

# Global and Local Image Warping

## Jun-Yan Zhu

16-726 Learning-based Image Synthesis, Spring 2025

# Logistics

---

## 1. Homework 0 released

### Assignment #0 - How to submit assignments?

Released on Monday 01/23/2023

**Due Date: Monday 01/30/2023 23:59**

**Download:**

Late Policy

- You have free 5 late days.
- You can use late days for assignments. A late day extends the deadline 24 hours.
- Once you have used all 5 late days, the penalty is 10% for each additional late day.

### 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:
```

# Image Transformations

---

image filtering: change **range** of image

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

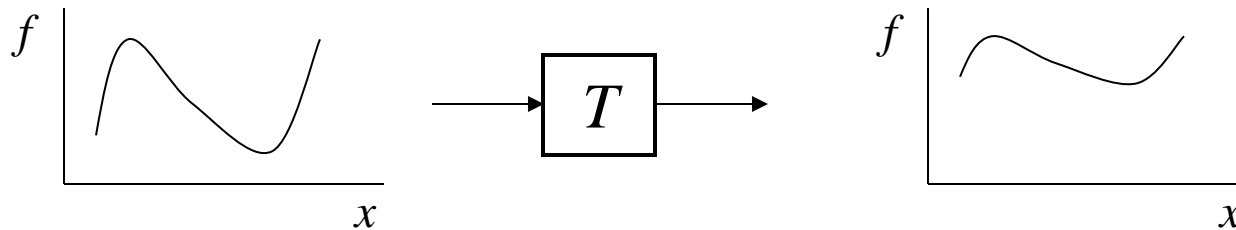
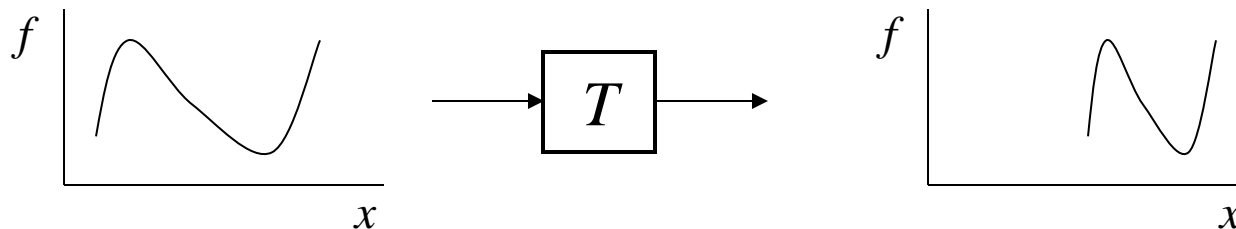


image warping: change **domain** of image

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



# Image Transformations

---

image filtering: change **range** of image

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

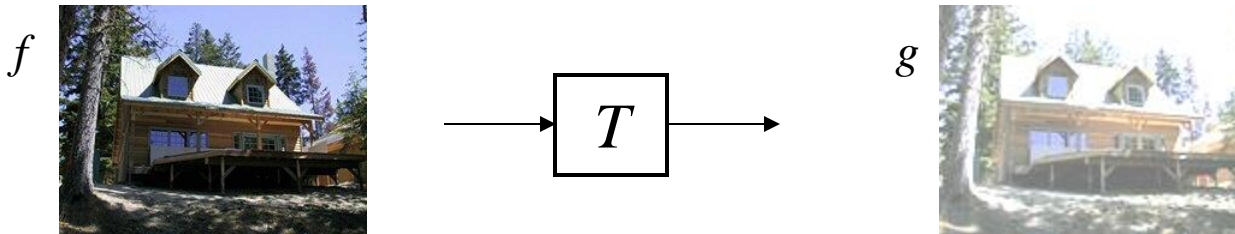
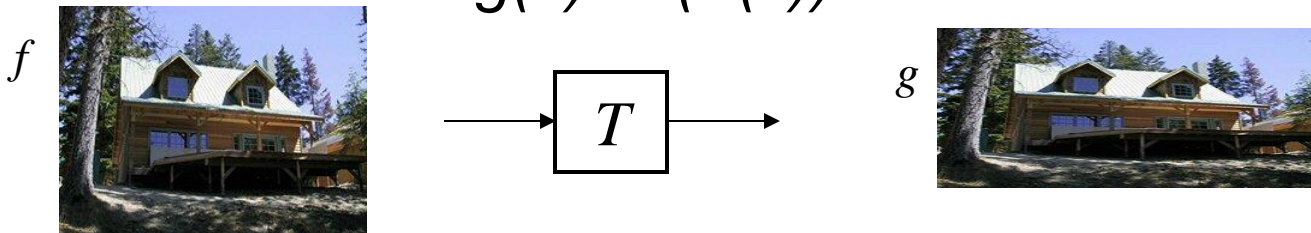


image warping: change **domain** of image

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



# Parametric (global) warping

---

Examples of parametric warps:



translation



rotation



aspect



affine



perspective



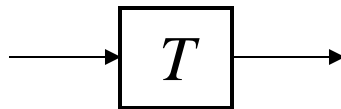
cylindrical

# 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 a linear  $T$  as a matrix:

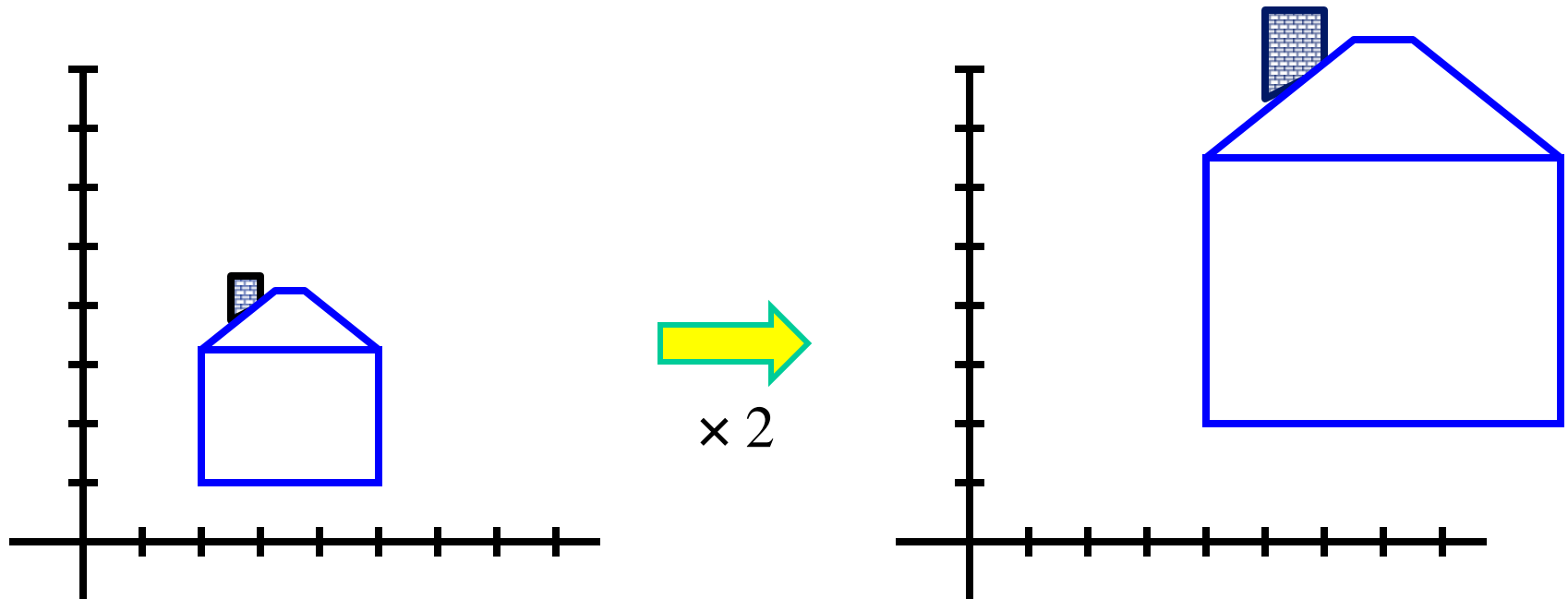
$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

# Scaling

---

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

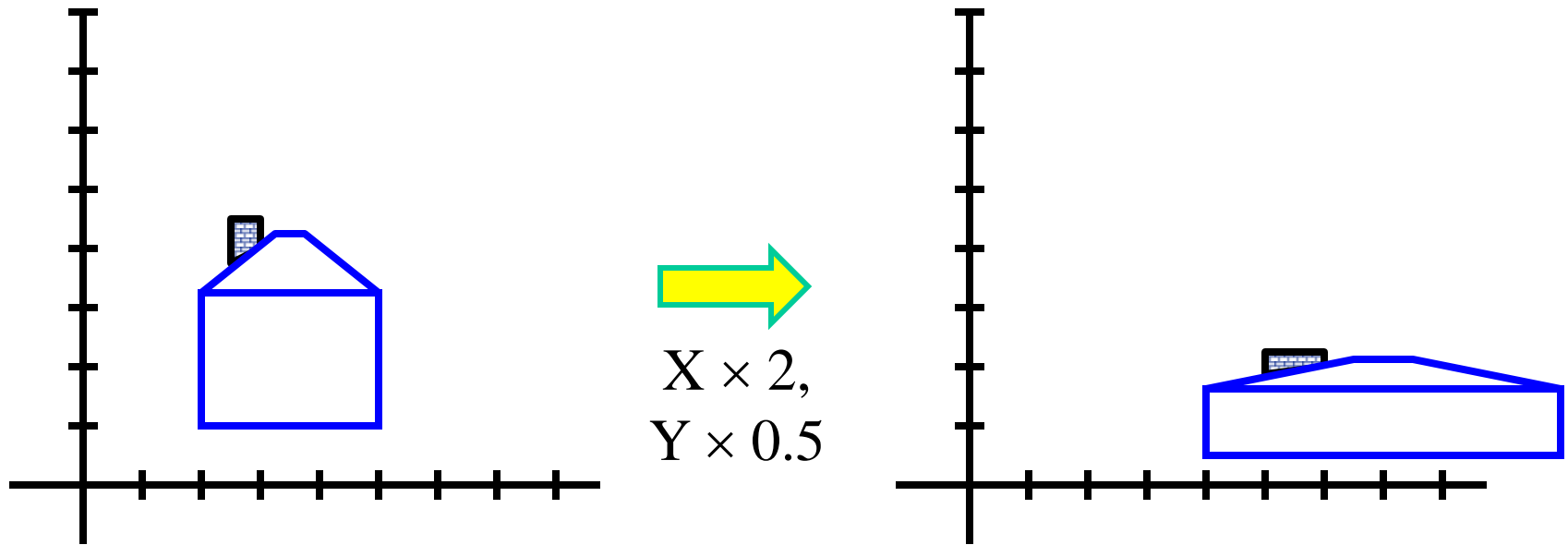
*Uniform scaling* means this scalar is the same for all components:



# Scaling

---

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



# Scaling

---

Scaling operation:

$$x' = ax$$

$$y' = by$$

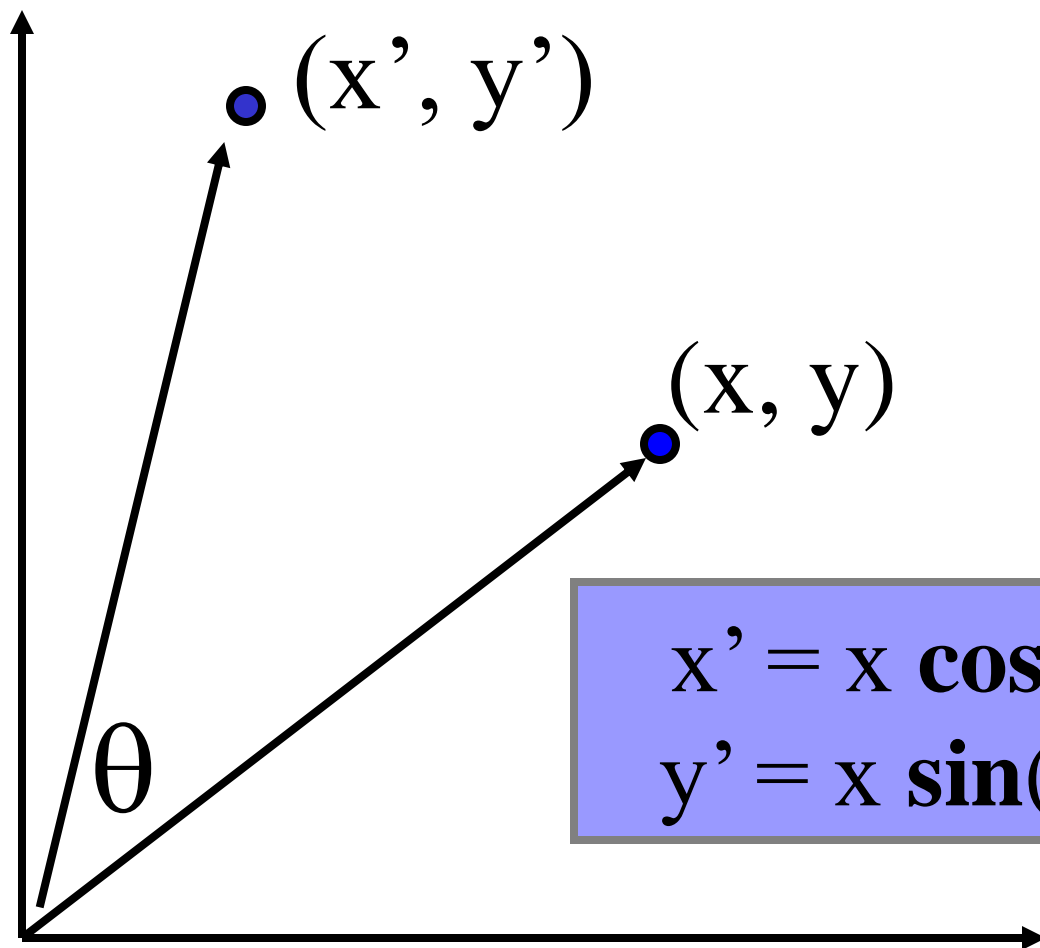
Or, in matrix form:

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

What's inverse of S?

# 2-D Rotation

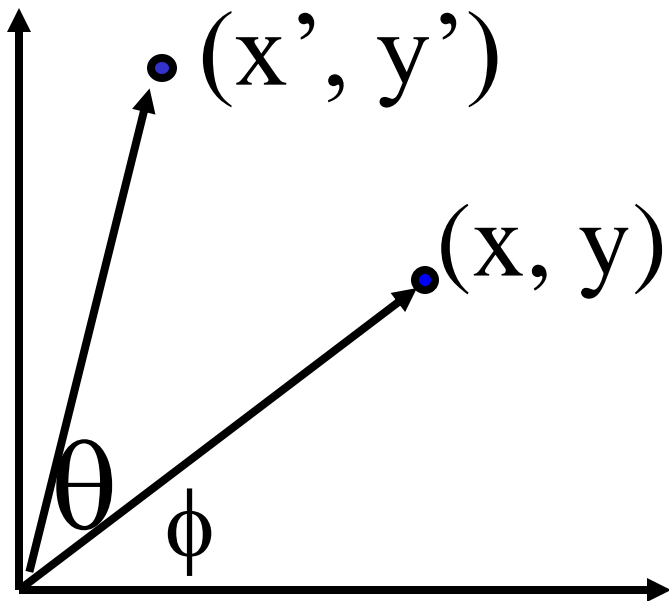
---



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

# 2-D Rotation

---



$$x = r \cos (\phi)$$

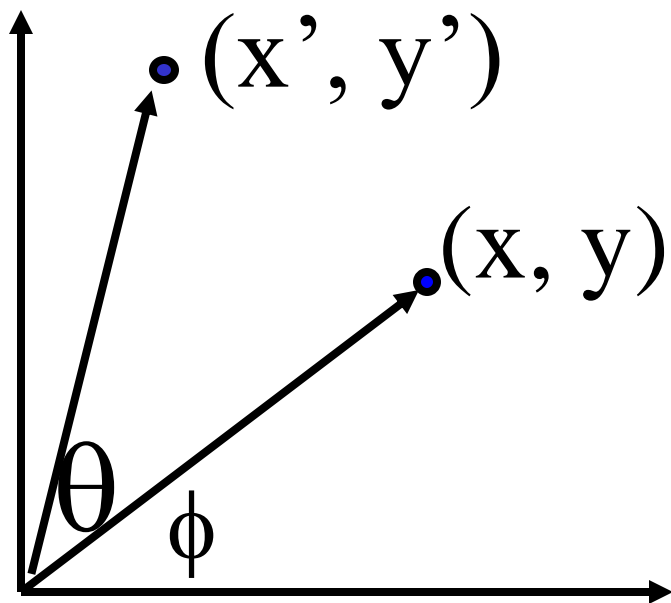
$$y = r \sin (\phi)$$

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

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

# 2-D Rotation

---



$$x = r \cos(\phi)$$

$$y = r \sin(\phi)$$

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

