Image Warping / Morphing

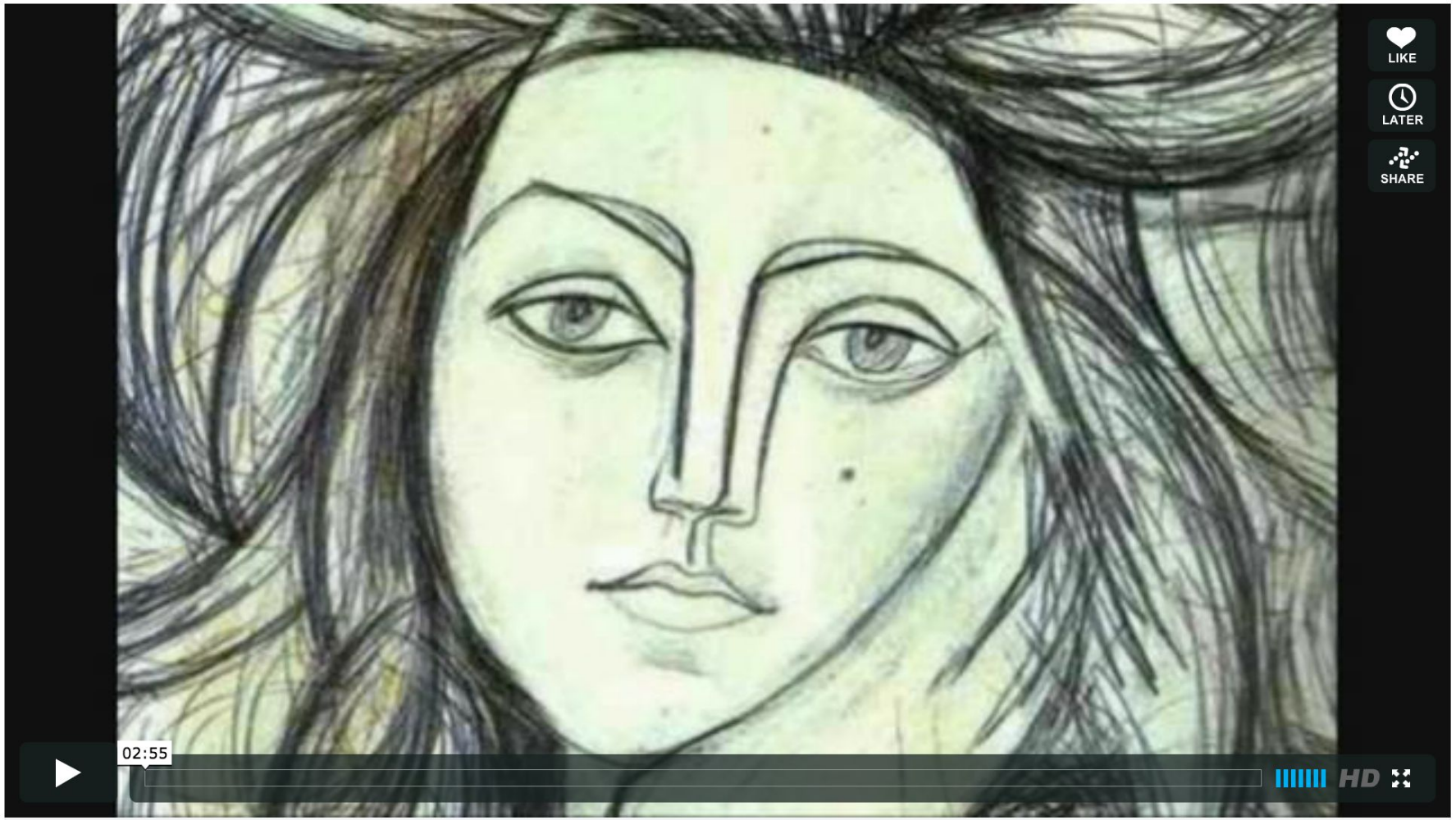
[Wolberg 1996, Recent Advances in Image Morphing]

Computational Photography

Connelly Barnes

Some slides from Fredo Durand, Bill Freeman, James Hays

Morphing Video: Women in Art



- http://www.vimeo.com/1456037

Terminator 2 Morphing (1991)

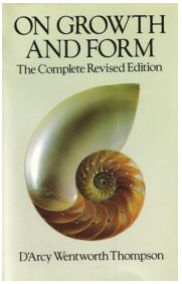
Terminator 2 Clip (YouTube)

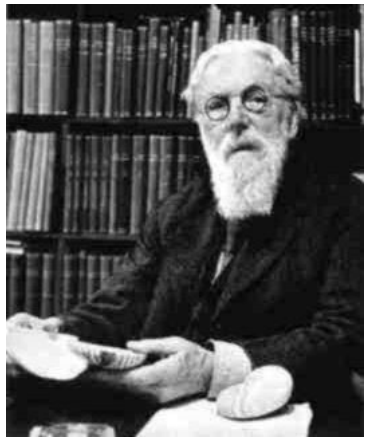
Image Warping in Biology

- D'Arcy Thompson

  http://en.wikipedia.org/wiki/D'Arcy_Thompson

- Importance of shape and structure in evolution





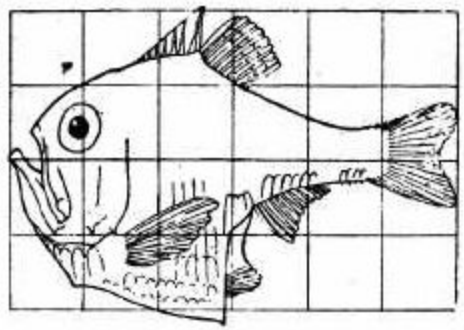


Fig. 517. *Argyropelecus Olfersi.*

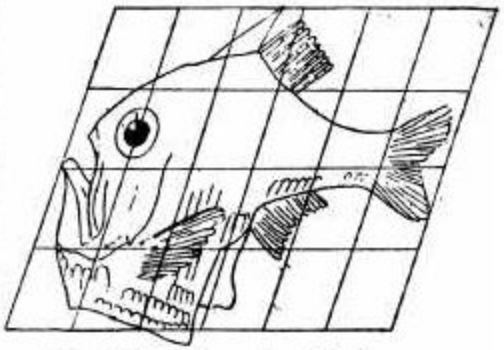
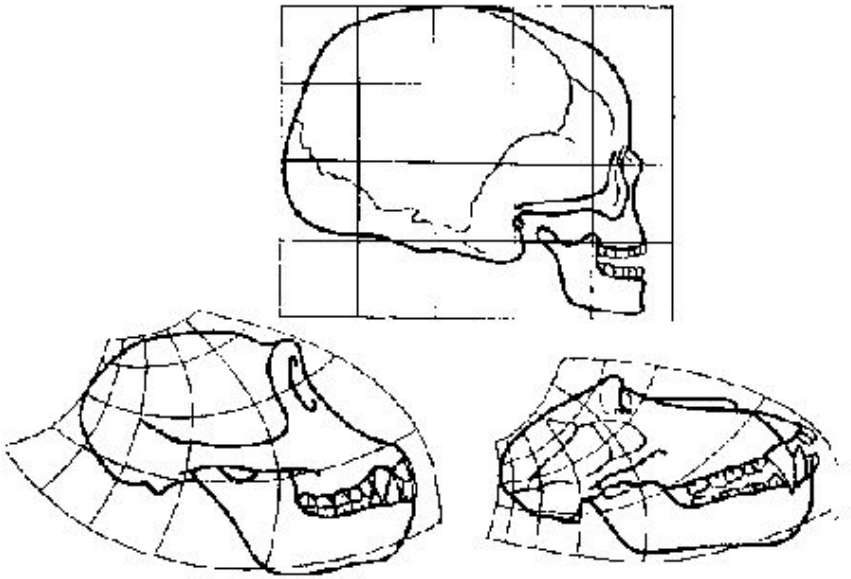


