Geometric Transformations

Srikumar Ramalingam
School of Computing
University of Utah

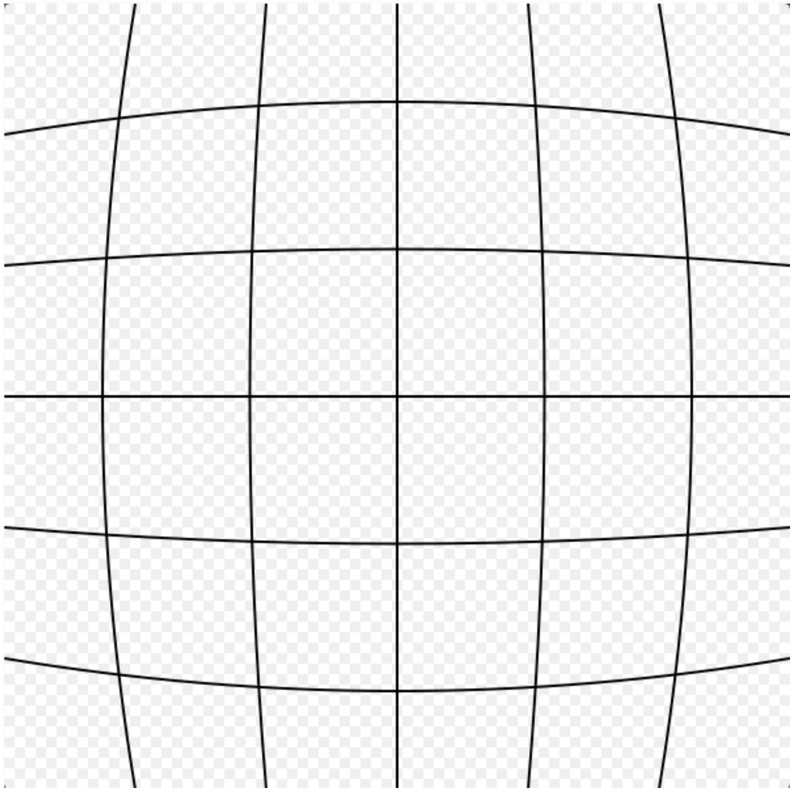
[Slides borrowed from Ross Whitaker and Jinxiang Chai (TAMU)]

Geometric Transformations

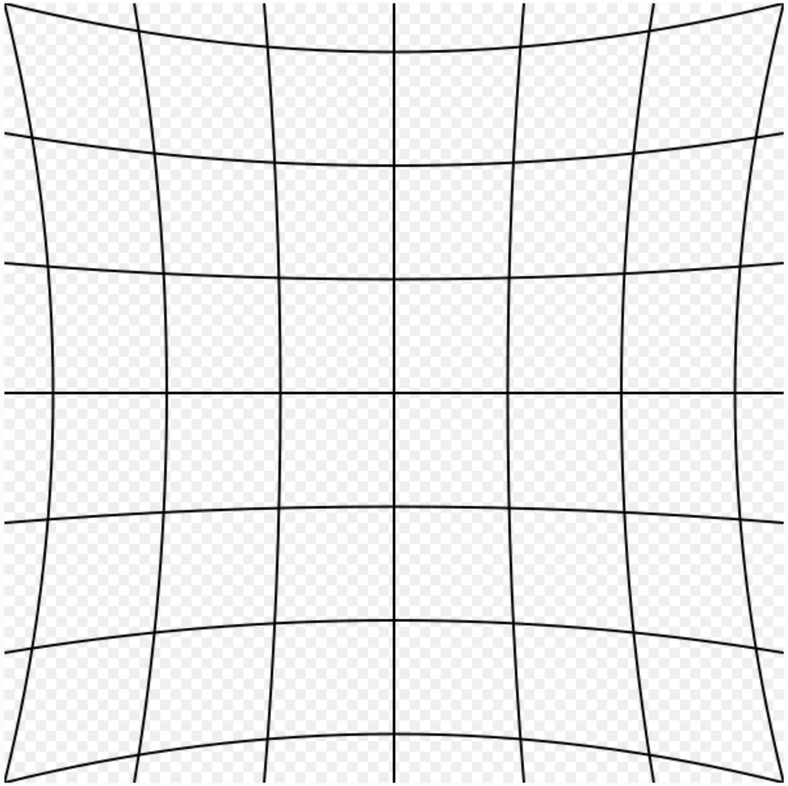
- Grayscale transformations -> operate on range/output
- Geometric transformations -> operate on image domain
  - Coordinate transformations
  - Moving image content from one place to another
- Two parts:
  - Define transformation
  - Resample grayscale image in new coordinates

