Robot Vision

TTK4255

Lecture 11 – Visual Odometry

Annette Stahl

(Annette.Stahl@ntnu.no)

Department of Engineering Cybernetics – ITK

NTNU, Trondheim

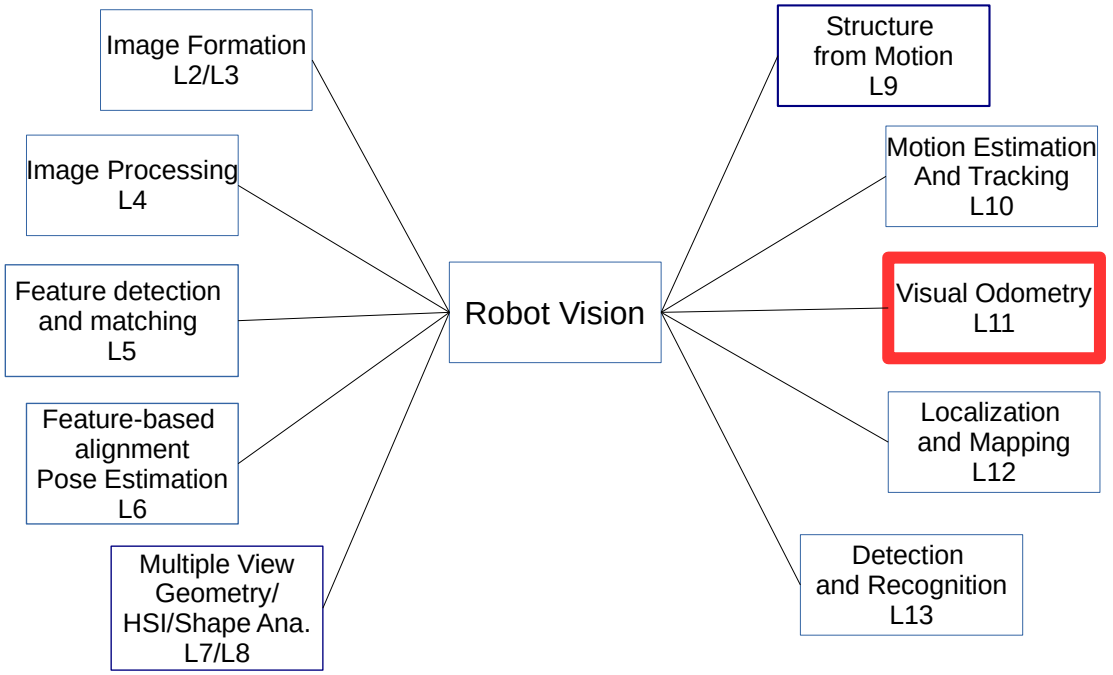
Spring Semester

16. March 2020

Lecture 11 – Visual Odometry

Annette Stahl (Annette.Stahl@ntnu.no)

Simen Haugo (Simen.Haugo@ntnu.no)



Outline of the fourth lecture:

- Visual Odometry definition + brief history

Visual Odometry Tutorial

This lecture is mainly based on

- Scaramuzza, D., Fraundorfer, F.,

  Visual Odometry: Part I - The First 30 Years and Fundamentals,

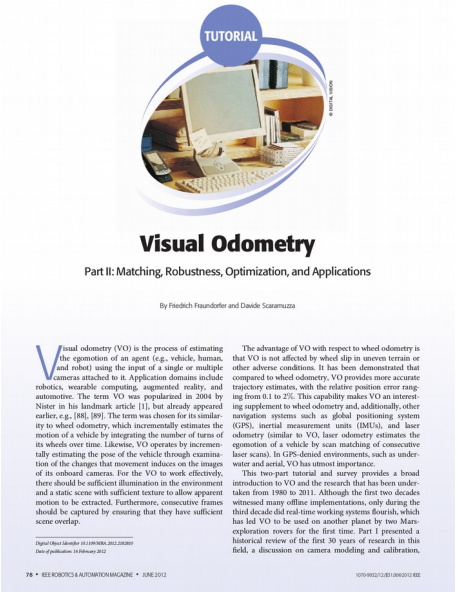
  IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

and partly on

- Fraundorfer, F., Scaramuzza, D.,

  Visual Odometry: Part II - Matching, Robustness, and Applications,

  IEEE Robotics and Automation Magazine, Volume 19, issue 1, 2012.



Visual Odometry

Visual Odometry (VO) is directly linked to **Structure from motion (SfM)**:

Basic Principle behind VO: Simple iteration of two-view SfM (cf. Lecture 09)

Visual Odometry consists in

- Estimating the motion (egomotion) of a robot/vehicle (agent) by using **only visual input** of a **single or multiple cameras** attached to it.

  → **incrementally** (as a new frame arrives) estimating the pose of the robot/vehicle (6DoF) by examining changes that motion induces on the images of its onboard cameras **in real time**

  → guarantees **local consistency**

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

Odometry

What is Odometry?

- Route Measure
- Estimating change in position over time

Wheeled odometry

Incrementally estimates the motion of a vehicle by integrating the number of turns of its wheels over time

→ wheeled odometry is affected by wheel slip

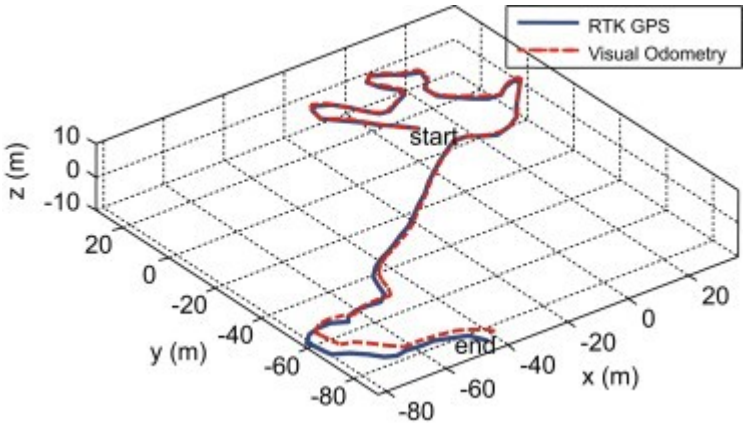
Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

Visual Odometry

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%)

→ this capabilities make VO an interesting supplement
to other navigation systems



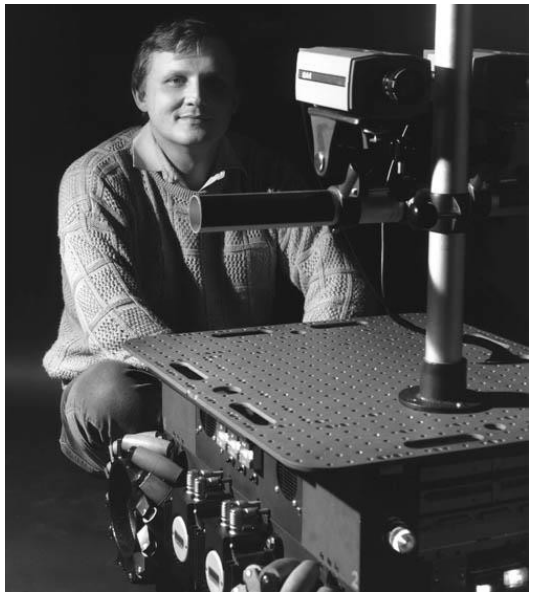
VO can be used as a **complement** to
- wheel odometry
- GPS
- inertial measurement units (IMUs)
- laser odometry (estimates the egomotion of a vehicle by scan-matching of consecutive laser scans)

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

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

Visual Odometry - VO

1980: First known VO real-time implementation on a robot by Hans Moraveck PhD-thesis (NASA/JPL) for Mars rovers using one sliding camera (sliding stereo).



“slider stereo”: a single camera sliding on a rail

Robot moved in a stop-and-go fashion digitizing and analyzing images at every location.

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

Visual Odometry

First introduced 2004 by

Nistér, D., Naroditsky, O., Bergen, J.,

“Visual odometry,”

IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

→ Showed successful results on different vehicles (on-and off-road) using single or stereo vision

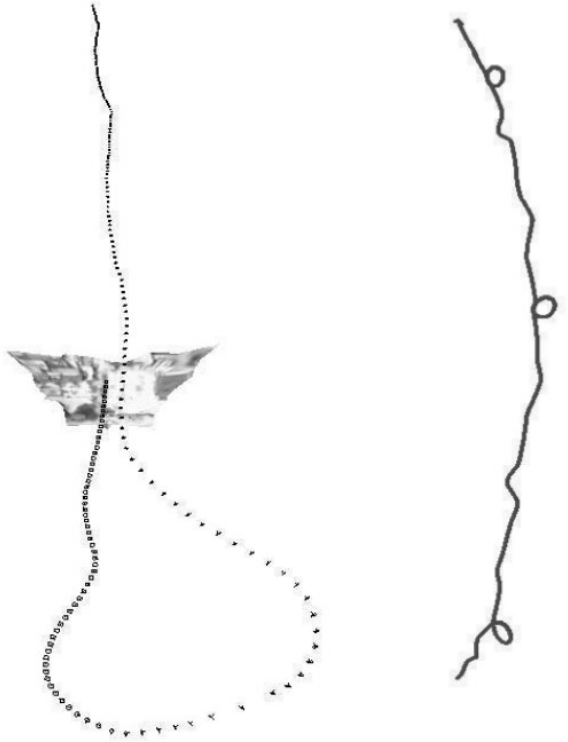


Figure 1: Left: Results with a single camera mounted obliquely on an aerial platform. The aeroplane flies at a low altitude and turns to make another sweep. The visual odometry shown was estimated in real-time with low delay. The result is based solely on visual input and no prior knowledge of the scene nor the motion is used. A textured triangulation in the frustum of the most recent camera position is also shown. Right: Results from a stereo pair mounted on a ground vehicle. The vehicle path is is over 600 meters long and includes three tight loops.

Visual Odometry

Assumptions:

- Sufficient illumination in the environment
- a static scene with
- Enough texture to allow apparent motion to be extracted
- Sufficient scene overlap between consecutive frames

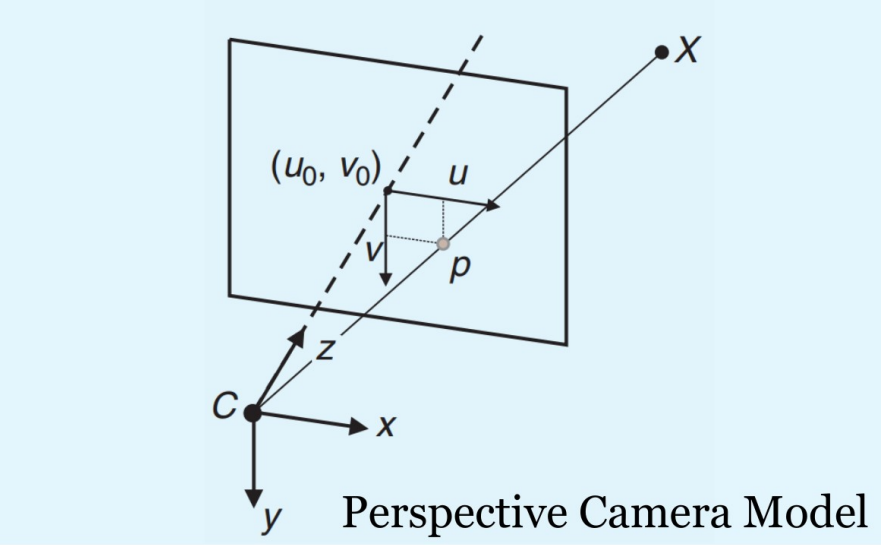
Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

