Geometric Transformations and Image Warping Chapter 2.6.5

Ross Whitaker

(modified by Guido Gerig)

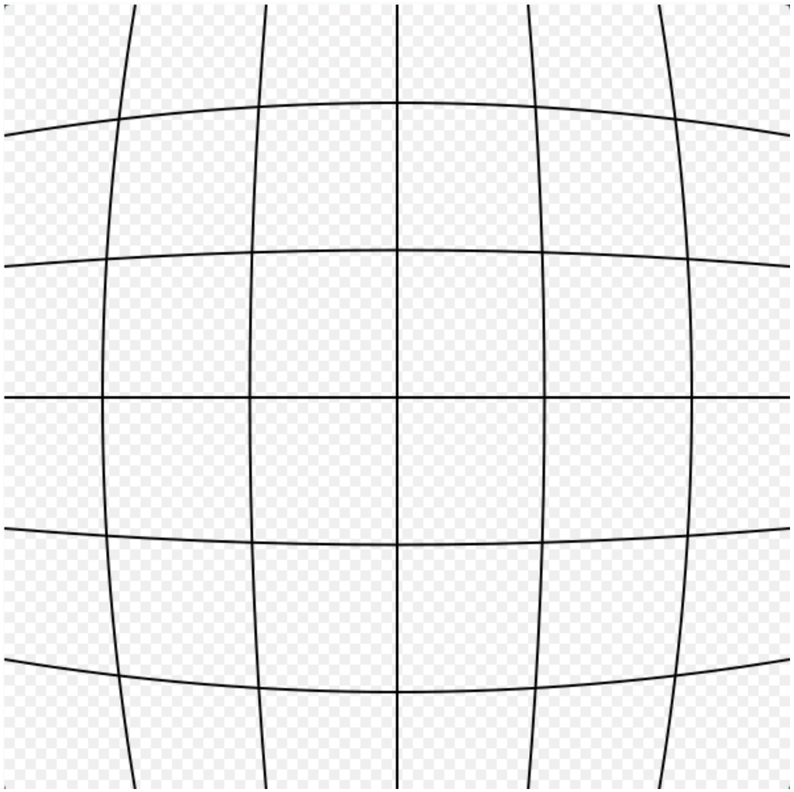
SCI Institute, School of Computing

University of Utah

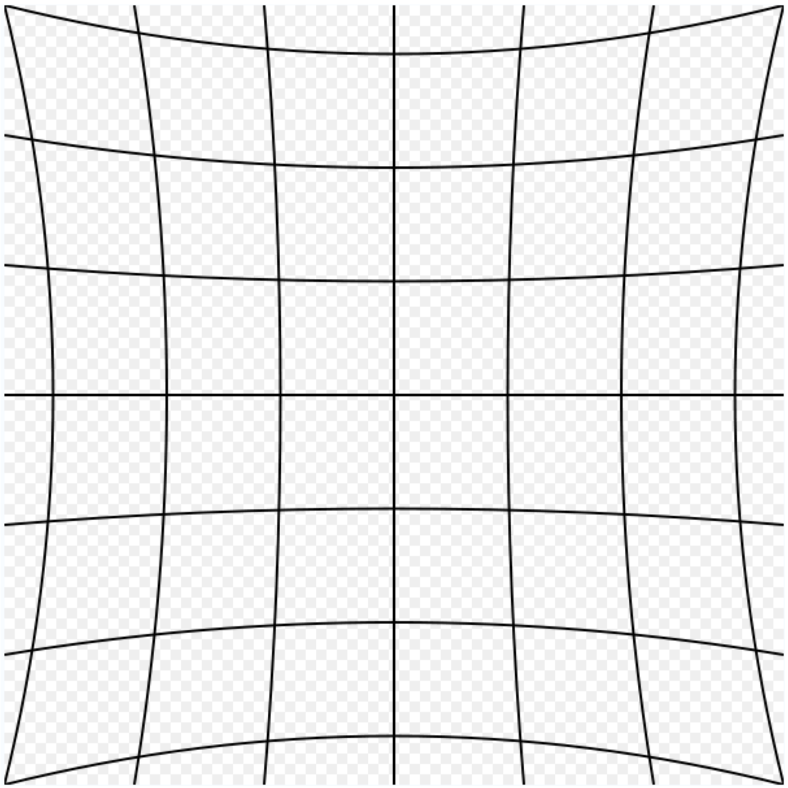
Geometric Transformations

- Greyscale transformations -> operate on range/output
- Geometric transformations -> operate on image domain
  - Coordinate transformations
  - Moving image content from one place to another
- Two parts:
  1. Define transformation
  2. Resample greyscale image in new coordinates

Geom Trans: Distortion From Optics



Barrel Distortion

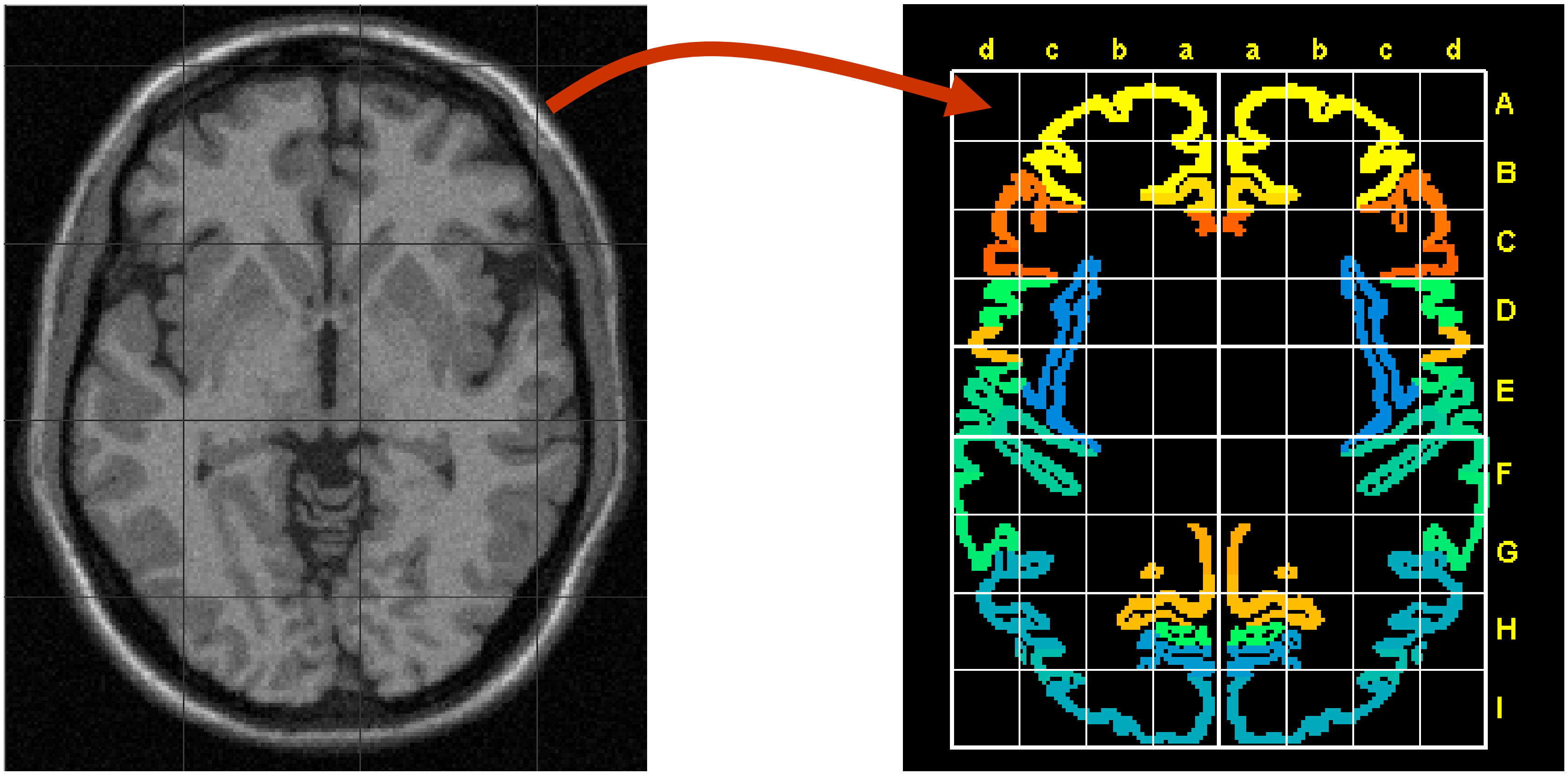


Pincushion Distortion

Radial distortion example

Geom Trans: Distortion From Optics

Geom. Trans.: Brain Template/Atlas



Geom. Trans.: Mosaicing of Series of Images

Domain Mappings Formulation

$$f \longrightarrow g$$

New image from old one

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = T(x,y) = \begin{pmatrix} T_1(x,y) \\ T_2(x,y) \end{pmatrix}$$

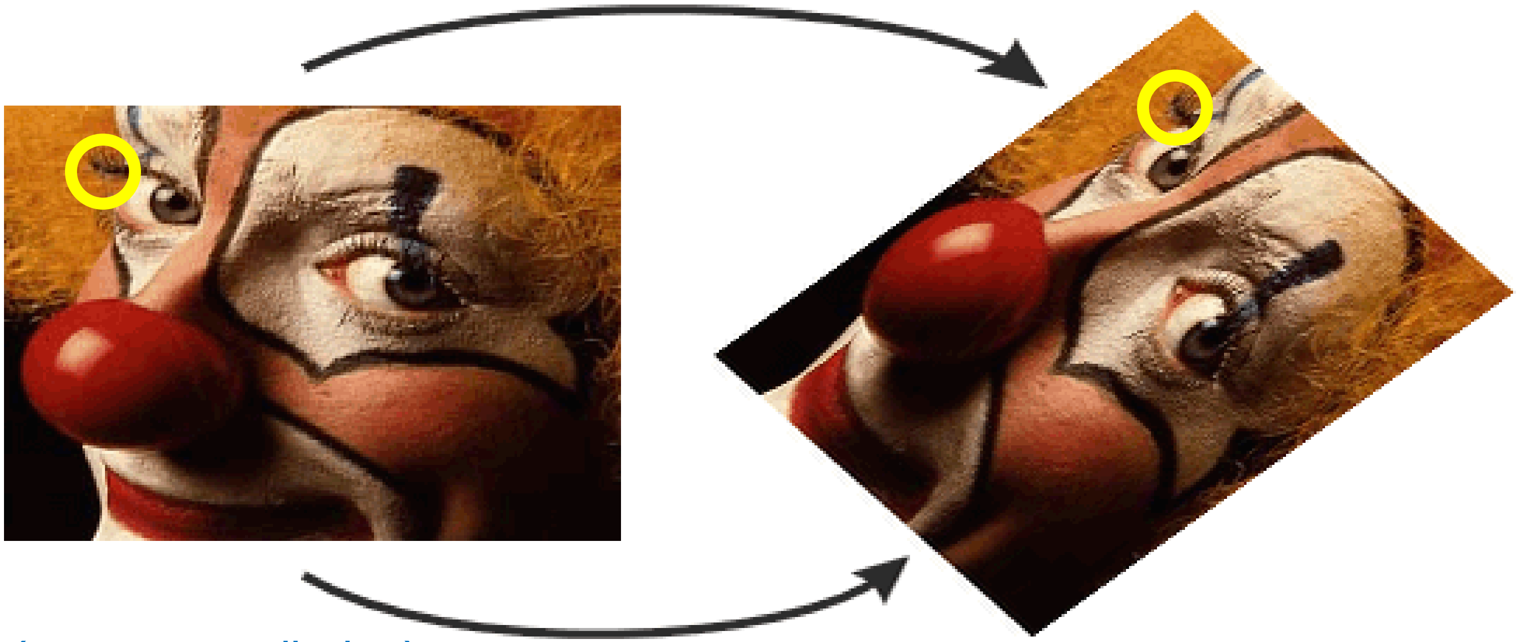
Coordinate transformation
Two parts – vector valued

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

$$g(x,y) = f(x',y') = \tilde{f}(x,y)$$

g is the same (intensity) image as f, but sampled on these new coordinates

Domain Mappings Formulation



(E. H. W. Meijering)

g is the same (intensity) image as f, but sampled on these new coordinates

Domain Mappings Formulation

$$\bar{x}' = T(\bar{x})$$

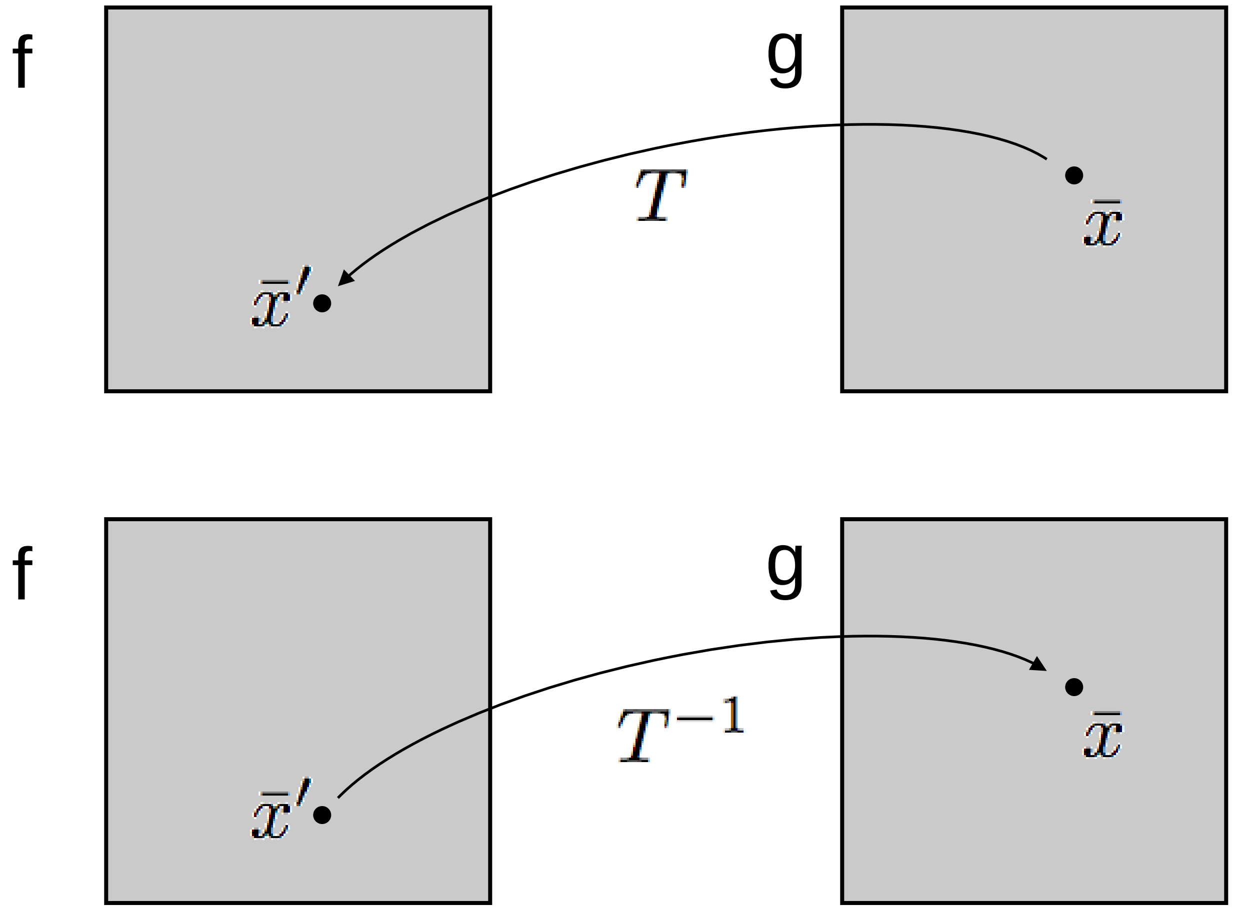
Vector notation is convenient. Bar used some times, depends on context.

$$g(\bar{x}) = \tilde{f}(\bar{x}) = f(\bar{x}') = f(T(\bar{x}))$$

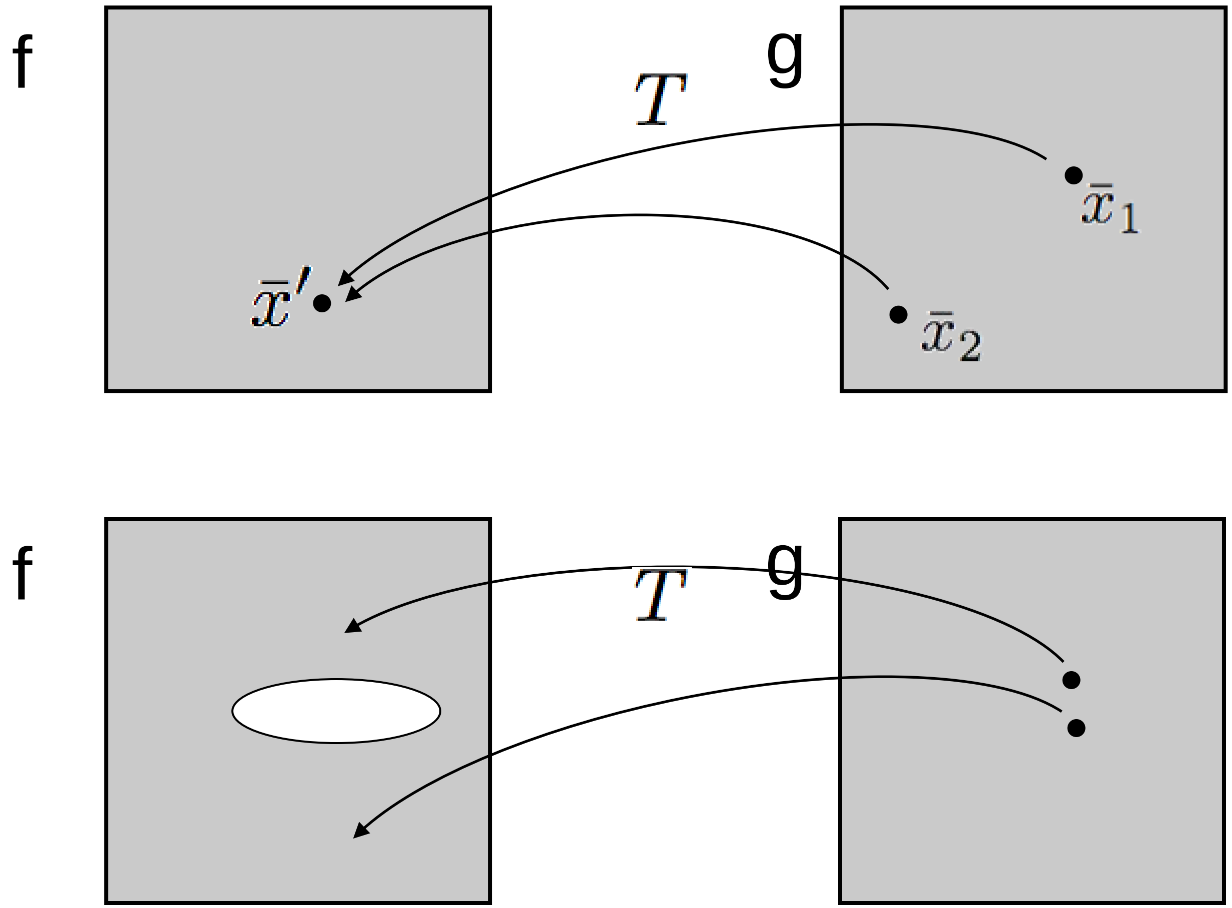
$$\bar{x} = T^{-1}(\bar{x}')$$

T may or may not have an inverse. If not, it means that information was lost.

