University of Zurich[UZH] | ETH zürich

Institute of Informatics – Institute of Neuroinformatics

ROBOTICS & PERCEPTION GROUP

Lecture 09
Multiple View Geometry 3

Davide Scaramuzza

Lab Exercise 6 - Today

- Room ETH HG E 1.1 from 13:15 to 15:00
- Work description: P3P algorithm and RANSAC

Outline

- Bundle Adjustment
- SFM with n views

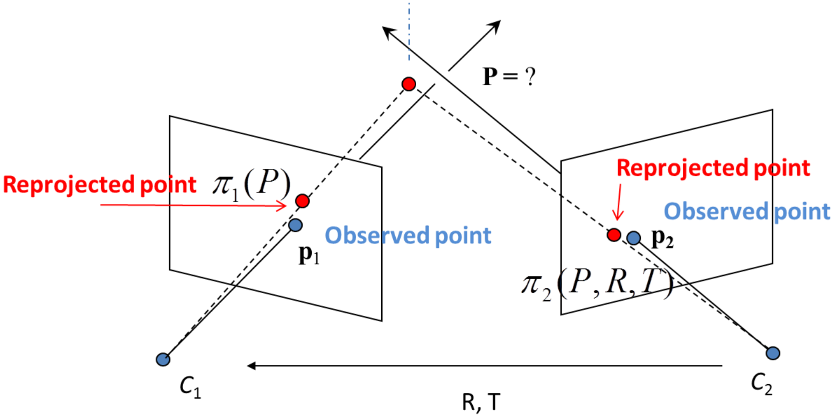
Bundle Adjustment (BA)

- **Non-linear, simultaneous refinement of structure P^i and motion $C = R, T$**
- It is used after linear estimation of R and T (e.g., after 8-point algorithm)
- Computes C, P^i by minimizing the Sum of Squared Reprojection Errors:

$$(P^i, C_1, C_2) = \arg\min_{R,T,P^i} \sum_{i=1}^{N} \left\| p_1^i - \pi_1(P^i, C_1) \right\|^2 + \left\| p_2^i - \pi_2(P^i, C_2) \right\|^2$$

NB: here, by C_1, C_2 we denote the **pose of** each camera in the **world** frame

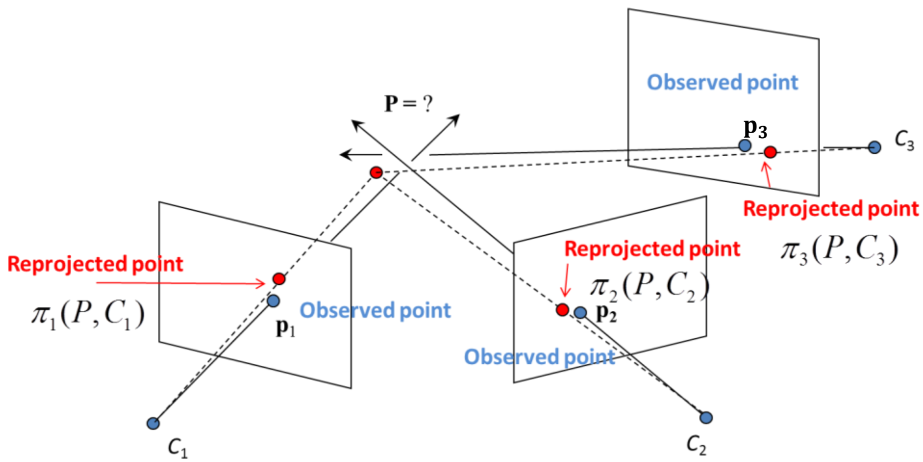
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)
- In order to not get stuck in local minima, the **initialization should be close the minimum**

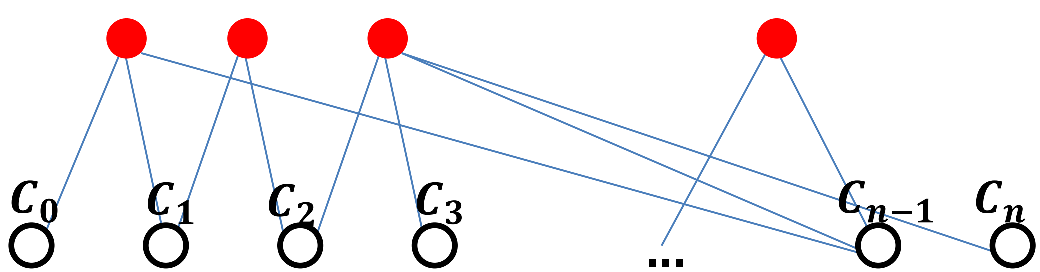


Bundle Adjustment (BA) for n Views

Minimizes the Sum of Squared Reprojection Errors **over each view k**

$$(P^i, C_k) = \arg\min_{P^i, C_k} \sum_k \sum_i \left\| p^i_k - \pi_k(P^i, C_k) \right\|^2$$





Outline

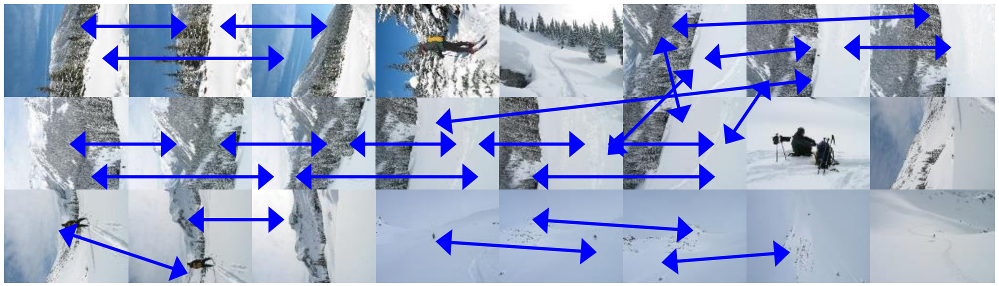
- Bundle Adjustment
- SFM with n views

Structure From Motion with n Views

- Compute initial structure and motion
  - **Hierarchical SFM**
  - **Sequential SFM**
- Refine simultaneously structure and motion through BA

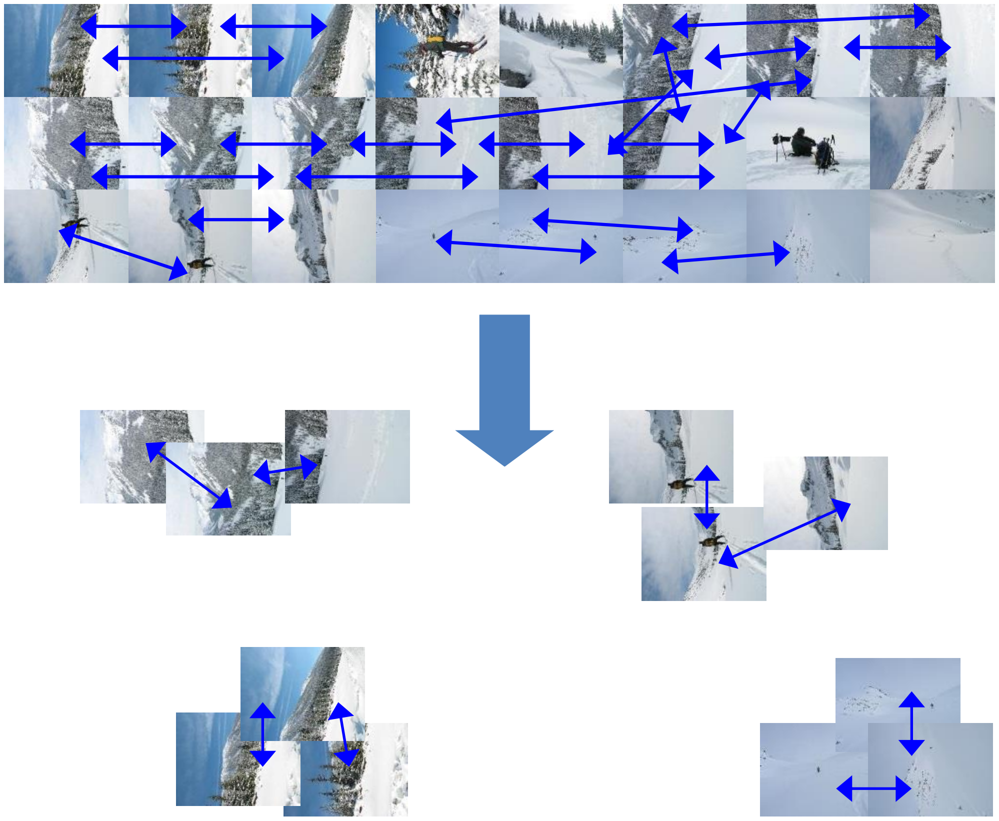
Hierarchical SFM

1. Extract and match features between nearby frames



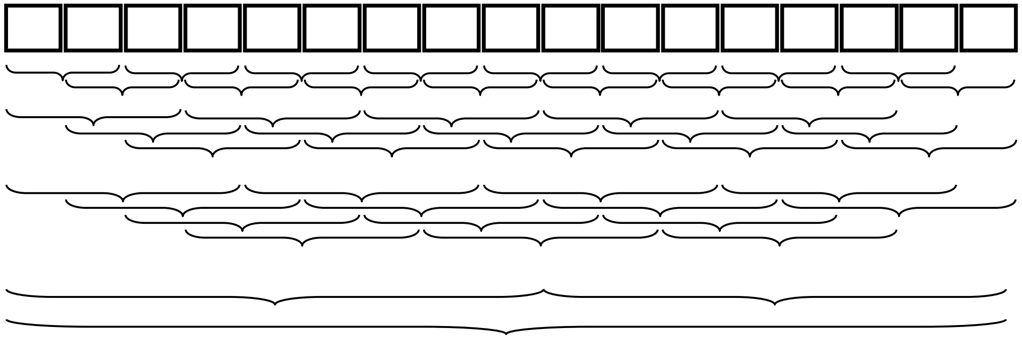
Hierarchical SFM

1. Extract and match features between nearby frames
2. Identify clusters consisting of 3 nearby frames:
3. Compute SFM for 3 views:
    1. Compute SFM between 1 and 2 and build point cloud
    2. Then merge 3rd view by running 3-point RANSAC between point cloud and 3rd view



