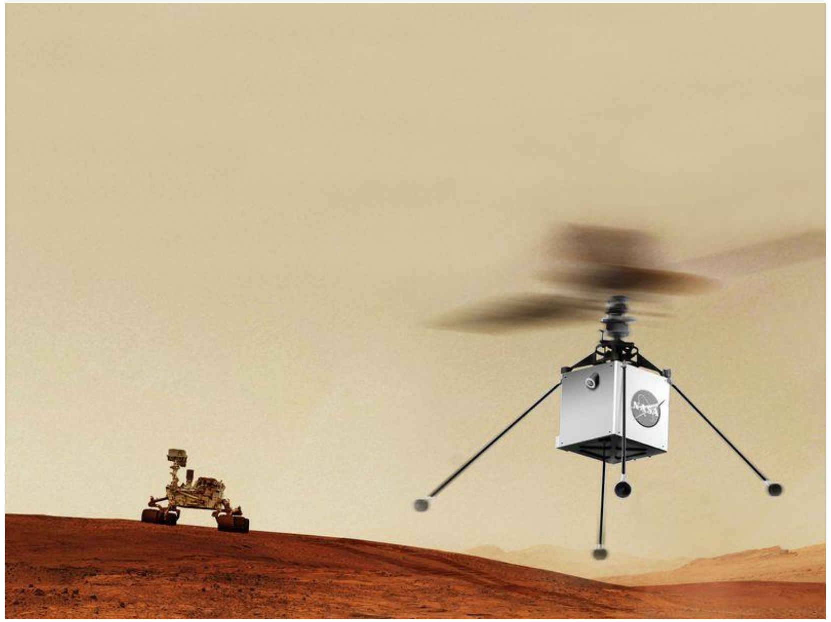
Vision Algorithms for Mobile Robotics

Lecture 10
Multiple View Geometry 4

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

Next week, seminar by NASA JPL

- When: Thursday December 2nd at 8:00 am followed by Lecture 11
- Title: “**Vision-Based Navigation for Mars Helicopters**”
- Who: Dr. Jeff Delaune: https://www-robotics.jpl.nasa.gov/people/Jeff_Delaune/

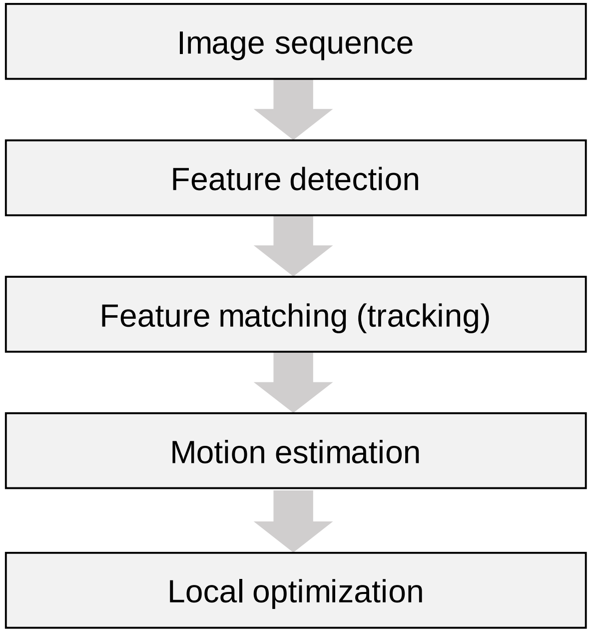


Jet Propulsion Laboratory
California Institute of Technology

Lab Exercise – Today

Intermediate VO integration for mini projects:

- problem statement
- details about what can/needs to be done
- we will show some of best examples from last years
- we will go through FAQ such as what can be added to get up to +0.5 mark

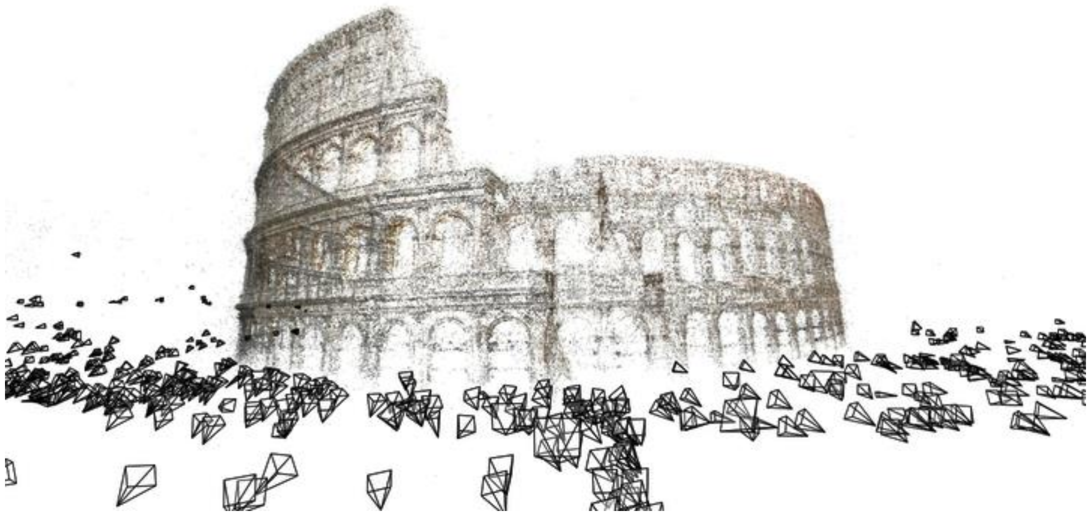


n-View Structure From Motion

- Compute initial structure and motion using either:
  - **Hierarchical SFM**
  - **Sequential SFM** $\rightarrow$ Visual Odometry (VO)
- Refine simultaneously structure and motion through BA

Hierarchical SFM applied to random internet images

- Reconstruction from 150,000 images from Flickr associated with the tags “Rome”
- 4 million 3D points. Cloud of 496 computers. 21 hours of computation!

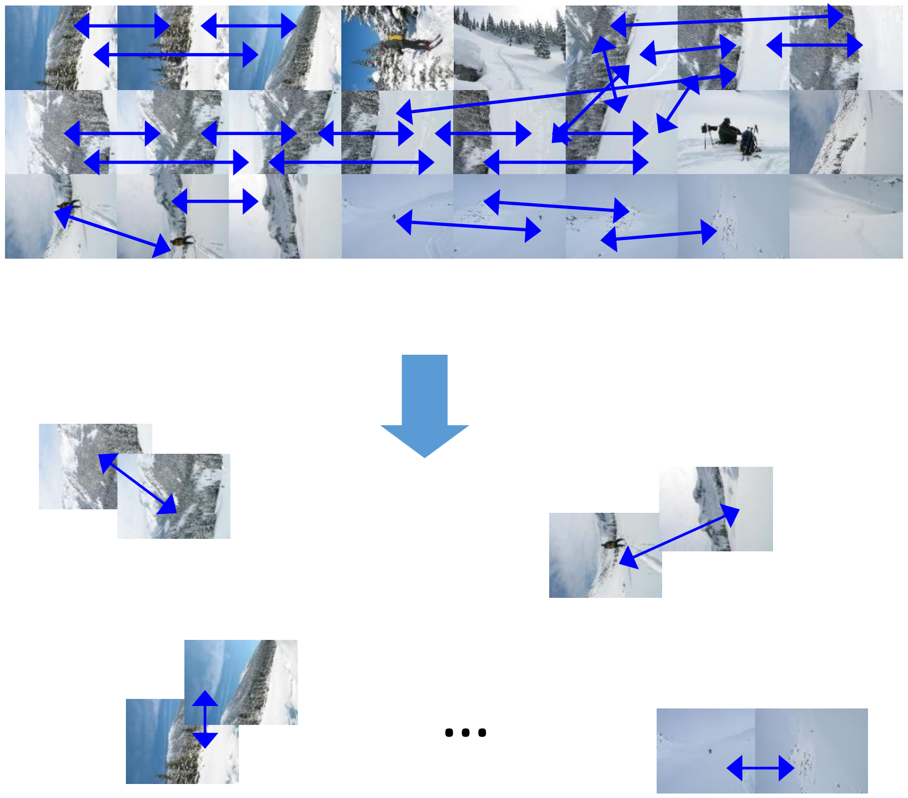


Agarwal, Snavely, Simon, Seitz, Szeliski, *Building Rome in a Day*, International Conference on Computer Vision (ICCV), 2009. [PDF, code, datasets]
Most influential paper of 2009

State of the art software: [COLMAP]

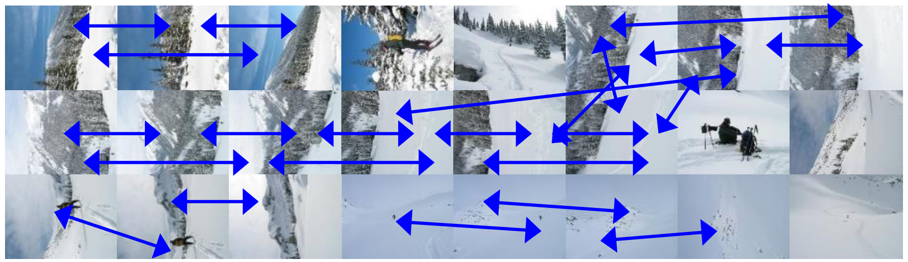
Hierarchical SFM

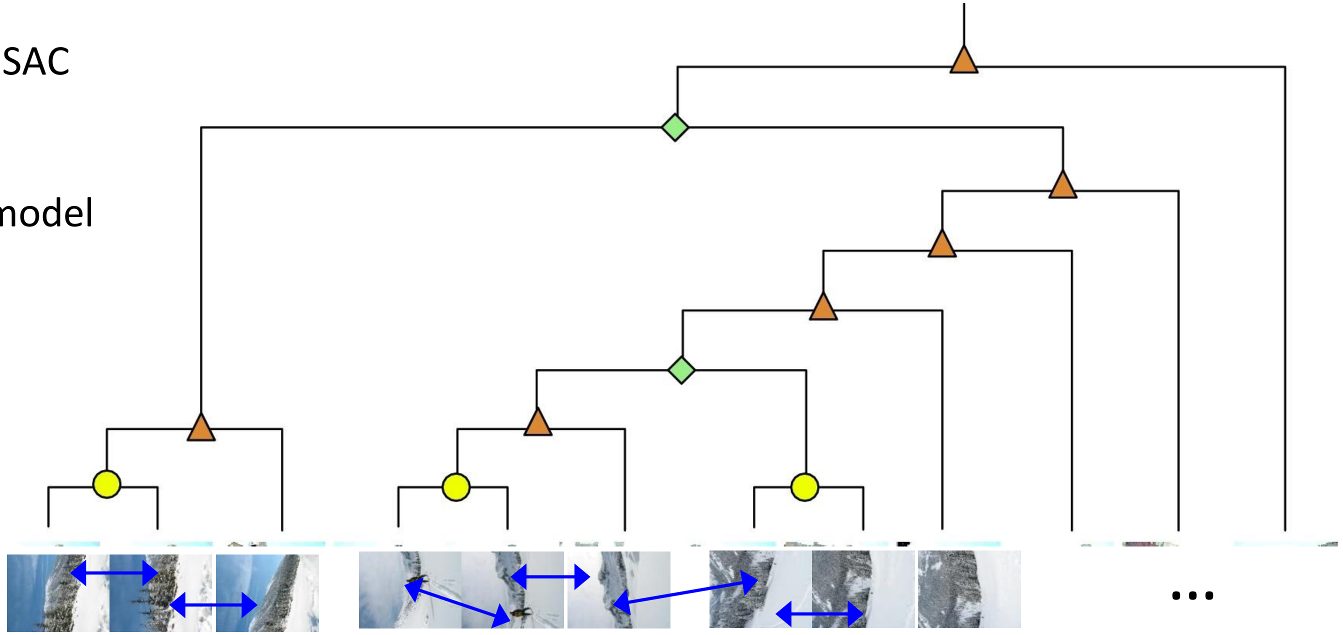
1. Extract and match features between nearby frames
2. Build clusters consisting of 2 nearby frames



Hierarchical SFM

1. Extract and match features between nearby frames
2. Build clusters consisting of 2 nearby frames
3. Extract topological tree (e.g., count number of SIFT matches)
4. Start from the terminal nodes
    1. Compute 2-view SFM and build 3D model (point cloud)
5. Iterate according to tree structure:
    1. Merge new view by running 3-point RANSAC between 3D model and 3rd view
    2. Merge near-by models using by running again 3-point RANSAC between one 3D model and one view of the other 3D model
    3. Bundle adjust





The circle ○ corresponds to the creation of a stereo-model, the triangle △ corresponds to applying PNP, the diamond ⋄ corresponds to a fusion of two partial independent models.

n-View Structure From Motion

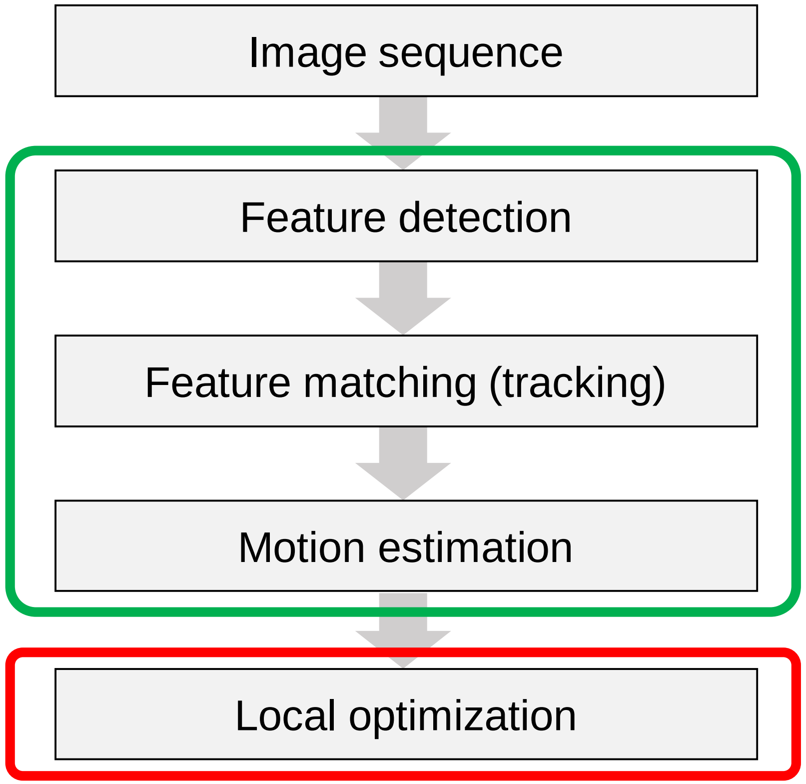
- Compute initial structure and motion using either:
    - **Hierarchical SFM**
    - **Sequential SFM** $\rightarrow$ Visual Odometry (VO)
- Refine simultaneously structure and motion through BA

