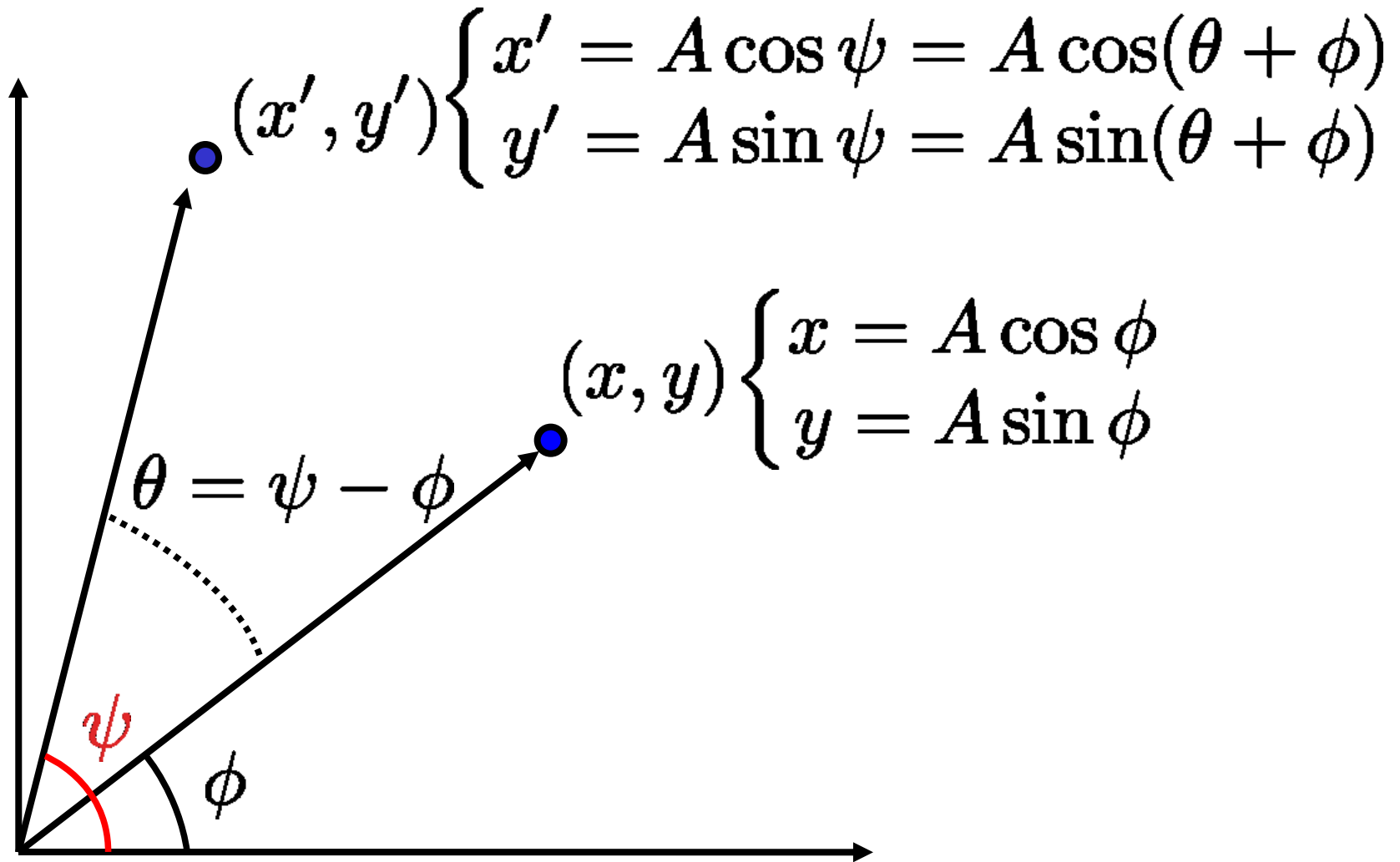
$$\begin{aligned}x' &= ax \\ y' &= by\end{aligned}$$

Or, in matrix form:

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

What's inverse of S?

2-D Rotation



$$\begin{cases} x = A \cos \phi \\ y = A \sin \phi \end{cases} \quad \begin{cases} x' = A \cos \psi = A \cos(\theta + \phi) \\ y' = A \sin \psi = A \sin(\theta + \phi) \end{cases}$$

$$\begin{cases} \cos(a \pm b) = \cos a \cos b \mp \sin a \sin b \\ \sin(a \pm b) = \sin a \cos b \pm \cos a \sin b \end{cases}$$

$$\begin{cases} x' = A \cos(\theta + \phi) = A \cos \theta \cos \phi - A \sin \theta \sin \phi \\ y' = A \sin(\theta + \phi) = A \sin \theta \cos \phi + A \cos \theta \sin \phi \end{cases}$$

$$\begin{cases} x' = A \cos(\theta + \phi) = x \cos \theta - y \sin \theta \\ y' = A \sin(\theta + \phi) = x \sin \theta + y \cos \theta \end{cases}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2-D Rotation

This is easy to capture in matrix form:

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

Even though sin and cos are nonlinear functions of 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 $\mathbf{R}^{-1} = \mathbf{R}^T$

2x2 Matrices

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

2D Identity?

$$x' = x$$

$$y' = y$$

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

2D Scale around (0,0)?

$$x' = s_x x$$

$$y' = s_y y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2x2 Matrices

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

2D Rotate around (0,0)?

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2D Shear?

$$\begin{aligned} x' &= x + h_x y \\ y' &= h_y x + y \end{aligned} \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & h_x \\ h_y & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

2x2 Matrices

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

2D Mirror about Y axis?

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

2D Mirror over (0,0)?

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

2x2 Matrices

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

2D Translation?

$$\begin{aligned}x' &= x + t_x \\ y' &= y + t_y\end{aligned}\quad \text{NO!}$$

Only linear 2D transformations
can be represented with a 2x2 matrix.
Translations are affine transformations!

All 2D Linear Transformations

Linear transformations are combinations of ...

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

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

Properties of linear transformations:

- Origin maps to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} \begin{pmatrix} i & j \\ k & l \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

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:

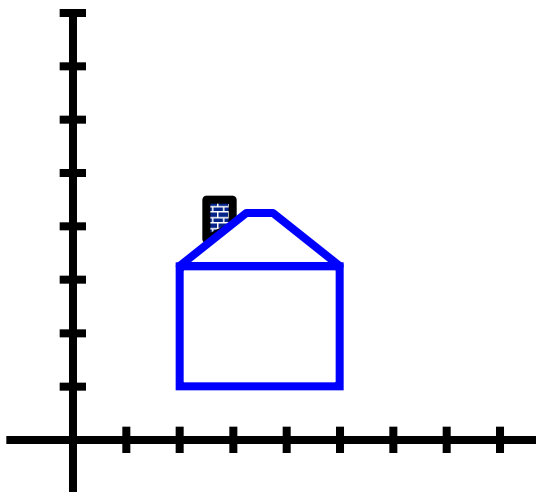
$$\mathbf{T} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$


Translation

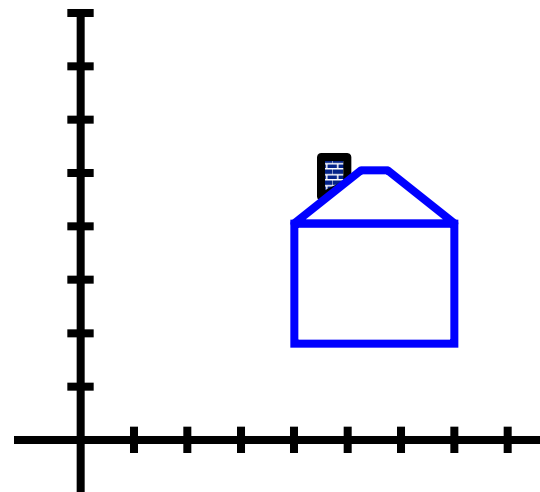
Example of translation

Homogeneous Coordinates

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




 $t_x = 2$
 $t_y = 1$



Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Translate

Scale

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & \beta_x & 0 \\ \beta_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Rotate

Shear

Matrix Composition

Transformations can be combined by matrix multiplication

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

$$p' = \mathbf{T}(t_x, t_y) \quad \mathbf{R}(\theta) \quad \mathbf{S}(s_x, s_y) \quad p$$

Does the order of the multiplication matter?

Imagine we want to rotate an object located at (a,b) over its center

Matrix Composition

Transformations can be combined by matrix multiplication

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -a \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

3rd
Translate back to (a,b)

2nd
Rotate

1st
Translate to the origin

Imagine we want to rotate an object located at (a,b) over its center

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

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

Will the last coordinate w always be 1?

Projective Transformations

Projective transformations ...

