NUS National University of Singapore | School of Computing

CS4277 / CS5477
3D Computer Vision

Lecture 3:
Rigid Body Motion and Robust Homography Estimation

Assoc. Prof. Lee Gim Hee

AY 2022/23

Semester 2

Course Schedule

Week	Date	Topic	Assignments
1	11 Jan	2D and 1D projective geometry	**Assignment 0:** Getting started with Python (Ungraded)
2	18 Jan	3D projective geometry, Circular points and Absolute conic	
3	25 Jan	Rigid body motion and Robust homography estimation	
4	01 Feb	Camera models and calibration	**Assignment 1:** Metric rectification and robust homography (10%) **Due:** 2359hrs, 07 Feb
5	08 Feb	Single view metrology	**Assignment 2**: Affine 3D measurement from vanishing line and point (10%) **Due:** 2359hrs, 14 Feb
6	15 Feb	The Fundamental and Essential matrices	
-	22 Feb	Semester Break	No lecture
7	**01 Mar**	**Mid-term Quiz (20%)**	**In-person Quiz (LT 15, 1900hrs – 2000hrs)**
8	08 Mar	Absolute pose estimation from points or lines	
9	15 Mar	Three-view geometry from points and/or lines	
10	22 Mar	Structure-from-Motion (SfM) and bundle adjustment	**Assignment 3:** SfM and Bundle adjustment (10%) **Due:** 2359hrs, 28 Mar
11	29 Mar	Two-view and multi-view stereo	**Assignment 4:** Dense 3D model from multi-view stereo (10%) **Due:** 2359hrs, 04 Apr
12	05 Apr	3D Point Cloud Processing	
13	12 Apr	Neural Field Representations	

Final Exam: 03 MAY 2023

Learning Outcomes

- Students should be able to:

1. Explain the concepts of SE(3) group and use it to describe rigid body motions in the 3D space.
2. Show the existence of homography.
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 2, 3 and 4.

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

Three-Dimensional Euclidean Space

- We will use $\mathbb{E}^3$ to denote the familiar three-dimensional Euclidean space.
- Every point $p \in \mathbb{E}^3$ can be identified with a point in $\mathbb{R}^3$ with three Cartesian coordinates:

$$\boldsymbol{X} \doteq [X_1, X_2, X_3]^T = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \in \mathbb{R}^3.$$

- Through such assignment, we established a one-to-one correspondence between $\mathbb{E}^3$ and $\mathbb{R}^3$.

Three-Dimensional Euclidean Space

Definition (Vector).

- Bound vector: If p has coordinates $\boldsymbol{X}$ and q has coordinates $\boldsymbol{Y}$, then v has coordinates:

$$v \doteq \boldsymbol{Y} - \boldsymbol{X} \quad \in \mathbb{R}^3.$$

- Free vector: Two pairs of points $(p\,, q)$ and (p', q') with coordinates satisfying $\boldsymbol{Y} - \boldsymbol{X} = \boldsymbol{Y}' - \boldsymbol{X}'$ define the same free vector.

- Free vector does not depend on its base point.

Dot and Cross Products in $\mathbb{E}^3$

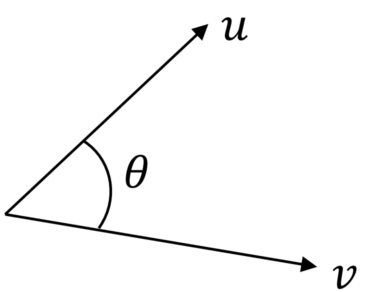
- The dot product of two vectors u and v is defined as

$$\langle u, v \rangle \doteq u^T v = u_1 v_1 + u_2 v_2 + u_3 v_3, \quad \forall u, v \in \mathbb{R}^3.$$

- Dot product can be used to measure distance:

$$\sqrt{\langle v, v \rangle} = \sqrt{v_1^2 + v_2^2 + v_3^2} \;,$$

- And angle between two vectors:



$$\cos\theta = \frac{\langle u, v \rangle}{\|u\| \|v\|}.$$

$\langle u, v \rangle = 0 \Rightarrow$ *orthogonal* vectors

Dot and Cross Products in $\mathbb{E}^3$

- Given two vectors $\mathrm{v}, \mathrm{u} \in \mathbb{R}^3$, their cross product is a third vector with coordinates given by:

$$u \times v \doteq \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix} \quad \in \mathbb{R}^3.$$

- Cross product of two vectors is linear in each of its arguments:

$$u \times (\alpha v + \beta w) = \alpha u \times v + \beta u \times w, \ \forall\, \alpha, \beta \in \mathbb{R}.$$

Dot and Cross Products in $\mathbb{E}^3$

- The cross product of two vectors is orthogonal to each of its factors:

$$\langle u \times v, u \rangle = \langle u \times v, v \rangle = 0, \quad u \times v = -v \times u.$$

- Thus, the order of the factors defines an *orientation*, i.e. if we change the order of the factors, the cross product changes sign.

Dot and Cross Products in $\mathbb{E}^3$

- The cross product can be represented by a map from $\mathbb{R}^3$ to $\mathbb{R}^3$: $v \mapsto u \times v$.

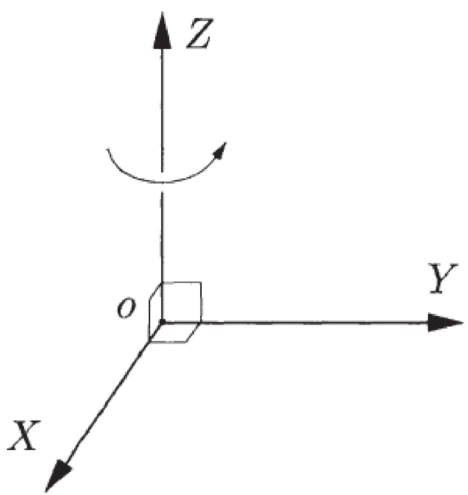
- This map is linear in v and therefore can be represented by a matrix:

$$\widehat{u} \doteq \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}.$$

- Hence, we can write $u \times v = \hat{u}v$. Note that u is a 3 x 3 skew-symmetric matrix, i.e. $\hat{u}^\top = -\widehat{u}$.

Right-Hand Rule

- For a standard Cartesian frame, the cross product of the principal axes *X* and Y gives the principal axis *Z.*
- The cross product therefore conforms to the *right-hand rule*.



Easy to verify that:

$$e_1 \doteq [1,0,0]^T, \; e_2 \doteq [0,1,0]^T \in \mathbb{R}^3,$$

we have:

$$e_1 \times e_2 = [0,0,1]^T \doteq e_3.$$

Image source: Y. Ma, S. Soatto, J. Kosecka, S. S. Sastry, " An invitation to 3-D vision.

Coordinate Frames

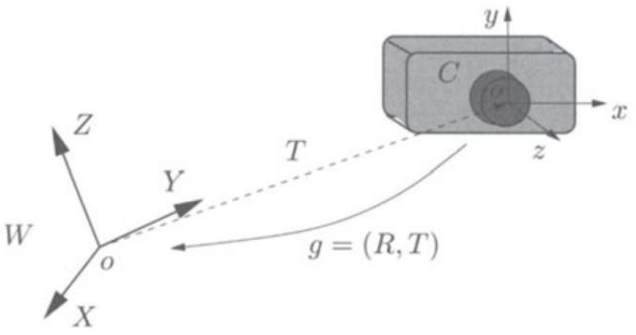
- A rigid object can always be associated with a right-handed orthonormal frame, which we call the *object coordinate frame* or the *body coordinate frame.*

- And its rigid-body motion can be entirely specified by the motion of such a frame with respect to a reference frame, which we call the *world frame*.

Coordinate Frames

Example: A camera frame F_C : (x, y, z), moving relative to a world reference frame F_W : *(X, Y, Z)* selected in advance.

The configuration of the camera is then determined by two components:



1. The translation vector T between the origins *O* of F_W and F_C .

2. The relative orientation R between the coordinate axes of F_W and F_C .

Image source: Y. Ma, S. Soatto, J. Kosecka, S. S. Sastry, " An invitation to 3-D vision.

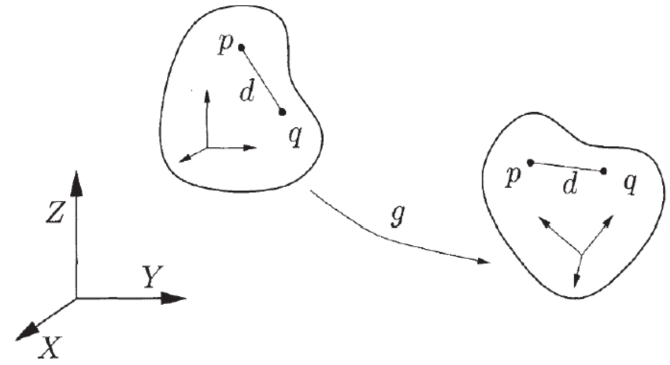
Special Euclidean Transformation

- **Definition:**

A map $g : \mathbb{R}^3 \mapsto \mathbb{R}^3$ *is a* rigid-body motion *or a* special Euclidean transformation *if it preserves the norm and the cross product of any two vectors,*

1. *norm:* $\|g_*(v)\| = \|v\|, \ \forall v \in \mathbb{R}^3,$ (distance preserving)
2. *cross product:* $g_*(u) \times g_*(v) = g_*(u \times v), \ \forall u, v \in \mathbb{R}^3,$ (orientation preserving).

The collection of all such motions or transformations is denoted by $SE(3)$.



Special Euclidean Transformation

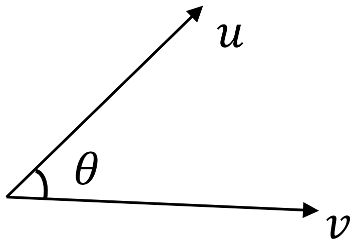
Preservation of angles:

- The inner product $\langle ., . \rangle$ can be expressed in terms of the norm $\|.\|$ by the *polarization identity*:

$$\langle u, v \rangle = \frac{1}{4} \left(\|u + v\|^2 - \|u - v\|^2 \right)$$

- Since $\|u + v\| = \|g_*(u) + g_*(v)\|$, we can conclude that, for any rigid-body motion g,

$$\langle u, v \rangle = \langle g_*(u), g_*(v) \rangle, \quad \forall u, v \in \mathbb{R}^3.$$



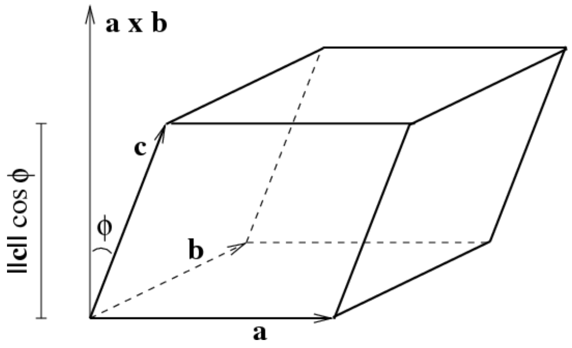
- Angle between two vectors: $\cos\theta = \frac{\langle u,v \rangle}{\|u\|\|v\|} = \frac{\langle g_*(u), g_*(v) \rangle}{\|g_*(u)\|\|g_*(v)\|}$.