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 (**mapping**)
  - Refine structure and motion through Bundle Adjustment (BA) (**optimization**)

VO Flow Chart: review (Lecture 01)

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



Front-end: outputs the *relative pose* between the *last two frames*

Back-end: “*adjusts*” the relative poses among *multiple recent frames*

VO Flow Chart: review (Lecture 01)

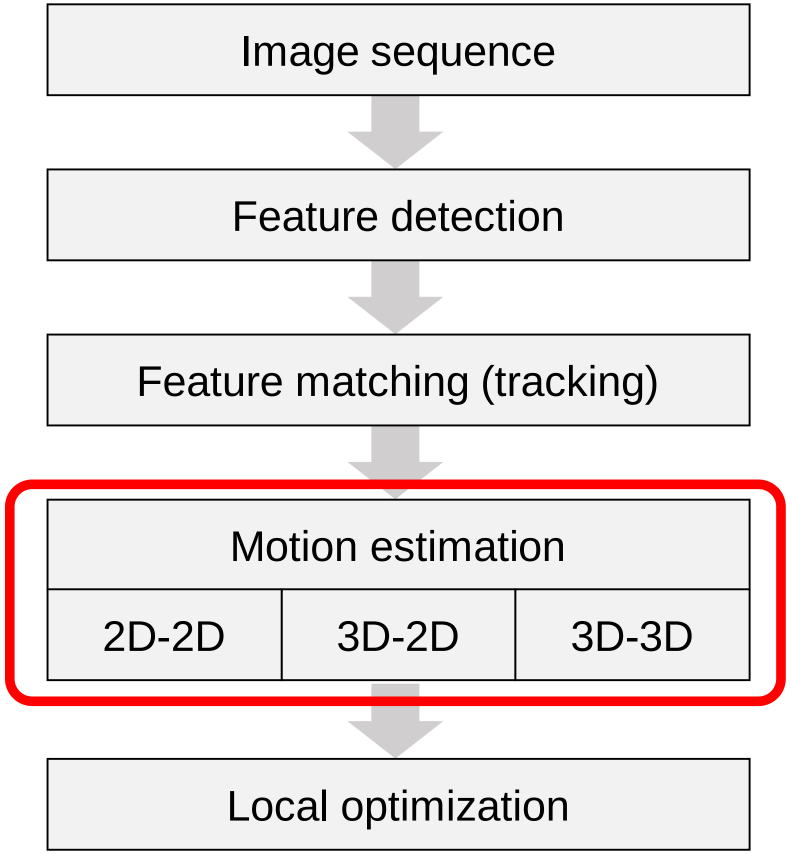
- VO computes the camera path incrementally (pose after pose)

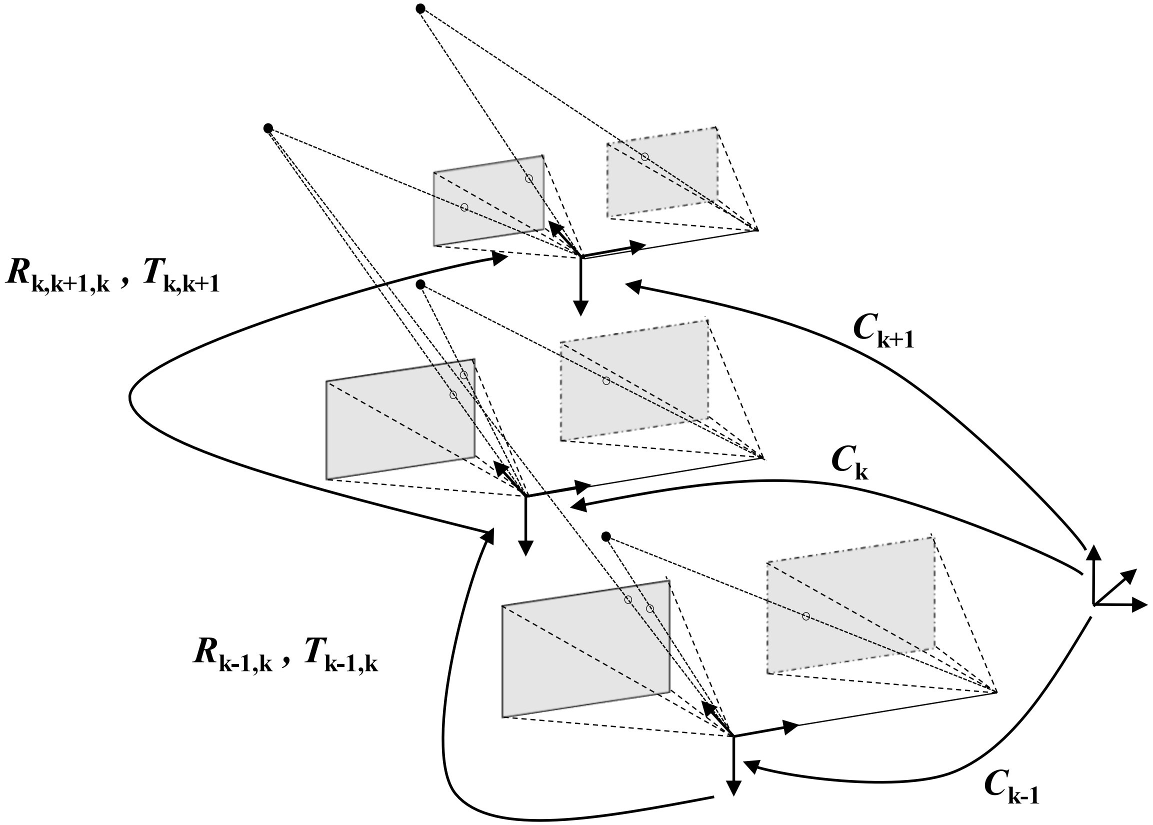
Image sequence
↓
Feature detection
↓
Feature matching (tracking)
↓
Motion estimation
↓
Local optimization

Features tracked over multiple recent frames overlaid on the last frame

VO Flow Chart: review (Lecture 01)

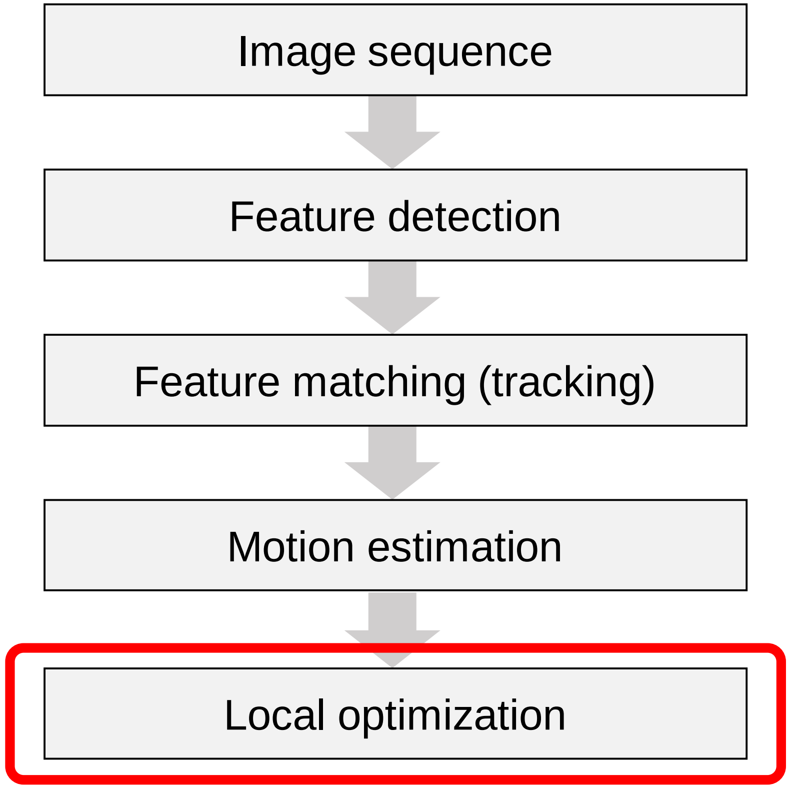
- VO computes the camera path incrementally (pose after pose)





VO Flow Chart: review (Lecture 01)

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



Example: Bundle Adjustment:

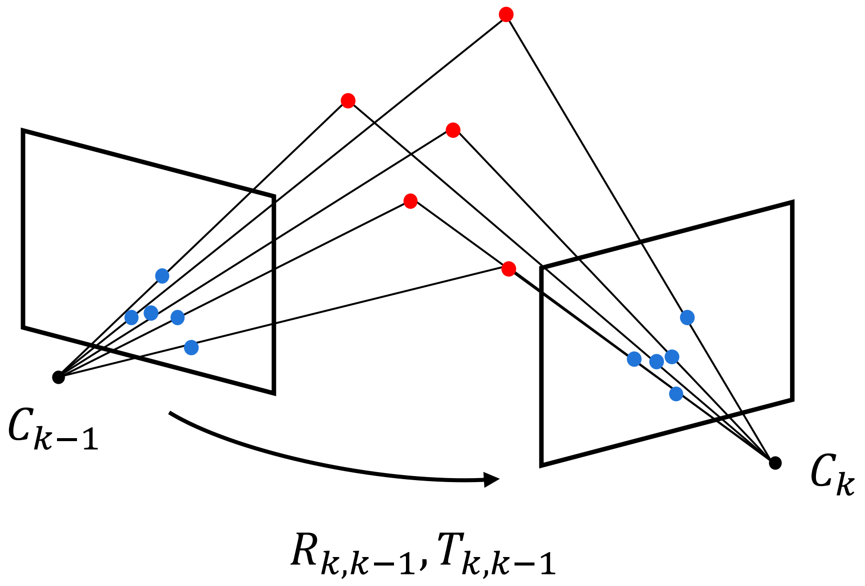
$$P^i, C_2, \dots, C_k = argmin_{P^i, C_2, \dots, C_k} \sum_{k=1}^{n} \sum_{i=1}^{N} \left\| p_k^i - \pi\left(P^i, K_k, C_k\right) \right\|^2$$

Or Pose-Graph Optimization (see later)

2D-to-2D (already seen: Lecture 08)

Motion from 2D-to-2D feature correspondences

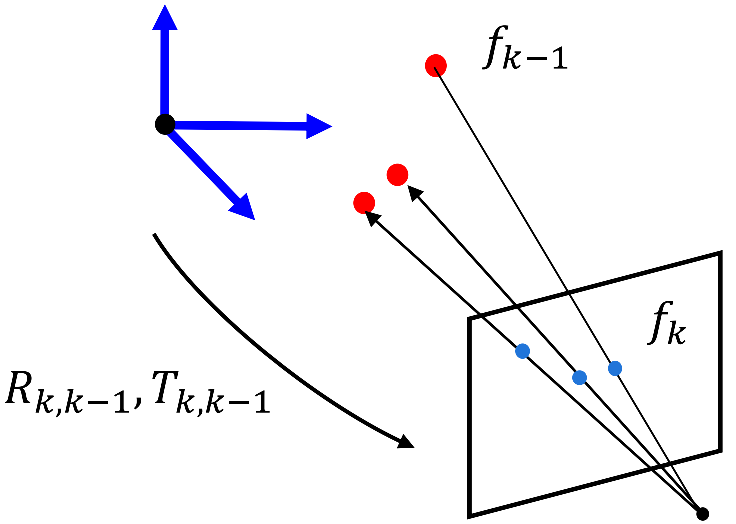
- Both feature **correspondences** f_{k-1} and f_k are specified in **image coordinates (2D)**
- The **minimal-case** solution involves **5** feature correspondences
- Popular algorithms: **5- and 8-point algorithms**



3D-to-2D (already seen: Lecture 03)

Motion from 3D-to-2D feature correspondences (i.e., Perspective from n Points: PnP problem)

- f_{k-1} is specified in 3D and f_k in 2D
- **Minimal case**:
  - DLT algorithm: minimal case: 6 points from 3D objects, or 4 from planar objects
  - P3P algorithm: minimal case: 3 points (+1 for disambiguation)
  - EPNP algorithm: for more than 4 points