Fig. 518. *Sternoptyx diaphana.*



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

Slide by Durand and Freeman

Cambrian Explosion

Opabinia

Hallucigenia

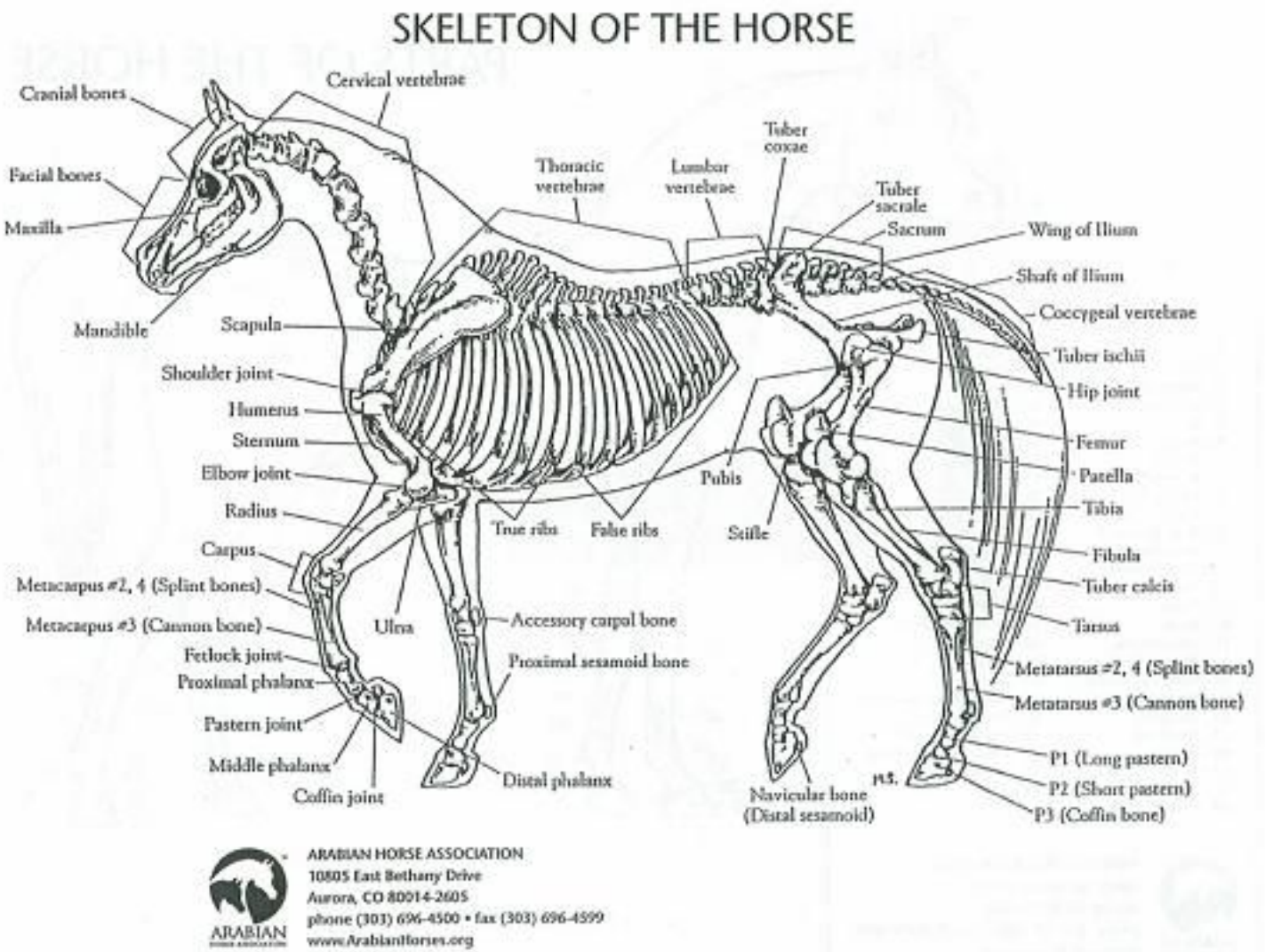
Pikaia

Marrella

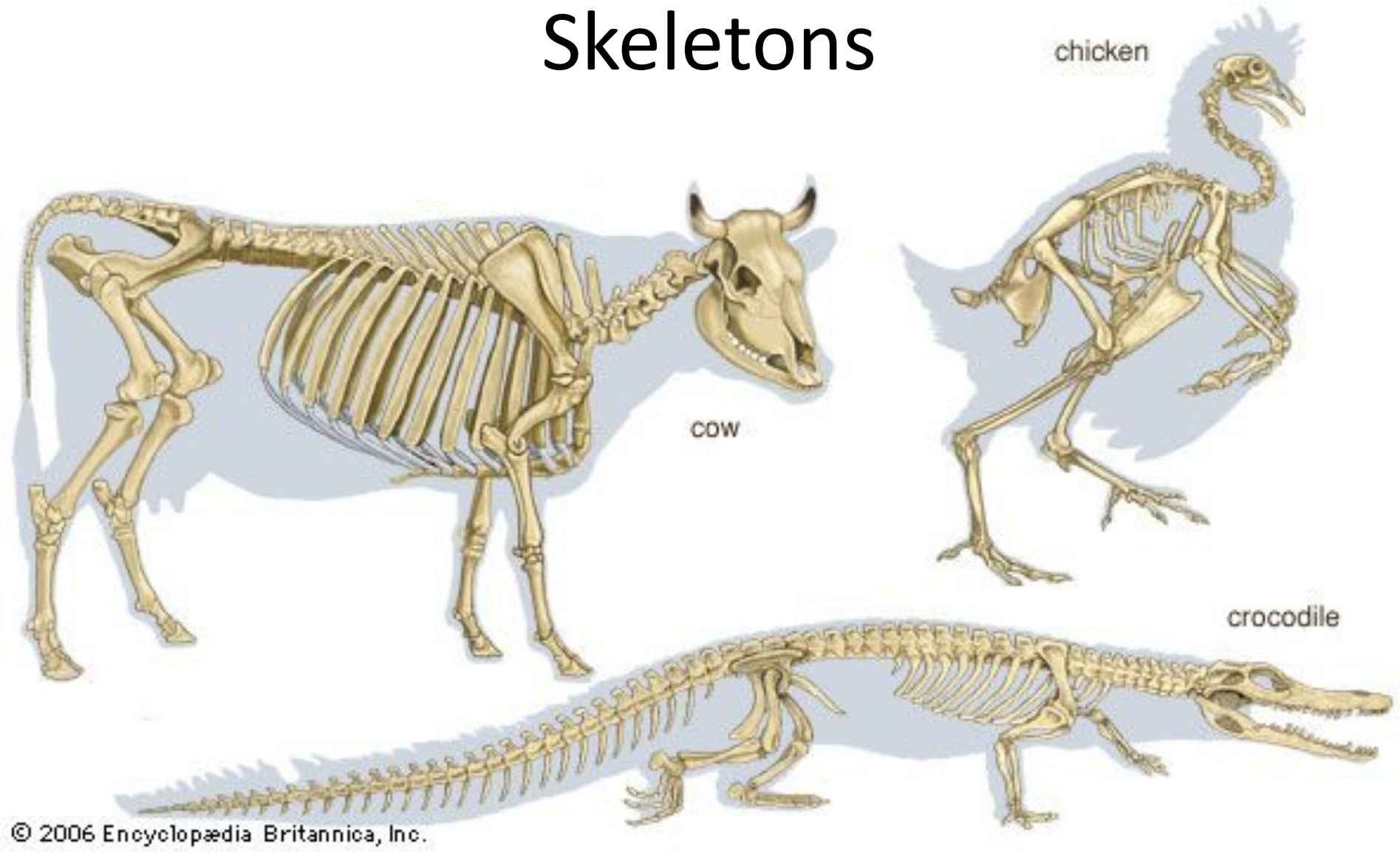
Aysheaia

Source: http://www.earthlearningidea.com/

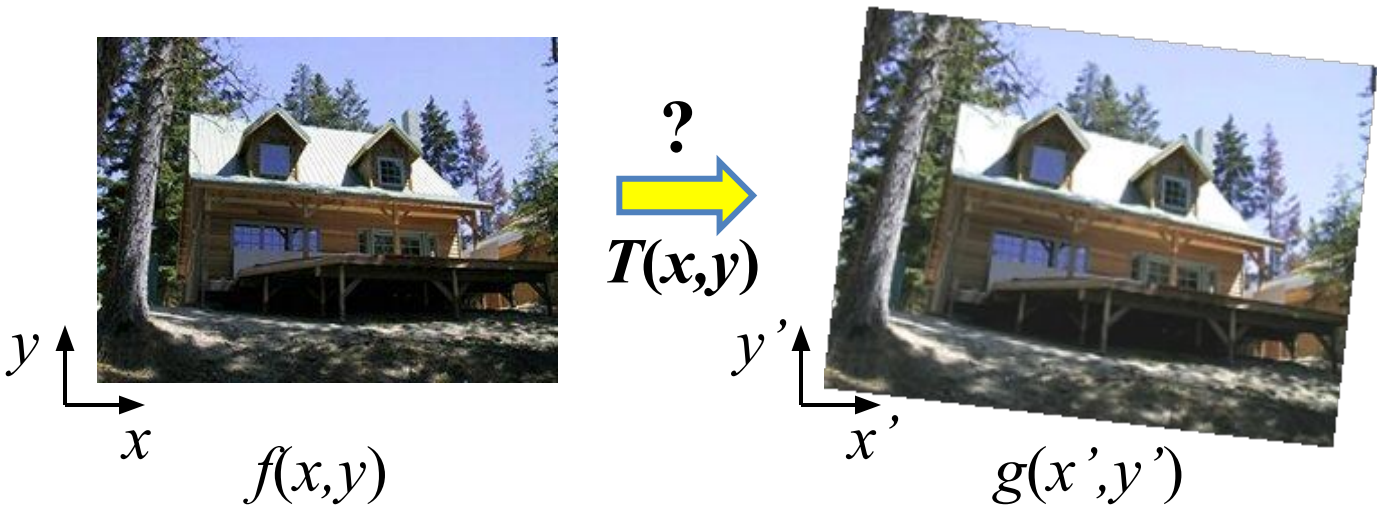
Skeletons


SKELETON OF THE HORSE
Cranial bones
Cervical vertebrae
Facial bones
Maxilla
Mandible
Scapula
Shoulder joint
Humerus
Sternum
Elbow joint
Radius
Carpus
Metacarpus #2, 4 (Splint bones)
Metacarpus #3 (Cannon bone)
Fetlock joint
Proximal phalanx
Pastern joint
Middle phalanx
Coffin joint
Thoracic vertebrae
Lumbar vertebrae
Tuber coxae
Tuber sacrale
Sacrum
Wing of Ilium
Shaft of Ilium
Coccygeal vertebrae
Tuber ischii
Hip joint
Femur
Patella
Tibia
Fibula
Tuber calcis
Tarsus
Metatarsus #2, 4 (Splint bones)
Metatarsus #3 (Cannon bone)
P1 (Long pastern)
P2 (Short pastern)
P3 (Coffin bone)
Pubis
Stifle
True ribs
False ribs
Ulna
Accessory carpal bone
Proximal sesamoid bone
Distal phalanx
Navicular bone
(Distal sesamoid)
ARABIAN HORSE ASSOCIATION
10805 East Bethany Drive
Aurora, CO 80014-2605
phone (303) 696-4500 • fax (303) 696-4599
www.ArabianHorses.org

Skeletons

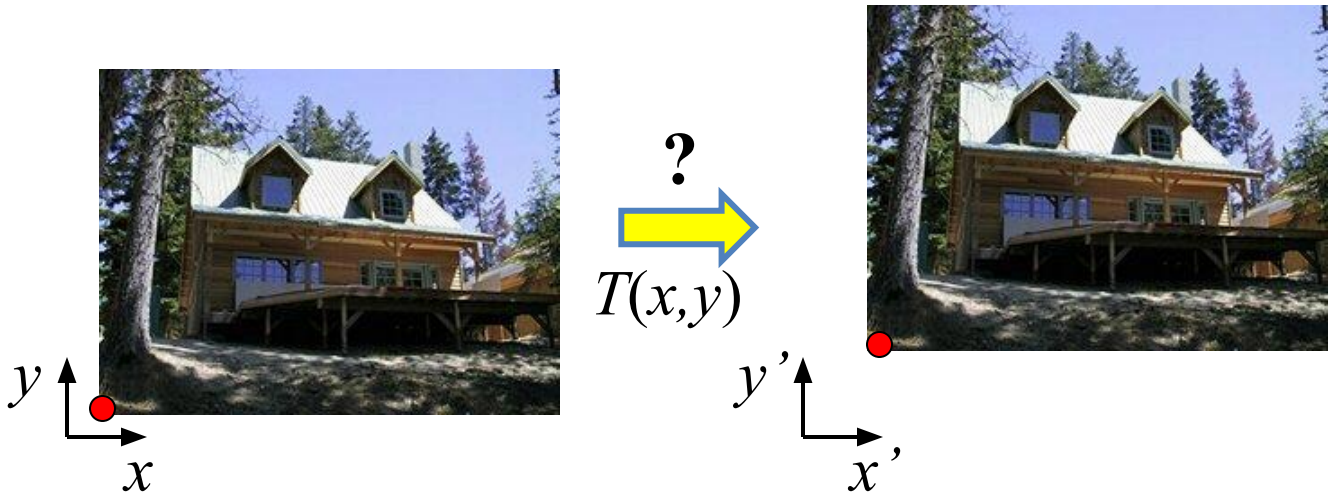


Recovering Transformations



- What if we know *f* and *g* and want to recover the transform T?
  - e.g. better align photographs you've taken
  - willing to let user provide correspondences
    - How many do we need?

