Computational Photography and Video:
Warping, Morphing and Mosaics

Prof. Marc Pollefeys

Dr. Gabriel Brostow

ETH

inf | Informatik Computer Science

Today's schedule

- Last week's recap
- Warping
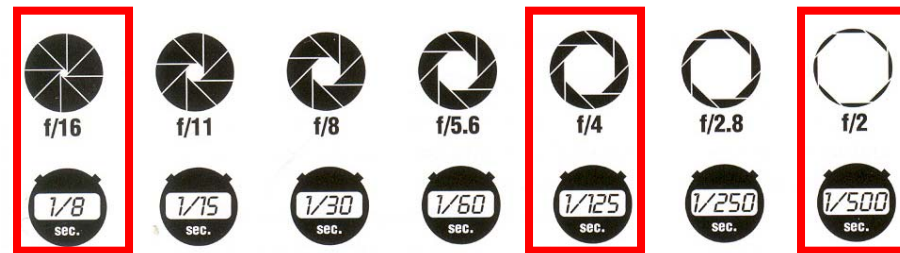- Morphing
- Mosaics

ETH

Today's schedule

- Last week's recap
- Warping
- Mosaics
- Morphing

ETH
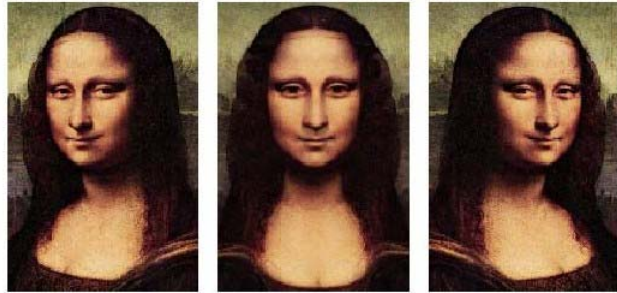
Exposure: shutter vs. aperture

- Trade-off motion blur vs. depth-of-field

Small aperture (deep depth of field), slow shutter speed (motion blurred). In this scene, a small aperture (f/16) produced great depth of field; the nearest paving stones as well as the farthest trees are sharp. But to admit enough light, a slow shutter speed (1/8 sec) was needed; it was too slow to show moving pigeons sharp. It also meant that a tripod had to be used to hold the camera steady.
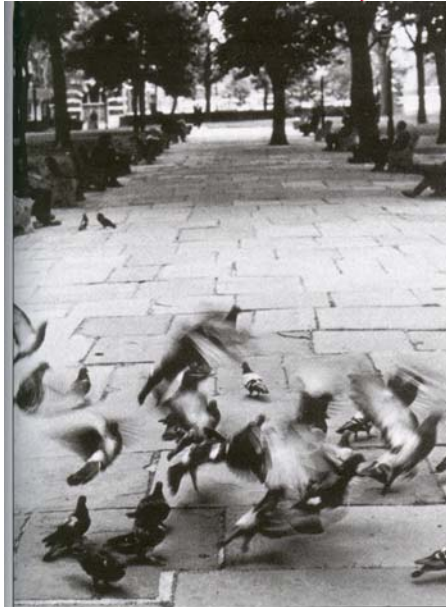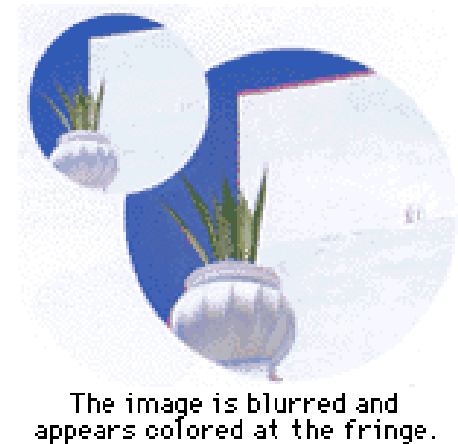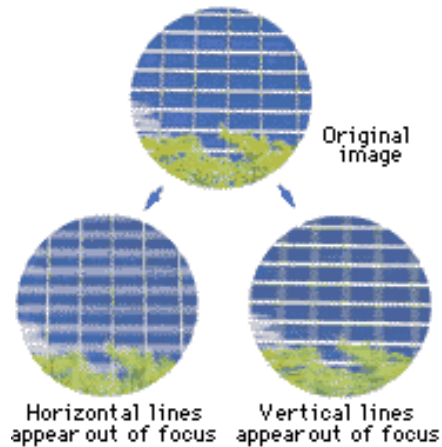
Medium aperture (moderate depth of field), medium shutter speed (some motion sharp). A medium aperture (f/4) and shutter speed (1/125 sec) sacrifice some background detail to produce recognizable images of the birds. But the exposure is still too long to show the motion of the birds' wings sharply.
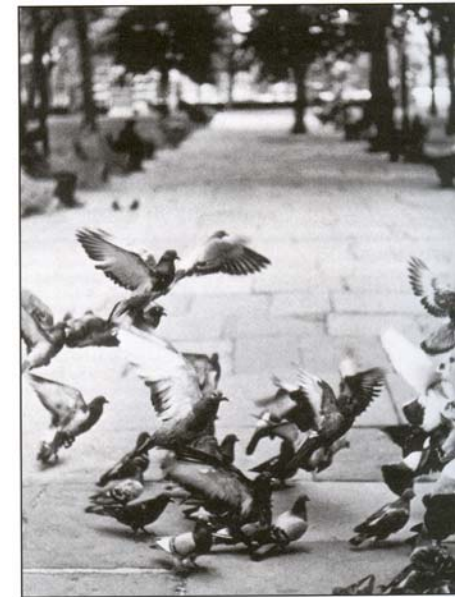
Large aperture (shallow depth of field), fast shutter speed (motion sharp). A fast shutter speed (1/500 sec) stops the motion of the pigeons so completely that the flapping of individual wings can be seen. A large aperture (f/2) made this possible, but at a cost of reduced depth of field.

(also gain/ISO sensitivity)

Imperfect lenses: aberrations, etc.

Original image

Horizontal lines appear out of focus

Vertical lines appear out of focus

The image is blurred and appears colored at the fringe.

ETH
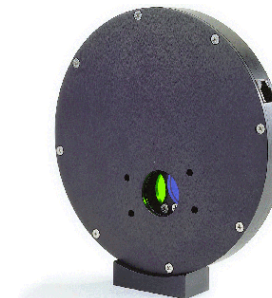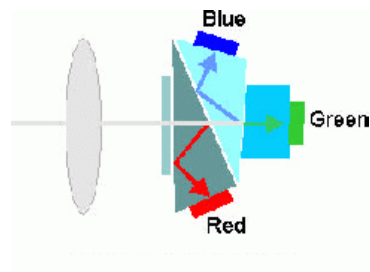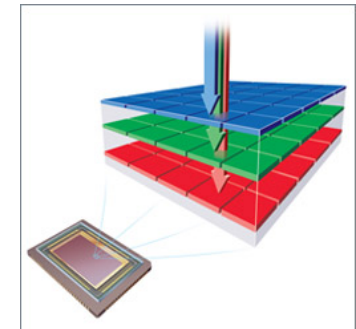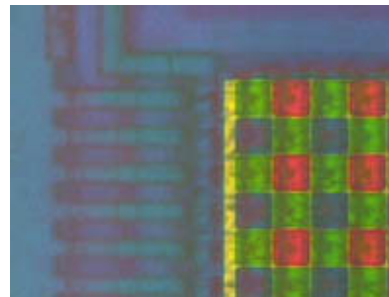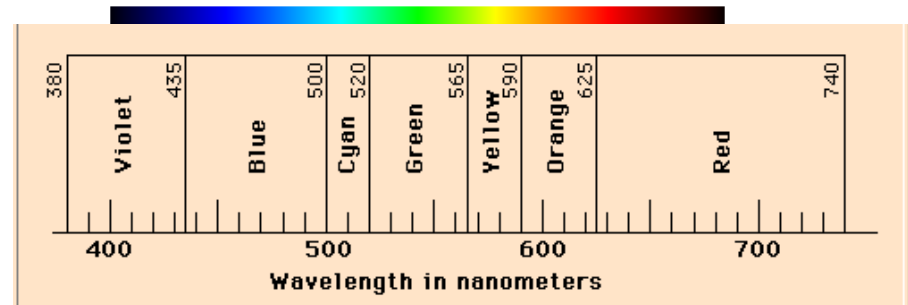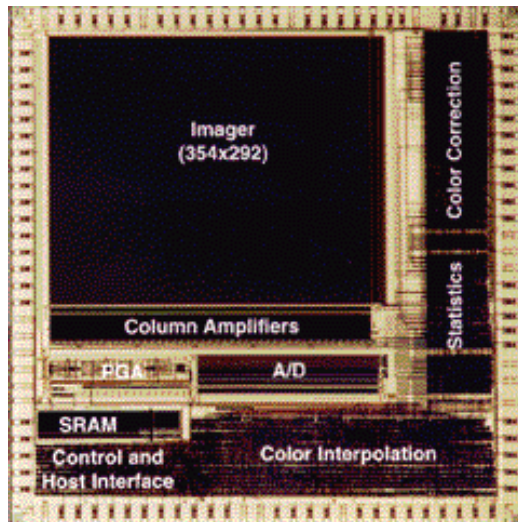
Sensors and color

Schedule	Computational Photography and Video	
20 Feb	Introduction to Computational Photography	
27 Feb	More on Cameras, Sensors and Color	Assignment 1: Color
5 Mar	Warping, morphing and mosaics	Assignment 2: Alignment
12 Mar	Image pyramids, Graphcuts	Assignment 3: Blending
19 Mar	Dynamic Range, HDR imaging, tone mapping	Assignment 4: HDR
26 Mar	*Easter holiday – no classes*	
2 Apr	TBD	Project proposals
9 Apr	TBD	Papers
16 Apr	TBD	Papers
23 Apr	TBD	Papers
30 Apr	TBD	Project update
7 May	TBD	Papers
14 May	TBD	Papers
21 May	TBD	Papers
28 May	TBD	Final project presentation

ETH

Today's schedule

- Last week's recap
- Warping
- Mosaics
- Morphing

ETH

Image Warping

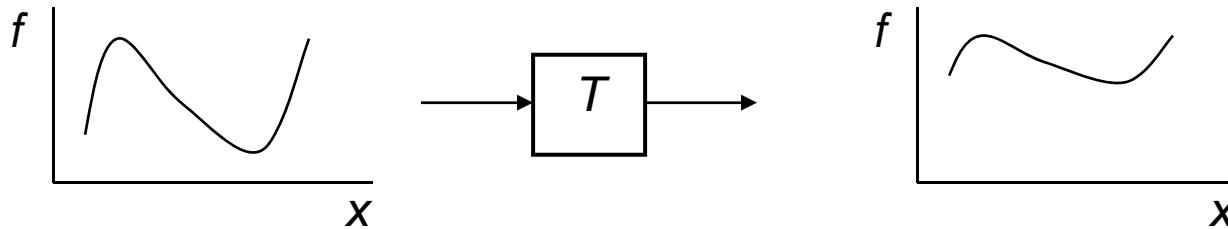image filtering: change ***range*** of image

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

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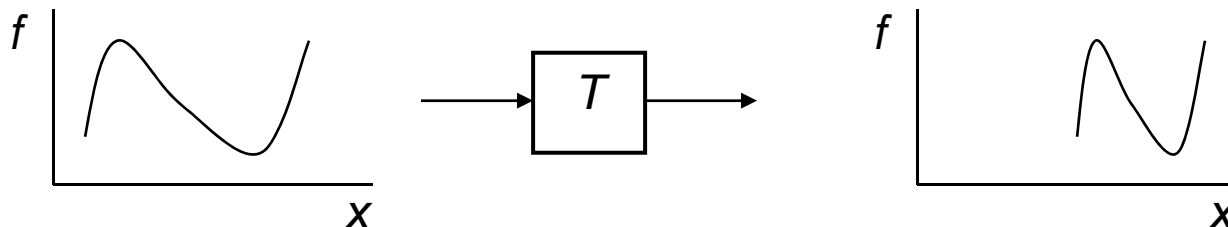
$$g(x) = f(T(x))$$

Image Warping

image filtering: change **range** of image

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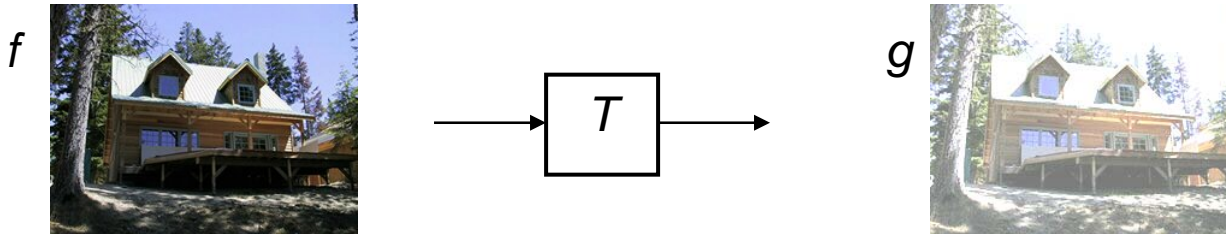
image warping: change **domain** of image

$$g(x) = f(T(x))$$
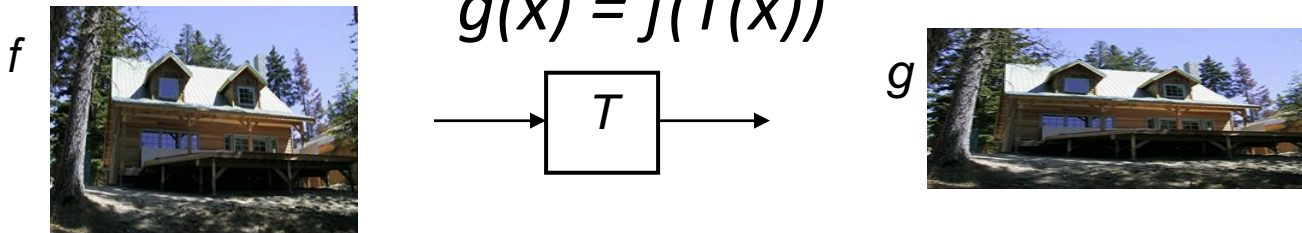
Parametric (global) warping

- Examples of parametric warps:

translation

rotation

aspect

affine

perspective

cylindrical

ETH

Parametric (global) warping
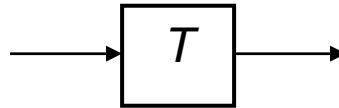
p = (x,y)   **p'** = (x',y')

- 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 *T* as a matrix:

  p' = **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:

