# Tutorial on Visual Odometry

**Davide Scaramuzza**

University of Zurich

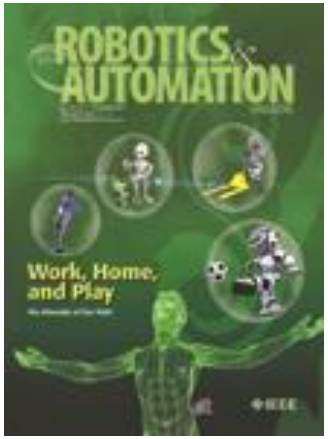
Robotics and Perception Group

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

University of Zurich[UZH]

ROBOTICS & PERCEPTION GROUP

# References






- Scaramuzza, D., Fraundorfer, F., **Visual Odometry: Part I** - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

- Fraundorfer, F., Scaramuzza, D., **Visual Odometry: Part II** - Matching, Robustness, and Applications, IEEE Robotics and Automation Magazine, Volume 19, issue 1, 2012.

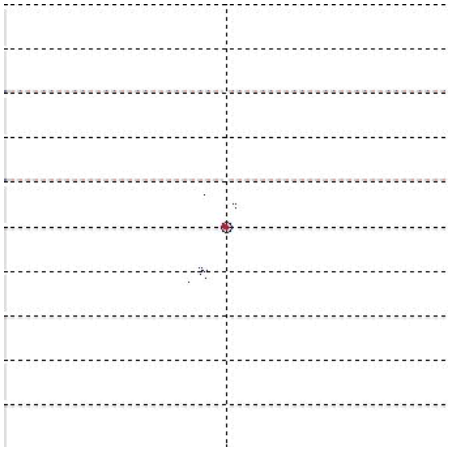
# What is Visual Odometry (VO) ?

VO is the process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras

**input**

Image sequence (or video stream)
from one or more cameras attached to a moving vehicle

**output**



$$R_0, R_1, \ldots, R_i$$

$$t_0, t_1, \ldots, t_i$$

Camera trajectory (3D structure is a plus):

# Assumptions

- Sufficient illumination in the environment
- Dominance of static scene over moving objects
- Enough texture to allow apparent motion to be extracted
- Sufficient scene overlap between consecutive frames

**Is any of these scenes good for VO? Why?**

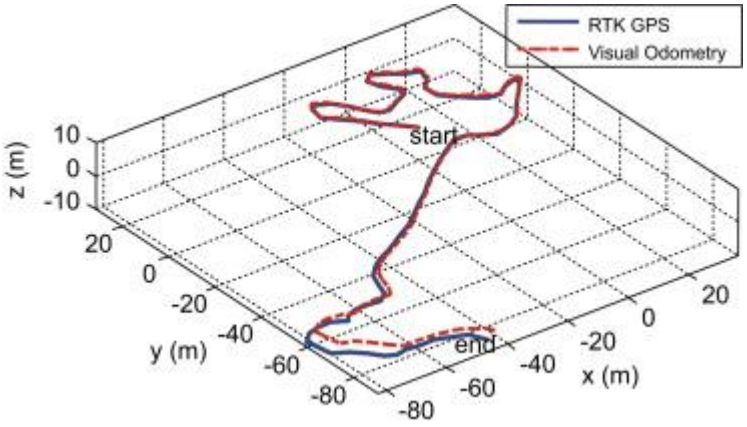
# Why VO ?

- Contrary to wheel odometry, VO is not affected by wheel slip in uneven terrain or other adverse conditions.

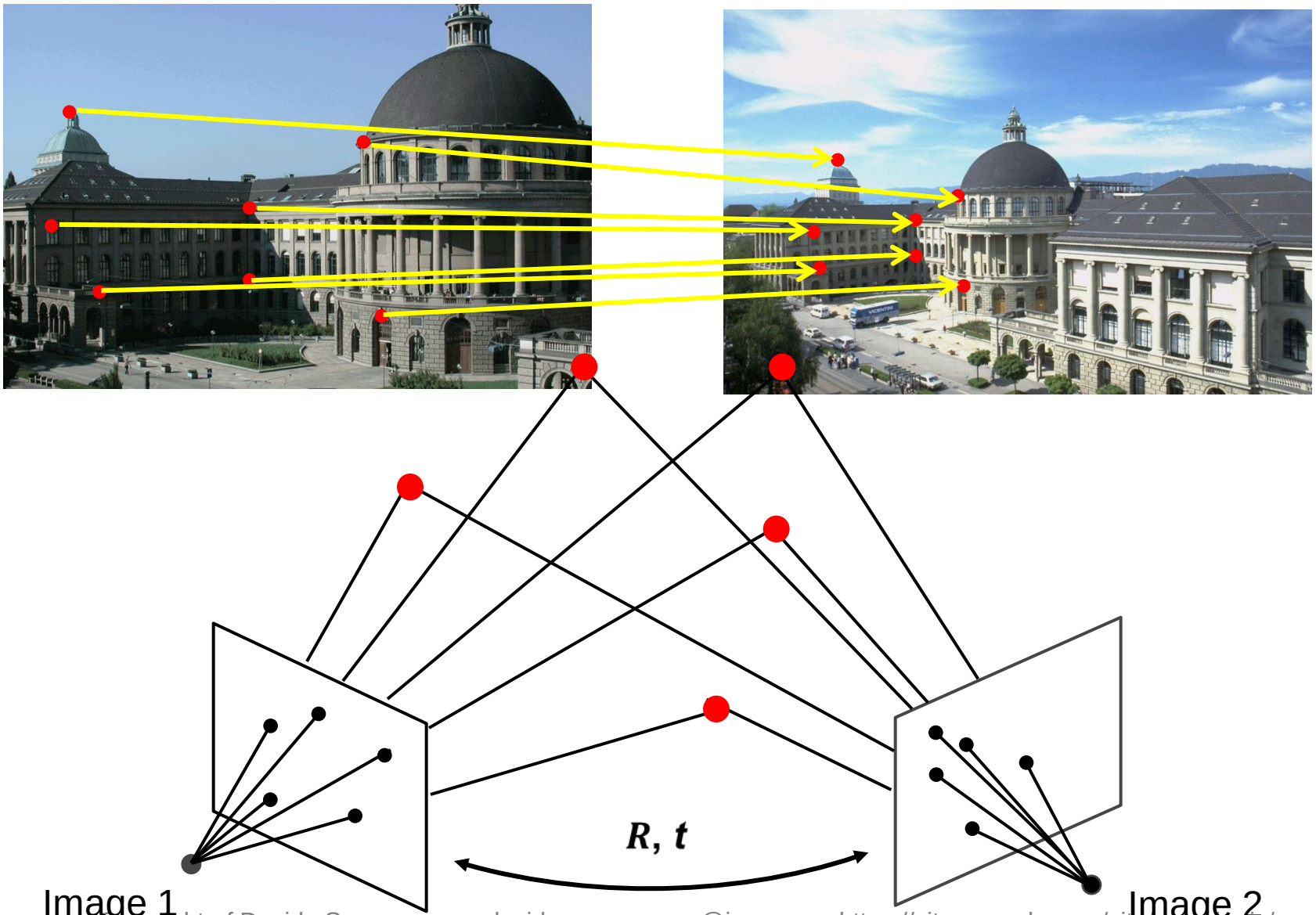
- More accurate trajectory estimates compared to wheel odometry
  (relative position error 0.1% – 2%)

- VO can be used as a complement to
  - wheel odometry
  - GPS
  - inertial measurement units (IMUs)
  - laser odometry

- In GPS-denied environments, such as underwater and aerial, VO has utmost importance

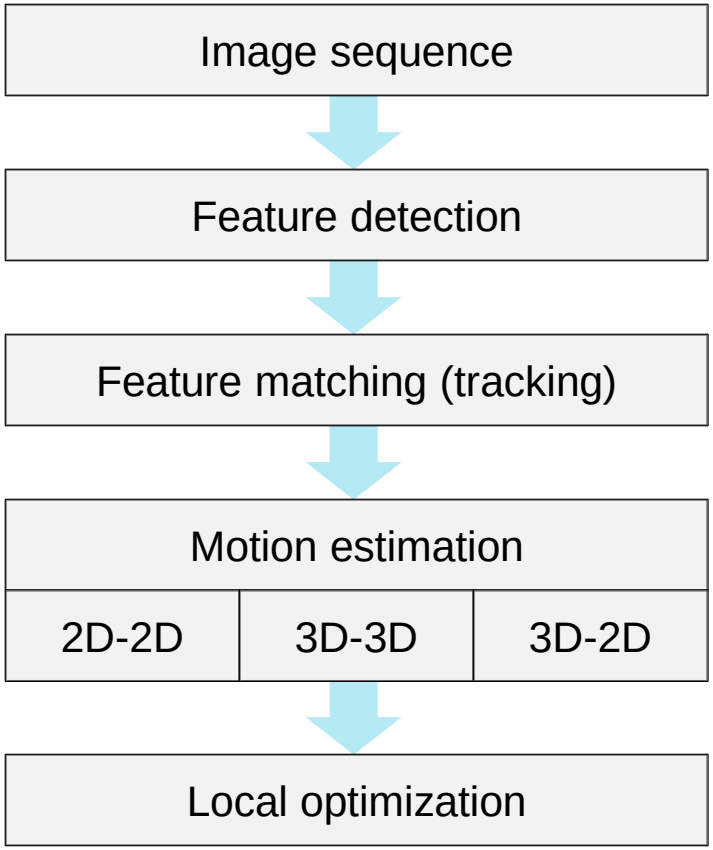


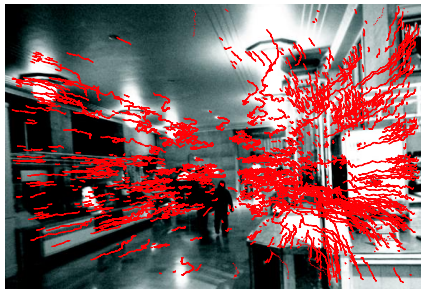
# Working Principle



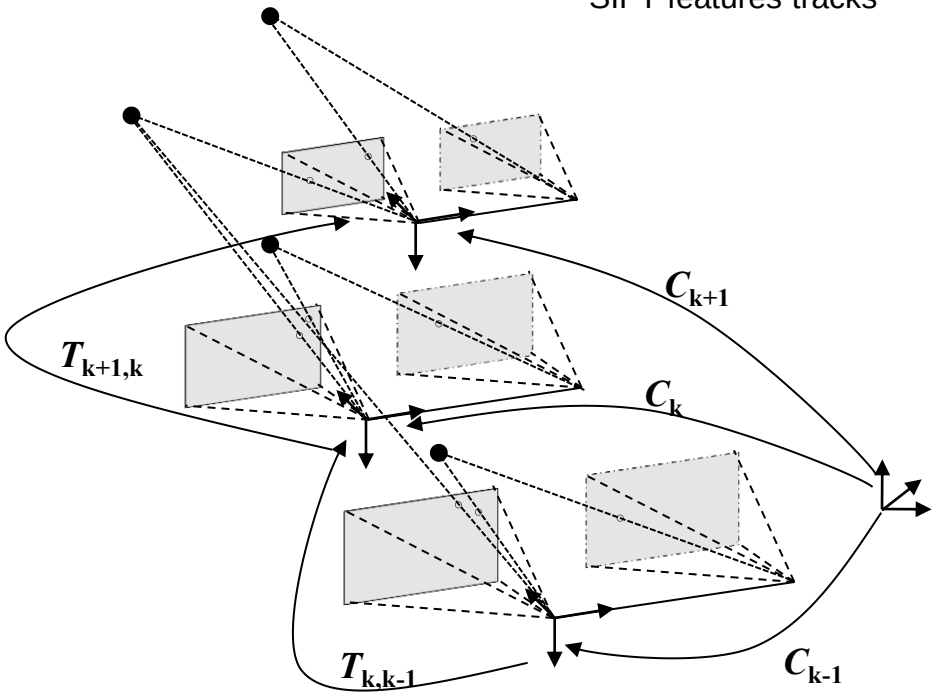
# VO Flow Chart

- VO computes the camera path incrementally (pose after pose)



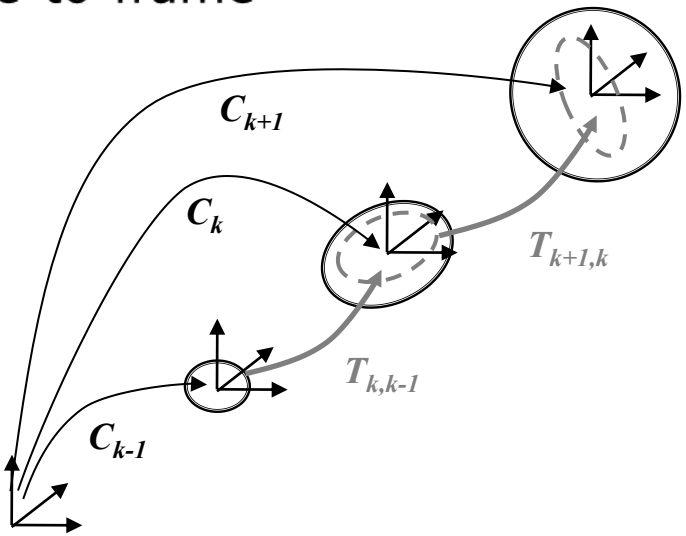



SIFT features tracks



# VO Drift

- The errors introduced by each new frame-to-frame motion accumulate over time
- This generates a drift of the estimated trajectory from the real path




The uncertainty of the camera pose at $C_k$ is a combination of the uncertainty at $C_{k-1}$ (black solid ellipse) and the uncertainty of the transformation $T_{k,k-1}$ (gray dashed ellipse)

# VO or Structure from Motion (SFM) ? (1/2)

- SFM is more general than VO and tackles the problem of 3D reconstruction of both the structure and camera poses from unordered image sets

- The final structure and camera poses are typically refined with an offline optimization (i.e., bundle adjustment), whose computation time grows with the number of images

**This video can be seen at**
**http://youtu.be/kxtQqYLRaSQ**

Reconstruction from 3 million images from Flickr.com
Cluster of 250 computers, 24 hours of computation!
Paper: “Building Rome in a Day”, ICCV’09

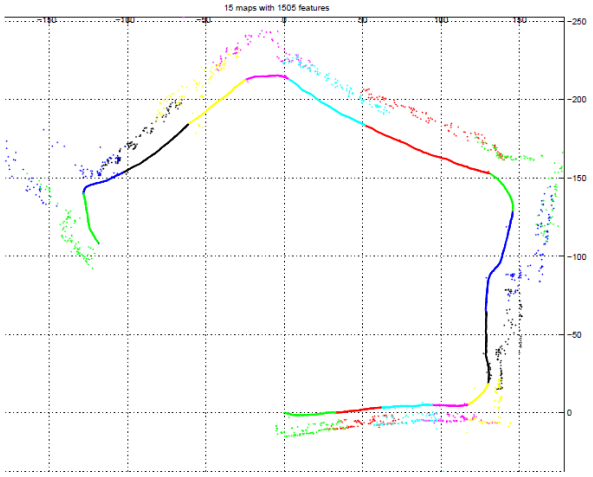
# VO or Structure from Motion (SFM) ? (2/2)

- VO is a particular case of SFM
- VO focuses on estimating the 3D motion of the camera sequentially (as a new frame arrives) and in real time.
- Bundle adjustment can be used (but it's optional) to refine the local estimate of the trajectory
- Terminology: sometimes SFM is used as a synonym of VO

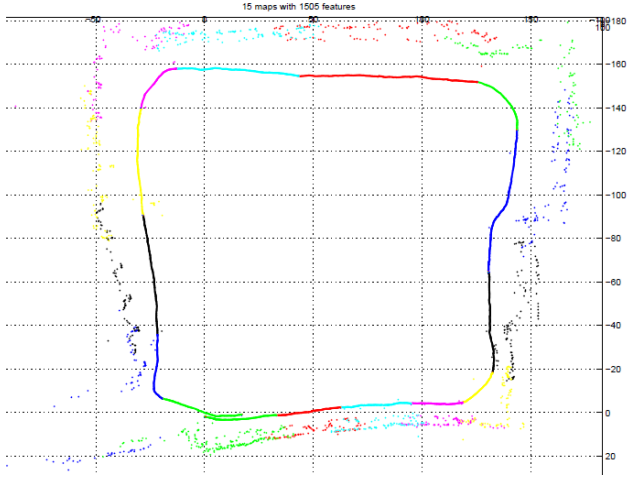
# VO vs. Visual SLAM (1/2)

- The goal of SLAM in general is to obtain a global, consistent estimate of the robot path. This is done through identifying loop closures. When a loop closure is detected, this information is used to reduce the drift in both the map and camera path (global bundle adjustment).

- Conversely, VO aims at recovering the path incrementally, pose after pose, and potentially optimizing only over the last $m$ poses path (windowed bundle adjustment)




**Before loop closing**

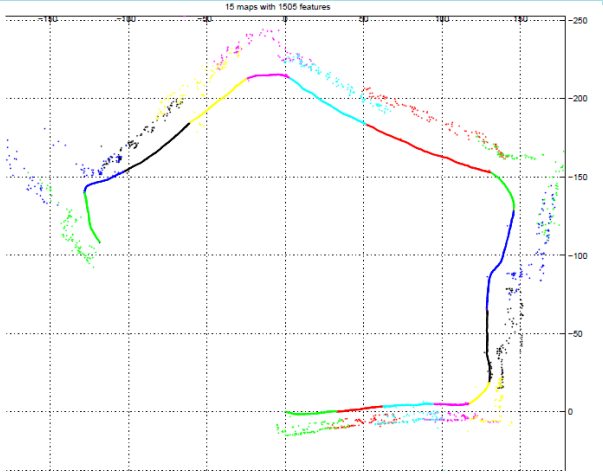



**After loop closing**

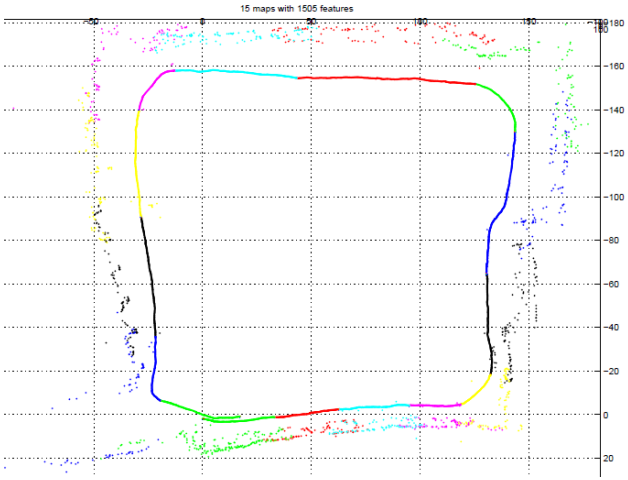
Image courtesy of Clemente et al. RSS'07

# VO vs. Visual SLAM (2/2)

- VO only aims to the local consistency of the trajectory
- SLAM aims to the global consistency of the trajectory and of the map
- VO can be used as a building block of SLAM
- VO is SLAM before closing the loop!
- The choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation.
- VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera.




**Visual odometry**




**Visual SLAM**

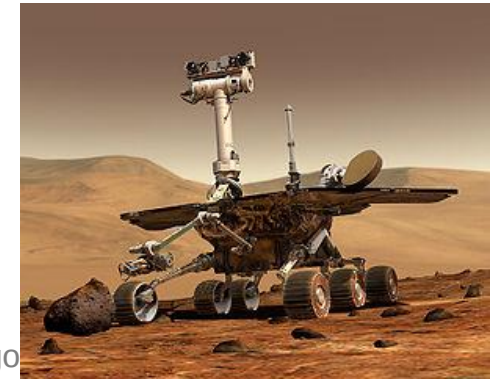
Image courtesy of Clemente et al. RSS'07

# Outline

- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion

# Brief history of VO

- **1996**: The term VO was coined by Srinivasan to define motion orientation in honey bees.
- **1980**: First known stereo VO real-time implementation on a robot by Moraveck PhD thesis (NASA/JPL) for Mars rovers using a sliding camera. Moravec invented a predecessor of Harris detector, known as Moravec detector
- **1980 to 2000**: The VO research was dominated by NASA/JPL in preparation of 2004 Mars mission (see papers from Matthies, Olson, etc. From JPL)
- **2004**: VO used on a robot on another planet: Mars rovers Spirit and Opportunity
- **2004**. VO was revived in the academic environment by Nister «Visual Odometry» paper.
  The term VO became popular.



