Tutorial on Visual Odometry

Davide Scaramuzza

Robotics and Perception Group
http://rpg.ifi.uzh.ch
University of Zurich

Outline

- Theory
- Open Source Algorithms

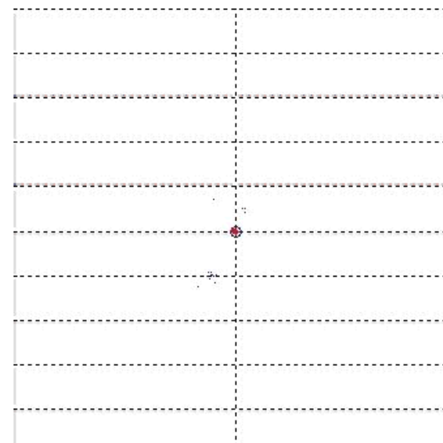
What is Visual Odometry (VO) ?

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

input

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

output



$$R_0, R_1, \dots, R_i$$

$$t_0, t_1, \dots, t_i$$

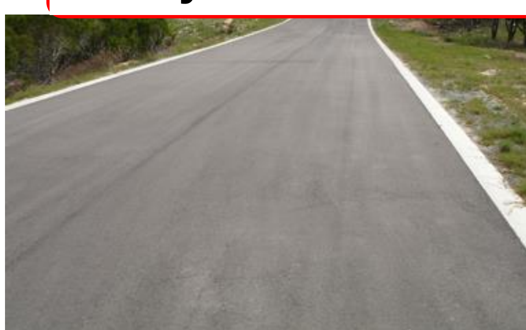
Camera trajectory (3D structure is a plus):

Assumptions

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



Is any of these scenes good for VO? Why?




A Brief history of VO

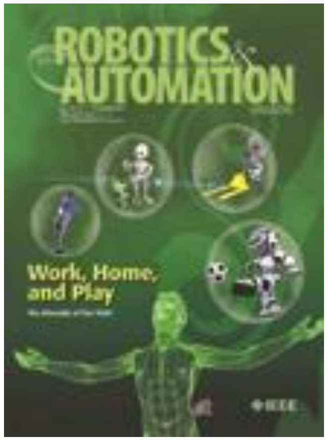
- **1980**: First known VO real-time implementation on a robot by **Hans Moraveck** PhD thesis **(NASA/JPL)** for Mars rovers using one sliding camera (*sliding stereo*).

- **1980 to 2000**: The VO research was dominate
2004 Mars mission (see papers from Matthies

- **2004**: VO used on a robot on another planet: M

- **2004**. VO was revived in the academic environ
by Nister «Visual Odometry» paper.
The term VO became popular.