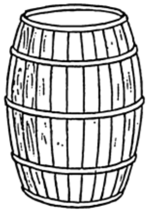
Geom Trans: Distortion From Optics

Barrel Distortion



Pincushion Distortion





Straight lines bulge out as in a barrel

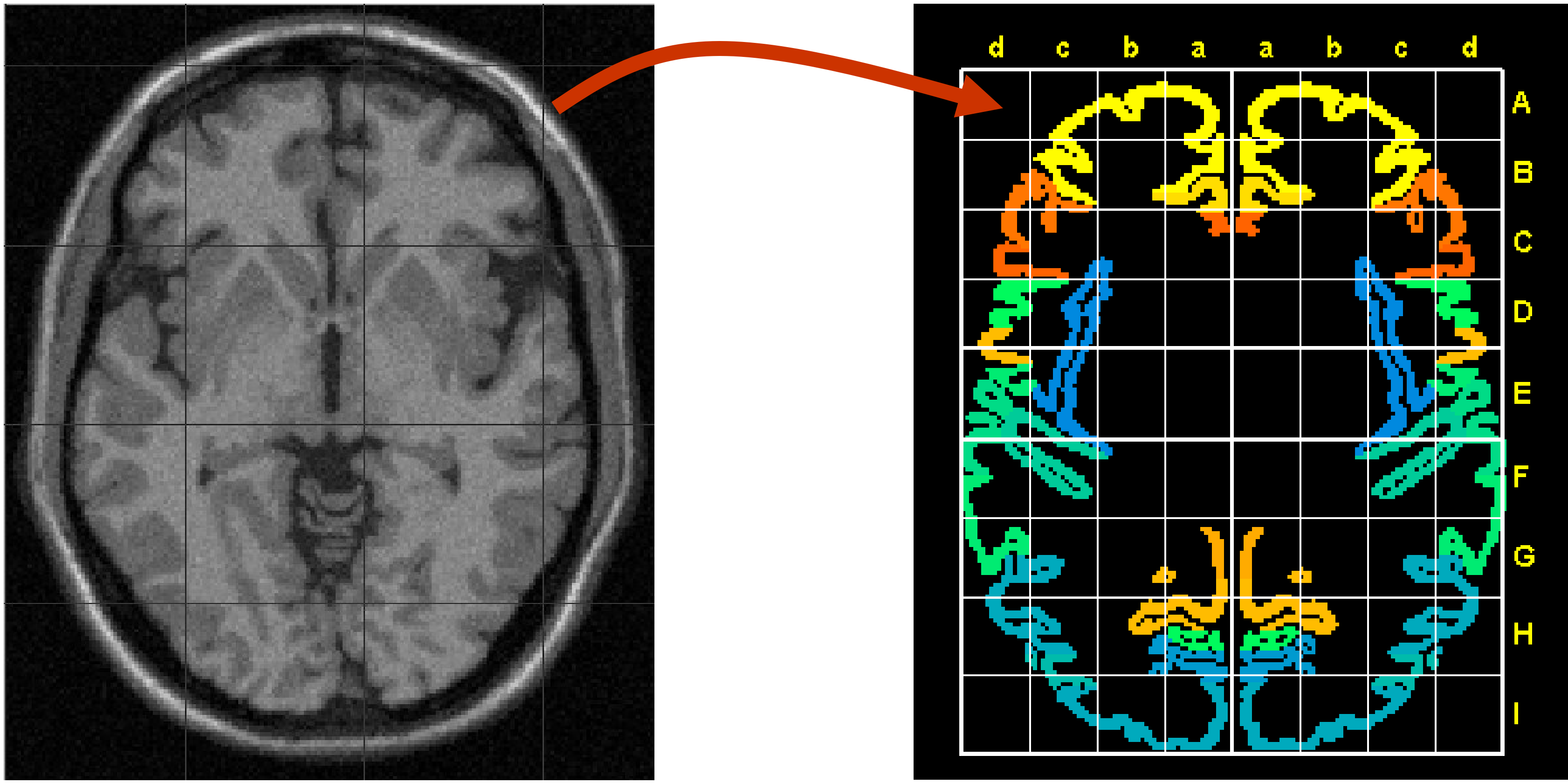
Corners of points form elongated points as in a cushion