# Outline

- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion

# Problem Formulation

- An agent is moving through an environment and taking images with a rigidly-attached camera system at discrete times $k$

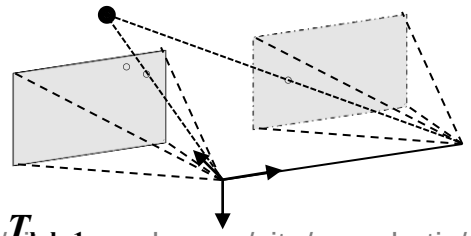
- In case of a monocular system, the set of images taken at times $k$ is denoted by

$$I_{0:n} = \{I_0, \dots, I_n\}$$

- In case of a stereo system, there are a left and a right image at every time instant, denoted by

$$I_{l,0:n} = \{I_{l,0}, \dots, I_{l,n}\},$$
$$I_{r,0:n} = \{I_{r,0}, \dots, I_{r,n}\}$$

- In case of a stereo system, without loss of generality, the coordinate system of the left camera can be used as the origin



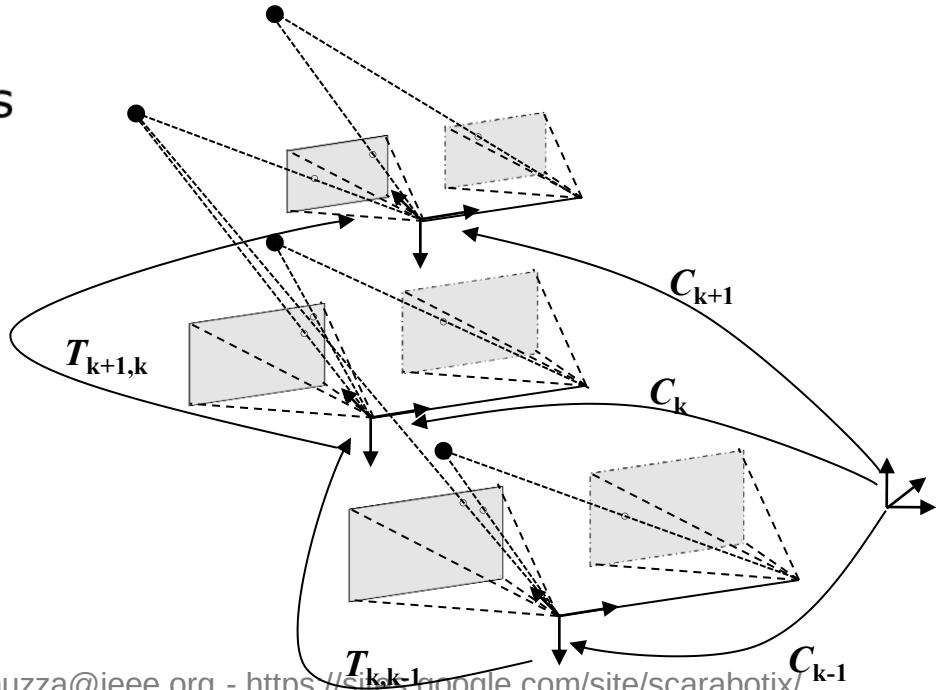
# Problem Formulation

- Two camera positions at adjacent time instants $k-1$ and $k$ are related by the rigid body transformation

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

- The set $T_{0:n} = \{T_1, \ldots, T_n\}$ contains all subsequent motions all subsequent motions.

- Finally, the set of camera poses $C_{0:n} = \{C_0, \ldots, C_n\}$ contains the transformations of the camera with respect to the initial coordinate frame at $k = 0$

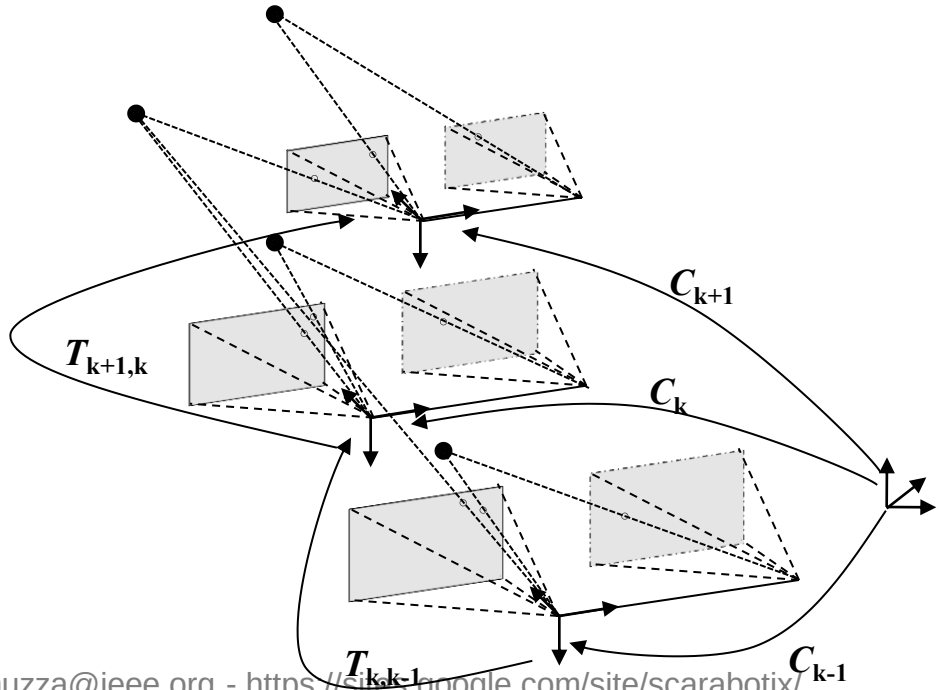


# Problem Formulation

- The current pose $C_n$ can be computed by concatenating all the transformations $T_k$, $k = 1 \dots n$, and, therefore,

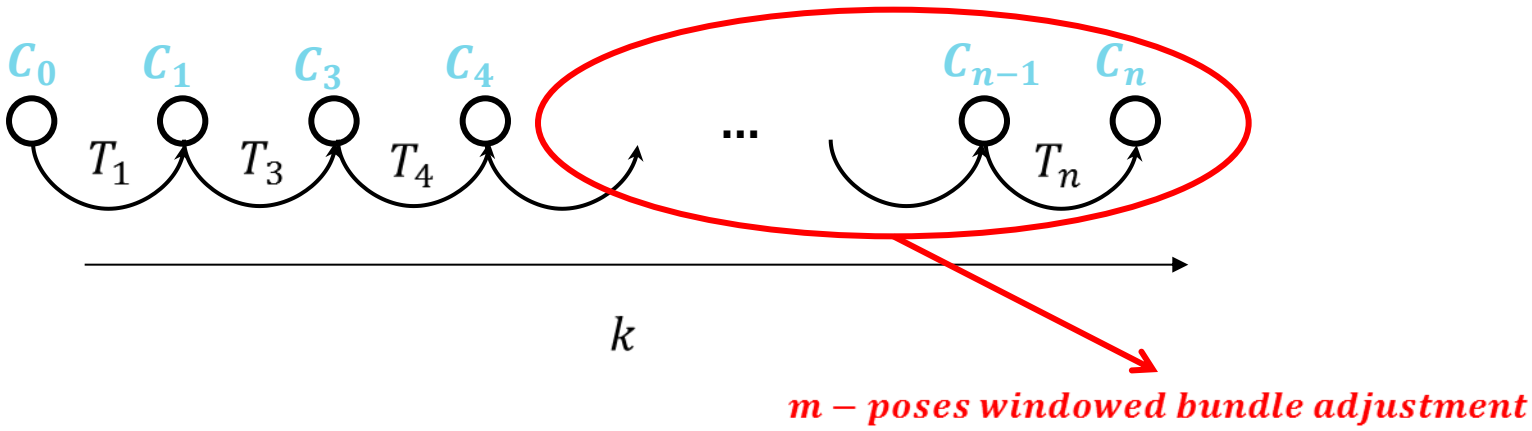
$$C_n = C_{n-1} T_n$$

with $C_0$ being the camera pose at the instant $k = 0$, which can be set arbitrarily by the user



# Problem Formulation

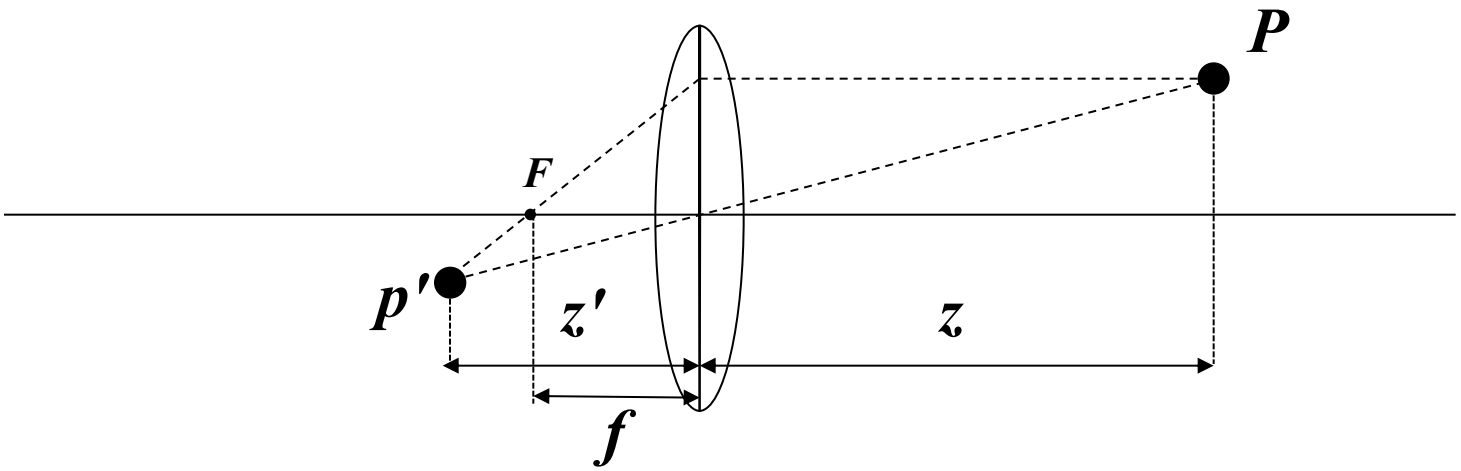
- The main task in VO is to compute the relative transformations $T_k$ from the images $I_k$ and $I_{k-1}$ and then to concatenate the transformations to recover the full trajectory $C_{0:n}$ of the camera.
- This means that VO recovers the path incrementally, pose after pose.
- An iterative refinement over last *m* poses can be performed after this step to obtain a more accurate estimate of the local trajectory.



# Outline

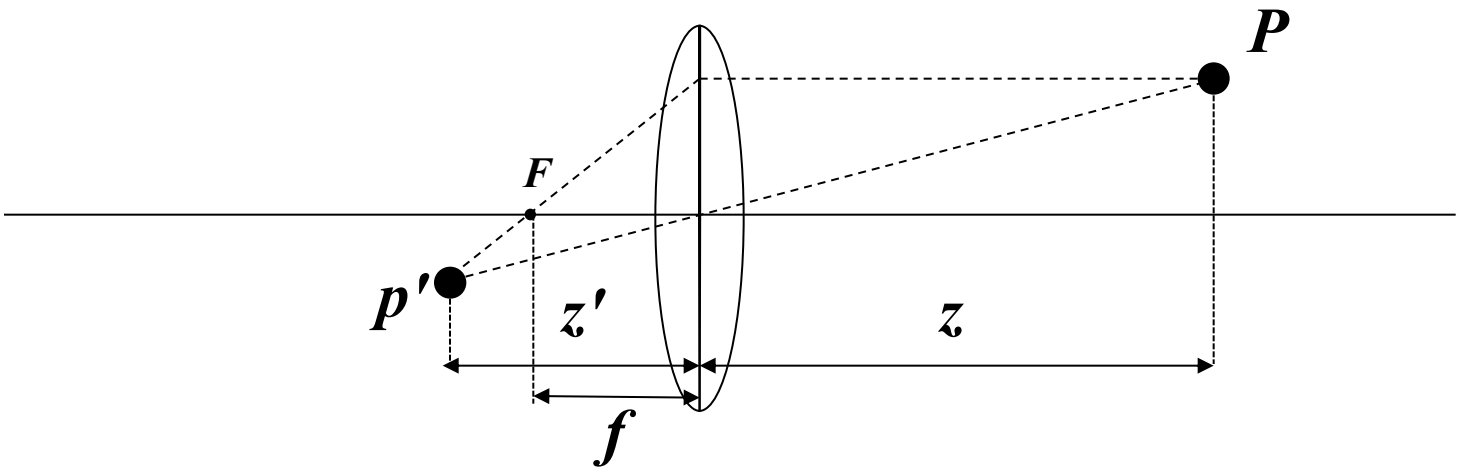
- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion

# Thin Lens Model



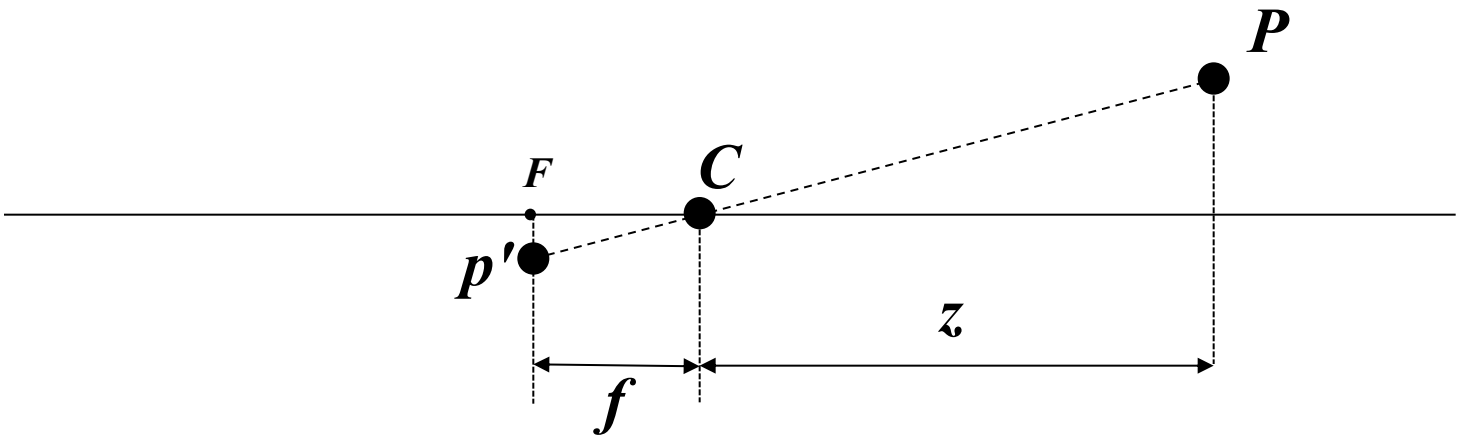

$$\frac{1}{z}+\frac{1}{z'}=\frac{1}{f}$$

# Thin Lens Model



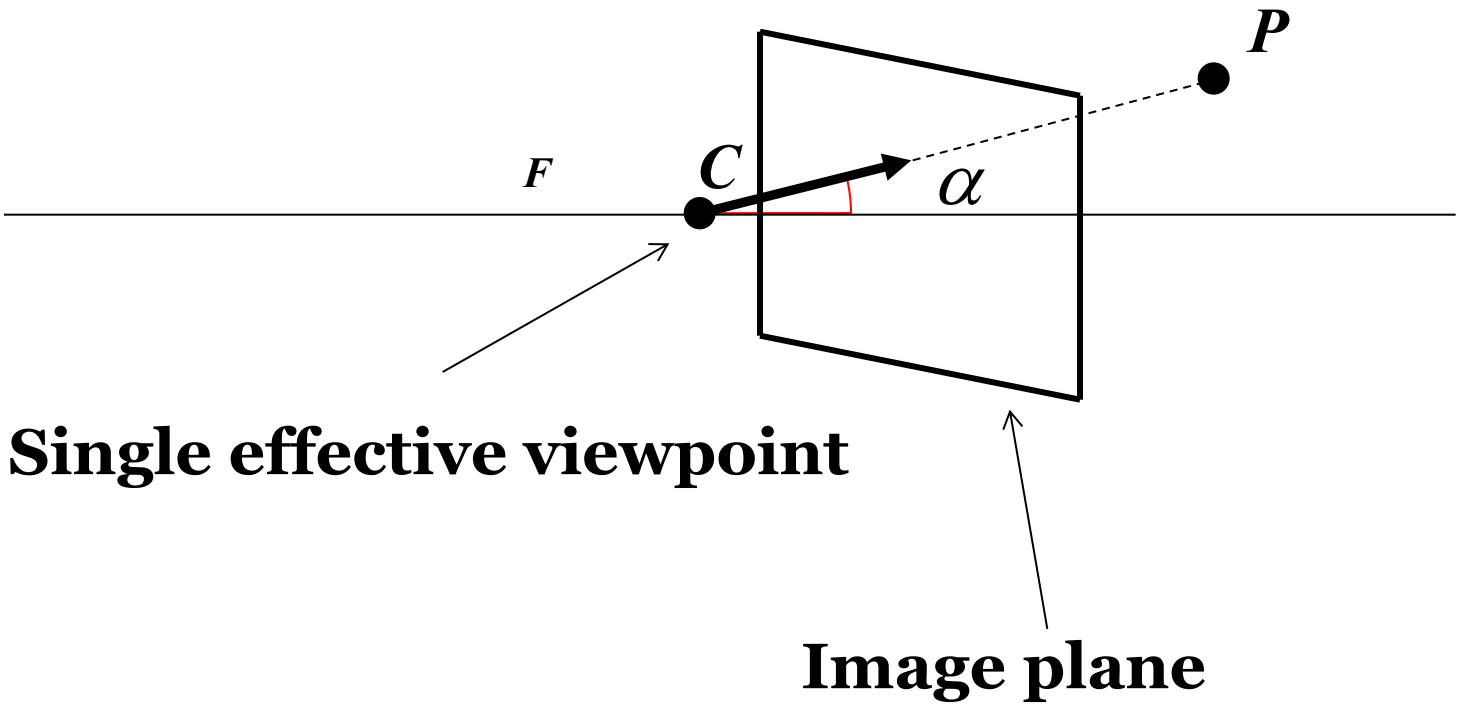

$$z >> z' \quad \Rightarrow \frac{1}{z'} \approx \frac{1}{f} \quad \Rightarrow z' = f$$

# The Pin-Hole Approximation



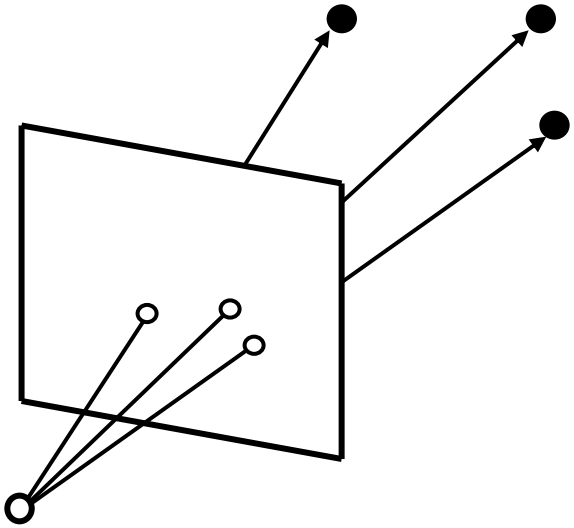

$$z >> z' \quad \Rightarrow \frac{1}{z'} \approx \frac{1}{f} \quad \Rightarrow z' = f$$