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

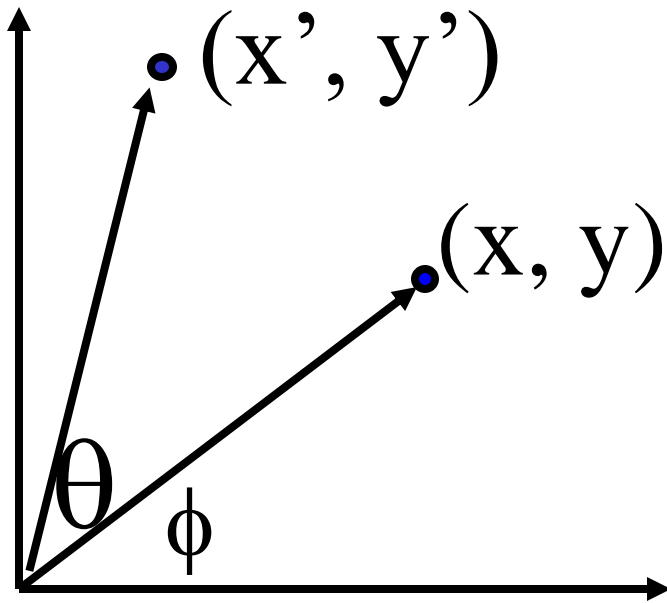
Trig Identity...

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

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

# 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...

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

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

# 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  $\theta$ ,

- ***$x'$  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?

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

2D Scale around (0,0)?

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

# 2x2 Matrices

---

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

2D Rotate around (0,0)?

$$\begin{aligned}x' &= \cos \Theta * x - \sin \Theta * y \\y' &= \sin \Theta * x + \cos \Theta * y\end{aligned}$$

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

2D Shear?

$$\begin{aligned}x' &= x + sh_x * y \\y' &= sh_y * x + y\end{aligned}$$

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

# 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?

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

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

# All 2D Linear Transformations

---

Linear transformations are combinations of ...

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

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

Properties of linear transformations:

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

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

# 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

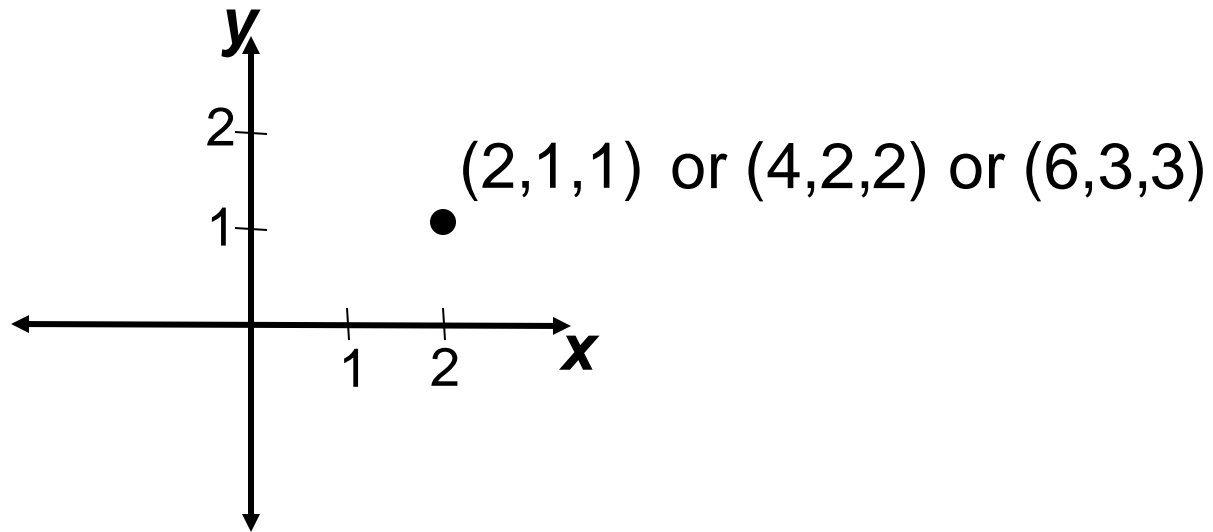
$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# 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



Convenient  
coordinate system to  
represent many  
useful  
transformations

# 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:**

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

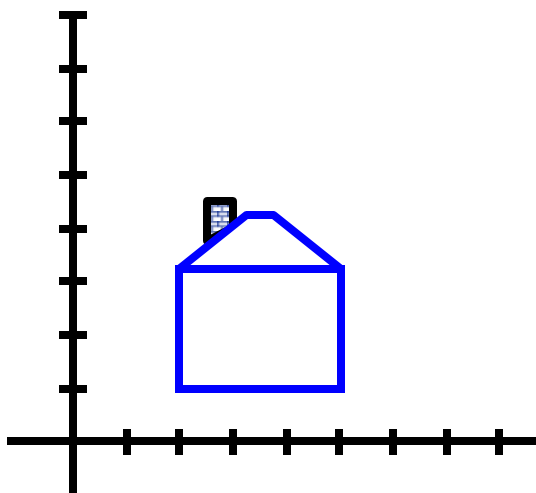
# Translation

Example of translation

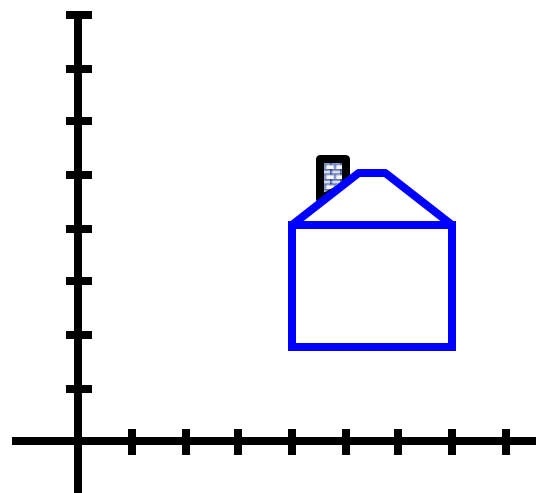
Homogeneous Coordinates



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



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



# 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

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

**p'** = **T**(t<sub>x</sub>,t<sub>y</sub>) **R**( $\Theta$ ) **S**(s<sub>x</sub>,s<sub>y</sub>) **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 ...

