Global and Local Image Warping
Jun-Yan Zhu

16-726 Learning-based Image Synthesis, Spring 2022

Logistics

1. Homework 0 released

   Deadline: 02/01/2022

How to submit assignments? (in general)

For each assignment, you will be required to submit two packages unless specified otherwise: your **code** and your **webpage**:

Submit code to Canvas

For every assignment you should create a `main.py` that can be used to run all your code for the assignment, and a `README.md` file that contains all required documentation. Place all source code used to generate your results, as well as any documentation required to use the code, in a folder named `andrewid_code_projX` where X is the hw number. Zip the whole folder and submit the zip to Canvas. Here is an example of your folder structure:

```
zhiqiul_code_proj1/
    main.py
    README.md
    utils.py
    ....
## zip the whold folder to zhiqiul_code_proj1.zip:
```

Submit your webpage to the class website

We will use Andrew File System(AFS) to store and display webpages. Here is a step by step tutorial:

1. Place your website under folder `projX` and zip it. Please make sure that your main report page is called `index.html` so browsers open it automatically. X is the hw number.
2. Remote Copy. Use WinSCP or your favorite scp/ftp tool to copy all your files to your Andrew home directory `scp projX.zip andrew_id@linux.andrew.cmu.edu:/home/andrew_id/`.
3. Log in to a Unix Andrew machine: `ssh andrew_id@linux.andrew.cmu.edu`
4. File Transfer. Unzip your website and copy the folder to your project directory: `cp -r projX/`

Logistics

2. Hybrid mode (in-person + remote + rotation)

   Group A: Monday

   Group B: Wednesday

3. Optional reading list: read after each class

4. Student Presentation sign-up sheet.

   Data-driven Graphics

   Generative Models

   Conditional Image Synthesis

   Image Editing with Optimization

   Face Modeling

   3D and Video Synthesis

Student Presentation

Data-Driven Graphics (2000-2010)

- Graphcut Texture, SIGGRAPH 2003.

- Interactive Digital Photomontage, SIGGRAPH 2004.

- GrabCut, SIGGRAPH 2004.

- Hybrid Images, SIGGRAPH 2006.

- Photo Clip Art, SIGGARPH 2007.

- Image Deformation Using Moving Least Squares. SIGGRAPH 2006.

- Seam Carving for Content-Aware Image Resizing, SIGGRAPH 2007.

- Sketch2Photo, SIGGRAPH Asia 2009.

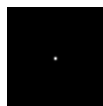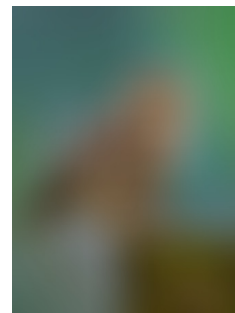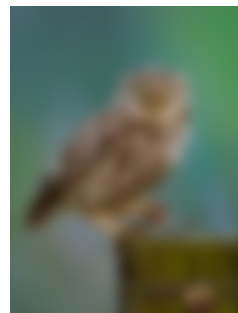- Coordinates for Instant Image Cloning, SIGGRAPH 2009
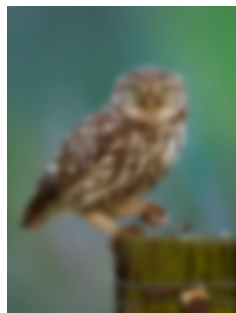
- …

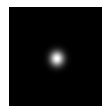# Review: how to create an image
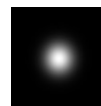
- Pointwise Processing:
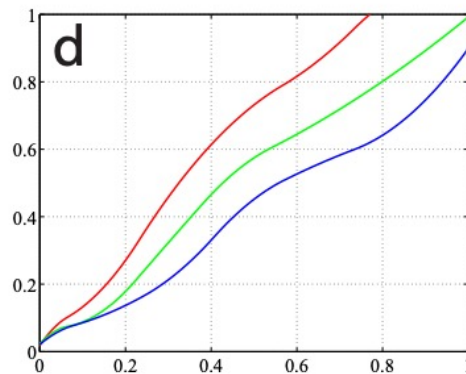
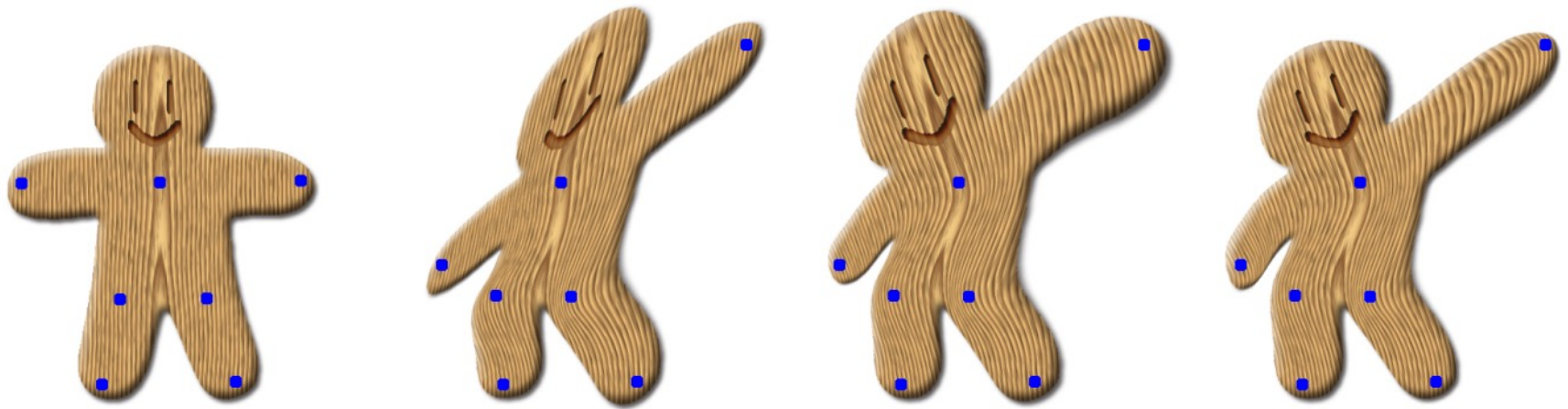- Image Filtering

σ = 1 pixel σ = 5 pixels σ = 10 pixels σ = 30 pixels

5

Global and Local Image Warping

Jun-Yan Zhu

16-726 Learning-based Image Synthesis, Spring 2022

Image Transformations

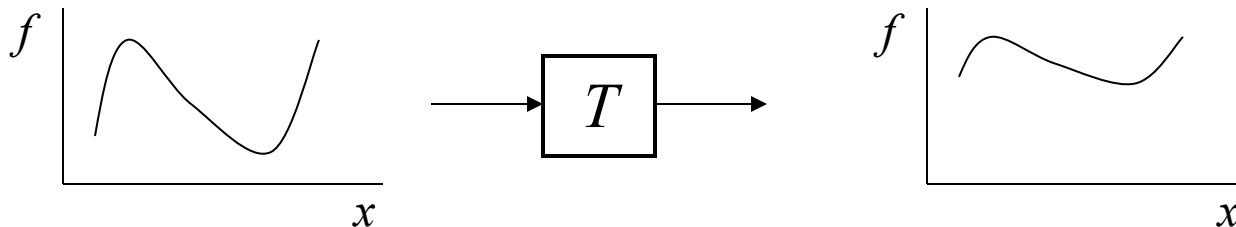image filtering: change **range** of image

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

image warping: change **domain** of image
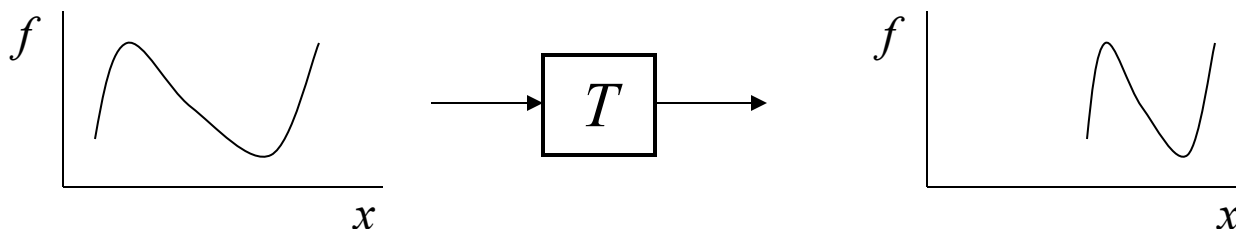
$$g(x) = f(T(x))$$

Image Transformations

image filtering: change **range** of image
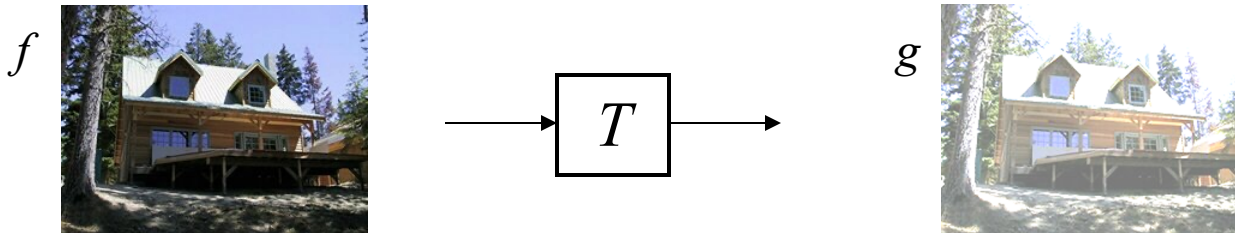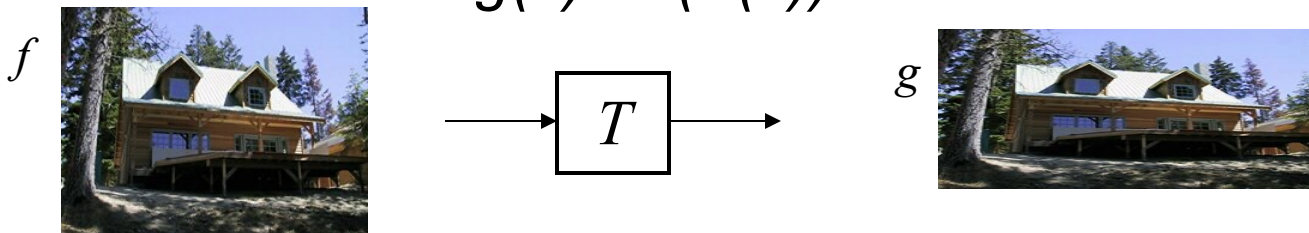
$$g(x) = T(f(x))$$

image warping: change **domain** of image

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

Parametric (global) warping

Examples of parametric warps:
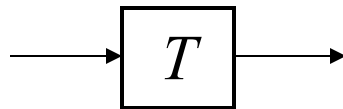

translation

rotation

aspect

affine

perspective

cylindrical

Parametric (global) warping

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

Transformation T is a coordinate-changing machine:

$$p' = T(p)$$

What does it mean that *T* is global?

