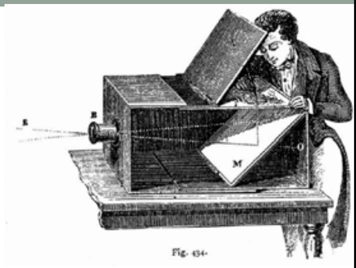
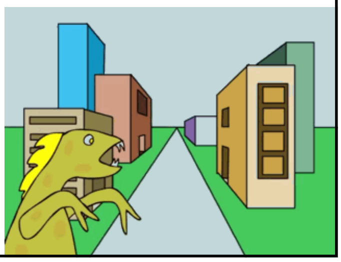
DIGITAL IMAGE PROCESSING



Lecture 8
Geometric transformations
Tammy Riklin Raviv
Electrical and Computer Engineering
Ben-Gurion University of the Negev

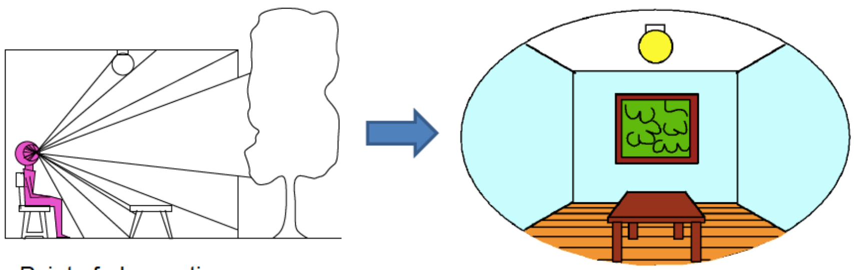


Geometry Transformations

- Pinhole camera model
- Homogeneous coordinates
- 2D geometric transformations
- Fitting and Alignment – brief intro

Image Formation: 3D -> 2D

3D world *2D image*

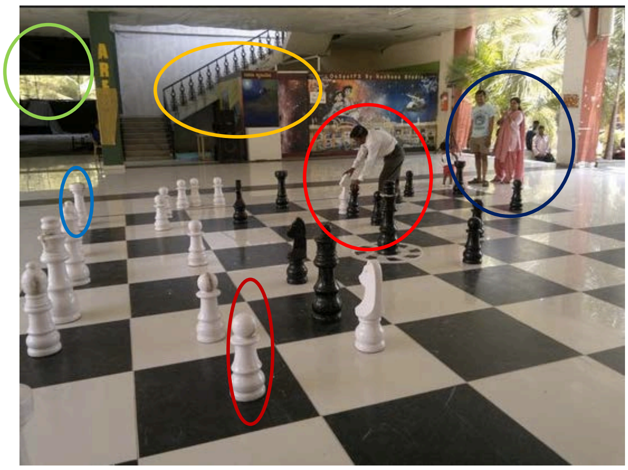


Point of observation

Figures © Stephen E. Palmer, 2002

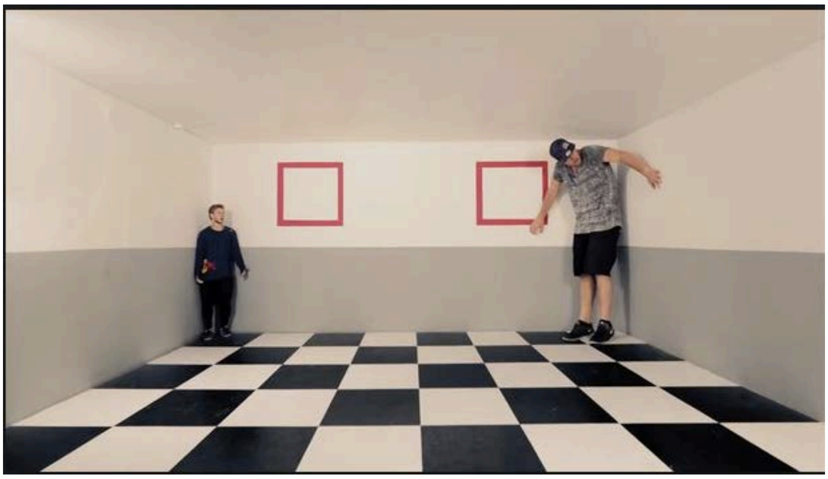
2D -> 3D Scene Understanding

Single view Geometry

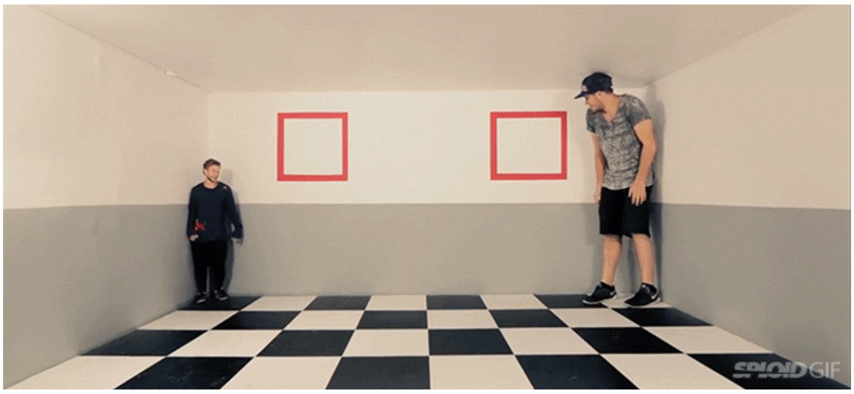


https://media-cdn.tripadvisor.com/media/photo-s/03/ae/fa/45/floor-chess-board.jpg

2D -> 3D Scene Understanding



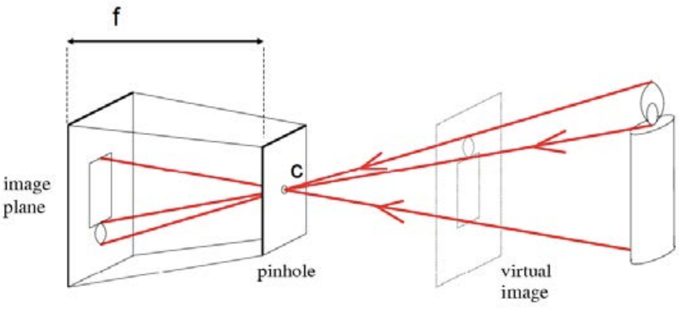
Jason Paul's Freerunning Illusions



https://www.gizmodo.com.au/2015/08/these-super-fun-illusions-really-messes-with-your-perspective/