$\times 2$

ETH
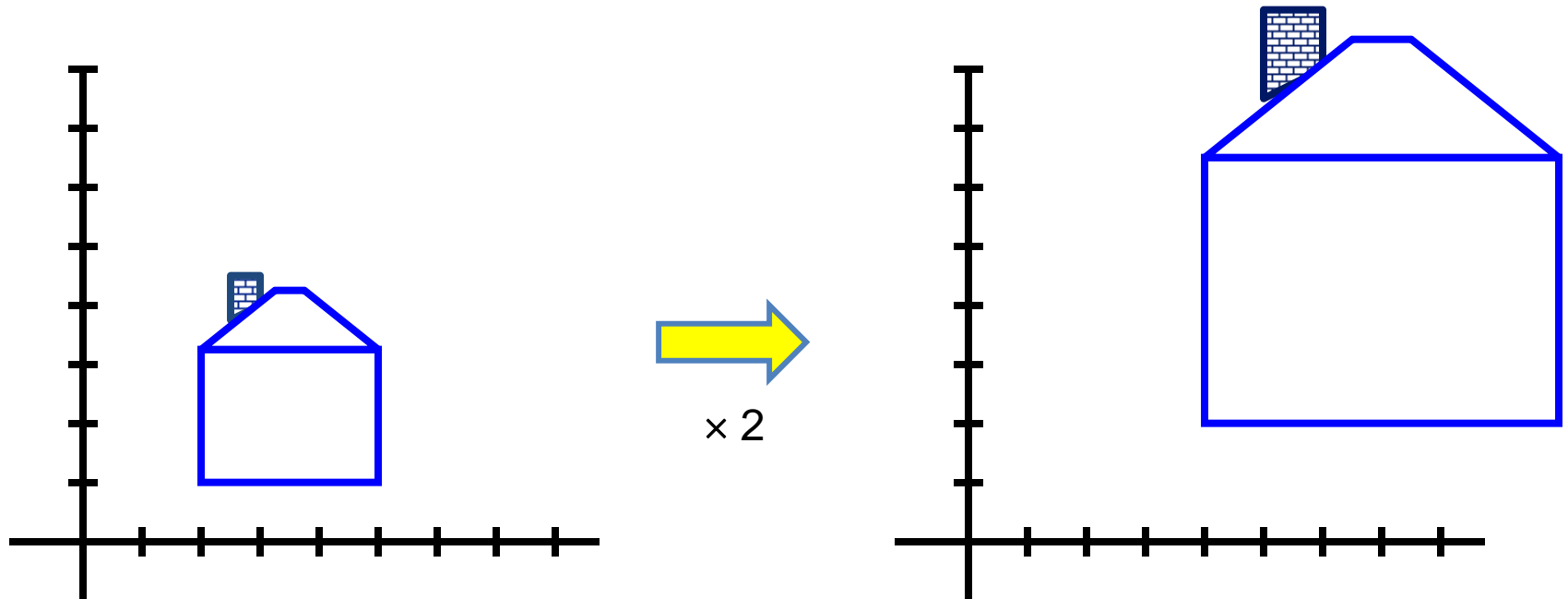
Scaling

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

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

ETH
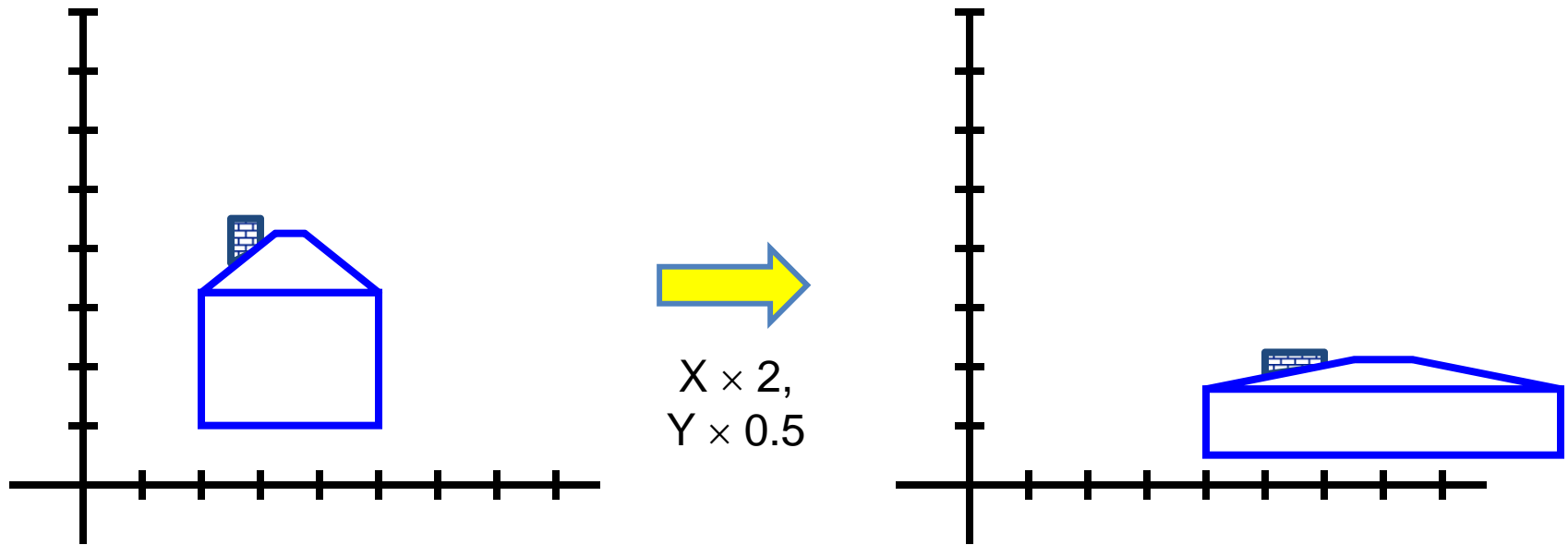
Scaling

- Scaling operation:

$$x' = ax$$

$$y' = by$$

- Or, in matrix form:

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

scaling matrix S

What's inverse of S?

2-D Rotation

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

ETH
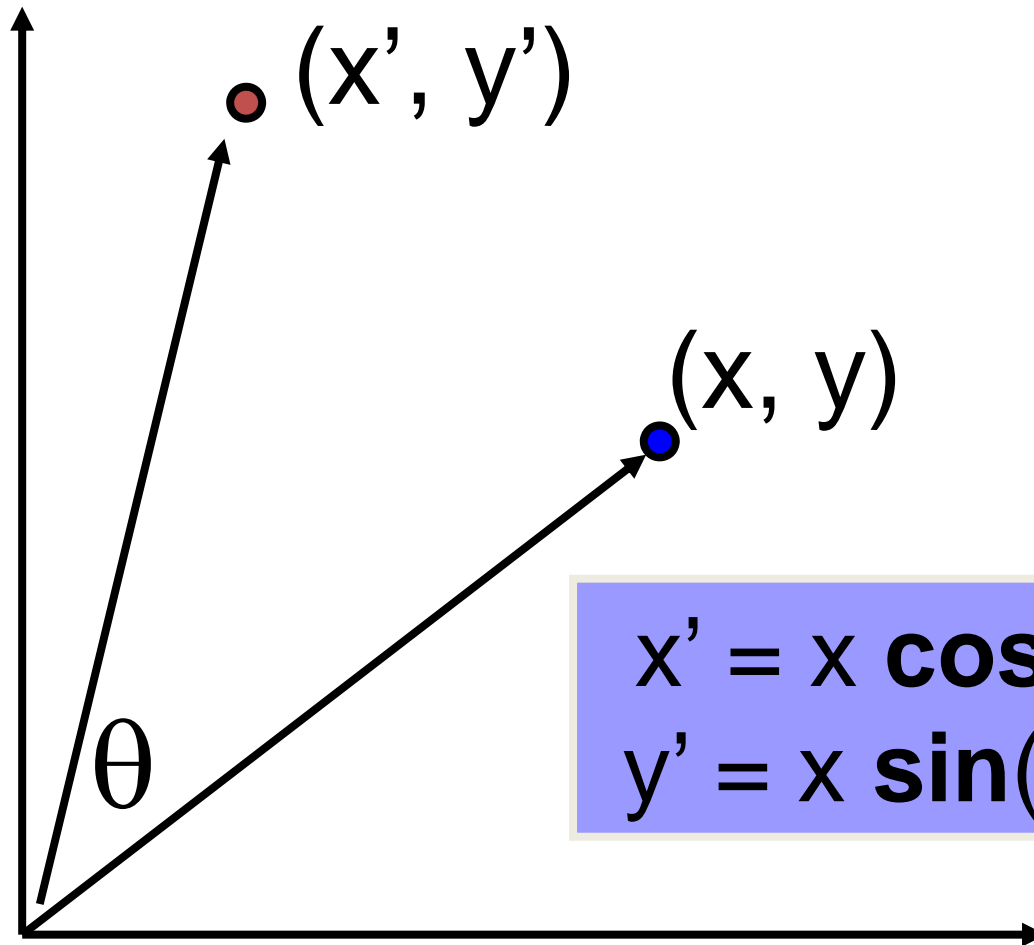
2-D Rotation

• This is easy to capture in matrix form:

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

• 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 $\quad \mathbf{R}^{-1} = \mathbf{R}^{T}$

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)?

$$x' = s_x * x$$
$$y' = s_y * y$$

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

ETH

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?

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

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

ETH

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)?

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

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

ETH

2x2 Matrices

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

2D Translation?

$$x' = x + t_x$$

NO!

$$y' = y + t_y$$

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

ETH

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

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

ETH

Homogeneous Coordinates

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

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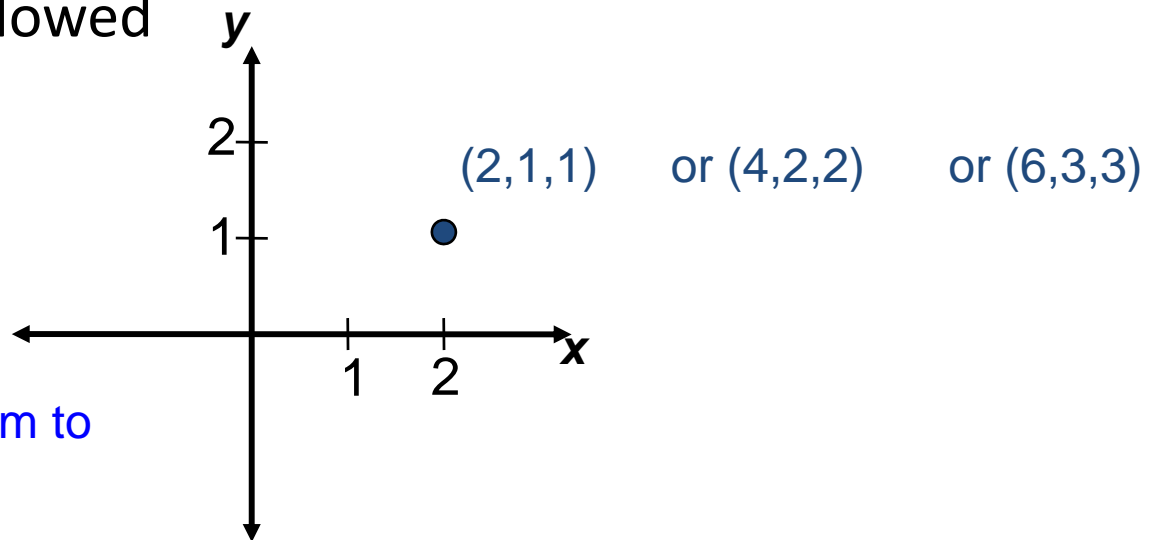
ETH

Homogeneous Coordinates

Homogeneous coordinates

– represent coordinates in 2 dimensions with a 3-vector

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

ETH

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

Homogeneous Coordinates

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

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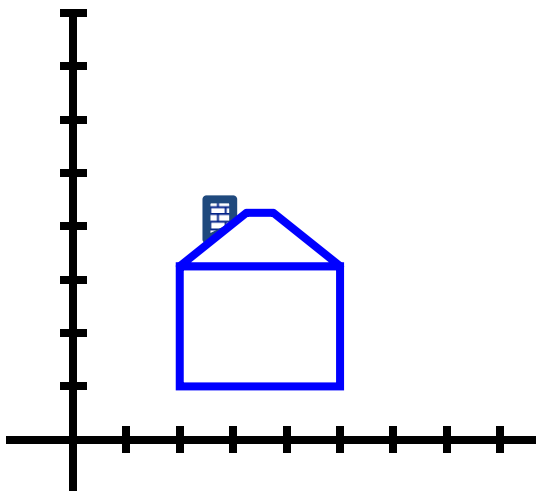$$y' = y + t_y$$

A: Using the rightmost column:

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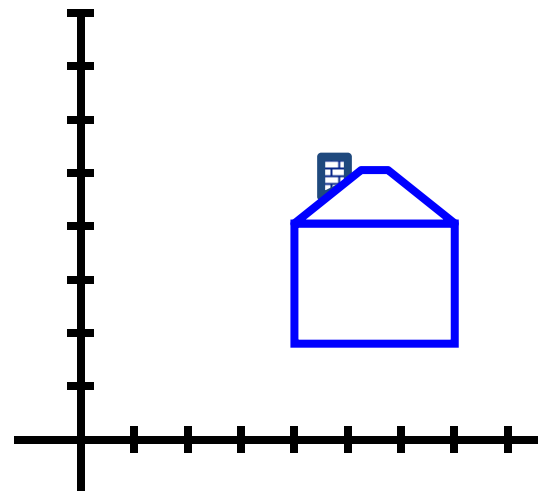
Translation

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

ETH

Affine Transformations

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

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

ETH

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

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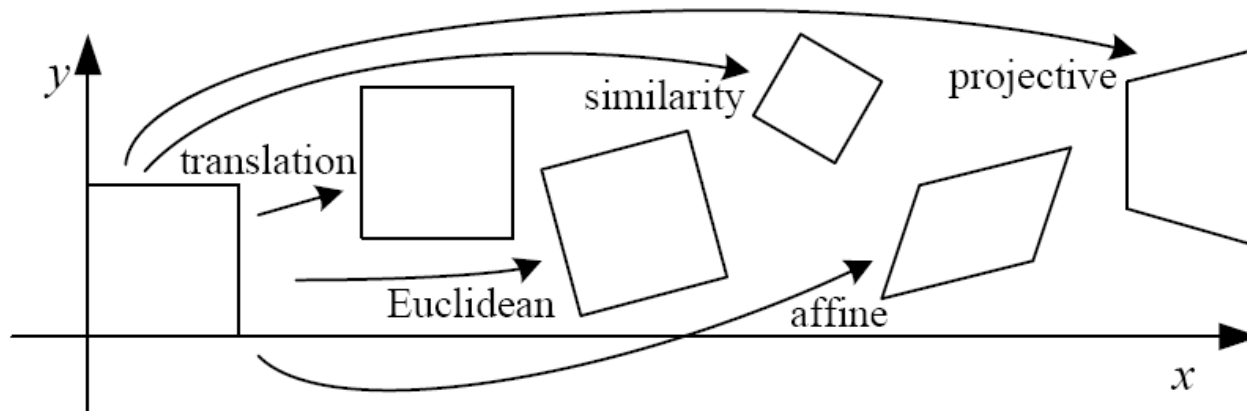
ETH

Matrix Composition

- Transformations can be combined by matrix multiplication

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

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

2D image transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\ \boldsymbol{I}\ \vert\ \boldsymbol{t}\ \right]_{2\times3}$			□
rigid (Euclidean)	$\left[\ \boldsymbol{R}\ \vert\ \boldsymbol{t}\ \right]_{2\times3}$			◇
similarity	$\left[\ s\boldsymbol{R}\ \vert\ \boldsymbol{t}\ \right]_{2\times3}$			◇
affine	$\left[\ \boldsymbol{A}\ \right]_{2\times3}$			▱
projective	$\left[\ \tilde{\boldsymbol{H}}\ \right]_{3\times3}$			⬡