Domain Mappings



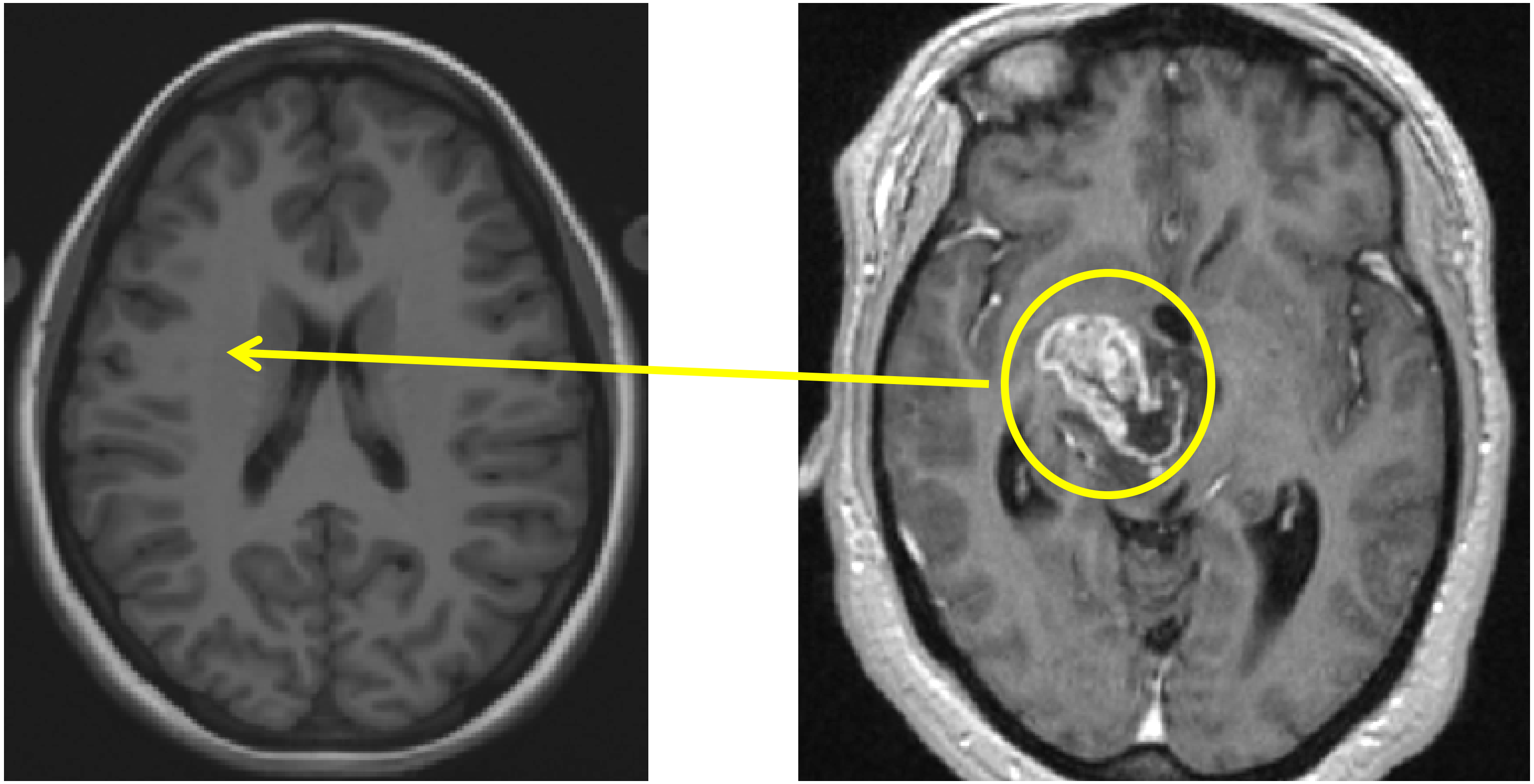
No Inverse?



Not "one to one"

Not "onto" - doesn't cover f

Example



Transformation Examples

- Linear $\bar{x}' = A\bar{x} + \bar{x}_0$ $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

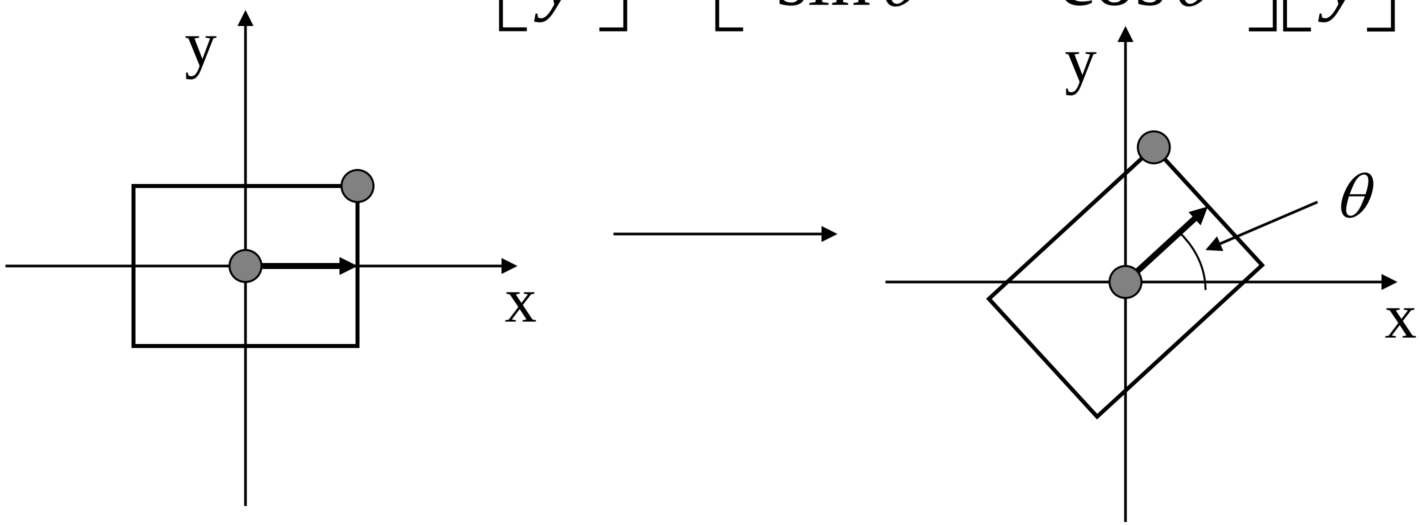
$$x' = ax + by + x_0$$
$$y' = cx + dy + y_0$$

- Rotation, Translation, Scaling?

2D Rotation

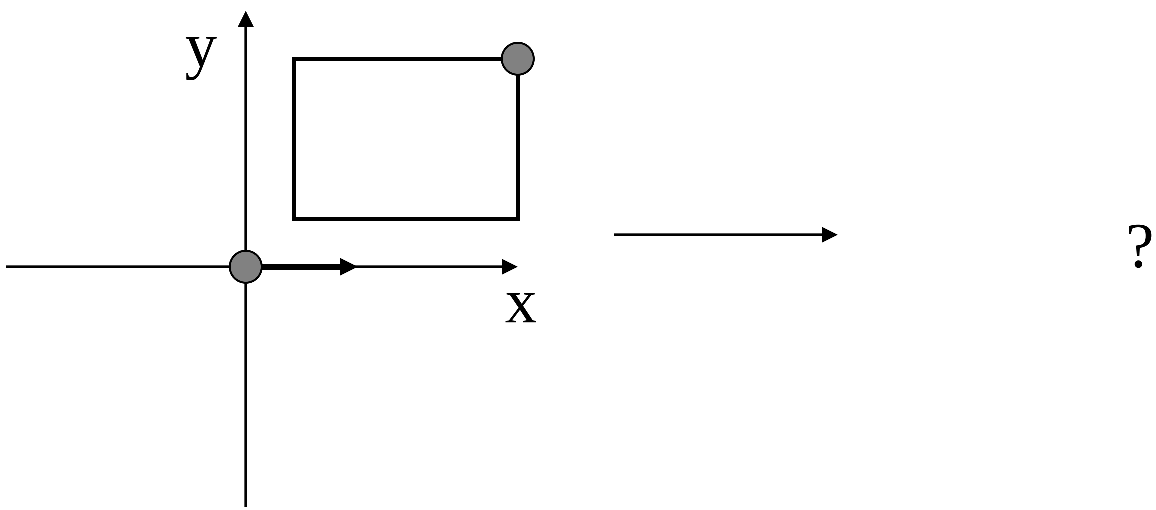
- Rotate counter-clockwise about the origin by an angle θ

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



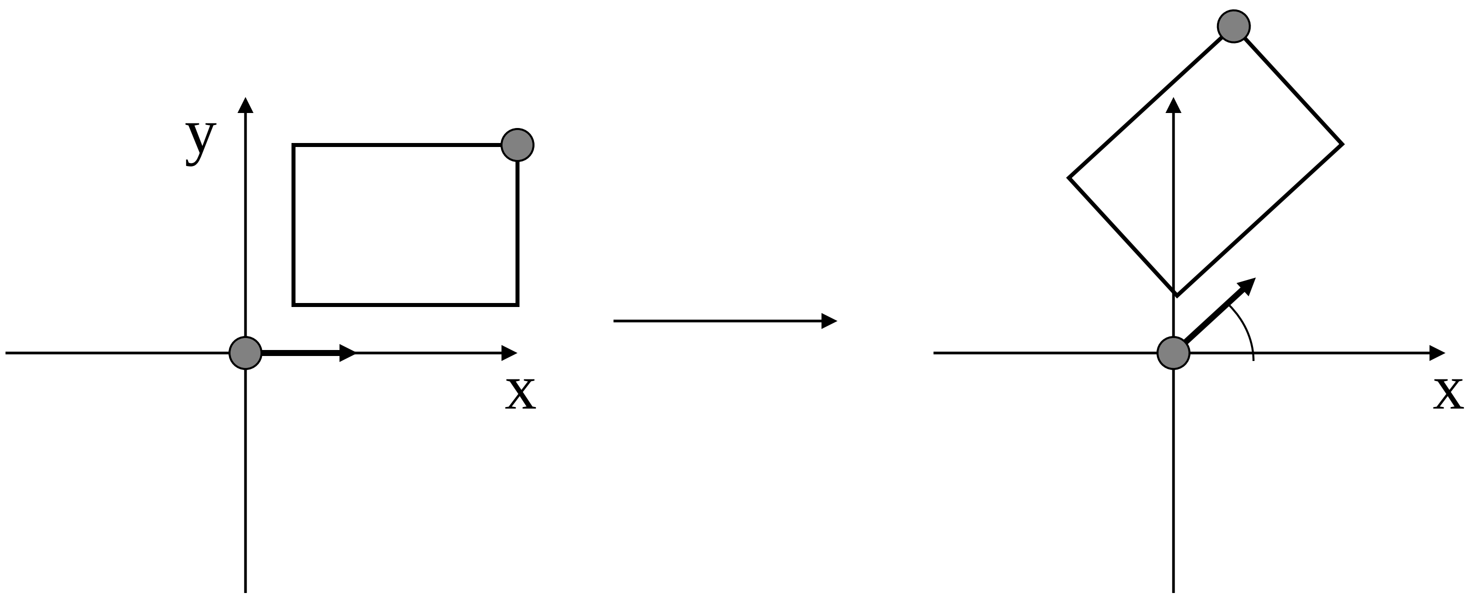
Rotating About An Arbitrary Point

- What happens when you apply a rotation transformation to an object that is not at the origin?



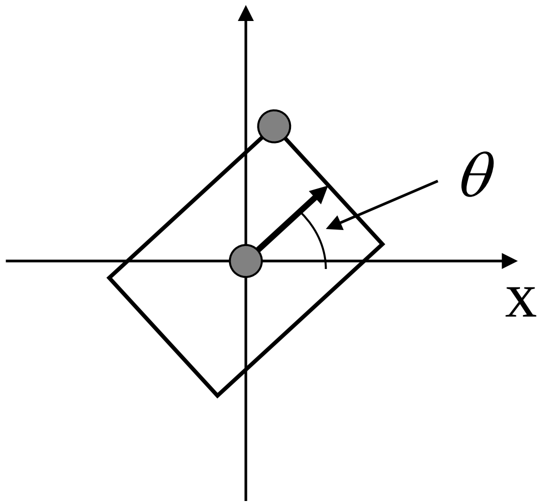
Rotating About An Arbitrary Point

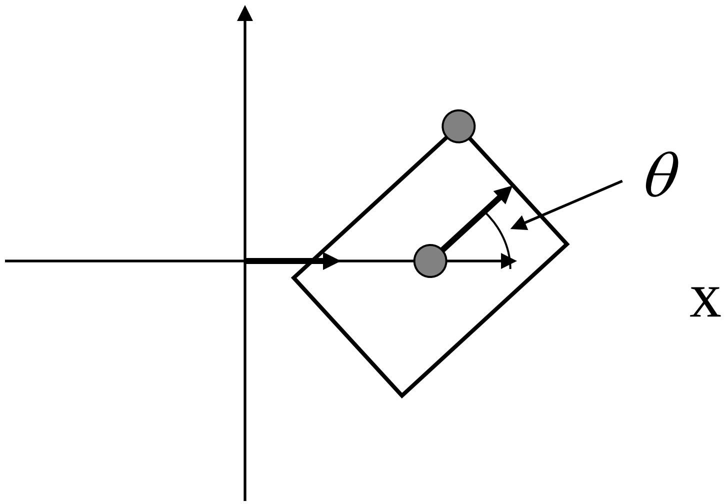
- What happens when you apply a rotation transformation to an object that is not at the origin?
  - It translates as well



Now: First Rotate, then Translate

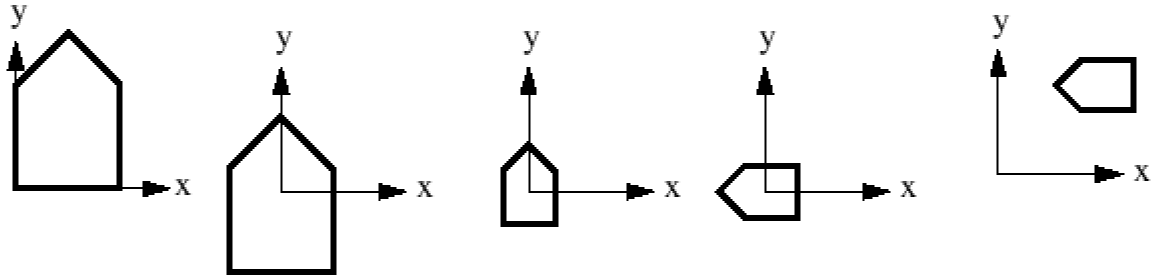
- Rotation followed by translation is **not the same** as translation followed by rotation:
- $T(R(\text{object})) \neq R(T(\text{object}))$





Series of Transformations

2D Object: Translate, scale, rotate, translate again



$$\vec{P'} = T2 + (R \cdot S \cdot (T1 + \vec{P}))$$

☛ Problem: Rotation, scaling, shearing are multiplicative transforms, but translation is additive.

Excellent Materials for self study

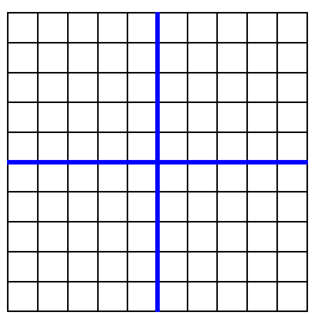
http://groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture07/Slide01.html

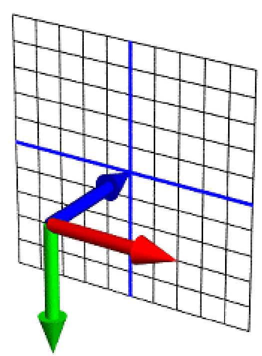
Problems with this Form

- Must consider Translation and Rotation separately
- Computing the inverse transform involves multiple steps
- Order matters between the R and T parts

$$R(T(\bar{x})) \neq T(R(\bar{x}))$$

These problem can be remedied by considering our 2 dimensional image plane as a 2D subspace within 3D.





