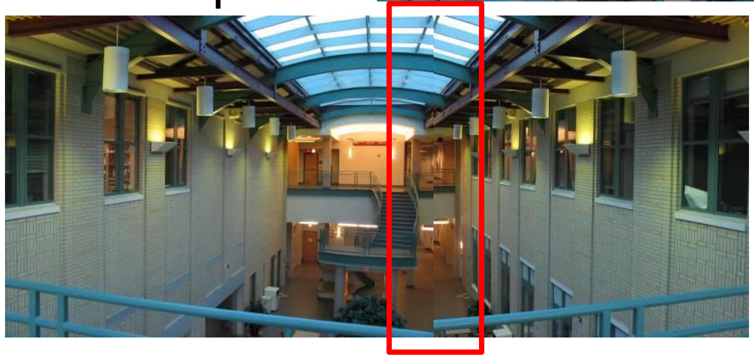
8. Homography

Mosaics: Stitching Images Together

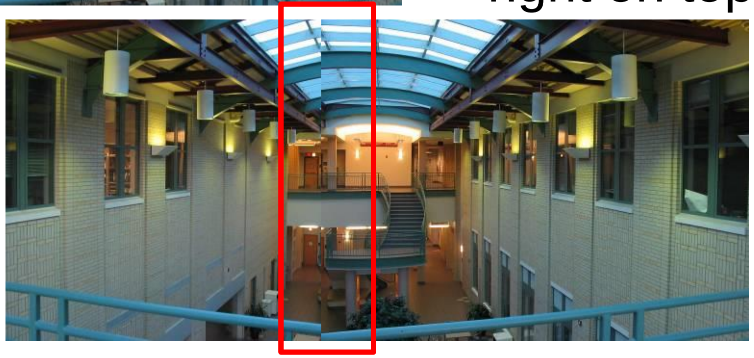
virtual wide-angle camera

Image Alignment: a quick experiments

left on top



right on top

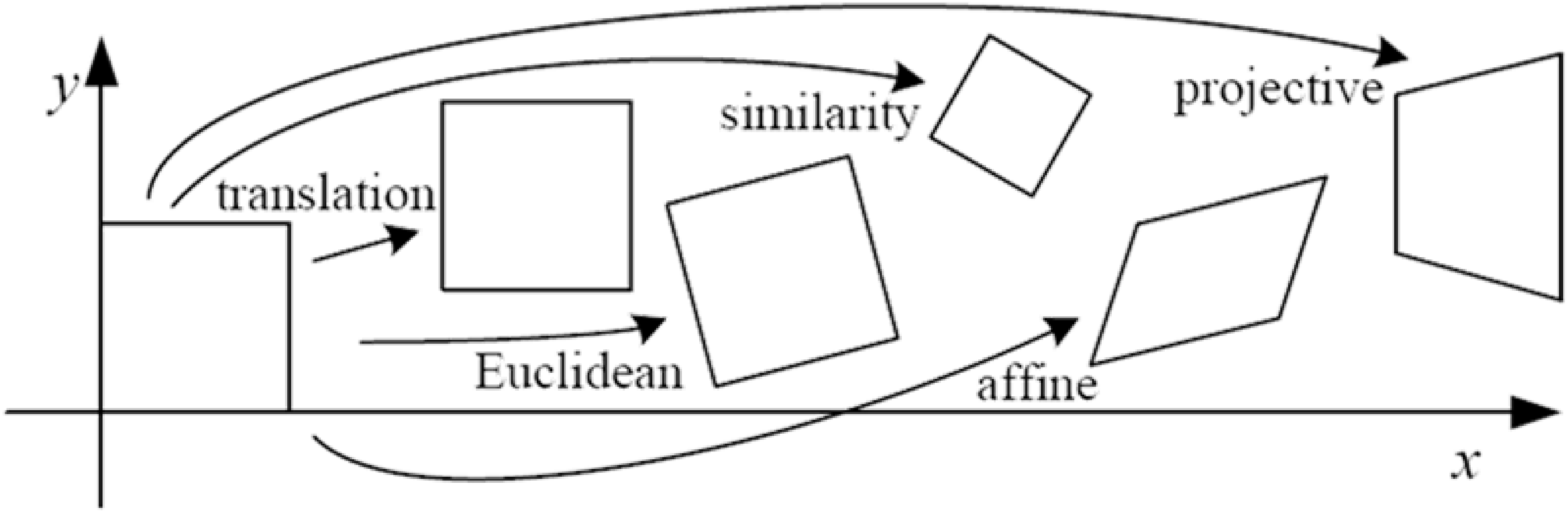


Translations are not enough to align the images

Motion Models

- What is an appropriate transformation:
  - translation?
  - rotation?
  - scale?
  - affine?
  - perspective?

Planar Linear Motion Models



Common Linear Transformations

- Uniform scaling by *s*:

(0,0) 

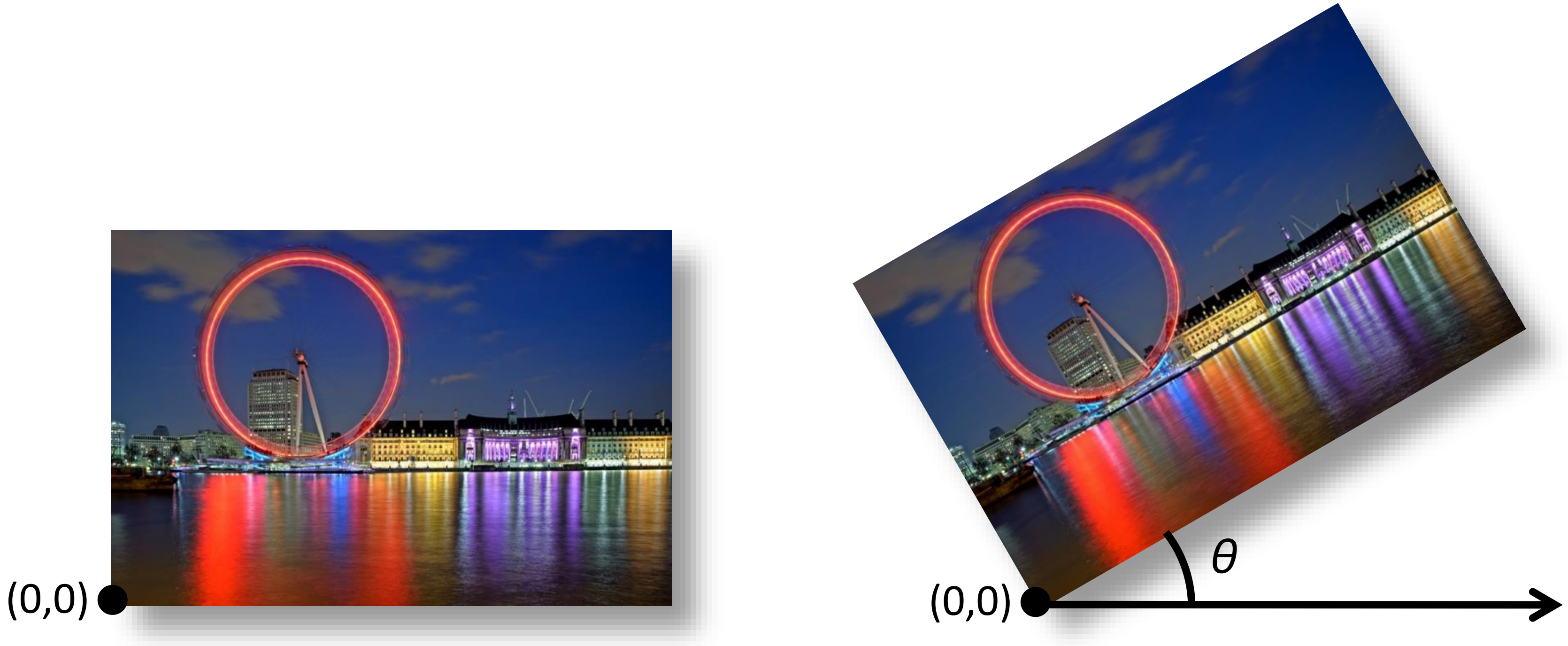
(0,0) 

$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?

Common Linear Transformations

- Rotation by angle θ (about the origin)



$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

What is the inverse?

For rotations:

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

2x2 Matrices

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

2D mirror about Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D mirror across line y = x?

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

2x2 Matrices

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

2D Translation?

$$
\begin{aligned}
x' &= x + t_x \\
y' &= y + t_y
\end{aligned}
$$

NO!

Homogeneous Coordinates

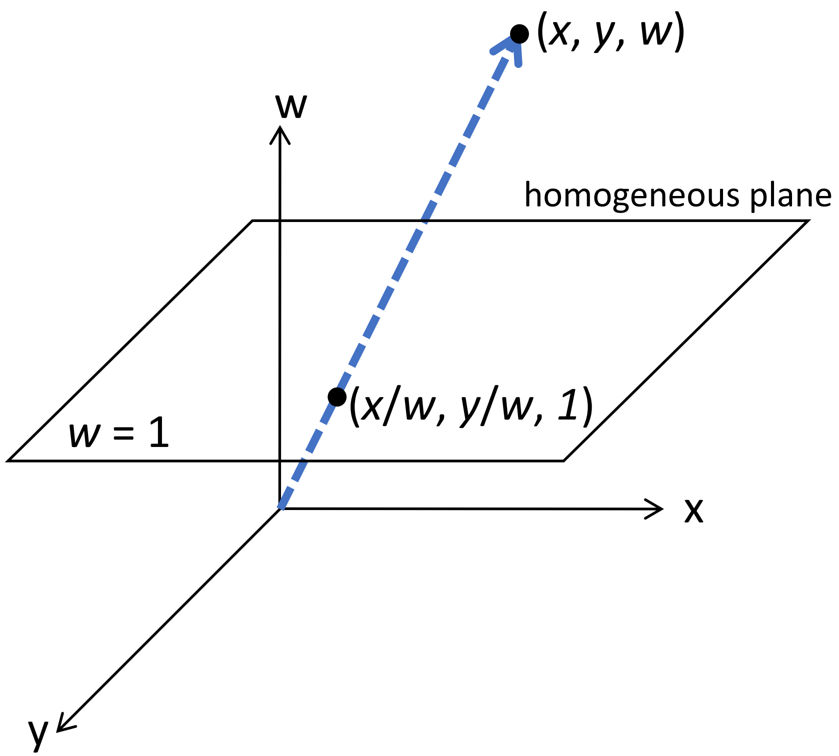
Trick: add one more coordinate:

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

homogeneous image coordinates

Converting *from* homogeneous coordinates

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



Translation

- Solution: homogeneous coordinates to the rescue

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

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

Affine Transformations

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

any transformation with last row [0 0 1] we call an *affine* transformation

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

Basic Affine Transformations

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

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_{\boldsymbol{x}} & 0 & 0 \\ 0 & \boldsymbol{s}_{\boldsymbol{y}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

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

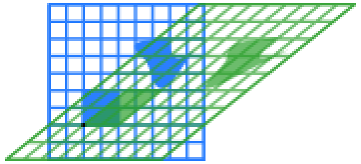
2D *in-plane* rotation

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

Shear (x direction)

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

Shear (y direction)



Affine Transformations

- Affine transformations are combinations of …
  - Rotation, Scaling, Shearing, and
  - Translations

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

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

Is this an affine transformation?