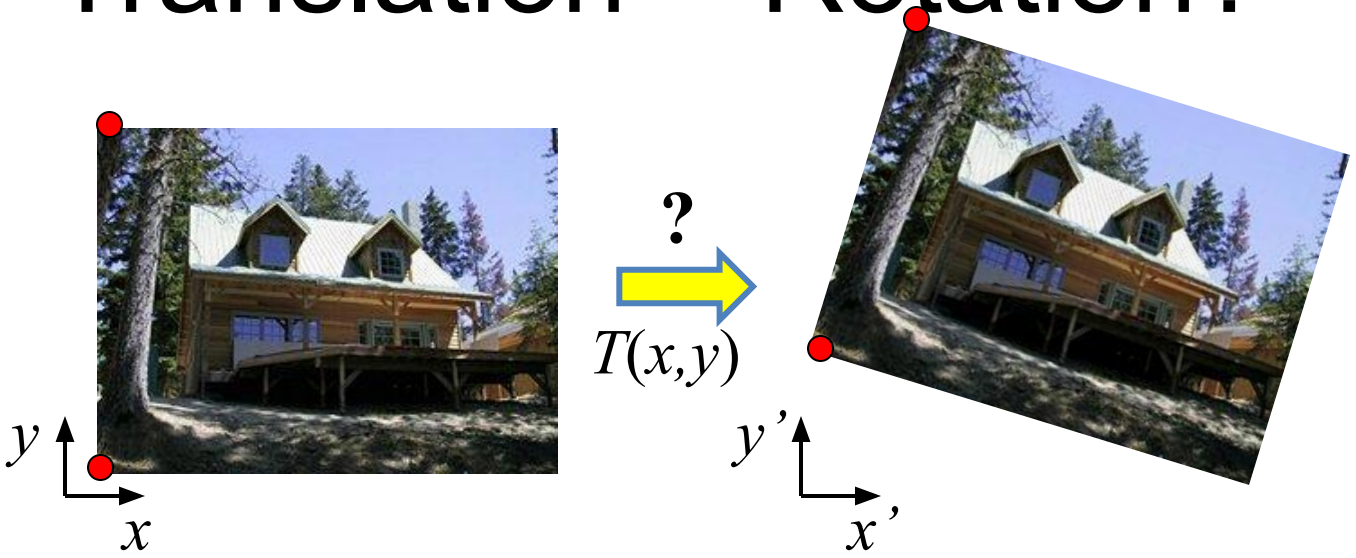
Translation: # correspondences?



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

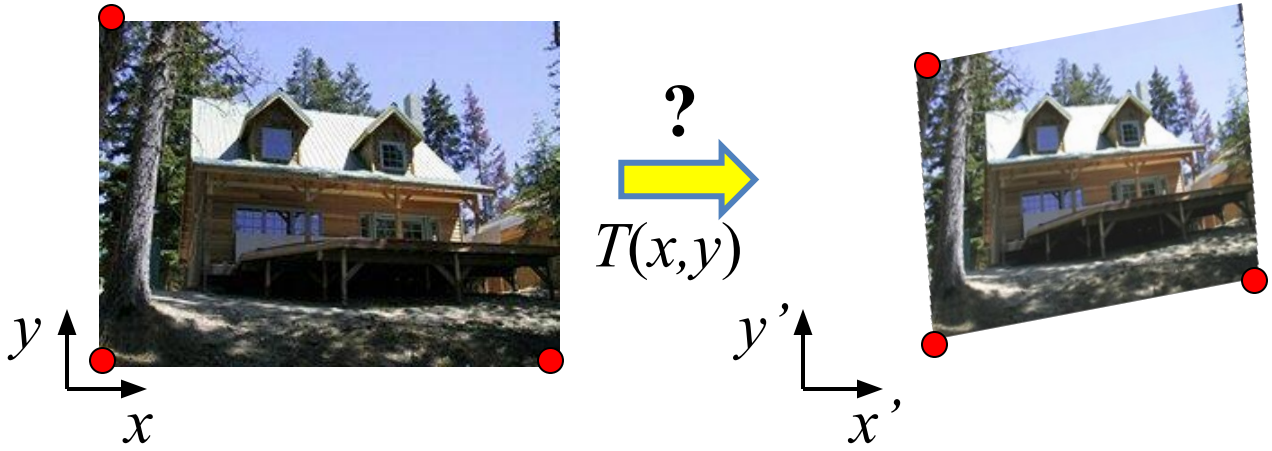
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translation + Rotation?



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

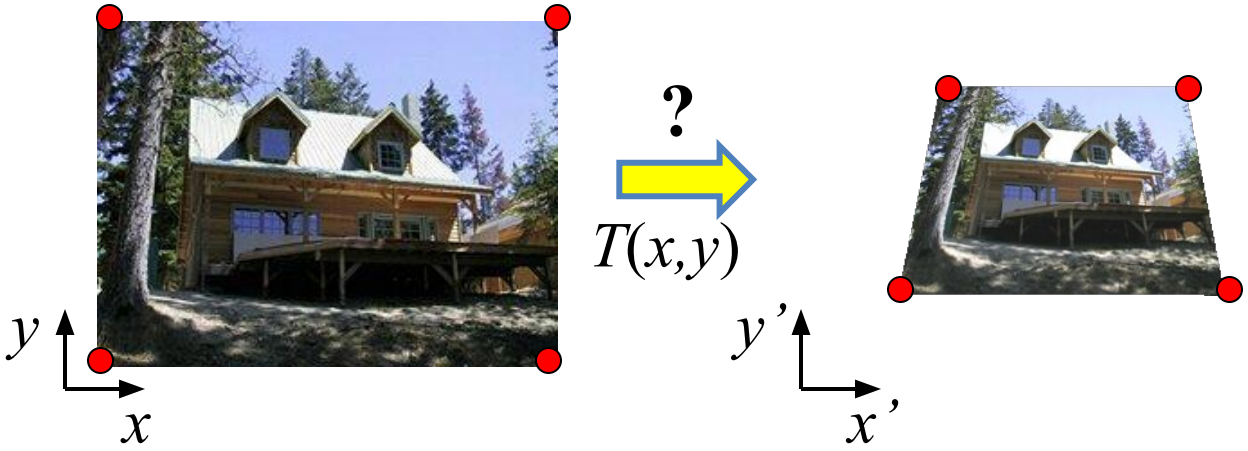
Affine: # correspondences?



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

$$T(x,y) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

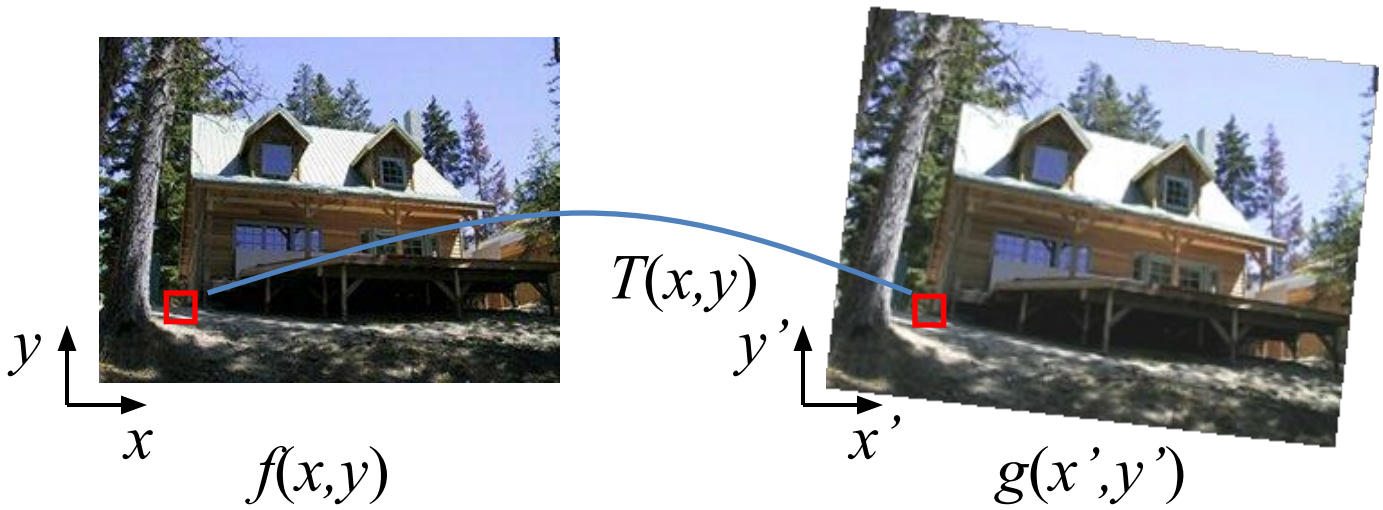
Projective / Homography



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

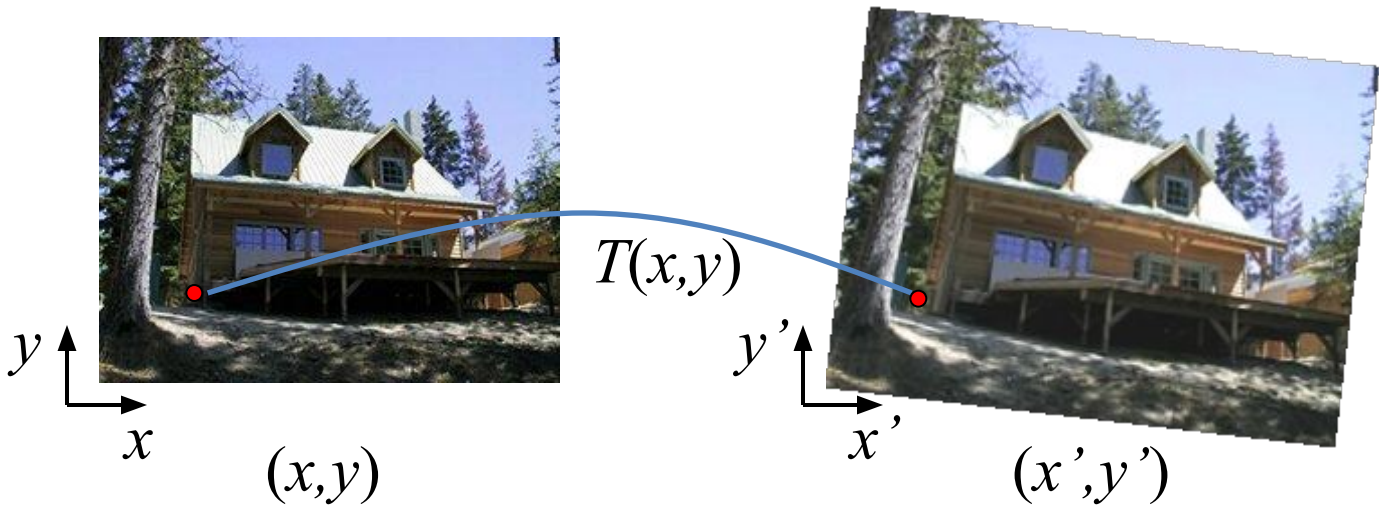
$$T(x,y) = h\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right) \qquad h(x,y,z) = (x/z, y/z)$$

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*))?