Mono and Stereo Visual Odometry

Monocular Visual Odometry

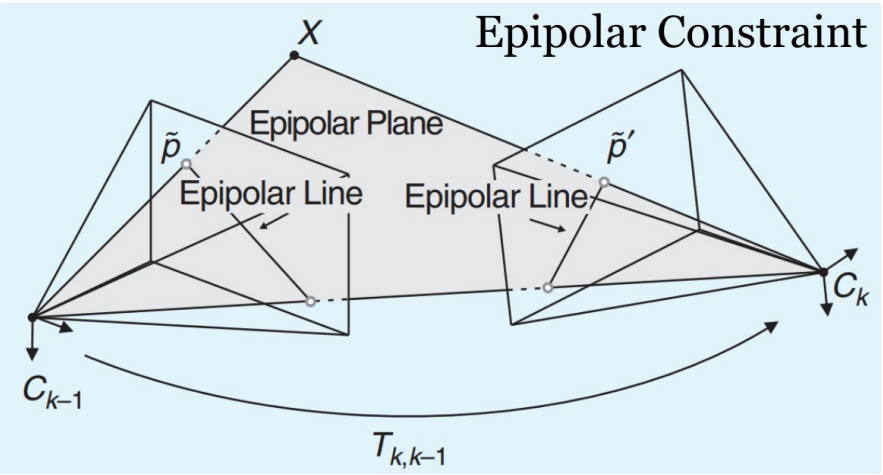
- Rel. motion and 3D str. must be computed
- A single camera = angle sensor
- Motion scale is unobservable

  (it must be synthesized)
- Best used in **hybrid** methods



Stereo Visual Odometry

- Solves the scale problem
- Feature depth between images
- Degenerates to the monocular case when the distance to the scene is much larger than the stereo baseline



Scaramuzzaand Fraundorfer, 2011

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

Visual Odometry Problem

- Images are taking at discrete time instants k
  - Mono case: $I_{0:n} = \{I_0, ..., I_n\}$
  - Stereo case: $I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$ $I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$
- Two camera positions are related by the rigid body transformation

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

- Set of all subsequent motions

$$T_{1:n} = \{T_{1,0}, ..., T_{n,n-1}\} = \{T_1, ..., T_n\}$$

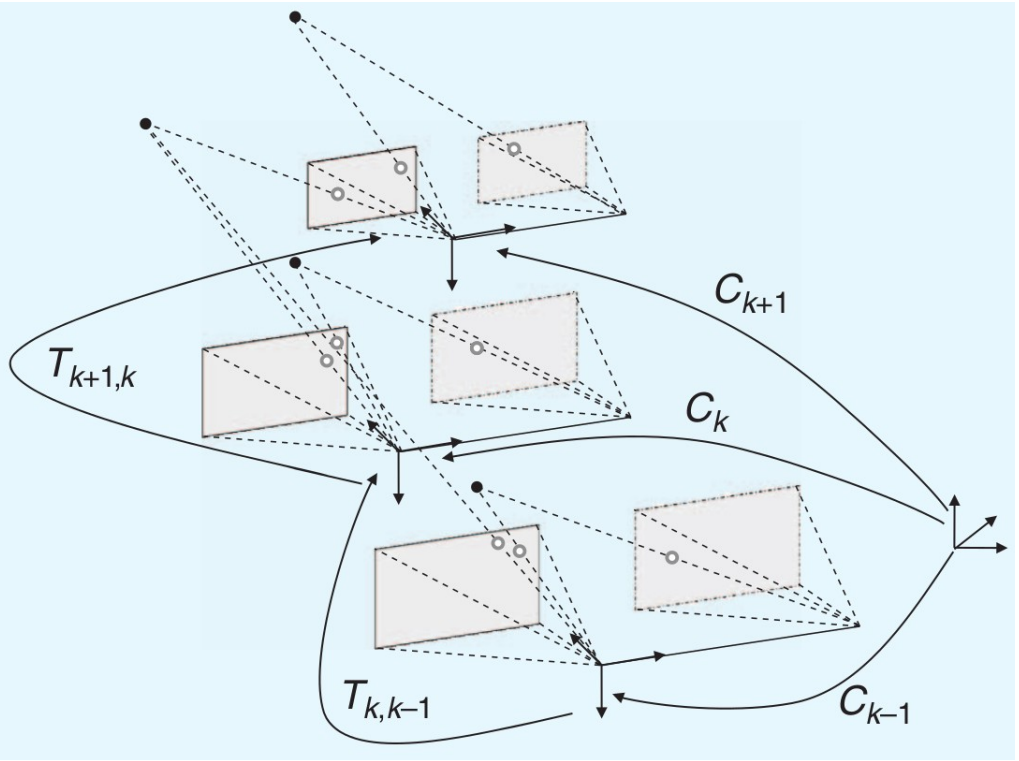
- Set of camera poses wrt the initial frame 0

$$C_{0:n} = \{C_0, ..., C_n\}$$

- Compute current pose n by concatenating all the transformations $(k=1,\ldots,n)$

$$C_n = C_{n-1}T_n \text{ with } C_0$$

the initial camera pose



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

Visual Odometry Problem

Main task in VO:

- Compute relative transformations T_k from the images I_k and I_{k-1} and then
- Concatenate the transformation to recover the full trajectory of the camera

→ recover path incrementally pose after pose

→ recover the full trajectory $C_{0:n}$ of the camera

Windowed-bundle adjustment (to reduce drift)**:**

Iteratively refine over the last m poses by minimizing the sum of squared reprojection errors of the reconstructed 3D points (3D map) over the last m images

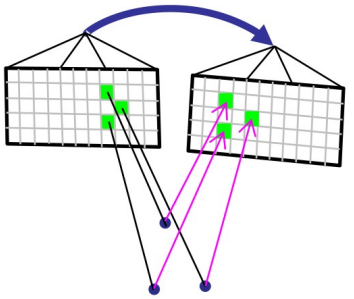
3D points are obtained by triangulation of the image points

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

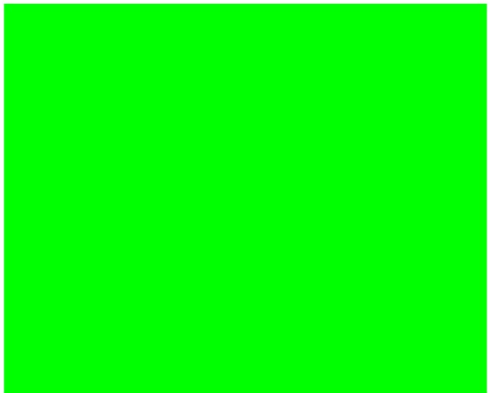
Direct Image Alignment

Minimize per pixel-intensity difference

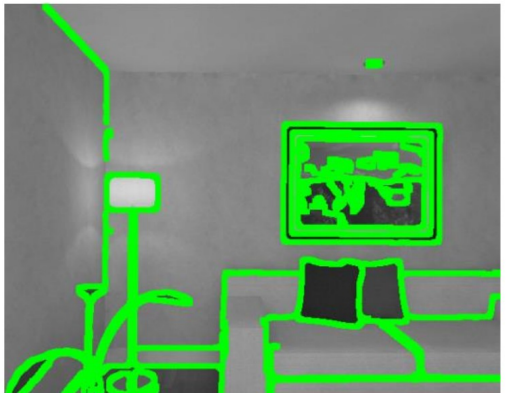
$$T_{k,k-1} = \arg\min_T \sum_i \|I_k(\boldsymbol{u}'_i) - I_{k-1}(\boldsymbol{u}_i)\|^2_\sigma$$



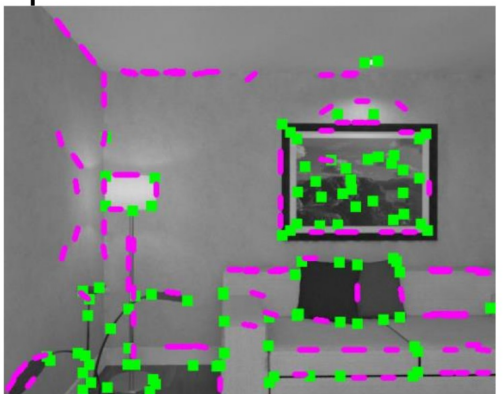
Dense

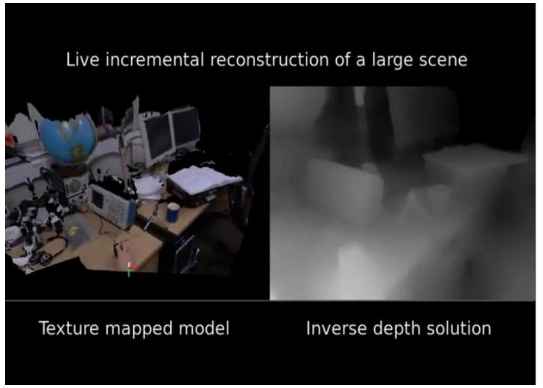


Semi-dense

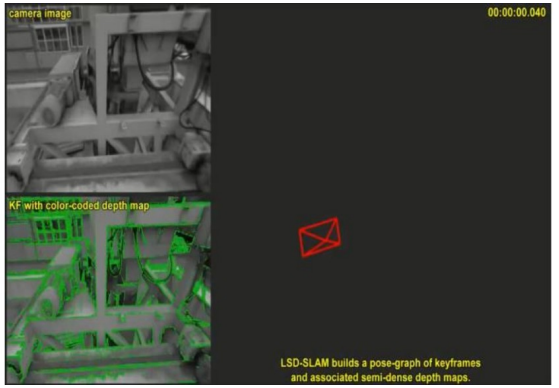


Sparse





DTAM [Newcombe et al. '11]
300'000+ pixels



LSD [Engel et al. 2014]
~10'000 pixels

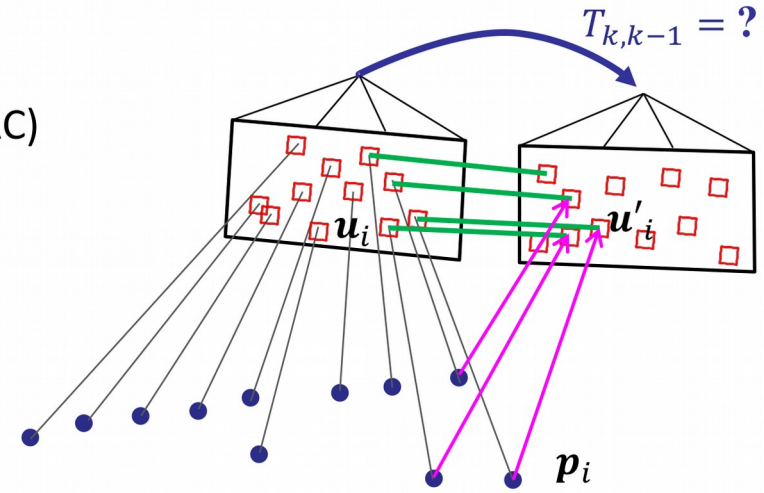
SVO [Forster et al. 2014]
100-200 features x 4x4 patch
~ 2,000 pixels

Irani & Anandan, "All About Direct Methods," Vision Algorithms: Theory and Practice, Springer, 2000

Feature based and Direct Methods

Feature-based methods

1. Extract & match features (+RANSAC)
2. Minimize **Reprojection error** minimization



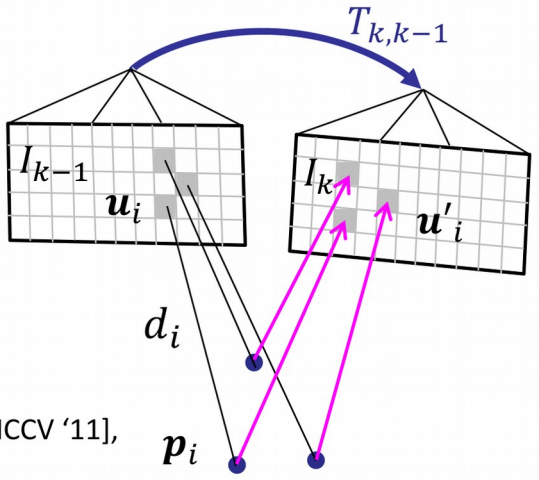
$$T_{k,k-1} = \arg\min_T \sum_i \|\boldsymbol{u'}_i - \pi(\boldsymbol{p}_i)\|^2_\Sigma$$

Direct methods

1. Minimize **photometric error**

$$T_{k,k-1} = \arg\min_T \sum_i \|I_k(\boldsymbol{u'}_i) - I_{k-1}(\boldsymbol{u}_i)\|^2_\sigma$$

where $\boldsymbol{u'}_i = \pi\left(T \cdot (\pi^{-1}(\boldsymbol{u}_i) \cdot d)\right)$



