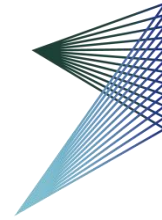University of
Zurich**UZH**

**ETH**zürich

Institute of Informatics – Institute of Neuroinformatics

ROBOTICS &
PERCEPTION
GROUP

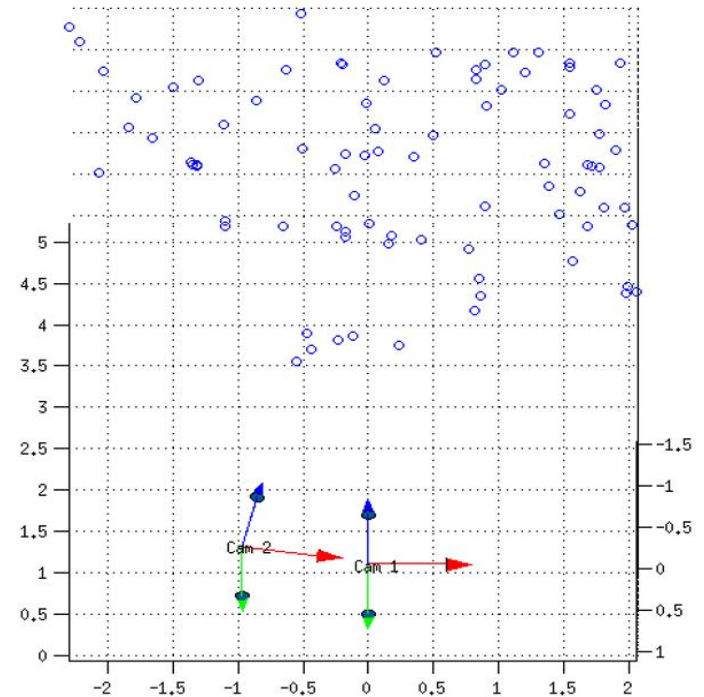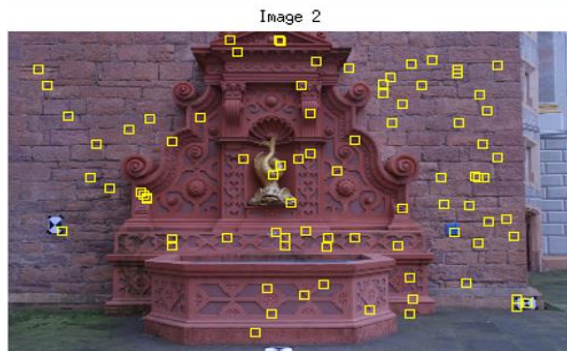# Lecture 08
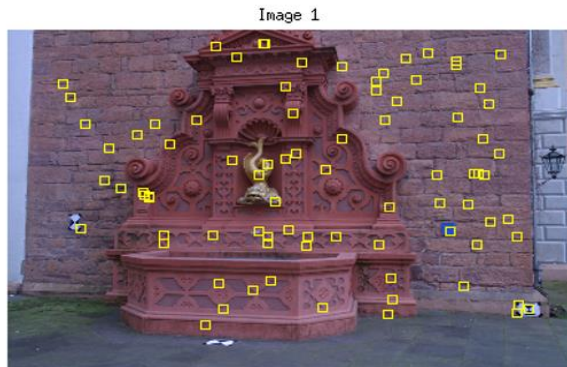# Multiple View Geometry 2

Davide Scaramuzza
http://rpg.ifi.uzh.ch/

# Lab Exercise 6 - Today afternoon

➢ Room ETH HG E 1.1 from 13:15 to 15:00

➢ Work description:  8-point algorithm

Estimated poses and 3D structure

# Our student projects on Robotics, ML, and Perception

- [http://rpg.ifi.uzh.ch/student_projects.php](http://rpg.ifi.uzh.ch/student_projects.php)

**University of Zurich** | **ETH** *zürich* — ROBOTICS & PERCEPTION GROUP

Department of Informatics - Institute of Neuroinformatics - Robotics and Perception Group

News
People
Research
Publications
Software/Datasets
Open Positions
Student Projects
Teaching
Media Coverage
Awards
Gallery
Contact

g+
GitHub
You Tube

## Student Projects

### How to apply

To apply, please send your CV, your Ms and Bs transcripts by email to all the contacts indicated below the project description. Do not apply on SiROP . Since Prof. Davide Scaramuzza is affiliated with ETH, there is no organizational overhead for ETH students. Custom projects are occasionally available. If you would like to do a project with us but could not find an advertized project that suits you, please contact Prof. Davide Scaramuzza directly to ask for a tailored project (sdavide at ifi.uzh.ch).

Upon successful completion of a project in our lab, students may also have the opportunity to get an internship at one of our numerous industrial and academic partners worldwide (e.g., NASA/JPL, University of Pennsylvania, UCLA, MIT, Stanford, ...).

### Exploring Adaptive Control Methods for High-Performance Quadrotor Control - Available

**Description:** Standard control algorithms require a very accurate model of the system to be controlled in order to take full advantage of system dynamic capabilities. However, such an accurate model is often not available (e.g., due to time-varying parameters or due to hard-to-model effects such as aerodynamic drag). One method to overcome this problem is to apply adaptive nonlinear control where model uncertainties and unknowns are concurrently learned while controlling the system. The goal of this project is to explore both vintage and recent adaptive control methods for the in-flight identification of a quadrotor's physical parameters (e.g., mass, inertia, thrust coefficients, drag) and adaptation of its controller.

**Contact Details:** Please send your CV and transcript to: Dario Brescianini, brescianini (at) ifi (dot) uzh (dot) ch
**Thesis Type:** Semester Project / Master Thesis
See project on SiROP

### An Open-Source Real-Time Event Camera Simulator for Robot Applications - Available

**Description:** Event cameras are revolutionary sensors that work radically differently from standard cameras. Instead of capturing intensity images at a fixed rate, event cameras measure per-pixel intensity changes asynchronously at the time they occur [1]. Since these sensors are not readily available and expensive, we have recently published an open source event camera simulator [2] that allows simulating arbitrary 3D camera motions in 3D scenes. The goal of this project is to close the loop and integrate the event camera simulator in a real-time robot simulation framework such that different robotic platforms with event camera sensors can be simulated and react based upon the measured events. [1]: D. Scaramuzza, Tutorial on Event-based Vision for High-Speed Robotics. http://www.rit.edu/kgcoe/iros15workshop/papers/IROS2015-WASRoP-Invited-04-slides.pdf [2]: H. Rebecq, D. Gehrig, D. Scaramuzza, ESIM: an Open Event Camera Simulator, Conference on Robot Learning, 2018

**Contact Details:** Please send your CV and transcript to: Dario Brescianini, brescianini (at) ifi (dot) uzh (dot) ch
**Thesis Type:** Semester Project / Bachelor Thesis
See project on SiROP

### Implementation of Feature Detector, Tracker, Matcher on CUDA - Available

**Description:** The goal of this thesis is to implement a Feature Detector (ex. Harris, Fast, SUSAN, LoG, DoG), Descriptor (SURF, BRISK, BRIEF, ORB, FREAK) and Matcher/Tracker using the nVidia Cuda Framework for high-efficiency real-time execution on a nVidia Jetson TX2 computer. Focus should be laid on exploiting new computational architectures arising from Machine Learning, such as fast convolutions and GPU-aided computation.

**Goal:** The thesis should provide a comparison of execution speed with respect to execution on an ARM CPU. The end

### High-Performance Simulation of Spiking Neural Network on GPUs - Available

**Description:** One major complication in research of biologically-inspired spiking neural Networks (SNNs) is simulation performance on conventional hardware (CPU/GPU). Computation in SNNs is dominated by operations on sparse tensors but usually this potential benefit is ignored to save development time. However, the exploitation of sparsity could be beneficial to scale simulation of SNNs to larger datasets. Requirements: - Experience with deep learning frameworks (e.g. TensorFlow or PyTorch) - Excellent programming skills and experience in CUDA

**Goal:** In this project, you will leverage sparse computation to develop high-performance simulations of SNNs that can be used for optimization. This will help to scale experiments and drastically improve results obtained by SNNs.

**Contact Details:** Mathias Gehrig, mgehrig (at) ifi (dot) uzh (dot) ch
**Thesis Type:** Semester Project
See project on SiROP

### Reinforcement learning for car racing in GTS - Available

**Description:** The Gran Turismo SPORT (GTS) for the Sony Playstation 4 is a modern car racing game which can simulate accurate physics of the car racing environment, including detailed models of various race cars and racing tracks. Data-driven approaches (e.g. reinforcement learning) have shown to be a promising method for solving such a continuous control problem where the dynamics of the environment are hard to model. We trained a neural network policy end-to-end using policy search methods for car racing in GTS. The policy has shown results comparable with human players and exceeded the performance of model-based agents. However, the model-free agent requires a large number of sampling steps in the environment, and thus, suffers from high sample complexities. -- Programming experience in C++/Python and TensorFlow/Pytorch. Reinforcement learning, deep learning or machine learning experience.

**Goal:** The goal of this master thesis is to first investigate reinforcement learning methods (imitation learning, model-free, model-based, hybrid methods, etc) for car racing, and then, train an agent that can drive faster than human experts. Ultimately, we aim to race the car with other players on the same track such that it also learn how to avoid dynamic obstacles.

**Contact Details:** Yunlong Song (song@ifi.uzh.ch), Elia Kaufmann (ekaufmann@ifi.uzh.ch)
**Thesis Type:** Master Thesis
See project on SiROP

### Learning-Guided MPC Flight - Available

**Description:** Model predictive control (MPC) is a versatile optimization-based control method that allows to incorporate constraints directly into the control problem. The advantages of MPC can be seen in its ability to accurately control dynamical systems that include large time delays and high-order dynamics. Recent advances in compute hardware allow to run MPC even on compute-constrained quadrotors. While model-predictive control can deal with complex systems and constraints, it still assumes the existence of a reference trajectory. With this project we aim to guide the MPC to a feasible reference trajectory by using a neural

# 2-View Geometry: Recap

- **Depth from stereo (i.e., stereo vision)**
  - **Assumptions:** K, T and R are known.
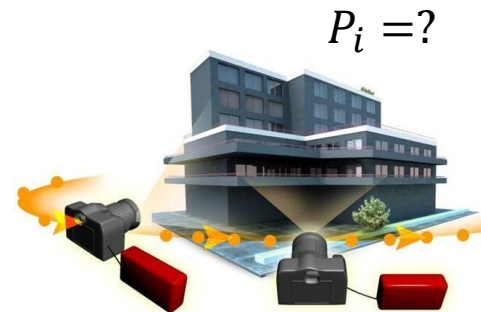  - **Goal**: Recover the 3D structure from images

$$P_i = ?$$
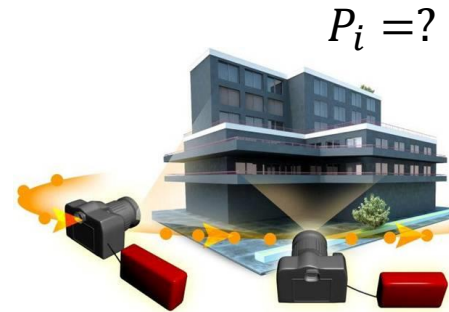
$$K_1, R_1, T_1$$

$$K_2, R_2, T_2$$

- **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

$$P_i = ?$$

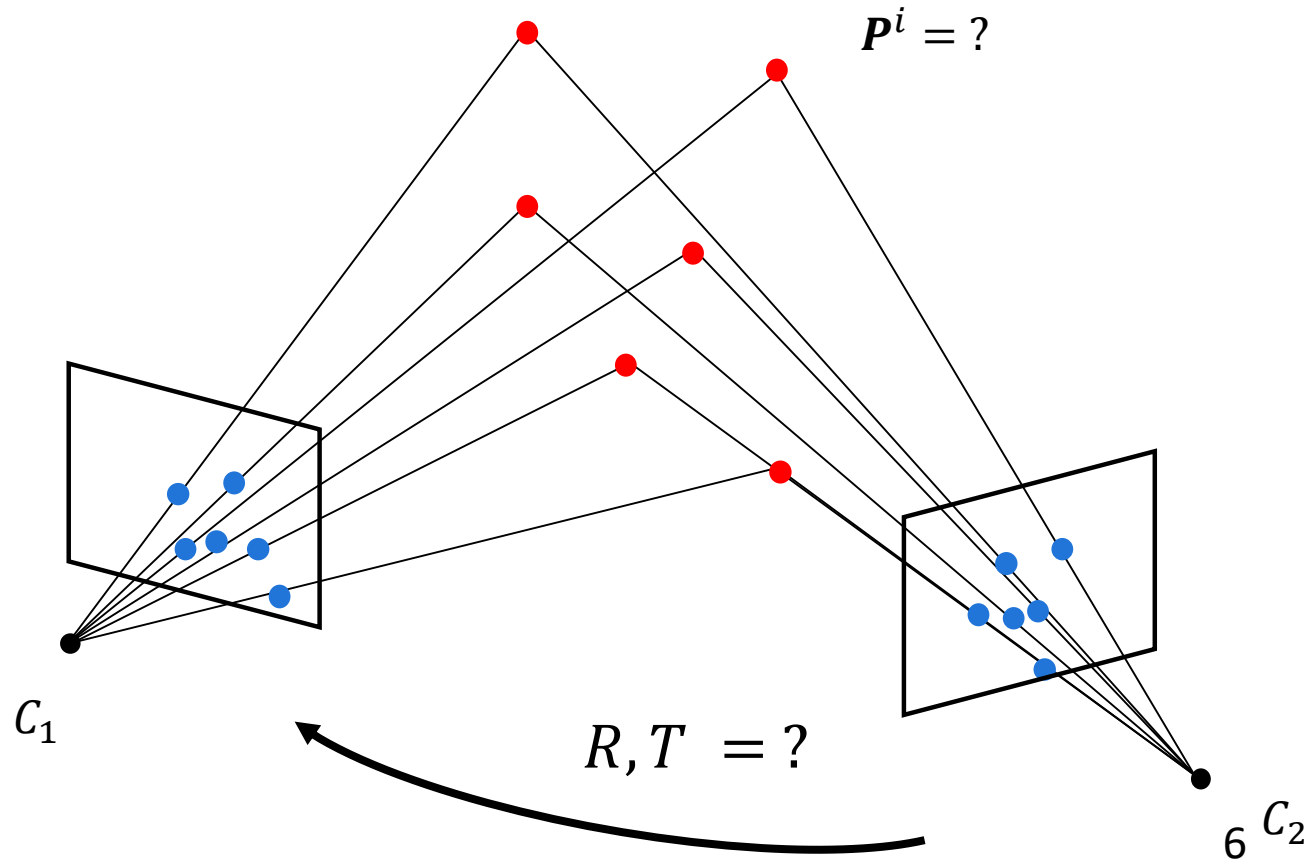$$K_1, R_1, T_1 = ?$$

$$K_2, R_2, T_2 = ?$$

4

# 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:

$$\lambda_1 \begin{bmatrix} u^i_1 \\ v^i_1 \\ 1 \end{bmatrix} = K_1 [I|0] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}$$

$$\lambda_2 \begin{bmatrix} u^i_2 \\ v^i_2 \\ 1 \end{bmatrix} = K_2 [R|T] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}$$



$\boldsymbol{P}^i = ?$

$C_1$

$R, T \; = ?$

$C_2$

# Structure from Motion (SFM)

- Two variants exist:

  - **Calibrated** camera(s) $\Rightarrow K_1, K_2$ **are known**

  - **Uncalibrated** camera(s) $\Rightarrow K_1, K_2$ **are unknown**



$P^i = ?$
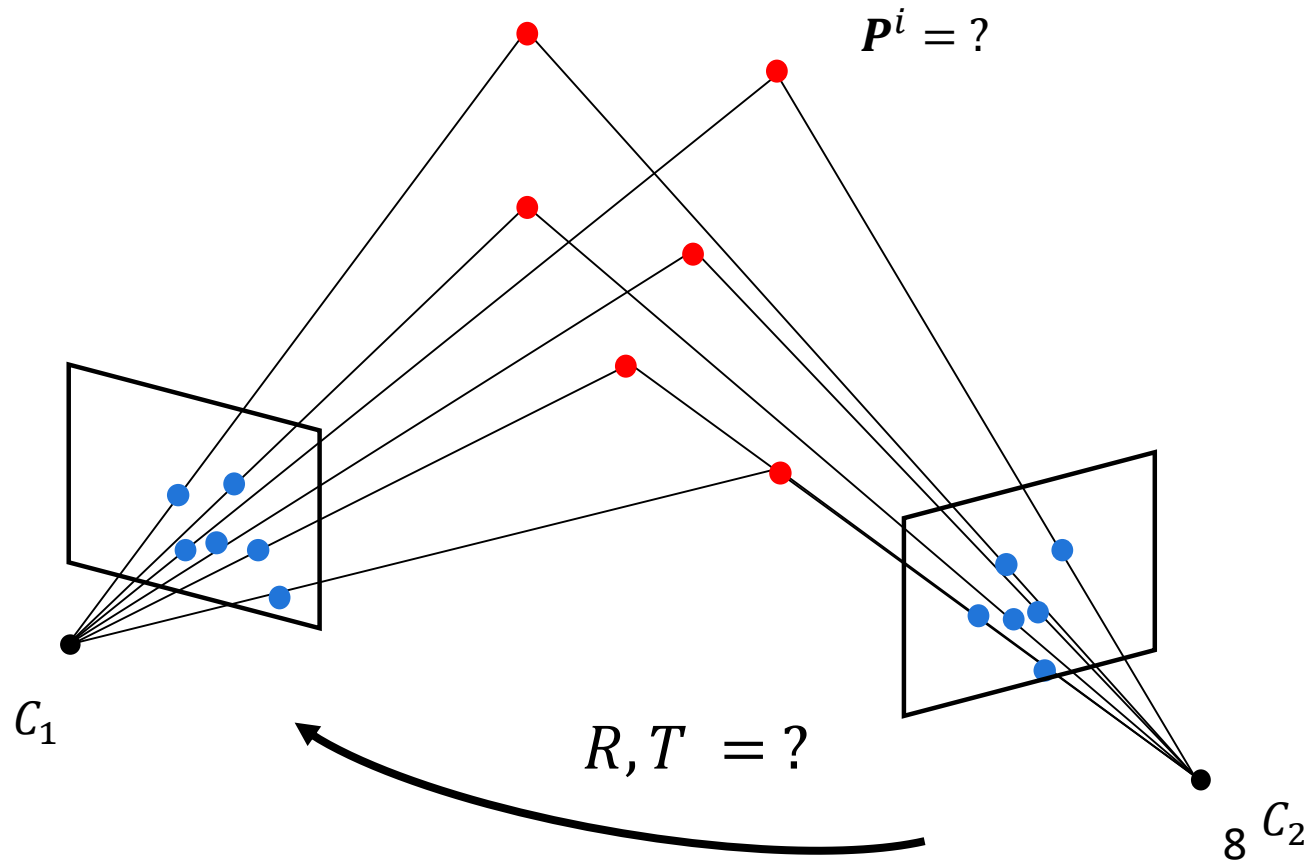
$C_1$

$R, T = ?$

$C_2$
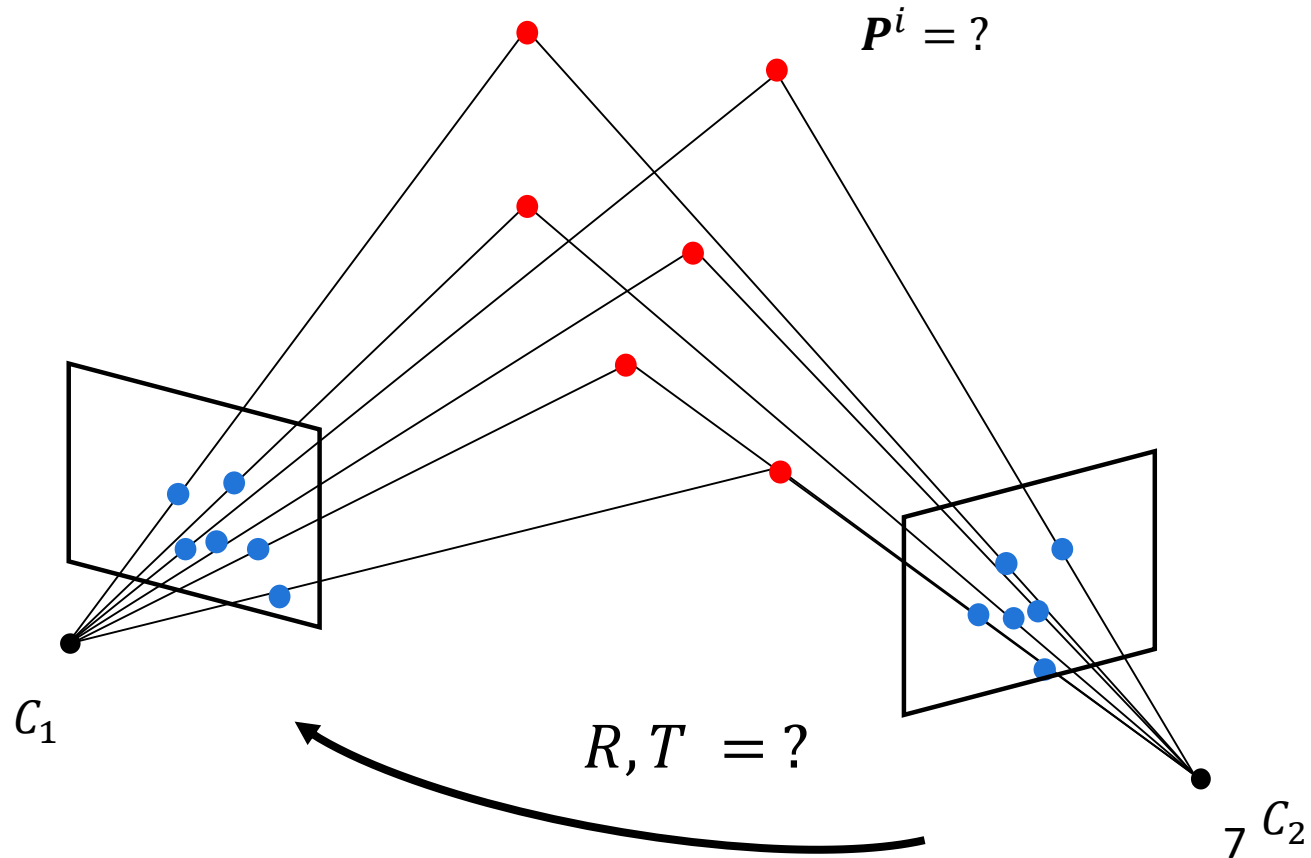
# Structure from Motion (SFM)

- 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

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

$$\lambda_1 \begin{bmatrix} \bar{u}^i_1 \\ \bar{v}^i_1 \\ 1 \end{bmatrix} = [I|0] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}$$

$$\lambda_2 \begin{bmatrix} \bar{u}^i_2 \\ \bar{v}^i_2 \\ 1 \end{bmatrix} = [R|T] \cdot \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}$$

$\boldsymbol{P}^i = ?$

$C_1$

$R, T = ?$

$C_2$

# Scale Ambiguity

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



Similarity
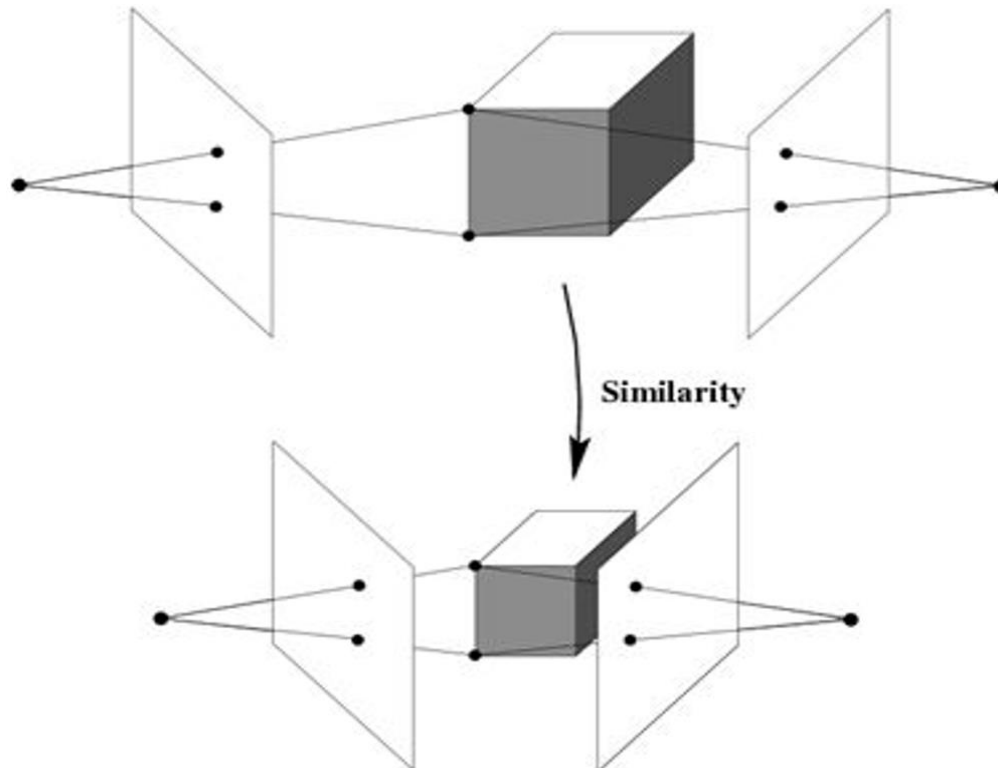
# Scale Ambiguity

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

- Does a solution exist?
  - If and only if the *number of independent equations $\geq$ number of unknowns* $\Rightarrow 4n \geq 5 + 3n \Rightarrow$ **$n \geq 5$**
  - First attempt to identify the solutions by Kruppa in 1913 (see slide 17).

E. Kruppa, Zur Ermittlung eines Objektes aus zwei Perspektiven mit Innerer Orientierung, *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. IIa.*, 1913. – English Translation plus original paper by Guillermo Gallego, Arxiv, 2017
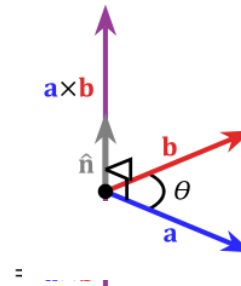
# Cross Product (or Vector Product)

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

- 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:

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

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

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

- So **c** is perpendicular to both **a** and **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**
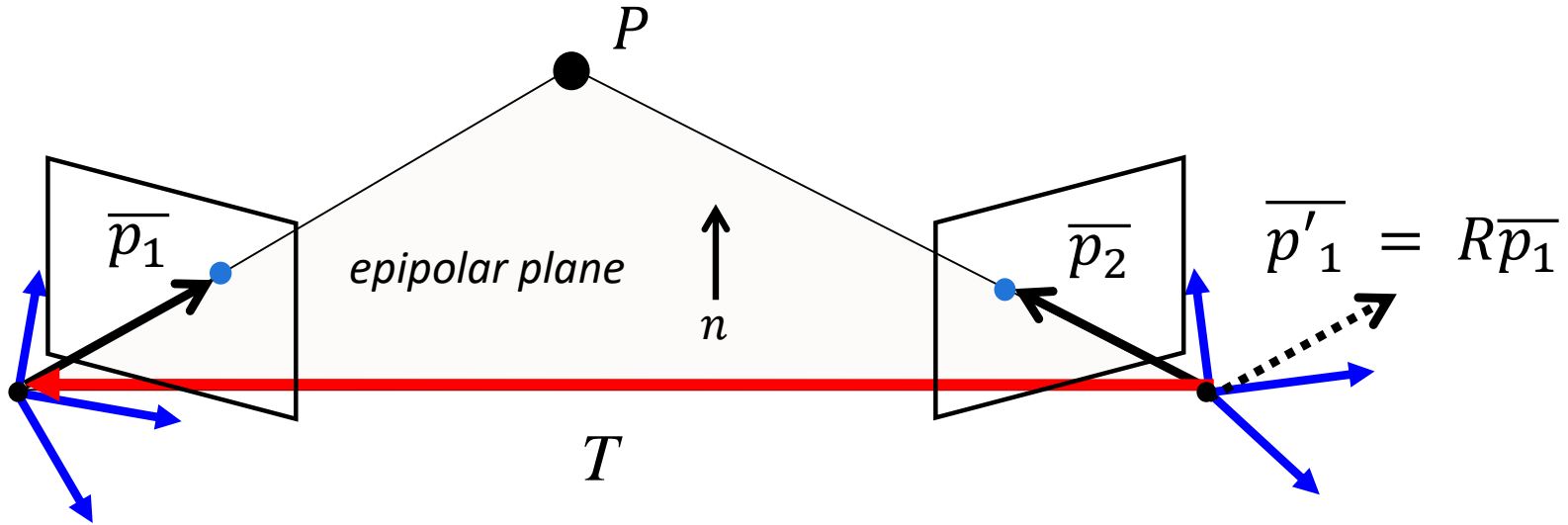
$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}]_\times \mathbf{b}$$

Can we solve the estimation of relative motion (R,T) independently of the estimation of the structure (3D points)?

The next couple of slides prove that this is possible. Once R,T are known, the 3D points can be triangulated using the triangulation algorithms from Lecture 7 (slides 30-36)

# Epipolar Geometry

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



*epipolar plane*

$P$

$n$

$\overline{p_1}$  $\overline{p_2}$

$\overline{p'}_1 = R\overline{p_1}$

$T$

$\overline{p}_1, \overline{p}_2, T$ are coplanar:

$$\overline{p}_2^T \cdot n = 0 \;\Rightarrow\; \overline{p}_2^T \cdot (T \times \overline{p'}_1)) = 0 \quad \Rightarrow\; \overline{p}_2^T (T \times (R\overline{p}_1)) = 0$$

$$\Rightarrow \overline{p}_2^T [T]_\times R \; \overline{p}_1 = 0 \quad \Rightarrow \boxed{\overline{p}_2^T E \; \overline{p}_1 = 0 \quad \textit{epipolar constraint}}$$

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

14

# Epipolar Geometry

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

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

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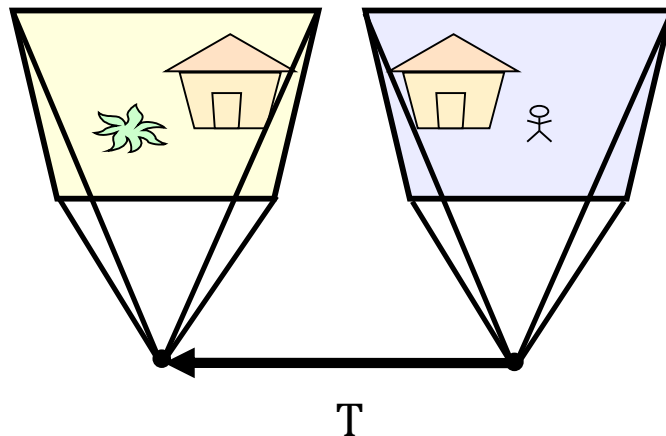


$$\mathbf{R} = \mathbf{I}_{3\times 3}$$

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

# 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 and that there can be at up to 11 solutions.

- However, in 1988, Demazure showed that there are actually at most **10 distinct solutions**.

- In 1996, Philipp proposed an iterative algorithm to find these solutions.

