University of Zurich[UZH] | ETH zürich

ROBOTICS & PERCEPTION GROUP

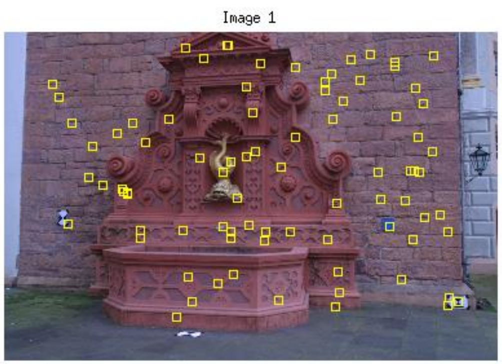
Institute of Informatics – Institute of Neuroinformatics

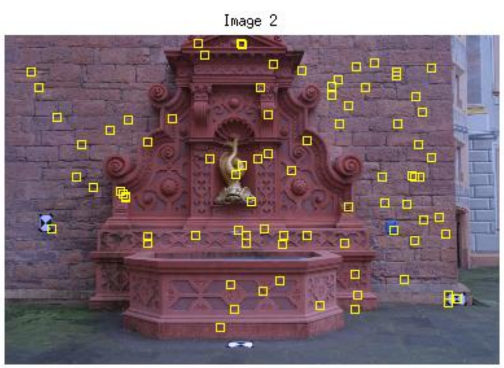
Lecture 08
Multiple View Geometry 2

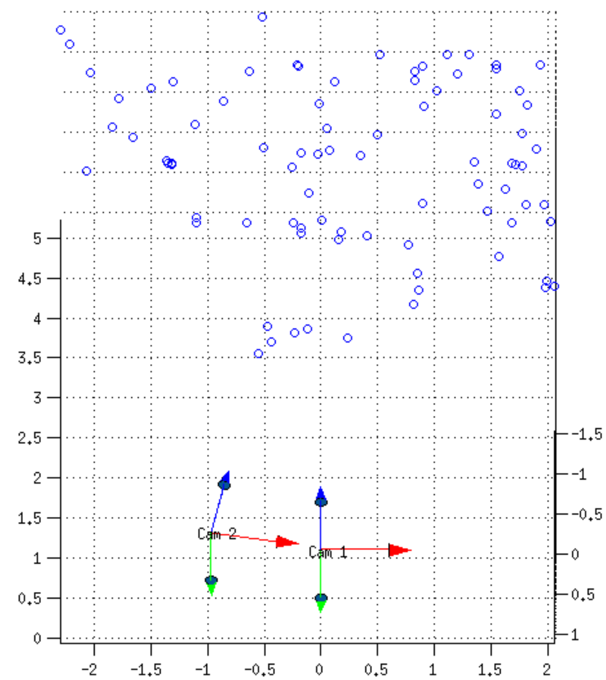
Davide Scaramuzza

Lab Exercise 5 - Today afternoon

- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: 8-point algorithm



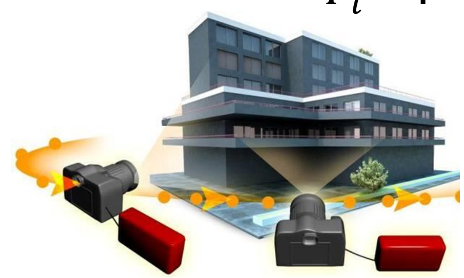




Estimated poses and 3D structure

2-View Geometry: Recap

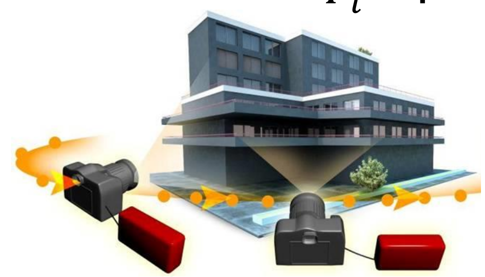
$P_i = ?$



K_1, R_1, T_1 K_2, R_2, T_2

- **Depth from stereo (i.e., stereo vision)**
  - **Assumptions:** K, T and R are known.
  - **Goal**: Recover the 3D structure from images
- **2-view Structure From Motion:**
  - **Assumptions**: none (K, T, and R are unknown).
  - **Goal**: Recover simultaneously 3D scene structure, camera poses (up to scale), and intrinsic parameters from two different views of the scene

$P_i = ?$



$K_1, R_1, T_1 = ?$ $K_2, R_2, T_2 = ?$

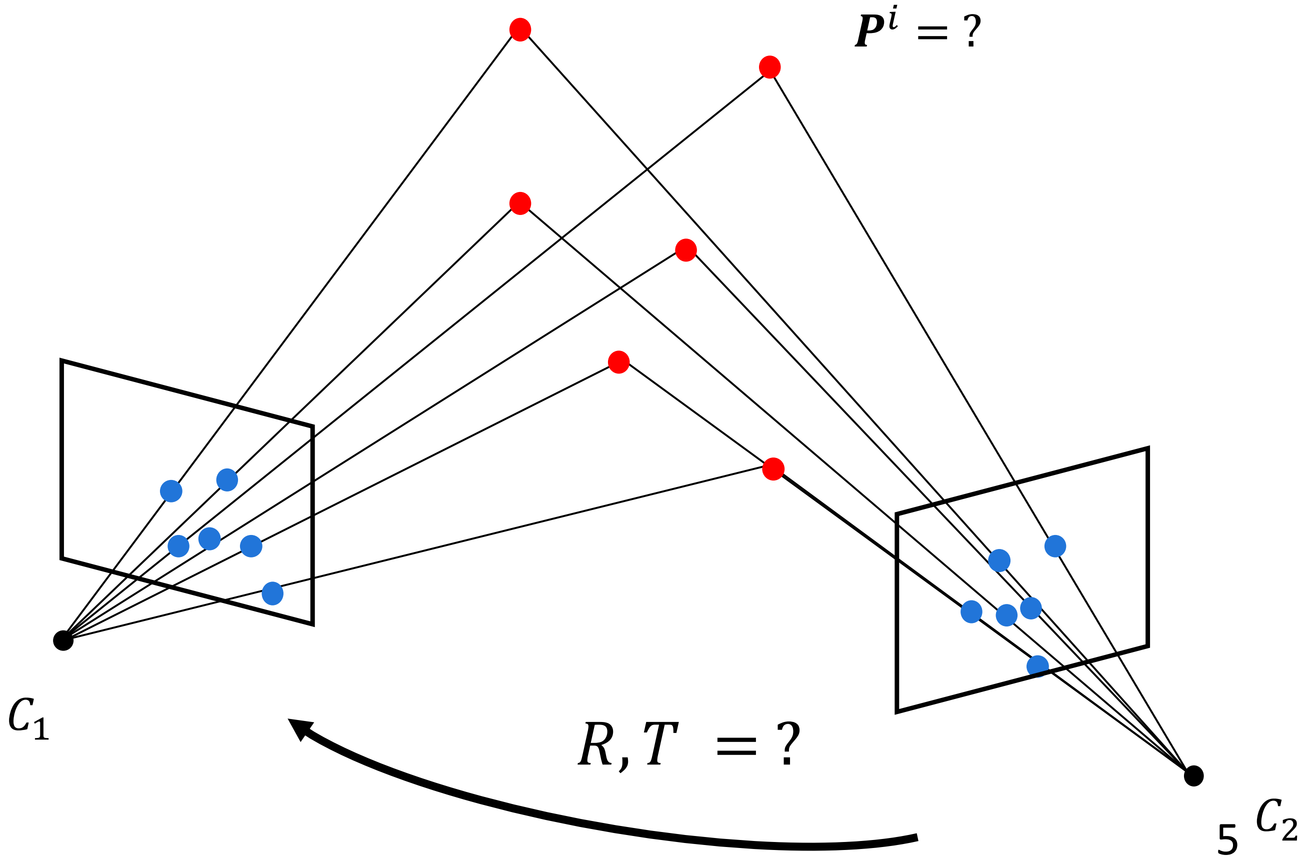
Outline

- Two-View Structure from Motion
- Robust Structure from Motion

Structure from Motion (SFM)

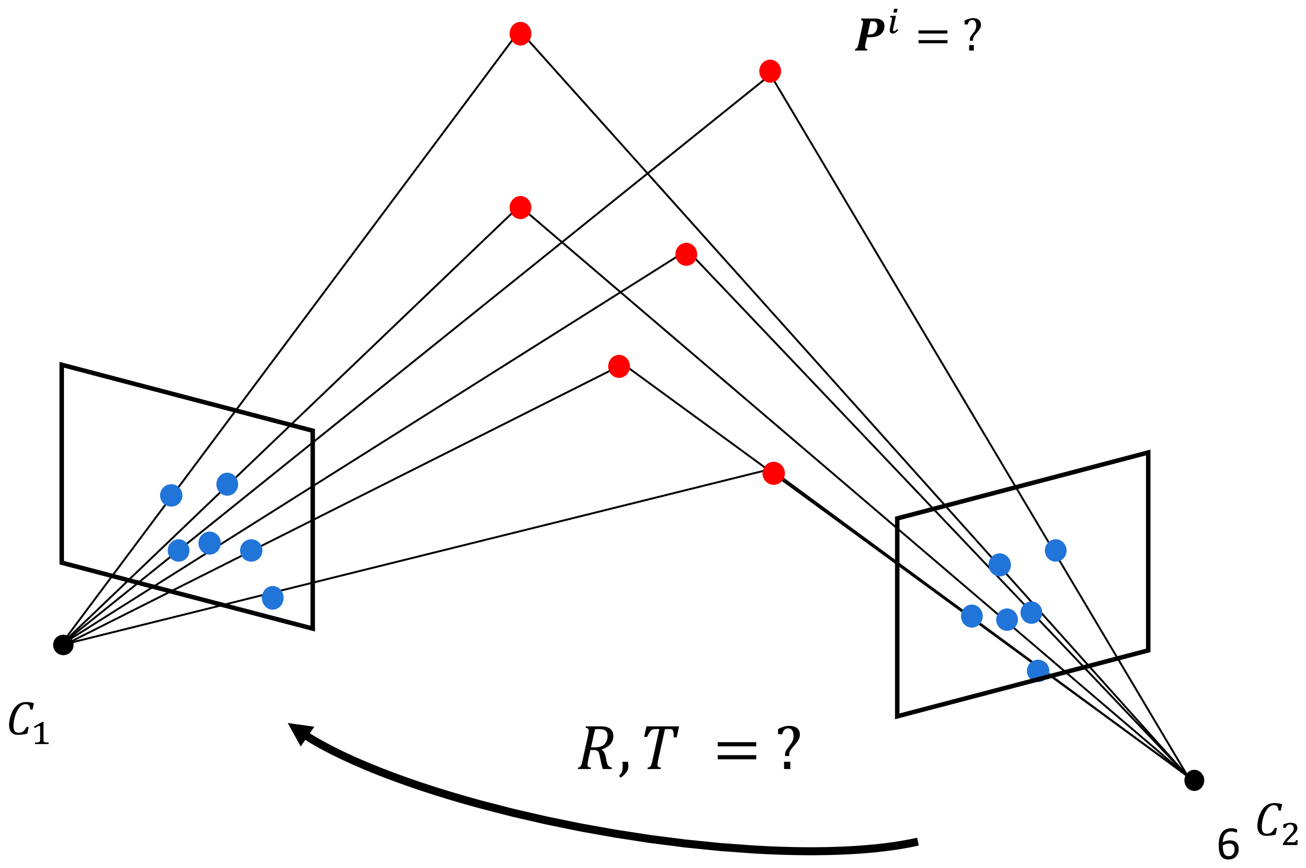
- **Problem formulation:** Given n point *correspondences* between two images, $\{p^i{}_1 = (u^i{}_1, v^i{}_1),\ p^i{}_2 = (u^i{}_2, v^i{}_2)\}$, simultaneously estimate the 3D points $\boldsymbol{P}^i$, the camera relative-motion parameters $(\boldsymbol{R}, \boldsymbol{T})$, and the camera intrinsics $\boldsymbol{K}_1, \boldsymbol{K}_2$ that satisfy:

$$
\begin{cases}
\lambda_1 \begin{bmatrix} u^i{}_1 \\ v^i{}_1 \\ 1 \end{bmatrix} = K_1 \left[I \middle| 0 \right] \cdot \begin{bmatrix} X^i{}_w \\ Y^i{}_w \\ Z^i{}_w \\ 1 \end{bmatrix} \\
\lambda_2 \begin{bmatrix} u^i{}_2 \\ v^i{}_2 \\ 1 \end{bmatrix} = K_2 \left[R \middle| T \right] \cdot \begin{bmatrix} X^i{}_w \\ Y^i{}_w \\ Z^i{}_w \\ 1 \end{bmatrix}
\end{cases}
$$



Structure from Motion (SFM)

- Two variants exist:
  - **Calibrated** camera(s) ⇒ $\boldsymbol{K_1}, \boldsymbol{K_2}$ **are known**
  - **Uncalibrated** camera(s) ⇒ $\boldsymbol{K_1}, \boldsymbol{K_2}$ **are unknown**

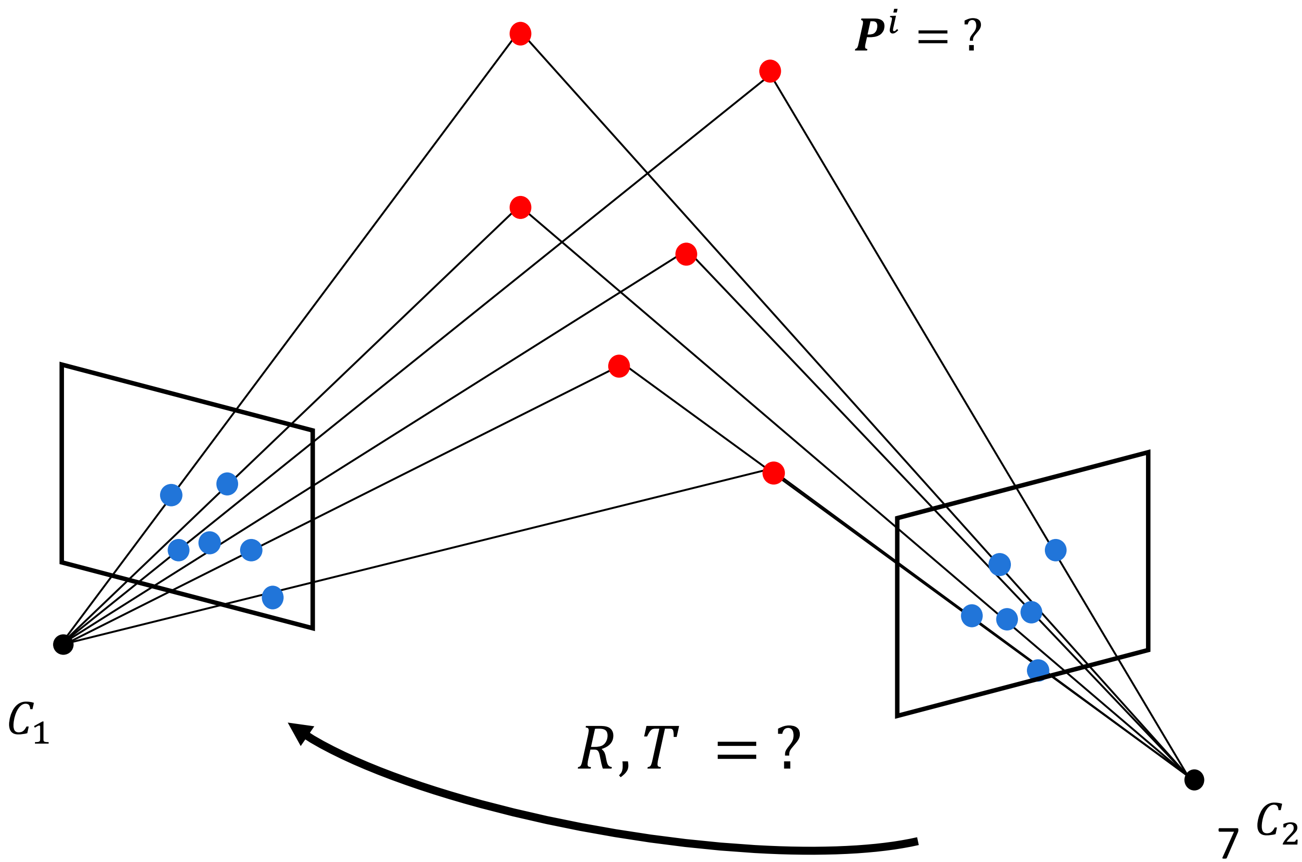


Structure from Motion (SFM)

- Let's study the case in which the cameras are **calibrated**
- For convenience, let's use *normalized image coordinates*
- Thus, we want to find $\boldsymbol{R}, \boldsymbol{T}, \boldsymbol{P}^i$ that satisfy

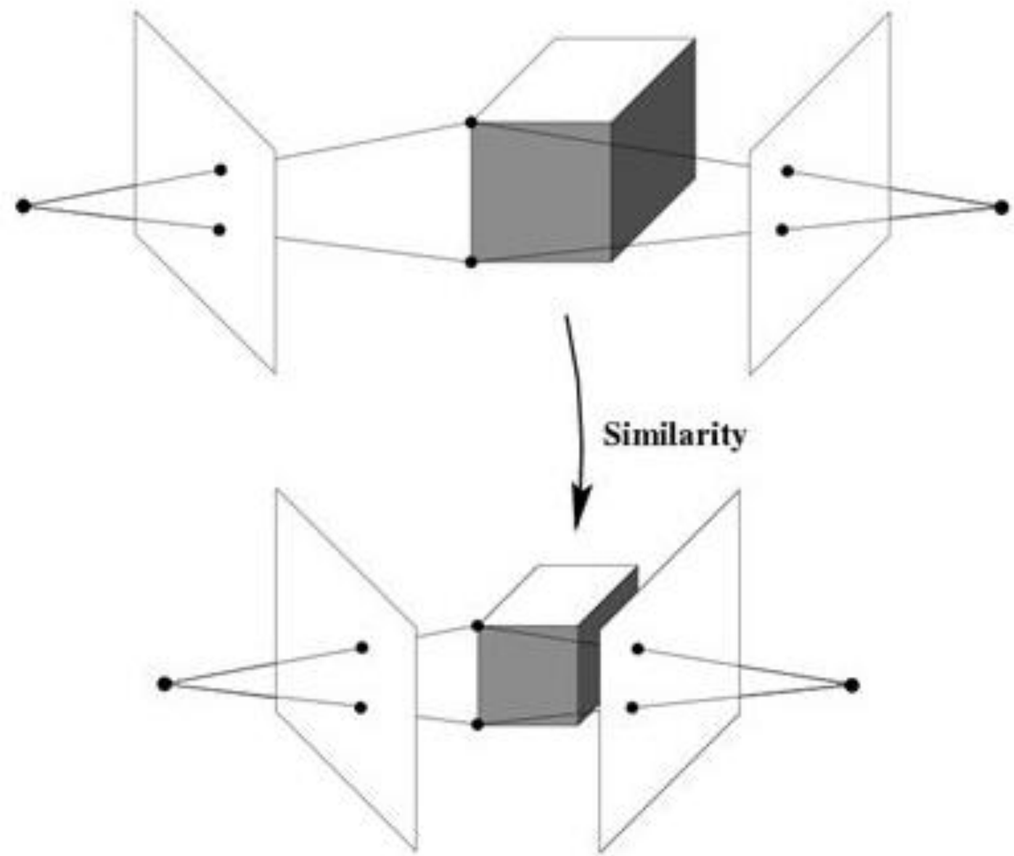
$$\begin{bmatrix} \bar{u} \\ \bar{v} \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$\begin{cases} \lambda_1 \begin{bmatrix} \bar{u}^i{}_1 \\ \bar{v}^i{}_1 \\ 1 \end{bmatrix} = [I|0] \cdot \begin{bmatrix} X^i{}_w \\ Y^i{}_w \\ Z^i{}_w \\ 1 \end{bmatrix} \\ \lambda_2 \begin{bmatrix} \bar{u}^i{}_2 \\ \bar{v}^i{}_2 \\ 1 \end{bmatrix} = [R|T] \cdot \begin{bmatrix} X^i{}_w \\ Y^i{}_w \\ Z^i{}_w \\ 1 \end{bmatrix} \end{cases}$$



Scale Ambiguity

If we rescale the entire scene by a constant factor (i.e., similarity transformation), the projections (in pixels) of the scene points in both images remain exactly the same:



Scale Ambiguity

- In monocular vision, it is therefore **not possible** to recover the absolute scale of the scene!
  - Stereo vision?
- Thus, only **5 degrees of freedom** are measurable:
  - **3** parameters to describe the **rotation**
  - **2** parameters for the **translation up to a scale** (we can only compute the direction of translation but not its length)

Structure From Motion (SFM)

- How many knowns and unknowns?
  - $\mathbf{4n}$ **knowns:**
    - n correspondences; each one $(u^i{}_1, v^i{}_1)$ and $(u^i{}_2, v^i{}_2)$, $i = 1 \dots n$
  - $\mathbf{5 + 3n}$ **unknowns**
    - 5 for the motion up to a scale (rotation-> 3, translation->2)
    - $3n =$ number of coordinates of the n 3D points

- Does a solution exist?
  - If and only if the *number of independent equations* ≥ *number of unknowns* $\Rightarrow 4n \geq 5 + 3n \Rightarrow \boxed{\mathbf{n \geq 5}}$
  - First analytical solution for 5 points by Kruppa in 1913. The equations yield to a 10 degree order polynomial, which has up to 10 solutions including complex ones.