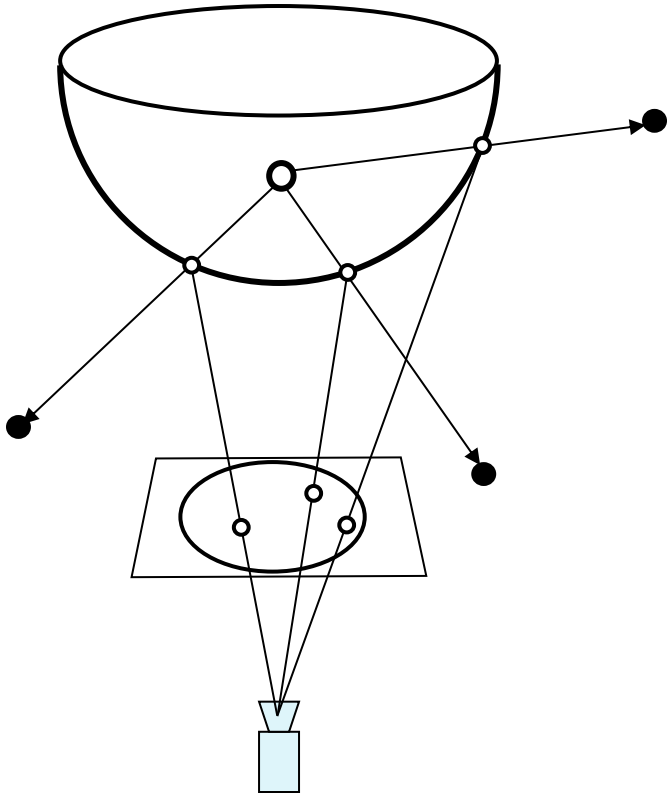
# The Pin-Hole Approximation



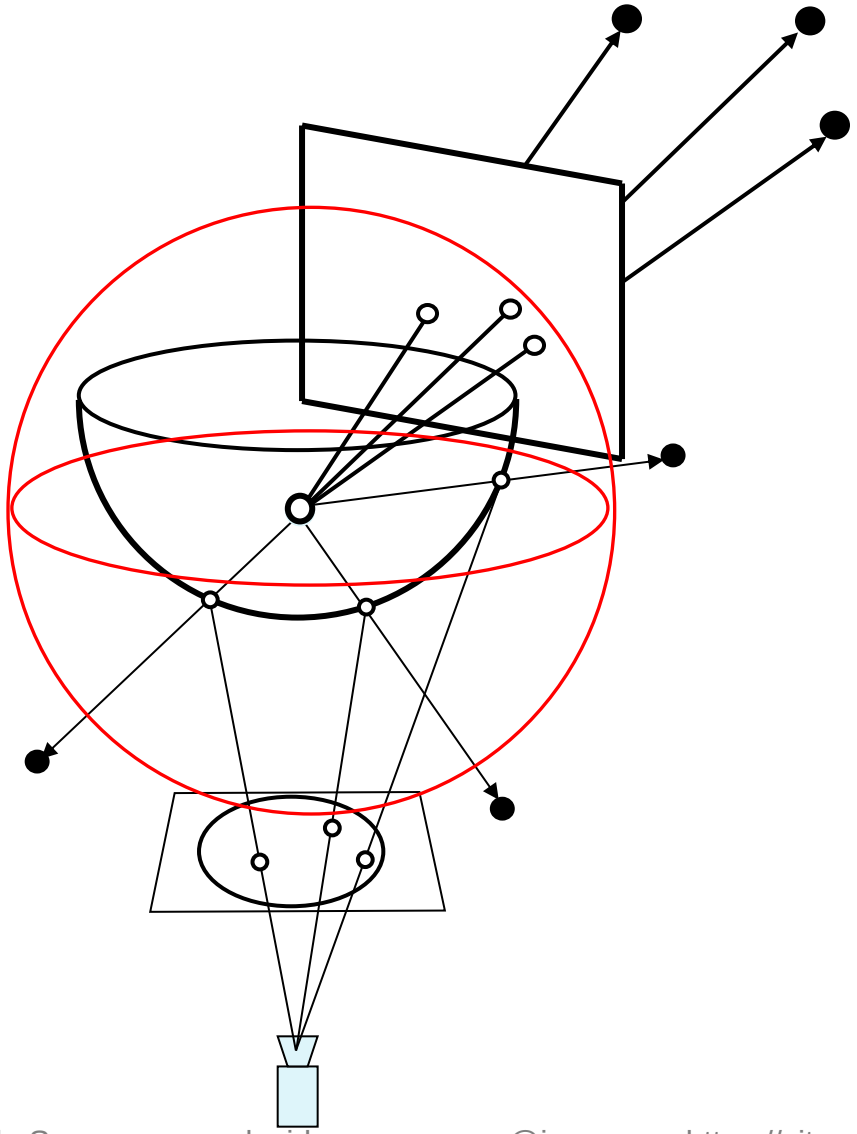
# Omnidirectional Camera Model

# Equivalence between Perspective and Omnidirectional Model

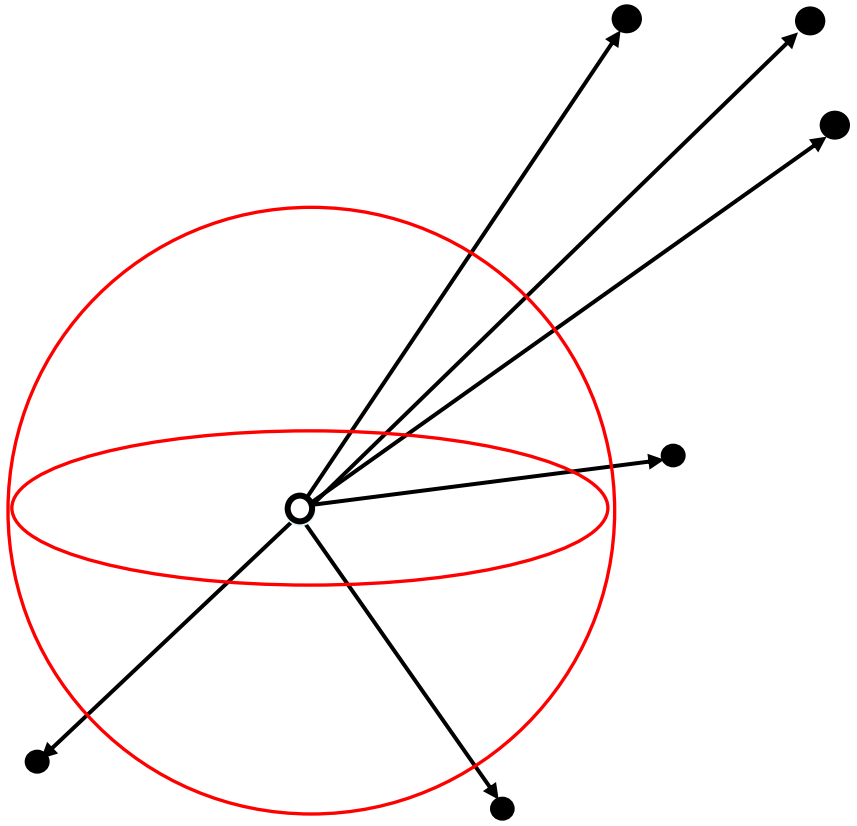




# Equivalence between Perspective and Omnidirectional Model



# Equivalence between Perspective and Omnidirectional Model: The Spherical Model



- Always possible after the camera has been calibrated!

# Equivalence between Perspective and Omnidirectional Model: The Spherical Model

- For convenience, points are projected on the unit sphere. Why?
- In the perspective case, is it better to use the perspective or the spherical model?

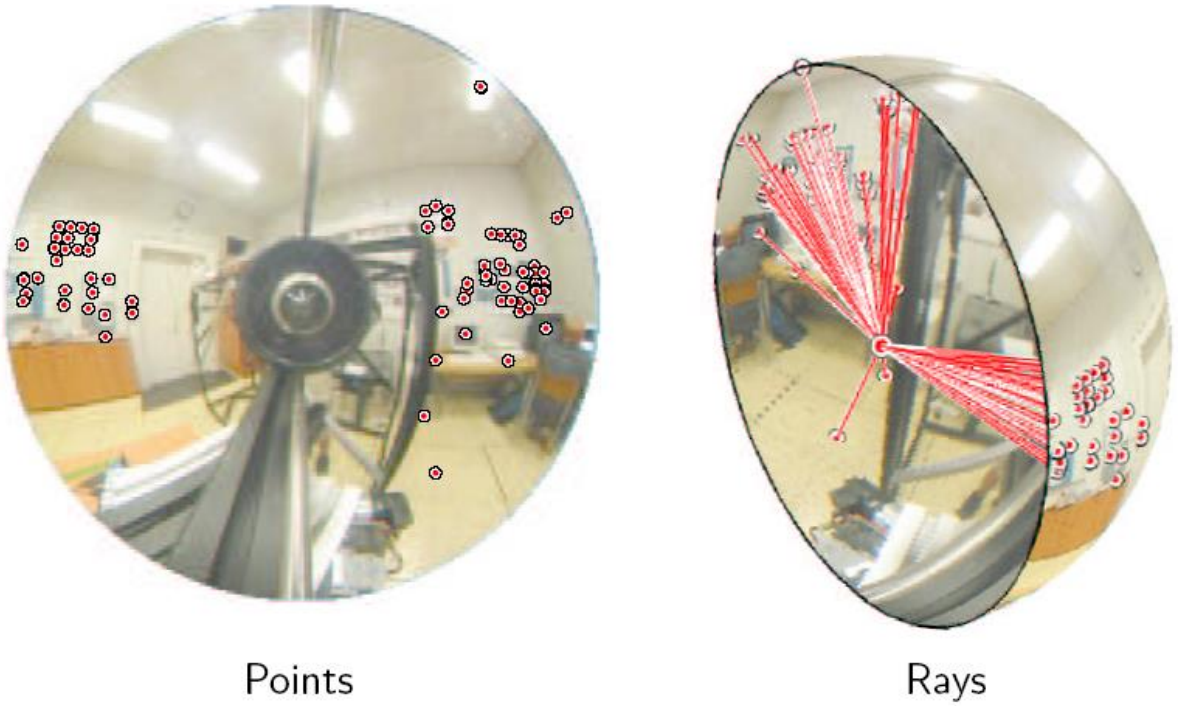
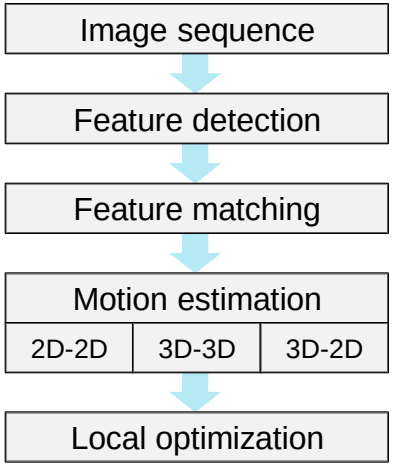



Image courtesy of Micusik & Pajdla, ACCV'04

# Outline

- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion



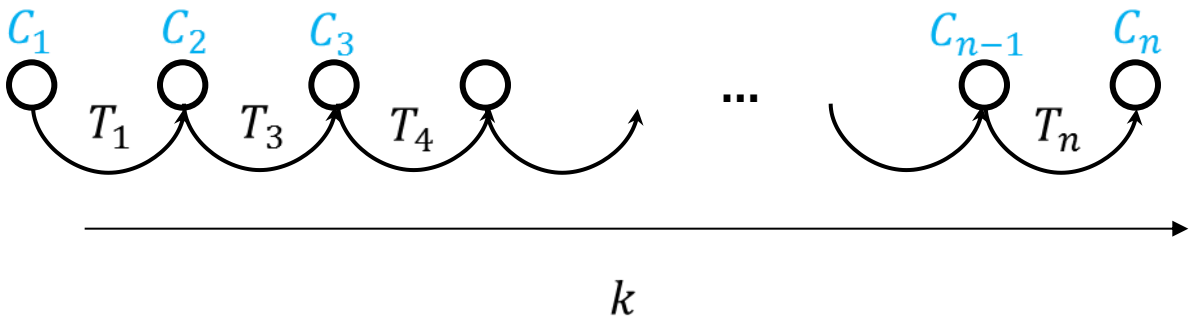
# Motion Estimation

- Motion estimation is the core computation step performed for every image in a VO system

- It computes the camera motion $T_k$ between the previous and the current image:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

- By concatenation of all these single movements, the full trajectory of the camera can be recovered

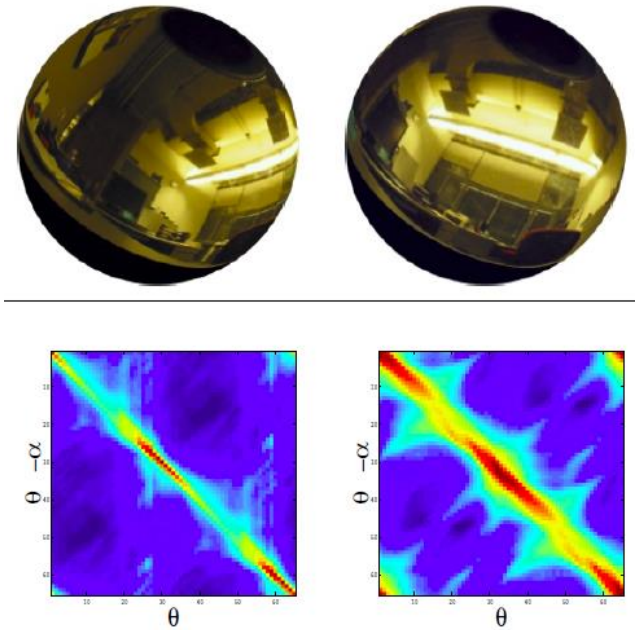


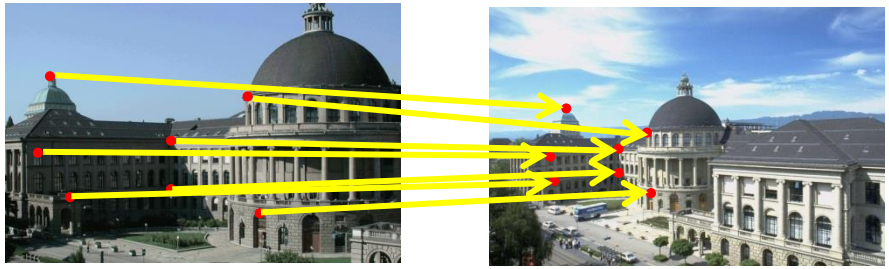
# Appearance-based or Featured-based ?

- There are two main approaches to compute the relative motion $T_k$

- Appearance-based methods use the intensity information of all the pixels in the two input images

- Feature-based methods only use salient and repeatable features extracted (or tracked) across the images



Makadia et al. «Correspondence-free structure from motion», IJCV'07

# Appearance-based or Featured-based ?

- Global methods are less accurate than feature-based methods and are computationally more expensive.

- Feature-based methods require the ability to match (or track) robustly features across frames but are faster and more accurate than global methods. Therefore, most VO implementations are feature based.

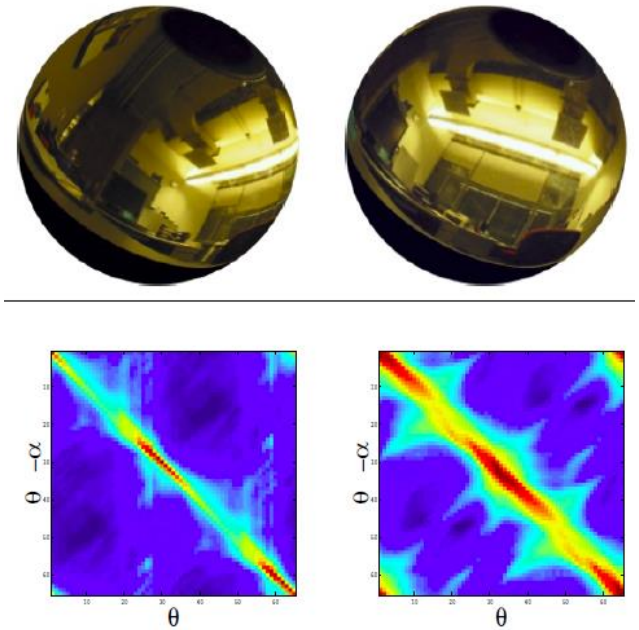
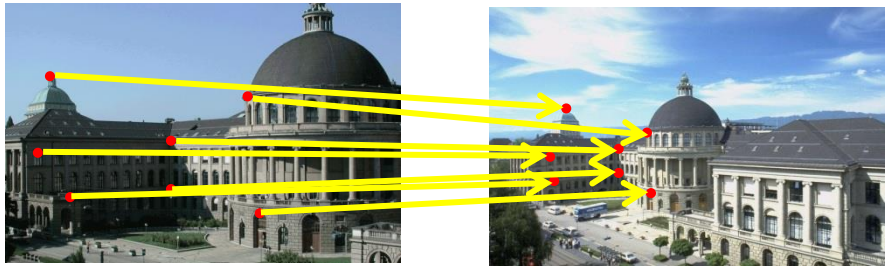



Image courtesy of Makadia et al., IJCV'07



# Motion Estimation

| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |

- Depending on whether the feature correspondences $f_{k-1}$ and $f_k$ are specified in 2D or 3D, there are three different cases:

- **2D-to-2D**: both $f_{k-1}$ and $f_k$ are specified in 2D image coordinates

- **3D-to-3D**: both $f_{k-1}$ and $f_k$ are specified in 3D To do this, it is necessary to triangulate 3D points at each time instant, for instance, by using a stereo camera system

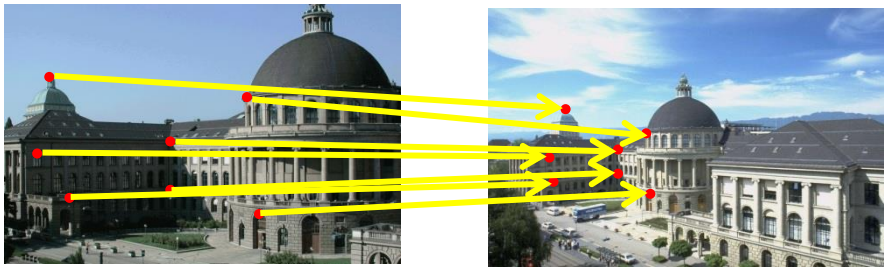
- **3D-to-2D**: $f_{k-1}$ are specified in 3D and $f_k$ are their corresponding 2D reprojections on the image $I_k$

# 2D-to-2D
## Motion from Image Feature Correspondences

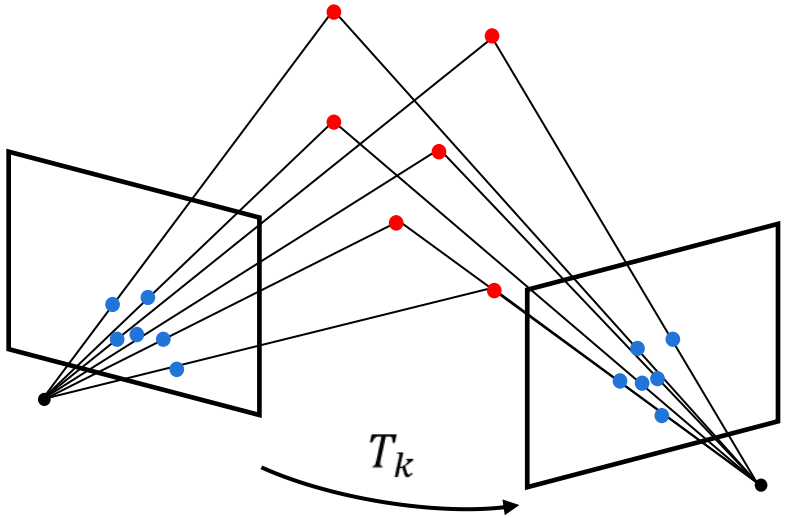
| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |

- Both $f_{k-1}$ and $f_k$ are specified in 2D
- The minimal-case solution involves 5-point correspondences
- The solution is found by determining the transformation that minimizes the reprojection error of the triangulated points in each image

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$



$I_{k-1}$ $I_k$

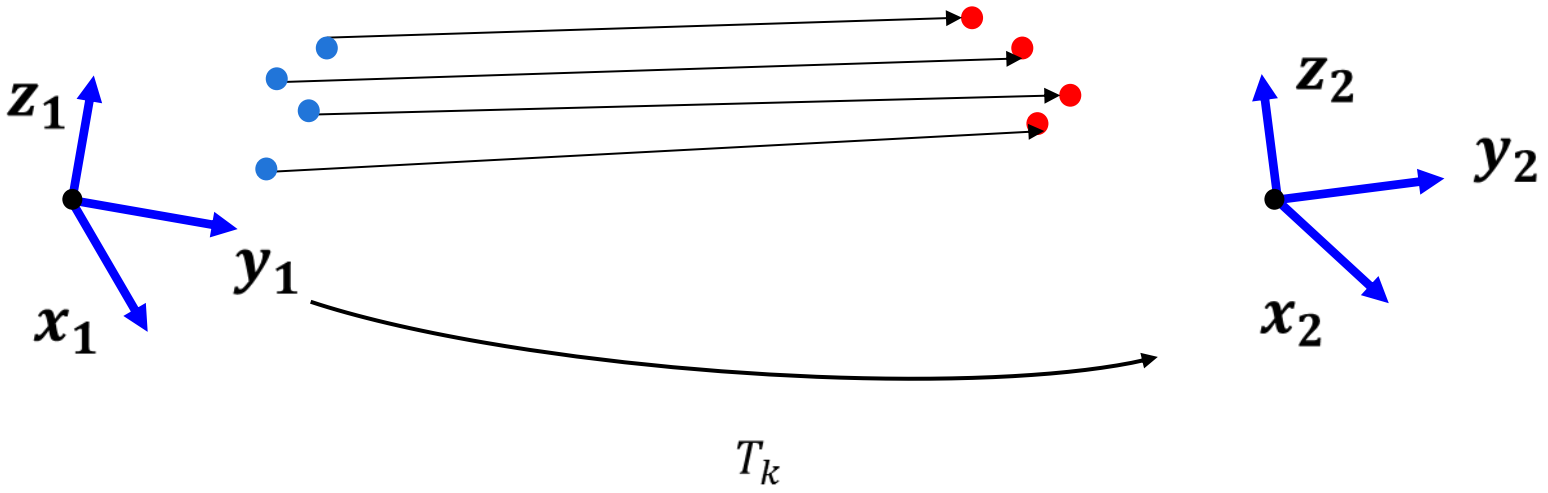


# 3D-to-3D
## Motion from 3D-3D Point Correspondences

| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |

- Both $f_{k-1}$ and $f_k$ are specified in 3D
- To do this, it is necessary to triangulate 3D points (e.g. use a stereo camera)
- The minimal-case solution involves 3 non-collinear correspondences
- The solution is found by determining the aligning transformation that minimizes the 3D-3D distance

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{T_k} \sum_i ||\tilde{X}_k^i - T_k \tilde{X}_{k-1}^i||$$

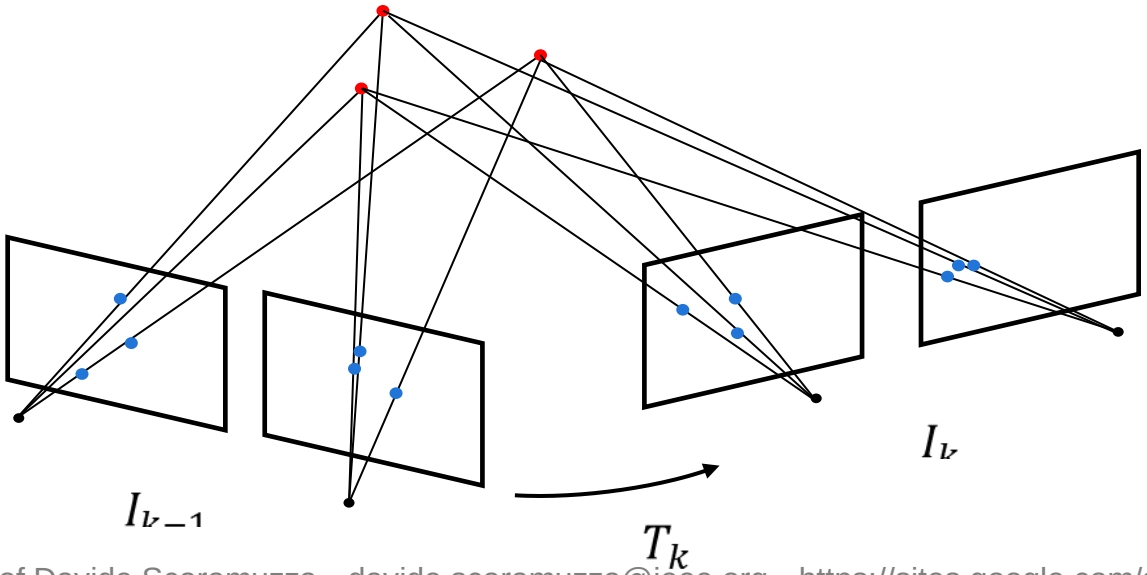


# 3D-to-3D
## Motion from 3D-3D Point Correspondences

| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |

- Both $f_{k-1}$ and $f_k$ are specified in 3D
- To do this, it is necessary to triangulate 3D points (e.g. use a stereo camera)
- The minimal-case solution involves 3 non-collinear correspondences
- The solution is found by determining the aligning transformation that minimizes the 3D-3D distance