Where do we go from here?

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

what happens when we mess with this row?

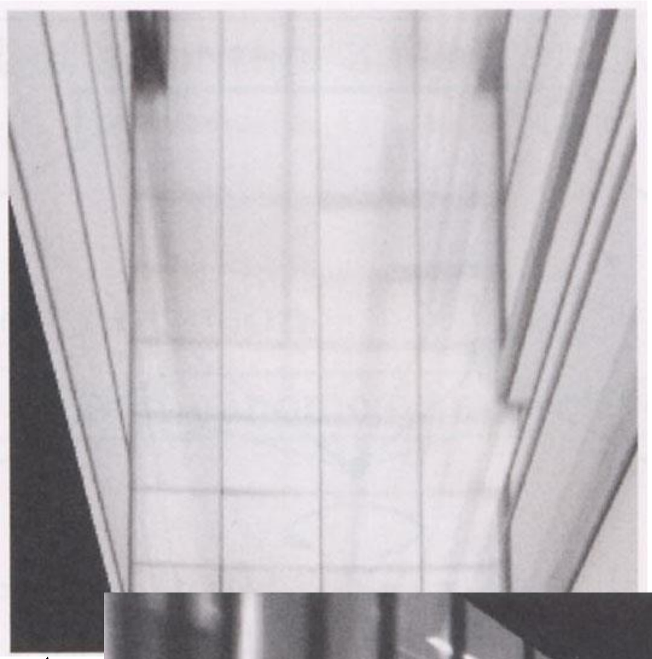
affine transformation

Projective Transformations aka Homographies aka Planar Perspective Maps

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)

Image warping with homographies



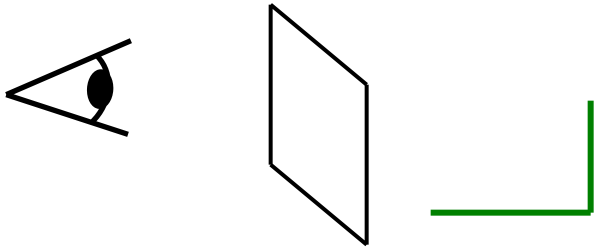
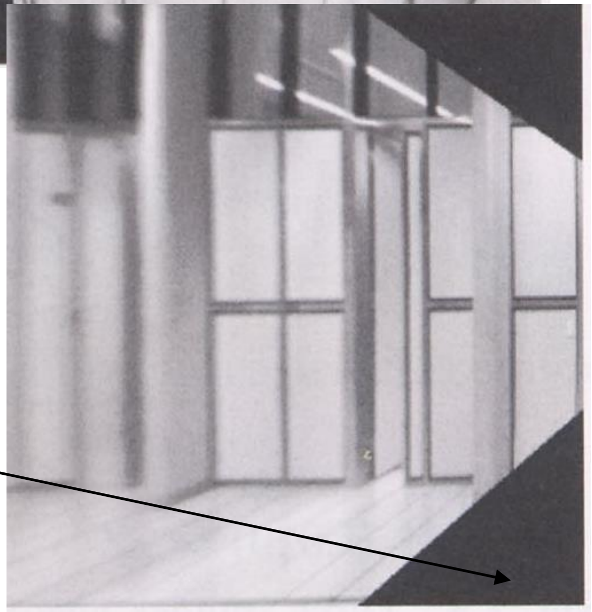


image plane in front



black area
where no pixel
maps to

Homographies

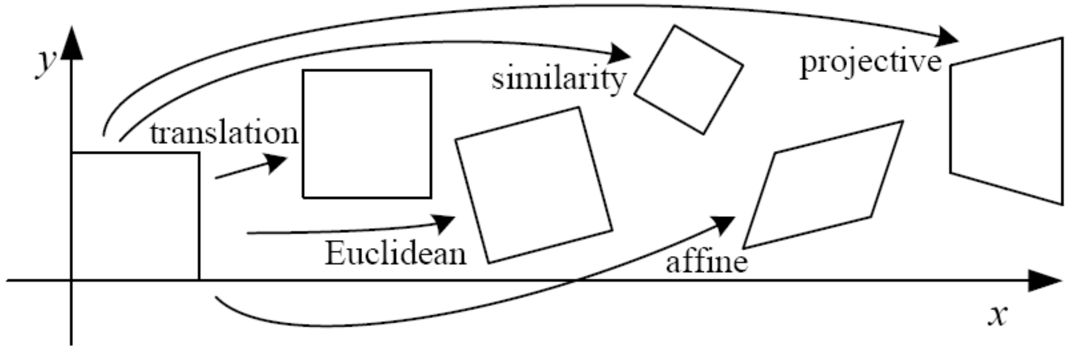
Projective Transformations

- Projective transformations …
  - 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:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

2D Linear Transformations



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	

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Examples

fronto-parallel view

affine view

perspective view

distant camera, zoomed in

Examples

original view

synthetic rotations

Examples

two
original
images

rectified and stitched

Questions?

Homography Warping

- The matrix representation

$$w\underbrace{\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}}_{\mathrm{x}'} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathrm{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathrm{x}}$$

w means x and x' are equal up to a scale;

w does not affect the results (per definition of homogenous coordinates)

- To apply a homography H
  - Compute **x’** = **Hx** (regular matrix multiply)
  - Convert **x’** from homogeneous to image coordinates
    - divide by w (third) coordinate

Solving for Homographies -- the basic idea

$$\boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x} \quad \Rightarrow \quad w\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Find the homography H from feature correspondences
  - E.g. pairs of matched SIFT/Harris corners, x and x'
- How?
  - Reshape $\boldsymbol{H}$ to a vector $\boldsymbol{h} = [a, b, c, d, e, f, g, h]^T$
  - $\boldsymbol{x}$ and $\boldsymbol{x}'$ are both known parameters
  - Each pair of feature points provide linear equations of $\boldsymbol{h}$
  - More pairs provide linear equations: $\boldsymbol{Ah} = \boldsymbol{b}$
  - How many correspondences do we need?

Direct Linear Transformation (DLT)

$$\mathbf{x}_i = (x_i, y_i, 1)^{\mathsf{T}} \qquad \mathbf{x}'_i = (x'_i, y'_i, 1)^{\mathsf{T}}$$

$$\mathbf{x}'_i \propto \mathbf{H}\mathbf{x}_i \quad \Rightarrow \quad \mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = 0$$

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}} \\ \mathbf{h}^{2\mathsf{T}} \\ \mathbf{h}^{3\mathsf{T}} \end{pmatrix} \qquad \mathbf{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i - \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i - x'_i\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \\ x'_i\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i - y'_i\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

Direct Linear Transformation (DLT)

$$\mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \begin{bmatrix} 0^\top & -\mathbf{x}_i^\top & y'_i\mathbf{x}_i^\top \\ \mathbf{x}_i^\top & 0^\top & -x'_i\mathbf{x}_i^\top \\ -y'_i\mathbf{x}_i^\top & x'_i\mathbf{x}_i^\top & 0^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

$$\mathbf{A}_i\mathbf{h} = 0$$

Direct Linear Transformation (DLT)

- Equations are linear in $\mathbf{h}$, $\mathbf{A}_i\mathbf{h} = 0$
- Only 2 out of 3 are linearly independent (indeed, 2 eq/pt)

$$\begin{bmatrix} 0^\top & -\mathbf{x}_i^\top & y_i'\mathbf{x}_i^\top \\ \mathbf{x}_i^\top & 0^\top & -x_i'\mathbf{x}_i^\top \\ -y_i'\mathbf{x}_i^\top & x_i'\mathbf{x}_i^\top & 0^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

(only drop third row if $w_i' \neq 0$)

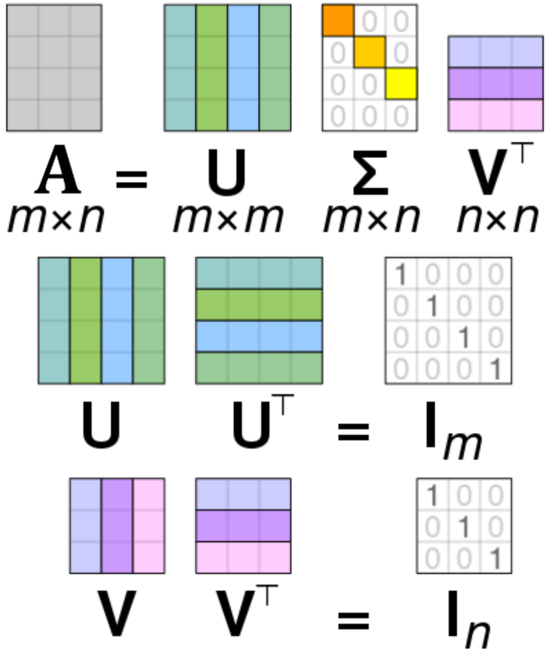
Direct Linear Transformation (DLT)

- Solving for H from 4 points (each point provide two independent equations of h)

$$\mathbf{Ah} = 0$$

- size A is 8x9 or 12x9, but rank 8
- Trivial solution is **h**=0
- 1-D null-space yields solution of interest
  - pick for example the one with $\|\mathbf{h}\| = 1$
- No exact solution because of inexact measurement, i.e. “noise”
  - Minimize $\|\mathbf{Ah}\|$ with constraint $\|\mathbf{h}\| = 1$
    (solution is to take the SVD of **A**, and get the last column of the matrix **V**)

Singular Value Decomposition



ortho-normal diagonal ortho-normal

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

unit norm constraint

m x n m x m m x n n x n

$$= \sum_{i=1}^{9} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$$

m x 1 1 x n

Each column of V represents a solution for $\mathbf{A}\boldsymbol{h} = \mathbf{0}$

where the singular value represents the reprojection error

DLT Algorithm

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{x_i \leftrightarrow x_i'\}$, determine the 2D homography matrix H such that $x_i' = Hx_i$

Algorithm

(i) For each correspondence $x_i \leftrightarrow x_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble n 2x9 matrices A_i into a single $2n$x9 matrix A

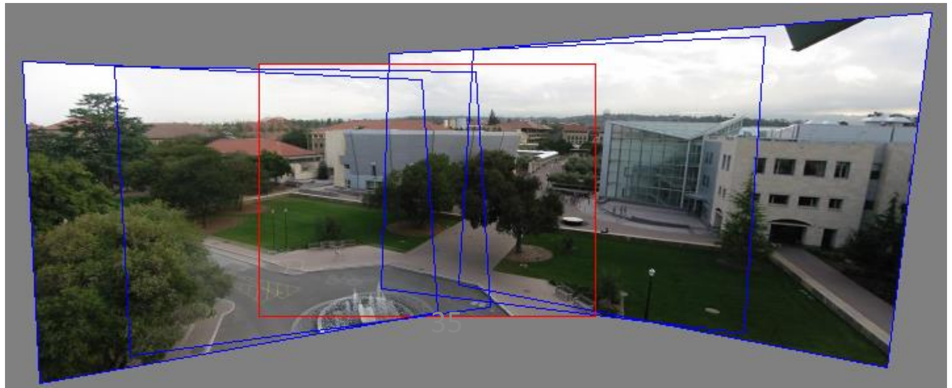
(iii) Obtain SVD of A. Solution for h is last column of V

(iv) Determine H from h

Questions?