Geom Trans: Distortion From Optics

Barrel Distortion

Pincushion Distortion

Geom. Trans.: Brain Template/Atlas



Atlas provides an invariant reference frame and allows one to match or compare different brains.

Geom. Trans: Mosaicing

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

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

New image Old image

Domain Mappings Formulation

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

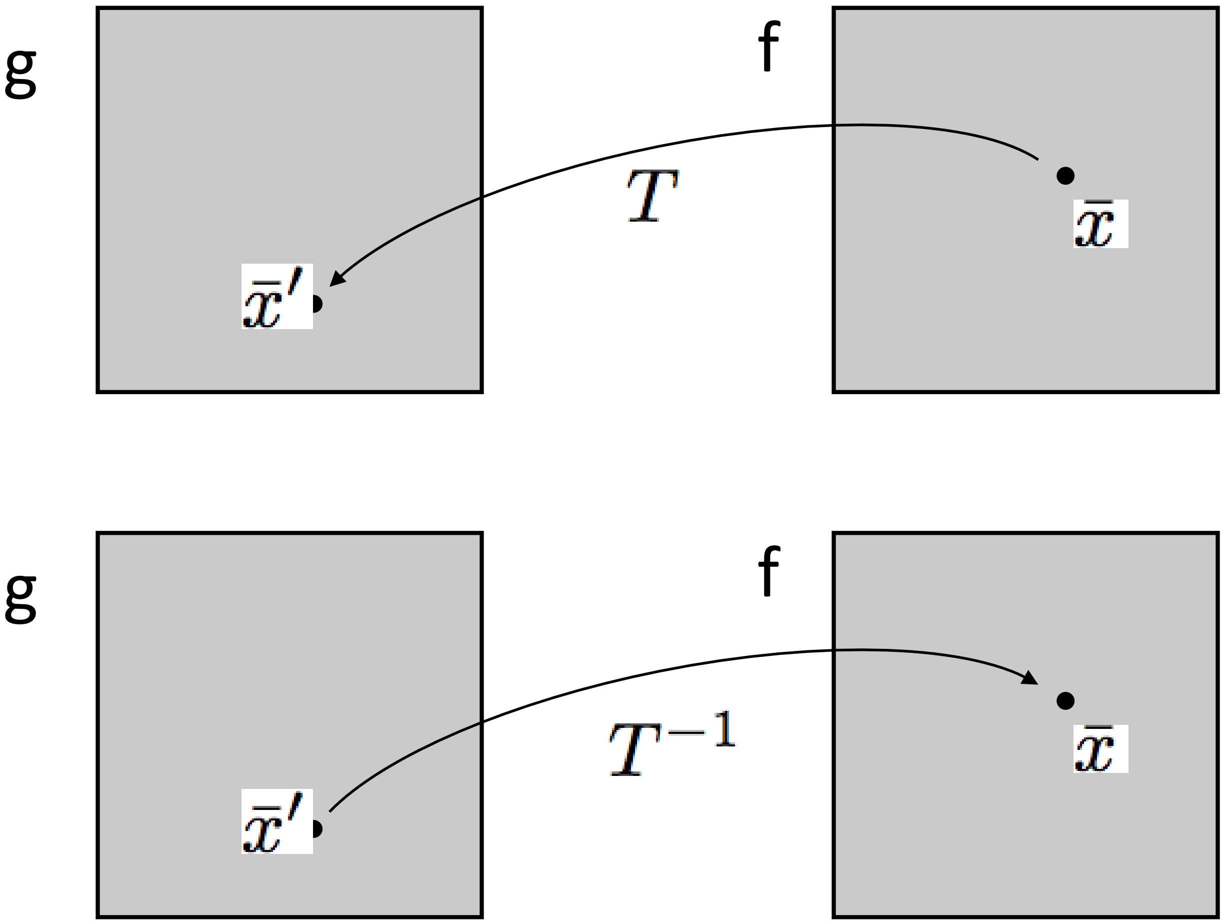
Vector notation is convenient.
Bar used some times, depends on context.