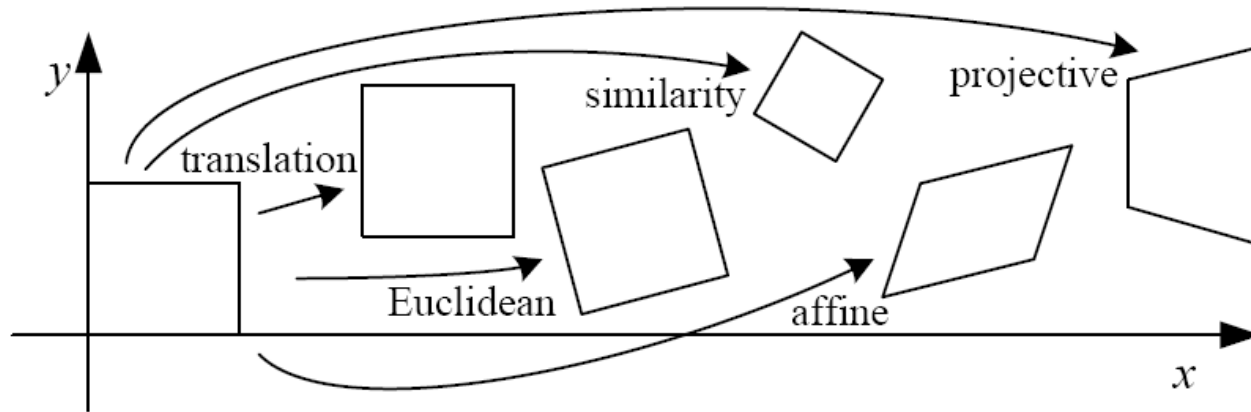
- Affine transformations, and
- Projective warps

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

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

2D image transformations

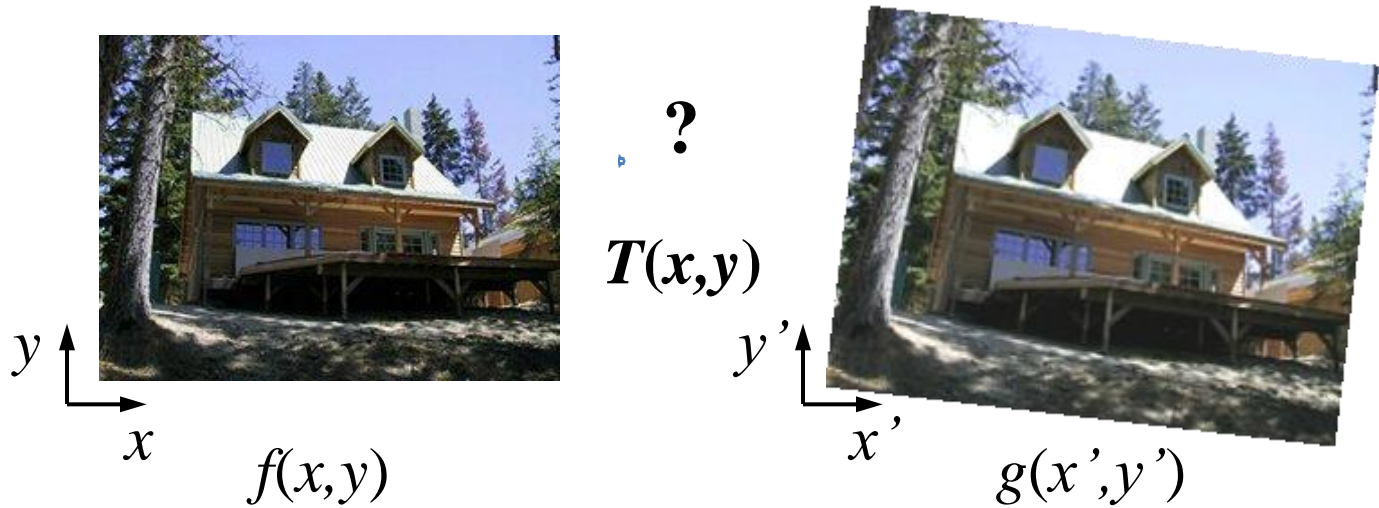


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$			
rigid (Euclidean)	$\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$			
similarity	$\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$			
affine	$\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$			
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$			

These transformations are a nested set of groups

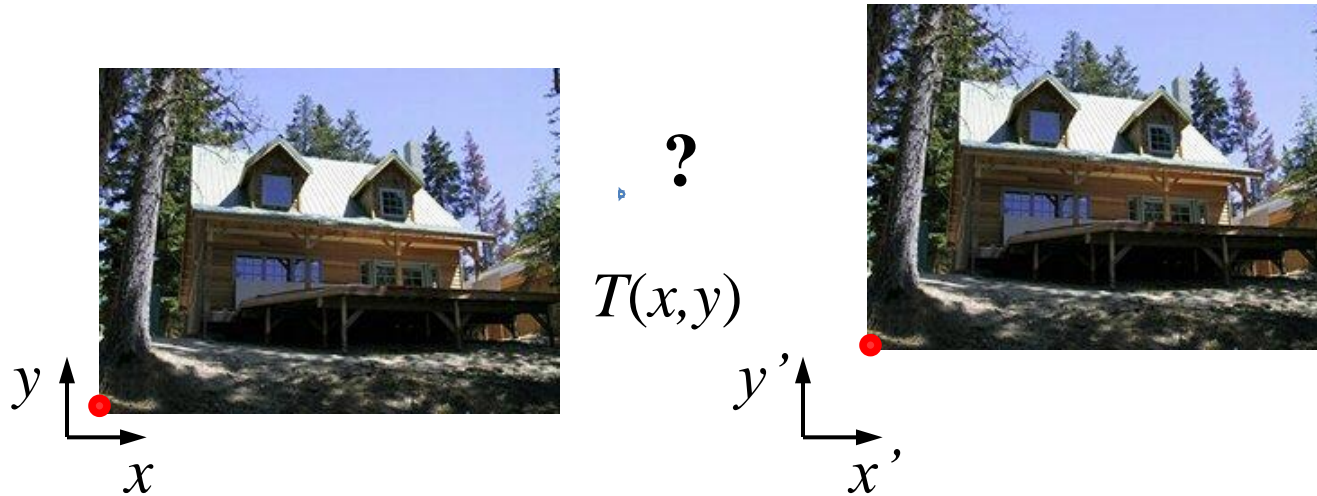
- Closed under composition (their composition is a member of the group) and its inverse is a member

Recovering Transformations



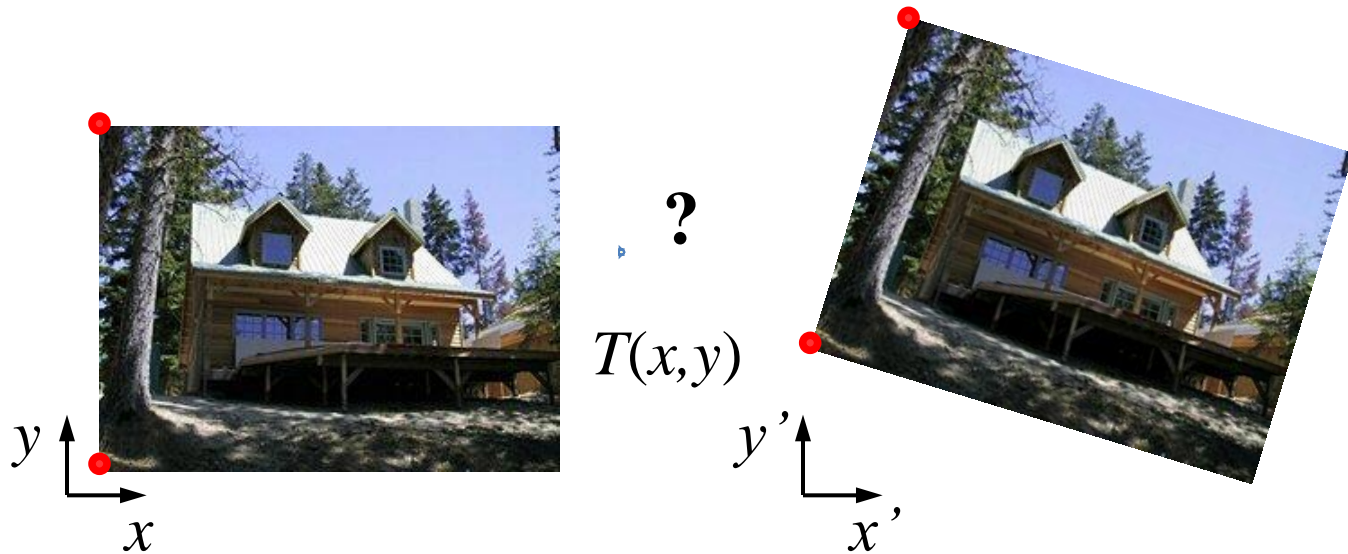
- What if we know f and g and want to recover the transform T ?
 - willing to let user provide correspondences
 - How many do we need?

Translation: # correspondences?