- 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.
D. Nister, An Efficient Solution to the Five-Point Relative Pose Problem, PAMI, 2004, PDF

# The 8-point algorithm

- The Essential matrix E is defined by

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

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

  provides a linear equation:

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

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

# The 8-point algorithm

- For $n$ points, we can write

$$
\begin{bmatrix}
\bar{u}_2^{\,1}\bar{u}_1^{\,1} & \bar{u}_2^{\,1}\bar{v}_1^{\,1} & \bar{u}_2^{\,1} & \bar{v}_2^{\,1}\bar{u}_1^{\,1} & \bar{v}_2^{\,1}\bar{v}_1^{\,1} & \bar{v}_2^{\,1} & \bar{u}_1^{\,1} & \bar{v}_1^{\,1} & 1 \\
\bar{u}_2^{\,2}\bar{u}_1^{\,2} & \bar{u}_2^{\,2}\bar{v}_1^{\,2} & \bar{u}_2^{\,2} & \bar{v}_2^{\,2}\bar{u}_1^{\,2} & \bar{v}_2^{\,2}\bar{v}_1^{\,2} & \bar{v}_2^{\,2} & \bar{u}_1^{\,2} & \bar{v}_1^{\,2} & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\bar{u}_2^{\,n}\bar{u}_1^{\,n} & \bar{u}_2^{\,n}\bar{v}_1^{\,n} & \bar{u}_2^{\,n} & \bar{v}_2^{\,n}\bar{u}_1^{\,n} & \bar{v}_2^{\,n}\bar{v}_1^{\,n} & \bar{v}_2^{\,n} & \bar{u}_1^{\,n} & \bar{v}_1^{\,n} & 1
\end{bmatrix}
\begin{bmatrix}
e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33}
\end{bmatrix} = 0
$$

Q (this matrix is **known**)

$\bar{E}$ (this matrix is **unknown**)

# The 8-point algorithm

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

**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 $||Qx||^2 = x^T Q^T Qx$).
- 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 how 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

$$\overline{\boldsymbol{p}}_2^{\top} \cdot \boldsymbol{E}\overline{\boldsymbol{p}}_1 = \left\| \overline{\boldsymbol{p}}_2 \right\| \left\| \boldsymbol{E}\overline{\boldsymbol{p}}_1 \right\| \cos(\theta)$$

We can see that this product depends on the angle $\theta$ between $\overline{\boldsymbol{p}}_1$ and the normal $n = \boldsymbol{E}\boldsymbol{p}_1$ to the epipolar plane. It is non zero when $\overline{\boldsymbol{p}}_1, \overline{\boldsymbol{p}}_2$, and $\boldsymbol{T}$ are not coplanar.
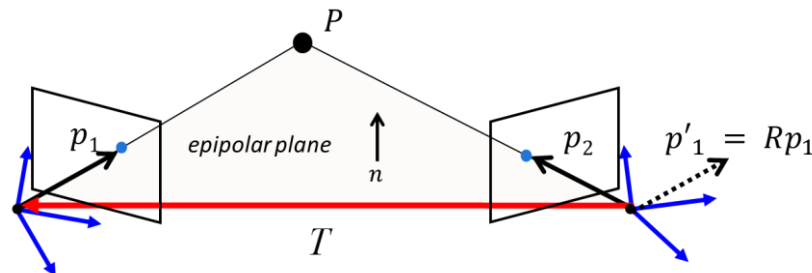
# Extract R and T from E
## (this slide will not be asked at the exam)

- Singular Value Decomposition: $E = U \sum V^T$

- Enforcing rank-2 constraint: set smallest singular value of $\sum$ to 0:

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

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

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

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

$$t = K_2 \hat{t}$$

$$R = K_2 \hat{R} K_1^{-1}$$

# 4 possible solutions of R and T



(a)

(b)
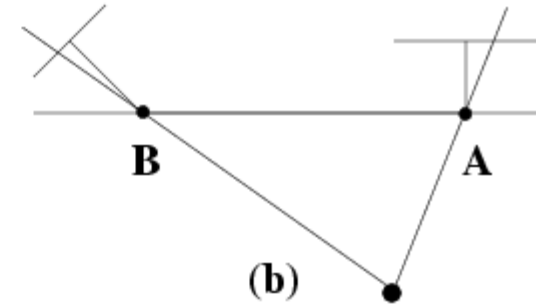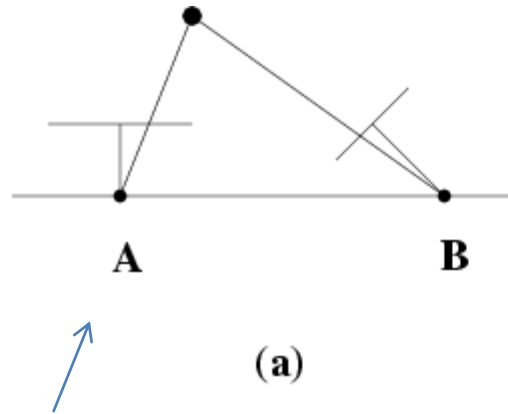
Only one solution where points are in front of both cameras

(c)

(d)
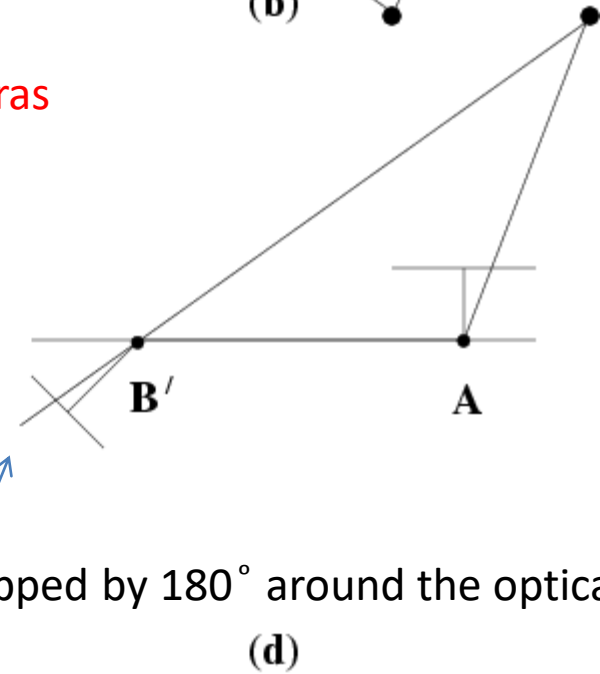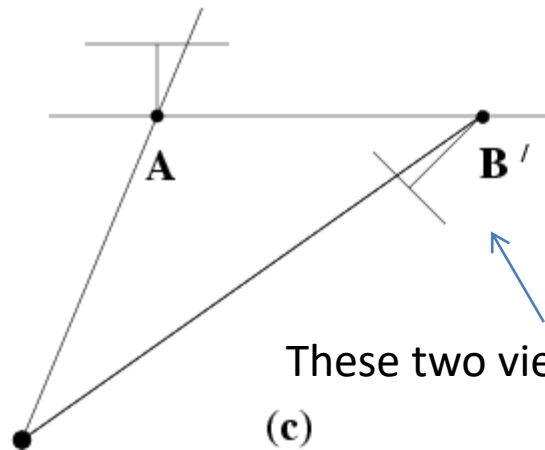
These two views are flipped by 180° around the optical axis

# Structure from Motion (SFM)

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

$$P^i = ?$$

$$C_1$$

$$R, T = ?$$

$$C_2$$
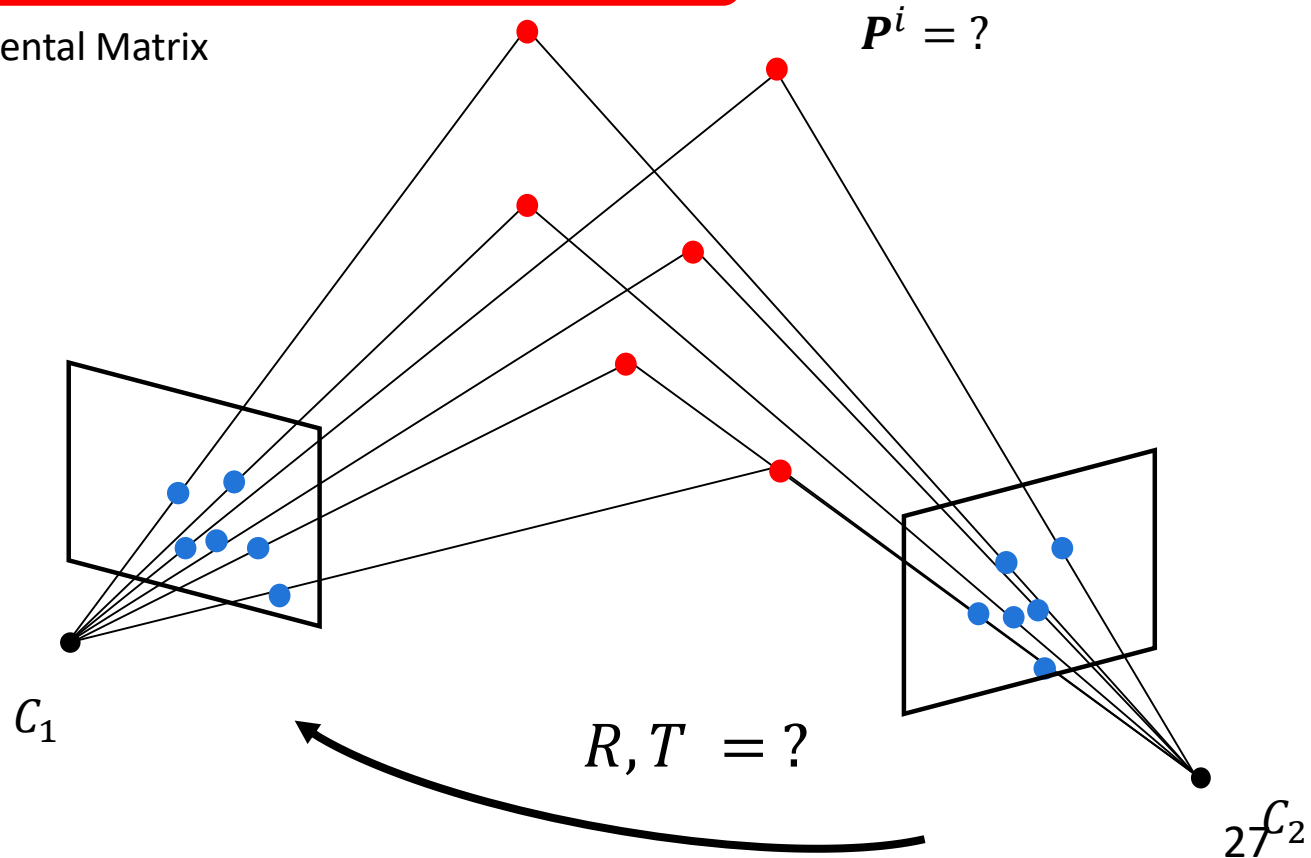
# The Fundamental Matrix

- Before, we assumed to know the camera intrinsic parameters and we used normalized image coordinates to get the epipolar constraint for **calibrated cameras**:

$$\begin{bmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{bmatrix} = K_1^{-1} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} \qquad \begin{bmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{bmatrix} = K_2^{-1} \begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}$$

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

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

# The Fundamental Matrix

- By substituting the definition of normalized coordinates into the epipolar constraint, we get the epipolar constraint for **uncalibrated cameras:**

$$\begin{bmatrix} u_2^i \\ v_2^i \\ 1 \end{bmatrix}^{\mathrm{T}} \mathrm{K}_2^{-\mathrm{T}} \ \mathrm{E} \ \mathrm{K}_1^{-1} \begin{bmatrix} u_1^i \\ v_1^i \\ 1 \end{bmatrix} = 0$$

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

Fundamental Matrix

$$\left. \begin{matrix} F = \mathrm{K}_2^{-\mathrm{T}} \ \mathrm{E} \ \mathrm{K}_1^{-1} \\ E = [T]_\times R \end{matrix} \right\} \Rightarrow \boxed{F = \mathrm{K}_2^{-\mathrm{T}} \ [T]_\times R \ \mathrm{K}_1^{-1}}$$

# 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:

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

- Advantage: we work directly in pixel coordinates

# Problem with 8-point algorithm

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

33

# Problem with 8-point algorithm

$$\begin{bmatrix} 250906.36 & 183269.57 & 921.81 & 200931.10 & 146766.13 & 738.21 & 272.19 & 198.81 & 1.00 \\ 2692.28 & 131633.03 & 176.27 & 6196.73 & 302975.59 & 405.71 & 15.27 & 746.79 & 1.00 \\ 416374.23 & 871684.30 & 935.47 & 408110.89 & 854384.92 & 916.90 & 445.10 & 931.81 & 1.00 \\ 191183.60 & 171759.40 & 410.27 & 416435.62 & 374125.90 & 893.65 & 465.99 & 418.65 & 1.00 \\ 48988.86 & 30401.76 & 57.89 & 298604.57 & 185309.58 & 352.87 & 846.22 & 525.15 & 1.00 \\ 164786.04 & 546559.67 & 813.17 & 1998.37 & 6628.15 & 9.86 & 202.65 & 672.14 & 1.00 \\ 116407.01 & 2727.75 & 138.89 & 169941.27 & 3982.21 & 202.77 & 838.12 & 19.64 & 1.00 \\ 135384.58 & 75411.13 & 198.72 & 411350.03 & 229127.78 & 603.79 & 681.28 & 379.48 & 1.00 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

~10000  ~10000  ~100  ~10000  ~10000  ~100  ~100  ~100  1

⚠ Orders of magnitude difference
between column of data matrix
→ 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, PAMI'97]

34

# Normalized 8-point algorithm (1/3)

- This can be fixed using a normalized 8-point algorithm [Hartley'97], 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

(0,0)            (700,0)

$$\begin{bmatrix} \dfrac{2}{700} & 0 & -1 \\ 0 & \dfrac{2}{500} & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

(-1,-1)            (1,-1)

(0,0)

(0,500)            (700,500)

(-1,1)            (1,1)

Hartley, In defense of the eight-point algorithm, PAMI'97, PDF

35
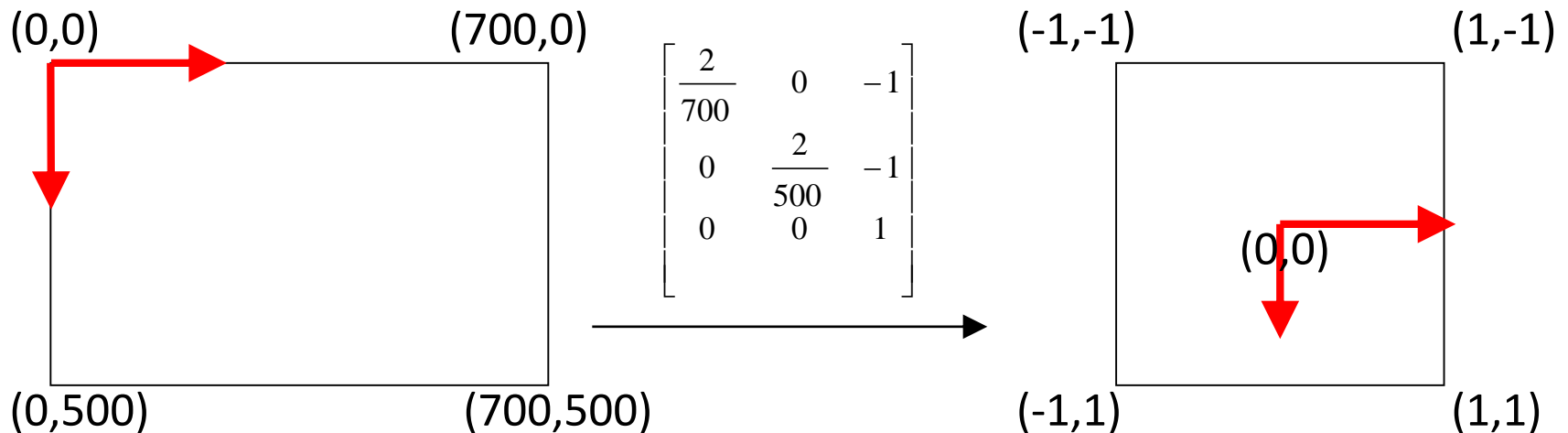
# Normalized 8-point algorithm (2/3)

- In the original 1997 paper, Hartley proposed to rescale the two point sets such that the centroid of each set is 0 and the mean standard deviation $\sqrt{2}$, so that the "average" point is equal to $[0, 0, 1]^{\mathsf{T}}$ (in homogeneous coordinates).

- This can be done for every point as follows:

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

where $\mu = (\mu_x, \mu_y) = \frac{1}{N}\sum_{i=1}^{n} p^i$ is the centroid and $\sigma = \frac{1}{N}\sum_{i=1}^{n}\left\|p^i - \mu\right\|^2$ is the mean standard deviation of the point set.