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

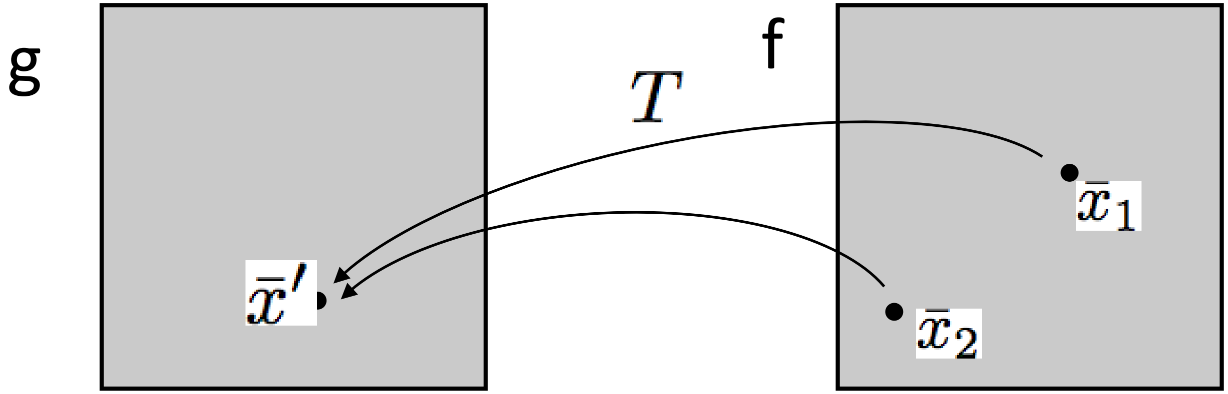
T may or may not have an inverse. If not, it means that information was lost.