Jason Paul's Freerunning Illusions

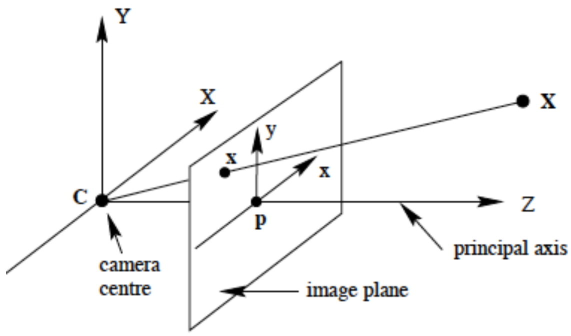
Pinhole Camera Model



f = focal length
c = center of the camera

Figure from Forsyth

Pinhole Camera Geometry



Pinhole Camera Geometry

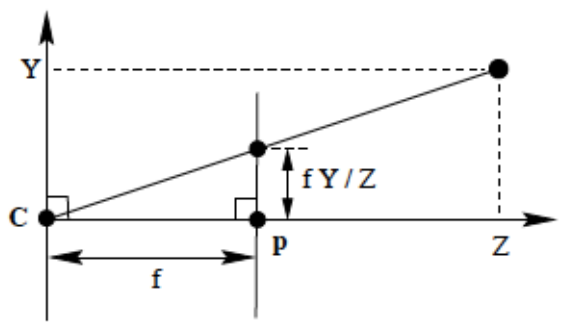


Image Formation

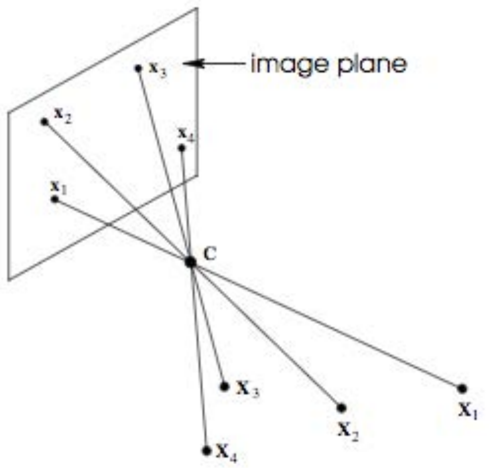


Image Formation

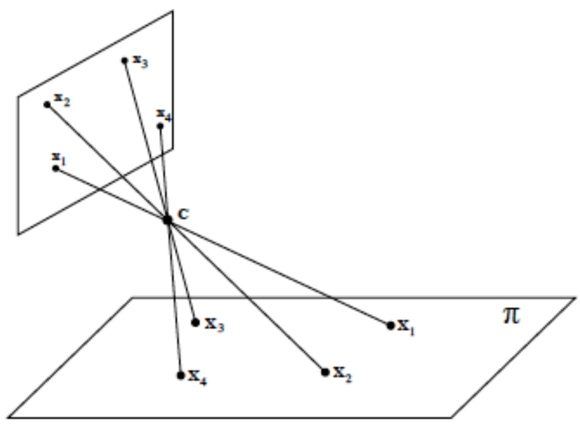


Image Formation

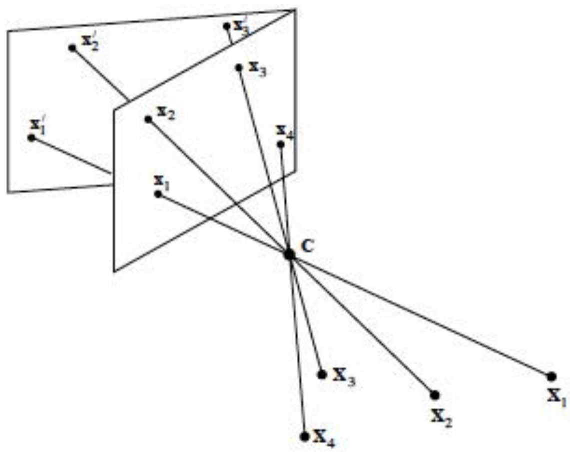


Image Formation

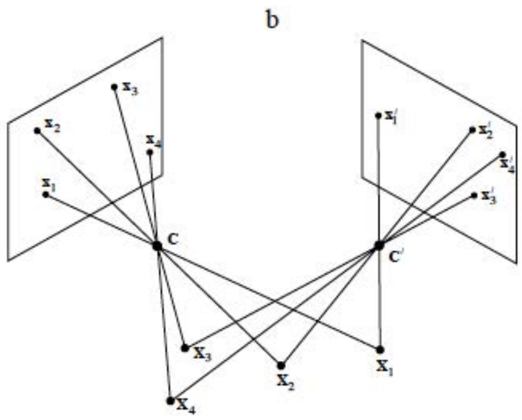
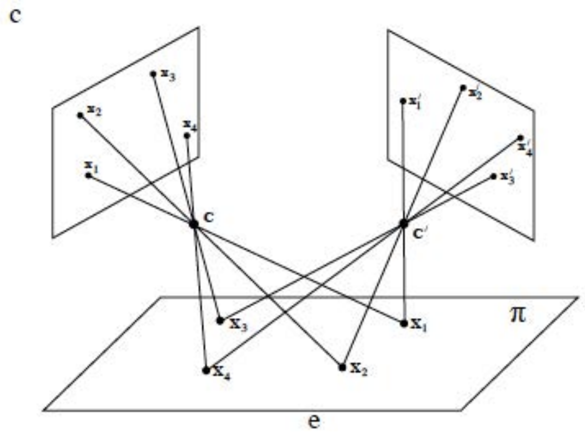
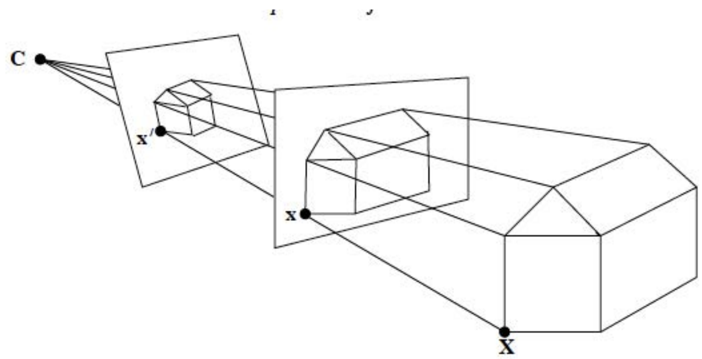


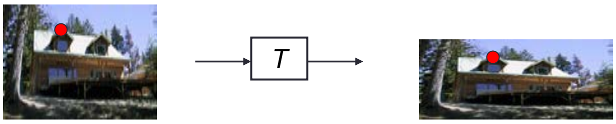
Image Formation



Cone of Rays



Parametric (global) transformations



p = (x,y) **p’** = (x’,y’)

Transformation T is a coordinate-changing machine:
p’ = *T*(p)

What does it mean that *T* is global?