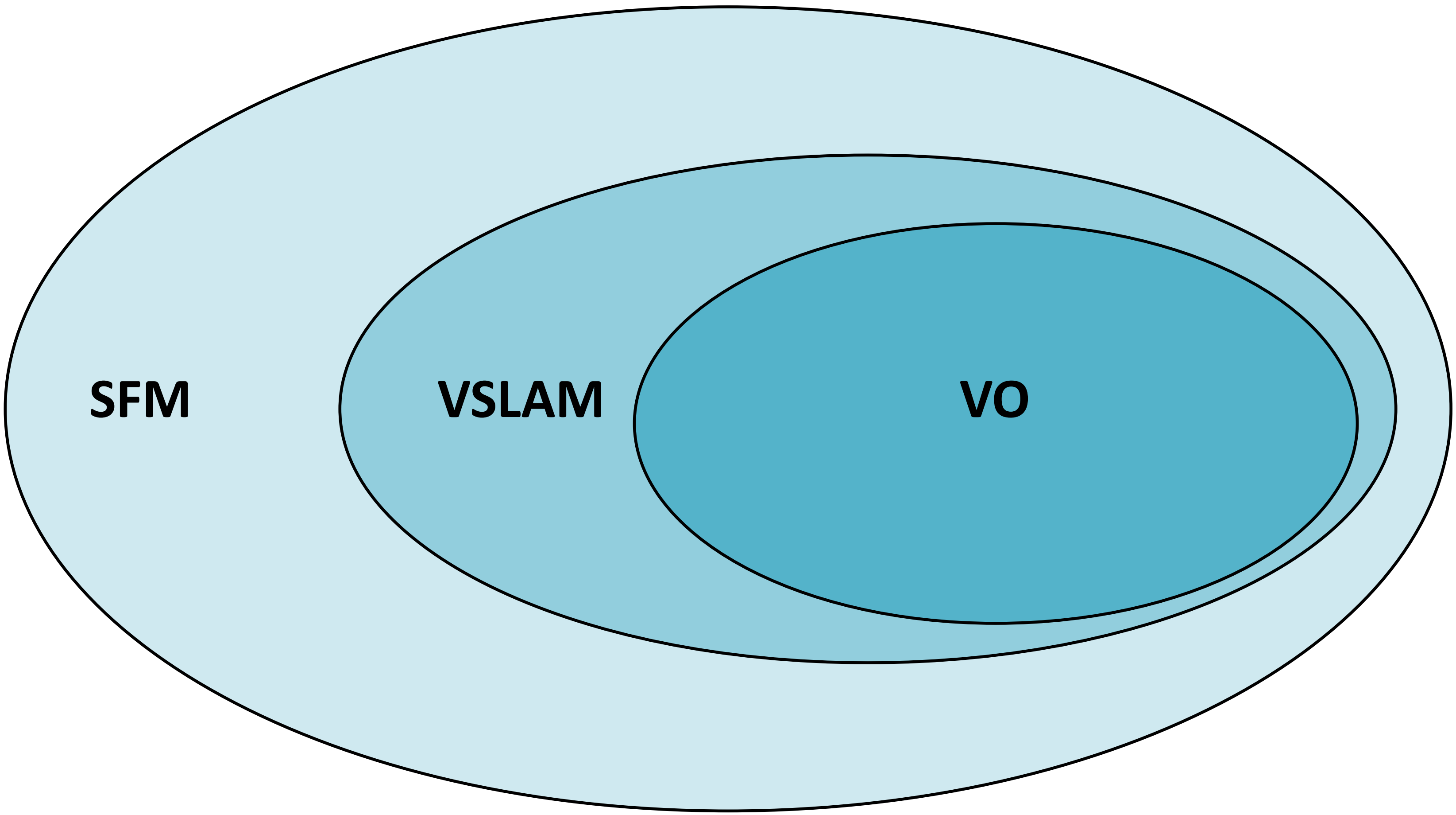
References



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

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

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

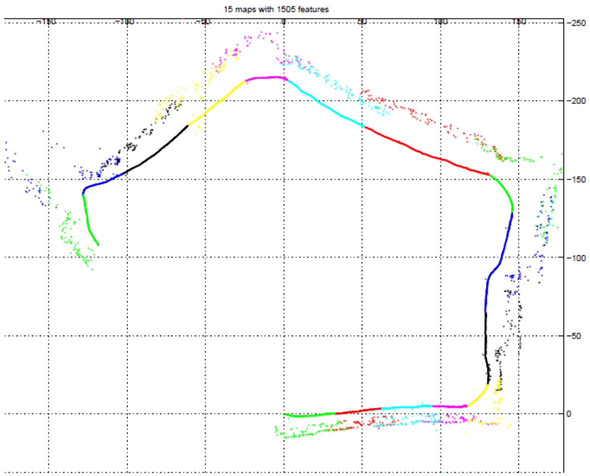
- VO is a **particular case** of SFM

- VO focuses on estimating the 3D motion of the camera **sequentially** (as a new frame arrives) and in **real time**.

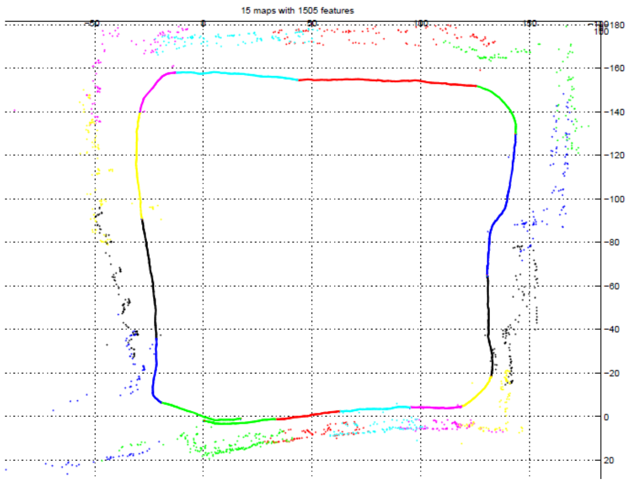
- Terminology: sometimes SFM is used as a synonym of VO

VO vs. Visual SLAM

- 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 <u>building block</u> of SLAM
- **VO is SLAM before closing the loop!**
- The choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation.
- VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera.