E. Kruppa, Zur Ermittlung eines Objektes aus zwei Perspektiven mit Innerer Orientierung, *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. IIa.*, 1913. [PDF]

Cross Product (or Vector Product)

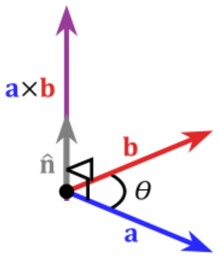
$$\vec{a} \times \vec{b} = \vec{c}$$

- Vector cross product takes two vectors and returns a third vector that is perpendicular to both inputs, with a direction given by the right-hand rule and a magnitude equal to the area of the parallelogram that the vectors span:

$$\vec{a} \cdot \vec{c} = 0$$

$$\vec{b} \cdot \vec{c} = 0$$

$$\|\vec{c}\| = \|\vec{a}\|\|\vec{b}\| \sin(\theta)$$

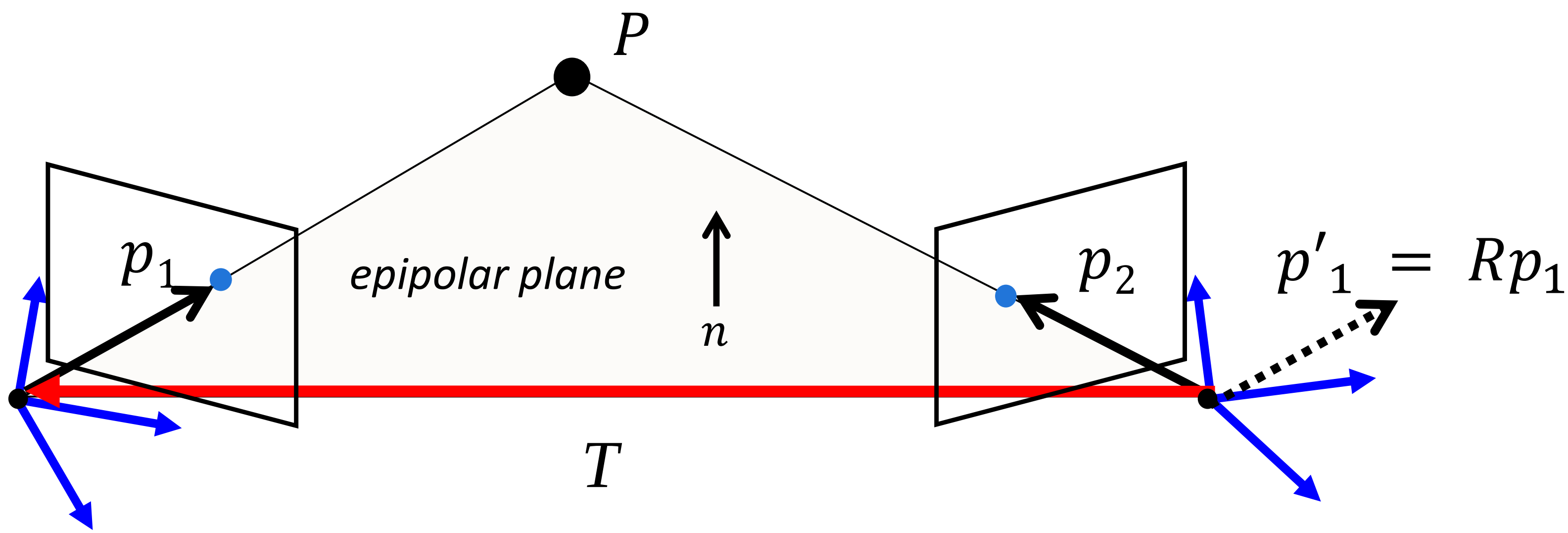


- So $\boldsymbol{c}$ is perpendicular to both $\boldsymbol{a}$ and $\boldsymbol{b}$ (which means that the dot product is 0)
- Also, recall that the cross product of two parallel vectors is 0
- The **cross product** between **a** and **b** can also be expressed in matrix form as the product between the **skew-symmetric matrix** of **a** and a vector **b**

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}]_\times \mathbf{b}$$

Epipolar Geometry

$$\bar{p}_1 = \begin{bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{bmatrix} \qquad \bar{p}_2 = \begin{bmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{bmatrix}$$



p_1, p_2, T are coplanar:

$$p_2^T \cdot n = 0 \;\Rightarrow\; p_2^T \cdot (T \times p_1') = 0 \;\Rightarrow\; p_2^T \cdot (T \times (Rp_1)) = 0$$

$$\Rightarrow\; p_2^T \, [T]_\times R \; p_1 = 0 \;\Rightarrow\; \boxed{p_2^T \; E \; p_1 = 0 \quad \textit{epipolar constraint}}$$

$$\boxed{\mathrm{E} = [T]_\times R \qquad \textit{essential matrix}}$$

Epipolar Geometry

$$\bar{p}_1 = \begin{bmatrix} \bar{u}_1 \\ \bar{v}_1 \\ 1 \end{bmatrix} \quad \bar{p}_2 = \begin{bmatrix} \bar{u}_2 \\ \bar{v}_2 \\ 1 \end{bmatrix} \quad \textit{Normalized image coordinates}$$

$$\bar{p}_2^T \; E \; \bar{p}_1 = 0 \qquad \textit{Epipolar constraint or Longuet-Higgins equation (1981)}$$

$$\mathrm{E} = [T]_\times R \qquad \textit{Essential matrix}$$

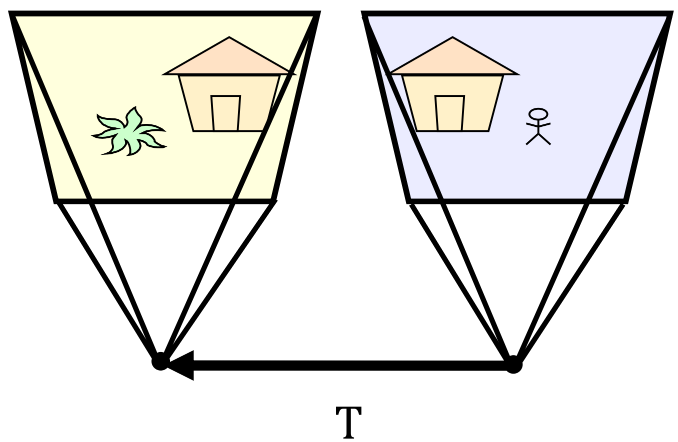
- The Essential Matrix can be decomposed into R and T recalling that $\mathrm{E} = [T]_\times R$ Four distinct solutions for R and T are possible.

H. Christopher Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, Nature, 1981, PDF.

Exercise

- Compute the Essential matrix for the case of two rectified stereo images

Rectified case



$$\mathbf{R} = \mathbf{I}_{3\times 3}$$

$$\mathrm{T} = \begin{bmatrix} -b \\ 0 \\ 0 \end{bmatrix} \rightarrow [\mathrm{T}]_{\times} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix} \qquad \rightarrow E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix}$$

How to compute the Essential Matrix?

Image 1

Image 2

- If we don't know **R** and **T**, can we estimate **E** from two images?
- Yes, given at least 5 correspondences

How to compute the Essential Matrix?

- Kruppa showed in 1913 that 5 image correspondences is the minimal case. However, his solution was not efficient.
- In 1996, Philipp proposed an iterative solution
- Only in 2004, the first efficient and non iterative solution was proposed. It uses Groebner basis decomposition [Nister, CVPR'2004]..

- The first popular solution uses 8 points and is called **the 8-point algorithm** or **Longuet-Higgins algorithm** (1981). Because of its ease of implementation, it is still used today (e.g., NASA rovers).

H. Christopher Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, Nature, 1981, PDF.

The 8-point algorithm

- The Essential matrix E is defined by

$$\bar{p}_2^T \; E \; \bar{p}_1 = 0$$

- Each pair of point correspondences $\bar{\boldsymbol{p}}_{\boldsymbol{1}} = (\bar{u}_1, \bar{v}_1, 1)^T, \quad \bar{\boldsymbol{p}}_{\boldsymbol{2}} = (\bar{u}_2, \bar{v}_2, 1)$

  provides a linear equation:

$$\bar{p}_2^T \; E \; \bar{p}_1 = 0 \qquad\qquad E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}$$

$$\bar{u}_2\bar{u}_1 e_{11} + \quad \bar{u}_2\bar{v}_1 e_{12} + \quad \bar{u}_2 e_{13} + \quad \bar{v}_2\bar{u}_1 e_{21} + \quad \bar{v}_2\bar{v}_1 e_{22} + \quad \bar{v}_2 e_{23} + \quad \bar{u}_1 e_{31} + \quad \bar{v}_1 e_{32} + \quad e_{33} = 0$$

The 8-point algorithm

- For n points, we can write

$$\underbrace{\begin{bmatrix} \bar{u}_2^{1}\bar{u}_1^{1} & \bar{u}_2^{1}\bar{v}_1^{1} & \bar{u}_2^{1} & \bar{v}_2^{1}\bar{u}_1^{1} & \bar{v}_2^{1}\bar{v}_1^{1} & \bar{v}_2^{1} & \bar{u}_1^{1} & \bar{v}_1^{1} & 1 \\ \bar{u}_2^{2}\bar{u}_1^{2} & \bar{u}_2^{2}\bar{v}_1^{2} & \bar{u}_2^{2} & \bar{v}_2^{2}\bar{u}_1^{2} & \bar{v}_2^{2}\bar{v}_1^{2} & \bar{v}_2^{2} & \bar{u}_1^{2} & \bar{v}_1^{2} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{u}_2^{n}\bar{u}_1^{n} & \bar{u}_2^{n}\bar{v}_1^{n} & \bar{u}_2^{n} & \bar{v}_2^{n}\bar{u}_1^{n} & \bar{v}_2^{n}\bar{v}_1^{n} & \bar{v}_2^{n} & \bar{u}_1^{n} & \bar{v}_1^{n} & 1 \end{bmatrix}}_{\text{Q (this matrix is \textbf{known})}} \underbrace{\begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{bmatrix}}_{\bar{E} \text{ (this matrix is \textbf{unknown})}} = 0$$

The 8-point algorithm

$$Q \cdot \overline{E} = 0$$

Minimal solution

- $Q_{(n \times 9)}$ should have rank 8 to have a unique (up to a scale) non-trivial solution $\bar{E}$
- Each point correspondence provides 1 independent equation
- Thus, 8 point correspondences are needed

Over-determined solution

- *n* > 8 points
- A solution is to minimize $||Q\bar{E}||^2$ subject to the constraint $||\bar{E}||^2 = 1$.
  The solution is the eigenvector corresponding to the smallest eigenvalue of the matrix $Q^T Q$ (because it is the unit vector x that minimizes $||Qx||^2 = x^T Q^T Q x$).
- It can be solved through Singular Value Decomposition (SVD). Matlab instructions:
  - `[U,S,V] = svd(Q);`
  - `Ev = V(:,9);`
  - `E = reshape(Ev,3,3)';`

- **Degenerate Configurations**
  - The solution of the eight-point algorithm is degenerate when the 3D points are coplanar. Conversely, the five-point algorithm works also for coplanar points

8-point algorithm: Matlab code

- A few lines of code. Go to the exercise this afternoon to learn to implement it ☺

8-point algorithm: Matlab code

```
function E = calibrated_eightpoint( p1, p2)

p1 = p1'; % 3xN vector; each column = [u;v;1]
p2 = p2'; % 3xN vector; each column = [u;v;1]

Q = [p1(:,1).*p2(:,1) , ...
     p1(:,2).*p2(:,1) , ...
     p1(:,3).*p2(:,1) , ...
     p1(:,1).*p2(:,2) , ...
     p1(:,2).*p2(:,2) , ...
     p1(:,3).*p2(:,2) , ...
     p1(:,1).*p2(:,3) , ...
     p1(:,2).*p2(:,3) , ...
     p1(:,3).*p2(:,3) ] ;

[U,S,V] = svd(Q);
Eh = V(:,9);

E = reshape(Eh,3,3)';
```

Interpretation of the 8-point algorithm

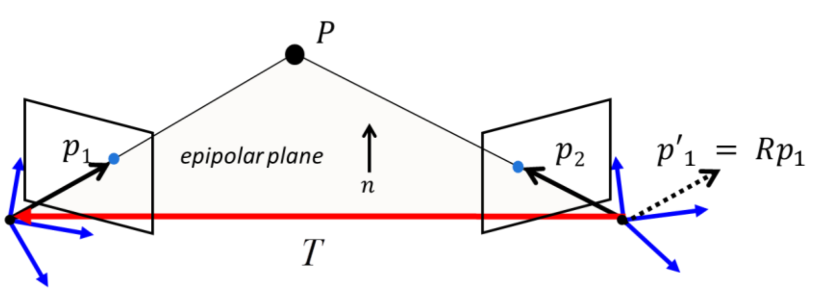
The 8-point algorithm seeks to minimize the following **algebraic error**

$$\sum_{i=1}^{N} (\overline{p}^{i\,T}_{2} \boldsymbol{E}\, \overline{p}^{i}_{1})^2$$

Using the definition of dot product, it can be observed that

$$\overline{\boldsymbol{p}}_{\boldsymbol{2}}^{\top} \cdot \boldsymbol{E}\boldsymbol{p}_1 = \|\boldsymbol{p}_{\boldsymbol{2}}\| \|\boldsymbol{E}\boldsymbol{p}_1\| \cos(\theta)$$

We can see that this product depends on the angle θ between $\boldsymbol{p}_1$ and the normal $\boldsymbol{E}\boldsymbol{p}_1$ to the epipolar plane. It is non zero when $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{T}$ are not coplanar.



Extract R and T from E
(this slide will not be asked at the exam)

- Singular Value Decomposition: $E = U \sum V^T$
- Enforcing rank-2 constraint: set smallest singular value of $\sum$ to 0:

$$\sum = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \cancel{\sigma_3} \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

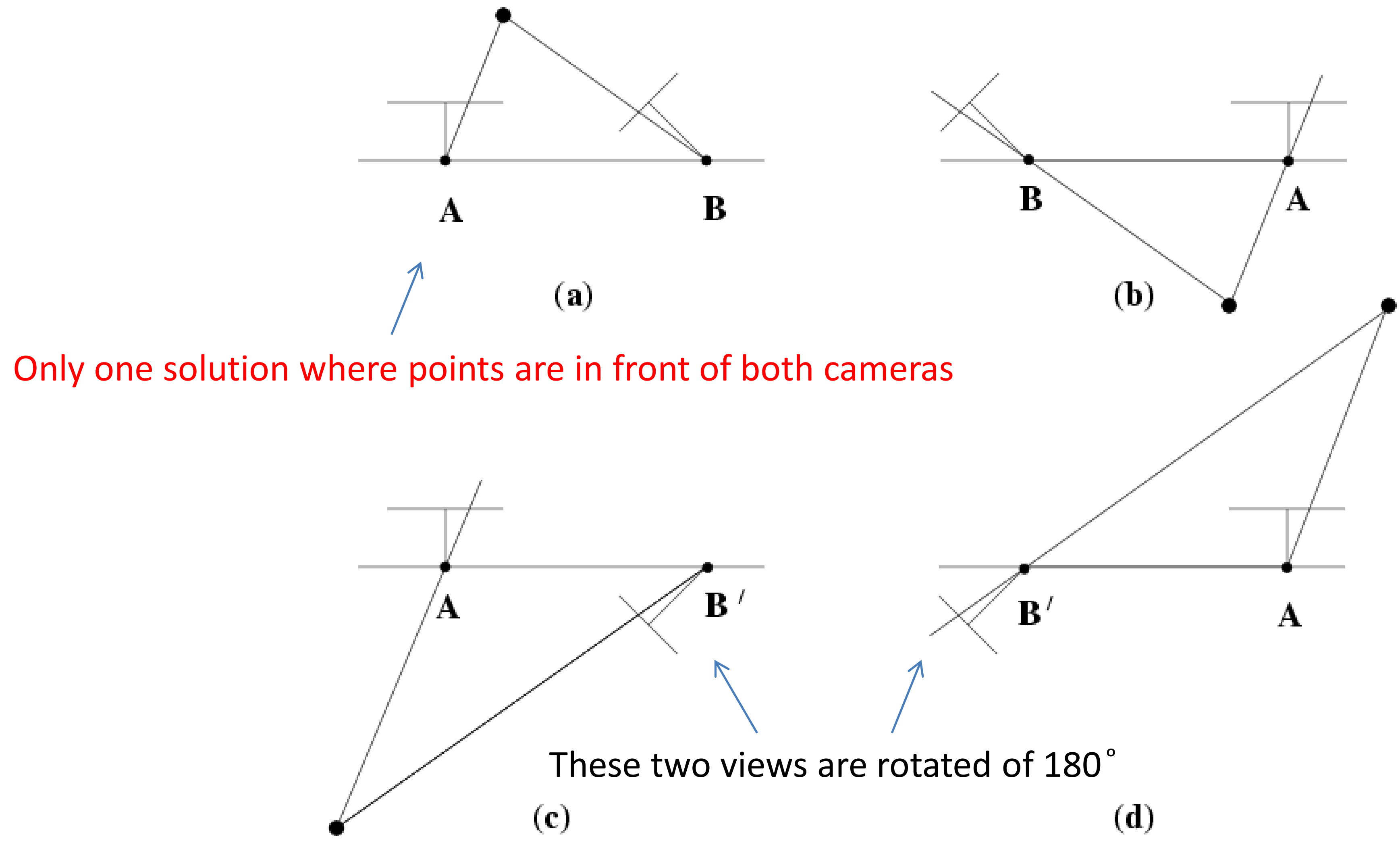
$$\hat{T} = U \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \sum V^T$$

