NUS National University of Singapore | School of Computing

CS4277 / CS5477
3D Computer Vision

Lecture 3:
Absolute Conic and Robust Homography Estimation

Asst. Prof. Lee Gim Hee

AY 2019/20

Semester 2

Course Schedule

Week	Date	Topic	Assignments
1	15 Jan	2D and 1D projective geometry	
2	22 Jan	Circular points and 3D projective geometry	
3	29 Jan	No Lecture	
4	05 Feb	Absolute conic and robust homography estimation (Video Lecture)	**Assignment 1**: Panoramic stitching (15%)
5	12 Feb	Camera models and calibration	
6	19 Feb	The fundamental and essential matrices	**Due:** Assignment 1 **Assignment 2**: Camera calibration (15%)
-	26 Feb	Semester Break	No lecture
7	04 Mar	Multiple-view geometry from points and/or lines	**Due:** Assignment 2 **Assignment 3**: Relative and absolute pose estimation (20%)
8	11 Mar	Absolute pose estimation from points and/or lines	
9	18 Mar	Structure-from-Motion (SfM) and Visual Simultaneous Localization and Mapping (vSLAM)	**Due:** Assignment 3
10	25 Mar	Two-view and multi-view stereo	**Assignment 4:** Dense 3D model from multi-view stereo (20%)
11	01 Apr	Generalized cameras	
12	08 Apr	Factorization and non-rigid structure-from-motion	**Due:** Assignment 4
13	15 Apr	Auto-Calibration	

*Make-up lecture (15 Feb, Sat 9.30am – 12.30pm): **Single view metrology** (Webcast will be provided)

Learning Outcomes

- Students should be able to:

1. Describe the plane at infinity and its invariance under affine transformation.
2. Describe the absolute conic (and its absolute dual quadrics) and its invariance under similarity transformation.
3. Explain the difference between the algebraic, geometric and Sampson errors, and apply them on homography estimation.
4. Use the RANSAC algorithm for robust estimation.

Acknowledgements

- A lot of slides and content of this lecture are adopted from:

1. R. Hartley, and Andrew Zisserman: “Multiple view geometry in computer vision”, Chapter 3 and 4.

2. Y. Ma, S. Soatto, J. Kosecka, S. S. Sastry, “ An invitation to 3-D vision”, Chapter 5.3.

The Plane at Infinity

- The plane at infinity has the canonical position $\boldsymbol{\pi}_\infty = (0, 0, 0, 1)^T$ in affine 3-space.

$\pi_\infty^T D = 0$

- It contains the directions $\boldsymbol{D} = (X_1, X_2, X_3, 0)^T$, and enables the identification of affine properties such as parallelism, particularly:

i. Two planes are parallel if, and only if, their line of intersection is on $\boldsymbol{\pi}_\infty$.

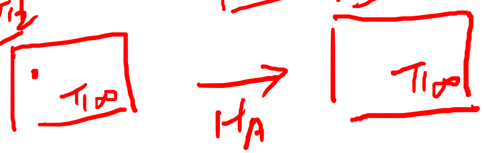
ii. A line is parallel to another line, or to a plane, if the point of intersection is on $\boldsymbol{\pi}_\infty$.

The Plane at Infinity

- The plane at infinity, $\boldsymbol{\pi}_\infty$, is a fixed plane under the projective transformation H if, and only if, H is an affinity, i.e.

$$\boldsymbol{\pi}'_\infty = H_A^{-T}\boldsymbol{\pi}_\infty = \begin{bmatrix} \mathrm{A}^{-T} & \mathbf{0} \\ -\mathrm{t}^T\mathrm{A}^{-T} & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}}_{\pi_\infty} = \boldsymbol{\pi}_\infty$$

- **Remarks:**



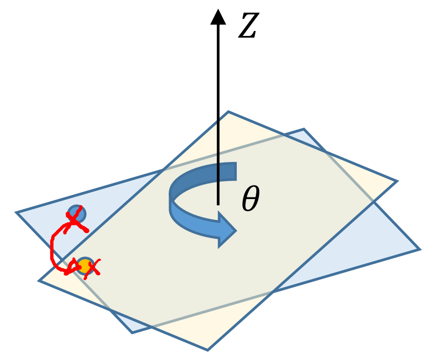
1. The plane $\boldsymbol{\pi}_\infty$ is, in general, only fixed as a set under an affinity; it is not fixed pointwise.

2. Under a particular affinity (for example a Euclidean motion) there may be planes in addition to $\boldsymbol{\pi}_\infty$ which are fixed. However, only $\boldsymbol{\pi}_\infty$ is fixed under any affinity.

The Plane at Infinity

Example: Consider the Euclidean transformation represented by the matrix

$$\mathrm{H_E} = \begin{bmatrix} \mathrm{R} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$



- This is a rotation by θ about the Z-axis with a zero translation, hence, there is a pencil of fixed planes orthogonal to the z-axis.

- The planes are fixed as sets, but not pointwise as any (finite) point (not on the axis) is rotated in horizontal circles by this Euclidean action.

The Plane at Infinity

Example:

- Algebraically, the fixed planes of H are the eigenvectors of H^T, i.e.

$$H^{-T}\mathbf{v} = \lambda\mathbf{v} \;\Leftrightarrow\; H^{-T}\boldsymbol{\pi} = \lambda\boldsymbol{\pi},$$

- $\lambda, \mathbf{v}$ are the eigenvalues and eigenvectors of H^T and H^{-T}.

- In this case, the eigenvalues and eigenvectors of $H_{\mathrm{E}}^{\mathrm{T}}$ are $\{e^{i\theta}, e^{-i\theta}, 1, 1\}$ and

$$\mathbf{E}_1 = \begin{pmatrix} 1 \\ i \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{E}_2 = \begin{pmatrix} 1 \\ -i \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{E}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{E}_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

- The eigenvectors $\mathbf{E}_1$ and $\mathbf{E}_2$ are imaginary planes, and will not be discussed further.

The Plane at Infinity

Example:

$$\mathbf{E}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{E}_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

In addition to $\mathbf{E}_4$ (i.e. the plane at infinity), we can see that there is a pencil of fixed planes spanned by $\mathbf{E}_3$ and $\mathbf{E}_4$ under H_E, i.e.

$$\boldsymbol{\pi} = \mu\mathbf{E}_3 + \lambda\mathbf{E}_4.$$

The Plane at Infinity

- We will see in Lecture 6 that uncalibrated two-view reconstructions lead to projective ambiguity.
- The identified π_∞ can be used to remove the projective ambiguity, where affine properties can be measured.

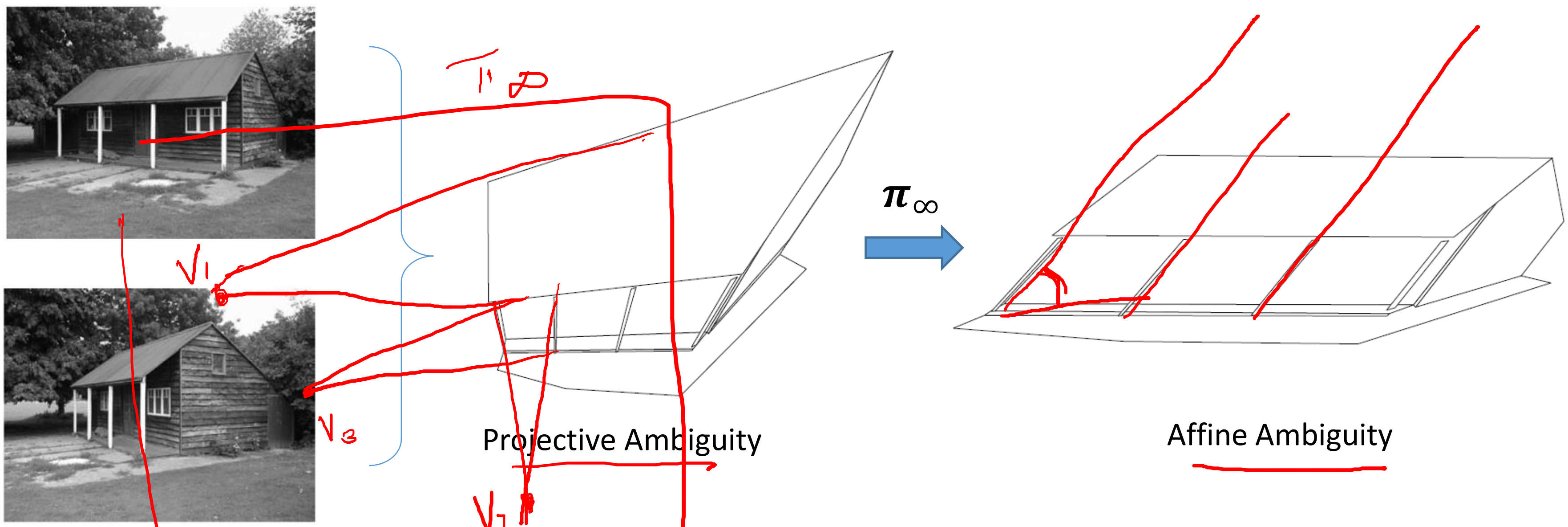


Image source: "Multiple View Geometry in Computer Vision", Richard Hartley and Andrew Zisserman

The Absolute Conic

- The absolute conic, Ω_∞, is a (point) conic on $\boldsymbol{\pi}_\infty$.
- In a metric frame $\boldsymbol{\pi}_\infty = (0,0,0,1)^T$, and points on Ω_∞ satisfy

$$\left.\begin{array}{r} X_1^2 + X_2^2 + X_3^2 \\ X_4 \end{array}\right\} = 0.$$

- Note that two equations are required to define Ω_∞.

The Absolute Conic

- For directions on $\boldsymbol{\pi}_\infty$ (i.e. points with $X_4 = 0$) the defining equation can be written

$$(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3)\mathrm{I}(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3)^\mathsf{T} = 0$$

- So that Ω_∞ corresponds to a conic C with matrix $\mathrm{C} = I$; it is thus a conic of purely imaginary points on $\boldsymbol{\pi}_\infty$.

- The conic Ω_∞ is a geometric representation of the 5 additional degrees of freedom required to specify metric properties in an affine coordinate frame.

The Absolute Conic

- The absolute conic, Ω_∞ , is a fixed conic under the projective transformation H if, and only if, H is a similarity transformation.

Proof:

Since the absolute conic lies in $\boldsymbol{\pi}_\infty$, a transformation fixing it must fix $\boldsymbol{\pi}_\infty$, and hence must be affine, i.e.

$$\mathrm{H_A} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & \mathbf{1} \end{bmatrix}.$$

Sim.

$$\underbrace{H^{-T}\Omega_\infty}\, H^{-1}$$

$$\Downarrow$$

At $\boldsymbol{\pi}_\infty$, $\Omega_\infty = I_{3\times3}$, and since it is fixed by H_A, one has $\mathrm{A}^{-T}\mathrm{IA}^{-1} = \mathrm{I}$ (up to scale), and taking inverses gives $\mathrm{AA}^\mathrm{T} = \mathrm{I}$. $\not\!\!\not$

This means that A is orthogonal, hence a scaled rotation, or scaled rotation with reflection, i.e. similarity transform.

□

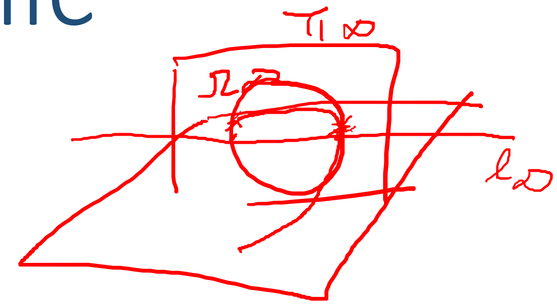
The Absolute Conic

- Even though Ω_∞ does not have any real points, it shares the properties of any conic:

1. The conic Ω_∞ is only fixed as a set by a general similarity; it is not fixed pointwise.

   Remark: This means that under a similarity a point on Ω_∞ may travel to another point on Ω_∞, but it is not mapped to a point off the conic.

The Absolute Conic



2. All circles intersect Ω_∞ in two points.

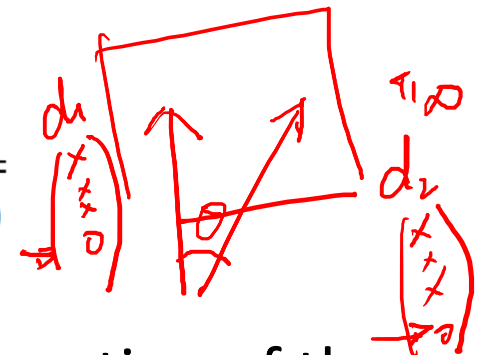
   Remarks: Suppose the support plane of the circle is $\boldsymbol{\pi}$. Then $\boldsymbol{\pi}$ intersects $\boldsymbol{\pi}_\infty$ in a line, and this line intersects Ω_∞ in two points. These two points are the circular points of $\boldsymbol{\pi}$.