Homogeneous Coordinates

- Use three numbers to represent a point
- *(x,y)=(wx,wy,w)* for any constant $w \neq 0$
  - Typically, *(x,y)* becomes *(x,y,1)*
  - To go backwards, divide by *w*
- Translation can now be done with matrix multiplication!

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{xx} & a_{xy} & b_x \\ a_{yx} & a_{yy} & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Basic Transformations

- Translation: $\begin{bmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix}$ Rotation: $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- Scaling: $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Transformation Examples

- Linear $\bar{x}' = A\bar{x} + \bar{x}_0$ $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

$$x' = ax + by + x_0$$
$$y' = cx + dy + y_0$$

- Homogeneous coordinates

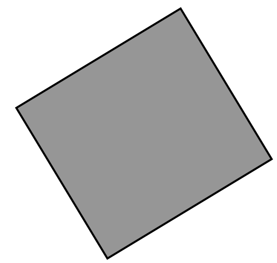
$$\bar{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad A = \begin{pmatrix} a & b & x_0 \\ c & d & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\bar{x}' = A\bar{x}$$

Special Cases of Linear

- Translation $$A = \begin{pmatrix} 0 & 0 & x_0 \\ 0 & 0 & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

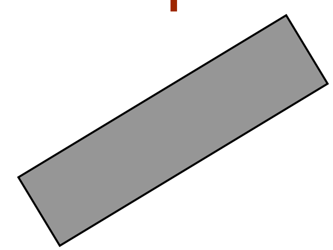
- Rotation $$A = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



- Rigid = rotation + translation
- Scaling $$A = \begin{pmatrix} p & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

p, q < 1 : expand

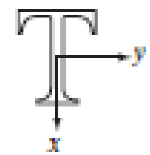
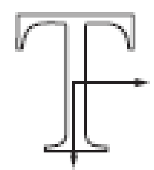
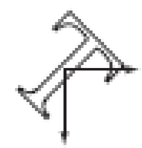
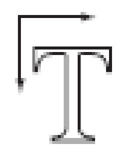
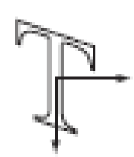
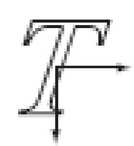
  - Include forward and backward rotation for arbitary axis



- Skew

- Reflection

Linear Transformations

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

$$[x \;\; y \;\; 1] = [v \;\; w \;\; 1]\,\mathbf{T} = [v \;\; w \;\; 1]\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

Cascading of Transformations

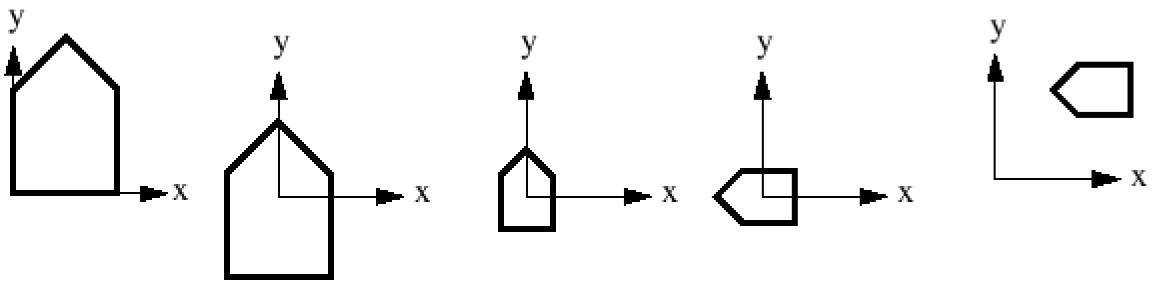
Demo:
http://groups.csail.mit.edu/graphics/classes/6.837/F01/Lecture07/Slide09.html

Homogeneous Coordinates: A general view

- Acknowledgement: Greg Welch, Gary Bishop, Siggraph 2001 Course Notes (Tracking).

Series of Transformations

2D Object: Translate, scale, rotate, translate again



$$\vec{P'} = T2 + (R \cdot S \cdot (T1 + \vec{P}))$$

☛ Problem: Rotation, scaling, shearing are multiplicative transforms, but translation is additive.

Solution: Homogeneous Coordinates

- In 2D: add a third coordinate, w
- Point $[x,y]^T$ expanded to $[x,y,w]^T$
- Scaling: force w to 1 by $[x,y,w]^T/w \rightarrow [x/w,y/w,1]^T$

$$\vec{P} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \text{ where } w \neq 0 \text{ and typically } w = 1$$

Resulting Transformations

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boxed{T = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}}$$

new: $\boxed{\overrightarrow{P'} = T2 \cdot R \cdot S \cdot T1 \cdot \vec{P}}$

before: $\overrightarrow{P'} = T2 + (R \cdot S \cdot (T1 + \vec{P}))$

Linear Transformations

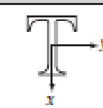
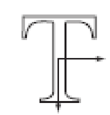
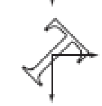
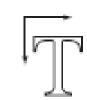
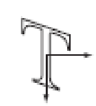
- Also called “affine”
  - 6 parameters
- Rigid -> 3 parameters
- Invertability
  - Invert matrix $T^{-1}(\bar{x}) = A^{-1}\bar{x}$
- What does it mean if A is not invertible?

Affine: General Linear Transformation

$$\bar{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad A = \begin{pmatrix} a & b & x_0 \\ c & d & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\bar{x}' = A\bar{x}$$

6 parameters for Trans (2), Scal (2), Rot (1), Shear X and Shear Y → 7 Parameters ?????

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Affine: General Linear Transformation

$$\bar{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad A = \begin{pmatrix} a & b & x_0 \\ c & d & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\bar{x}' = A\bar{x}$$

6 parameters for Trans (2), Scal (2), Rot (1), Shear X and Shear Y → 7 Parameters ?????

1)

Rot 90deg

Shear X

Rot -90deg

2)

Shear Y

Shear Y can be formulated as Shear X applied to rotated image -> There is only one Shear parameter

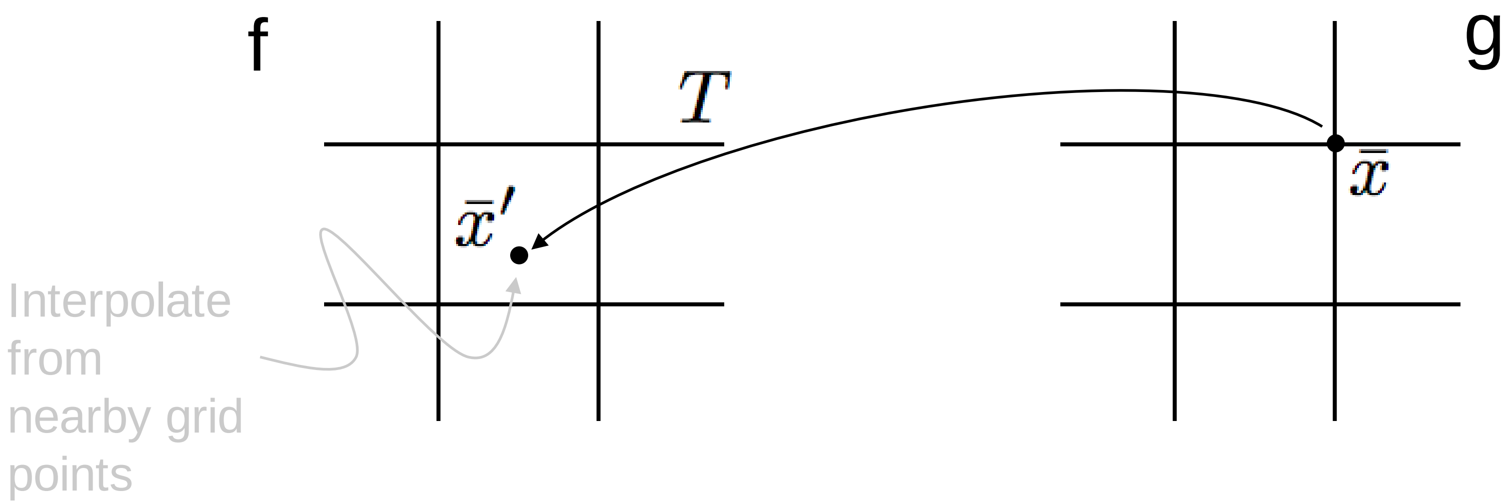
Implementation

Two major procedures:

1. Definition or estimation of transformation type and parameters
2. Application of transformation: Actual transformation of image

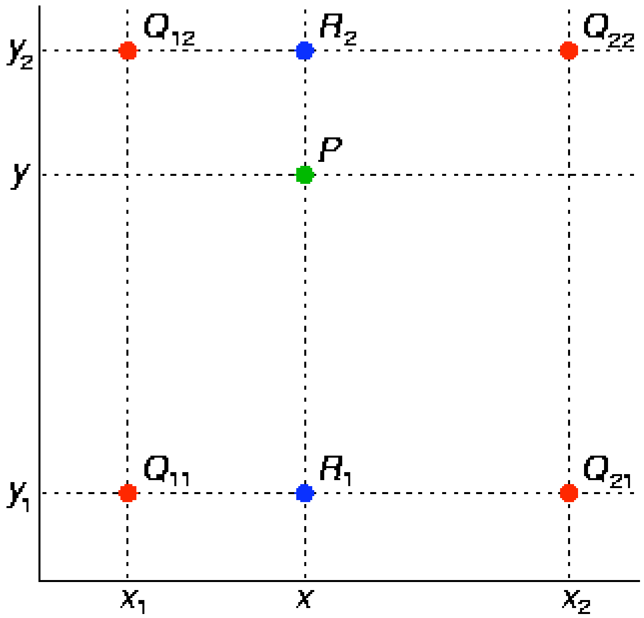
Implementation – Two Approaches to Apply Transf.

1. Pixel filling – forward mapping g to f
   - T() takes you from coords in g() to coords in f()
   - Need random access to pixels in f()
   - Sample grid for g(), interpolate f() as needed



Interpolation: Bilinear

- Successive application of linear interpolation along each axis



$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

Source: WIkipedia

Bilinear Interpolation

- *Not* linear in x, y

$$f(x,y) \approx \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1).$$

$$b_1 + b_2 x + b_3 y + b_4 xy$$

$$\begin{aligned} b_1 &= f(0,0) \\ b_2 &= f(1,0) - f(0,0) \\ b_3 &= f(0,1) - f(0,0) \\ b_4 &= f(0,0) - f(1,0) \\ &\quad - f(0,1) + f(1,1). \end{aligned}$$

Bilinear Interpolation

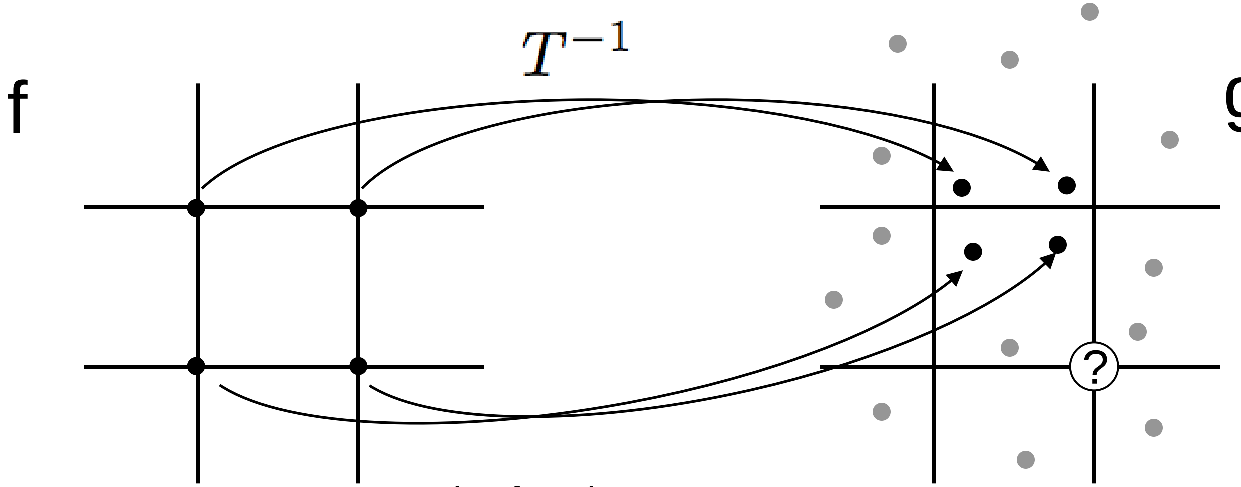
- Convenient form
  - – Normalize to unit grid [0,1]x[0,1]

$$f(x,y) \approx f(0,0)\,(1-x)(1-y) + f(1,0)\,x(1-y) + f(0,1)\,(1-x)y + f(1,1)xy.$$

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

Implementation – Two Approaches

2. Splatting – backward mapping f to g
   - T^{-1}() takes you from coords in f() to coords in g()
   - You have f() on grid, but you need g() on grid
   - Push grid samples onto g() grid and do interpolation from <u>unorganized</u> data (kernel)



Nearby points are not organized – "scattered"

Scattered Data Interpolation With Kernels
Shepard's method

- Define kernel
  - Falls off with distance, radially symmetric

$$K(\bar{x}_1, \bar{x}_2) = K(|\bar{x}_1 - \bar{x}_2|)$$

Kernel examples

$$K(\bar{x}_1, \bar{x}_2) = \frac{1}{2\pi\sigma^2} e^{\frac{|\bar{x}_1 - \bar{x}_2|^2}{2\sigma^2}}$$

$$K(\bar{x}_1, \bar{x}_2) = \frac{1}{|\bar{x}_1 - \bar{x}_2|^p}$$

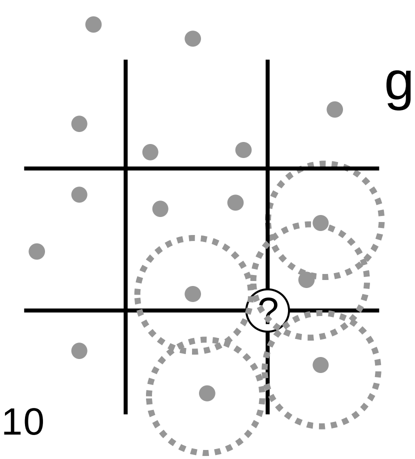
$$g(x) = \frac{1}{\sum_{j=1}^{N} w_j} \sum_{i=1}^{N} w_i f\left(x'_i\right)$$

$$w_j = K\left(|\bar{x} - T^{-1}(\bar{x}'_j)\right)$$

Required grid coordinates in g

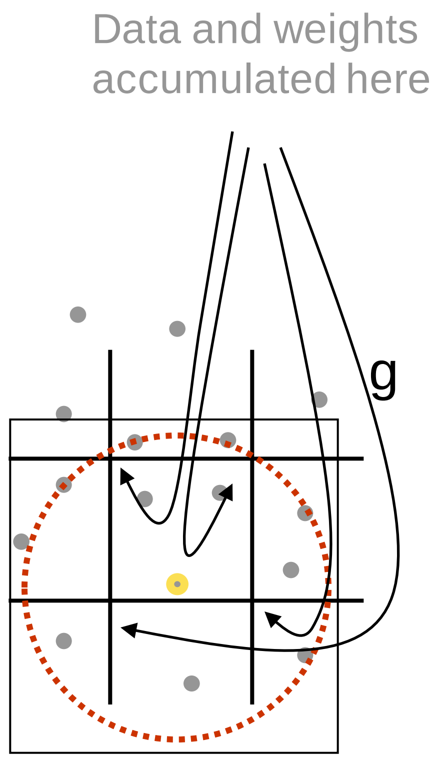
Transformed coord. from f

Grid coordinates in f



Shepard's Method Implementation

- If points are dense enough
  - Truncate kernel
  - For each point in f()
    - Form a small box around it in g() – beyond which truncate
    - Put weights and data onto grid in g()
  - Divide total data by total weights: B/A

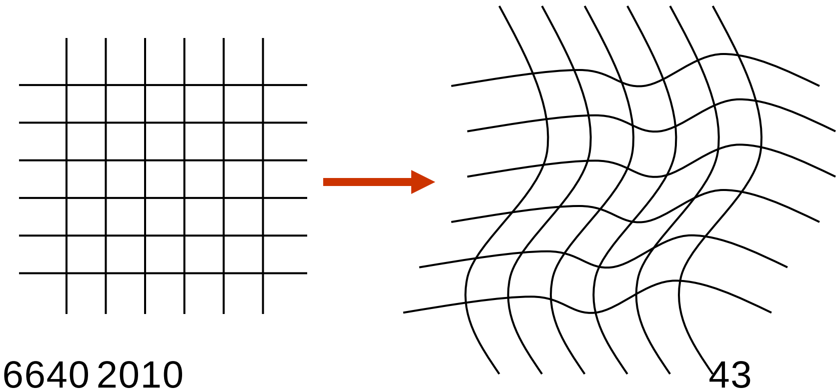


$$A = \sum_{j=1}^{N} w_j \qquad B = \sum_{i=1}^{N} w_i f\left(T^{-1}(x_i')\right)$$