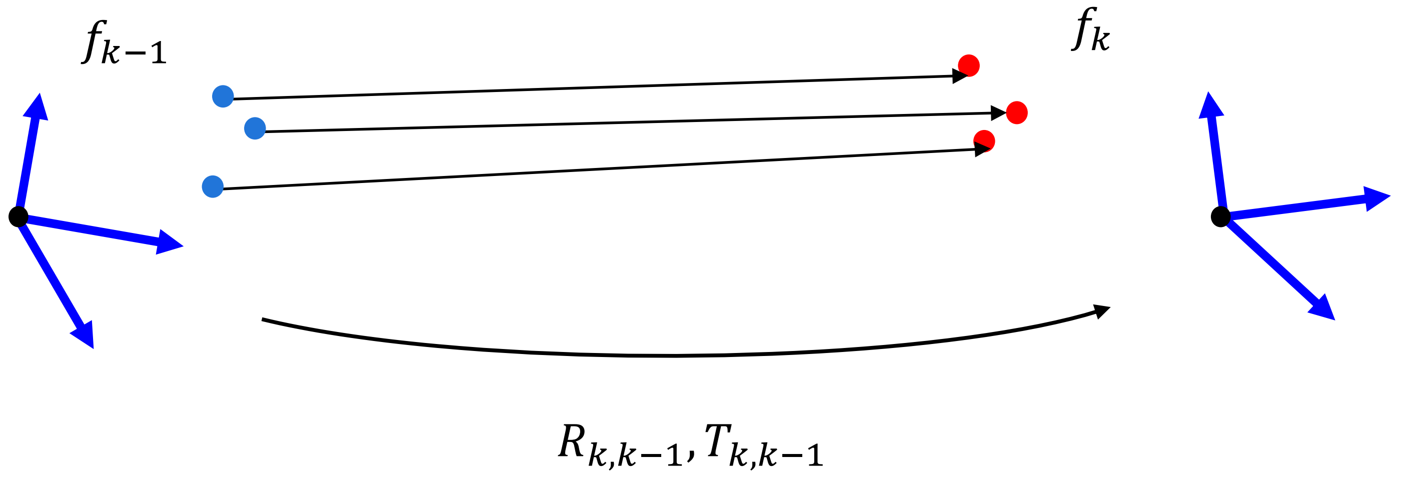


3D-to-3D

- **Motion from 3D-to-3D feature correspondences** (also known as point cloud registration problem)
- Both f_{k-1} and f_k are specified in 3D. To do this, it is necessary to first triangulate 3D points (e.g. use a stereo camera)
- The **minimal-case** solution involves **3 non-collinear** correspondences
- Popular algorithm: [Arun'87]
- Consists of solving the following system of equations with R and T as unknowns:

$$\begin{bmatrix} X^i_{k-1} \\ Y^i_{k-1} \\ Z^i_{k-1} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X^i_k \\ Y^i_k \\ Z^i_k \\ 1 \end{bmatrix}$$

where i is the feature ID.



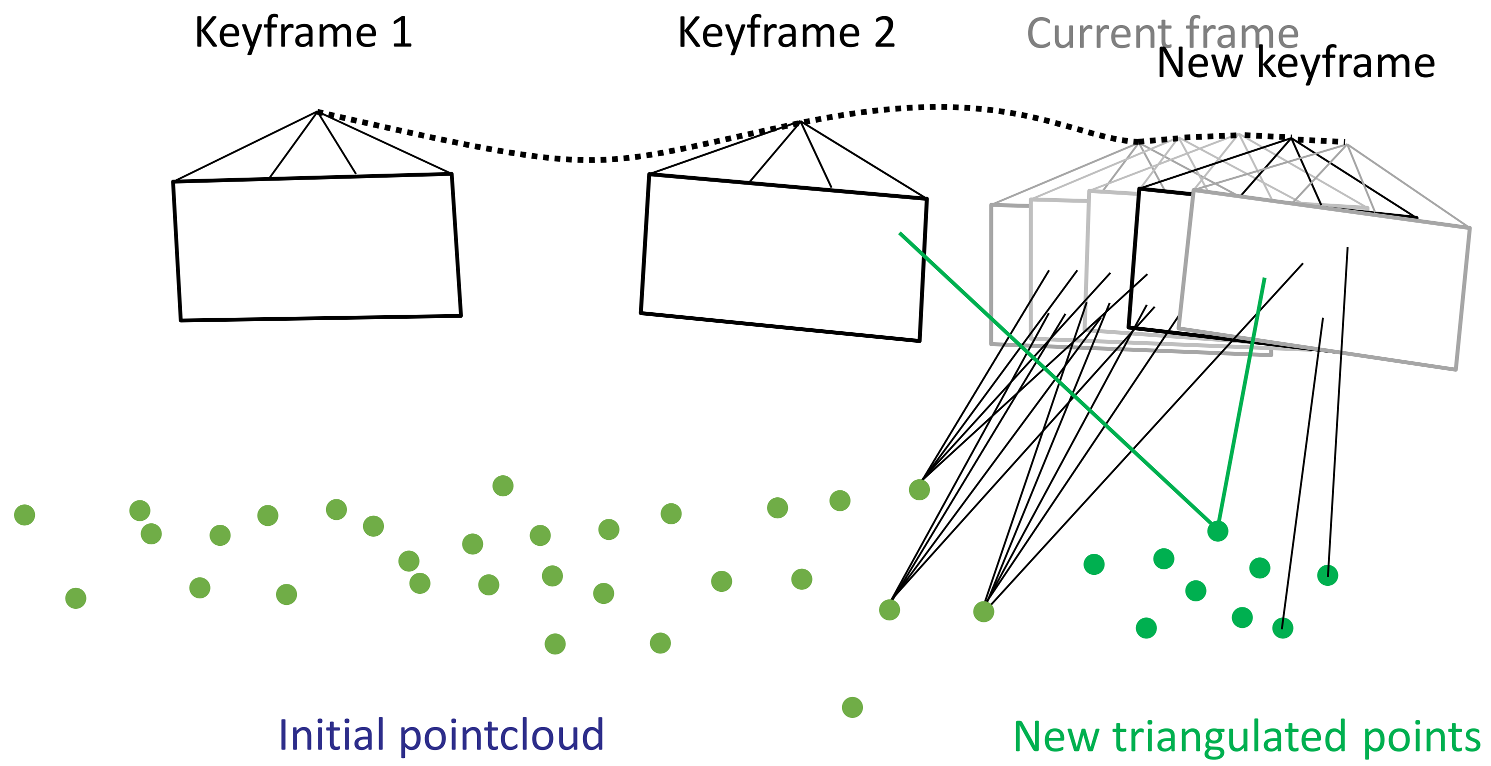
Arun, Huang, Blostein, "Least-Squares Fitting of Two 3-D Point Sets," Transactions on Pattern Analysis and Machine Intelligence (PAMI), 1987. PDF

Motion Estimation: Recap

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

Case Study: Monocular VO (i.e., single camera VO)

This pipeline was initially proposed in PTAM (Parallel Tracking & Mapping) [Klein, ISMAR'07]



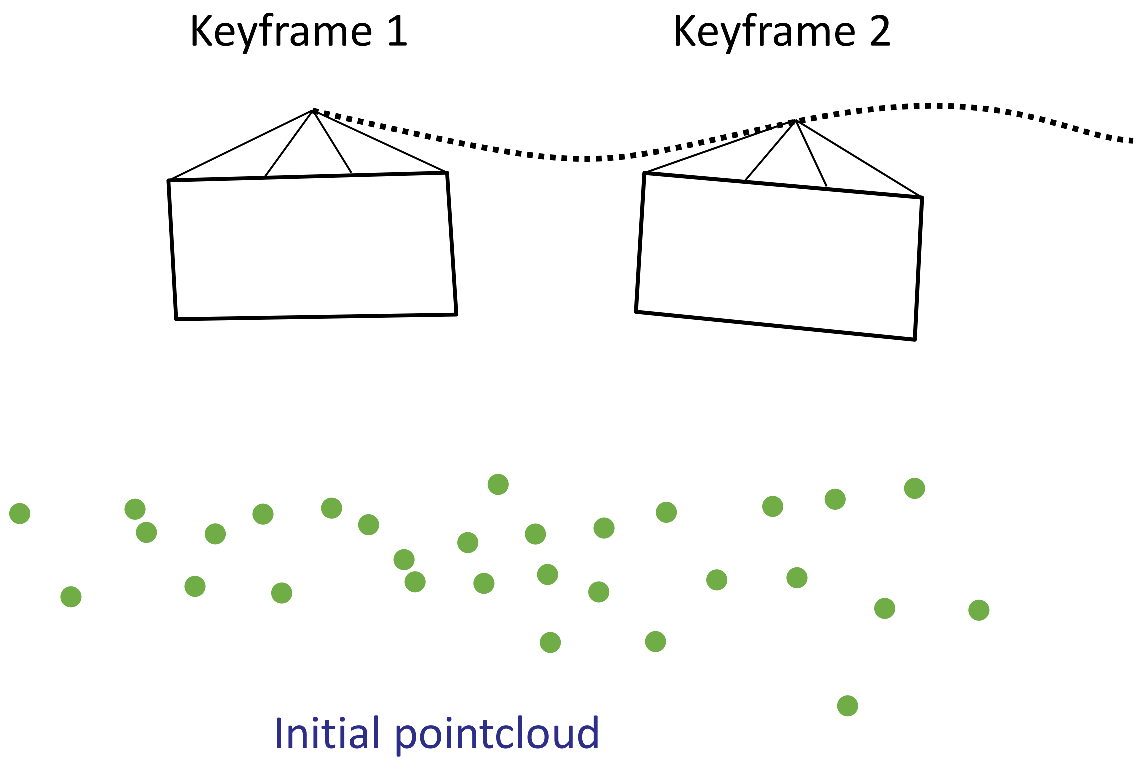
Klein, Murray, *Parallel Tracking and Mapping for Small AR Workspaces*, International Symposium on Mixed and Augmented Reality (ISMAR), 2007.
[PDF, code, videos](). **Best paper award**

Case Study: Monocular VO (i.e., single camera VO)

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

1. Bootstrapping (i.e., initialization)

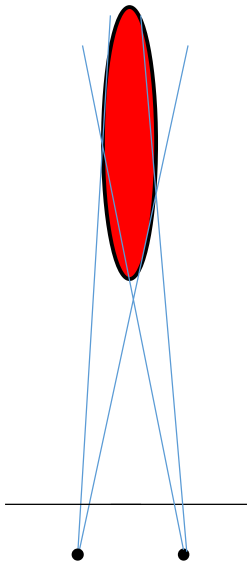
- Initialize structure and motion from 2 views: e.g., **5- or 8-point RANSAC**
- Refine structure and motion (**Bundle Adjustment**)
- How far should the two frames (i.e., keyframes) be?



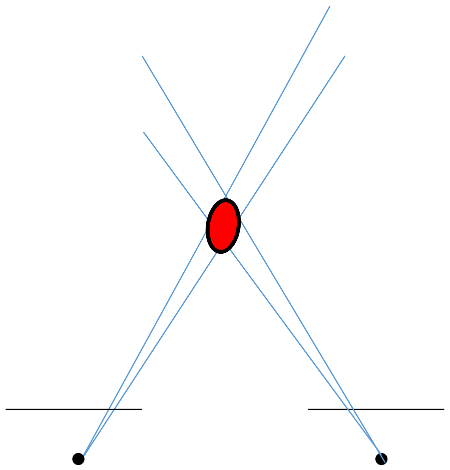
Case Study: Monocular VO (i.e., single camera VO)

2. Keyframe selection (i.e., skipping frames)

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



Small baseline → large depth uncertainty

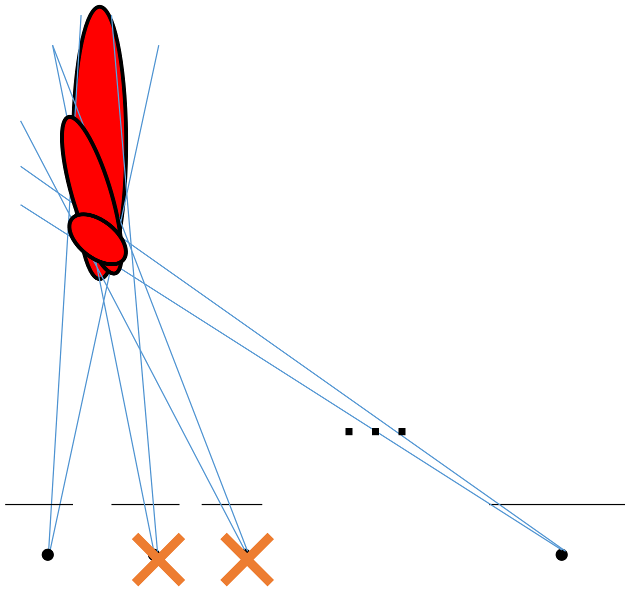


Large baseline → small depth uncertainty

Case Study: Monocular VO (i.e., single camera VO)

2. Keyframe selection (i.e., skipping frames)

- 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, normalized by the average distance from the scene, falls below a certain threshold. The selected frames are called ***keyframes***
- **Rule of the thumb:** add a keyframe when $\frac{\textit{keyframe distance}}{\textit{average-depth}} > \textit{threshold (~10-20 \%)}$