Hierarchical SFM

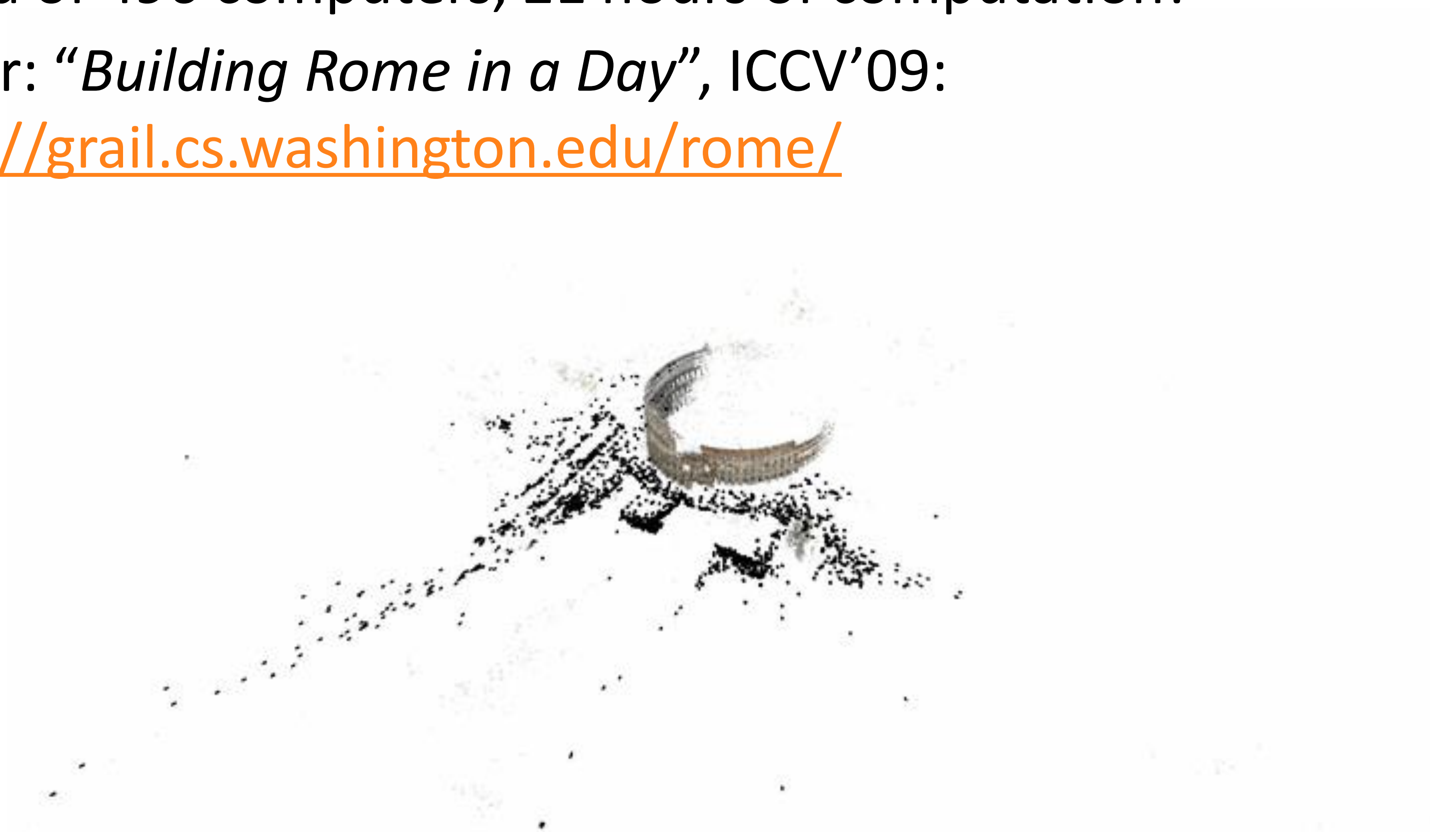
1. Extract and match features between nearby frames
2. Identify clusters consisting of 3 nearby frames:
3. Compute SFM for 3 views:
   1. Compute SFM between 1 and 2 and build point cloud
   2. Then merge 3rd view by running 3-point RANSAC between point cloud and 3rd view
4. Merge clusters pairwise and refine (BA) both structure and motion



How do you merge clusters?

Hierarchical SFM: Example

- Reconstruction from 150,000 images from Flickr.com associated with the tags “Rome” and “Roma”
- Cloud of 496 computers, 21 hours of computation!
- Paper: “*Building Rome in a Day*”, ICCV’09: http://grail.cs.washington.edu/rome/



Structure From Motion with n Views

- Compute initial structure and motion
  - **Hierarchical SFM**
  - **Sequential SFM**
- Refine simultaneously structure and motion through BA

Sequential SFM - also called Visual Odometry (VO)

- Initialize structure and motion from 2 views (**bootstrapping**)
- For each additional view
  - Determine pose (**localization**)
  - Extend structure (i.e., extract and triangulate new features)
  - Refine both pose and structure (BA)

A Brief history of 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*).

A Brief history of 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*).
- **1980 to 2000**: The VO research was dominated by **NASA/JPL** in preparation of the **2004 mission to Mars**
- **2004**: VO was used on a robot on another planet: Mars rovers Spirit and Opportunity (see seminal paper from NASA/JPL, 2007)
- **2004**. VO was revived in the academic environment by **David Nister**’s «Visual Odometry» paper. The term VO became popular.

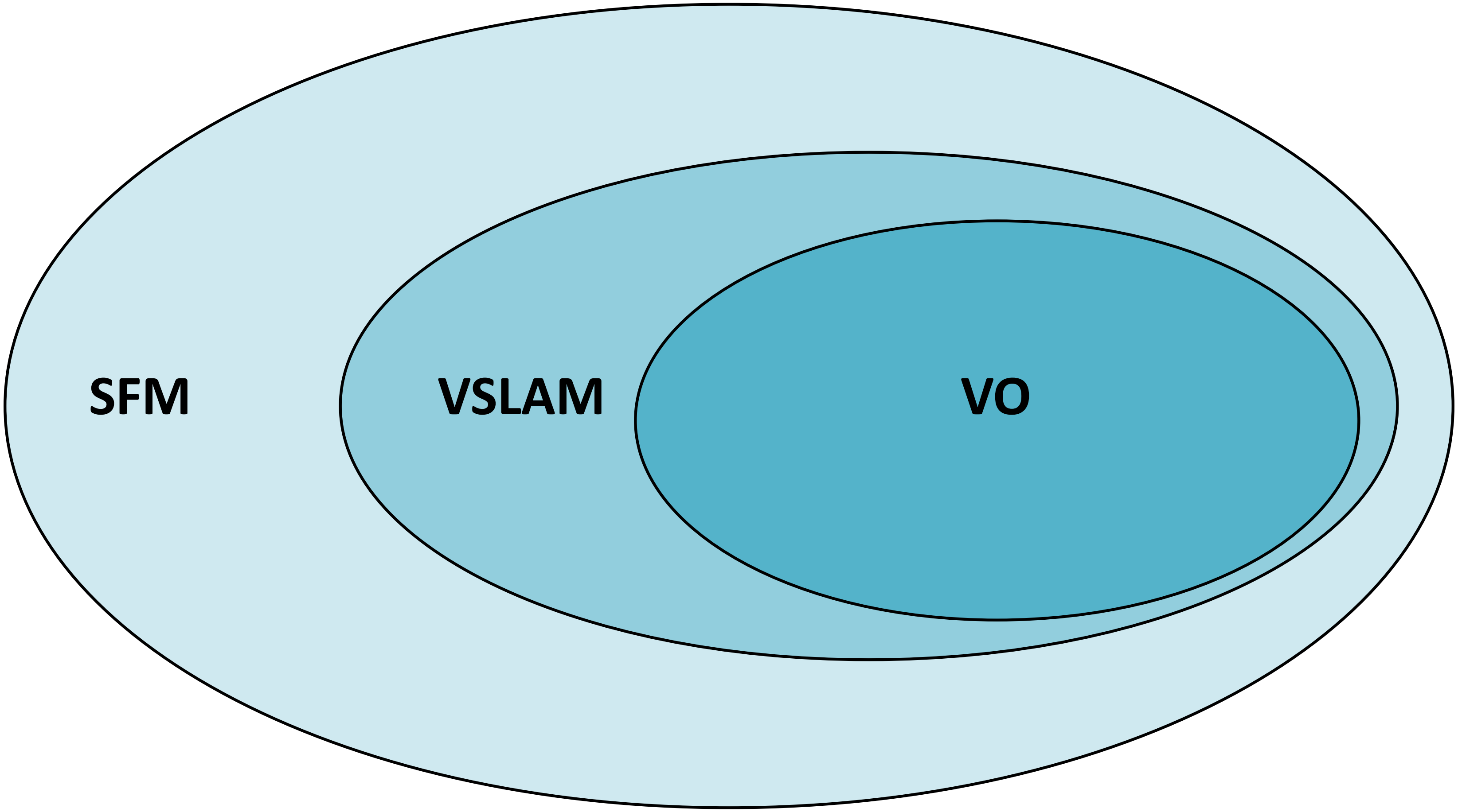
More about history and tutorials

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

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

- 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. PDF

VO vs VSLAM vs SFM



Structure from Motion (SFM)

SFM is more general than VO and tackles the problem of 3D reconstruction and 6DOF pose estimation from **unordered image sets**

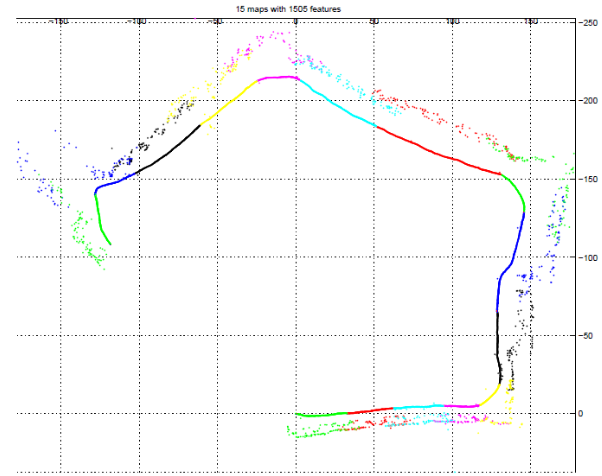
Reconstruction from 3 million images from Flickr.com
Cluster of 250 computers, 24 hours of computation!
Paper: “Building Rome in a Day”, ICCV’09

VO vs SFM

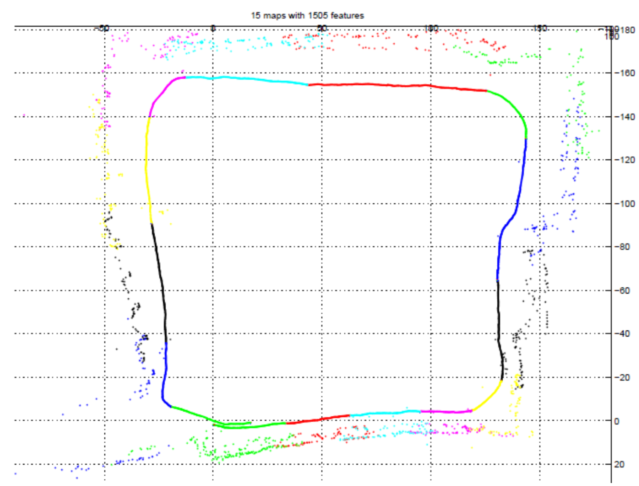
- VO is a **particular case** of SFM
- VO focuses on estimating the 3D motion of the camera **sequentially** (as a new frame arrives) and in **real time**.
- Terminology: sometimes SFM is used as a synonym of VO

VO vs. Visual SLAM

- **Visual Odometry**
  - Focus on incremental estimation/**local consistency**
- **Visual SLAM**: Simultaneous Localization And Mapping
  - Focus on **globally consistent** estimation
  - **Visual SLAM = visual odometry + loop detection + graph optimization**
- VO **sacrifices consistency for real-time performance**, without the need to keep track of all the previous history of the camera.