- This transformation can be expressed in matrix form using homogeneous coordinates:

$$\widehat{p^i} = \begin{bmatrix} \dfrac{\sqrt{2}}{\sigma} & 0 & -\dfrac{\sqrt{2}}{\sigma}\mu_x \\ 0 & \dfrac{\sqrt{2}}{\sigma} & -\dfrac{\sqrt{2}}{\sigma}\mu_y \\ 0 & 0 & 1 \end{bmatrix} p^i$$

Hartley, In defense of the eight-point algorithm, PAMI'97, [PDF](#)

# 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 **normalized** $\widehat{F}$ with 8-p. algorithm using normalized coordinates $\widehat{p_1}$, $\widehat{p_2}$

3.  Compute **unnormalized** F from $\widehat{F}$: $F = B_2^\top \widehat{F} B_1$

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

$$p_2^\top B_2^\top \qquad \widehat{F} \qquad B_1 p_1$$

$$F = B_2^\top \widehat{F} B_1$$

# Can $R, T, K_1, K_2$ be extracted from F?

- In general **no**: infinite solutions exist
- However, if the coordinates of the principal points of each camera are known and the two cameras have the same focal length $f$ in pixels, then $R, T, f$ can determined uniquely

# Comparison between Normalized and non-normalized algorithm
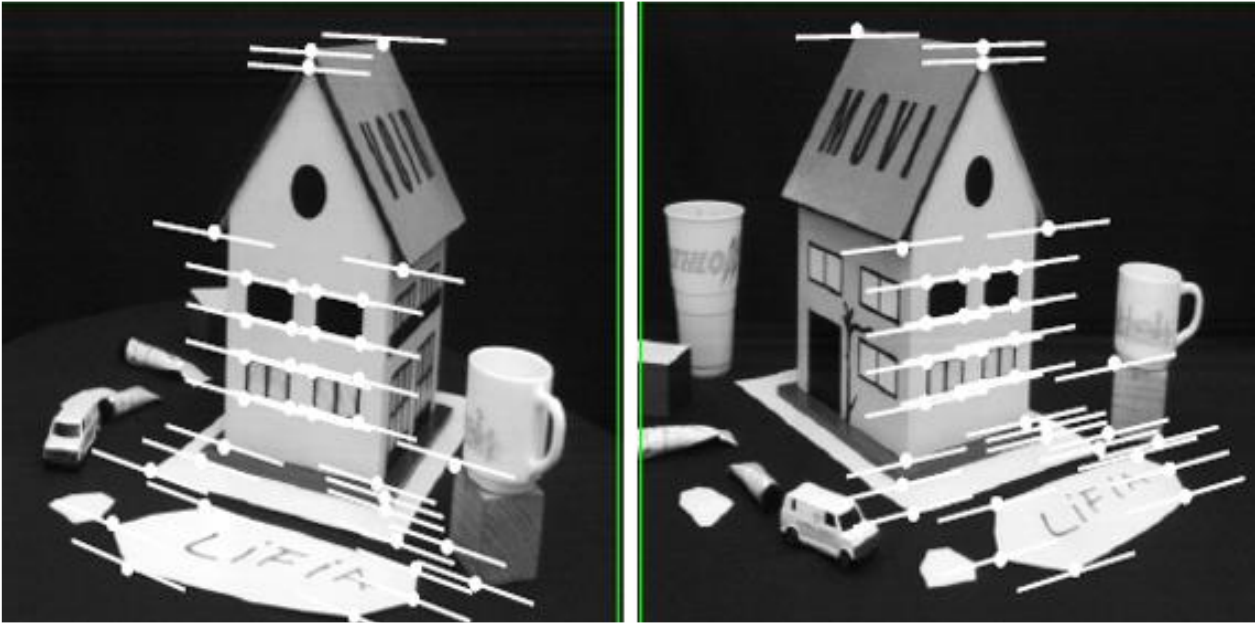


|  | 8-point | Normalized 8-point | Nonlinear refinement |
|---|---|---|---|
| Avg. Ep. Line Distance 1 | 2.33 pixels | 0.92 pixel | 0.86 pixel |

# Error Measures

➤ The **quality of the estimated Essential matrix** can be measured using different error metrics.

➤ The first one is the **algebraic error** that is defined directly by the Epipolar Constraint:

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

<span style="color:red">Remember Slide 24 for the geometrical interpretation of this error<br>What is the drawback with this error measure?</span>

➤ This error will exactly be 0 if E is computed from just 8 points (because in this case a non-overdetermined solution exists). For more than 8 points, it may 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 **squared cosines of the angle from the epipolar plane**: err $= \sum_i (\cos(\theta_i))^2$

➤ From slide 24, we obtain: $\cos(\theta) = \dfrac{\boldsymbol{p}^T_2 \cdot \boldsymbol{E}\boldsymbol{p}_1}{\|\boldsymbol{p}_2\|\|\boldsymbol{E}\boldsymbol{p}_1\|}$

$\mathbf{P} = ?$

$l_1 = E^T p_2$

*Epipolar plane*

$\mathbf{p}_1$

$\mathbf{p_2}$

$n$

$C_1$

$C_2$
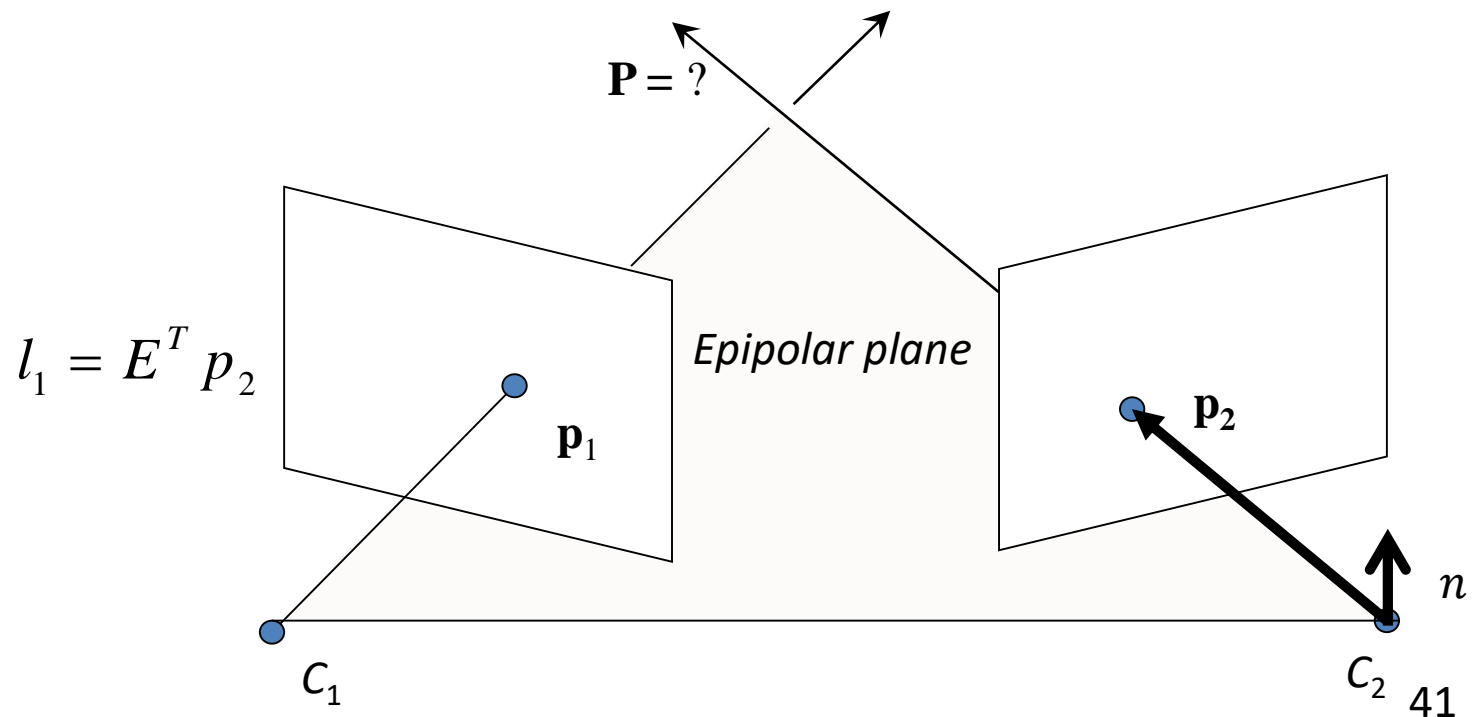
# Epipolar Line Distance

➢ 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



$\mathbf{P} = ?$

$l_1 = F^T p_2$

*epipolar plane*

$l_2 = Fp_1$

$\mathbf{p_1}$

$\mathbf{p_2}$

$C_1$

$C_2$
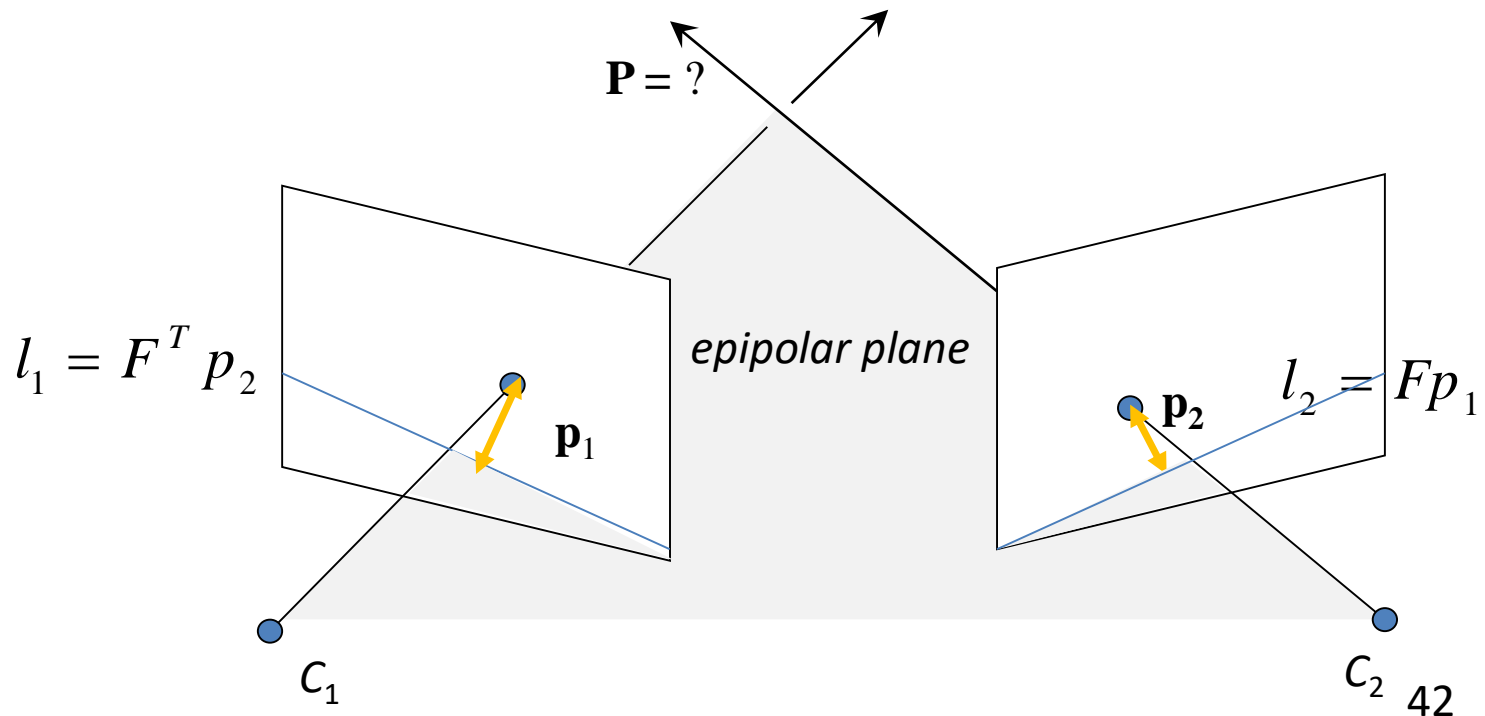
# Reprojection Error

➤ Sum of the **Squared Reprojection Errors**

$$err = \sum_{i=1}^{N} \left\| \mathbf{p}_1^i - \pi_1(\mathbf{P}^i) \right\|^2 + \left\| \mathbf{p}_2^i - \pi_2(\mathbf{P}^i, \mathbf{R}, \mathbf{T}) \right\|^2$$

➤ Computation is expensive because requires point triangulation

➤ **However it is the most popular because more accurate**



How to compute P? See Slides 30-36 of Lecture 07

$\mathbf{P} = ?$

**Reprojected point** $\pi_1(P)$

**Observed point**

$\mathbf{p}_1$

**Reprojected point**

**Observed point**

$\mathbf{p}_2$

$\pi_2(P)$

$C_1$

$C_2$

R, T
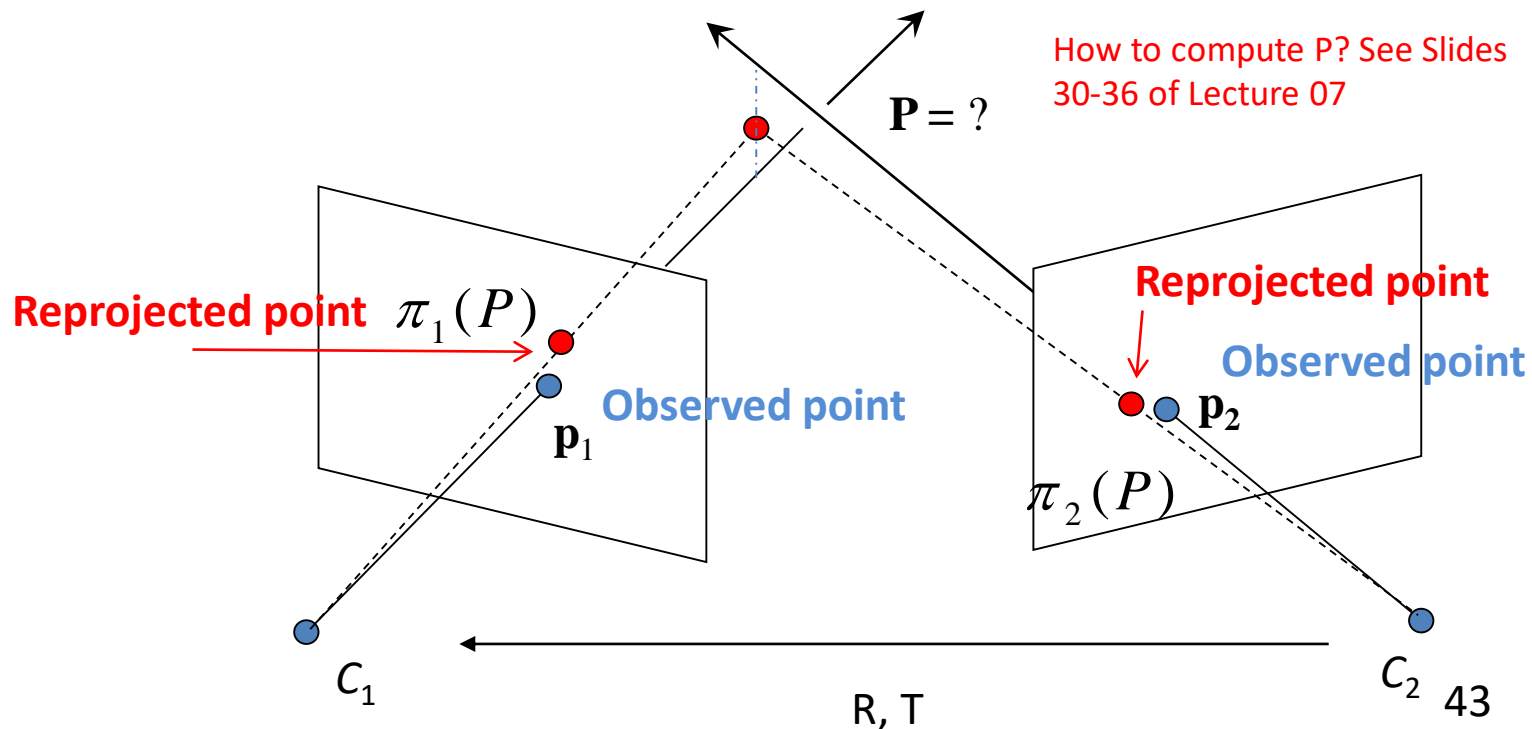
# Outline

- 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**
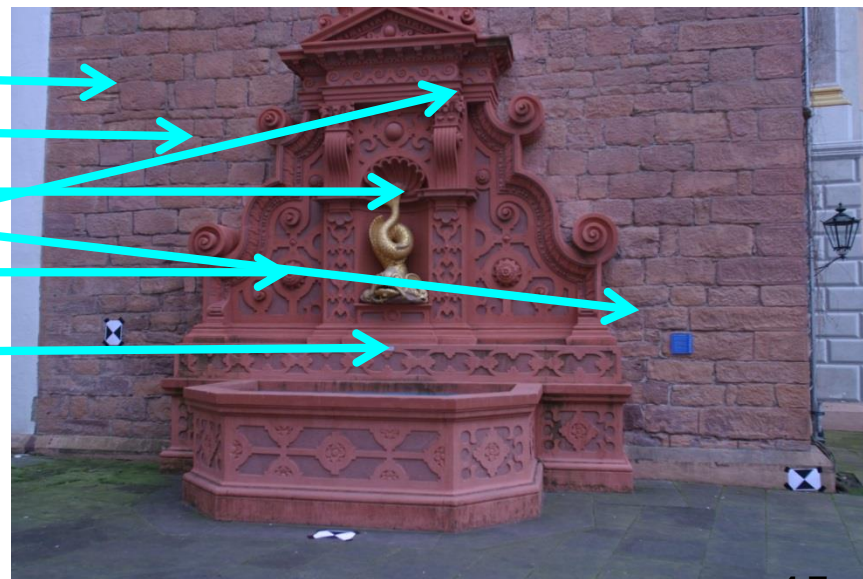