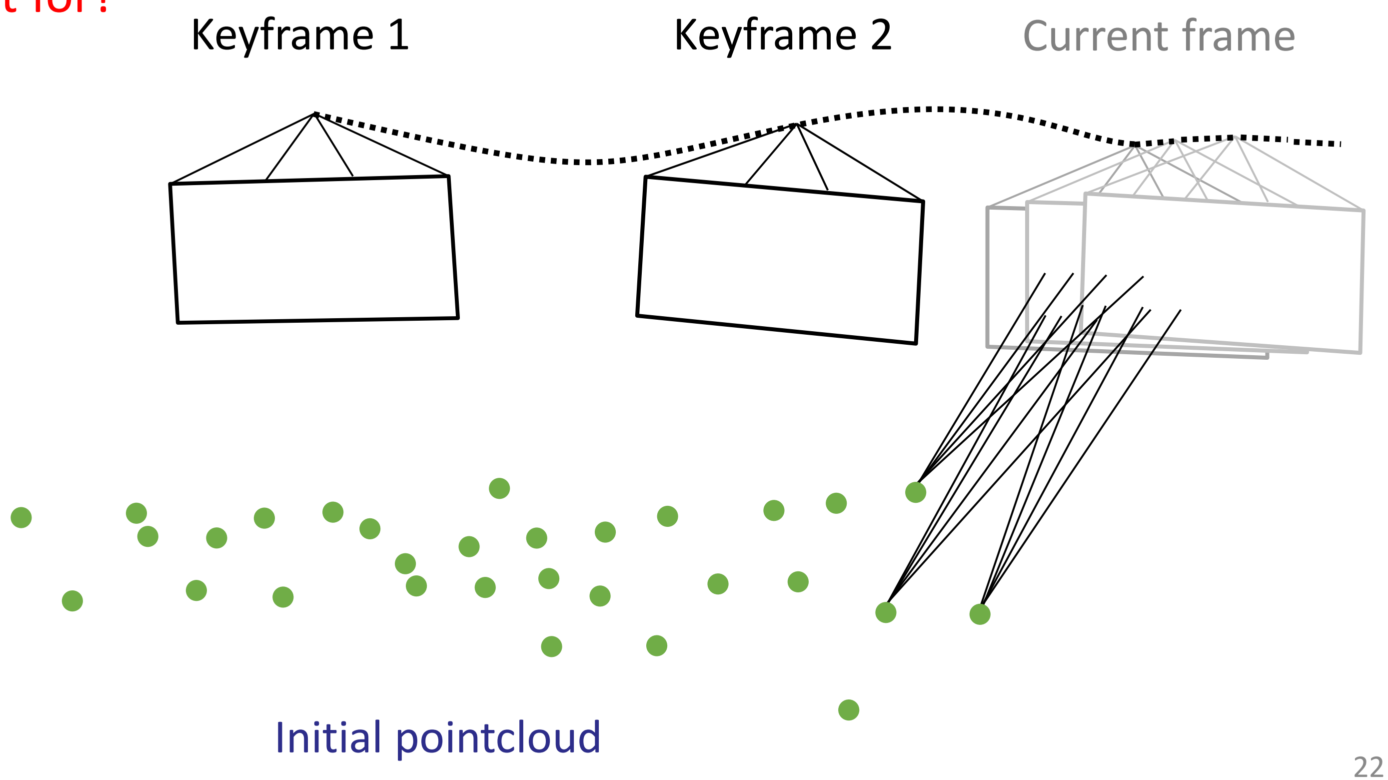
Where does this come from?

Case Study: Monocular VO (i.e., single camera VO)

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

3. Localization (i.e., pose estimation from a given point cloud)

- Given a 3D point cloud (map), determine the pose of each additional view
- What algorithm is used?
- How far from the last keyframe can we use it for?

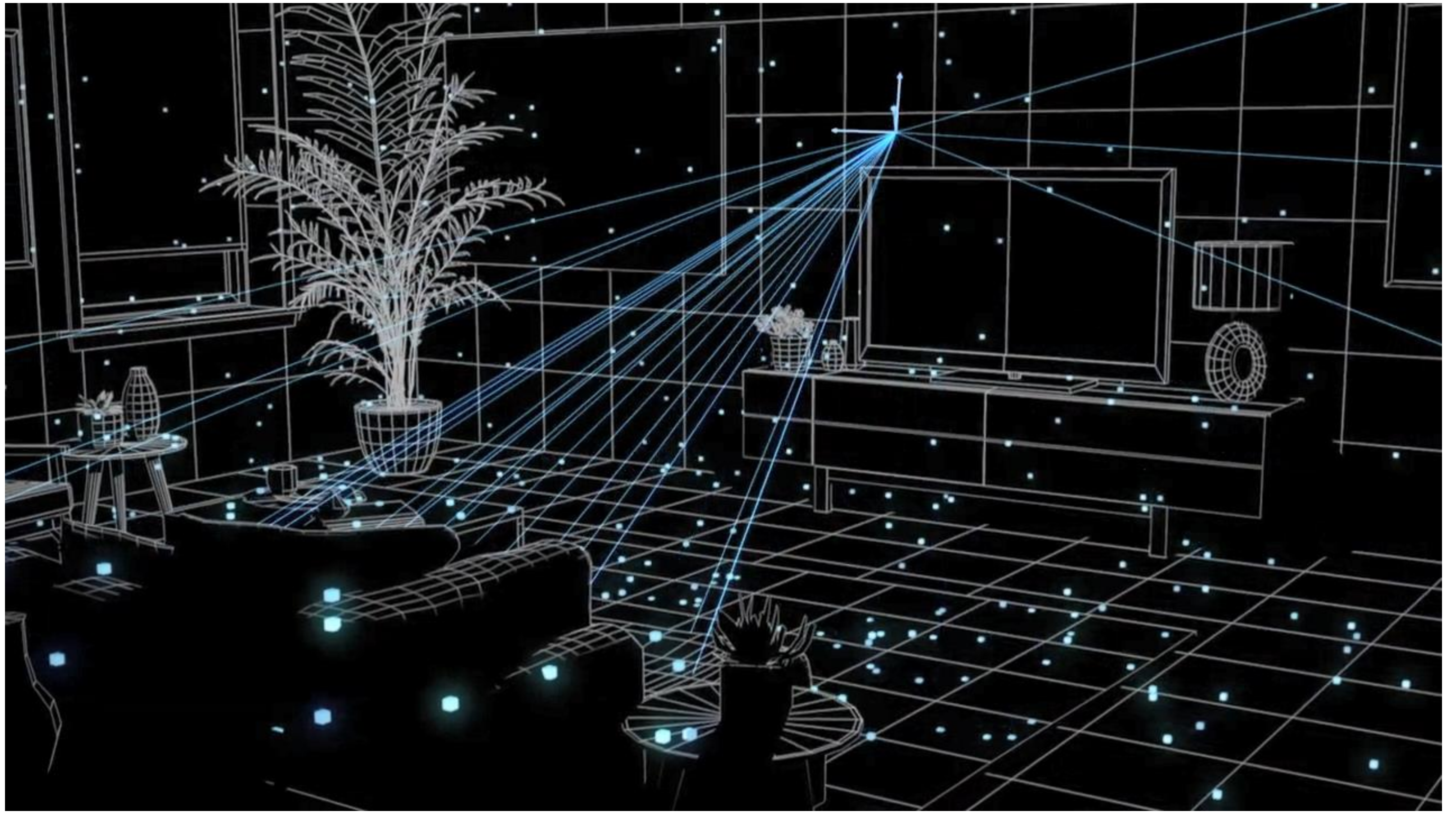


Case Study: Monocular VO (i.e., single camera VO)

3. Localization (i.e., pose estimation from a given point cloud)

- Given a 3D point cloud (map), determine the pose of each additional view

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

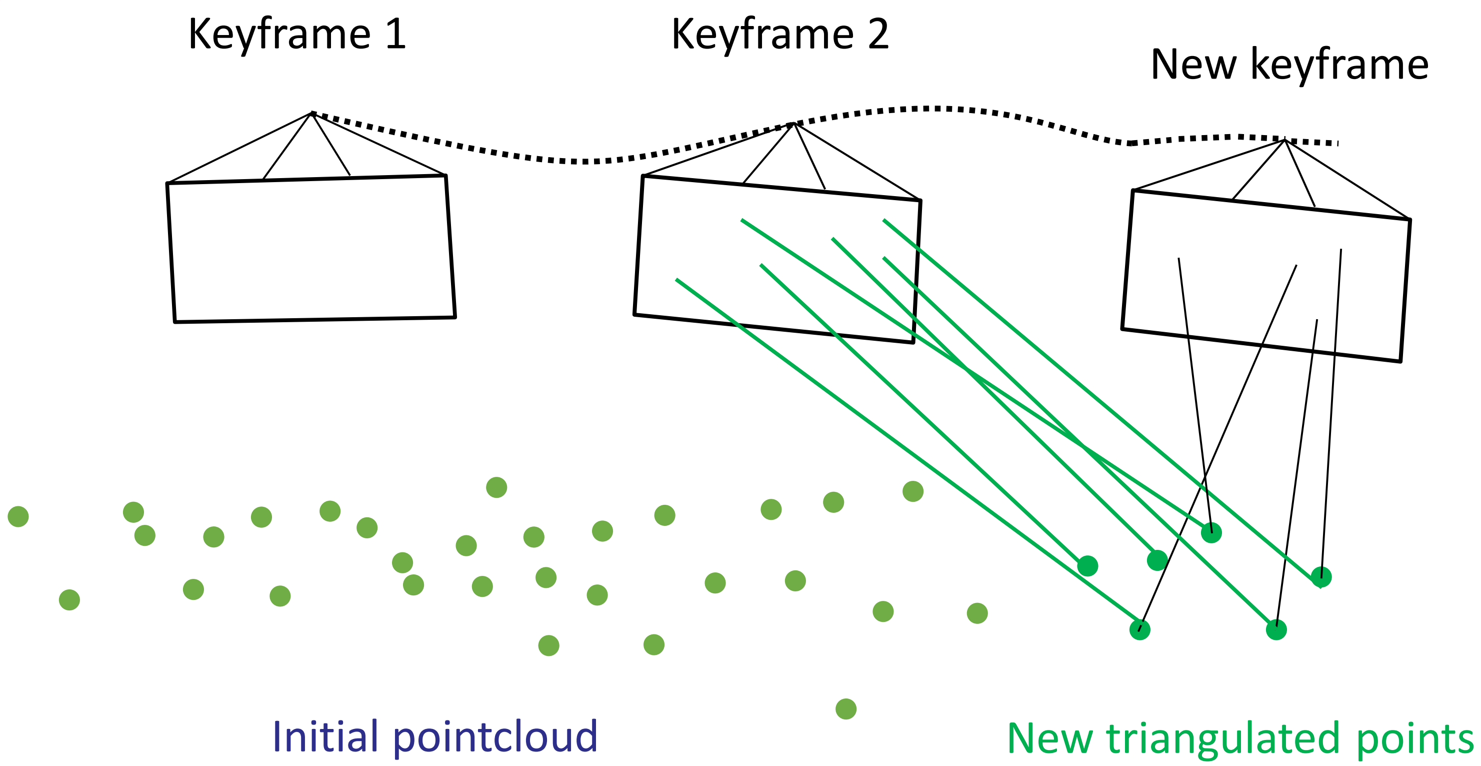


Video of Oculus Insight (the VIO used in Oculus Quest): built by former Zurich-Eye team, today Facebook Zurich.

Case Study: Monocular VO (i.e., single camera VO)

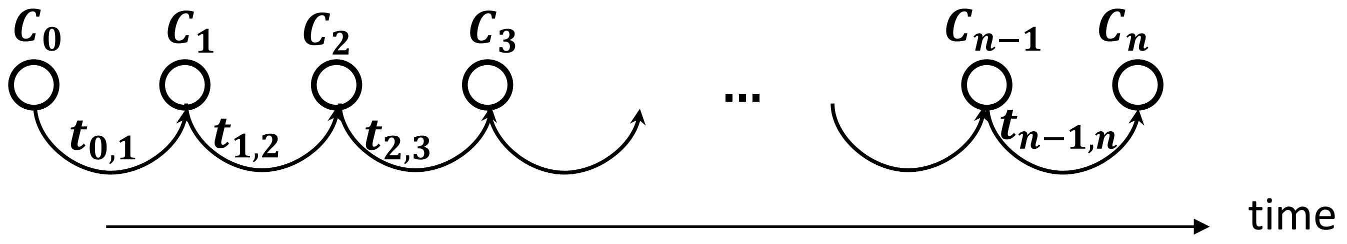
4. Extend Structure (i.e., mapping)

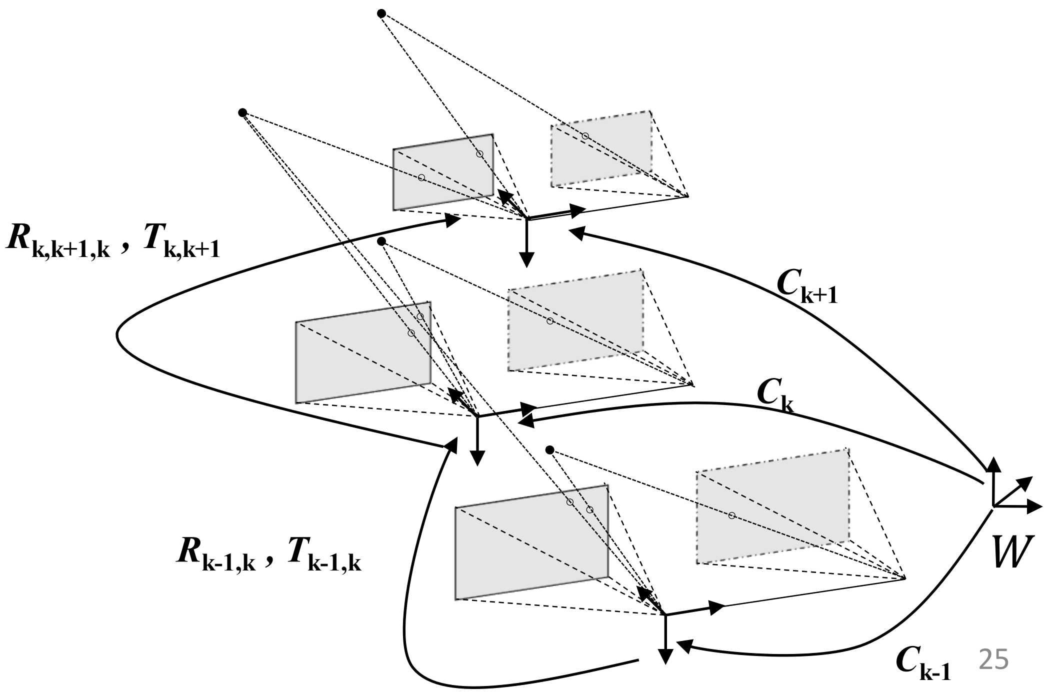
- Extract and triangulate new features
- Is it necessary to do this at every frame or can we just do it at keyframes?
- What are the pros and cons?