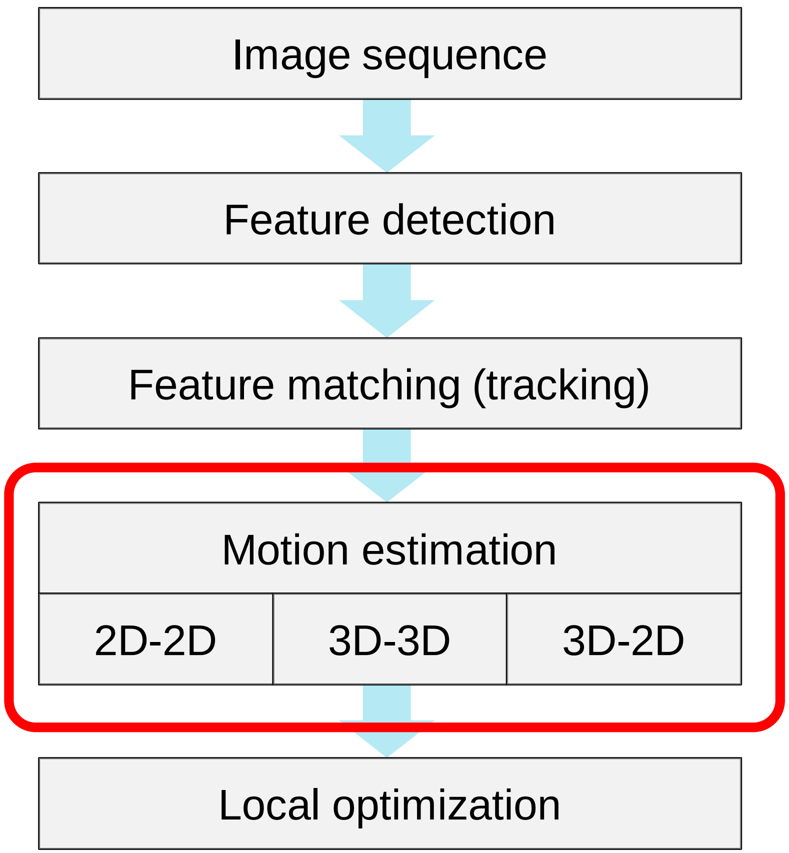
Visual odometry

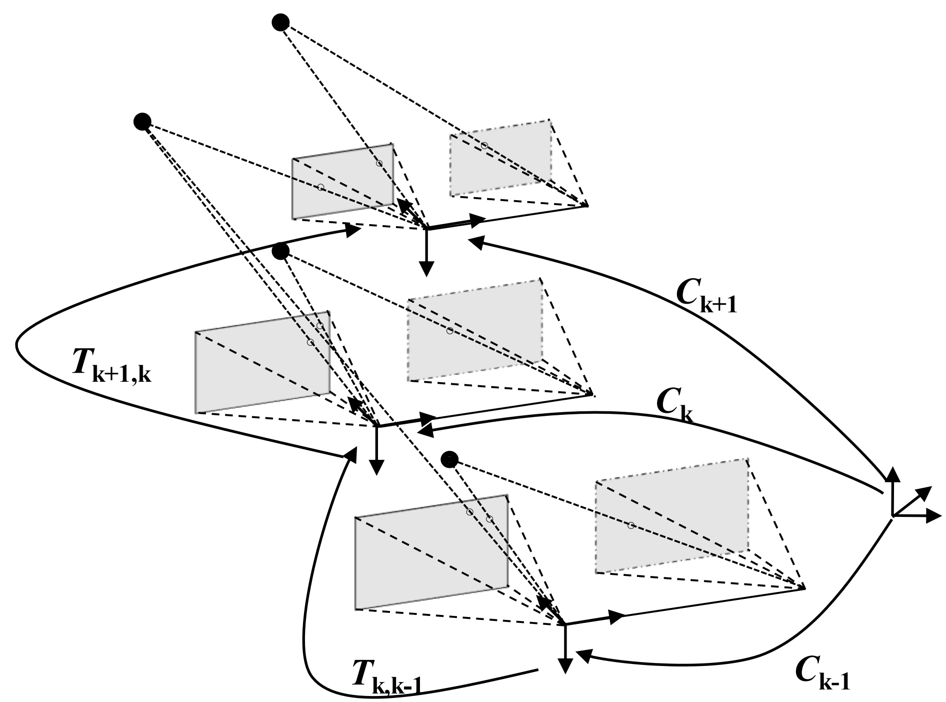


Visual SLAM

VO Flow Chart

VO computes the camera path incrementally (pose after pose)





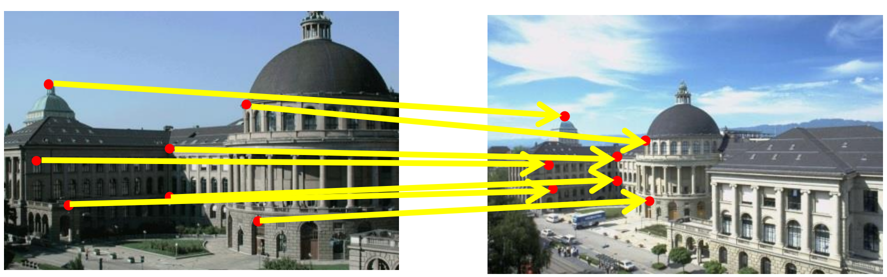
2D-to-2D

Motion from Image Feature Correspondences

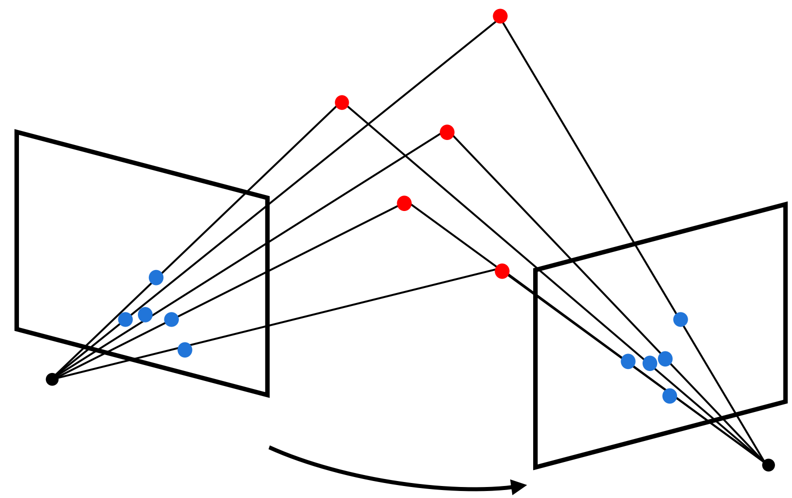
- Both feature points f_{k-1} and f_k are specified **in 2D**
- The minimal-case solution involves **5-point** correspondences
- The solution is found by minimizing the reprojection error:

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

- Popular algorithms: 8- and 5-point algorithms [Hartley'97, Nister'06]



I_{k-1} I_k



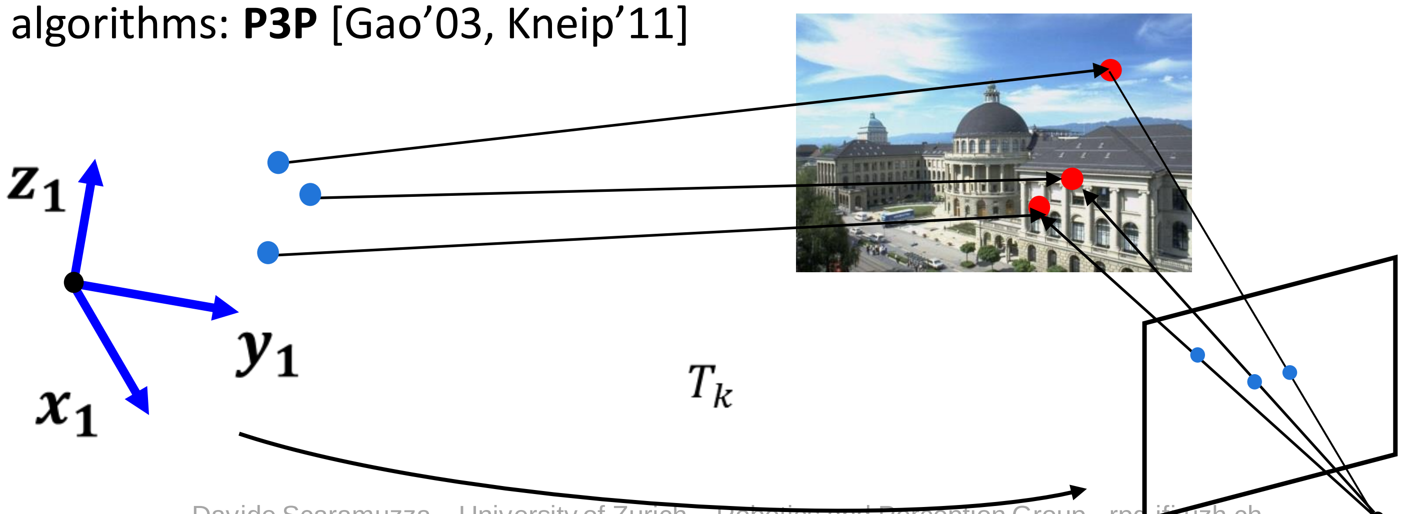
3D-to-2D

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** (+1 for disambiguating the 4 solutions)
- The solution is found by minimizing the reprojection error:

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

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



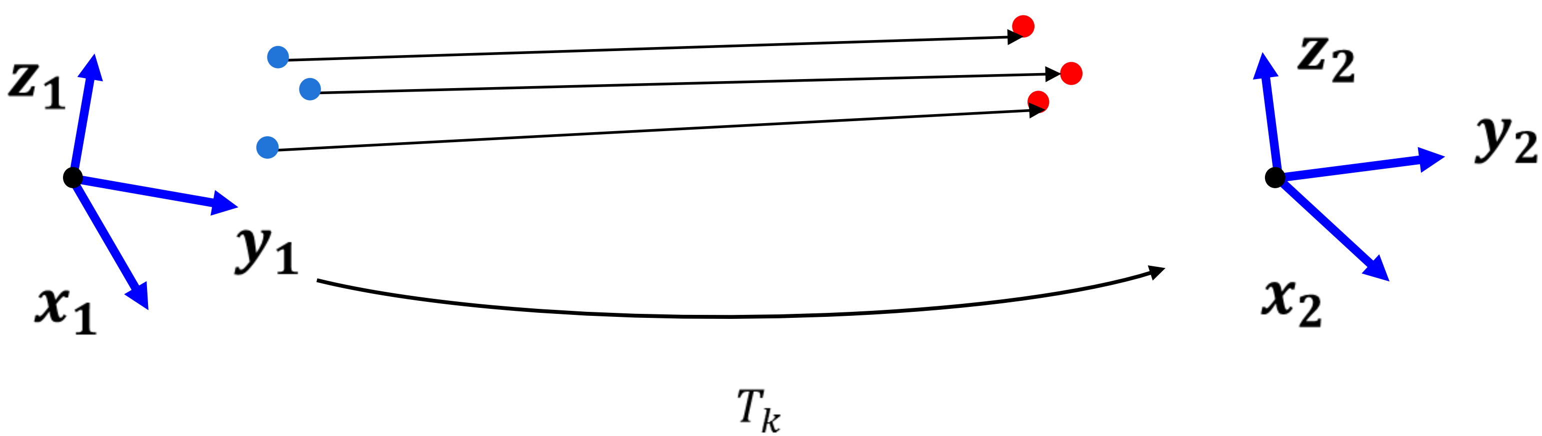
3D-to-3D

Motion from 3D-3D Point Correspondences (point cloud registration)

- 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 minimizing the 3D-3D Euclidean distance:

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

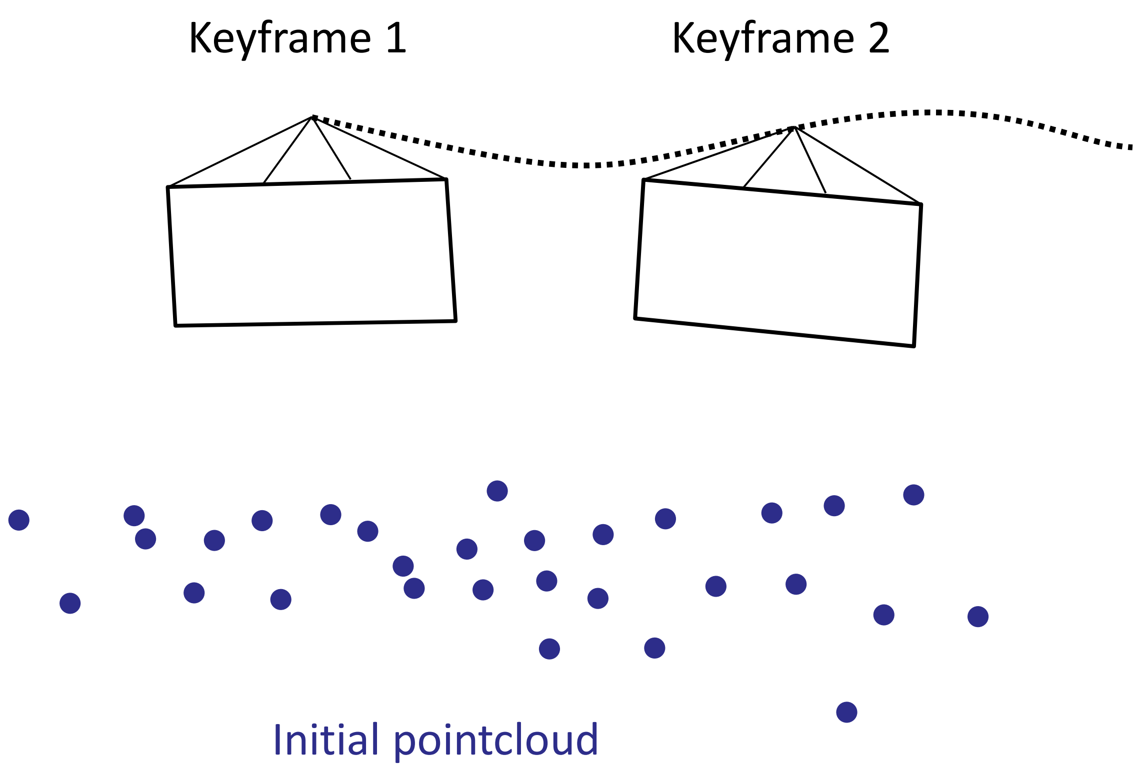
- Popular algorithm: [Arun'87] for global registration, ICP for local refinement or Bundle Adjustment (BA)



Case Study:
Monocular Visual Odometry

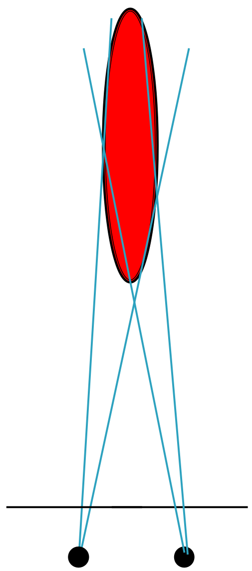
Monocular VO (i.e., with a single camera)

- **Bootstrapping**
  - Initialize structure and motion from 2 views: e.g., 8-point algorithm + RANSAC
  - Refine structure and motion (BA)
  - How far should the two frames (i.e., keyframes) be?

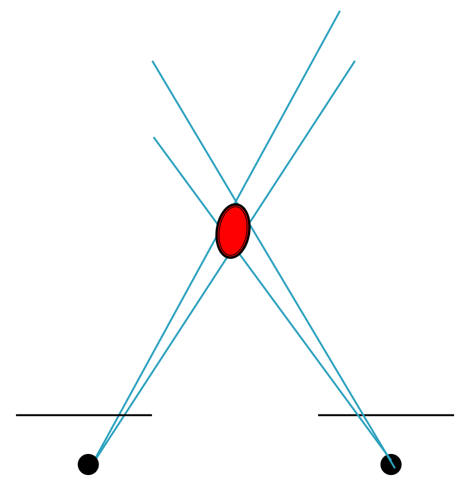


Skipping frames (Keyframe Selection)

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



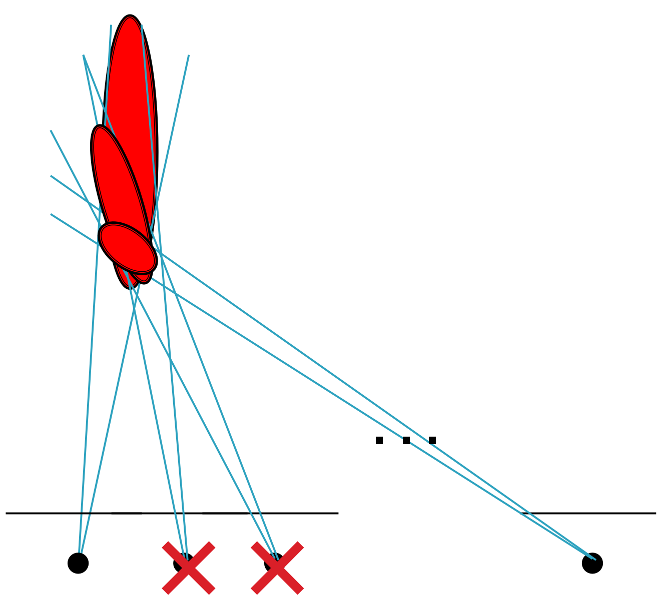
Small baseline → large depth uncertainty



Large baseline → small depth uncertainty

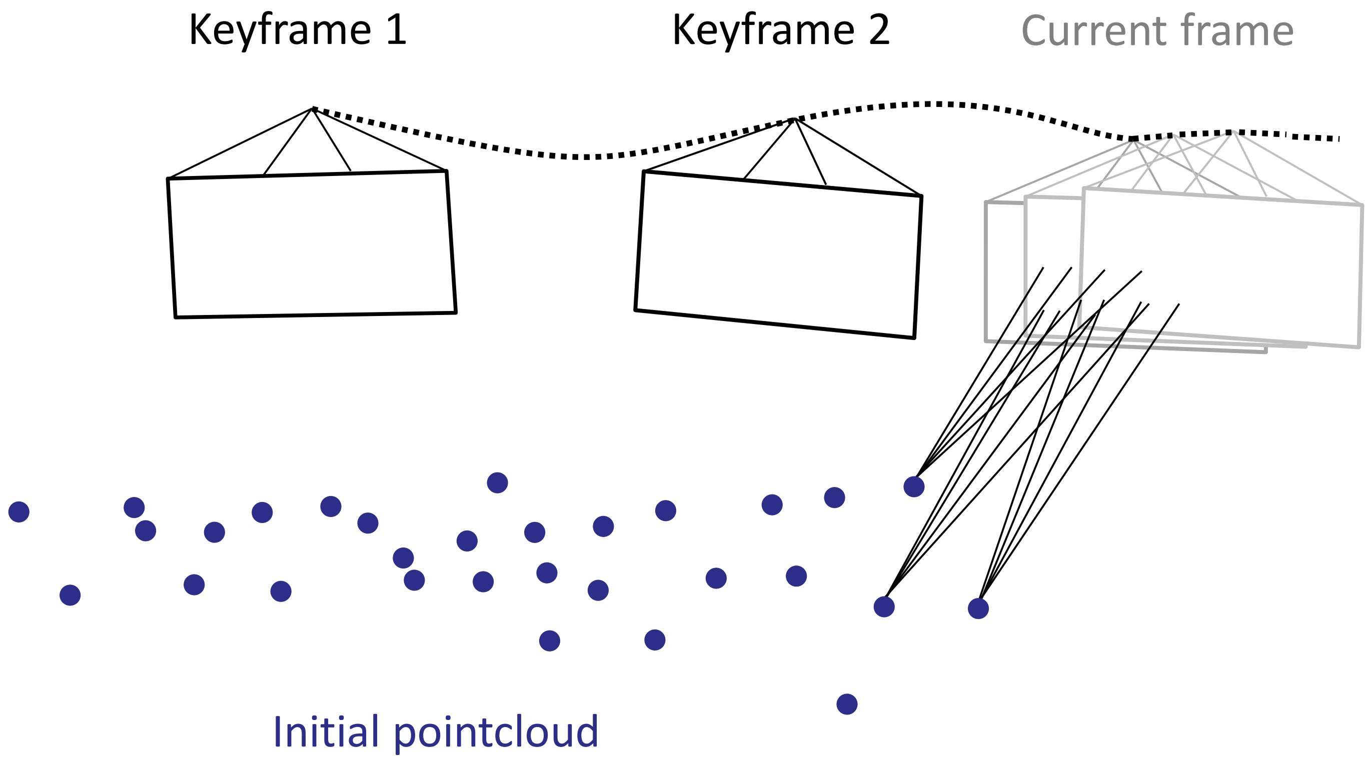
Skipping frames (Keyframe Selection)