Image 1                    Image 2

# 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**
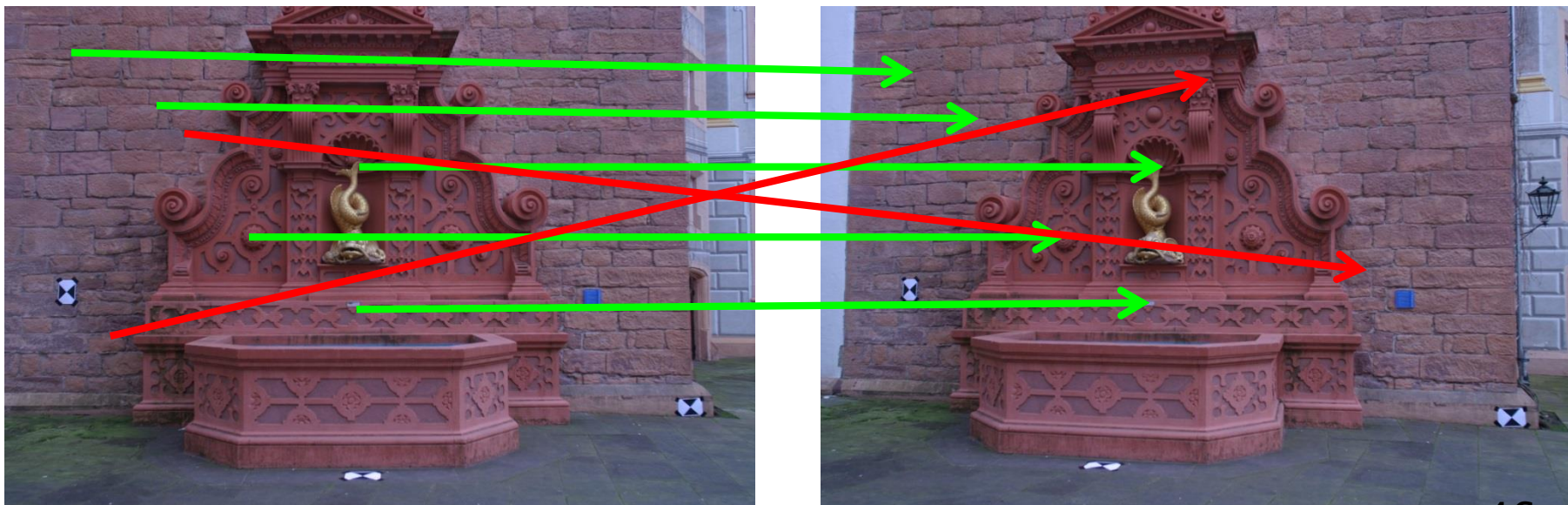


Image 1　　　　　　　　　　　　　　　　　Image 2

# Influence of Outliers on Motion Estimation
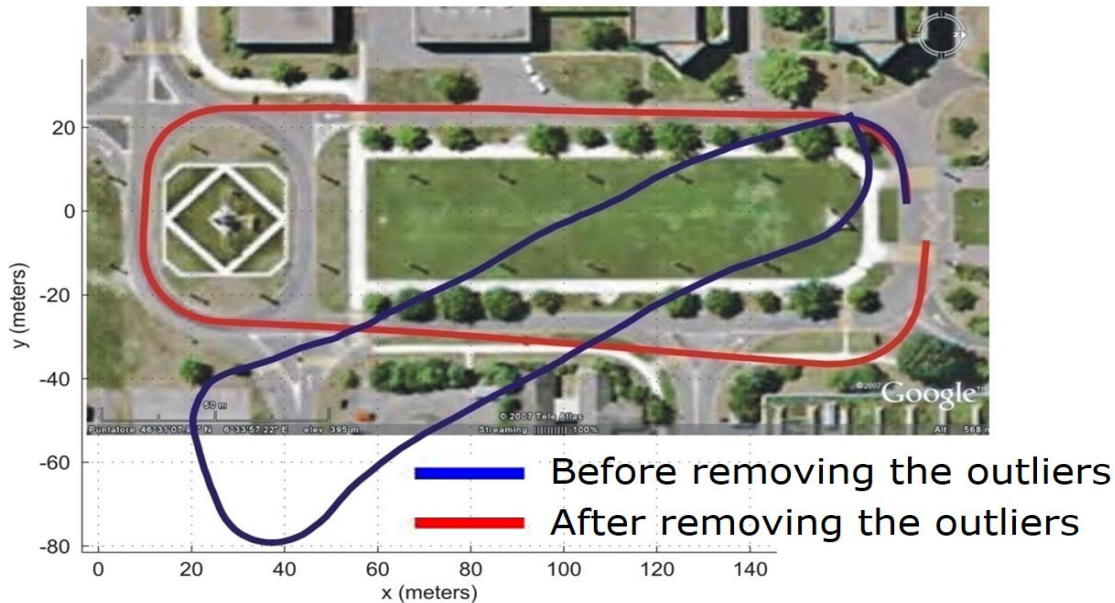


➢ Error at the loop closure: 6.5 m
➢ Error in orientation:        5 deg
➢ Trajectory length:           400 m

—— Before removing the outliers
—— After removing the outliers

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**
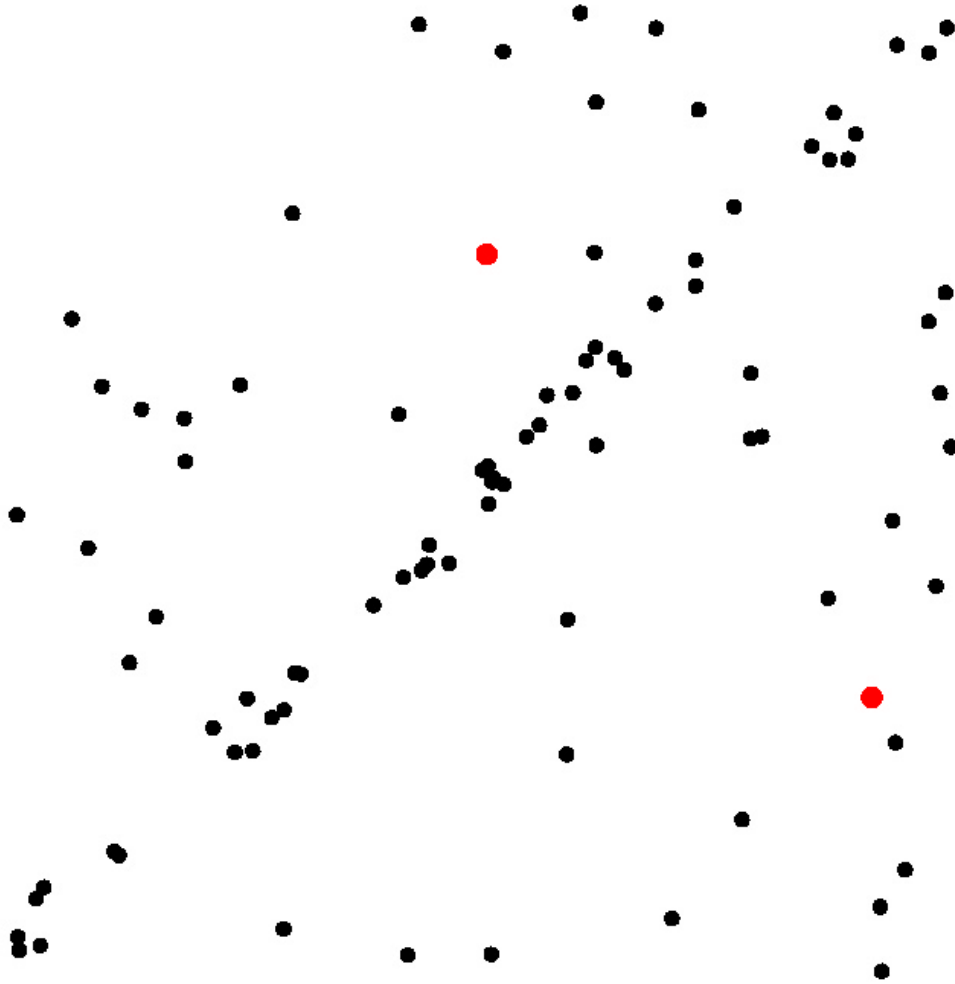
# RANSAC
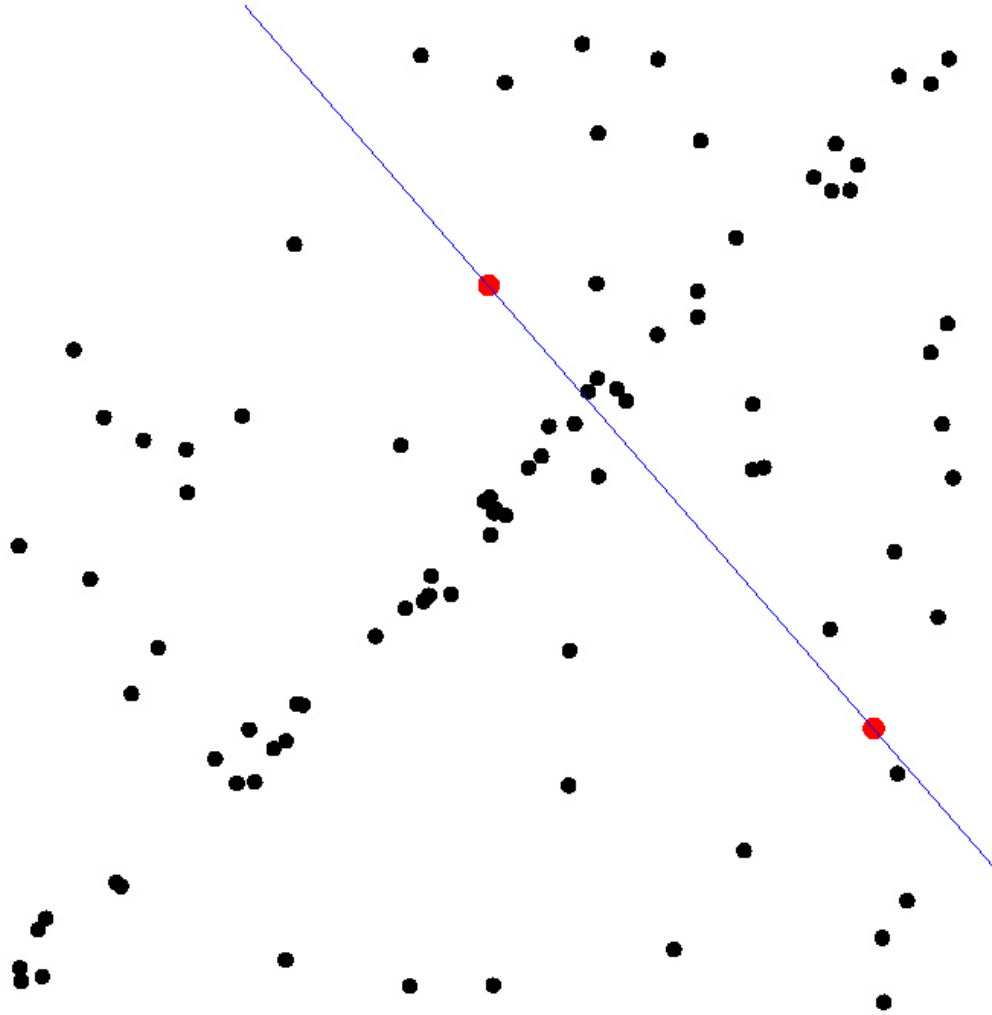