3. All spheres intersect $\boldsymbol{\pi}_\infty$ in Ω_∞ .

The Absolute Conic

- The angle between two lines with directions (3-vectors) $\mathbf{d}_1$ and $\mathbf{d}_2$ is given by:

$$\cos\theta = \frac{(\mathbf{d}_1^{\top}\Omega_\infty \mathbf{d}_2)}{\sqrt{(\mathbf{d}_1^{\top}\Omega_\infty \mathbf{d}_1)(\mathbf{d}_2^{\top}\Omega_\infty \mathbf{d}_2)}}$$



- where $\mathbf{d}_1$ and $\mathbf{d}_2$ are the points of intersection of the lines with the plane $\boldsymbol{\pi}_\infty$ containing the conic Ω_∞.

- And Ω_∞ is the matrix representation of the absolute conic in that plane.

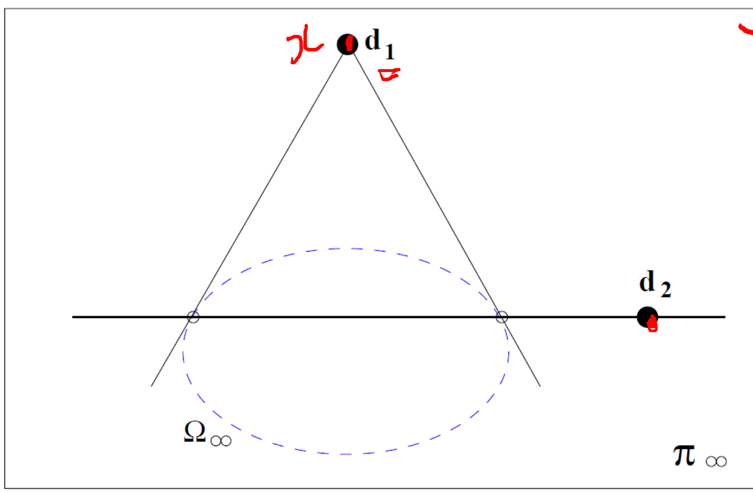
The Absolute Conic: Orthogonality and Polarity

- Two directions $\mathbf{d}_1$ and $\mathbf{d}_2$ are orthogonal if

$$\Rightarrow \quad \mathbf{d}_1^T \Omega_\infty \mathbf{d}_2 = 0. \qquad \cos 90^\circ = 0$$

- Thus orthogonality is encoded by conjugacy with respect to Ω_∞.

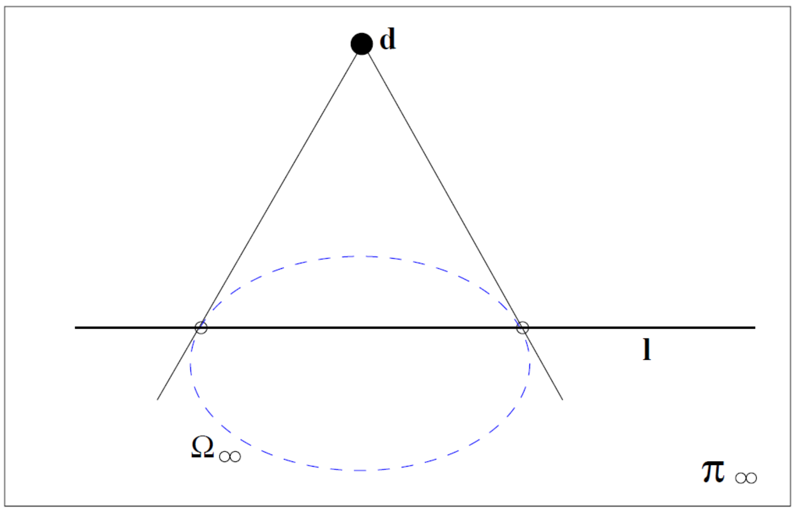
$y \leftarrow l = Cx$ (polar, pole) $\qquad y^T C x = 0$ (d_2, d_1)



On $\boldsymbol{\pi}_\infty$ orthogonal directions $\mathbf{d}_1$, $\mathbf{d}_2$ are conjugate with respect to the conic Ω_∞.

Image source: “Multiple View Geometry in Computer Vision”, Richard Hartley and Andrew Zisserman

The Absolute Conic: Orthogonality and Polarity



$l = Cx$

$l = \Omega_\infty d$

A plane normal direction $\mathbf{d}$ and the intersection line $\mathbf{l}$ of the plane with $\boldsymbol{\pi}_\infty$ are in pole–polar relation with respect to Ω_∞.

Image source: “Multiple View Geometry in Computer Vision”, Richard Hartley and Andrew Zisserman

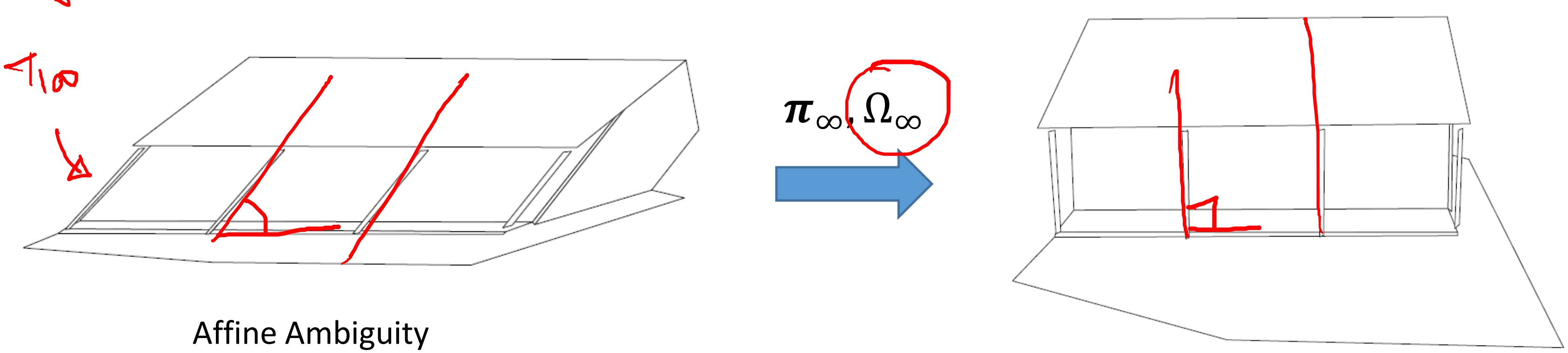
The Absolute Conic: Orthogonality and Polarity

- We will see in Lecture 5 that the imaged absolute conic can be used to recover the camera intrinsics, i.e. calibration.

- Furthermore, we will see in Lecture 6 that both the absolute conic and plane at infinity can be used to remove affine distortion, hence the metric properties can be measured.

Projective

π_∞



Affine Ambiguity

Similarity Ambiguity

Image source: “Multiple View Geometry in Computer Vision”, Richard Hartley and Andrew Zisserman

The Absolute Dual Quadric

- The dual of the absolute conic Ω_∞ is a degenerate dual quadric in 3-space called the absolute dual quadric, and denoted Q^*_∞.
- **Geometrically** Q^*_∞ consists of the planes tangent to Ω_∞, so that Ω_∞ is the "rim" of Q^*_∞, hence called a rim quadric.
- **Algebraically** Q^*_∞ is represented by a 4×4 homogeneous matrix of rank 3, with the canonical form:

$$Q^*_\infty = \begin{bmatrix} \mathrm{I} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix}.$$

rank deficient.

The Absolute Dual Quadric

- The dual quadric Q^*_∞ is a degenerate quadric.
- There are 8 degrees of freedom (a symmetric matrix has 10 independent elements, but the irrelevant scale and zero determinant).

The Absolute Dual Quadric

- The absolute dual quadric, Q^*_∞, is fixed under the projective transformation H if, and only if, H is a similarity.

Proof:

Since Q^*_∞ is a dual quadric, it is fixed under H if and only if $Q^*_\infty = HQ^*_\infty H^T$. Applying this with an arbitrary transform

$$\mathrm{H} = \begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{v}^\mathsf{T} & k \end{bmatrix}, \text{ we get } \begin{bmatrix} \mathrm{I} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix} = \begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{v}^\mathsf{T} & k \end{bmatrix} \begin{bmatrix} \mathrm{I} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} \mathrm{A}^\mathsf{T} & \mathbf{v} \\ \mathbf{t}^\mathsf{T} & k \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{AA}^\mathsf{T} & \mathrm{A}\mathbf{v} \\ \mathbf{v}^\mathsf{T}\mathrm{A}^\mathsf{T} & \mathbf{v}^\mathsf{T}\mathbf{v} \end{bmatrix}$$

Q^*_∞

Sim.

#

The Absolute Dual Quadric

Proof:

$AA^T = I$

$$\begin{bmatrix} \mathtt{I} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix} = \begin{bmatrix} \mathtt{A}\mathtt{A}^\mathsf{T} & \mathtt{A}\mathbf{v} \\ \mathbf{v}^\mathsf{T}\mathtt{A}^\mathsf{T} & \mathbf{v}^\mathsf{T}\mathbf{v} \end{bmatrix}$$

which must be true up to scale.

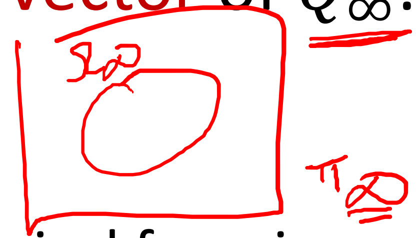
By inspection, this equation holds if and only if $\mathbf{v} = \mathbf{0}$ and A is a scaled orthogonal matrix (scaling, rotation and possible reflection).

In other words, H is a similarity transform.

$\square$

The Absolute Dual Quadric

- The plane at infinity $\boldsymbol{\pi}_\infty$ is the null-vector of Q^*_∞.



Remarks:

This is easily verified when Q^*_∞ has its canonical form in a metric frame since then, with $\boldsymbol{\pi}_\infty = (0, 0, 0, 1)^T$, $\mathrm{Q}^*_\infty \boldsymbol{\pi}_\infty = \mathbf{0}$.

$$\begin{pmatrix} I_{3\times 3} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This property holds in any frame as may be readily seen algebraically from the transformation properties of planes and dual quadrics: if $\mathbf{X}' = H\mathbf{X}$, then $\mathrm{Q}^*_\infty{}' = H\mathrm{Q}^*_\infty H^T$, $\boldsymbol{\pi}_\infty{}' = H^{-T}\boldsymbol{\pi}_\infty$, and

$$\mathrm{Q}^*_\infty{}'\boldsymbol{\pi}'_\infty = (\mathrm{H}\,\mathrm{Q}^*_\infty \cancel{\mathrm{H}^\top)\mathrm{H}^{-\top}}\boldsymbol{\pi}_\infty = \mathrm{H}\mathrm{Q}^*_\infty\boldsymbol{\pi}_\infty = \mathbf{0}.$$

$= 0.$

The Absolute Dual Quadric

- The angle between two planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ is given by

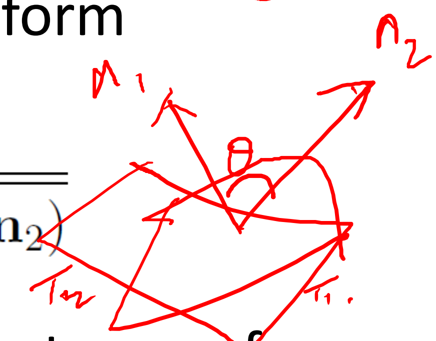
$$\cos\theta = \frac{\boldsymbol{\pi}_1^{\mathsf{T}} \mathrm{Q}_\infty^* \boldsymbol{\pi}_2}{\sqrt{\left(\boldsymbol{\pi}_1^{\mathsf{T}} \mathrm{Q}_\infty^* \boldsymbol{\pi}_1\right)\left(\boldsymbol{\pi}_2^{\mathsf{T}} \mathrm{Q}_\infty^* \boldsymbol{\pi}_2\right)}}.$$

$$\begin{pmatrix} I_{3\times 3} & 0 \\ 0 & 0 \end{pmatrix}$$

Proof:

Consider two planes with Euclidean coordinates $\boldsymbol{\pi}_1 = (\boldsymbol{n}_1^T, \cancel{d_1})^T$, $\boldsymbol{\pi}_2 = (\boldsymbol{n}_2^T, \cancel{d_2})^T$. In a Euclidean frame, Q_∞^* has the form

$$\mathrm{Q}_\infty^* = \begin{bmatrix} \mathrm{I} & \mathbf{0} \\ \mathbf{0}^{\mathsf{T}} & 0 \end{bmatrix}$$, and we get $$\cos\theta = \frac{\mathbf{n}_1^{\mathsf{T}}\mathbf{n}_2}{\sqrt{\left(\mathbf{n}_1^{\mathsf{T}}\mathbf{n}_1\right)\left(\mathbf{n}_2^{\mathsf{T}}\mathbf{n}_2\right)}}$$



which is the angle between the planes expressed in terms of a scalar product of their normals.