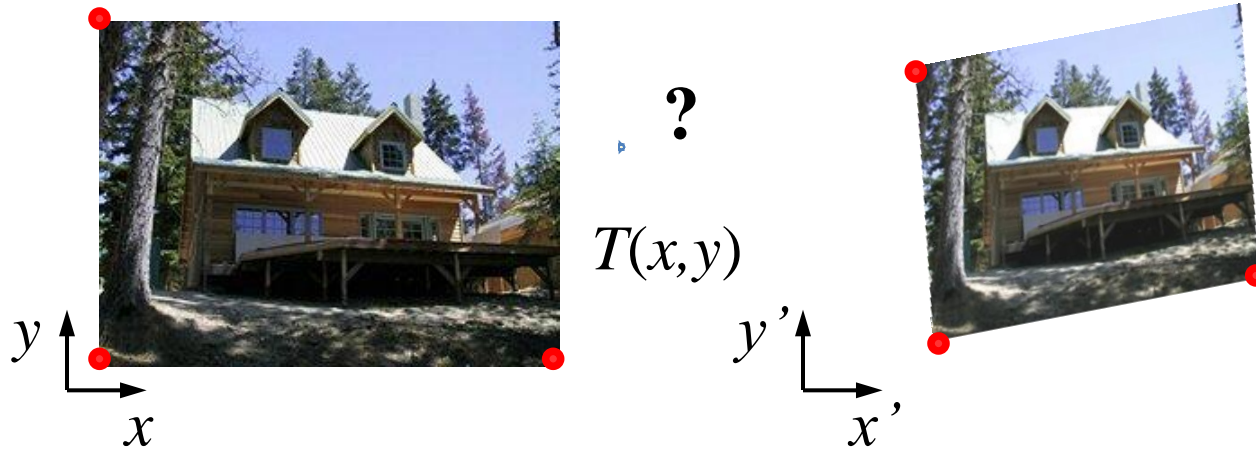
- How many Degrees of Freedom?
- How many correspondences needed for translation?
- What is the transformation matrix?

Euclidian: # correspondences?



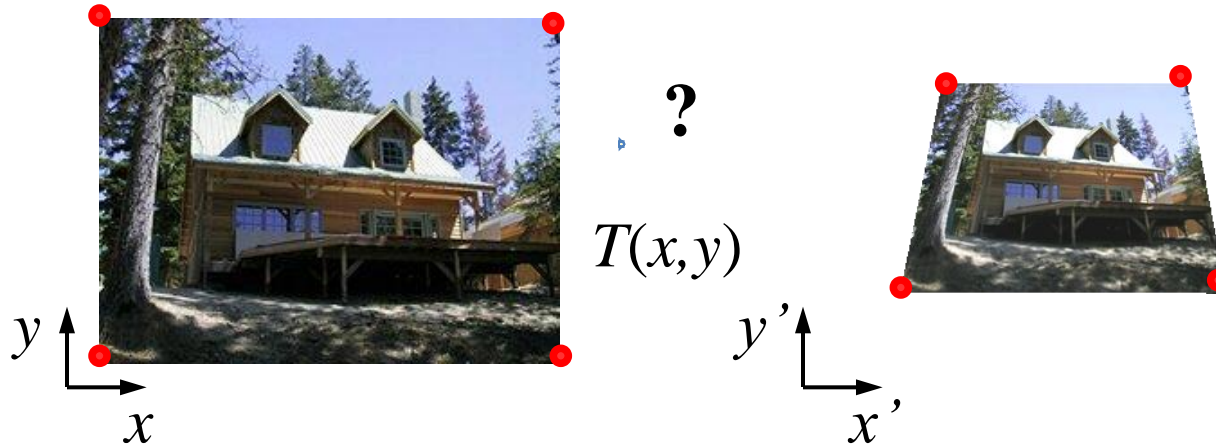
- How many DOF?
- How many correspondences needed for translation+rotation?

Affine: # correspondences?



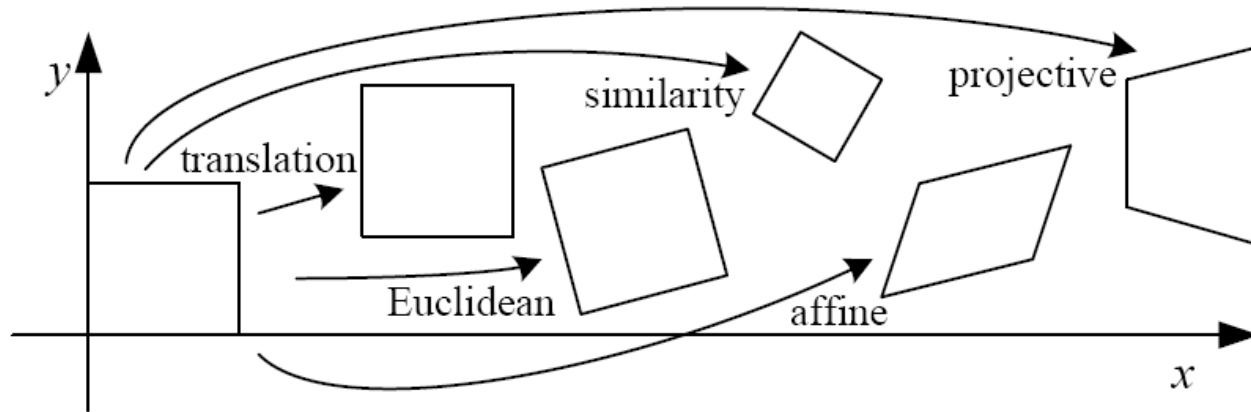
- How many DOF?
- How many correspondences needed for affine?

Projective: # correspondences?



- How many DOF?
- How many correspondences needed for projective?

2D image transformations

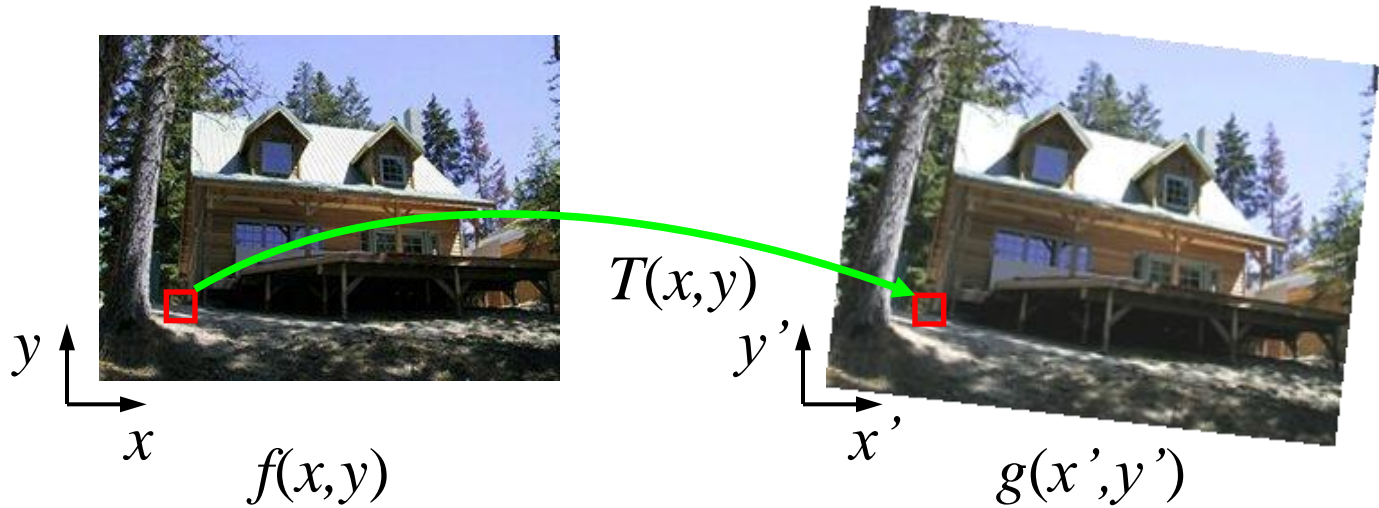


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles + ...	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

These transformations are a nested set of groups

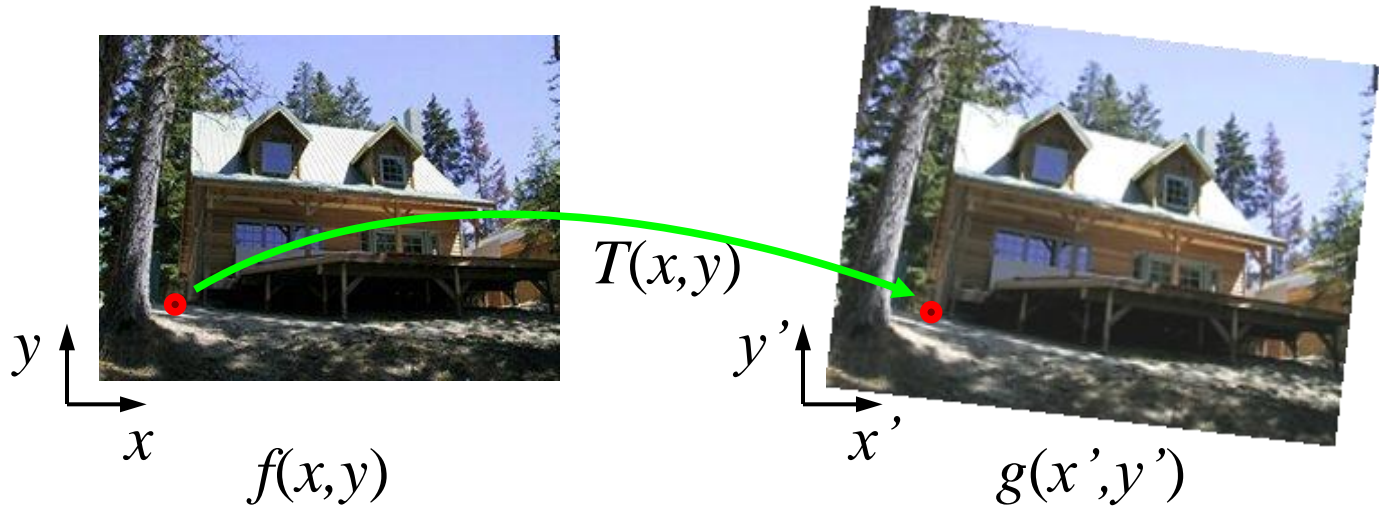
- Closed under composition and inverse is a member