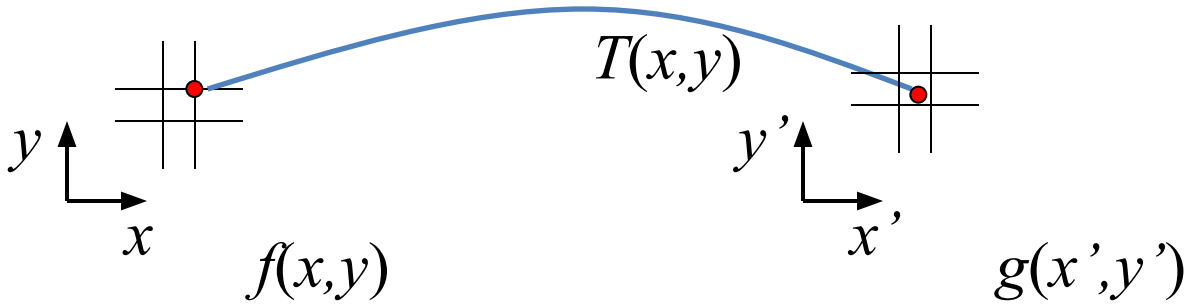
Forward warping



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

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

Forward warping

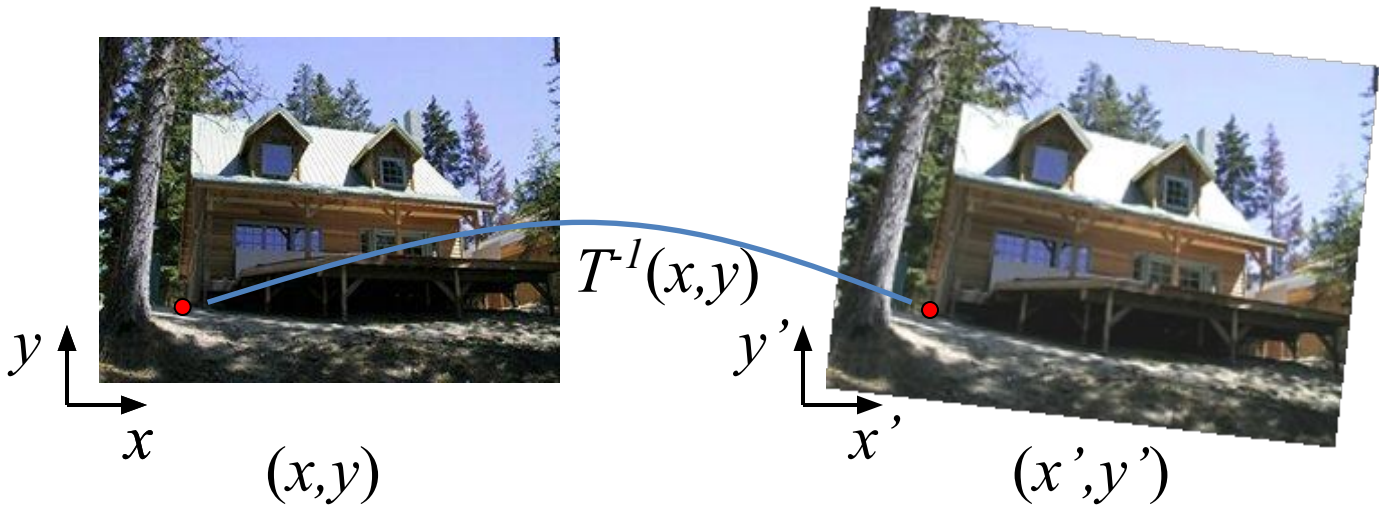


Q: what if pixel lands "between" two pixels?

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

- Known as "splatting"
- Can also interpolate points in target image: `griddata` (Matlab), `scipy.interpolate.griddata` (Python)

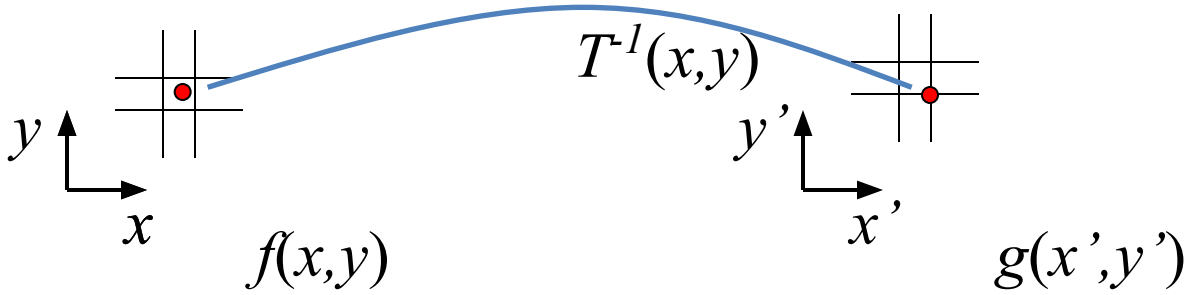
Inverse warping



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

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

Inverse warping



Q: what if pixel comes from "between" two pixels?

A: *Interpolate* color value from neighbors

– nearest neighbor, bilinear, Gaussian, bicubic

– See `interp2` (Matlab), `scipy.interpolate.interp2d` (Python)

Forward vs. inverse warping

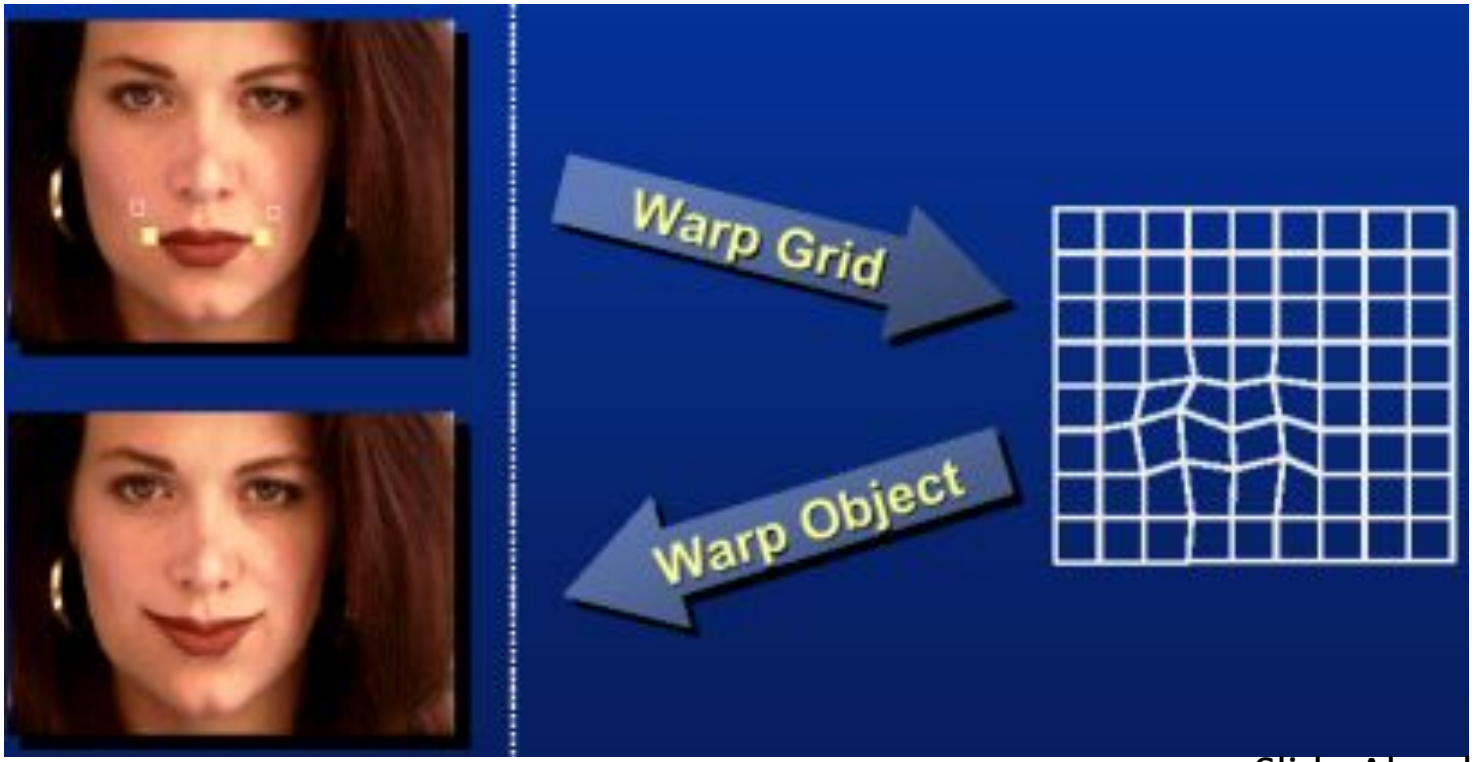
- Q: Which is better?

Forward vs. inverse warping

- Q: Which is better?
- A: Usually inverse – eliminates holes
  - However, it requires an invertible warp function
  - Not always possible

How to Obtain Warp Field?