- Is the same for any point p
- can be described by just a few numbers (parameters)
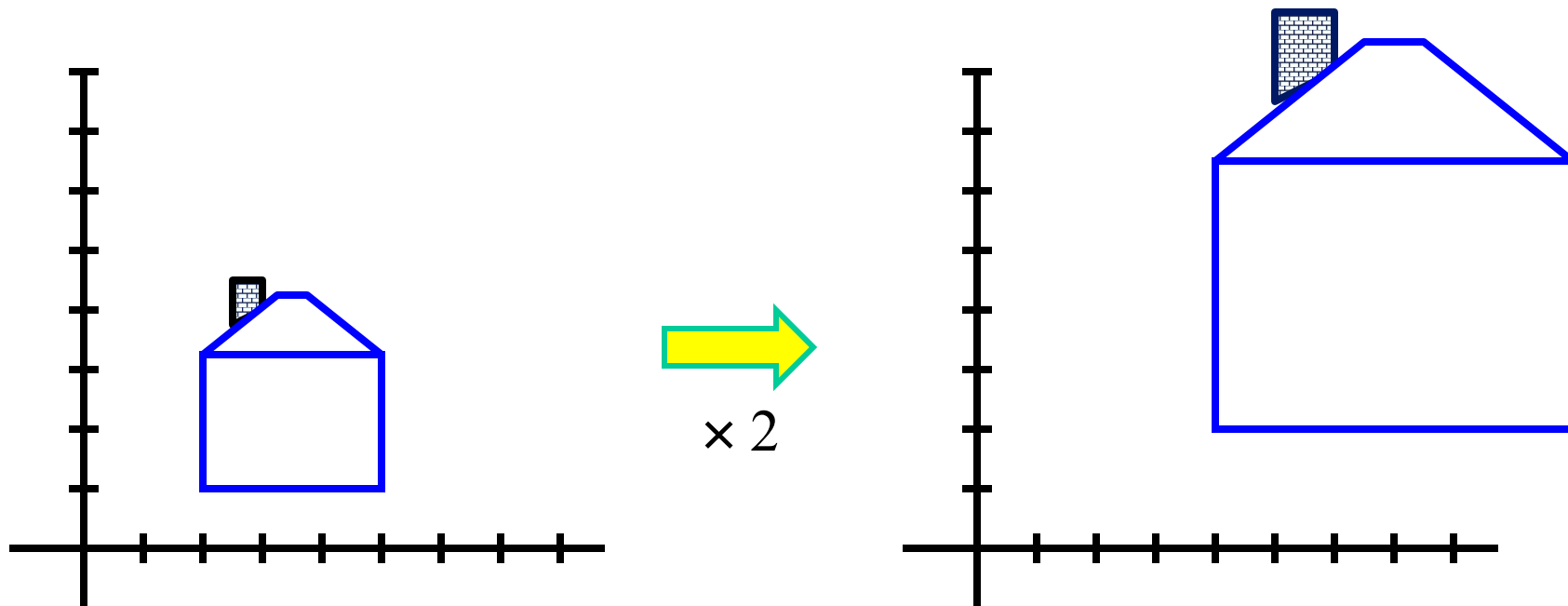
Let's represent a <u>linear</u> *T* as a matrix:

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

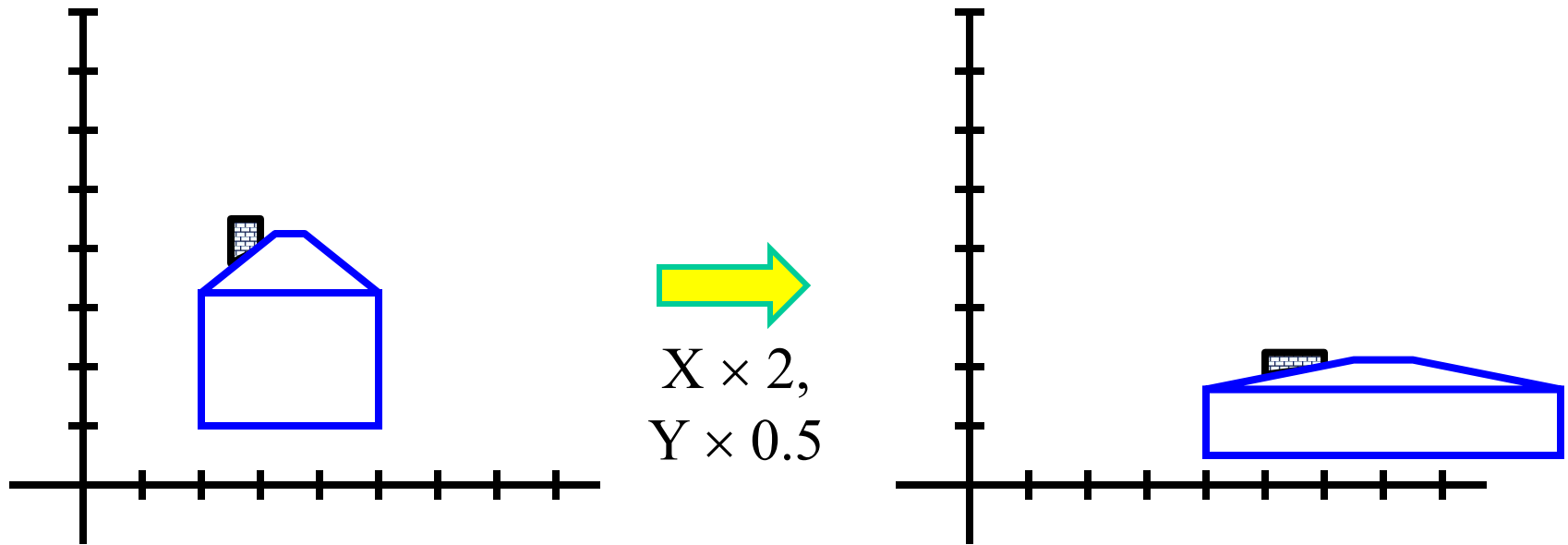$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling

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

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

$\times 2$

Scaling

Non-uniform scaling: different scalars per component:

$$X \times 2,$$
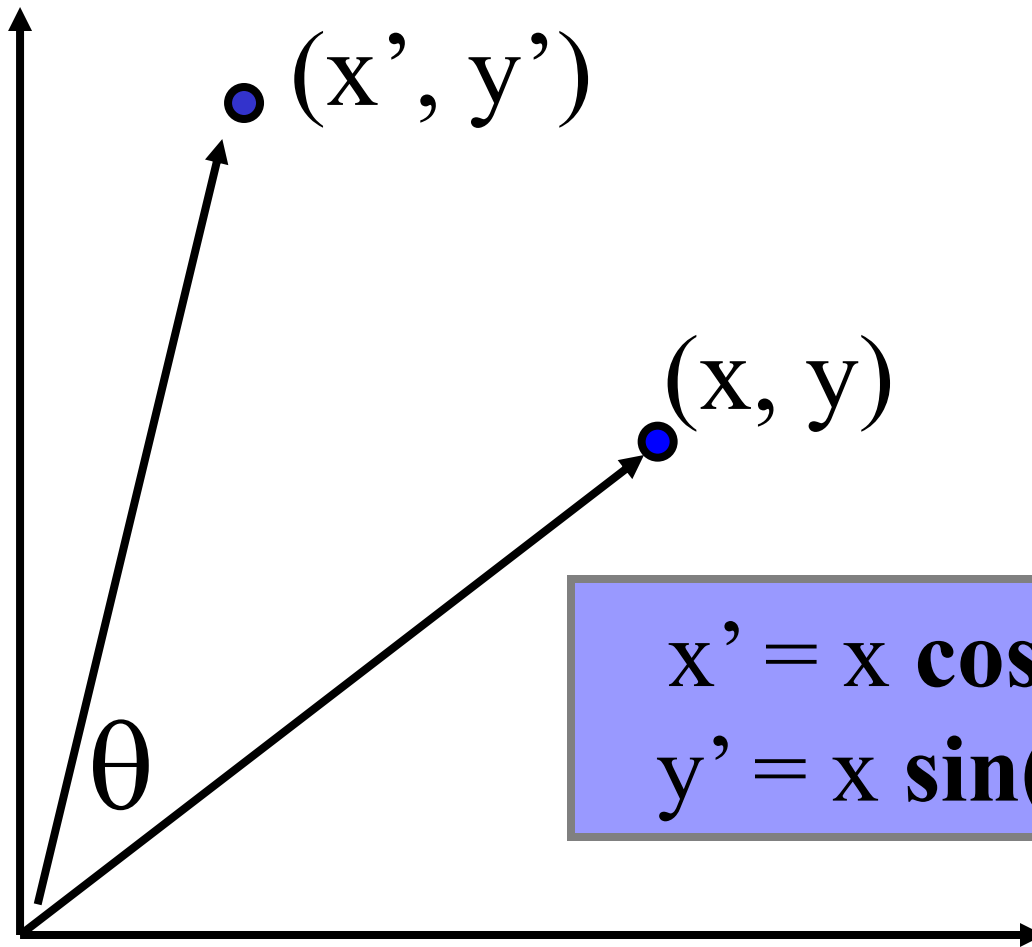$$Y \times 0.5$$

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

$$x' = x \, \mathbf{cos}(\theta) - y \, \mathbf{sin}(\theta)$$
$$y' = x \, \mathbf{sin}(\theta) + y \, \mathbf{cos}(\theta)$$

2-D Rotation

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

(x', y')

(x, y)

θ

ϕ

2-D Rotation

$x = r \cos(\phi)$

$y = r \sin(\phi)$

$x' = r \cos(\phi + \theta)$

$y' = r \sin(\phi + \theta)$

Trig Identity…

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$

$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

(x', y')

(x, y)

θ

ϕ

2-D Rotation

$x = r \cos(\phi)$

$y = r \sin(\phi)$

$x' = r \cos(\phi + \theta)$
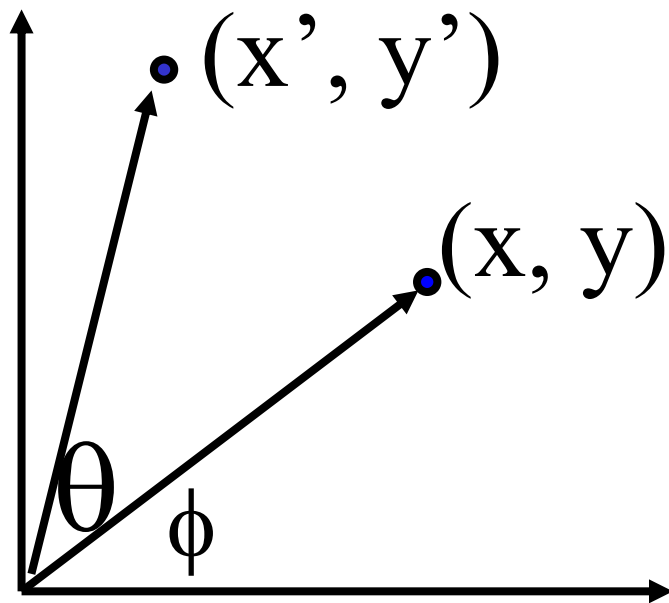
$y' = r \sin(\phi + \theta)$
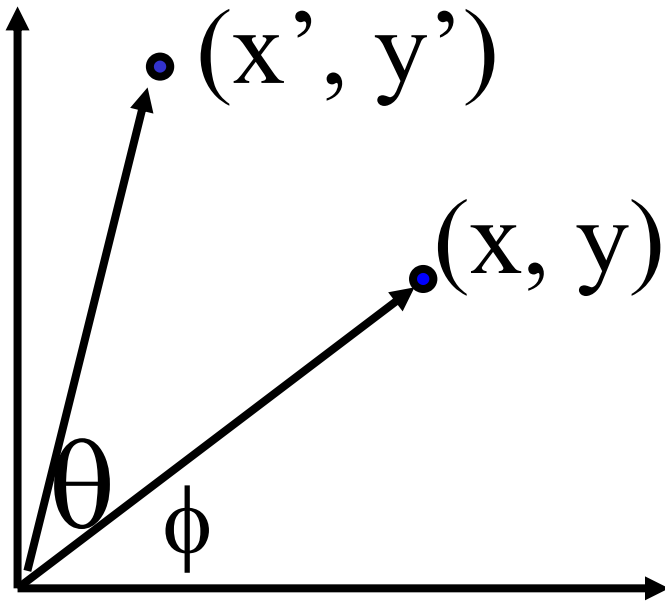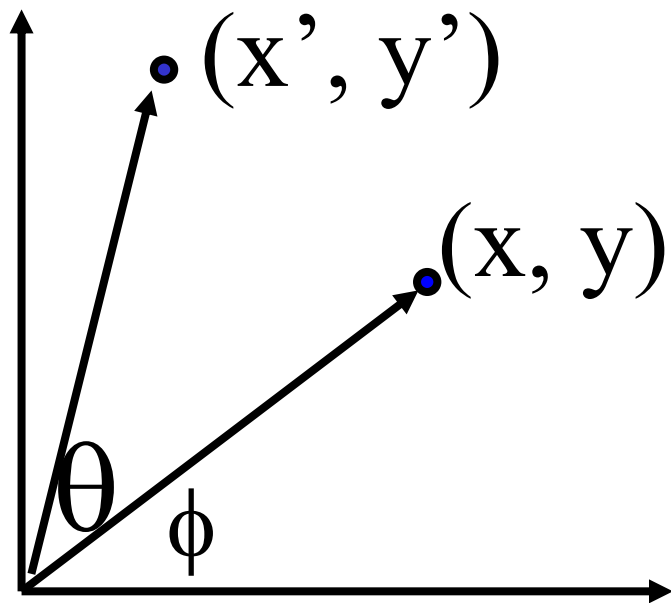
Trig Identity…

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$

$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

Substitute…

$x' = x \, \textbf{cos}(\theta) - y \, \textbf{sin}(\theta)$

$y' = x \, \textbf{sin}(\theta) + y \, \textbf{cos}(\theta)$

(x', y')

(x, y)

θ

ϕ

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 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?