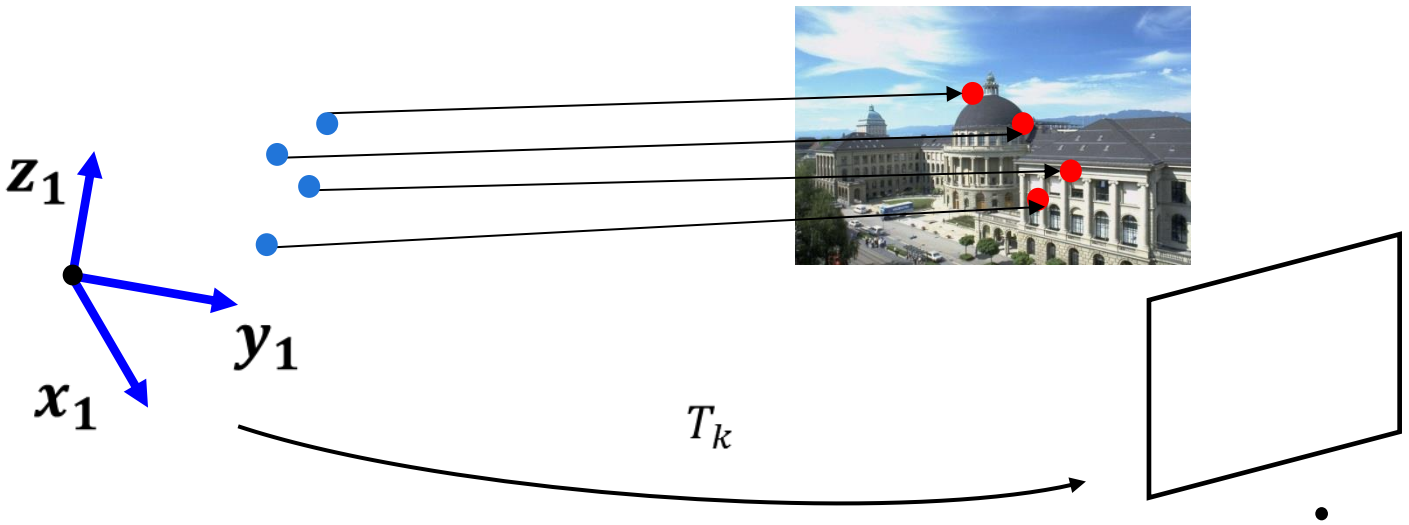
$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{T_k} \sum_i ||\tilde{X}_k^i - T_k \tilde{X}_{k-1}^i||$$



# 3D-to-2D
## Motion from 3D Structure and Image Correspondences

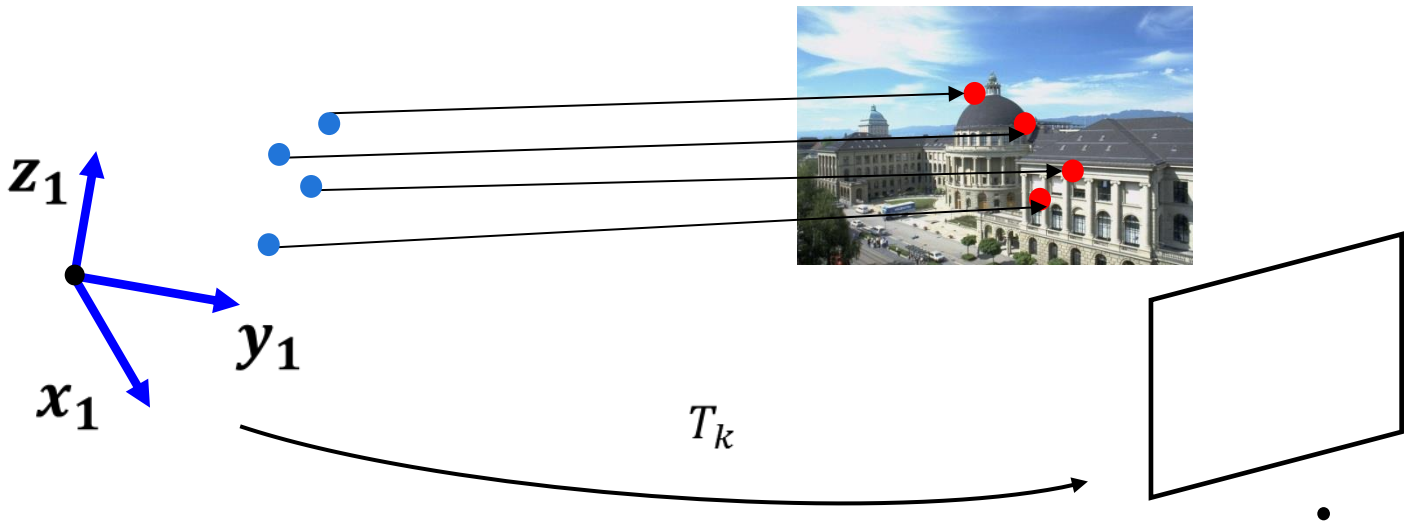
- $f_{k-1}$ is specified in 3D and $f_k$ in 2D
- This problem is known as *camera resection* or PnP (perspective from *n* points)
- The minimal-case solution involves 3 correspondences
- The solution is found by determining the transformation that minimizes the reprojection error

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{T_k} \sum_i \| p_k^i - \hat{p}_{k-1}^i \|^2$$



# 3D-to-2D
## Motion from 3D Structure and Image Correspondences

- In the monocular case, the 3D structure needs to be triangulated from two adjacent camera views (e.g., $I_{k-2}$ and $I_{k-1}$) and then matched to 2D image features in a third view (e.g., $I_k$).

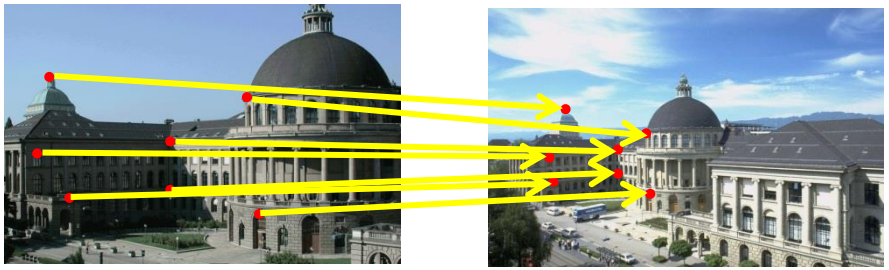


# 2D-to-2D
## Motion from Image Feature Correspondences

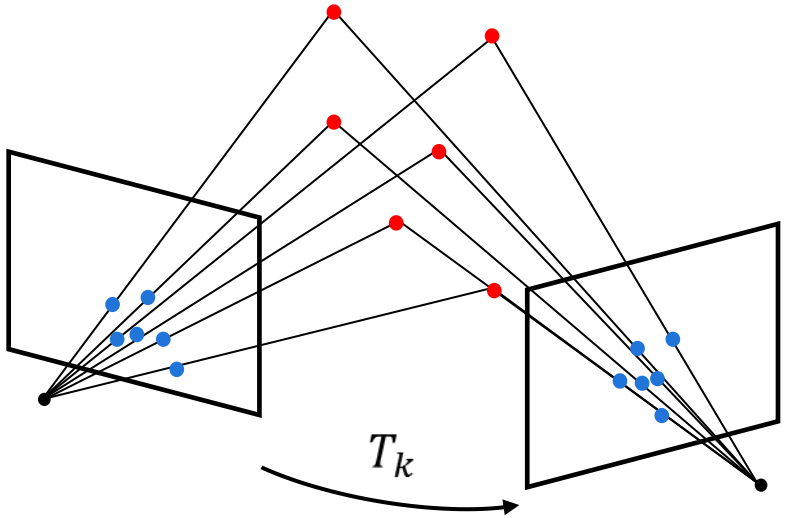
| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |

- Both $f_{k-1}$ and $f_k$ are specified in 2D
- The minimal-case solution involves 5-point correspondences
- The solution is found by determining the transformation that minimizes the reprojection error of the triangulated points in each image

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$

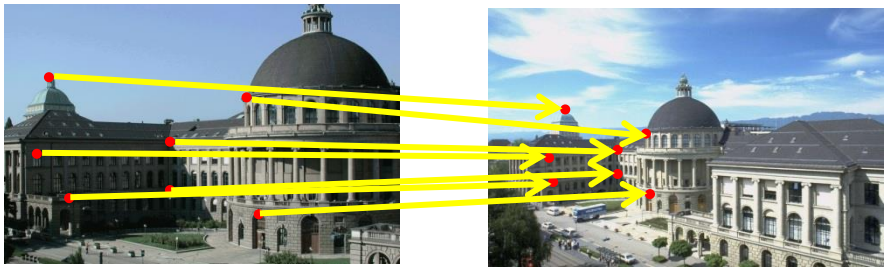


$I_{k-1}$ $I_k$

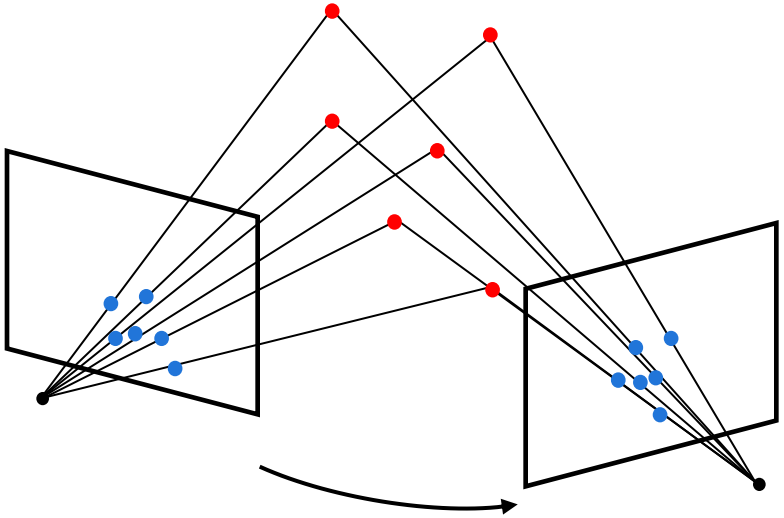


# 2D-to-2D
## Motion from Image Feature Correspondences

| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |



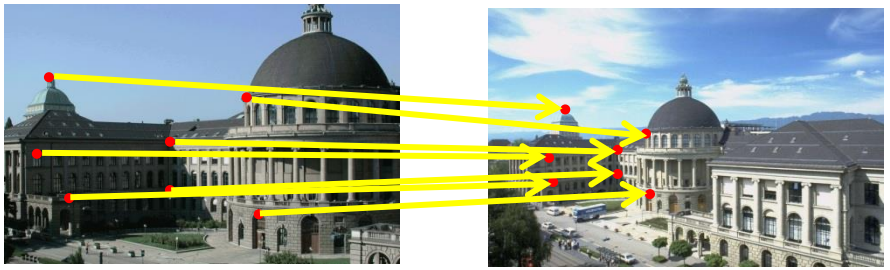
$I_{k-1}$ $I_k$



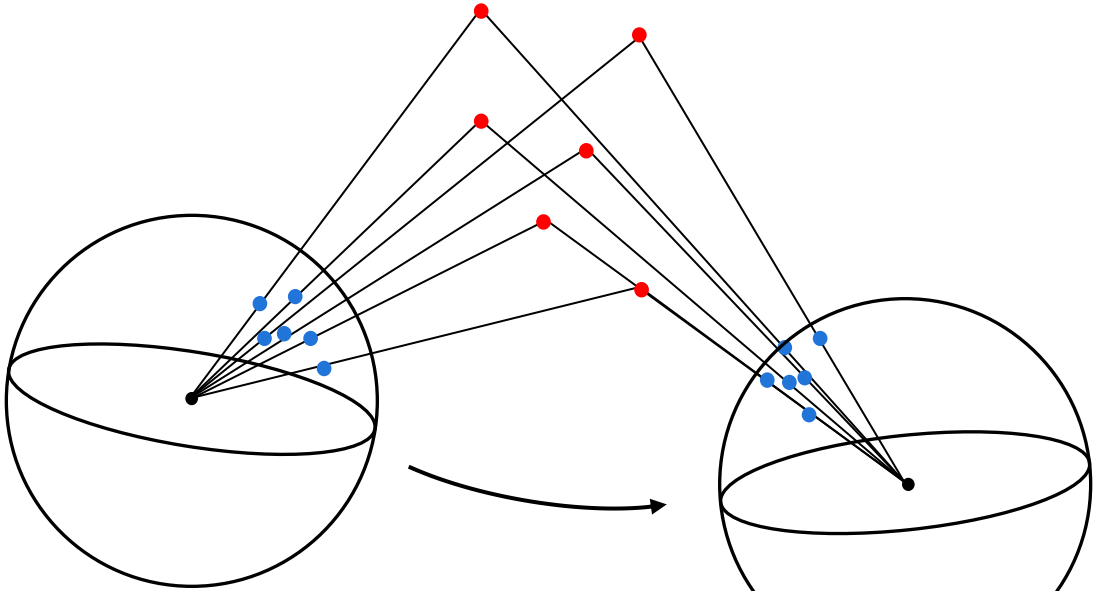
$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = ?$$

# 2D-to-2D
## Motion from Image Feature Correspondences

| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |



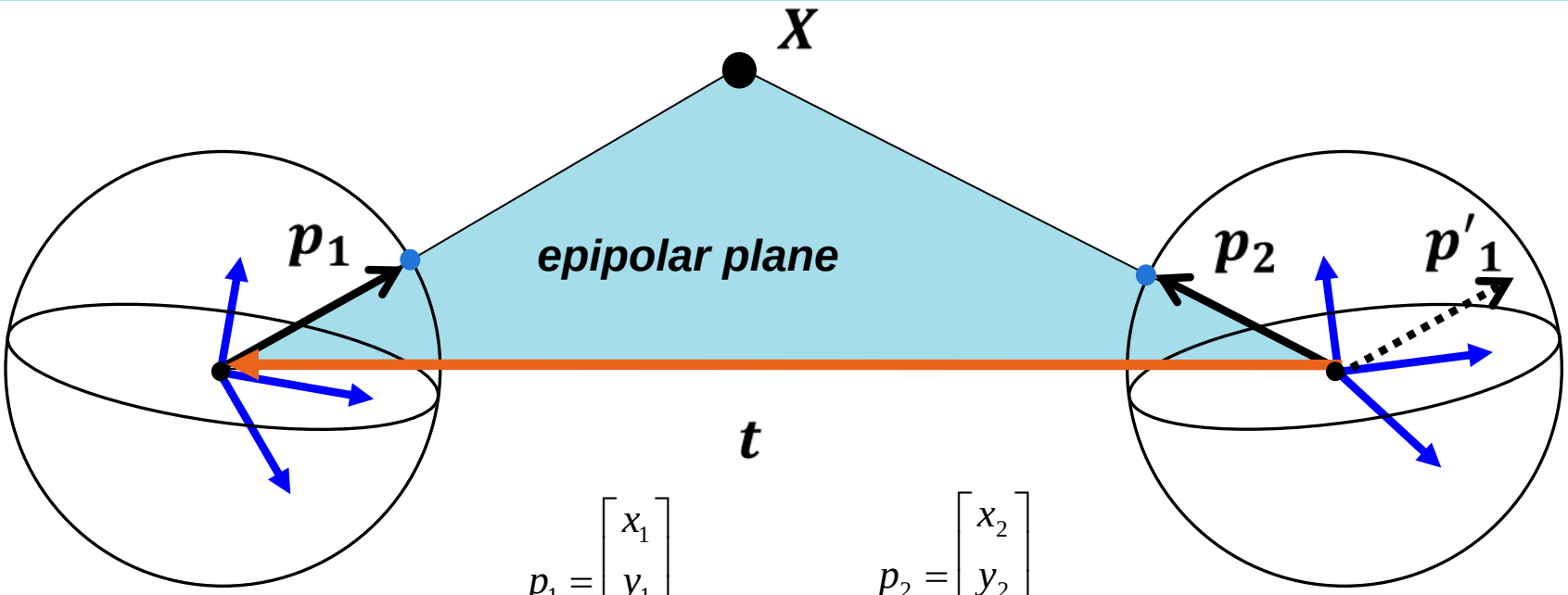
$I_{k-1}$ $I_k$



$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = ?$$

# 2D-to-2D
## Epipolar Geometry






$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \qquad p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

**$p_1$, $p_2$, $T$ are coplanar:**

$$p_2^T \cdot (t \times p_1') = 0 \qquad \Rightarrow p_2^T \cdot (t \times (R p_1)) = 0$$

$$\Rightarrow p_2^T [t]_\times R \; p_1 = 0 \qquad \Rightarrow p_2^T \; E \; p_1 = 0 \qquad \textit{Epipolar constraint}$$

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

# 2D-to-2D
## Epipolar Geometry



$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \qquad p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$ ***Image coordinates on the Unit sphere***

$$p_2^T \, E \, p_1 = 0$$ ***Epipolar constraint***

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

$$[\boldsymbol{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

- The Essential Matrix can be computed directly from the image coordinates (using SVD).
- At least 5 points needed! [Kruppa, 1913]. The more points, the better!
- The Essential Matrix can be decomposed into $\boldsymbol{R}$ and $\boldsymbol{t}$ (again using SVD)

# 2D-to-2D
## Computing the Essential Matrix



- The Essential matrix can be computed from 5 point correspondences using [Nister'2003] algorithm (5-*point algorithm)*
- *The 5-p algorithm* has become the standard for 2D-to-2D motion estimation, however, its implementation is not straightforward
- A simple and straightforward solution for $n \geq 8$ noncoplanar points is the Longuet-Higgins' *8-p algorithm*, which is summarized here:
- Let $p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$, $p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$ be the coordinates one feature correspondence
- $E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} e_{11} \\ \vdots \\ e_{33} \end{bmatrix}$
- ${p_2}^{\mathrm{T}} E p_1 = 0 \Rightarrow [\boldsymbol{x_1x_2}\ \boldsymbol{y_1x_2}\ \boldsymbol{z_1x_2}\ \boldsymbol{x_1y_2}\ \boldsymbol{y_1y_2}\ \boldsymbol{z_1y_2}\ \boldsymbol{x_1z_2}\ \boldsymbol{y_1z_2}\ \boldsymbol{z_1z_2}]E = 0$

  which can be solved with SVD

# 2D-to-2D Algorithm

| Motion estimation | | |
|---|---|---|
| 2D-2D | 3D-3D | 3D-2D |

**Algorithm 1: VO from 2D-to-2D correspondeces**

1 Capture new frame $I_k$
2 Extract and match features between $I_{k-1}$ and $I_k$,
3 Compute essential matrix for image pair $I_{k-1}$, $I_k$
4 Decompose essential matrix into $R_k$ and $t_k$, and form $T_k$
5 Compute relative scale and rescale $t_k$ accordingly
6 Concatenate transformation by computing $C_k = C_{k-1}T_k$
7 Repeat from 1

**How do we compute the relative scale between $I_{k-2}, I_{k-1}$, and $I_k$ ?**

# 2D-to-2D
# Relative Scale Computation



- The absolute scale of the translation cannot be computed from two images.
- However, it is possible to compute relative scales for the subsequent transformations.
- One way of doing this is to triangulate 3D points $X_{k-1}$ and $X_k$ from two subsequent image pairs. From corresponding 3D points, the relative distances between any combination of two 3D points can be computed.
- The proper scale can then be determined from the distance ratio *r* between a point pair in $X_{k-1}$ and a pair in $X_k$

$$r = \frac{||X_{k-1,i} - X_{k-1,j}||}{||X_{k,i} - X_{k,j}||}$$

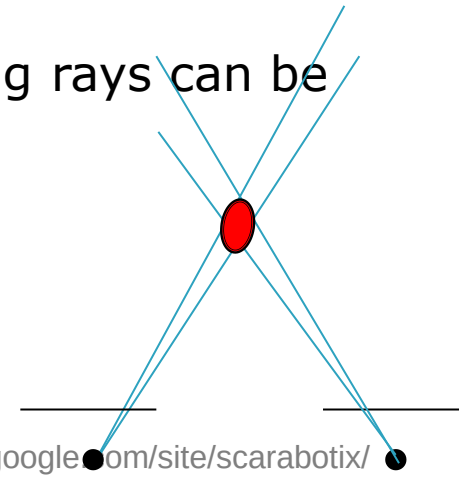
- For robustness, the scale ratios for many point pairs are computed and the mean (or in presence of outliers the median) is used

# Motion Estimation: Summary

| Type of correspondences | Monocular | Stereo |
|---|---|---|
| 2D-2D | X | X |
| 3D-3D | | X |
| 3D-2D | X | X |

# Triangulation and Keyframe Selection

- Some of the previous motion estimation methods require triangulation of 3D points
- Triangulated 3D points are determined by intersecting backprojected rays from 2D image correspondences of at least two image frames
- In reality, they never intersect due to
  - image noise,
  - camera model and calibration errors,
  - and feature matching uncertainty
- The point at minimal distance from all intersecting rays can be taken as an estimate of the 3D point position

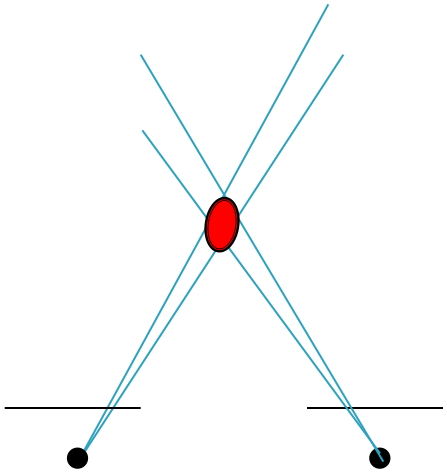


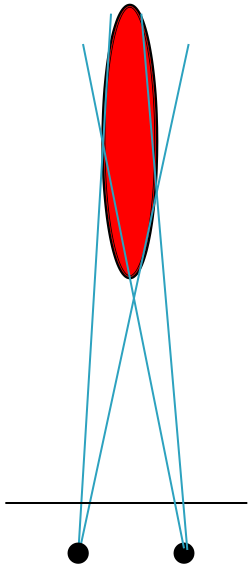
# Triangulation and Keyframe Selection

- When frames are taken at nearby positions compared to the scene distance, 3D points will exibit large uncertainty