Special Euclidean Transformation

Preservation of volume:

- From the definition of a rigid-body motion, one can show that it also preserves the so-called *triple product* among three vectors:

$$\langle g_*(u), g_*(v) \times g_*(w) \rangle = \langle u, v \times w \rangle.$$



$$\begin{aligned} \text{Volume} &= \text{area of base} \cdot \text{height} \\ &= \|\mathbf{a} \times \mathbf{b}\| \, \|\mathbf{c}\| \, |\cos\phi| = |(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}|. \end{aligned}$$

Images source: https://mathinsight.org/image/volume_parallelepiped

Orthogonal Matrix Representation of Rotations

- A point $\mathbf{X}$ can be written as a linear combination of the basis in F_W:

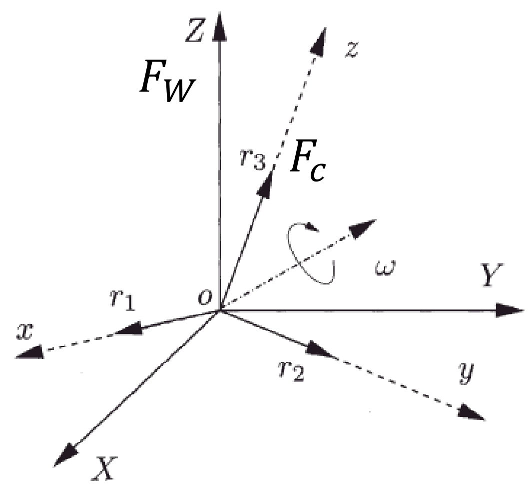
$$\mathbf{X} = x\begin{bmatrix}1\\0\\0\end{bmatrix} + y\begin{bmatrix}0\\1\\0\end{bmatrix} + z\begin{bmatrix}0\\0\\1\end{bmatrix}.$$

- A change of basis implies we want to write $\mathbf{X}$ as a linear combination of a set of new basis in F_C:

$$\mathbf{X} = x'\mathbf{r}_1 + y'\mathbf{r}_2 + z'\mathbf{r}_3$$

$$= \underbrace{[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3]}_{\text{3x3 rotation matrix that transforms } \mathbf{X}' \text{ into } \mathbf{X}}\begin{bmatrix}x'\\y'\\z'\end{bmatrix}.$$

3x3 rotation matrix that transforms $\mathbf{X}'$ into $\mathbf{X}$



Orthogonal Matrix Representation of Rotations

- Since r_1, r_2, r_3 form an orthonormal frame, it follows that:

$$r_i^T r_j = \delta_{ij} \doteq \begin{cases} 1 & \text{for} \quad i = j, \\ 0 & \text{for} \quad i \neq j, \end{cases} \qquad \forall i, j \in \{1, 2, 3\}.$$

- This can be written in matrix form as:

$$R_{wc}^T R_{wc} = R_{wc} R_{wc}^T = I \quad \Rightarrow \quad R_{wc}^{-1} = R_{wc}^T.$$

- Any matrix that satisfies the above identity is called an *orthogonal matrix*.

Orthogonal Matrix Representation of Rotations

- Since r_1, r_2, r_3 form a right-handed frame, we have:

$$\det(R_{wc}) = +1.$$

- Hence, R_{wc} is a *special orthogonal matrix,* the word "special" indicates that it is orientation-preserving.

- The space of all such special orthogonal matrices in $\mathbb{R}^3$ is usually denoted by:

$$SO(3) \doteq \left\{ R \in \mathbb{R}^{3\times 3} \mid R^T R = I, \det(R) = +1 \right\}.$$

Orthogonal Matrix Representation of Rotations

- We can show that rotations indeed preserve both the inner and cross product of vectors.

Proof:

1. Preservation of dot product:

$$(Rv)^\top(Ru) = v^\top \underbrace{(R^\top R)}_{=1} u = v^\top u.$$

$\square$

2. Preservation of cross product:

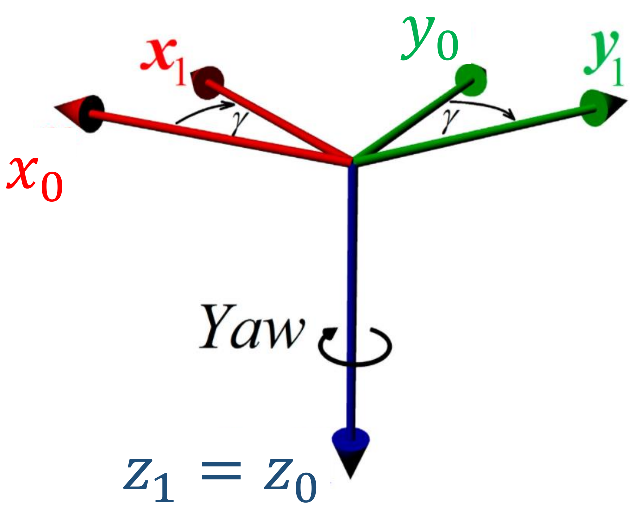
$$(Rv) \times (Ru) = \det R \, \underbrace{(R^{-1})^\top}_{=R} (v \times u)$$

$$= R(v \times u)$$

$\square$

Euler Angles to Rotation Matrix

- The yaw angle γ is the rotation around the z-axis.



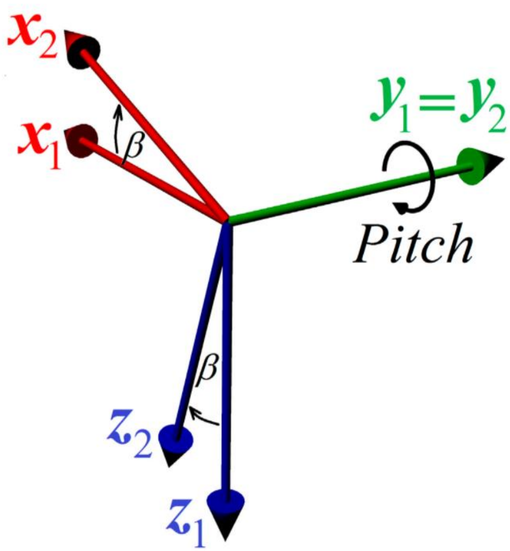
$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Longrightarrow \boldsymbol{X}_0 = R_{01}\boldsymbol{X}_1$$

Image Source: http://www.mdpi.com/1424-8220/15/3/7016/htm

Euler Angles to Rotation Matrix

- The pitch angle β is the rotation around the y-axis.



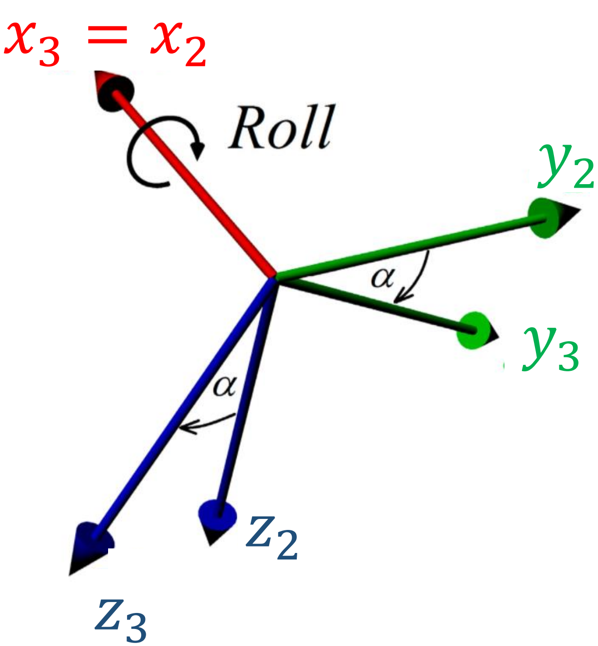
$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$\Longrightarrow \boldsymbol{X}_1 = R_{12}\boldsymbol{X}_2$$

Image Source: http://www.mdpi.com/1424-8220/15/3/7016/htm

Euler Angles to Rotation Matrix

- The roll angle α is the rotation around the x-axis.



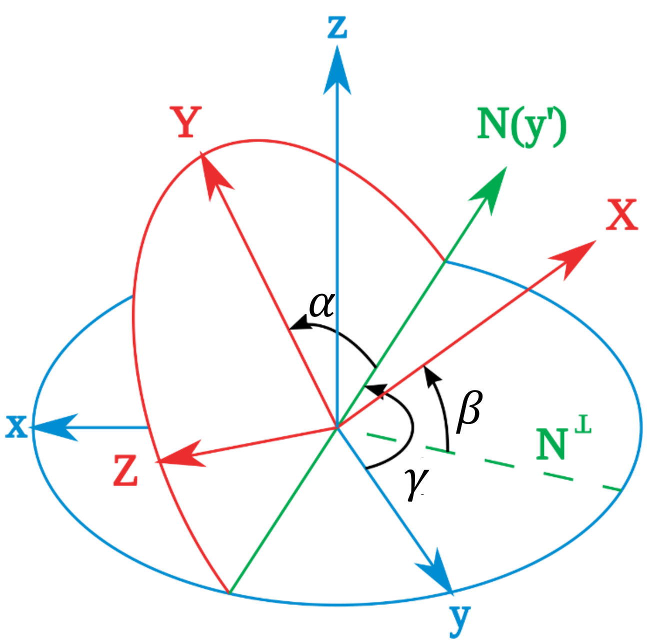
$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$\Longrightarrow \boldsymbol{X}_2 = R_{23}\boldsymbol{X}_3$$

Image Source: http://www.mdpi.com/1424-8220/15/3/7016/htm

Euler Angles to Rotation Matrix

- Tait–Bryan angles *z-y′-x″* sequence, i.e. rotation around z, rotation around y' and finally rotation around x''.



$$R_{03} = R_{01} R_{12} R_{23}$$

$$= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$\Longrightarrow \boldsymbol{X}_0 = R_{03} \boldsymbol{X}_3$$

Image Source: https://en.wikipedia.org/wiki/Euler_angles

Rigid-body motion and its Representations

- The point p on the object w.r.t F_W is represented by the vector X_w
- X_w is simply the sum of the translation $T_{wc} \in \mathbb{R}^3$ in F_c and X_C *in* F_W.
- Since X_c is the point p in F_C, it becomes $R_{wc}X_c$ in F_W, *where* $R_{wc} \in SO(3)$.
- We get: $\boldsymbol{X}_w = R_{wc}\boldsymbol{X}_c + T_{wc}$.