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

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

2D Scale around (0,0)?

$$\boldsymbol{x}' = \boldsymbol{s}_x * \boldsymbol{x}$$
$$\boldsymbol{y}' = \boldsymbol{s}_y * \boldsymbol{y}$$

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

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?

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

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

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

Properties of linear transformations:

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

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

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

Homogeneous Coordinates

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

$$x' = x + t_x$$

$$y' = y + t_y$$

Homogeneous Coordinates
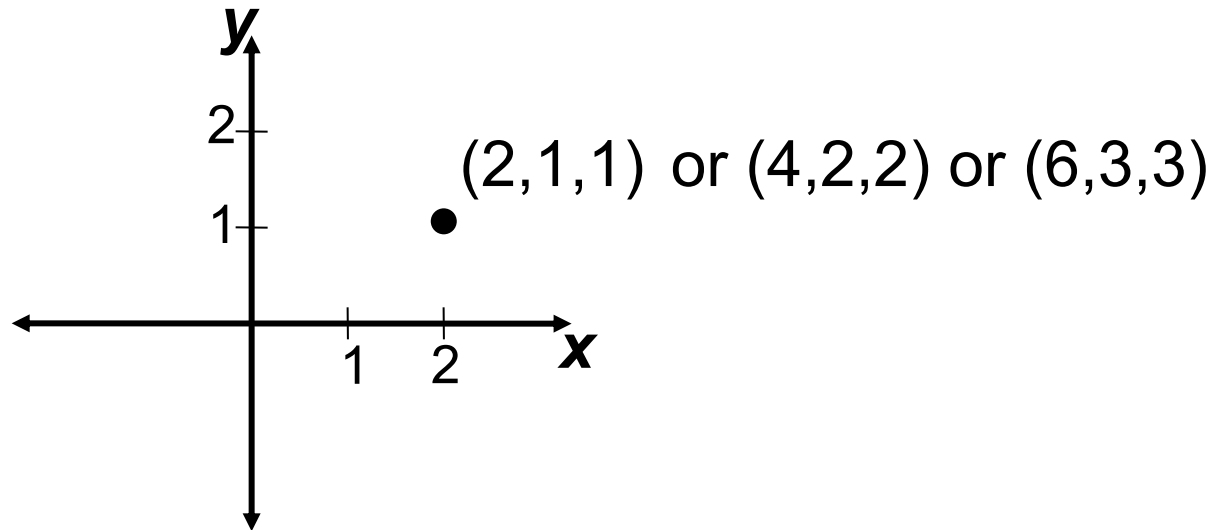
Homogeneous coordinates

- represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

$(2,1,1)$ or $(4,2,2)$ or $(6,3,3)$

Convenient coordinate system to represent many useful transformations

30

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:

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

Translation

Example of translation

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$
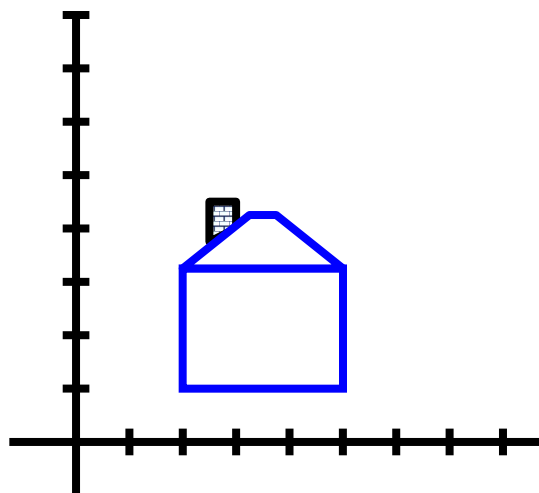
$t_x = 2$
$t_y = 1$
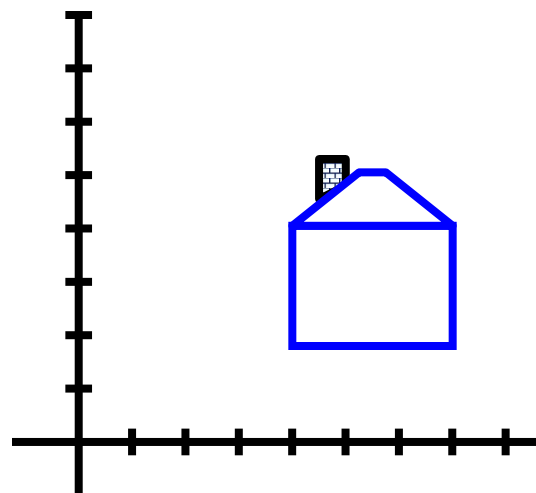
Basic 2D Transformations

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

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

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

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

Shear

Matrix Composition

Transformations can be combined by
matrix multiplication

$$
\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}
$$

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

Does the order of multiplication matter?

Affine Transformations

Affine transformations are combinations of …

- Linear transformations, and
- Translations

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

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

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

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

2D image transformations
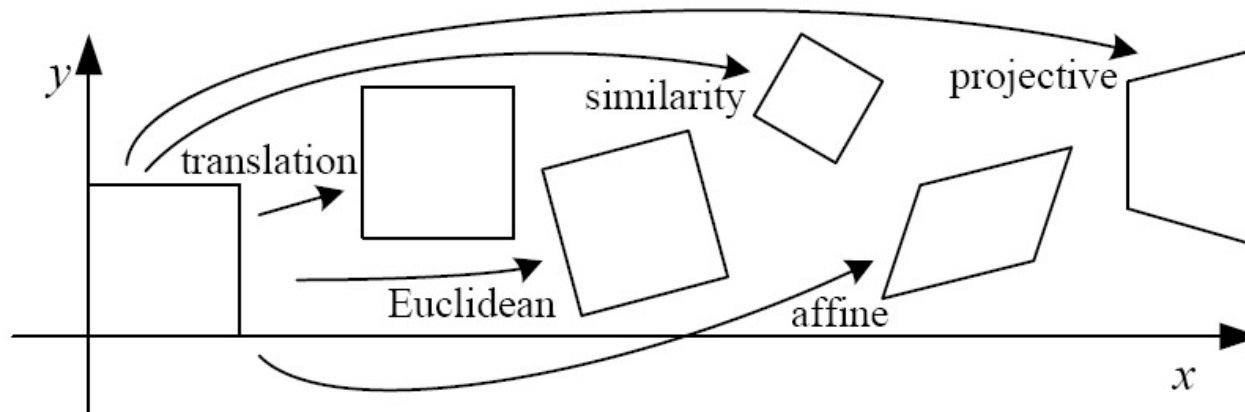
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c|c} I & t \end{array}\right]_{2\times3}$			
rigid (Euclidean)	$\left[\begin{array}{c|c} R & t \end{array}\right]_{2\times3}$			
similarity	$\left[\begin{array}{c|c} sR & t \end{array}\right]_{2\times3}$			
affine	$\left[\begin{array}{c} A \end{array}\right]_{2\times3}$			
projective	$\left[\begin{array}{c} \tilde{H} \end{array}\right]_{3\times3}$			

These transformations are a nested set of groups
   • Closed under composition and inverse is a member

Image Warping in Biology

D'Arcy Thompson

Importance of shape and structure in evolution
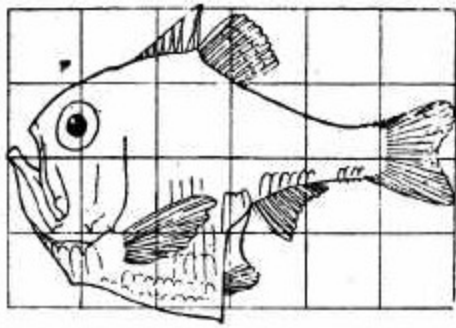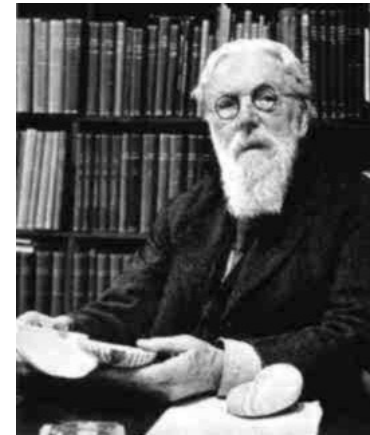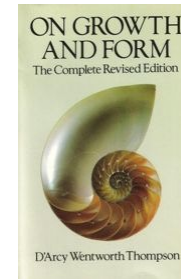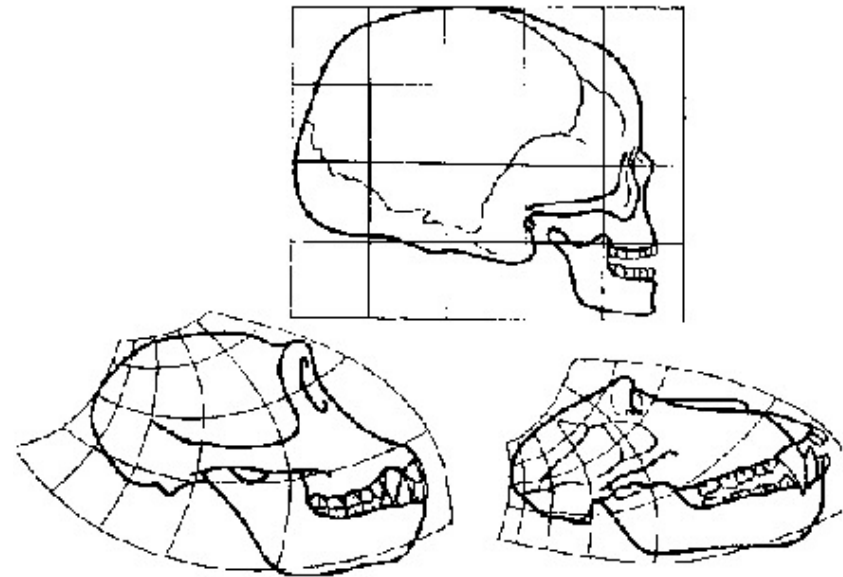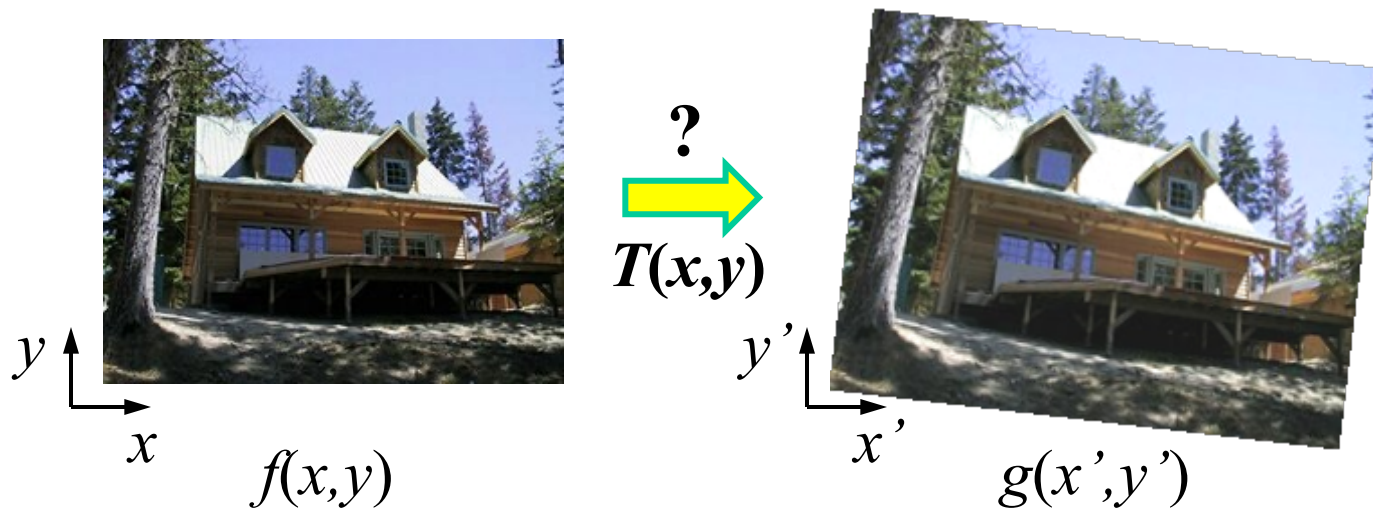