$$\hat{T} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & t_x \\ -t_y & t_x & 0 \end{bmatrix} \Rightarrow \hat{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

$$\hat{R} = U \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} V^T$$

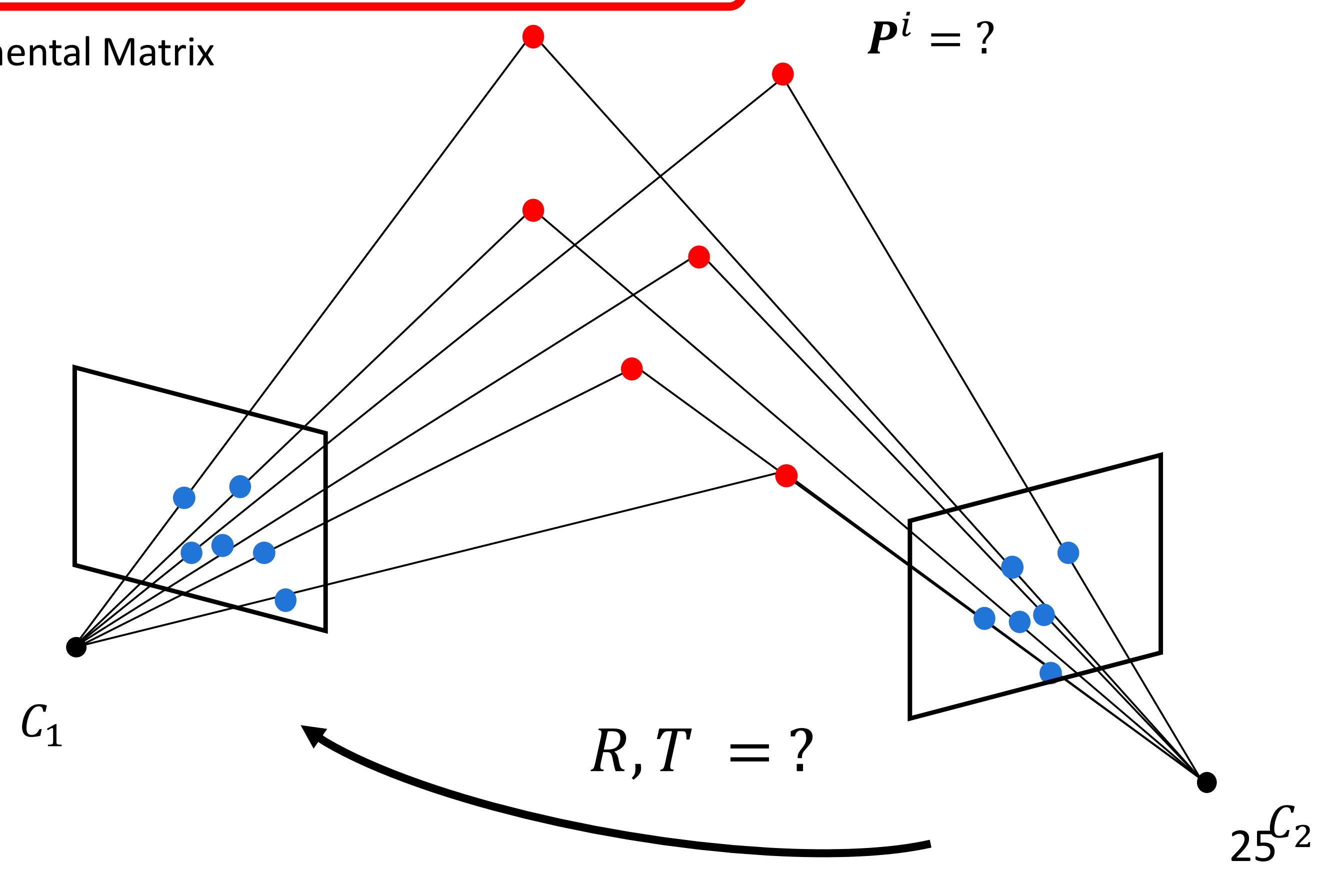
$$\boxed{\begin{aligned} t &= K_2 \hat{t} \\ R &= K_2 \hat{R} K_1^{-1} \end{aligned}}$$

4 possible solutions of R and T



Structure from Motion (SFM)

- Two variants exist:
  - **Calibrated** camera(s) ⇒ $\boldsymbol{K_1, K_2}$ **are known**
    - Uses the Essential Matrix
  - **Uncalibrated** camera(s) ⇒ $\boldsymbol{K_1, K_2}$ **are unknown**
    - Uses the Fundamental Matrix



The Fundamental Matrix

- Before, we assumed to know the camera intrinsic parameters and we used normalized image coordinates

$$\bar{p}_2^T \; E \; \bar{p}_1 = 0$$

$$\begin{bmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{bmatrix}^T E \begin{bmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{bmatrix} = K_1^{-1} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{bmatrix} = K_2^{-1} \begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}^T K_2^{-T} \; E \; K_1^{-1} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}^T \boxed{F} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = 0$$

Fundamental Matrix

$$\left.\begin{aligned} F &= K_2^{-T} \; E \; K_1^{-1} \\ E &= [T]_\times R \end{aligned}\right\} \Rightarrow \boxed{F = K_2^{-T} \, [T]_\times R \; K_1^{-1}}$$

The 8-point Algorithm for the Fundamental Matrix

- The same 8-point algorithm to compute the essential matrix from a set of normalized image coordinates can also be used to determine the Fundamental matrix

$$\begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}^{\mathrm{T}} \mathrm{F} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = 0$$

Problem with 8-point algorithm

$$\begin{bmatrix} u_2^1 u_1^1 & u_2^1 v_1^1 & u_2^1 & v_2^1 u_1^1 & v_2^1 v_1^1 & v_2^1 & u_1^1 & v_1^1 & 1 \\ u_2^2 u_1^2 & u_2^2 v_1^2 & u_2^2 & v_2^2 u_1^2 & v_2^2 v_1^2 & v_2^2 & u_1^2 & v_1^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_2^n u_1^n & u_2^n v_1^n & u_2^n & v_2^n u_1^n & v_2^n v_1^n & v_2^n & u_1^n & v_1^n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Problem with 8-point algorithm

250906.36	183269.57	921.81	200931.10	146766.13	738.21	272.19	198.81	1.00
2692.28	131633.03	176.27	6196.73	302975.59	405.71	15.27	746.79	1.00
416374.23	871684.30	935.47	408110.89	854384.92	916.90	445.10	931.81	1.00
191183.60	171759.40	410.27	416435.62	374125.90	893.65	465.99	418.65	1.00
48988.86	30401.76	57.89	298604.57	185309.58	352.87	846.22	525.15	1.00
164786.04	546559.67	813.17	1998.37	6628.15	9.86	202.65	672.14	1.00
116407.01	2727.75	138.89	169941.27	3982.21	202.77	838.12	19.64	1.00
135384.58	75411.13	198.72	411350.03	229127.78	603.79	681.28	379.48	1.00
~10000	~10000	~100	~10000	~10000	~100	~100	~100	1

$$\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

!

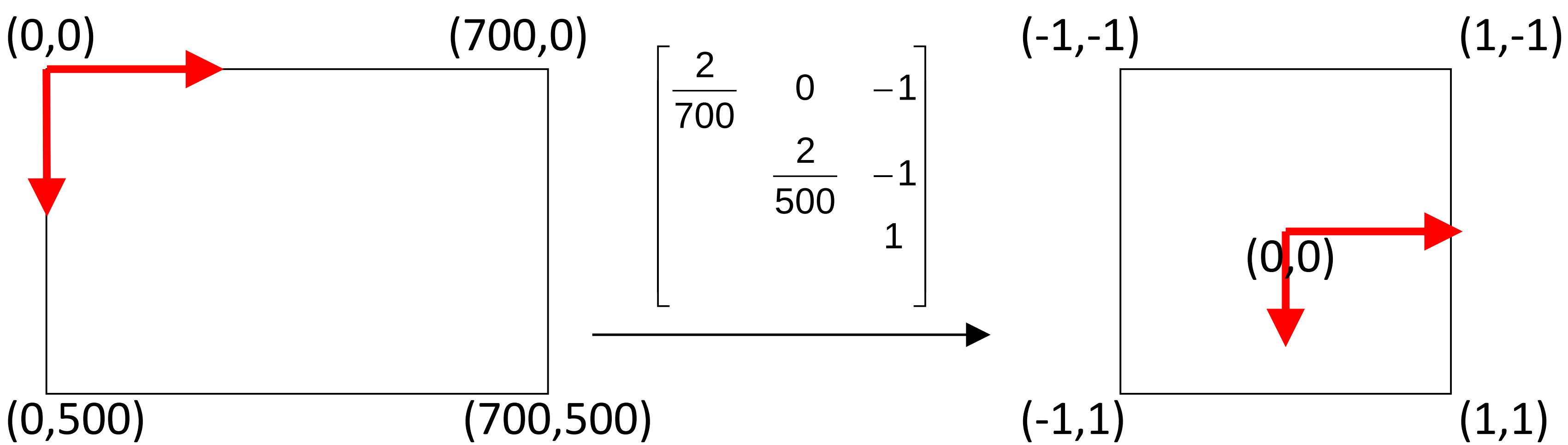
Orders of magnitude difference
between column of data matrix
$\rightarrow$ least-squares yields poor results

- Poor numerical conditioning, which makes results very sensitive to noise
- Can be fixed by rescaling the data: *Normalized 8-point algorithm* [Hartley, 1995]

Normalized 8-point algorithm (1/3)

- This can be fixed using a normalized 8-point algorithm, which estimates the Fundamental matrix on a set of **Normalized correspondences** (with better numerical properties) and **then unnormalizes** the result to obtain the fundamental matrix for the **given (unnormalized) correspondences**

- **Idea:** Transform image coordinates so that they are in the range $\sim[-1,1] \times [-1,1]$
- One way is to apply the following rescaling and shift



Normalized 8-point algorithm (2/3)

- A more popular way is to rescale the two point sets such that the centroid of each set is 0 and the mean standard deviation $\sqrt{2}$.
- This can be done for every point as follows:

$$\widehat{p^i} = \frac{\sqrt{2}}{\sigma}(p^i - \mu)$$

- Where $\mu = \frac{1}{N}\sum_{i=1}^{n} p^i$ is the centroid of the set and $\sigma = \frac{1}{N}\sum_{i=1}^{n}\left\|p^i - \mu\right\|^2$ is the mean standard deviation.
- This transformation can be expressed in matrix form using homogeneous coordinates:

$$\widehat{p^i} = \begin{bmatrix} \frac{\sqrt{2}}{\sigma} & 0 & -\frac{\sqrt{2}}{\sigma}\mu^x \\ 0 & \frac{\sqrt{2}}{\sigma} & -\frac{\sqrt{2}}{\sigma}\mu^y \\ 0 & 0 & 1 \end{bmatrix} p^i$$

Normalized 8-point algorithm (3/3)

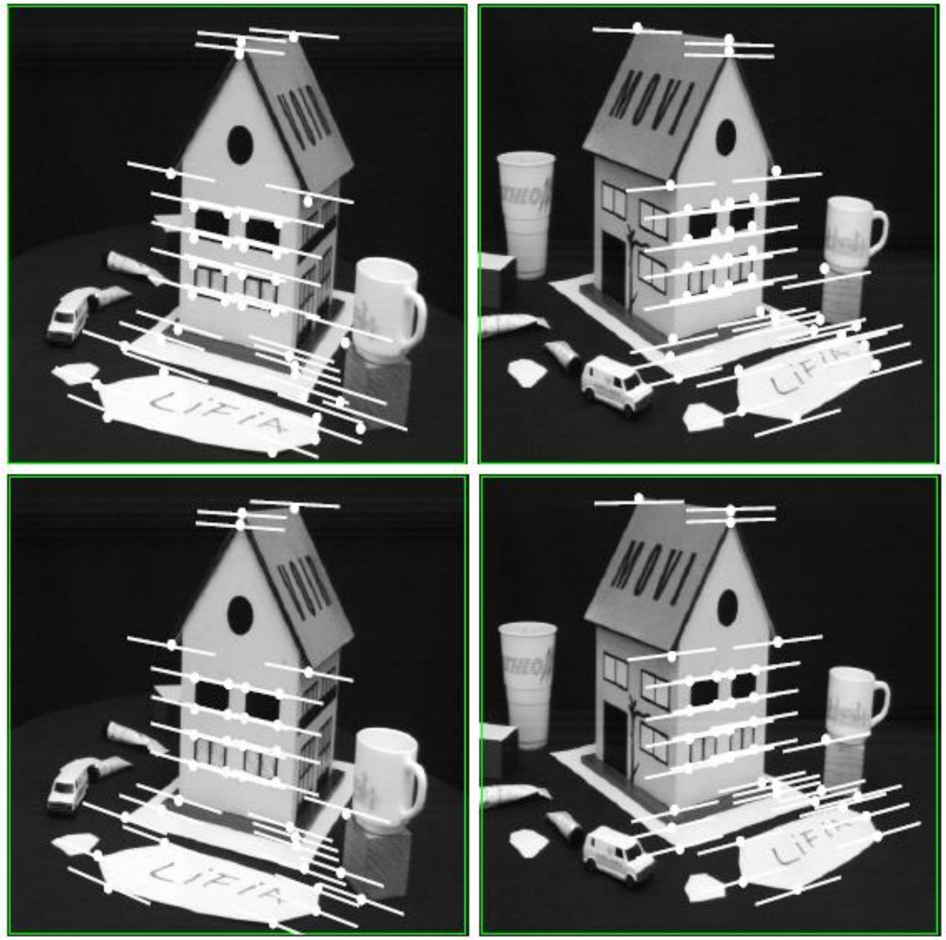
The Normalized 8-point algorithm can be summarized in three steps:

1. Normalize point correspondences: $\widehat{p_1} = B_1 p_1$, $\widehat{p_2} = B_2 p_2$
2. Estimate $\hat{F}$ using normalized coordinates $\widehat{p_1}$, $\widehat{p_2}$
3. Compute F from $\hat{F}$: $\mathrm{F} = \mathrm{B}_2^\top \hat{\mathrm{F}}\, \mathrm{B}_1$

$$\widehat{p_2}^{T} \hat{\mathrm{F}}\ \widehat{p_1} = 0$$

$$\underbrace{\boxed{B_2{}^\top p_2^\top}\quad \hat{\mathrm{F}}\quad \boxed{B_1{}^\top p_1^\top}}_{\mathrm{F} = \mathrm{B}_2^\top \hat{\mathrm{F}}\, \mathrm{B}_1}$$

Comparison between Normalized and non-normalized algorithm



	8-point	Normalized 8-point	Nonlinear refinement
Av. Reprojection error 1	2.33 pixels	0.92 pixel	0.86 pixel
Av. Reprojection error 2	2.18 pixels	0.85 pixel	0.80 pixel

Error Measures

- The quality of the estimated Fundamental matrix can be measured using different cost functions.
- The first one is the algebraic error that is defined directly in the Epipolar Constraint:

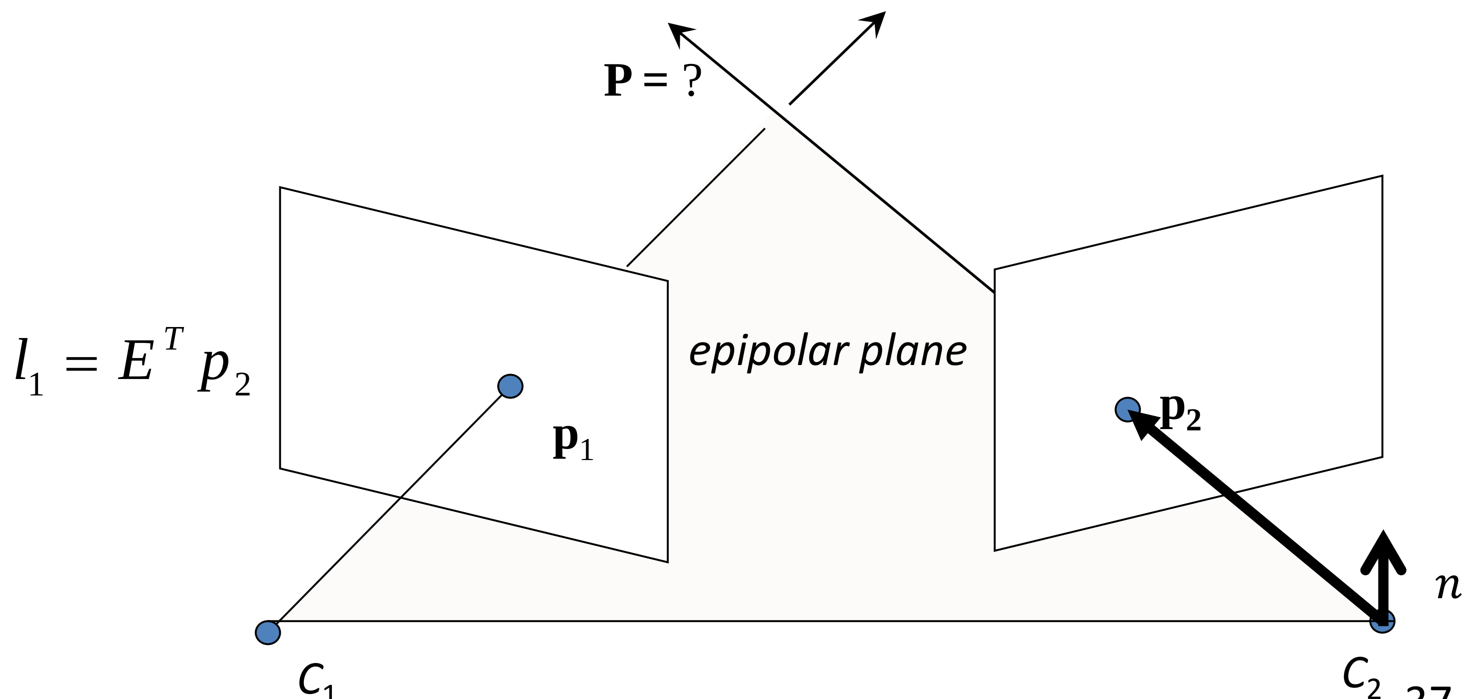
$$err = \sum_{i=1}^{N} \left(\overline{p}_2^{i\,T} \boldsymbol{E}\, \overline{p}_1^{i} \right)^2$$

Remember Slide 22 for the geometrical interpretation of this error
What is the drawback with this error measure?

- This error will exactly be 0 if E is computed from just 8 points (because in this case a solution exists). For more than 8 points, it will not be 0 (due to image noise or outliers (overdetermined system)).
- There are alternative error functions that can be used to measure the quality of the estimated Fundamental matrix: the **Directional Error**, the **Epipolar Line Distance**, or the **Reprojection Error**.