- 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 $\frac{\textit{keyframe distance}}{\textit{average-depth}} > \textit{threshold (~10-20 \%)}$



Monocular VO (i.e., with a single camera)

- **Localization**
  - Determine the pose of each additional view
    - How?
    - How long can I do that?



Localization

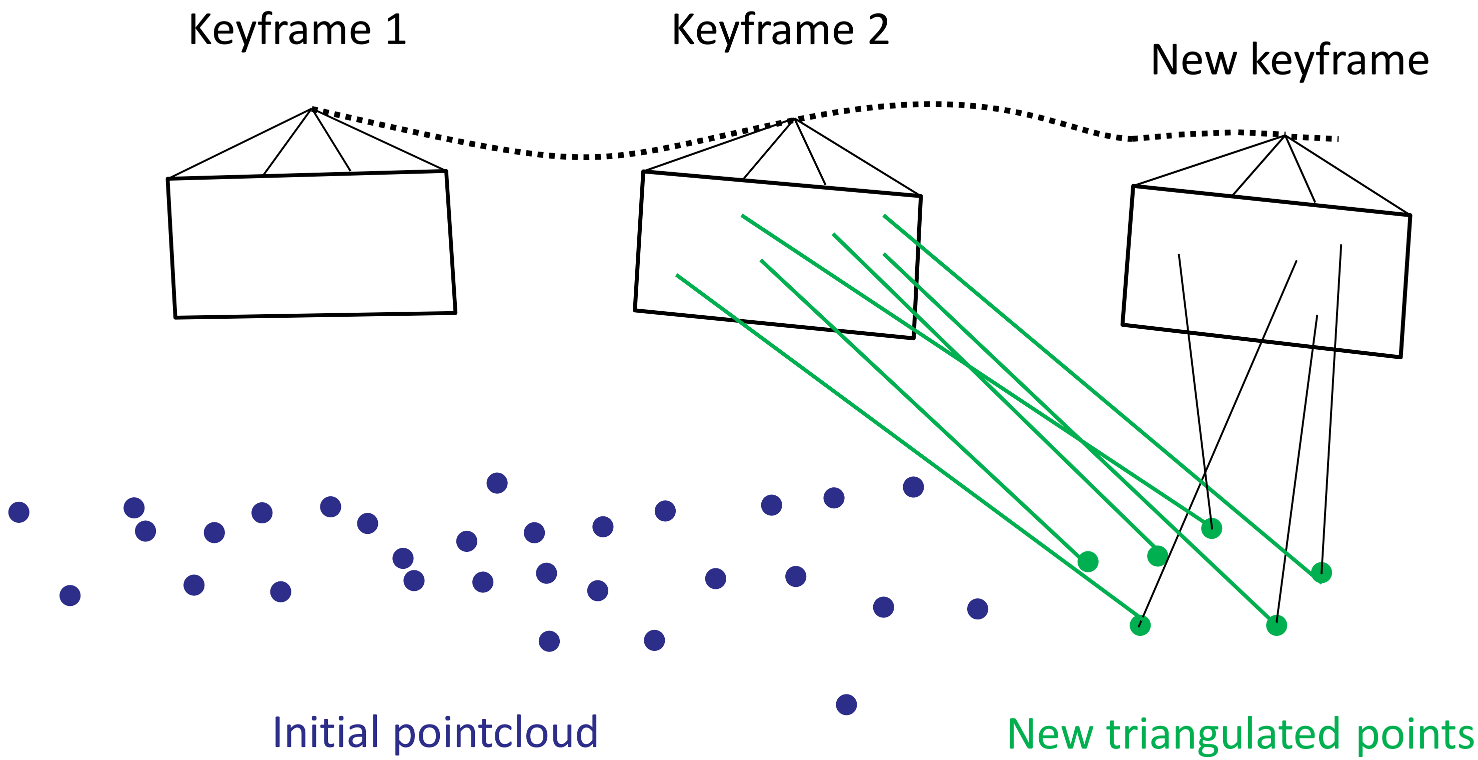
- Compute camera pose from known 3D-to-2D feature correspondences
  - Extract correspondences (how?)
  - Solve for R and t (K is known)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \left[R \mid T \right] \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- What's the minimal number of required point correspondences?
  - Lecture 3:
    - 6 for linear solution (DLT algorithm)
    - 3 for a non linear solution (P3P algorithm)

Extend Structure

- Extract and triangulate new features
  - Is it necessary to do this for every frame or can we just do it for keyframes?
  - What are the pros and cons?



Monocular Visual Odometry: putting all pieces together

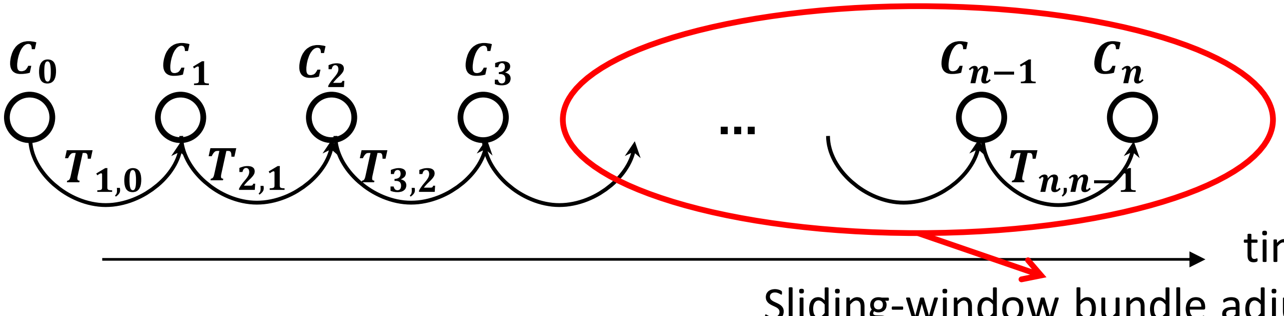
- We denote the relative motion between adjacent keyframes:

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

- By concatenation of all these transformations, the full trajectory of the camera can be recovered:

$$C_k = T_{k,k-1} C_{k-1}$$

- A non-linear refinement (BA) over the last m poses (+ visible structure) can be performed to get a more accurate estimate of the local trajectory



Loop Closure Detection (i.e., Place Recognition)

- **Relocalization problem:**
  - During VO, tracking can be lost (due to occlusions, low texture, quick motion, illumination change)

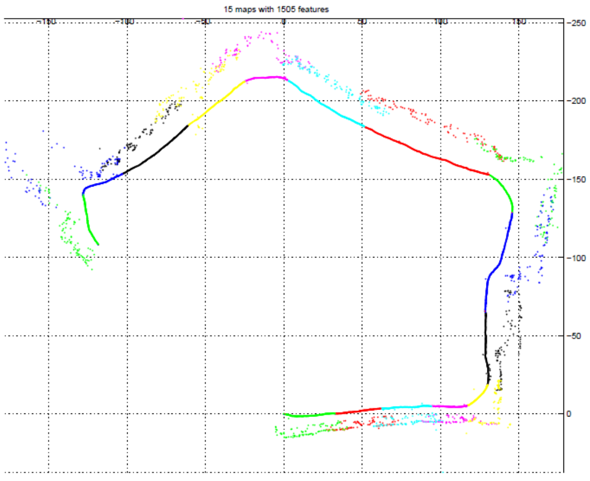
- Solution: **Re-localize** camera pose and continue

- **Loop closing problem**
  - When you go back to a previously mapped area:
    - **Loop detection**: to avoid map duplication
    - **Loop correction**: to compensate the accumulated drift
  - In both cases you need a place recognition technique

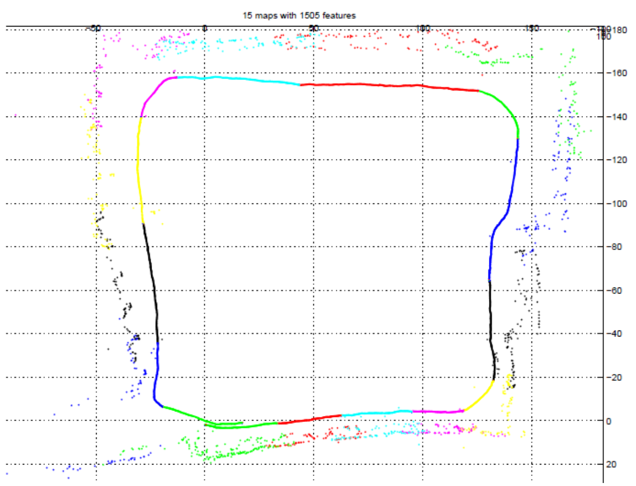
We will address place recognition in Lecture 12