Image warping



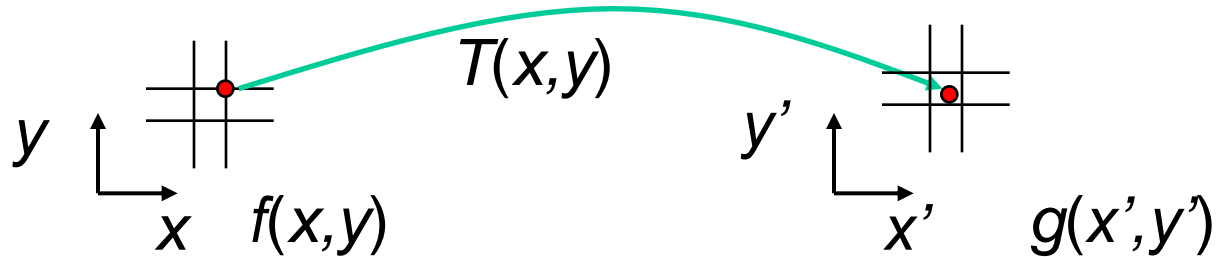
Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

Forward warping



Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

Forward warping



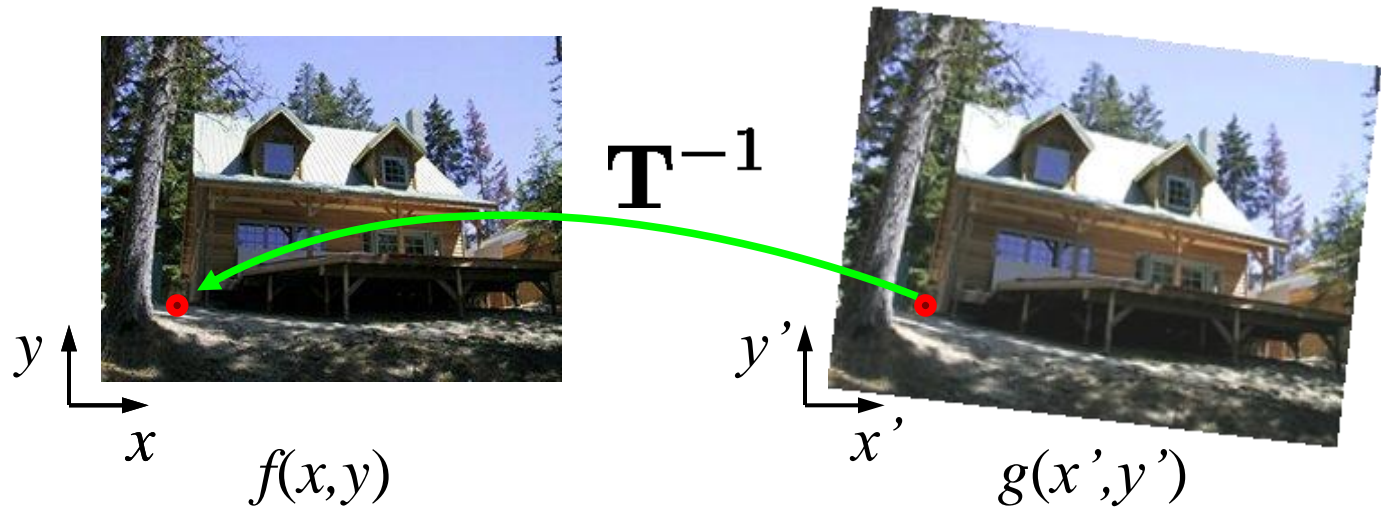
Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands “between” two pixels?

A: distribute color among neighboring pixels (x',y')

- Known as “splatting”
- Check out `griddata` in Matlab

Inverse warping

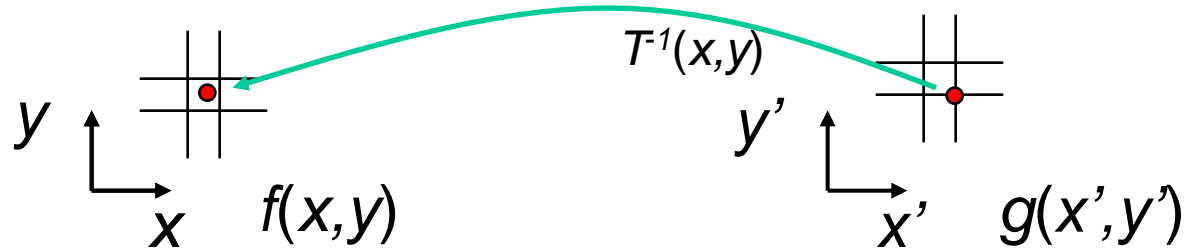


Get each pixel $g(x',y')$ from its corresponding location

$(x, y) = \mathbf{T}^{-1}(x', y')$ in the first image

Q: what if pixel comes from “between” two pixels?

Inverse warping



Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

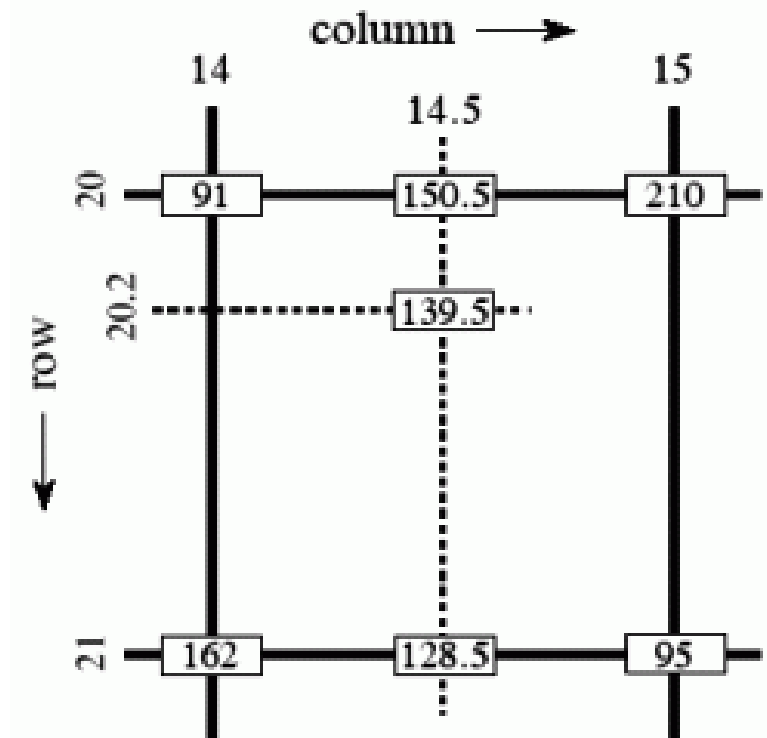
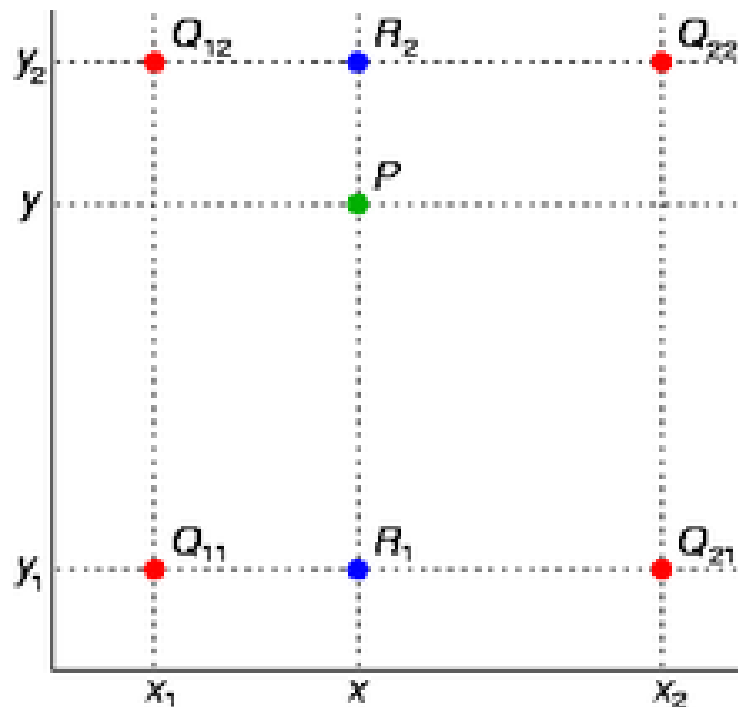
Q: what if pixel comes from “between” two pixels?