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

Image Warping in Biology

- D'Arcy Thompson

  http://www-groups.dcs.st-and.ac.uk/~history/Miscellaneous/darcy.html
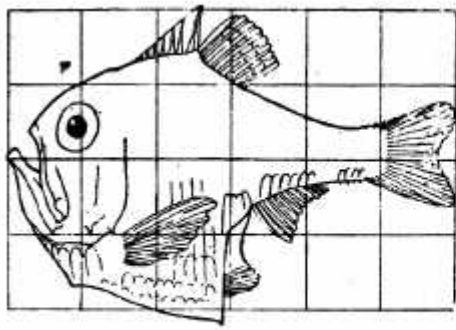
  http://en.wikipedia.org/wiki/D'Arcy_Thompson
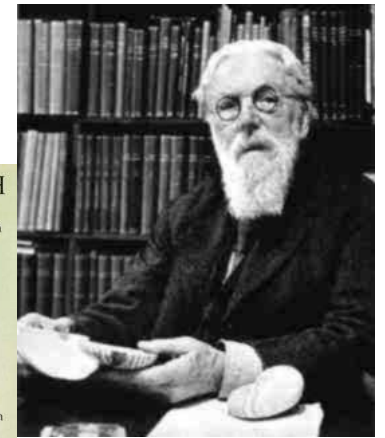
- Importance of shape and structure in evolution

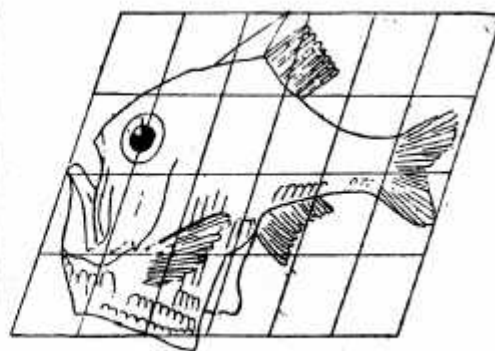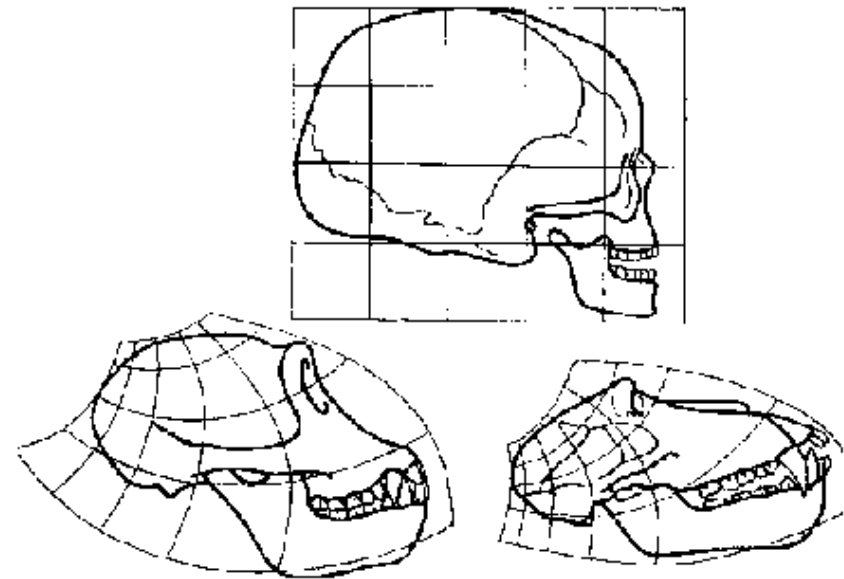Fig. 517.   *Argyropelecus Olfersi.*   Fig. 518.   *Sternoptyx diaphana.*

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

Slide by Durand and Freeman

Recovering Transformations

y x $f(x,y)$

?

$T(x,y)$

y' x' $g(x',y')$
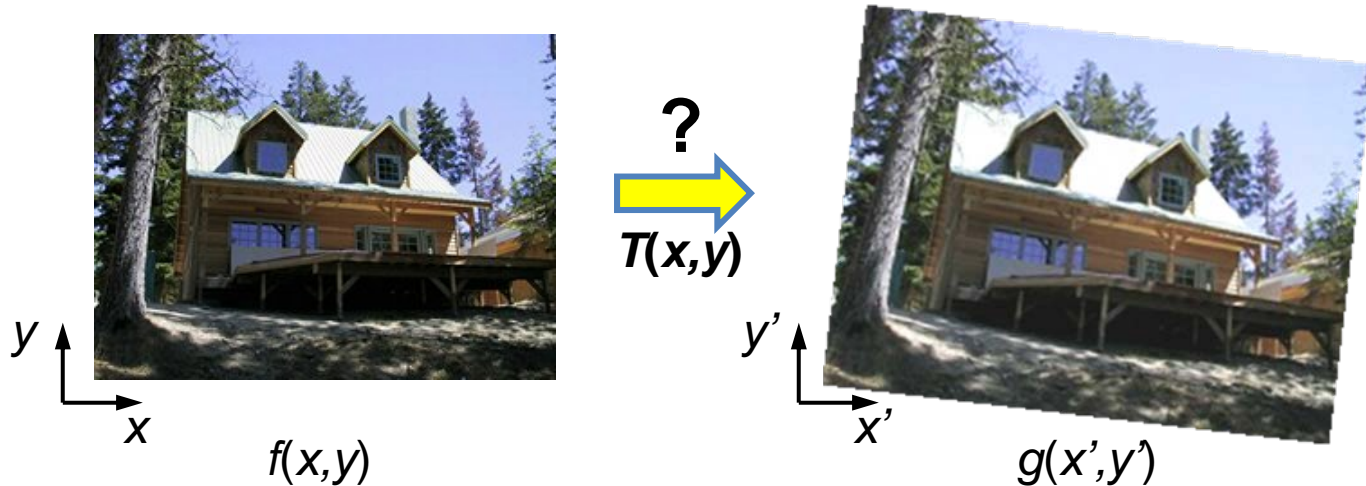
- What if we know f and g and want to recover the transform T?
  - willing to let user provide correspondences
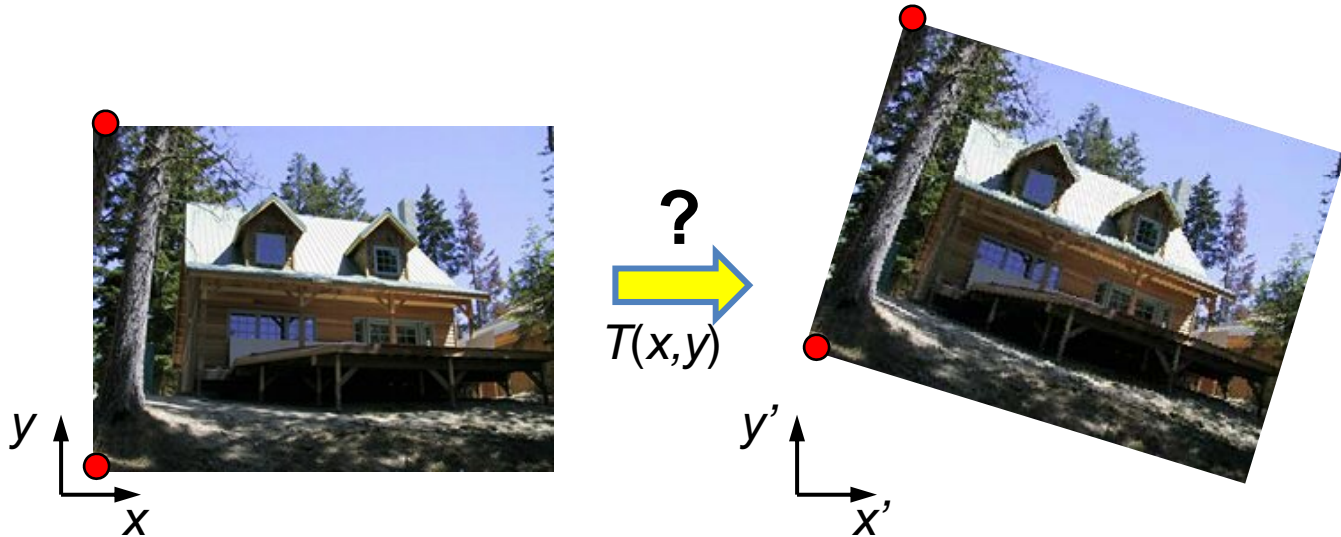   - How many do we need?

ETH

Translation: # correspondences?

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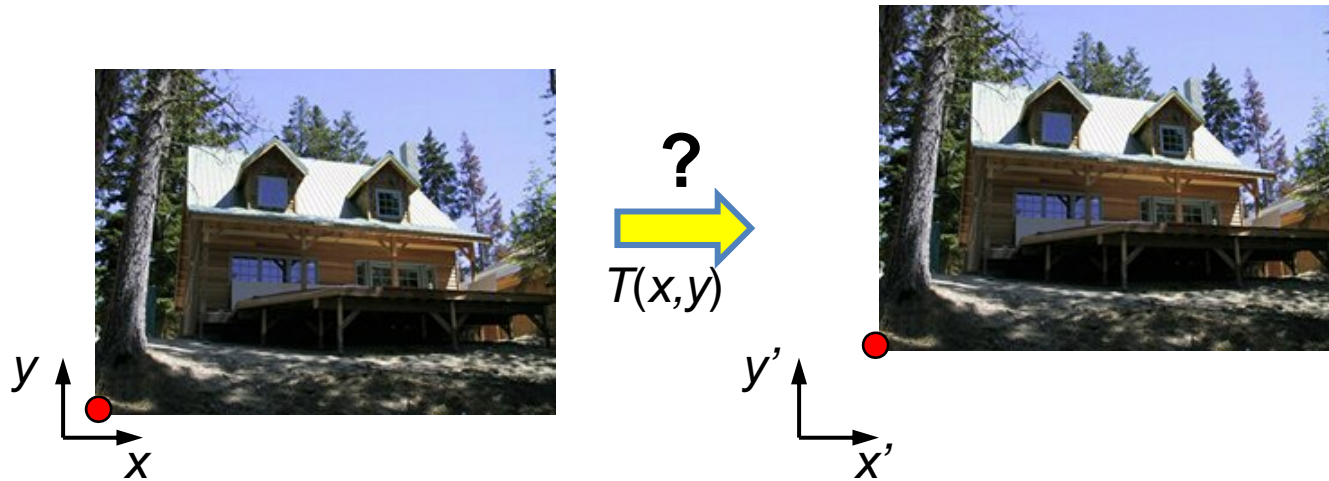
- How many correspondences needed for translation?

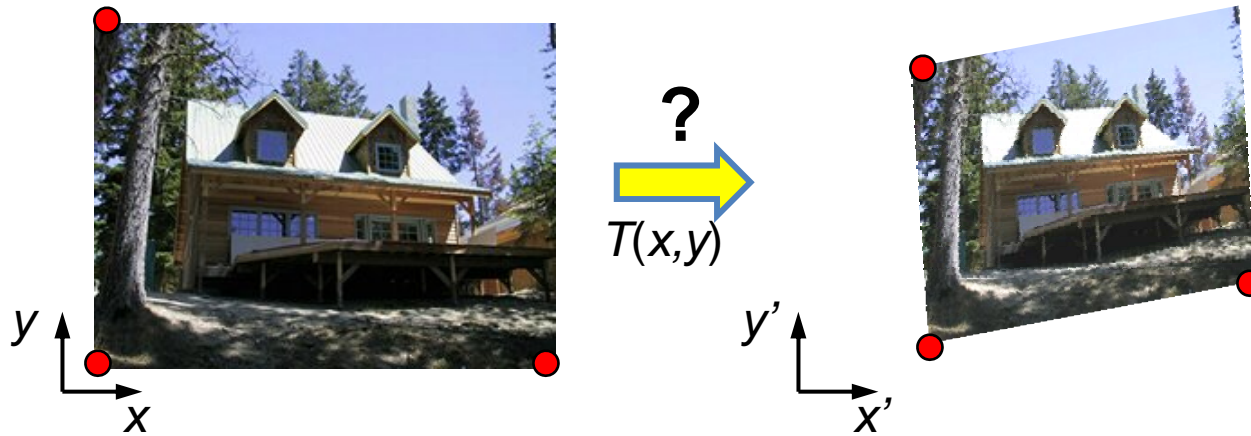- How many Degrees of Freedom?

- What is the transformation matrix?

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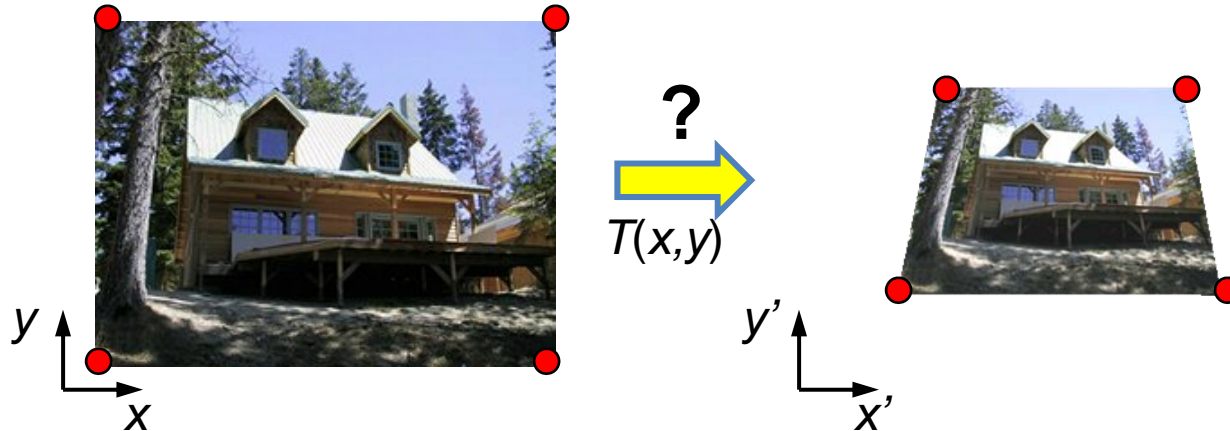
Euclidian: # correspondences?

$T(x,y)$

y
x

y'
x'

- How many correspondences needed for translation+rotation?

- How many DOF?
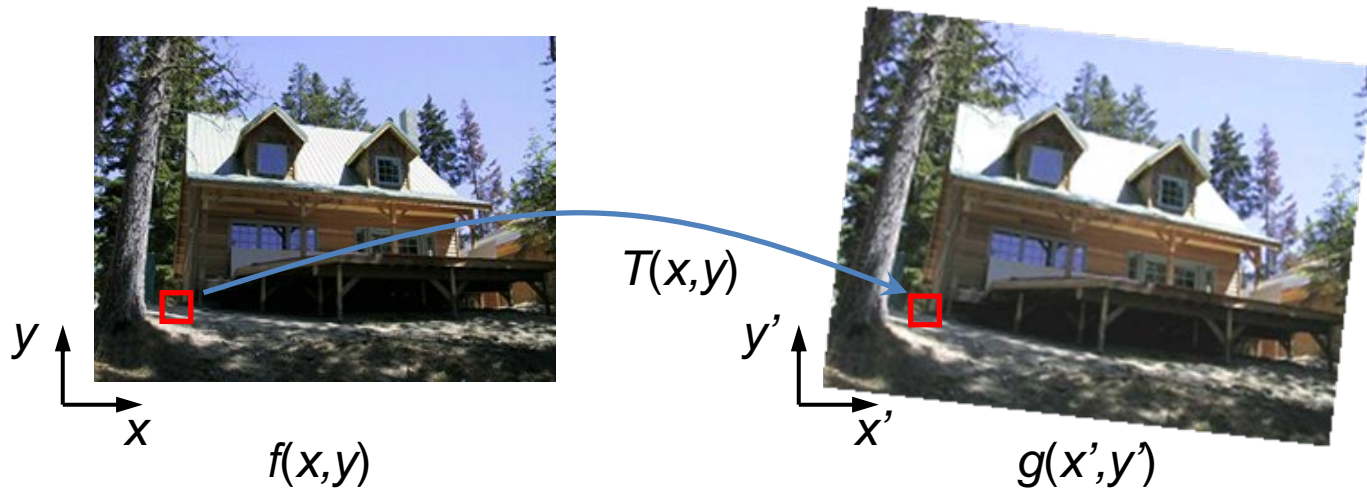
ETH

Affine: # correspondences?