- *T* is the same for any point p
  T can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix
p’ = **T**p

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Homogeneous Coordinates

Converting to *homogeneous* coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogeneous Coordinates

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

One extra step:

$$x' = u/w$$
$$y' = v/w$$

Homogeneous Coordinates

Invariant to scaling

$$k \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

Homogeneous Coordinates

Cartesian Coordinates

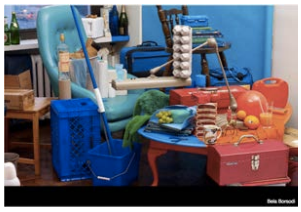
Point in Cartesian is ray in Homogeneous

Homogeneous Coordinates

- Line equation: ax + by + c = 0 $$line_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}$$
- Append 1 to pixel coordinate to get homogeneous coordinate $$p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$
- Line given by cross product of two points $$line_{ij} = p_i \times p_j$$
- Intersection of two lines given by cross product of the lines $$q_{ij} = line_i \times line_j$$

What are Geometric Transformations?

Translation



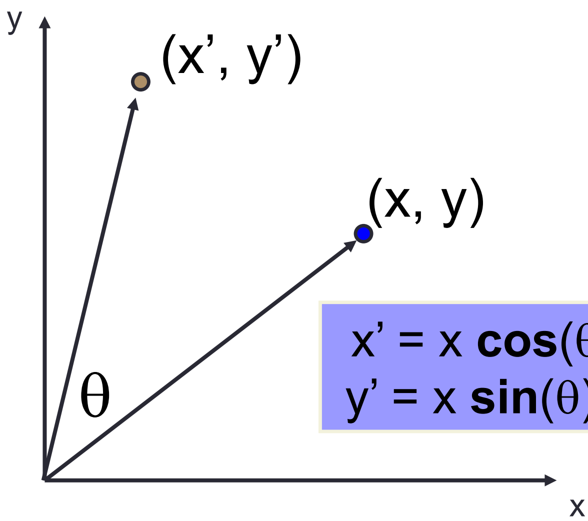
t_x

t_y

Preserves: Orientation

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

2-D Rotation



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

2-D Rotation

This is easy to capture in matrix form:

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

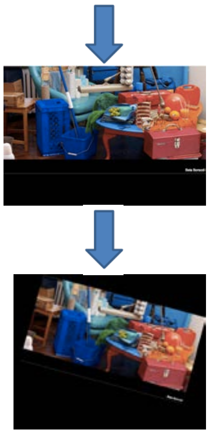
Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***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 –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

Rotation and Translation

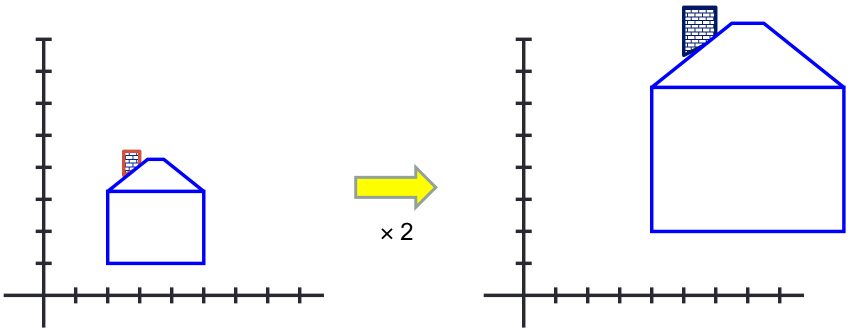


$$\begin{bmatrix} cos(\theta) & -sin(\theta) & t_x \\ sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

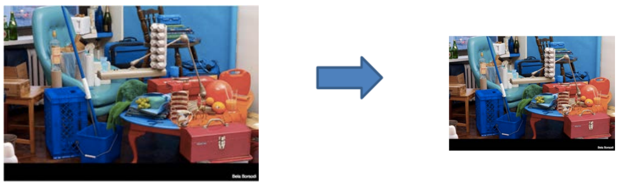
Translations and Rotations are not commutative

Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:

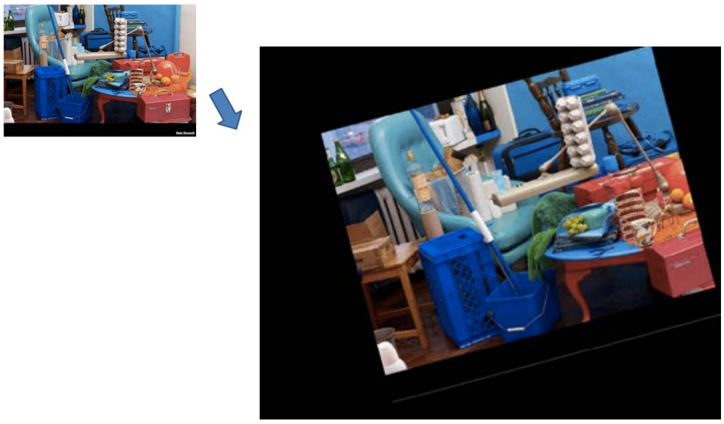


Scale



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

Similarity Transformations

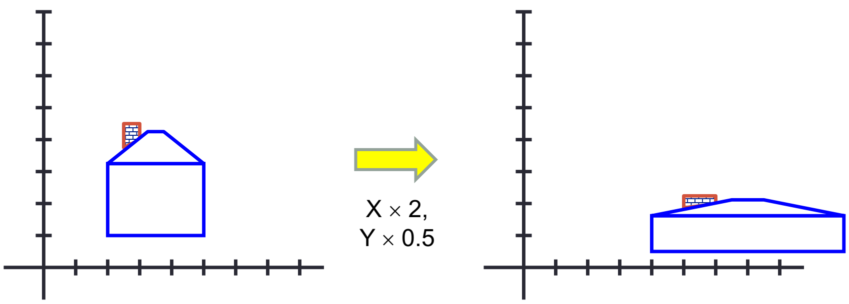


Similarity transform (4 DoF) = translation + rotation + scale

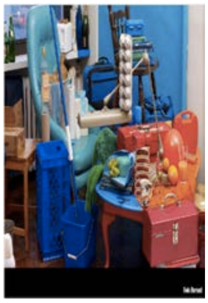
Preserves: Angles

Aspect Ratio

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



Aspect Ratio

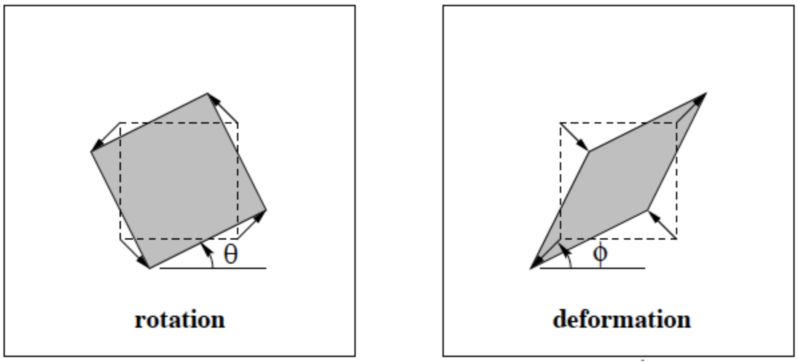


$$\begin{bmatrix} a & 0 & 0 \\ 0 & \frac{1}{a} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Shear

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

Planar Affine Deformation



$R(\theta)$ $\qquad$ $R(-\phi)DR(\phi)$

Basic 2D Transformations

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

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

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

Rotate

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

Translate

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

Affine

Affine is any combination of translation, scale, rotation, and shear

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

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

or

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

Properties of affine transformations:

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

What is missing?