- **Therefore, 3D-3D motion estimation methods will drift much more quickly than 3D-2D and 2D-2D methods**

- In fact, the uncertainty introduced by triangulation affects the motion estimation. In fact, in the 3D-to-3D case the 3D position error is minimized, while in the 3D-to-2D and 2D-to-2D cases is the image reprojection error

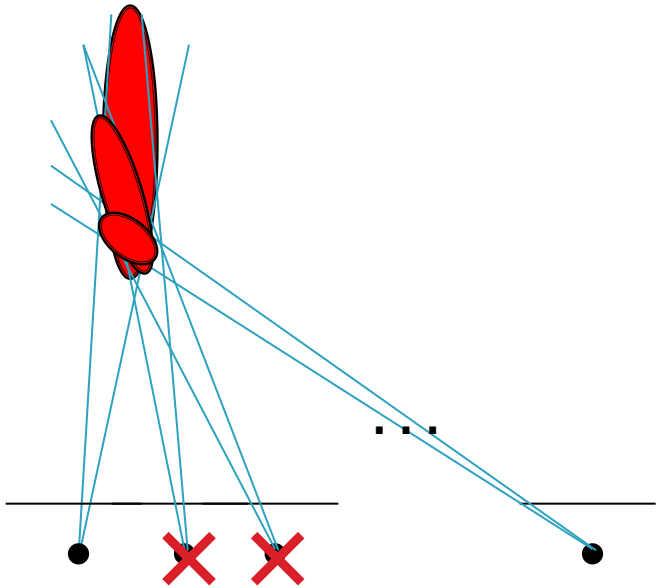




# Triangulation and Keyframe Selection

- One way to avoid this consists of skipping frames until the average uncertainty of the 3D points decreases below a certain threshold. The selected frames are called *keyframes*

- Keyframe selection is a very important step in VO and should always be done before updating the motion

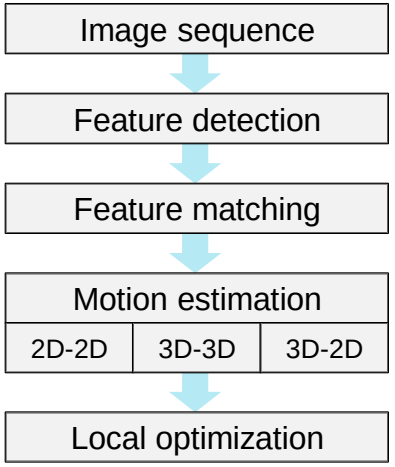


# Summary Considerations

- In the Stereo vision case, 3D-2D method exhibits less drift than 3D-3D method
- Stereo vision has the advantage over monocular vision that both motion and structure are computed in the absolute scale. It also exhibits less drift.
- When the distance to the scene is much larger than the stereo baseline, stereo VO degenerates into monocular VO
- Keyframes should be selected carefully to reduce drift
- Regardless of the chosen motion computation method, local bundle adjustment (over the last *m* frames) should be always performed to compute a more accurate estimate of the trajectory. **After bundle adjustment, the effects of the motion estimation method are much more alleviated** (as long as the initialization is close to the solution)

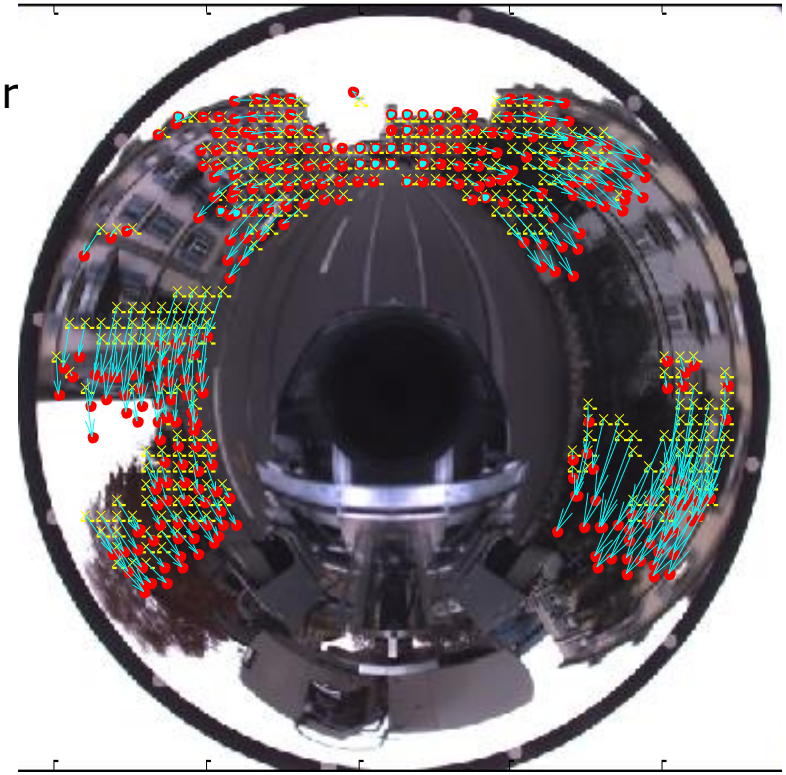
# Outline

- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion

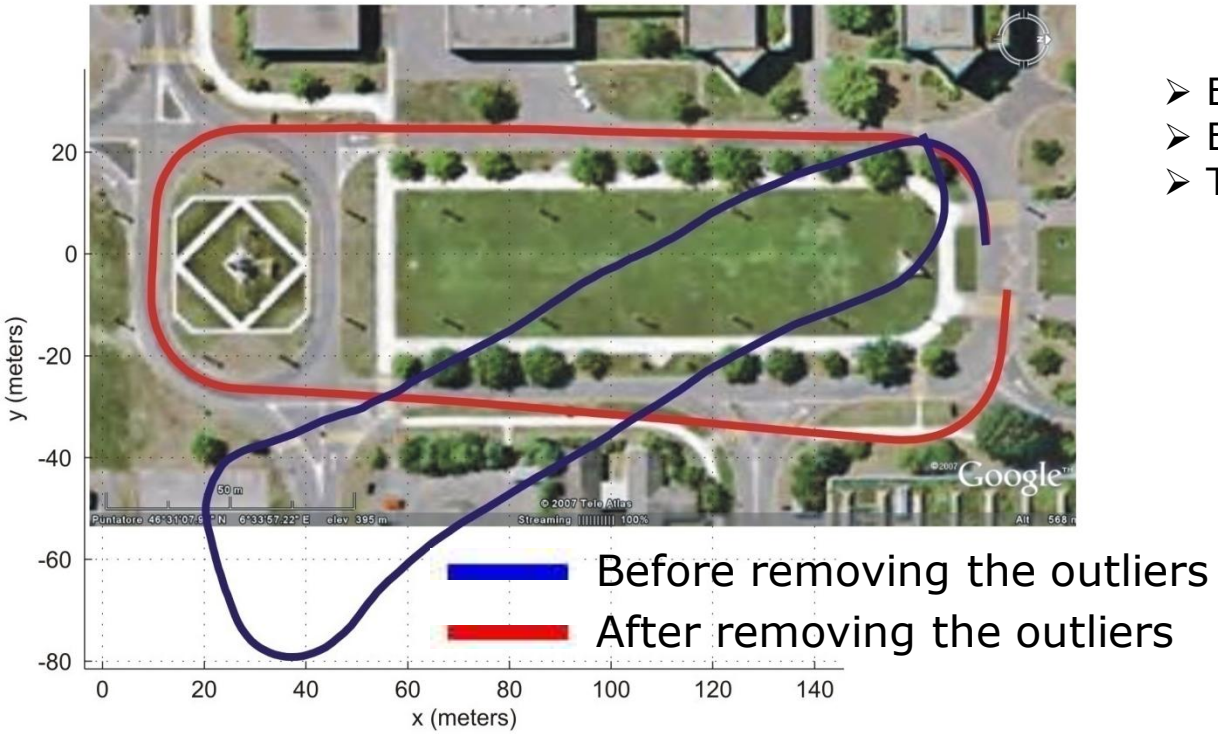


# Robust Estimation

- Matched points are usually contaminated by outliers, that is, wrong data associations
- Possible causes of outliers are
  - image noise,
  - occlusions,
  - blur,
  - changes in view point and illumination for which the mathematical model of the feature detector or descriptor does not account for
- For the camera motion to be estimated accurately, outliers must be removed
- This is the task of Robust Estimation



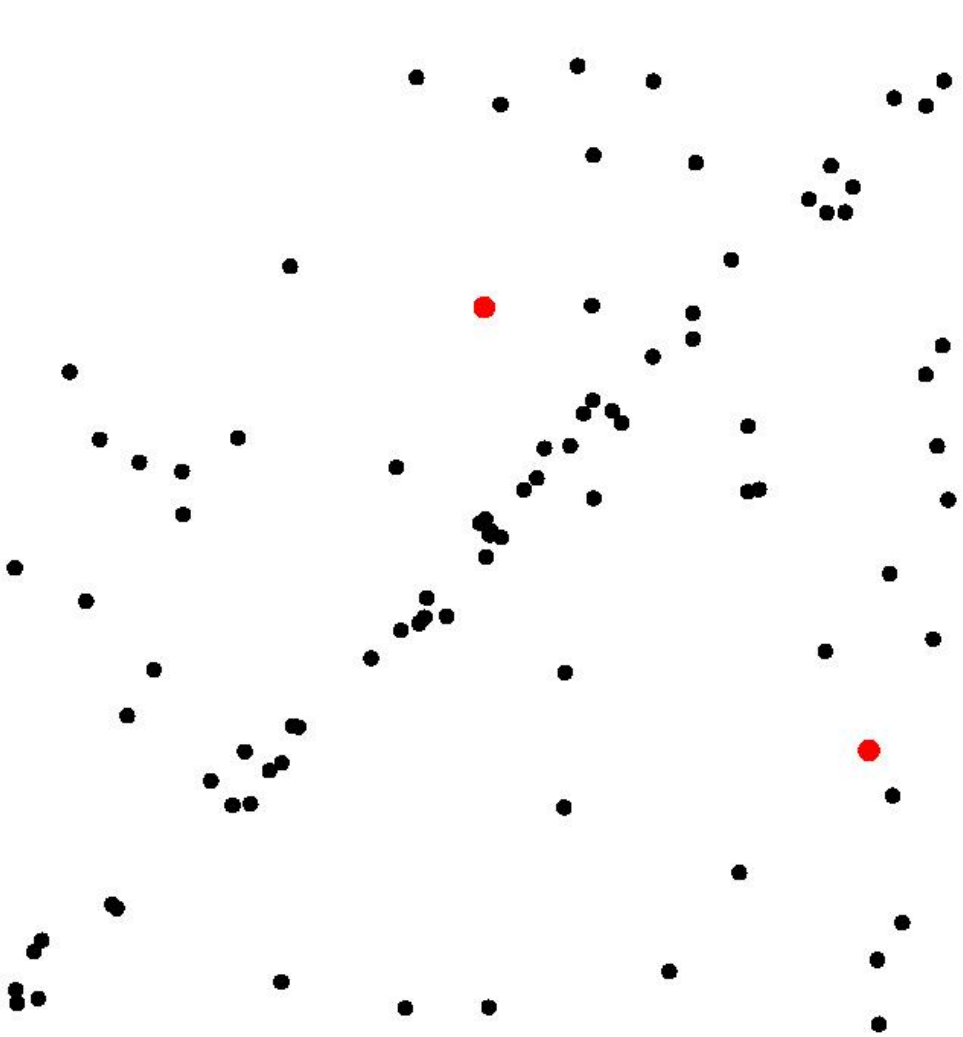
# Influence of Outliers on Motion Estimation




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

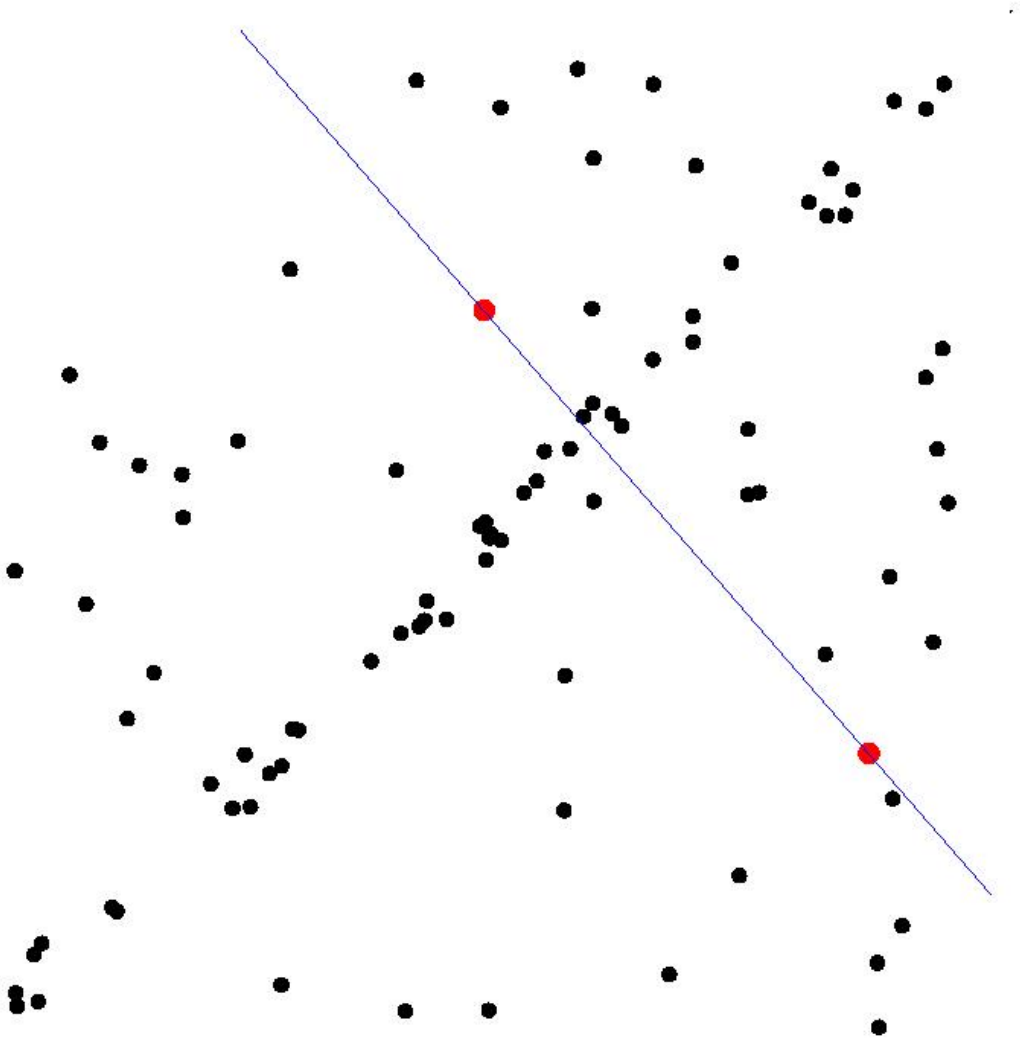
# RANSAC Example: Line Extraction

# RANSAC Example: Line Extraction



- Select sample of 2 points at random

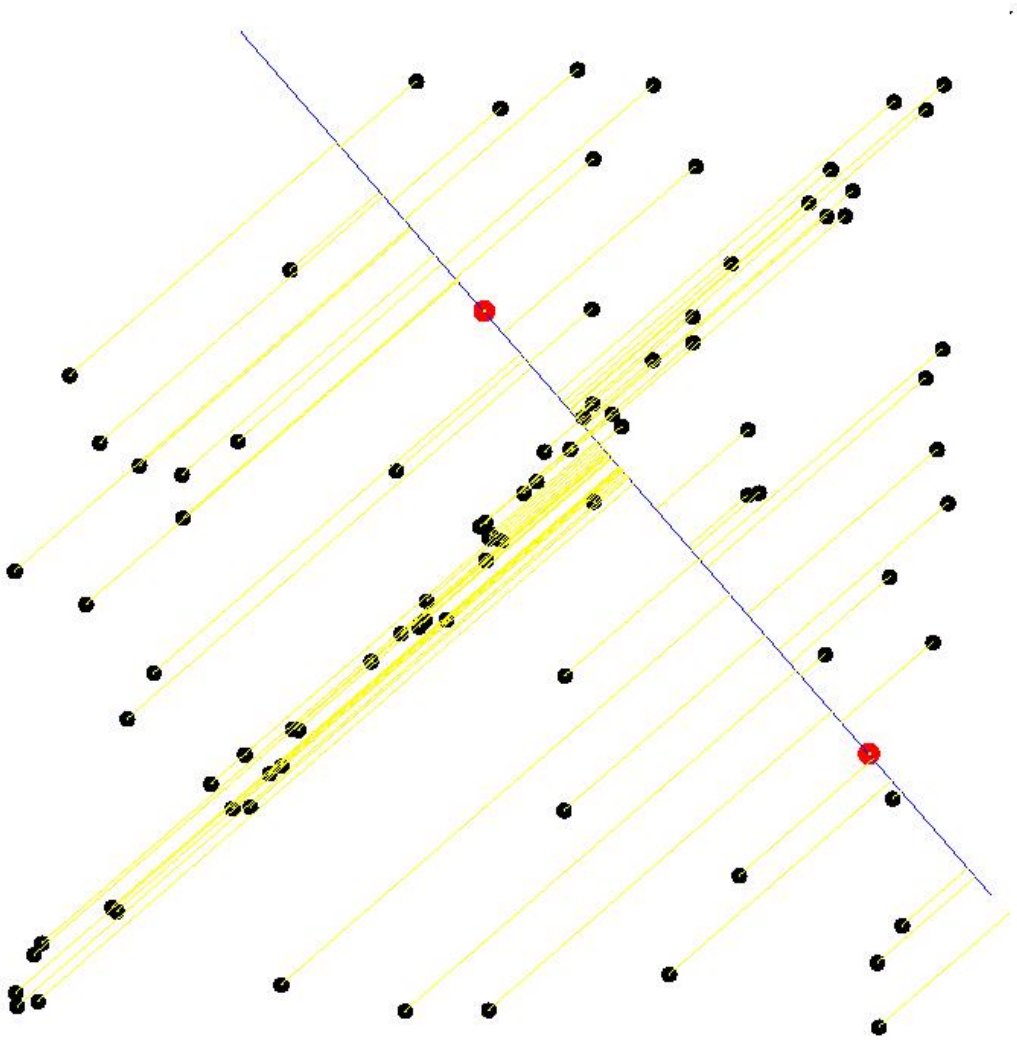
# RANSAC Example: Line Extraction



- Select sample of 2 points at random

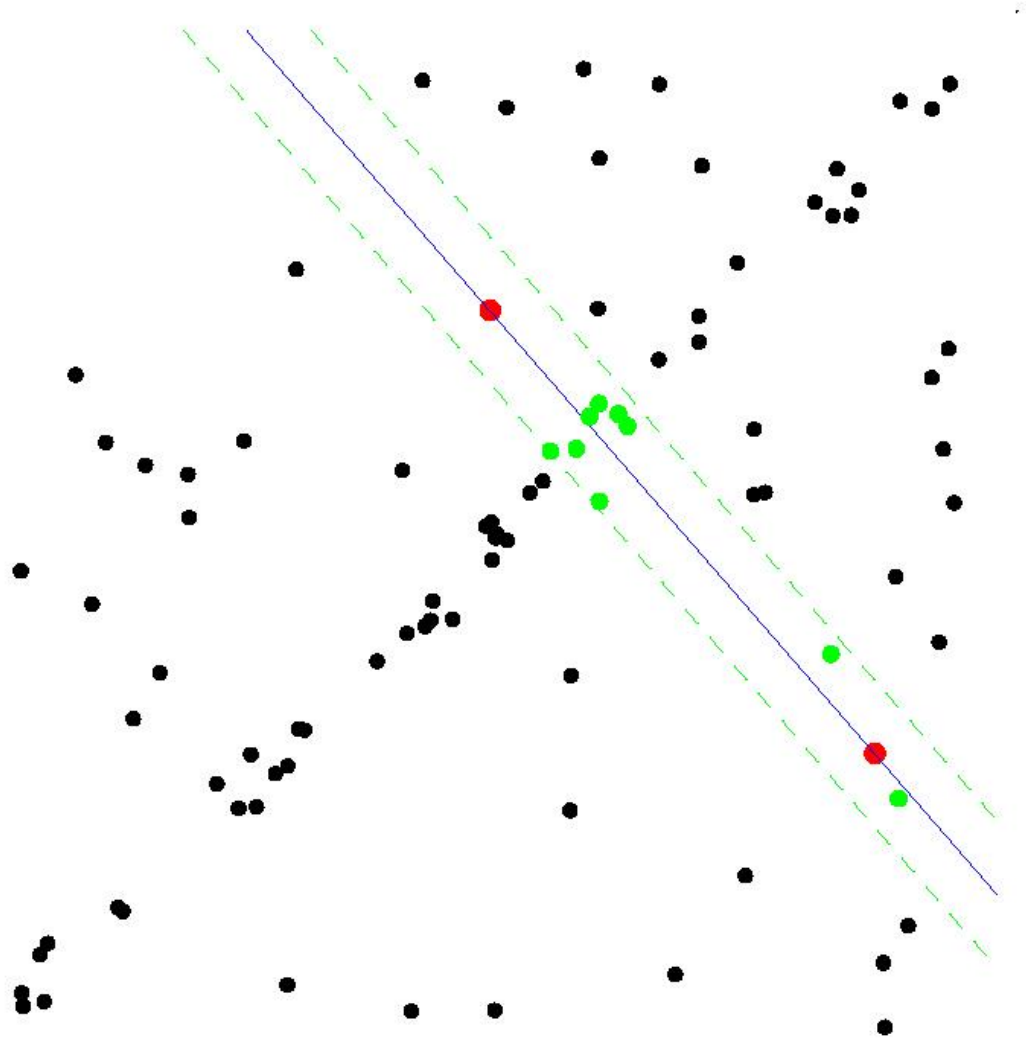
- Calculate model parameters that fit the data in the sample