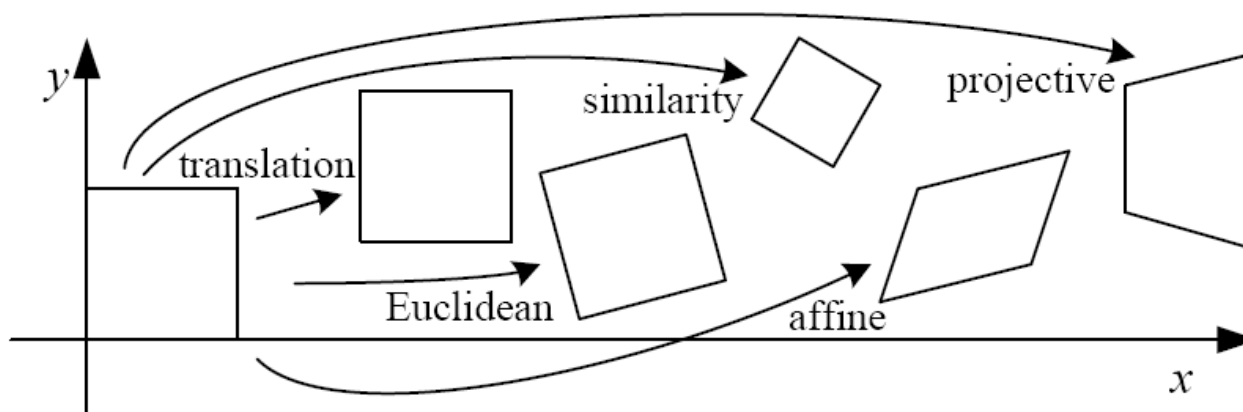
- Affine transformations, and
- Projective warps

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

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

# 2D image transformations



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

These transformations are a nested set of groups

- Closed under composition and inverse is a member



# Image Warping in Biology

D'Arcy Thompson

<http://www-groups.dcs.st-and.ac.uk/~history/Miscellaneous/darcy.html>

[http://en.wikipedia.org/wiki/D'Arcy\\_Thompson](http://en.wikipedia.org/wiki/D'Arcy_Thompson)

Importance of shape and structure in evolution

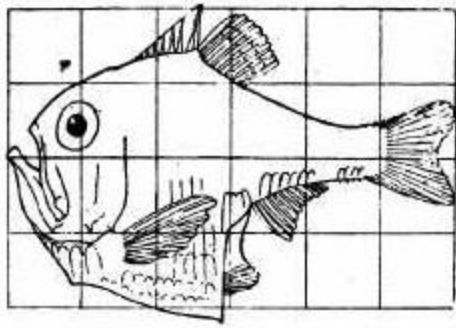
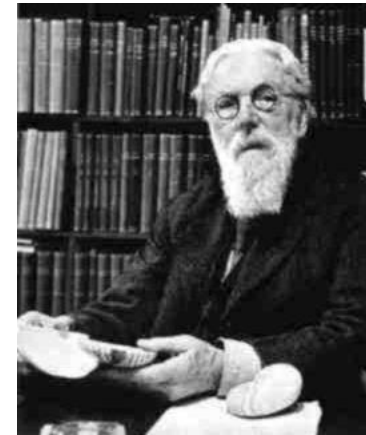
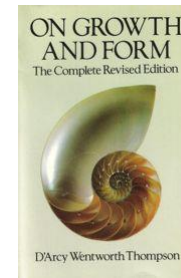


Fig. 517. *Argyropelecus Olfersi*.

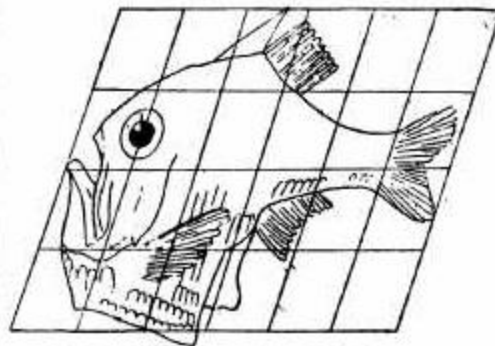
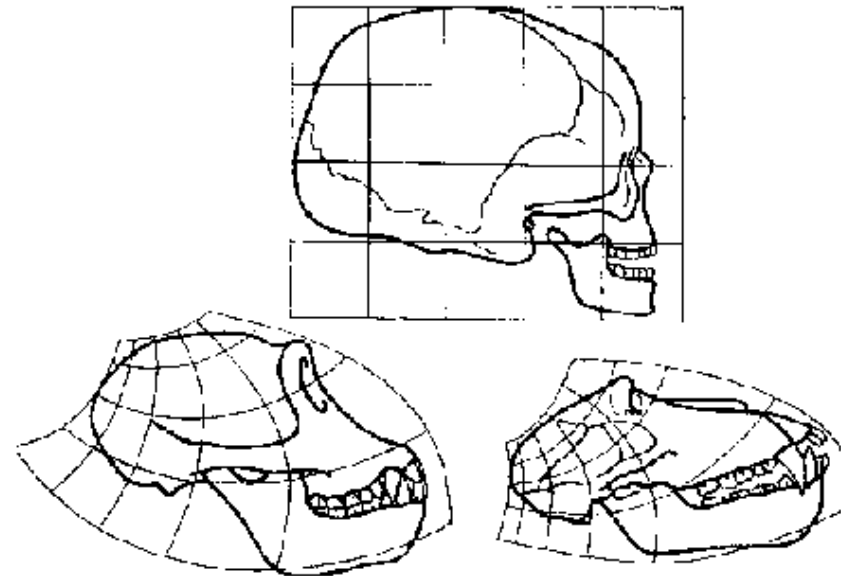


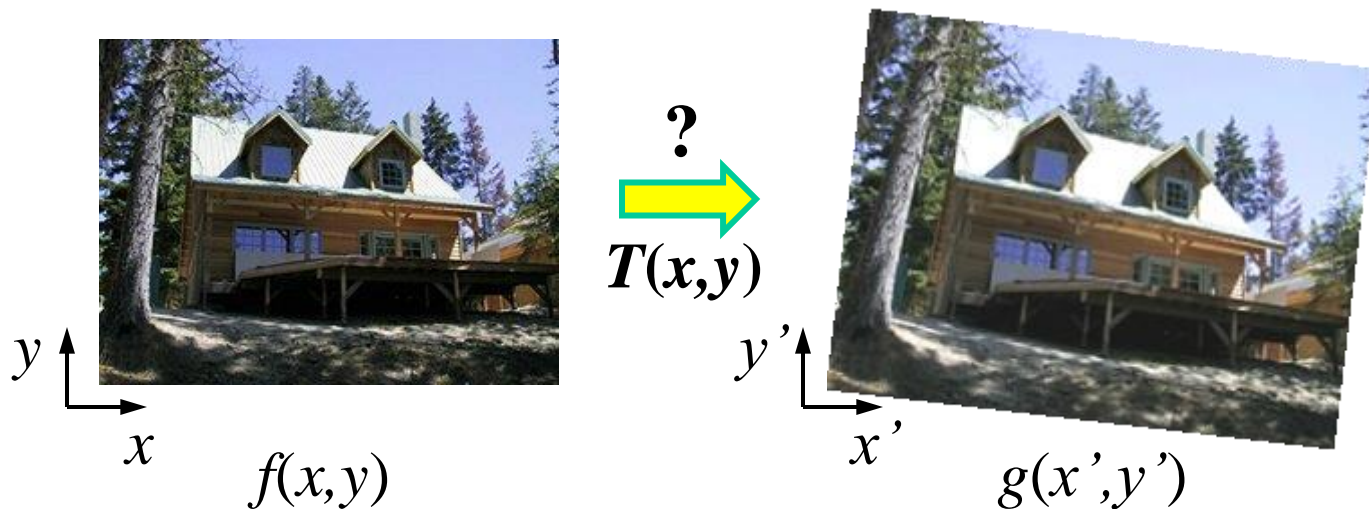
Fig. 518. *Sternoptyx diaphana*.