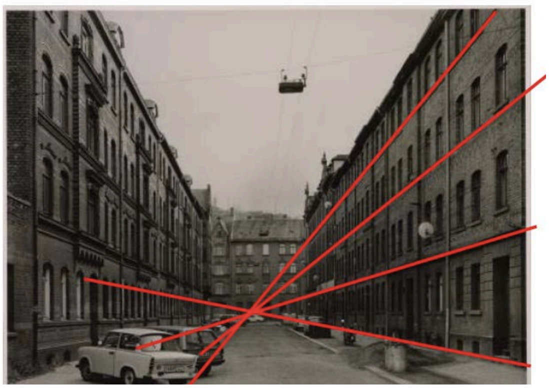
Canaletto

Are there any other planar transformations?

Vanishing Points and Lines

Photo from online Tate collection

Note on Estimating Vanishing Points



Use multiple lines for better accuracy

… but lines will not intersect at exactly the same point in practice

One solution: take mean of intersecting pairs

… bad idea!

Instead, minimize angular differences

General Affine

We already used these

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

Projective Transformations

a.k.a. Homographies

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$x' = u/w$$
$$y' = v/w$$

“keystone” distortions

Projective Transformations

Similarity Affine Projective

Projective Transformations

Projective transformations are combos of

- Affine transformations, and
- Projective warps

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

Properties of projective transformations:

- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

Homogeneous Coordinates

Converting to *homogeneous* coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogeneous Coordinates

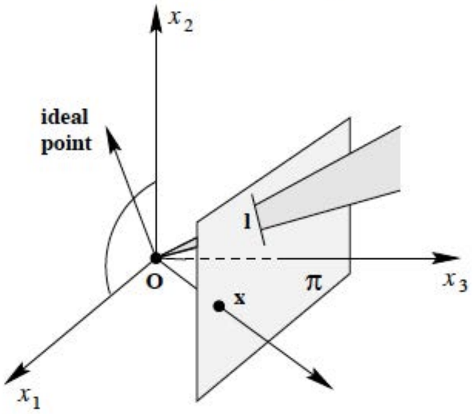
- Line equation: ax + by + c = 0 $\quad line_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}$
- Append 1 to pixel coordinate to get homogeneous coordinate $\quad p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$
- Line given by cross product of two points $\quad line_{ij} = p_i \times p_j$
- Intersection of two lines given by cross product of the lines $\quad q_{ij} = line_i \times line_j$

Example: Intersection of Parallel lines

$$x = 1 \qquad l = (-1, 0, 1)^T$$
$$x = 2 \qquad l' = (-1, 0, 2)^T$$

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ -1 & 0 & 1 \\ -1 & 0 & 2 \end{vmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

2D Projective Plane



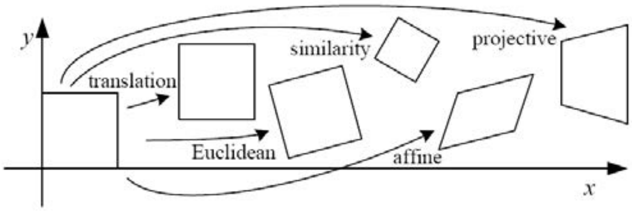
Projective Transformations

Projective transformations are combos of
- Affine transformations, and
- Projective warps

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

Properties of projective transformations:
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

2D image transformations (reference table)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

'Homography'

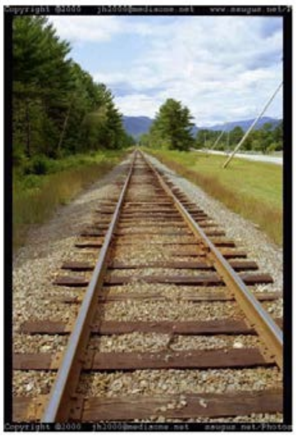
Szeliski 2.1

Bonus: Geometric transformations

Use different geometric transformations to deform
different images (or the same image)
and merge the deformed images
in a creative way.
Explain in text (math) what have you done.
Best *outcome* winner gets extra .5 bonus point.

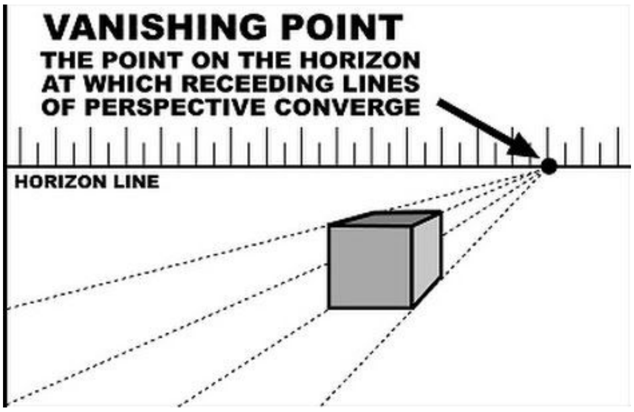
https://www.google.co.il/search?q=giant+image+small&client=firefox-
b&dcr=0&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwiPz5LDgJjYAhW
HF-wKHQtYCkcQsAQIJg&biw=1127&bih=739#imgrc=XIX7j4WkGBQ7uM:

Vanishing Points and Lines

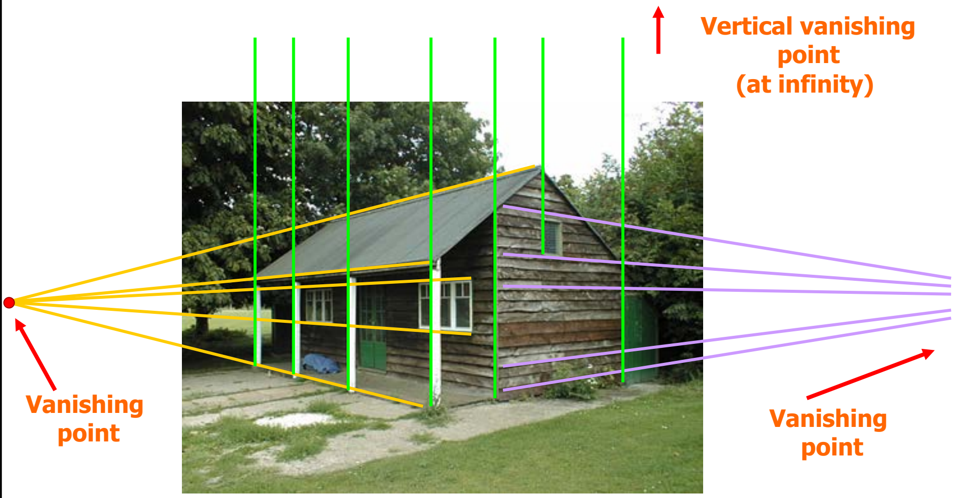


Parallel lines in the world intersect in the image at a "vanishing point"