Directional Error

- Sum of the Angular Distances to the Epipolar plane: $\mathrm{err} = \sum_i (\cos(\theta_i))^2$
- From slide 22, we obtain: $\cos(\theta) = \left(\dfrac{\boldsymbol{p}^T{}_2 \cdot \boldsymbol{E}\boldsymbol{p}_1}{\|\boldsymbol{p}^T{}_2\| \, \|\boldsymbol{E}\boldsymbol{p}_1\|} \right)^2$

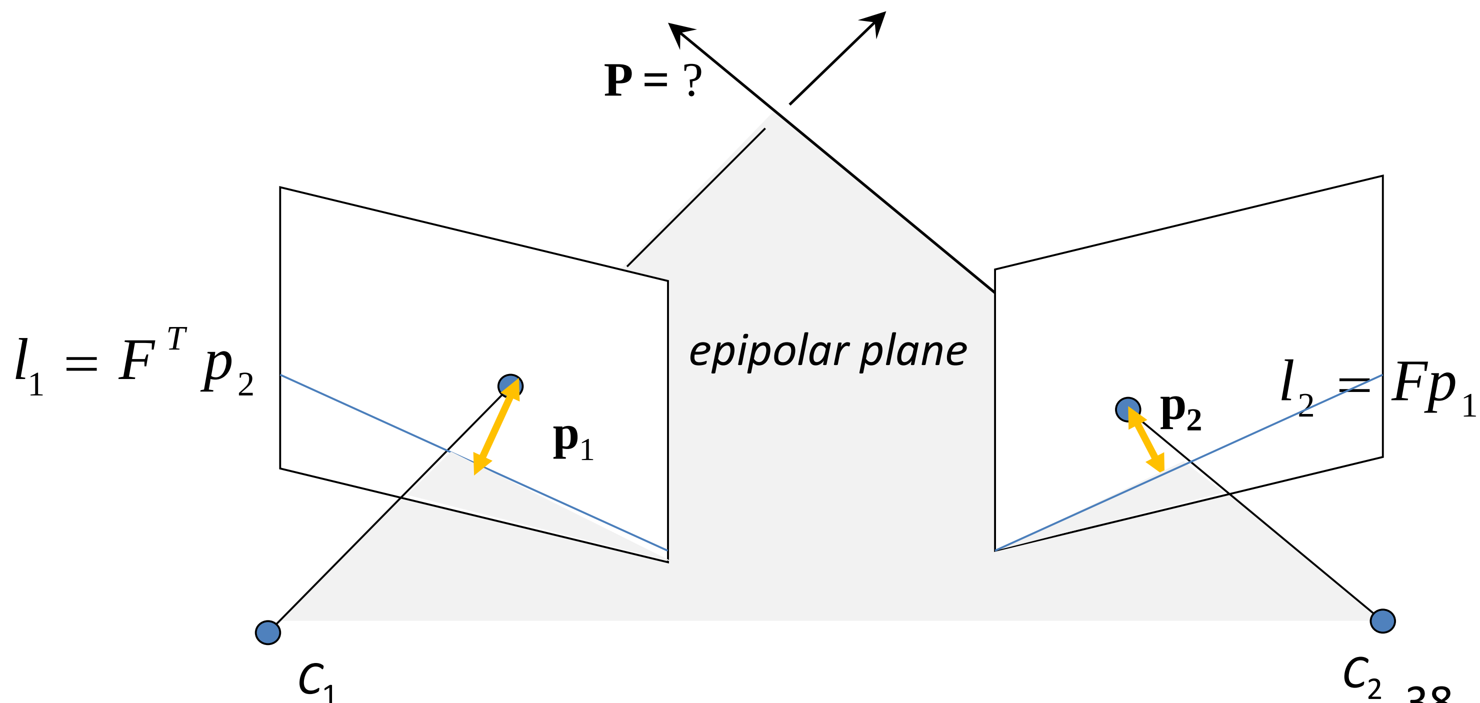


Epipolar Line Distance

- Sum of **Squared Epipolar-Line-to-point Distances**

$$err = \sum_{i=1}^{N} d^2(p_1^i, l_1^i) + d^2(p_2^i, l_2^i)$$

- Cheaper than reprojection error because does not require point triangulation

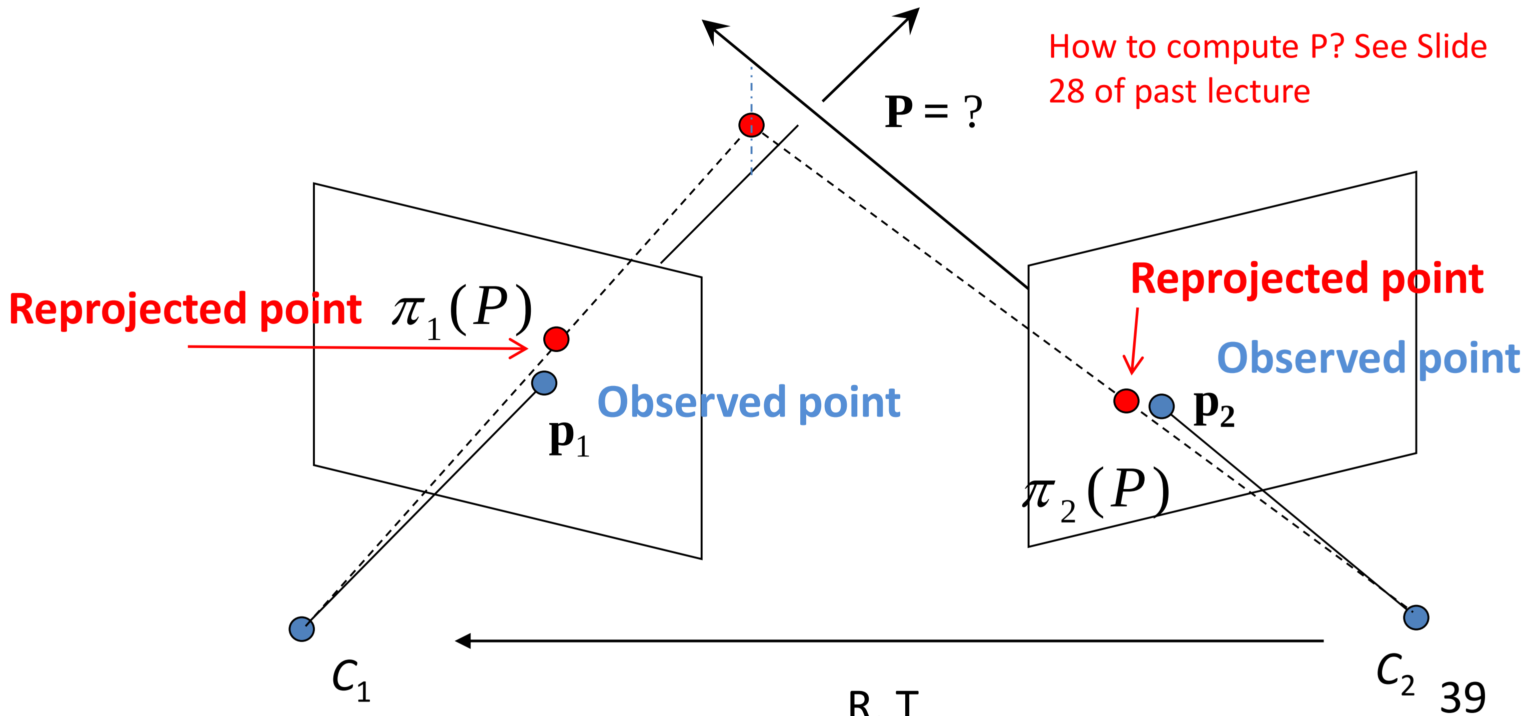


Reprojection Error

- Sum of the **Squared Reprojection Errors**

$$err = \sum_{i=1}^{N} \left\| p_1^i - \pi_1(P^i) \right\|^2 + \left\| p_2^i - \pi_2(P^i, R, T) \right\|^2$$

- Computation is expensive because requires point triangulation
- **However it is the most popular because more accurate**



Outline

- Two-View Structure from Motion
- Robust Structure from Motion

Robust Estimation

- Matched points are usually contaminated by **outliers** (i.e., wrong image matches)
- Causes of outliers are:
  - changes in view point (including scale) and illumination
  - image noise
  - occlusions
  - blur
- For the camera motion to be estimated accurately, outliers must be removed
- This is the task of **Robust Estimation**

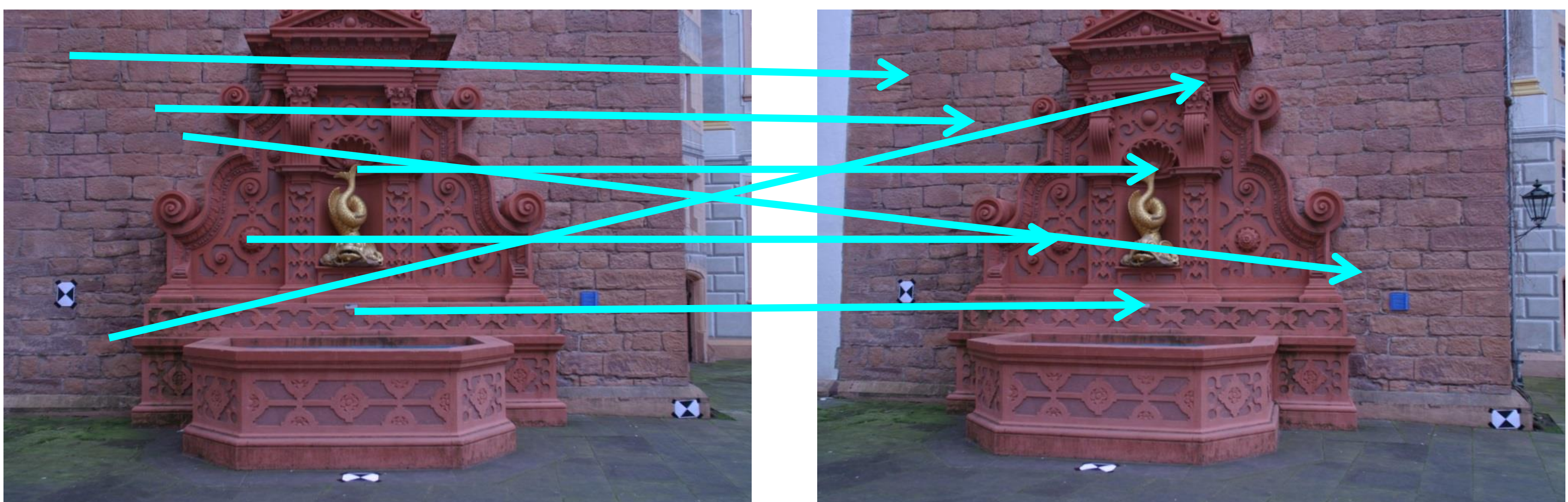


Image 1 Image 2

Robust Estimation

- Matched points are usually contaminated by **outliers** (i.e., wrong image matches)
- Causes of outliers are:
  - changes in view point (including scale) and illumination
  - image noise
  - occlusions
  - blur
- For the camera motion to be estimated accurately, outliers must be removed
- This is the task of **Robust Estimation**

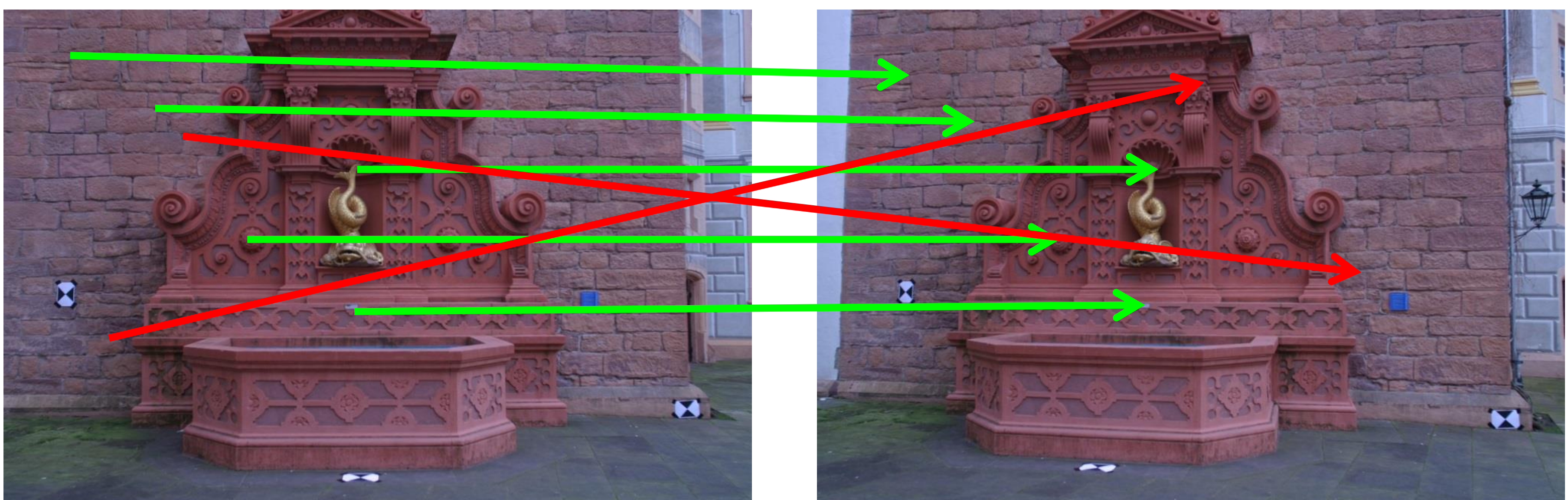
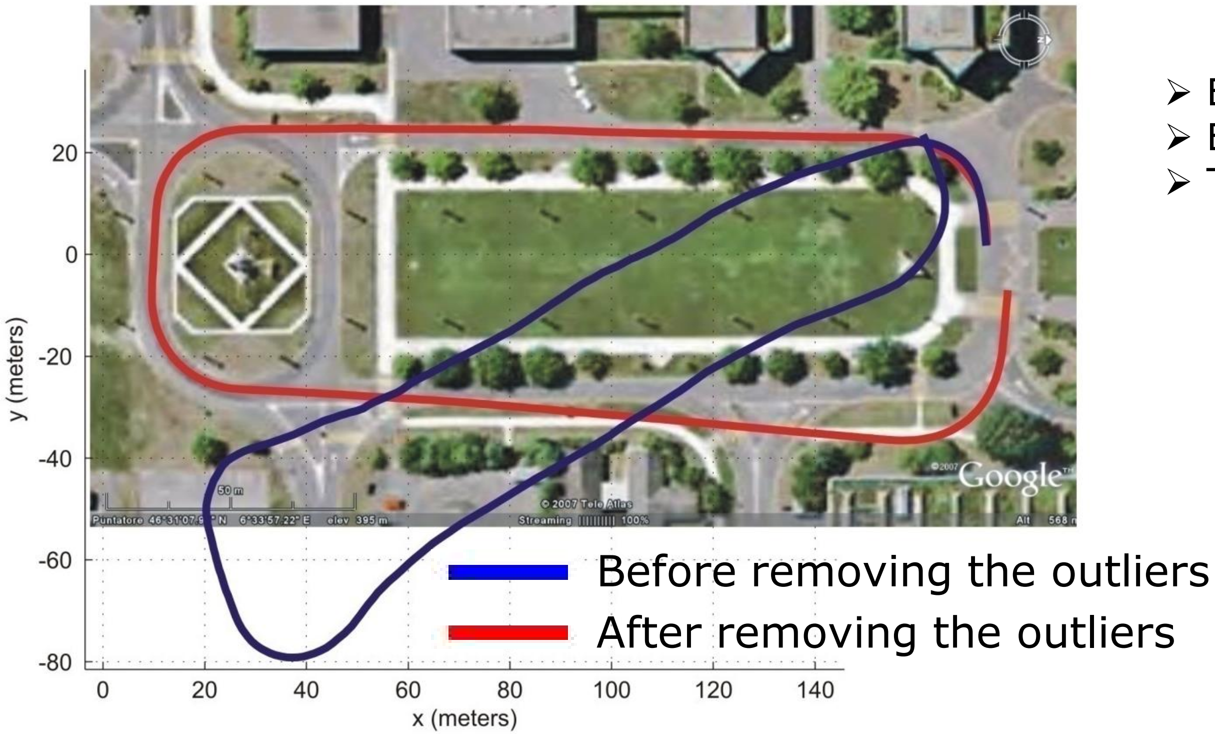


Image 1 Image 2

Influence of Outliers on Motion Estimation



- Error at the loop closure: 6.5 m
- Error in orientation: 5 deg
- Trajectory length: 400 m

Outliers can be removed using RANSAC [Fishler & Bolles, 1981]

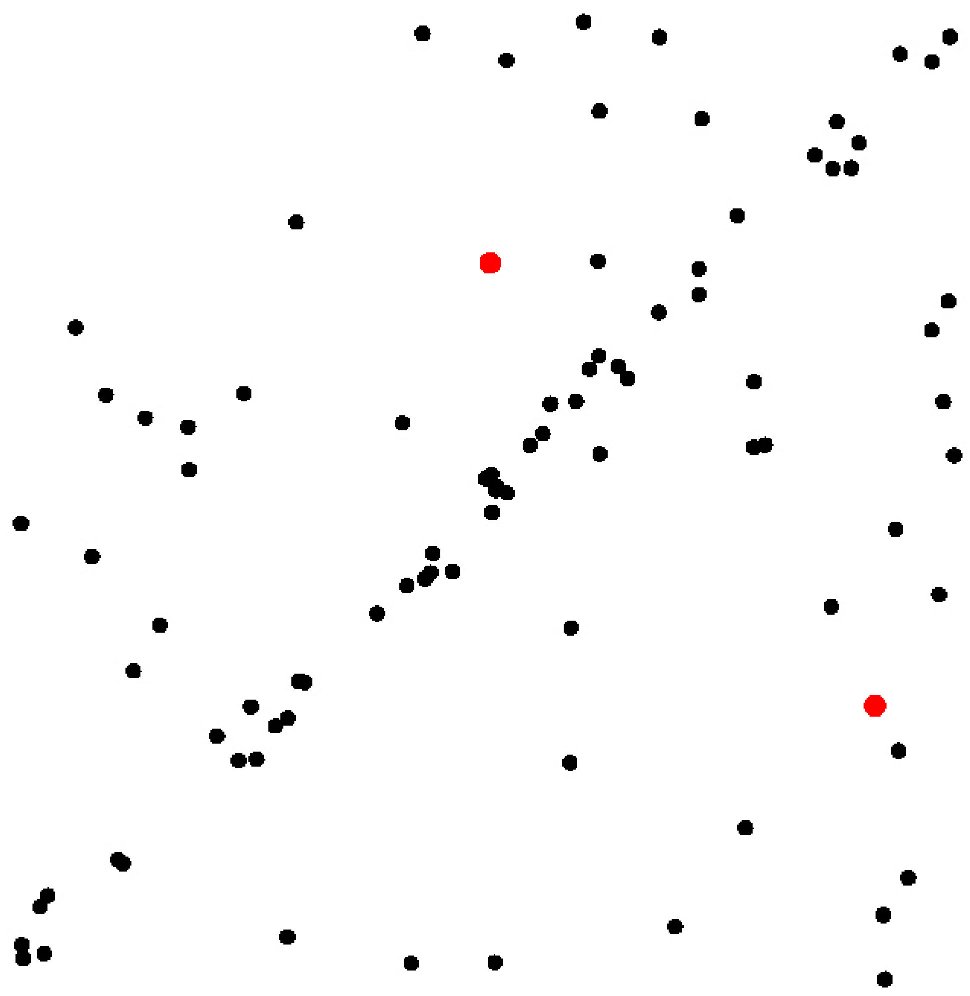
RANSAC (RAndom SAmple Consensus)