ESTIMATION OF TRANSFORMATIONS

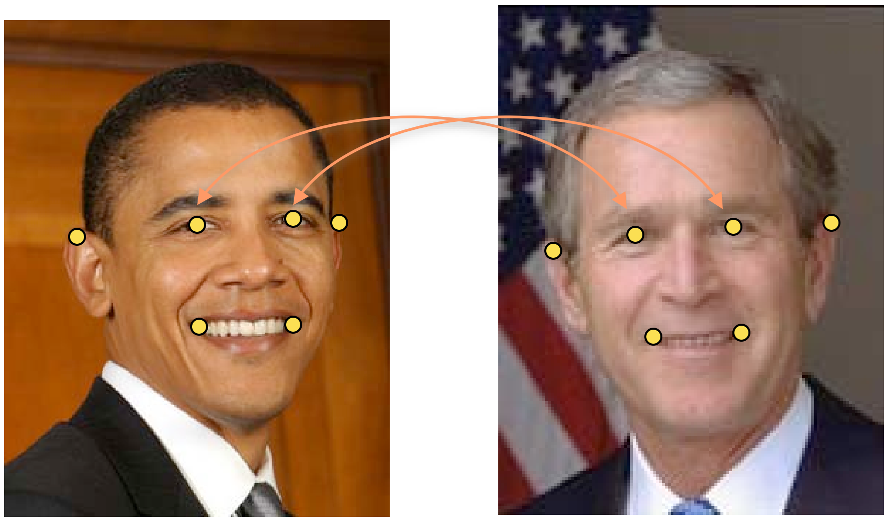
Determine Transformations

- All polynomials of (x,y)
- Any vector valued function with 2 inputs
- How to construct transformations?
  - Define form or class of a transformation
  - Choose parameters within that class
    - Rigid - 3 parameters (T, R)
    - Affine - 6 parameters



Correspondences

- Also called “landmarks” or “fiducials”

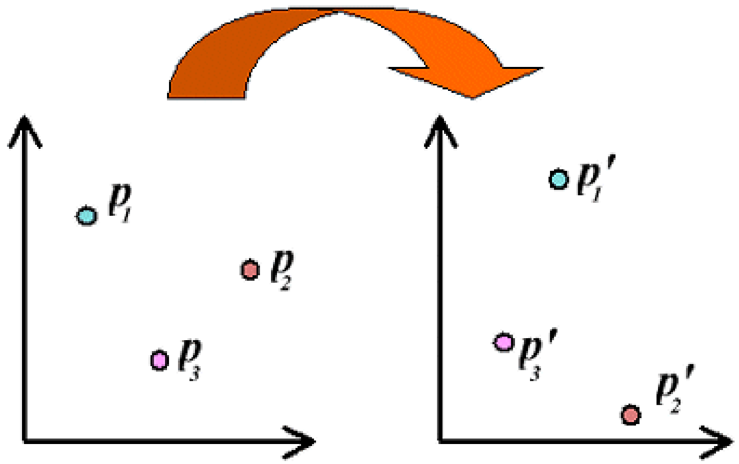


$\bar{c}_1, \bar{c}'_1$
$\bar{c}_2, \bar{c}'_2$
$\bar{c}_3, \bar{c}'_3$
$\bar{c}_4, \bar{c}'_4$
$\bar{c}_5, \bar{c}'_5$
$\bar{c}_6, \bar{c}'_6$

Question: How many landmarks for affine T?

- Estimation of 6 parameters → 3 corresponding point pairs with (x,y) coordinates

The coordinates of three corresponding points uniquely determine and Affine Transform



If we know where we would like at least three points to map to, we can solve for an Affine transform that will give this mapping.

Transformations/Control Points Strategy

1. Define a functional representation for T with k parameters (β) $T(\beta, \bar{x})$
   $\beta = (\beta_1, \beta_2, \ldots, \beta_K)$
2. Define (pick) N correspondences
3. Find β so that

$$\bar{c}_i' = T(\beta, \bar{c}_i) \;\; i = 1, \ldots, N$$

4. If overconstrained (K < 2N) then solve

$$\arg\min_{\beta} \left[\sum_{i=1}^{N} \left(\bar{c}_i' - T(\beta, \bar{c}_i) \right)^2 \right]$$

Example Affine Transformation: 3 Corresponding Landmarks

Solution Method

We've used this technique several times now. We set up 6 linear equations in terms of our 6 unknown values. In this case, we know the coordinates before and after the mapping, and we wish to solve for the entries in our Affine transform matrix.

This gives the following solution:

$$\mathbf{X}^{-1}\mathbf{x}' = \mathbf{a}$$

$$\underbrace{\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \end{bmatrix}}_{\mathbf{x}'} = \underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix}}_{\mathbf{a}}$$

Example: Quadratic

Transformation

$$T_x = \beta_x^{00} + \beta_x^{10}x + \beta_x^{01}y + \beta_x^{11}xy + \beta_x^{20}x^2 + \beta_x^{02}y^2$$
$$T_y = \beta_y^{00} + \beta_y^{10}x + \beta_y^{01}y + \beta_y^{11}xy + \beta_y^{20}x^2 + \beta_y^{02}y^2$$

Denote $\bar{c}_i = (c_{x,i}, c_{y,i})$

Correspondences must match

$$c'_{y,i} = \beta_y^{00} + \beta_y^{10}c_{x,i} + \beta_y^{01}c_{y,i} + \beta_y^{11}c_{x,i}c_{y,i} + \beta_y^{20}c_{x,i}^2 + \beta_y^{02}c_{y,i}^2$$
$$c'_{x,i} = \beta_x^{00} + \beta_x^{10}c_{x,i} + \beta_x^{01}c_{y,i} + \beta_x^{11}c_{x,i}c_{y,i} + \beta_x^{20}c_{x,i}^2 + \beta_x^{02}c_{y,i}^2$$

Note: these equations are linear in the unkowns

Write As Linear System

$$
\begin{pmatrix}
\begin{matrix} 1 & c_{x,1} & c_{y,1} & c_{x,1}c_{y,1} & c_{x,1}^2 & c_{y,1}^2 \\ 1 & c_{x,2} & c_{y,2} & c_{x,2}c_{y,2} & c_{x,2}^2 & c_{y,2}^2 \\ & & & \vdots & & \\ 1 & c_{x,N} & c_{y,N} & c_{x,N}c_{y,N} & c_{x,N}^2 & c_{y,N}^2 \end{matrix} & 0 \\
0 & \begin{matrix} 1 & c_{x,1} & c_{y,1} & c_{x,1}c_{y,1} & c_{x,1}^2 & c_{y,1}^2 \\ 1 & c_{x,2} & c_{y,2} & c_{x,2}c_{y,2} & c_{x,2}^2 & c_{y,2}^2 \\ & & & \vdots & & \\ 1 & c_{x,N} & c_{y,N} & c_{x,N}c_{y,N} & c_{x,N}^2 & c_{y,N}^2 \end{matrix}
\end{pmatrix}
\begin{pmatrix} \beta_x^{00} \\ \beta_x^{10} \\ \beta_x^{01} \\ \beta_x^{11} \\ \beta_x^{20} \\ \beta_x^{02} \\ \beta_y^{00} \\ \beta_y^{10} \\ \beta_y^{01} \\ \beta_y^{11} \\ \beta_y^{20} \\ \beta_y^{02} \end{pmatrix}
=
\begin{pmatrix} c'_{x,1} \\ c'_{x,2} \\ \vdots \\ c'_{x,N} \\ c'_{y,1} \\ c'_{y,2} \\ \vdots \\ c'_{y,N} \end{pmatrix}
$$

$$Ax = b$$

A – matrix that depends on the (unprimed) correspondences and the transformation

x – unknown parameters of the transformation

b – the primed correspondences

Linear Algebra Background

$$Ax = b$$

$$\begin{array}{rcl} a_{11}x_1 + \ldots + a_{1N}x_N & = & b_1 \\ a_{21}x_1 + \ldots + a_{2N}x_N & = & b_2 \\ \ldots & & \ldots \\ a_{M1}x_1 + \ldots + a_{MN}x_N & = & b_M \end{array}$$

Simple case: A is square (M=N) and invertable (det[A] not zero)

$$A^{-1}Ax = Ix = x = A^{-1}b$$

(Numerics: Don't find A inverse. Use Gaussian elimination or some kind of decomposition of A.)

Linear Systems – Other Cases

- (M<N) or (M = N and the equations are degenerate or *singular):*
  - System is underconstrained – lots of solutions
- Approach
  - Impose some extra criterion on the solution
  - Find the one solution that optimizes that criterion
  - *Regularizing* the problem

Linear Systems – Other Cases

- M > N (e.g. more points than parameters):
  - System is overconstrained
  - *No* solution
- Approach
  - Find solution that is best compromise
  - Minimize squared error (least squares)

$$x = \arg\min_{\mathbf{x}} |\mathrm{Ax} - \mathrm{b}|^2$$

Solving Least Squares Systems

- Pseudoinverse (normal equations)

$$A^T A x = A^T b$$
$$x = (A^T A)^{-1} A^T b$$

  - Issue: often not well conditioned (nearly singular)
- Alternative: *singular value decomposition SVD*

Singular Value Decomposition

$$\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix} = UWV^T = \begin{pmatrix} & & \\ & U & \\ & & \end{pmatrix} \begin{pmatrix} w_1 & & & 0 \\ & w_2 & & \\ & & \cdots & \\ & & \cdots & \\ 0 & & & w_N \end{pmatrix} \begin{pmatrix} & V^T & \end{pmatrix}$$

$$I = U^T U = U U^T = V^T V = V V^T$$

Invert matrix A with SVD

$$A^{-1} = V W^{-1} U^T \qquad W^{-1} = \begin{pmatrix} \frac{1}{w_1} & & & 0 \\ & \frac{1}{w_2} & & \\ & & \cdots & \\ & & \cdots & \\ 0 & & & \frac{1}{w_N} \end{pmatrix}$$

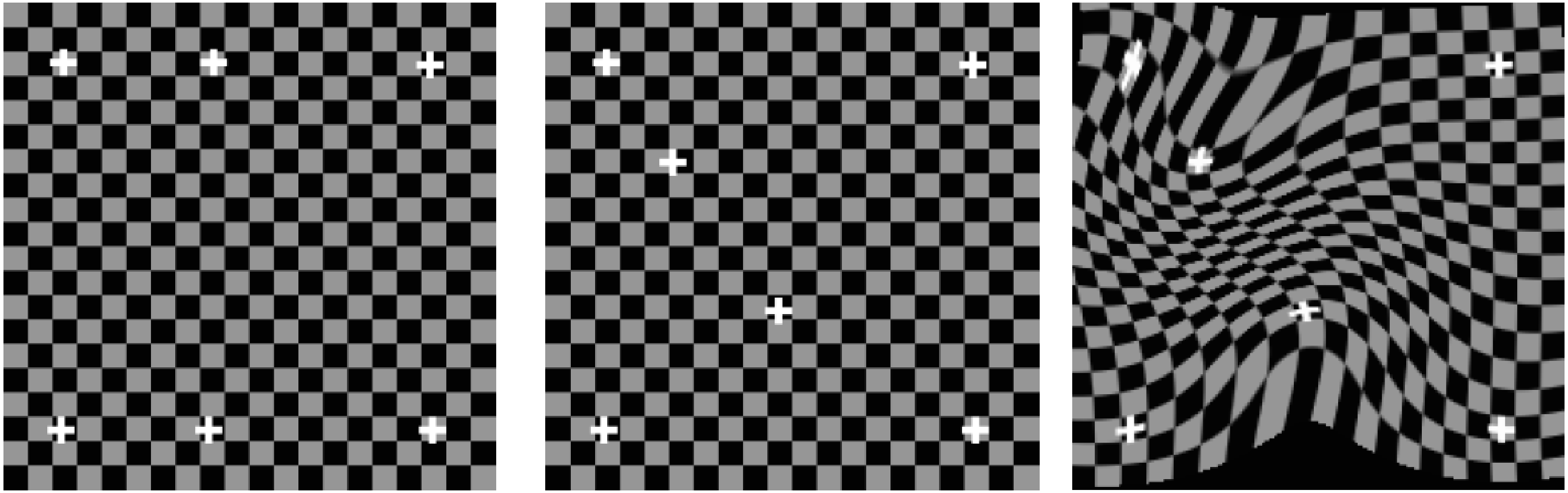
SVD for Singular Systems

- If a system is singular, some of the w's will be zero

$$x = VW^*U^Tb$$

$$w_j^* = \begin{cases} 1/w_j & |w_j| > \epsilon \\ 0 & \text{otherwise} \end{cases}$$

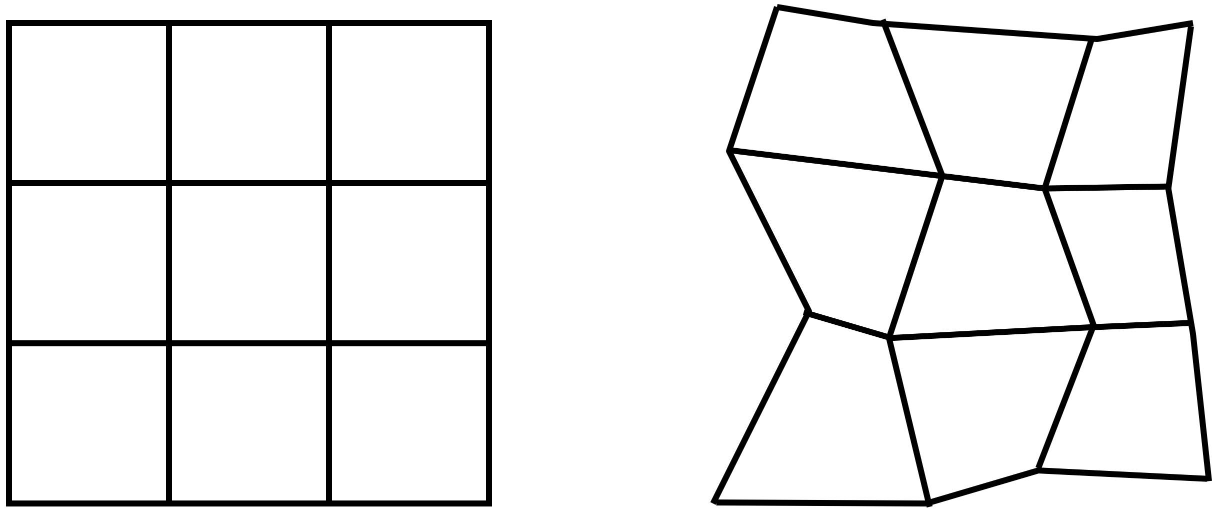
- Properties:
  - Underconstrained: solution with shortest overall length
  - Overconstrained: least squares solution



SPECIFYING "WARPS" VIA SPARSE SET OF LANDMARKS

Specifying Warps – Another Strategy

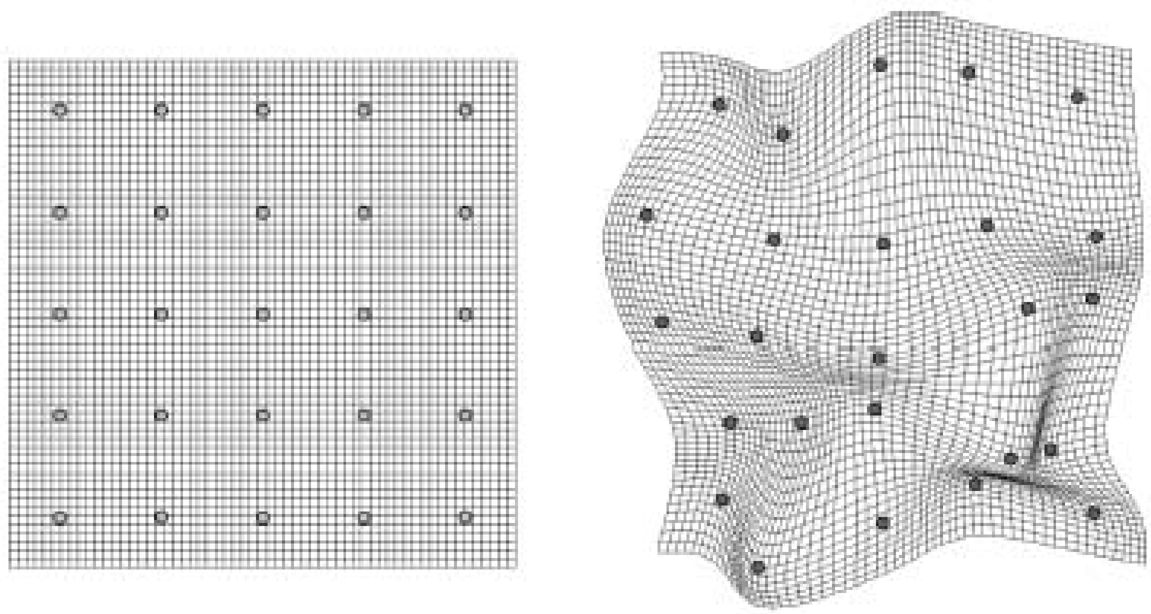
- Let the # DOFs in the warp equal the # of control points (x1/2)
  - Interpolate with some grid-based interpolation
    - E.g. binlinear, splines



Landmarks Not On Grid

- Landmark positions driven by application
- Interpolate transformation at unorganized correspondences
  - *Scattered data interpolation*
- How do we do scattered data interpolation?
  - Idea: use kernels!
- *Radial basis functions*
  - Radially symmetric functions of distance to landmark

Concept



(a) (b)

Figure 1. Warping a 2D mesh with RBFs: a) original mesh; b) mesh after warping.