Vanishing Points and Lines



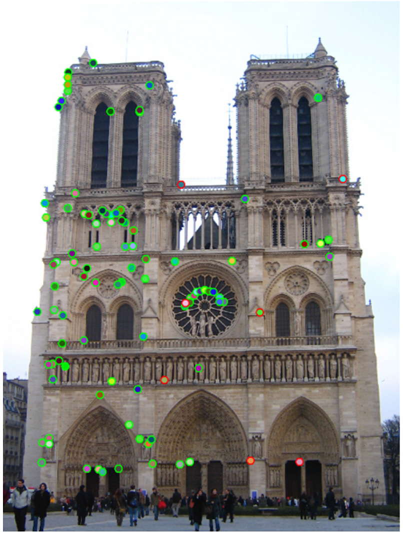
Example: vanishing points and lines

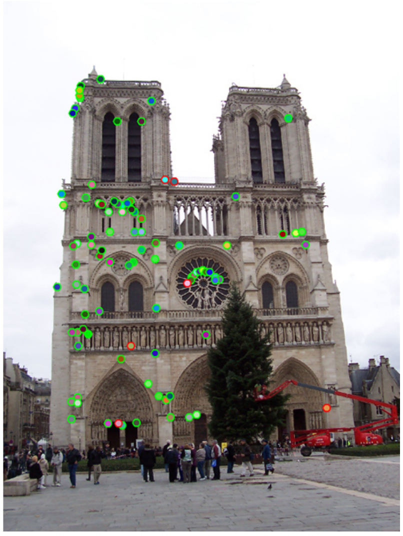


Slide from Efros, Photo from Criminisi

Vanishing Objects

Given matches, what is the transformation?





Fitting and Alignment

Fitting:

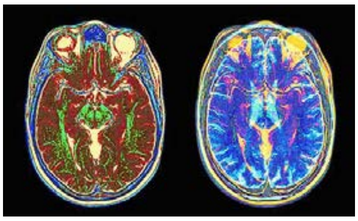
Find the parameters of a model that best fit the data.

Alignment:

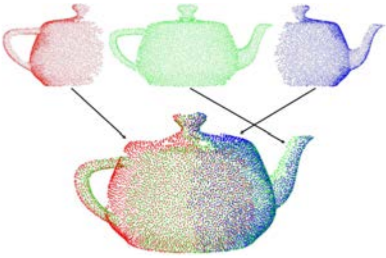
Find the parameters of the transformation that best aligns matched points.

Image Alignment: Panography

Other applications



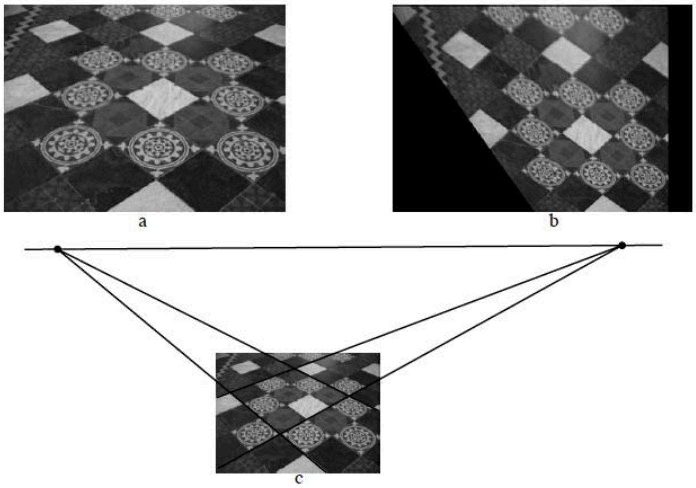
Medical imaging: match brain



Robotics: match point clouds

Kwok and Tang

Affine rectification with vanishing lines



Fitting and Alignment

- Challenges
  - Design a suitable **goodness of fit** measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method
    - Avoid local optima
    - Find best parameters quickly
  - Typically want to solve for a global transformation that accounts for **the most** true correspondences
    - Noise (typically 1-3 pixels)
    - Outliers (often 50%)
    - Many-to-one matches or multiple objects

Fitting and Alignment: Methods

- Global optimization / search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

Fitting and Alignment: Methods

- Global optimization / search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

2D Alignment with Least Squares

Given a set of matched feature points $\{(\mathbf{x}_i, \mathbf{x}_i')\}$

and a planar parametric transformation of the form:

$$\mathbf{x}' = f(\mathbf{x}; p)$$

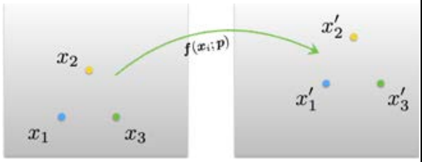
How can we estimate p ?

$$||\boldsymbol{x}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

$$E_{LS} = \sum_i ||f(\mathbf{x}_i; p) - \mathbf{x}_i'||^2$$

Predicted measured

Residual
(projection error)



2D Alignment with Least Squares

$$E_{LS} = \sum_i ||f(\mathbf{x}_i; p) - \mathbf{x}_i'||^2$$

Find parameters that minimize squared error

$$\hat{\boldsymbol{p}} = \arg\min_{\boldsymbol{p}} \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}_i'\|^2$$

2D Alignment with Least Squares

General form of linear least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\text{LLS}} = \sum_i |\boldsymbol{a}_i \boldsymbol{x} - \boldsymbol{b}_i|^2$$

$$= \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 \quad \text{(matrix form)}$$

This function is quadratic.

How do you find the root of a quadratic?

2D Alignment with Least Squares

Minimize the error:

Expand

$$E_{\text{LLS}} = \boldsymbol{x}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} - 2\boldsymbol{x}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Take derivative, set to zero

$$(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b} \quad \text{(normal equation)}$$

Solve for x

$$\boxed{\boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}}$$

2D Alignment with Least Squares

Translation

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

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

2D Alignment with Least Squares

Affine transformations

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

- How many unknowns?
- How many equations per match?
- How many matches do we need?