$$g(\bar{x}') = f(\bar{x})$$

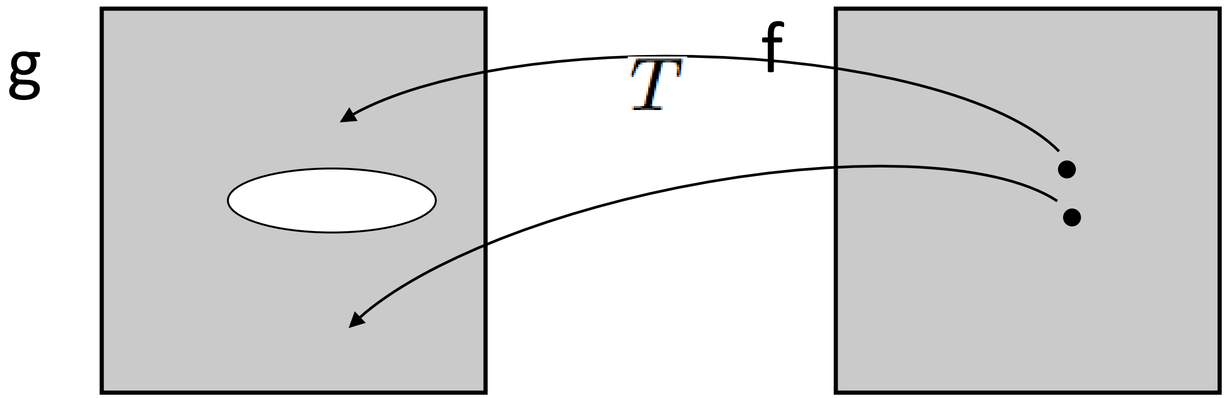
Domain Mappings



No Inverse?



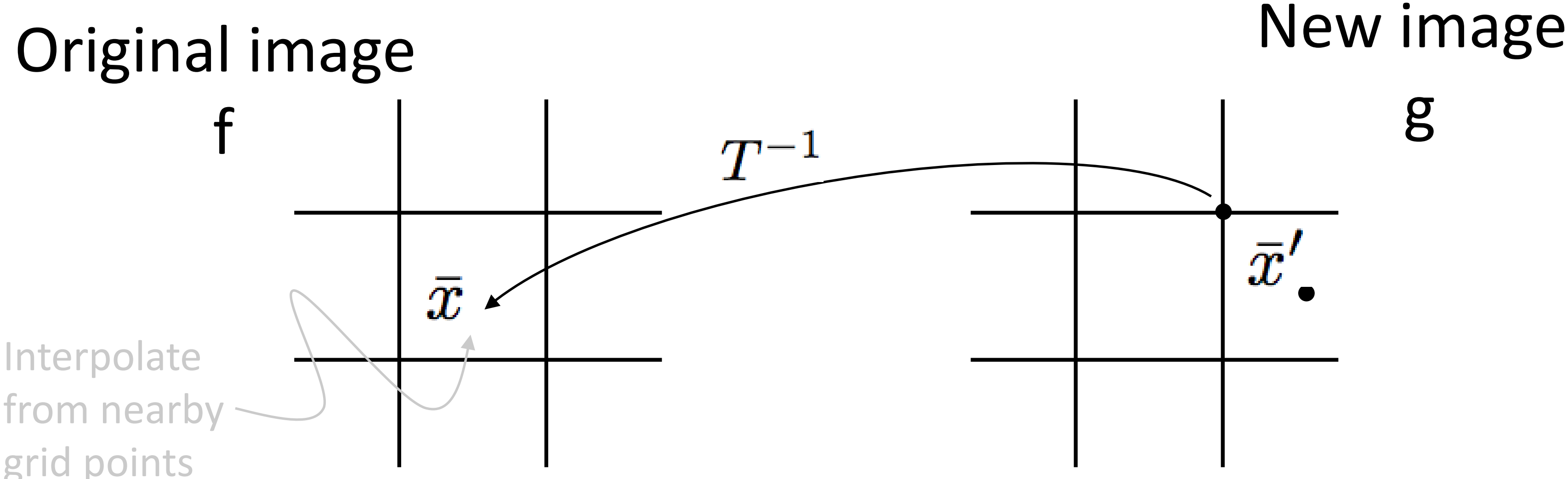
Not "one to one"