Concept

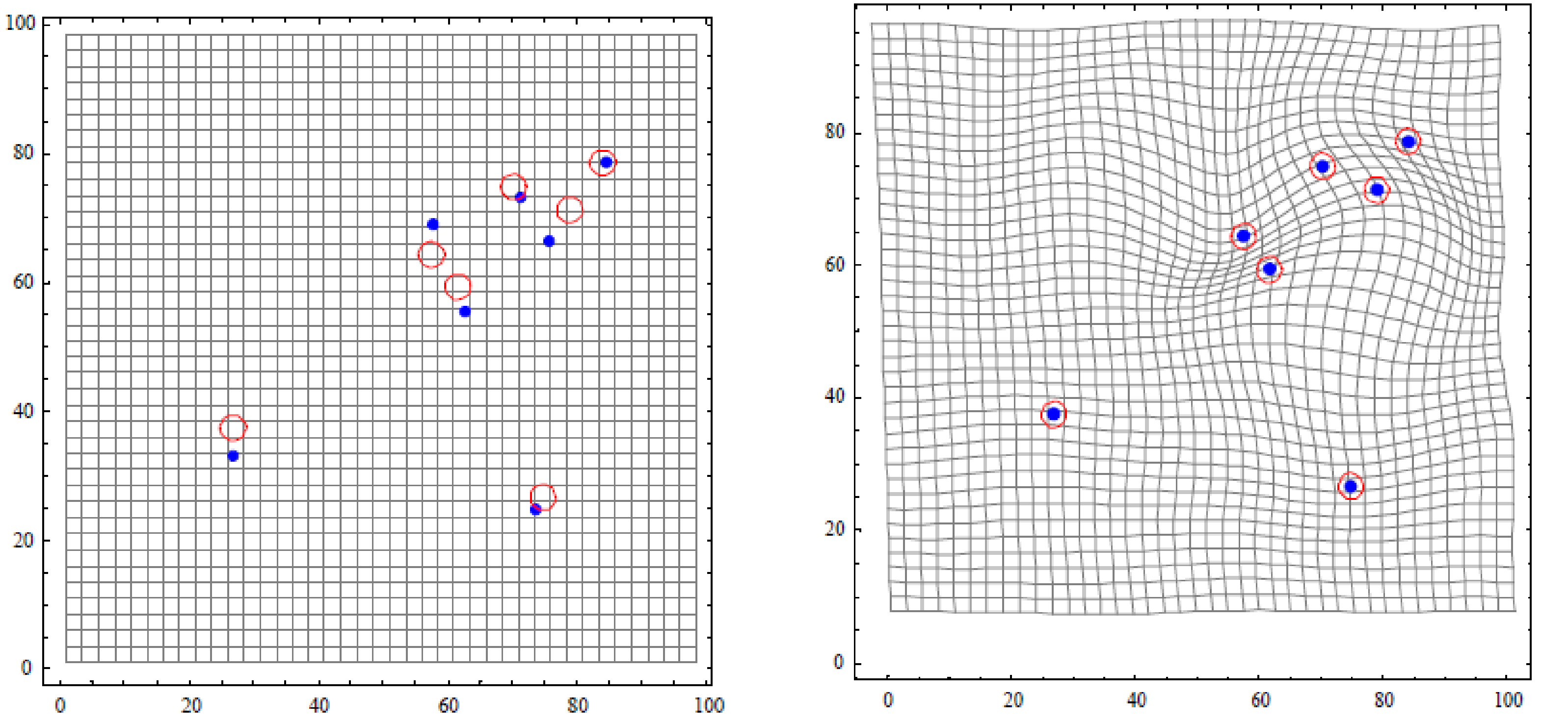


Fig. 5 Radial basis interpolation of a regular grid, based on the random motion of 7 landmarks.

Warping a Neuro-Anatomy Atlas on 3D MRI Data with Radial Basis Functions
H.E. Bennink, J.M. Korbeeck, B.J. Janssen, B.M. ter Haar Romeny

Concept

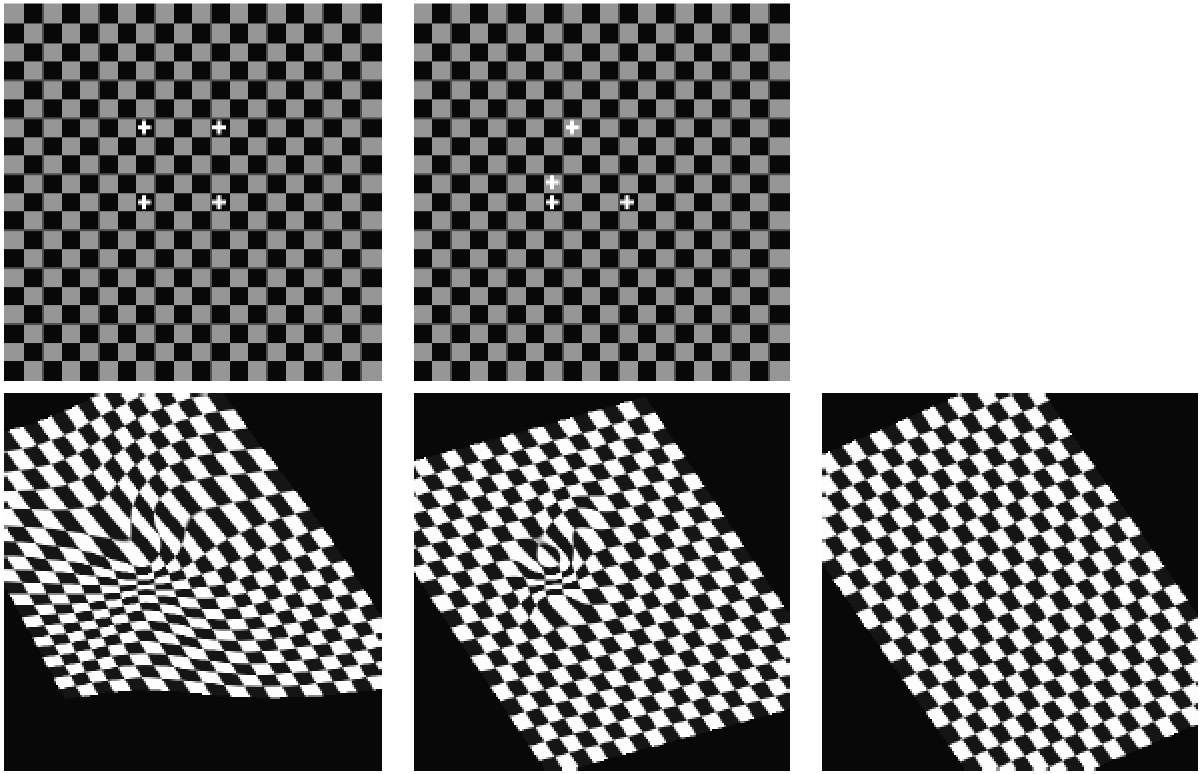


Figure 8: Generalizing affine mappings in different ways. **Top**: Position of source and target anchor points. **Bottom** (left to right): thin-plate warp, Gaussian warp, affine least-square warp ($\lambda = \infty$). In all cases the mapping can be well approximated by an affine mapping far away from the anchors. In the thin-plate case this affine map is different at different regions, unlike the Gaussian case in which the same affine component appearing in the definition of the mapping dominates the transformation in all areas away from the anchors.

RBFs – Formulation

- Represent T as weighted sum of basis functions

$$\underbrace{\mathrm{T}(\bar{x}) = \sum_{i=1}^{N} k_i \phi_i(\bar{x})}_{\text{Sum of radial basis functions}} \qquad \phi_i(\bar{x}) = \underbrace{\phi\left(||\bar{x} - \bar{x}_i||\right)}_{\text{Basis functions centered at positions of data}}$$

- Need interpolation for vector-valued function, T:

$$\begin{aligned} T^x(\bar{x}) &= \textstyle\sum_{i=1}^{N} k_i^x \phi_i(\bar{x}) \\ T^y(\bar{x}) &= \textstyle\sum_{i=1}^{N} k_i^y \phi_i(\bar{x}) \end{aligned}$$

Choices for ϕ

- Gaussian: $g(t) = \exp(-0.5(t^2/\sigma^2))$
- Multiquadratics: $g(t) = 1/\mathrm{Sqrt}(t^2+c^2)$, where c is least distance to surrounding points

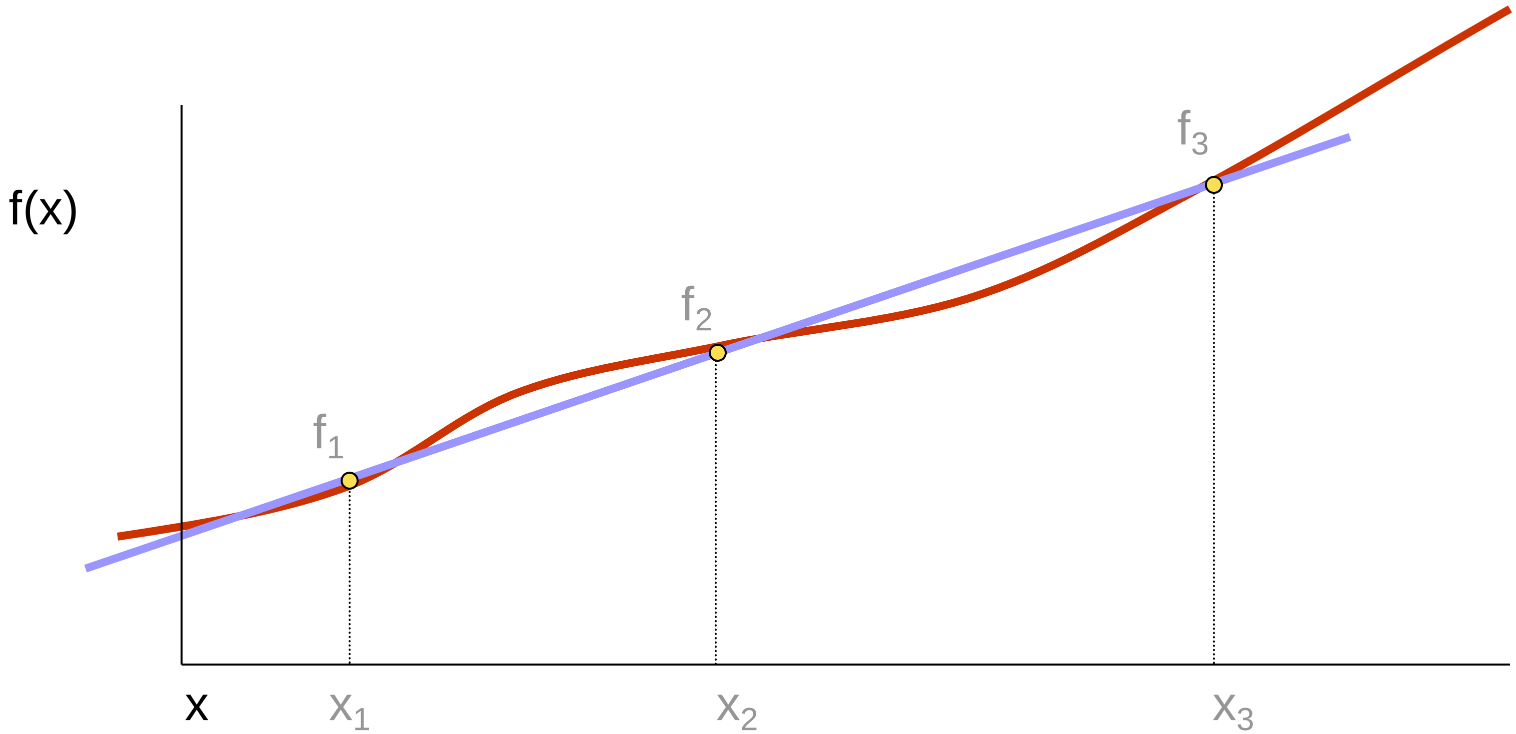
Solve For k's With Landmarks as Constraints

- Find the k's so that T(x) fits at data points

$$\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} k_1^x \\ k_2^x \\ \vdots \\ k_N^x \\ k_1^y \\ k_2^y \\ \vdots \\ k_N^y \end{pmatrix} = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_N \\ y'_1 \\ y'_2 \\ \vdots \\ y'_N \end{pmatrix}$$

$$B = \begin{pmatrix} \phi_1(\bar{x}_1) & \phi_2(\bar{x}_1) & \dots & \phi_N(\bar{x}_1) \\ \phi_1(\bar{x}_2) & \phi_2(\bar{x}_2) & \dots & \phi_N(\bar{x}_2) \\ \vdots & & & \\ \phi_1(\bar{x}_N) & \phi_2(\bar{x}_N) & \dots & \phi_N(\bar{x}_N) \end{pmatrix}$$

Issue: RBFs Do Not Easily Model Linear Trends



RBFs – Formulation w/Linear Term

- Represent T as weighted sum of basis functions and linear part

$$\mathrm{T}(\bar{x}) = \underbrace{\sum_{i=1}^{N} k_i \phi_i(\bar{x})}_{\text{Sum of radial basis functions}} + \underbrace{p_2 y + p_1 x + p_o}_{\text{Linear part of transformation}} \qquad \phi_i(\bar{x}) = \phi\left(\underbrace{||\bar{x} - \bar{x}_i||}_{\text{Basis functions centered at positions of data}}\right)$$

- Need interpolation for vector-valued function, T:

$$\begin{aligned} T^x(\bar{x}) &= \sum_{i=1}^{N} k_i^x \phi_i(\bar{x}) + p_2^x y_+ p_1^x x + p_o^x \\ T^y(\bar{x}) &= \sum_{i=1}^{N} k_i^y \phi_i(\bar{x}) + p_2^y y_+ p_1^y x + p_o^y \end{aligned}$$

RBFs – Linear System

$$\begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} k_1^x \\ k_2^x \\ \vdots \\ k_N^x \\ p_2^x \\ p_1^x \\ p_0^x \\ k_1^y \\ k_2^y \\ \vdots \\ k_N^y \\ p_2^y \\ p_1^y \\ p_0^y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ x'_1 \\ x'_2 \\ \vdots \\ x'_N \\ 0 \\ 0 \\ 0 \\ y'_1 \\ y'_2 \\ \vdots \\ y'_N \end{pmatrix}$$

$$B = \begin{pmatrix} x_1 & x_2 & \dots & x_N & 0 & 0 & 0 \\ y_1 & y_2 & \dots & y_N & 0 & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 \\ \phi_{11} & \phi_{12} & \dots & \phi_{1N} & y_1 & x_1 & 1 \\ \phi_{21} & \phi_{22} & \dots & \phi_{2N} & y_2 & x_2 & 1 \\ \vdots & & & & & & \\ \phi_{N1} & \phi_{N2} & \dots & \phi_{NN} & y_N & x_N & 1 \end{pmatrix}$$

RBFs – Solution Strategy

- Find the k's and p's so that T() fits at data points
  - The k's can have no linear trend (force it into the p's)
- Constraints -> linear system

$$T^x(\bar{x}_i) = x'_i \qquad T^y(\bar{x}_i) = y'_i$$

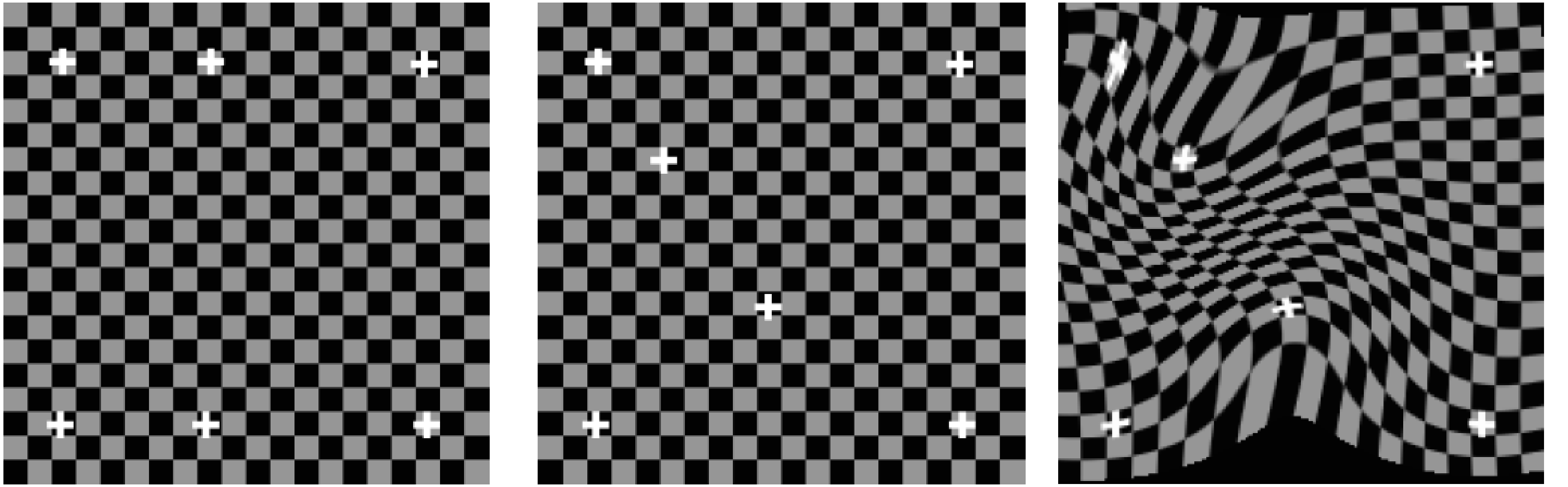
Correspondences must match

$$\sum_{i=1}^{N} k_i^x = 0 \qquad \sum_{i=1}^{N} k_i^y = 0$$

$$\sum_{i=1}^{N} k_i^x \bar{x}_i = \bar{0} \qquad \sum_{i=1}^{N} k_i^y \bar{x}_i = \bar{0}$$