2D Alignment with Least Squares

For the Affine transformation $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x};\boldsymbol{p})$

$$\boldsymbol{x}' = \mathbf{M}\boldsymbol{x} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Vectorize transformation parameters

Affine

$$\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}$$

Notation in general form $\quad \boldsymbol{b} \qquad \mathbf{A} \qquad \boldsymbol{x}$

2D Alignment with Least Squares

$$E_{LS} = \sum_i ||f(\mathbf{x}_i; p) - \mathbf{x}'_i||^2$$

There is a linear relationship between the transformation parameters (Translation, Similarity, Affine) and the differences between the coordinates: $\Delta\mathbf{x} = \mathbf{x}' - \mathbf{x} = J(\mathbf{x})p$

where $J = \dfrac{\partial f}{\partial p}$ is the Jacobian of the transformation f with respect to the motion parameters p

2D Alignment with Least Squares

$$
\begin{aligned}
E_{\text{LLS}} &= \sum_i \|\boldsymbol{J}(\boldsymbol{x}_i)\boldsymbol{p} - \Delta\boldsymbol{x}_i\|^2 \\
&= \boldsymbol{p}^T \left[\sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i)\right] \boldsymbol{p} - 2\boldsymbol{p}^T \left[\sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta\boldsymbol{x}_i\right] + \sum_i \|\Delta\boldsymbol{x}_i\|^2 \\
&= \boldsymbol{p}^T \boldsymbol{A}\boldsymbol{p} - 2\boldsymbol{p}^T\boldsymbol{b} + c.
\end{aligned}
$$

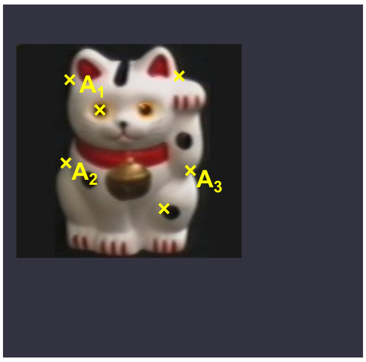
The minimum can be found by solving the symmetric positive definite (SPD) system of normal equations: $\mathbf{A}p = b$

where $\boldsymbol{A} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i)$ Hessian $\boldsymbol{b} = \sum_i \boldsymbol{J}^T(\boldsymbol{x}_i)\Delta\boldsymbol{x}_i$

2D Coordinate Transformations and Jacobians

Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$

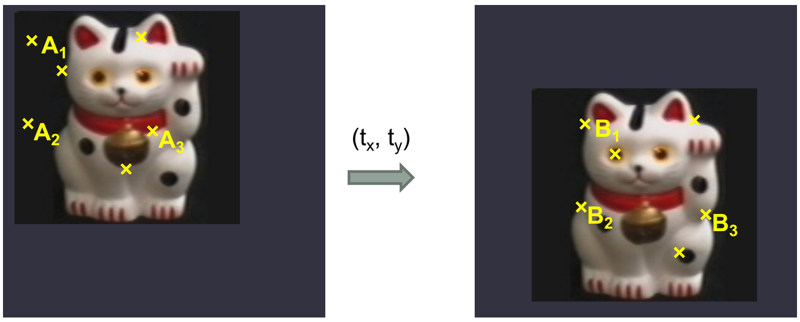
Example: solving for translation



Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



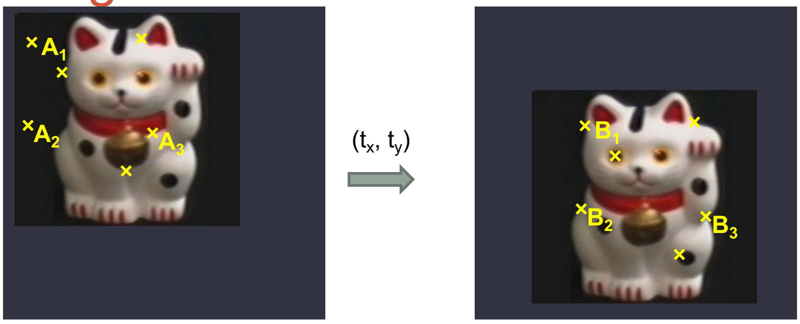
Least squares solution

1. Write down objective function
2. Derived solution
   a) Compute derivative
   b) Compute solution
3. Computational solution
   a) Write in form Ax=p
   b) Solve using closed-form solution

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

Solving for translation



for the case of translation –

the average translation

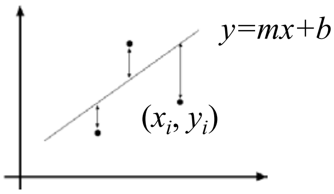
between corresponding points or,

equivalently, the translation of the point centroids

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$

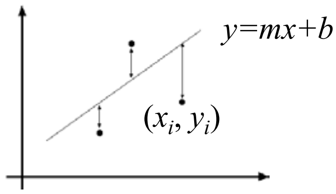


Modified from S. Lazebnik

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{Ap})^T \mathbf{y} + (\mathbf{Ap})^T (\mathbf{Ap})$$

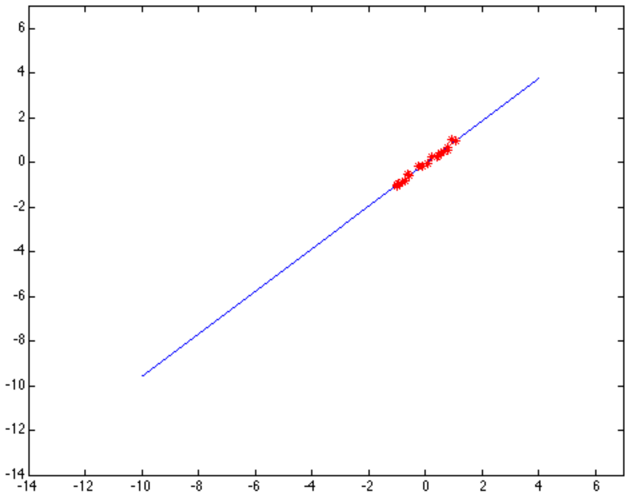
$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

Matlab: `p = A \ y;`

$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{y}$$ (Closed form solution)