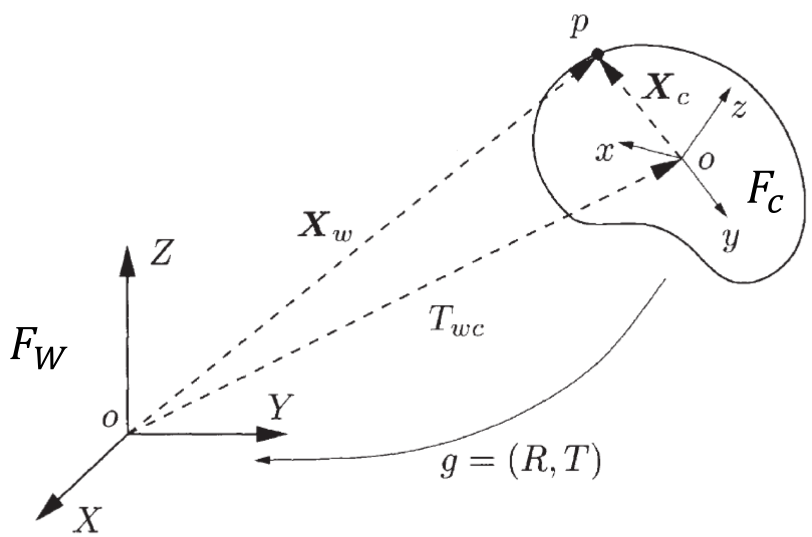


Image source: Y. Ma, S. Soatto, J. Kosecka, S. S. Sastry, " An invitation to 3-D vision.

Homogeneous Representation

- The transformation $\boldsymbol{X}_w = R_{wc}\boldsymbol{X}_c + T_{wc}$ can be written in a "linear form" as:

$$\bar{\boldsymbol{X}}_w = \begin{bmatrix} \boldsymbol{X}_w \\ 1 \end{bmatrix} = \begin{bmatrix} R_{wc} & T_{wc} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_c \\ 1 \end{bmatrix} \doteq g_{wc}\, \bar{\boldsymbol{X}}_c.$$

- g is the homogeneous representation given by:

$$g = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \quad \in \mathbb{R}^{4\times 4}.$$

Homogeneous Representation

- The homogeneous representation of g gives rise to a natural matrix representation of the special Euclidean transformations:

$$SE(3) \doteq \left\{ g = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \middle| R \in SO(3), T \in \mathbb{R}^3 \right\} \quad \subset \mathbb{R}^{4\times 4}.$$

- $\forall g_1, g_2 \in SE(3)$, we have

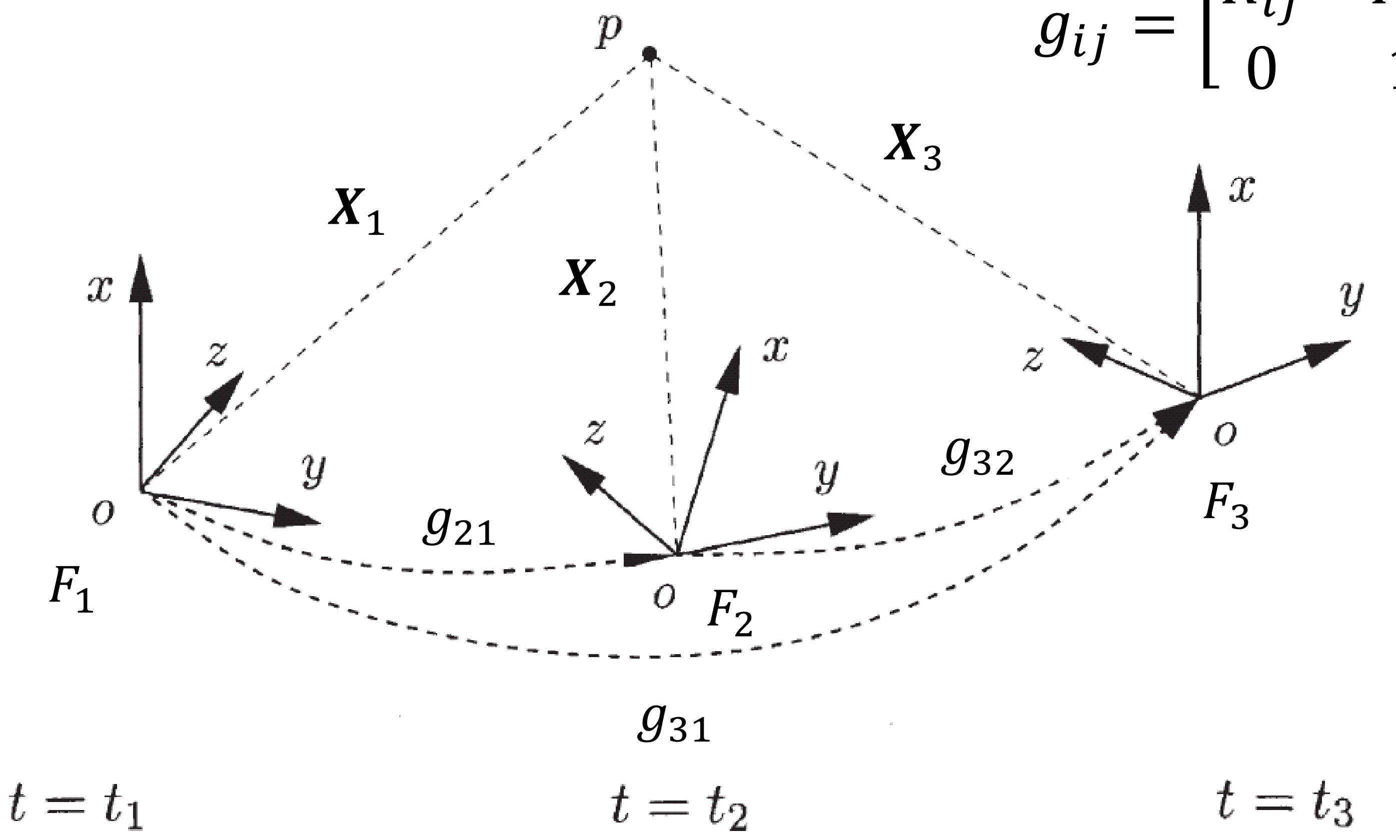
$$g_1 g_2 = \begin{bmatrix} R_1 & T_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & T_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 T_2 + T_1 \\ 0 & 1 \end{bmatrix} \quad \in SE(3)$$

and $$g^{-1} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T T \\ 0 & 1 \end{bmatrix} \quad \in SE(3).$$

Composition of Rigid-body Motions

- Given three camera frames at time $t = t_1, t_2, t_3$, respectively.

$$g_{ij} = \begin{bmatrix} R_{ij} & T_{ij} \\ 0 & 1 \end{bmatrix} \in SE(3)$$



Composition of Rigid-body Motions

- Then we have the following relationship between coordinates of the same point p at different frames:

$$\boldsymbol{X}_2 = g_{21}\boldsymbol{X}_1, \quad \boldsymbol{X}_3 = g_{32}\boldsymbol{X}_2, \quad \boldsymbol{X}_3 = g_{31}\boldsymbol{X}_1.$$

- This implies the following *composition rule*:

$$g_{32}g_{21} = g_{31},$$

since

$$\boldsymbol{X}_3 = g_{32}\boldsymbol{X}_2 = g_{32}g_{21}\boldsymbol{X}_1 = g_{31}\boldsymbol{X}_1,$$

Composition of Rigid-body Motions

- The same composition rule implies the *rule of inverse*:

$$g_{21}^{-1} = g_{12}.$$

- since $g_{21}g_{12} = g_{22} = I$.

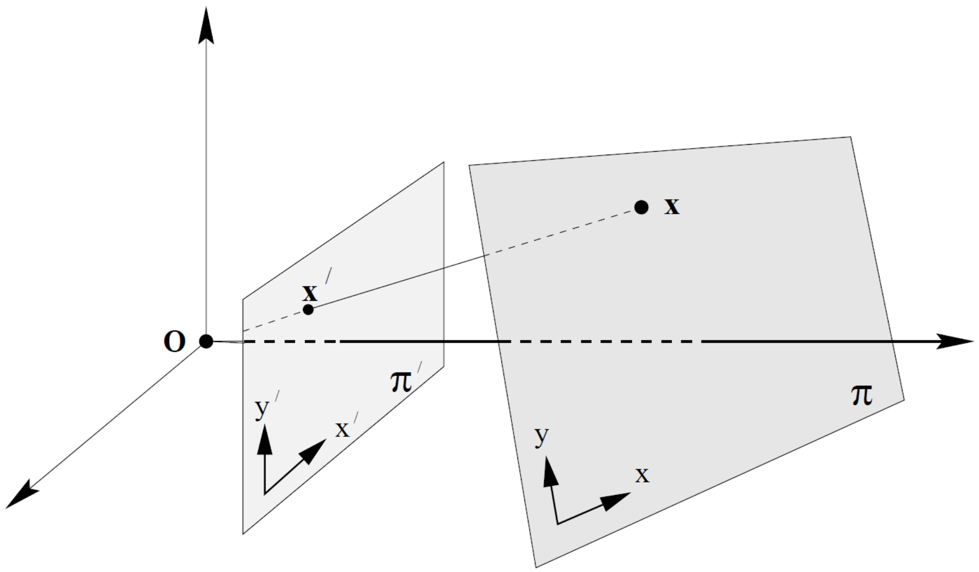
- In general, the composition rules in homogeneous representation is given by:

$$\boldsymbol{X}_i = g_{ij}\boldsymbol{X}_j, \quad g_{ik} = g_{ij}g_{jk}, \quad g_{ij}^{-1} = g_{ji}.$$

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 $\mathbf{x}' = \mathrm{H}\mathbf{x}$.