- Move control points to specify a spline warp
- Spline produces a smooth vector field $T(x, y)$



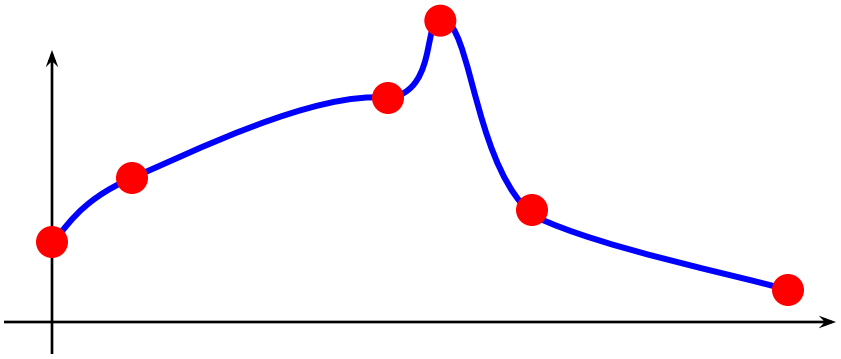
Slide Alyosha Efros

Warp as Interpolation

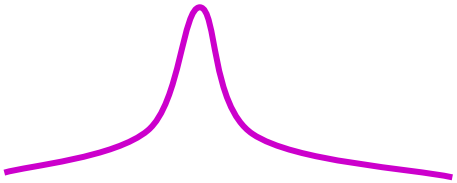
- **We are looking for a warping field**
  - A function that given a 2D point, returns a warped 2D point
- **We have a sparse number of correspondences**
  - These specify values of the warping field
- **This is an interpolation problem**
  - Given sparse data, find smooth function

Interpolation in 1D

- We are looking for a function f
- We have N data points: x_i, y_i
  - Scattered: spacing between x_i is non-uniform
- We want f so that
  - For each i, $f(x_i)=y_i$
  - f is smooth
- Depending on notion of smoothness, different f

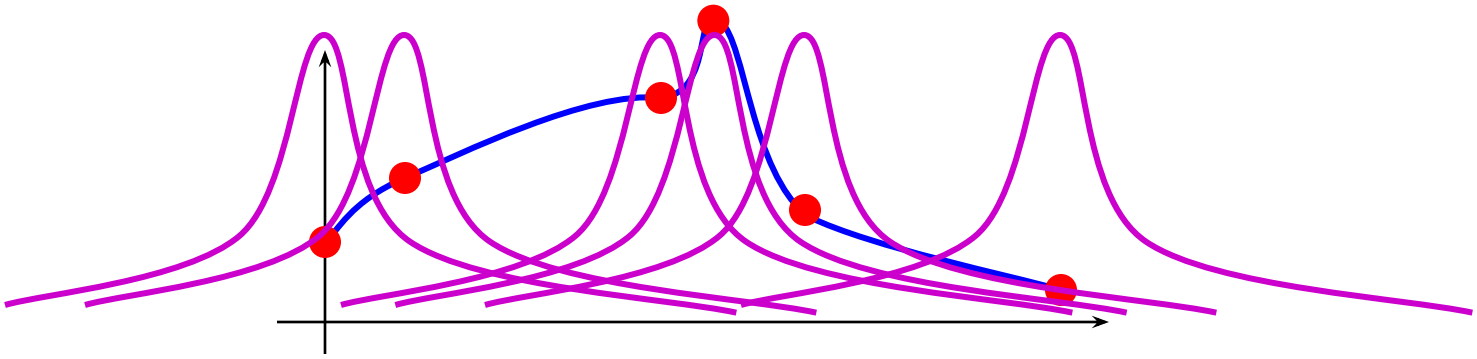


Radial Basis Functions (RBF)

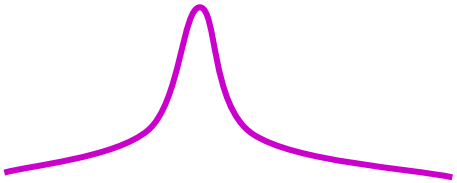


- Place a smooth kernel R centered on each data point x_i

$$f(z) = \sum \alpha_i R\left(\left\| z - x_i \right\|\right)$$



Radial Basis Functions (RBF)

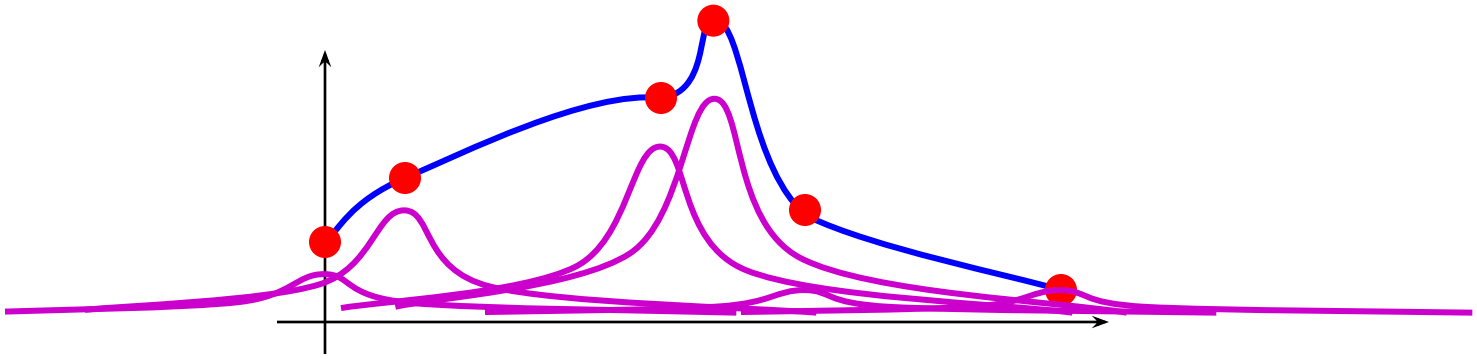


- Place a smooth kernel *R* centered on each data point x_i

$$f(z) = \sum \alpha_i R\left(\left\| z - x_i \right\|\right)$$

- Find weights α_i to make sure we interpolate the data

for each i, $f(x_i)=y_i$



Radial Basis Function Kernels

Linear $R(r) = r$

Cubic $R(r) = r^3$

Quintic $R(r) = r^5$

Thin plate $R(r) = r^2 \log r$

Inverse $R(r) = 1 / \sqrt{(r/w)^2 + 1}$

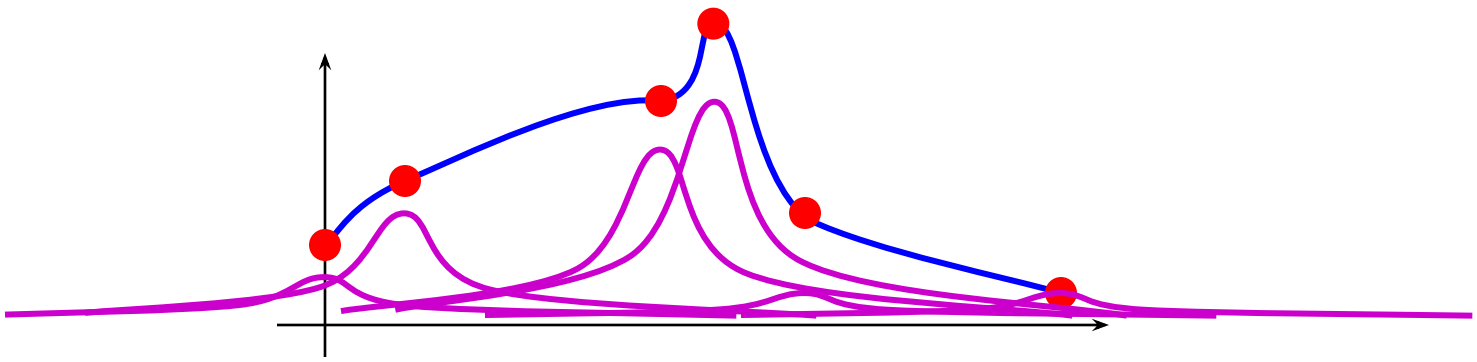
Multiquadratic $R(r) = \sqrt{(r/w)^2 + 1}$

Solve RBF Interpolation Problem

$$f(z) = \sum \alpha_i R\left(\left\| z - x_i \right\|\right)$$

For each *j*, $\sum \alpha_i R\left(\left\| x_j - x_i \right\|\right) = y_j$

- In 1D: *N* equations, *N* unknowns, linear solver.
- In *n*-D: Denote $\boldsymbol{\alpha}_i, \mathbf{x}_i, \mathbf{y}_i \in \mathbf{R}^m$
  Solve *Nm* equations in *Nm* unknowns $\boldsymbol{\alpha}_i$.



RBF Summary

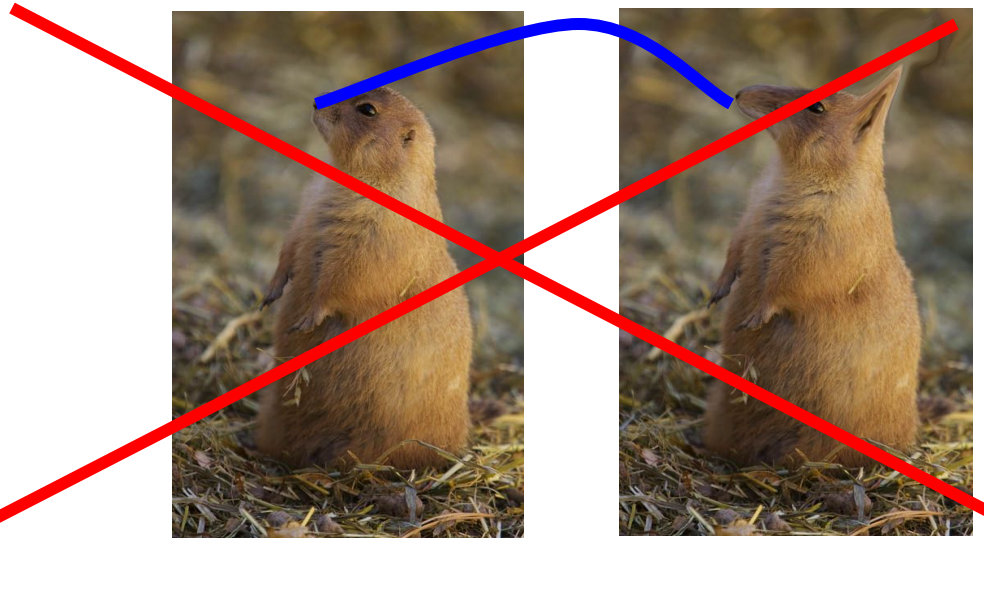
- Interpolates “scattered data”, or data defined only at a few sparse locations.
- Basis functions have infinite extent…