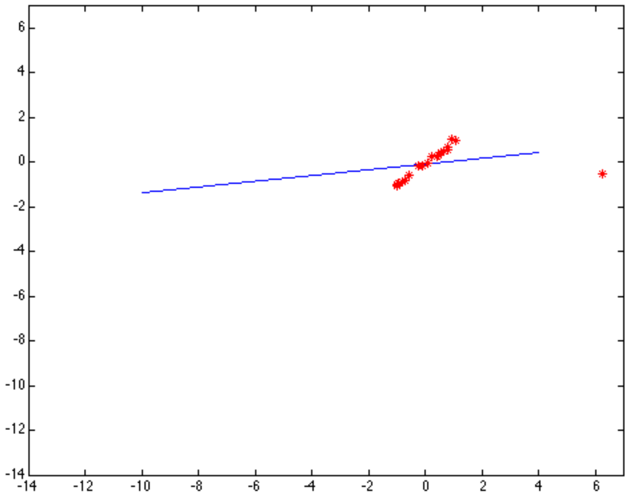
Modified from S. Lazebnik

Simple example: Fitting a line



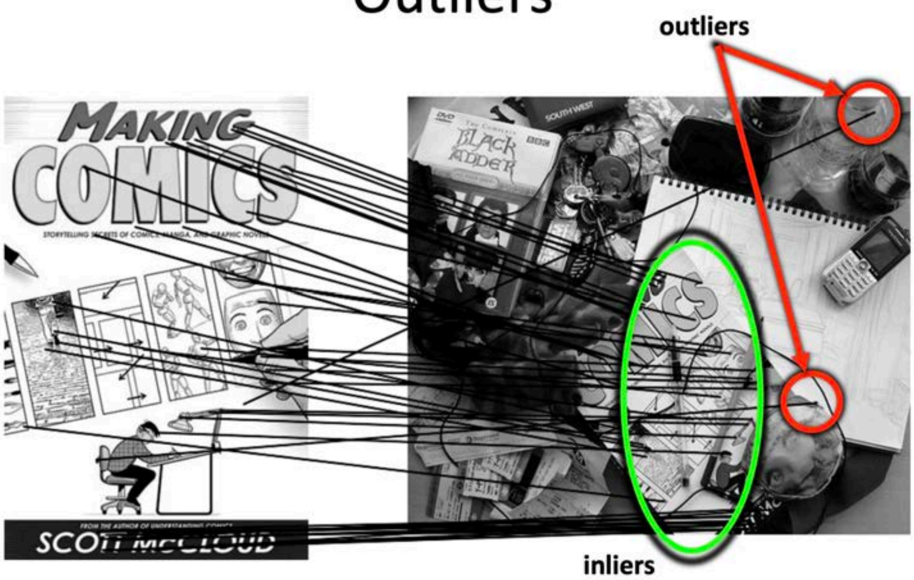
Least squares: Robustness to noise

- Least



Problem: squared error heavily penalizes outliers

Outliers



Least squares (global) optimization

Good

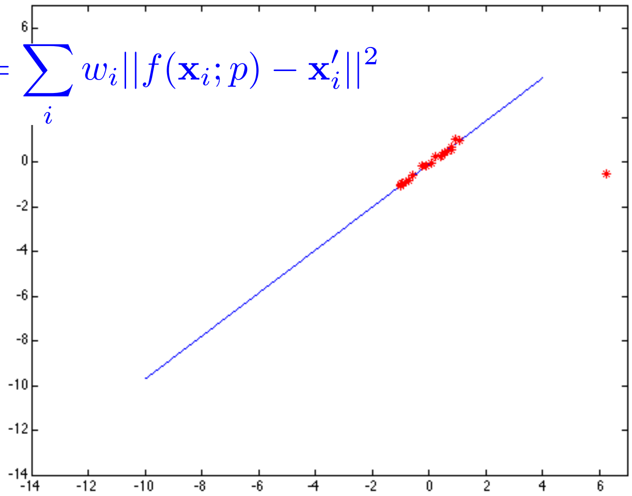
- Clearly specified objective
- Optimization is easy

Bad

- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

Robust (Weighted) Least Square

$$E_{WLS} = \sum_i w_i ||f(\mathbf{x}_i; p) - \mathbf{x}'_i||^2$$



The effect of the outlier is minimized

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Scale of robust function should be chosen adaptively based on median residual
- Least squares solution can be used for initialization

Other ways to search for parameters for when no closed form solution exists

Line search

1. For each parameter, step through values and choose value that gives best fit
2. Repeat (1) until no parameter changes

Grid search

1. Propose several sets of parameters, evenly sampled in the joint set
2. Choose best (or top few) and sample joint parameters around the current best; repeat

Gradient descent

1. Provide initial position (e.g., random)
2. Locally search for better parameters by following gradient

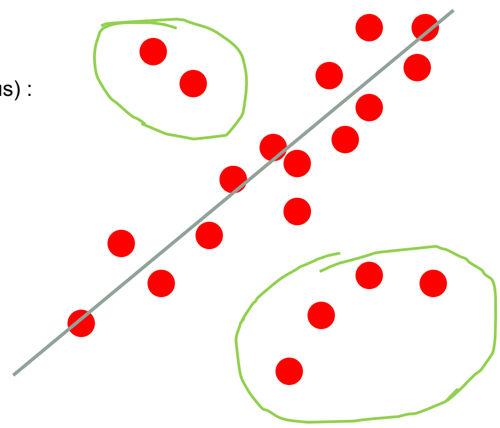
Hypothesize and test

1. Propose parameters
   - Try all possible
   - Each point votes for all consistent parameters
   - Repeatedly sample enough points to solve for parameters
2. Score the given parameters
   - Number of consistent points, possibly weighted by distance
3. Choose from among the set of parameters
   - Global or local maximum of scores
4. Possibly refine parameters using inliers

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.

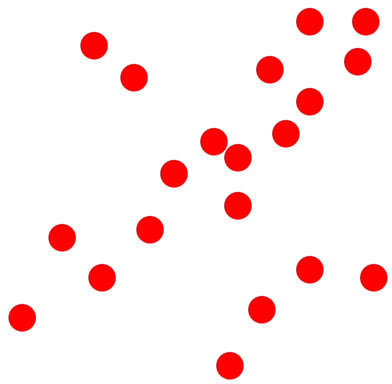


This data is noisy, but we expect a good fit to a known model.

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



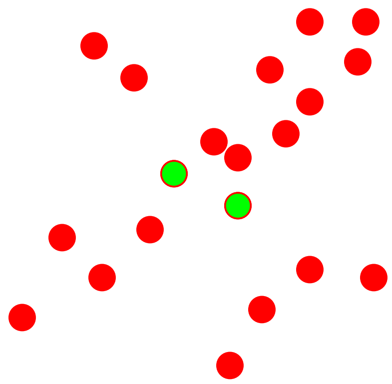
Algorithm:

1. **Sample** (randomly) the number of points *s* required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

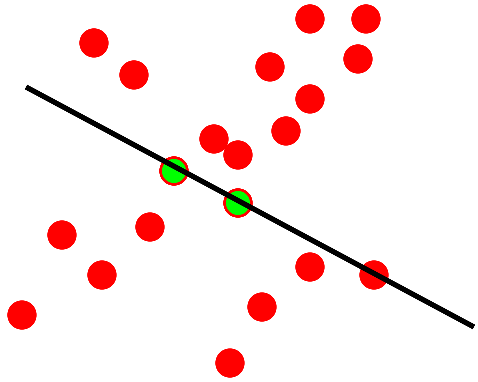
1. **Sample** (randomly) the number of points required to fit the model (s=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



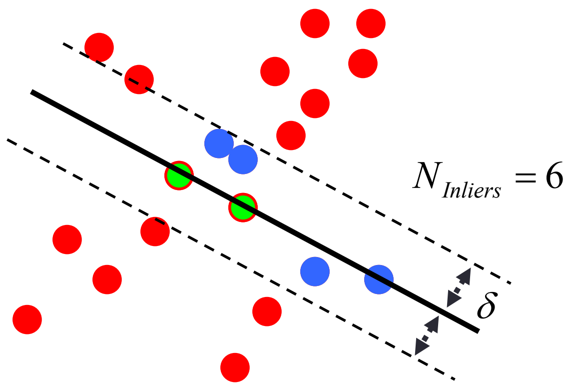
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (s=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example