The Absolute Dual Quadric

Remarks:

$x' = HX$

If the planes and Q^*_∞ are projectively transformed,

π_1, π_2

$$\cos\theta = \frac{\boldsymbol{\pi}_1^\mathsf{T} Q^*_\infty \boldsymbol{\pi}_2}{\sqrt{\left(\boldsymbol{\pi}_1^\mathsf{T} Q^*_\infty \boldsymbol{\pi}_1\right)\left(\boldsymbol{\pi}_2^\mathsf{T} Q^*_\infty \boldsymbol{\pi}_2\right)}}.$$

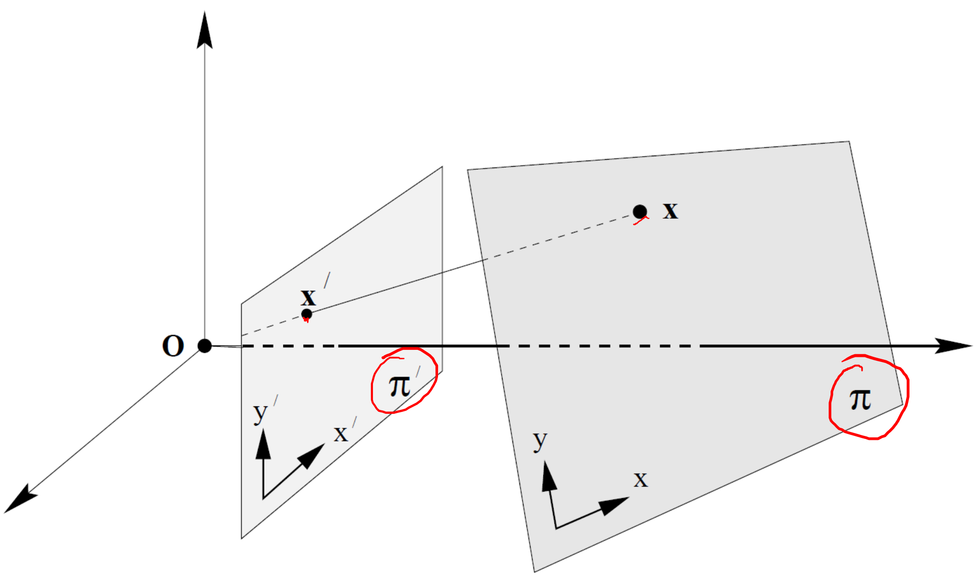
will still determine the angle between planes due to the (covariant) transformation properties of planes and dual quadrics.

Exercise: Prove it!

Planar Projective Transformations

We have seen in Lecture 1:

- Central projection maps points on one plane to points on another plane.
- And represented by a linear mapping of homogeneous coordinates $\mathbf{x}' = H\mathbf{x}$.

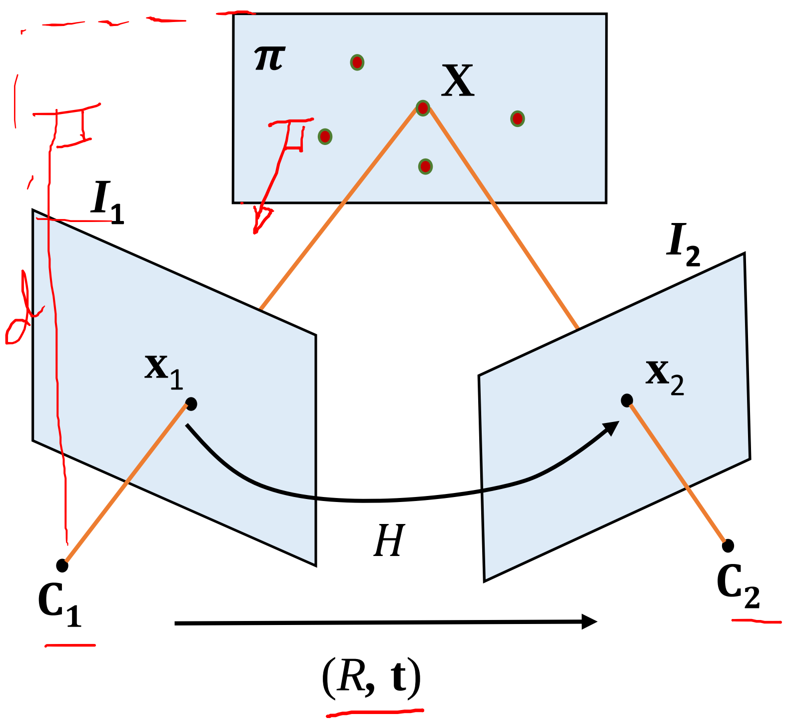


This is also known as Homography!

Image source: "Multiple View Geometry in Computer Vision", Richard Hartley and Andrew Zisserman

Existence of Projective Homography

1. Planar scene:



- $\mathbf{X}_1$ and $\mathbf{X}_2$ is the 3D point $\mathbf{X}$ expressed in $\mathbf{C}_1$ and $\mathbf{C}_2$ respectively:

$$\Rightarrow \mathbf{X}_2 = R\mathbf{X}_1 + \mathbf{t}.$$

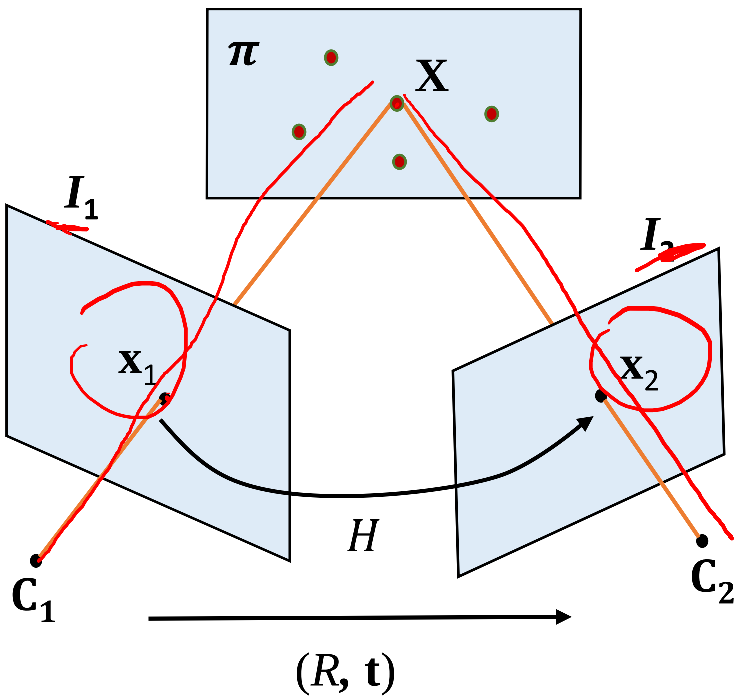
- $\mathbf{N} = [n_1, n_2, n_3]^T$ is the unit normal vector representing the plane $\boldsymbol{\pi}$ w.r.t $\mathbf{C}_1$, and d is the perpendicular distance from plane to $\mathbf{C}_1$:

$$\mathbf{N}^T\mathbf{X}_1 = n_1X + n_2Y + n_3Z = d,$$

$$\Rightarrow \frac{\mathbf{N}^T\mathbf{X}_1}{d} = 1, \ \forall\, \mathbf{X}_1 \in \boldsymbol{\pi}.$$

Existence of Projective Homography

1. Planar scene:



- Combining the two equations, we get

$$\mathbf{X}_2 = \left(R + \frac{\mathbf{t}\mathbf{N}^T}{d}\right)\mathbf{X}_1,$$

- Since $\lambda_1 \mathbf{x}_1 = \mathbf{X}_1$ and $\lambda_2 \mathbf{x}_2 = \mathbf{X}_2$, we get

$$\lambda \mathbf{x}_2 = \underbrace{\left(R + \frac{\mathbf{t}\mathbf{N}^T}{d}\right)}_{H} \mathbf{x}_1$$

Existence of Projective Homography

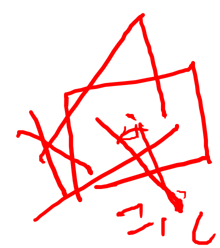
2. Plane at infinity: Scene is very far away from the camera, e.g. aerial images, i.e.

$$H = \left(R + \frac{\mathbf{t}\mathbf{N}^T}{d}\right) \Rightarrow H_\infty = \lim_{d \to \infty}\left(R + \frac{\mathbf{t}\mathbf{N}^T}{d}\right) = R.$$

$\rightarrow 0.$

This is the same as pure rotation, i.e. $\mathbf{t} = (0,0,0)^T$:

$$H = \left(R + \frac{\mathbf{t}\mathbf{N}^T}{d}\right) \quad \Rightarrow H = R.$$

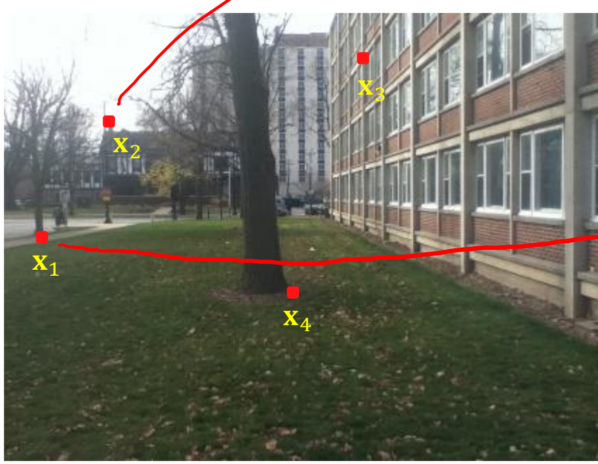


2D Homography

- **Given**: A set of points correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ between two images.

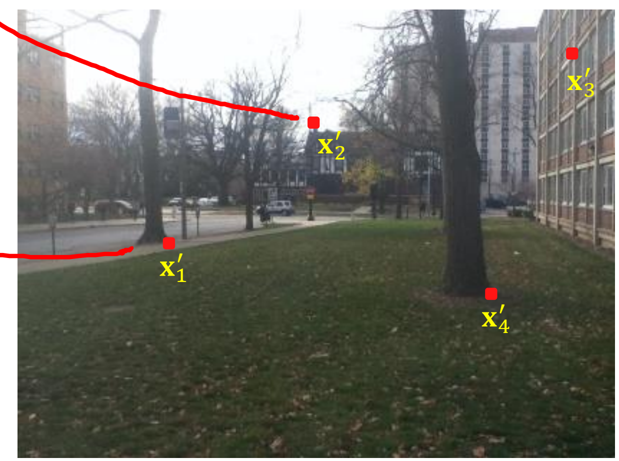
- **Compute**: The 2D Homography, *H* such that $H\mathbf{x}_i = \mathbf{x}'_i$ for each i.

3x3



$\mathbb{P}^2 \rightarrow \mathbb{P}^2$

Point correspondences on image planes undergo 2D Homography



Number of Measurements Required?

Question:

How many corresponding points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ are required to compute H?

Number of Measurements Required?

Answer:

- The number of degrees of freedom and number of constraints give a lower bound:

1. 8 degrees of freedom for H, i.e. 9 entries less 1 for up to scale.

   3×3

2. We will see that each point correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ gives 2 constraints.

   4pt 4×2 = 8

- As a consequence, it is necessary to specify four point correspondences in order to constrain H fully.

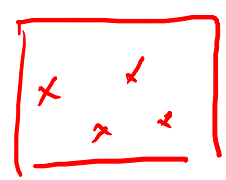
Approximate Solutions

- It will be seen that if exactly four correspondences are given, then an exact solution for the matrix H is possible.

8 dof

- This is the minimal solution, which is important for the number of RANSAC loops for robust estimation (details later).

- Since points are measured inexactly ("noise"), more than four correspondences are usually used to obtain a least-squares solution (details later).



Direct Linear Transformation (DLT) Algorithm

- We begin with a simple linear algorithm for determining H given a set of four point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$.