Visual odometry



Visual SLAM

Image courtesy from [Clemente, RSS'07]

VO Working Principle

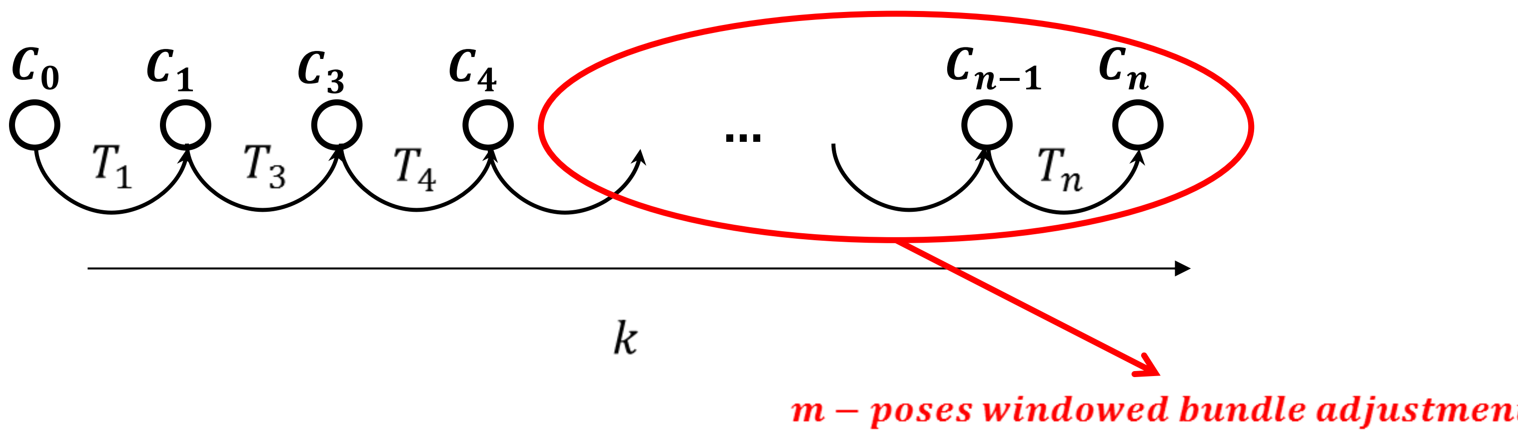
1. Compute the relative motion T_k from images I_{k-1} to image I_k

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

2. Concatenate them to recover the full trajectory

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

3. An optimization over the last m poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment)



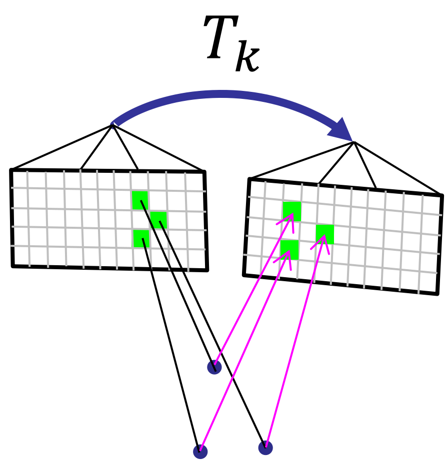
Front-End vs *Back-End*

- The *front-end* is responsible for
  - Feature extraction, matching, and outlier removal
  - Loop closure detection
- The *back-end* is responsible for the pose and structure optimization (e.g., iSAM, g2o, Google Ceres)

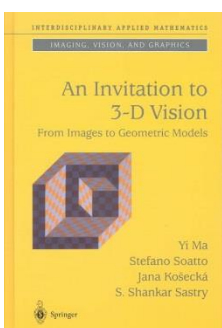
How do we estimate the relative motion T_k ?