A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear, Gaussian, bicubic
- Check out `interp2` in Matlab

Bilinear Interpolation

$$f(x, y) \approx \begin{bmatrix} 1 - x & x \end{bmatrix} \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix}.$$



Forward vs. inverse warping

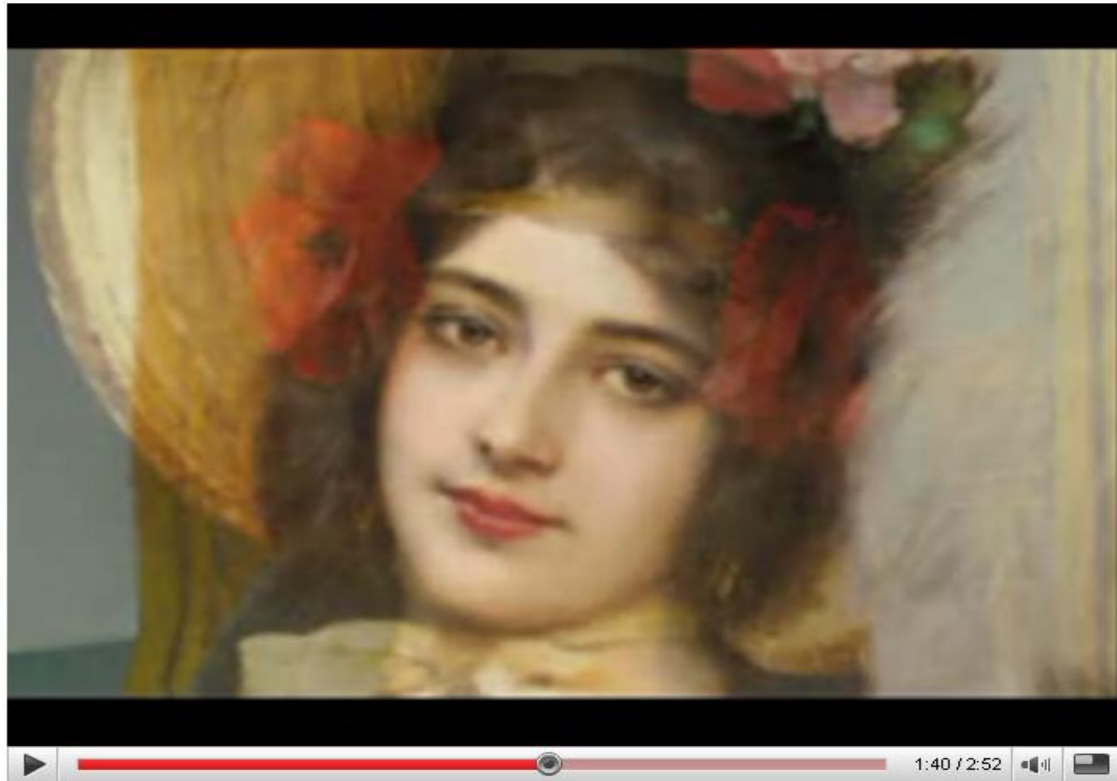
Q: which is better?

A: Usually inverse—eliminates holes

- however, it requires an invertible warp function

Image morphing

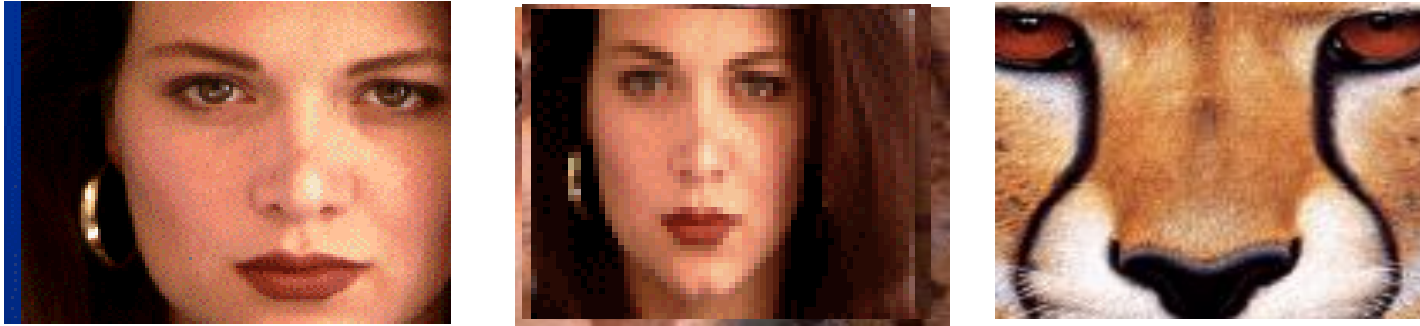
Women in art



[watch in high quality](#)

http://youtube.com/watch?v=nUDIoN-_Hxs

Morphing = Object Averaging



The aim is to find “an average” between two objects

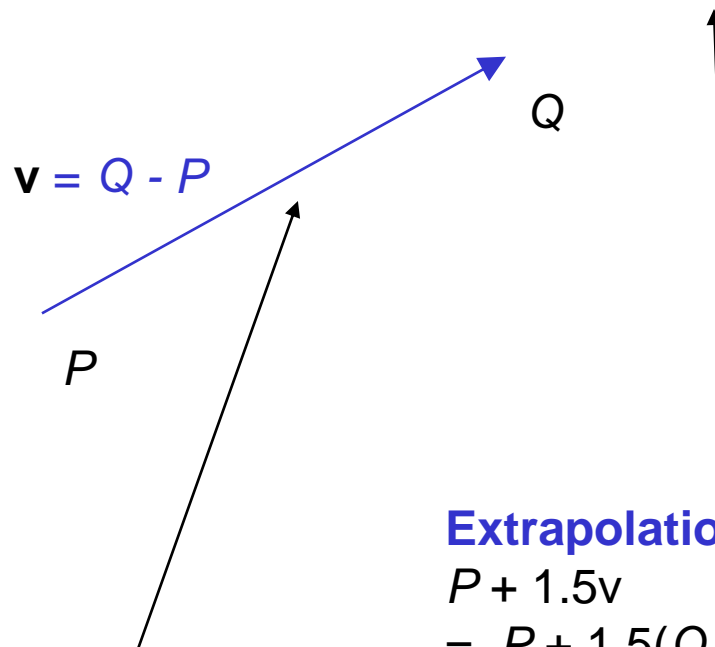
- Not an average of two images of objects...
- ...but an image of the average object!
- How can we make a smooth transition in time?
 - Do a “weighted average” over time t

How do we know what the average object looks like?

- We haven't a clue!
- But we can often fake something reasonable
 - Usually required user/artist input

Averaging Points

What's the average
of P and Q?



$$\begin{aligned} P + 0.5\mathbf{v} \\ &= P + 0.5(\mathbf{Q} - \mathbf{P}) \\ &= 0.5P + 0.5Q \end{aligned}$$

Extrapolation: $t < 0$ or $t > 1$

$$\begin{aligned} P + 1.5\mathbf{v} \\ &= P + 1.5(\mathbf{Q} - \mathbf{P}) \\ &= -0.5P + 1.5Q \quad (t=1.5) \end{aligned}$$

Linear Interpolation

New point: $(1-t)P + tQ$
 $0 < t < 1$

P and Q can be anything:

- points on a plane (2D) or in space (3D)
- Colors in RGB (3D)
- Whole images (m-by-n D)... etc.

Idea #1: Cross-Dissolve



Interpolate whole images:

$$\text{Imagehalfway} = (1-t) * \text{Image1} + t * \text{image2}$$

This is called **cross-dissolve** in film industry

But what if the images are not aligned?

Idea #2: Align, then cross-dissolve



Align first, then cross-dissolve

- Alignment using global warp – picture still valid

Dog Averaging



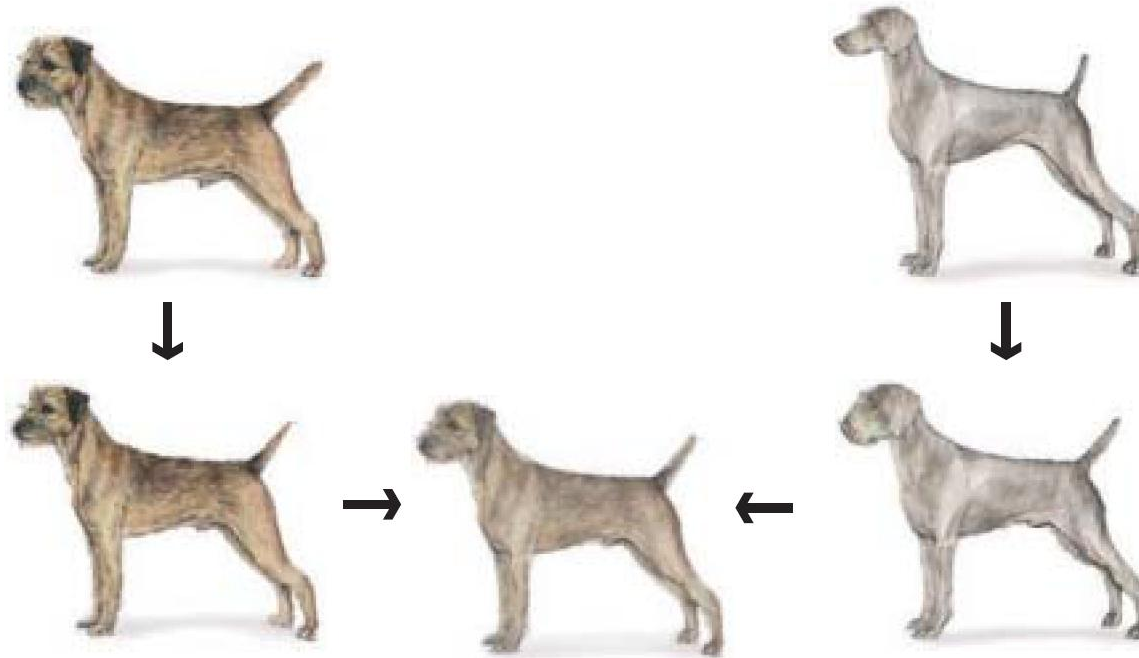
What to do?