- Let us denote $H\mathbf{x}_i = \mathbf{x}'_i$ in terms of vector cross product:

$$\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \mathbf{0}, \text{ where } \mathtt{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix} \text{ and } \mathbf{x}'_i = (x'_i, y'_i, w'_i)^{\mathsf{T}}.$$

- The cross product may then be given explicitly as:

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

Direct Linear Transformation (DLT) Algorithm

- Since $\mathbf{h}^{j\mathsf{T}}\mathbf{x}_i = \mathbf{x}_i^{\mathsf{T}}\mathbf{h}^j$ for $j = 1, \ldots, 3$, the cross product can be written in a linear form:

Only first 2 rows are independent!

$$\left[\begin{array}{ccc} \mathbf{0}^{\mathsf{T}} & -w_i'\mathbf{x}_i^{\mathsf{T}} & y_i'\mathbf{x}_i^{\mathsf{T}} \\ w_i'\mathbf{x}_i^{\mathsf{T}} & \mathbf{0}^{\mathsf{T}} & -x_i'\mathbf{x}_i^{\mathsf{T}} \\ \cancel{-y_i'\mathbf{x}_i^{\mathsf{T}}} & \cancel{x_i'\mathbf{x}_i^{\mathsf{T}}} & \cancel{\mathbf{0}^{\mathsf{T}}} \end{array}\right] \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

$$\underbrace{\left[\begin{array}{ccc} \mathbf{0}^{\mathsf{T}} & -w_i'\mathbf{x}_i^{\mathsf{T}} & y_i'\mathbf{x}_i^{\mathsf{T}} \\ w_i'\mathbf{x}_i^{\mathsf{T}} & \mathbf{0}^{\mathsf{T}} & -x_i'\mathbf{x}_i^{\mathsf{T}} \end{array}\right]}_{2\times 9} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}\text{, or}$$

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

- The third row is obtained, up to scale, from the sum of x_i' times the first row and y_i times the second.

Direct Linear Transformation (DLT) Algorithm

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

- A_i is a 2 x 9 matrix, and $\mathbf{h}$ is a 9-vector made up of all elements in H, i.e.

$$\mathbf{h} = \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix}, \qquad \mathtt{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

- With h_i the i–th element of $\mathbf{h}$.

- Note that w_i is normally chosen as 1.

$x_i = (x_i, y_i, w_i)^T$

1

Direct Linear Transformation (DLT) Algorithm

9x1 vector

- $\mathbf{h}$ has 8 degrees of freedom and each point correspondence gives two constraints.
- A minimum of $\boxed{\text{4-point correspondences}}$ is needed to solve for h, i.e. $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, for $i \geq 4$.

$A_i h = 0$

2x9 9x1

- Stacking all equations together, we get:

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

9x1 vector.

- A is now a $2i \times 9$ matrix.

Least –Squares Solution

- In real image measurements, the point correspondences are corrupted with noise.

- An exact solution for $\mathrm{A}\mathbf{h} = 0$ does not exist!

$h = 0$ ✗

- Instead, we seek to minimize $\|\mathrm{A}\mathbf{h}\|$ over $\mathbf{h}$, subjected to the constraint of $\boxed{\|\mathbf{h}\| = 1}$.

- This is the least-squares solution of $\mathbf{h}$ and can obtained by taking the 9-vector right null-space of A.

Singular Value Decomposition (SVD)

- **Right null-space:** right singular vector that correspondences to the smallest singular value, i.e. σ_9 in the Singular Value Decomposition (SVD) of A, i.e. v_9,

$$\mathrm{svd}(\mathrm{A}) = \underbrace{[u_1, u_2, \dots u_9]}_{\text{Left singular vectors (2i x 9)}} \underbrace{\begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_9 \end{bmatrix}}_{\text{Singular values (9 x 9)}} \underbrace{[v_1, v_2, \dots v_9]^T}_{\text{Right singular vectors (9 x 9)}}$$

Singular Value Decomposition (SVD)

- In general, for a given $m \times n$ matrix A, where $m > n$ and $\mathrm{rank(A)} = r$, its Singular Value Decomposition is given by:

≡ noise free !

$Ax = 0$

$$\underbrace{\mathrm{A} = [u_1 \quad \dots \quad u_n]}_{\text{Left singular vectors (m x n)}} \underbrace{\begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \sigma_{n-r} & 0 \\ 0 & 0 & \dots & 0 & \sigma_n \end{bmatrix}}_{\text{Singular values (n x n)}} \underbrace{[v_1 \quad \dots \quad v_n]^T}_{\text{Right singular vectors (n x n)}} = \mathrm{U}\Sigma V^T$$

Left singular vectors (m x n)

Singular values (n x n)

$\sigma_1 > \sigma_2 > \sigma_3 > \cdots > \sigma_n$

Right singular vectors (n x n)

$A = u_i \sigma_i V_i^T$

$AV_i = u_i \sigma_i$

$\sigma_n = 0$

$\exists AV_i = 0$

- $\sigma_{n-r}, \dots, \sigma_n = 0$, i.e. $\mathrm{rank(A)} = r$ if A is NOT corrupted by noise and an exact solution for $\mathrm{A}\mathbf{h} = 0$ exists!

$h = V_i$

Singular Value Decomposition (SVD)

- If A is corrupted by noisy measurements, $\sigma_{n-r}, \ldots, \sigma_n \neq 0$.

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \qquad \sigma_1 > \sigma_2 > \cdots > \sigma_n$$

- Since U and V are orthogonal matrices, and Σ is a diagonal matrix, we have:

$$A = \mathrm{U}\Sigma V^T \Rightarrow AV = U\Sigma$$

$$Av_i = u_i \sigma_i$$

- $||Av_i||$ is minimized when $u_i\sigma_i$ is at its minimum, i.e. smallest singular value, i.e. σ_n.

Singular Value Decomposition (SVD)

- The solution of the problem:

$$\boxed{\underset{\mathbf{h}}{\operatorname{argmin}} \|\mathrm{A}\mathbf{h}\|, \quad \text{s.t.} \quad \|\mathbf{h}\| = 1}$$

is given by setting $\underline{\mathbf{h} = v_n}$. $= \sigma_n$

least singular value.

- We note that the constraint of $\|\mathbf{h}\| = 1$ is satisfied since $[v_1 \quad \cdots \quad v_n]^T$ an orthogonal matrix where the rows and columns are unit norm, respectively.

Direct Linear Transformation (DLT) Algorithm

<u>Objective</u>

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

$Ah = 0.$

<u>Algorithm</u>

(i) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$ compute A_i. Usually only two first rows needed.

(ii) Assemble n 2×9 matrices A_i into a single $2n \times 9$ matrix A.

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

(iv) Determine H from $\mathbf{h}$.

$3 \times 3.$

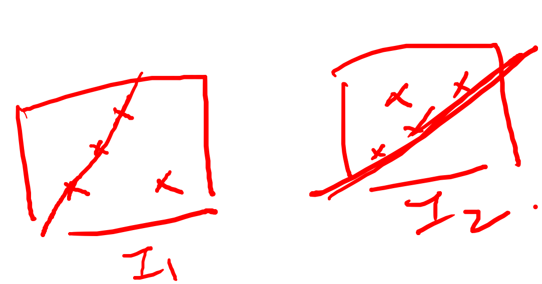
Slide credit: Marc Pollefeys

Homography: Degeneracy

8 dof.

8 constraints

$$Ah = 0$$



- Rank of matrix *A* drops below 8 if three of the minimum four points correspondences are collinear.

- In this case, we cannot solve for h, i.e. critical configuration or degeneracy.

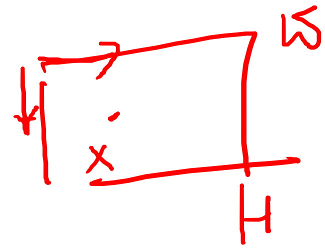
rank(A) < 8

- It is important to check that selected points are NOT in the critical configuration, i.e. collinear.

Importance of Normalization

Problem:

For a point $(x,y,w)^T = (100,100,1)^T$,



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

$$\begin{array}{ccccccccc} \sim 10^2 & \sim 10^2 & 1 & \sim 10^2 & \sim 10^2 & 1 & \sim 10^4 & \sim 10^4 & \sim 10^2 \end{array}$$

ill-conditioned.

$Ah = 0.$

Orders of magnitude difference -

This causes bad behavior in the SVD solution!

Solution: Data normalization

Importance of Normalization

Monte Carlo simulation:

- 5 points subjected to 0.1 pixel Gaussian noise are used to compute an identity homography matrix in 100 trials.
- Computed homography is used to transfer a further point into the second image in each trial.
- Results show that homographies computed from unnormalized data is less accurate.

A

$x_i \leftrightarrow x_i'$

H=I

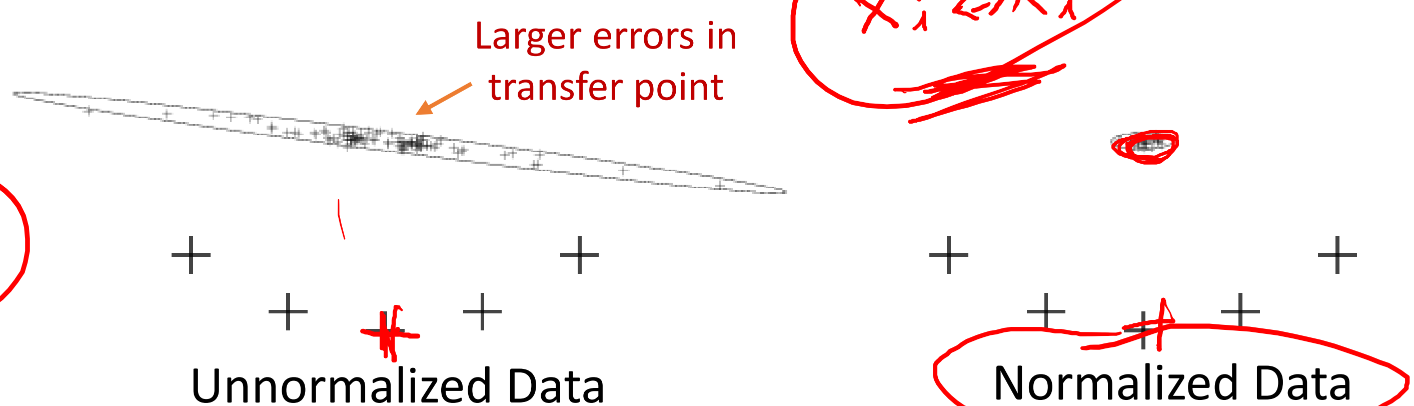


Image Source: R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision"

Data Normalization

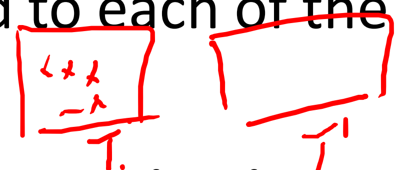
similarity

- Data normalization is carried out by a transformation of the points is as follows:

   i. Points are translated so that their centroid is at the origin. $(0,0,1)^T$

   ii. Points are then scaled so that the average distance from the origin is equal to sqrt(2).

   iii. Transformation is applied to each of the two images independently.



- This means that the average point is equal to $(1,1,1)^T$ after normalization
- ⇒ no magnitude difference in linear equation Ah=0.

Normalized DLT Algorithm

Data normalization is an essential step in the DLT algorithm.

It must not be considered optional!

Objective

Given n≥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) Normalize points $\tilde{x}_i = T_{norm} x_i, \tilde{x}_i' = T'_{norm} x_i'$

(ii) Apply DLT algorithm to $\tilde{x}_i \leftrightarrow \tilde{x}_i'$,

(iii) Denormalize solution $H = T'^{-1}_{norm} \tilde{H} T_{norm}$

Sim Transformation

$$\Rightarrow T_{norm} = \begin{bmatrix} s & 0 & -sc_x \\ 0 & s & -sc_y \\ 0 & 0 & 1 \end{bmatrix}$$

c: centroid of all data points

$$s = \frac{\sqrt{2}}{\bar{d}}$$

where $\bar{d}$: mean distance of all points from centroid.