Fig. 517. *Argyropelecus Olfersi.*
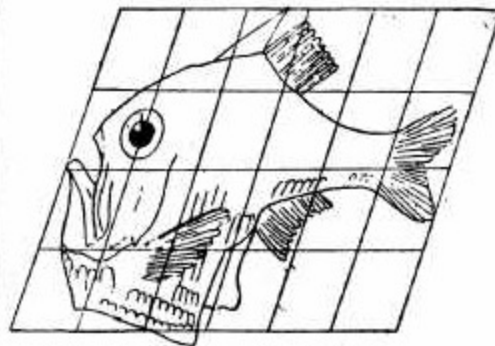
Fig. 518. *Sternoptyx diaphana.*

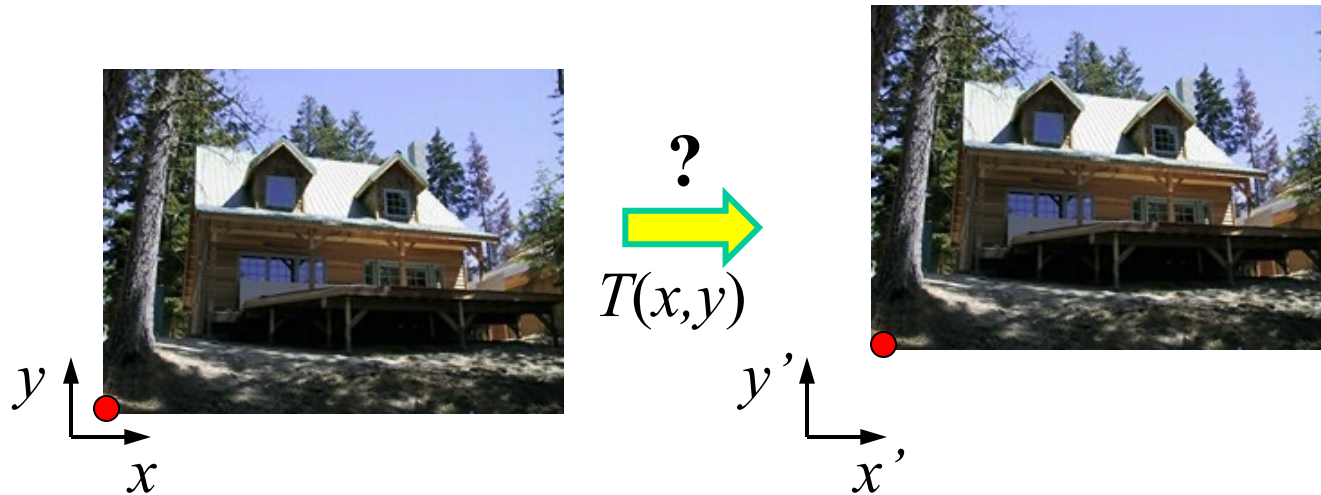Skulls of a human, a chimpanzee and a baboon and transformations between them

38

Recovering Transformations

$f(x,y)$ $T(x,y)$ $g(x',y')$

What if we know *f* and *g* and want to recover the transform T?

- e.g. better align images from Project 1
- willing to let user provide correspondences
  - How many do we need?

Translation: # correspondences?

?

$T(x,y)$

y

x

y'

x'

How many correspondences needed for translation?

How many Degrees of Freedom?

What is the transformation matrix?

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

40

Euclidian: # correspondences?

$T(x,y)$

y

x

y'

x'
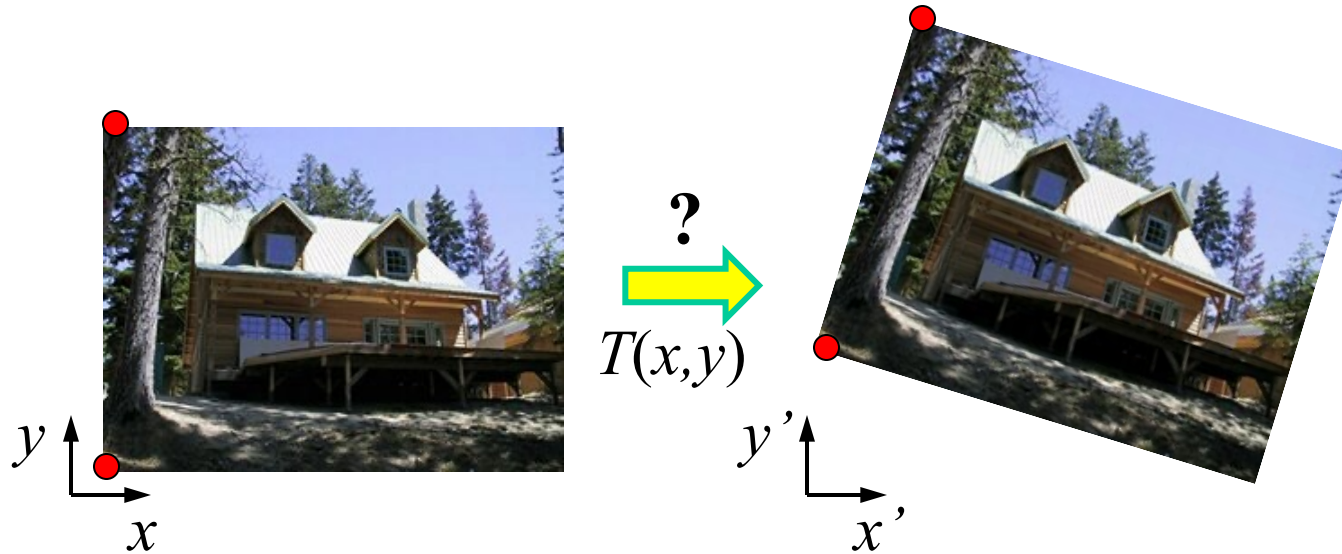
How many correspondences needed for translation+rotation?

How many DOF?

Affine: # correspondences?

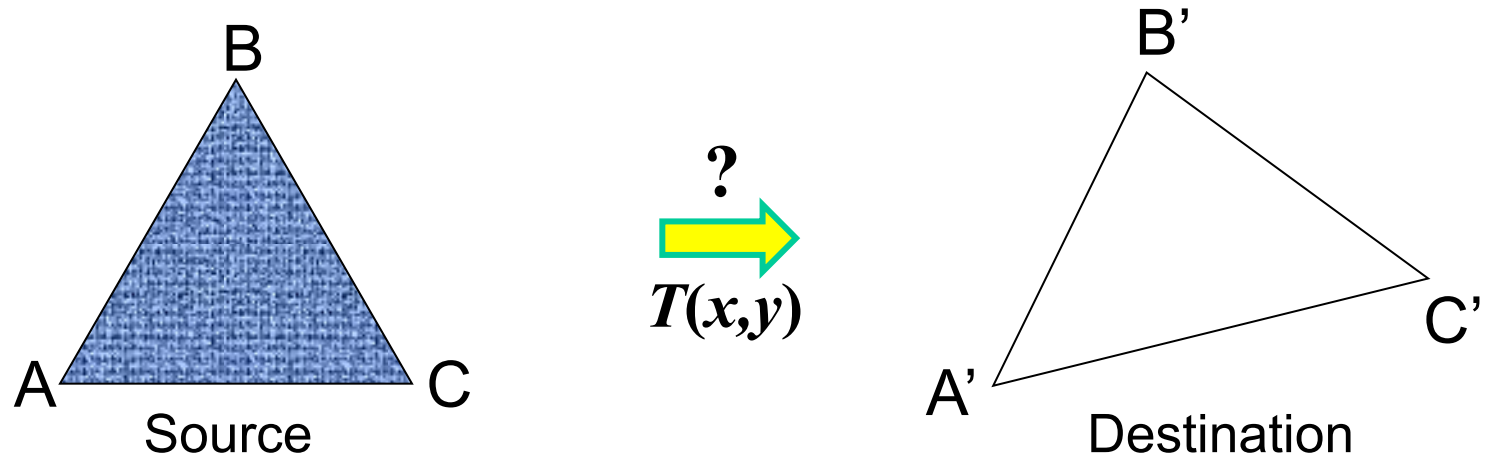$T(x,y)$
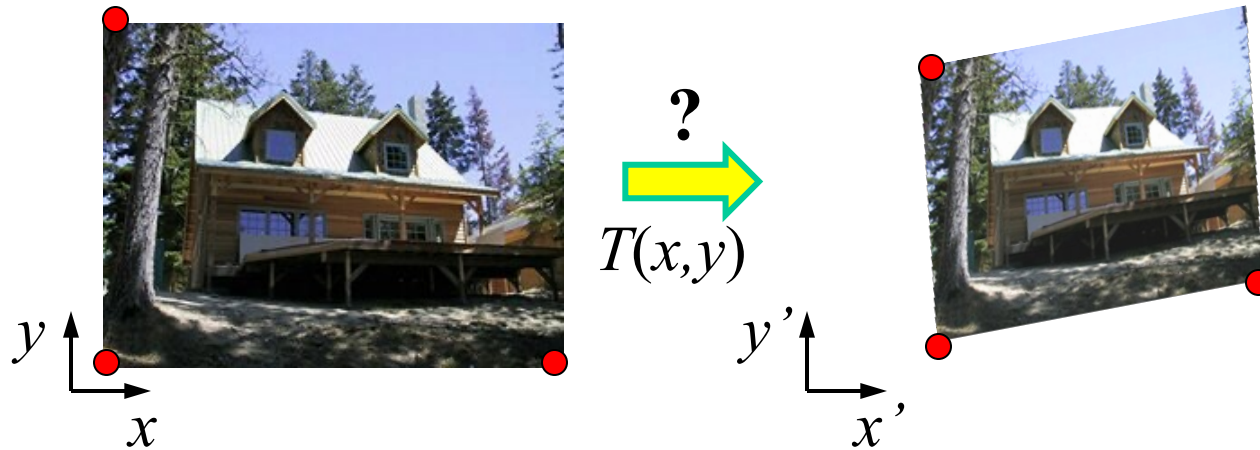
How many correspondences needed for affine?

How many DOF?

Projective: # correspondences?