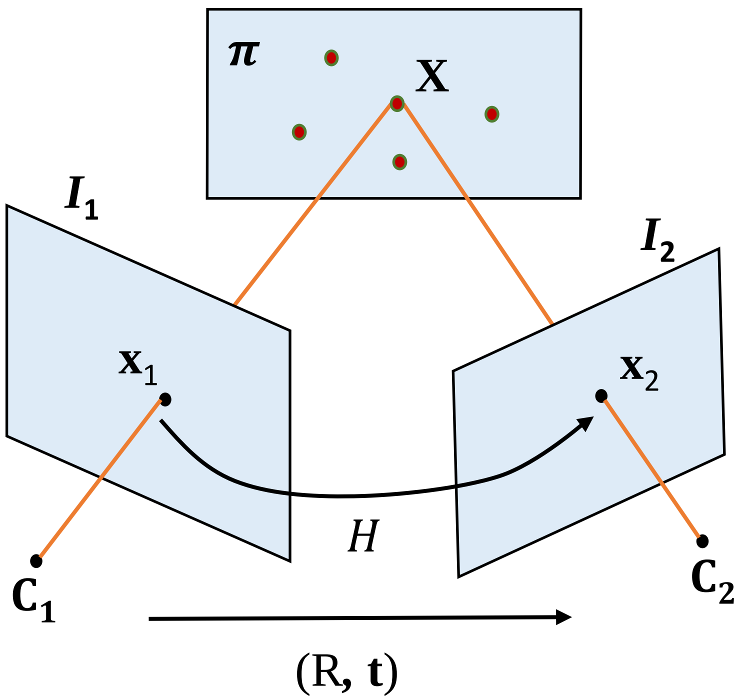


This is also known as Homography!

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:

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

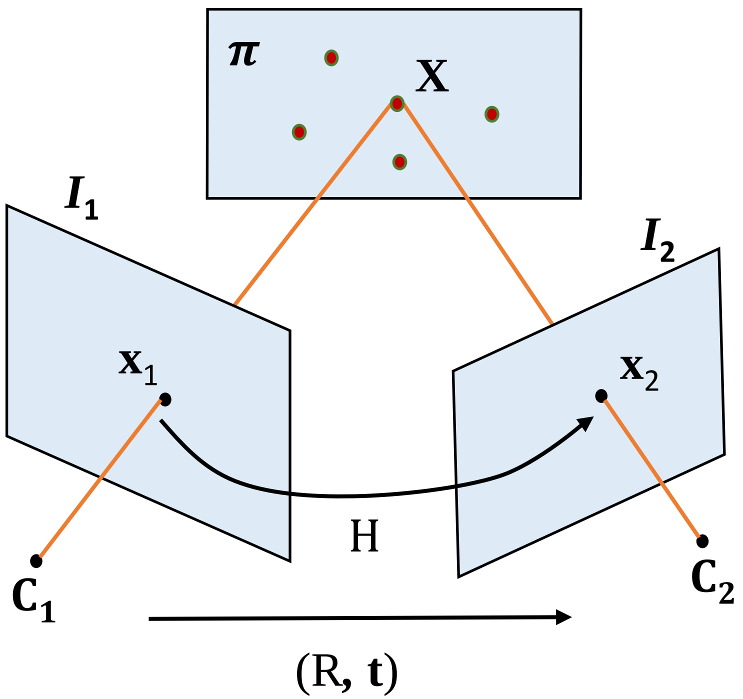
- $\mathbf{N} = [n_1, n_2, n_3]^\top$ is the unit normal vector representing the plane $\boldsymbol{\pi}$ w.r.t $\mathbf{C}_1$, and d is the perpendicular distance from plane to $\mathbf{C}_1$:

$$\mathbf{N}^\top \mathbf{X}_1 = n_1 X + n_2 Y + n_3 Z = d,$$

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

Existence of Projective Homography

1. Planar scene:



- Combining the two equations, we get

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

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

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

Existence of Projective Homography

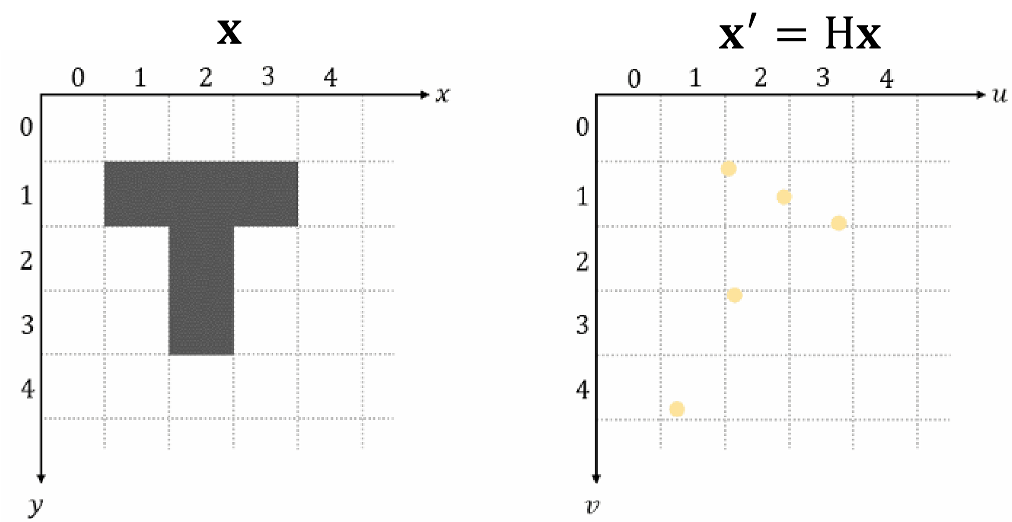
2. Plane at infinity: Scene is very far away from the camera, e.g., aerial images, i.e.

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

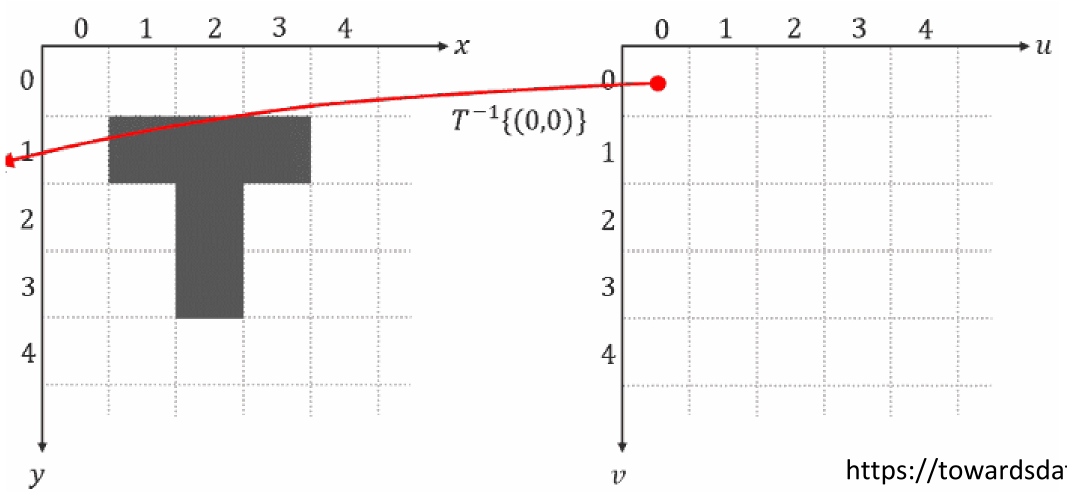
This is the same as pure rotation, i.e., $\mathbf{t} = (0,0,0)^{\top}$:

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

Reverse Mapping



Directly computing $\mathbf{x}' = \mathrm{H}\mathbf{x}$ leads to holes.

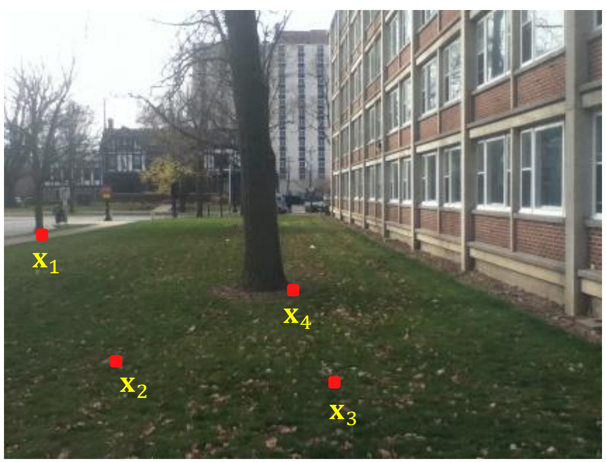


Instead, we should create a lookup table of $\mathrm{H}^{-1}\mathbf{x}' = \mathbf{x}$.

https://towardsdatascience.com/spatial-transformer-tutorial-part-1-forward-and-reverse-mapping-8d3f66375bf5

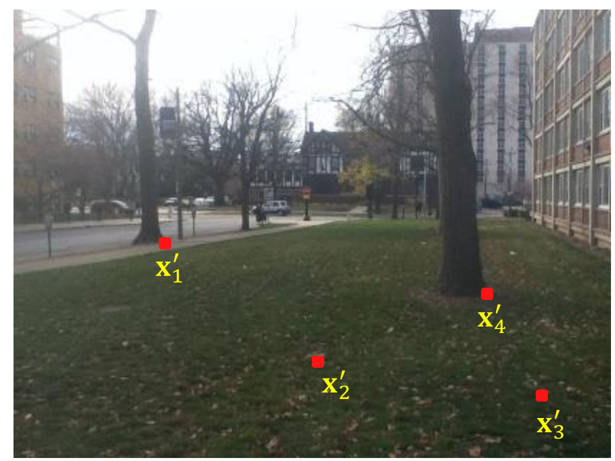
2D Homography

- **Given**: A set of points correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$ between two images.
- **Compute**: The 2D Homography, H such that $\mathrm{H}\mathbf{x}_i = \mathbf{x}_i'$ for each i.



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

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

- Therefore, 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.
- 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 $\mathrm{H}\mathbf{x}_i = \mathbf{x}_i'$ in terms of vector cross product:

$$\mathbf{x}_i' \times \mathrm{H}\mathbf{x}_i = \mathbf{0}, \text{ where } \mathrm{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\top}\mathbf{x}_i \\ \mathbf{h}^{2\top}\mathbf{x}_i \\ \mathbf{h}^{3\top}\mathbf{x}_i \end{pmatrix} \text{ and } \mathbf{x}_i' = (x_i', y_i', w_i')^\top.$$

- The cross product may then be given explicitly as:

$$\mathbf{x}_i' \times \mathrm{H}\mathbf{x}_i = \begin{pmatrix} y_i'\mathbf{h}^{3\top}\mathbf{x}_i - w_i'\mathbf{h}^{2\top}\mathbf{x}_i \\ w_i'\mathbf{h}^{1\top}\mathbf{x}_i - x_i'\mathbf{h}^{3\top}\mathbf{x}_i \\ x_i'\mathbf{h}^{2\top}\mathbf{x}_i - y_i'\mathbf{h}^{1\top}\mathbf{x}_i \end{pmatrix}.$$

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!

$$\begin{bmatrix} \mathbf{0}^{\mathsf{T}} & -w_i'\mathbf{x}_i^{\mathsf{T}} & y_i'\mathbf{x}_i^{\mathsf{T}} \\ w_i'\mathbf{x}_i^{\mathsf{T}} & \mathbf{0}^{\mathsf{T}} & -x_i'\mathbf{x}_i^{\mathsf{T}} \\ \cancel{-y_i'\mathbf{x}_i^{\mathsf{T}}} & \cancel{x_i'\mathbf{x}_i^{\mathsf{T}}} & \cancel{\mathbf{0}^{\mathsf{T}}} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

$$\Downarrow$$

$$\begin{bmatrix} \mathbf{0}^{\mathsf{T}} & -w_i'\mathbf{x}_i^{\mathsf{T}} & y_i'\mathbf{x}_i^{\mathsf{T}} \\ w_i'\mathbf{x}_i^{\mathsf{T}} & \mathbf{0}^{\mathsf{T}} & -x_i'\mathbf{x}_i^{\mathsf{T}} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}, \text{ or} \qquad \mathtt{A}_i\mathbf{h} = \mathbf{0}.$$

- 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.