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

# Recovering Transformations

---

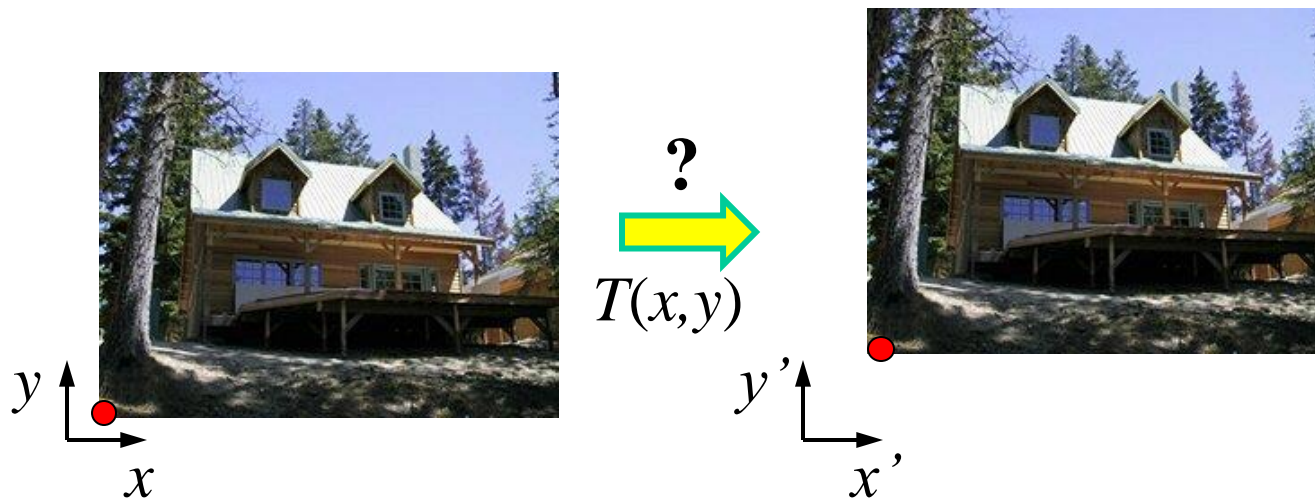


What if we know  $f$  and  $g$  and want to recover the transform  $T$ ?

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

# Translation: # correspondences?

---



How many correspondences needed for translation?

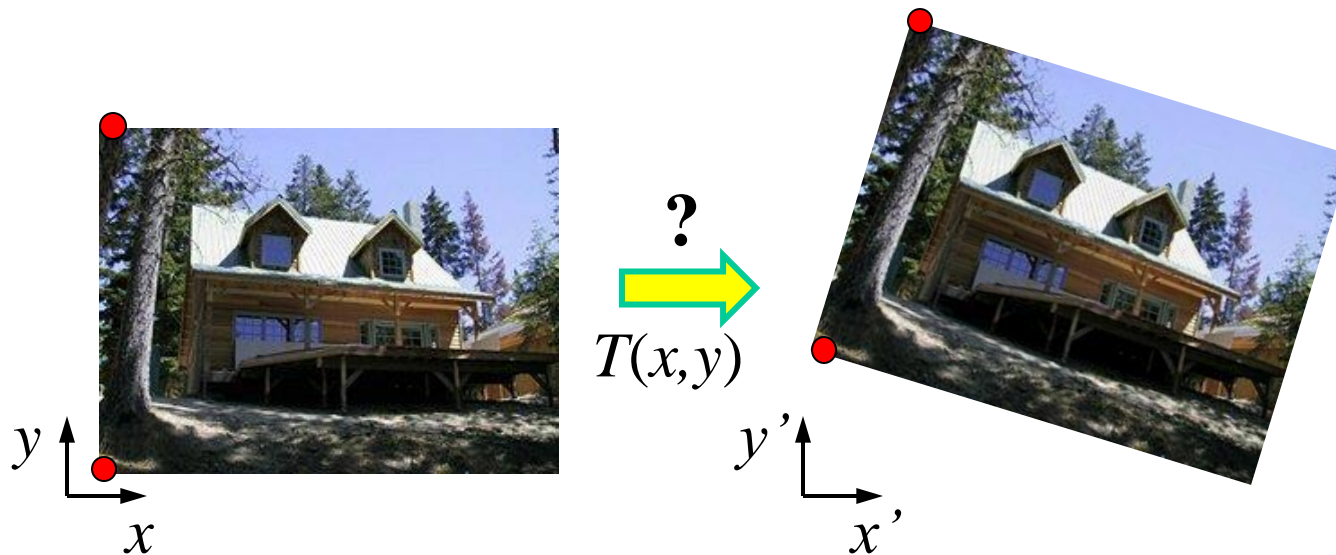
How many Degrees of Freedom?

What is the transformation matrix?

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

# Euclidian: # correspondences?

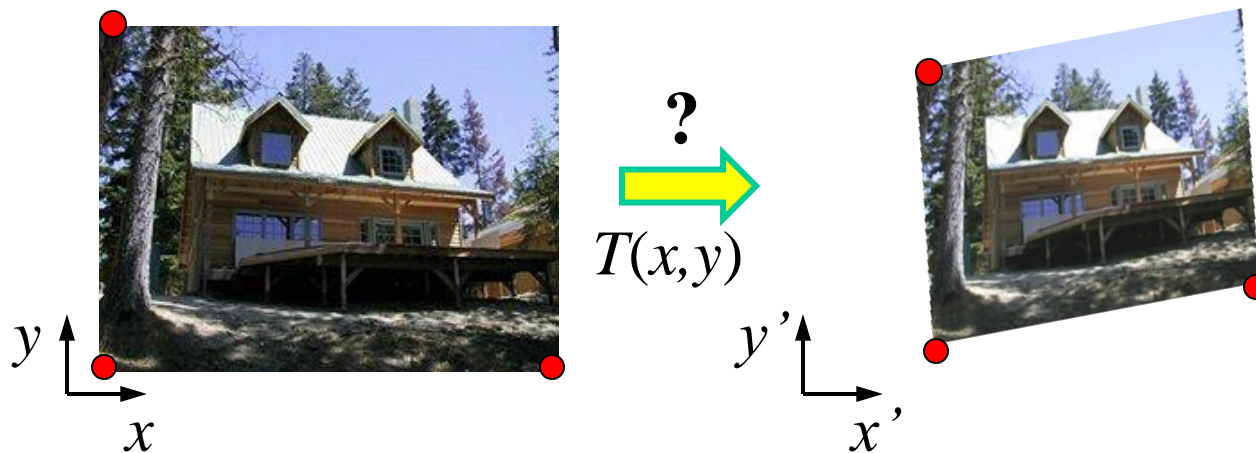
---



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

# Affine: # correspondences?

---

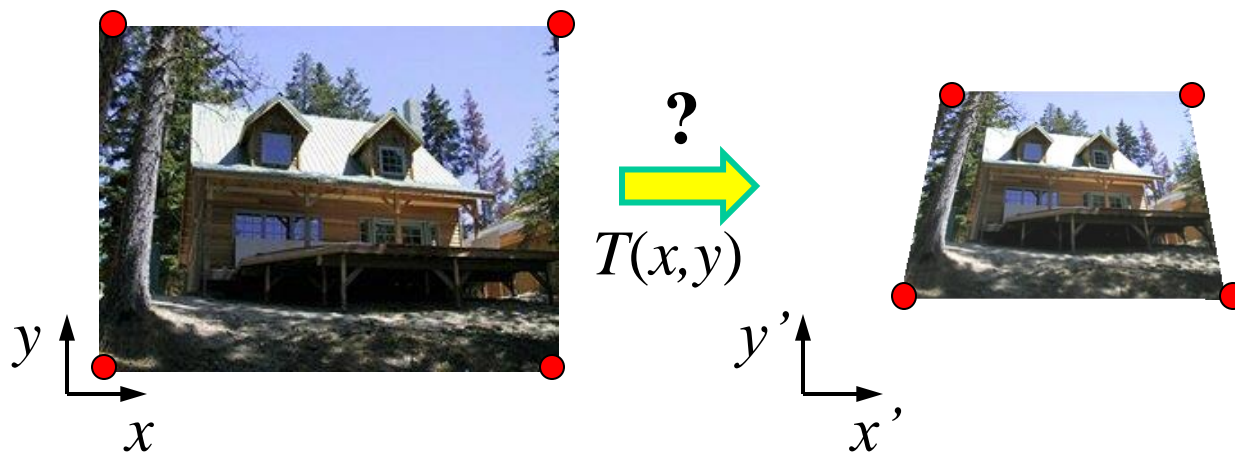


How many correspondences needed for affine?

How many DOF?

# Projective: # correspondences?

---

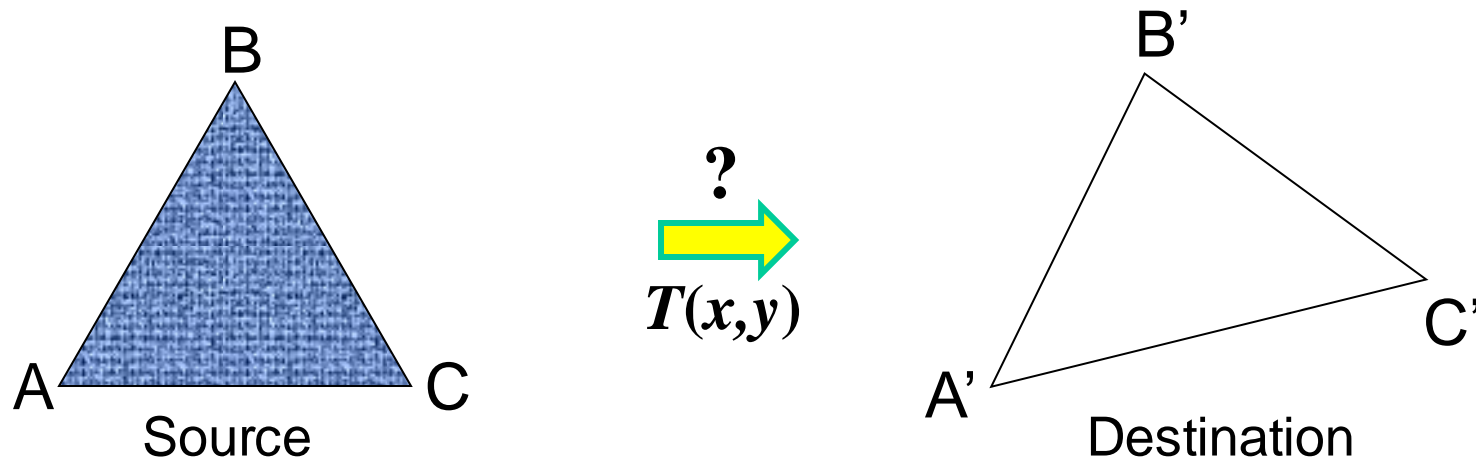


How many correspondences needed for projective?

How many DOF?

# Example: warping triangles

---



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.