VO: putting all pieces together

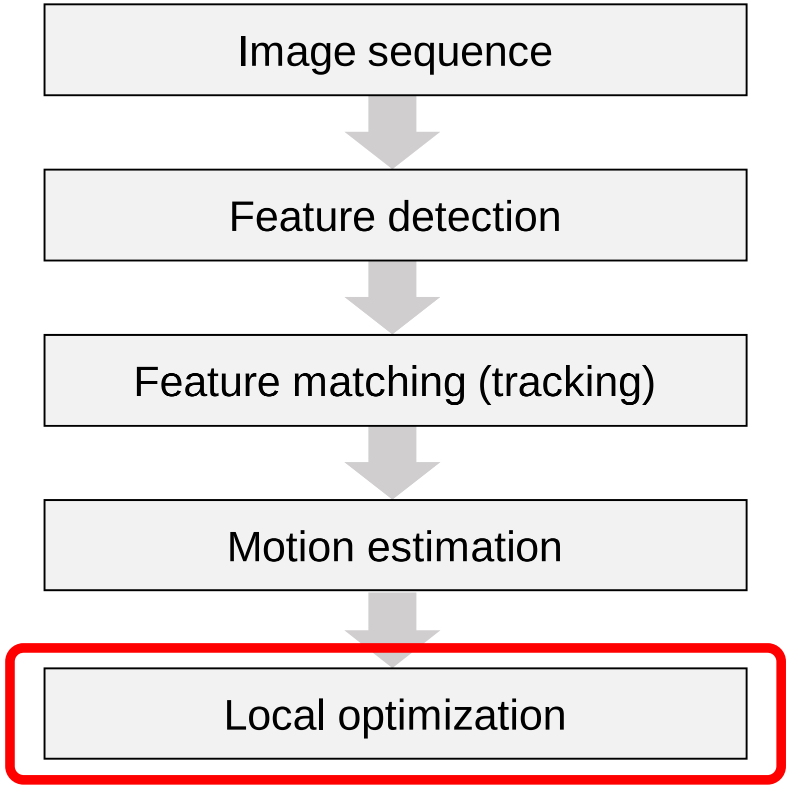
- Let the **relative motion** between image I_{k-1} and image I_k be: $t_{k-1,k} = \begin{bmatrix} R_{k-1,k} & T_{k-1,k} \\ 0 & 1 \end{bmatrix}$
- Let C_{k-1} be the **previous camera pose in the world reference frame**
- Then, the **current pose $\boldsymbol{C_k}$ in the world frame** is given by: $C_k = C_{k-1} t_{k-1,k}$

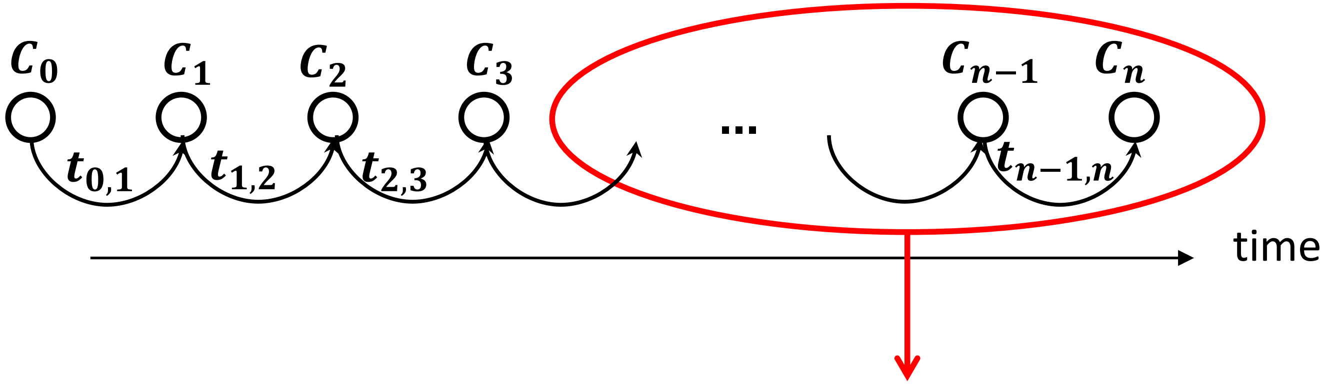




Local Optimization

VO flowchart:

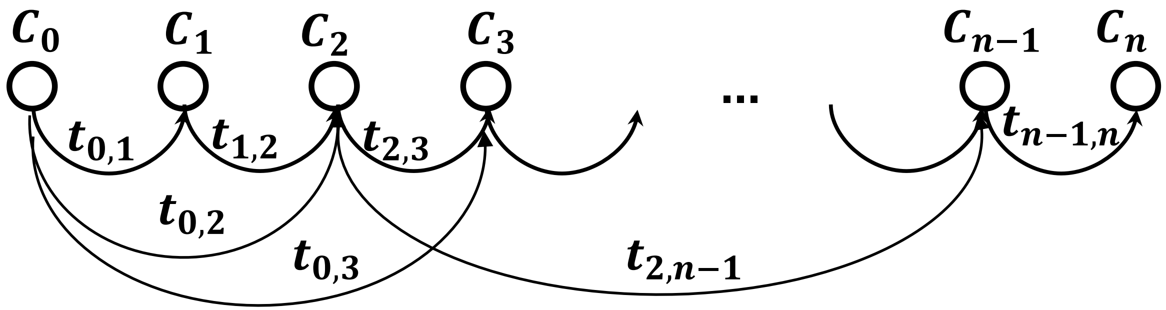




Sliding-window bundle adjustment
or Pose-Graph Optimization (see next slide)

Pose-Graph Optimization

- So far we assumed that the transformations are between consecutive frames

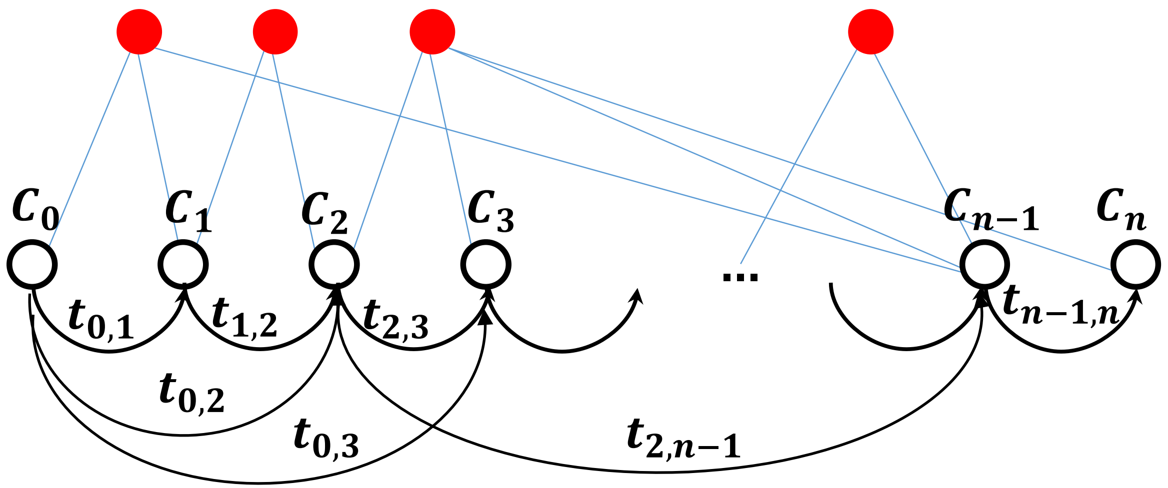


- However, transformations can also be computed between **non-adjacent frames:** $\boldsymbol{t_{j,i}}$ (e.g., when features from previous keyframes are still observed). They can be used as additional constraints to improve camera poses by solving:

$$\{C_1, \dots, C_n\} = argmin_{\{C_1,\dots,C_n\},} \sum_i \sum_j \left\| C_i - C_j t_{j,i} \right\|^2$$

- 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 exist: g2o, GTSAM, SLAM++, Google Ceres

Bundle Adjustment (BA)



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

$$P^i, C_1, \dots, C_n = argmin_{X^i, C_1, \dots, C_n,} \sum_{k=1}^{n} \sum_{i=1}^{N} \rho\left(p_k^i - \pi\left(P^i, K_k, C_k\right)\right)$$

- $\rho()$ is the **Huber or Tukey norm**
- Gauss-Newton or Levenberg-Marquadt are typically used to minimize it. For large graphs, efficient open-source tools exist: [g2o](), [GTSAM](), [SLAM++](), [Google Ceres]()

Bundle Adjustment vs Pose-graph Optimization

- BA is **more precise** than pose-graph optimization because it adds additional constraints (*landmark constraints*)
- But **more costly**: $O\left((qN + lm)^3\right)$ with N being the number of points, m the number of 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**)

More efficient BA algoritms have recently been developed:

[1] Demmel, Schubert, Sommer, Cremers, Usenko, Square Root Marginalization for Sliding-Window Bundle Adjustment, IEEE International Conference on Computer Vision (ICCV), 2021. Paper, Video, Code.

[2] Demmel, Sommer, Cremers, Usenko, Square Root Bundle Adjustment for Large-Scale Reconstruction, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. Paper, Video, Code.

Place Recognition

During VO, two problems can occur:

- **Relocalization problem:** camera pose estimation can fail due to:
    1. Feature **tracking can be lost** (due to occlusions, low texture, quick motion, illumination change)
    2. In case of monocular VO: **pure rotation followed by translation** (why?)

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

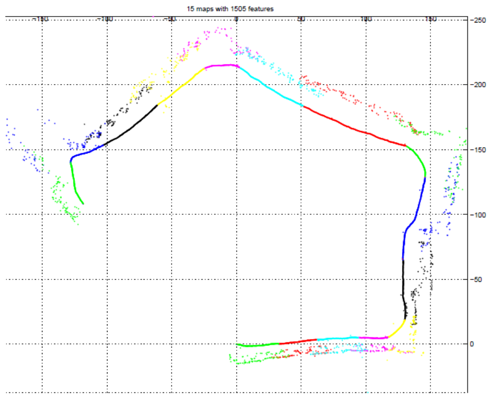
- **Loop closing problem**
    - When you go back to a previously mapped area:
        - **Loop closure detection**: to avoid map duplication
        - **Loop correction**: to compensate the accumulated drift
    - In both cases you need a place recognition technique

We will address place recognition in Lecture 12

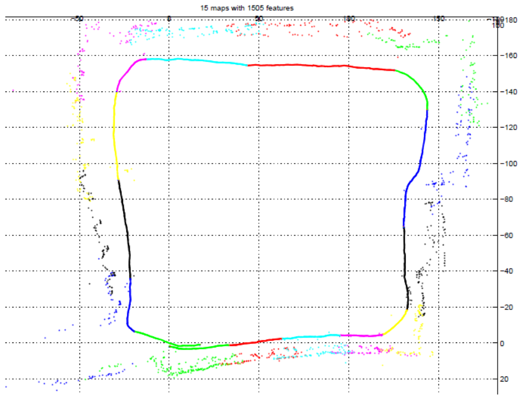
VO vs. Visual SLAM (recap from Lecture 01)

- **Visual Odometry**
  - Focus on incremental estimation
  - **Guarantees local consistency** (i.e., estimated trajectory is locally correct, but not globally, i.e. from the start to the end)

- **Visual SLAM** (Simultaneous Localization And Mapping)
  - **SLAM = visual odometry + loop detection & closure**
  - **Guarantees global consistency** (the estimated trajectory is globally correct, i.e. from the start to the end)



Visual odometry



Visual SLAM

Image courtesy of [Clemente et al., RSS'07]

Open Source Monocular VO and SLAM algorithms

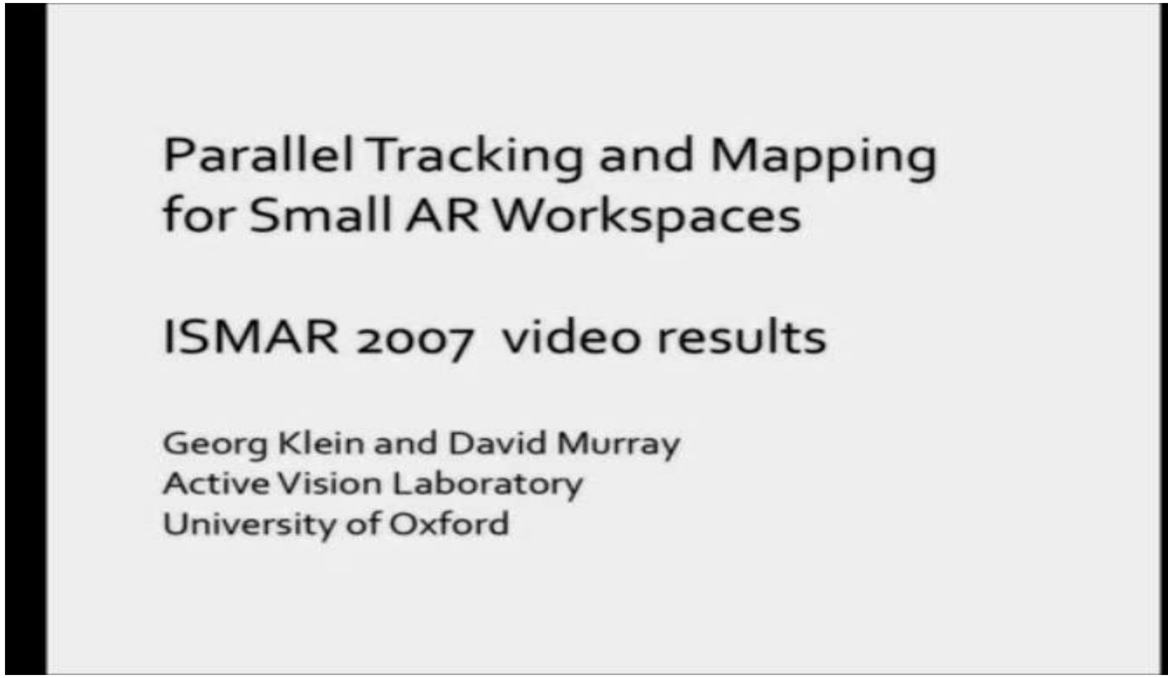
- PTAM
- ORB-SLAM
- SVO
- LSD-SLAM
- DSO