- RANSAC is the **standard method for model fitting in the presence of outliers** (very noisy points or wrong data)
- It can be applied to all sorts of problems where the goal is to **estimate the parameters of a model from the data** (e.g., camera calibration, Structure from Motion, DLT, PnP, P3P, Homography, etc.)
- Let’s review RANSAC for line fitting and see how we can use it to do Structure from Motion

M. A.Fischler and R. C.Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Graphics and Image Processing, 1981.

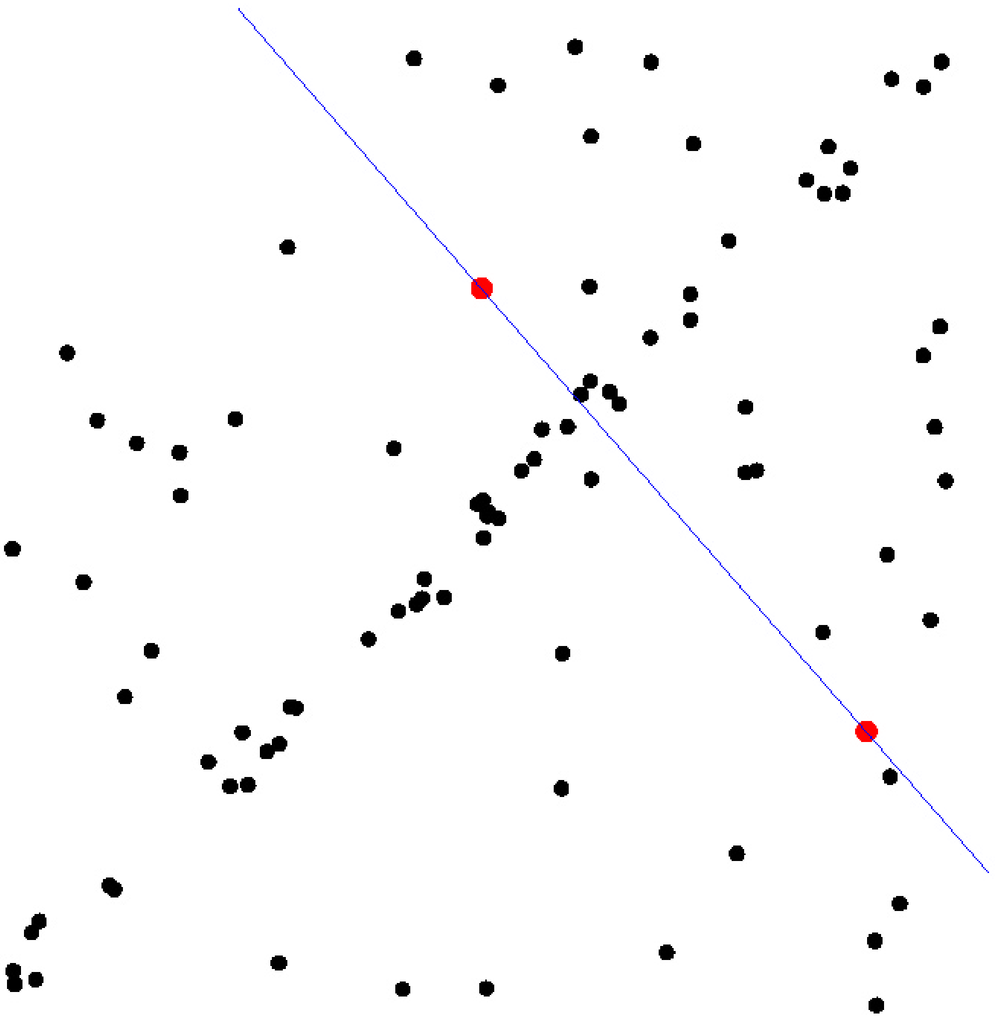
RANSAC

RANSAC



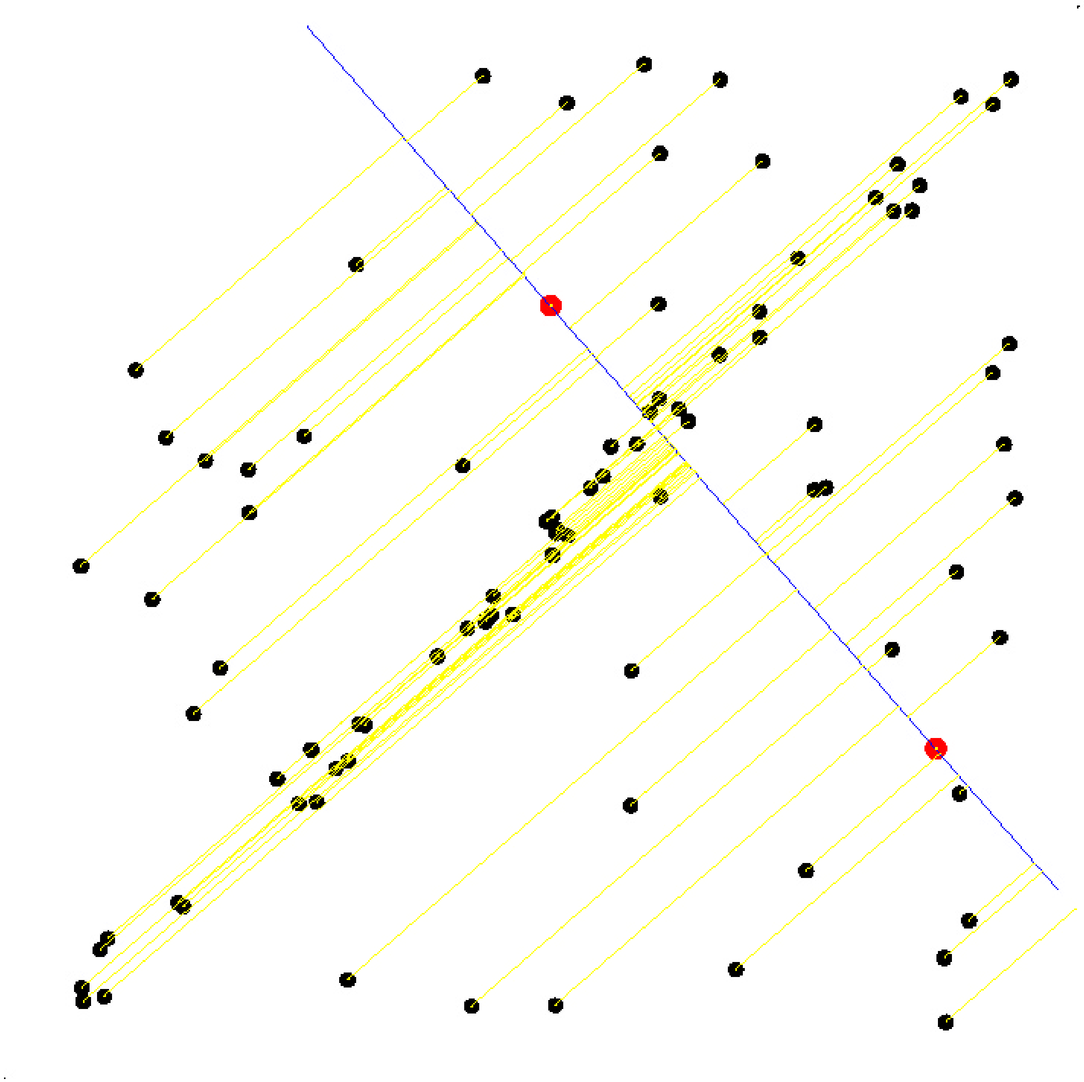
- **Select sample of 2 points at random**

RANSAC



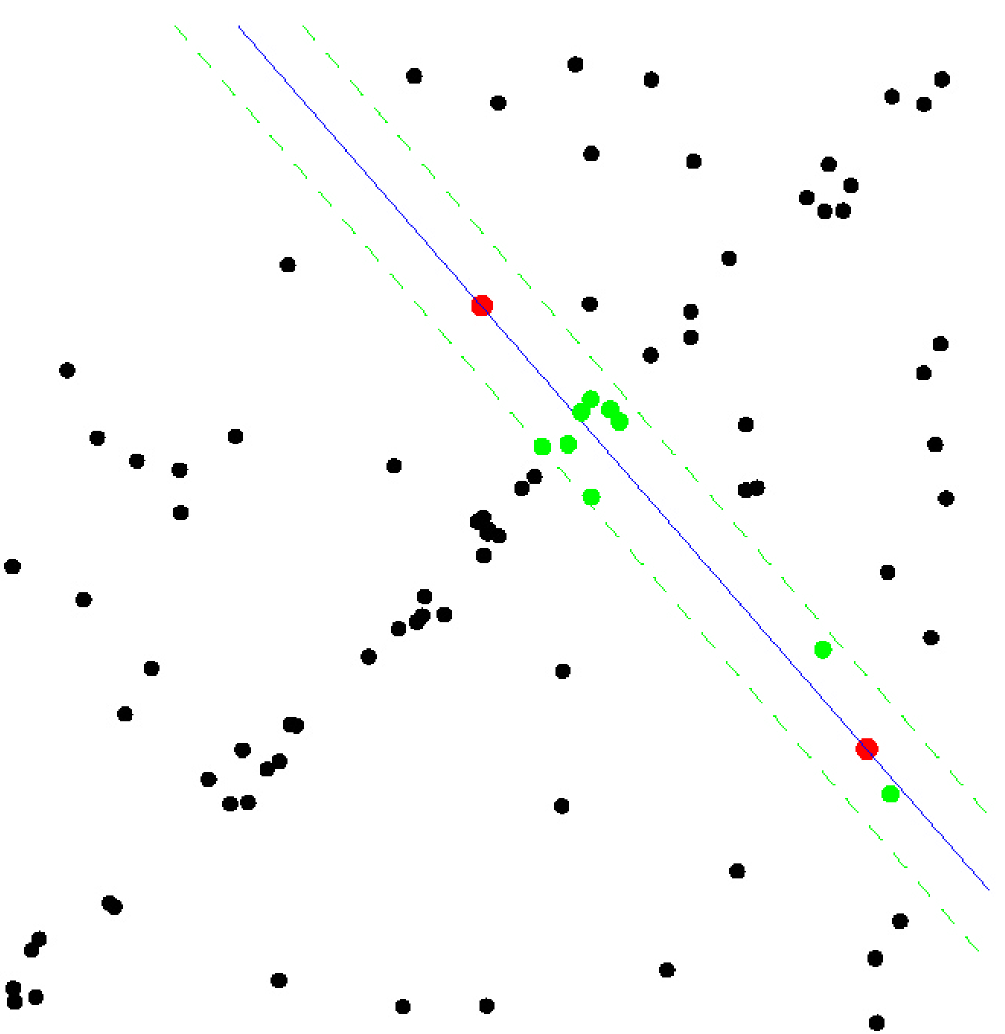
- Select sample of 2 points at random
- **Calculate model parameters that fit the data in the sample**

RANSAC



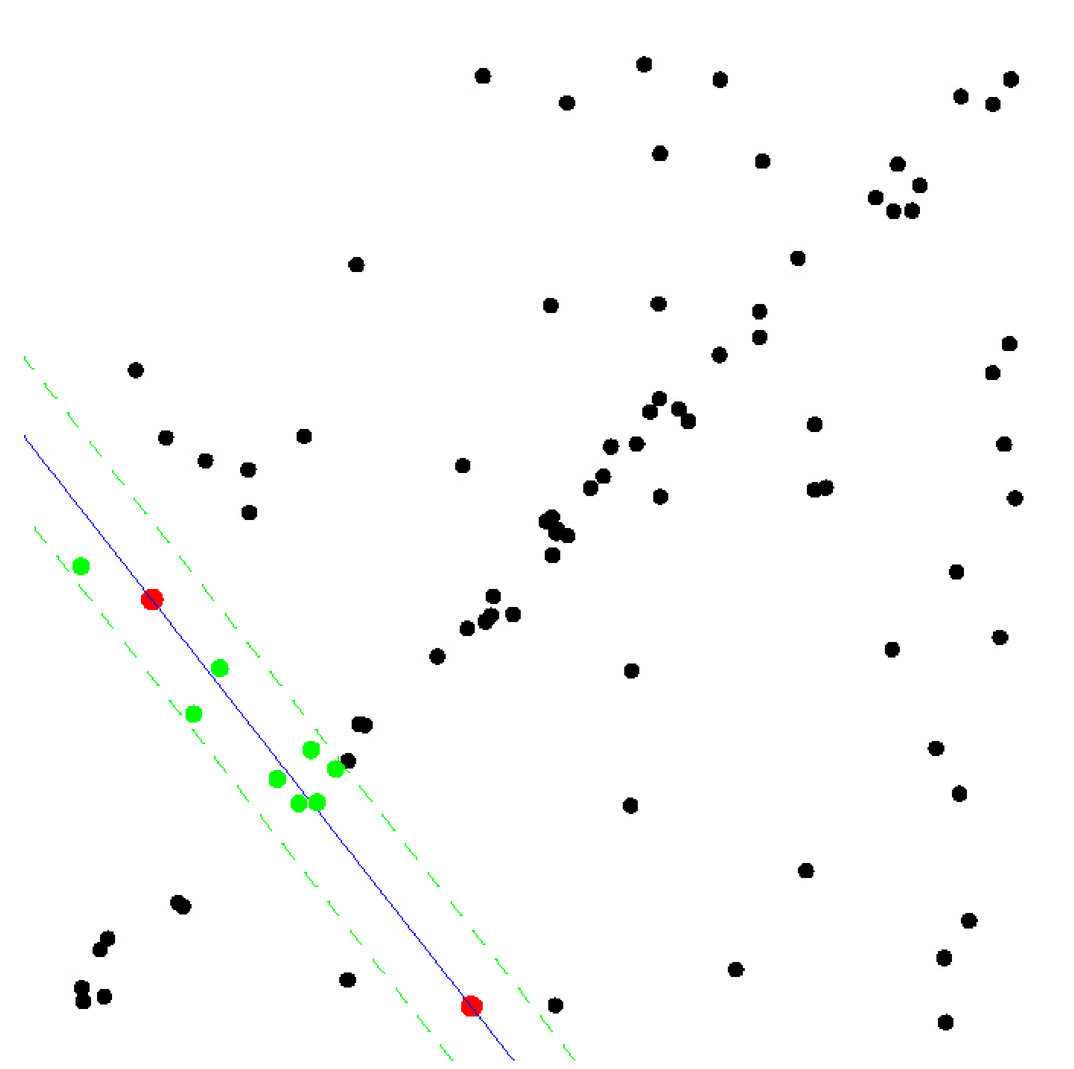
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

RANSAC



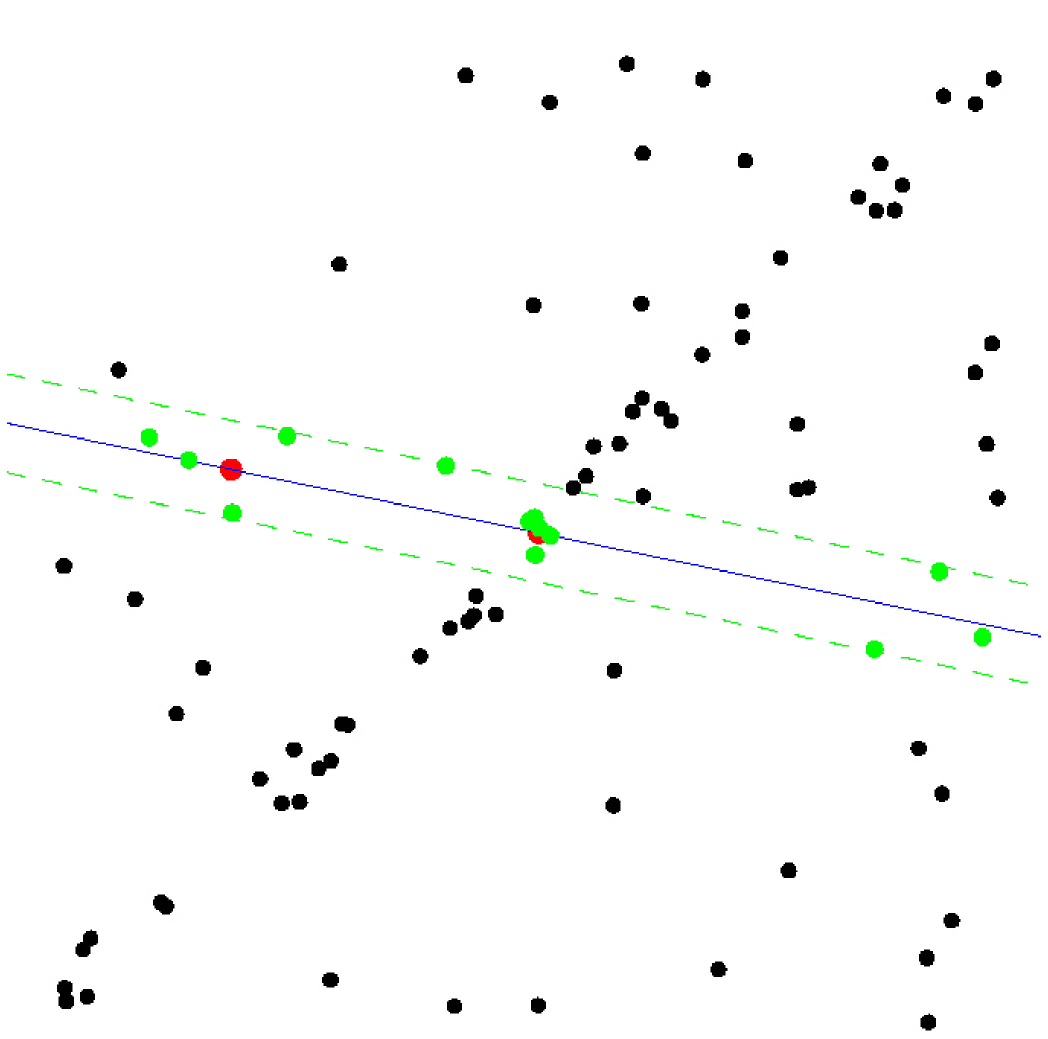
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- **Select data that supports current hypothesis**

RANSAC



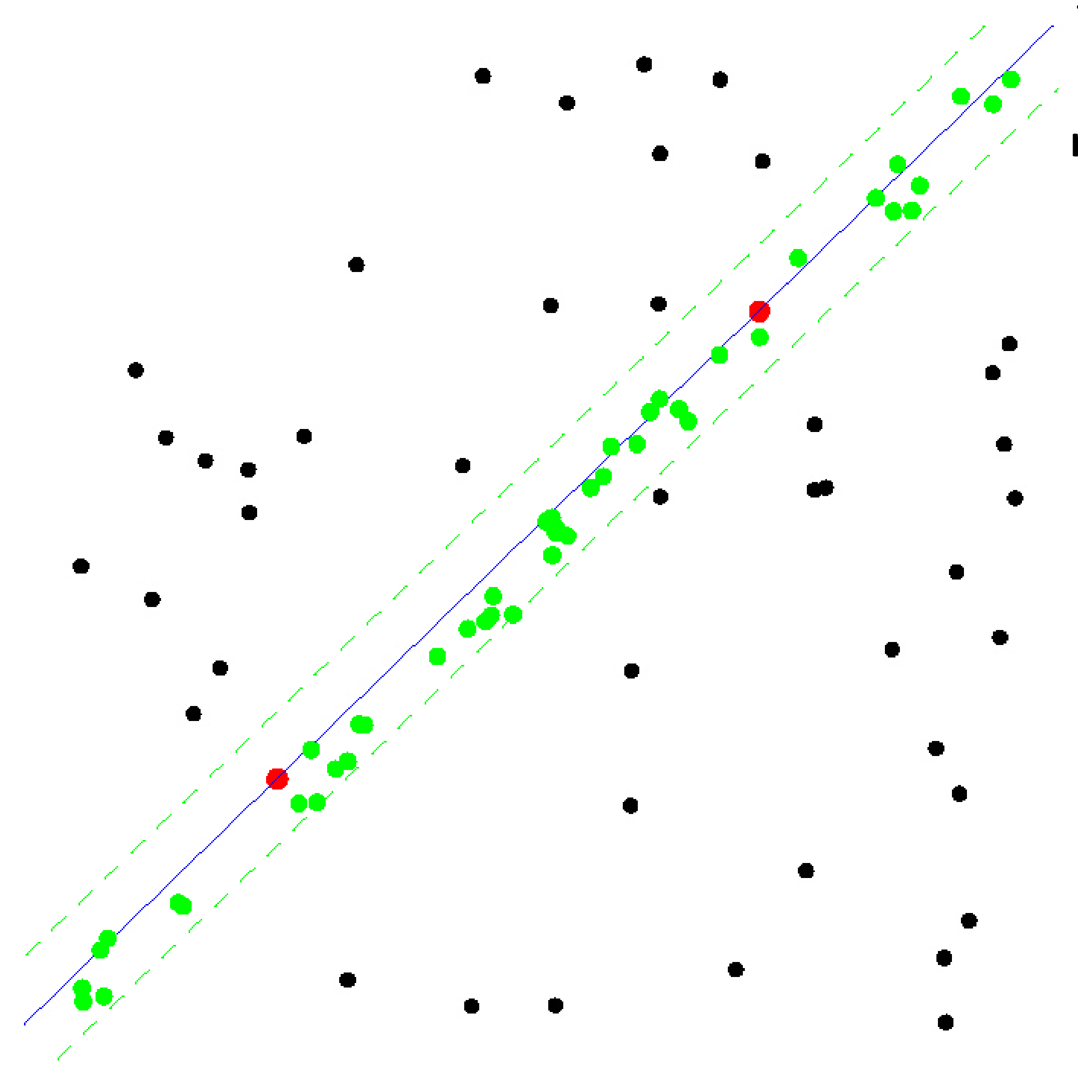
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- **Repeat**

RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- **Repeat**

RANSAC



Select the set with the maximum number of inliers obtained within k iterations

RANSAC

How many iterations does RANSAC need?