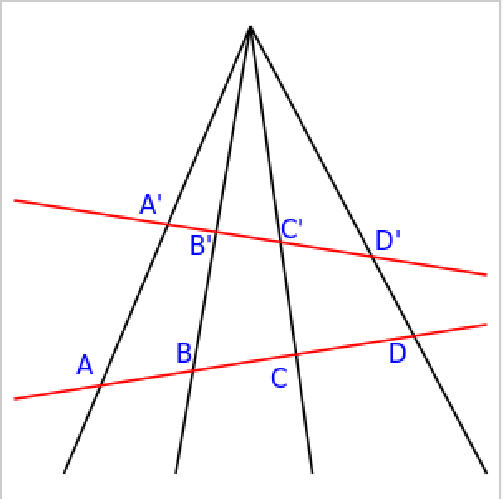
Overall Pipeline

- 3D rotation ➔ homography image transformation
  - Using homogeneous coordinates to represent pixel position
- Use feature correspondence
- Solve the homography model
- Warp all images to a reference one
  1. Pick one image (red)
  2. Warp the other images towards it by solving the homography transformation (usually, one by one)
- Use your favorite blending



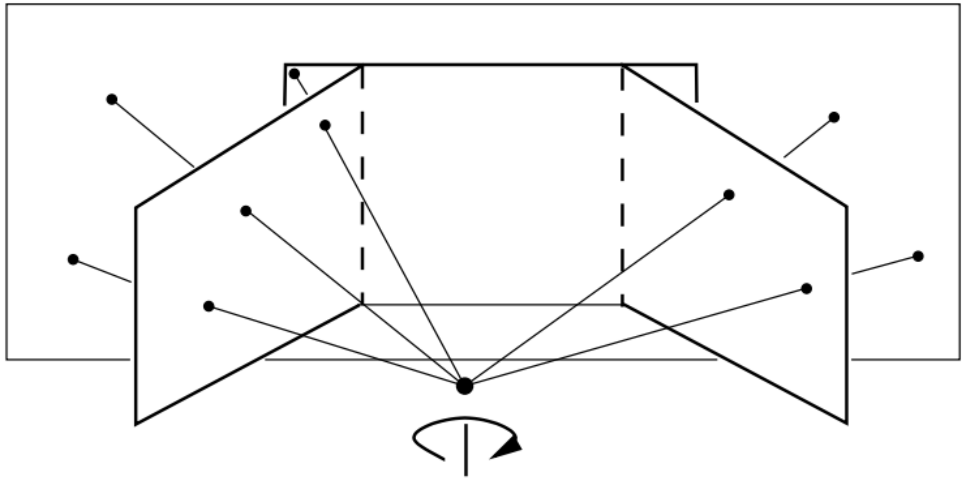
Why the Homography Model?

- Projective Transformation
  - Historically developed from the study of central projection
  - Maps lines to lines, points to points, but does not preserve parallelism
- The two images of a rotating camera



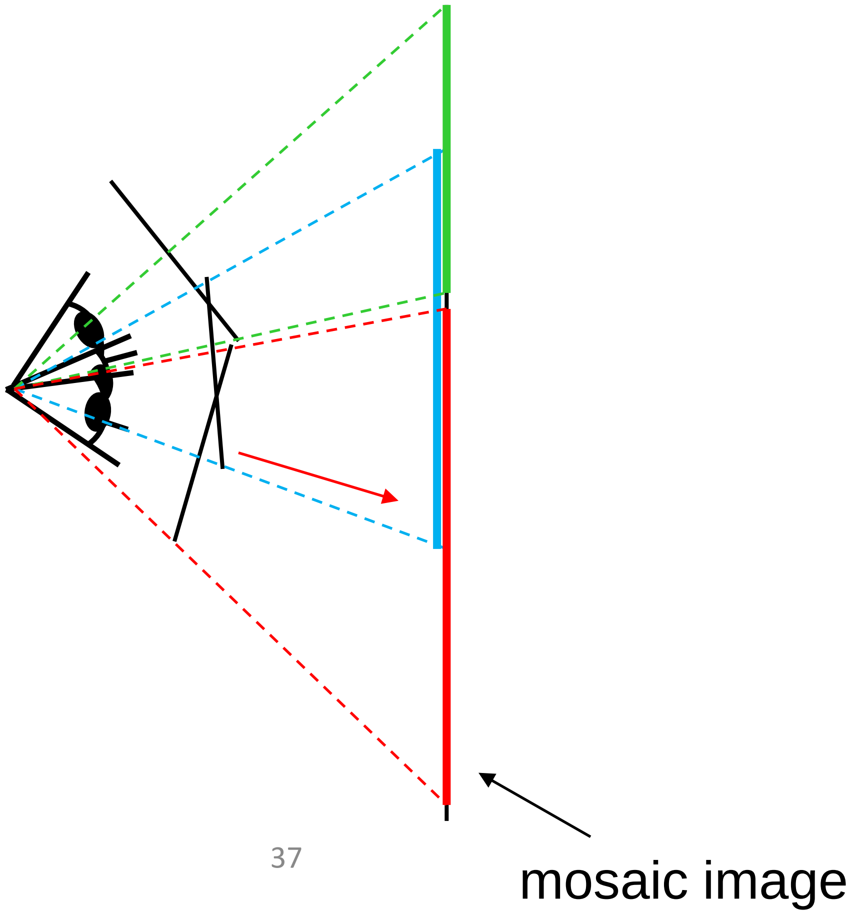
Points A, B, C, D and A', B', C', D' are related by a perspectivity, which is a projective transformation.