Not "onto" - doesn't cover g

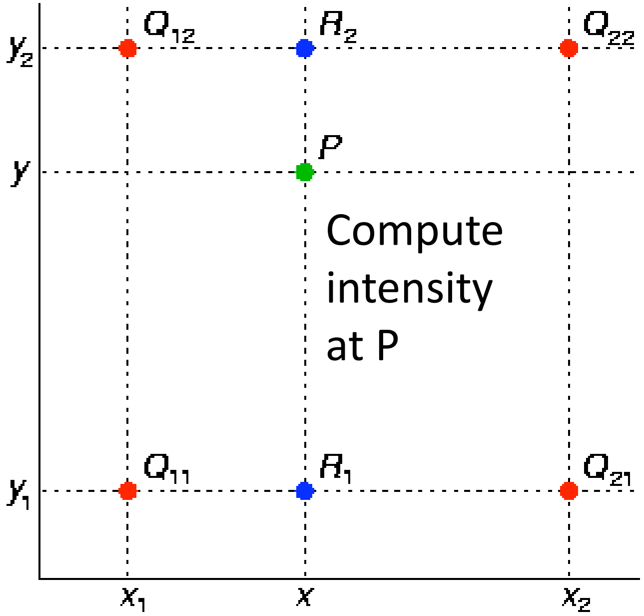
Implementation – Two Approaches

- Backward mapping
  - T^{-1}() takes you from coordinates in g() to coordinates 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
- We are given intensity values at $Q_{12}, Q_{22}, Q_{11}, \text{and } Q_{21}$.
- First, we compute intensity values at R_1 and R_2 using linear interpolation. Then we compute at P using linear interpolation.



Source: Wikipedia

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

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

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

Bilinear Interpolation

- Not linear in x, y

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

Bilinear Interpolation

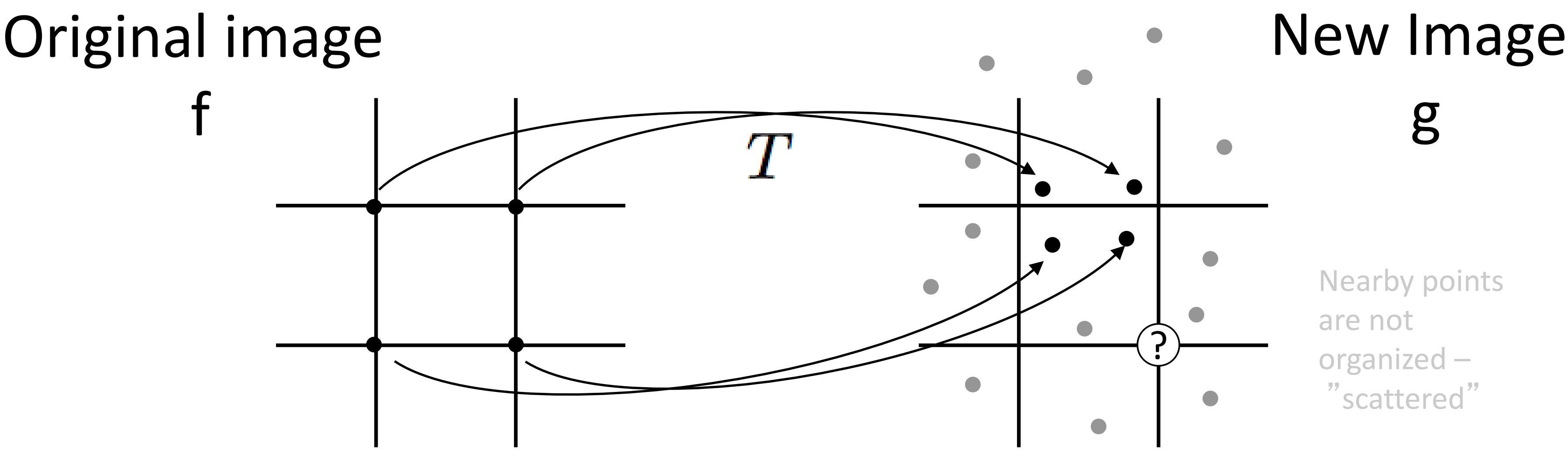
- Convenient form
  - Normalize to unit grid [0,1]x[0,1]

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

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

Implementation – Two Approaches

- Forward Mapping
  - T() takes you from coordinates in f() to coordinates in g()
  - You have f() on grid, but you need g() on grid
  - Push grid samples onto g() grid and do interpolation from unorganized data (kernel)