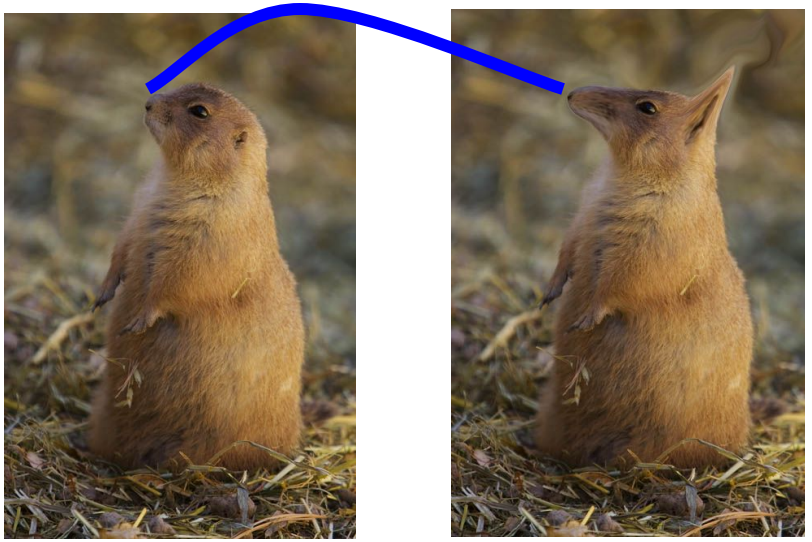
- Python: scipy.interpolate.Rbf

- MATLAB: Google “matlab rbf interpolation” (3rd party code)

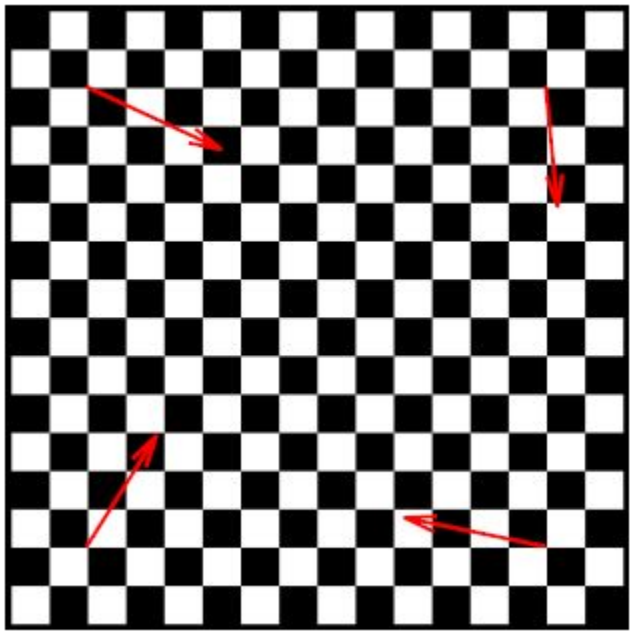
Applying a warp: use inverse

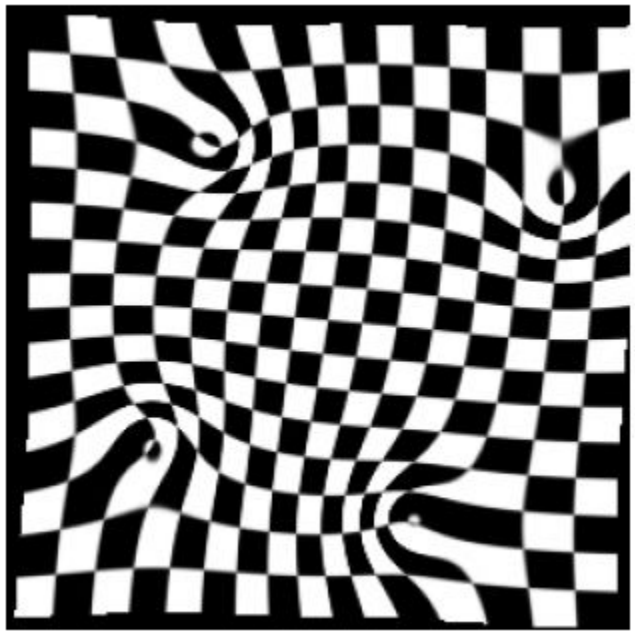
- Forward warp:
  - For each pixel in **input** image
    - Paste color **to warped** location in output
  - Problem: gaps
- Inverse warp
  - For each pixel in **output** image
    - Lookup color **from inverse-warped** location