- Cross-dissolve doesn't work
- Global alignment doesn't work
 - Cannot be done with a global transformation (e.g. affine)
- Any ideas?

Feature matching!

- Nose to nose, tail to tail, etc.
- This is a local (non-parametric) warp

Idea #3: Local warp, then cross-dissolve



Morphing procedure

For every frame t ,

1. Find the average shape (the “mean dog” 😊)
 - local warping
2. Find the average color
 - Cross-dissolve the warped images

Local (non-parametric) Image Warping



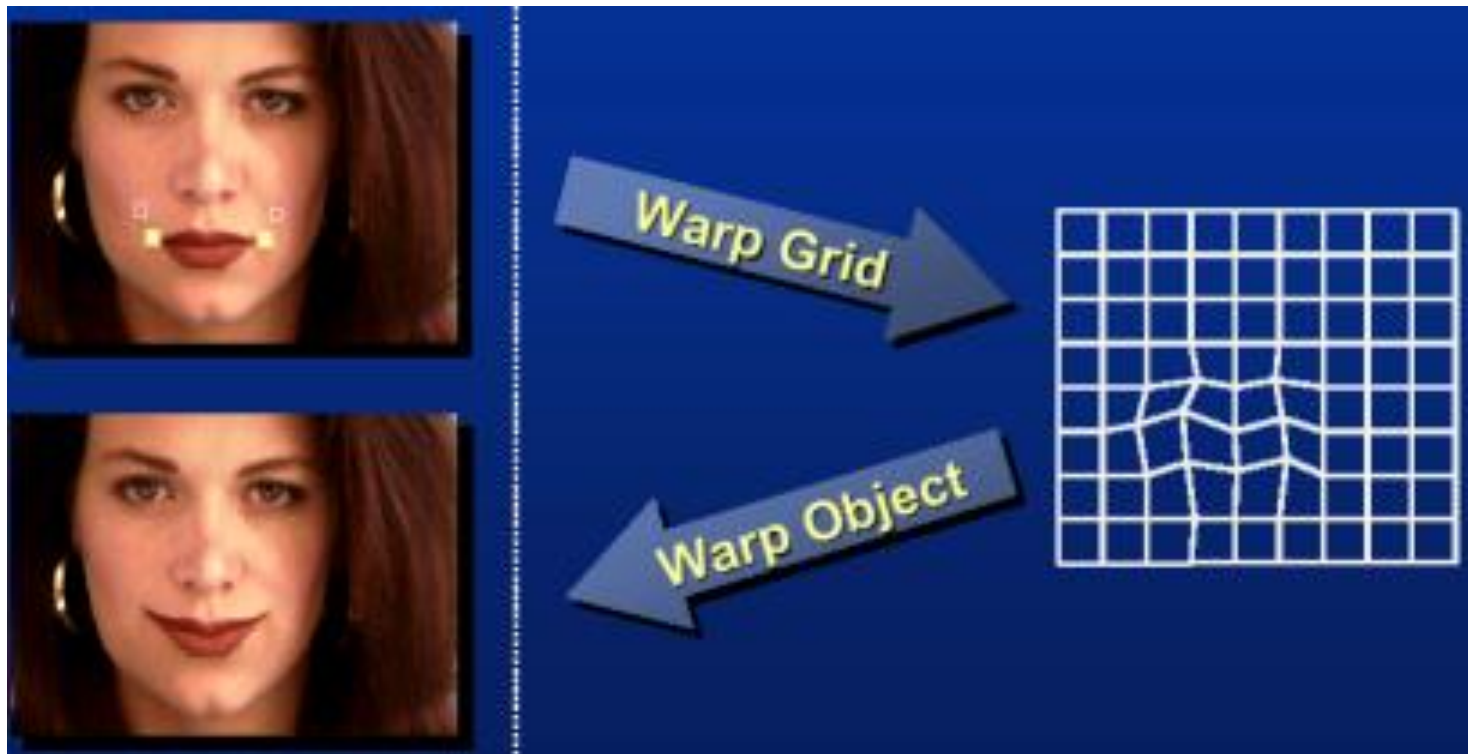
Need to specify a more detailed warp function

- Global warps were functions of a few (2,4,8) parameters
- Non-parametric warps $u(x,y)$ and $v(x,y)$ can be defined independently for every single location x,y !
- Once we know vector field u,v we can easily warp each pixel (use backward warping with interpolation)

Image Warping – non-parametric

Move control points to specify a spline warp

Spline produces a smooth vector field



Warp specification - dense

How can we specify the warp?

Specify corresponding *spline control points*

- *interpolate* to a complete warping function



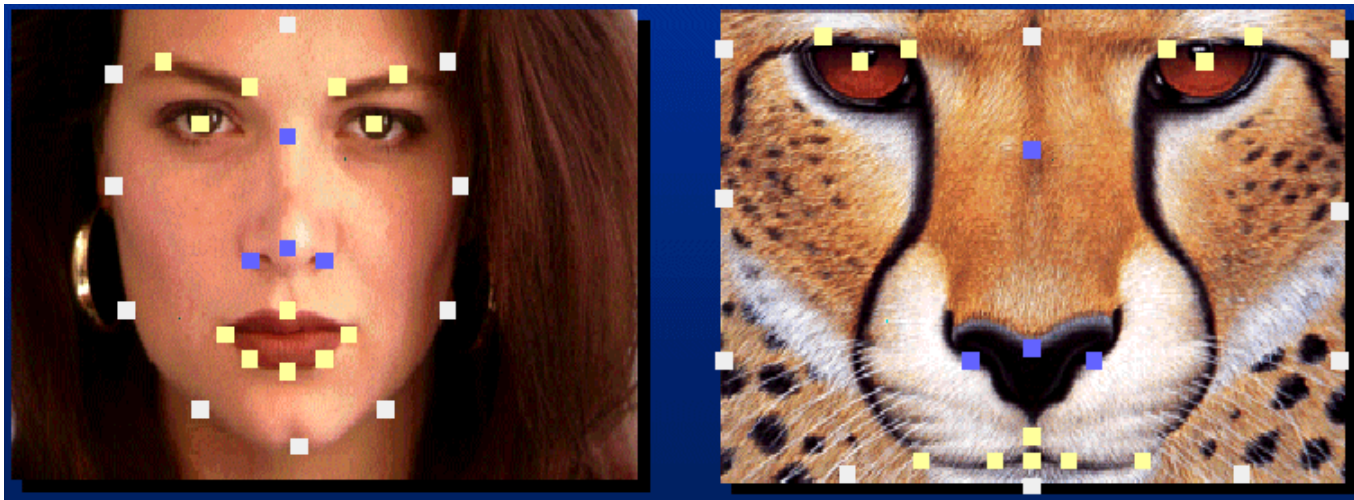
But we want to specify only a few points, not a grid

Warp specification - sparse

How can we specify the warp?

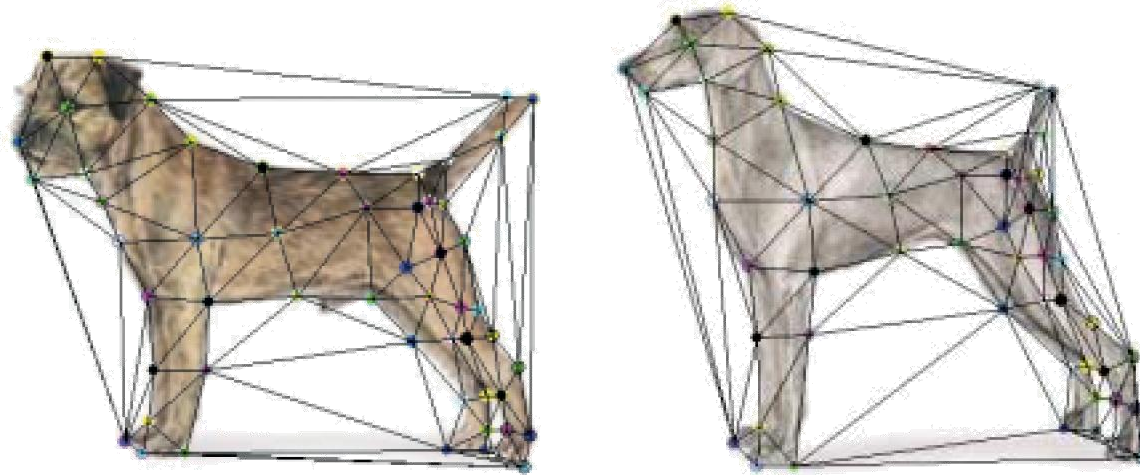
Specify corresponding *points*

- *interpolate* to a complete warping function
- How do we do it?