[Jin,Favaro,Soatto'03] [Silveira, Malis, Rives, TRO'08], [Newcombe et al., ICCV '11], [Engel et al., ECCV'14], [Forster et al., ICRA'14]

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

Feature based and Direct Methods

Feature based

- ✓ Large frame-to-frame motions
- ✓ Accuracy: Efficient optimization of structure and motion (Bundle Adjustment)
- × Slow due to costly feature extraction and matching
- × Matching Outliers (RANSAC)

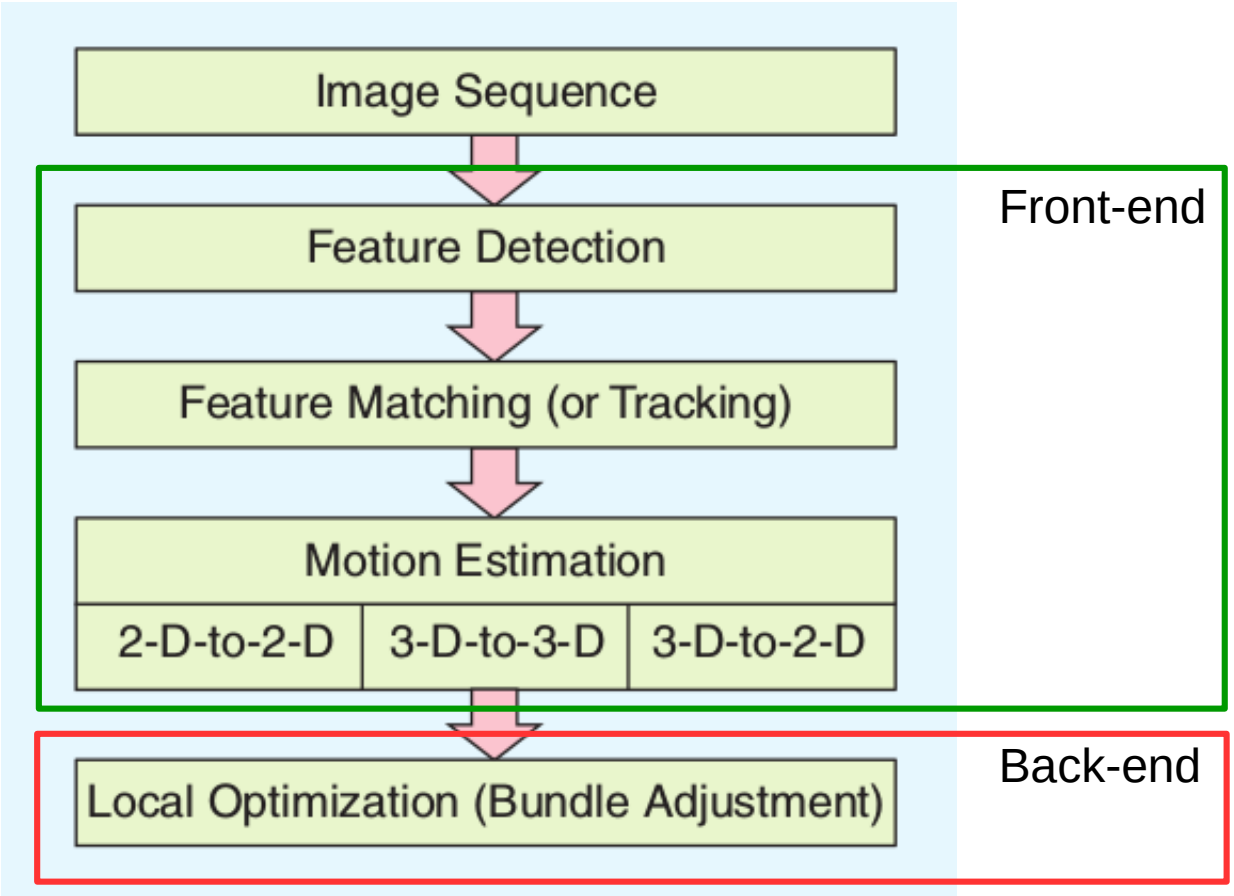
Direct

- ✓ All information in the image can be exploited (precision, robustness)
- ✓ Increasing camera frame-rate reduces computational cost per frame
- × Limited frame-to-frame motion
- × Joint optimization of dense structure and motion too expensive

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

VO Work Flow

VO computes pose after pose incrementally resulting in the camera path



The **front-end** is responsible for
- Feature extraction
- Matching
- Outlier removal
- Loop closure detection

The **back-end** is responsible for
- pose and structure optimization (e.g., iSAM, g2o, Google Ceres)

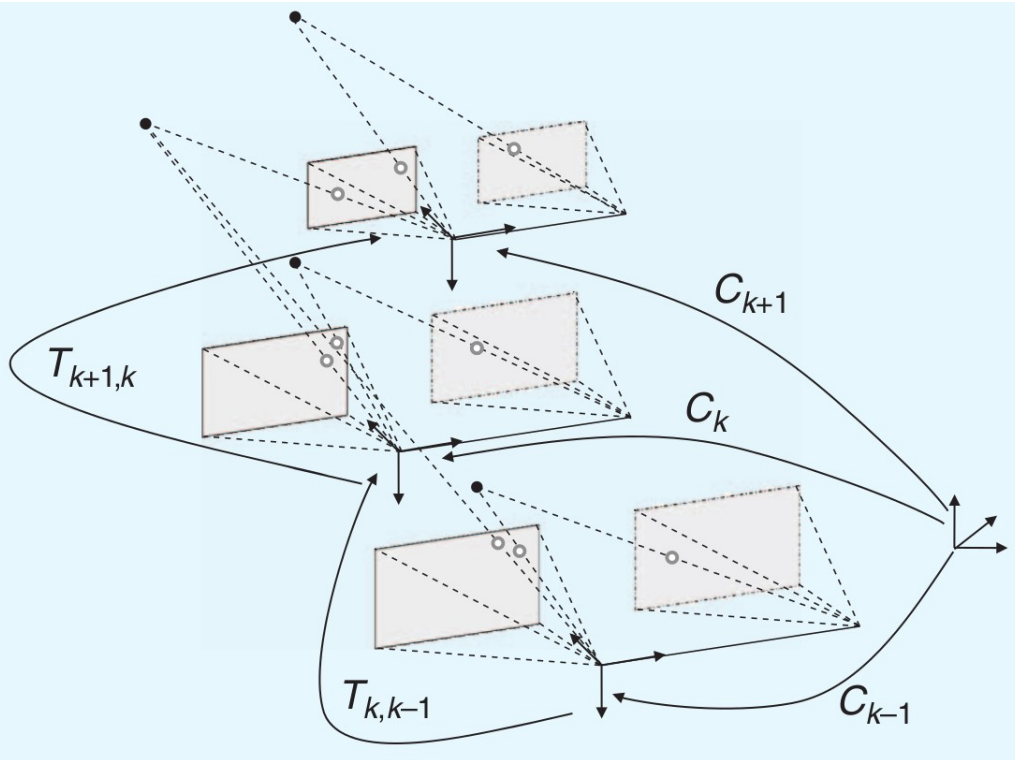
Scaramuzza et al., Visual Odometry: Part I, IEEE Robotics and Automation Magazine, 18 (4), 2011

Motion Estimation

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

The camera motion between the current image and the previous image is computed.

How can the transformation T_k between the two images I_k and I_{k-1} be computed from two sets of corresponding features f_{k-1} and f_k at time instances k-1 and k respectively.



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

Motion Estimation – Feature Corr.

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

3-D to 3-D: f_{k-1} and f_k are specified in 3-D coordinates

3-D points are triangulated at each time instant (utilize f.e. stereo camera system).

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

Monocular case: 3-D structure needs to be triangulated from two adjacent camera views (e.g., I_{k-2} I_{k-1}) and then matched to 2-D image features in a third view (e.g., I_k).
→ matches over at least three views are necessary.

Note: features can be points or lines. In general point features are used in VO

Type of correspondences	Monocular	Stereo
2D to 2D	x	x
3D to 3D		x
3D to 2D	x	x

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

2D to 2D - Motion from Image Feature Corr.

Estimate Essential Matrix
Geometric relations between two images I_k and I_{k-1} of a calibrated camera
Contains the camera motion parameters up to scale for translation

$$E_k \simeq [T_k]_\times R_k = \hat{T}_k R_k \qquad \text{(notation from L07)}$$
$$E_k \simeq \hat{t}_k R_k \qquad \text{(notation from DS_VO1)}$$

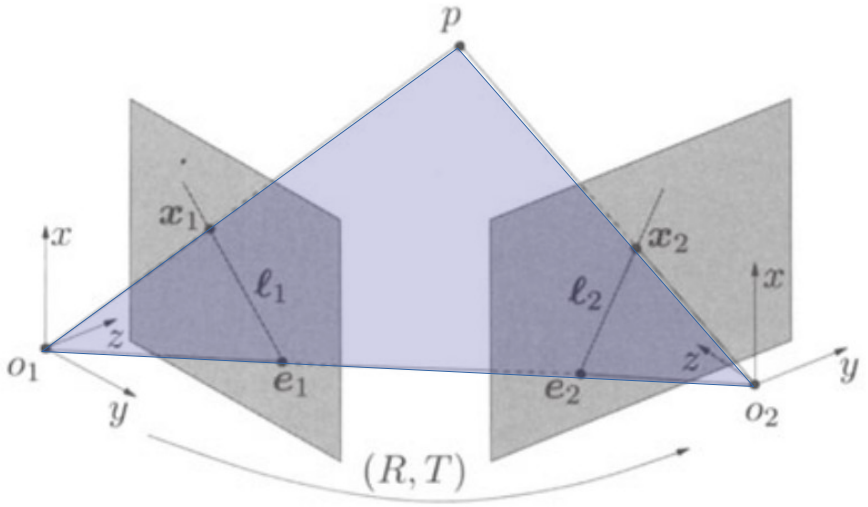
Algorithm 1. VO from 2-D-to-2-D correspondences.

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).

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

Epipolar Constraint

$$X_r = RX_l + T$$
$$\lambda_r \mathbf{x}_r = R\lambda_l \mathbf{x}_l + T$$
$$\lambda_r [T]_\times \mathbf{x}_r = [T]_\times R\lambda_l \mathbf{x}_l + [T]_\times T$$
$$\mathbf{x}_r^\top \lambda_r [T]_\times \mathbf{x}_r = \mathbf{x}_r^\top [T]_\times R\lambda_l \mathbf{x}_l$$
$$0 = \mathbf{x}_r^\top [T]_\times R\mathbf{x}_l$$
$$0 = \mathbf{x}_r^\top E\mathbf{x}_l$$



- The **Epipolar constraint** holds for every pair of corresponding points
- Two images of the same point $\mathbf{p} = \mathbf{x}$ from two camera positions with relative pose $(R,\ T)$, ($R \in SO(3)$ relative orientation, $T \in \mathbb{R}^3$ relative position) satisfy the epipolar constraint equation.
- Where $E = [T]_\times R \in \mathbb{R}^{3\times 3}$ is the **essential matrix**
- The epipolar constraint gives the relative pose between two cameras
- The Essential Matrix can be decomposed into R and T recalling that four distinct solutions for R and T are possible. [LonguetHiggins1981]

Recall L07: Decompose E

Singular Value Decomposition:

$$E = USV^\top$$

Enforce rank-2: set smallest singular value of S to 0:

$$S = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The **four** solutions are:

$$\hat{T} = U \begin{bmatrix} 0 & \pm 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} SU^\top$$

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

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

Select solution which is in front of the camera.

2D to 2D - Motion from Image Feature Corr.