Image I_{k-1}

Image I_k

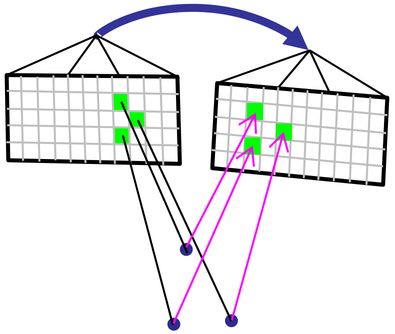


$$T_k = \arg\min_{\mathbf{T}} \iint_{\bar{\mathcal{R}}} \rho \left[I_k\Big(\pi\big(\mathbf{T} \cdot \pi^{-1}(\mathbf{u}, d_{\mathbf{u}})\big)\Big) - I_{k-1}(\mathbf{u}) \right] d\mathbf{u}$$



“An Invitation to 3D Vision”, Ma, Soatto, Kosecka, Sastry, Springer, 2003

Direct Image Alignment



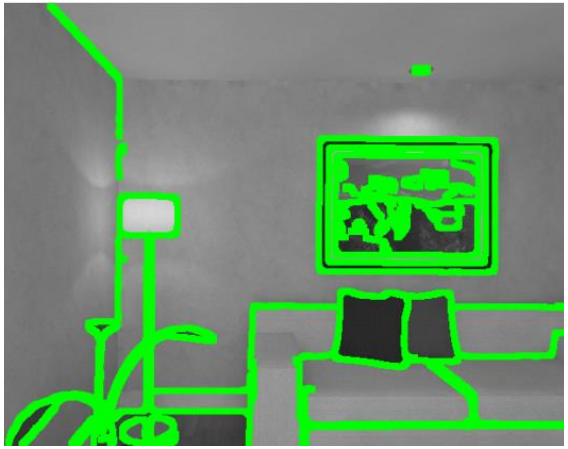
It minimizes the **per-pixel intensity difference**

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

Dense

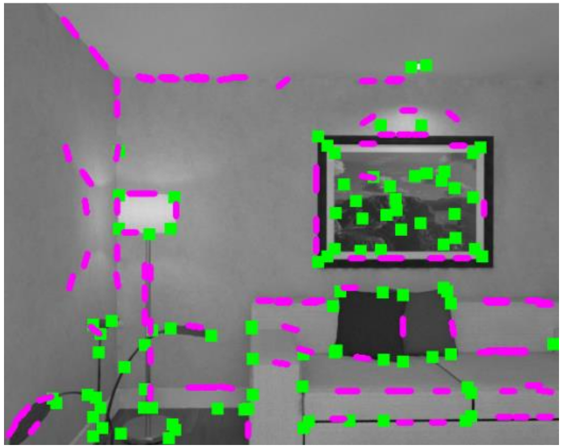
DTAM [Newcombe et al. '11]
300'000+ pixels