From Wikipedia

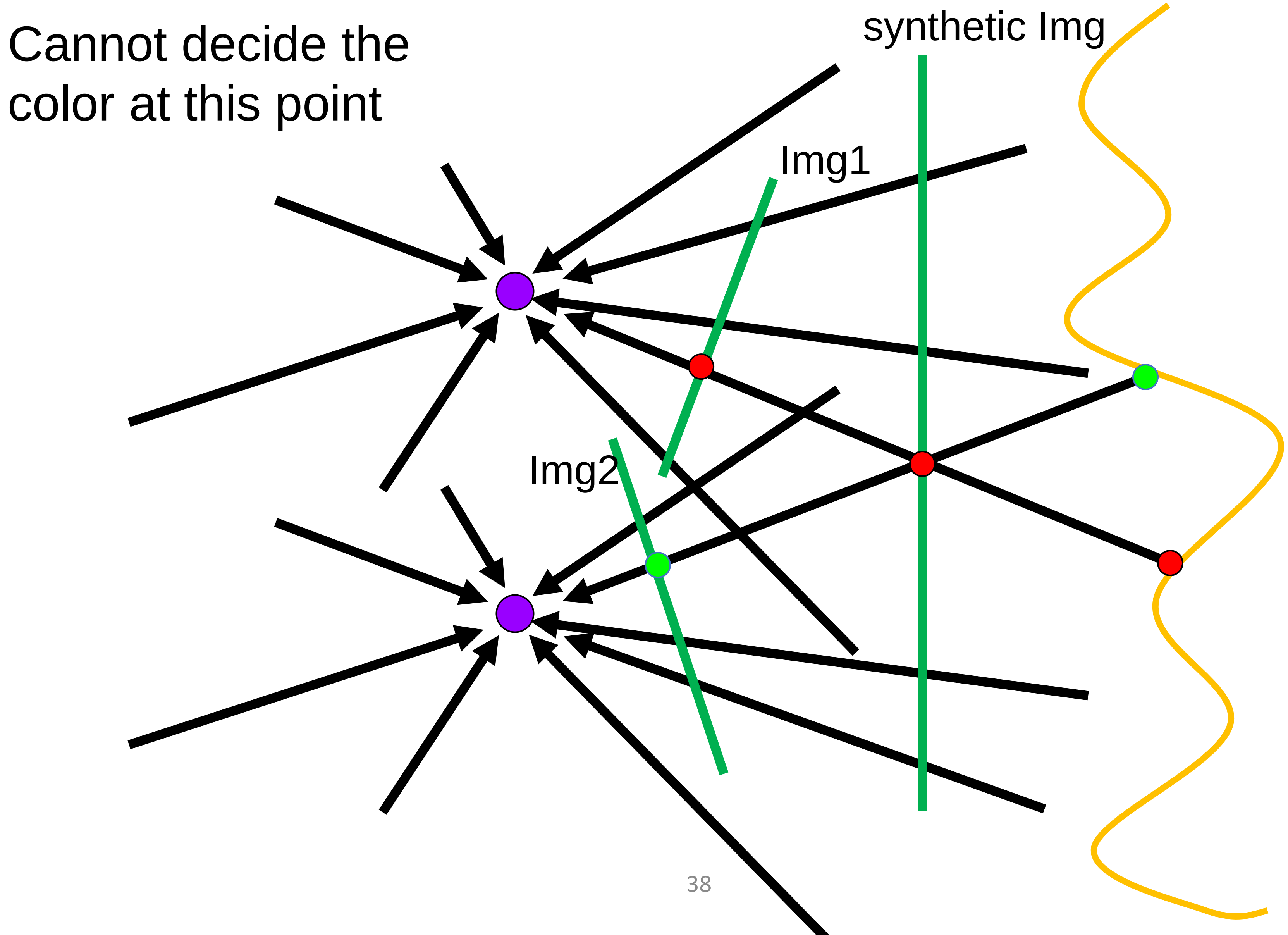


An Intuitive Explanation

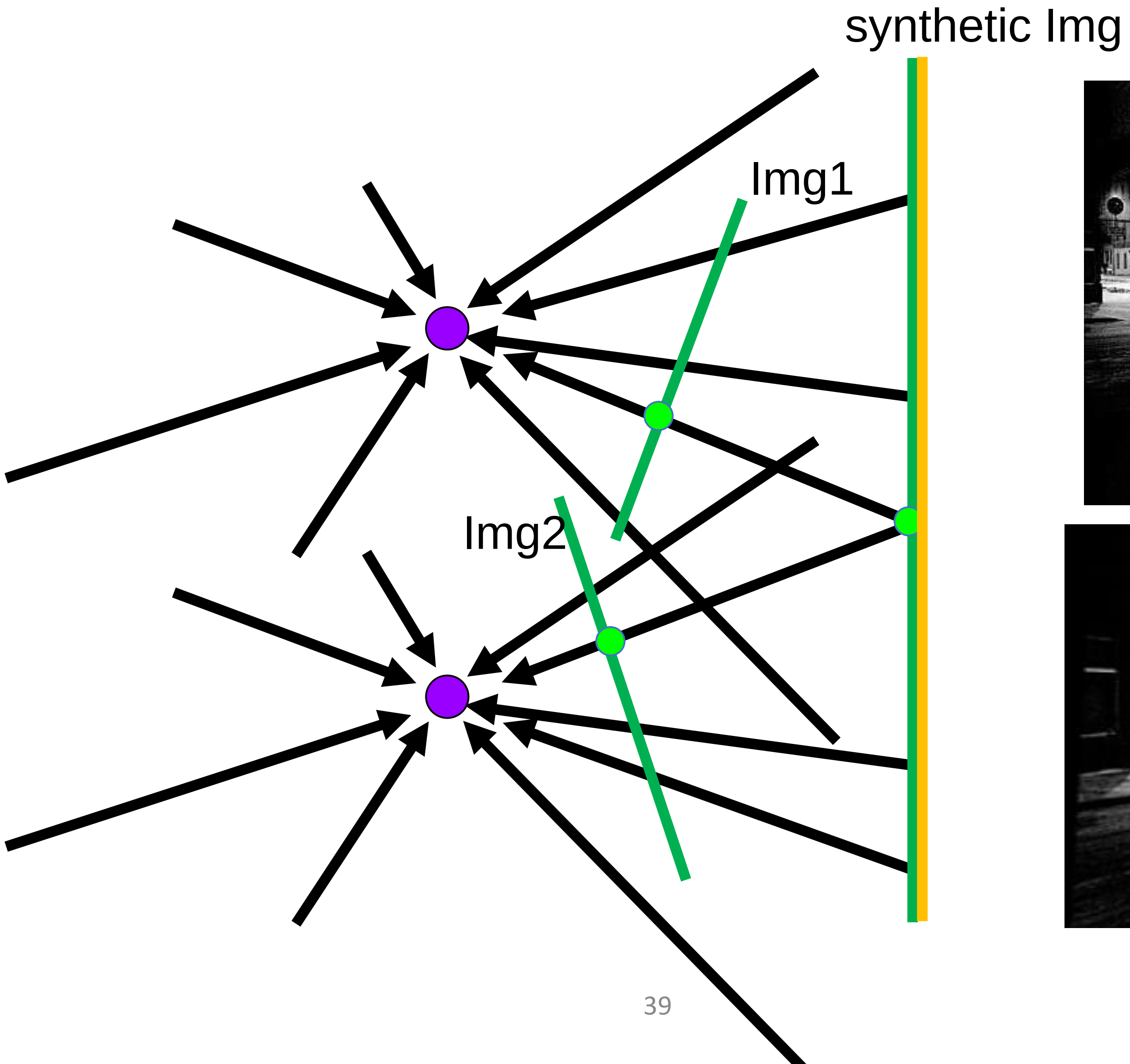
- The mosaic has a natural interpretation in 3D
  - The images are reprojected onto a common plane
  - The mosaic is the image of that plane



Mosaic for Changing Camera Center?



Works for Planar Scenes



Questions?

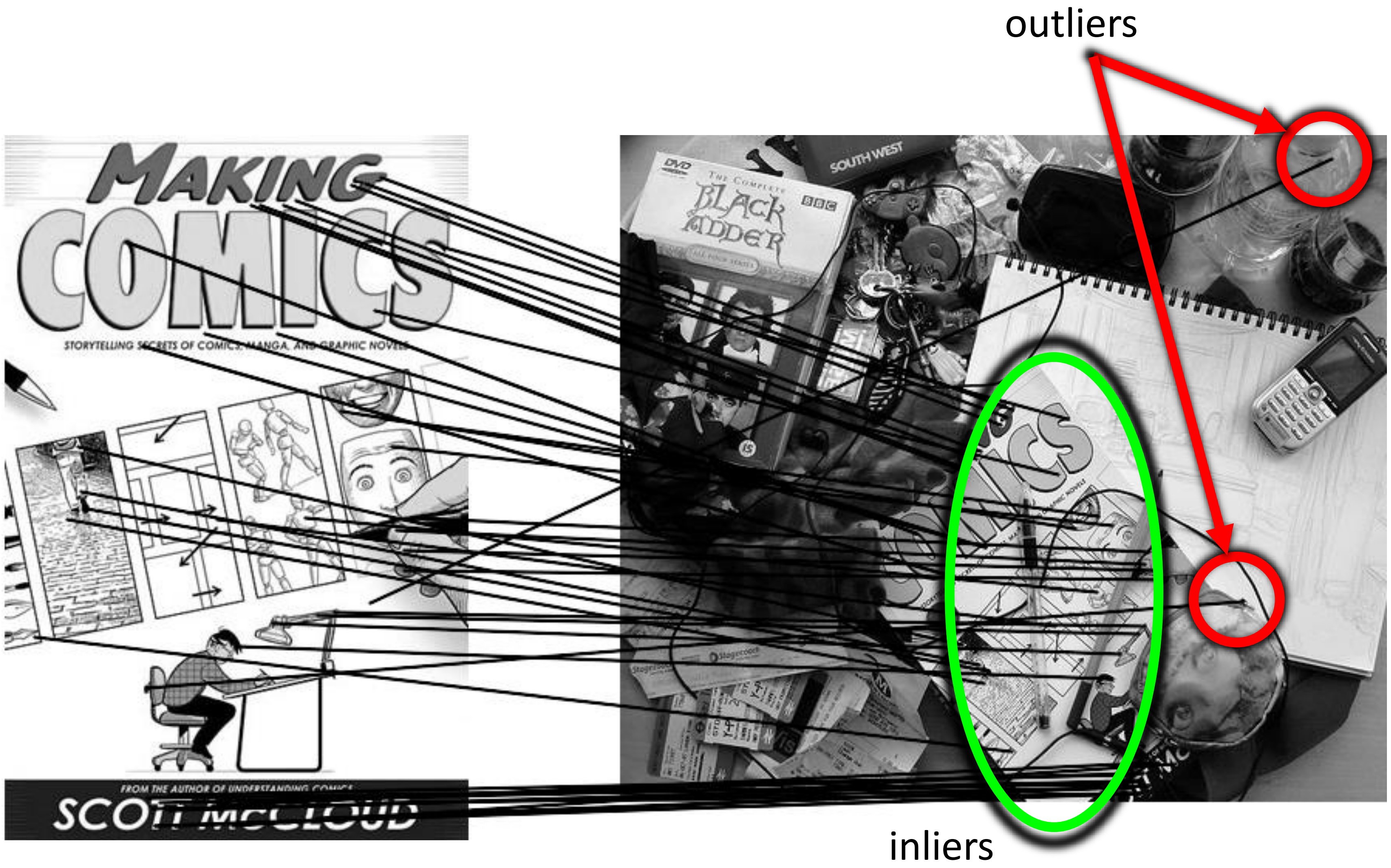
Mosaic Stitching Algorithm

Given images A and B

1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

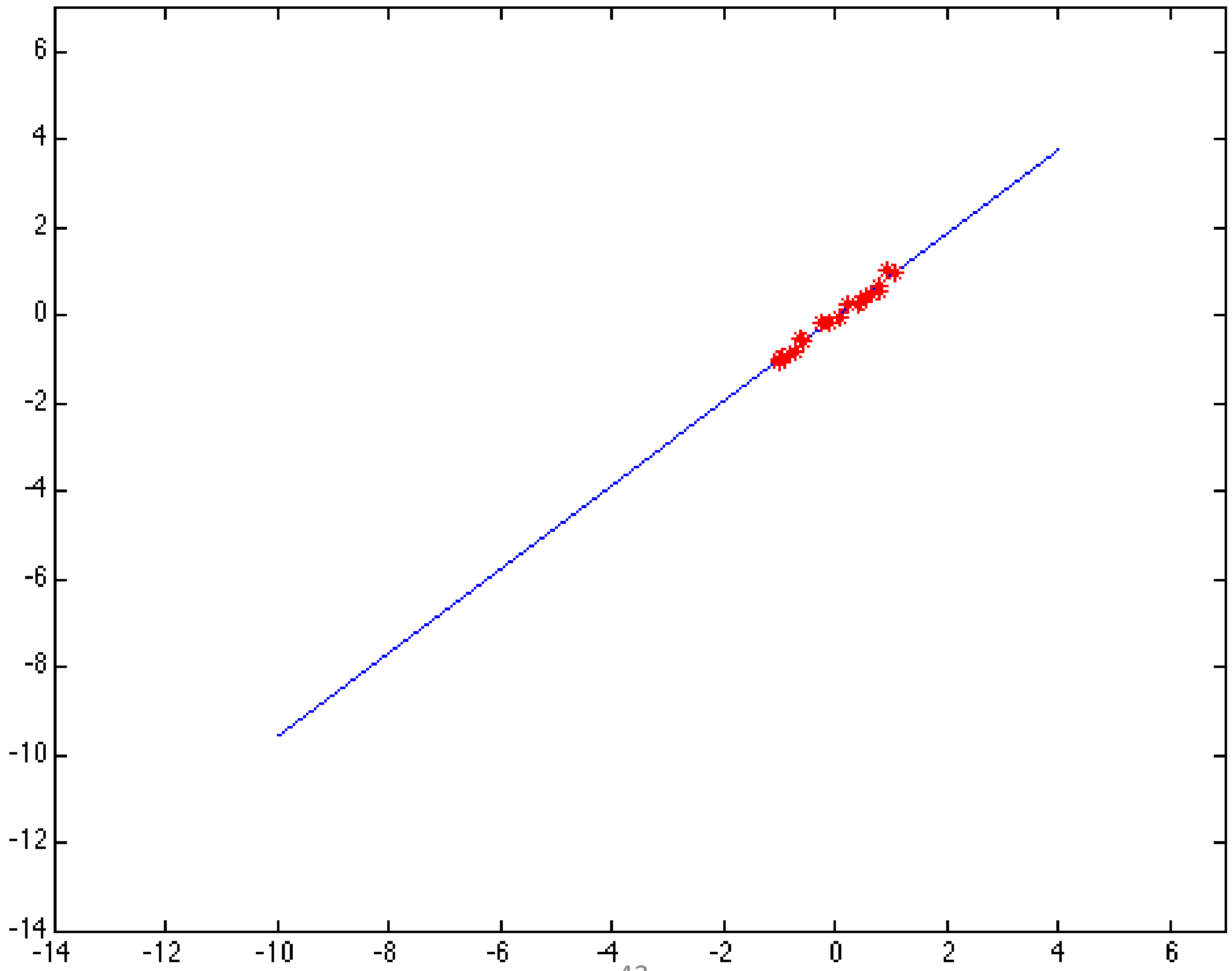
What could go wrong?