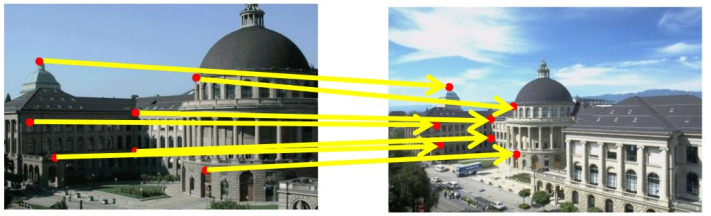
Motion from Image Feature Correspondences

- Both feature points f_{k-1} and f_k are specified in 2D
- The minimal-case solution involves 5-point correspondences
- Use estimate R, t as initial values for the non-linear optimization
- The solution is found by minimizing the image reprojection error

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

- Popular algorithms: 8- and 5-point algorithms
  [Hartley'97, Nister'04]

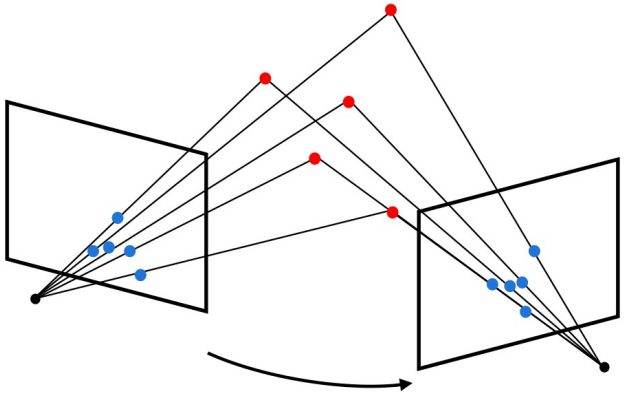
D. Nister, "An efficient solution to the five-point relative pose problem," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pp. 756-770, June 2004.



I_{k-1} I_k

Algorithm 1. VO from 2-D-to-2-D correspondences.

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).



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

3D to 2D – Motion from 3D Structure and Image Feature Correspondences

Motion estimation from 3D to 2D correspondences is more accurate than from 3D to 3D correspondences because it minimizes the image reprojection error instead of the 3D to 3D feature position error.
The transformation

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

Is computed from the 3D to 2D correspondences

Algorithm 3. VO from 3-D-to-2-D Correspondences.

1) Do only once:
1.1) Capture two frames I_{k-2}, I_{k-1}
1.2) Extract and match features between them
1.3) Triangulate features from I_{k-2}, I_{k-1}
2) Do at each iteration:
2.1) Capture new frame I_k
2.2) Extract features and match with previous frame I_{k-1}
2.3) Compute camera pose (PnP) from 3-D-to-2-D matches
2.4) Triangulate all new feature matches between I_k and I_{k-1}
2.5) Iterate from 2.1).

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

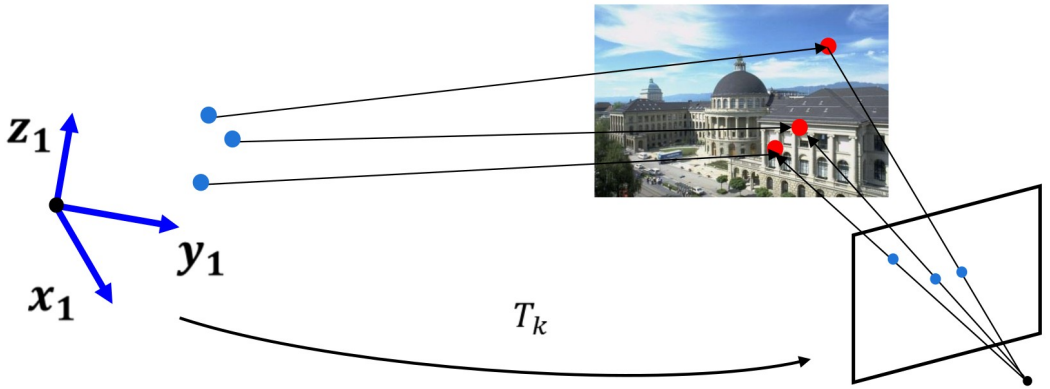
3D to 2D – Motion from 3D Structure and Image Feature Correspondences

- 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)
- For n>=6 DLT (linear system → SVD)
- The minimal-case solution involves 3 correspondences (+1 for disambiguating the 4 solutions)
- The solution is found by minimizing the image reprojection error

- Popular algorithms: P3P [Gao'03, Kneip'11]

Algorithm 3. VO from 3-D-to-2-D Correspondences.

1) Do only once:
1.1) Capture two frames I_{k-2}, I_{k-1}
1.2) Extract and match features between them
1.3) Triangulate features from I_{k-2}, I_{k-1}
2) Do at each iteration:
2.1) Capture new frame I_k
2.2) Extract features and match with previous frame I_{k-1}
2.3) Compute camera pose (PnP) from 3-D-to-2-D matches
2.4) Triangulate all new feature matches between I_k and I_{k-1}
2.5) Iterate from 2.1).

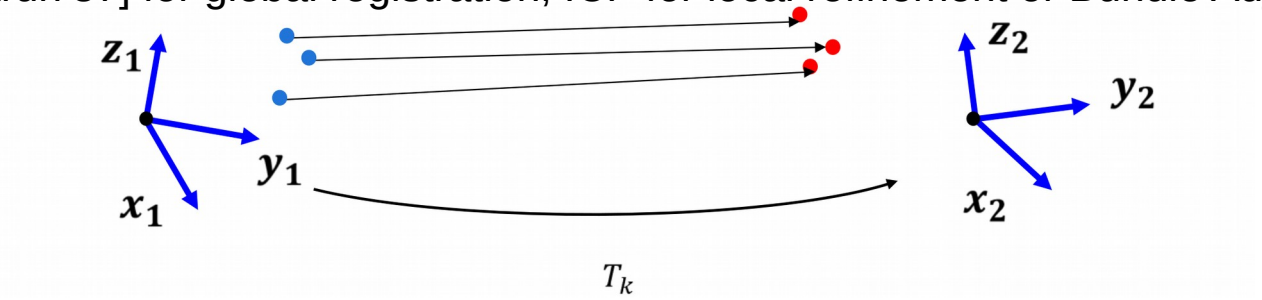


Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

3D to 3D – Motion from 3D Structure Corr.

- Motion from 3D-3D point correspondences (point cloud registration)
- Both f_{k-1} and f_k are specified in 3D.
- Triangulate 3D points (e.g. use a stereo camera)
- The minimal-case solution involves 3 **non-collinear correspondences**
- The solution is found by minimizing the 3D-3D Euclidean distance between the two 3D feature sets

- Popular algorithm: [Arun'87] for global registration, ICP for local refinement or Bundle Adjustment



Algorithm 2. VO from 3-D-to-3-D correspondences.

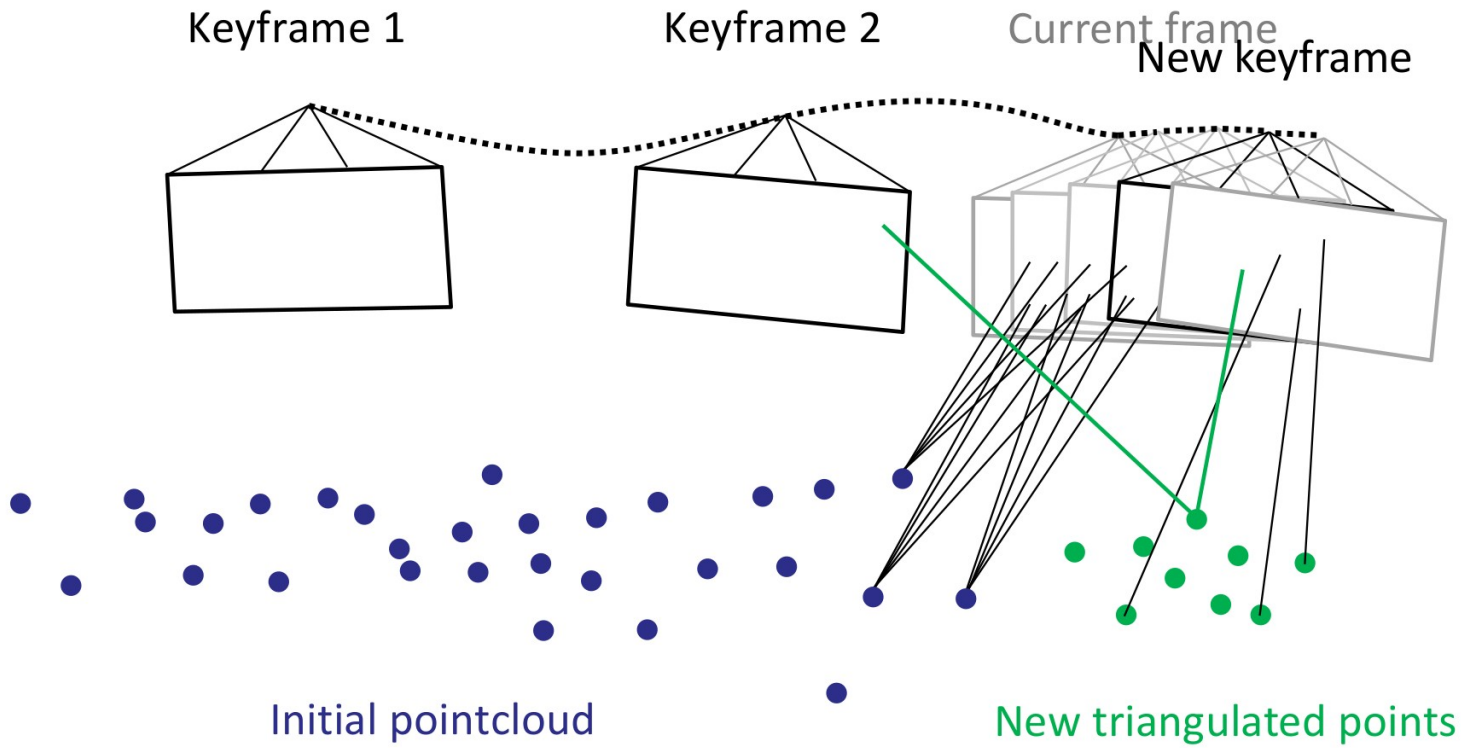
1) Capture two stereo image pairs $I_{l,k-1}, I_{r,k-1}$ and $I_{l,k}, I_{r,k}$
2) Extract and match features between $I_{l,k-1}$ and $I_{l,k}$
3) Triangulate matched features for each stereo pair
4) Compute T_k from 3-D features X_{k-1} and X_k
5) Concatenate transformation by computing $C_k = C_{k-1}T_k$
6) Repeat from 1).

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

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

Keyframe based Monocular VO

Initialization
- Initialize structure and motion from 2 views: e.g., 5- or 8-point RANSAC
- Refine structure and motion (Bundle Adjustment)
- How far should the two frames (i.e., keyframes) be?



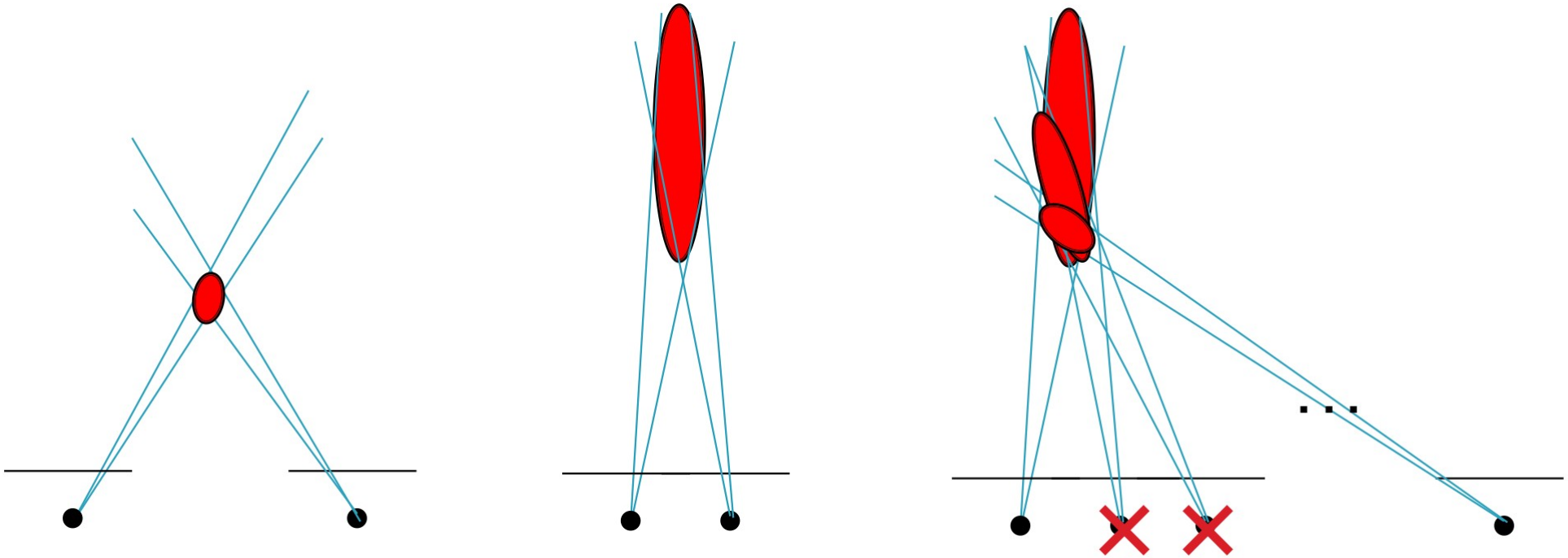
Typical visual odometry pipeline used in many algorithms
[Nister'04, PTAM'07, LIBVISO'08, LSD-SLAM'14, SVO'14, ORB-SLAM'15]