Recall: VO vs. Visual SLAM

- **Visual SLAM = visual odometry + loop detection + graph optimization**



Visual odometry

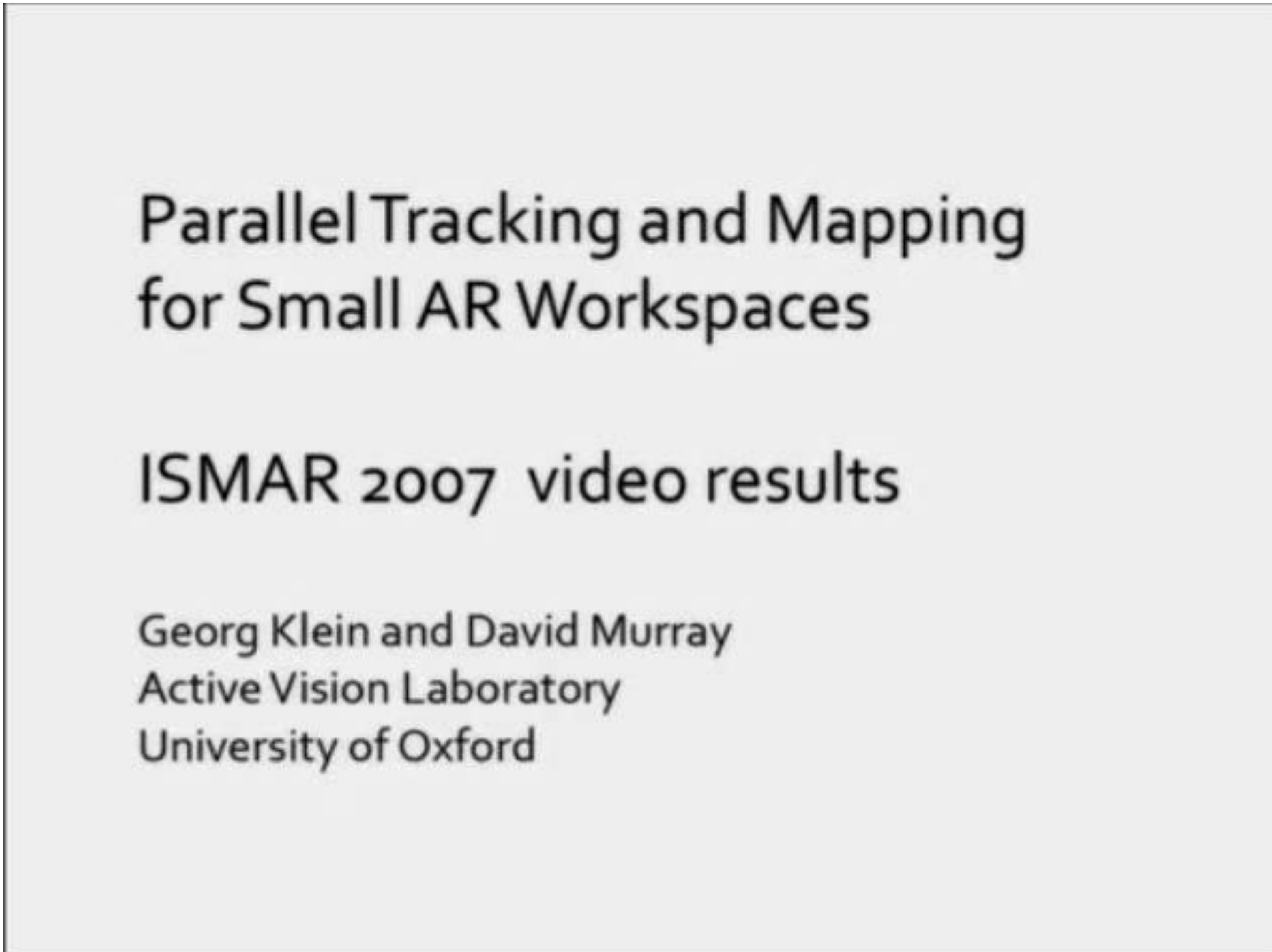


Visual SLAM

Open Source Monocular VO and SLAM algorithms

- **PTAM** [Klein, 2007] -> Oxford, Murray's lab
- **ORB-SLAM** [Mur-Artal, T-RO, 15] -> Zaragoza, Tardos' lab
- **LSD-SLAM** [Engel, ECCV'14] -> Munich, Cremers' lab
- **DSO** [Engel'16] -> Munich, Cremers' lab
- **SVO** [Forster, ICRA'14, TRO'17] -> Zurich, Scaramuzza's lab

PTAM: Parallel Tracking and Mapping for Small AR Workspaces



ORB-SLAM [Mur-Artal, TRO'15]

- **Feature based**
  - ORB feature = FAST corner + Oriented Rotated Brief descriptor
  - Binary descriptor
  - Very fast to compute and compare
  - Minimizes reprojection error

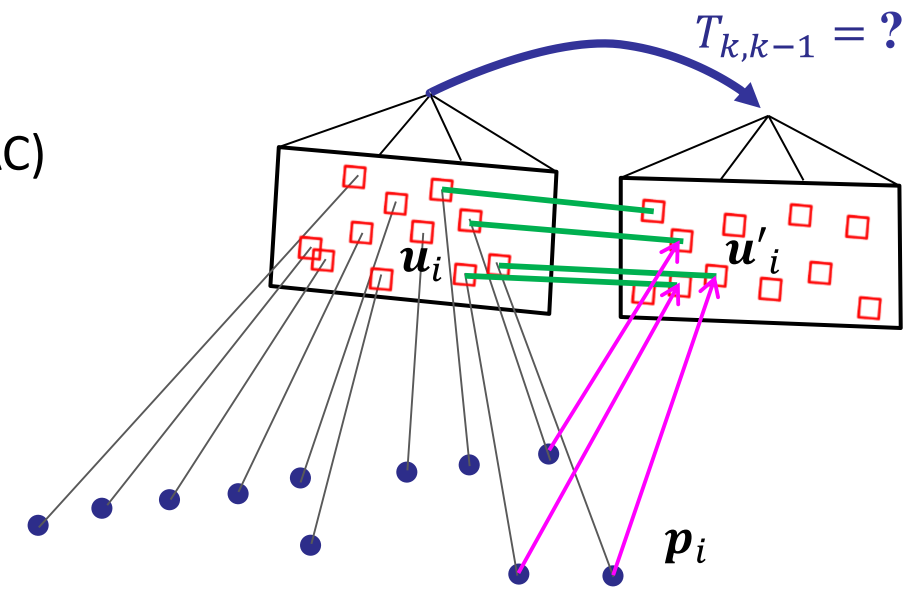
- Includes:
  - **Loop closing**
  - **Relocalization**
  - Final optimization

- **Real-time (30Hz)**

[Mur-Artal, Montiel, Tardos, ORB-SLAM: Large-scale Feature-based SLAM, TRO'15]

Feature-based methods

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



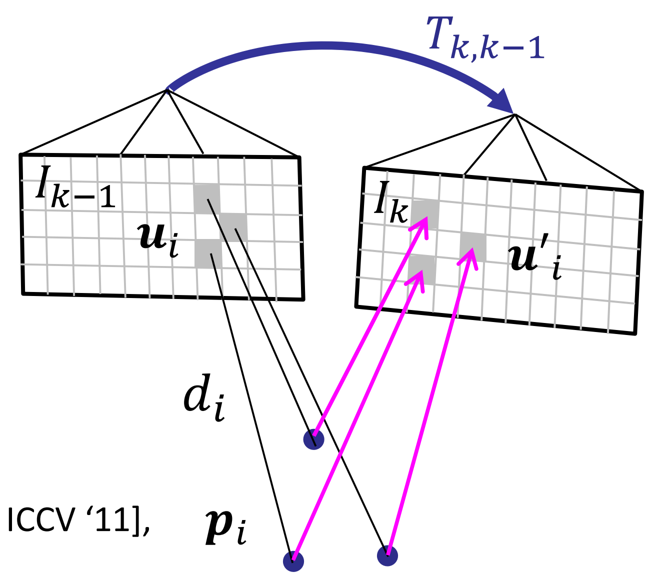
$$T_{k,k-1} = \arg\min_T \sum_i \| \boldsymbol{u}'_i - \pi(\boldsymbol{p}_i) \|^2_\Sigma$$

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



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

Feature-based methods

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

$$T_{k,k-1} = \arg\min_{T} \sum_{i} \| \boldsymbol{u}'_i - \pi(\boldsymbol{p}_i) \|_{\Sigma}^{2}$$

✓ 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 methods

1. Minimize **photometric error**

$$T_{k,k-1} = \arg\min_{T} \sum_{i} \| I_k(\boldsymbol{u}'_i) - I_{k-1}(\boldsymbol{u}_i) \|_{\sigma}^{2}$$

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

✓ 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

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

LSD-SLAM [Engel, ECCV'14]

- **Direct** (photometric error) **+ Semi-Dense** formulation
  - 3D geometry represented as semi-dense depth maps
  - Minimizes **photometric error**
  - **Separateley** optimizes poses & structure

Download from https://vision.in.tum.de/research/vslam/lsdslam

- Includes:
  - **Loop closing**
  - **Relocalization**
  - Final optimization

- **Real-time (30Hz)**

[Engel, Schoeps, Cremers, LSD-SLAM: Large-scale Semi-Dense SLAM, ECCV'14]

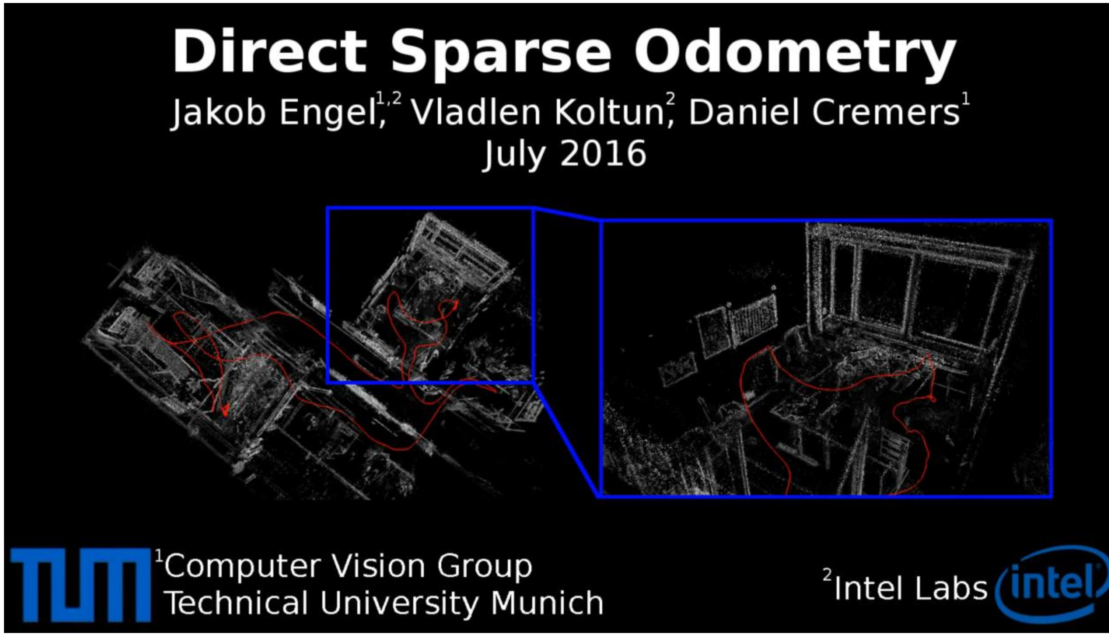
DSO [Engel, PAMI'17]