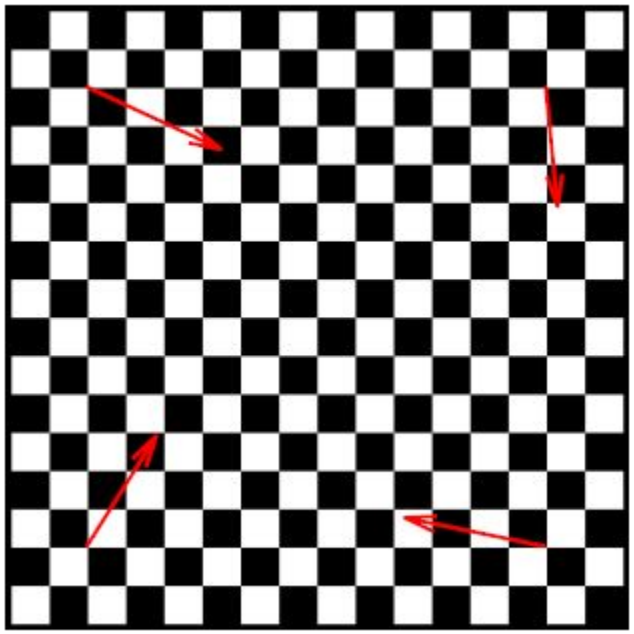
Example

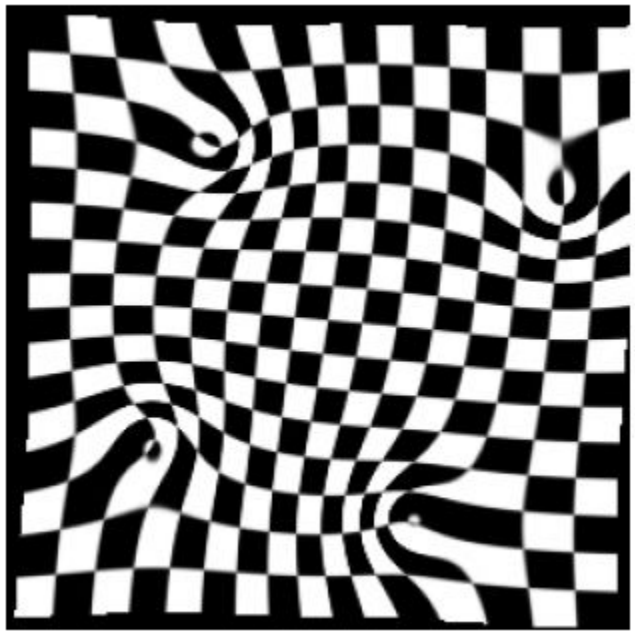




Example

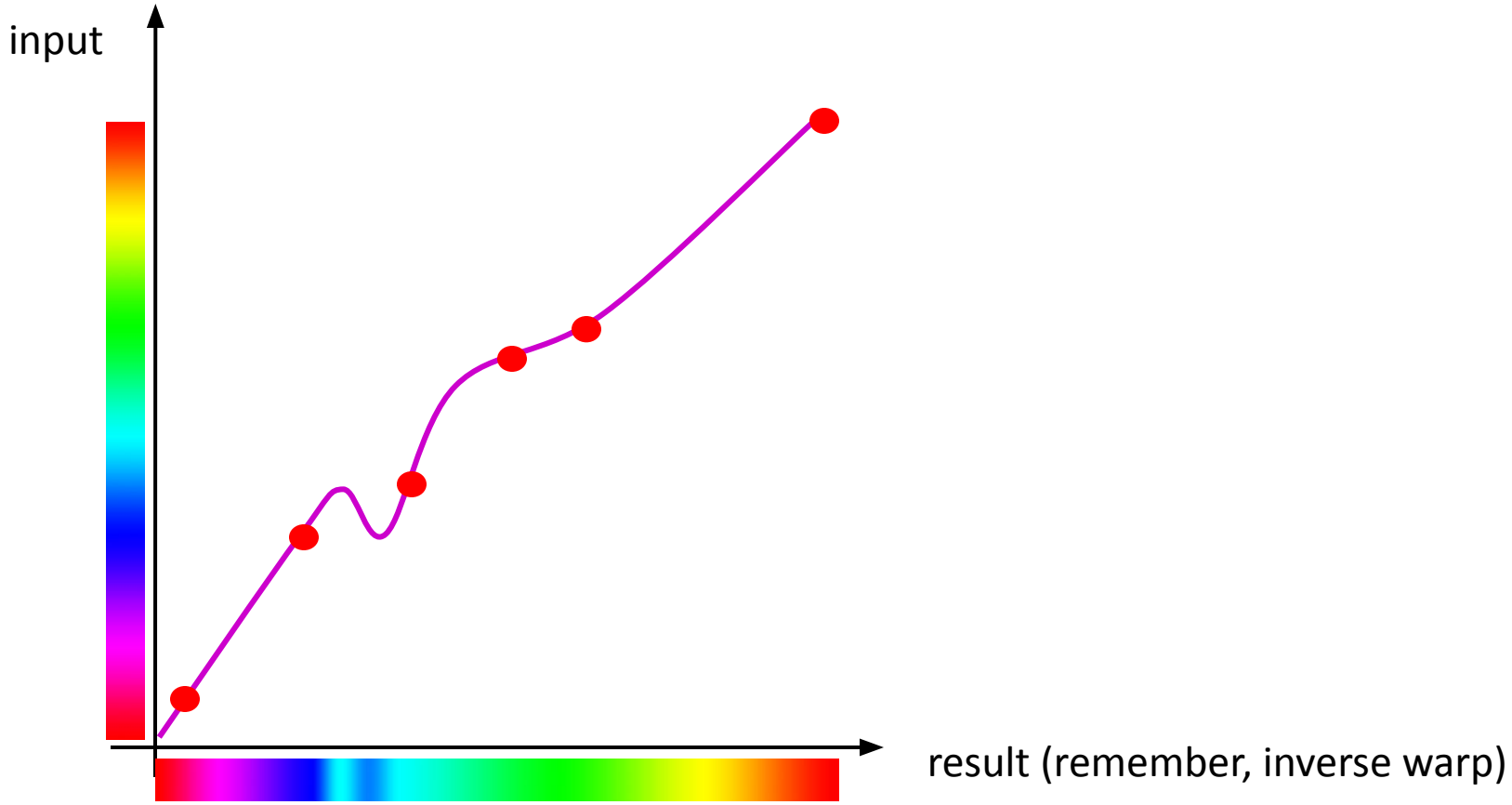
- Fold problems
  - Oh well…





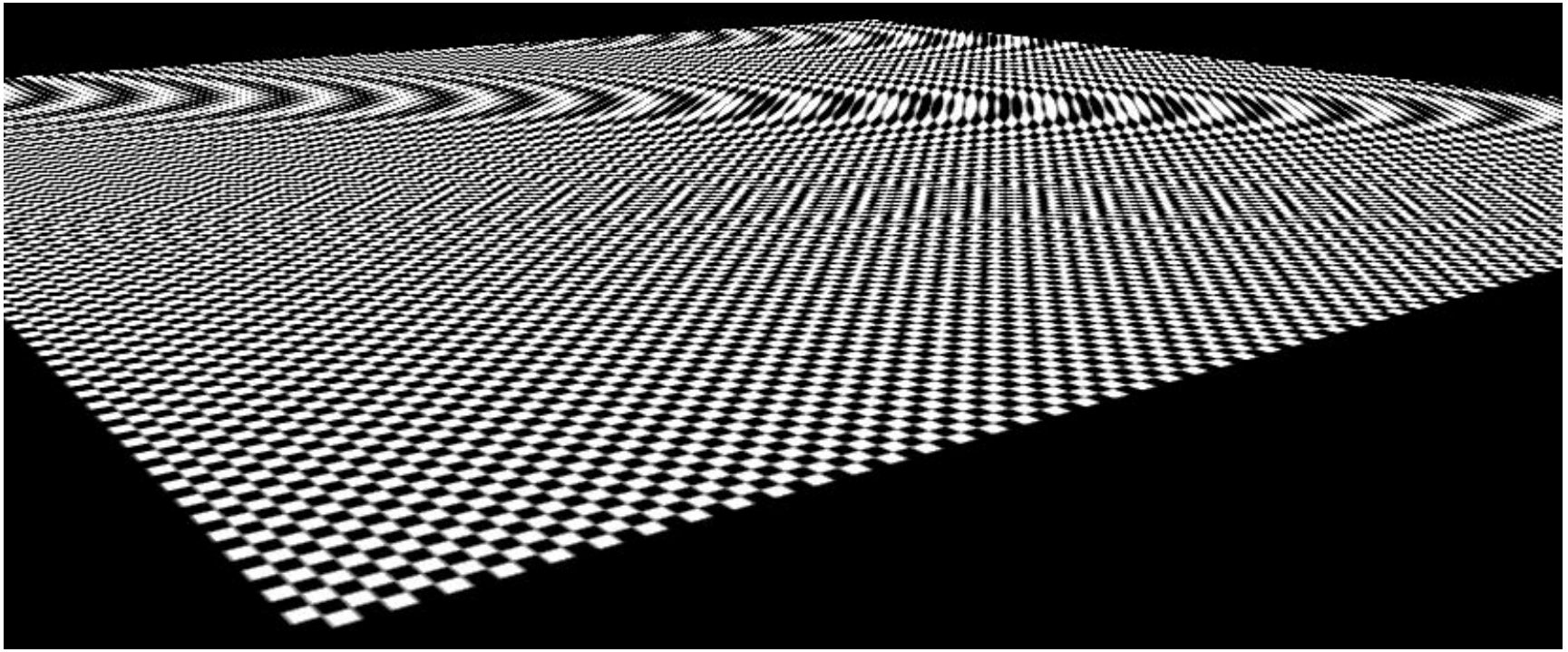
1D equivalent of folds

- No guarantee that our 1D RBF is monotonic



Aliasing Issues with Warping

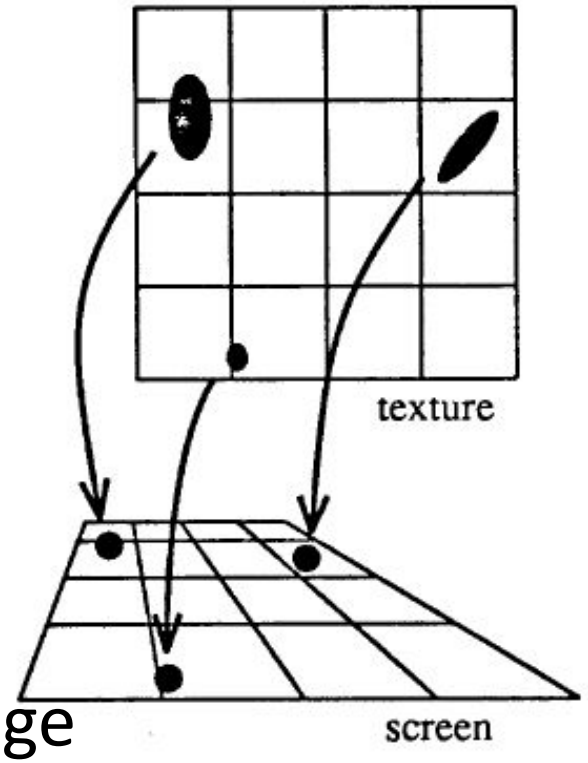
- *Aliasing* can happen if warps are extreme. This is especially noticeable during animation.



Aliasing Solution

- Use an ellipsoidal Gaussian:

$$G(x,y) = G_{\sigma_1}(x)G_{\sigma_2}(y)$$



- "Elliptical Weighted Average" (EWA)
- Filter is deformed based on warping.
- For inverse warping, each output (warped) pixel does a weighted average of nearby pixels against the filter.
- Can approximate with circular Gaussian.

[Paul Heckbert Master's Thesis]

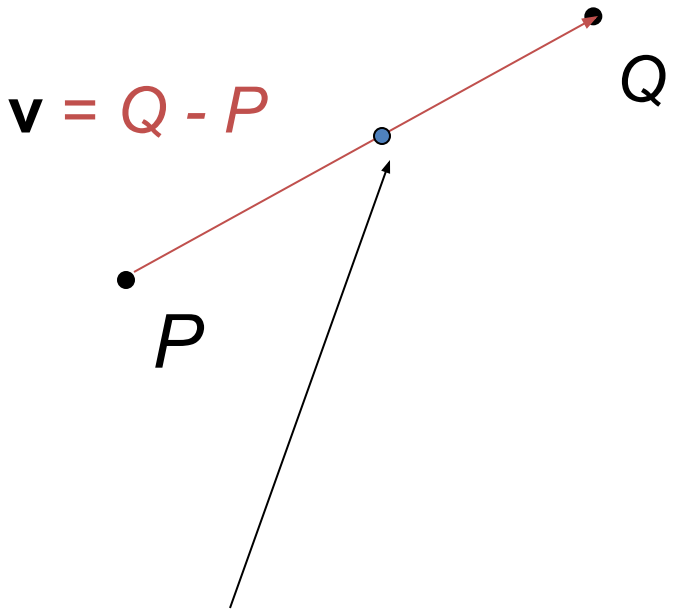
Morphing = Object Averaging



- 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

Linear Interpolation

How can we linearly transition between point *P* and point *Q*?



$P + t\,\mathbf{v}$

$= (1-t)P + tQ,$ e.g. t = 0.5

- 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

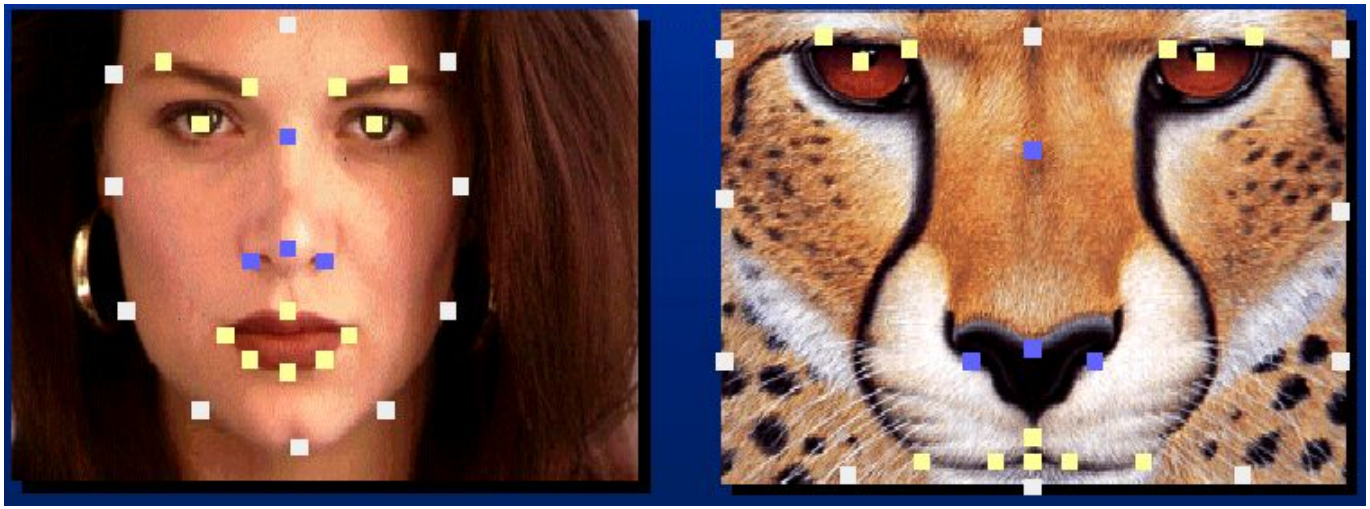
- Interpolate whole images:
- $Image_{halfway} = (1-t)*Image_1 + t*image_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #2: Align, then cross-disolve