Keyframe Selection

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

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

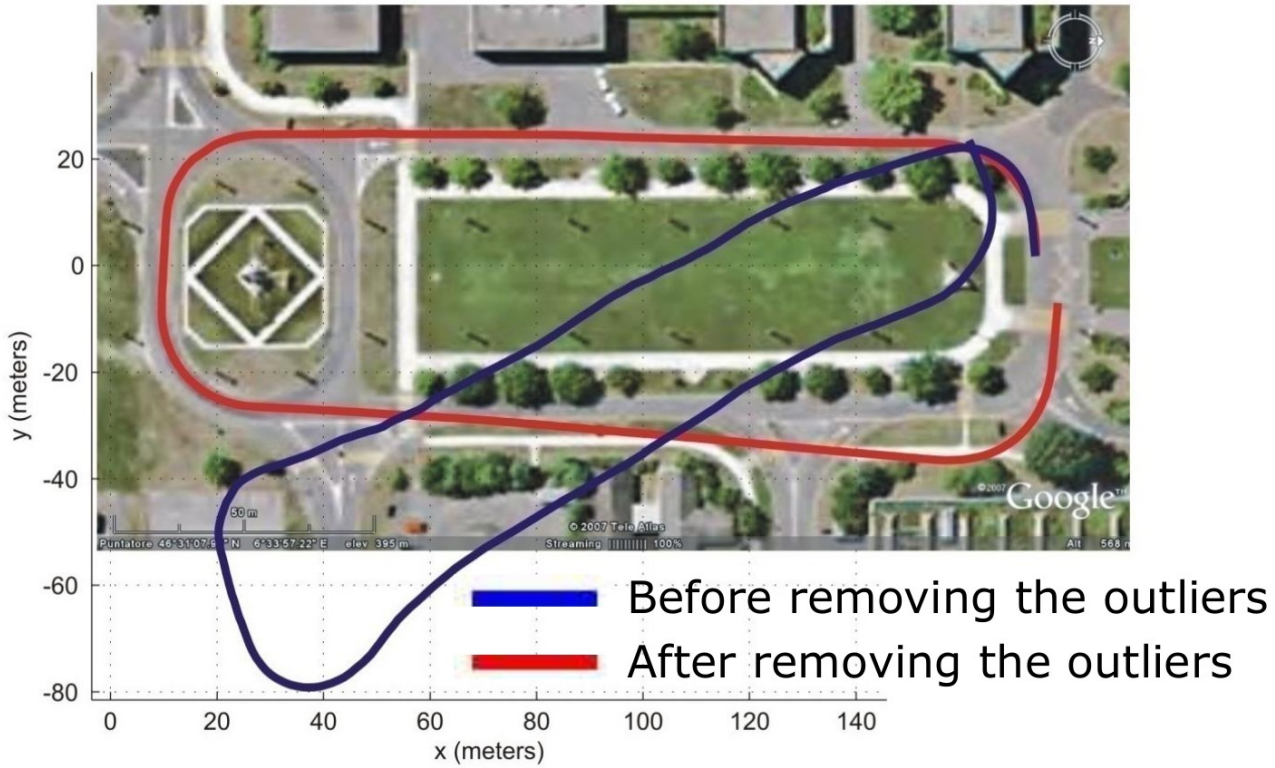
Rule of the thumb: add a keyframe when keyframe distance > threshold (~10-20 %)



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

Robust Estimation

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



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

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

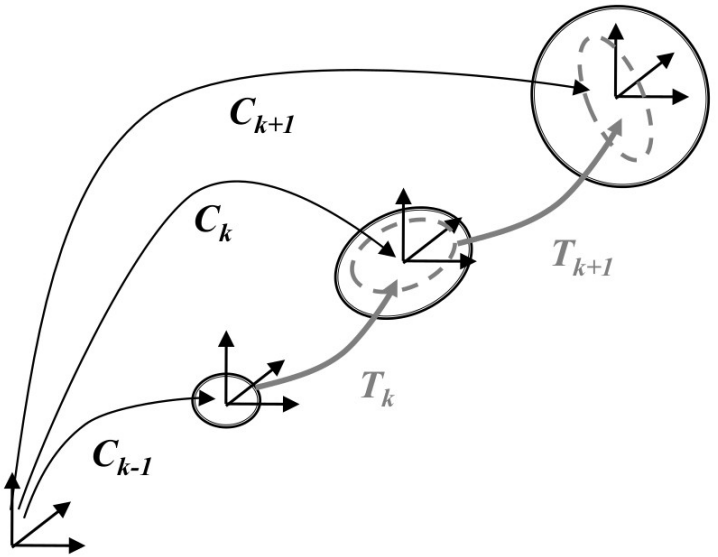
Error Propagation

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 Σ_k is

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

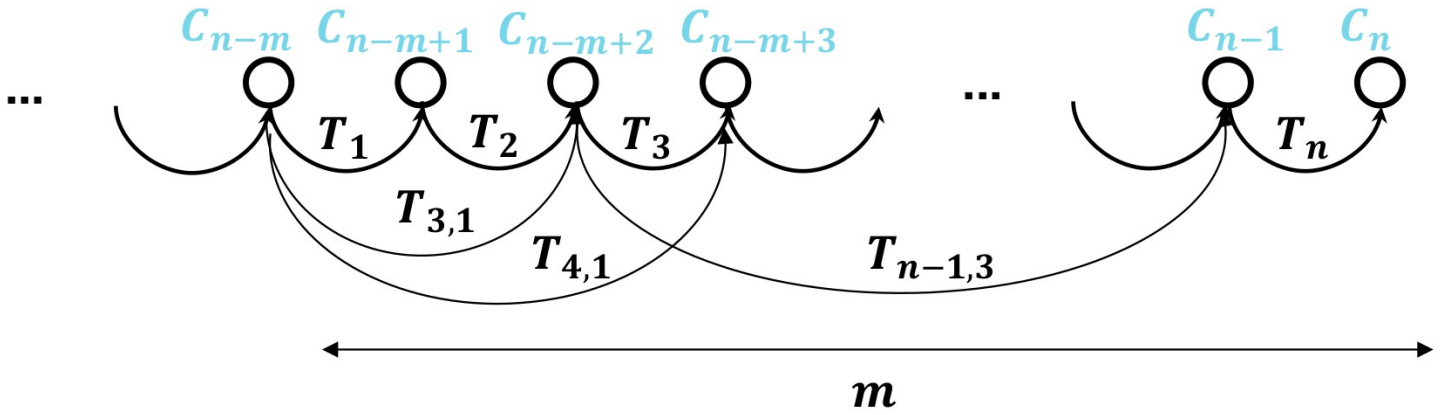


The camera-pose uncertainty is always increasing when concatenating transformations. Thus, it is important to keep the uncertainties of the individual transformations small

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

Pose-graph Optimization

So far we assumed that the transformations are between consecutive frames



- Transformations can be computed also between non-adjacent frames $T_{e_{ij}}$ (e.g., when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following:

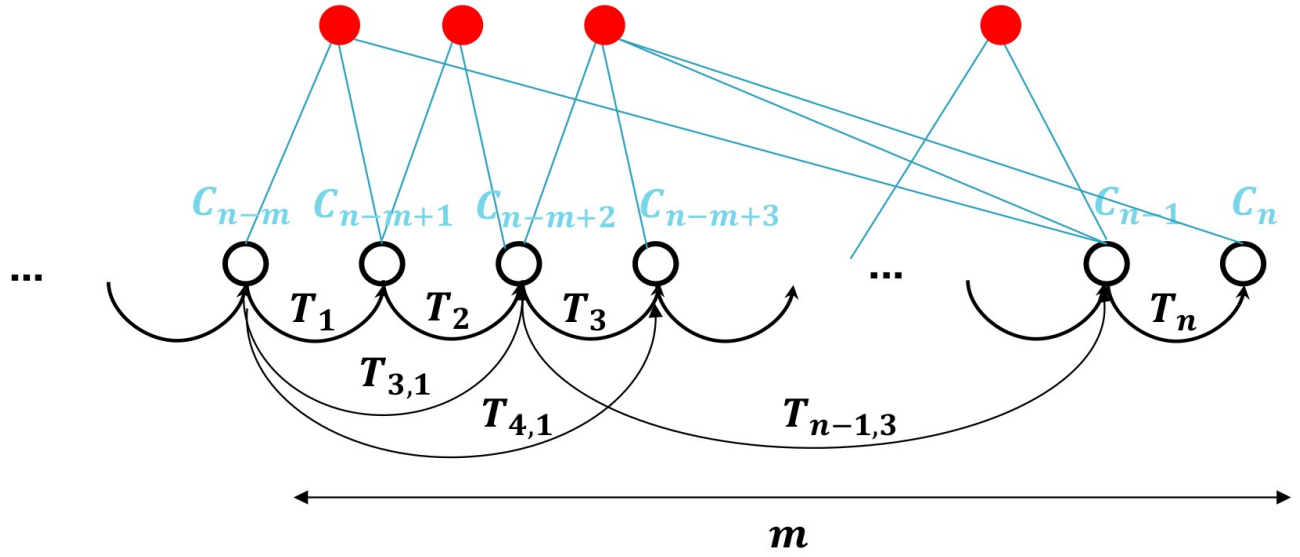
$$\sum_{e_{ij}} \left\| C_i - T_{e_{ij}} C_j \right\|^2$$

- For efficiency, only the last m keyframes are used
- Gauss-Newton or Levenberg-Marquadt are typically used to minimize it. For large graphs, efficient open-source tools: g2o, GTSAM, Google Ceres

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

Bundle Adjustment

- Similar to pose-graph 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 to the minimum
- Gauss-Newton or Levenberg-Marquadt can be used. For large graphs, efficient open-source software exists: GTSAM, g2o, Google Ceres can be used.

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

BA vs. Pose-graph Optimization

BA is more precise than pose-graph optimization because it adds additional constraints (landmark constraints)

But more costly: with M and N being the number of points and cameras poses and q and l the number of parameters for points and camera poses.

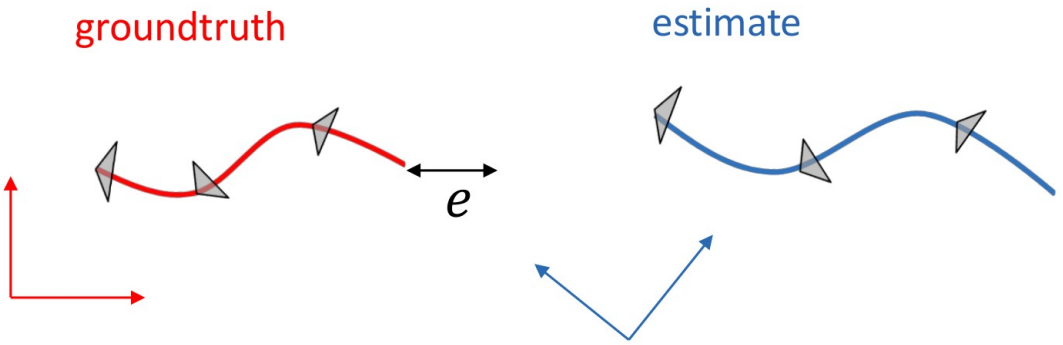
Workarounds:

- A small window size limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible.
- It is possible to reduce the computational complexity by just optimizing over the camera parameters and keeping the 3-D landmarks fixed, e.g., (motion-only BA)

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

Accuracy Evaluation

- Evaluation is not trivial
- Not obvious what to measure
  - Different reference frame
  - Different scale
  - Different times stamps



Align the first poses and measure the end-pose error?