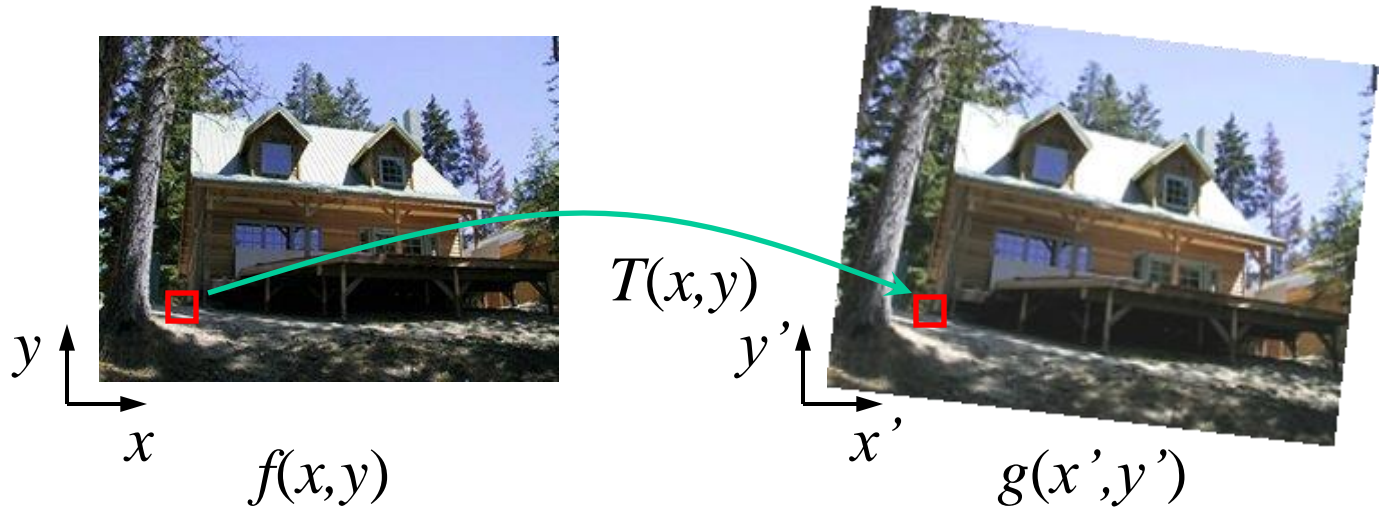
What kind of transformation is T?

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

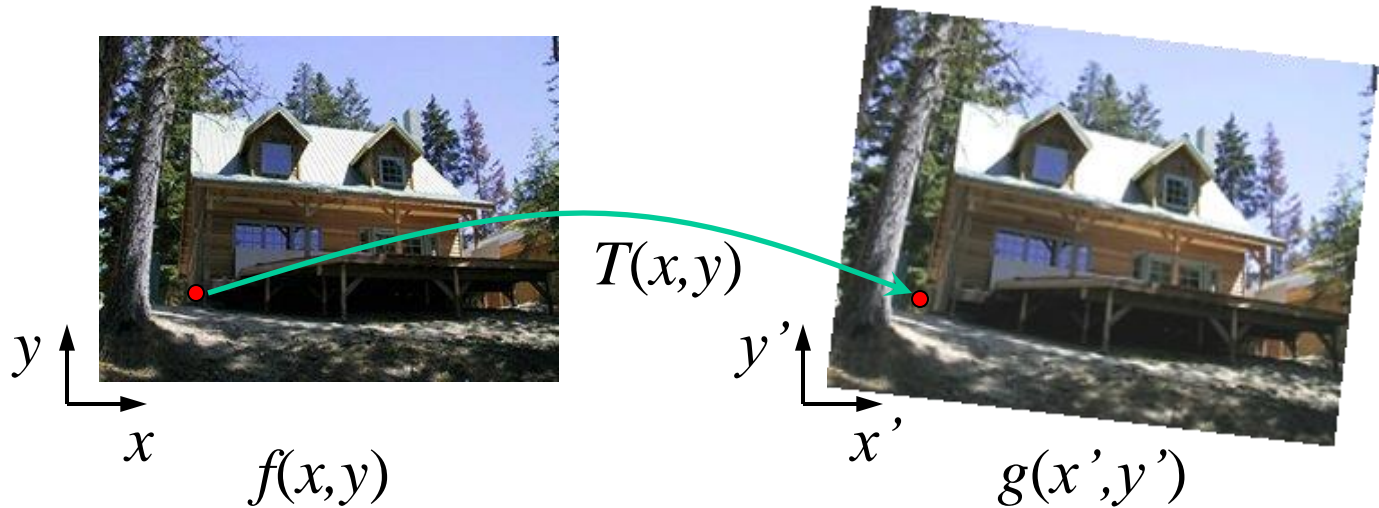
---



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

---

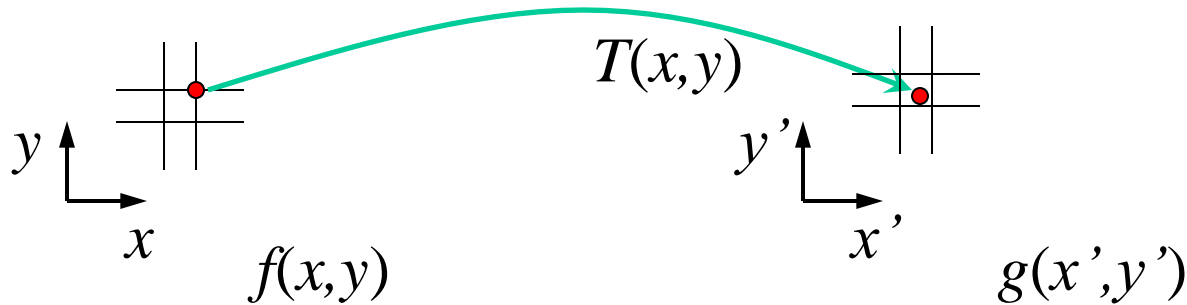


Send each pixel  $f(x,y)$  to its corresponding location  
 $(x',y') = T(x,y)$  in the second image

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

# Forward warping

---



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

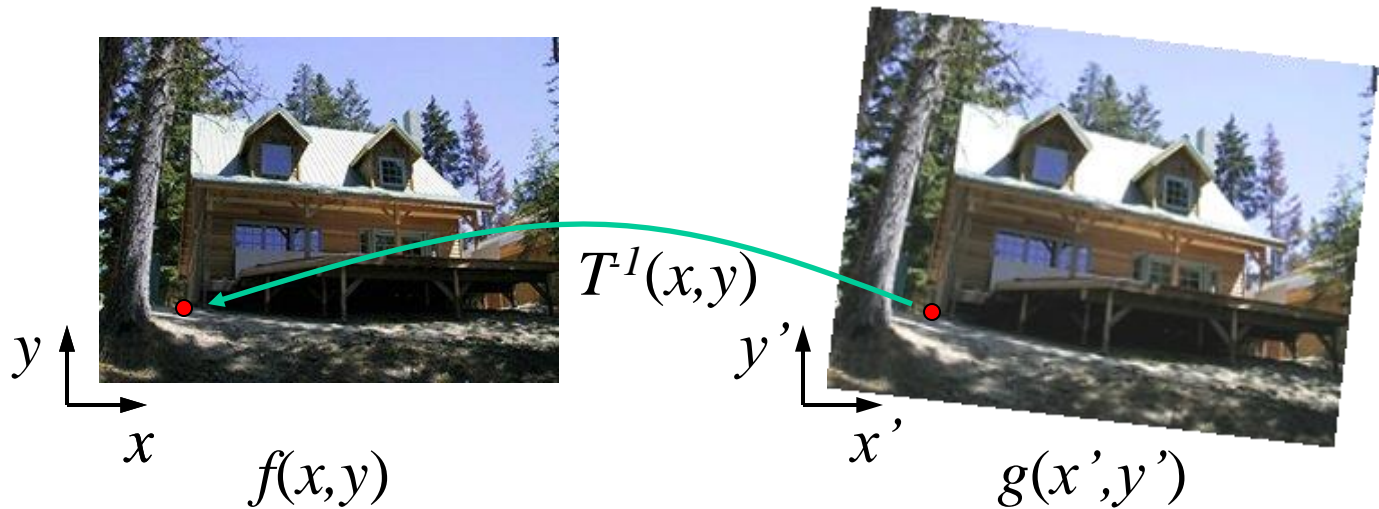
Q: what if pixel lands “between” two pixels?

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

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

# Inverse warping

---

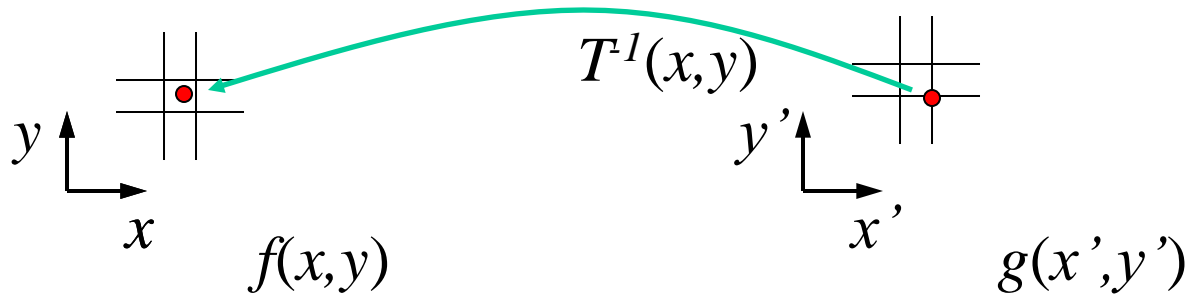


Get each pixel  $g(x',y')$  from its corresponding location  
 $(x,y) = T^{-1}(x',y')$  in the first image

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

# 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 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  $t$

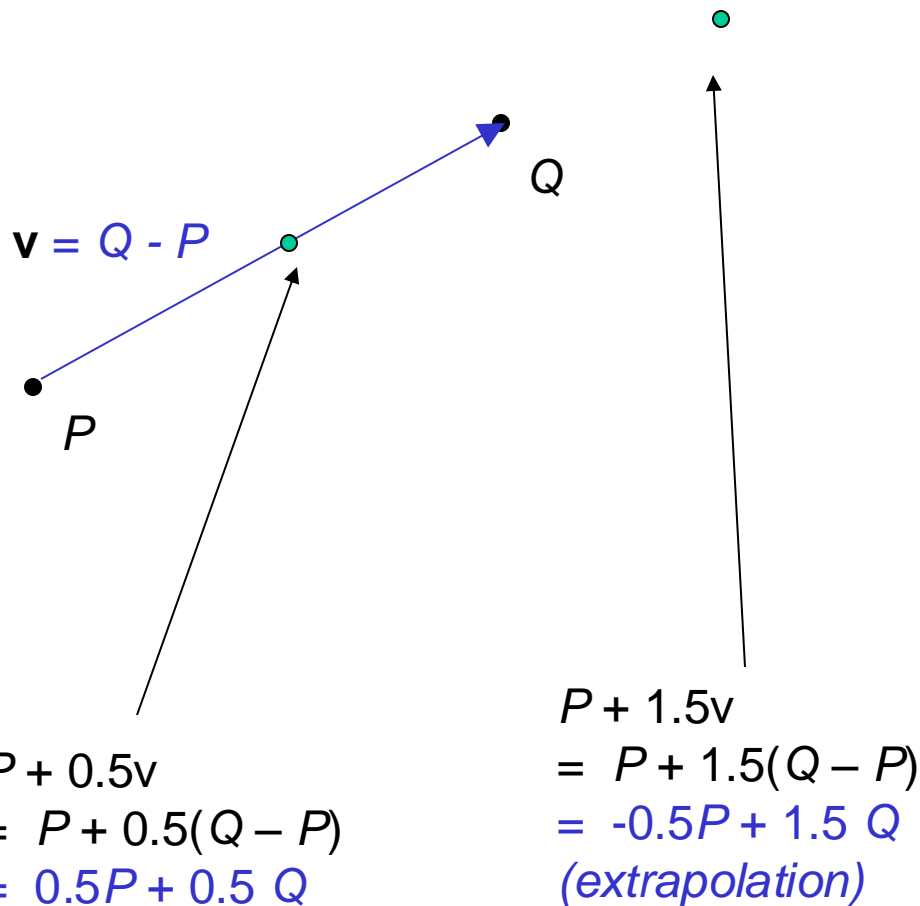
How do we know what the average object looks like?