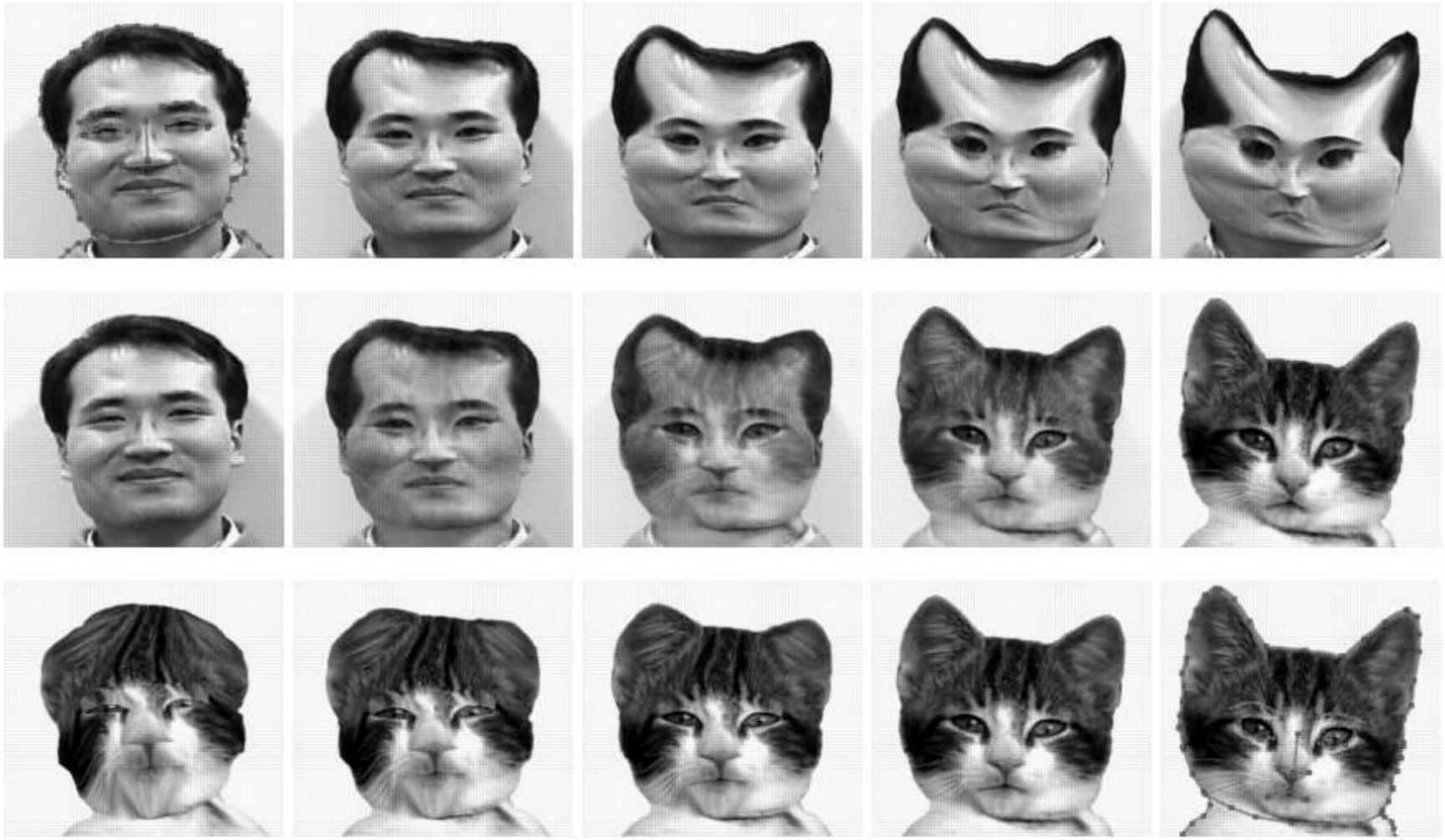
- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Full Morphing

A  B

- What if there is no simple global function that aligns two images?
- User specifies corresponding feature points
- Construct warp animations A -> B and B -> A
- Cross dissolve these

Full Morphing



Full Morphing

Image A

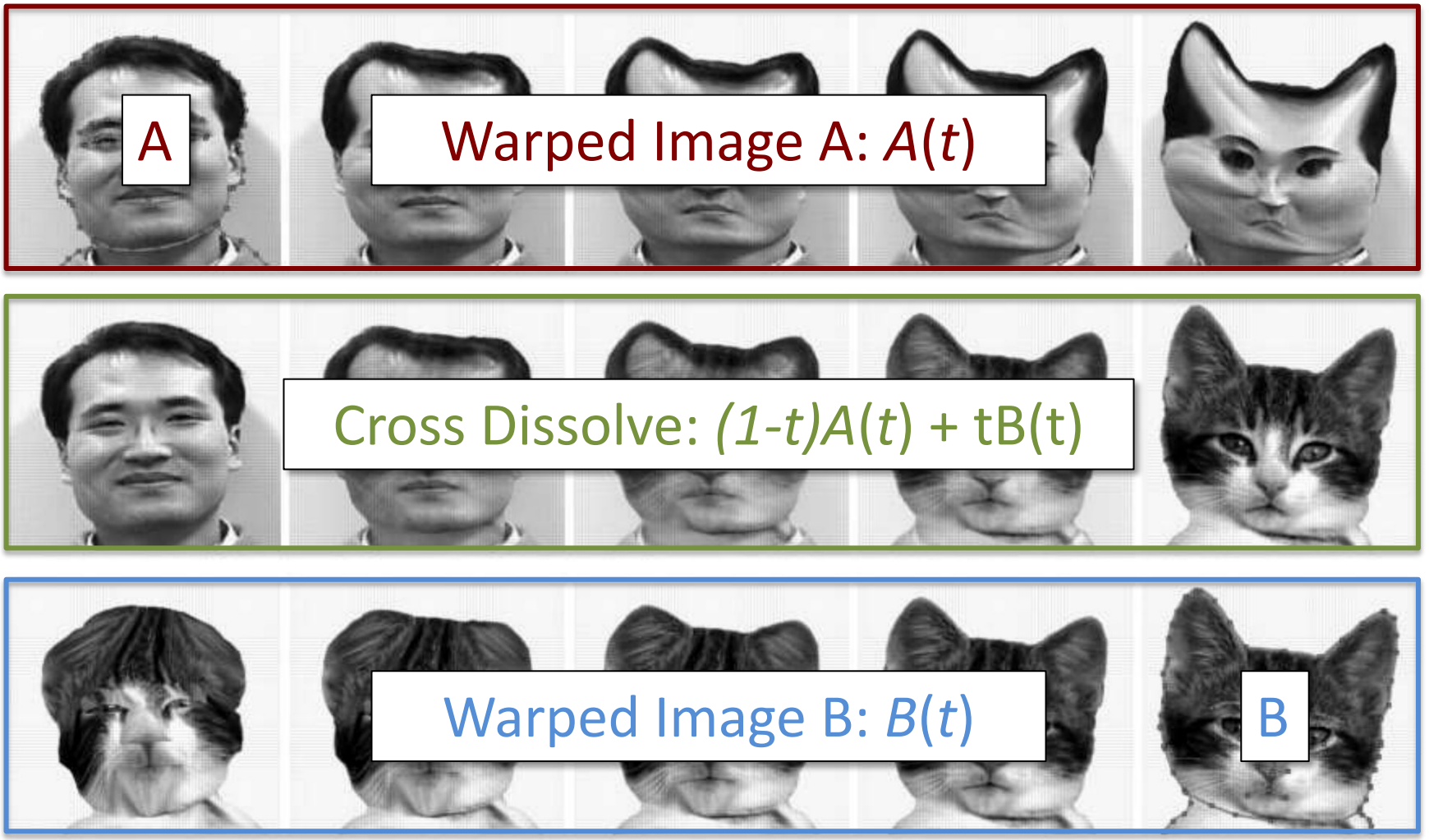
Image B

1. Find warping fields from user constraints (points or lines):
   Warp field $T_{AB}(x, y)$ that maps A pixel to B pixel
   Warp field $T_{BA}(x, y)$ that maps B pixel to A pixel

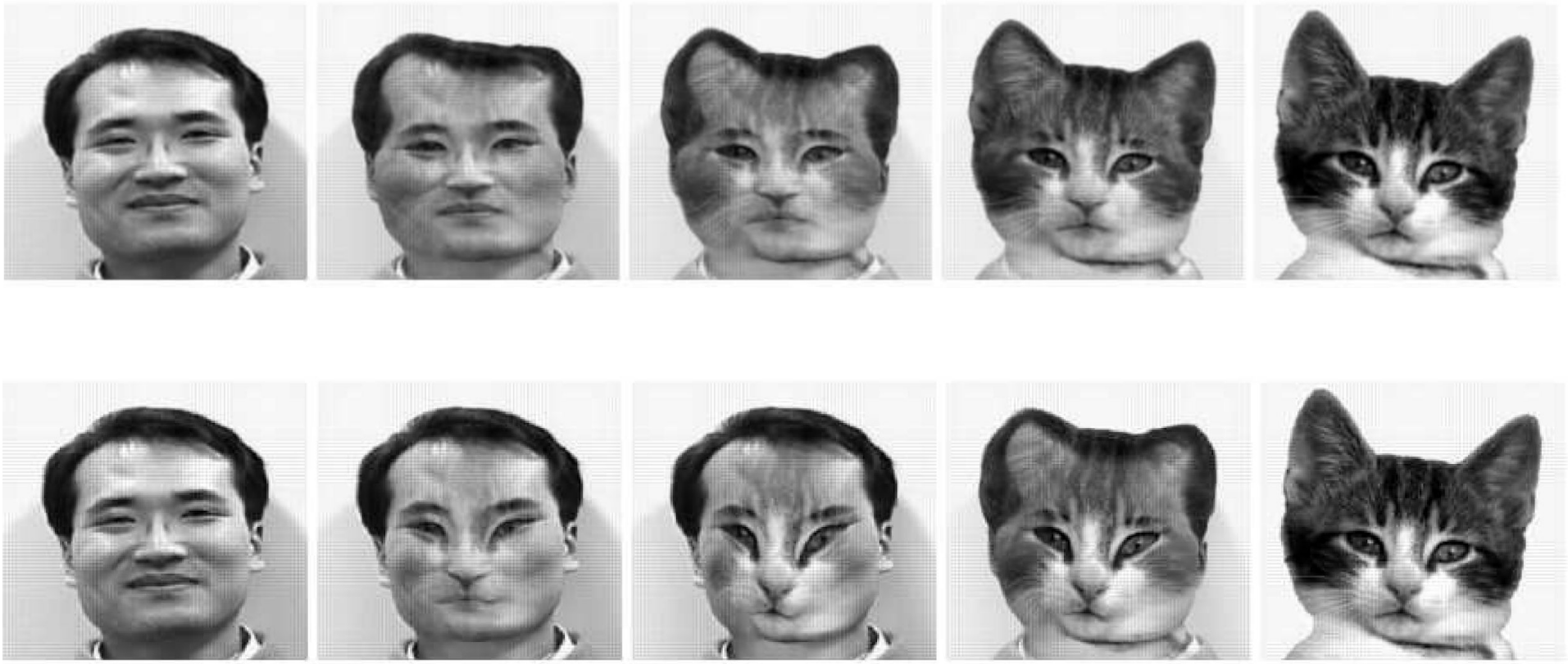
2. Make video $A(t)$ that warps A over time to the shape of B
   Start warp field at identity and linearly interpolate to T_{BA}
   Construct video $B(t)$ that warps B over time to shape of A

3. Cross dissolve these two videos.

Full Morphing



Catman!



Conclusion

- Illustrates general principle in graphics:
  - First register, then blend
- Avoids ghosting

Michael Jackson - Black or White