Scattered Data Interpolation With Kernels Shepard's method

- Define kernel
  - Falls off with distance, radially symmetric

Kernel examples

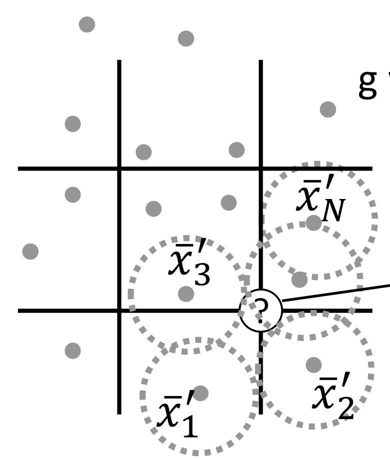
$$K(\bar{x}_1, \bar{x}_2) = \frac{1}{2\pi\sigma^2} e^{-\frac{|\bar{x}_1 - \bar{x}_2|^2}{2\sigma^2}}$$

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

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

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

$$w_j = K(|\bar{x}' - \bar{x}_j'|)$$



New image g with known values at locations $\bar{x}_1', \bar{x}_2', \dots, \bar{x}_N'$

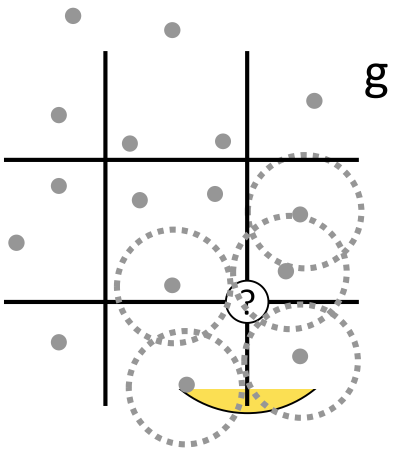
Intensity at $\bar{x}'$?

Modified Shepard's Method

- If points are dense enough
  - Truncate kernel
  - For each point in g()
    - Form a small circle around it in g() – beyond which truncate
    - Put weights and data onto grid in g()
  - Value at a specific location x'.

$$w_j = \frac{\max(0, R - |\bar{x}' - \bar{x}_j'|)}{R\,|\bar{x}' - \bar{x}_j'|}$$

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



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

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

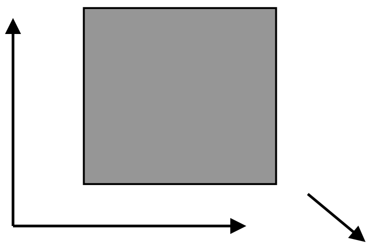
3x1 homogeneous vectors from now onwards

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

Special Cases of Linear Transformations

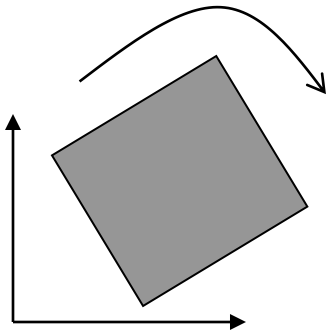
- Translation

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



- Rotation

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



- Scaling

$$A = \begin{pmatrix} p & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

  - Include forward and backward rotation for arbitrary axis

Special Cases of Linear Transformations

- Skew matrix

$$A = \begin{pmatrix} 1 & p & 0 \\ q & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Reflection matrix (special case of scaling)

$$A = \begin{pmatrix} p & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{p} = -1, \boldsymbol{q} = 1$$