- 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?



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:

$$\text{Image}_{\text{halfway}} = (1-t) \cdot \text{Image}_1 + t \cdot \text{Image}_2$$

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

# 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

---



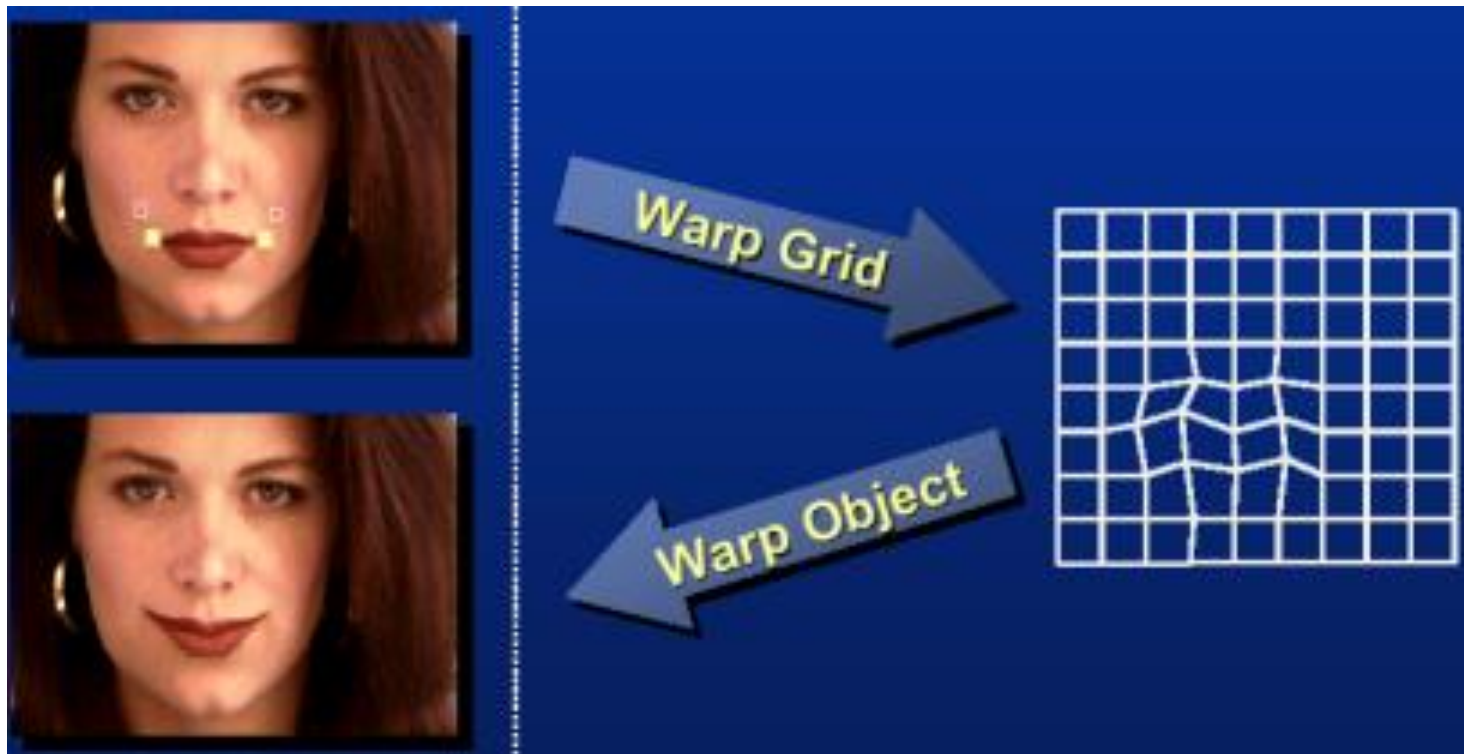
Need to specify a more detailed warp function

- 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

---

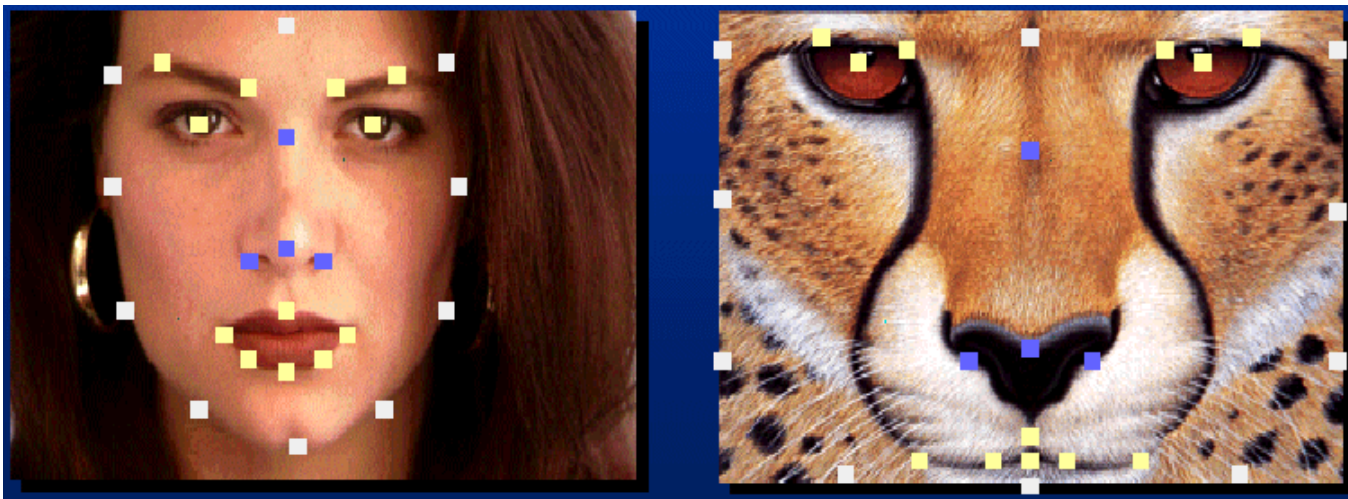
Define vector field to specify a dense warp



# Warp specification - sparse

---

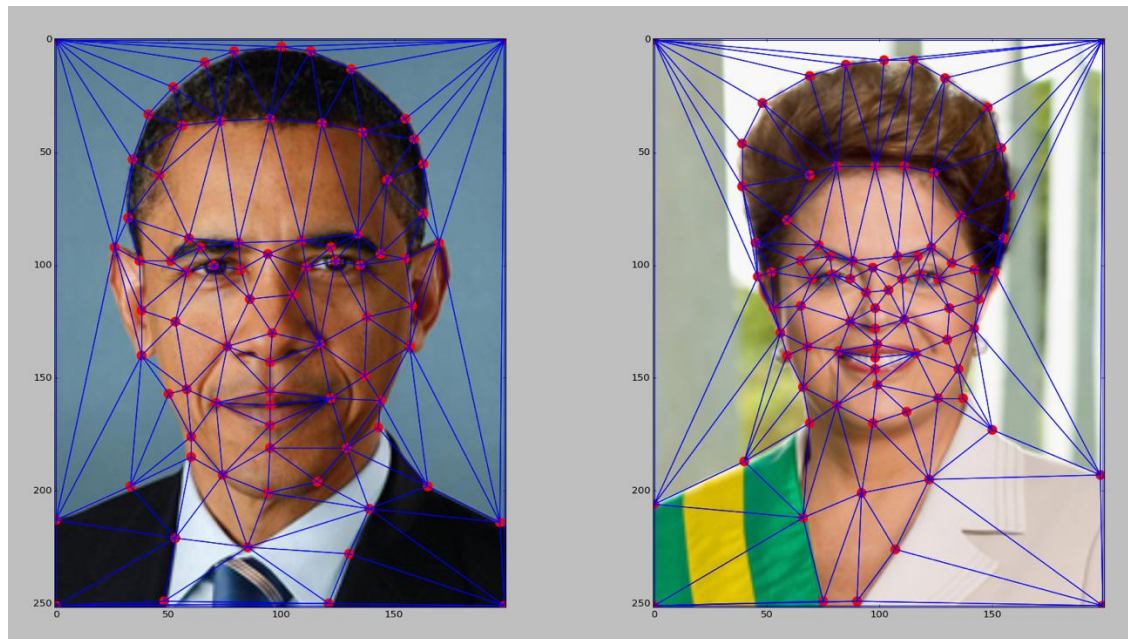
How can we specify a sparse warp?



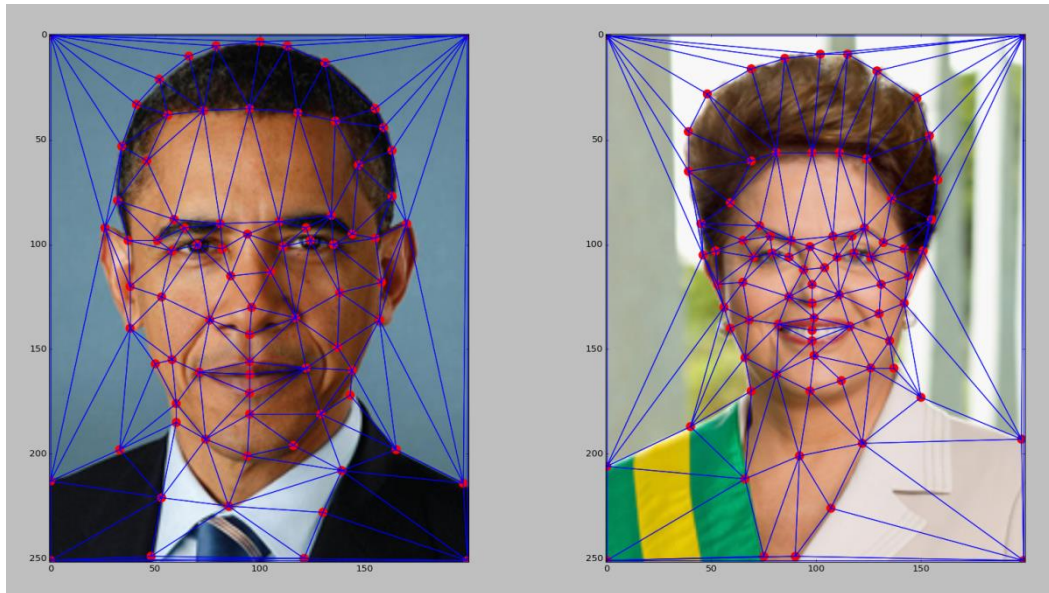
How do we go from feature points to pixels?