$T(x,y)$
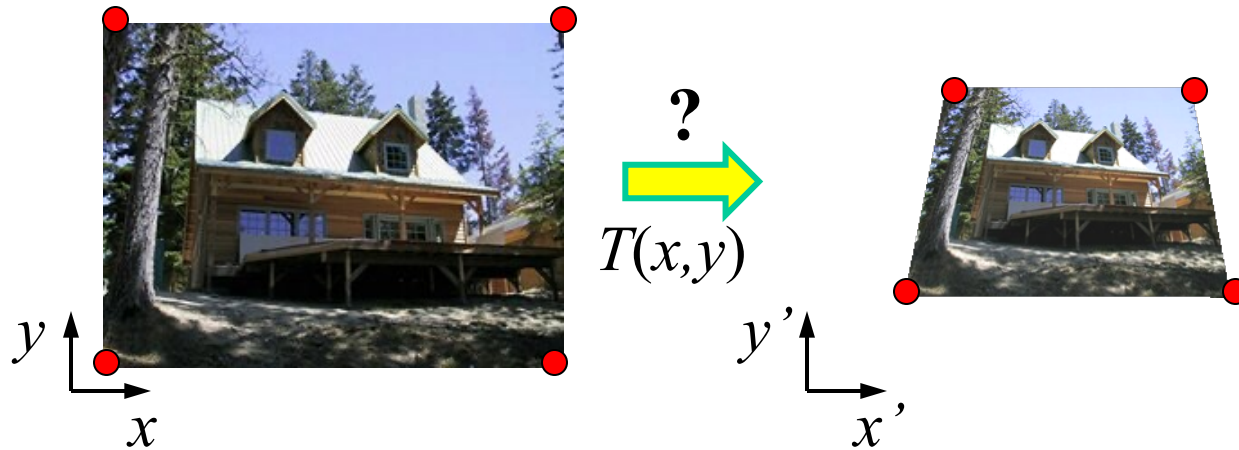
How many correspondences needed for projective?
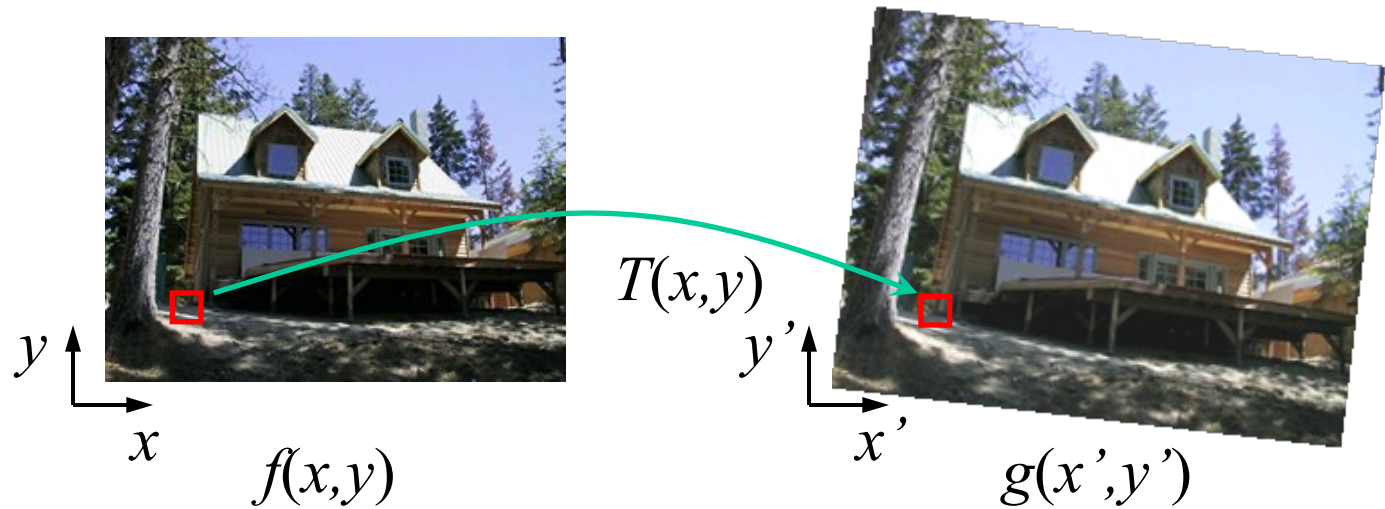
How many DOF?

Example: warping triangles

B

B'

?

$T(x,y)$

C'

A   C

A'

Source

Destination

Given two triangles: ABC and A'B'C' in 2D (12 numbers)

Need to find transform T to transfer all pixels from one to the other.

What kind of transformation is T?

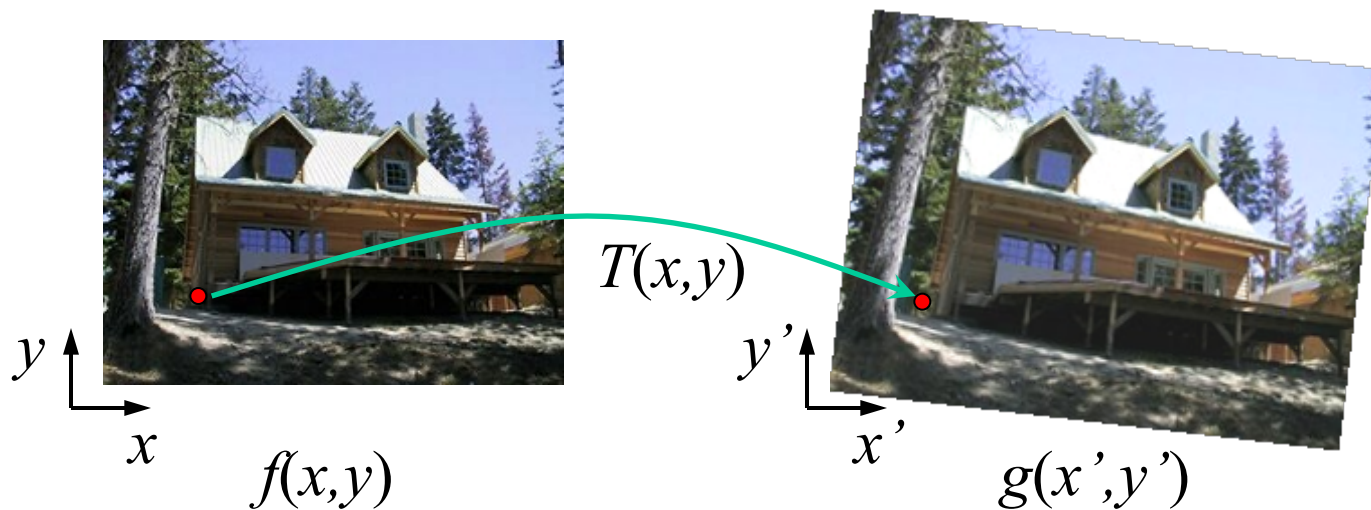How can we compute the transformation matrix:

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

Image warping

$T(x,y)$

$f(x,y)$

$g(x',y')$

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))*?
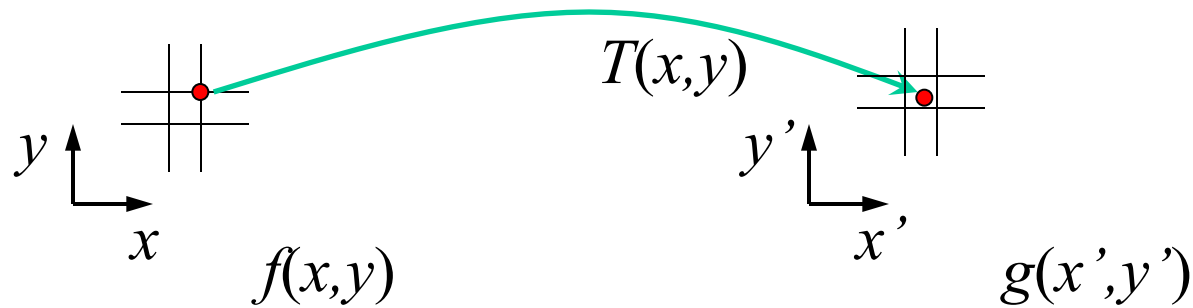
Forward warping

Send each pixel *f*(*x*,*y*) to its corresponding location

(*x'*,*y'*) = *T*(*x*,*y*) in the second image

Q: what if pixel lands "between" two pixels?

Forward warping

$T(x,y)$

y ↑
x →
$f(x,y)$

y' ↑
x' →
$g(x',y')$

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')
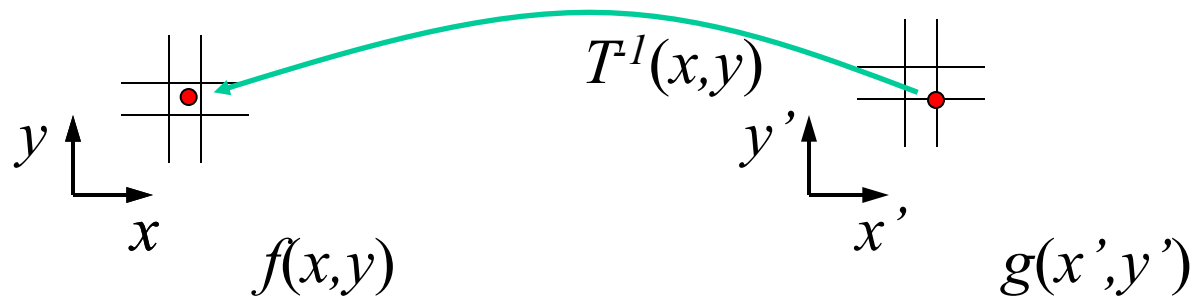
– Known as "splatting"

– Check out `griddata` in Matlab

Inverse warping
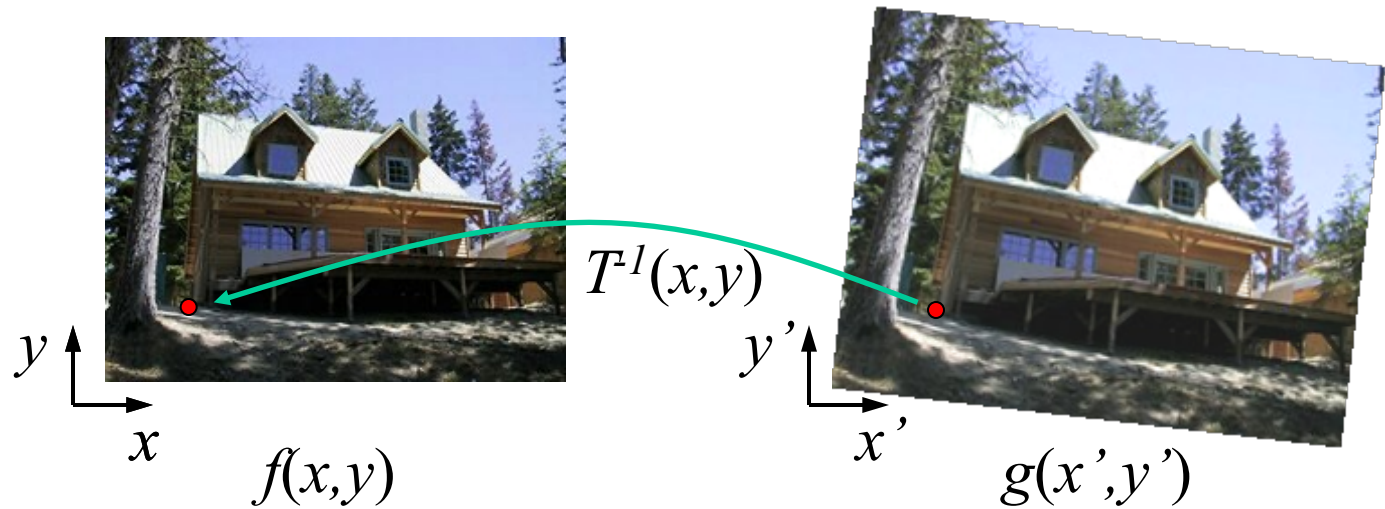
$T^{-1}(x,y)$

y

x

$f(x,y)$

y'

x'

$g(x',y')$

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?