How do we go from feature points to pixels?

Triangular Mesh

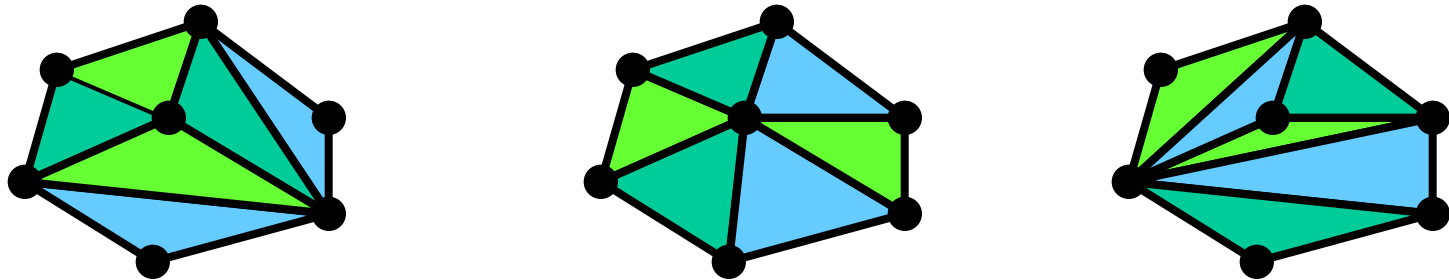


1. Input correspondences at key feature points
2. Define a triangular mesh over the points
 - Same mesh (triangulation) in both images!
 - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately from source to destination
 - Affine warp with three corresponding points (just like the exercise)

Triangulations

A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.

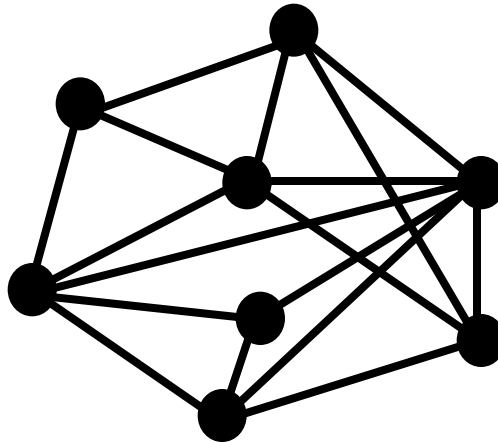
There are an exponential number of triangulations of a point set.



An $O(n^3)$ Triangulation Algorithm

Repeat until impossible:

- Select two sites.
- If the edge connecting them does not intersect previous edges, keep it.



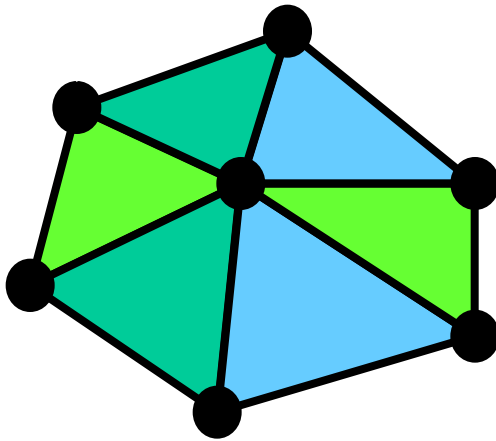
“Quality” Triangulations

Let $\alpha(T) = (\alpha_1, \alpha_2, \dots, \alpha_{3t})$

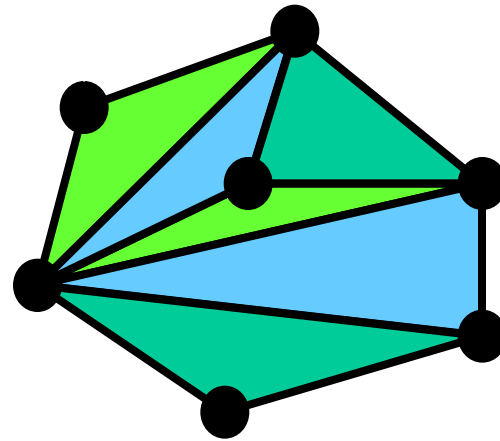
be the vector of angles in the triangulation T in increasing order. A triangulation will be “better” than if

$\alpha(T_1) > \alpha(T_2)$ lexicographically

The Delaunay triangulation is the “best” maximizes smallest angles



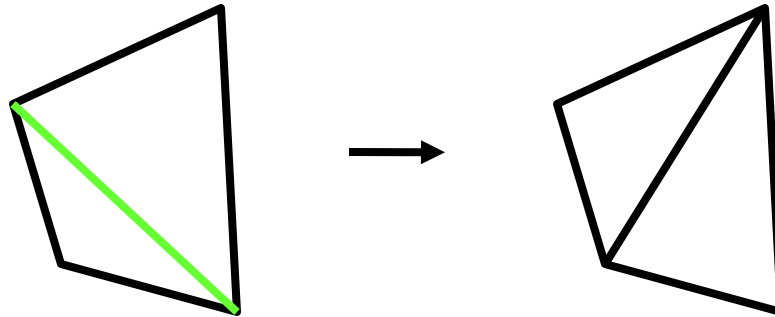
good



bad

Improving a Triangulation

In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.

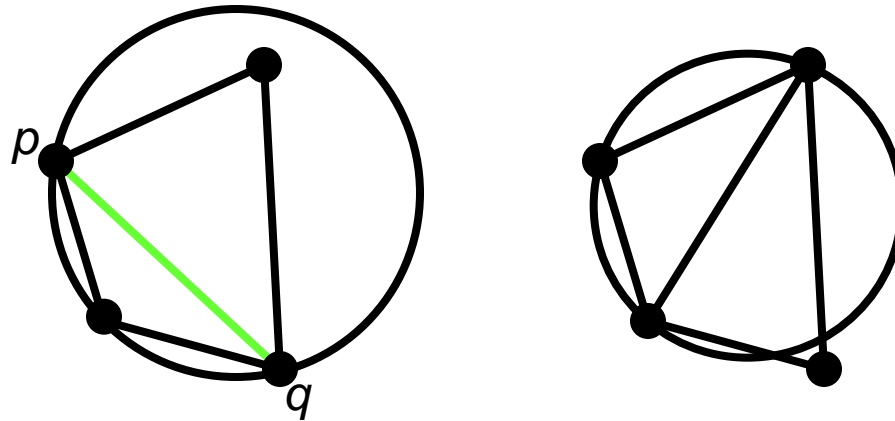


If an edge flip improves the triangulation, the first edge is called *illegal*.

Illegal Edges

Lemma: An edge pq is illegal iff one of its opposite vertices is inside the circle defined by the other three vertices.

Proof: By Thales' theorem.



Theorem: A Delaunay triangulation does not contain illegal edges.

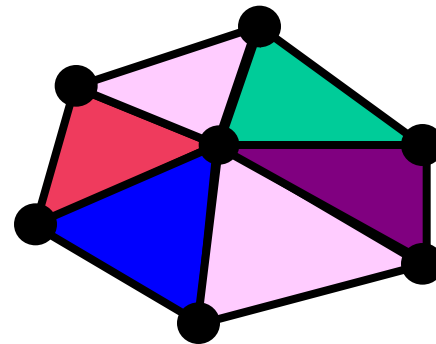
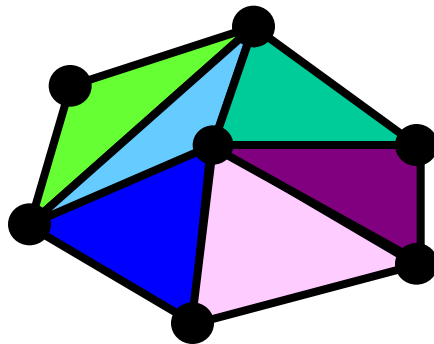
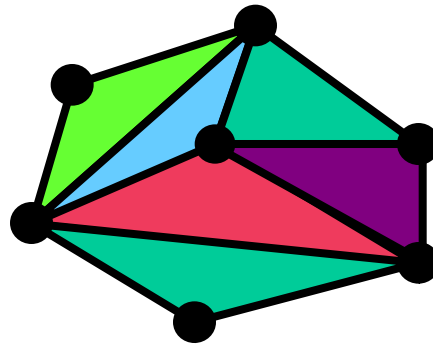
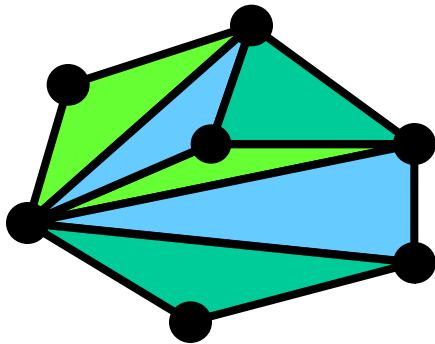
Corollary: A triangle is Delaunay if the circle through its vertices is empty of other sites.