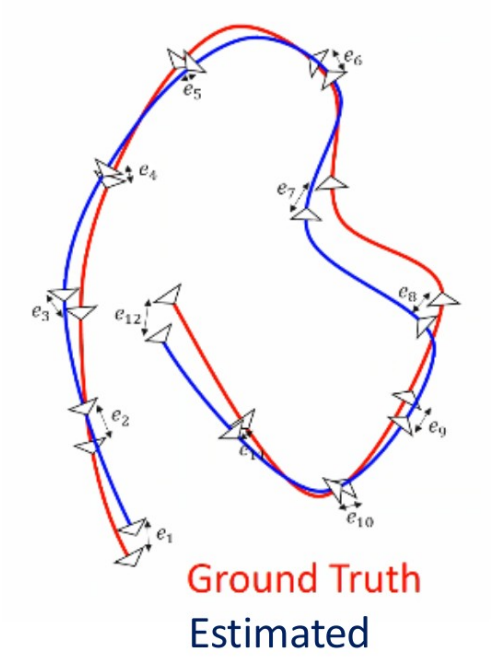
- How many poses should be used for the alignment?
- Not robust:
  - Most VIOs are non-deterministic (e.g., RANSAC, multithreading)
    → every time you run your VIO on the same dataset, you get different results
- Not meaningful:
  - too sensitive to the trajectory shape
  - does not capture the error statistics

→ **Utilize other Evaluation Measures**

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

Absolute Trajectory Error (ATE)

- RMSE of the aligned estimate and the groundtruth.
- Advantage: Estimated Single number metric
- Disadvantage: Many parameters to specify



Step 1: Align the trajectory

$$\underset{R,T,s}{\operatorname{argmin}} \sum_{i=0}^{N} \|\hat{t}_i - sRt_i - T\|^2$$

groundtruth positions

estimated positions

Alignment parameters

Step 2: Root mean squared errors between the aligned estimate and the groundtruth.

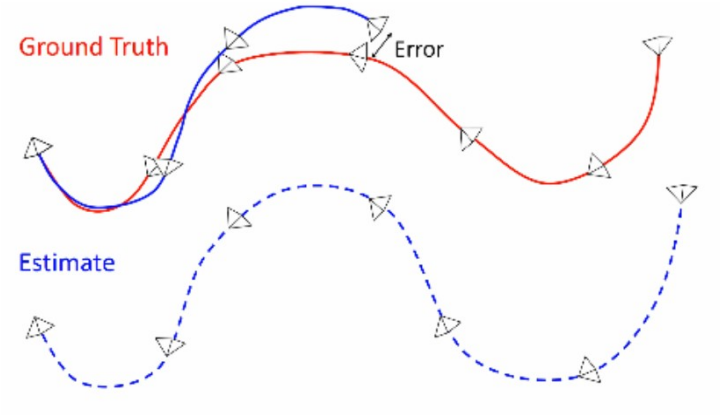
$$\sqrt{\frac{\sum_{i=1}^{N} \|\hat{t}_i - sRt_i - T\|^2}{N}}$$

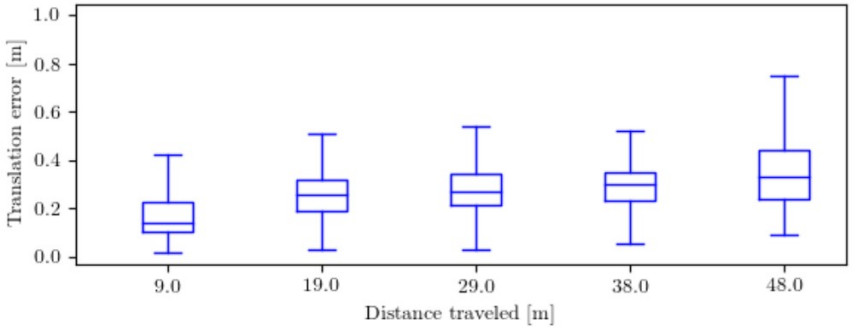
Sturm et al., "A benchmark for the evaluation of RGB-D SLAM systems." IROS 2012.
Zhang et al., "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." IROS'18.

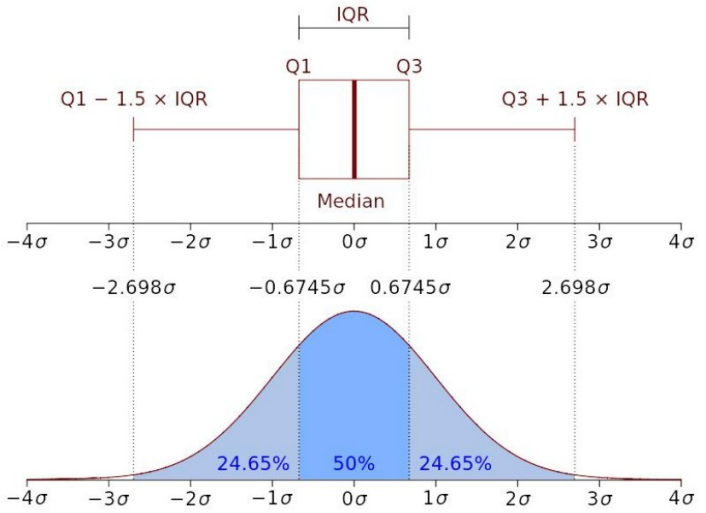
Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

Relative Trajectory Error (RTE)

- Relative Error (Odometry Error) - Statistics of sub-trajectories of specified lengths:
- Calculate errors for all the sub-trajectories of certain lengths.







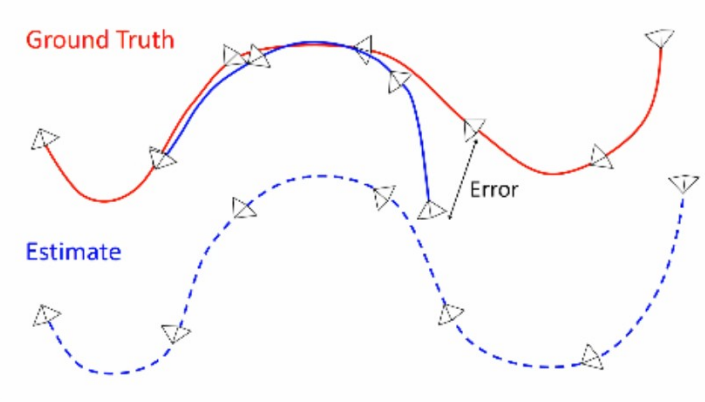
Advantage: Informative statistics
Disadvantage: Complicated to compute and rank

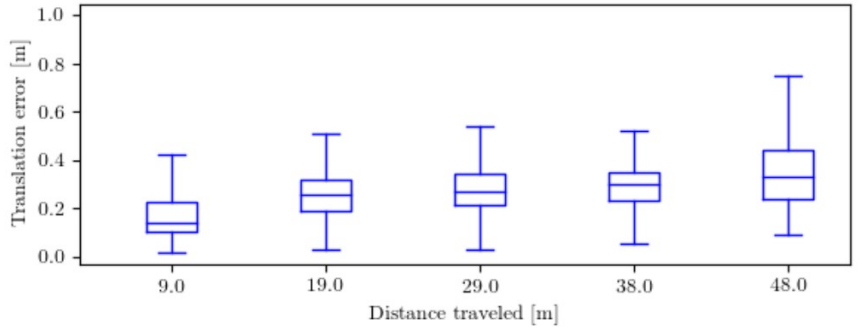
Geiger et al. "Are we ready for autonomous driving? the KITTI vision benchmark suite." CVPR 2012.
Zhang et al., "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." IROS'18.

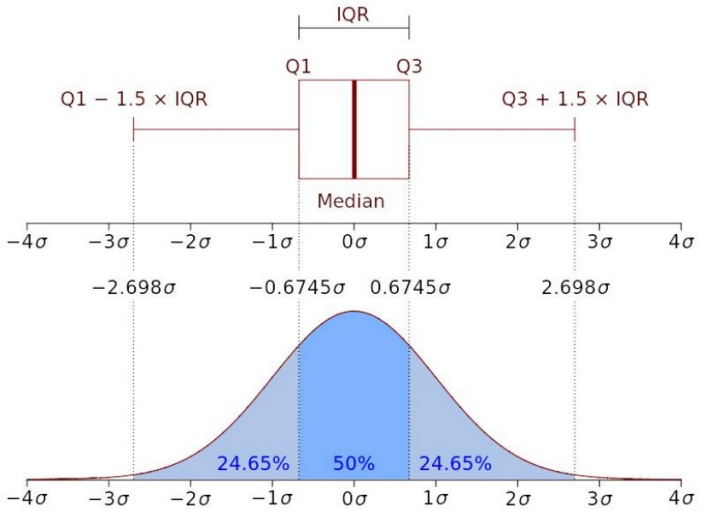
Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

Relative Trajectory Error (RTE)

- Relative Error (Odometry Error) - Statistics of sub-trajectories of specified lengths:
- Calculate errors for all the sub-trajectories of certain lengths.







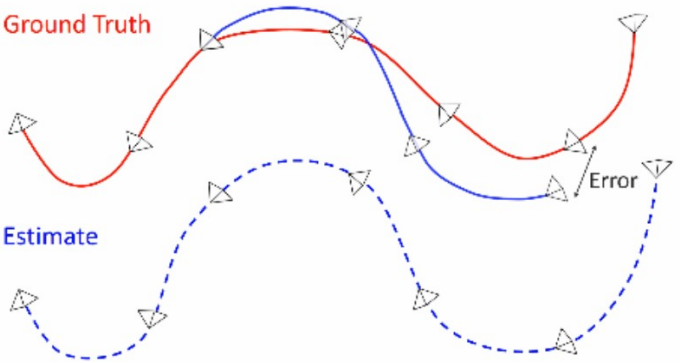
Advantage: Informative statistics
Disadvantage: Complicated to compute and rank

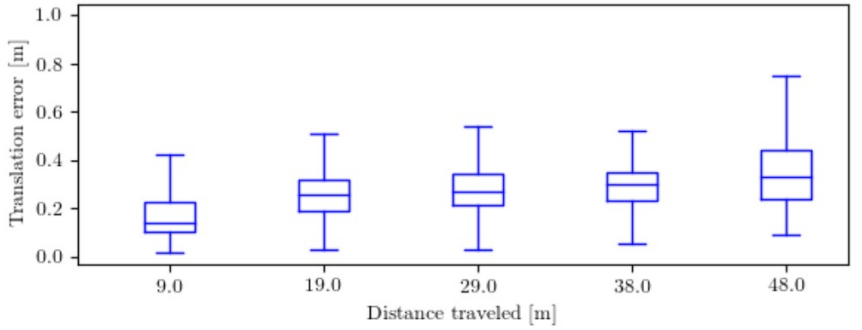
Geiger et al. "Are we ready for autonomous driving? the KITTI vision benchmark suite." CVPR 2012.
Zhang et al., "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." IROS'18.

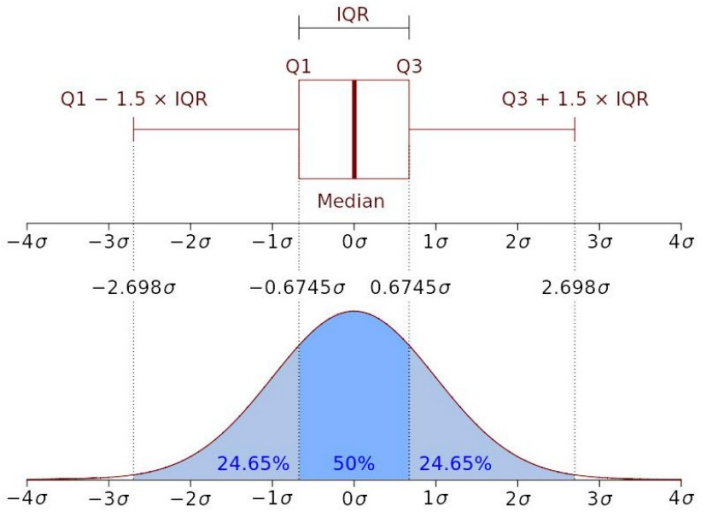
Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

Relative Trajectory Error (RTE)

- Relative Error (Odometry Error) - Statistics of sub-trajectories of specified lengths:
- Calculate errors for all the sub-trajectories of certain lengths.