# RANSAC Example: Line Extraction



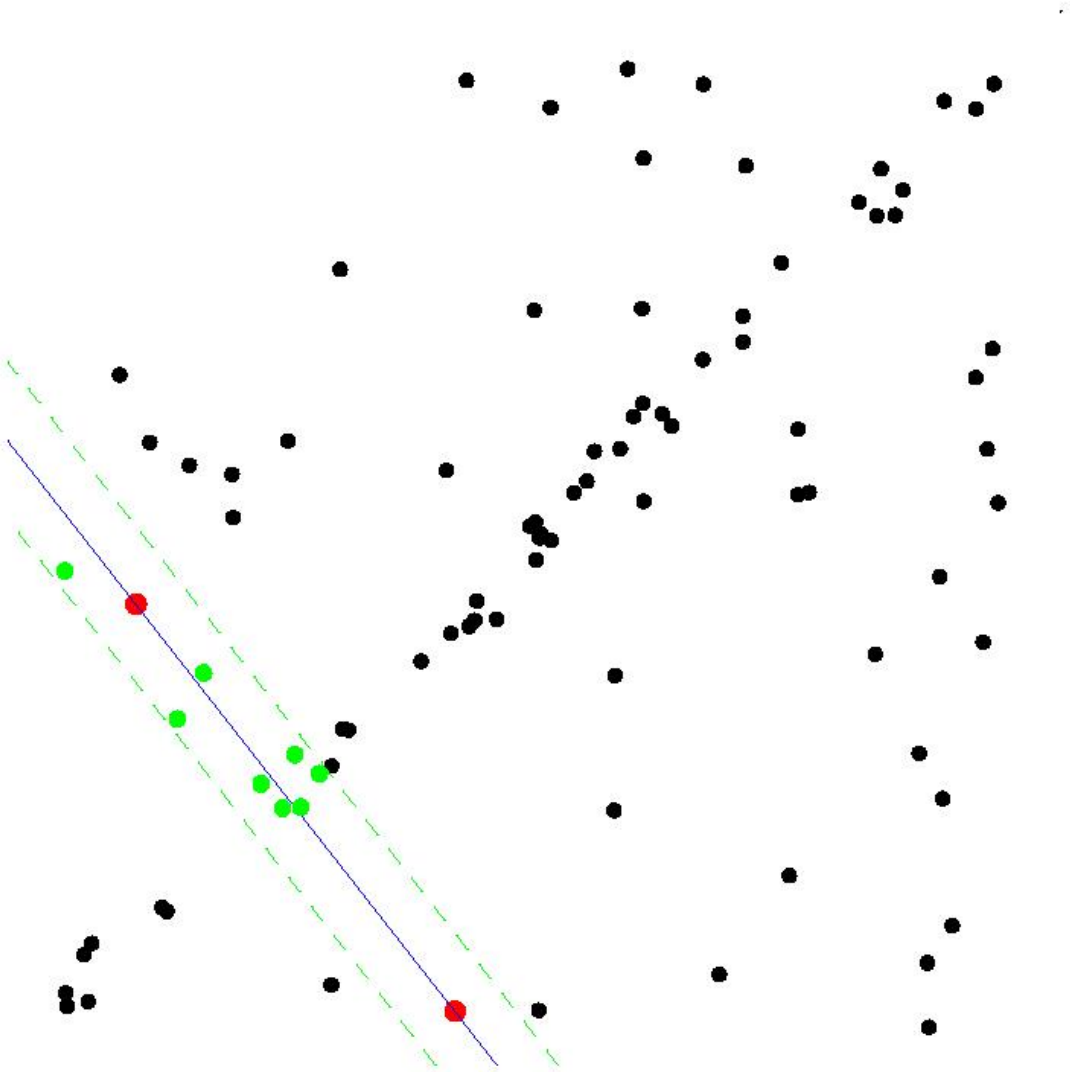
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point

# RANSAC Example: Line Extraction



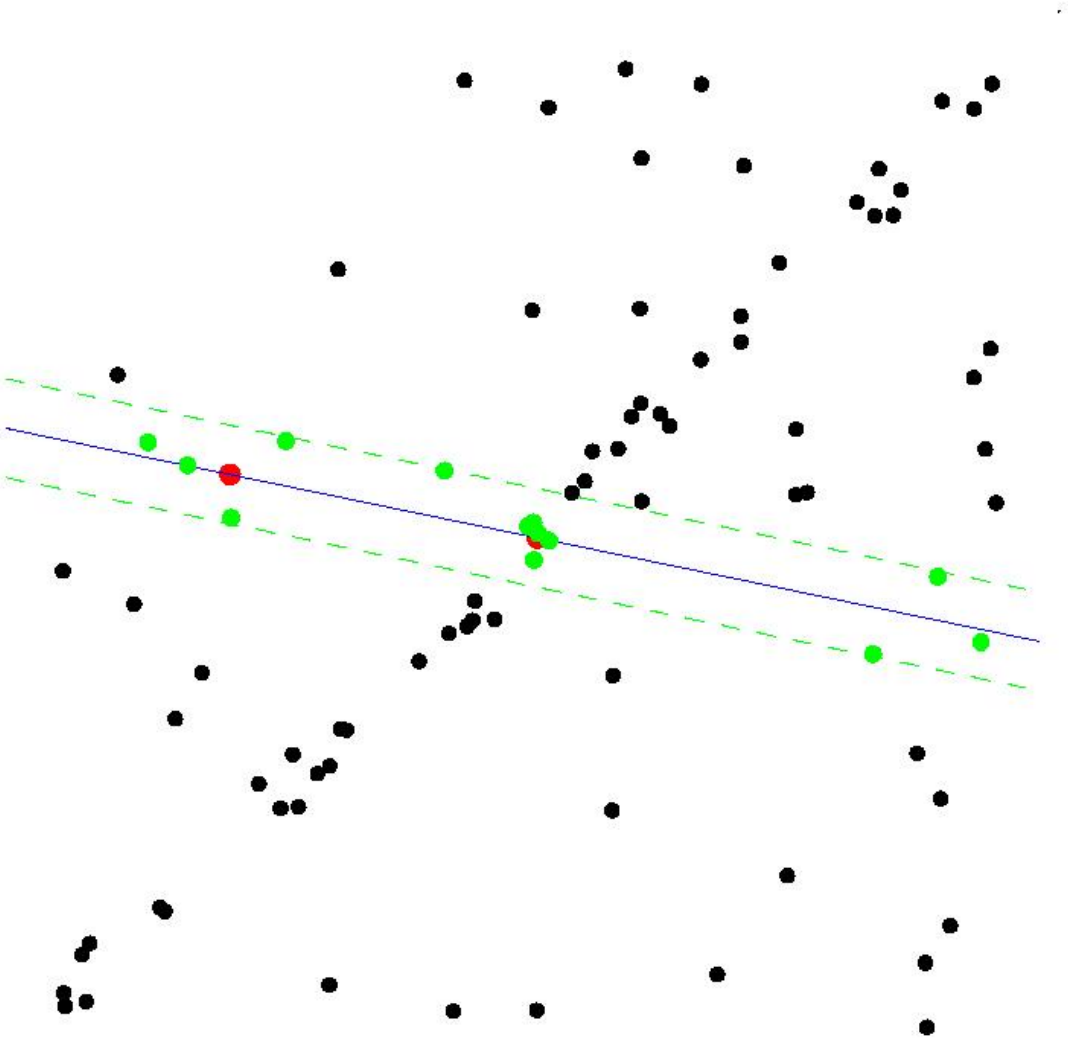
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis

# RANSAC Example: Line Extraction



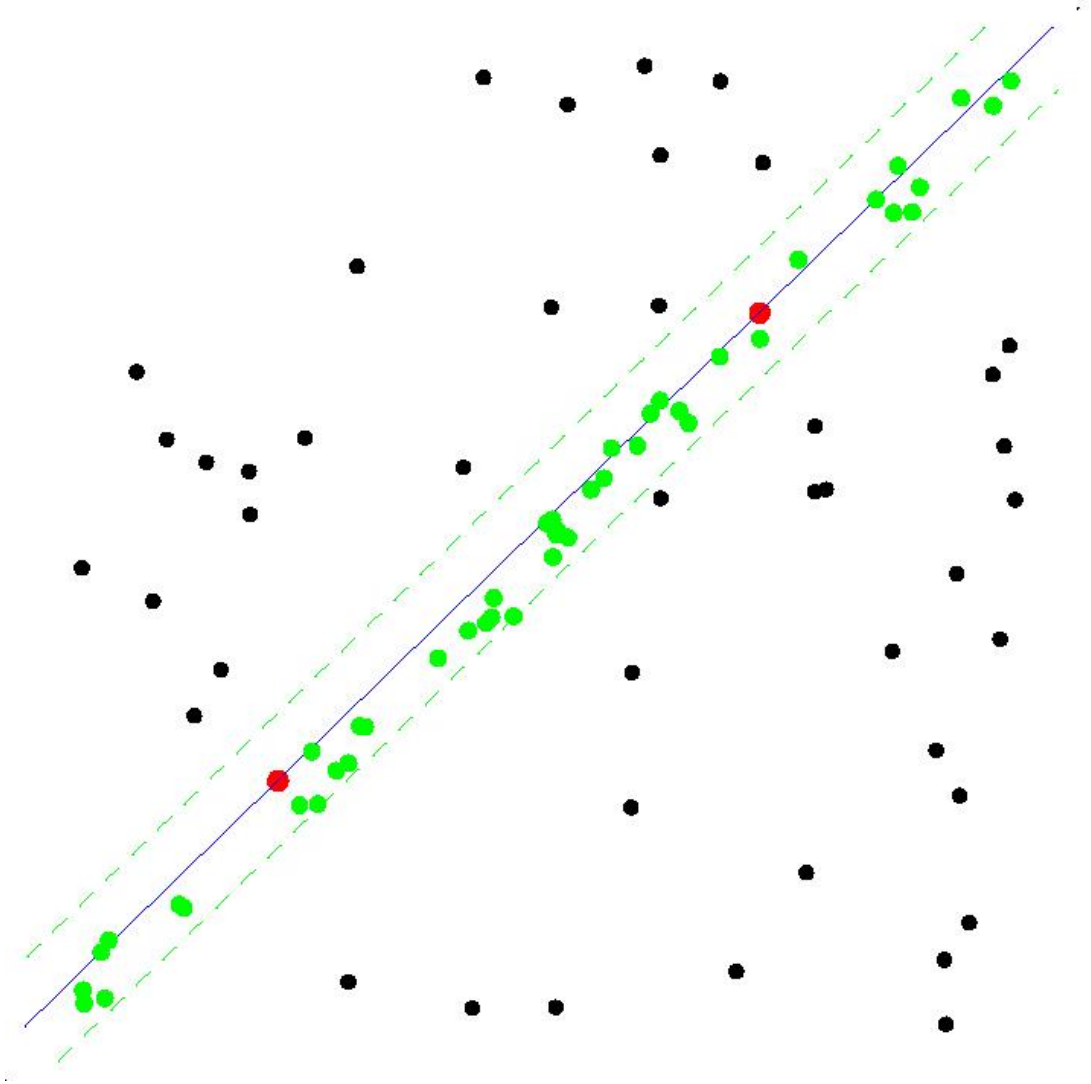
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- Repeat sampling

# RANSAC Example: Line Extraction



- 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 sampling

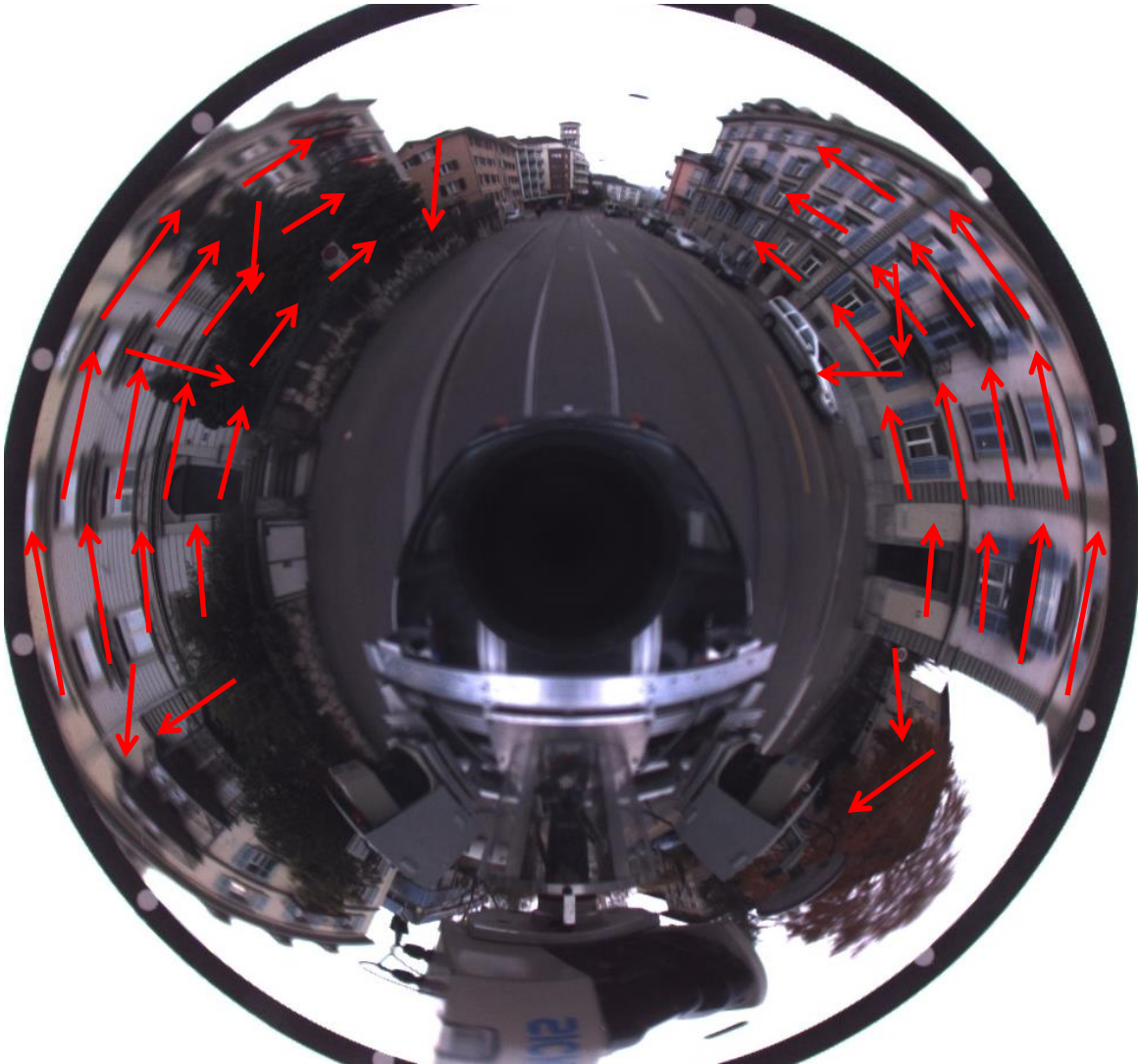
# RANSAC Example: Line Extraction



**ALL-INLIER SAMPLE**

# RANSAC [Fishler & Bolles, 1981]

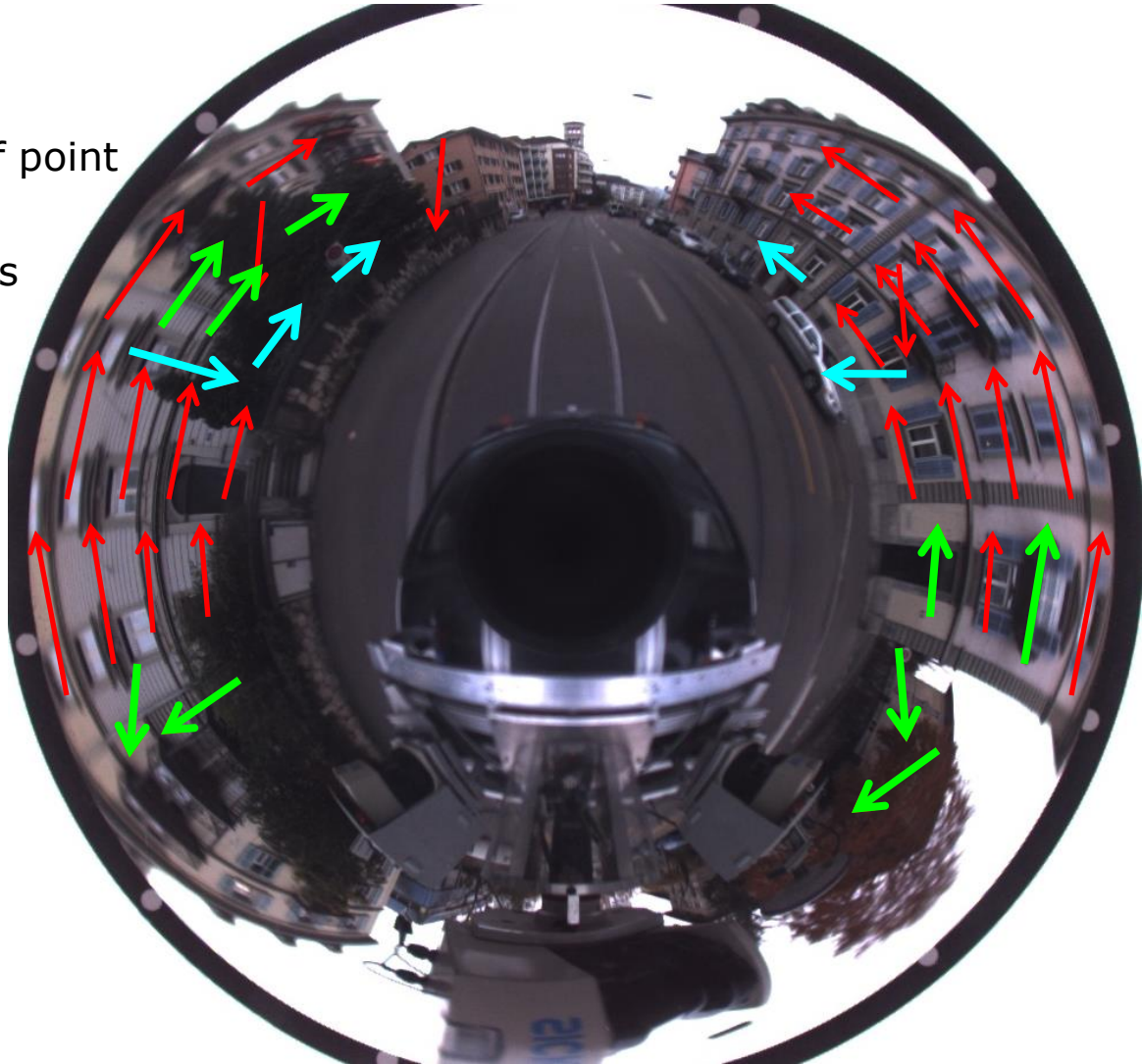
Has been established as the standard method for motion estimation in the presence of outliers



# RANSAC [Fishler & Bolles, 1981]

Has been established as the standard method for motion estimation in the presence of outliers

1. Randomly select a minimal set of point correspondences
2. Compute motion and count inliers
3. Repeat from 1



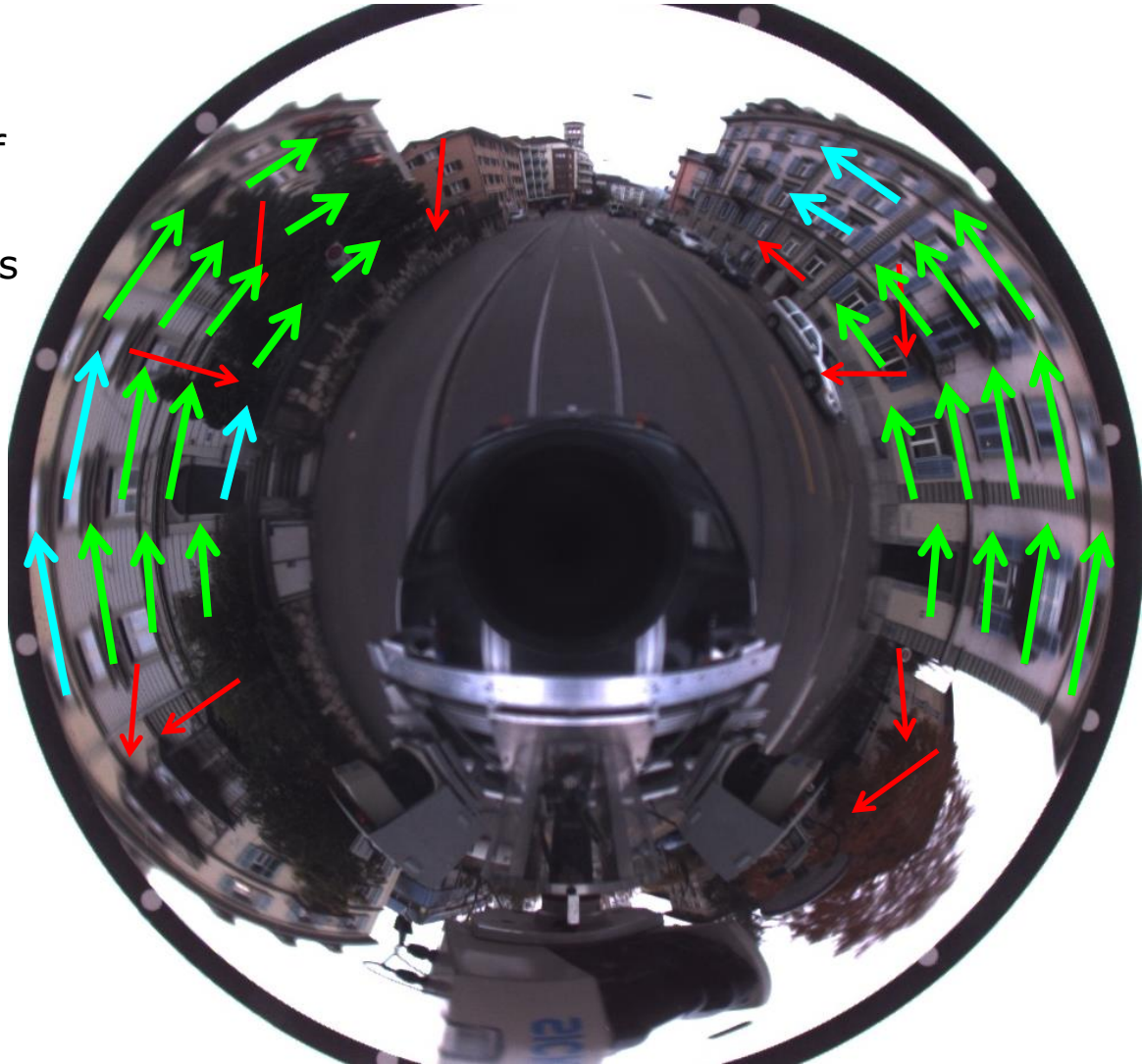
# RANSAC [Fishler & Bolles, 1981]