# Triangular Mesh

---

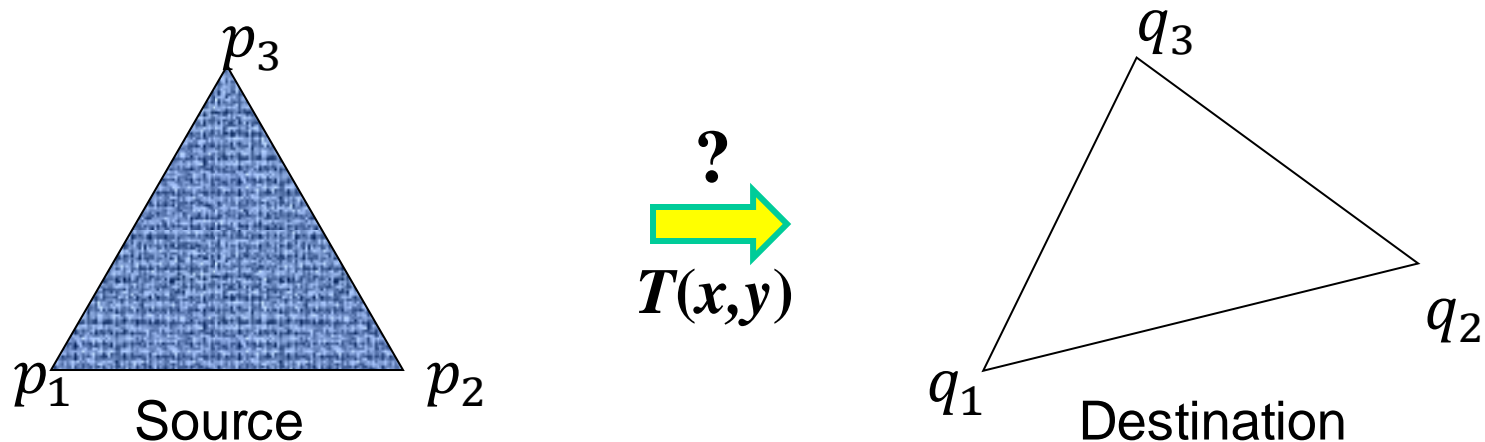


1. Input correspondences at key feature points
2. Define a triangular mesh over the points
  - Same mesh in both images!
  - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately from source to destination
  - How do we warp a triangle?



# Warping triangles

---



Given two triangles:  $p_1p_2p_3$  and  $q_1q_2q_3$  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:

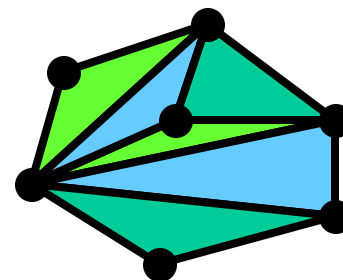
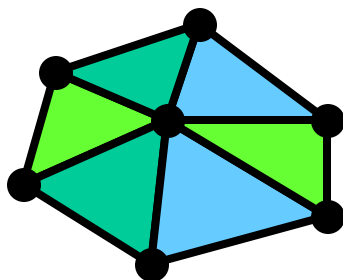
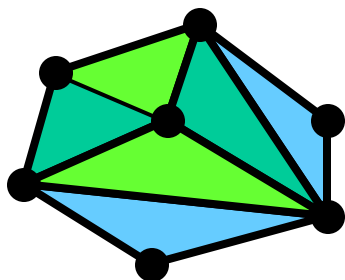
$$\begin{aligned} p &= (x, y) \\ q &= (x', y') \end{aligned} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Triangulations

---

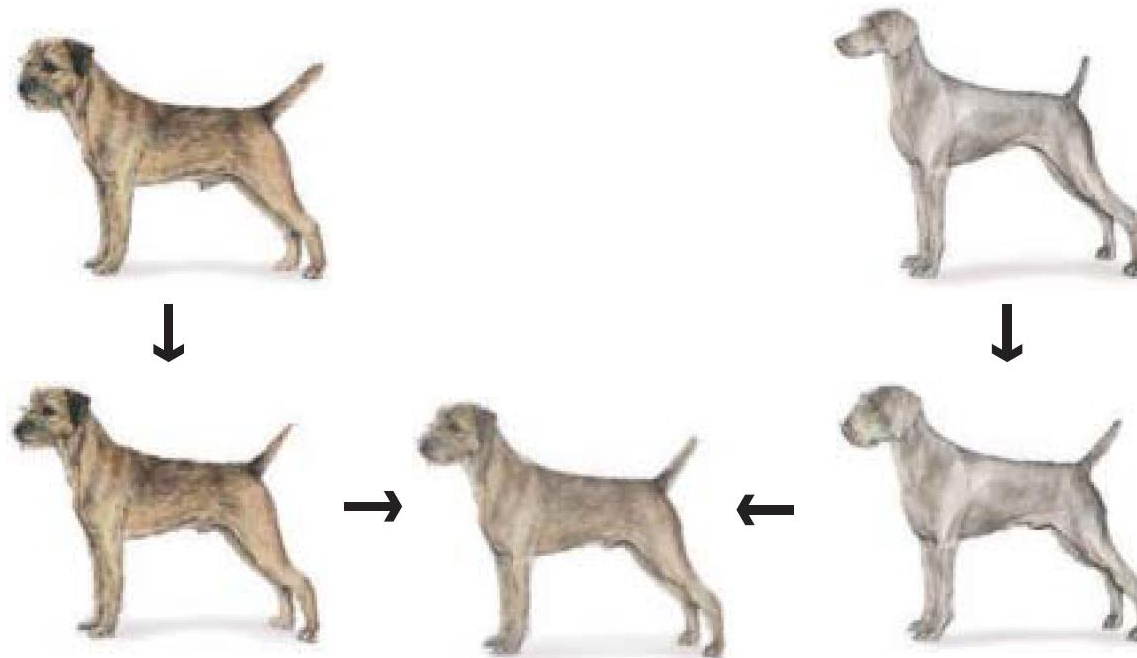
A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.

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



# Full Morphing Procedure

---



Morphing procedure:

*for every  $t$ ,*

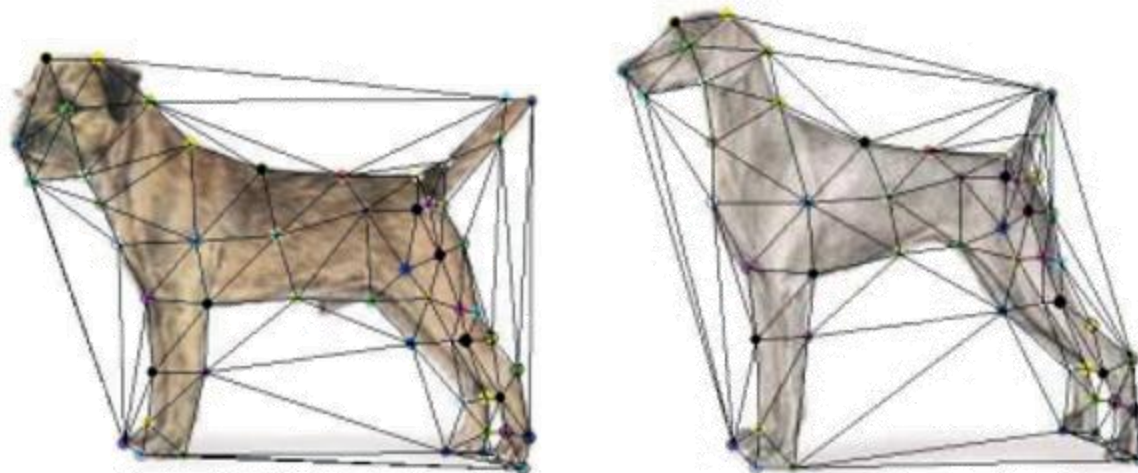
1. Find the average shape (the “mean dog” 😊)
  - local warping
2. Find the average color
  - Cross-dissolve the warped images

# 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:

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

**cross-dissolve!**

# Morphing & matting

---

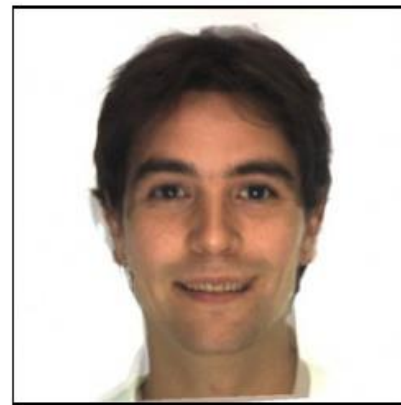
Extract foreground first to avoid artifacts in the background



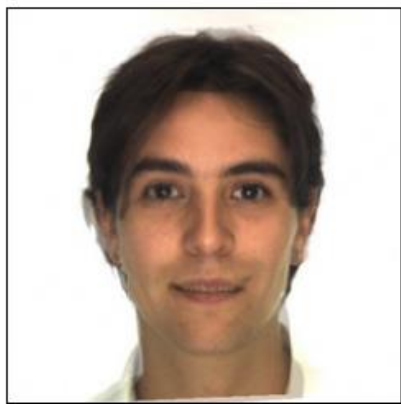
(c)  $\alpha = 0.0$



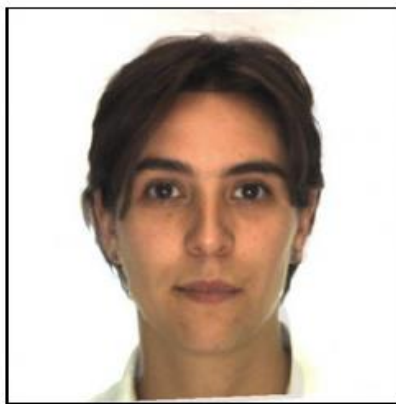
(d)  $\alpha = 0.2$