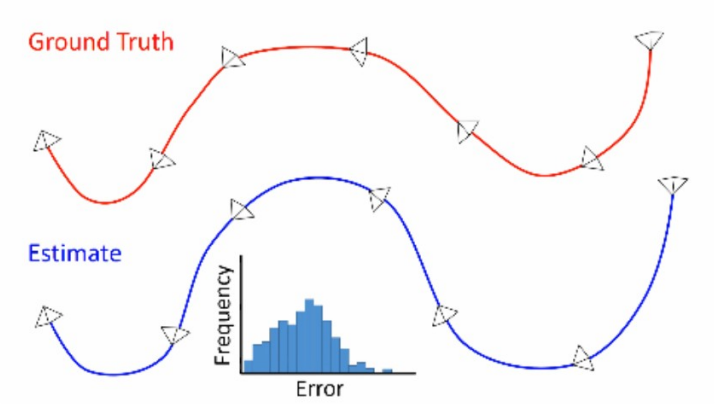
Advantage: Informative statistics
Disadvantage: Complicated to compute and rank

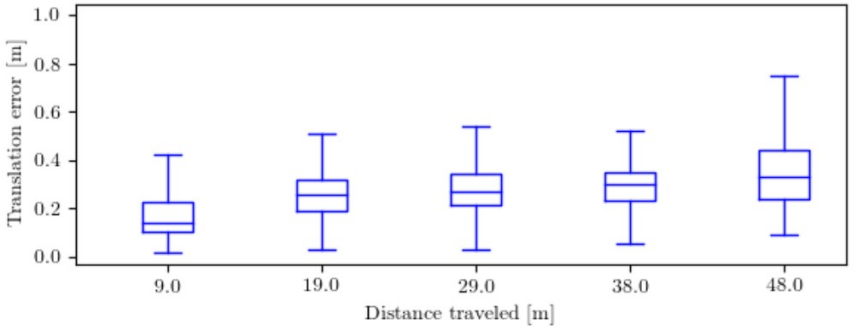
Geiger et al. "Are we ready for autonomous driving? the KITTI vision benchmark suite." CVPR 2012.
Zhang et al., "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." IROS'18.

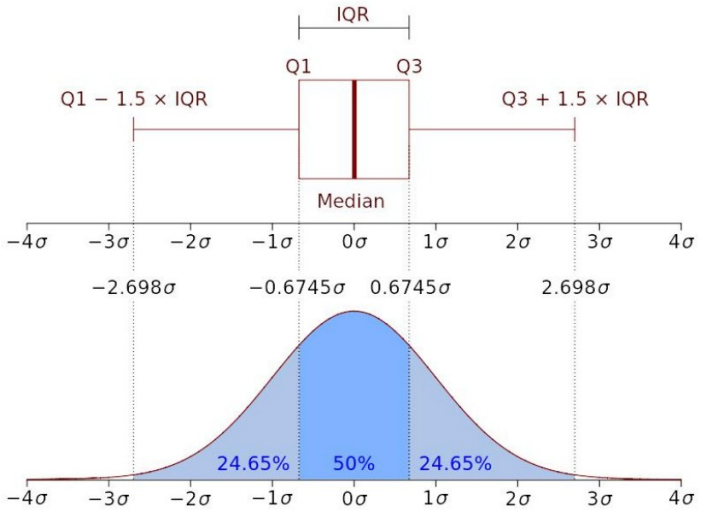
Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

Relative Trajectory Error (RTE)

- Relative Error (Odometry Error) - Statistics of sub-trajectories of specified lengths:
- Calculate errors for all the sub-trajectories of certain lengths.







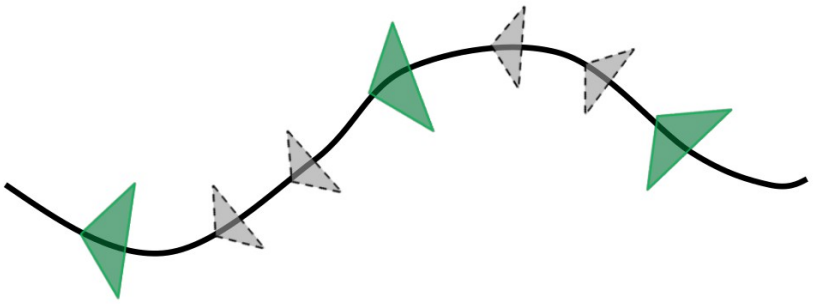
Advantage: Informative statistics
Disadvantage: Complicated to compute and rank

Geiger et al. "Are we ready for autonomous driving? the KITTI vision benchmark suite." CVPR 2012.
Zhang et al., "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." IROS'18.

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

Trajectory Accuracy: Error Metrics

- Both ATE and RTE are widely used in practice, but:
- Many details need to be specified:
  - Number of poses used for the alignment (also, frames or keyframes?)
  - Type of transformation used for the alignment:
  - SE(3) for stereo VO, Sim(3) for monocular VO, 4DOF for VIO (Sub-trajectory lengths in RTE)
- Evaluation tool to facilitate reproducible evaluation: Trajectory Evaluation Toolbox



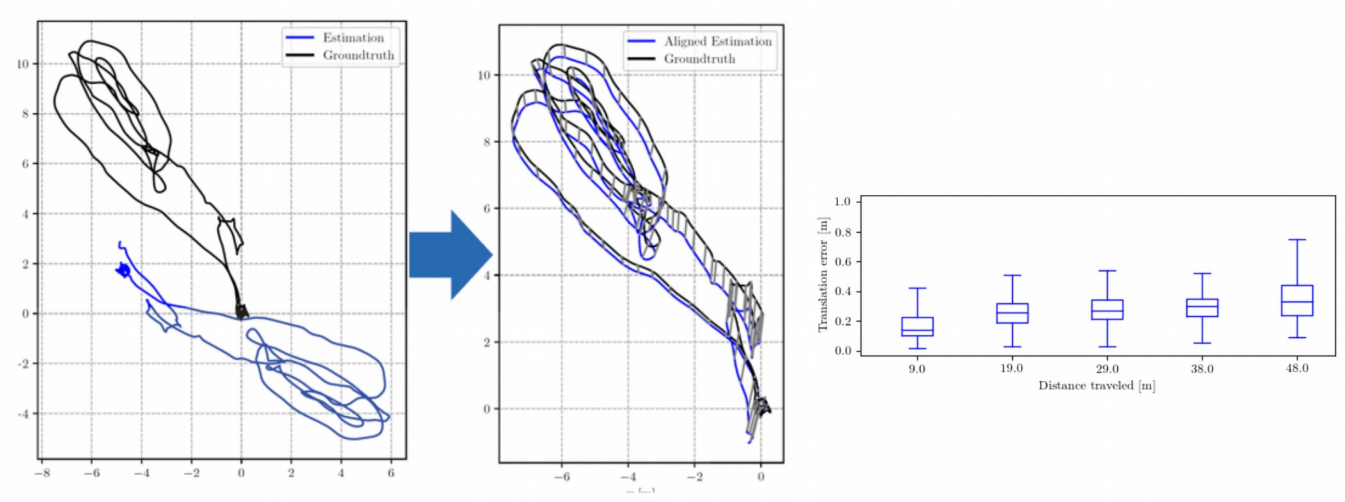
White: Normal frames (used for real time pose update)
Green: Keyframes (usually updated after BA)

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

Trajectory Evaluation Toolbox

- Designed to make trajectory evaluation easy
- Implements different alignment methods depending on the sensing modalities:

  SE(3) for stereo, sim(3) for monocular, 4DOF for VIO
- Implements Absolute Trajectory Error and Relative Error
- Automated evaluation of different algorithms on multiple datasets (for N runs)
- Code: https://github.com/uzh-rpg/rpg_trajectory_evaluation [Zhang, IROS'18]



Zhang et al., "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry." IROS'18.

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

Visual Odometry - Drift

All visual odometry algorithms suffer from motion drift due to the integration of the relative displacements between consecutive poses which unavoidably accumulates errors overtime.

This drift becomes evident usually after a few hundred meters.

But the results may vary depending on the abundance of features in the environment, the resolution of the cameras, the presence of moving objects like people or other passing vehicles, and the illumination conditions.

The reason visual odometry is becoming more and more popular in both robotics and automotive is that drift can be canceled if the vehicle revisits a place that has already been observed previously.

The possibility of performing location recognition (or place recognition) is one of the main advantages of vision compared to other sensor modalities

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

Loop Closure

First observation

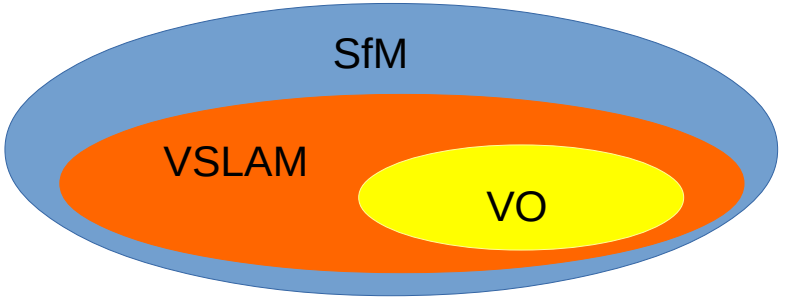
Second observation after a loop

Loop constraints are very valuable constraints for pose graph optimization

- 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 (GIST descriptors) or local image descriptors (e.g., SIFT, BRIEF, BRISK features)
- Image retrieval is the problem of finding the most similar image of a template image in a database of billion images (image retrieval). This can be solved efficiently with Bag of Words [Sivic'03, Nister'06, FABMAP, Galvez-Lopez'12 (DBoW2)]

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

SfM – VO and VSLAM



Structure from Motion (SfM): The problem of recovering relative camera poses and three-dimensional (3-D) structure from a set of camera images (calibrated or non-calibrated).
SfM is more **general** and tackles the problem of 3-D reconstruction of both the structure and camera poses from **sequentially ordered or unordered image sets**.

Visual Odometry (VO) is a special case of SfM: The final structure and camera poses are typically refined with an offline optimization (i.e., bundle adjustment (BA)), whose computation time grows with the number of images.
VO focuses on estimating the 3-D motion of the **camera sequentially** - as a new frame arrives - and in **real time**. Bundle adjustment can be used to refine the local estimate of the trajectory. VO aims at recovering the robot path incrementally, pose after pose, and potentially optimizing only over the last n poses of the path (**windowed bundle adjustment**). This sliding window optimization can be considered equivalent to building a local map in SLAM; however, the philosophy is different: in VO, we only care about **local consistency of the trajectory** and the **local map is used to obtain a more accurate estimate of the local trajectory**. VO can be used as a building block for a complete SLAM algorithm to recover the incremental of the camera including loop closure;

Visual Simultaneous Localization and Mapping (VSLAM) aims to obtain a global, consistent estimate of the robot path. This implies keeping a track of a map of the environment to realize when the robot returns to a previously visited area - **loop closure.** When a loop closure is detected, this information is used to reduce the drift in both the map and camera path. Understanding when a loop closure occurs and efficiently integrating this new constraint into the current map are two of the main issues in SLAM.) SLAM is concerned with the global map consistency.

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

VO - SLAM

- 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.

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

VO versus VSLAM

Visual Odometry

- incremental estimation
- local consistency

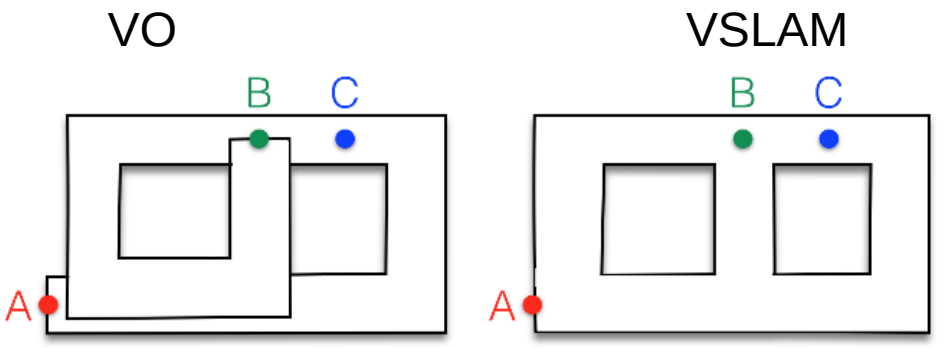


Fig. 1: Left: map built from odometry. The map is homotopic to a long corridor that goes from the starting position A to the final position B. Points that are close in reality (e.g., B and C) may be arbitrarily far in the odometric map. Right: map build from SLAM. By leveraging loop closures, SLAM estimates the actual topology of the environment, and "discovers" shortcuts in the map.