- Ideally: check all possible combinations of **2** points in a dataset of **N** points.
- Number of all pairwise combinations: **N(N-1)/2**

  ⇨ computationally unfeasible if **N** is too large.

  example: 1000 points ⇨ need to check all 1000*999/2 ≅ **500’000** possibilities!

- Do we really need to check all possibilities or can we stop RANSAC after some iterations?
  Checking a **subset** of combinations is enough if we have a **rough** estimate of the percentage of inliers in our dataset

- This can be done in a probabilistic way

RANSAC

How many iterations does RANSAC need?

- ***w*** := number of inliers/N
  N := total number of data points
  ⇨ ***w*** : fraction of inliers in the dataset ⇨ ***w =*** P(selecting an inlier-point out of the dataset)
- Assumption: the 2 points necessary to estimate a line are selected independently
  ⇨ w^2 = P(both selected points are inliers)
  ⇨ $1-w^2$ = P(at least one of these two points is an outlier)
- Let ***k*** := no. RANSAC iterations executed so far
- ⇨ $(1-w^2)^k$ = P(RANSAC never selected two points that are both inliers)
- Let ***p*** := P(probability of success)
- ⇨ $1-p = (1-w^2)^k$ and therefore :

$$k = \frac{\log(1-p)}{\log(1-w^2)}$$

RANSAC

How many iterations does RANSAC need?

- The number of iterations $\boldsymbol{k}$ is

$$k = \frac{\log(1 - p)}{\log(1 - w^2)}$$

- ⇨ knowing the fraction of inliers $\boldsymbol{w}$, after $\boldsymbol{k}$ RANSAC iterations we will have a probability $\boldsymbol{p}$ of finding a set of points free of outliers

- Example: if we want a probability of success $\boldsymbol{p}$=99% and we know that $\boldsymbol{w}$=50% ⇨ $\boldsymbol{k}$=16 iterations – these are dramatically fewer than the number of all possible combinations! **As you can see, the number of points does not influence the estimated number of iterations, only w does!**

- In practice we only need a rough estimate of $\boldsymbol{w}$.
  More advanced variants of RANSAC estimate the fraction of inliers and adaptively update it at every iteration (how?)

RANSAC applied to Line Fitting

1. Initial: let *A* be a set of *N* points
2. **repeat**
3. Randomly select a sample of 2 points from *A*
4. Fit a line through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points whose distance < *d*)
7. Store these inliers
8. **until** maximum number of iterations ***k*** reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

RANSAC applied to general model fitting

1. Initial: let *A* be a set of *N* points
2. **repeat**
3. Randomly select a sample of ***s*** points from *A*
4. **Fit a model** from the ***s*** points
5. Compute the **distances** of all other points from this model
6. Construct the inlier set (i.e. count the number of points whose distance < *d*)
7. Store these inliers
8. **until** maximum number of iterations ***k*** reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

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

The Three Key Ingredients of RANSAC

In order to implement RANSAC for Structure From Motion (SFM), we need three key ingredients:

1. What's the **model** in SFM?
2. What's the **minimum number of points** to estimate the model?
3. How do we compute the distance of a point from the model? In other words, can we define a **distance metric** that measures how well a point fits the model?

Answers

1. **What's the model** in SFM?
   - The **Essential Matrix** (for calibrated cameras) or the **Fundamental Matrix** (for uncalibrated cameras)
   - Alternatively, **R** and **T**

2. What's the **minimum number of points** to estimate the model?
   1. We know that 5 points is the theoretical minimum number of points
   2. However, if we use the *8-point algorithm*, then **8** is the minimum

3. How do we compute the **distance** of a point from the model?
   1. We can use the epipolar constraint ($\bar{p}_2^\top E \bar{p}_1 = 0$ or $p_2^\top F p_1 = 0$) to measure how well a point correspondence verifies the model E or F, respectively. However, the **Directional error**, the **Epipolar line distance**, or the **Reprojection error (even better)** are used (we already saw why)

Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows

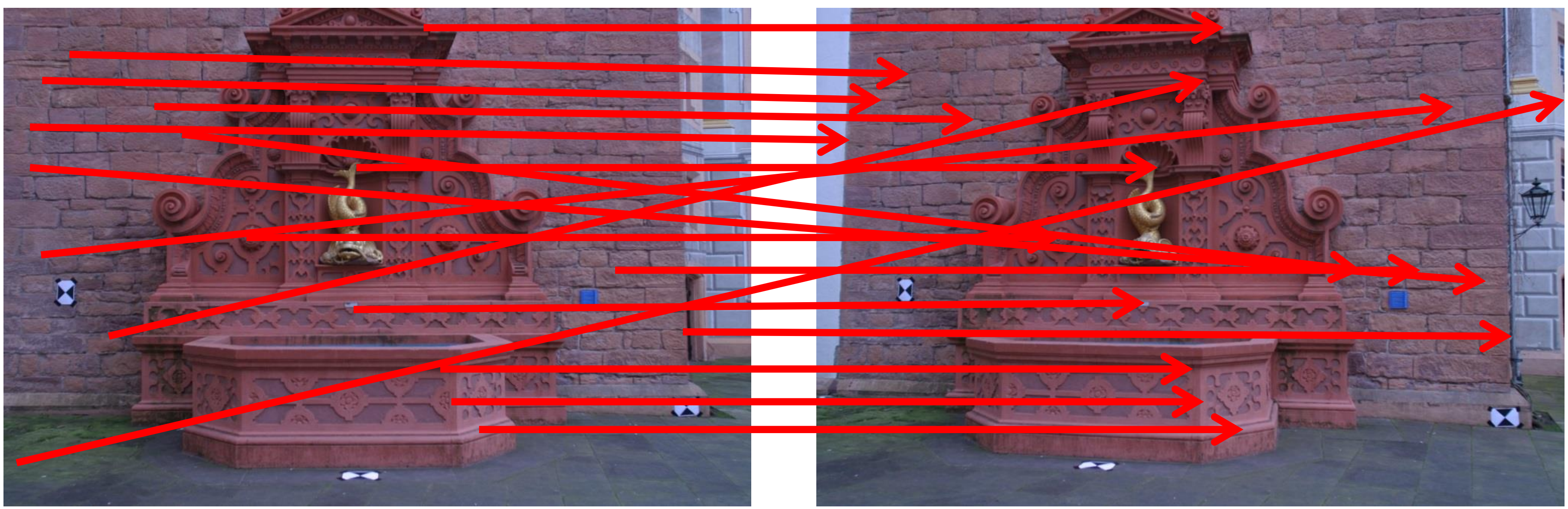


Image 1 Image 2

Example: 8-point RANSAC applied to SfM

- Let's consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

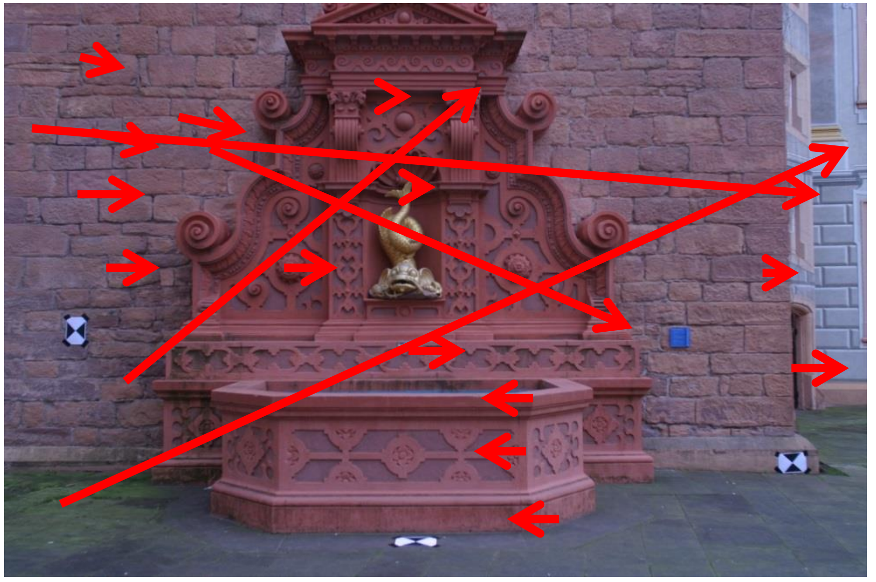


Image 1

Example: 8-point RANSAC applied to SfM

- Let's consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences

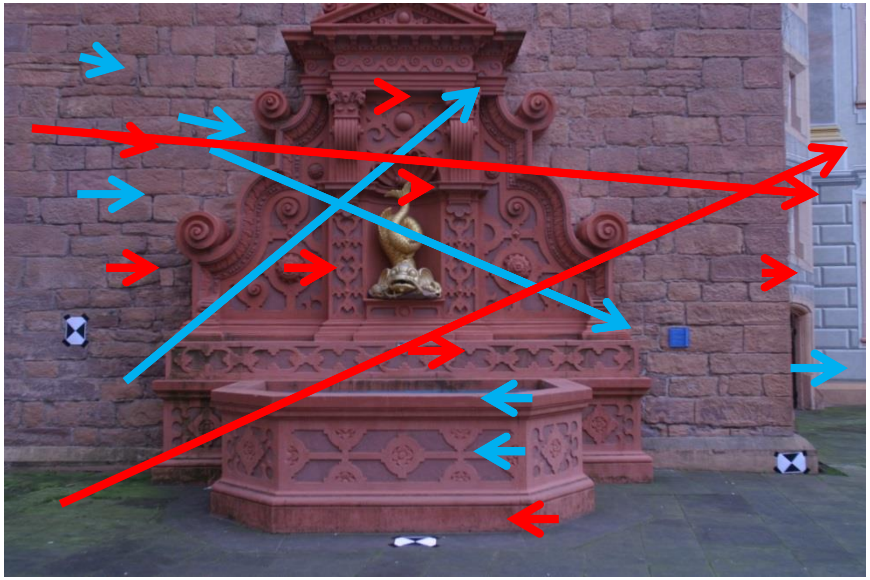


Image 1

Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers

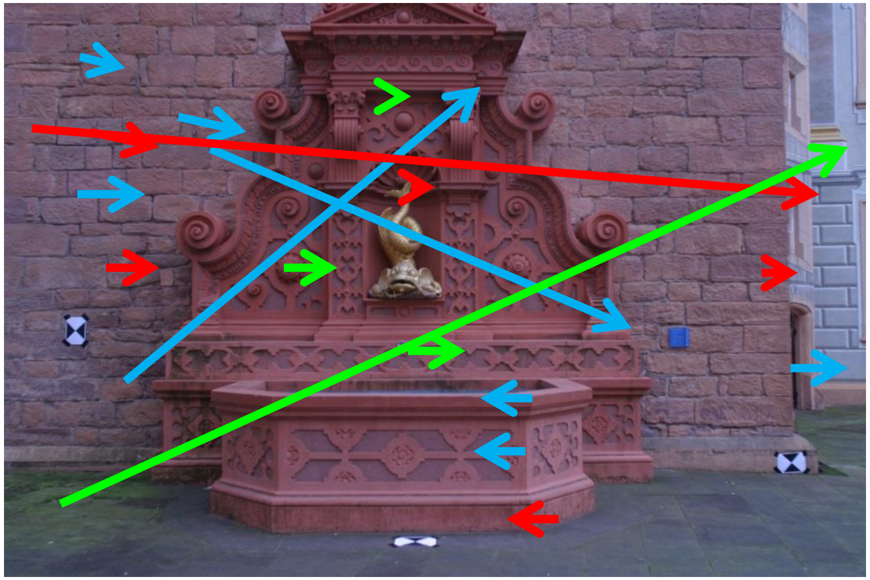


Image 1

Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers
3. Repeat from 1

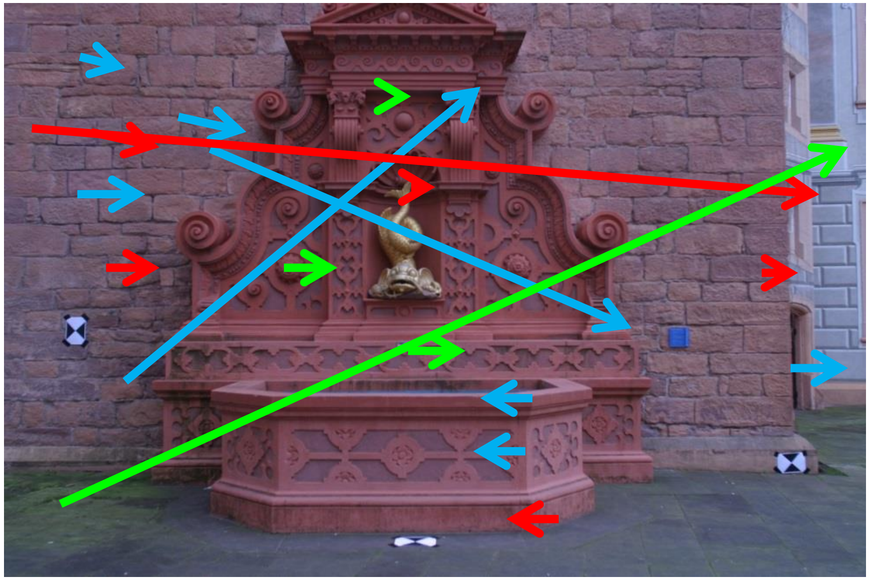


Image 1

Example: 8-point RANSAC applied to SfM

- Let's consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

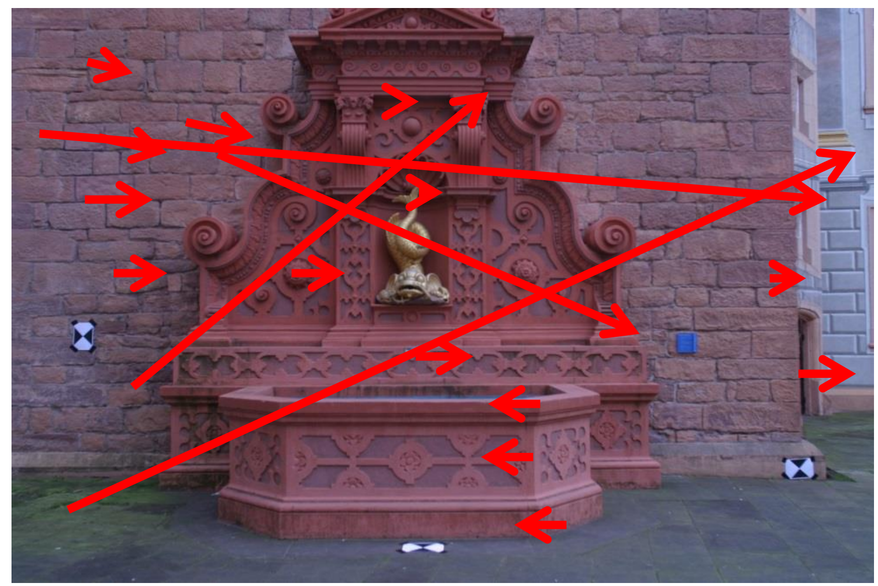


Image 1

Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences

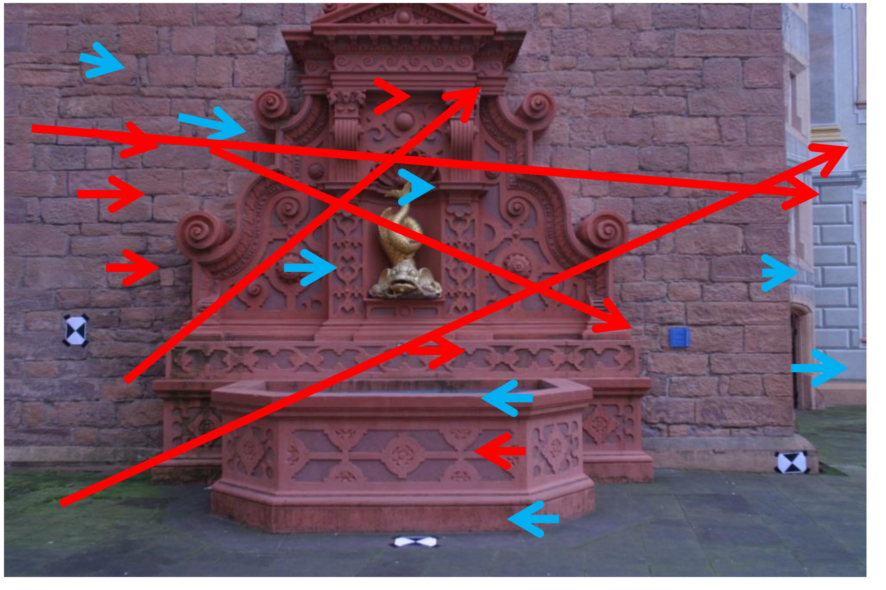


Image 1

Example: 8-point RANSAC applied to SfM

- Let’s consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers

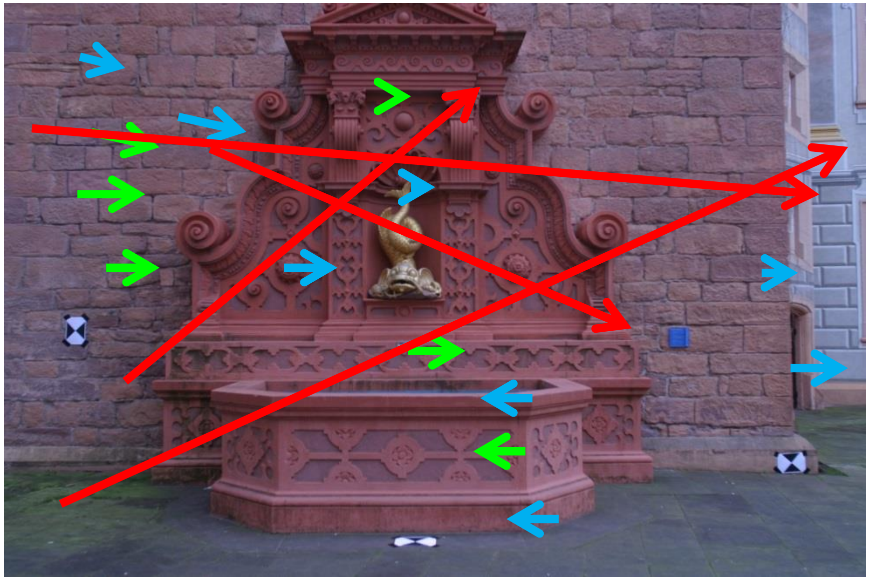


Image 1

Example: 8-point RANSAC applied to SfM

- Let's consider the following image pair and its image correspondences (e.g., Harris, SIFT, etc.), denoted by arrows
- For convenience, we overlay the features of the second image in the first image and use arrows to denote the motion vectors of the features