Indirect methods: Minimize the feature reprojection error

Direct methods: Minimize the feature photometric error

PTAM: Parallel Tracking and Mapping

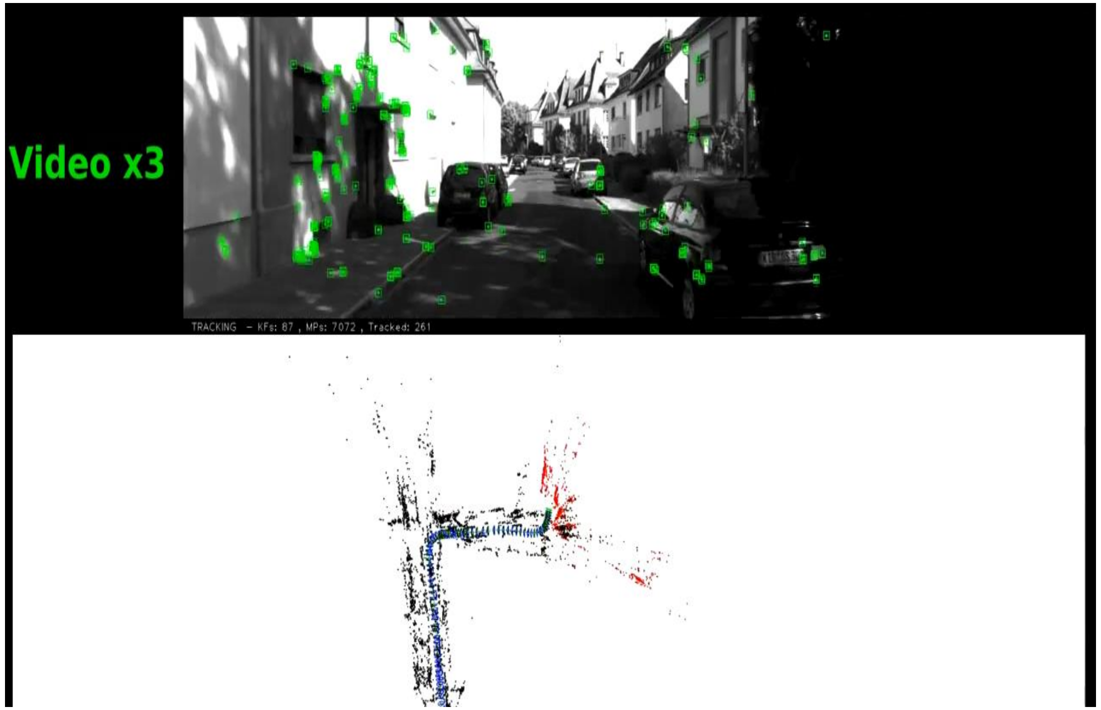
- Monocular only
- **Feature based**
  - FAST corners + patch descriptors
  - Minimizes reprojection error
  - **Jointly optimizes poses & structure** (sliding window BA)
- First to propose **keyframe-based VO**
- First to propose **alternation of localization** (i.e., camera tracking) and **mapping**
- Tracking and mapping running in **two independent threads**: updated map is used by localization thread asynchronously, as soon it becomes available
- Includes:
  - **Relocalization**
  - No global optimization, only local
- **Real-time (30Hz)**, however global optimization is not done in real time but asynchronously every once in a while



Klein, Murray, *Parallel Tracking and Mapping for Small AR Workspaces*, International Symposium on Mixed and Augmented Reality (ISMAR), 2007. PDF, code, videos. **Best paper award**

ORB-SLAM

- Supports both **monocular and stereo** cameras
- **Feature based**
  - FAST corners + ORB descriptors
  - ORB: binary descriptor, very fast to compute and match (Hamming distance)
  - **Jointly optimizes poses & structure** (sliding window BA)
- **Same workflow as PTAM** (keyframe based, alternation of localization and mapping as independent threads)
- Includes:
  - **Loop closing**
  - **Relocalization**
  - **Final optimization**
- **Real-time (30Hz)**, however global optimization is not done in real time but asynchronously every once in a while

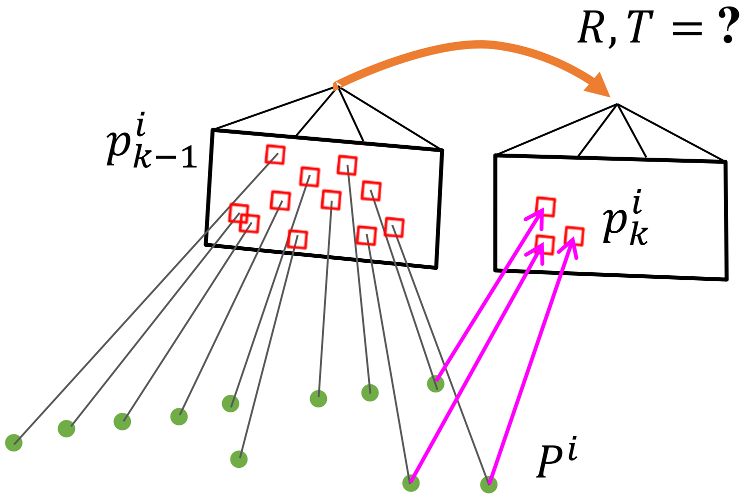


Mur-Artal, Montiel, Tardos, *ORB-SLAM: Large-scale Feature-based SLAM*, IEEE Transactions on Robotics (T-RO), 2015. PDF, code, videos.

Indirect vs Direct Methods

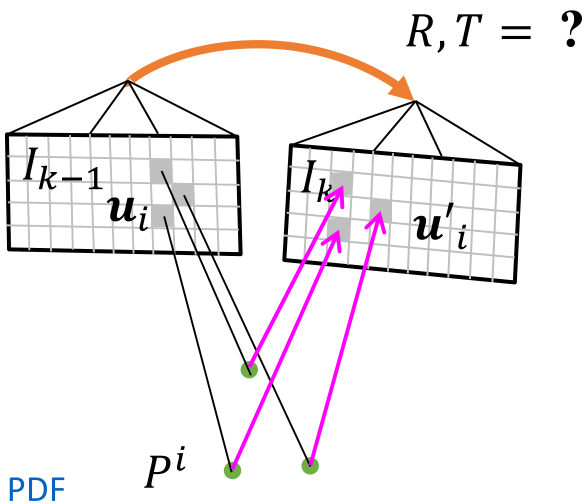
- **Indirect methods**
  1. Extract & match features + 3-point RANSAC
  2. Bundle Adjust by minimizing the **Reprojection Error**:

$$P^i, R, T = \arg\min_{P^i,R,T} \sum_{i=1}^{N} \rho\left(p_k^i - \pi(P^i, K, R, T)\right)$$



- **Direct methods**
  1. No feature extraction & no RANSAC needed.
     Instead, directly minimize **Photometric Error**:

$$P^i, R, T = \arg\min_{P^i,R,T} \sum_{i=1}^{N} \rho\left(I_{k-1}(p_{k-1}^i) - I_k\left(\pi(P^i, K, R, T)\right)\right)$$



What are their pros and cons?

Irani, Anandau, *All about direct methods*, Springer'99. PDF

Indirect vs Direct Methods

- **Indirect methods**
  1. Extract & match features + 3-point RANSAC
  2. Bundle Adjust by minimizing the **Reprojection Error**:

$$P^i, R, T = \arg\min_{P^i,R,T} \sum_{i=1}^{N} \rho\left(p_k^i - \pi(P^i, K, R, T)\right)$$

✓ Can cope with large frame-to-frame motions (large basin of convergence)

✗ Slow due to costly feature extraction, matching, and outlier removal (e.g., RANSAC)

- **Direct methods**
  1. No feature extraction & no RANSAC needed.
     Instead, directly minimize **Photometric Error**:

$$P^i, R, T = \arg\min_{P^i,R,T} \sum_{i=1}^{N} \rho\left(I_{k-1}(p_{k-1}^i) - I_k\left(\pi(P^i, K, R, T)\right)\right)$$

✓ All information in the image can be exploited (higher accuracy, higher robustness to motion blur and weak texture (i.e., weak gradients))

✓ Increasing the camera frame-rate reduces computational cost per frame (no RANSAC needed)

✗ Very sensitive to intial value → limited frame-to-frame motion (small basin of convergence)

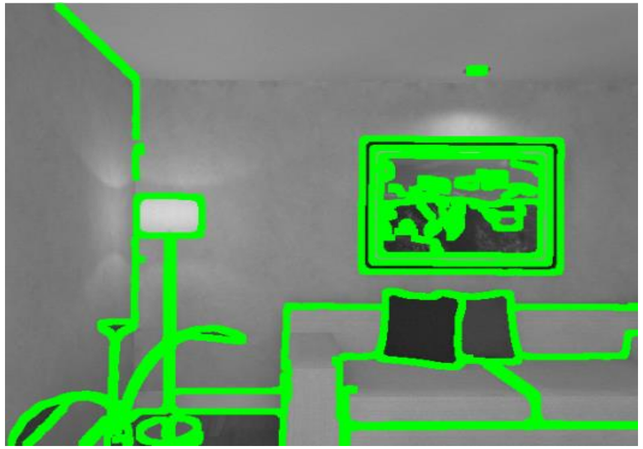
Irani, Anandau, *All about direct methods*, Springer'99. PDF

Direct Methods: Dense, Semi-dense, Sparse

Dense methods
track every pixel

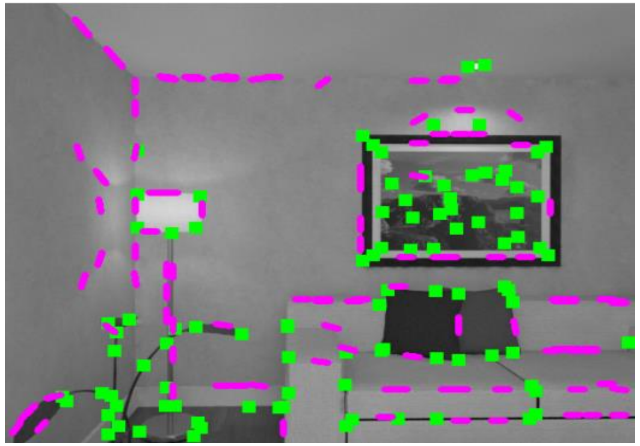
In a VGA image: 300'000+ pixels

Semi-Dense methods
track only edges



In a VGA image: ~10,000 pixels

Sparse methods
track sparse pixels



In a VGA image: ~2,000 pixels

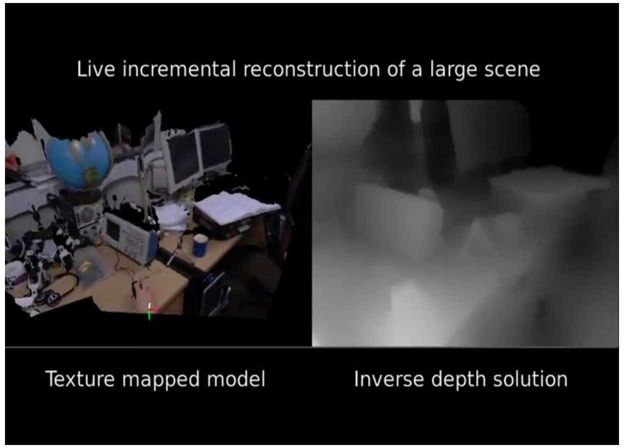
Forster, Zhang, Gassner, Werlberger, Scaramuzza, *SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems*, IEEE Transactions on Robotics (T-RO), 2017. PDF.]