Inverse warping

$T^{-1}(x,y)$

y
x
$f(x,y)$

y'
x'
$g(x',y')$

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

Forward vs. inverse warping

Q: which is better?

A: usually inverse—eliminates holes

- however, it requires an invertible warp function—not always possible...

Morphing = Object Averaging

The aim is to find "an average" between two objects

- Not an average of two <u>images of objects</u>…
- …but an image of the <u>average object</u>!
- 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?

$\mathbf{v} = Q - P$

Q

P
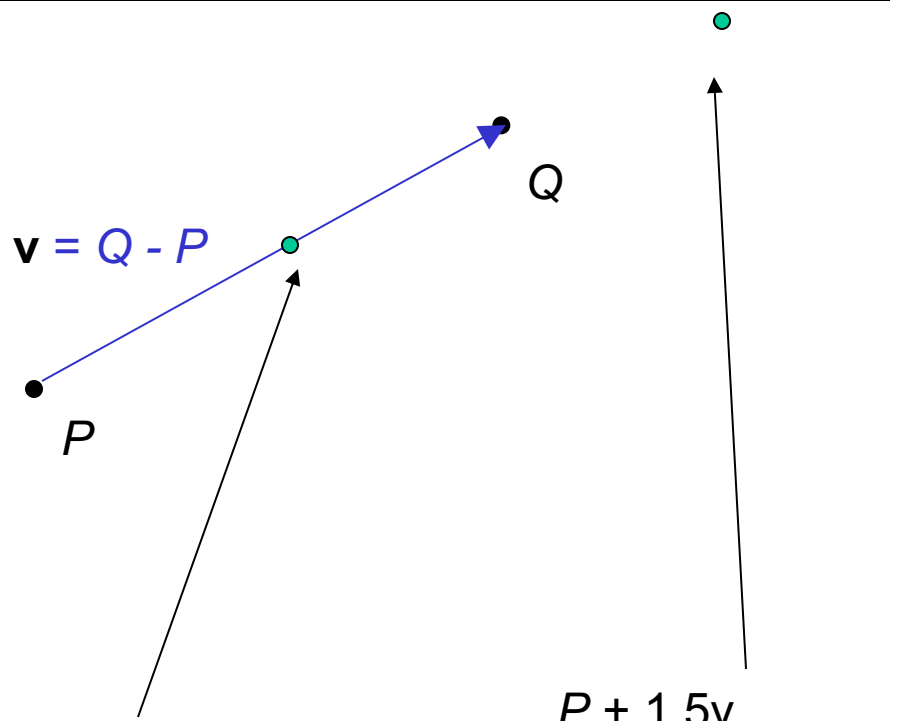
$P + 0.5v$
$= P + 0.5(Q - P)$
$= 0.5P + 0.5Q$

$P + 1.5v$
$= P + 1.5(Q - P)$
$= -0.5P + 1.5Q$
(extrapolation)

Linear Interpolation
(Affine Combination):
New point $aP + bQ$,
defined only when $a+b = 1$
So $aP+bQ = aP+(1-a)Q$

P and Q can be anything:

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

Idea #1: Cross-Dissolve

Interpolate whole images:

$$Image_{halfway} = (1-t)*Image_1 + t*image_2$$

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

But what is the images are not aligned?

Idea #2: Align, then cross-disolve

Align first, then cross-dissolve
- Alignment using global warp – picture still valid

Global warp not always enough!

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

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)

Warp specification -- dense

Define vector field to specify a dense warp
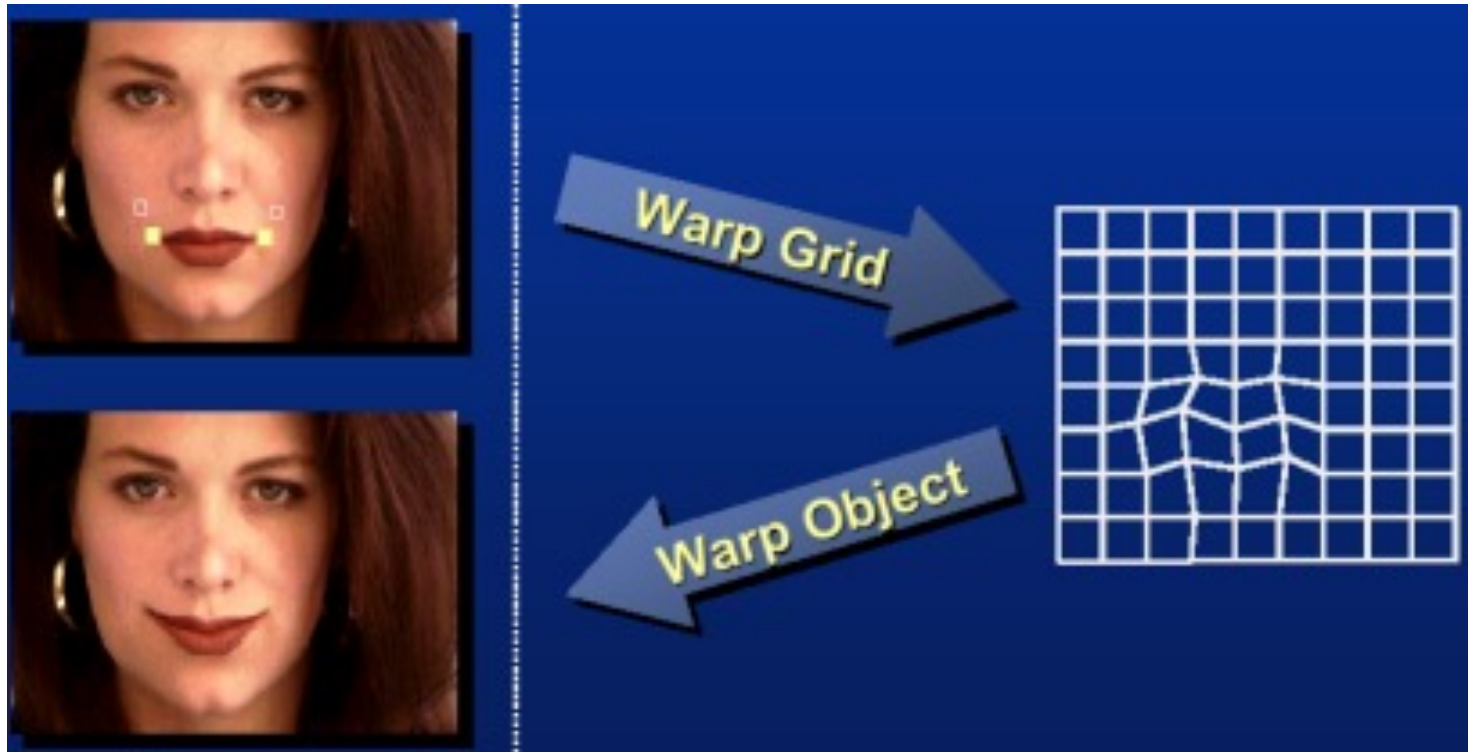
Warp specification - sparse

How can we specify a sparse warp?
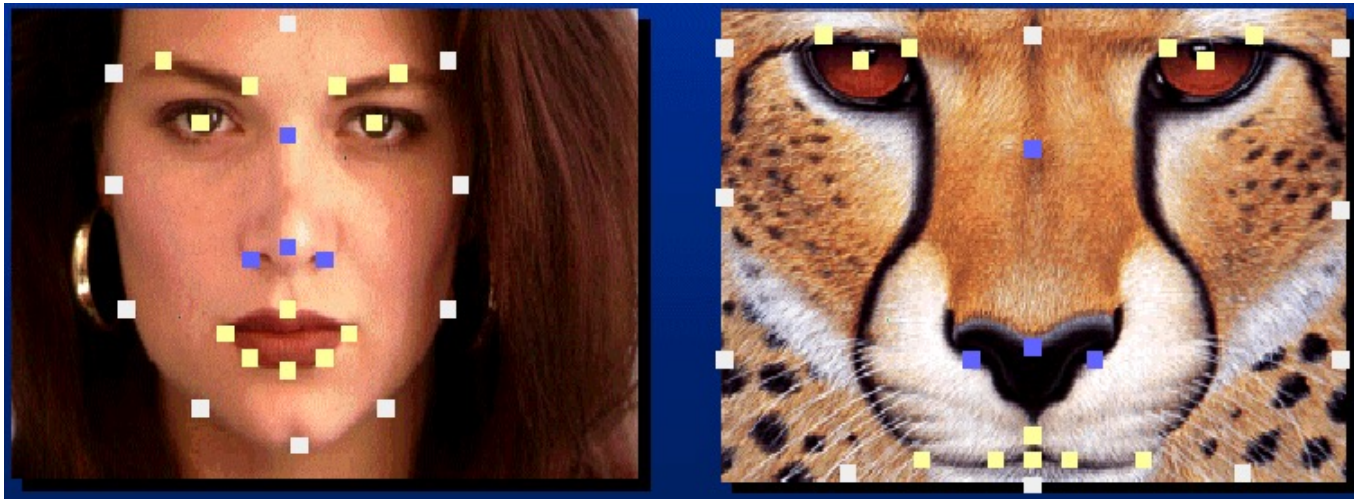
How do we go from feature points to pixels?

Triangular Mesh
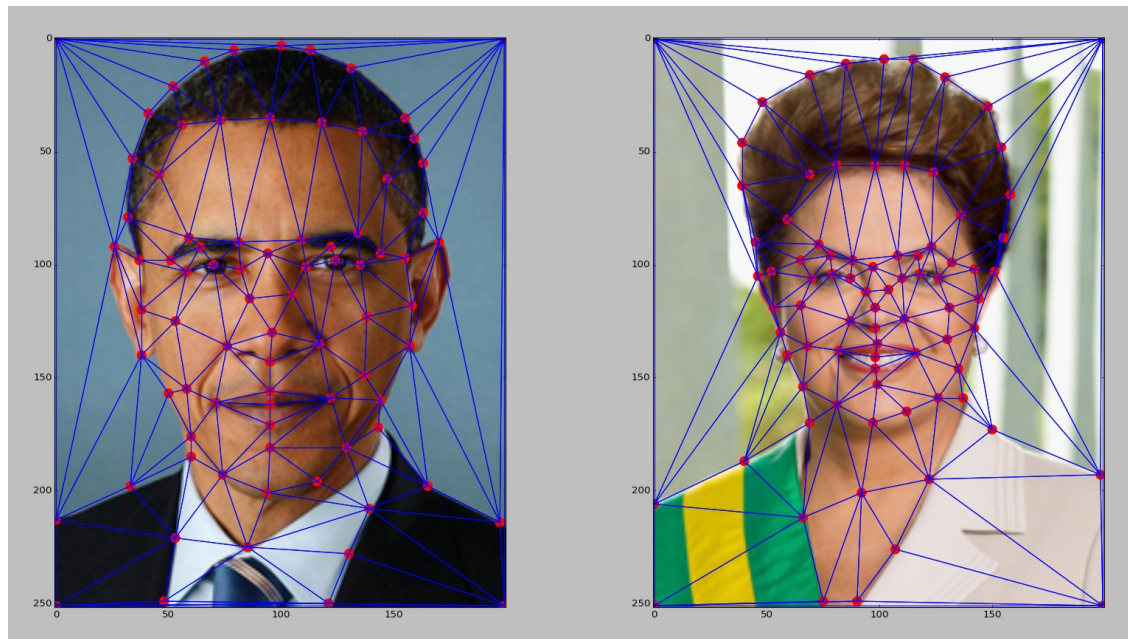
1. Input correspondences at key feature points

2. Define a triangular mesh over the points

   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences

3. Warp each triangle separately from source to destination

   - How do we warp a triangle?