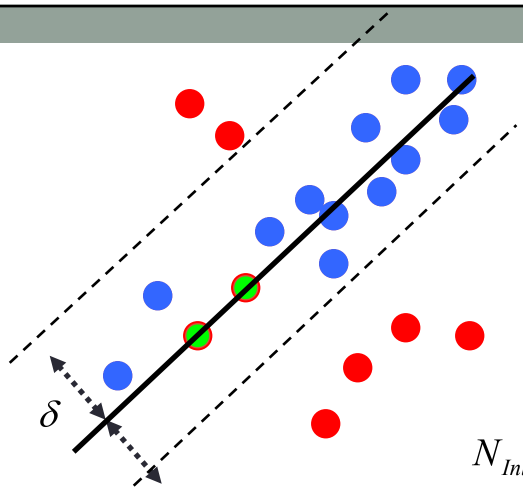


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (s=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

$$N_{Inliers} = 14$$

1. **Sample** (randomly) the number of points required to fit the model (*s*=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

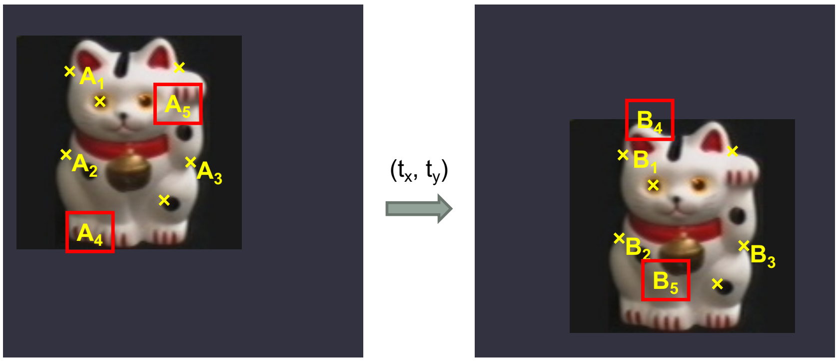
Repeat 1-3 until the best model is found with high confidence

Example: solving for translation

Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



Problem: outliers

RANSAC solution

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

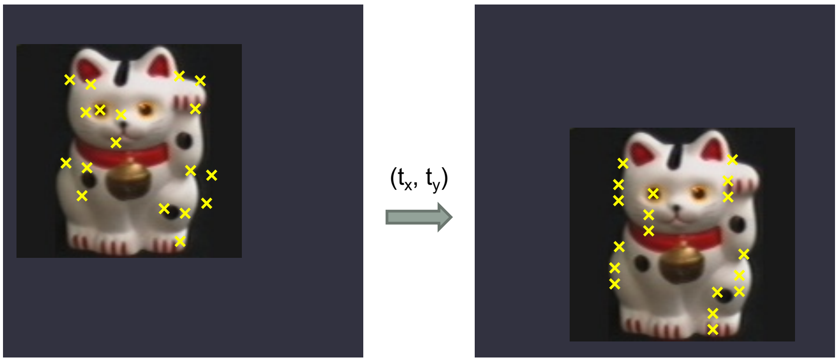
- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

What if we want to align… but we have no matched pairs?

- Hough transform and RANSAC not applicable



Problem: no initial guesses for correspondence

Iterative Closest Points (ICP) Algorithm

Goal:

Estimate transform between two dense point sets S_1 and S_2

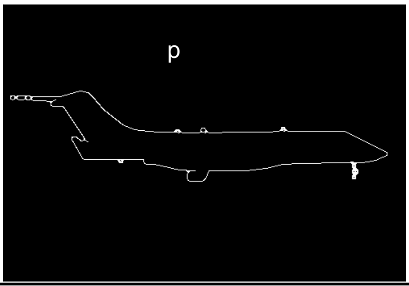
1. **Initialize** transformation
   - Compute difference in mean positions, subtract
   - Compute difference in scales, normalize
2. **Assign** each point in S_1 to its nearest neighbor in S_2
3. **Estimate** transformation parameters T
   - Least squares or robust least squares, e.g., rigid transform
4. **Transform** the points in S_1 using estimated parameters T
5. **Repeat** steps 2-4 until change is very small (convergence)

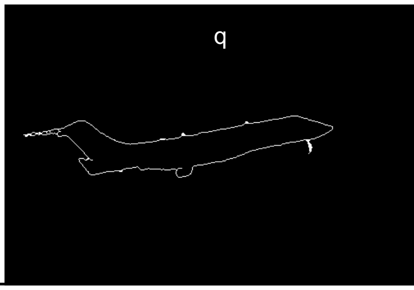
ICP demonstration

Bouaziz et al.

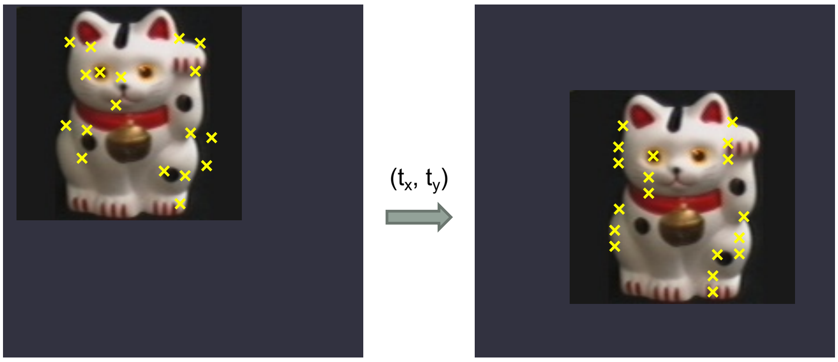
Example: aligning boundaries

1. Extract edge pixels $p_1..p_n$ and $q_1..q_m$
2. Compute initial transformation (e.g., compute translation and scaling by center of mass, variance within each image)
3. Get nearest neighbors: for each point p_i find corresponding $\text{match(i)} = \underset{j}{\text{argmin}}\, dist(pi, qj)$
4. Compute transformation ***T*** based on matches
5. Transform points ***p*** according to ***T***
6. Repeat 3-5 until convergence





Example: solving for translation



Problem: no initial guesses for correspondence

ICP solution

1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Algorithm Summaries

- Least Squares Fit
  - Closed form solution
  - Robust to noise
  - Not robust to outliers
- Robust Least Squares
  - Improves robustness to outliers
  - Requires iterative optimization
- RANSAC
  - Robust to noise and outliers
  - Works with a moderate number of parameters (e.g, 1-8)
- Iterative Closest Point (ICP)
  - For local alignment only: does not require initial correspondences
  - Sensitive to initialization
- Hough transform
  - Robust to noise and outliers
  - Can fit multiple models
  - Only works for a few parameters (1-4 typically)