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

- **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

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

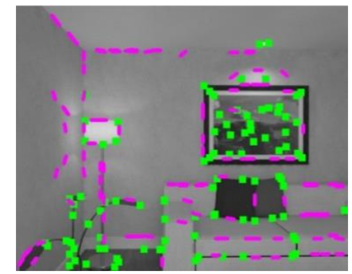
- **Real-time (30Hz)**



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

SVO [Forster, ICRA'14, TRO'17]

Edgelet **Corner**



- **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
- **SVO 2.0 includes**
  - Fish-eye & Omni cameras
  - Multi-camera systems

Meant for high speed!
- **400 fps** on i7 laptops
- **100 fps** on smartphone PC

Download from http://rpg.ifi.uzh.ch/svo2.html

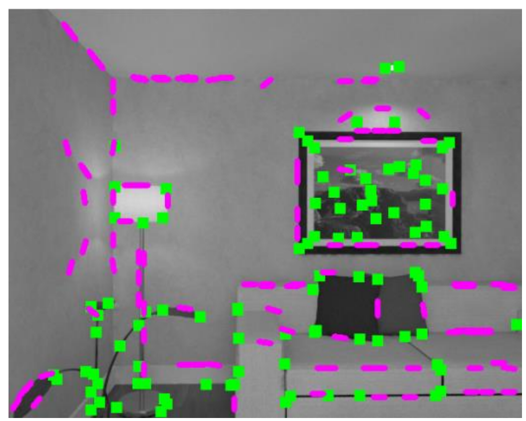
[Forster, Pizzoli, Scaramuzza, SVO: Fast, Semi-Direct Visual Odometry ICRA'14, TRO'17]

Comparison SVO, DSO, ORB-SLAM, LSD-SLAM [Forster, TRO'17]

Comparison is done on public benchmarks:

- EUROC-MAV
- ICL-NUIM (synthetic)
- TUM-RGBD

EUROC



ICL-NUIM

TUM-RGBD

[Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO'17]

Accuracy EUROC dataset [Forster, TRO'17]

	Monocular			
	SVO (edgelets + prior)	ORB-SLAM (no loop, real-time)	DSO (real-time)	LSD-SLAM (no loop-closure)
Machine Hall 01	0.10	0.61	0.05	0.18
Machine Hall 02	0.12	0.72	0.05	0.56
Machine Hall 03	0.41	1.70	0.26	2.69
Machine Hall 04	0.43	6.32	0.24	2.13
Machine Hall 05	0.30	5.66	0.15	0.85
Vicon Room 1 01	0.07	1.35	0.47	1.24
Vicon Room 1 02	0.21	0.58	0.10	1.11
Vicon Room 1 03	×	0.63	0.66	×
Vicon Room 2 01	0.11	0.53	0.05	×
Vicon Room 2 02	0.11	0.68	0.19	×
Vicon Room 2 03	1.08	1.06	1.19	×

TABLE I: Absolute translation errors (RMSE) in meters of the EUROC dataset after translation and scale alignment with the ground-truth trajectory and averaging over five runs. Loop closure detection and optimization was deactivated for ORB and LSD-SLAM to allow a fair comparison with SVO. The results of ORB-SLAM and DSO were obtained from [42].

[Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO'17]

Processing times of SVO, LSD-SLAM, ORB-SLAM

	Mean	St.D.	CPU@20 fps
SVO Mono	2.53	0.42	55 ±10%
ORB Mono SLAM (No loop closure)	29.81	5.67	187 ±32%
LSD Mono SLAM (No loop closure)	23.23	5.87	236 ±37%
DSO	20.12	4.03	181 ±27%

TABLE II: The first and second column report mean and standard devitation of the processing time in milliseconds on a laptop with an Intel Core i7 (2.80 GHz) processor. Since all algorithms use multi-threading, the third column reports the average CPU load when providing new images at a constant rate of 20 Hz.

[Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO'17]

Processing Times of SVO

- **Laptop** (Intel i7, 2.8 GHz): up to 400 fps
- **Smartphone**, ARM Cortex-A9, 1.7 GHz (Odroid): Up to 100 fps

Timing results on an Intel Core i7 (2.80 GHz) laptop processor:

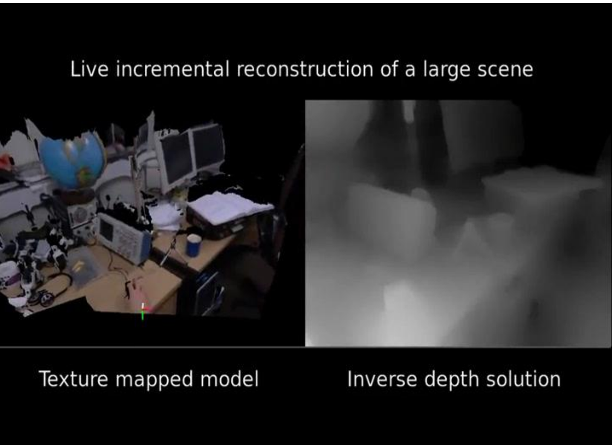
	Thread	Intel i7 [ms]
Sparse image alignment	1	0.66
Feature alignment	1	1.04
Optimize pose & landmarks	1	0.42
Extract features	2	1.64
Update depth filters	2	1.80

[Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO'17]

Direct Methods: Dense vs Semi-dense vs Sparse [TRO'16]

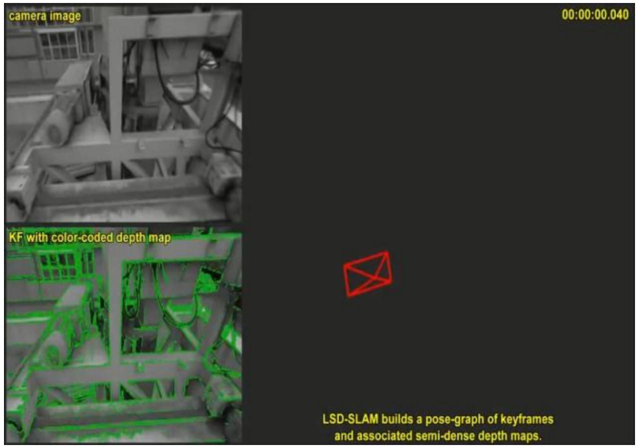
Dense	Semi-Dense	Sparse

Dense



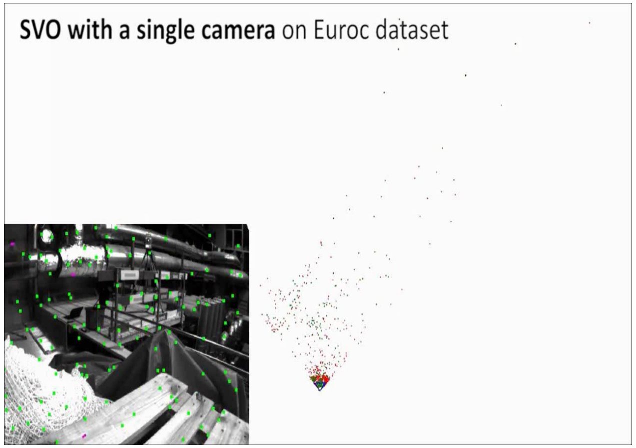
DTAM [Newcombe '11] REMODE [Pizzoli'14]
300'000+ pixels

Semi-Dense



LSD-SLAM [Engel'14]
~10,000 pixels

Sparse

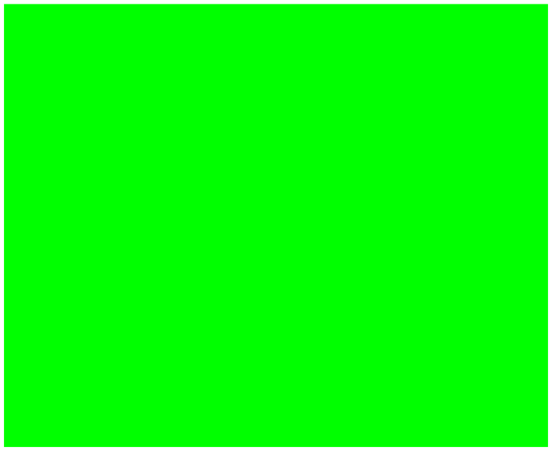


SVO [Forster'14]
100-200 x 4x4 patches ≅ 2,000 pixels

[Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO'17]

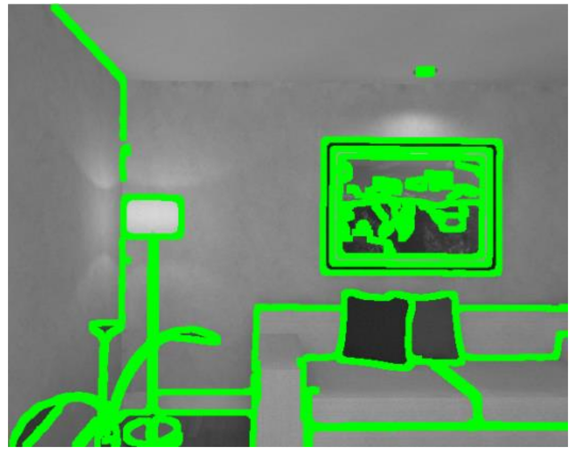
Direct Methods: Dense vs Semi-dense vs Sparse [TRO’16]