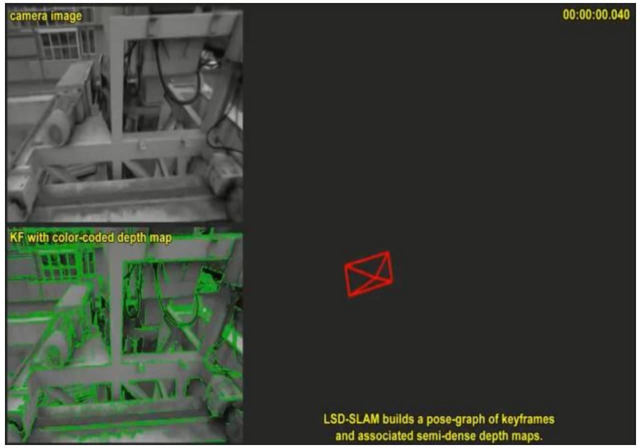
Direct Methods: Dense, Semi-dense, Sparse

Dense methods
track every pixel

Semi-Dense methods
track only edges

Sparse methods
track sparse pixels





In a VGA image: 300’000+ pixels

In a VGA image: ~10,000 pixels

In a VGA image: ~2,000 pixels
e.g., 120 feature patches × (4×4 pixels per patch)

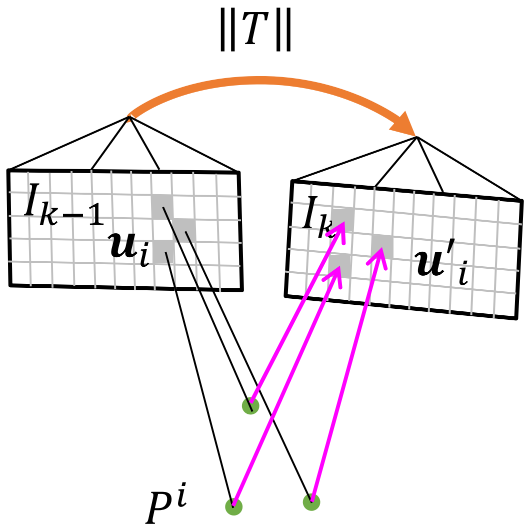
DTAM [Newcombe ‘11], REMODE [Pizzoli’14]

LSD-SLAM [Engel’14]

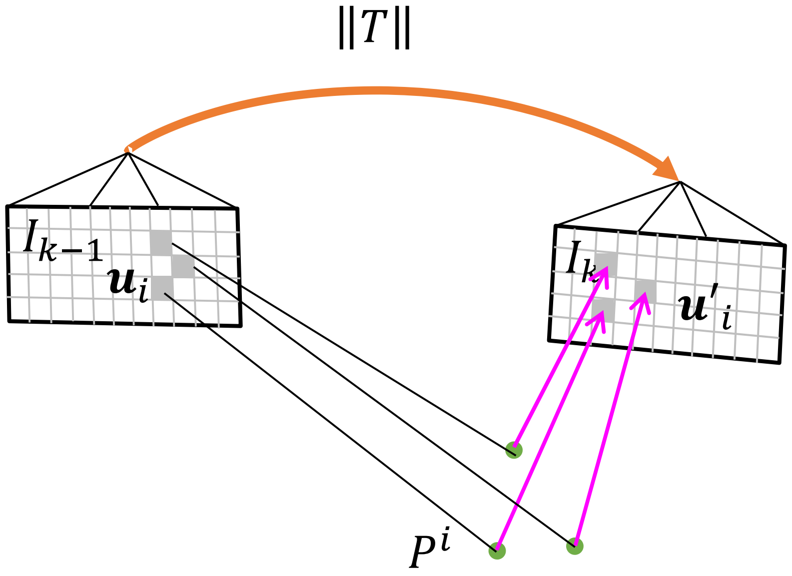
SVO [Forster’14], DSO [Engel’17]

Direct Methods: Dense, Semi-dense, Sparse

- What is the influence of the motion baseline on the convergence rate of direct methods?



For **small motion** baselines, $\|T\|$,
the **photometric error is usually small**

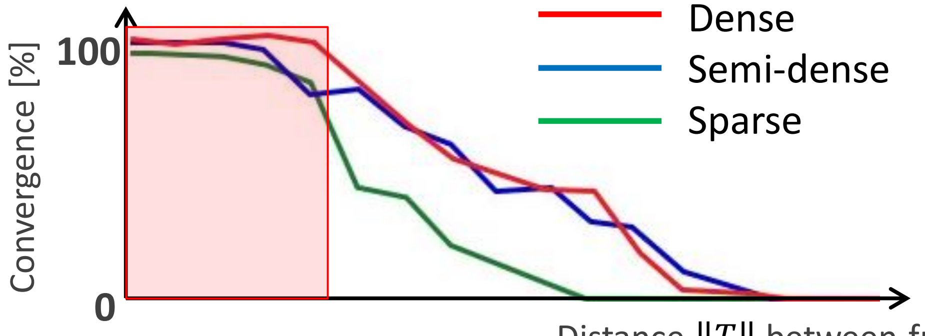


For **large motion** baselines, $\|T\|$,
the **photometric error is usually large**

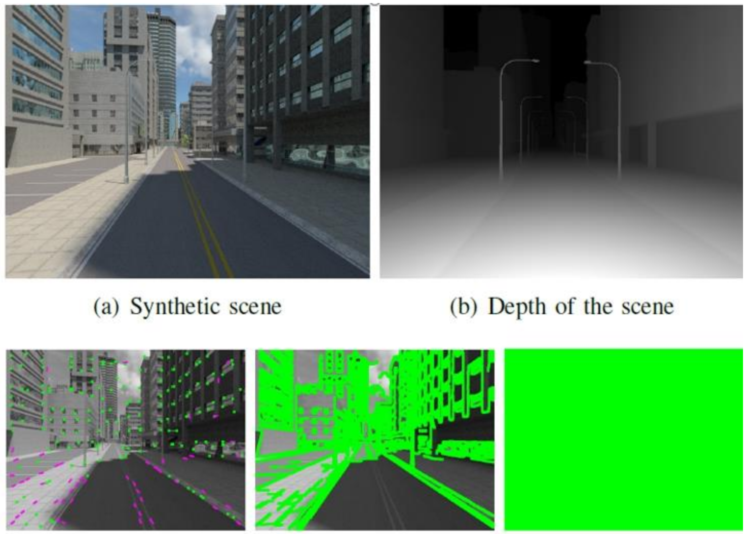
Forster, Zhang, Gassner, Werlberger, Scaramuzza, *SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems*, IEEE Transactions on Robotics (T-RO), 2017. [PDF](PDF).]

What is the influence of the motion baseline on the convergence rate of direct methods?

We can use **photorealistic simulation** to answer this question by generating thousands of data



(when the two images have 0 motion baseline, we say their overlap is 100%; as the baseline increases, their overlap decreases)



(a) Synthetic scene (b) Depth of the scene

(c) Sparse (d) Semi-Dense (e) Dense

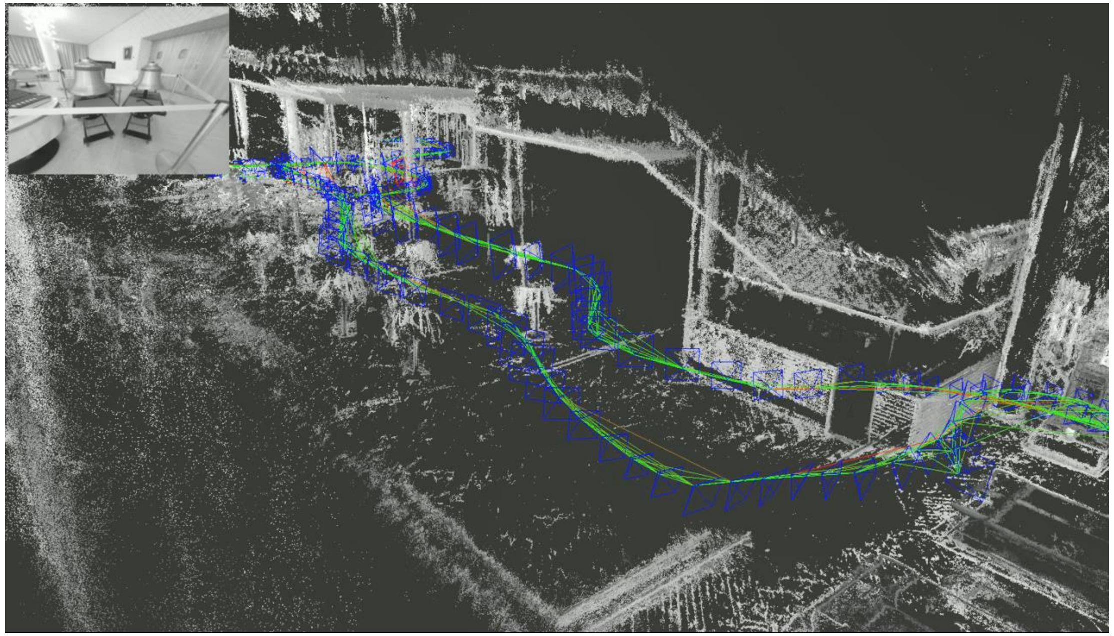
Simulated dataset from [here]

- Findings:
  - **Dense and Semi-dense** behave similarly
  - **Weak gradients are not informative** for the optimization
  - **Dense methods** are only useful **with motion blur, defocus**, and **weak- texture** regions
  - **Sparse methods behave equally well as dense or semi-dense methods** for small motion baselines

Forster, Zhang, Gassner, Werlberger, Scaramuzza, *SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems*, IEEE Transactions on Robotics (T-RO), 2017. PDF.]

LSD-SLAM

- Supports both **monocular and stereo** cameras
- **Direct** (photometric error) **+ Semi-Dense** formulation
  - **3D structure** represented as **semi-dense** depth map
  - Minimizes **photometric error**
  - **Separateley** optimizes poses & structure (sliding window)
- **Same workflow as PTAM** (keyframe based, alternation of localization and mapping as independent threads)
- Includes:
  - **Loop closing**
  - **Relocalization**
  - **Final optimization**
- **Real-time (30Hz)**, however global optimization is not done in real time but asynchronously every once in a while

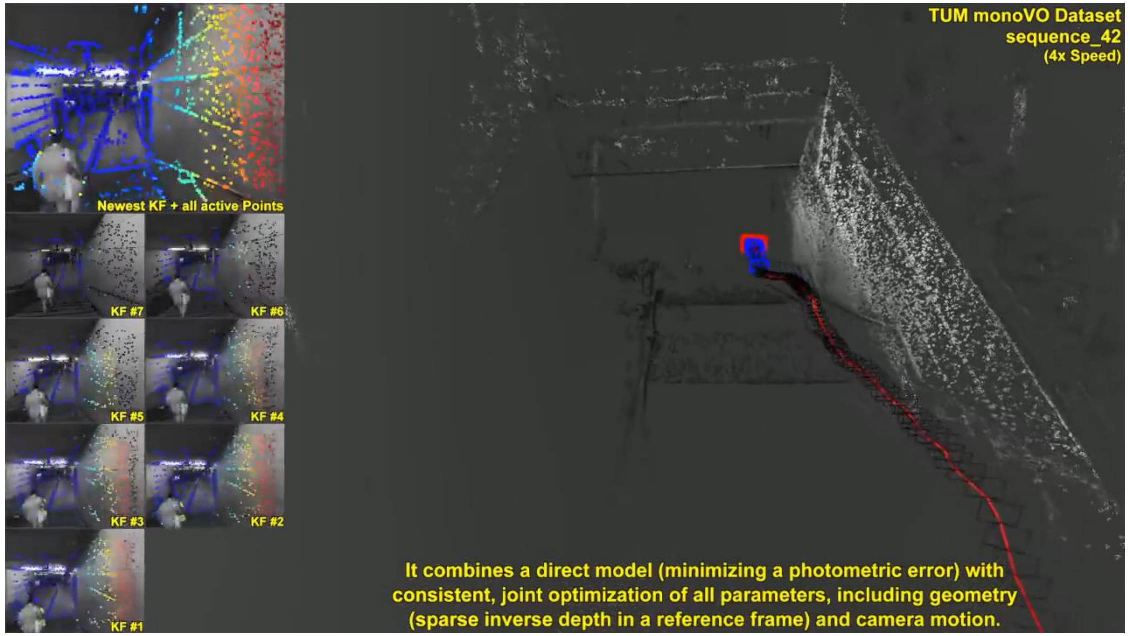


Engel, Schoeps, Cremers, *LSD-SLAM: Large-scale Semi-Dense SLAM*, European Conference on Computer Vision (ECCV), 2014. PDF, code, videos.

DSO

- Supports both **monocular and stereo** cameras
- **Direct** (photometric error) + **Sparse** formulation
  - **3D structure** represented as **sparse large gradients'** depth map
  - Minimizes **photometric error**
  - **Jointly optimizes poses & structure** (sliding window)
  - Incorporates photometric correction to compensate exposure time change (Δt_{k-1}, Δt_k)

$$P^i, R, K = \arg\min_{P^i, R, K} \sum_{i=1}^{N} \rho\left(I_{k-1}\left(p_{k-1}^i\right) - \frac{\Delta t_{k-1}}{\Delta t_k} I_k\left(\pi\left(P^i, K, R, T\right)\right)\right)$$

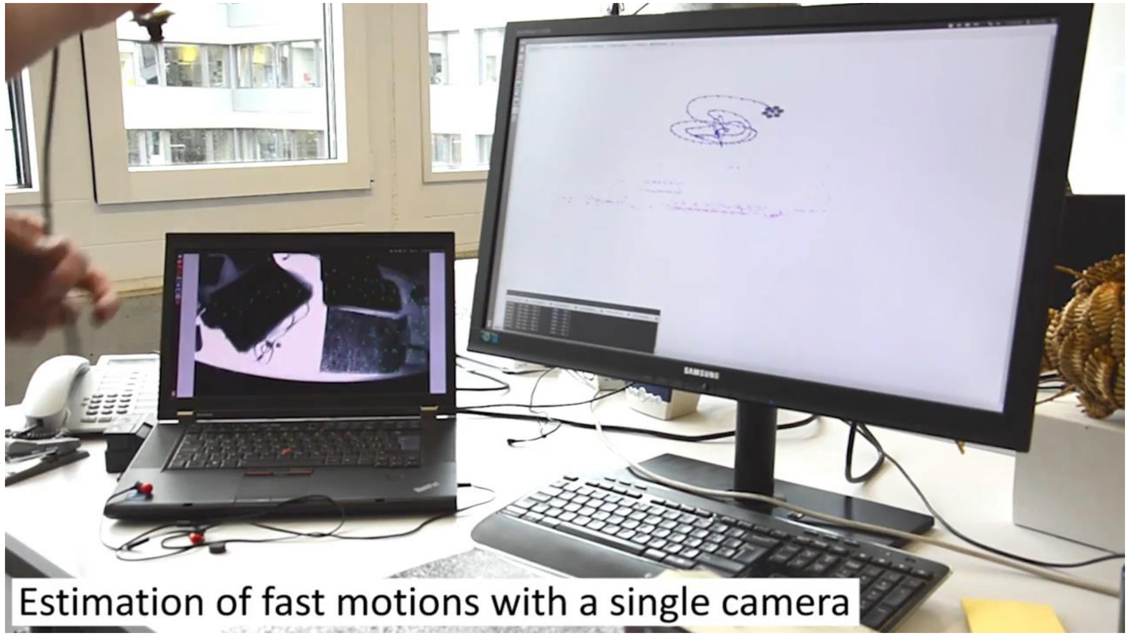


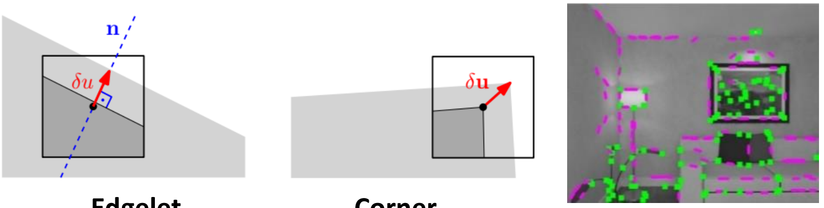
- **Same workflow as PTAM** (keyframe based, alternation of localization and mapping as independent threads)
- **Real-time (30Hz)**, however global optimization is not done in real time but asynchronously every once in a while

Engel, Koltun, Cremers, *DSO: Direct Sparse Odometry*, IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 2017. PDF, code, and videos.