Image courtesy of [Cadena et al., 2016]

Visual Simultaneous Localization and Mapping

- SLAM = visual odometry + loop detection & closure
- global consistency

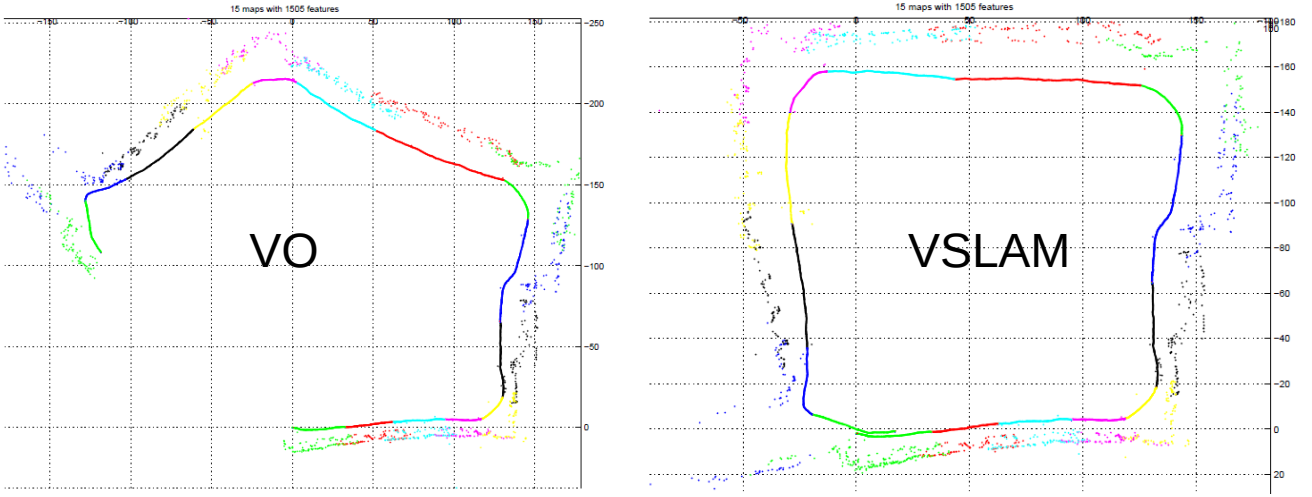
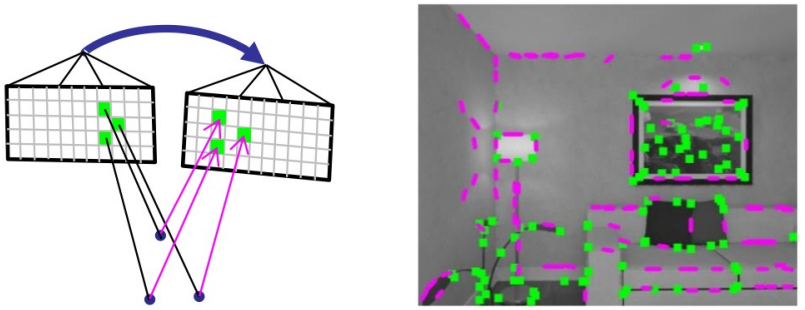


Image courtesy of [Clemente et al., RSS2007]

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

SVO: Fast, Semi-Direct Visual Odometry

[Forster, Pizzoli, Scaramuzza, ICRA'14]:

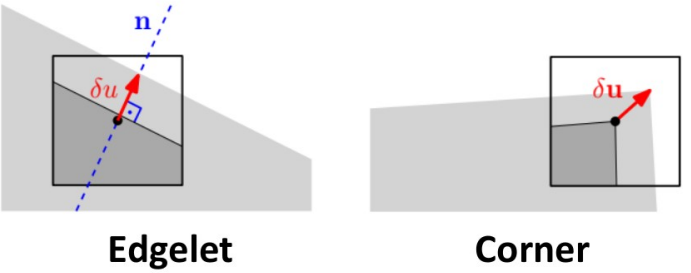


Direct (minimizes photometric error)

- Corners and edgelets
- Frame-to-frame motion estimation

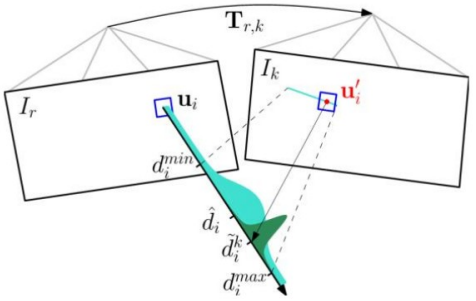
Feature-based (minimizes reprojection error)

- Frame-to-Keyframe pose refinement



Mapping

Probabilistic depth estimation of 3D points



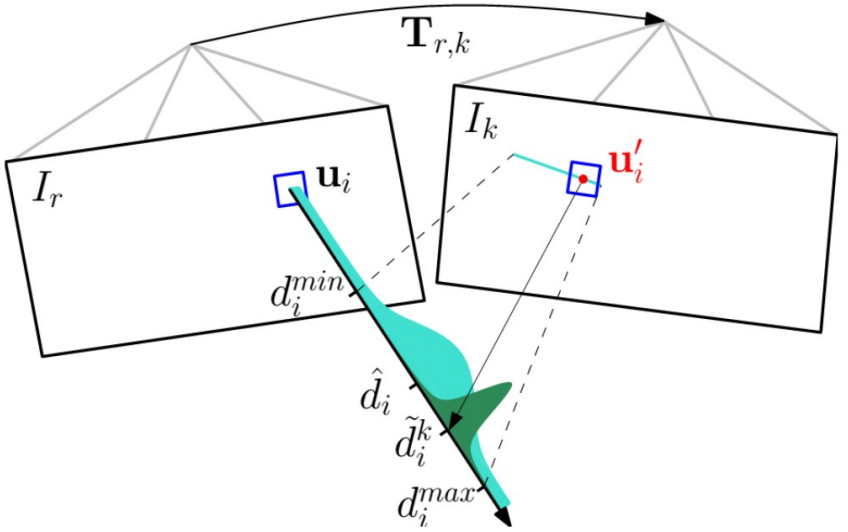
SVO: Fast, Semi-Direct Visual Odometry

[Forster, Pizzoli, Scaramuzza, ICRA'14]:

Probabilistic Depth Estimation

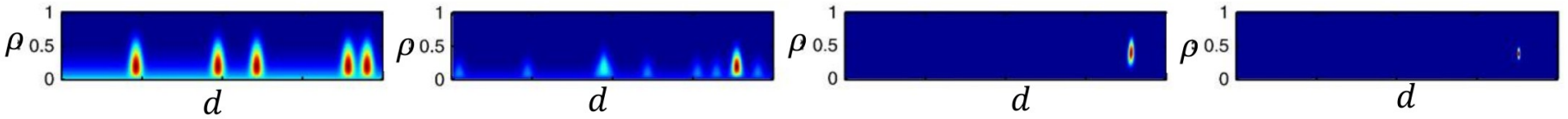
Depth-Filter:

- Depth Filter for every feature
- Recursive Bayesian depth estimation



Mixture of Gaussian + Uniform distribution

$$p(\tilde{d}_i^k | d_i, \rho_i) = \rho_i \mathcal{N}(\tilde{d}_i^k | d_i, \tau_i^2) + (1 - \rho_i)\mathcal{U}(\tilde{d}_i^k | d_i^{\min}, d_i^{\max})$$



SVO: Fast, Semi-Direct Visual Odometry

Processing Times of SVO

Laptop (Intel i7, 2.8 GHz): 400 frames per second

Embedded ARM Cortex-A9, 1.7 GHz: Up to 70 frames per second

Smartphone PC; 100 fps

Source Code

- Open Source available at: github.com/uzh-rpg/rpg_svo
- Works with and without ROS
- Closed-Source professional edition (SVO 2.0): available for companies
- SVO 2.0 includes Fish-eye & Omni cameras, Multi-camera systems

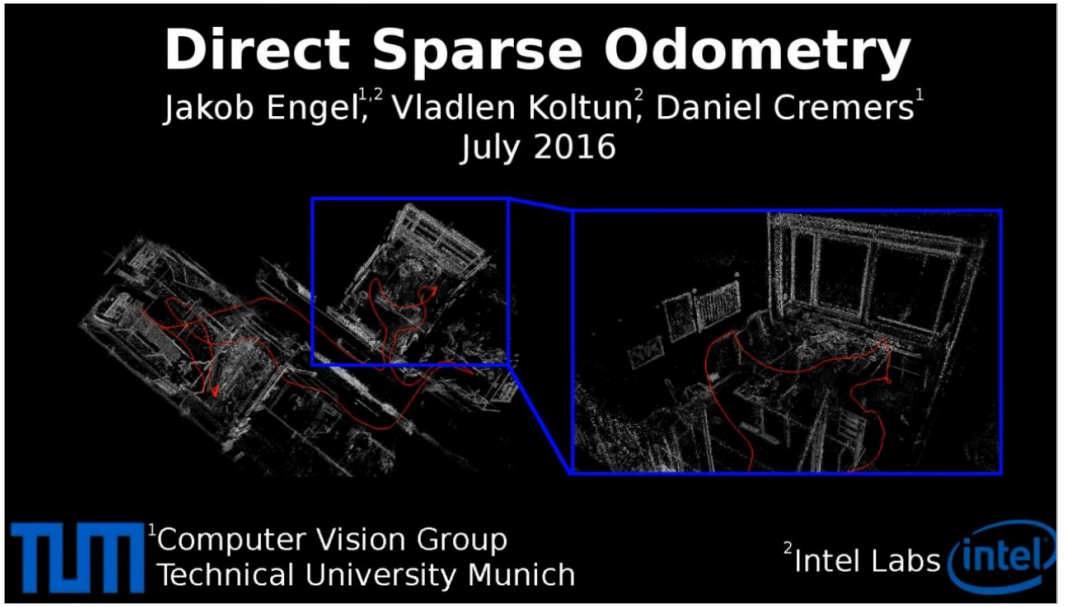
Direct Sparse Odometry (DSO)

[Engel, Koltun, Cremers, DSO: Direct Sparse Odometry, PAMI'17]

Direct (photometric error) + Sparse formulation:

- 3D geometry represented as sparse large gradients
- Minimizes photometric error
- Jointly optimizes poses & structure (sliding window)
- Incorporate photometric correction to compensate exposure time change
- Real-time (30Hz)

$$E_{pj} := \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{p}} w_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma$$



https://vision.in.tum.de/research/vslam/dso

Software

Visual Odometry

- Modified PTAM [Weiss et al.,]: (feature-based, mono): http://wiki.ros.org/ethzasl_ptam
- SVO [Forster et al.] (semi-direct, mono, stereo, multi-cameras): https://github.com/uzh-rpg/rpg_svo
- LIBVISO2 (feature-based, mono and stereo): http://www.cvlibs.net/software/libviso
- DSO [Engel'16] -> Munich, Cremers' lab

Understanding Check

- How can motion estimation be performed dependent on the dimensionality cases?
- Can you define VO?
- What is the relationship between VO, SfM and VSLAM?
- What are keyframes? Why do we need them and how can we select them?
- Are you able to define loop closure detection? Why do we need loops?
- Are you able to provide a list of the most popular open source VO algorithms?
- Are you able to describe the differences between feature-based methods and direct methods?
- Are you able to name accuracy evaluation measurements?
- Describe briefly SVO and DSO.

Literature

- 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.
- C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I.D. Reid, J.J. Leonard, Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age, IEEE Transactions on Robotics, Vol. 32, Issue 6, 2016.