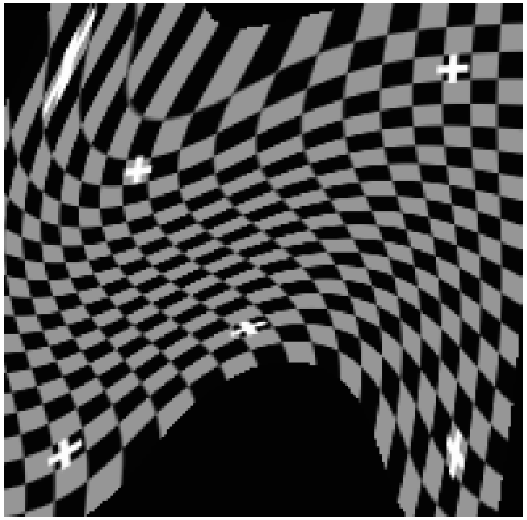
Keep linear part separate from deformation

RBF Warp – Example

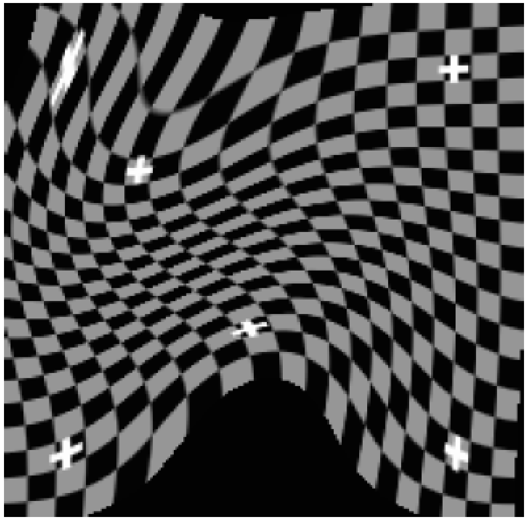


What Kernel Should We Use

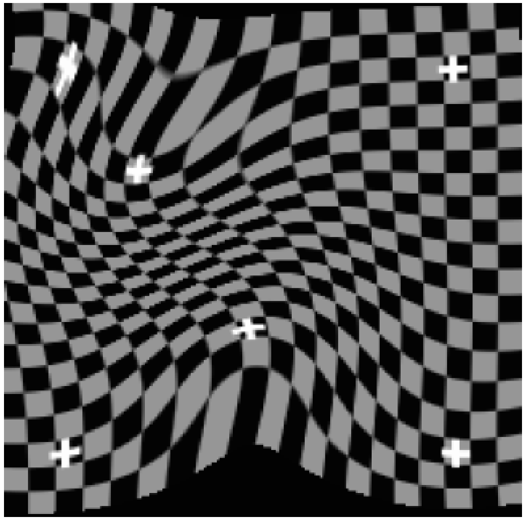
- Gaussian
  - Variance is free parameter – controls smoothness of warp



$\sigma = 2.5$



$\sigma = 2.0$



$\sigma = 1.5$

From: Arad et al. 1994

RBFs – Aligning Faces

Mona Lisa – Target

Venus – Source

Venus – Warped

Symmetry?

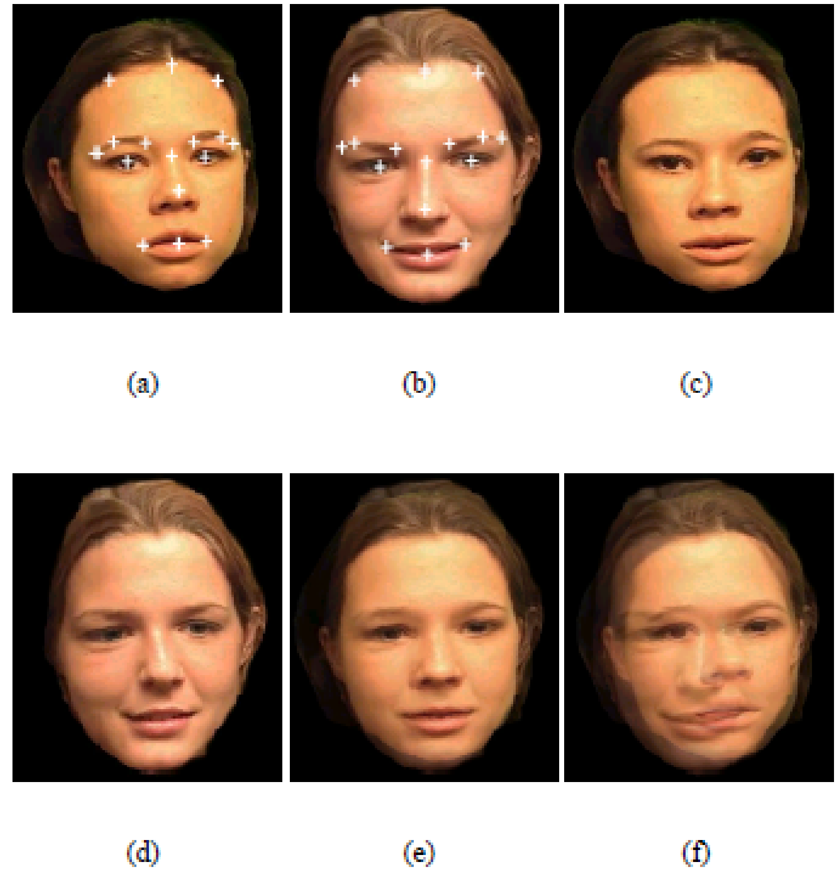


Figure 2. Image metamorphasis with RBFs: a) source image I_0; b) destination image I_1; c) forward warping I_0 with $d_{0\rightarrow 1}$; d) backward warping I_1 with $d_{1\rightarrow 0}$; e) result of morphing between I_0 and I_1; f) cross-dissolved image.

Image-based Talking Heads using Radial Basis Functions James D. Edge and Steve Maddock

Application

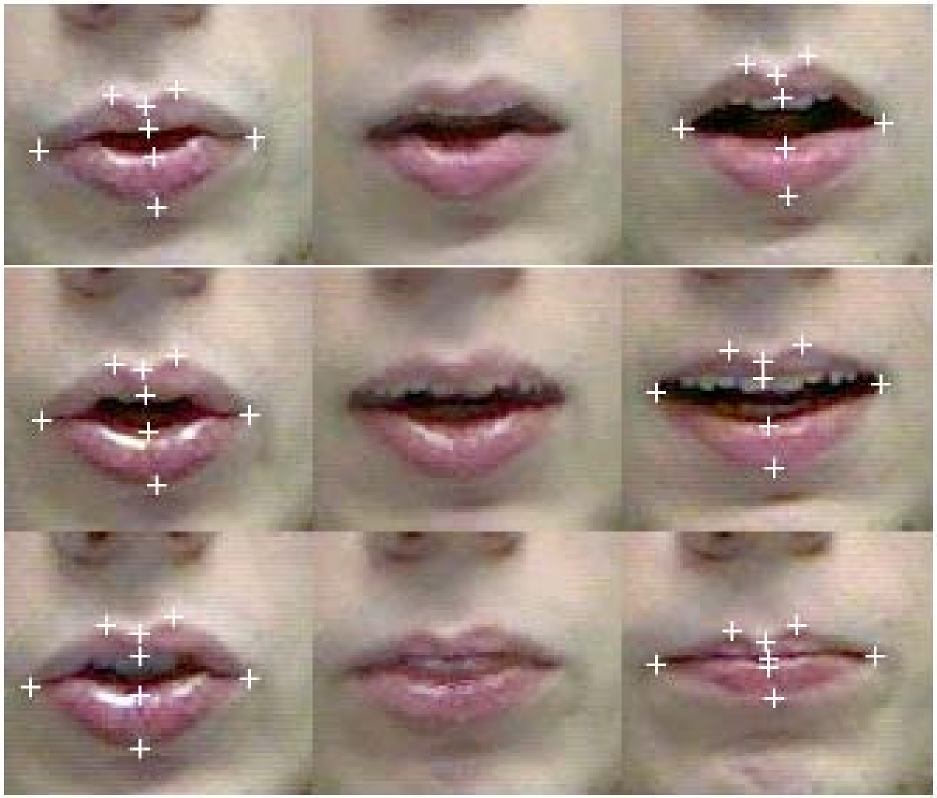


Figure 4. Synthesized viseme transitions. Central column contains transitional frames between the source and destination visemes.

- Modeling of lip motion in speech with few landmarks.
- Synthesis via motion of landmarks.

RBFs – Special Case: Thin Plate Splines

- A special class of kernels

$$\phi_i(x) = ||x - x_i||^2 \lg(||x - x_i||)$$

- Minimizes the distortion function (bending energy)

$$\int \left[\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right] dx dy.$$

  – No *scale* parameter. Gives *smoothest* results
  – Bookstein, 1989

Application: Image Morphing

- Combine shape and intensity with time parameter t
  - Just blending with amounts t produces “fade”

$$I(t) = (1-t)I_1 + tI_2$$

  - Use control points with interpolation in t

$$\bar{c}(t) = (1-t)\bar{c}_1 + t\bar{c}_2$$

  - Use c_1, c(t) landmarks to define T_1, and c_2,c(t) landmarks to define T_2

Image Morphing

- Create from blend of two warped images $I_t(\bar{x}) = (1-t)I_1\left(T_1(\bar{x})\right) + tI_2\left(T_2(\bar{x})\right)$

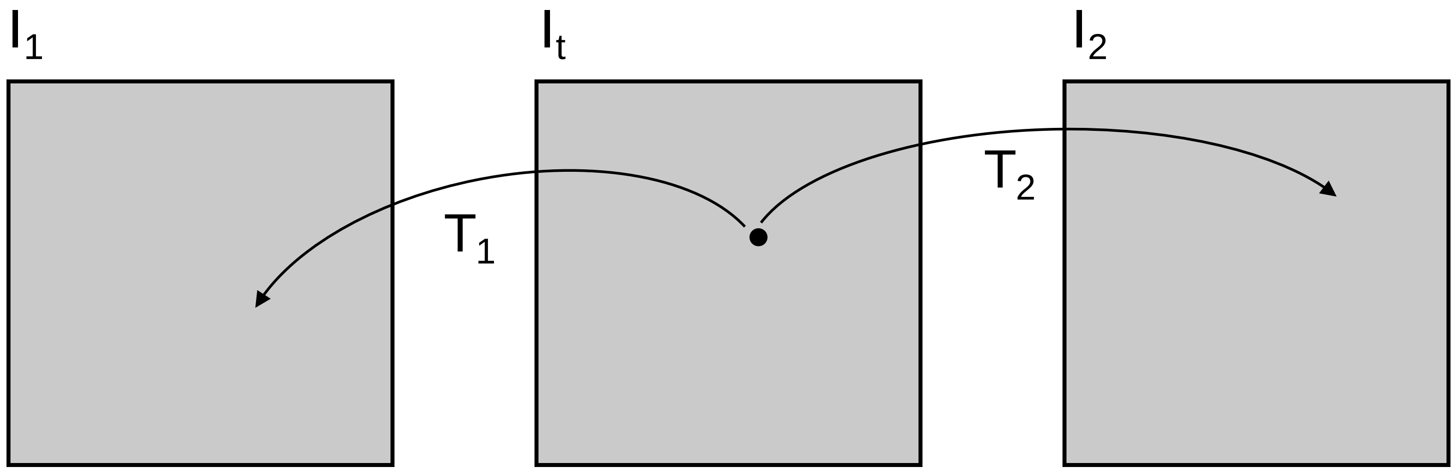


Image Morphing

Application: Image Templates/Atlases

- Build image templates that capture statistics of class of images
  – Accounts for shape and intensity
  – Mean and variability
- Purpose
  – Establish common coordinate system (for comparisons)
  – Understand how a particular case compares to the general population

Templates – Formulation

- N landmarks over M different subjects/samples

Correspondences

Images

$$I^j(\bar{x}) \qquad \bar{c}_i^j \qquad \begin{pmatrix} \bar{c}_1^1 & \dots & \bar{c}_N^1 \\ \vdots & & \vdots \\ \bar{c}_1^M & \dots & \bar{c}_N^M \end{pmatrix}$$

Mean of correspondences (template)

$$\hat{c}_i = \frac{1}{M} \sum_{j=1}^{M} \bar{c}_i^j$$

Transformations from mean to subjects

$$\bar{c}_i^j = T^j(\hat{c}_i)$$

Templated image

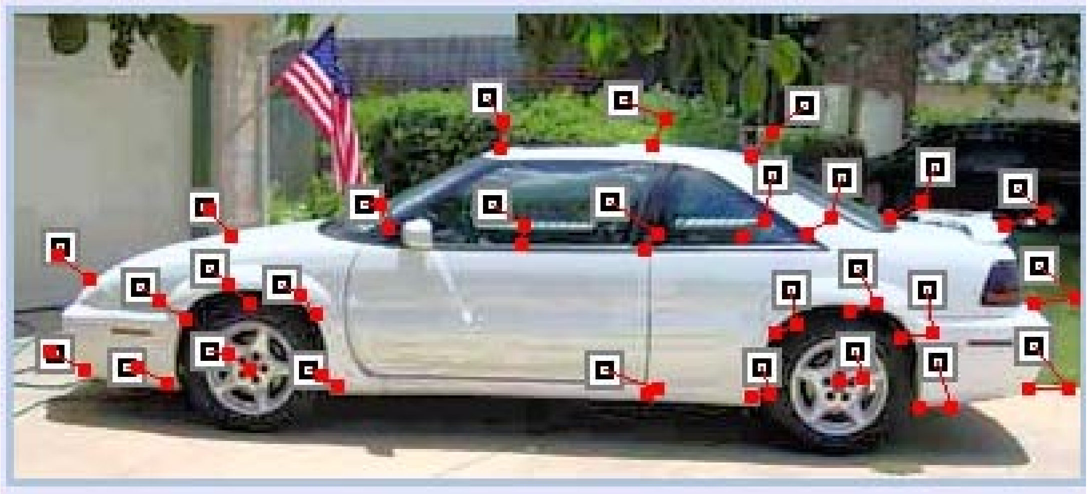
$$\hat{I}(\bar{x}) = \frac{1}{M} \sum_j I^j\left(T^j(\bar{x})\right)$$