$T(x,y)$

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

ETH

Projective: # correspondences?
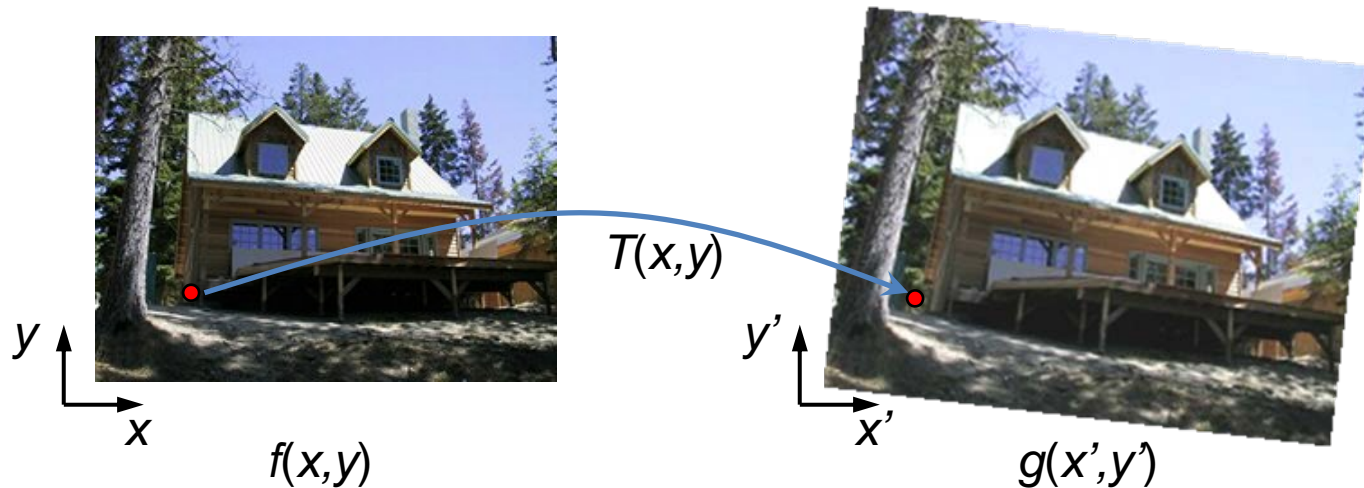

$T(x,y)$

- How many correspondences needed for projective?

- How many DOF?

ETH

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))*?
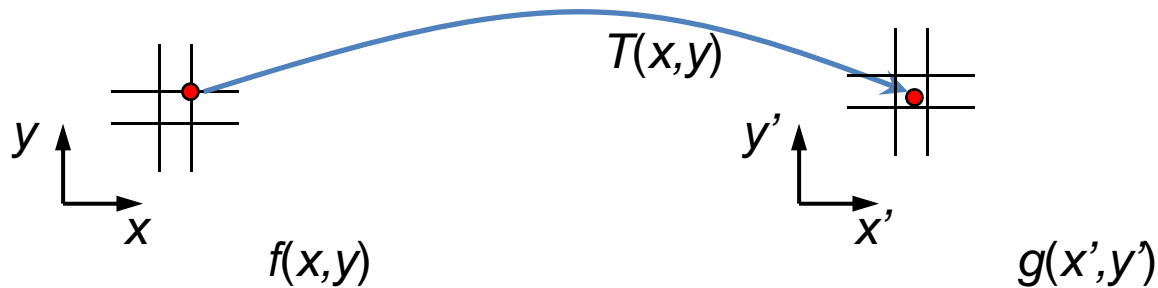
Forward warping

$T(x,y)$

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?

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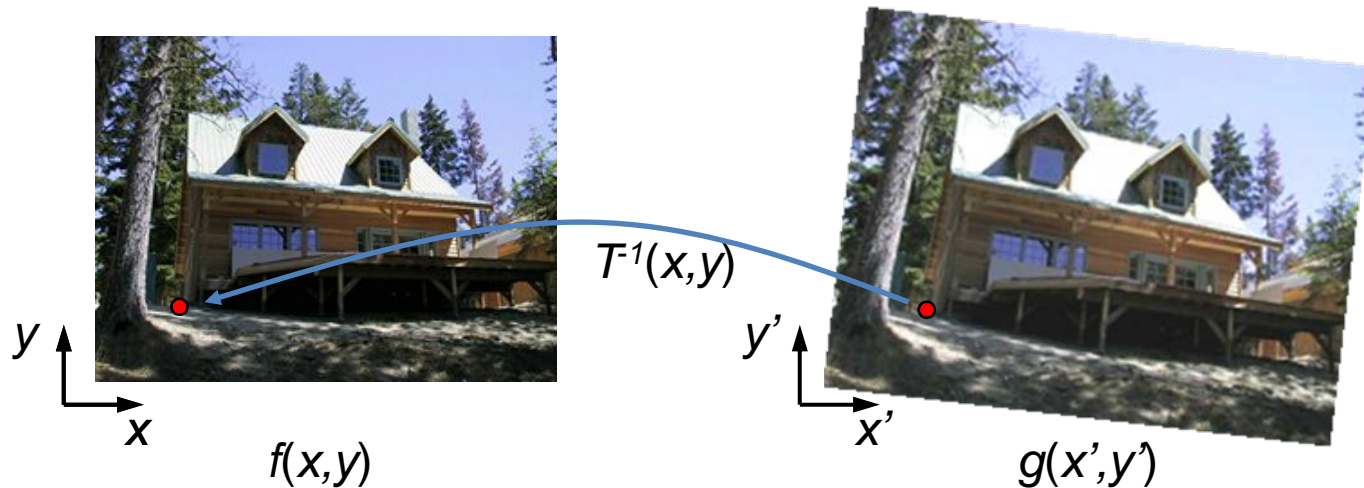
  $(x',y') = T(x,y)$ in the second image

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

A: distribute color among neighboring pixels (x',y')
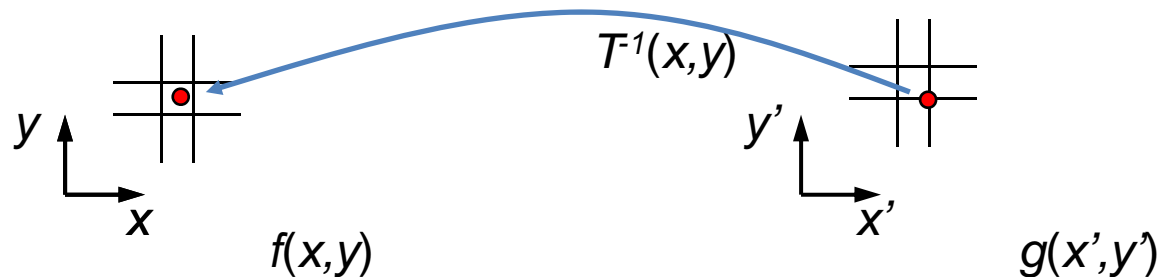
 – Known as "splatting"

 – 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

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

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

- 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...

ETH

Today's schedule

- Last week's recap
- Warping
- Mosaics
- Morphing

ETH

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV   = 200 x 135°

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
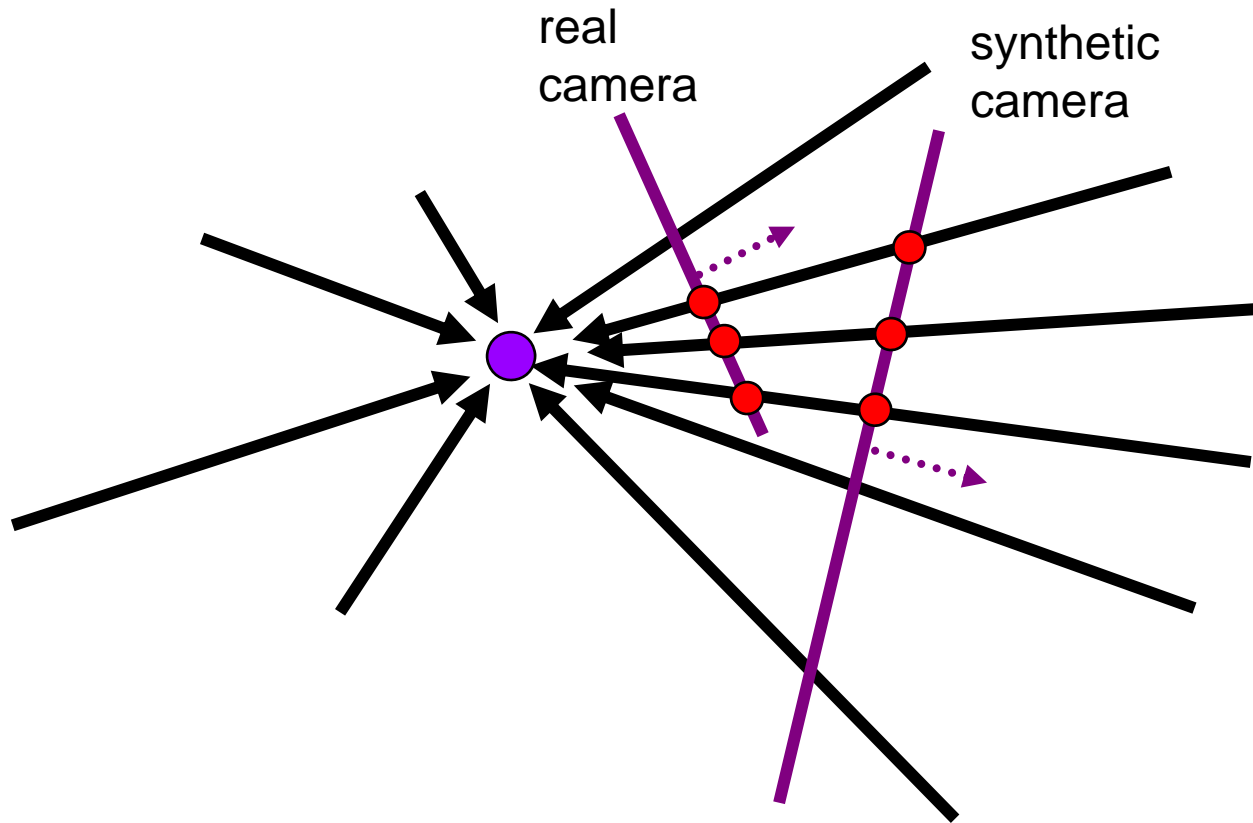 - Human FOV   = 200 x 135°

Slide from Brown & Lowe

Mosaics: stitching images together

virtual wide-angle camera

A pencil of rays contains all views

real
camera

synthetic
camera

Can generate any synthetic camera view
as long as it has **the same center of projection**!

ETH

How to do it?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - If there are more images, repeat

- …but **wait**, why should this work at all?
  - What about the 3D geometry of the scene?
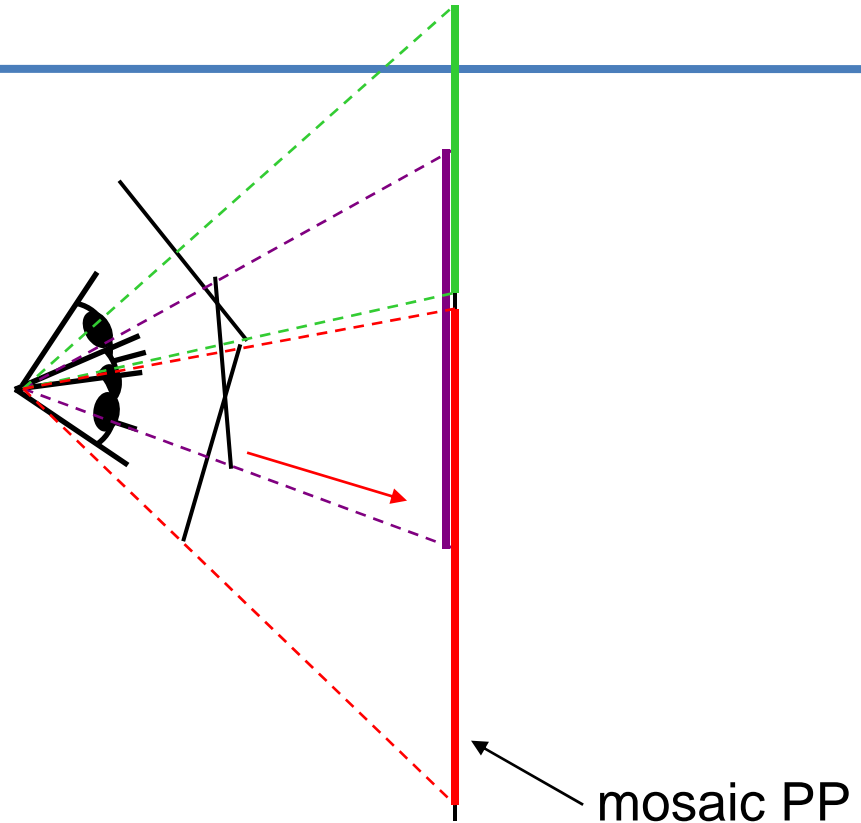 - Why aren't we using it?