(c) Ian Albuquerque Raymundo da Silva

Warping triangles

Source   Destination
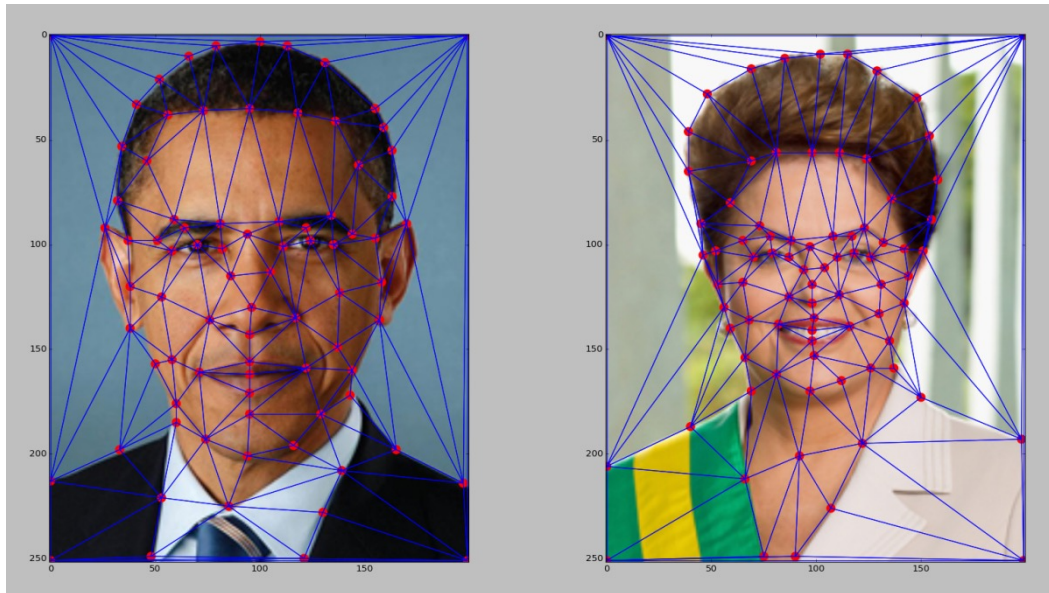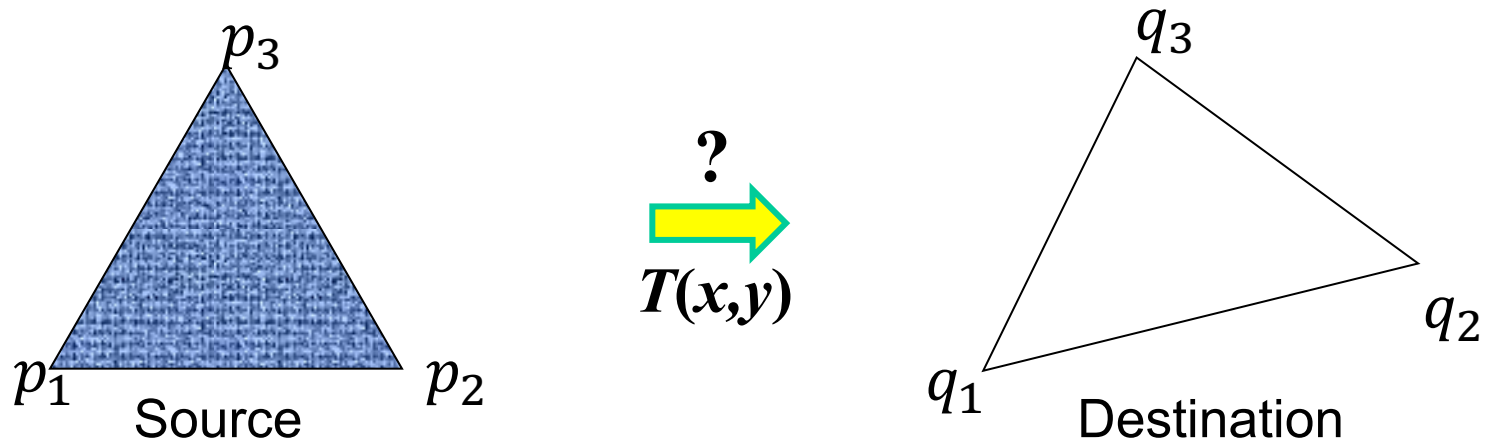
Given two triangles: $p_1 p_2 p_3$ and $q_1 q_2 q_3$ in 2D (12 numbers)

Need to find transform T to transfer all pixels from one to the other.

What kind of transformation is T?

How can we compute the transformation matrix:

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

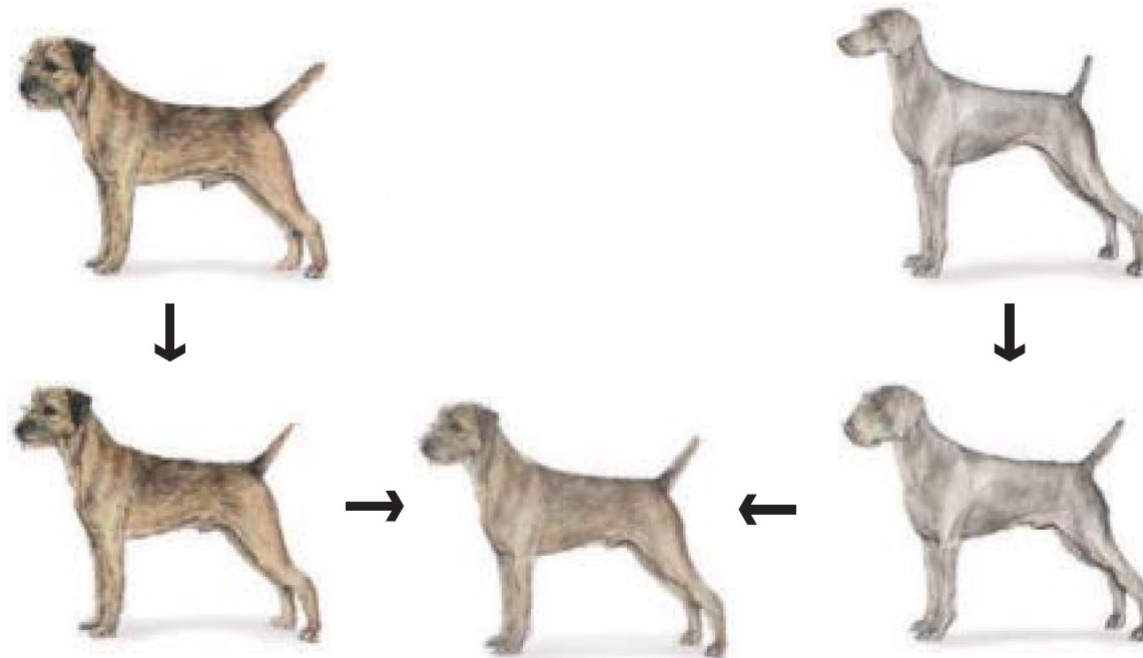$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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.
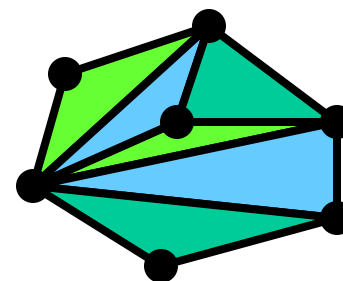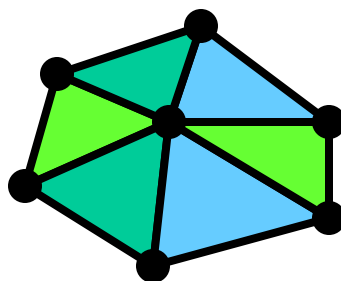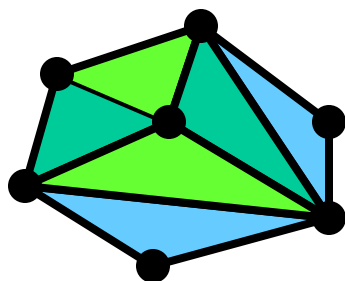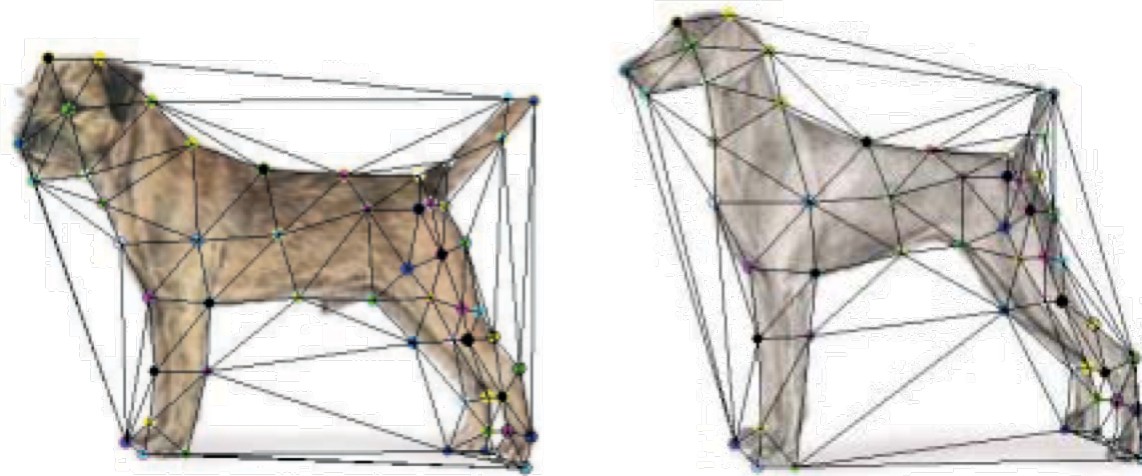
Full Morphing Procedure

Morphing procedure:

for every t,

1. Find the average shape (the "mean dog"☺)
   - local warping
2. Find the average color
   - Cross-dissolve the warped images

1. Create Average Shape

How do we create an intermediate warp at time t?

- Assume t = [0,1]
- Simple linear interpolation of each feature pair
    - $p \rightarrow q$
    - $(1-t) \cdot p + t \cdot q$ for corresponding features p and p'

2. Create Average Color

Interpolate whole images:

$$\text{Image}_{halfway} = (1-t)*\text{Image} + t*\text{image}'$$

cross-dissolve!

Morphing & matting

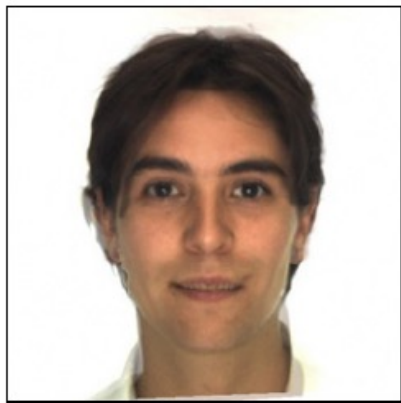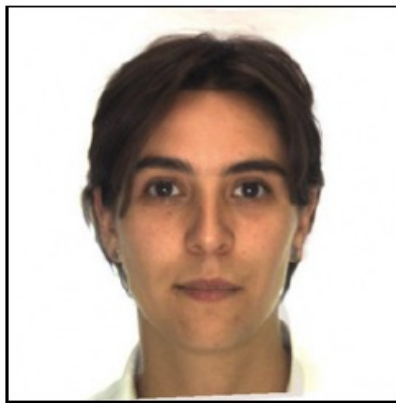Extract foreground first to avoid artifacts in the background
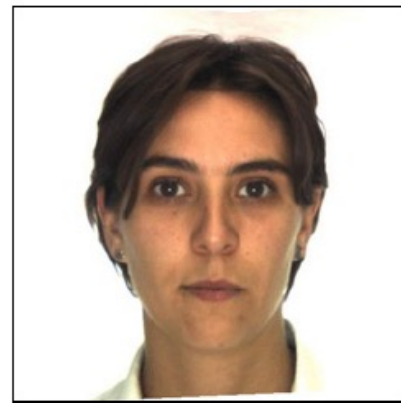

(c) $\alpha = 0.0$ (d) $\alpha = 0.2$ (e) $\alpha = 0.4$

(f) $\alpha = 0.6$ (g) $\alpha = 0.8$ (h) $\alpha = 1.0$

Slide by Durand and Freeman

66
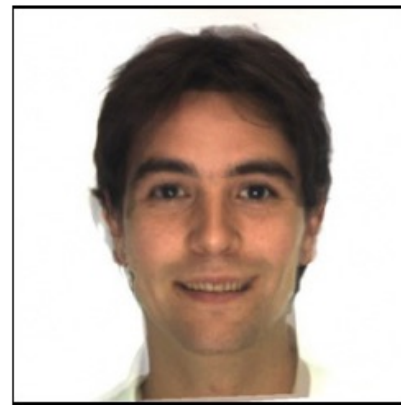
Amuse-bouche

By Philip Scott Johnson
Music: Bach's Sarabande from Suite for Solo Cello No. 1 in G Major, BWV 1007 performed by Yo-Yo Ma

Moving Least Square

What is a good local warping function $T(p) \rightarrow q$?