Different Cost Functions: Algebraic Distance

- The DLT algorithm minimizes the norm $||\mathrm{A}\mathbf{h}||$, where $\boldsymbol{\epsilon} = \mathrm{A}\mathbf{h}$ is called the residual vector.
- Each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$ contributes a partial error vector $\boldsymbol{\epsilon}_i$ (2×1), where the norm is called the algebraic distance:

$$d_{\text{alg}}(\mathbf{x}_i', \mathrm{H}\mathbf{x}_i)^2 = ||\boldsymbol{\epsilon}_i||^2 = \left\| \begin{bmatrix} \mathbf{0}^\mathsf{T} & -w_i'\mathbf{x}_i^\mathsf{T} & y_i'\mathbf{x}_i^\mathsf{T} \\ w_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x_i'\mathbf{x}_i^\mathsf{T} \end{bmatrix} \mathbf{h} \right\|^2 .$$

- Given a set of correspondences, the total algebraic error for the complete set is:

$$\sum_i d_{\text{alg}}(\mathbf{x}_i', \mathrm{H}\mathbf{x}_i)^2 = \sum_i ||\boldsymbol{\epsilon}_i||^2 = ||\mathrm{A}\mathbf{h}||^2 = ||\boldsymbol{\epsilon}||^2 .$$

Different Cost Functions: Algebraic Distance

$Ah = 0$

- The disadvantage is that the quantity that is minimized is not meaningful geometrically nor statistically.

- Nevertheless, it is a linear solution (and thus a unique), and is computationally inexpensive.

- Often solutions based on algebraic distance are used as a starting point for a non-linear minimization of a geometric cost function (details later).

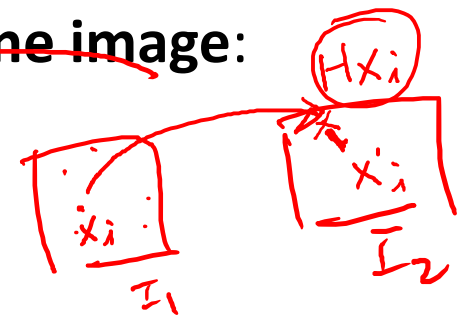
- The non-linear minimization gives the solution a final "polish".

Different Cost Functions: Geometric Distance

- The geometric distance in the image refers to the difference between the measured and estimated image coordinates.

- Let's first consider the transfer error in **one image**:

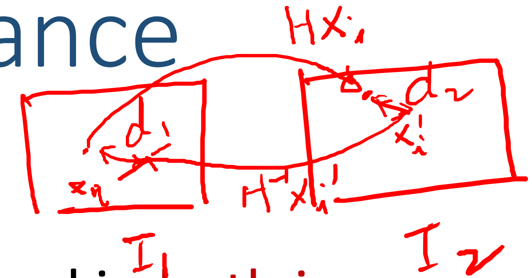
$$\arg\min_{H} \sum_i d(\mathbf{x}'_i, \mathrm{H}\mathbf{x}_i)^2.$$



- This is the Euclidean image distance in the second image between the measured point $\mathbf{x}'_i$ and the corresponding point $H\mathbf{x}_i$ transferred from the first image.

- The error is minimized over the estimated homography H.

Different Cost Functions: Geometric Distance



Symmetric Transfer Error

- More preferable that errors be minimized in both images, and not solely in the one.

- Symmetric transfer error considers the forward (H) and backward (H^{-1}) transformation:

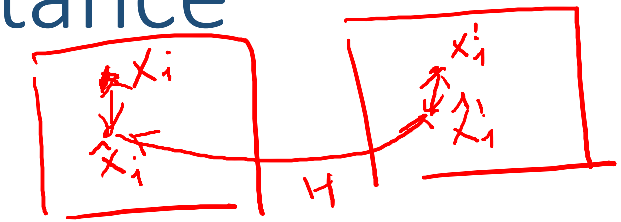
$$\underset{H}{\text{argmin}} \quad \sum_i d(\mathbf{x}_i, \mathrm{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathrm{H}\mathbf{x}_i)^2 .$$

- The first term is the transfer error in the first image, and the second term is the transfer error in the second image.

- Again the error is minimized over the estimated homography H.

Different Cost Functions: Geometric Distance

Reprojection Error



- We are seeking a homography $\widehat{H}$ and pairs of perfectly matched points $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_i'$ that minimize the total error function:

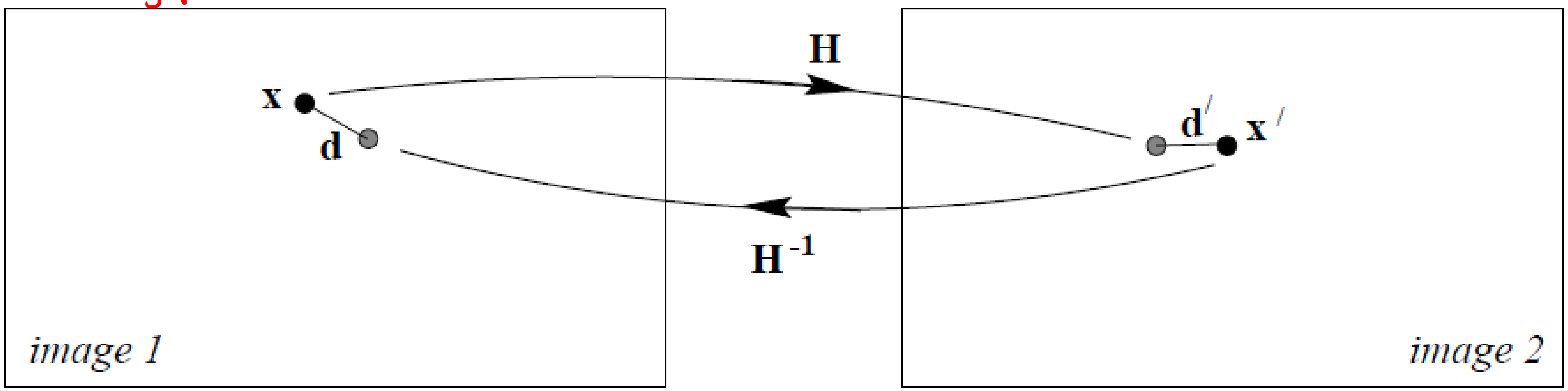
argmin
$H, \hat{x}_i$

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \quad \text{subject to } \hat{\mathbf{x}}'_i = \hat{\mathrm{H}}\hat{\mathbf{x}}_i \;\; \forall i.$$

- Minimizing this cost function involves determining both $\widehat{H}$ *and* a set of subsidiary correspondences $\{\hat{\mathbf{x}}_i\}$ and $\{\hat{\mathbf{x}}_i'\}$.

Symmetric Transfer Error (upper) Vs Reprojection Error

Symmetric



Reprojection

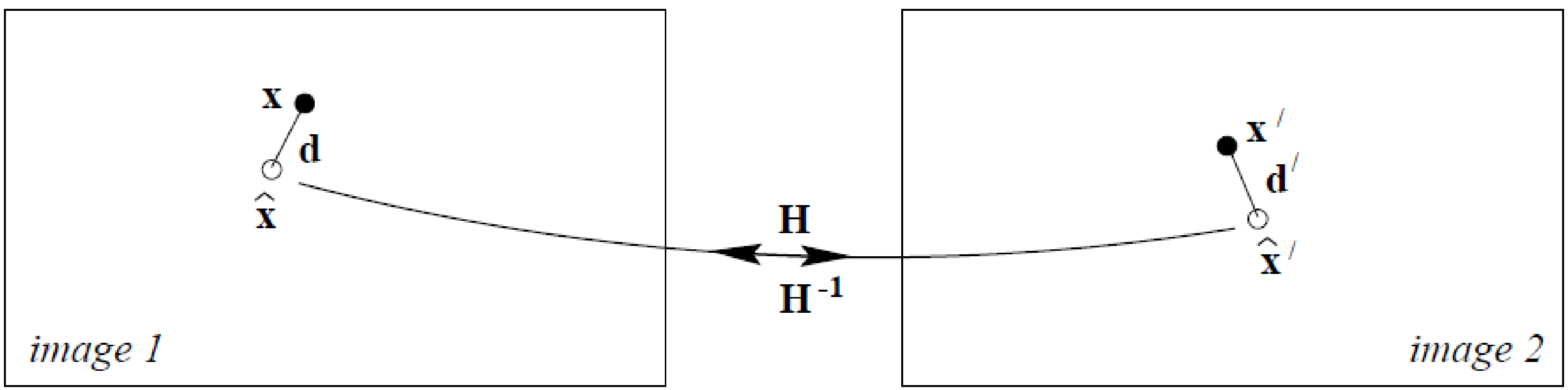


Image Source: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Different Cost Functions: Sampson Error

- The minimization of both homography H and points $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i$ makes the reprojection error accurate but also computationally complex.

$\underset{H,\ \hat{x}_i}{\arg\min}$

- Its complexity contrasts with the simplicity of minimizing the algebraic error.

- The Sampson error lies between the algebraic and geometric cost functions in terms of complexity, but gives a close approximation to reprojection error.

Different Cost Functions: Sampson Error

x_i x_i'

- Let $C_H(X) = \mathbf{0}$ denote the cost function $A\mathbf{h} = \mathbf{0}$ that is satisfied by the point $\mathbf{X} = (x, y, x', y')^T$ for a given homography H.

4-vector.

- We further denote $\hat{\mathbf{X}}$ as the desired point so that $C_H(\hat{\mathbf{X}}) = \mathbf{0}$ where $\delta_{\mathbf{X}} = \hat{\mathbf{X}} - \mathbf{X}$, and now the cost function may be approximated by a Taylor expansion:

$\hat{X} = \delta_X + X$

$$\mathcal{C}_{\mathrm{H}}(\mathbf{X} + \boldsymbol{\delta}_{\mathbf{X}}) = \mathcal{C}_{\mathrm{H}}(\mathbf{X}) + (\partial \mathcal{C}_{\mathrm{H}} / \partial \mathbf{X}) \boldsymbol{\delta}_{\mathbf{X}} = \mathbf{0}$$

Different Cost Functions: Sampson Error

- The approximated cost function can be rewritten as:

$$\mathrm{J}\boldsymbol{\delta}_{\mathbf{X}} = -\boldsymbol{\epsilon}$$

$\hat{X}-X$

$Ax=b$

where J is the partial-derivative matrix, and $\boldsymbol{\epsilon}$ is the cost $\mathrm{C}_H(\boldsymbol{X})$ associated with $\mathbf{X}$.

- The minimization problem now becomes: Find the vector $\delta_{\mathbf{X}}$ that minimizes $||\delta_{\mathbf{X}}||^2$ subject to $\boxed{\mathrm{J}\boldsymbol{\delta}_{\mathbf{X}} = -\boldsymbol{\epsilon}}$.

Different Cost Functions: Sampson Error

- Now $J\boldsymbol{\delta}_{\mathbf{X}} = -\boldsymbol{\epsilon}$ can be solved using the right pseudo inverse as:

$$\boldsymbol{\delta}_{\mathbf{X}} = -J^{\mathsf{T}}(JJ^{\mathsf{T}})^{-1}\boldsymbol{\epsilon},$$

right pseudo inverse.

- And the Sampson error is defined by the norm:

argmin

$$\|\boldsymbol{\delta}_{\mathbf{X}}\|^2 = \boldsymbol{\delta}_{\mathbf{X}}^{\mathsf{T}}\boldsymbol{\delta}_{\mathbf{X}} = \boldsymbol{\epsilon}^{\mathsf{T}}(JJ^{\mathsf{T}})^{-1}\boldsymbol{\epsilon}.$$

Different Cost Functions: Sampson Error

- For the 2D homography estimation problem, $\mathbf{X} = (x, y, x', y')^T$ where the measurements are $\mathbf{x} = (x, y, 1)^T$ and $\mathbf{x}' = (x', y', 1)^T$.

- And $\boldsymbol{\epsilon} = \mathrm{C}_H(\boldsymbol{X})$ is the algebraic error vector $\mathrm{A}_i\mathbf{h}$ (a 2-vector).

- $\mathrm{J} = \partial \mathcal{C}_{\mathrm{H}}(\mathbf{X})/\partial \mathbf{X}$ is a 2 x 4 matrix, where

$$J_{11} = \partial(-w_i' \mathbf{x}_i^{\mathsf{T}} \mathbf{h}^2 + y_i' \mathbf{x}_i^{\mathsf{T}} \mathbf{h}^3)/\partial x = -w_i' h_{21} + y_i' h_{31}.$$

Exercise: Derive the full expression of $\|\delta_{\mathbf{X}}\|^2$!