1. Randomly select 8 point correspondences
2. Fit the model to all other points and count the inliers
3. Repeat from 1 for ***k*** times

$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^8)}$$

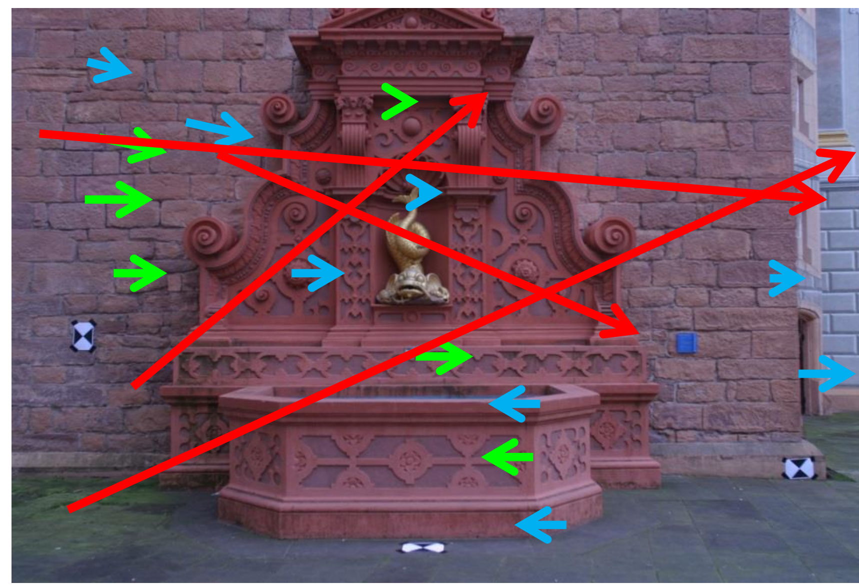


Image 1

RANSAC iterations $\boldsymbol{k}$ vs. $\boldsymbol{s}$

$\boldsymbol{k}$ is exponential in the number of points $\boldsymbol{s}$ necessary to estimate the model:

- **8-point RANSAC**
  - Assuming
    - $\boldsymbol{p}$ = 99%,
    - $\boldsymbol{\varepsilon}$ = 50% (fraction of outliers)
    - $\boldsymbol{s}$ = 8 points (8-point algorithm)

$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} = 1177 \quad iterations$$

- **5-point RANSAC**
  - Assuming
    - $\boldsymbol{p}$ = 99%,
    - $\boldsymbol{\varepsilon}$ = 50% (fraction of outliers)
    - $\boldsymbol{s}$ = 5 points (5-point algorithm of David Nister (2004))

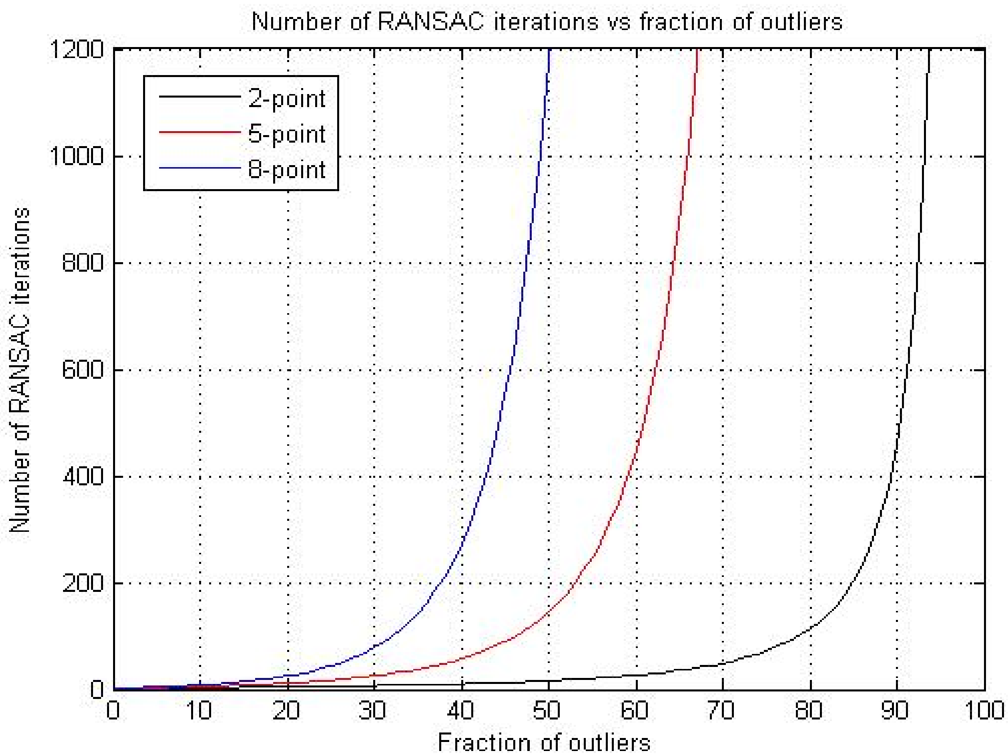
$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} = 145 \quad iterations$$

- **2-point RANSAC (e.g., line fitting)**
  - Assuming
    - $\boldsymbol{p}$ = 99%,
    - $\boldsymbol{\varepsilon}$ = 50% (fraction of outliers)
    - $\boldsymbol{s}$ = 2 points

$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} = 16 \quad iterations$$

RANSAC iterations $\boldsymbol{k}$ vs. $\boldsymbol{\varepsilon}$

- $\boldsymbol{k}$ is increases exponentially with the fraction of outliers $\boldsymbol{\varepsilon}$



RANSAC iterations

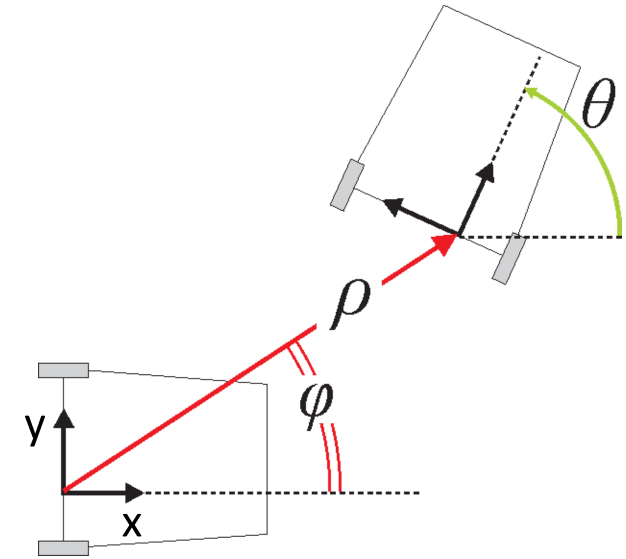
- As observed, $\boldsymbol{k}$ is exponential in the number of points $\boldsymbol{s}$ necessary to estimate the model
- The **8-point algorithm** is extremely simple and was very successful; however, it requires more than **1177 iterations**
- Because of this, there has been a large interest by the research community in **using smaller motion parameterizations** (i.e., smaller *s*)
- The first efficient solution to the minimal-case solution (5-point algorithm) took almost a century (Kruppa 1913 → Nister 2004)
- The **5-point RANSAC** (Nister 2004) only requires **145 iterations**; however:
  - The **5-point algorithm** can return **up to 10 solutions of E (worst case scenario)**
  - The **8-point algorithm** only returns a **unique solution of E**

Can we use less than 5 points?

Yes, if you use motion constraints!

Planar Motion

Planar motion is described by three parameters: *ϑ, φ, ρ*



$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\varphi \\ \rho\sin\varphi \\ 0 \end{bmatrix}$$

Let's compute the Epipolar Geometry

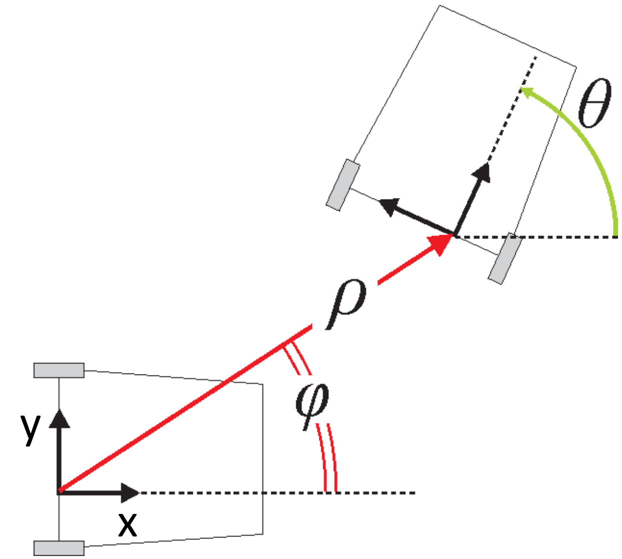
$$\mathrm{E} = [T]_\times R$$ *Essential matrix*

$$\overline{p}_2^T \; E \; \overline{p}_1 = 0$$ *Epipolar constraint*

Planar Motion

Planar motion is described by three parameters: *ϑ, φ, ρ*

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad T = \begin{bmatrix} \rho\cos\varphi \\ \rho\sin\varphi \\ 0 \end{bmatrix}$$



Let’s compute the Epipolar Geometry

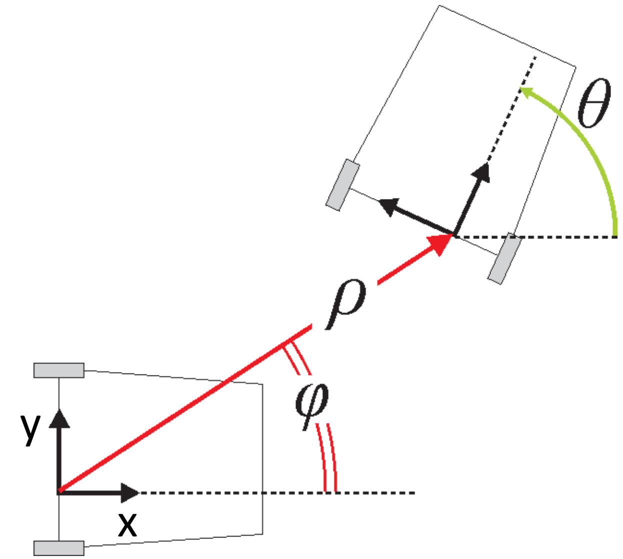
$$[T]_\times = \begin{bmatrix} 0 & 0 & \rho\sin\varphi \\ 0 & 0 & -\rho\cos\varphi \\ -\rho\sin\varphi & \rho\cos\varphi & 0 \end{bmatrix}$$

$$E = [T]_\times R = \begin{bmatrix} 0 & 0 & \rho\sin\varphi \\ 0 & 0 & -\rho\cos\varphi \\ -\rho\sin\varphi & \rho\cos\varphi & 0 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Planar Motion

Planar motion is described by three parameters: *ϑ, φ, ρ*

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\varphi \\ \rho\sin\varphi \\ 0 \end{bmatrix}$$



Let’s compute the Epipolar Geometry

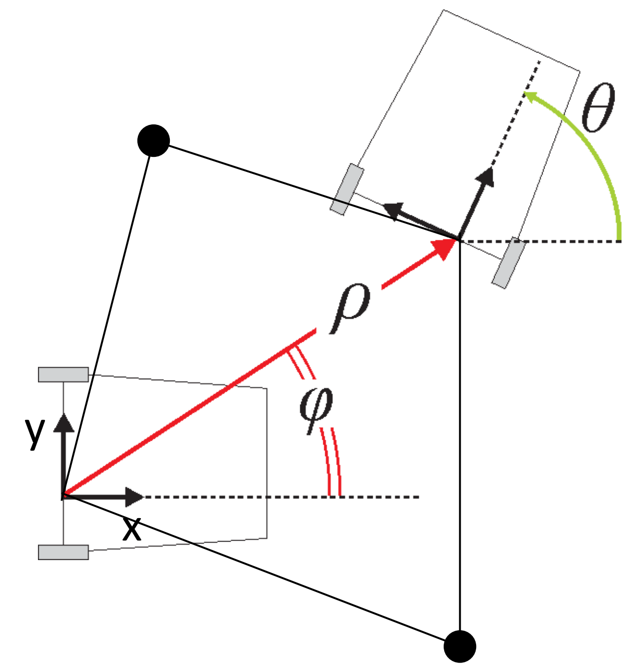
$$[T]_\times = \begin{bmatrix} 0 & 0 & \rho\sin\varphi \\ 0 & 0 & -\rho\cos\varphi \\ -\rho\sin\varphi & \rho\cos\varphi & 0 \end{bmatrix}$$

$$E = [T]_\times R = \begin{bmatrix} 0 & 0 & \rho\sin(\varphi) \\ 0 & 0 & -\rho\cos(\varphi) \\ -\rho\sin(\varphi-\theta) & \rho\cos(\varphi-\theta) & 0 \end{bmatrix}$$

Planar Motion

Planar motion is described by three parameters: ϑ, φ, ρ

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad T = \begin{bmatrix} \rho\cos\varphi \\ \rho\sin\varphi \\ 0 \end{bmatrix}$$



Observe that E has 2DoF (θ, ϕ, because ρ is the scale factor); thus, 2 correspondences are sufficient to estimate θ and ϕ ["2-Point RANSAC", Ortin, 2001]

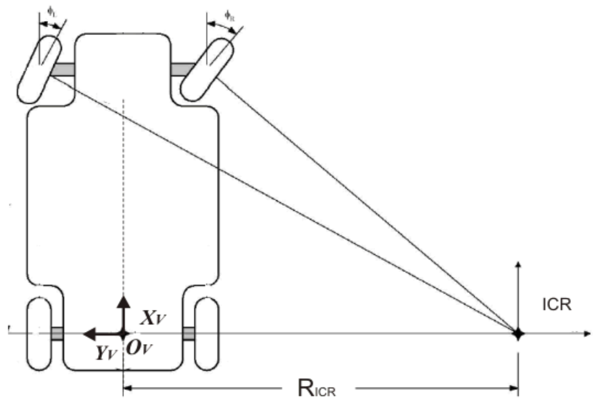
$$E = [T]_\times R = \begin{bmatrix} 0 & 0 & \rho\sin(\varphi) \\ 0 & 0 & -\rho\cos(\varphi) \\ -\rho\sin(\varphi-\theta) & \rho\cos(\varphi-\theta) & 0 \end{bmatrix}$$

Can we use less than 2 point correspondences?

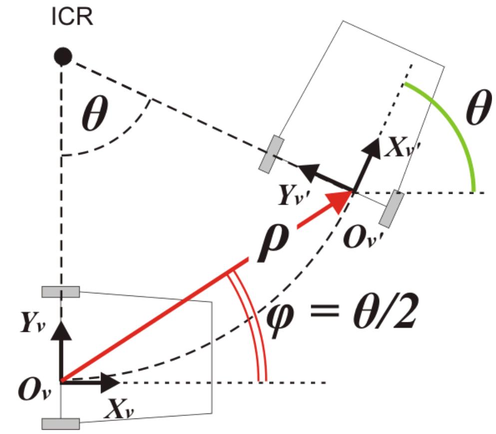
Yes, if we exploit wheeled vehicles with **non-holonomic** constraints

Planar & Circular Motion (e.g., cars)

Wheeled vehicles, like cars, follow locally-planar circular motion about the Instantaneous Center of Rotation (ICR)



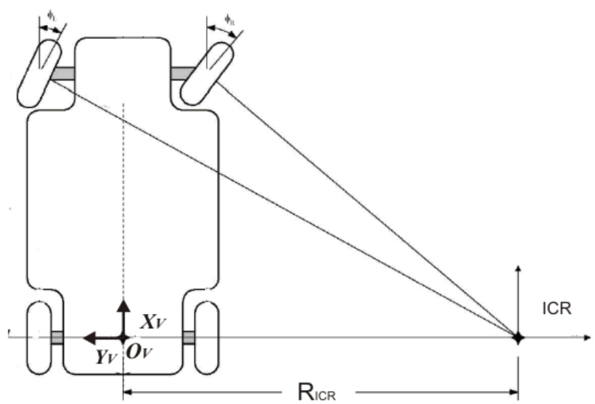
Example of Ackerman steering principle



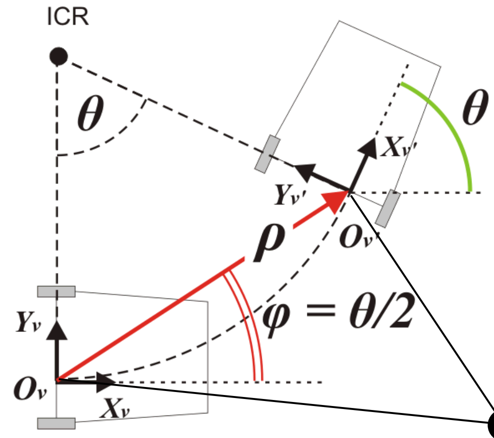
Locally-planar circular motion

Planar & Circular Motion (e.g., cars)

Wheeled vehicles, like cars, follow locally-planar circular motion about the Instantaneous Center of Rotation (ICR)