$$\boldsymbol{p} = \ \ 1, \boldsymbol{q} = -1$$

Linear Transformations

- Also called "affine"
  - 6 parameters

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

- Rigid -> 3 parameters

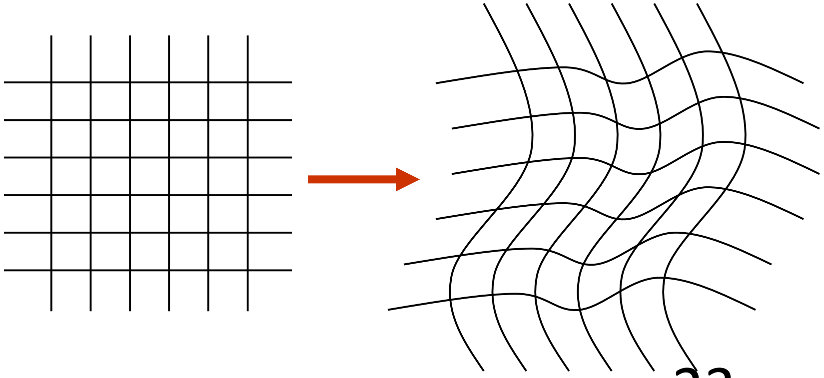
$$A = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & x_0 \\ \sin(\theta) & \cos(\theta) & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Invertibility

- Invertibility $T^{-1}(\bar{x}) = A^{-1}\bar{x}$
  - Invert matrix
- What does it mean if A is not invertible?

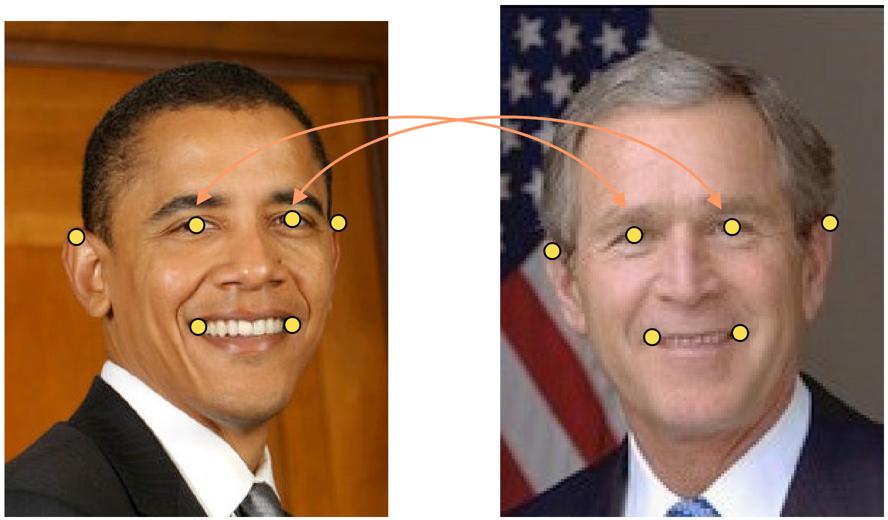
Other Transformations

- 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
    - Affine - 6 parameters



Correspondences

- Also called “landmarks” or “fiducials”



$\bar{c}_1, \bar{c}'_1$

$\bar{c}_2, \bar{c}'_2$

$\bar{c}_3, \bar{c}'_3$

$\bar{c}_4, \bar{c}'_4$

$\bar{c}_5, \bar{c}'_5$

$\bar{c}_6, \bar{c}'_6$

Transformations/Control Points Strategy

- Define a functional representation for T with k parameters (B) $T(\beta, \bar{x}) \quad \beta = (\beta_1, \beta_2, \ldots, \beta_K)$
- Define (pick) N correspondences
- Find B so that

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

- If over-constrained (K < 2N) then solve

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

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 unknowns

Write As Linear System

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

$$Ax = b$$

Transformation parameter vector, not to be confused with 3x1 homogenous vectors

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

x – unknown parameters of the transformation

b – the primed correspondences

Case 1: Linear Systems

$$Ax = b$$

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

Simple case: A is square (M=N) and invertible (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

Case 2: Linear Systems

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

Case 3: Linear Systems

- M > N
  - System is over-constrained
  - No solution
- Approach
  - Find solution that is best compromise
  - Minimize squared error (least squares)

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

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

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 $x = VW^*U^Tb$

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

- W* is obtained by replacing every non-zero entry by its reciprocal and transposing the matrix.
- Properties:
  - Under-constrained: solution with shortest overall length
  - Over-constrained: least squares solution