Dense



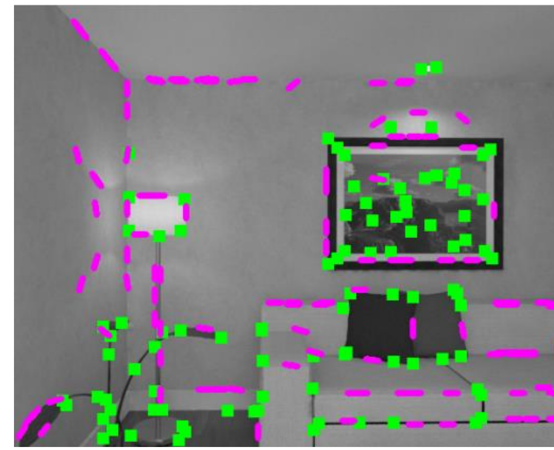
DTAM [Newcombe ‘11] REMODE [Pizzoli’14]
300’000+ pixels

Semi-Dense



LSD-SLAM [Engel’14]
~10,000 pixels

Sparse

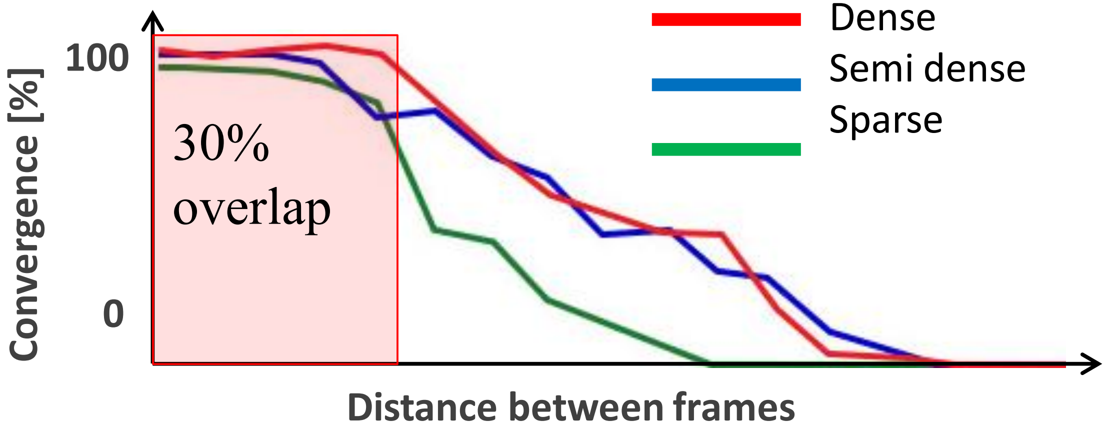


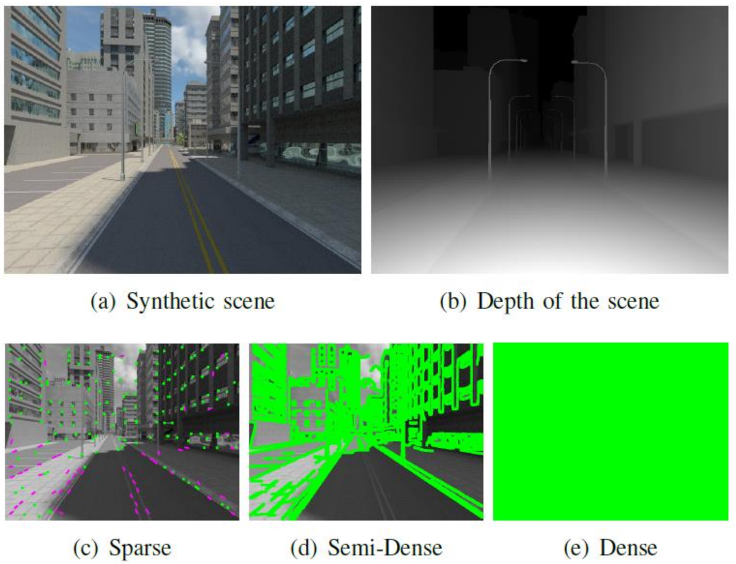
SVO [Forster’14]
100-200 x 4x4 patches ≅ 2,000 pixels

[Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO’17]

Direct Methods: Dense vs Semi-dense vs Sparse [TRO'16]

Robustness to motion baseline (computed from 1,000 Blender simulations)





Images from the synthetic Multi-FOV Zurich Urban Dataset

- **Dense** and **Semi-dense behave similarly**
  - weak gradients are not informative for the optimization)
- Dense only useful with **motion blur** and **defocus**
- **Sparse** methods behave equally well for image **overlaps up to 30%**

- [Forster, et al., SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO'17]
- Multi-FOV Zurich Urban Dataset: http://rpg.ifi.uzh.ch/fov.html

Position error: 5 mm, height: 1.5 m – Down-looking camera

Speed: 4 m/s, height: 3 m – Down-looking camera

Robustness to dynamic scenes (down-looking camera)

Automatic recovery from aggressive flight [ICRA'15]

[ICRA'10-17, AURO'12, RAM'14, JFR'15, RAL'17]

Tech Transfer activities

Parrot: Autonomous Inspection of Bridges and Power Masts

Parrot senseFly

Albris drone

Dacuda 3D (now Magic Leap)

- Fully immersive VR (running on iPhone)
- Powered by SVO

Dacuda's 3D divison

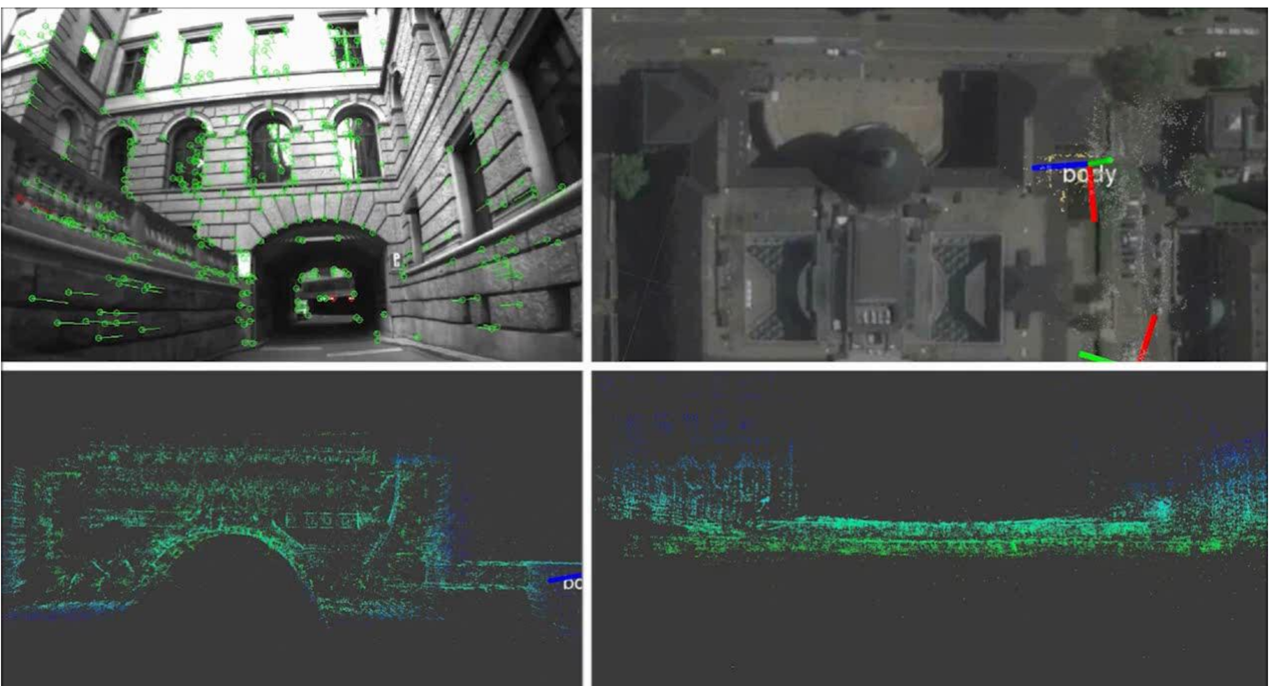
magic leap

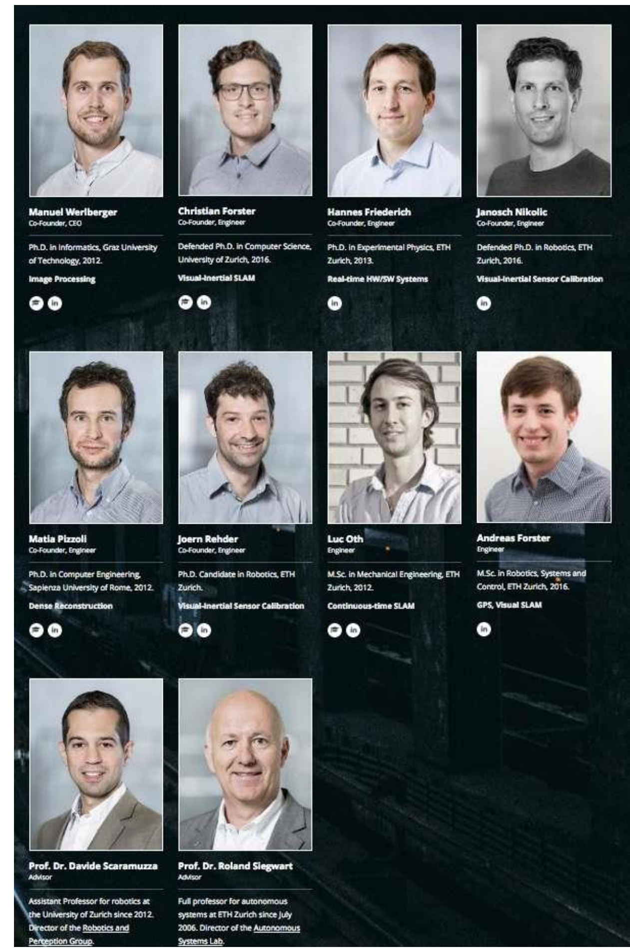
Zurich-Eye, first Wyss Zurich project

Vision-based Localization and Mapping Solutions for Mobile Robots

Created in Sep. 2015, **became Facebook-Oculus Zurich in Sep. 2016**

The Zurich Eye team is behind the new Oculus Santa Cruz





Zurich-Eye, first Wyss Zurich project

Vision-based Localization and Mapping Solutions for Mobile Robots

Created in Sep. 2015, **became Facebook-Oculus Zurich in Sep. 2016**

The Zurich Eye team is behind the new Oculus Santa Cruz