Outliers



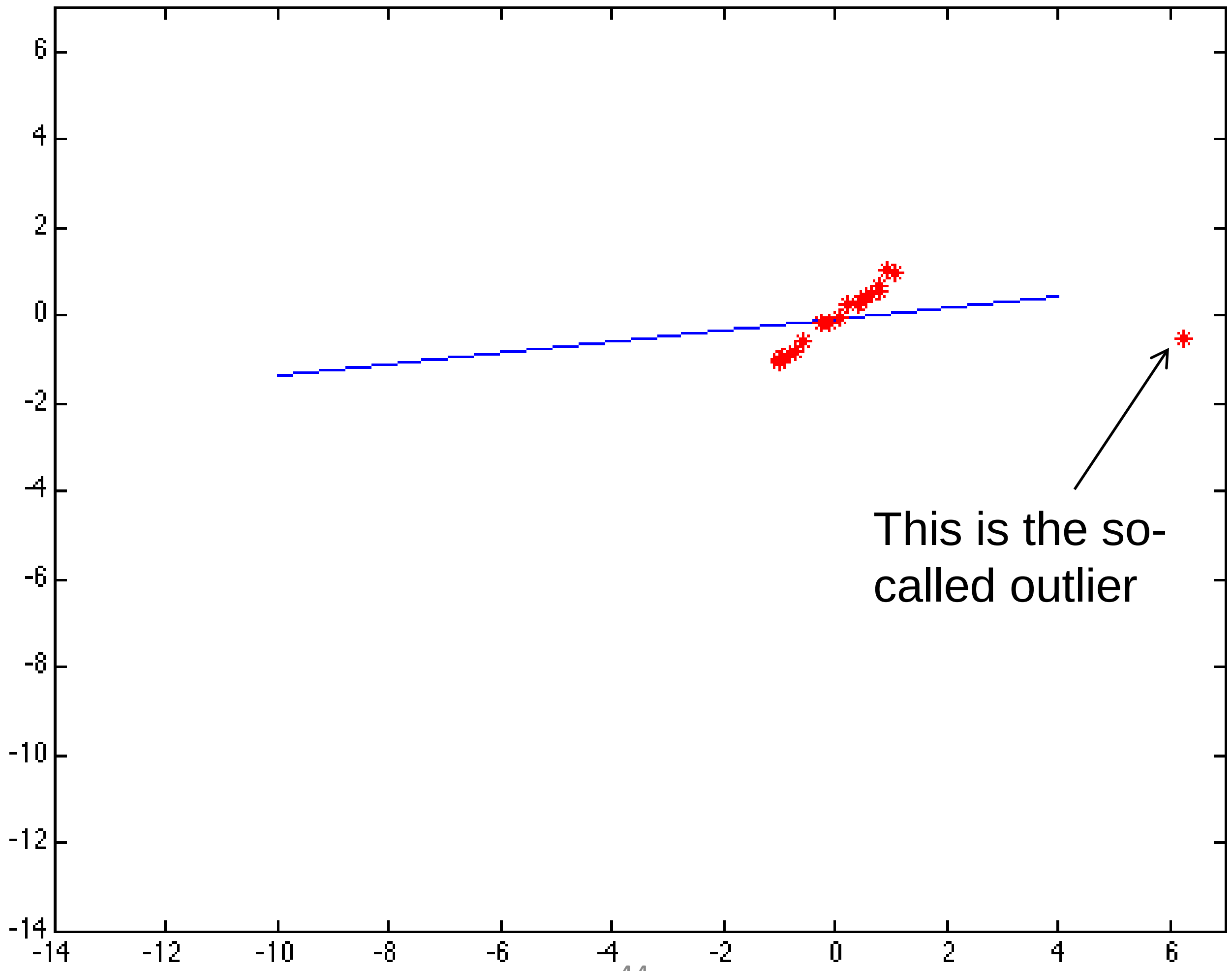
Problems of Least Square Fitting

- Let's consider a simpler example… linear regression



Problems of Least Square Fitting

- A single outlier could 'drag' the line away….



Robust Fitting

- Squared penalty can be a source of bias in the presence of noise points
  - One is an M-estimator
    - A error function with an upper bound
    - Square nearby, threshold far away
  - Another is RANSAC
    - Search for good points

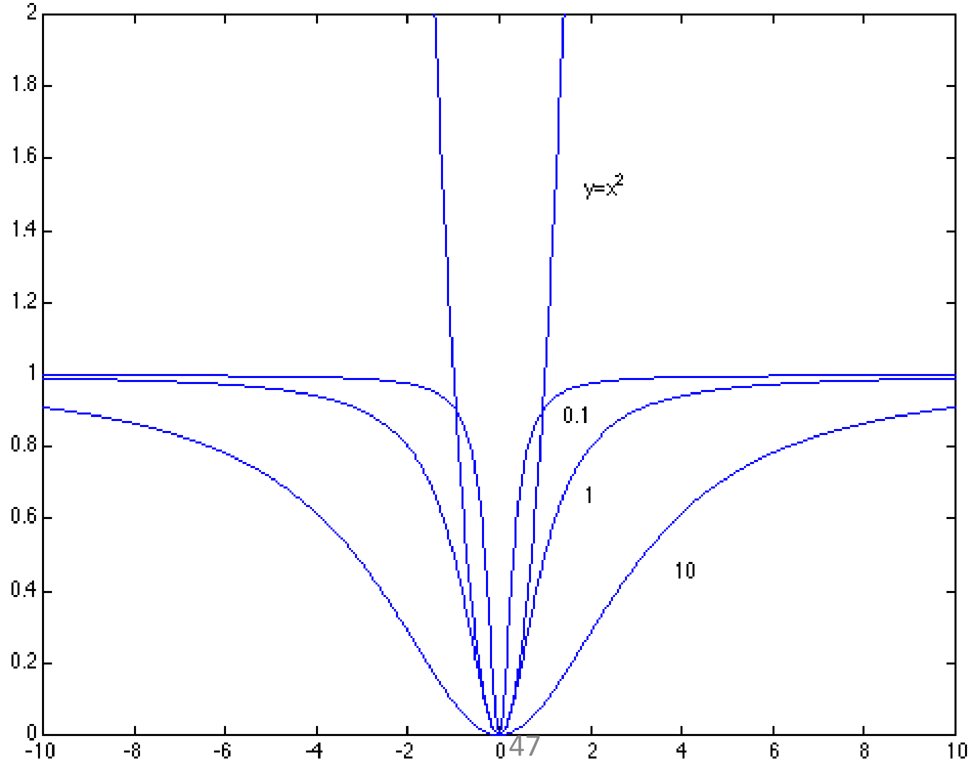
M-estimators

- Least square minimizes $\sum_i r^2(x_i, \theta)$
  - $r(x_i, \theta)$ is the residual/distance
  - In the case of line fitting, $r(\cdot) = (ax_i + by_i + c)$

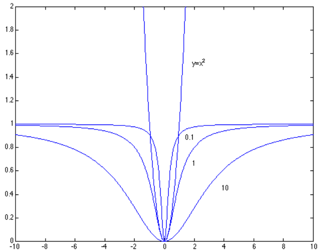
- M-estimators minimize $\sum_i \rho(r(\cdot); \sigma)$
  - There are different choices of the function $\rho(\cdot)$
  - Typically, the result objective function is even more nonlinear!!
    - Harder to optimize

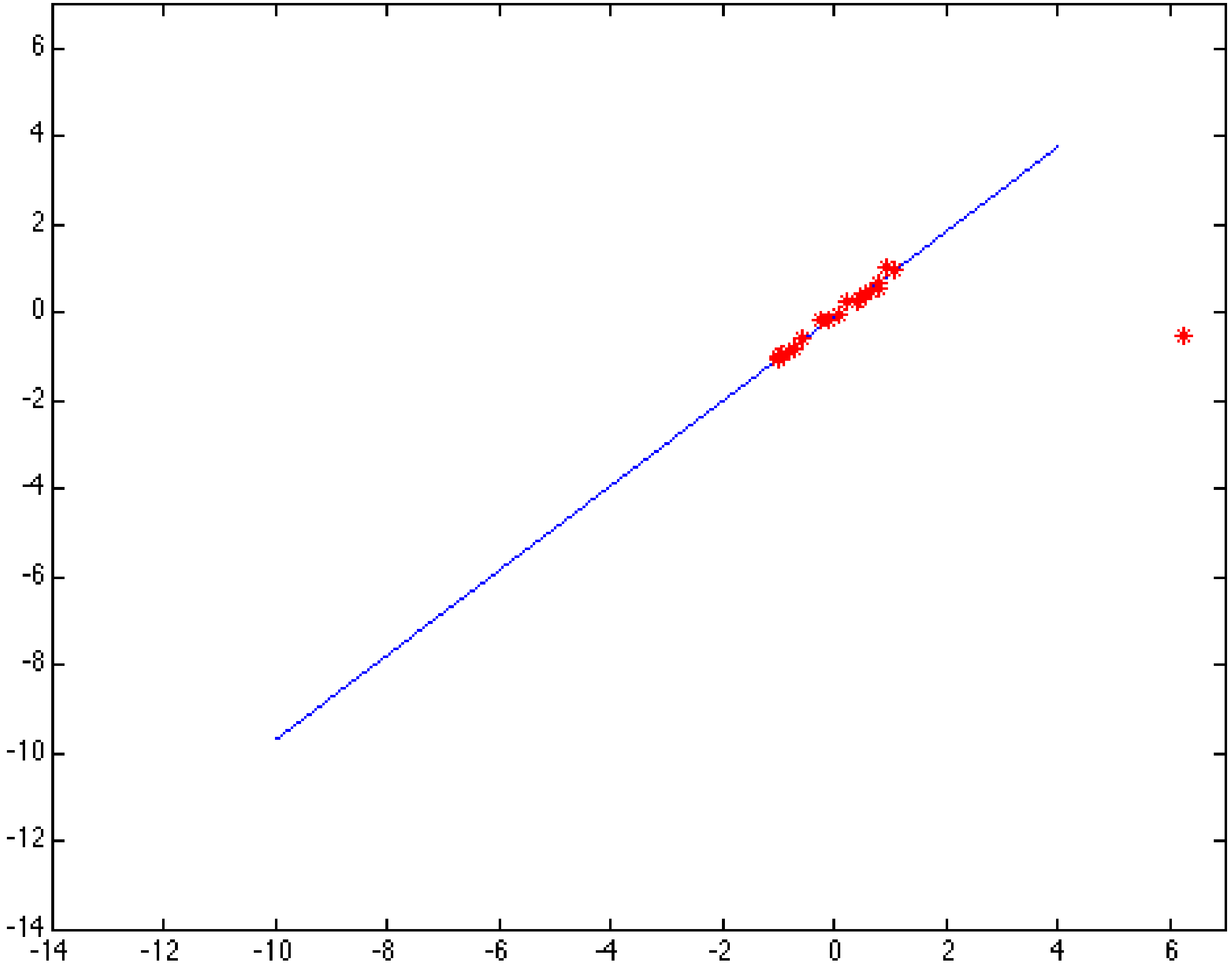
Typical Examples of $\boldsymbol{\rho}$

- A popular choice is $\rho(r;\sigma)=\dfrac{r^2}{r^2+\sigma^2}$
  - When r is too large, ρ becomes saturated to 1
  - When r is small, ρ functions like r^2
- ρ is controlled by the parameter σ