- Interpolation: need to satisfy control points $T(p_i) = q_i$
- Smoothness: T should be smooth;
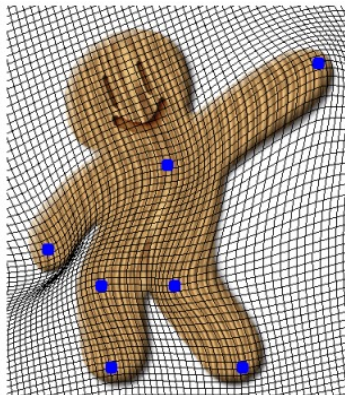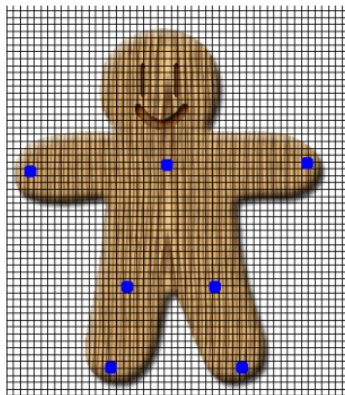- Identity: if $p_i = q_i$, T should be an identity mapping

Triangulation-based methods:

$$argmin_T \left\| T(p_i) - q_i \right\|^2 \text{ for 3 vertices in each triangle}$$

Moving least squares:

$$argmin_T w_i \left\| T(p_i) - q_i \right\|^2 \text{ for all the control points}$$
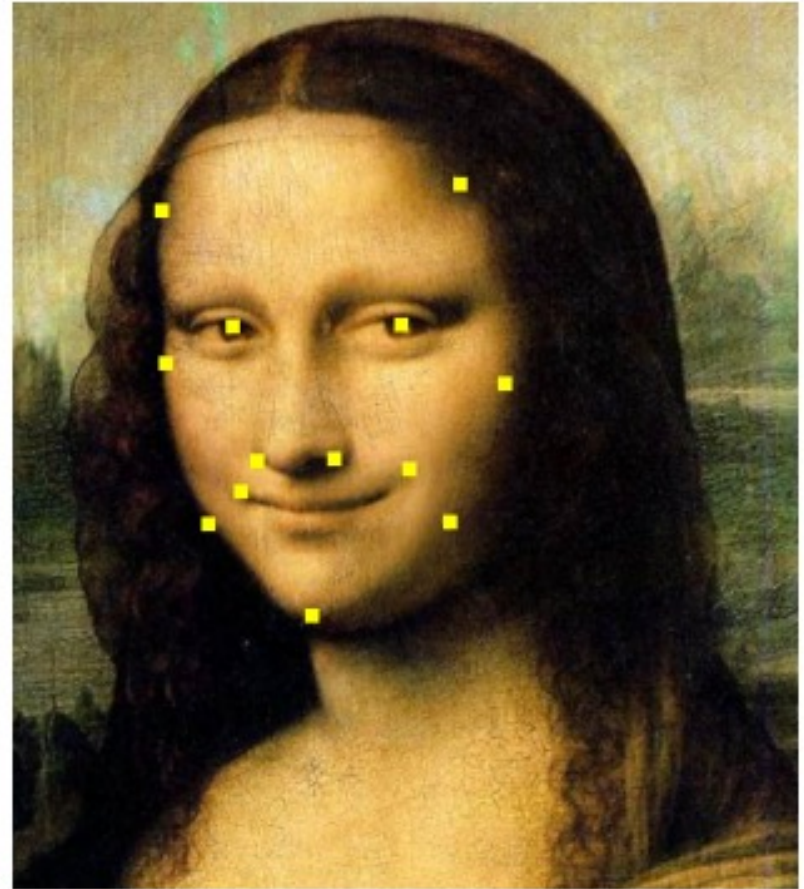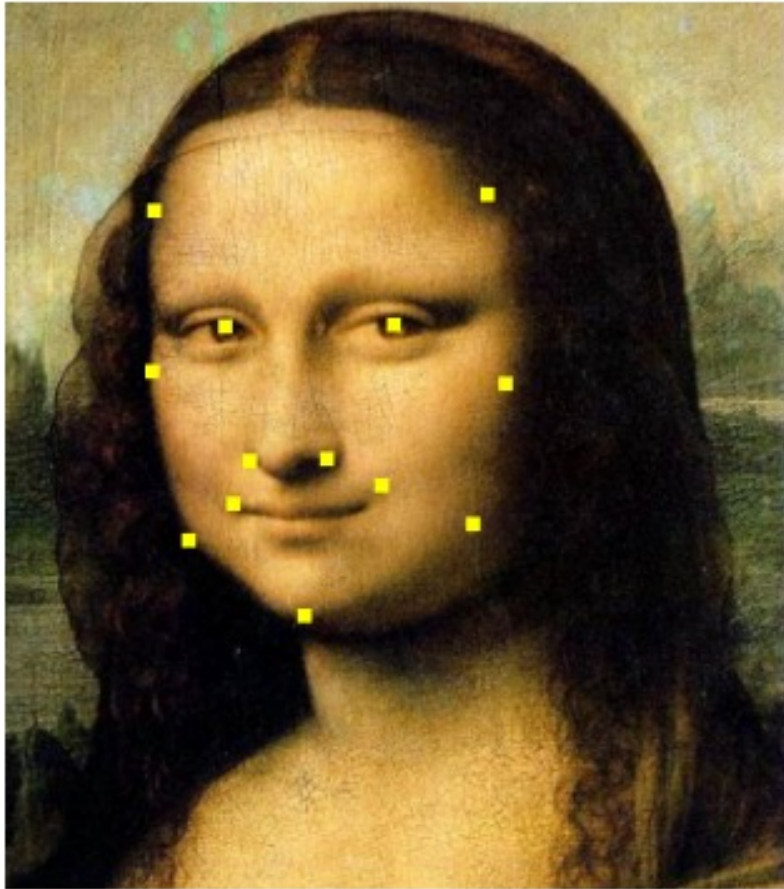
$$\text{Where } w_i = \frac{1}{|p_i - v|^{2\alpha}}$$
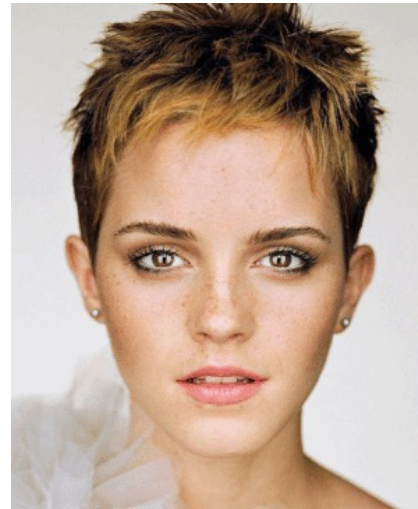
Moving Least Square
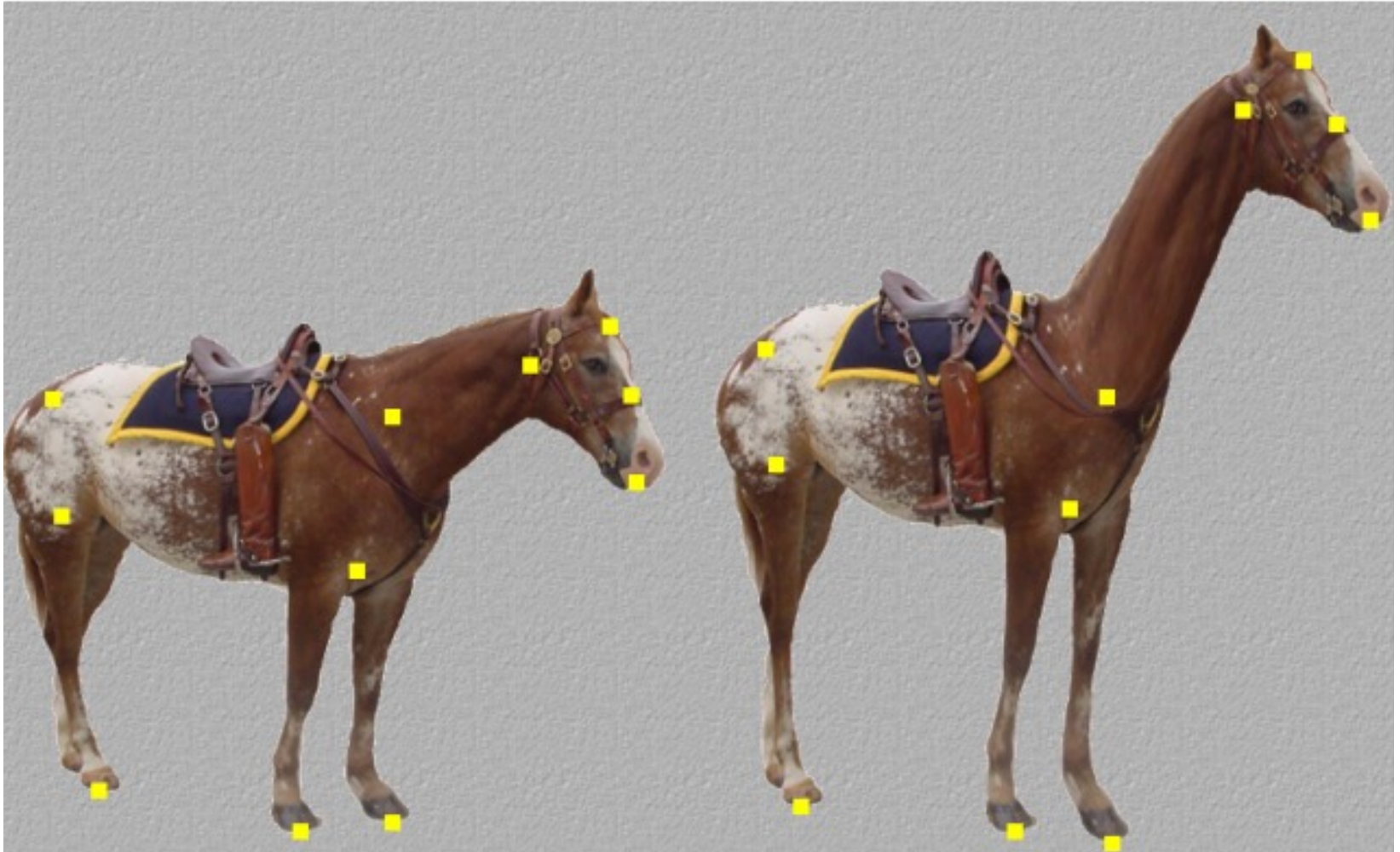
Moving Least Square

Thank You!

© Rachel Albert

16-726, Spring 2022

https://learning-image-synthesis.github.io/