Has been established as the standard method for motion estimation in the presence of outliers

1. Randomly select a minimal set of point correspondences
2. Compute motion and count inliers
3. Repeat N times

**The number of iterations needed grows exponentially with the outliers**

**~ 1000 iterations!**



# RANSAC Algorithm

**Algorithm 1: RANSAC**

1) Initial: let A be a set of N feature correspondences
2) repeat
2.1) Randomly select a sample of $s$ points from A
2.2) Fit a model to these points
2.3) Compute the distance of all other points to this model
2.4) Construct the inlier set (i.e. count the number of points whose distance from the model $< d$)
2.5) Store these inliers
2.6) until maximum number of iterations reached
3) The set with the maximum number of inliers is chosen as a solution to the problem
4) Estimate the model using all the inliers

# How many iterations of RANSAC ?

- The number of iterations $N$ that is necessary to guarantee that a correct solution is found can be computed by

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

- $\boldsymbol{s}$ is the number of points from which the model can be instantiated
- $\boldsymbol{\varepsilon}$ is the percentage of outliers in the data
- $\boldsymbol{p}$ is the requested probability of success

- Example: $\boldsymbol{p = 99\%,\ s = 5,\ \varepsilon = 50\% \Rightarrow N = 145}$

# How many iterations of RANSAC ?

- RANSAC is an iterative method and is **non deterministic** in that it **exhibits a different solution** on different runs; however, the solution tends to be stable when the number of iterations grows

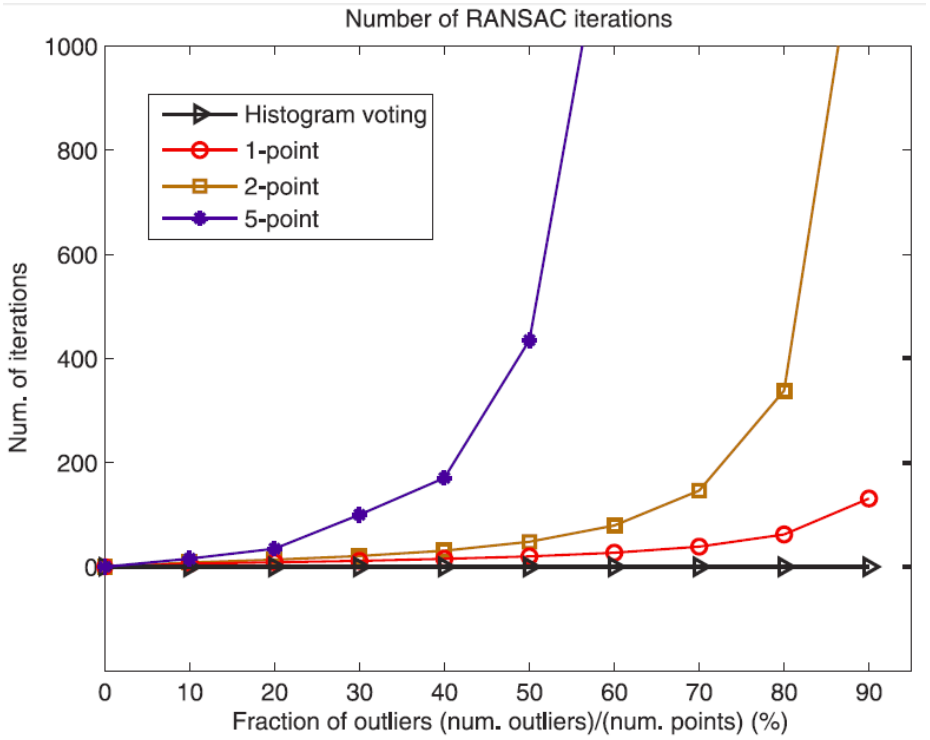
- For the sake of robustness, in many practical implementations $N$ is usually multiplied by a factor of 10

- More advanced implementations of RANSAC **estimate the fraction of inliers adaptively**, iteration after iteration (like in your exercise today)

# RANSAC iterations

- $N$ is exponential in the number of data points $s$ necessary to estimate the model
- Therefore, there is a high interest in using a minimal parameterization of the model

| Number of points ($s$): | 8 | 7 | 6 | 5 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| Number of iterations ($N$): | 1177 | 587 | 292 | 145 | 71 | 16 | 7 |



# What is the minimum number of points to estimate motion ?

*To estimate the motion of a calibrated camera in 6 DoF,*
*we need 5 points*
[Kruppa, 1913]

Why ?

# What is the minimum number of points to estimate motion ?

*In 6 DoF we would need 6 points ...*

*... but the scale is unobservable ...*

*... and therefore we only need 6 – 1 = 5 points*

["5-Point RANSAC", Nister, 2003]

General rule:

Minimum number of points = $N_{DoF}$ - 1

# What is the minimum number of points to estimate motion ?

The “5-Point RANSAC” typically needs ~1000 iterations

To reduce the number iterations, we should use a smaller number of points ( < 5 )

Is this possible?

*Yes, if we exploit motion constraints!*

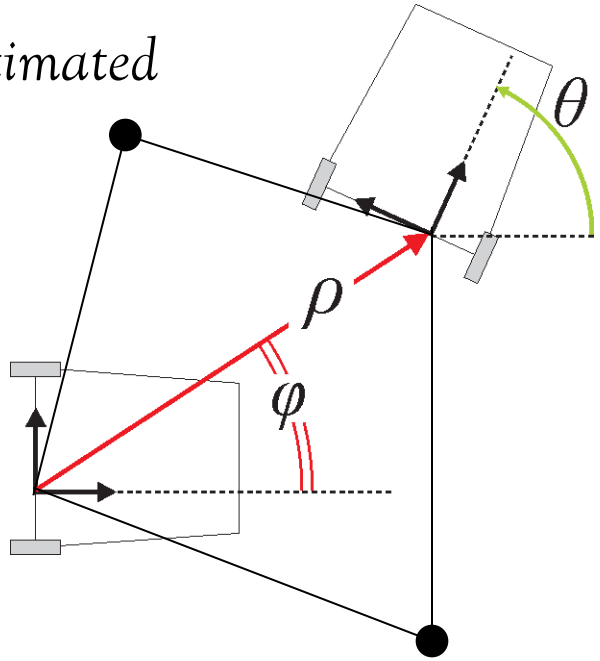
# What is the minimum number of points to estimate motion ?

*For planar motion, only 3 parameters need to be estimated*

$\theta, \varphi, \rho \Rightarrow$ *3 DoF*

*and therefore only 2 points are needed*

["2-Point RANSAC", Ortin, 2001]

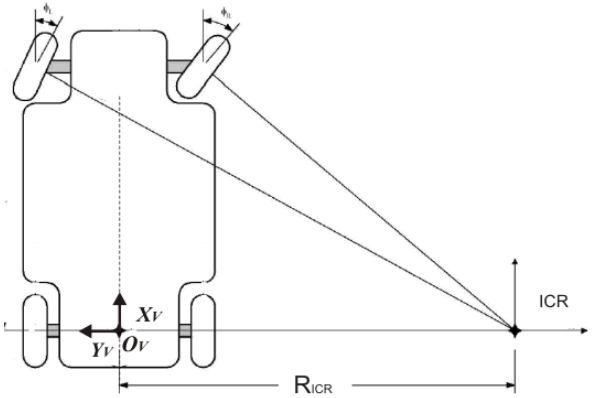



**Can we use an even smaller number of points?**

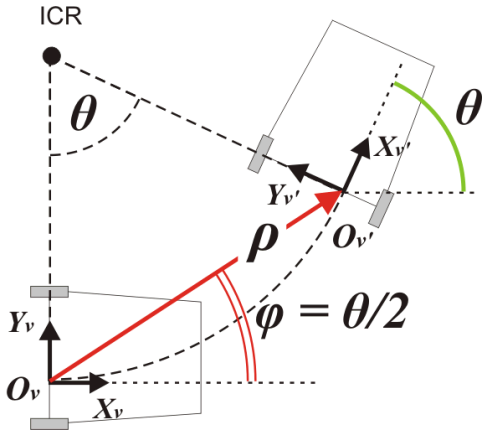
*Yes, if we exploit the vehicle* non-holonomic *constraints*

# Exploiting Vehicle's Non-Holonomic Constraints ?

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



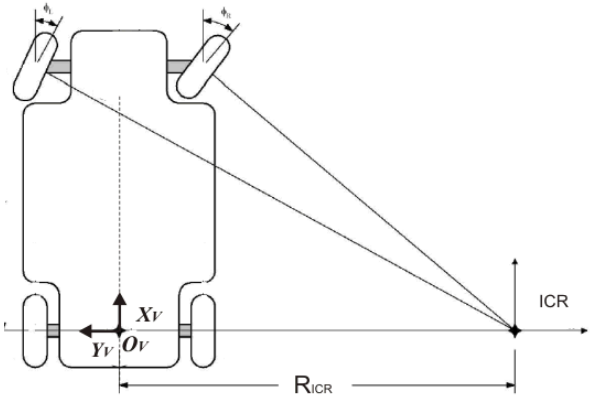

Example of Ackerman steering principle



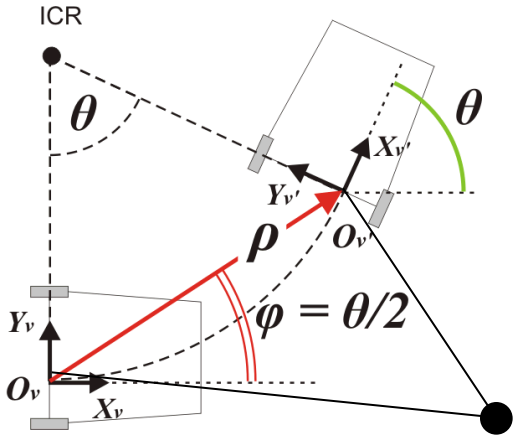

Locally circular motion

# Exploiting Vehicle's Non-Holonomic Constraints ?

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




Example of Ackerman steering principle




Locally circular motion

*$\varphi = \theta/2$ => only 2 parameters $(\theta, \rho)$ need to be estimated*

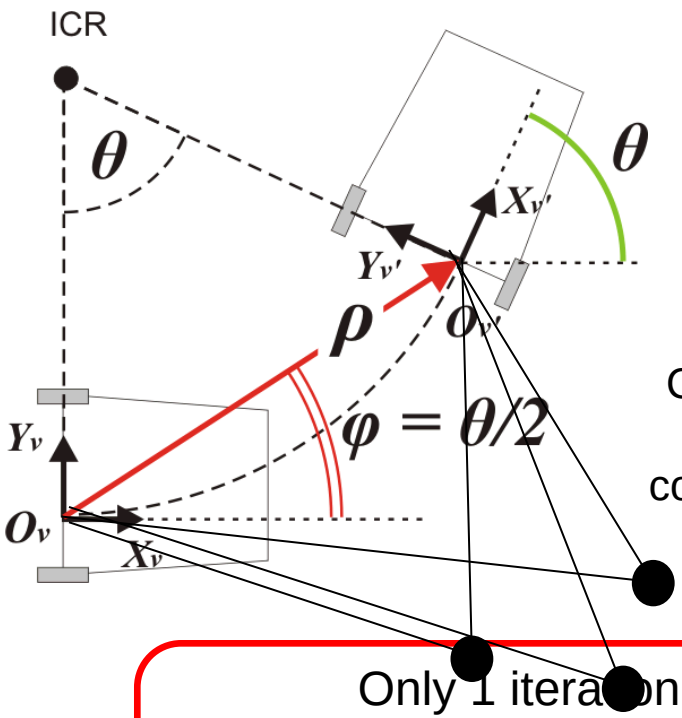
*and therefore only 1 point is needed*

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

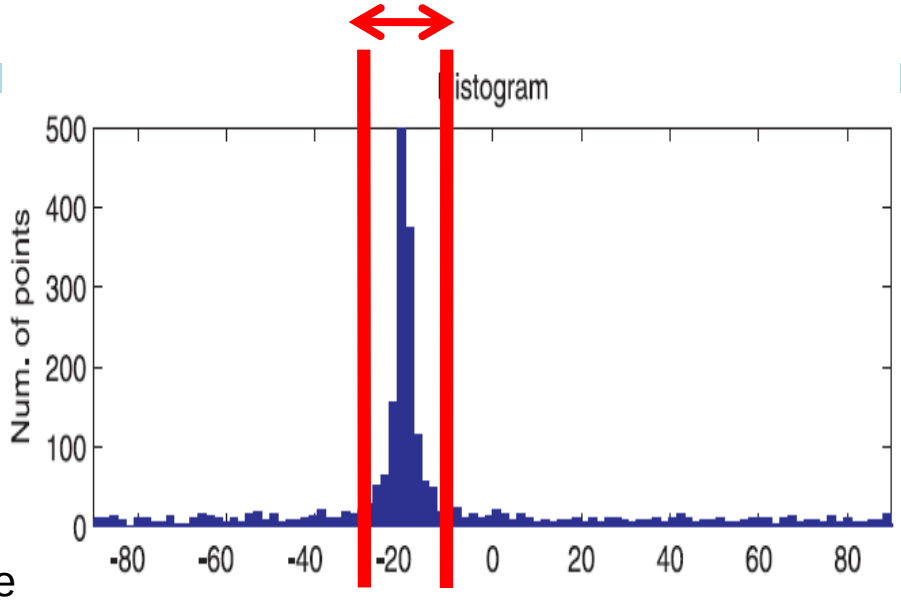

D. Scaramuzza. **1-Point-RANSAC Structure from Motion for Vehicle-Mounted Cameras by Exploiting Non-holonomic Constraints**. International Journal of Computer Vision, Volume 95, Issue 1, 2011

D. Scaramuzza. **Performance Evaluation of 1-Point-RANSAC Visual Odometry**. Journal of Field Robotics, Vol. 28, issue 5, 2011

# 1-Point RANSAC algorithm

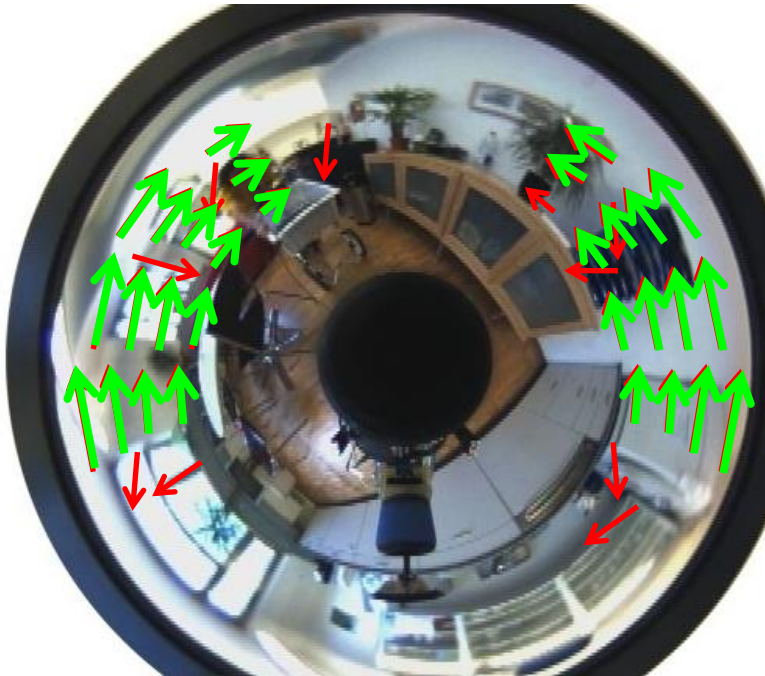



Compute $\theta$ for every point correspondence



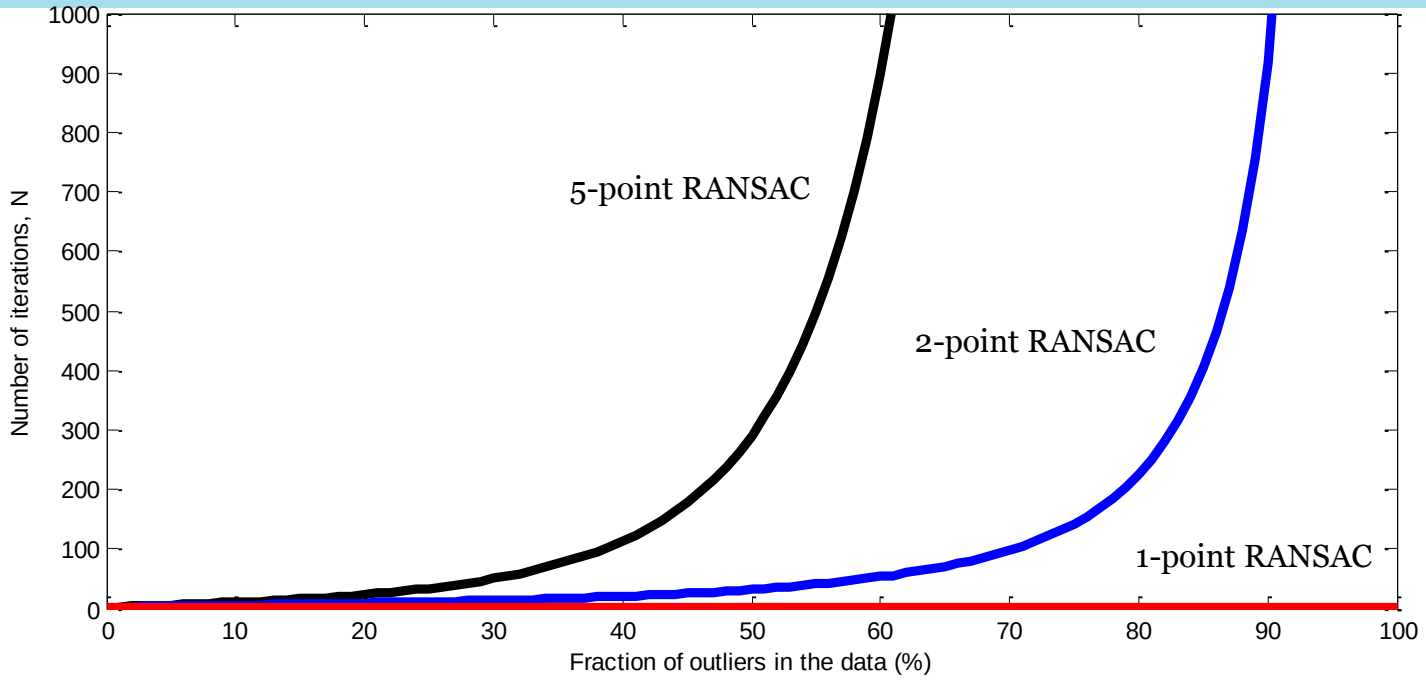

Only 1 iteration

The most efficient algorithm for removing outliers, up to 800 Hz



1-Point RANSAC is ONLY used to find the inliers.
Motion is then estimated from them in 6DOF