- Select sample of 2 points at random

- **Calculate model parameters that fit the data in the sample**

# RANSAC
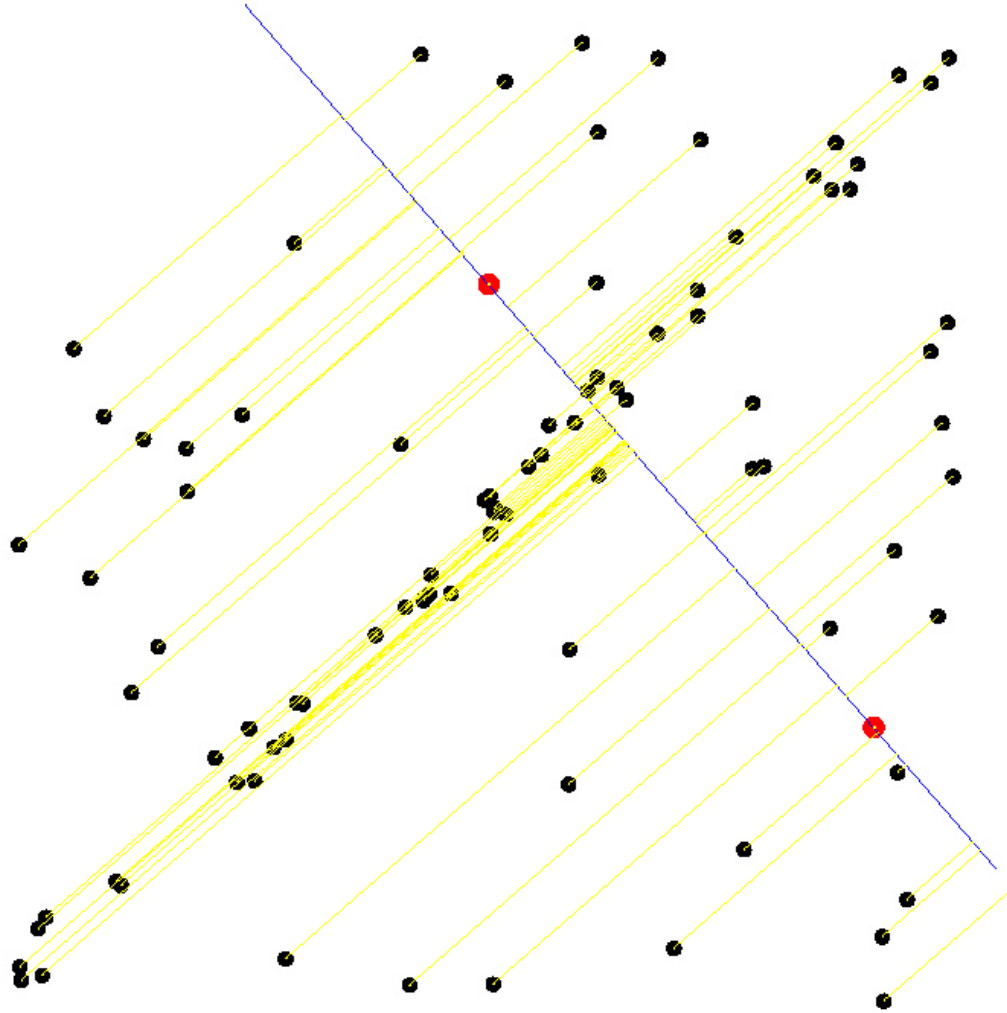


- Select sample of 2 points at random

- Calculate model parameters that fit the data in the sample

- **Calculate error function for each data point**

# RANSAC
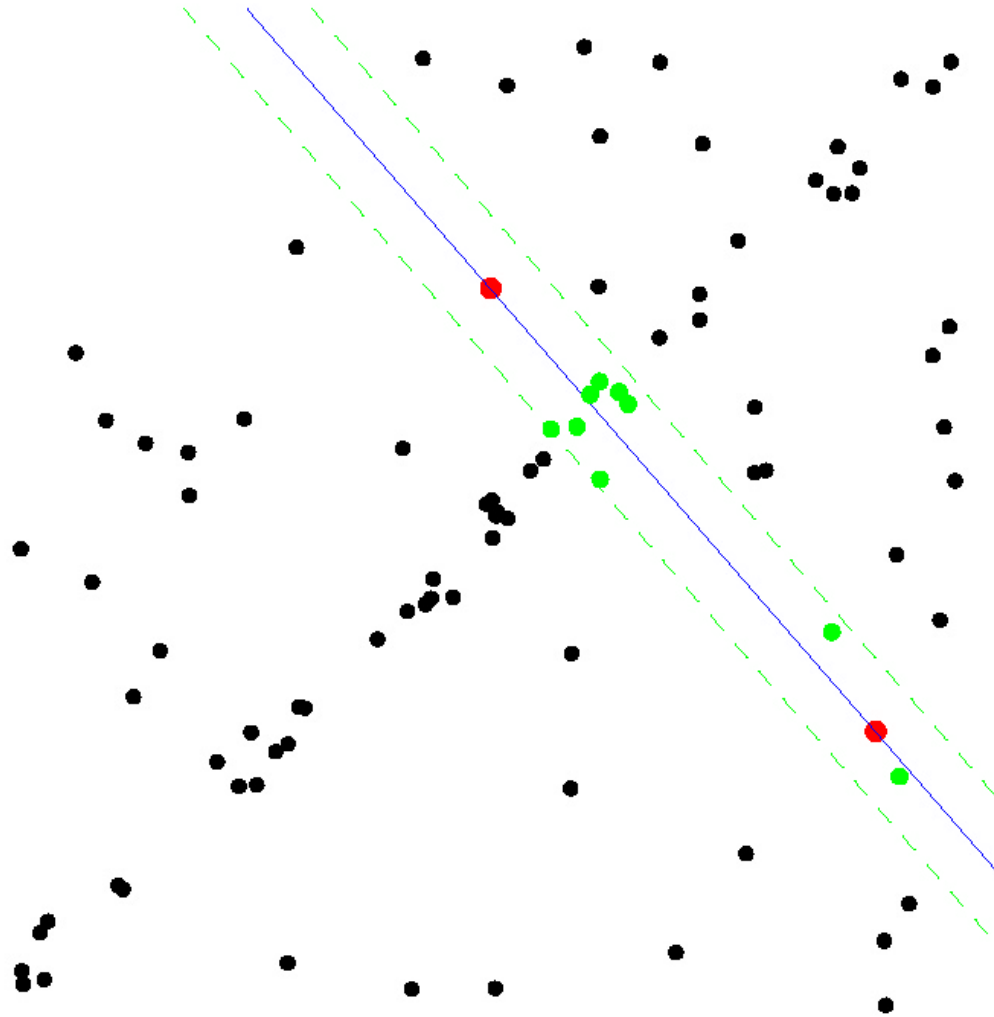


- Select sample of 2 points at random

- Calculate model parameters that fit the data in the sample

- Calculate error function for each data point

- **Select data that support current hypothesis**

# RANSAC
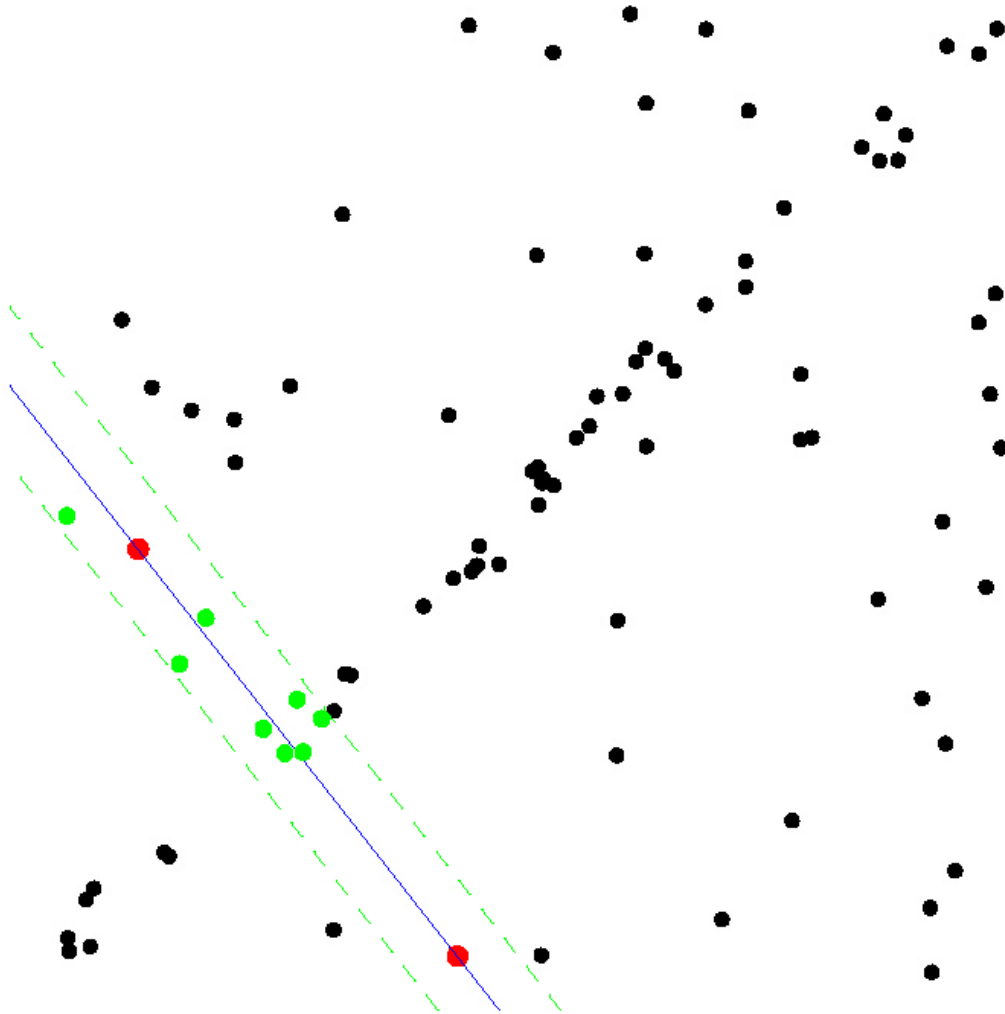


- Select sample of 2 points at random

- Calculate model parameters that fit the data in the sample

- Calculate error function for each data point

- Select data that support current hypothesis
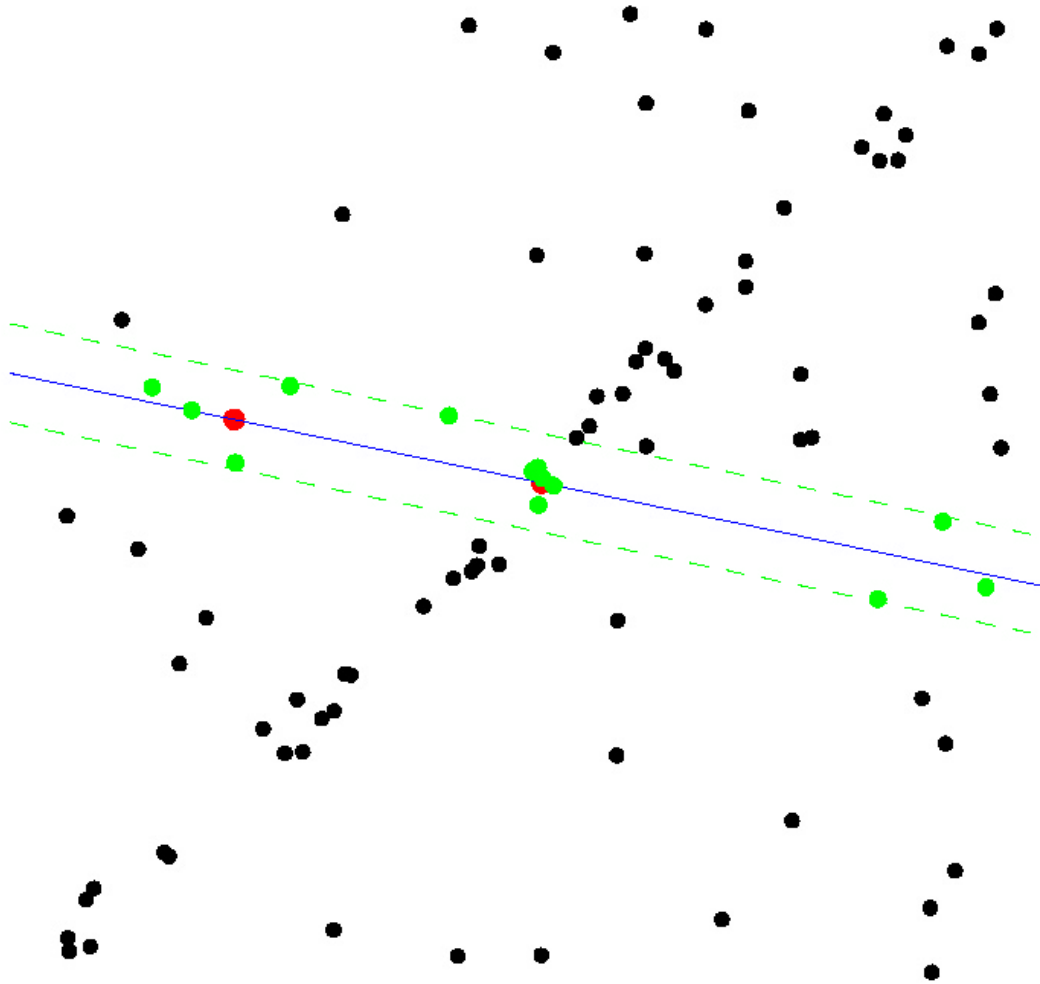
- **Repeat**

# 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 support current hypothesis

- **Repeat**
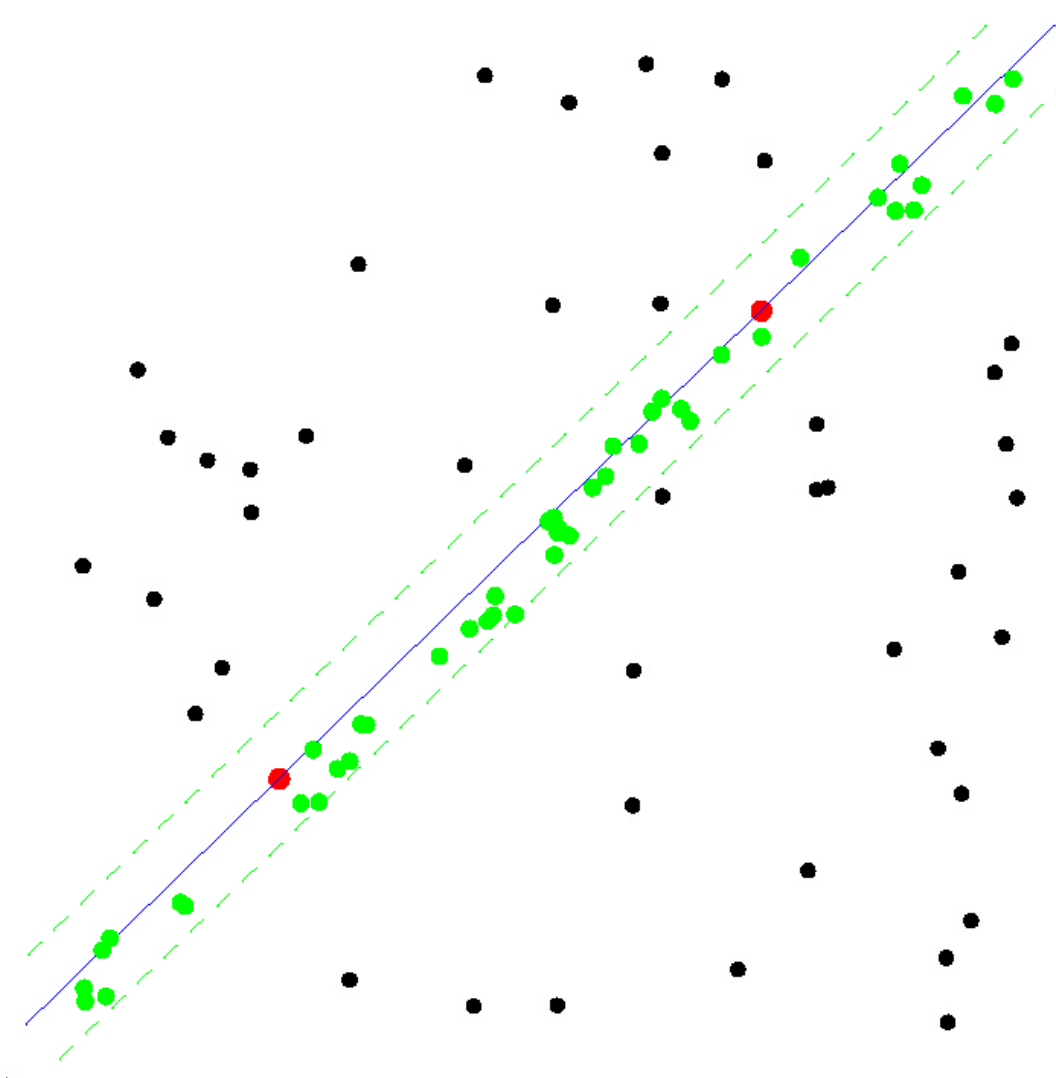
# RANSAC



**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 $k$ is

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

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

- Example: if we want a probability of success $p$=99% and we know that $w$=50% ⇨ $k$=16 iterations
  - these are **significantly fewer** than the number of **all possible combinations**!
  - **Notice: number of points does not influence minimum number of iterations $k$, only $w$ does!**

- In practice we only need a rough estimate of $w$. 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

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

# 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. Algebraic error ($\bar{p}_2^\top E \bar{p}_1 = 0$ or $p_2^\top F p_1 = 0$) (Slide 40)
   2. Directional error (Slide 41)
   3. Epipolar line distance (Slide 42)
   4. Reprojection error (Slide 43)

# 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
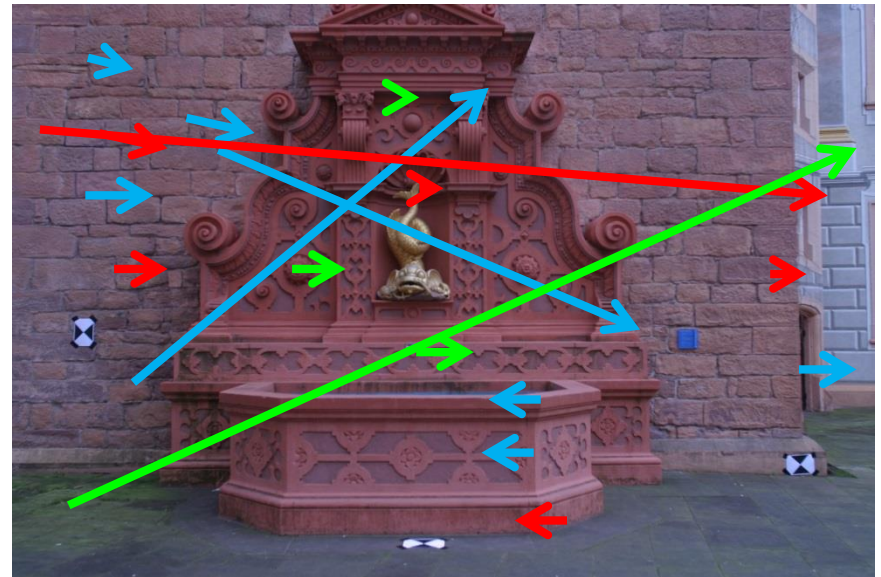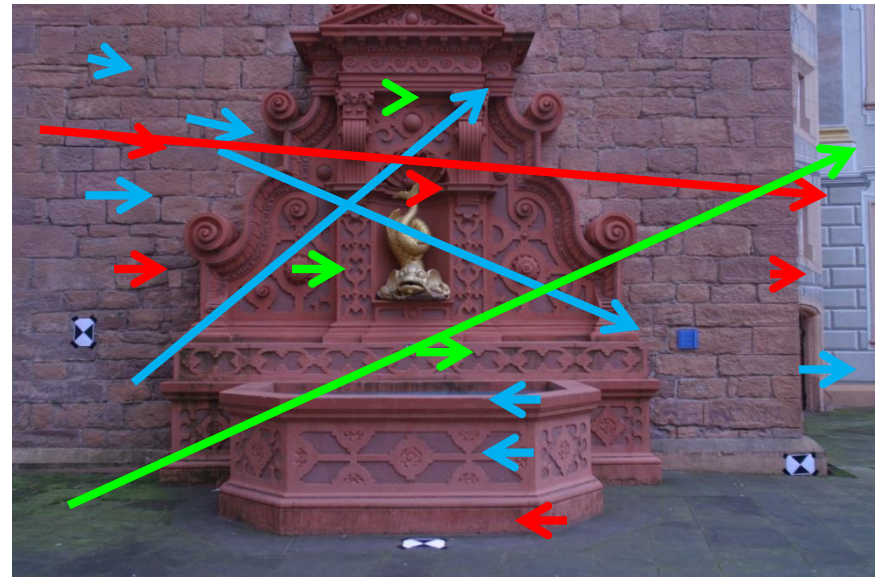