Semi-Dense



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

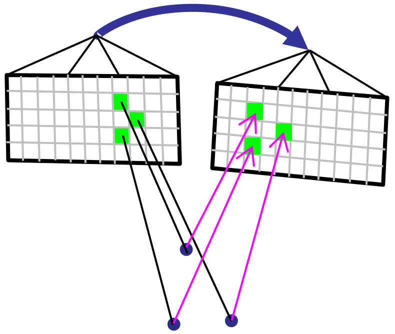
Sparse



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

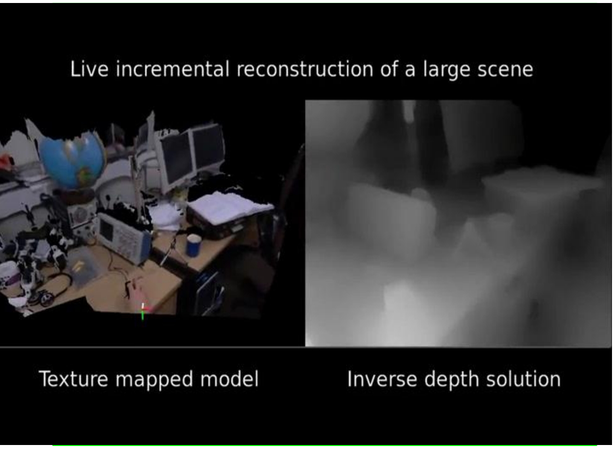
Direct Image Alignment



It minimizes the **per-pixel intensity difference**

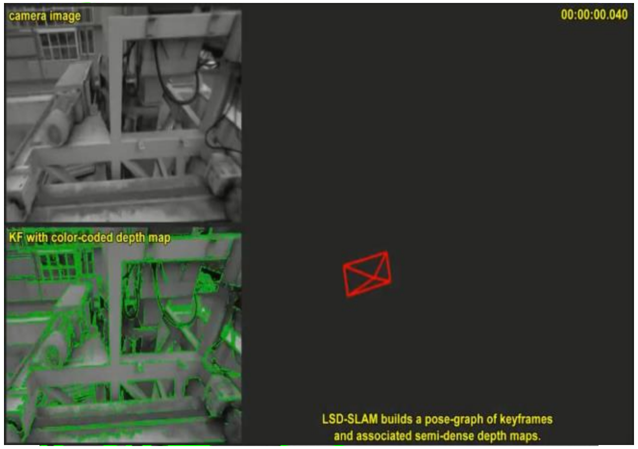
$$T_{k,k-1} = \arg\min_{T} \sum_{i} \| I_k(\boldsymbol{u}'_i) - I_{k-1}(\boldsymbol{u}_i) \|_\sigma^2$$

Dense



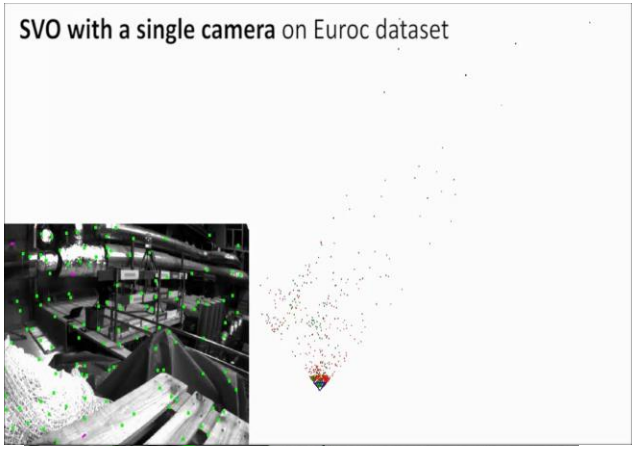
DTAM [Newcombe et al. '11]
300,000+ pixels