# Comparison of RANSAC algorithms

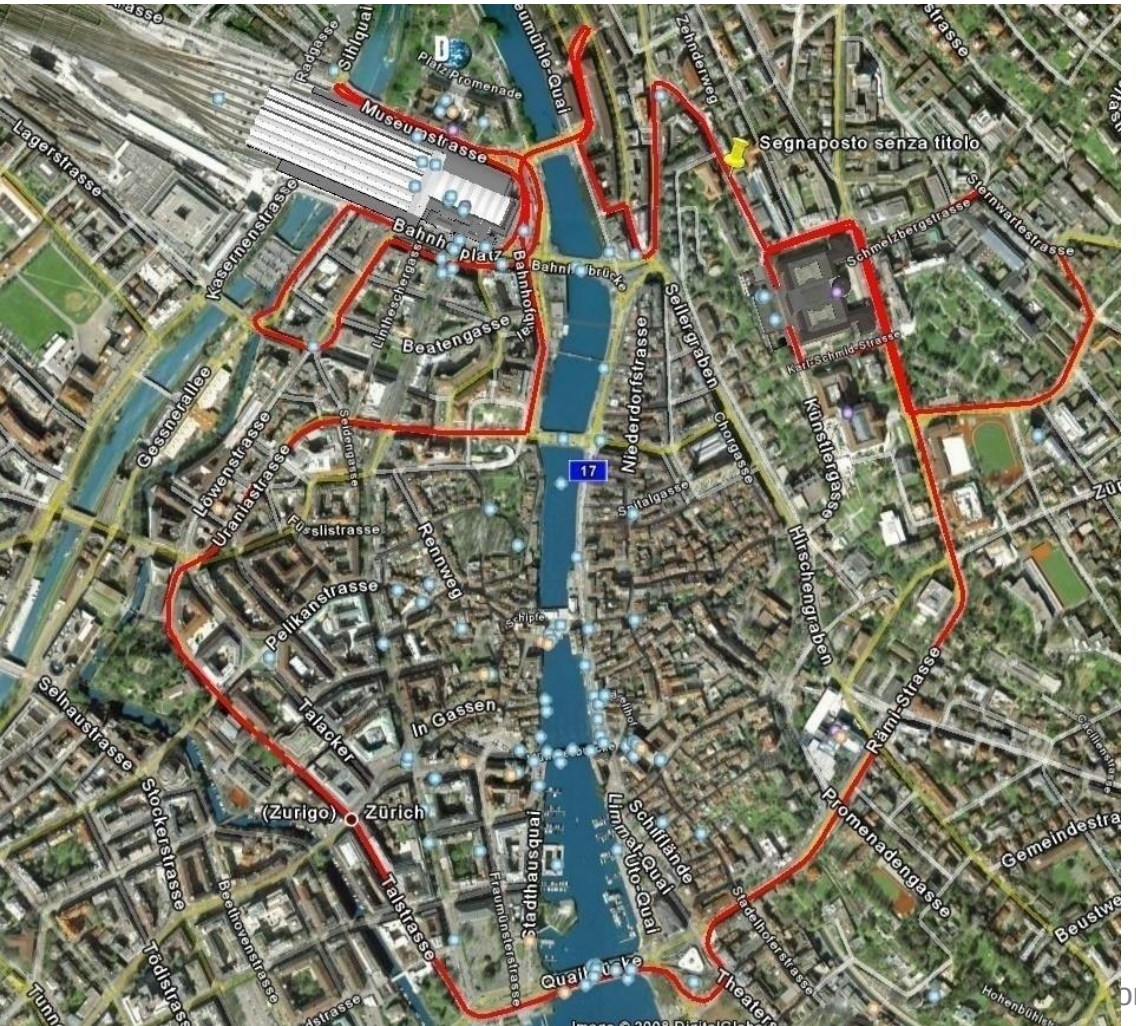


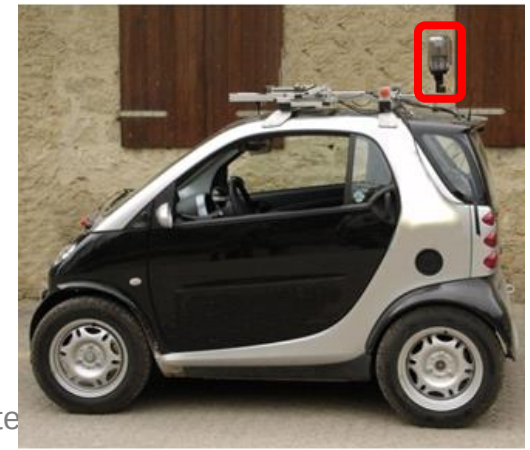

| | 5-Point RANSAC [Nister'03] | 2-Point RANSAC [Ortin'01] | 1-Point RANSAC [Scaramuzza, IJCV'11, JFR'11] |
|---|---|---|---|
| Number of iterations | **~1000** | **~100** | **1** |

Our proposed method

# Visual Odometry: Results in a Urban Environment

- 15,000 images collected in Zurich during a over 25 Km path
- Image resolution: 640 x 480

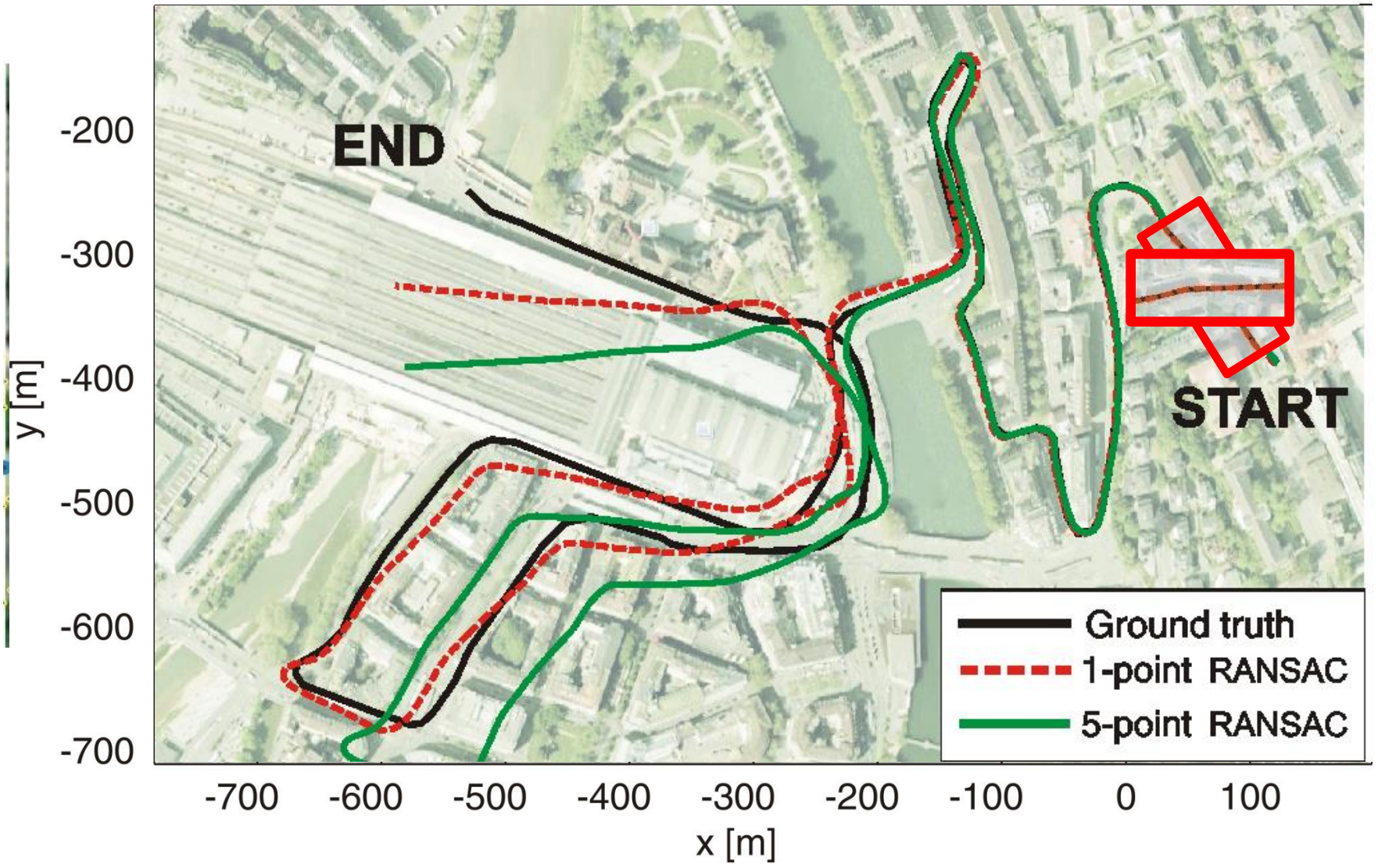


org - https://site

**This video can be seen at**
**http://youtu.be/t7uKWZtUjCE**

Comparison between visual odometry and ground truth



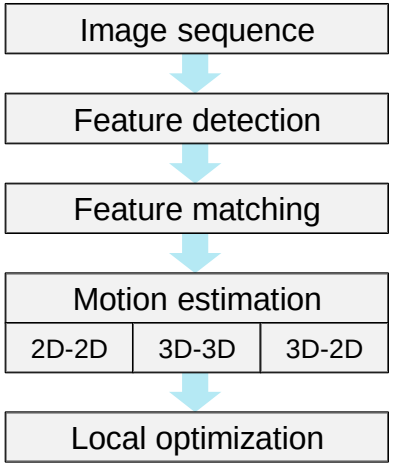
# Considerations about RANSAC

**Is it really better to use minimal sets in RANSAC?**

- If one is concerned with certain speed requirements, YES
- However, might not be a good choice if the image correspondences are very noisy: in this case, the motion estimated from a minimal set wil be inaccurate and will exhibit fewer inliers when tested on all other points
- Therefore, when the computational time is not a real concern and one deals with very noisy features, **using a non-minimal set may be better than using a minimal set**

# Outline

- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion

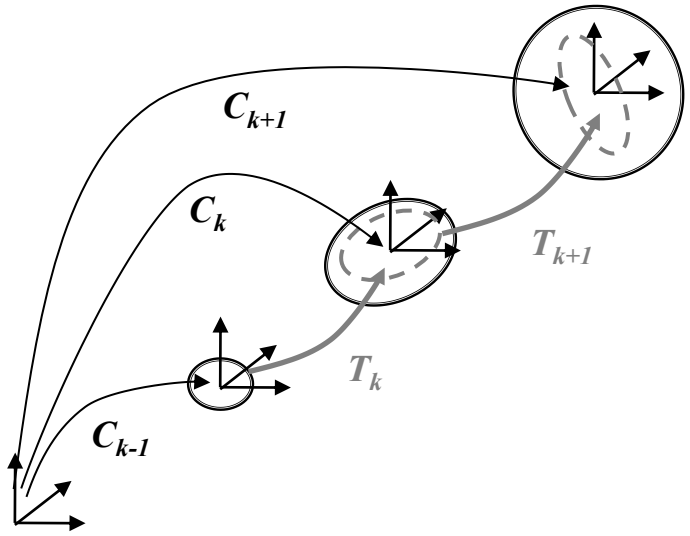


# Error Propagation

- The uncertainty of the camera pose $C_k$ is a combination of the uncertainty at $C_{k-1}$ (black-solid ellipse) and the uncertainty of the transformation $T_k$ (gray dashed ellipse)

- $C_k = f(C_{k-1}, T_k)$

- The combined covariance $\Sigma_k$ is

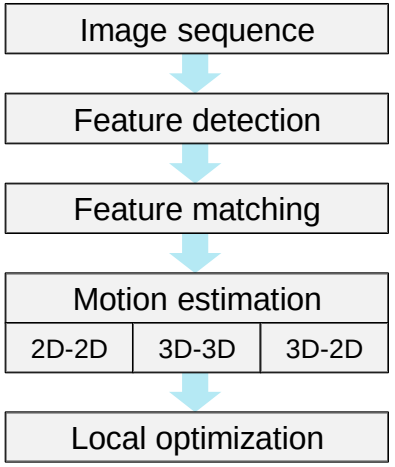



$$\begin{aligned}\Sigma_k &= J \begin{bmatrix} \Sigma_{k-1} & 0 \\ 0 & \Sigma_{k,k-1} \end{bmatrix} J^\top \\ &= J_{\vec{C}_{k-1}} \Sigma_{k-1} J_{\vec{C}_{k-1}}{}^\top + J_{\vec{T}_{k,k-1}} \Sigma_{k,k-1} J_{\vec{T}_{k,k-1}}{}^\top\end{aligned}$$

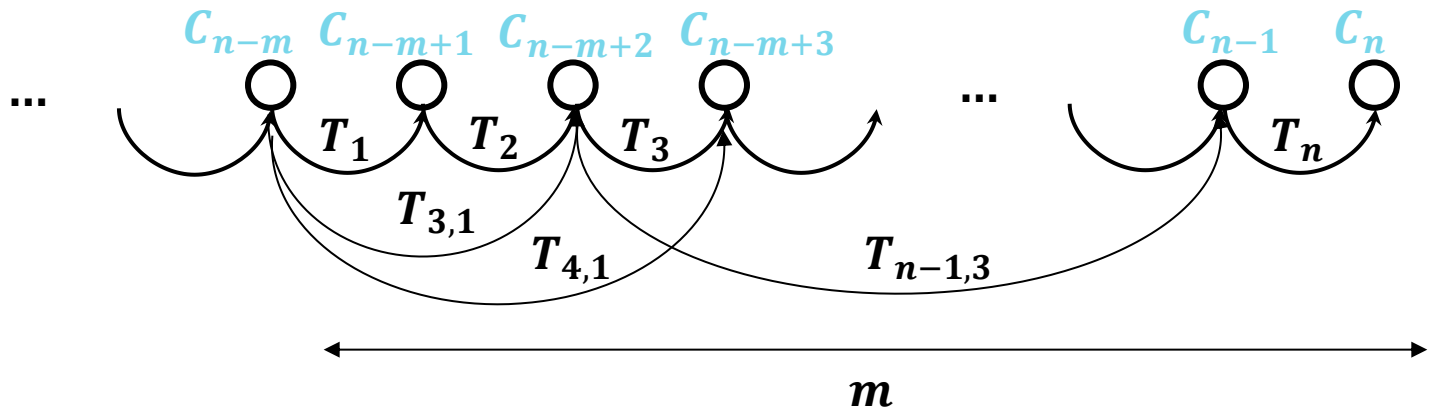
- The camera-pose uncertainty is always increasing when concatenating transformations. Thus, it is important to keep the uncertainties of the individual transformations small

# Outline

- Brief history of VO
- Problem formulation
- Camera modeling and calibration
- Motion estimation
- Robust estimation
- Error propagation
- Camera-pose optimization (bundle adjustment)
- Discussion



# Windowed Camera-Pose Optimization

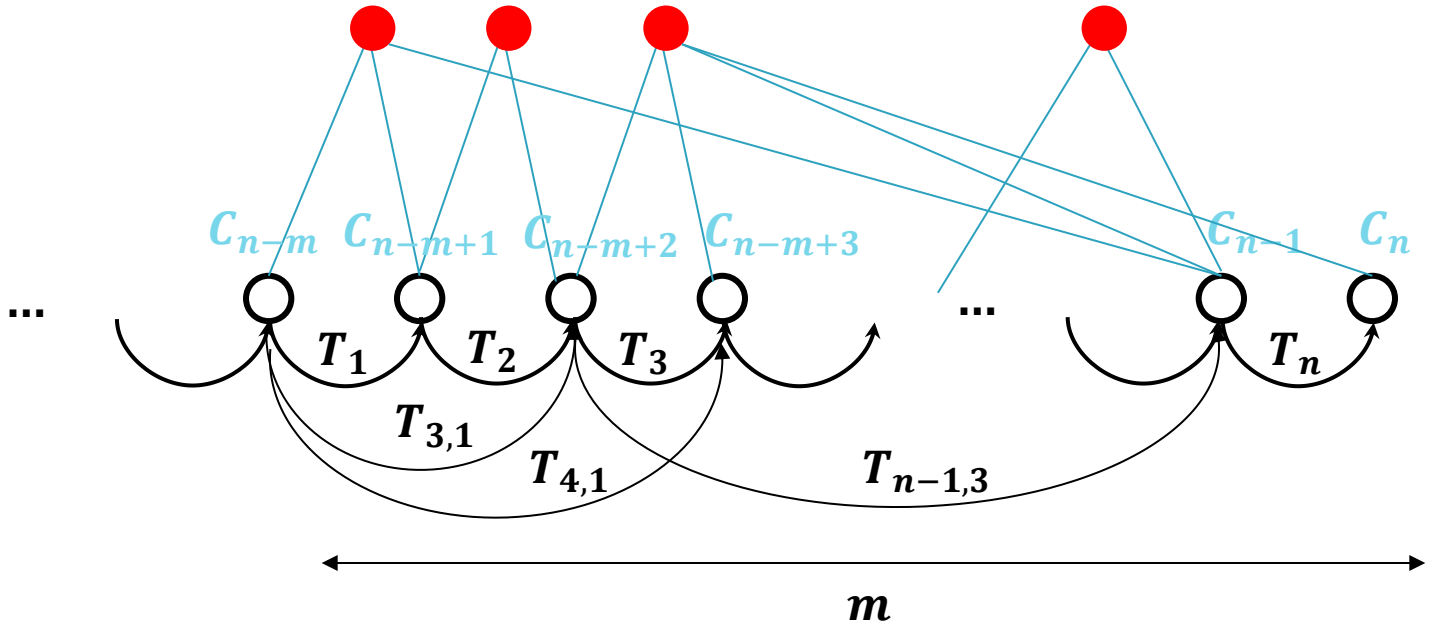



- So far we assumed that the transformations are between consecutive frames
- Transformations can be computed also between non-adjacent frames $T_{e_{ij}}$ and can be used as additional constraints to improve cameras poses by minimizing the following

$$\sum_{e_{ij}} \|C_i - T_{e_{ij}} C_j\|^2$$

- For efficiency, only the last $m$ keyframes are used
- Levenberg-Marquadt can be used

# Windowed Bundle Adjustment (BA)




- Similar to pose-optimization but it also optimizes 3D points

$$\arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$

- In order to not get stuck in local minima, the initialization should be close the minimum
- Levenberg-Marquadt can be used

# Loop Detection

- Loop constraints are very valuable constraints for pose graph optimization
- These constraints form graph edges between nodes that are usually far apart and between which large drift might have been accumulated.
- Events like reobserving a landmark after not seeing it for a long time or coming back to a previously-mapped area are called loop detections
- Loop constraints can be found by evaluating visual similarity between the current camera images and past camera images.
- Visual similarity can be computed using global image descriptors or local image descriptors (see lecture about Visual SLAM)

First observation

Second observation after a loop

Image courtesy of Cummins & Newman, IJRR'08

# Considerations about Bundle Adjustment

- Windowed BA reduces the drift compared to 2-view VO because incorporates constraints between several frames
- More precise than camera-pose optimization
- The choise of the window size *m* is governed by computational reasons
- The computational complexity of BA is $O\left((qN + lm)^3\right)$ with $N$ being the number of points, $m$ the number of poses, and $q$ and $m$ the number of parameters for points and camera poses

# Improving the Accuracy of VO

- Other sensors can be used such as
  - IMU (called inertial VO)
  - Compass
  - GPS
  - Laser
- An IMU combined with a single camera allows the estimation of the absolute scale. Why?
- Make sure that you have many points (thoudsands) which cover the image uniformly

# VO Applications

VO has successfully been applied within various technological fields

- Space exploration:
  - Planetary lander furing descent phase
  - Spirit and Opportunity Mars-exploration rovers
    - Since 2004, used VO in addition to dead-reckoning for about 6 Km
    - Especially in preence of wheel slip

- MAV navigation
  - European project SFLY
  - Vision-based MAVs at the Robotics and Perception Group (see http://rpg.ifi.uzh.ch/research_mav.html )

- Underwater vehicles

- Automotive industry

**The sFly video can be seen at http://youtu.be/_-p08o_oTO4**

# VO Applications

World-first mouse scanner

Currently distributed by LG: *SmartScan LG LSM100*

**This video can be seen at**
**http://youtu.be/A4NGXFv27AE**

# Software and Dataset

SOFTWARE AND DATASETS

| Author | Description | Link |
|---|---|---|
| Willow Garage | OpenCV: A computer vision library maintained by Willow Garage. The library includes many of the feature detectors mentioned in this tutorial (e.g., Harris, KLT, SIFT, SURF, FAST, BRIEF, ORB). In addition, the library contains the basic motion-estimation algorithms as well as stereo-matching algorithms. | http://opencv.willowgarage.com |
| Willow Garage | ROS (Robot Operating System): A huge library and middleware maintained by Willow Garage for developing robot applications. Contains a visual-odometry package and many other computer-vision-related packages. | http://www.ros.org |
| Willow Garage | PCL (Point Cloud Library): A 3D-data-processing library maintained from Willow Garage, which includes useful algorithms to compute transformations between 3D-point clouds. | http://pointclouds.org |
| Henrik Stewenius et al. | 5-point algorithm: An implementation of the 5-point algorithm for computing the essential matrix. | http://www.vis.uky.edu/~stewe/FIVEPOINT/ |
| Changchang Wu et al. | SiftGPU: Real-time implementation of SIFT. | http://cs.unc.edu/~ccwu/siftgpu |
| Nico Cornelis et al. | GPUSurf: Real-time implementation of SURF. | http://homes.esat.kuleuven.be/~ncorneli/gpusurf |
| Christopfer Zach | GPU-KLT: Real-time implementation of the KLT tracker. | http://www.inf.ethz.ch/personal/chzach/opensource.html |
| Edward Rosten | Original implementation of the FAST detector. | http://www.edwardrosten.com/work/fast.html |

# Software and Dataset

| | | |
|---|---|---|
| Michael Calonder | Original implementation of the BRIEF descriptor. | http://cvlab.epfl.ch/software/brief/ |
| Leutenegger et al. | BRISK feature detector. | http://www.asl.ethz.ch/people/lestefan/personal/BRISK |
| Jean-Yves Bouguet | Camera Calibration Toolbox for Matlab. | http://www.vision.caltech.edu/bouguetj/calib_doc |
| Davide Scaramuzza | OCamCalib: Omnidirectional Camera Calibration Toolbox for MATLAB. | https://sites.google.com/site/scarabotix/ocamcalib-toolbox |
| Christopher Mei | Omnidirectional Camera Calibration Toolbox for MATLAB | http://homepages.laas.fr/~cmei/index.php/Toolbox |
| Mark Cummins | FAB-MAP: Visual-word-based loop detection. | http://www.robots.ox.ac.uk/~mjc/Software.htm |
| Friedrich Fraundorfer | Vocsearch: Visual-word-based place recognition and image search. | http://www.inf.ethz.ch/personal/fraundof/page2.html |
| Manolis Lourakis | SBA: Sparse Bundle Adjustment | http://www.ics.forth.gr/~lourakis/sba |
| Christopher Zach | SSBA: Simple Sparse Bundle Adjustment | http://www.inf.ethz.ch/personal/chzach/opensource.html |
| Rainer Kuemmerle et al. | G2O: Library for graph-based nonlinear function optimization. Contains several variants of SLAM and bundle adjustment. | http://openslam.org/g2o |
| RAWSEEDS EU Project | RAWSEEDS: Collection of datasets with different sensors (lidars, cameras, IMUs, etc.) with ground truth. | http://www.rawseeds.org |
| SFLY EU Project | SFLY-MAV dataset: Camera-IMU dataset captured from an aerial vehicle with Vicon data for ground truth. | http://www.sfly.org |
| Davide Scaramuzza | ETH OMNI-VO: An omnidirectional-image dataset captured from the roof of a car for several kilometers in a urban environment. MATLAB code for visual odometry is provided. | http://sites.google.com/site/scarabotix |