Direct Linear Transformation (DLT) Algorithm

- $\mathbf{h}$ has 8 degrees of freedom and each point correspondence gives two constraints.

- A minimum of 4-point correspondences is needed to solve for h, i.e. $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, for $i \geq 4$.

- Stacking all equations together, we get:

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

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

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

Singular Value Decomposition (SVD)

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

$$\mathrm{A} = \underbrace{\begin{bmatrix} u_1 & \dots & u_m \end{bmatrix}}_{\text{Left singular vectors (m x m)}} \underbrace{\begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \sigma_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \sigma_n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Singular values (m x n)}} \underbrace{\begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}^\top}_{\text{Right singular vectors (n x n)}} = \mathrm{U}\Sigma V^\top$$

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

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

Singular Value Decomposition (SVD)

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

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

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

$$\mathrm{A}v_i = u_i\sigma_i$$

- $||\mathrm{A}v_i||$ 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:

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

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

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

Direct Linear Transformation (DLT) Algorithm

Objective

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

Algorithm

(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 $\mathbf{h}$.

Slide credit: Marc Pollefeys

Homography: Degeneracy

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

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

Importance of Normalization

Problem:

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

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

$\sim 10^2$ $\sim 10^2$ 1 $\sim 10^2$ $\sim 10^2$ 1 $\sim 10^4$ $\sim 10^4$ $\sim 10^2$

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.

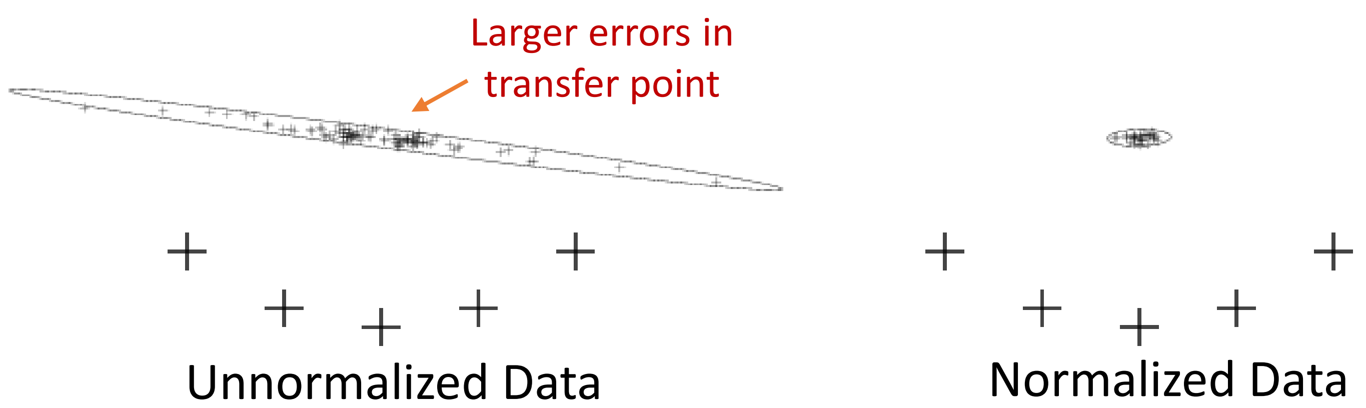


Image Source: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Data Normalization

- Data normalization is carried out by a transformation of the points as follows:
    1. Points are translated so that their centroid is at the origin.
    2. Points are then scaled so that the average distance from the origin is equal to sqrt(2).
    3. 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
- $\Rightarrow$ 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!

<u>Objective</u>

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

<u>Algorithm</u>

(i) Normalize points $\tilde{\mathrm{x}}_\mathrm{i} = \mathrm{T}_\mathrm{norm}\mathrm{x}_\mathrm{i}, \tilde{\mathrm{x}}'_\mathrm{i} = \mathrm{T}'_\mathrm{norm}\mathrm{x}'_\mathrm{i}$

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

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

$$\mathrm{T}_\mathrm{norm} = \begin{bmatrix} s & 0 & -sc_x \\ 0 & s & -sc_y \\ 0 & 0 & 1 \end{bmatrix}$$

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:

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

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

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

Different Cost Functions: Algebraic Distance

- 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**:

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

- This is the Euclidean image distance in the second image between the measured point $\mathbf{x}'_i$ and the corresponding point $\mathrm{H}\mathbf{x}_i$ transferred from the first image.
- The error is minimized over the estimated homography H.

Different Cost Functions: Geometric Distance

Symmetric Transfer Error

- 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:

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

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

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

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

Symmetric Transfer Error (upper) Vs Reprojection Error

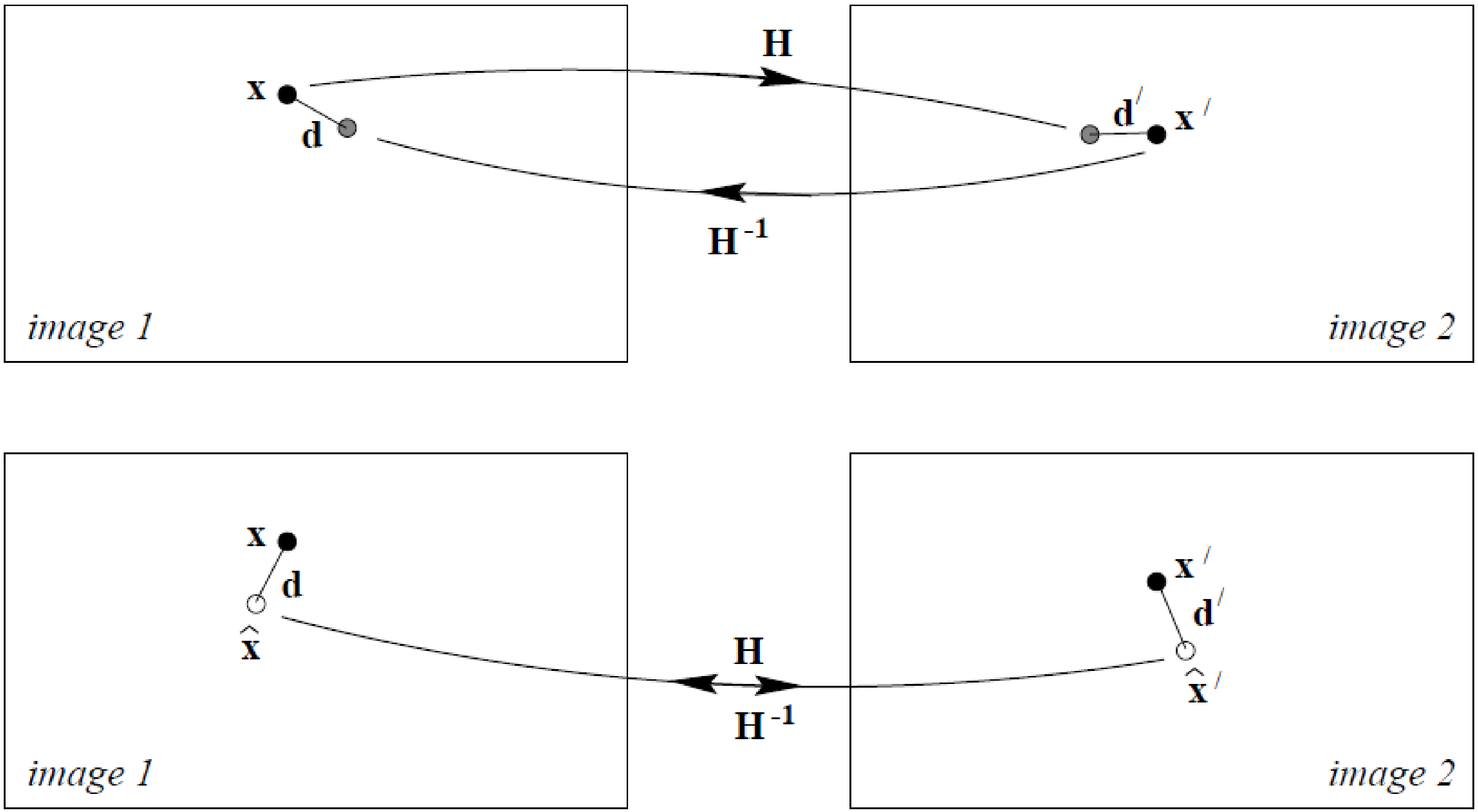


Image Source: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

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.

- 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

- Let $\mathrm{C_H}(\mathbf{X}) = \mathbf{0}$ denote the cost function $\mathrm{A}\mathbf{h} = \mathbf{0}$ that is satisfied by the point $\mathbf{X} = (x, y, x', y')^\top$ for a given homography H.

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

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

Different Cost Functions: Sampson Error

- The approximated cost function can be rewritten as:

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

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

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

Different Cost Functions: Sampson Error

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

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

- And the Sampson error is defined by the norm:

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

Different Cost Functions: Sampson Error

- For the 2D homography estimation problem, $\mathbf{X} = (x, y, x', y')^{\mathsf{T}}$ where the measurements are $\mathbf{x} = (x, y, 1)^{\mathsf{T}}$ and $\mathbf{x}' = (x', y', 1)^{\mathsf{T}}$.
- And $\boldsymbol{\epsilon} = \mathrm{C}_H(\mathbf{X})$ 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 :

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

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

  where

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

- 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:

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

  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:

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

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

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:

  $f : (\mathbf{h}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n) \mapsto (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_1', \ldots, \hat{\mathbf{x}}_n, \hat{\mathbf{x}}_n')$, 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 4*n*-vector.