Semi-Dense



LSD-SLAM [Engel et al. 2014]
~10,000 pixels

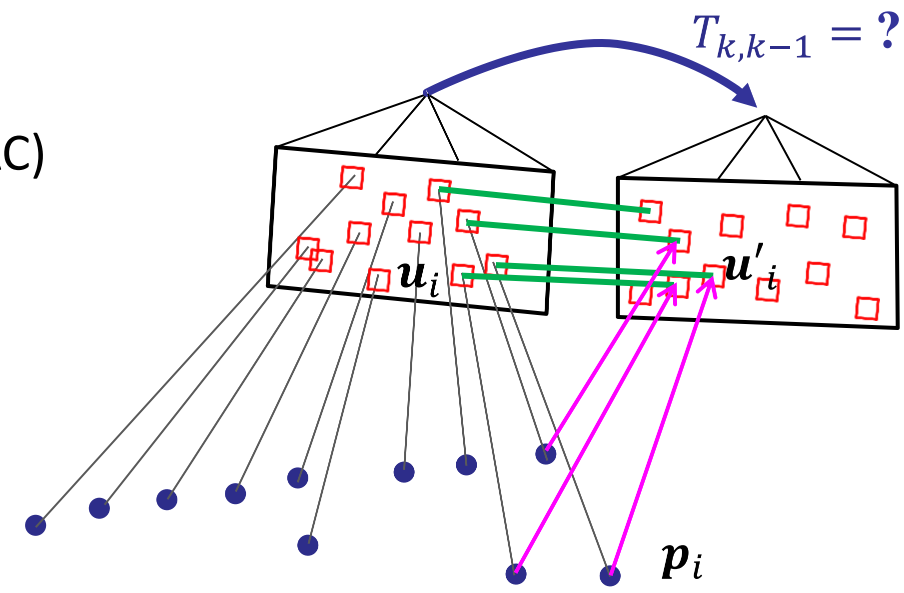
Sparse



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

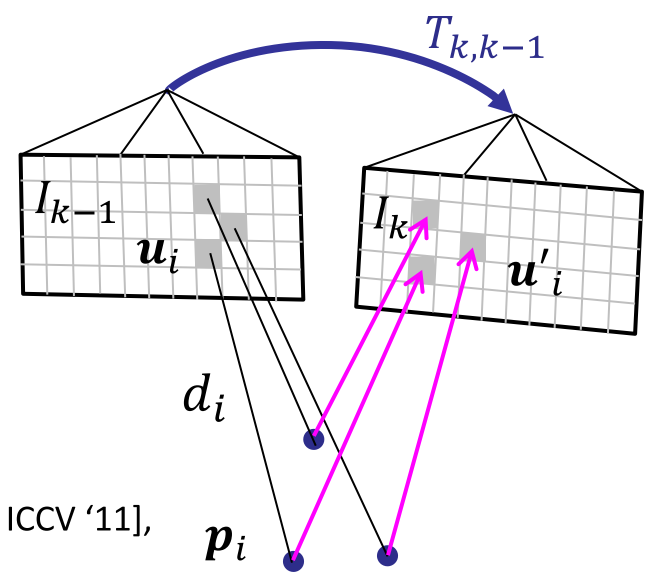
Feature-based methods



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

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

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

[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**

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

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

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

✓ All information in the image can be exploited (precision, robustness)

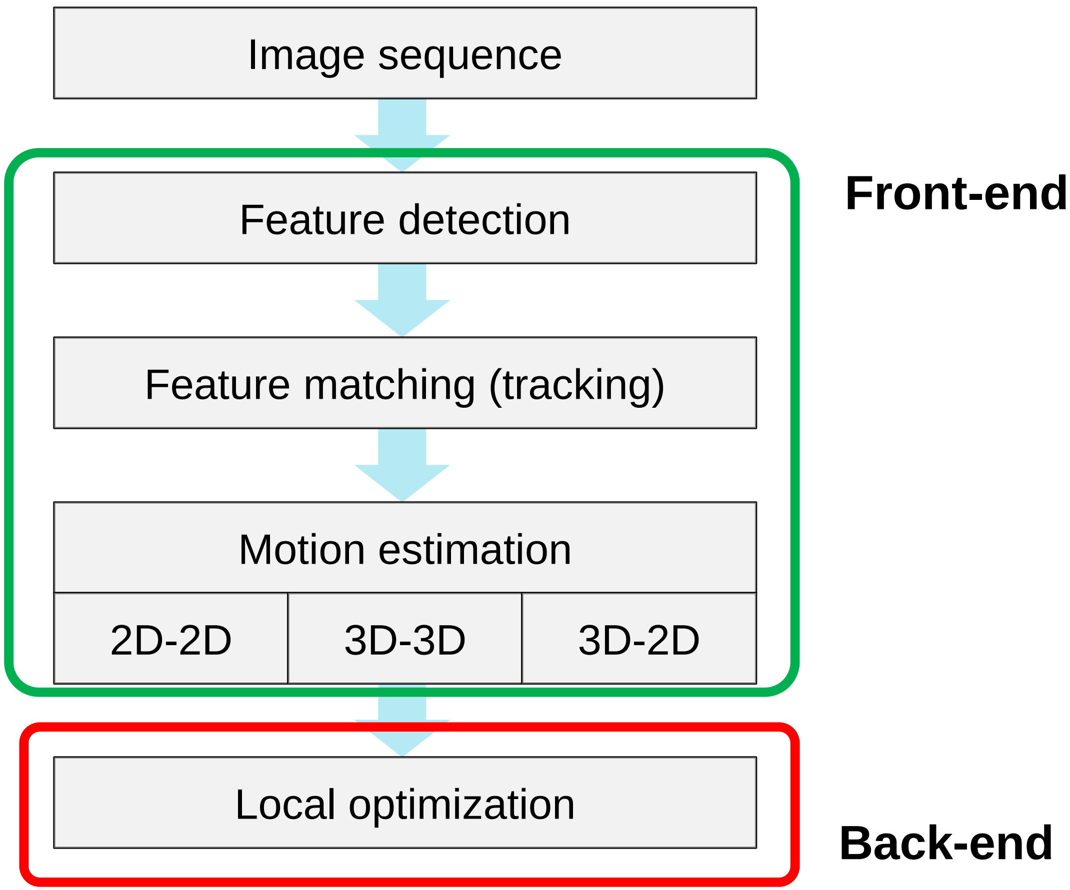
✓ Increasing camera frame-rate reduces computational cost per frame