(e)  $\alpha = 0.4$



(f)  $\alpha = 0.6$



(g)  $\alpha = 0.8$



(h)  $\alpha = 1.0$

# Amuse-bouche

---



By Philip Scott Johnson

59

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

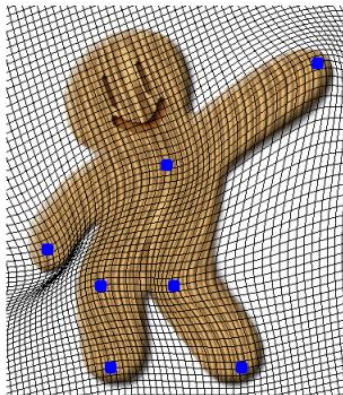
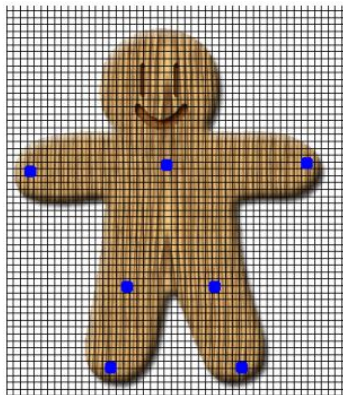
Triangulation-based methods:

$\operatorname{argmin}_T ||T(p_i) - q_i||^2$  for 3 vertices in each triangle

**Moving least squares:**

$\operatorname{argmin}_T w_i ||T(p_i) - q_i||^2$  for all the control points

Where  $w_i = \frac{1}{|p_i - v|^{2\alpha}}$



$v$ : current pixel

$\alpha$ : hyper-parameters

$p_i$ : source control points

$q_i$ : target control points

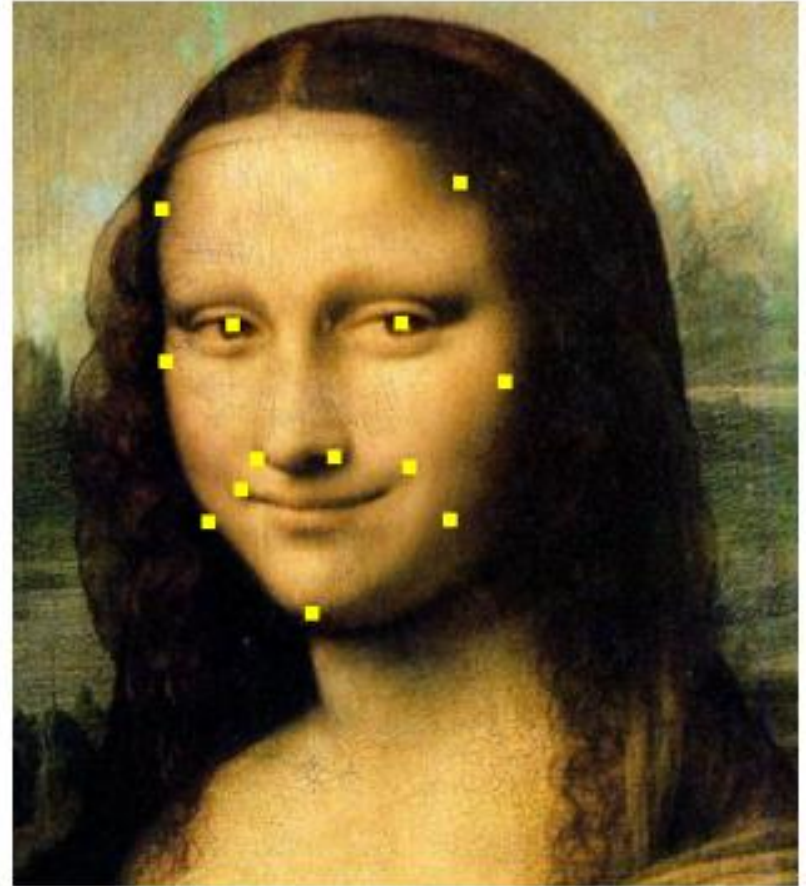
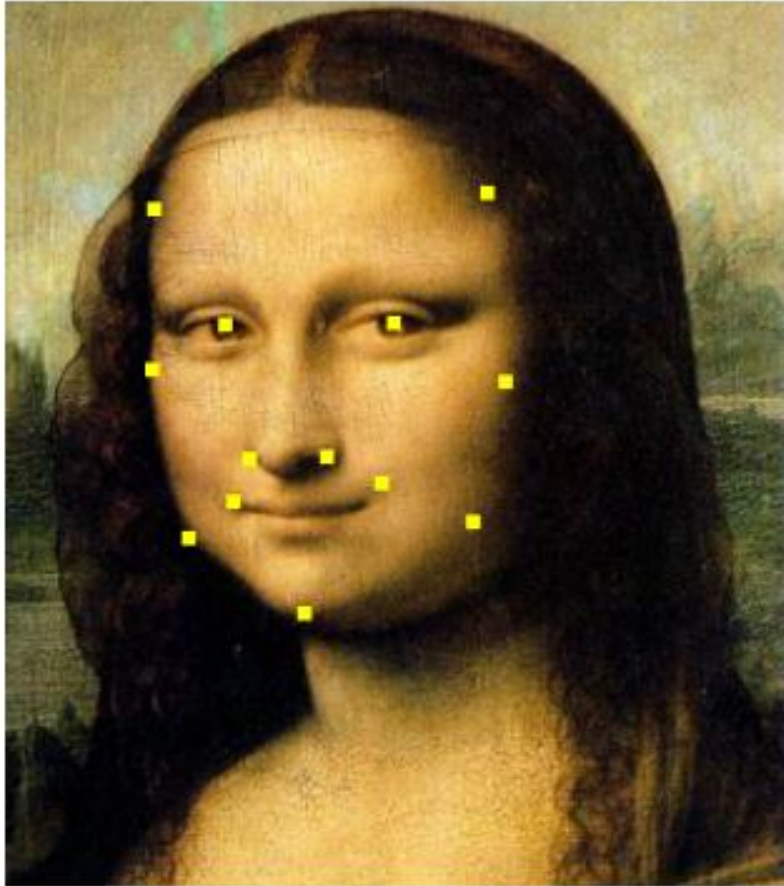
# Moving Least Square

---



# Moving Least Square

---



# Programming Project #1

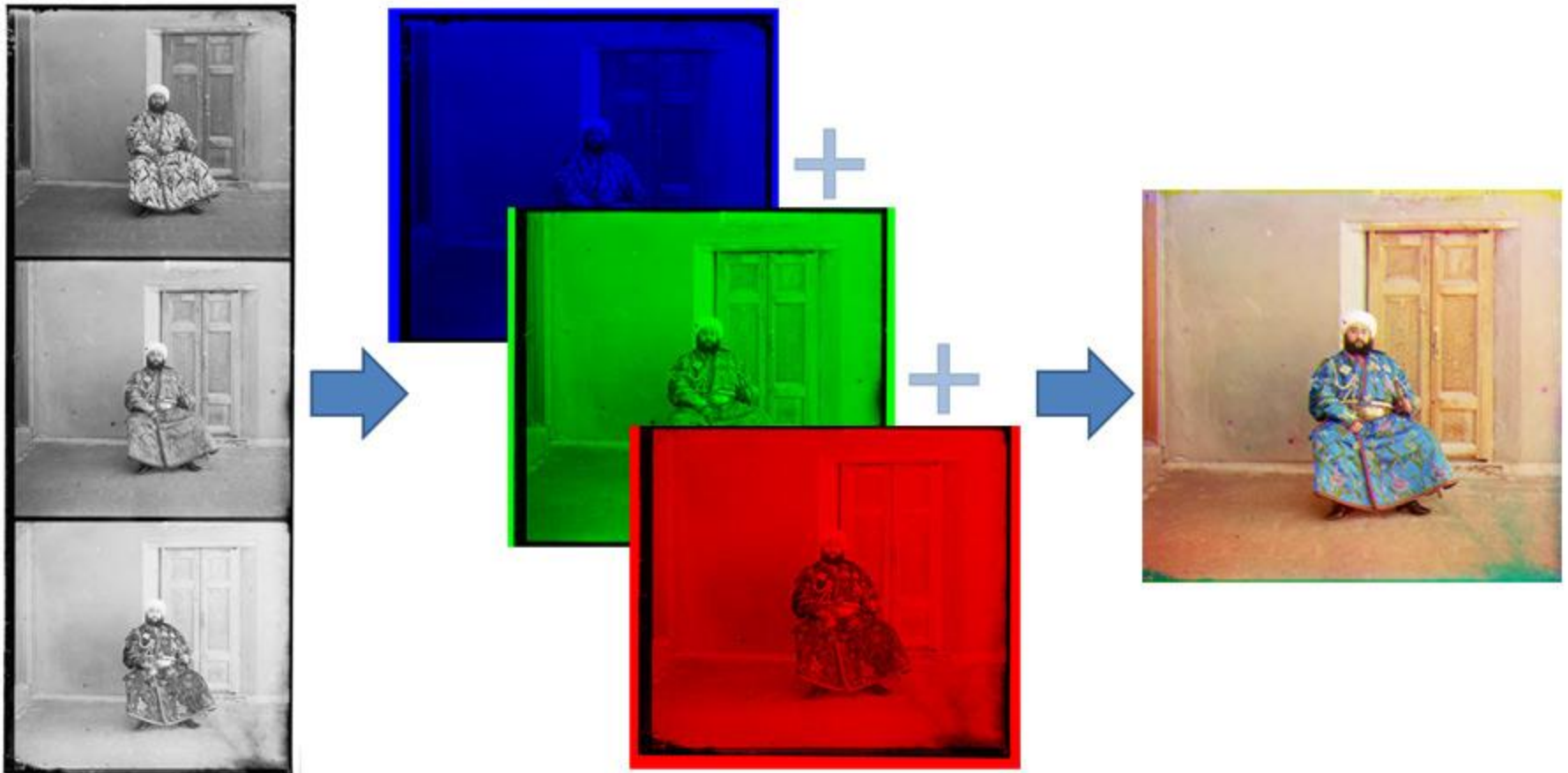
---

## Prokudin-Gorskii's Color Photography (1907)



# Programming Project #1

---



# Programming Project #1

---

- How to compare R,G,B channels?
- No right answer
  - Sum of Squared Differences (SSD):

$$ssd(u, v) = \sum_{(x,y) \in N} [I(u+x, v+y) - P(x, y)]^2$$

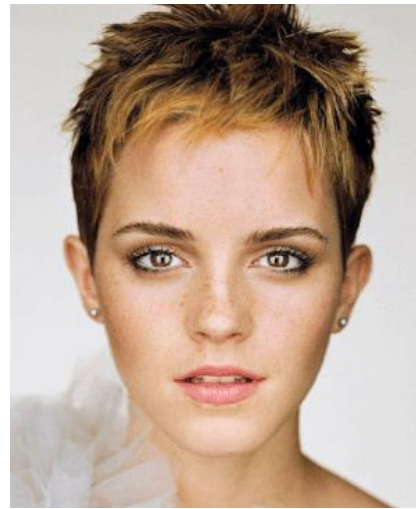
- Normalized Correlation (NCC):

$$ncc(u, v) = \frac{\sum_{(x,y) \in N} [I(u+x, v+y) - \bar{I}] [P(x, y) - \bar{P}]}{\sqrt{\sum_{(x,y) \in N} [I(u+x, v+y) - \bar{I}]^2 \sum_{(x,y) \in N} [P(x, y) - \bar{P}]^2}}$$



---

# Thank You!



© Rachel Albert

16-726, Spring 2025

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