Examples





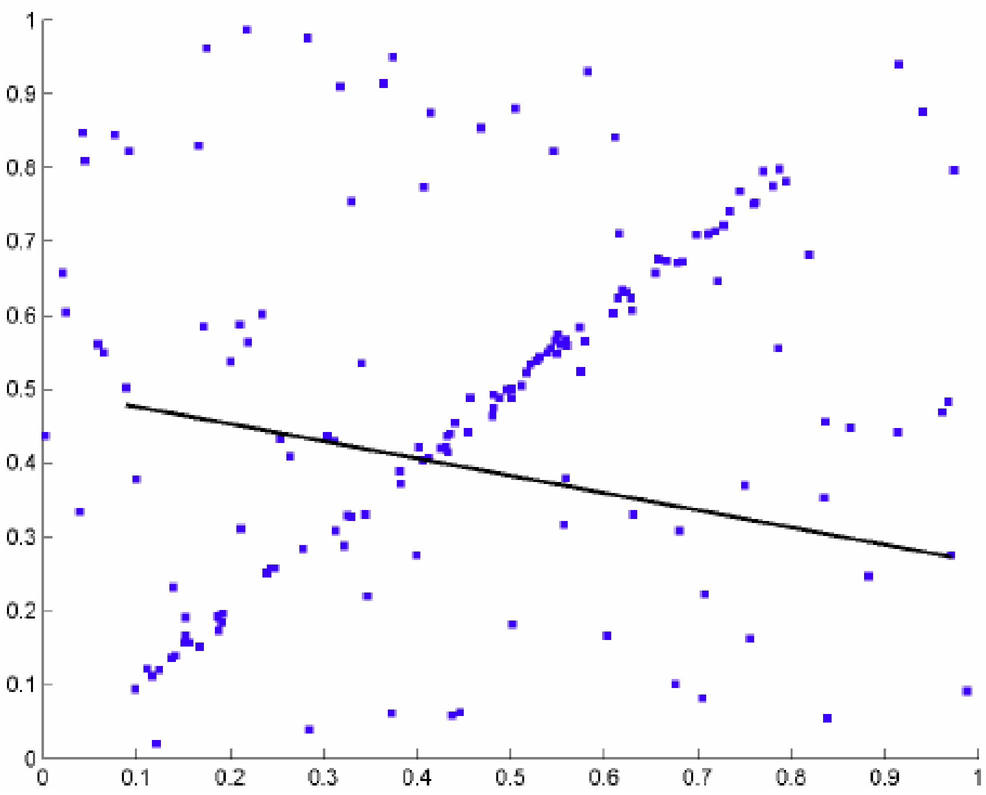
Questions?

The Idea of RANSAC

- Choose a small set of samples at random
  - the set should be as small as possible, so that it is more likely outlier-free (e.g. two points for line fitting)
- Fit a model – e.g. the line
- Count the number of inliers
  - Points that “agree” with the model are considered inliers
  - “Agree” = within a small distance of the line

- Repeat the random sampling many times, select the one with the largest number of inliers
- Take these inliers as true inliers and fit again

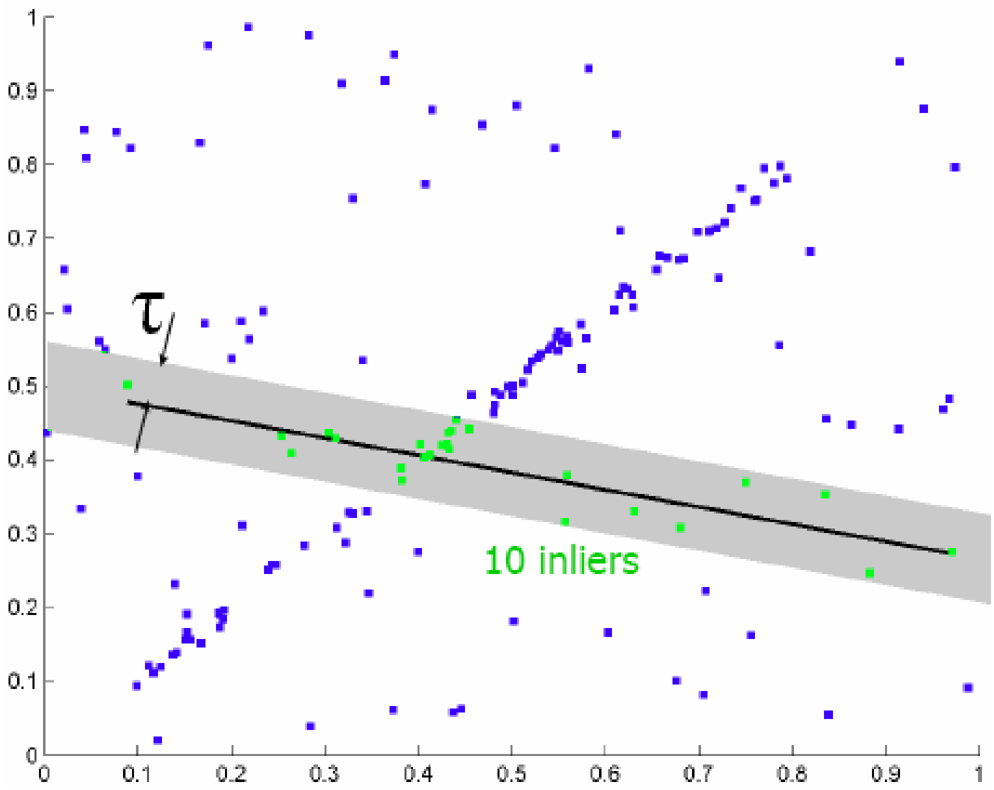
Example



1. sample randomly two points, get a line

RANSAC

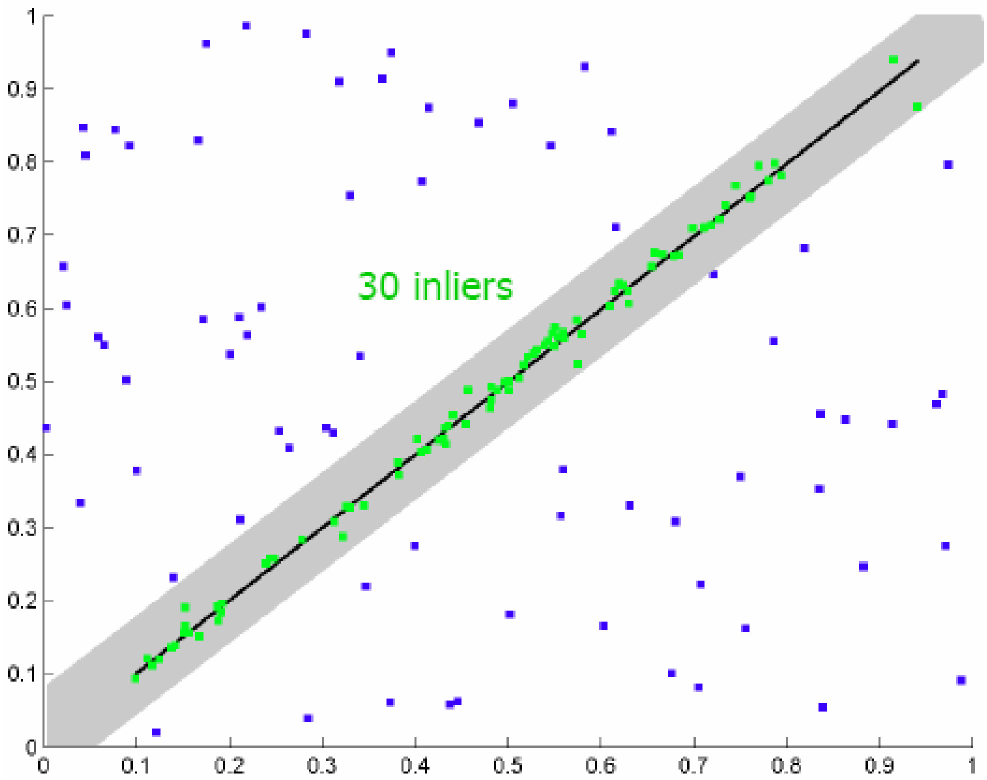
Example



1. sample randomly two points, get a line
2. count inliers for threshold T

RANSAC

Example



1. sample randomly two points, get a line
2. count inliers for threshold T
3. repeat N times and select model with most inliers

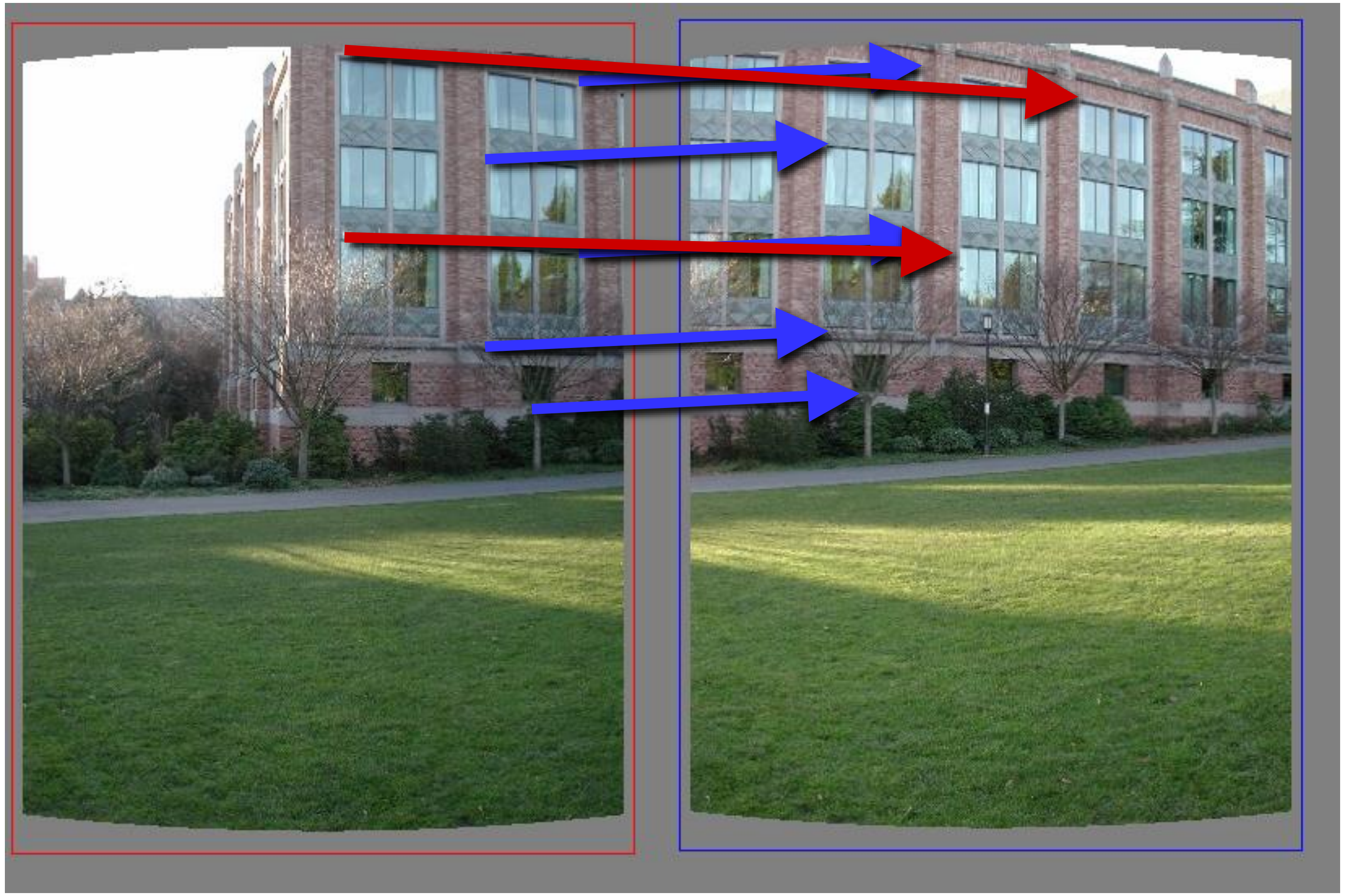
RANSAC

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the largest set of inliers, and fit the model again