Image 1      Image 2

# 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 on 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 on 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 on the first image and use arrows to denote the *motion vectors* of the features
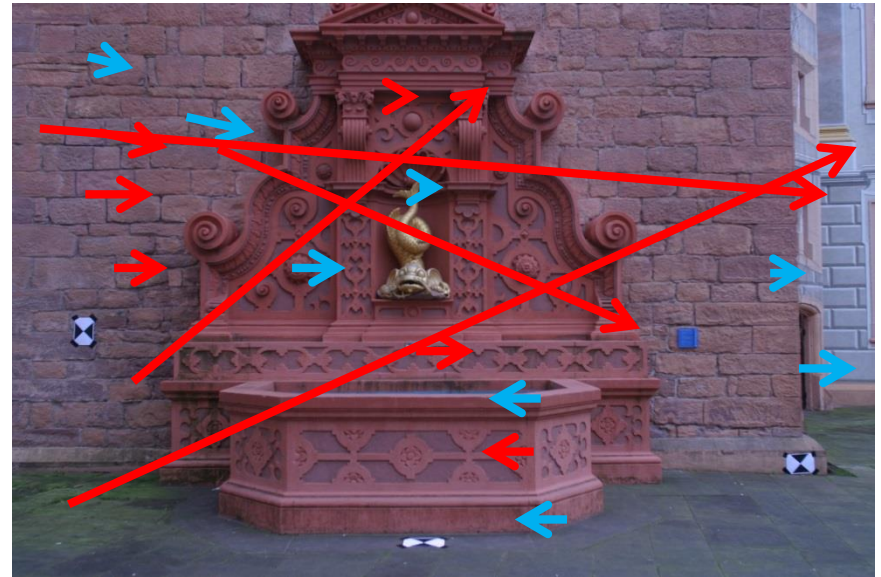
1. Randomly select 8 point correspondences

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 on the first image and use arrows to denote the *motion vectors* of the features

1. Randomly select 8 point correspondences

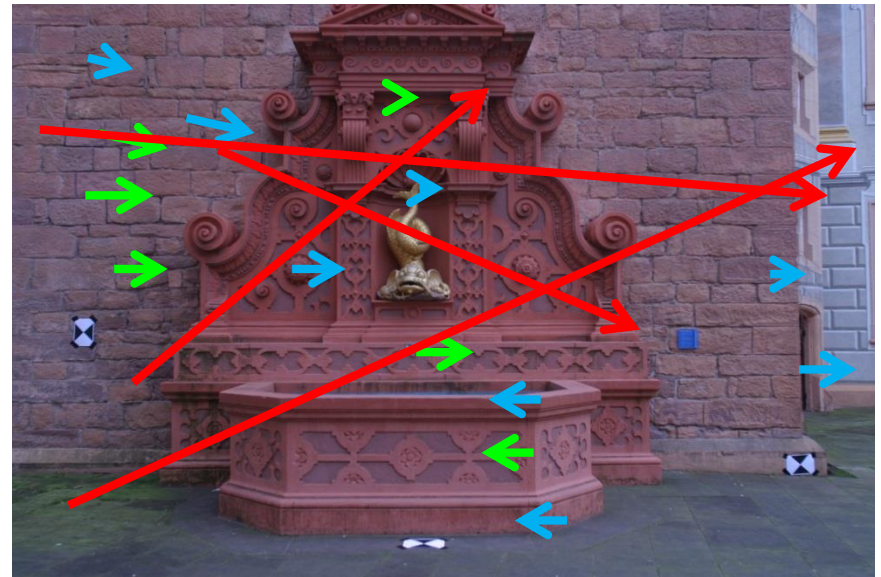2. Fit the model to all other points and count the inliers

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 on 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 on 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 on 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 on 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

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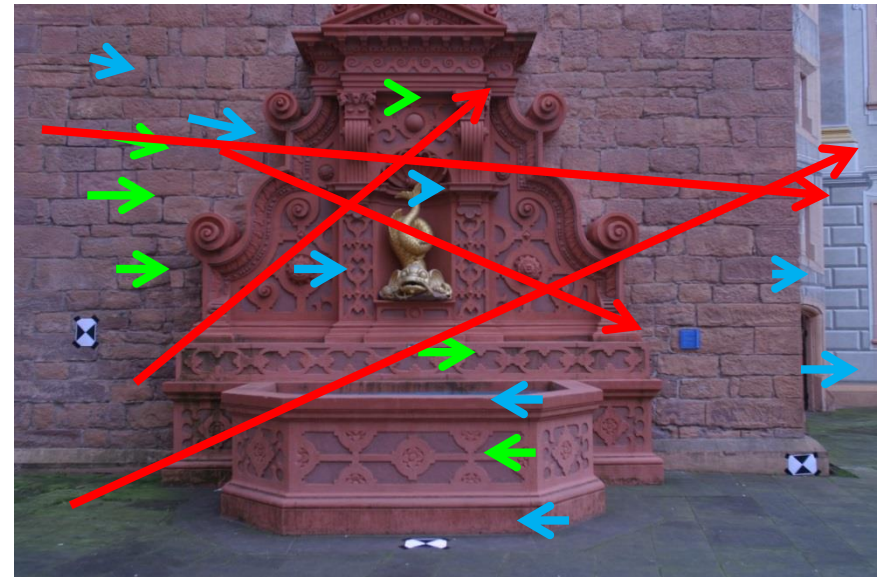


Image 1

# RANSAC iterations $k$ vs. $s$

$k$ is exponential in the number of points $s$ necessary to estimate the model:

- **8-point RANSAC**
  - Assuming
    - $p$ = 99%,
    - $\varepsilon$ = 50% (fraction of outliers)
    - $s$ = 8 points (8-point algorithm)
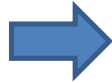
$$k = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} = 1177 \quad iterations$$

- **5-point RANSAC**
  - Assuming
    - $p$ = 99%,
    - $\varepsilon$ = 50% (fraction of outliers)
    - $s$ = 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
    - $p$ = 99%,
    - $\varepsilon$ = 50% (fraction of outliers)
    - $s$ = 2 points

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

# RANSAC iterations $k$ vs. $\varepsilon$

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



Number of RANSAC iterations vs fraction of outliers
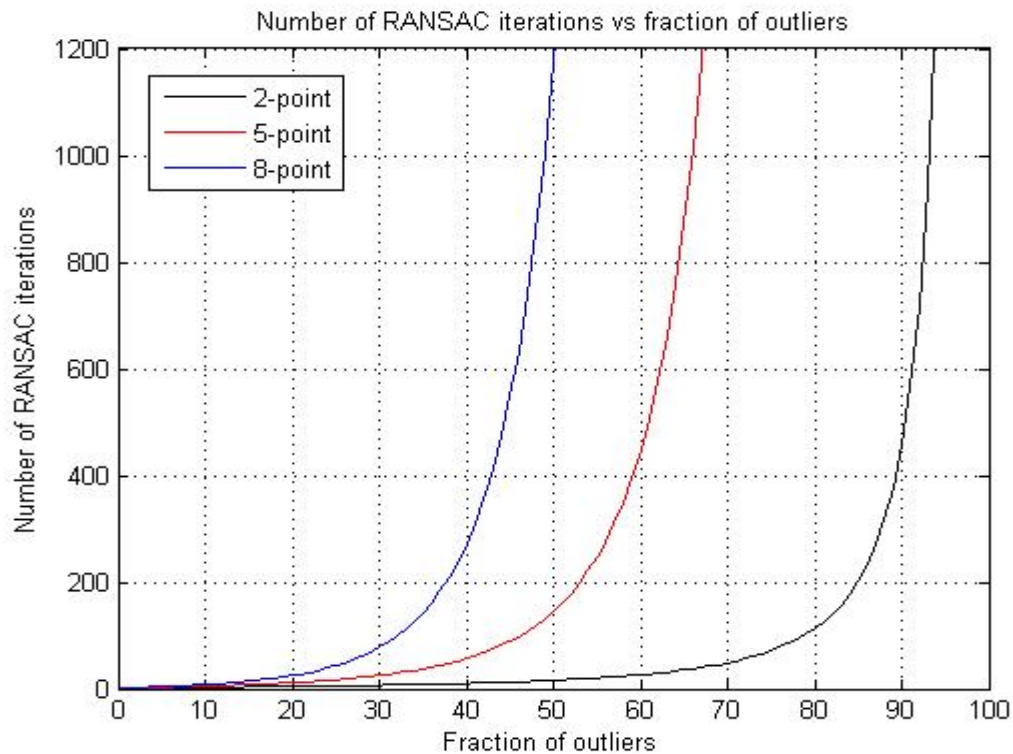
# RANSAC iterations

- As observed, $k$ is exponential in the number of points $s$ 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**

<span style="color:red">Can we use less than 5 points?</span>

Yes, if you use motion constraints!

# Planar Motion

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

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\varphi \\ \rho\sin\varphi \\ 0 \end{bmatrix}$$
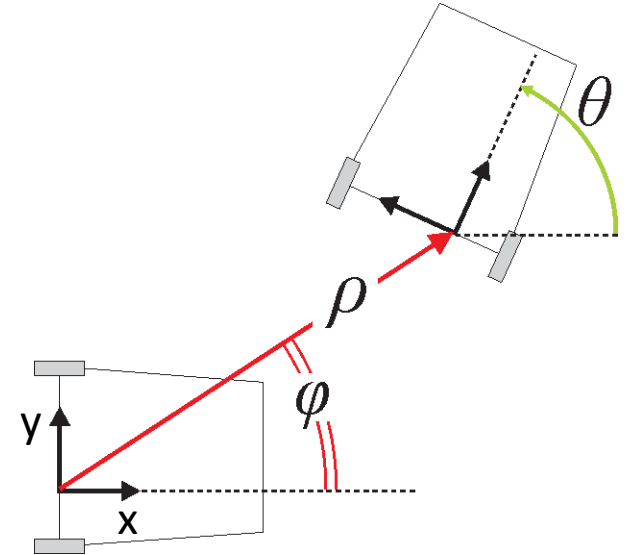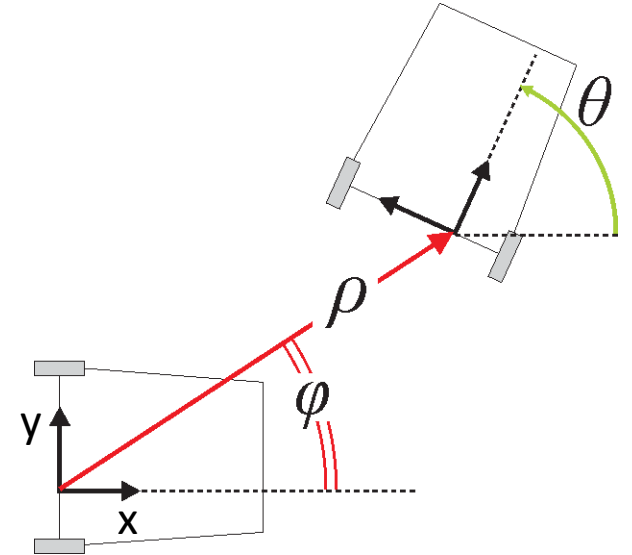
Let's compute the Epipolar Geometry

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

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

76

# Planar Motion

Planar motion is described by three parameters: $\vartheta, \varphi, \rho$

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



Let's compute the Epipolar Geometry
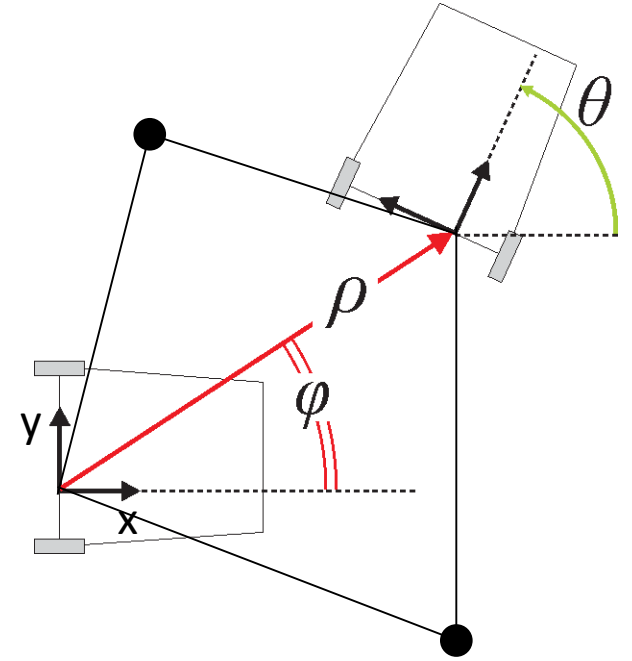
$$[T]_\times = \begin{bmatrix} 0 & 0 & \rho\sin\varphi \\ 0 & 0 & -\rho\cos\varphi \\ -\rho\sin\varphi & \rho\cos\varphi & 0 \end{bmatrix}$$

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

77

# Planar Motion

Planar motion is described by three parameters: $\vartheta, \varphi, \rho$

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



Let's compute the Epipolar Geometry

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

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

# Planar Motion

Planar motion is described by three parameters: $\vartheta, \varphi, \rho$

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



Observe that $E$ has 2DoF (θ, ɸ, because ρ is the scale factor); thus, 2 correspondences are sufficient to estimate θ and ɸ ["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)
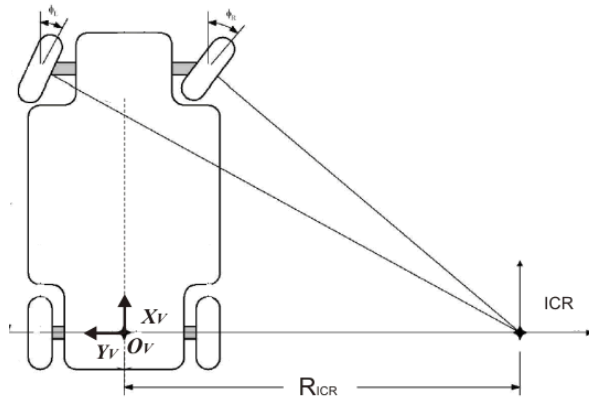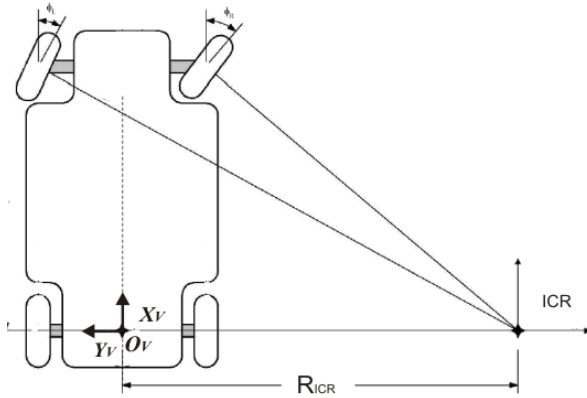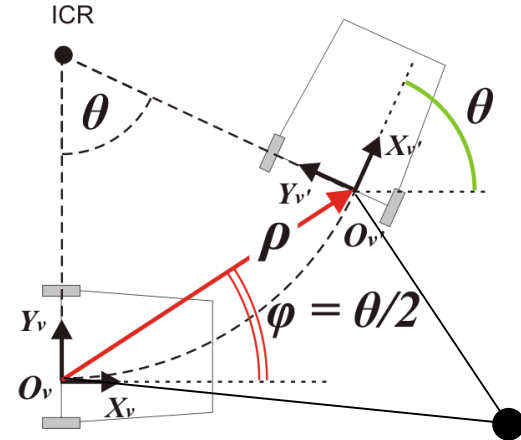