Example of Ackerman steering principle



Locally-planar circular motion

$$\varphi = \theta/2 \Rightarrow \text{only 1 DoF } (\theta);$$

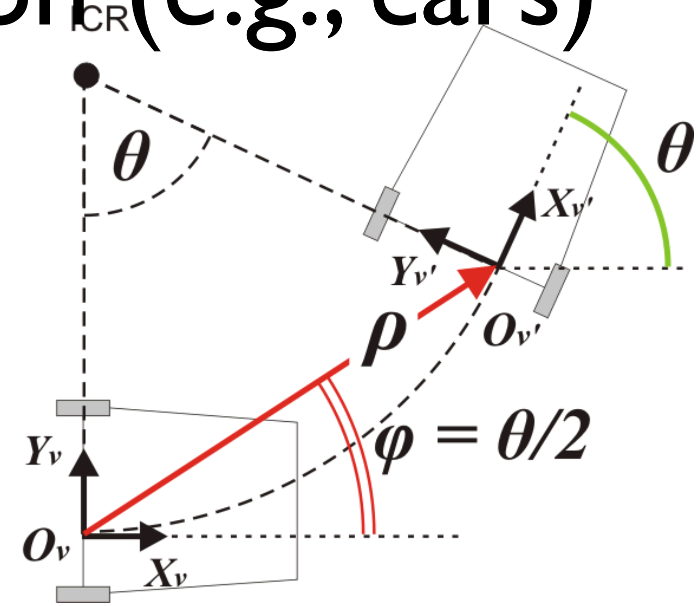
thus, only 1 point correspondence is needed

This is the smallest parameterization possible and results in the most efficient algorithm for removing outliers

Scaramuzza, **1-Point-RANSAC Structure from Motion for Vehicle-Mounted Cameras by Exploiting Non-holonomic Constraints**, International Journal of Computer Vision, 2011

Planar & Circular Motion (e.g., cars)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\frac{\theta}{2} \\ \rho\sin\frac{\theta}{2} \\ 0 \end{bmatrix}$$



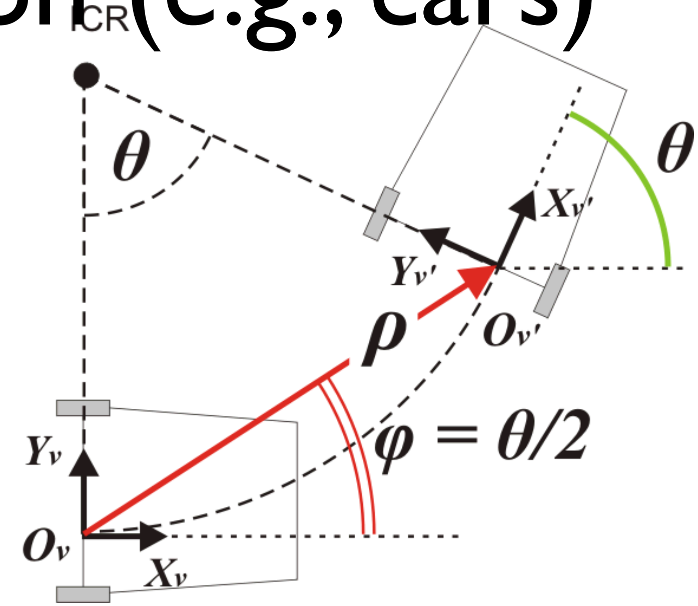
Let’s compute the Epipolar Geometry

$$\mathrm{E} = [T]_{\times} R \qquad \textit{Essential matrix}$$

$$\overline{p}_2^{\,T}\, E\, \overline{p}_1 = 0 \qquad \textit{Epipolar constraint}$$

Planar & Circular Motion (e.g., cars)

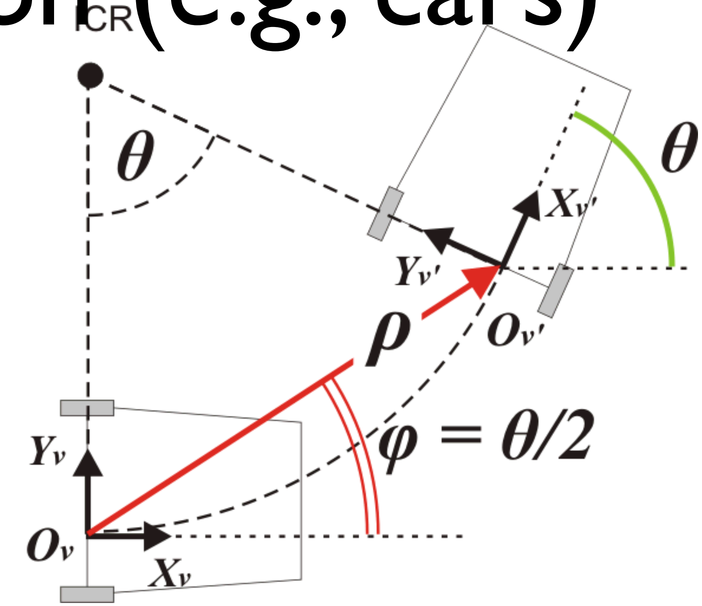
$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\frac{\theta}{2} \\ \rho\sin\frac{\theta}{2} \\ 0 \end{bmatrix}$$



Let's compute the Epipolar Geometry

$$E = [T]_\times R = \begin{bmatrix} 0 & 0 & \rho\sin\frac{\theta}{2} \\ 0 & 0 & -\rho\cos\frac{\theta}{2} \\ -\rho\sin\frac{\theta}{2} & \rho\cos\frac{\theta}{2} & 0 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \rho\sin\frac{\theta}{2} \\ 0 & 0 & \rho\cos\frac{\theta}{2} \\ \rho\sin\frac{\theta}{2} & -\rho\cos\frac{\theta}{2} & 0 \end{bmatrix}$$

Planar & Circular Motion (e.g., cars)



$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\frac{\theta}{2} \\ \rho\sin\frac{\theta}{2} \\ 0 \end{bmatrix}$$

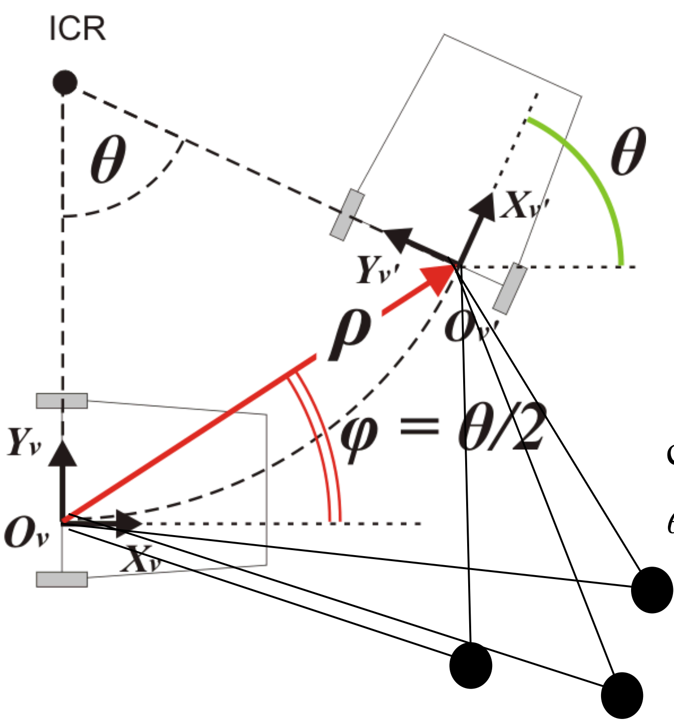
Let's compute the Epipolar Geometry

$$E = \rho \begin{bmatrix} 0 & 0 & \sin\frac{\theta}{2} \\ 0 & 0 & \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} & -\cos\frac{\theta}{2} & 0 \end{bmatrix}$$

$$p_2^T \; E \; p_1 = 0 \quad \Rightarrow \quad \sin\left(\frac{\theta}{2}\right)\cdot(u_2 + u_1) + \cos\left(\frac{\theta}{2}\right)\cdot(v_2 - v_1) = 0$$

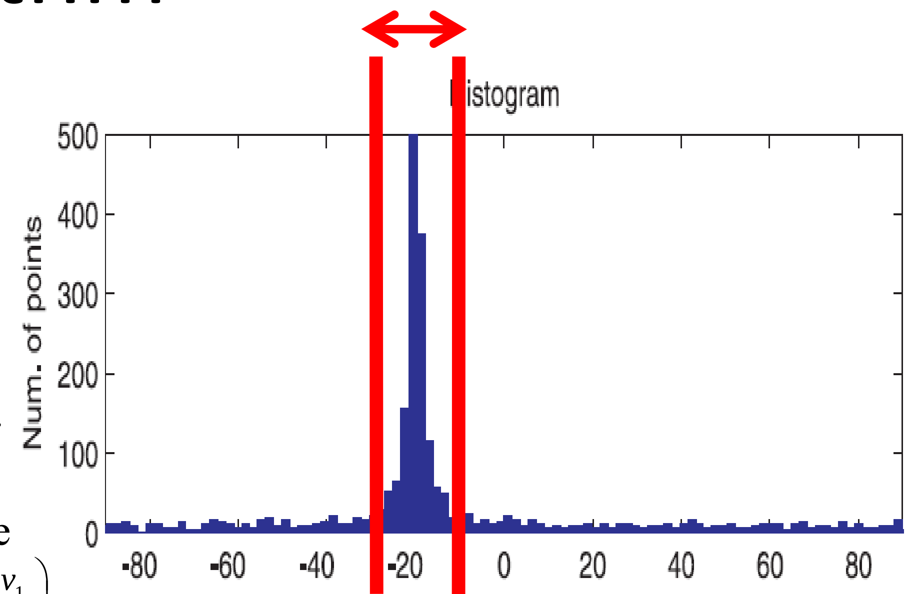
$$\boxed{\theta = -2\tan^{-1}\left(\frac{v_2 - v_1}{u_2 + u_1}\right)}$$

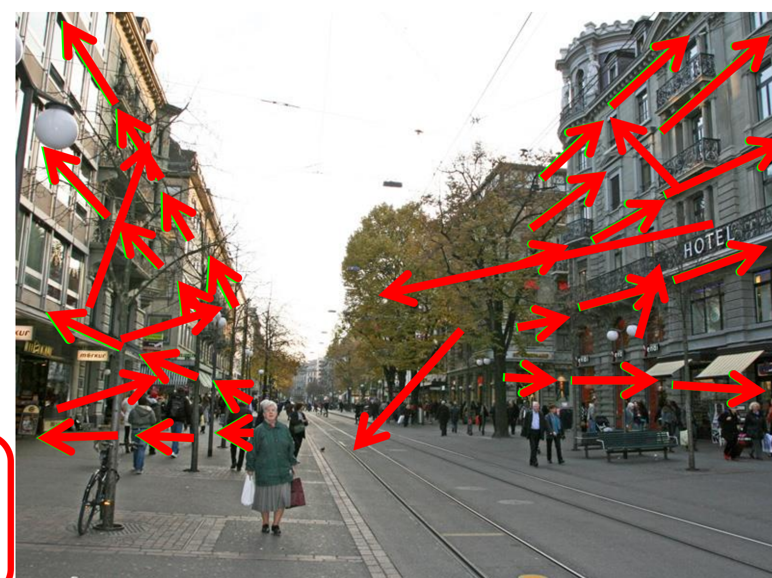
1-Point RANSAC algorithm



Compute θ for every point correspondence

$$\theta = -2\tan^{-1}\left(\frac{v_2 - v_1}{u_2 + u_1}\right)$$





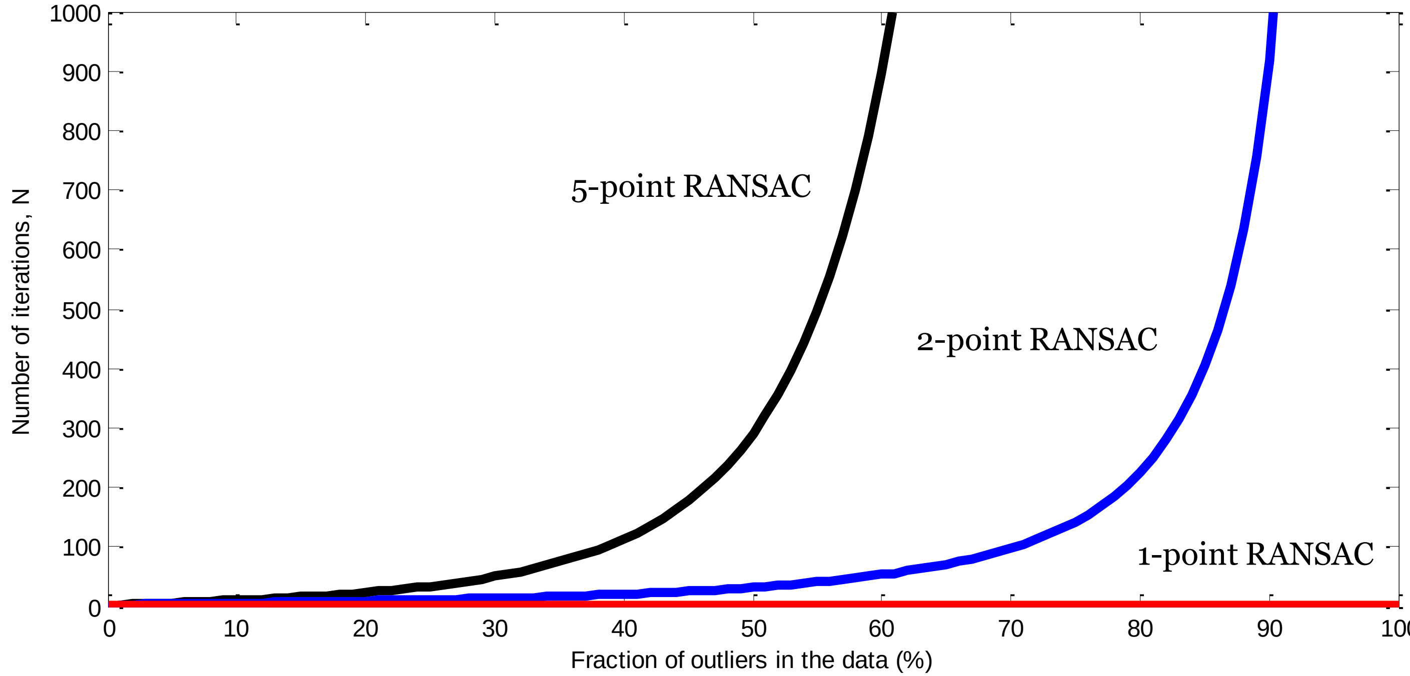
Only 1 iteration!

The most efficient algorithm for removing outliers, up to 1000 Hz

1-Point RANSAC is ONLY used to find the inliers.

Motion is then estimated from them in 6DOF

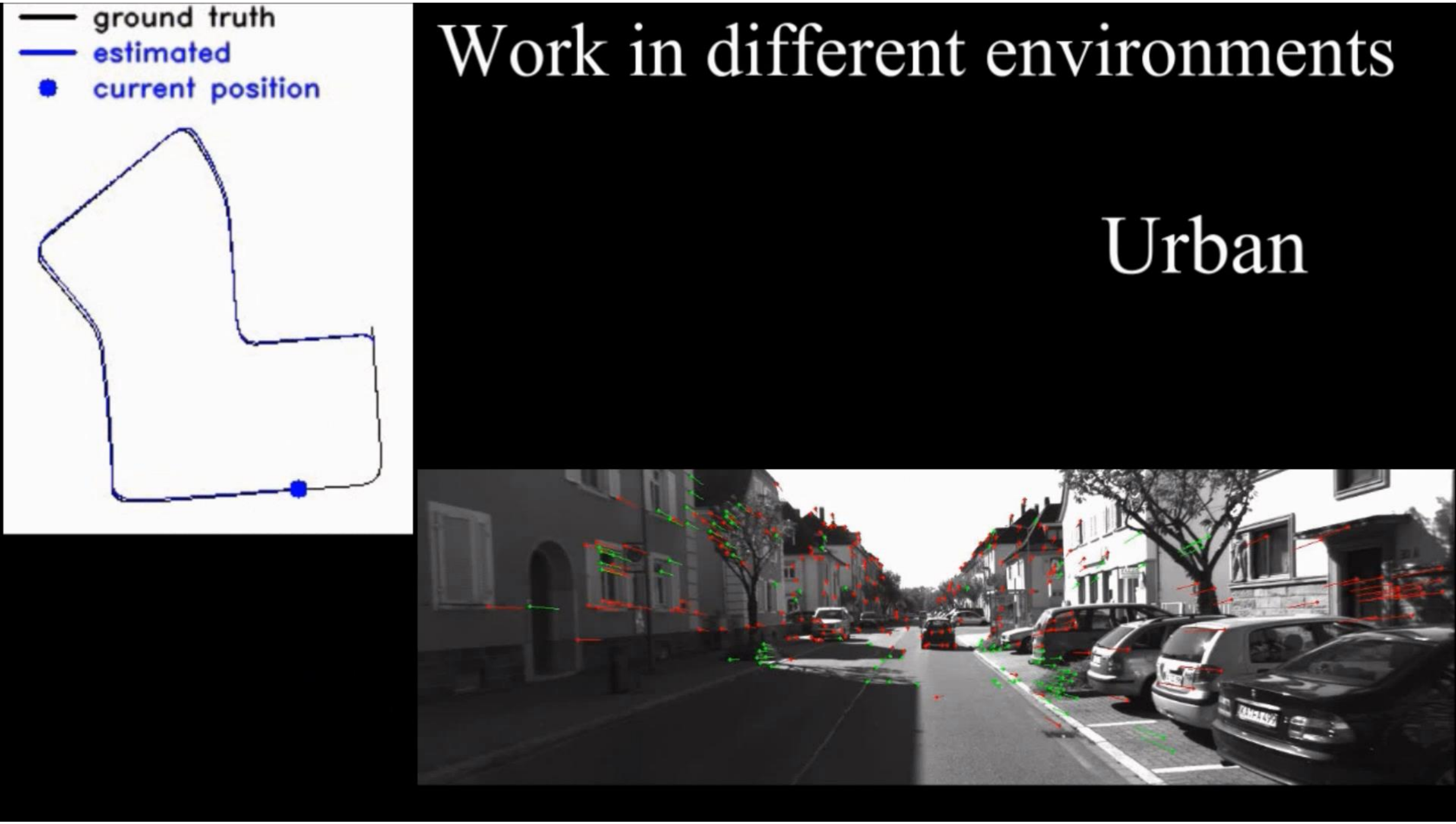
Comparison of RANSAC algorithms



$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \quad \text{where we typically use } p = 99\%$$

	8-Point RANSAC	5-Point RANSAC [Nister'03]	2-Point RANSAC [Ortin'01]	1-Point RANSAC [Scaramuzza, IJCV'10]
Numb. of iterations	**> 1177**	**>145**	**>16**	**=1**

Visual Odometry with 1-Point RANSAC



Scaramuzza, **1-Point-RANSAC Structure from Motion for Vehicle-Mounted Cameras by Exploiting Non-holonomic Constraints**, International Journal of Computer Vision, 2011

Things to remember

- SFM from 2 view
  - Calibrated and uncalibrated case
  - Proof of Epipolar Constraint
  - 8-point algorithm and algebraic error
  - Normalized 8-point algorithm
  - Algebraic, directional, Epipolar line distance, Reprojection error
  - RANSAC and its application to SFM
  - 8 vs 5 vs 1 point RANSAC, pros and cons

- Readings:
  - Ch. 14.2 of Corke book
  - CH. 7.2 of Szeliski book