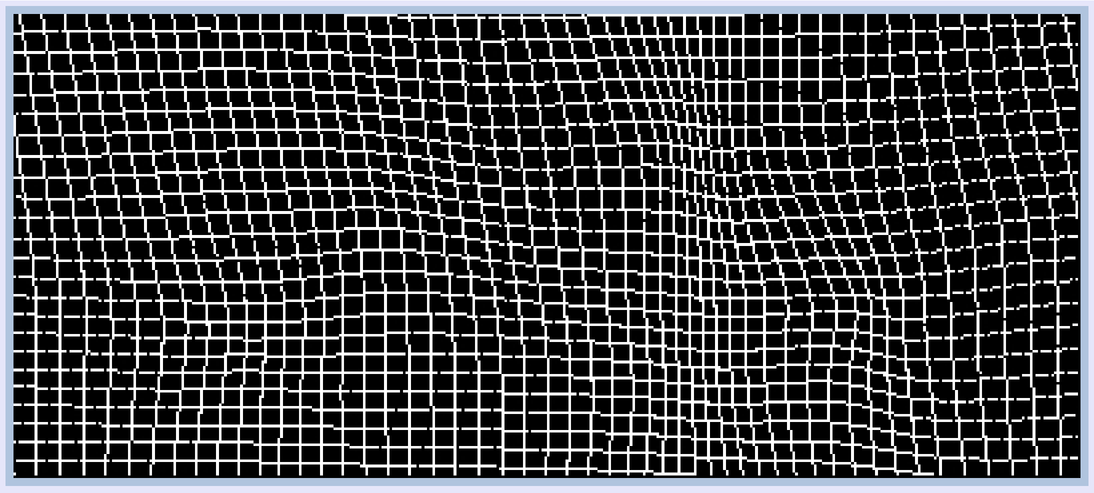
Cars



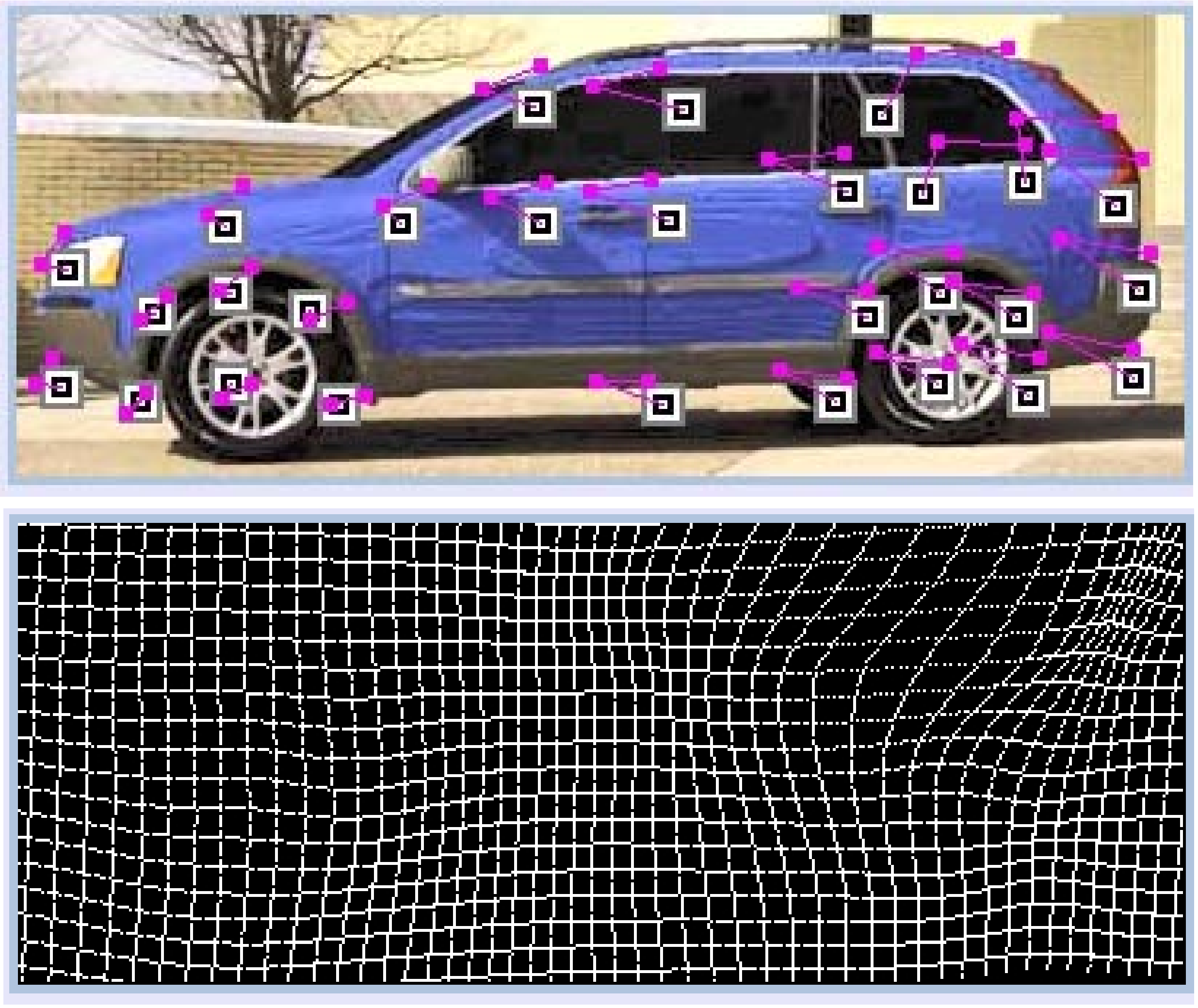



Car Landmarks and Warp

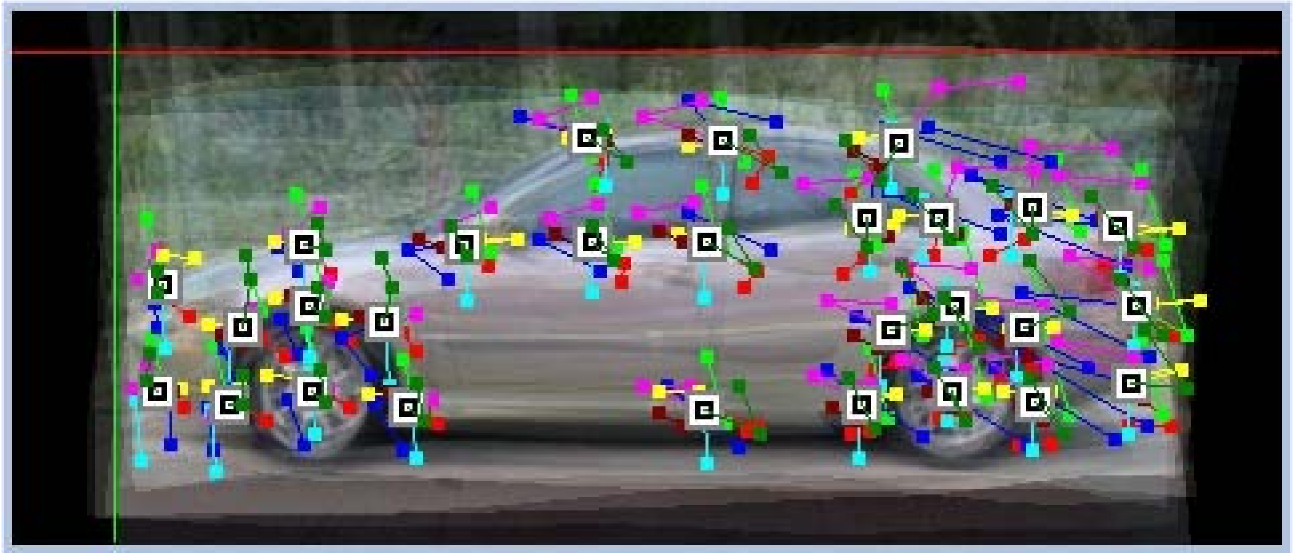




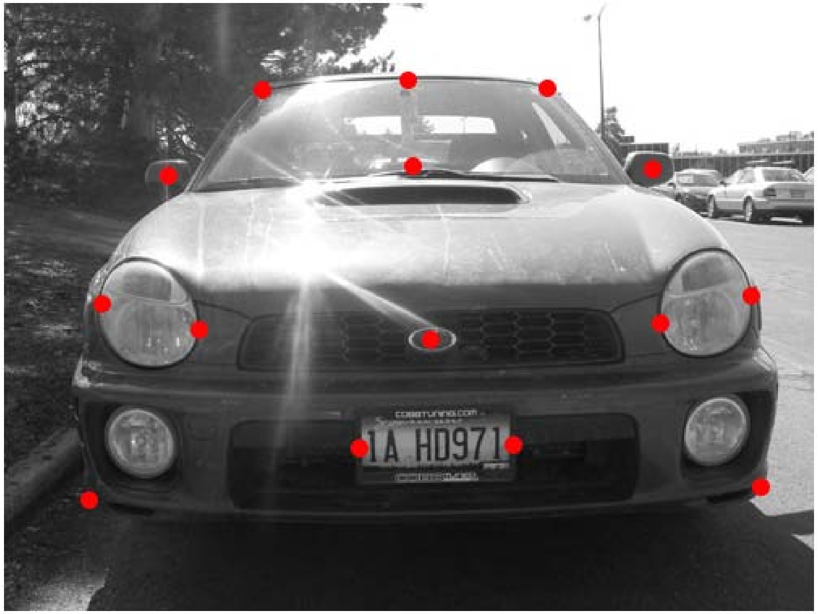
Car Landmarks and Warp



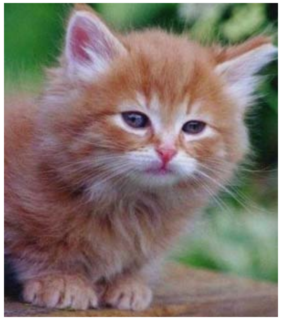
Car Mean



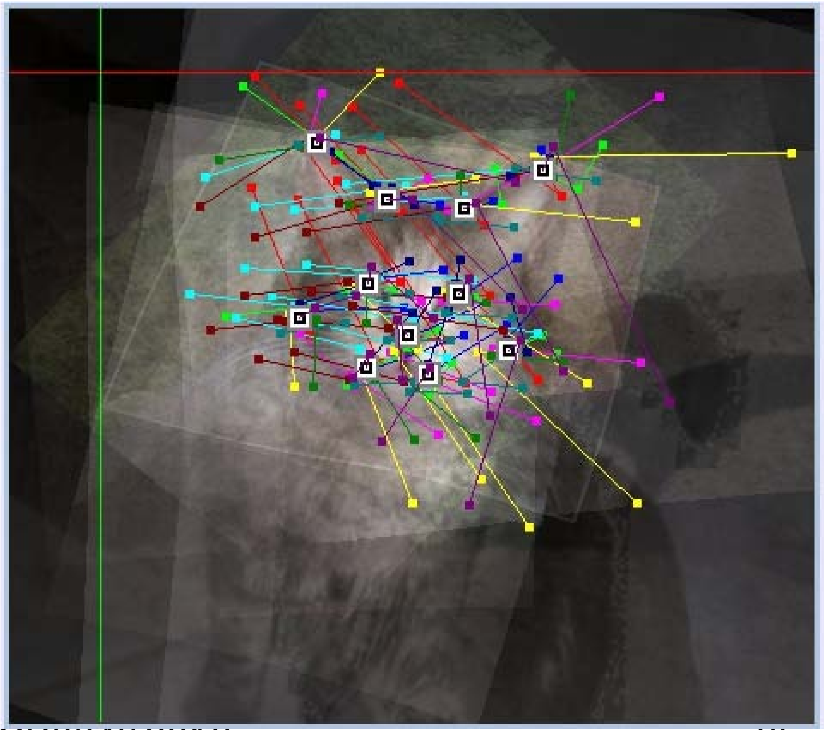
Cars



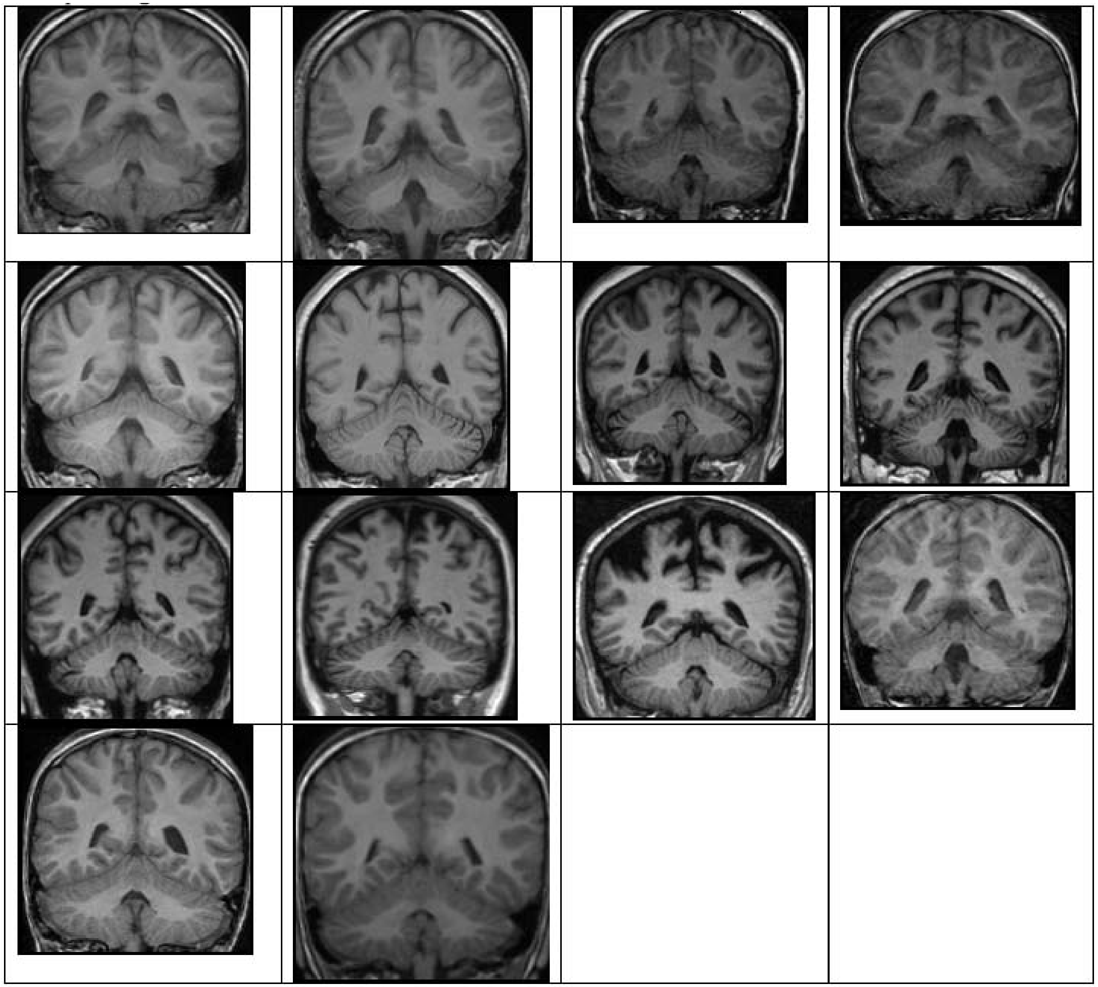
Cats

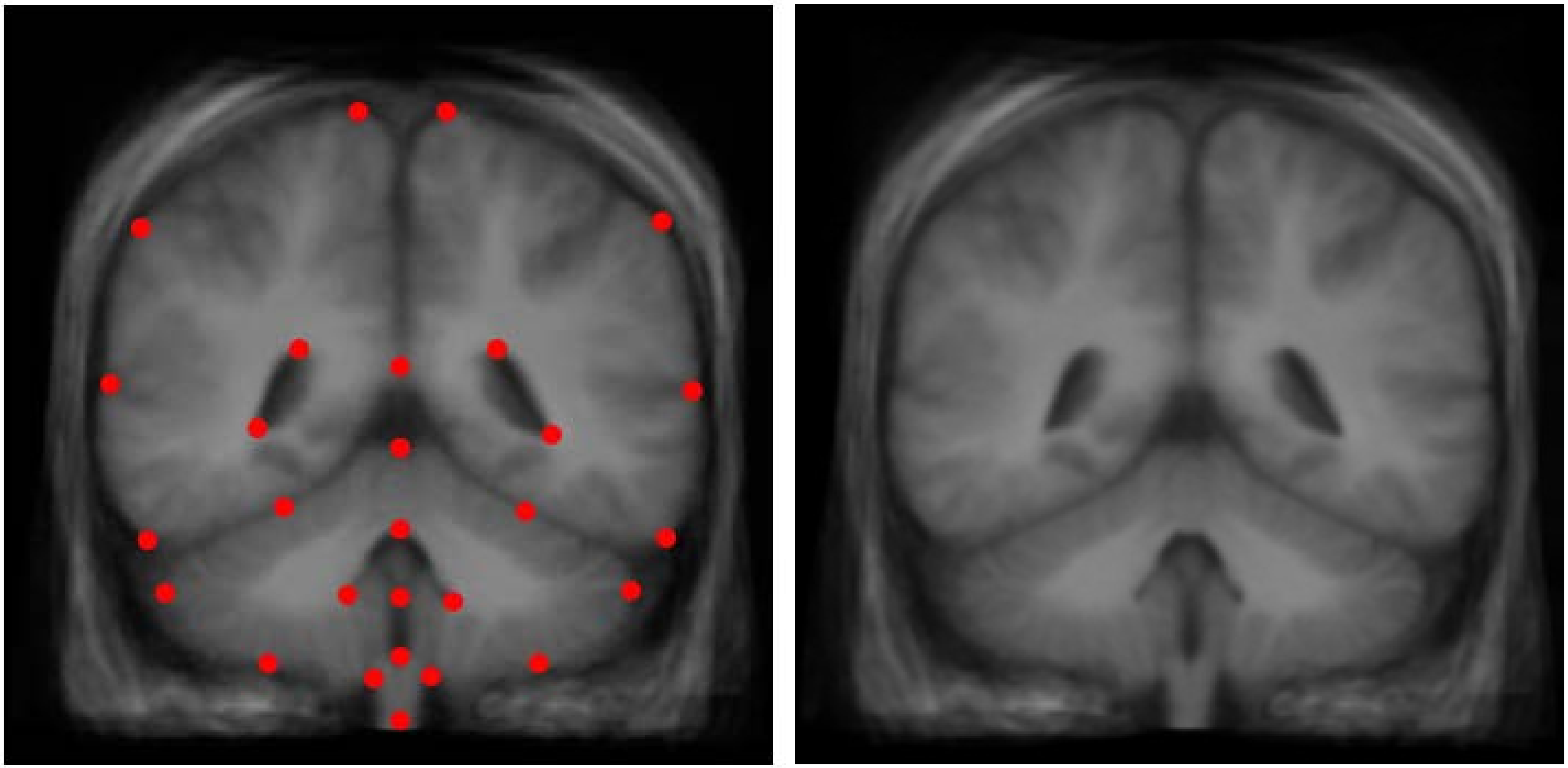




Brains



Brain Template

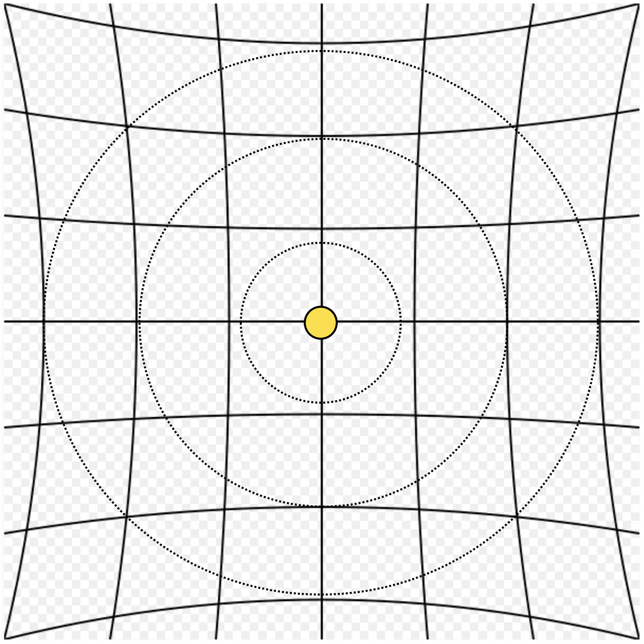


APPLICATIONS

Warping Application: Lens Distortion

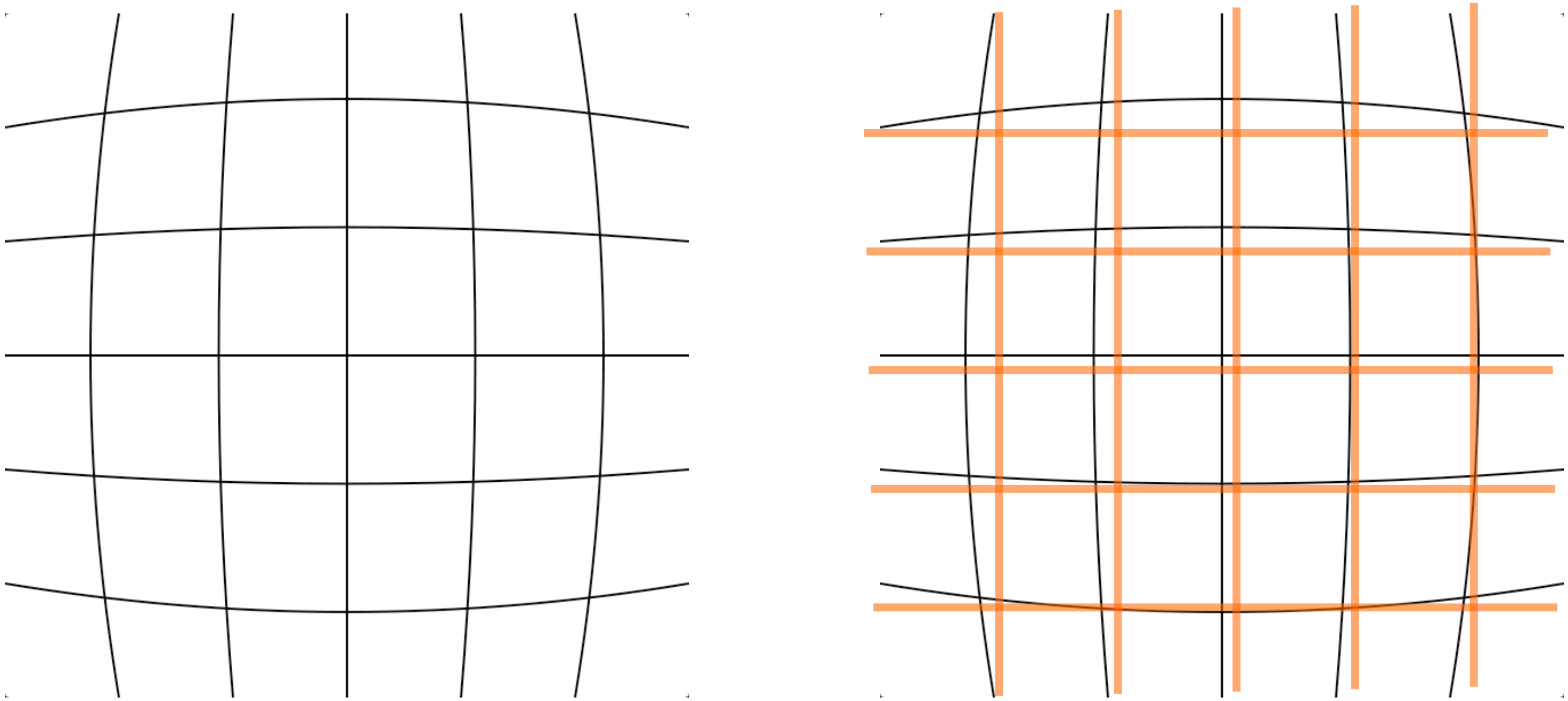
- Radial transformation – lenses are generally circularly symmetric
  - Optical center is known
  - Model of transformation:

$$\bar{x}' = \bar{x}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots\right)$$



Correspondences

- Take picture of known grid – crossings



- Measure set of landmark pairs → Estimate transformation, correct images

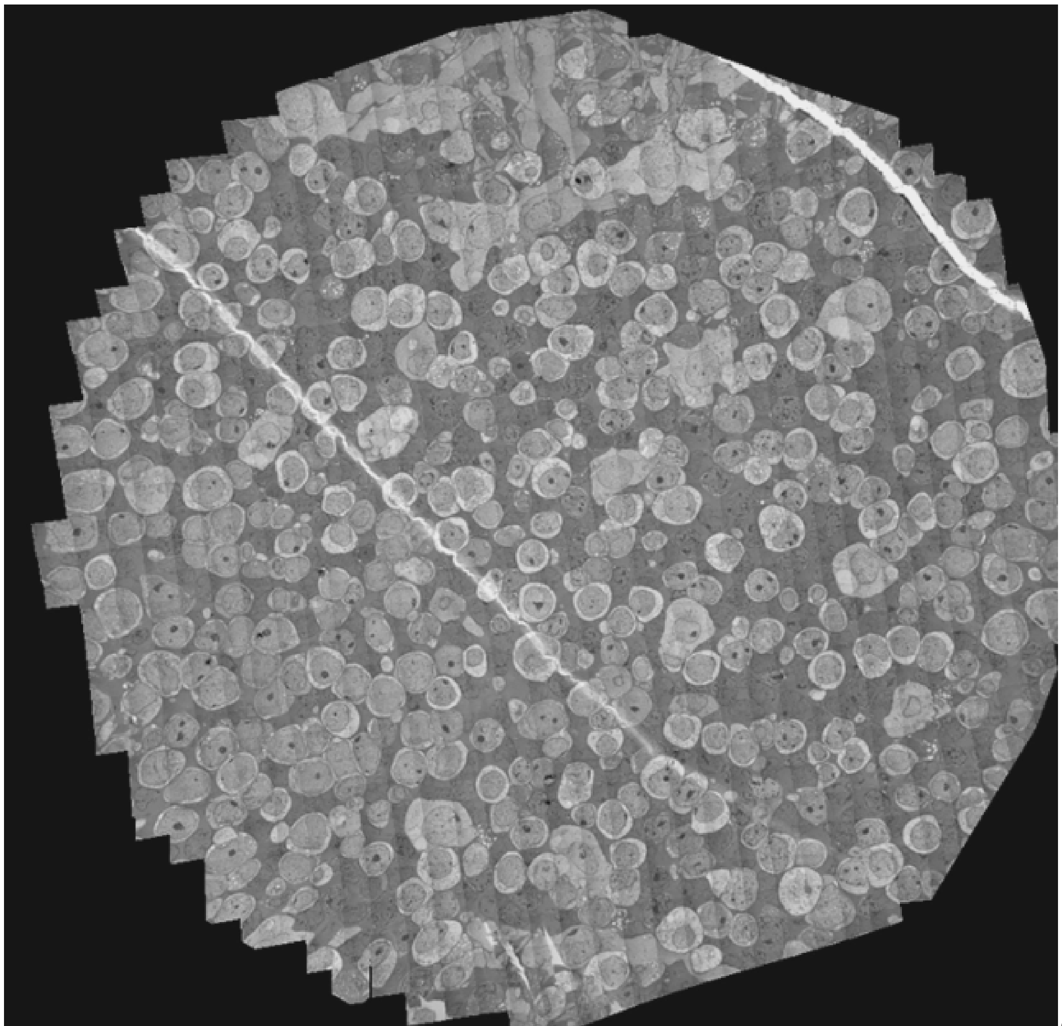
Image Mosaicing

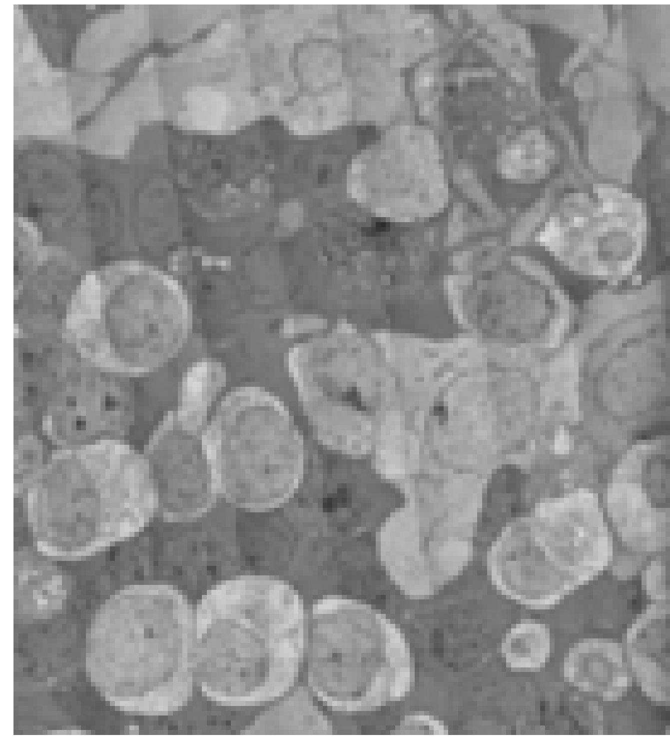
- Piecing together images to create a larger mosaic
- Doing it the old fashioned way
  - Paper pictures and tape
  - Things don’t line up
  - Translation is not enough
- Need some kind of warp
- Constraints
  - Warping/matching two regions of two different images only works when…

Applications

Saint-Guénolé Church of Batz-sur-Mer Equirectangular 360º by Vincent Montibus

Microscopy (Morane Eye Inst, UofU, T. Tasdizen et al.)





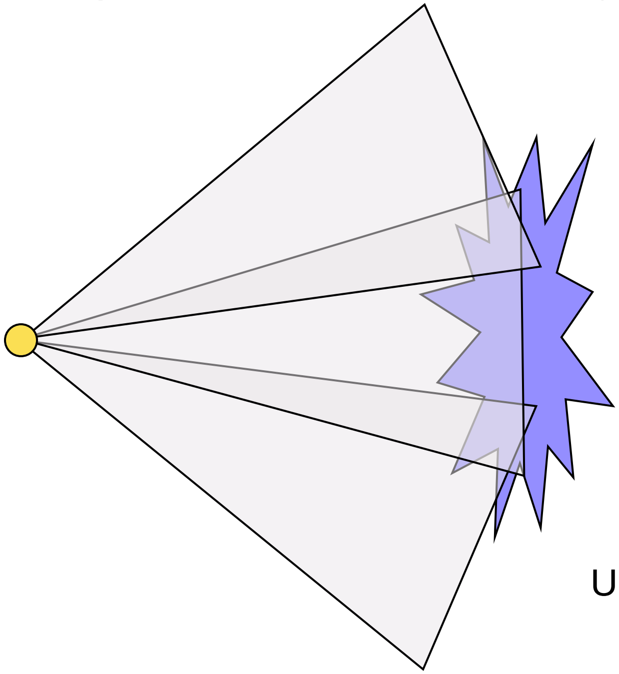

SONY
NEX-5


SONY
NEX-5

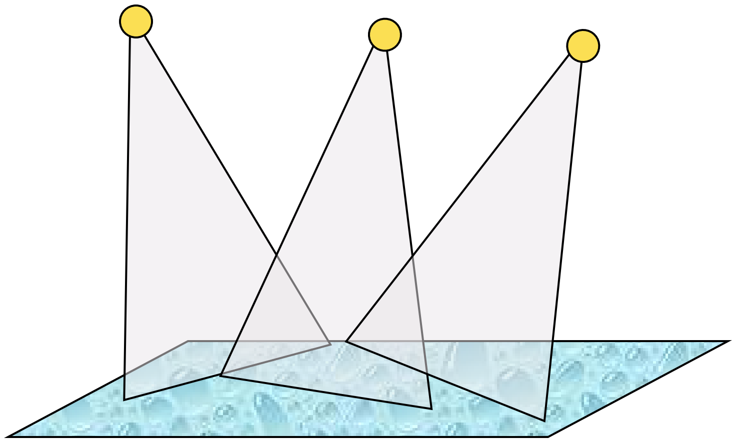
Special Cases

- Nothing new in the scene is uncovered in one view vs another
  - No ray from the camera gets behind another

1) Pure rotations–arbitrary scene

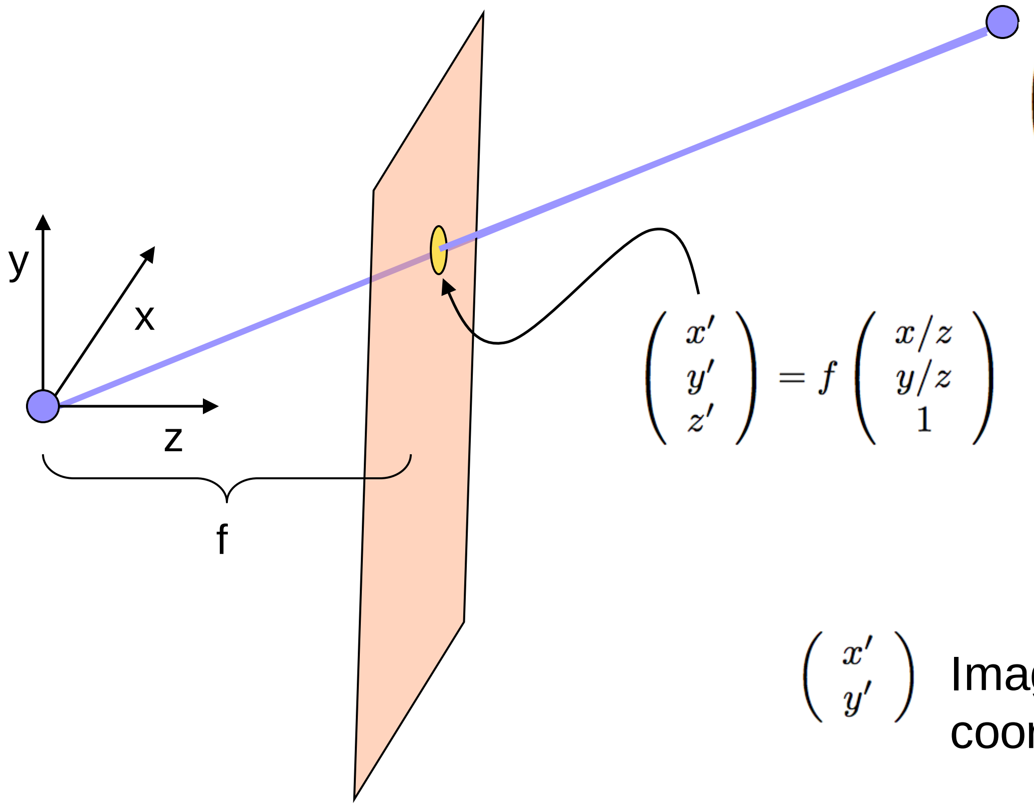


2) Arbitrary views of planar surfaces



3D Perspective and Projection

- Camera model



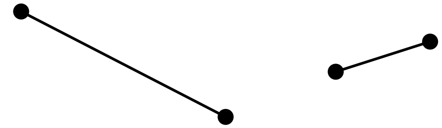
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = f \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix}$$

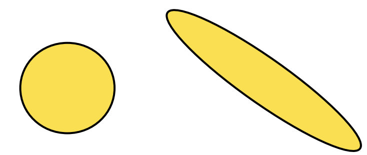
$\begin{pmatrix} x' \\ y' \end{pmatrix}$ Image coordinates