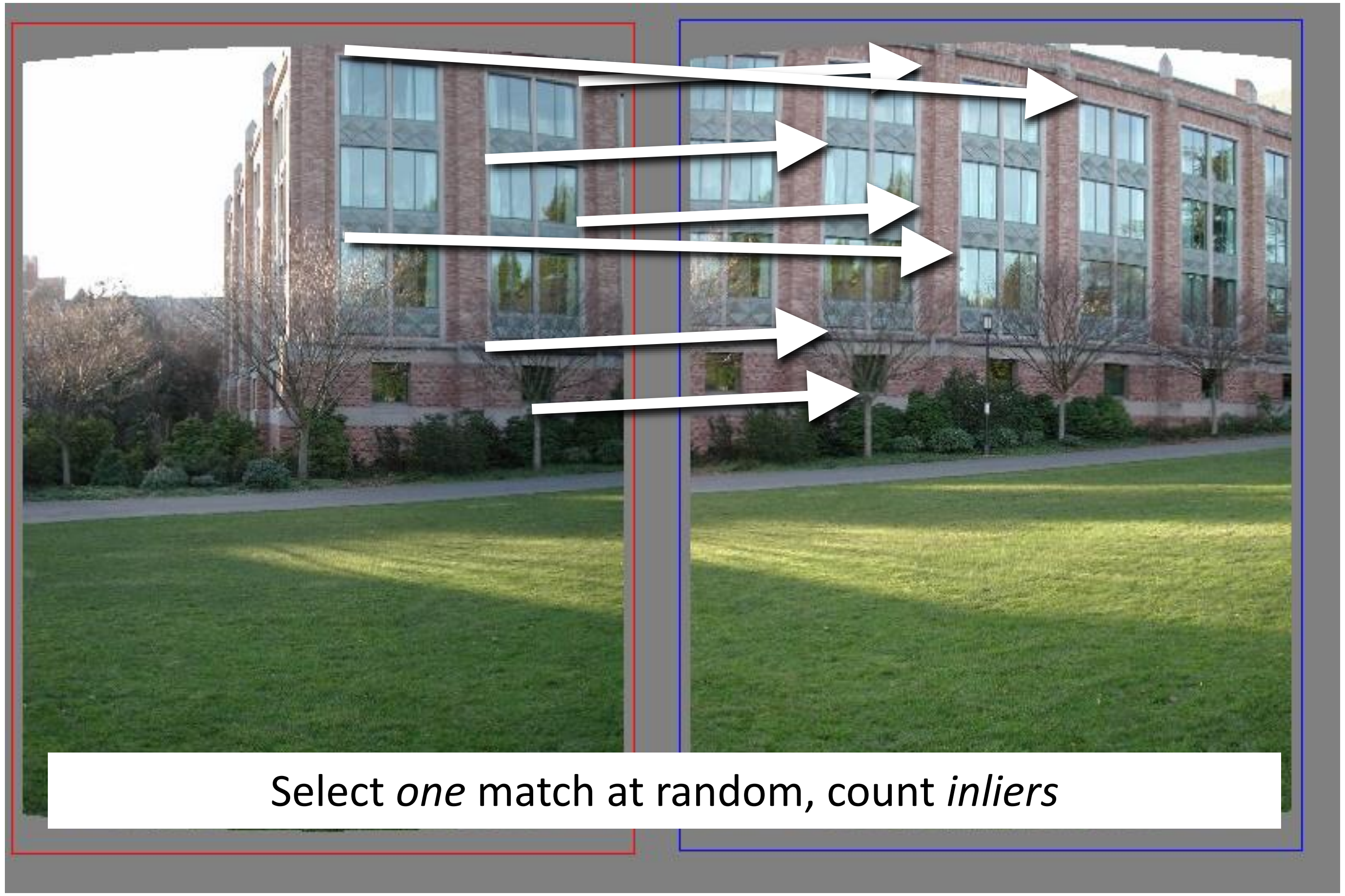
Translation Estimation

- Two “outlier”, five “inliers”