ETH

Aligning images

left on top

right on top

Translations are not enough to align the images

ETH

Image reprojection

mosaic PP

- The mosaic has a natural interpretation in 3D
  - The images are reprojected onto a common plane
  - The mosaic is formed on this plane
  - Mosaic is a *synthetic wide-angle camera*

Image reprojection

- ## Basic question
  - How to relate two images from the same camera center?
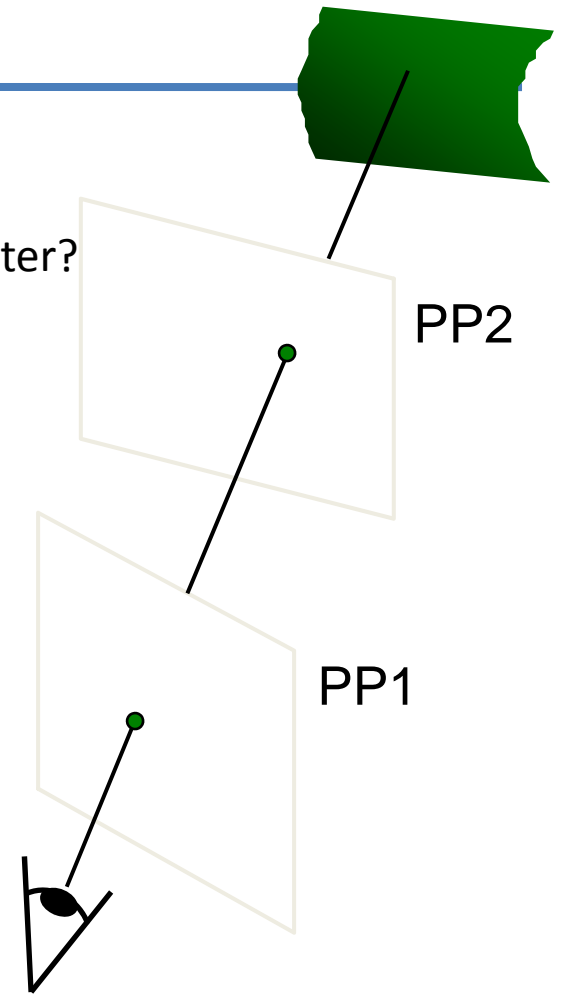   - how to map a pixel from PP1 to PP2

    Answer
    - Cast a ray through each pixel in PP1
    - Draw the pixel where that ray intersects PP2

PP2

PP1

But don't we need to know the geometry
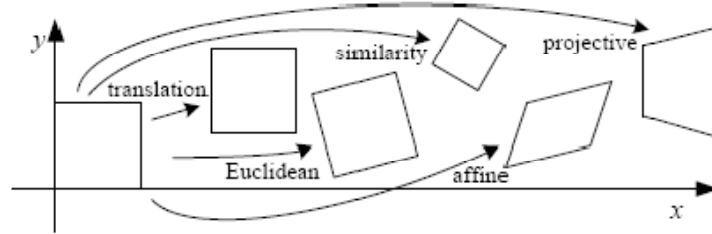of the two planes in respect to the eye?

Observation:
Rather than thinking of this as a 3D reprojection,
think of it as a 2D **image warp** from one image to another

Back to Image Warping

Which t-form is the right one for warping PP1 into PP2?
e.g. translation, Euclidean, affine, projective

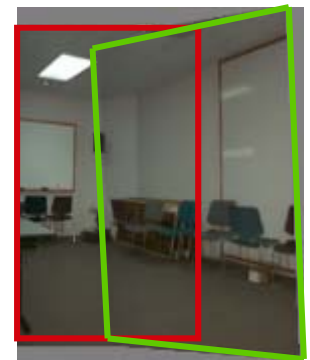**Translation**   **Affine**   **Perspective**

2 unknowns   **6 unknowns**   **8 unknowns**

Homography

- A: Projective – mapping between any two PPs with the same center of projection
    - rectangle should map to arbitrary quadrilateral
    - parallel lines aren't
    - but must preserve straight lines
    - same as: project, rotate, reproject

- called Homography

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\textbf{p'}$ $\qquad$ $\textbf{H}$ $\qquad$ $\textbf{p}$

To apply a homography **H**

- Compute **p'** = **Hp** (regular matrix multiply)
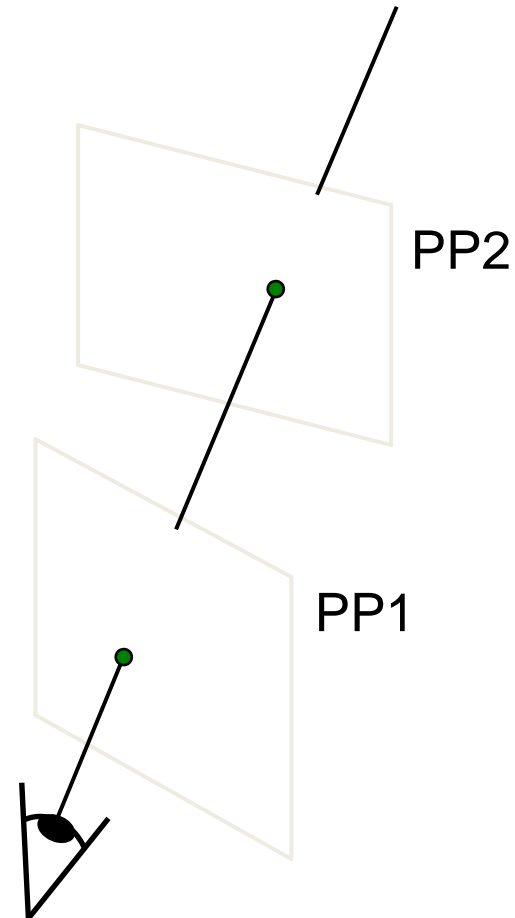- Convert **p'** from homogeneous to image coordinates

PP2

PP1

ETH
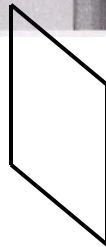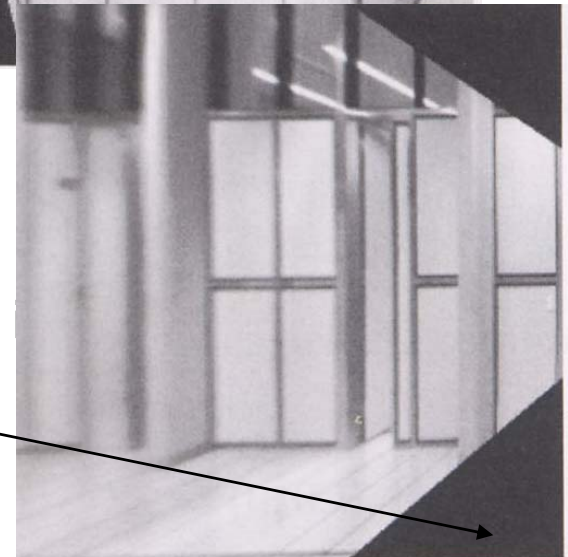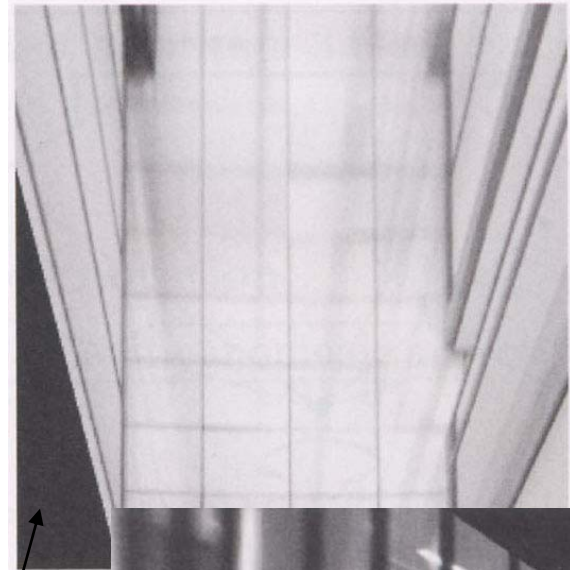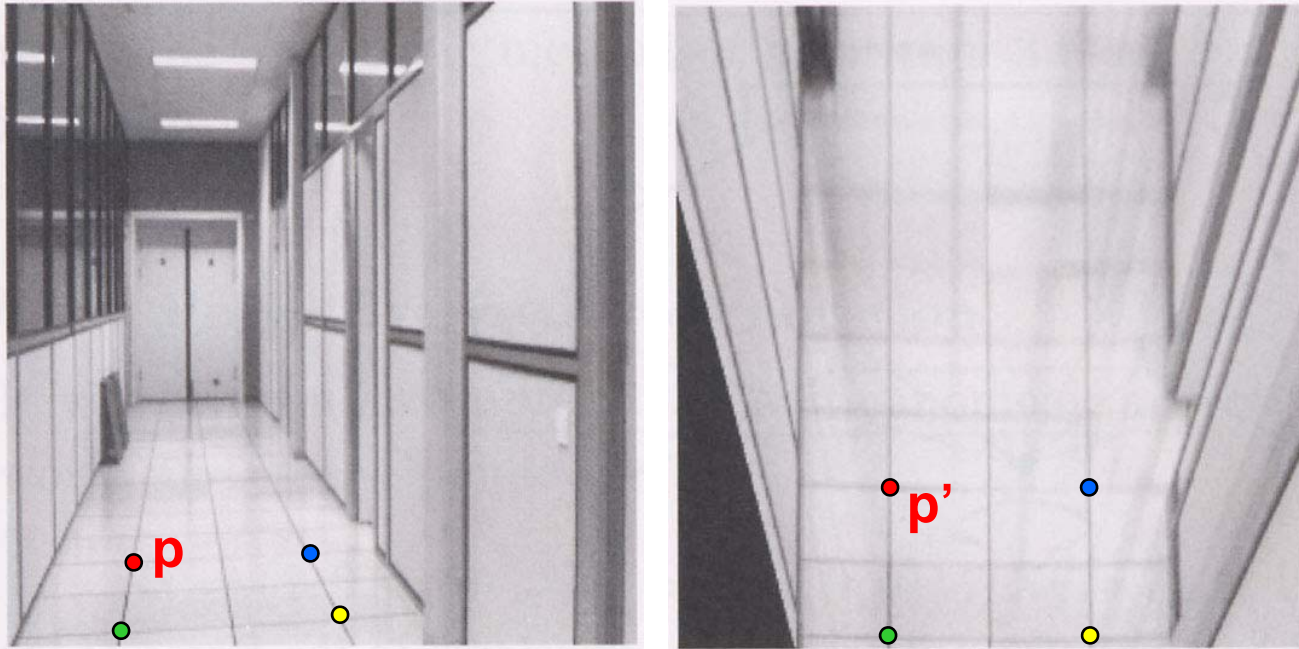
Image warping with homographies

image plane in front

black area
where no pixel
maps to

ETH

Image rectification

To unwrap (rectify) an image

- Find the homography **H** given a set of **p** and **p'** pairs
- How many correspondences are needed?
- Tricky to write H analytically, but we can <u>solve</u> for it!
- Find such H that "best" transforms points p into p'
- Use least-squares!

Computing a homography

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\lambda = h_3 \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{aligned} h_3 \mathbf{x} x' &= h_1 \mathbf{x} \\ h_3 \mathbf{x} y' &= h_2 \mathbf{x} \end{aligned}$$
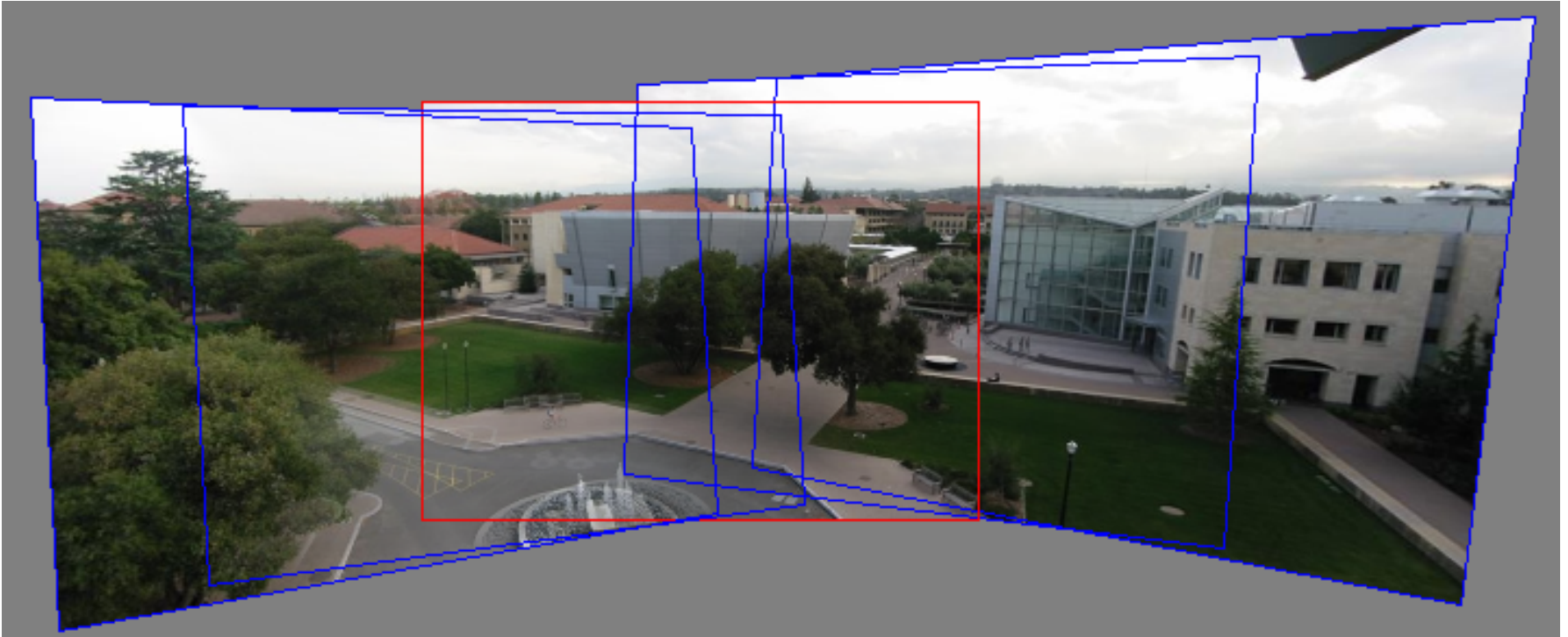
$$\begin{bmatrix} \mathbf{x}^\top & 0 & -x' \mathbf{x}^\top \\ 0 & \mathbf{x}^\top & -y' \mathbf{x}^\top \end{bmatrix} \begin{bmatrix} h_1^\top \\ h_2^\top \\ h_3^\top \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

2 equations/point

Stack matrix representing equations for 4 or more points

Solve through SVD, least-square solution is given by last right singular vector

$$\mathrm{USV}^\top . \mathrm{V}_9 = \sigma_9 \quad \text{(i.e. smallest singular value)}$$
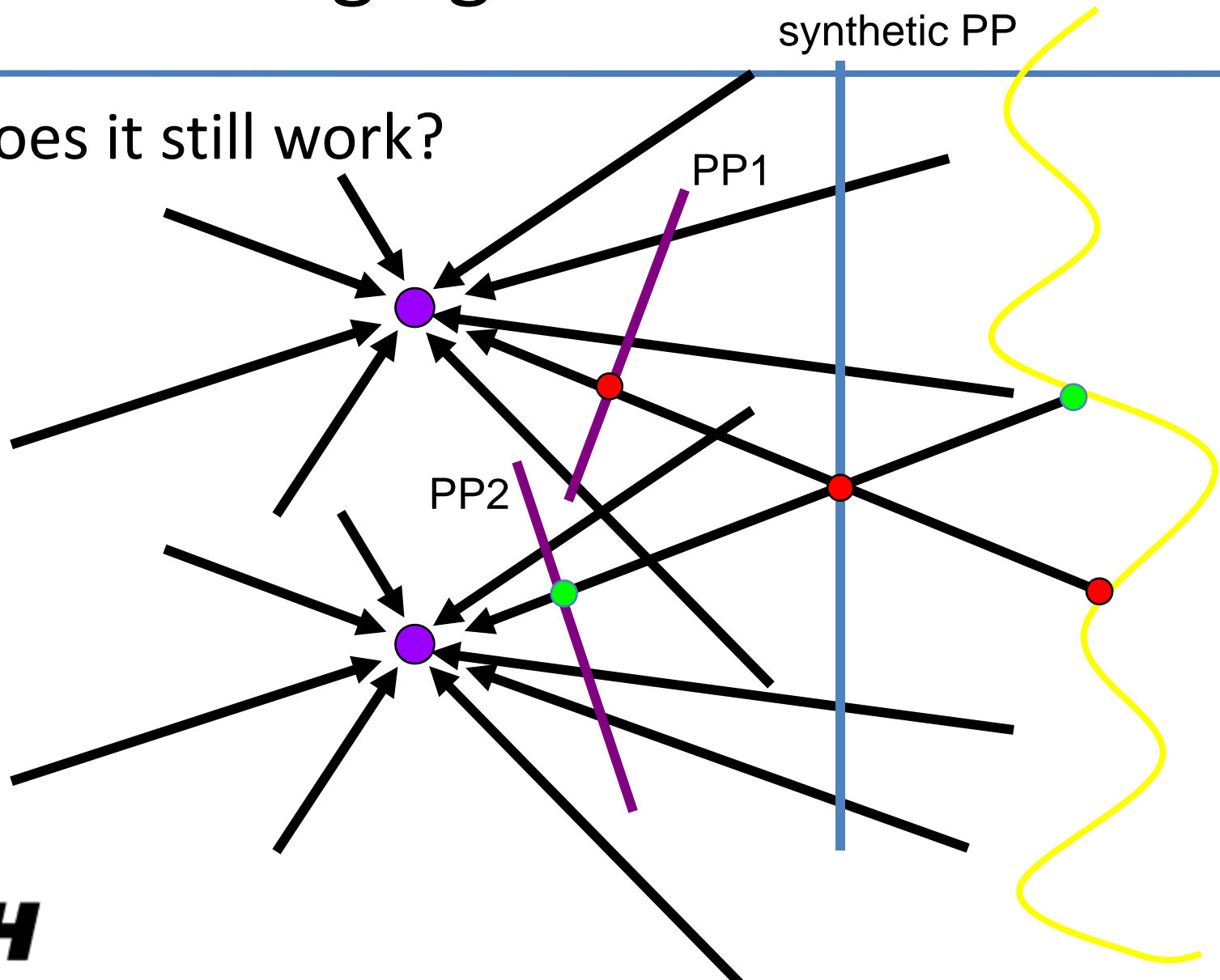
Numerically better if image coordinates scaled to [-1,1]