✗ Limited frame-to-frame motion

✗ Joint optimization of dense structure and motion too expensive

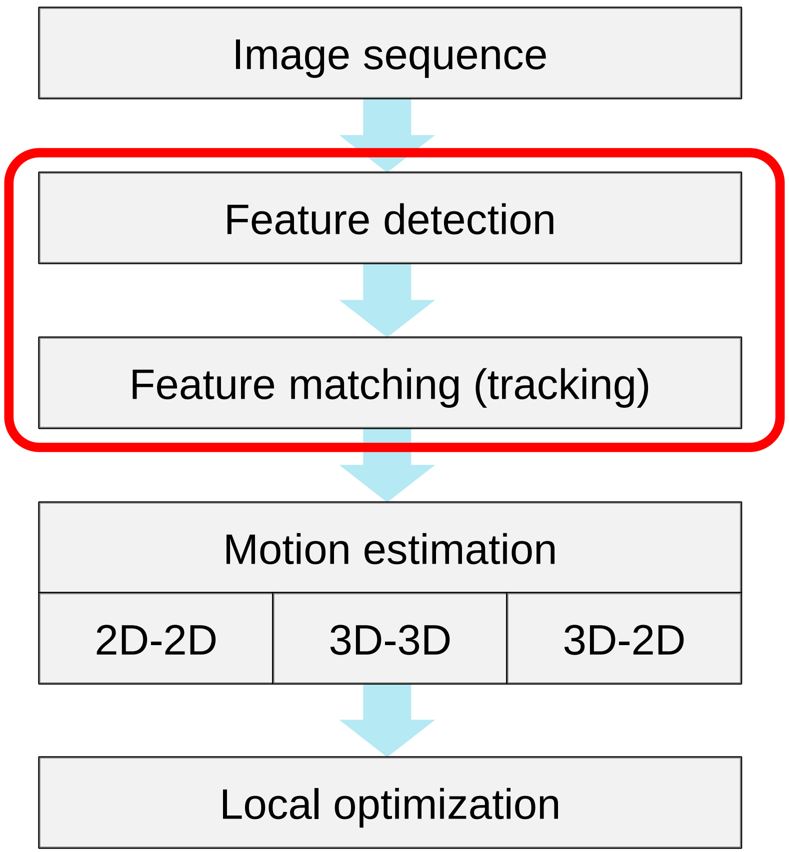
VO Flow Chart

VO computes the camera path incrementally (pose after pose)



VO Flow Chart

VO computes the camera path incrementally (pose after pose)



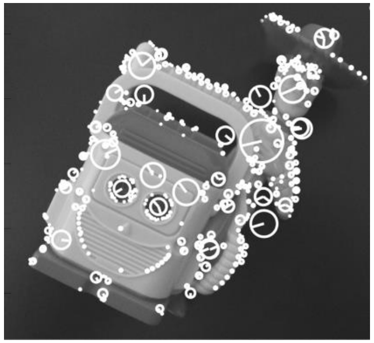
Example features tracks

Corners vs Blob Detectors

- A **corner** is defined as the intersection of one or more edges
  - A corner has high localization accuracy
    - Corner detectors are good for VO
  - It's **less distinctive than a blob**
  - E.g., *Harris, Shi-Tomasi, SUSAN, FAST*

Harris corners

- A **blob** is any other image pattern, **which is not a corner**, that significantly differs from its neighbors in intensity and texture
  - **Has less localization accuracy than a corner**
    - **Blob detectors are better for place recognition**
  - It's **more distinctive than a corner**
  - E.g., *MSER, LOG, DOG (SIFT), SURF, CenSurE*



SIFT features

VO Flow Chart

VO computes the camera path incrementally (pose after pose)