Example of Ackerman steering principle
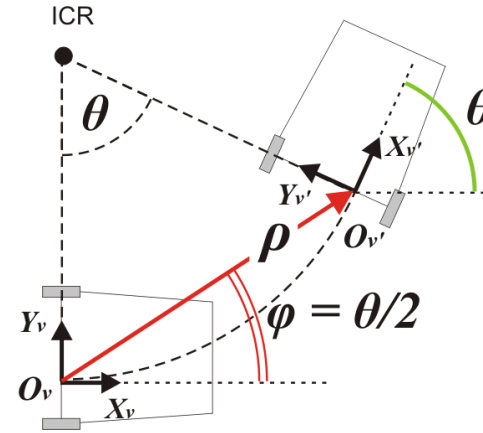

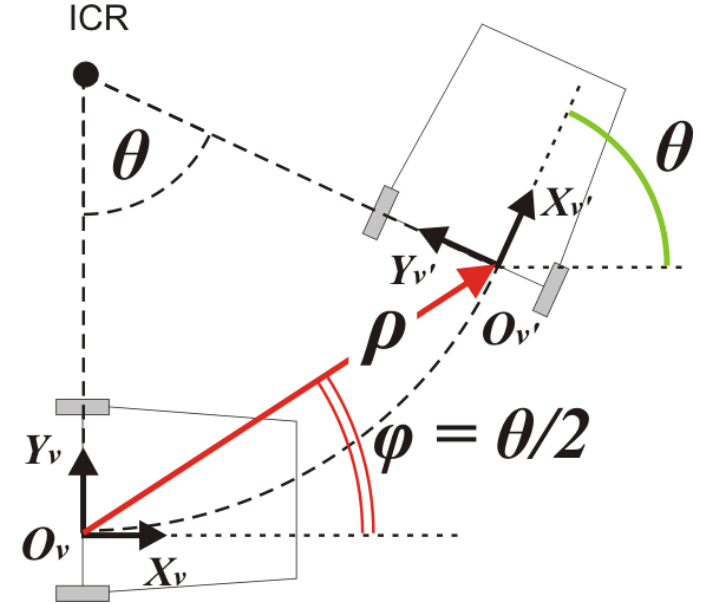
Locally-planar circular motion

# 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

$$\varphi = \theta/2 => only\ 1\ DoF\ (\theta);$$

*thus, only 1 point correspondence is needed* [Scaramuzza, IJCV'11]

**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)

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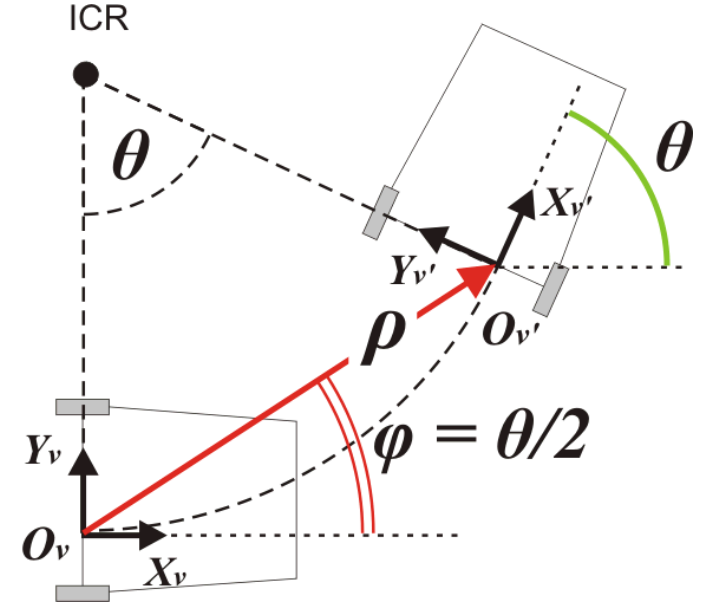


Let's compute the Epipolar Geometry

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

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

83

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



$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T = \begin{bmatrix} \rho\cos\dfrac{\theta}{2} \\ \rho\sin\dfrac{\theta}{2} \\ 0 \end{bmatrix}$$
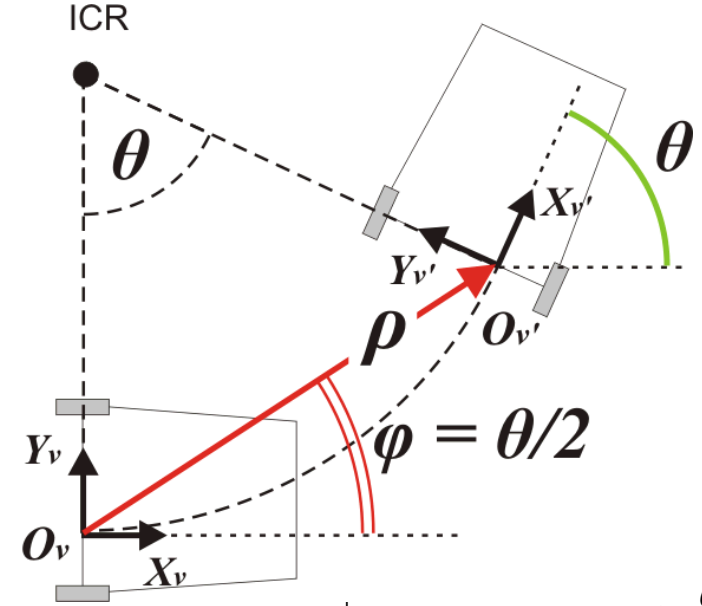
Let's compute the Epipolar Geometry

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

84

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

ICR

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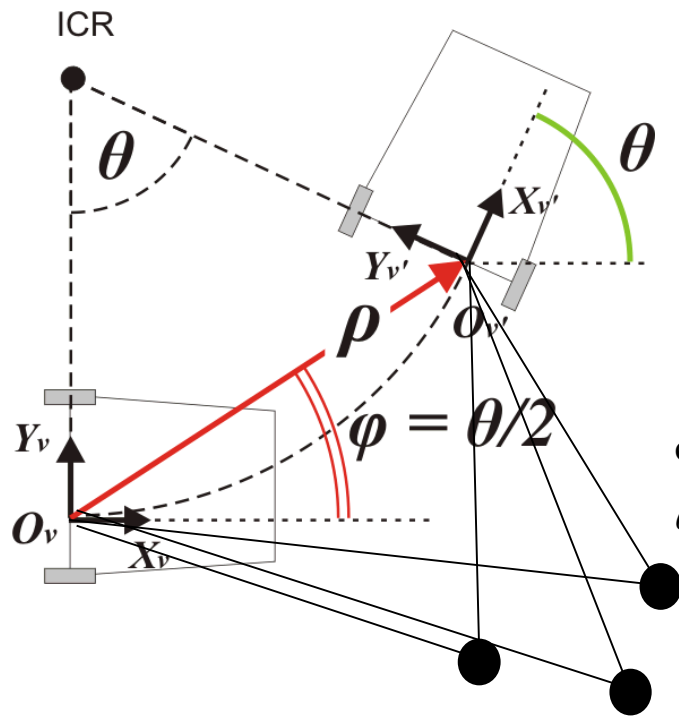$$

$\varphi = \theta/2$

Let's compute the Epipolar Geometry

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

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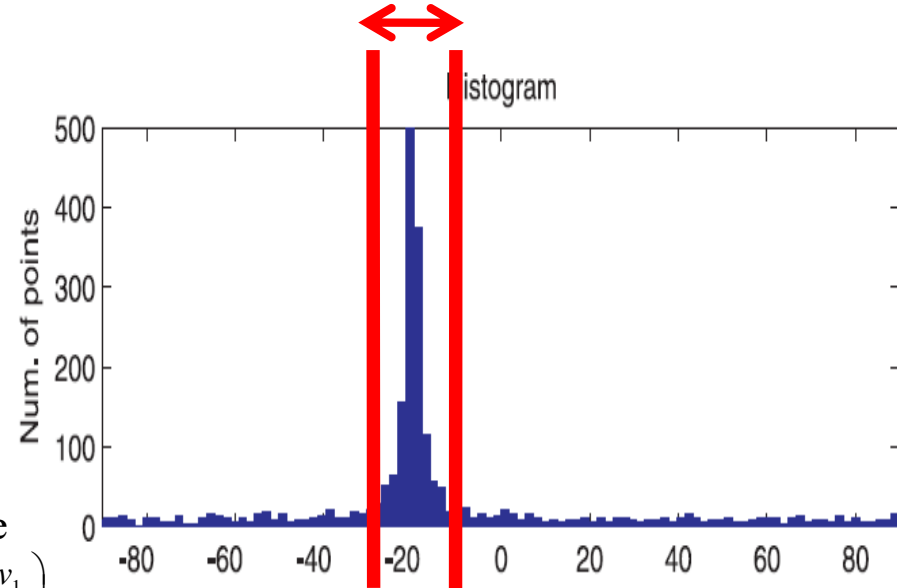$$

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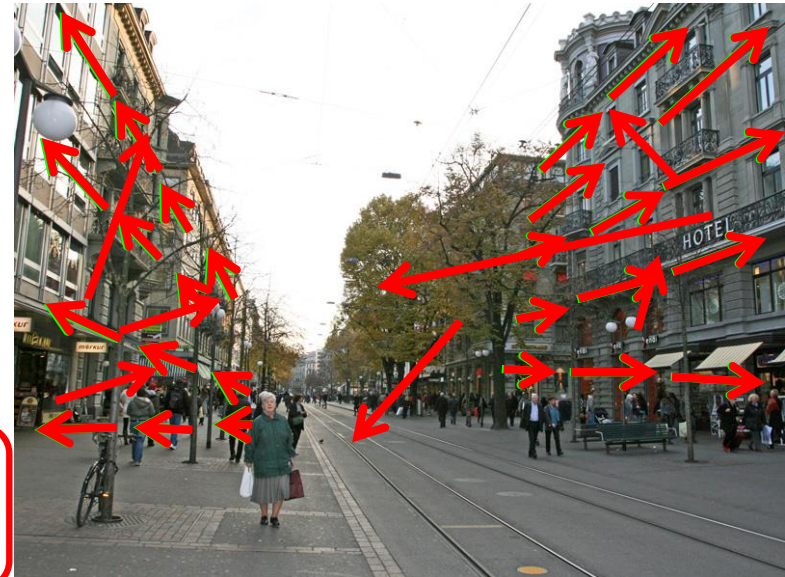$$

# 1-Point RANSAC algorithm
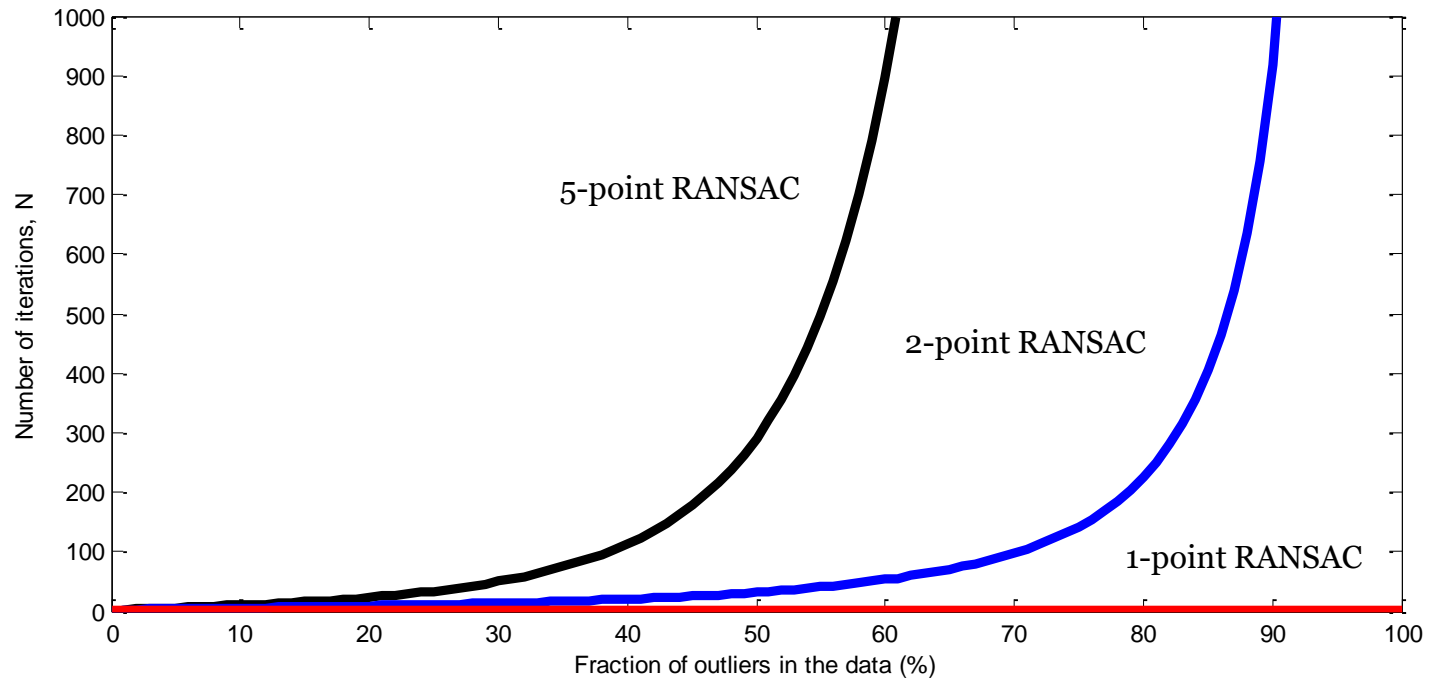


Compute $\theta$ for every point correspondence

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

Only 1 iteration!

The most efficient algorithm for removing outliers (<1ms)

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

Motion is then estimated from them in 6DOF
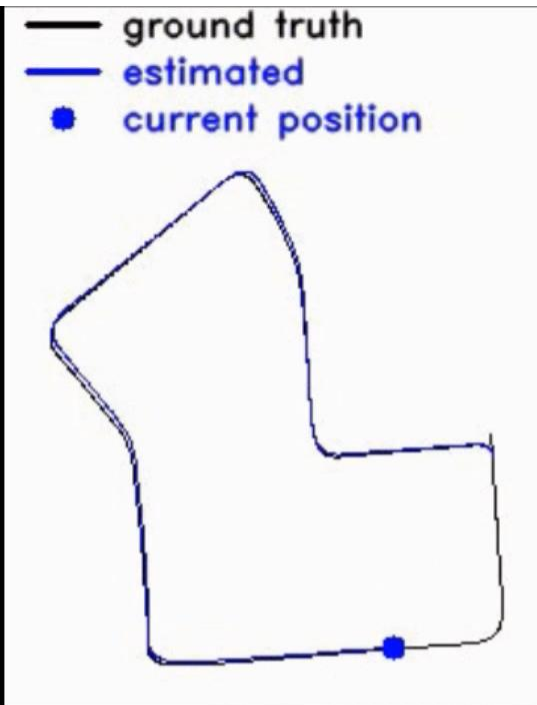
86

# Comparison of RANSAC algorithms



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

| | 8-Point RANSAC [Longuet-Higgins'81] | 5-Point RANSAC [Nister'04] | 2-Point RANSAC [Ortin'01] | 1-Point RANSAC [Scaramuzza'11] |
|---|---|---|---|---|
| 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

# Understanding Check

Are you able to answer the following questions?

- What's the minimum number of correspondences required for calibrated SFM and why?

- Are you able to derive the epipolar constraint?

- Are you able to define the essential matrix?

- Are you able to derive the 8-point algorithm?

- How many rotation-translation combinations can the essential matrix be decomposed into?

- Are you able to provide a geometrical interpretation of the epipolar constraint?

- Are you able to describe the relation between the essential and the fundamental matrix?

- Why is it important to normalize the point coordinates in the 8-point algorithm?

- Describe one or more possible ways to achieve this normalization.

- Are you able to describe the normalized 8-point algorithm?

- Are you able to provide quality metrics for the essential matrix estimation?

- Why do we need RANSAC?

- What is the theoretical maximum number of combinations to explore?

- After how many iterations can RANSAC be stopped to guarantee a given success probability?

- What is the trend of RANSAC vs. iterations, vs. the fraction of outliers, vs. the number of points to estimate the model?

- How do we apply RANSAC to the 8-point algorithm, DLT, P3P?

- How can we reduce the number of RANSAC iterations for the SFM problem?