Panoramas

1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend

ETH

changing camera center

synthetic PP

Does it still work?
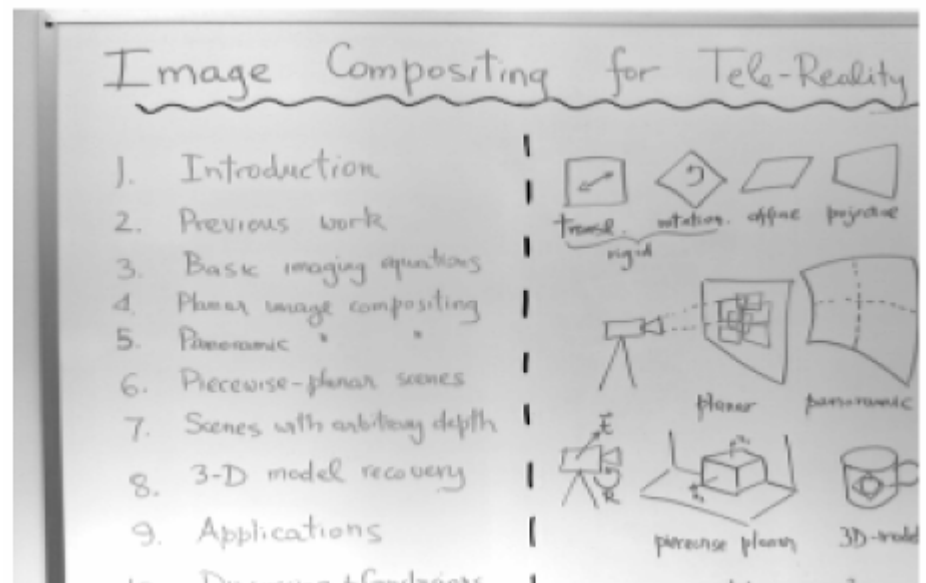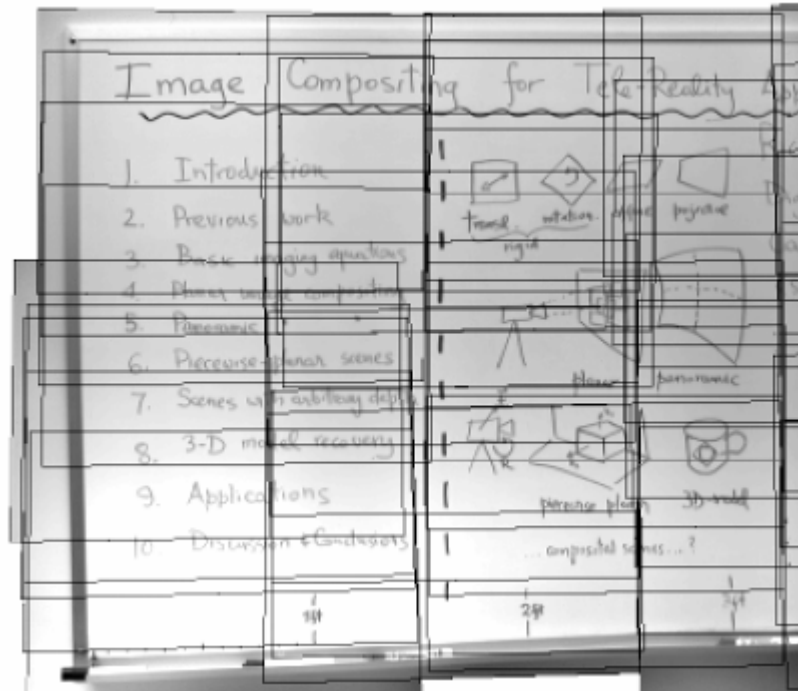
PP1

PP2

ETH

Planar scene (or far away)

PP3

PP1

PP2
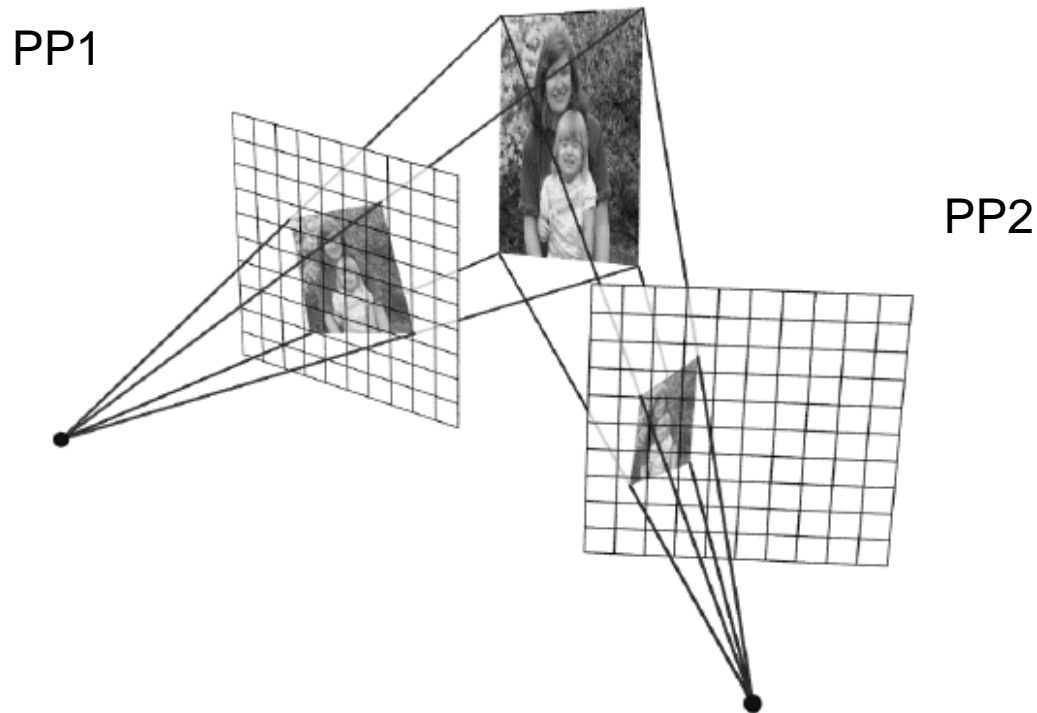
- PP3 is a projection plane of both centers of projection, so we are OK!
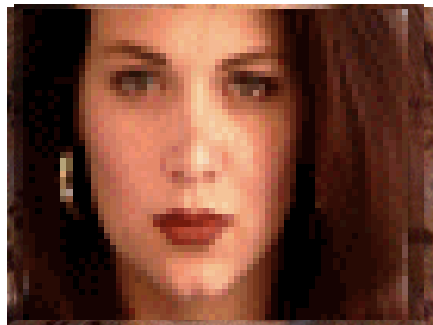- This is how big aerial photographs are made

ETH

Planar mosaic

Today's schedule

- Last week's recap
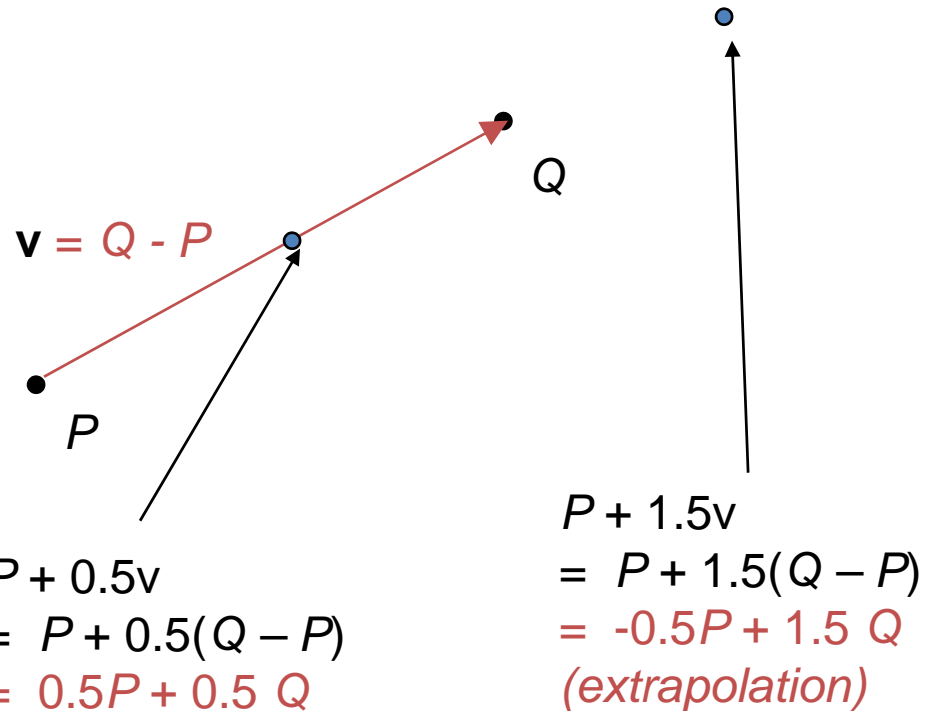- Warping
- Mosaics
- Morphing

ETH

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

ETH

Averaging Points

What's the average
of P and Q?

$\mathbf{v} = Q - P$

Q

P

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

$P + 1.5v$
$= P + 1.5(Q - P)$
$= -0.5P + 1.5\ Q$
(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.

ETH

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?

ETH

# Idea #2: Align, then cross-disolve

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

ETH

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

ETH

Idea #3: Local warp, then cross-dissolve
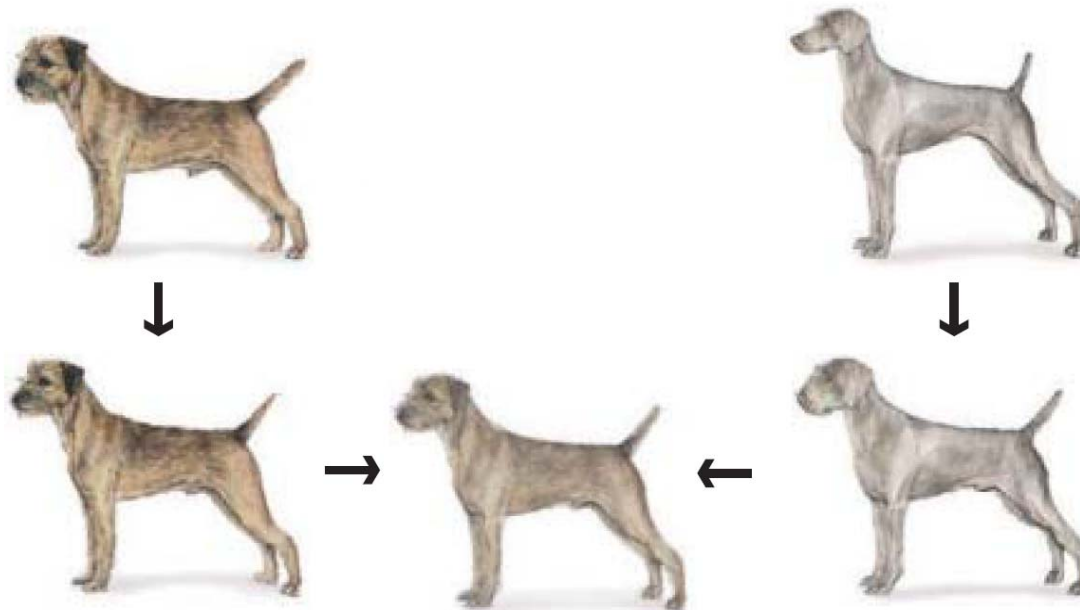
Morphing procedure:

for every t,

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

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)

ETH

Image Warping – non-parametric

- Move control points to specify a spline warp
- Spline produces a smooth vector field
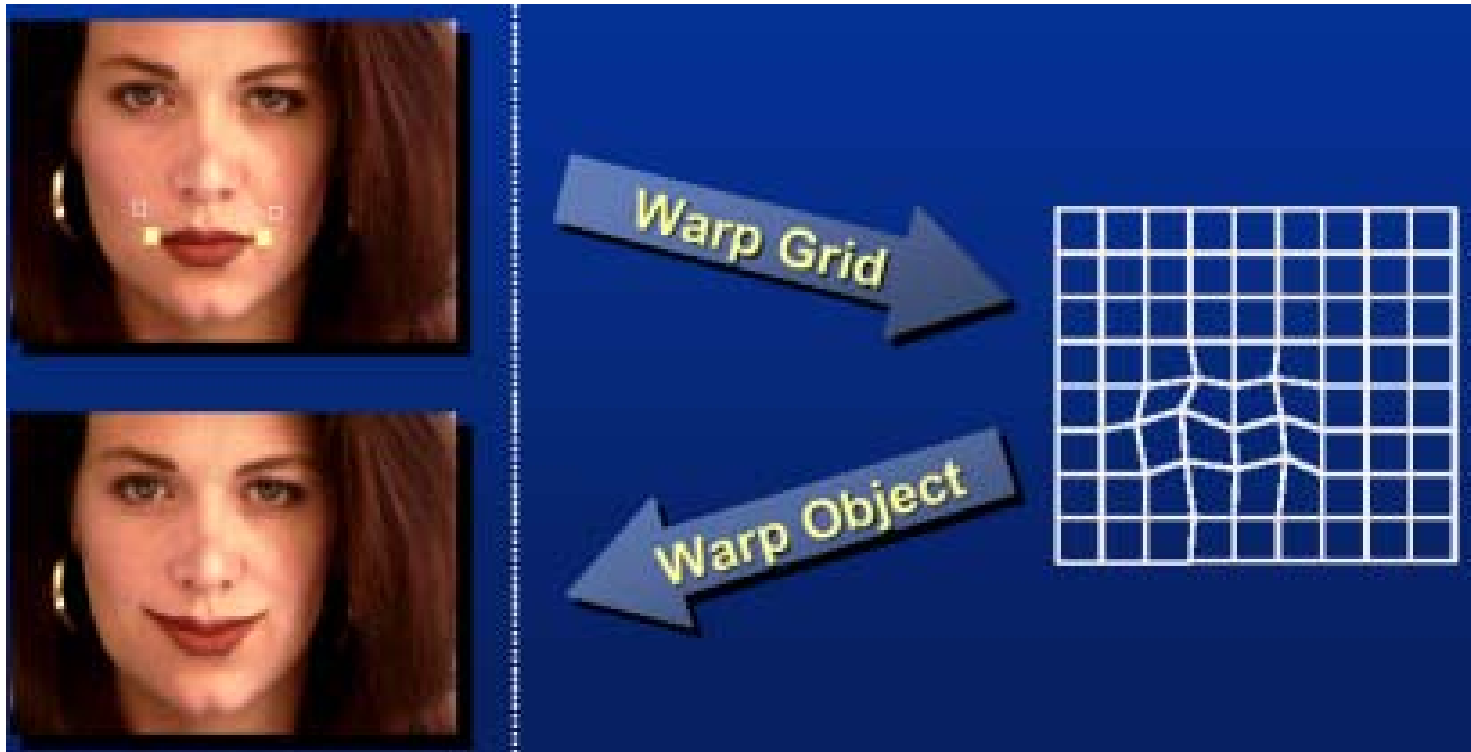
Warp specification - dense

- How can we specify the warp?