SVO

- Supports both **monocular, stereo, multi-camera systems** as well as omnidirectional models (fisheye and catadioptric)
- Combines **indirect** + **direct methods**
  - **Direct** methods for **frame-to-frame motion estimation**
  - **Indirect** methods for **frame-to-keyframe pose refinement**
- **Mapping**
  - **Probabilistic depth** estimation (heavy-tail Gaussian distribution)
- Includes:
  - **Loop closing,**
  - **Relocalization,**
  - **Final optimization**
- **Same workflow as PTAM** (keyframe based, alternation of localization and mapping as independent threads)
- **Faster than real-time: up to 400 fps** on i7 laptops and **100 fps** on smartphone PCs (Odroid (ARM)) or NVIDIA Jetson



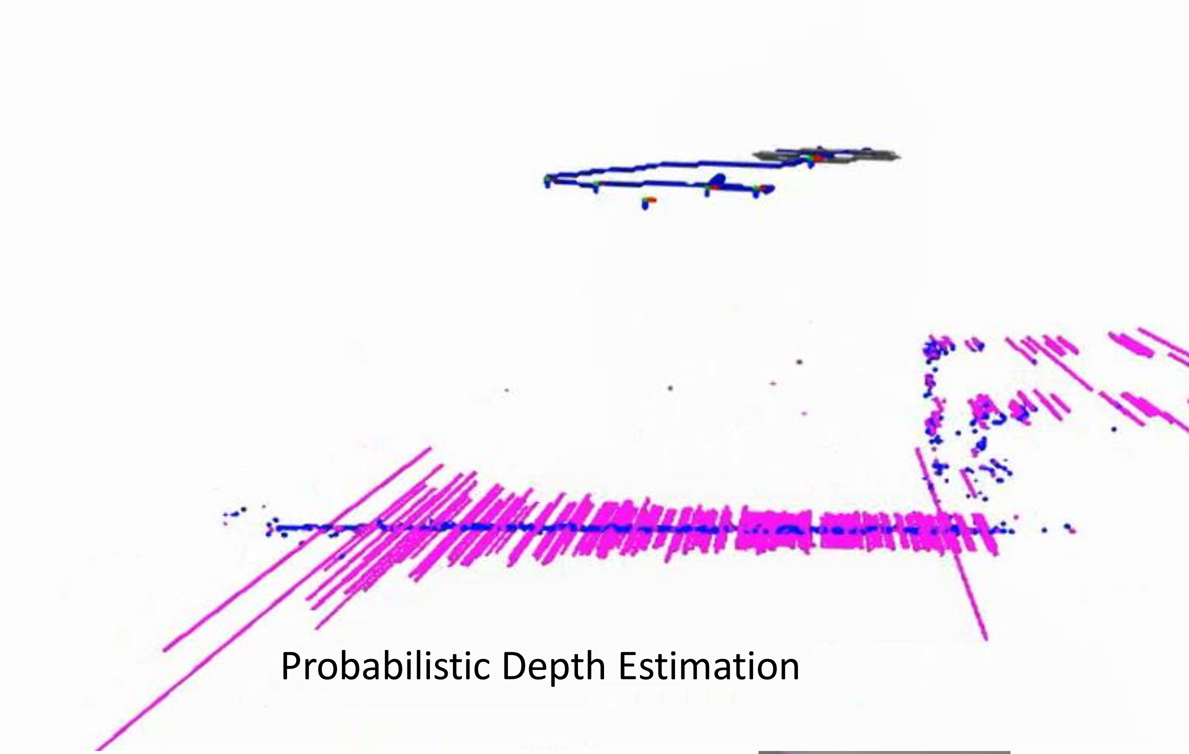


Edgelet **Corner**

Forster, Zhang, Gassner, Werlberger, Scaramuzza, *SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems*, IEEE Transactions on Robotics (T-RO), 2017. PDF, code, videos.

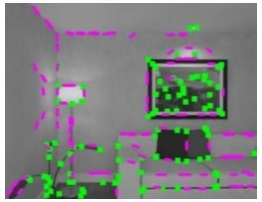
SVO

- Supports both **monocular, stereo, multi-camera systems** as well as omnidirectional models (fisheye and catadioptric)
- Combines **indirect** + **direct methods**
  - **Direct** methods for **frame-to-frame motion estimation**
  - **Indirect** methods for **frame-to-keyframe pose refinement**
- **Mapping**
  - **Probabilistic depth** estimation (heavy-tail Gaussian distribution)
- Includes:
  - **Loop closing,**
  - **Relocalization,**
  - **Final optimization**
- **Same workflow as PTAM** (keyframe based, alternation of localization and mapping as independent threads)
- **Faster than real-time: up to 400 fps** on i7 laptops and **100 fps** on smartphone PCs (Odroid (ARM)) or NVIDIA Jetson



Probabilistic Depth Estimation

Edgelet Corner



SWISS MADE

Forster, Zhang, Gassner, Werlberger, Scaramuzza, *SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems*, IEEE Transactions on Robotics (T-RO), 2017. PDF, code, videos.

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

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

Processing time in milliseconds

CPU load (100% = 1 core)

Forster, Zhang, Gassner, Werlberger, Scaramuzza, *SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems*, IEEE Transactions on Robotics (T-RO), 2017. PDF, code, videos.

SVO and its derivatives are used today in many of products…

- DJI drones
- Magic Leap AR headsets
- Oculus VR headsets
- Huawei phones
- Nikon cameras
- …

Autonomous quadrotor navigation in dynamic scenes (down-looking camera)
(running on Odroid U3 board (ARM Cortex A9 at 90fps)

20 m/s obstacle free autonomous quadrotor flight at DARPA FLA (2015)

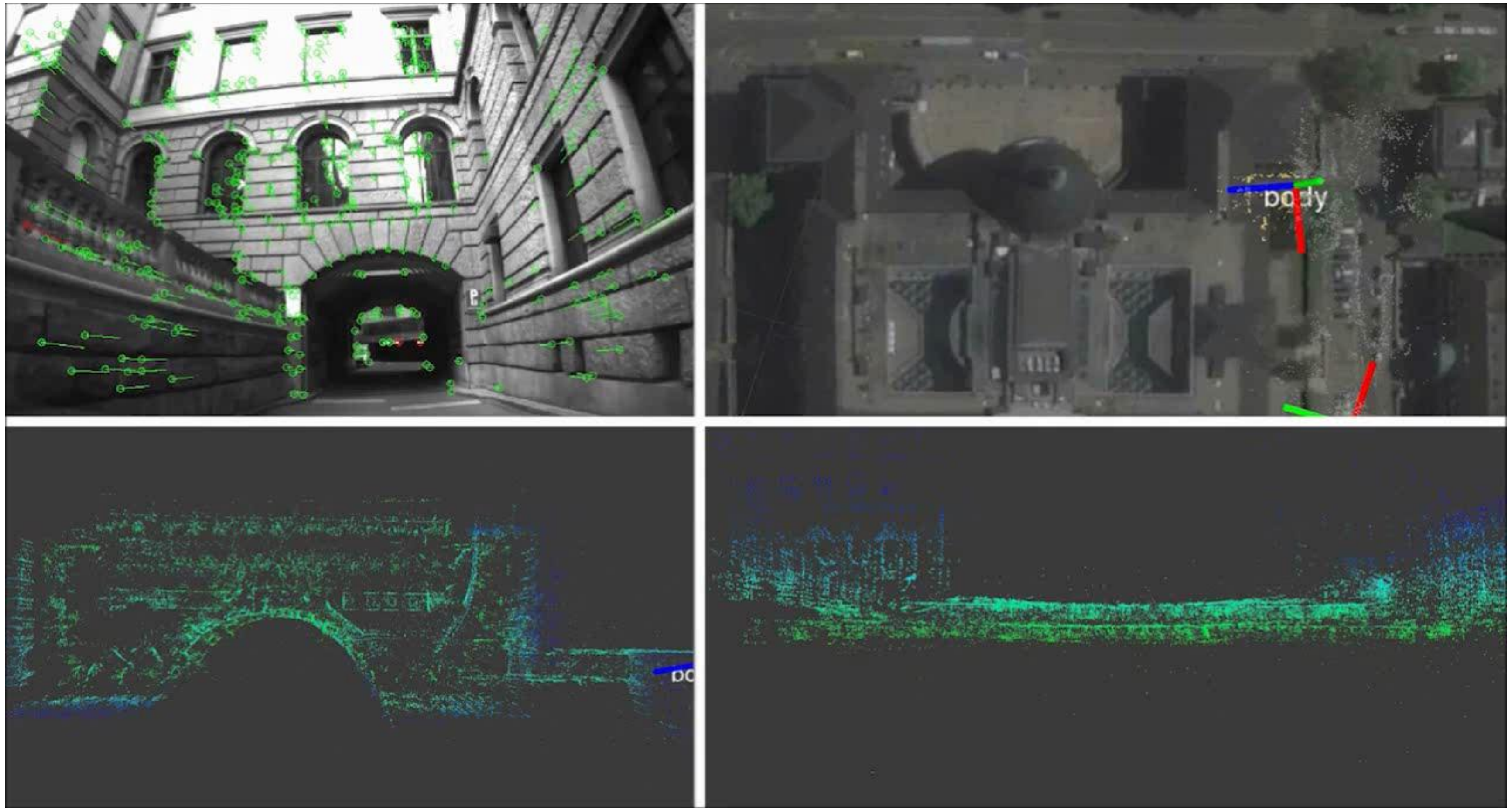
Throw-and-go (2015)
(inspired many products, like [DJI Tello drone])

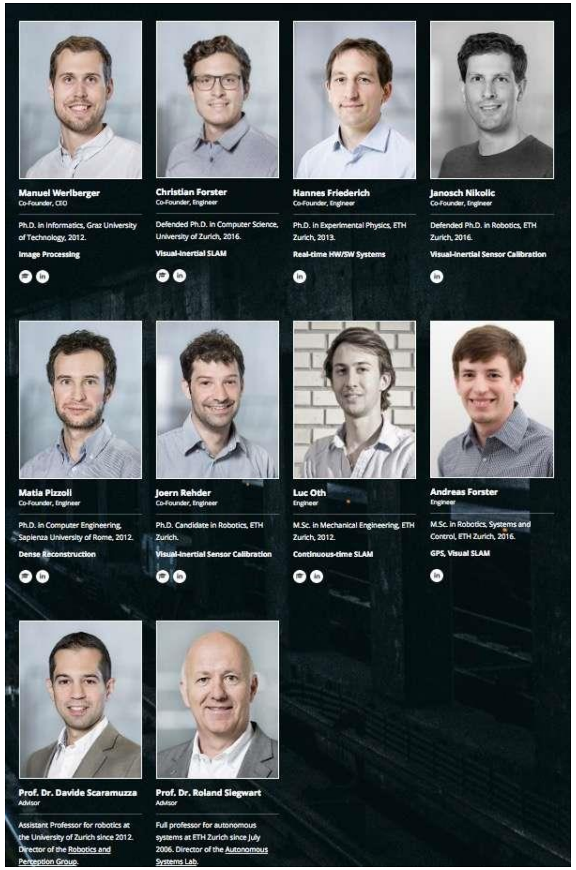
Virtual Reality with SVO running on an iPhone 6
(with company Dacuda at CES 2017. Dacuda is today Magic Leap Zurich)

More videos here: http://rpg.ifi.uzh.ch/svo_pro.html

Startup: “Zurich-Eye” – Today: Facebook-Oculus Zurich

- **Vision-based Localization and Mapping** systems for mobile robots
- Born in Sep. 2015, became **Facebook-Oculus Zurich** in Sep. 2016. Today, **200 employees**.





Startup: “Zurich-Eye” – Today: Facebook-Oculus Zurich

- **Vision-based Localization and Mapping** systems for mobile robots
- Born in Sep. 2015, became **Facebook-Oculus Zurich** in Sep. 2016. Today, **200 employees**.
- In 2018, Zurich-Eye launched **Oculus Quest** (2 million units sold so far)
- Christian Forster (Facebook Zurich & co-founder of Zurich-Eye) gave a lecture on Nov. 26, 2020, which will be shared on OLAT.

Things to remember

- Hierarchical SFM
- VO flowchart
  - Monocular VO
  - Stereo VO
  - Keyframe selection
- Bundle adjustment vs pose-graph optimization
- Indirect vs direct methods
- Direct methods: Dense, semi-dense, and sparse formulations
- Popular open-source VO algorithms

Readings

- 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

Understanding Check

Are you able to answer the following questions:

- Bundle Adjustment and Pose Graph Optimization. Mathematical expressions and illustrations. Pros and cons.
- Are you able to describe hierarchical and sequential SFM for monocular VO?
- What are the building blocks of visual odometry and SLAM?
- 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? How can we detect loop closures? (make link to other lectures)
- Are you able to describe the differences between feature-based methods and direct methods?
- Sparse vs semi-dense vs dense. What are their pros and cons?
- Are you able to provide a list of the most popular open source VO and VSLAM algorithms?
- Difference between SFM, VO, SLAM (see also lecture 01)