Iterative Minimization

- The Geometric and Sampson errors are usually minimized as the squared Mahalanobis distance :

$$\|\mathbf{X} - f(\mathbf{P})\|^2_\Sigma = (\mathbf{X} - f(\mathbf{P}))^\mathsf{T}\Sigma^{-1}(\mathbf{X} - f(\mathbf{P})) \ ,$$

$$\Rightarrow \quad \underset{\mathbf{P}}{\operatorname{argmin}} \ \|\mathbf{X} - f(\mathbf{P})\|^2_\Sigma \ ,$$

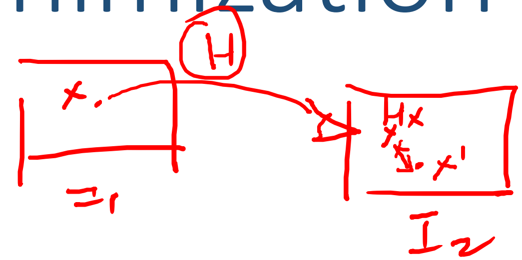
$\Downarrow$ Continuous.

where

- ➢ $\mathbf{X} \in \mathbb{R}^N$ is the measurement vector with covariance matrix Σ. $\Downarrow$ I
- ➢ $\mathbf{P} \in \mathbb{R}^M$ is the set of parameters to be optimized.
- ➢ A mapping function $f : \mathbb{R}^M \to \mathbb{R}^N$.

- This is an unconstrained continuous optimization that can be solved with solvers such as Gauss-Newton or Levenberg-Marquardt (details in Lecture 9).

Iterative Minimization



Error in one image:

- Measurement vector $\mathbf{X}$ is made up of the $2n$ inhomogeneous points $\mathbf{x}'_i$.
- Set of parameters to be optimized $\mathbf{P}$ is set as $\mathbf{h}$.
- Mapping function f is defined by:

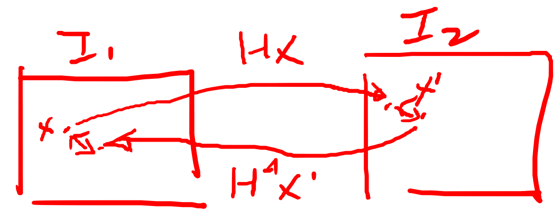
$$f : \mathbf{h} \mapsto (\mathrm{H}\mathbf{x}_1, \mathrm{H}\mathbf{x}_2, \ldots, \mathrm{H}\mathbf{x}_n)$$

  where the coordinates of points $\mathbf{x}_i$ in the first image is taken as a fixed input.

- We now find that $\|\mathbf{X} - f(\mathbf{h})\|^2$ becomes $\sum_i d(\mathbf{x}'_i, \mathrm{H}\mathbf{x}_i)^2$.

Iterative Minimization

Symmetric Transfer Error:



- Measurement vector $\mathbf{X}$ is a 4-vector made up of the inhomogeneous coordinates of the points $\mathbf{x}_i$ and $\mathbf{x}_i'$.

- Set of parameters to be optimized $\mathbf{P}$ is set as $\mathbf{h}$.

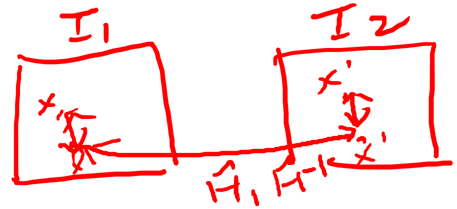
- Mapping function f is defined by:

$$f : \mathbf{h} \mapsto (\mathrm{H}^{-1}\mathbf{x}_1', \ldots, \mathrm{H}^{-1}\mathbf{x}_n', \mathrm{H}\mathbf{x}_1, \ldots, \mathrm{H}\mathbf{x}_n).$$

- We now find that $\|\mathbf{X} - f(\mathbf{h})\|^2$ becomes $\sum_i d(\mathbf{x}_i, \mathrm{H}^{-1}\mathbf{x}_i')^2 + d(\mathbf{x}_i', \mathrm{H}\mathbf{x}_i)^2$

Iterative Minimization

Reprojection Error:



- Measurement vector contains the inhomogeneous coordinates of all the points $\mathbf{x}_i$ and $\mathbf{x}'_i$.
- Set of parameters to be optimized is $\mathbf{P} = (\mathbf{h}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$.
- Mapping function f is defined by:

$$f : (\mathbf{h}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n) \mapsto (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}'_1, \ldots, \hat{\mathbf{x}}_n, \hat{\mathbf{x}}'_n) \text{, where } \hat{\mathbf{x}}'_i = \hat{\mathrm{H}}\hat{\mathbf{x}}_i.$$

- We can verify that $\|\mathbf{X} - f(\mathbf{h})\|^2$ becomes:

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \quad \text{subject to } \hat{\mathbf{x}}'_i = \hat{\mathrm{H}}\hat{\mathbf{x}}_i \quad \forall i$$

with $\mathbf{X}$ as a $4n$-vector.

Iterative Minimization

Sampson Approximation:

$x_i \,,\; x_i'$

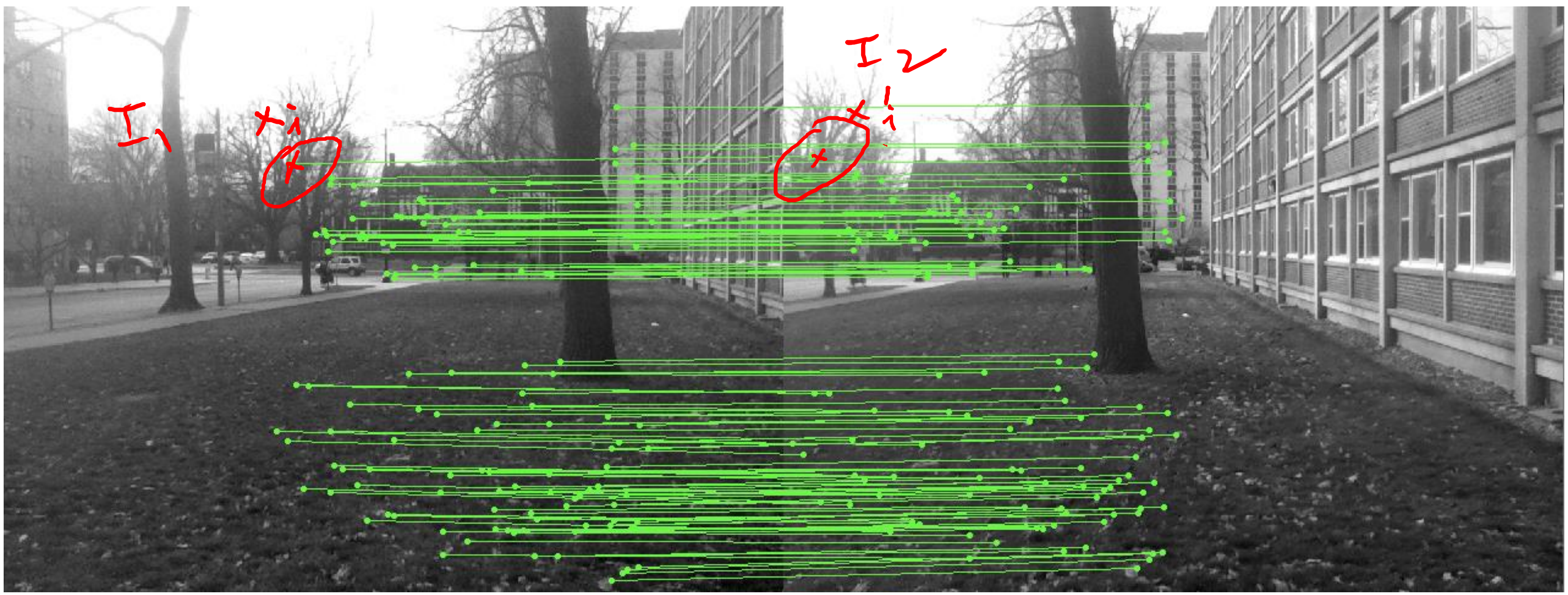
- Measurement vector $\mathbf{X} = (x, y, x', y')^T$.
- Set of parameters to be optimized $\mathbf{P}$ is set as $\mathbf{h}$.
- Here, we directly set $\mathbf{X} - f(\mathbf{h}) = \delta_{\mathbf{X}}$, and $\|\mathbf{X} - f(\mathbf{h})\|^2$ gives us the Sampson error:

$\delta_X = \hat{X} - X$

$$\|\boldsymbol{\delta}_{\mathbf{X}}\|^2 = \boldsymbol{\delta}_{\mathbf{X}}^{\mathsf{T}} \boldsymbol{\delta}_{\mathbf{X}} = \boldsymbol{\epsilon}^{\mathsf{T}} (\mathtt{J}\mathtt{J}^{\mathsf{T}})^{-1} \boldsymbol{\epsilon}.$$

Random Sample Consensus: RANSAC

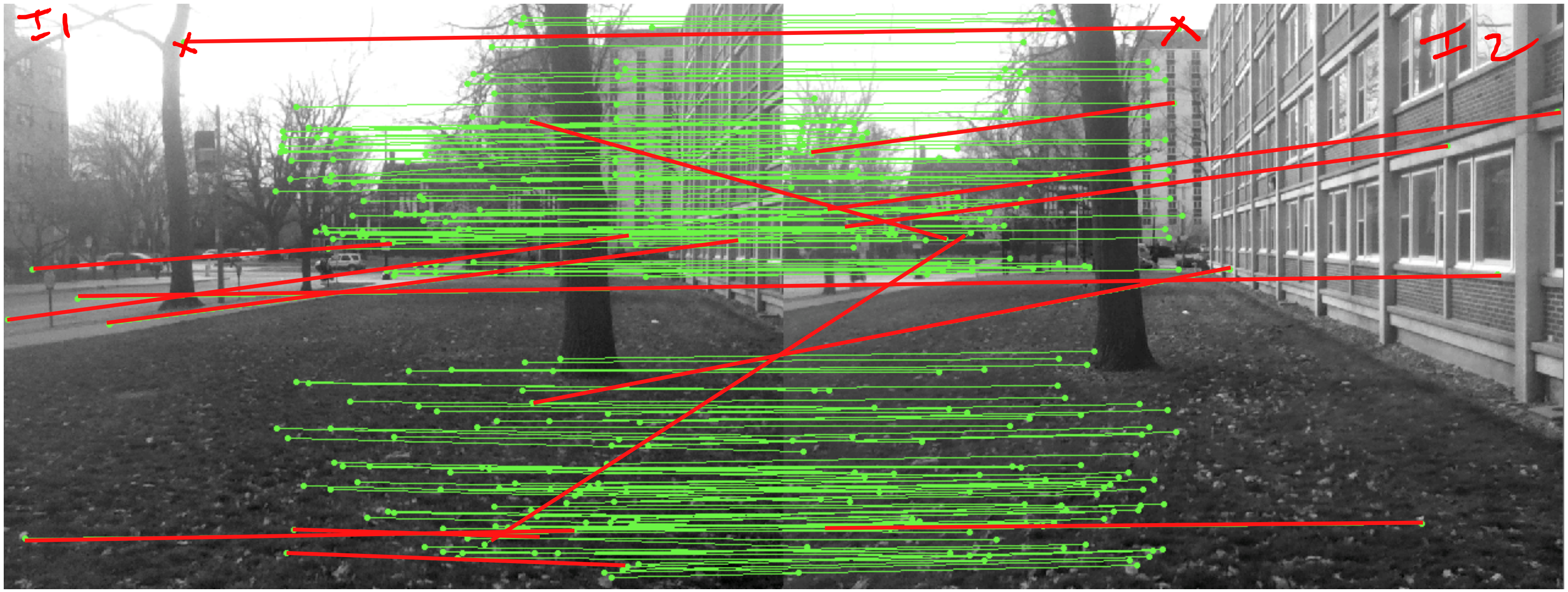
- Up to this point, we have assumed a set of correspondences with only ~~measurement noise~~.
  inlier



Random Sample Consensus: RANSAC

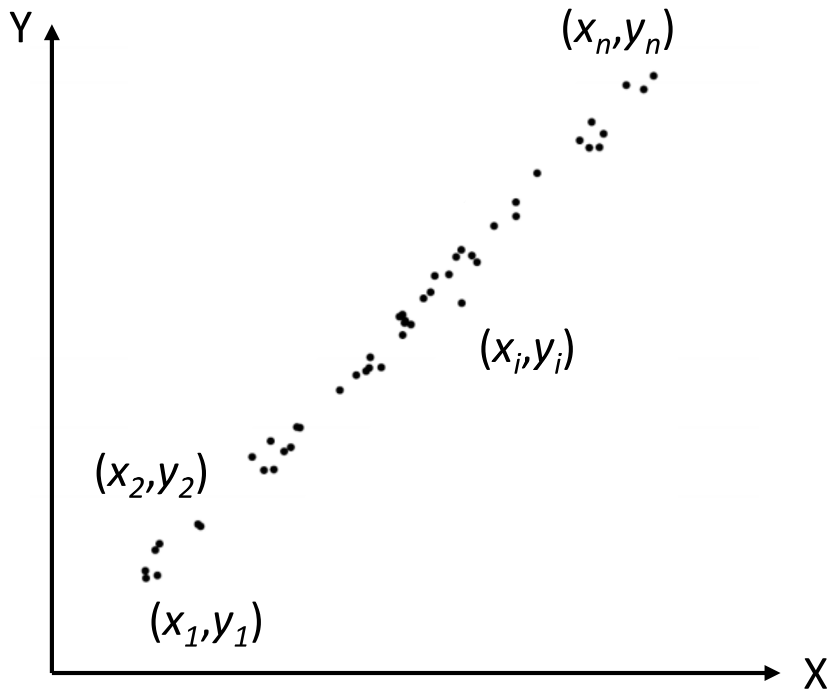
SIFT or ORB

- In reality, keypoint matching gives us many outliers.
- Outliers can severely disturb the least-squares estimation, and should be removed.