Iterative Minimization

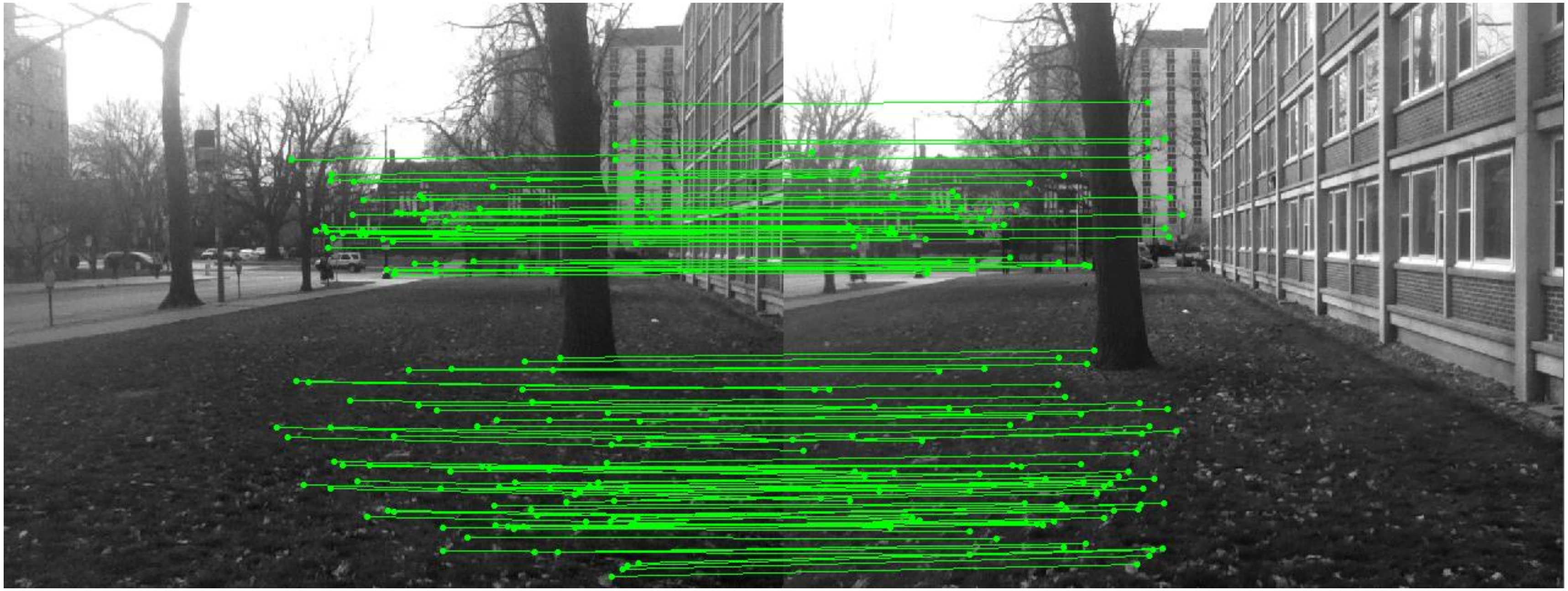
Sampson Approximation:

- Measurement vector $\mathbf{X} = (x, y, x', y')^\mathsf{T}$.
- 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:

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

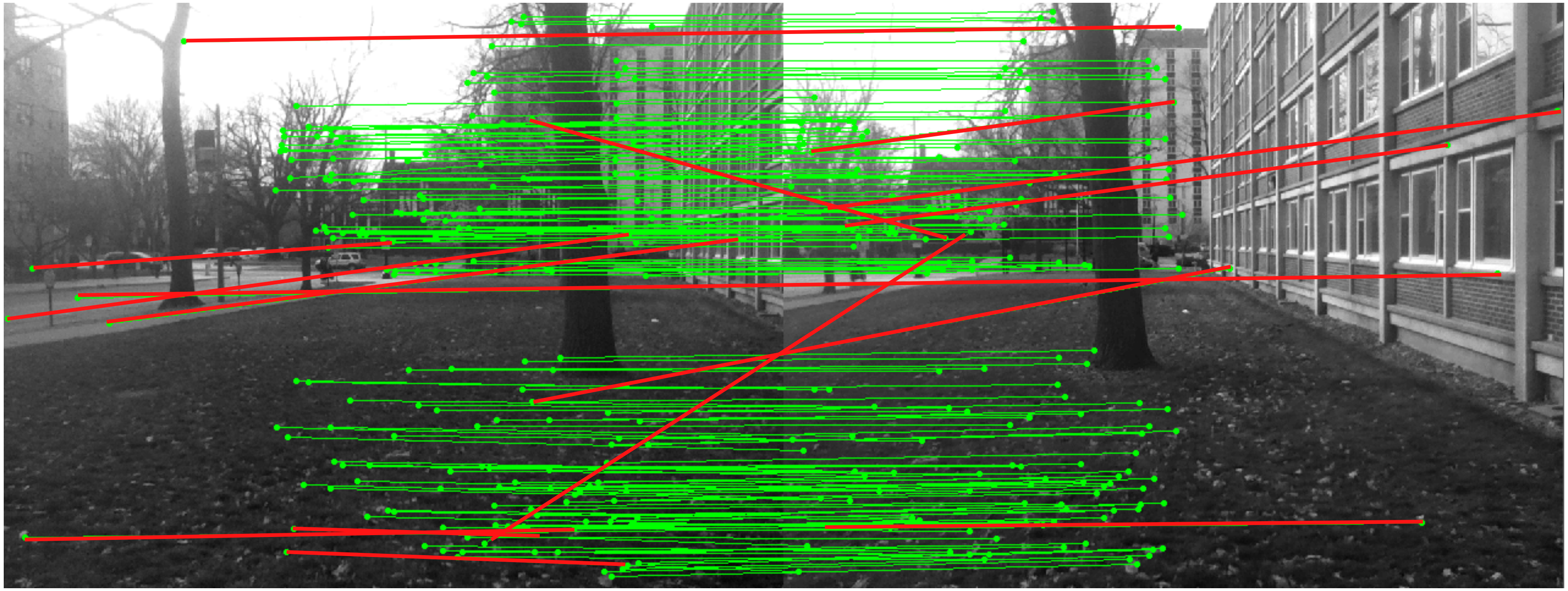
Random Sample Consensus: RANSAC

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



Random Sample Consensus: RANSAC

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



The RANSAC Song

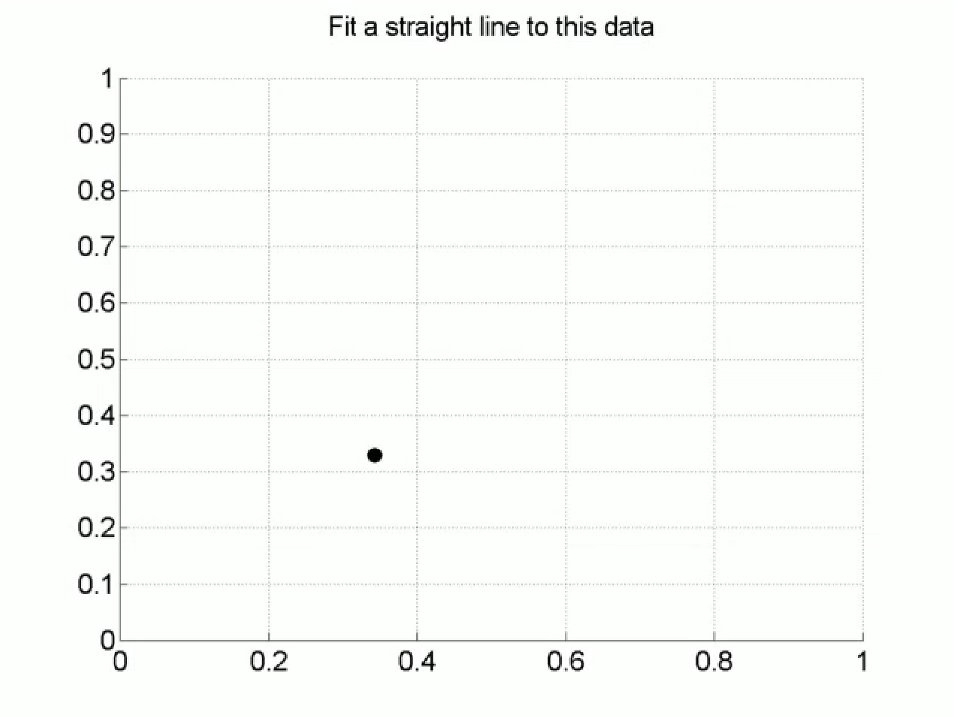
When you have outliers you may face much frustration
if you include them in a model fitting operation.
But if your model's fit to a sample set of minimal size,
the probability of the set being outlier-free will rise.
Brute force tests of all sets will cause computational constipation.

N random samples
will provide an example
of a fitted model uninfluenced by outliers. No need to test all combinations!

Each random trial should have its own unique sample set
and make sure that the sets you choose are not degenerate.
N, the number of sets, to choose is based on the probability
of a point being an outlier, and of finding a set that's outlier free.
Updating *N* as you go will minimise the time spent.

So if you gamble
that *N* samples are ample
to fit a model to your set of points, it's likely that you will win the bet.

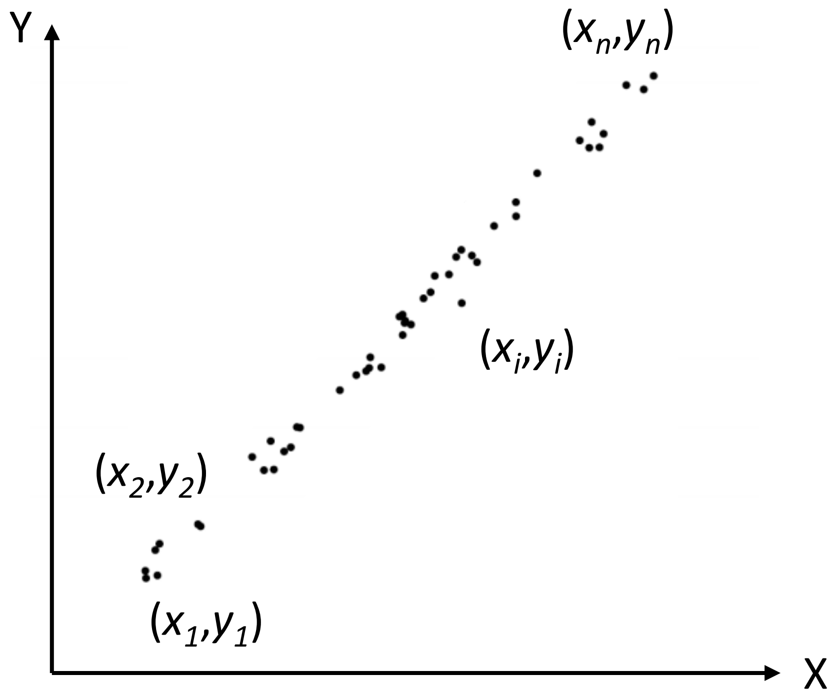
Select the set that boasts
that its number of inliers is the most (you're almost there).
Fit a new model just to those inliers and discard the rest,
an estimated model for your data is now possessed!
This marks the end point of your model fitting quest.