Specify corresponding *spline control points*

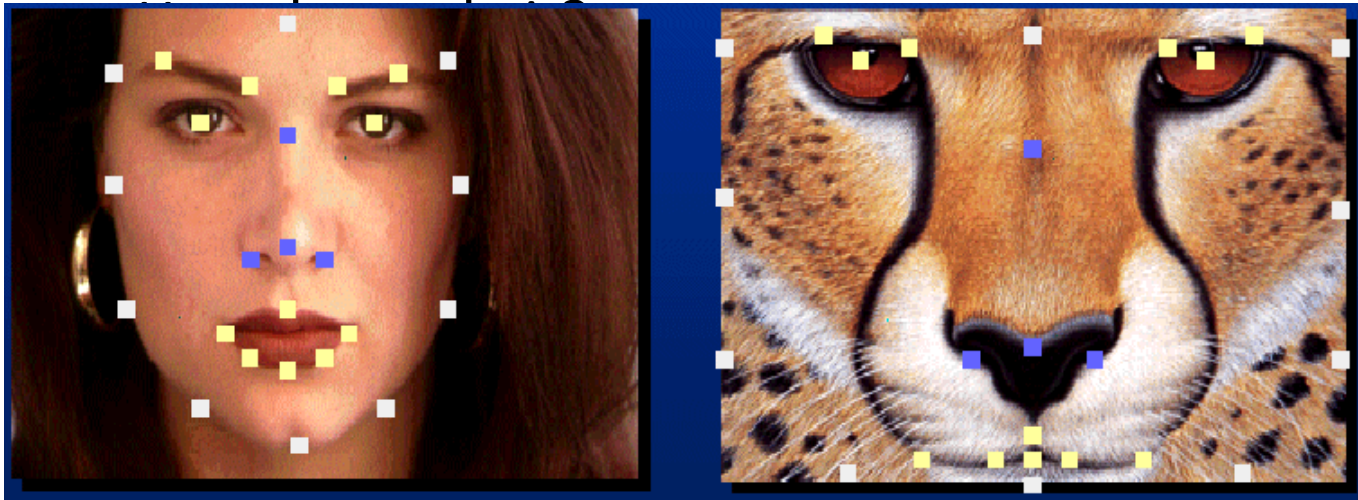- *interpolate* to a complete warping function

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

ETH

Warp specification - sparse

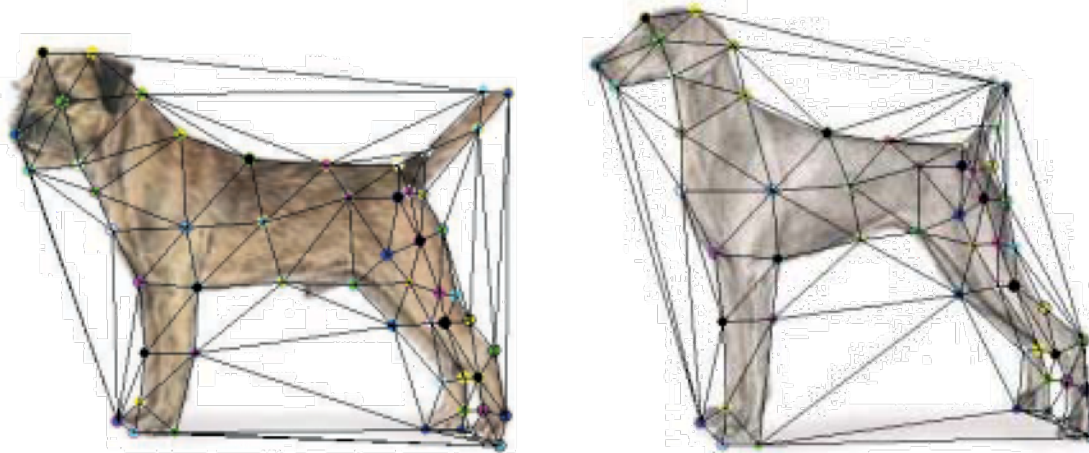- How can we specify the warp?

Specify corresponding *points*

- *interpolate* to a complete warping function

How do we go from feature points to pixels?

ETH

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?
   - 3 points = affine warp!
   - Just like texture mapping

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 images

ETH
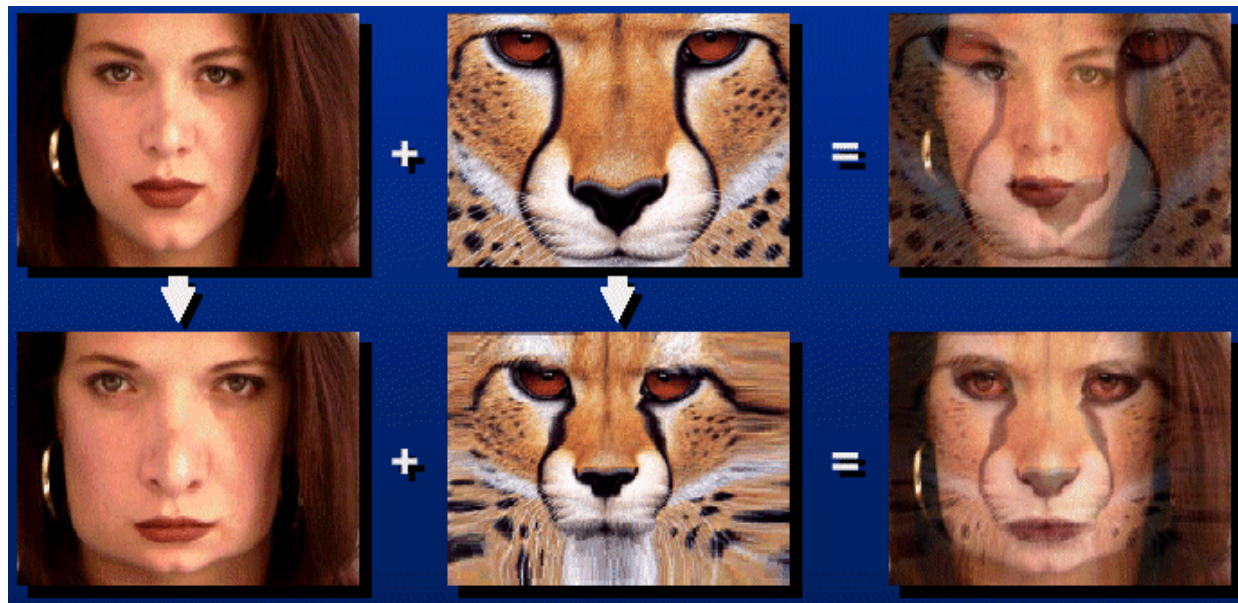
Warp interpolation

- How do we create an intermediate warp at time t?
  - Assume t = [0,1]
  - Simple linear interpolation of each feature pair
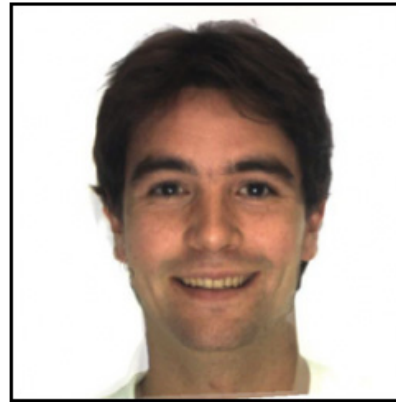 - (1-t)*p1+t*p0 for corresponding features p0 and p1
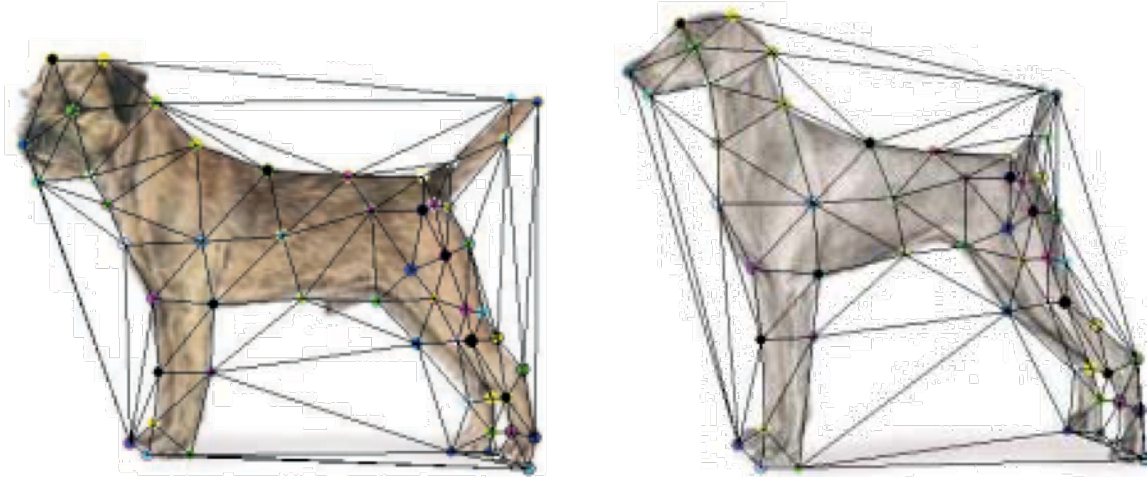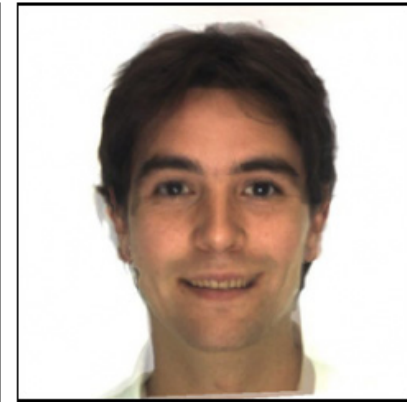
Morphing & matting

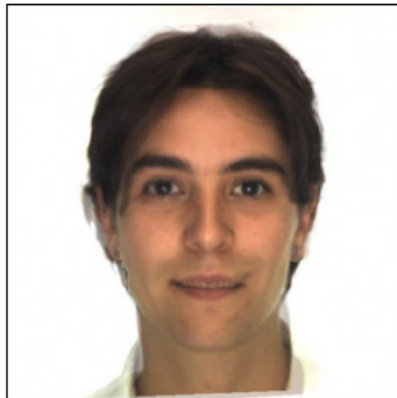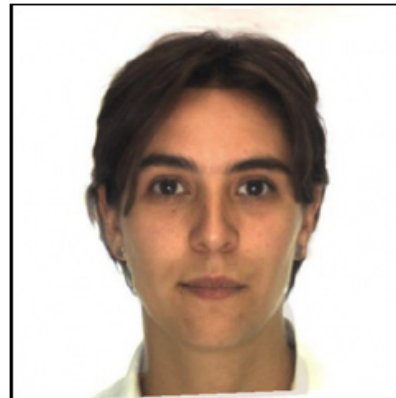- Extract foreground first to avoid artifacts in the background

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

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

Slide by Durand and Freeman

Women in Art video

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

ETH

Problem with morphing

- So far, we have performed linear interpolation of feature point positions

- But what happens if we try to morph between two views of the same object?
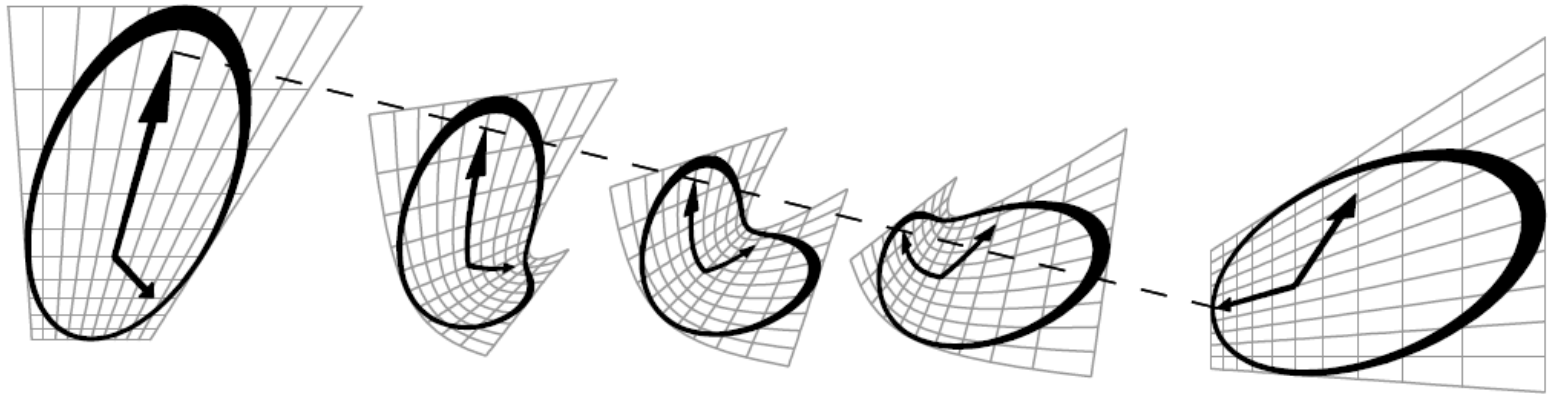
Figure 2: A Shape-Distorting Morph. Linearly interpolating two perspective views of a clock (far left and far right) causes a geometric bending effect in the in-between images. The dashed line shows the linear path of one feature during the course of the transformation. This example is indicative of the types of distortions that can arise with image morphing techniques.