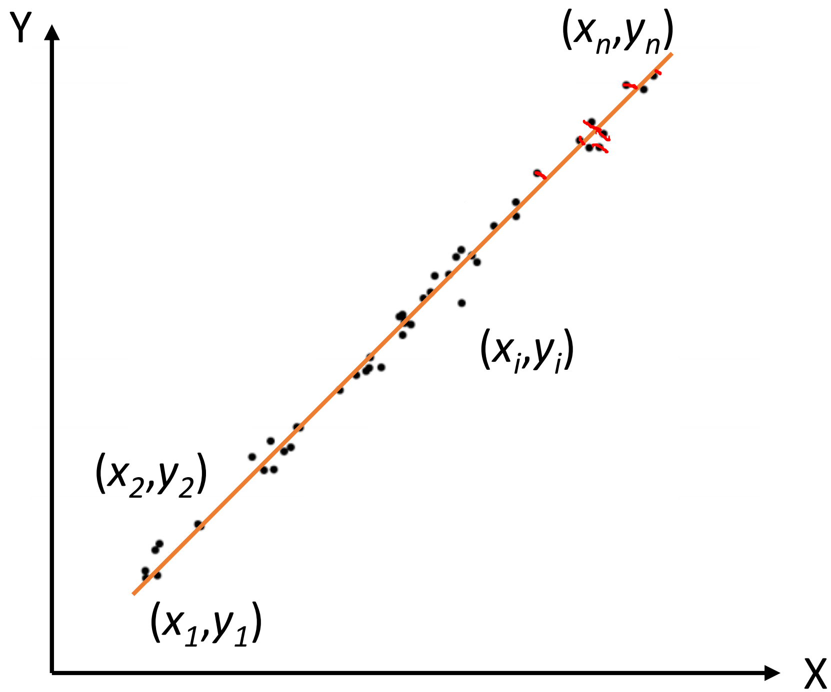
RANSAC: Line Fitting Example

- **Given**: n data points (x_i,y_i), for $i = 1, \dots, n$
- **Find**: Best fit line, i.e. two parameters (m,c) from the line equation $y_i = mx_i + c$, for $i = 1, \dots, n$



RANSAC: Line Fitting Example

- **Given**: n data points (x_i,y_i), for $i = 1, \ldots, n$
- **Find**: Best fit line, i.e. two parameters (m,c) from the line equation $y_i = mx_i + c,$ for $i = 1, \ldots, n$

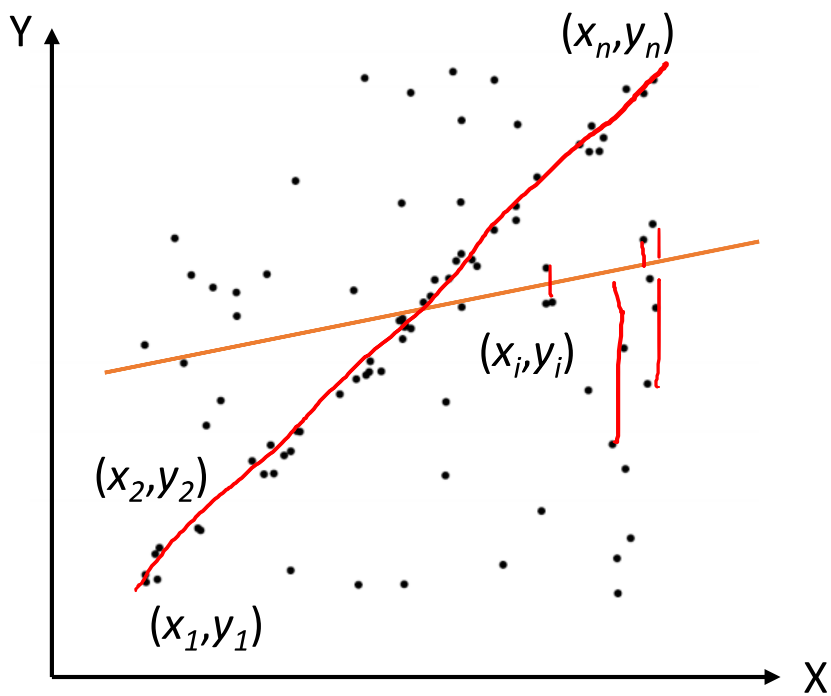


Least-squares solution:

$$\underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{n} \|y_i - (mx_i + c)\|^2$$

RANSAC: Line Fitting Example

- **Given**: n data points (x_i,y_i), for $i = 1, \dots, n$
- **Find**: Best fit line, i.e. two parameters (m,c) from the line equation $y_i = mx_i + c,$ for $i = 1, \dots, n$



Least-squares solution:

$$\underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{n} \| y_i - (mx_i + c) \|^2$$

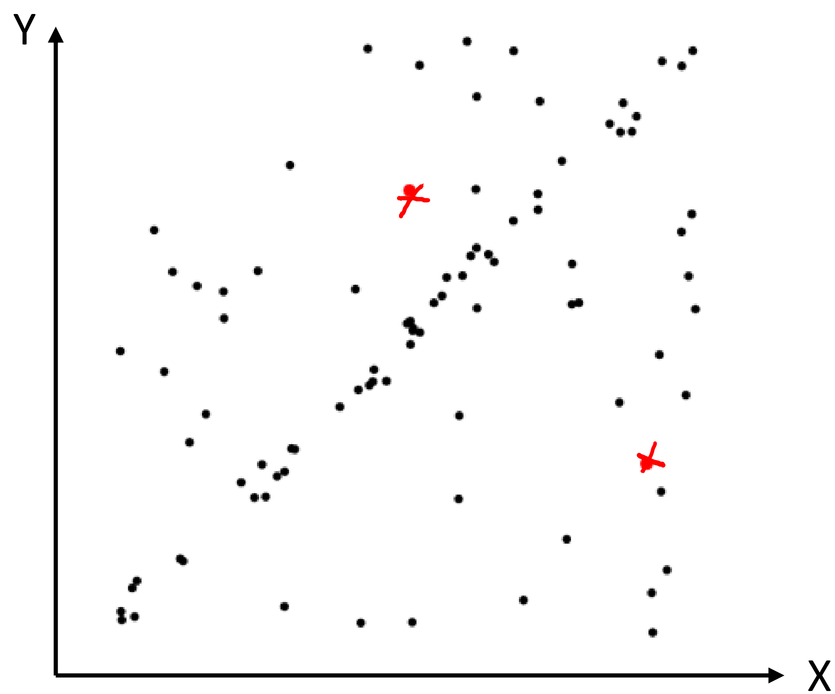
Least-squares fails when there's outliers!!!

RANSAC: Line Fitting Example

RANSAC Steps:

1. Randomly select minimal subset of points, i.e. 2 points

$y = mx + c$



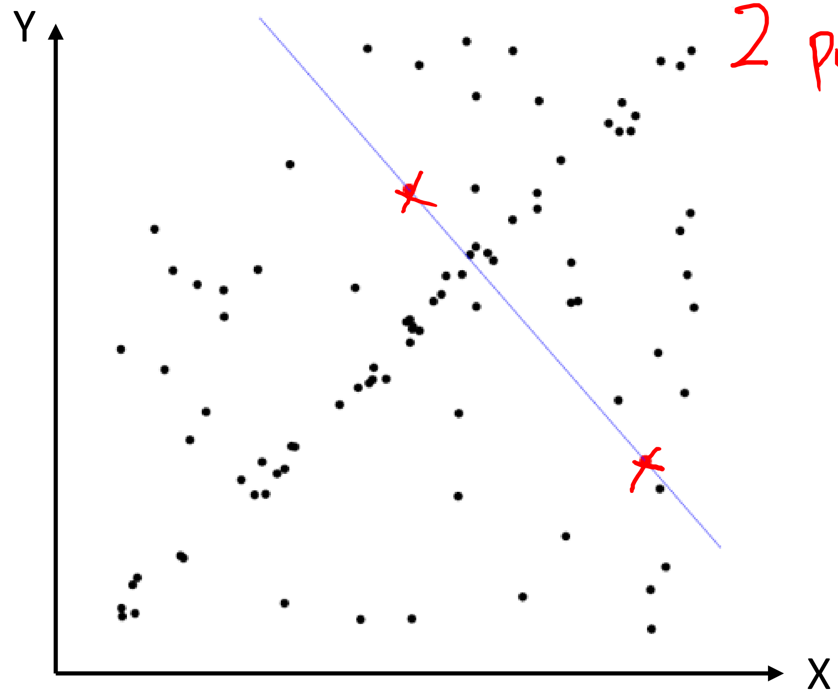
RANSAC: Line Fitting Example

RANSAC Steps:

2. Hypothesize a model

$y = mx + c$

hypothesis

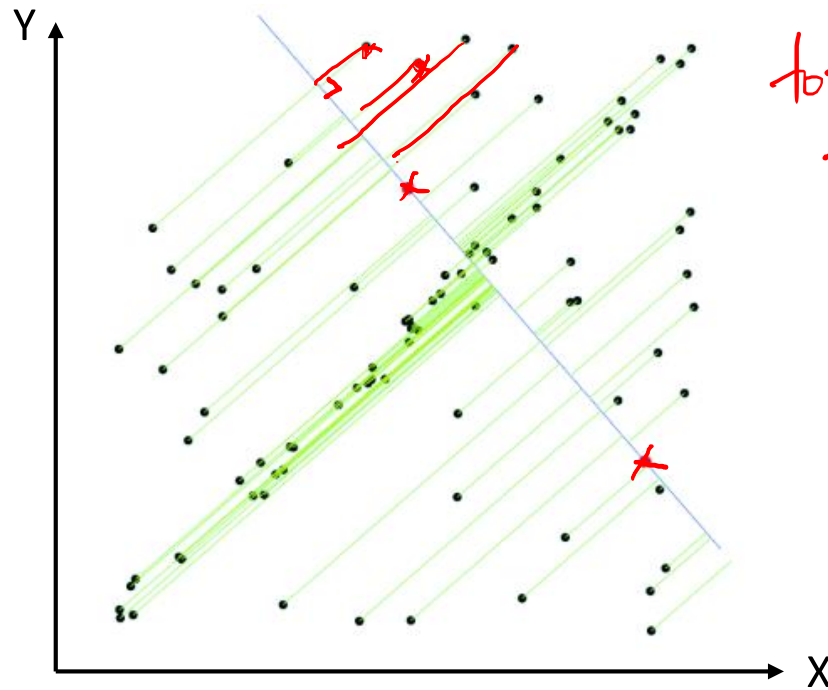


2 point

RANSAC: Line Fitting Example

RANSAC Steps:

3. Compute error function, i.e. shortest point to line distance

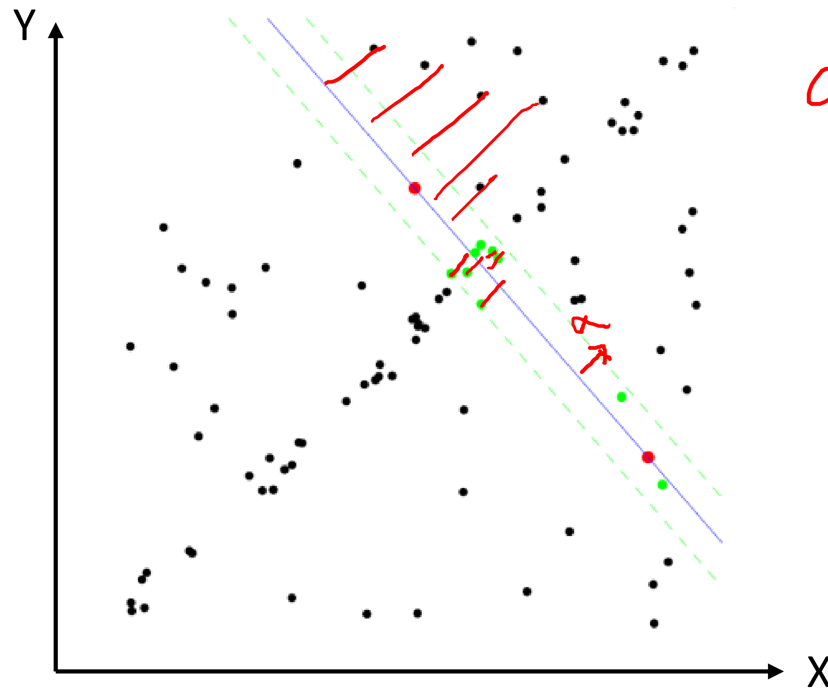


total error
$= \sum_i d_\perp$

RANSAC: Line Fitting Example

RANSAC Steps:

4. Select points consistent with model



$d_i \leq \alpha$

RANSAC: Line Fitting Example

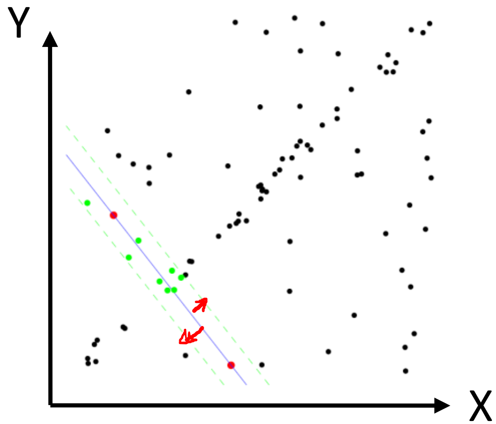
RANSAC Steps:

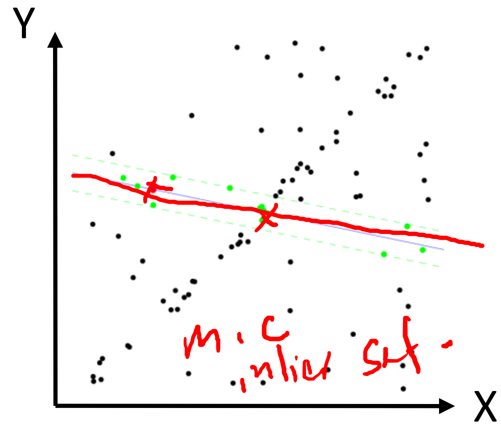
M iteration

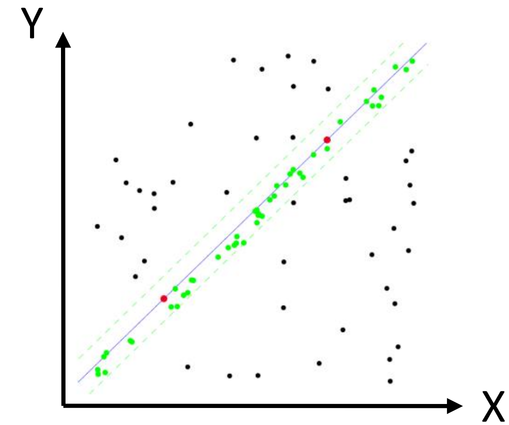
5. Repeat hypothesize-and-verify loop

iteration =1

iteration=2

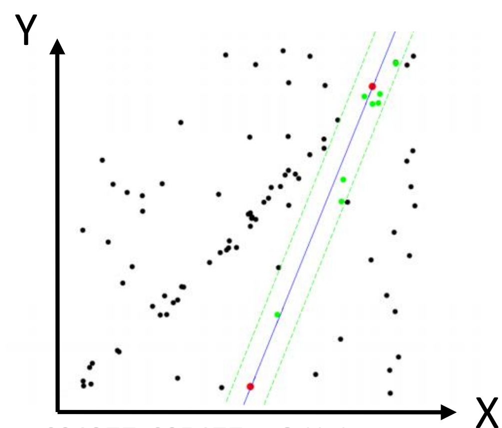






m, c

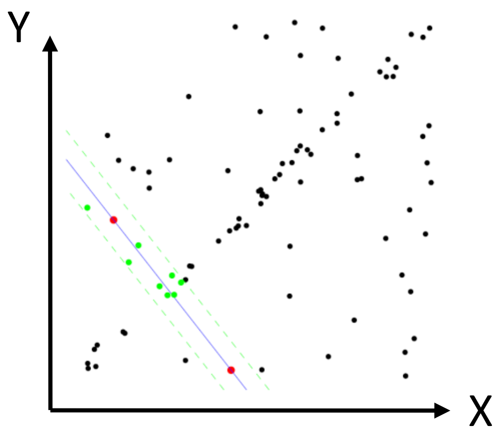
inlier set.

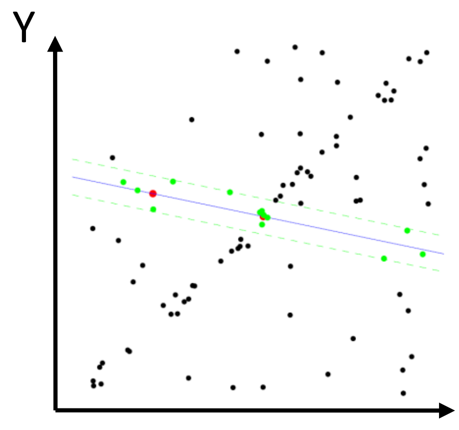


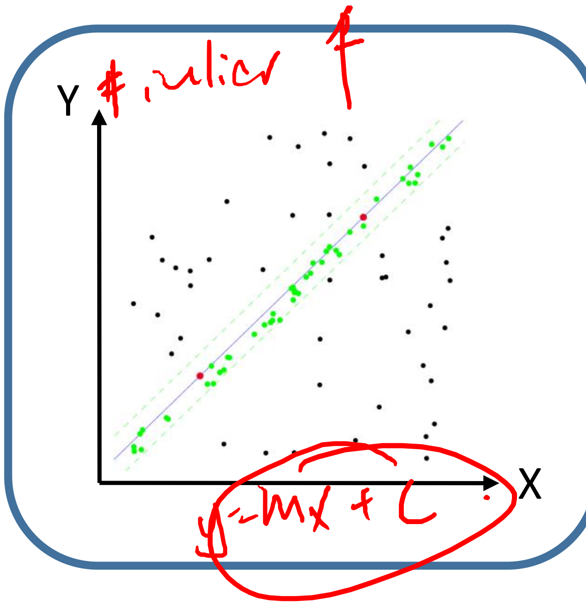
RANSAC: Line Fitting Example

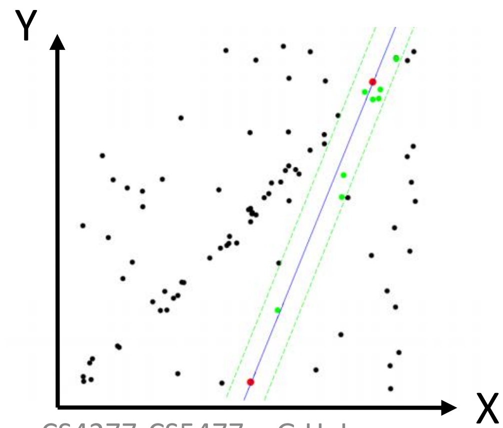
RANSAC Steps:

6. Select the hypothesis with the highest number of consistent points, i.e. inliers.









RANSAC Algorithm

Objective y=mx+c

Robust fit of a model to a data set S which contains outliers.

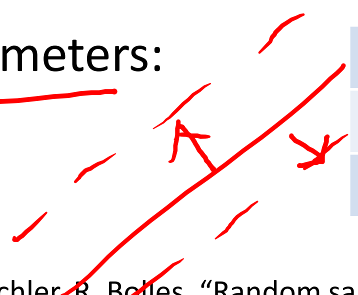
Algorithm 2-pt.

i. Randomly select a sample of s data points from S and instantiate the model from this subset.

ii. Determine the set of data points S_i which are within a distance threshold t of the model. The set S_i is the consensus set of the sample and defines the inliers of S.

iii. After N trials, select the largest consensus set S_i. The model is re-estimated using all the points in the subset S_i.

Three parameters:



Number of points	s
Distance threshold	t
Number of Samples	N

M. Fischler, R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", Communications ACM, 1981.

Choosing the Parameters

- Number of points, s: minimum Solution
  - Typically minimum number needed to fit the model.
  - e.g. 2 for line and 4 for homography.
- Distance threshold, t:
  - Usually chosen empirically.
  - But can be set as t^2=3.84σ^2 if the measurement error, i.e. zero-mean Gaussian noise with std. dev. Σ is known.
- Number of samples, N:

n points

2-pt.

- Exhaustive search of all sample is often unnecessary and infeasible.

$^nC_2$

Choosing the Parameters

- Number of samples, N

Probability that algorithm never selects a set of s points which all are inliers:

N sets of 2-pts

Probability that all s points are inliers

$$1 - p = \underbrace{(1 - w^s)}_{\text{probability that at least one of the } s \text{ points is an outlier}}{}^N$$

$$\Longrightarrow N = \frac{\log(1-p)}{\log(1-w^s)}$$

p: probability that at least one of the random samples of s points is free from outliers.

99%

w : probability that any selected point is an inlier.

Choosing the Parameters

- Number of samples, *N*:

$$N = \frac{\log(1 - p)}{\log(1 - w^s)}$$

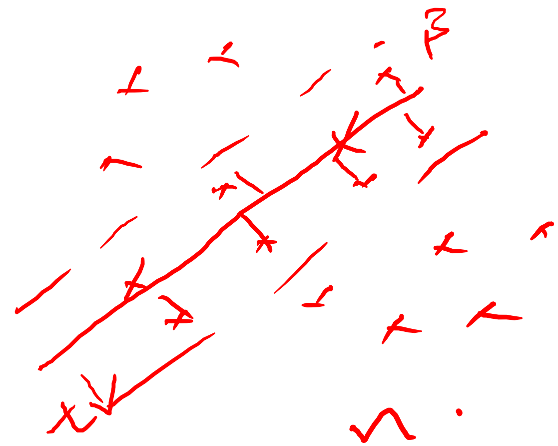
Sample size	Proportion of outliers $\epsilon = 1 - w$						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Table gives examples of N for $p = 0.99$ for a given s and ϵ.

Source: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Choosing *N* Adaptively

- Often *w* is unknown, we can choose the worst case, i.e. 50%.
- *w* can also be decided adaptively:



Adaptive RANSAC Algorithm

N=∞ , sample_count =0

while N > sample_count Repeat

1. Choose a sample and count #inliers
2. Set $w = \frac{\#\text{inliers}}{\#\text{points}}$ $\frac{\beta}{n}$.
3. $N = \frac{\log(1-p)}{\log(1-w^s)}$ with $p=0.99$
4. Increment sample_count by 1

Terminate

Robust 2D Homography Computation

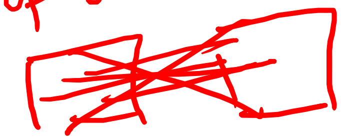
Objective

Compute the 2D homography between two images. in presence of outliers.

Algorithm

SIFT or ORB.

i. Interest points: Compute keypoints in each image.

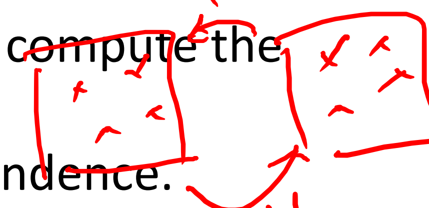


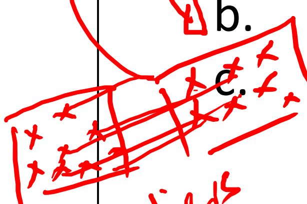
ii. Putative correspondences: Match keypoints using descriptors.

iii. RANSAC robust estimation: Repeat for *N* samples, where *N* is determined adaptively:

S minimal Sol^n.

- a. Select a random sample of 4 correspondences and compute the homography, H. hypothesize model
- b. Calculate the distance *d* for each putative correspondence.
- c. Compute the number of inliers consistent with H by the number of correspondences for which $d < t$ $d_i < t$





inliers

Choose the H with the largest number of inliers.

iv. Optimal estimation: re-estimate H from all correspondences classified as inliers.

Summary

- We have looked at how to:

1. Describe the plane at infinity and its invariance under affine transformation.
2. Describe the absolute conic (and its absolute dual quadrics) and its invariance under similarity transformation.
3. Explain the difference between the algebraic, geometric and Sampson errors, and apply them on homography estimation.
4. Use the RANSAC algorithm for robust estimation.