Source: http://danielwedge.com/ransac/

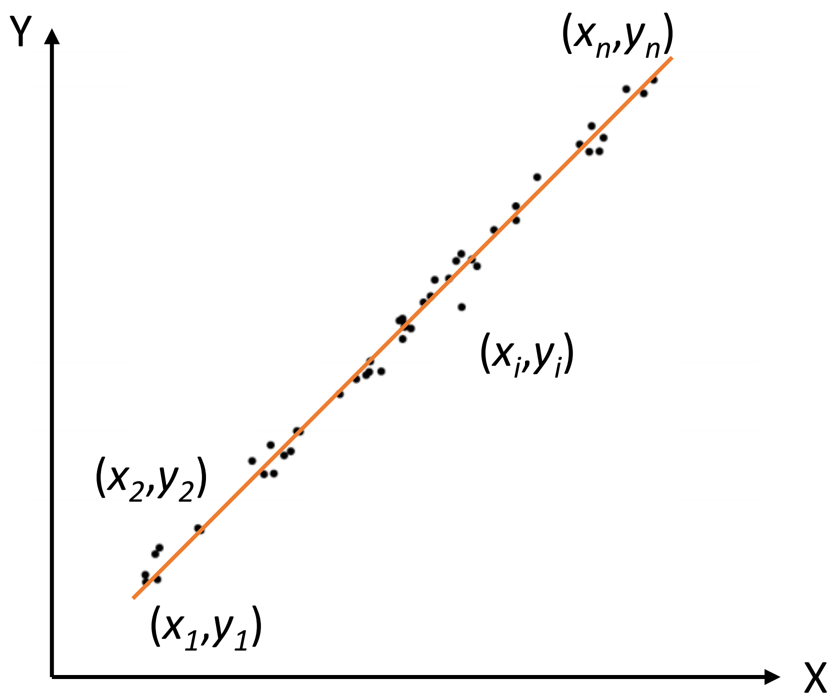
RANSAC: Line Fitting Example

- **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 $i = 1, \ldots, n$

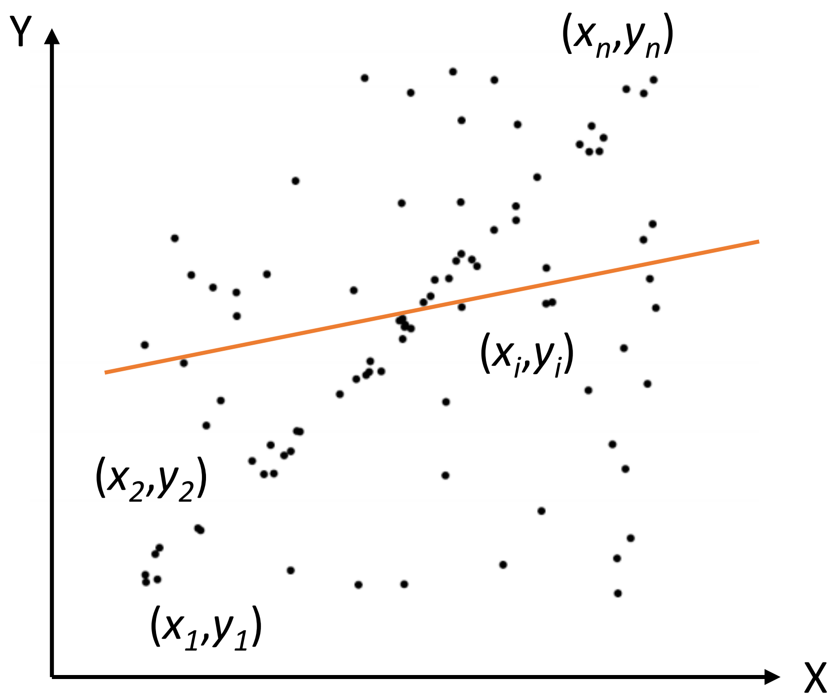


Least-squares solution:

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

RANSAC: Line Fitting Example

- **Given**: n data points (x_i,y_i), for $i = 1, \ldots, n$
- **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:

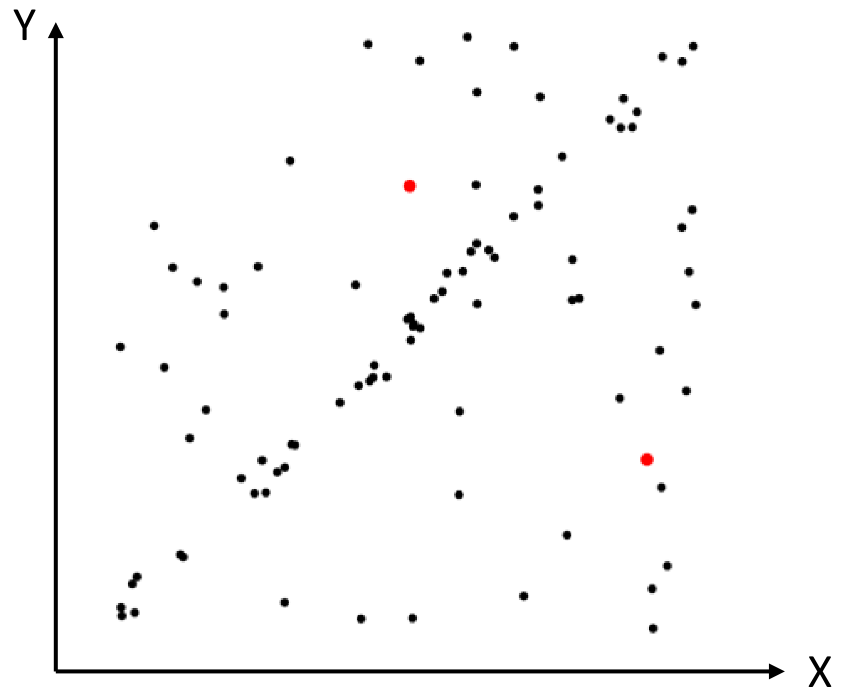
$$\underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{n} \|y_i - (mx_i + c)\|^2$$

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

RANSAC: Line Fitting Example

RANSAC Steps:

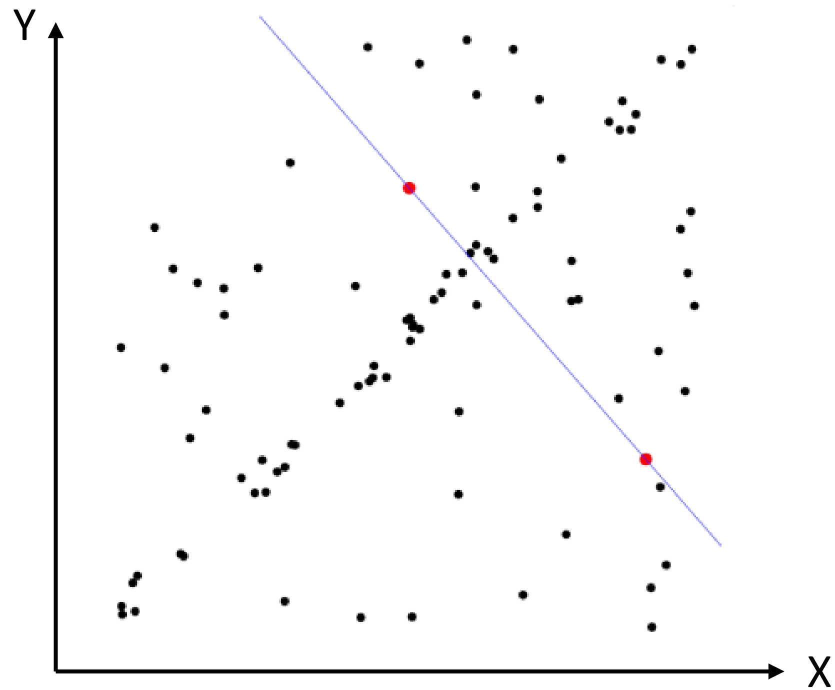
1. Randomly select minimal subset of points, i.e. 2 points



RANSAC: Line Fitting Example

RANSAC Steps:

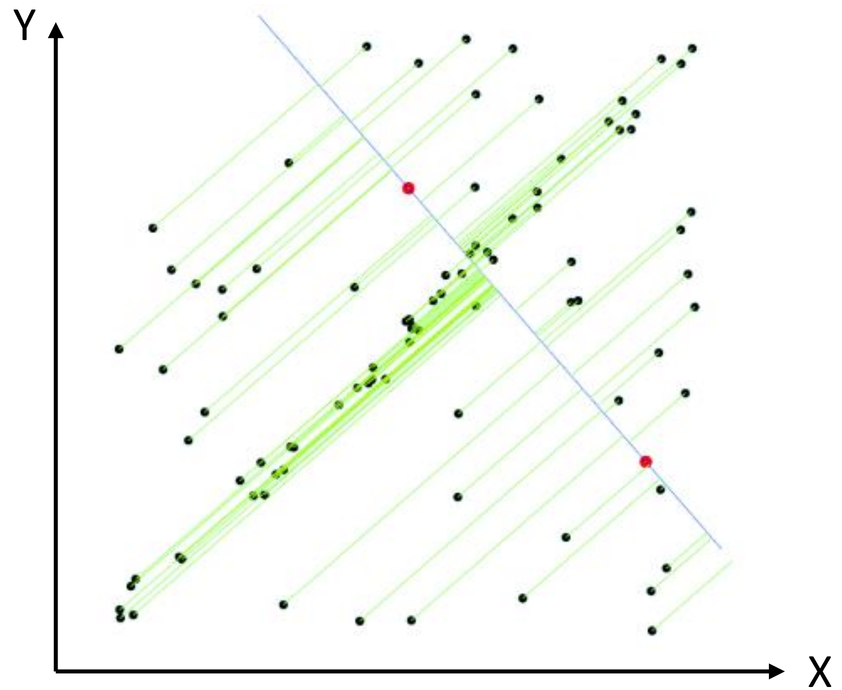
2. Hypothesize a model



RANSAC: Line Fitting Example

RANSAC Steps:

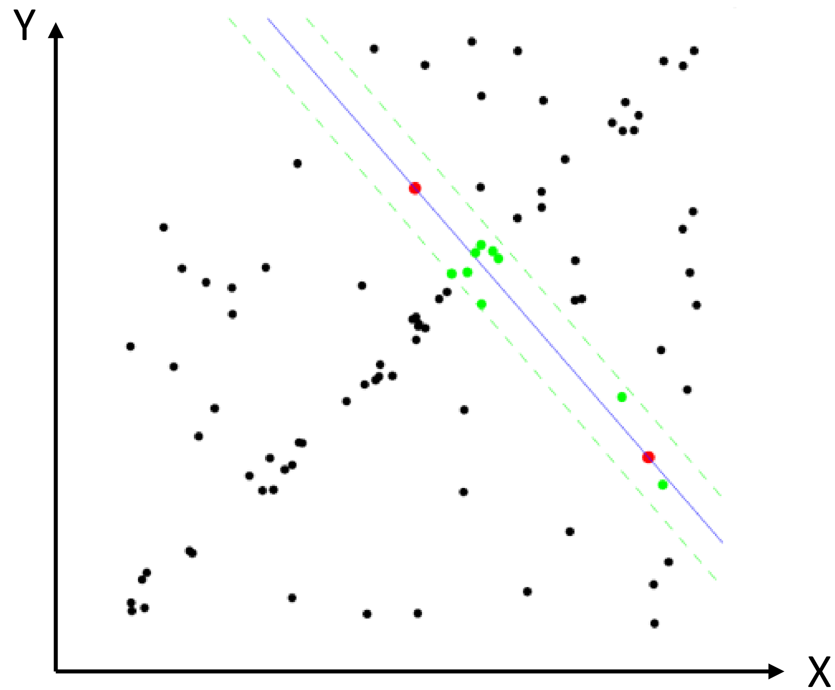
3. Compute error function, i.e. shortest point to line distance



RANSAC: Line Fitting Example

RANSAC Steps:

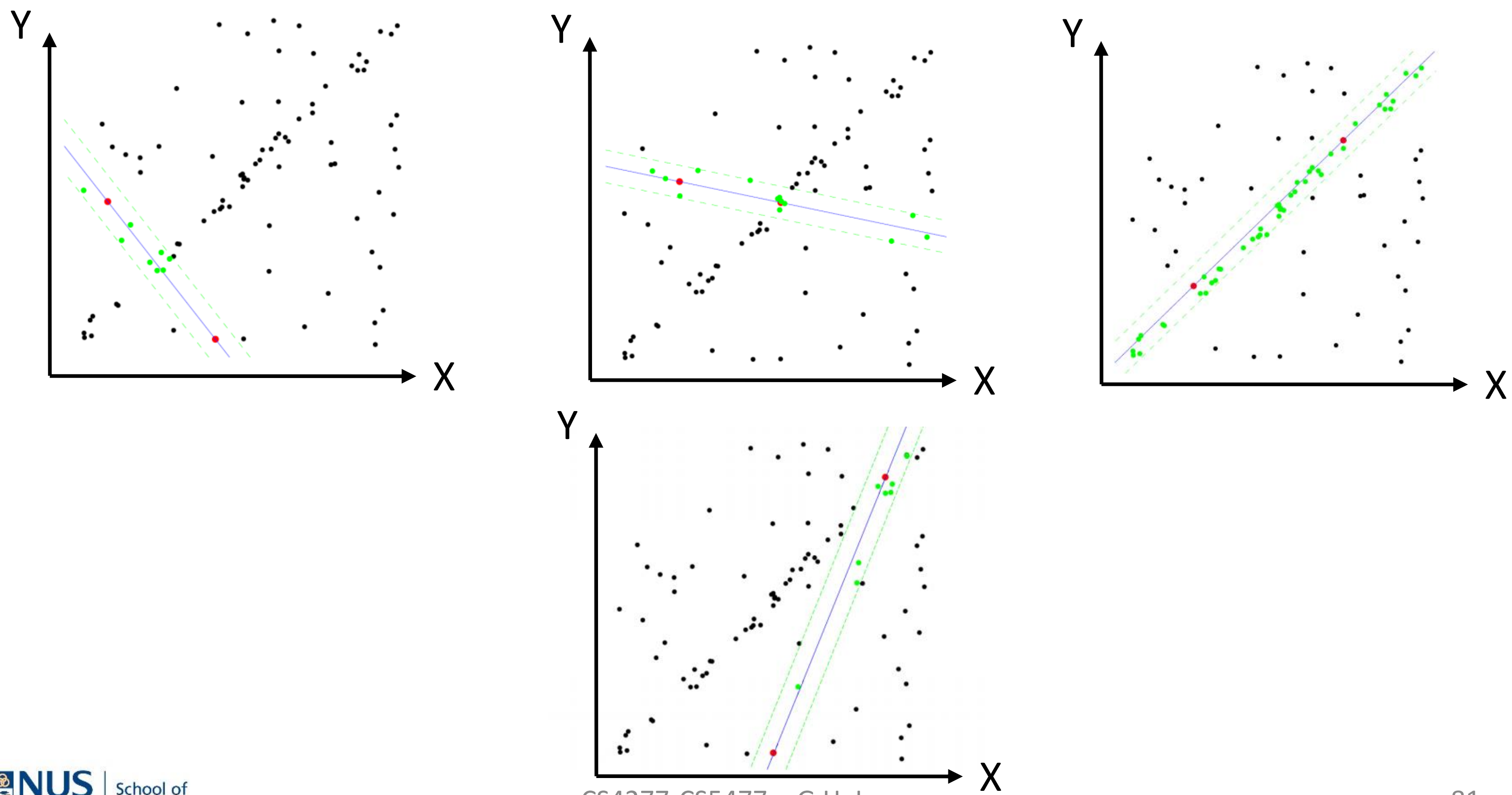
4. Select points consistent with model



RANSAC: Line Fitting Example

RANSAC Steps:

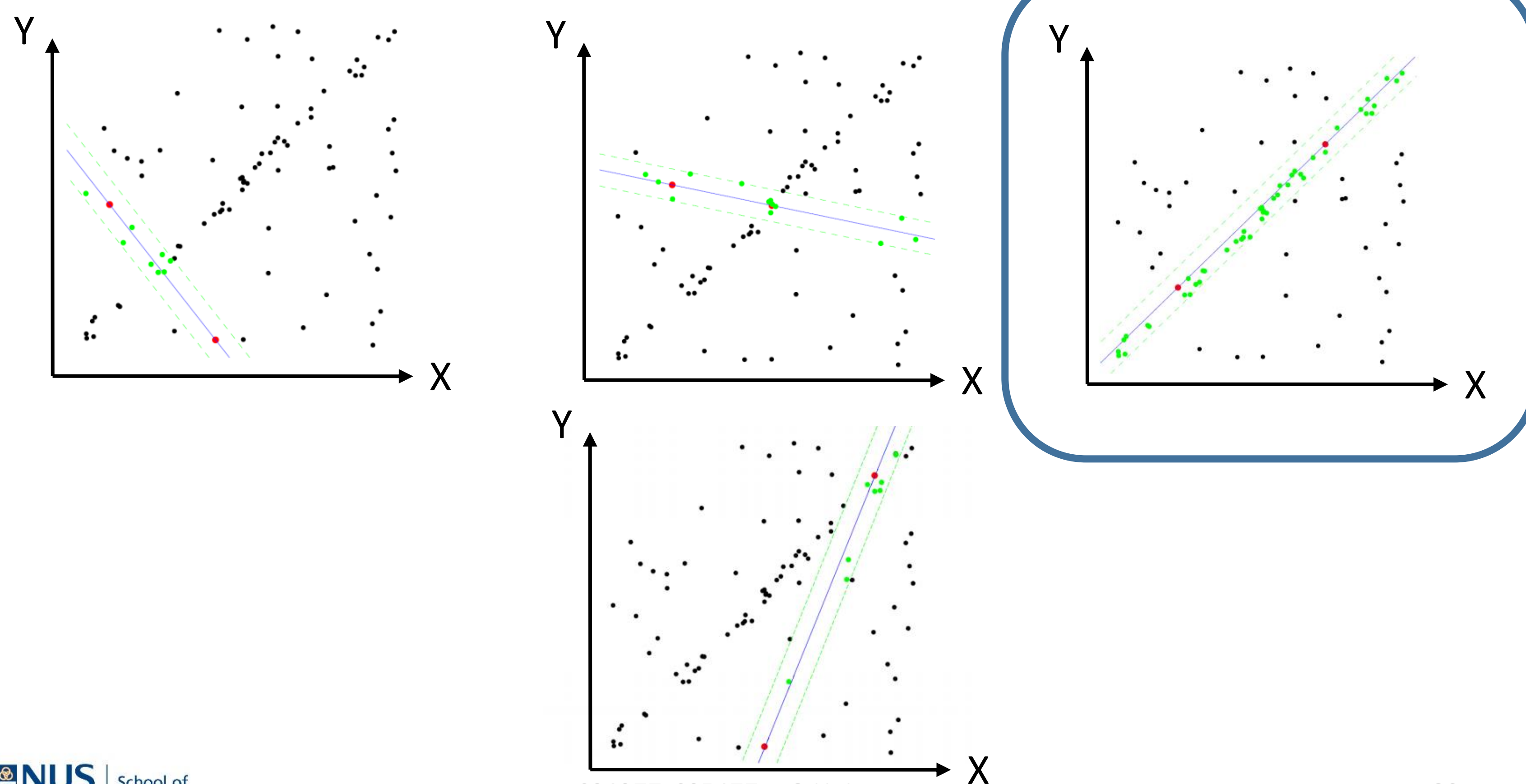
5. Repeat hypothesize-and-verify loop



RANSAC: Line Fitting Example

RANSAC Steps:

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



RANSAC Algorithm

Objective

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

Algorithm

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:
  - 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 $t^2=3.84\sigma^2$ if the measurement error, i.e., zero-mean Gaussian noise with std. dev. Σ is known.
- Number of samples, N:
  - Exhaustive search of all sample is often unnecessary and infeasible.

Choosing the Parameters

- Number of samples, *N*

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

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.

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 $w = \frac{\#\text{inliers}}{\#\text{points}}$
3. $\mathrm{N} = \frac{\log(1-\mathrm{p})}{\log(1-\mathrm{w}^{\mathrm{s}})}$ with *p=0.99*
4. Increment sample_count by 1

Terminate

Robust 2D Homography Computation

Objective

Compute the 2D homography between two images.

Algorithm

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:

a. Select a random sample of 4 correspondences and compute the homography, H.
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 $d < t$

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. Explain the concepts of SE(3) group and use it to describe rigid body motions in the 3D space.

2. Show the existence of homography.

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.