Corollary: The Delaunay triangulation is not unique if more than three sites are co-circular.

Naïve Delaunay Algorithm

Start with an arbitrary triangulation. Flip any illegal edge until no more exist.

Could take a long time to terminate.



Delaunay Triangulation by Duality

General position assumption: There are no four co-circular points.

Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.

Corollary: The DT may be constructed in $O(n \log n)$ time.

This is what Matlab's `delaunay` function uses.

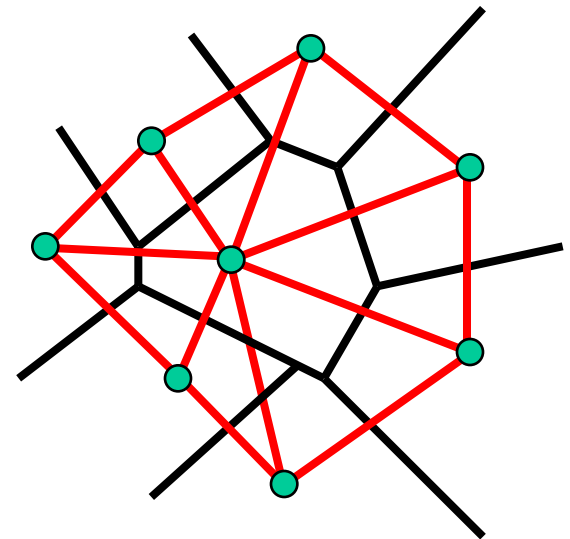
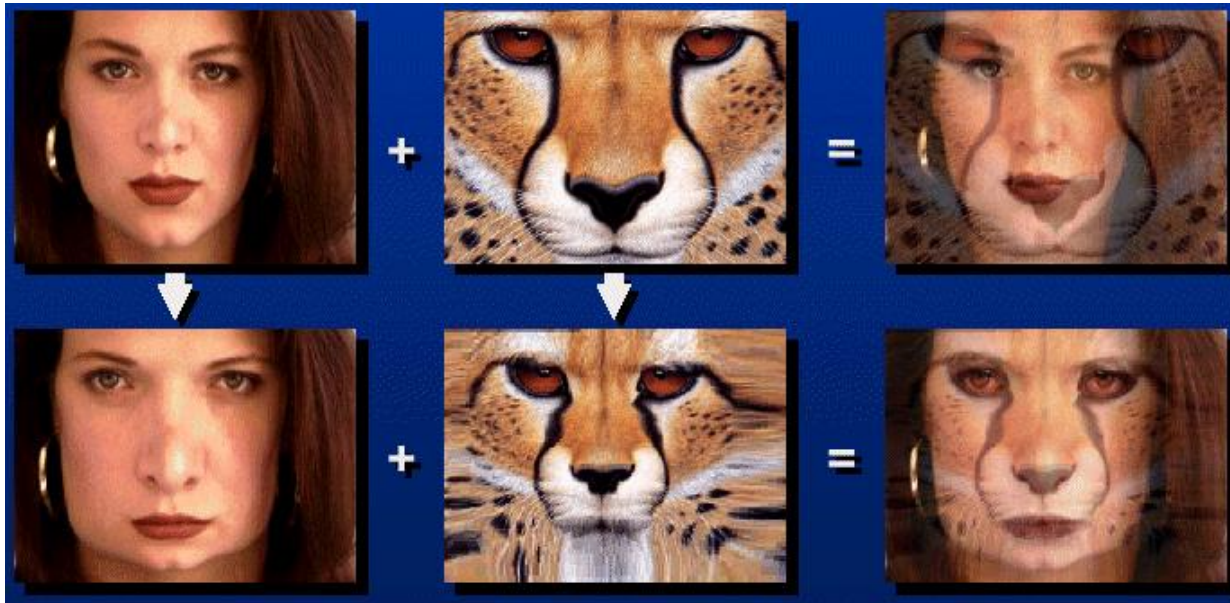


Image Morphing

We know how to warp one image into the other, but how do we create a morphing sequence?

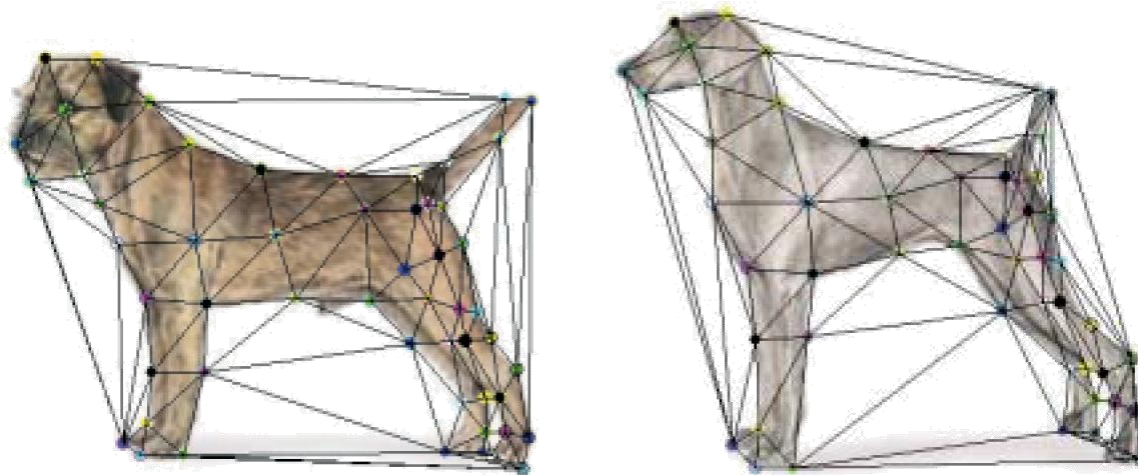
1. Create an intermediate shape (by interpolation)
2. Warp both images towards it
3. Cross-dissolve the colors in the newly warped



Warp interpolation

How do we create an intermediate warp at time t ?

- Assume $t = [0,1]$
- Simple linear interpolation of each feature pair
- $(1-t)*p1+t*p0$ for corresponding features $p0$ and $p1$



Morphing & matting

Extract foreground first to avoid artifacts
in the background



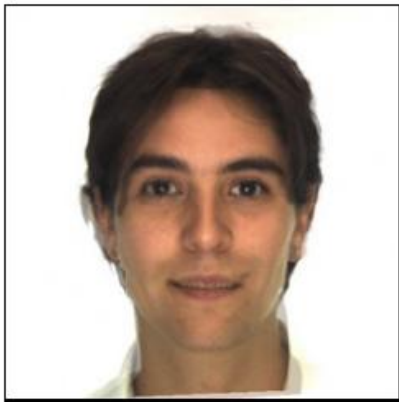
(c) $\alpha = 0.0$



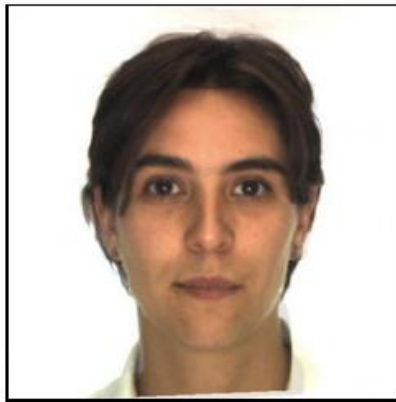
(d) $\alpha = 0.2$



(e) $\alpha = 0.4$



(f) $\alpha = 0.6$

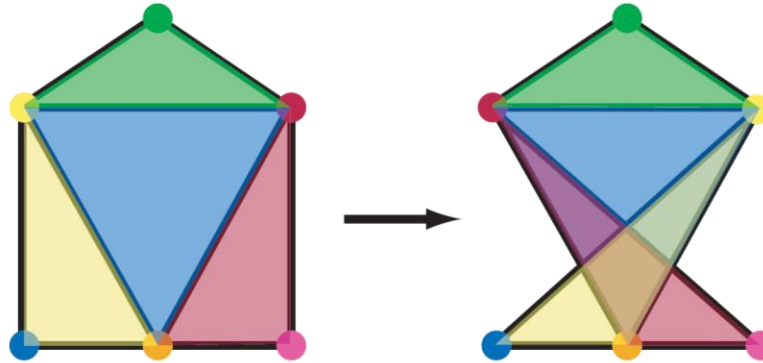


(g) $\alpha = 0.8$



(h) $\alpha = 1.0$

Other Issues



Beware of folding

- You are probably trying to do something 3D-ish

Morphing can be generalized into 3D

- If you have 3D data, that is!

Extrapolation can sometimes produce interesting effects

- Caricatures

Dynamic Scene



Black or White (MJ):

<http://www.youtube.com/watch?v=R4kLKv5gtxc>

Willow morph: <http://www.youtube.com/watch?v=uLUyuWo3pG0>