Perspective Projection Properties

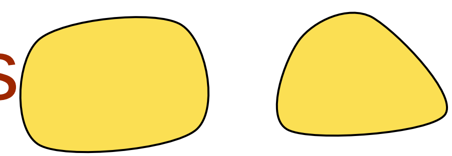
- Lines to lines (linear)



- Conic sections to conic sections



- Convex shapes to convex shapes



- Foreshortening

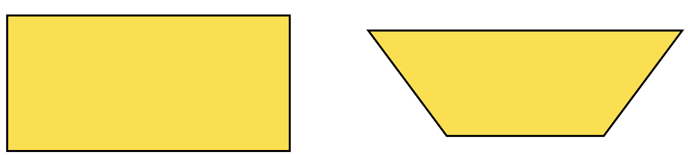


Image Homologies

- Images taken under cases 1,2 are perspectively equivalent to within a linear transformation
  - Projective relationships – equivalence is

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \equiv \begin{pmatrix} d \\ e \\ f \end{pmatrix} \Longleftrightarrow \begin{pmatrix} a/c \\ b/c \\ 1 \end{pmatrix} = \begin{pmatrix} d/f \\ e/f \\ 1 \end{pmatrix}$$

Transforming Images To Make Mosaics

Linear transformation with matrix P

$$\bar{x}^* = P\bar{x} \qquad P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{pmatrix}$$

$$\begin{aligned} x^* &= p_{11}x + p_{12}y + p_{13} \\ y^* &= p_{21}x + p_{22}y + p_{23} \\ z^* &= p_{31}x + p_{32}y + 1 \end{aligned}$$

Perspective equivalence

$$\begin{aligned} x' &= \frac{p_{11}x+p_{12}y+p_{13}}{p_{31}x+p_{32}y+1} \\ y' &= \frac{p_{21}x+p_{22}y+p_{23}}{p_{31}x+p_{32}y+1} \end{aligned}$$

Multiply by denominator and reorganize terms

$$\begin{aligned} p_{31}xx' + p_{32}yx' - p_{11}x - p_{12}y - p_{13} &= -x' \\ p_{31}xy' + p_{32}yy' - p_{21}x - p_{22}y - p_{23} &= -y' \end{aligned}$$

Linear system, solve for P

$$\begin{pmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' \\ & & & \vdots & & & & \\ -x_N & -y_N & -1 & 0 & 0 & 0 & x_Nx_N' & y_Nx_2' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y_1' & y_1y_1' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y_2' & y_2y_2' \\ & & & \vdots & & & & \\ 0 & 0 & 0 & -x_N & -y_N & -1 & x_Ny_N' & y_Ny_N' \end{pmatrix} \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{23} \\ p_{23} \\ p_{31} \\ p_{32} \end{pmatrix} = \begin{pmatrix} -x_1' \\ -x_2' \\ \vdots \\ -x_N' \\ -y_1' \\ -y_2' \\ \vdots \\ -y_N' \end{pmatrix}$$

Image Mosaicing

4 Correspondences

5 Correspondences

6 Correspondences

Mosaicing Issues

- Need a canvas (adjust coordinates/origin)
- Blending at edges of images (avoid sharp transitions)
- Adjusting brightnesses
- Cascading transformations