View morphing

- Seitz & Dyer

  http://www.cs.washington.edu/homes/seitz/vmorph/vmorph.htm

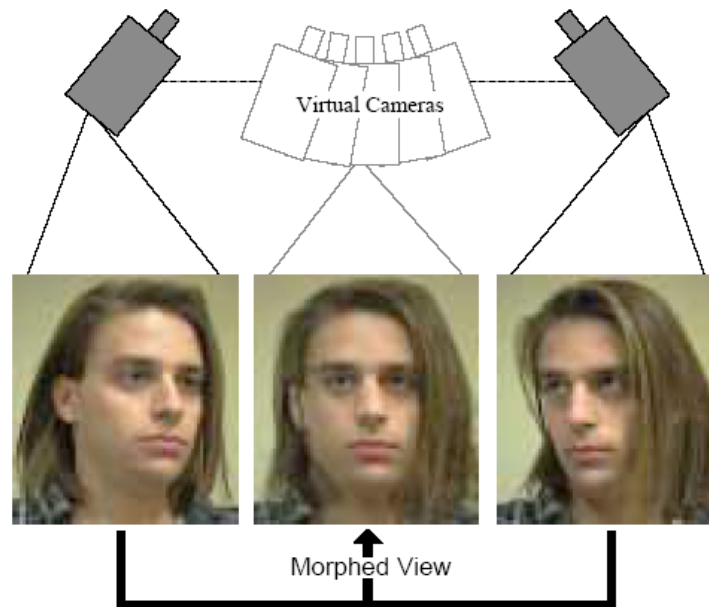- Interpolation consistent with 3D view interpolation

Figure 1: View morphing between two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.

Main trick

- Prewarp with a homography to "pre-align" images
- So that the two views are parallel
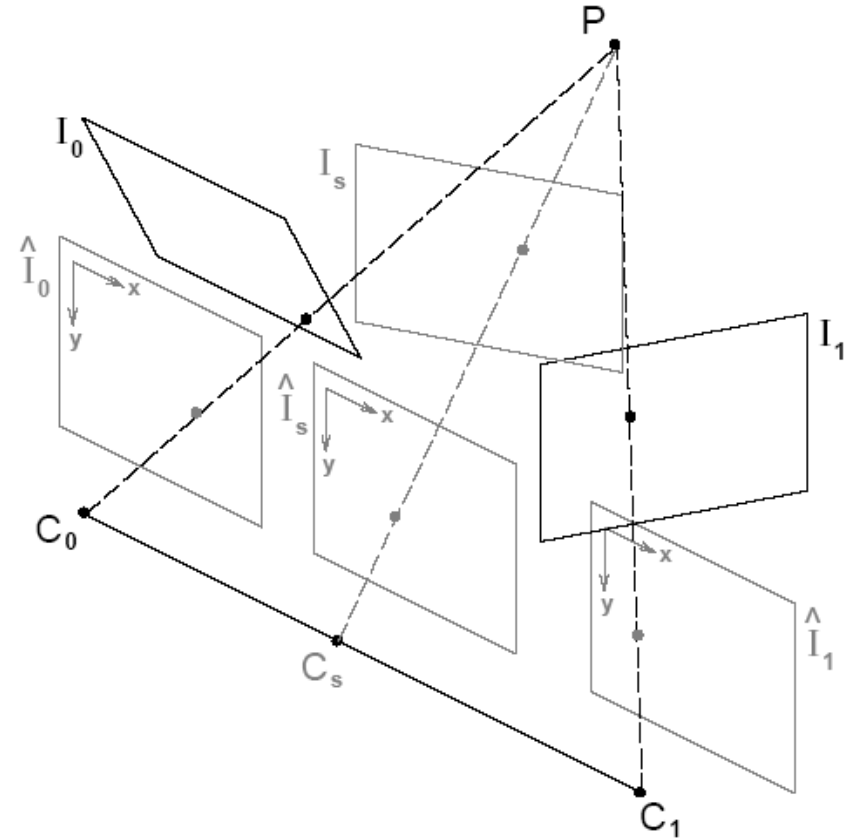 - Because linear interpolation works when views are parallel

Figure 4: View Morphing in Three Steps. (1) Original images $\mathcal{I}_0$ and $\mathcal{I}_1$ are prewarped to form parallel views $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. (2) $\hat{\mathcal{I}}_s$ is produced by morphing (interpolating) the prewarped images. (3) $\hat{\mathcal{I}}_s$ is postwarped to form $\mathcal{I}_s$.
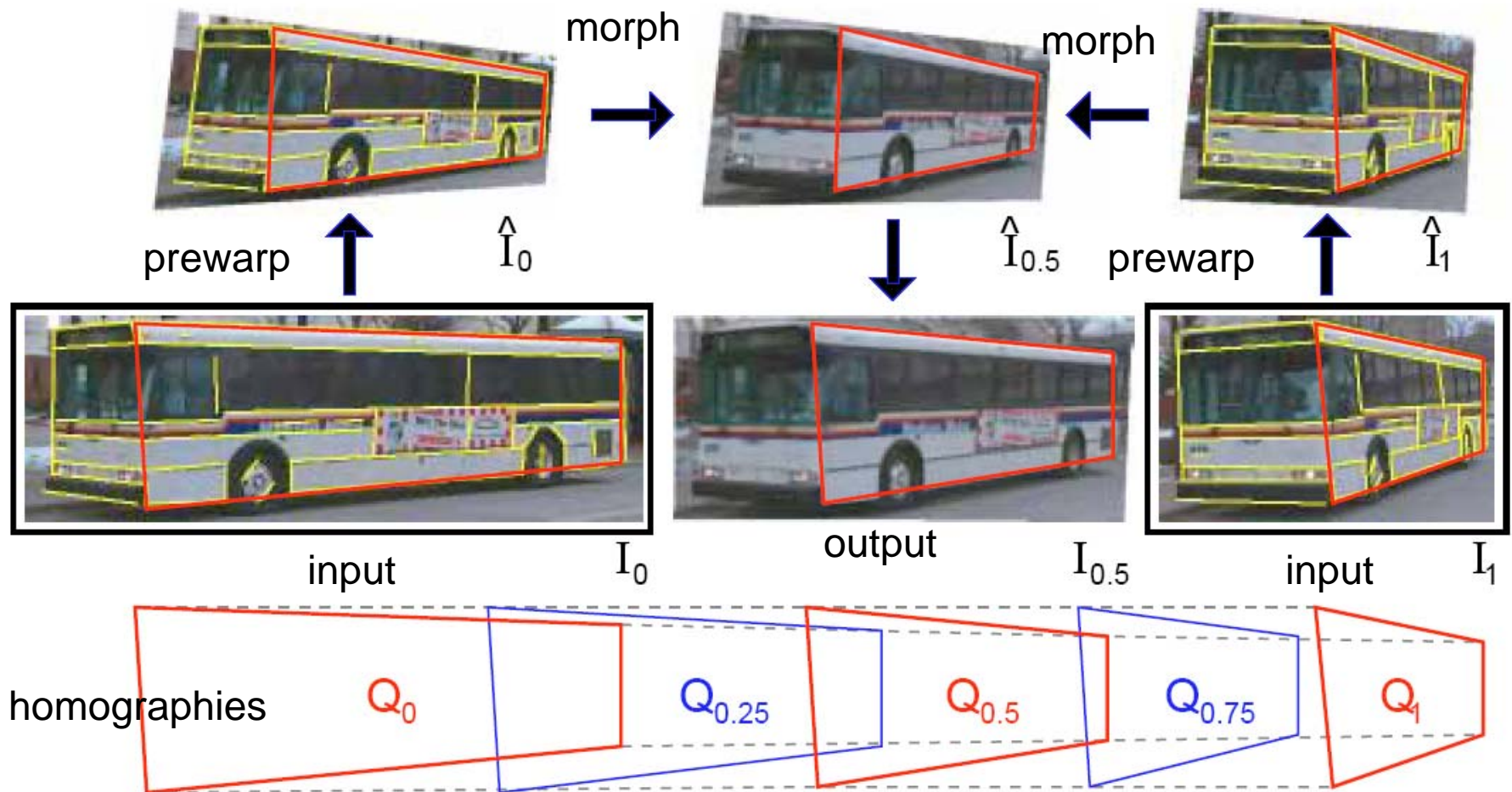
ETH

Figure 6: View Morphing Procedure: A set of features (yellow lines) is selected in original images $\mathcal{I}_0$ and $\mathcal{I}_1$. Using these features, the images are automatically prewarped to produce $\hat{\mathcal{I}}_0$ and $\hat{\mathcal{I}}_1$. The prewarped images are morphed to create a sequence of in-between images, the middle of which, $\hat{\mathcal{I}}_{0.5}$, is shown at top-center. $\hat{\mathcal{I}}_{0.5}$ is interactively postwarped by selecting a quadrilateral region (marked red) and specifying its desired configuration, $Q_{0.5}$, in $\mathcal{I}_{0.5}$. The postwarps for other in-between images are determined by interpolating the quadrilaterals (bottom).
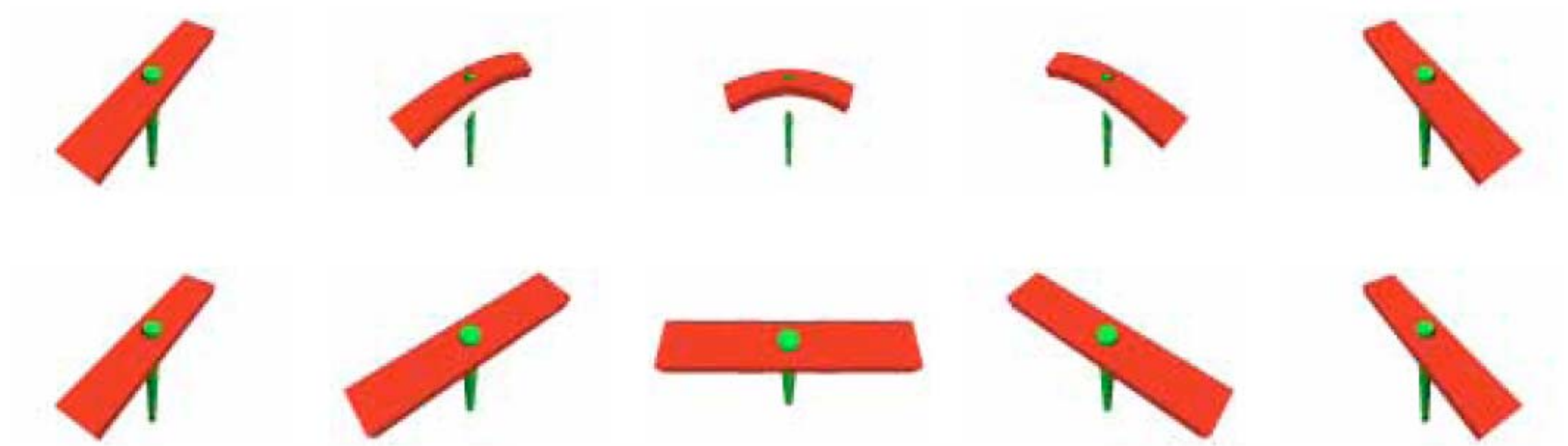
Figure 10: Image Morphing Versus View Morphing. Top: image morph between two views of a helicopter toy causes the in-between images to contract and bend. Bottom: view morph between the same two views results in a physically consistent morph. In this example the image morph also results in an extraneous hole between the blade and the stick. Holes can appear in view morphs as well.

ETH

Figure 9: Mona Lisa View Morph. Morphed view (center) is halfway between original image (left) and it's reflection (right).
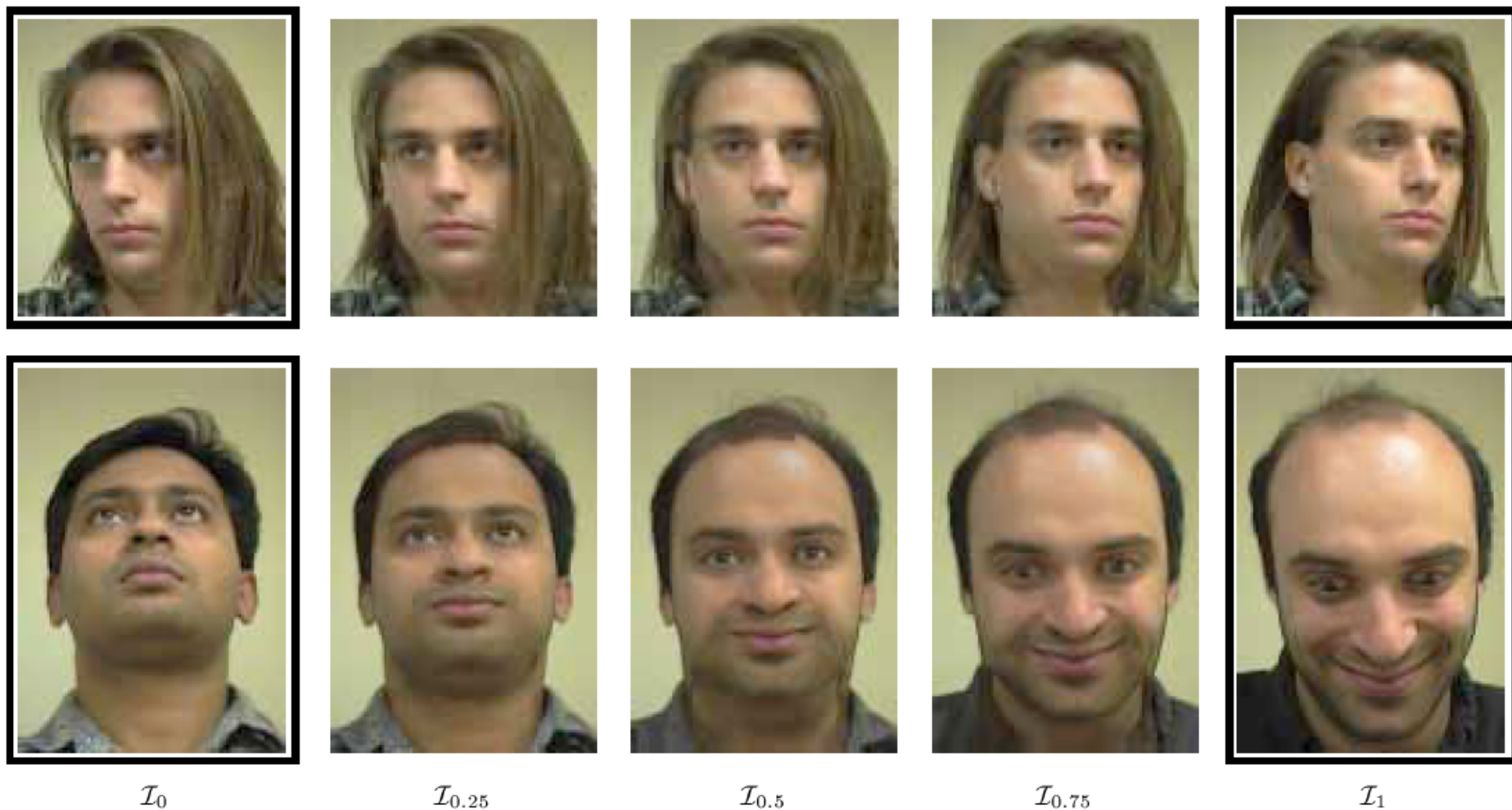
Figure 7: Facial View Morphs. Top: morph between two views of the same person. Bottom: morph between views of two different people. In each case, view morphing captures the change in facial pose between original images $\mathcal{I}_0$ and $\mathcal{I}_1$, conveying a natural 3D rotation.

Next week

- Image pyramids and graph-cuts

512   256   128   64   32   16   8

ETH