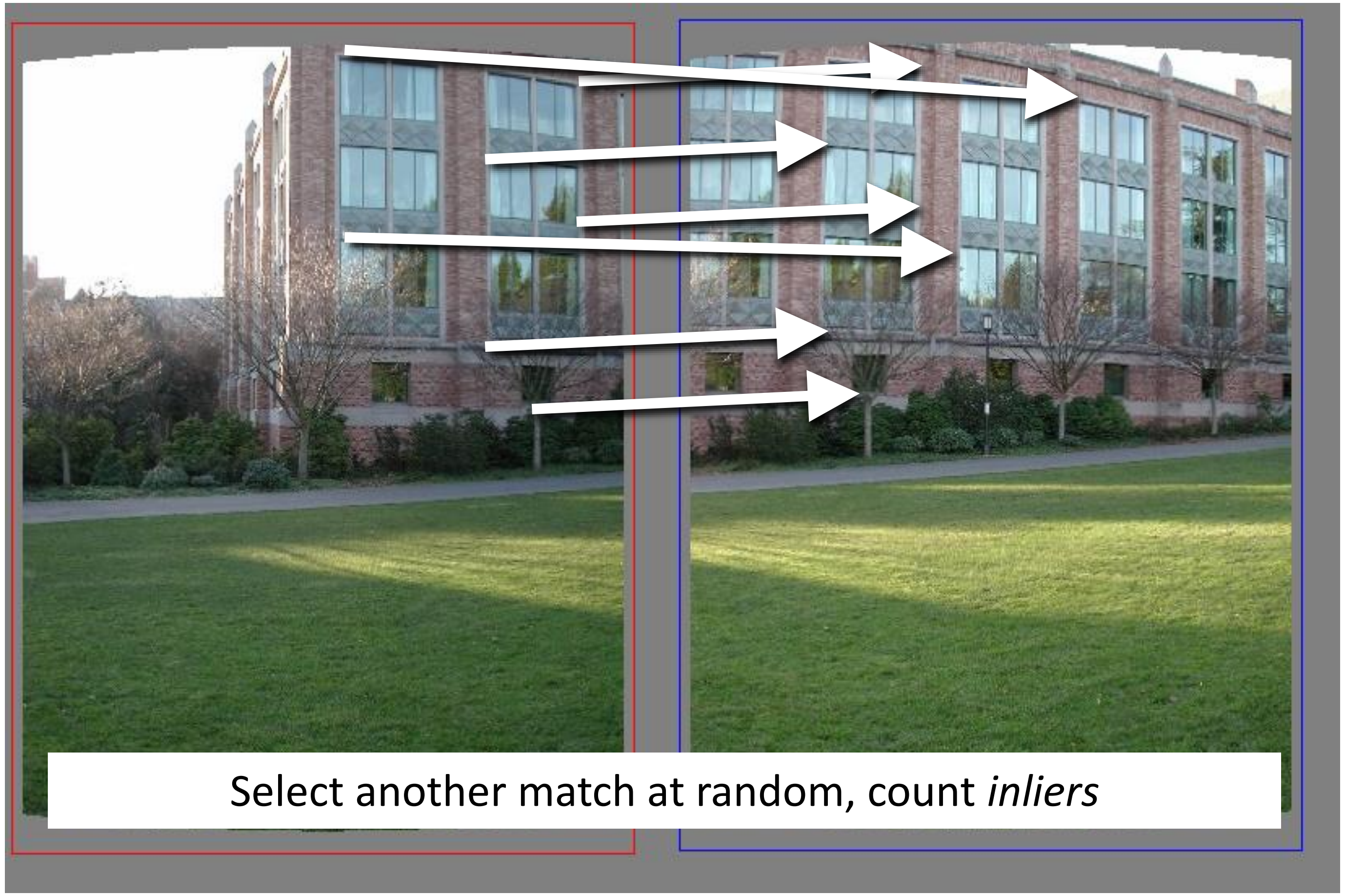
RAndom SAmple Consensus

- Randomly select one pair, count inliers, and find five inliers



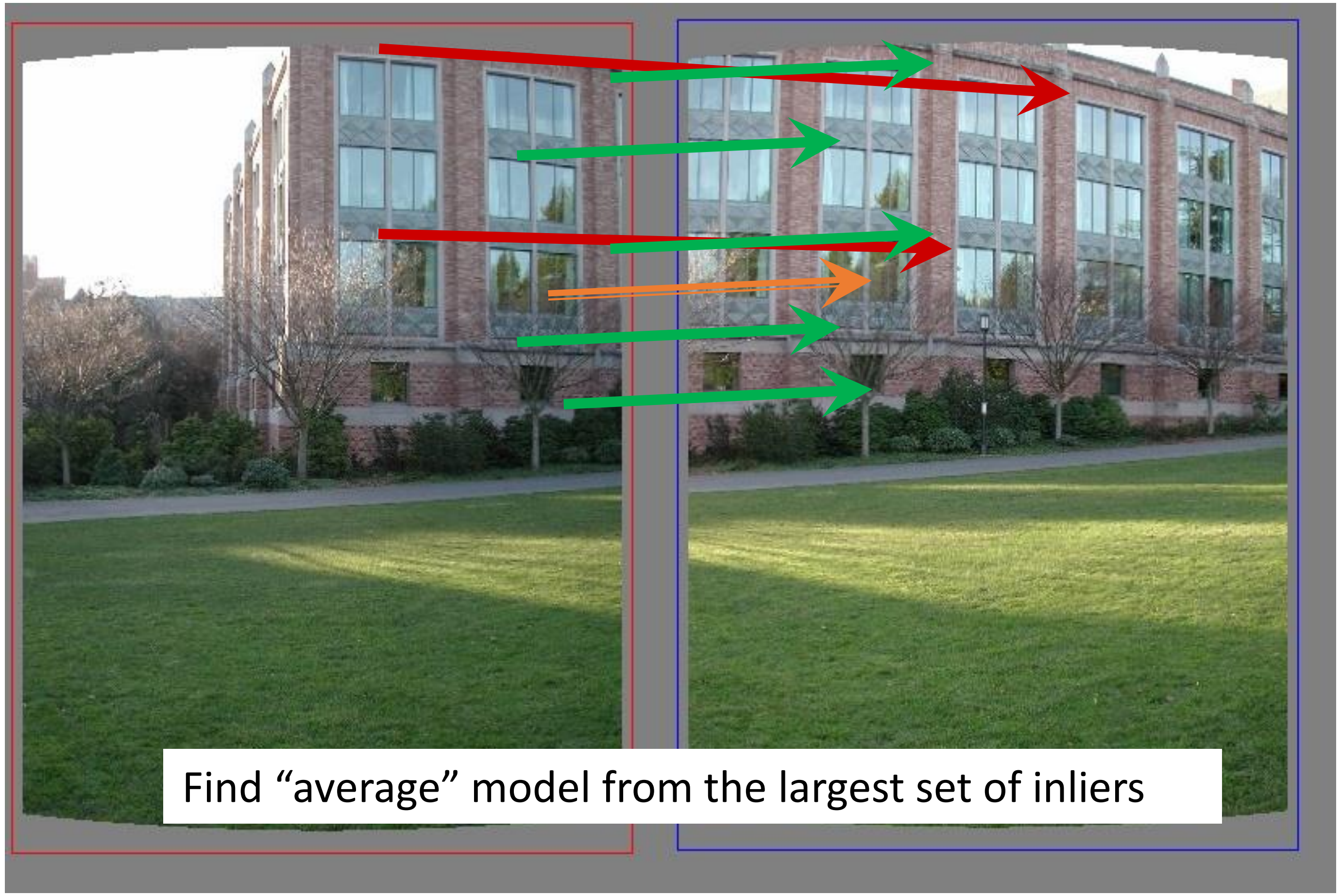
RAndom SAmple Consensus

- Randomly select one pair, count inliers, and find just one inlier



RAndom SAmple Consensus

- Randomly select one pair, count inliers, and find just one inlier



Find “average” model from the largest set of inliers

RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers
  - "All good matches are alike; every bad match is bad in its own way."

    – Tolstoy via Alyosha Efros

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
  - How many rounds do we need?
  - I will explain the calculation soon

Questions?

How Many Iterations?

- Suppose each time s samples are selected to fit a model
- Suppose e is the percentage of outlier
- Suppose N iterations have been run
- Success = at least one set of samples is outlier-free
  - So what is the probability p of success?
- The math:
  - The probability q that a set is outlier-free $q = (1 - e)^s$
  - The probability $(1 - p)$ that all sets have outliers

$$1 - p = (1 - q)^N$$

- Decide the confidence probability p (e.g. 0.99), then determine N

$$N = \log(1 - p) / \log(1 - (1 - e)^s)$$

How Many Iterations?

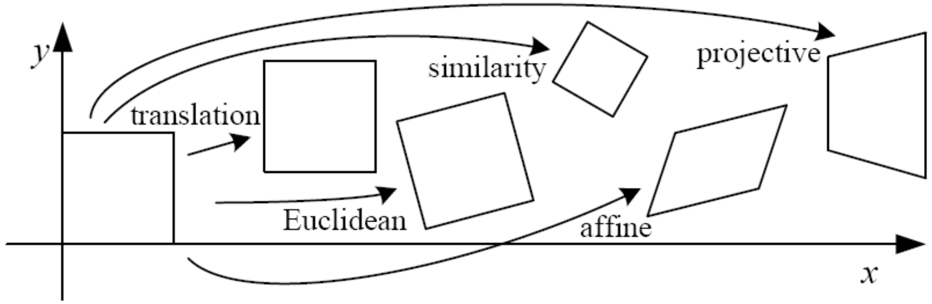
- Suppose each time s samples are selected to fit a model
- Suppose e is the percentage of outlier
- Decide the confidence probability p (e.g. 0.99), then determine N

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

How Big is *s*?

- For alignment, depends on the motion model
  - Here, each sample is a correspondence (pair of matching points)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8	straight lines	

Adaptively Determining N

e is often unknown a priori,

- so pick worst case, e.g. 50%,
- adapt if more inliers are found, e.g. 80% would yield e=0.2

The algorithm:

- $N=\infty$, *sample_count* =0
- While N >*sample_count* repeat
  - Choose a sample and count the number of inliers
  - Set e=1-(number of inliers)/(total number of points)
  - Recompute N from e
  - Increment the *sample_count* by 1
- Terminate

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice

- Cons
  - Parameters to tune
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

Questions?

Cost functions

- Algebraic distance
- Geometric distance
- Reprojection error

- Comparison
- Geometric interpretation

Algebraic Distance

DLT minimizes $\|\mathbf{Ah}\|$

$\mathbf{e} = \mathbf{Ah}$ residual vector

$\mathbf{e}_i$ the first two components of $\boldsymbol{e}$

$$d_{\mathrm{alg}}(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2 = \|\mathbf{e}_i\|^2 = \left\| \begin{bmatrix} 0^\mathsf{T} & -\mathbf{x}_i^\mathsf{T} & -y'_i \mathbf{x}_i^\mathsf{T} \\ -\mathbf{x}_i^\mathsf{T} & 0^\mathsf{T} & -x'_i \mathbf{x}_i^\mathsf{T} \end{bmatrix} \mathbf{h} \right\|^2$$

The *algebraic distance* between x_1, x_2:

$$d_{\mathrm{alg}}(\mathbf{x}_1, \mathbf{x}_2)^2 = a_1^2 + a_2^2 \text{ where } \mathbf{a} = (a_1, a_2, a_3)^\mathrm{T} = \mathbf{x}_1 \times \mathbf{x}_2$$

$$\sum_i d_{\mathrm{alg}}(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i)^2 = \sum_i \|\mathbf{e}_i\|^2 = \|\mathbf{Ah}\|^2 = \|\mathbf{e}\|^2$$

Not geometrically/statistically meaningful, but given good normalization it works fine and is very fast (use for initialization)

Geometric Distance

$\mathbf{x}$ measured coordinates
$\hat{\mathbf{x}}$ estimated coordinates

$d(.,.)$ Euclidean distance (in image)

- Error in one image

$$\hat{\mathbf{H}} = \underset{\mathrm{H}}{\operatorname{argmin}} \sum_i d\left(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i\right)^2$$

- Symmetric transfer error

$$\hat{\mathbf{H}} = \underset{\mathrm{H}}{\operatorname{argmin}} \sum_i d\left(\mathbf{x}_i, \mathbf{H}^{-1}\mathbf{x}'_i\right)^2 + d\left(\mathbf{x}'_i, \mathbf{H}\mathbf{x}_i\right)^2$$

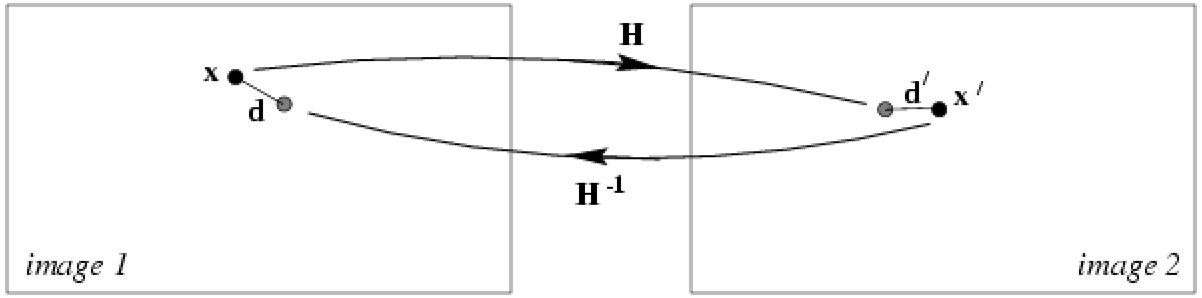
- Reprojection error

$$\left(\hat{\mathbf{H}}, \hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i\right) = \underset{\mathrm{H}, \hat{\mathrm{x}}_i, \hat{\mathrm{x}}'_i}{\operatorname{argmin}} \sum_i d\left(\mathbf{x}_i, \hat{\mathbf{x}}_i\right)^2 + d\left(\mathbf{x}'_i, \hat{\mathbf{x}}'_i\right)^2$$

$$\text{subject to } \hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\hat{\mathbf{x}}_i$$

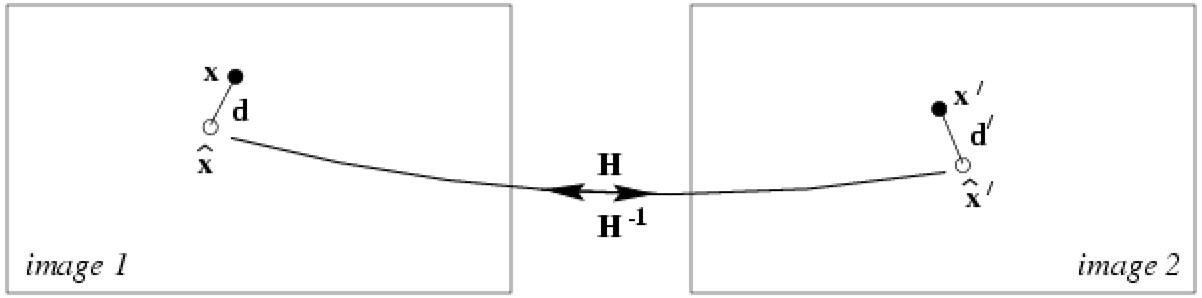
Visualization

Symmetric geometric error



$$d\left(\mathbf{x},\mathbf{H}^{-1}\mathbf{x}'\right)^2+d\left(\mathbf{x}',\mathbf{H}\mathbf{x}\right)^2$$

Reprojection error



$$d\left(\mathbf{x},\hat{\mathbf{x}}\right)^2+d\left(\mathbf{x}',\hat{\mathbf{x}}'\right)^2$$

Minimize the re-projection error

- The objective function

$$\{\mathbf{H}, \hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i\} = \arg\min \sum d(\mathbf{x}_\mathrm{i}, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_\mathrm{i}, \hat{\mathbf{x}}'_i)^2$$

$$= (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (x'_i - \hat{x}'_i)^2 + (y'_i - \hat{y}'_i)^2$$

$$\hat{x}'_i = \frac{a\hat{x}_i + b\hat{y}_i + c}{g\hat{x}_i + h\hat{y}_i + i} \qquad \hat{y}_i' = \frac{d\hat{x}_i + e\hat{y}_i + f}{g\hat{x}_i + h\hat{y}_i + i} \qquad \begin{bmatrix} \hat{x}'_i \\ \hat{y}'_i \\ 1 \end{bmatrix} \propto \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- Nonlinear, how do we minimize it?
  - General methods
    - Newton iteration
    - Levenberg-Marquardt

Questions?