Image sequence

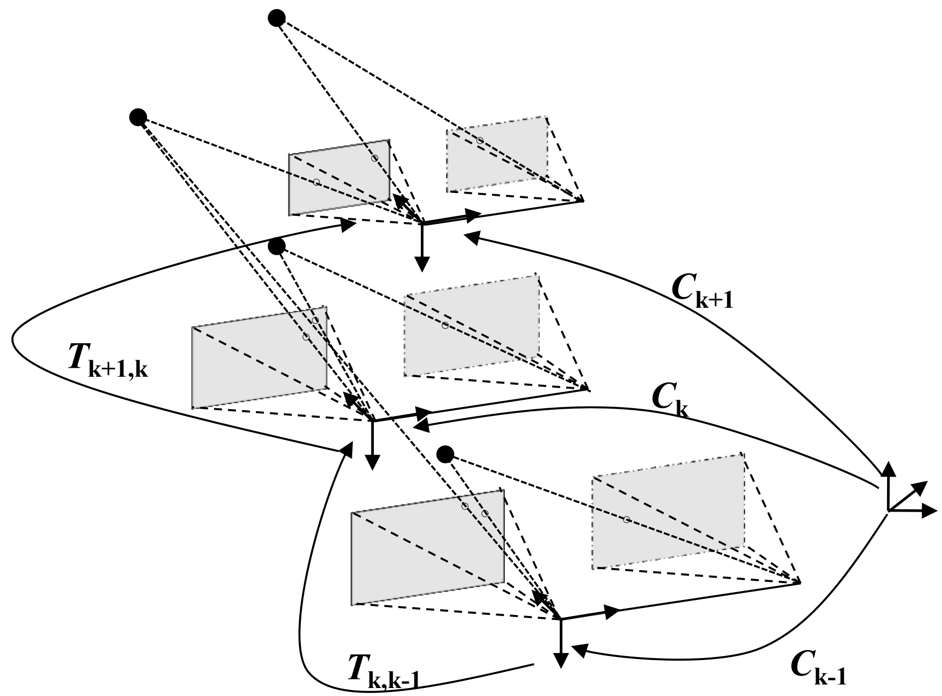
Feature detection

Feature matching (tracking)

Motion estimation

2D-2D	3D-3D	3D-2D

Local optimization



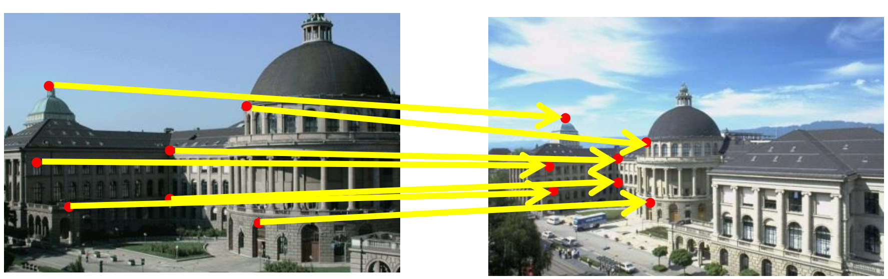
2D-to-2D

Motion from Image Feature Correspondences

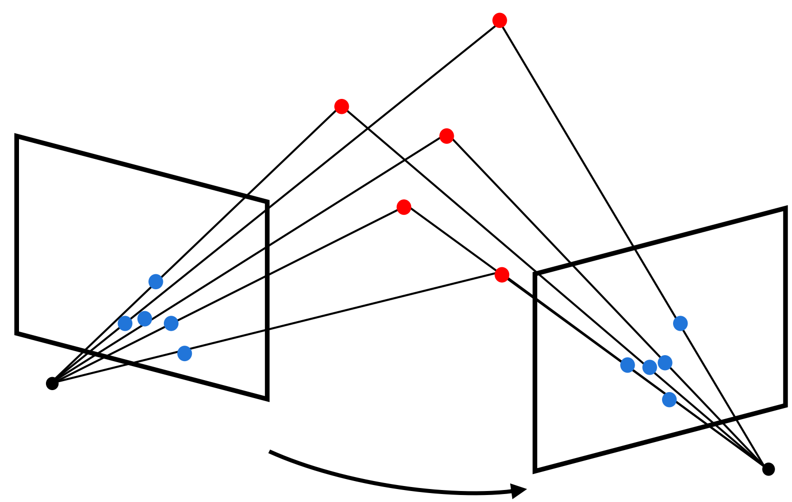
- Both feature points f_{k-1} and f_k are specified **in 2D**
- The minimal-case solution involves **5-point** correspondences
- The solution is found by minimizing the reprojection error:

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

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



I_{k-1} $\quad$ I_k



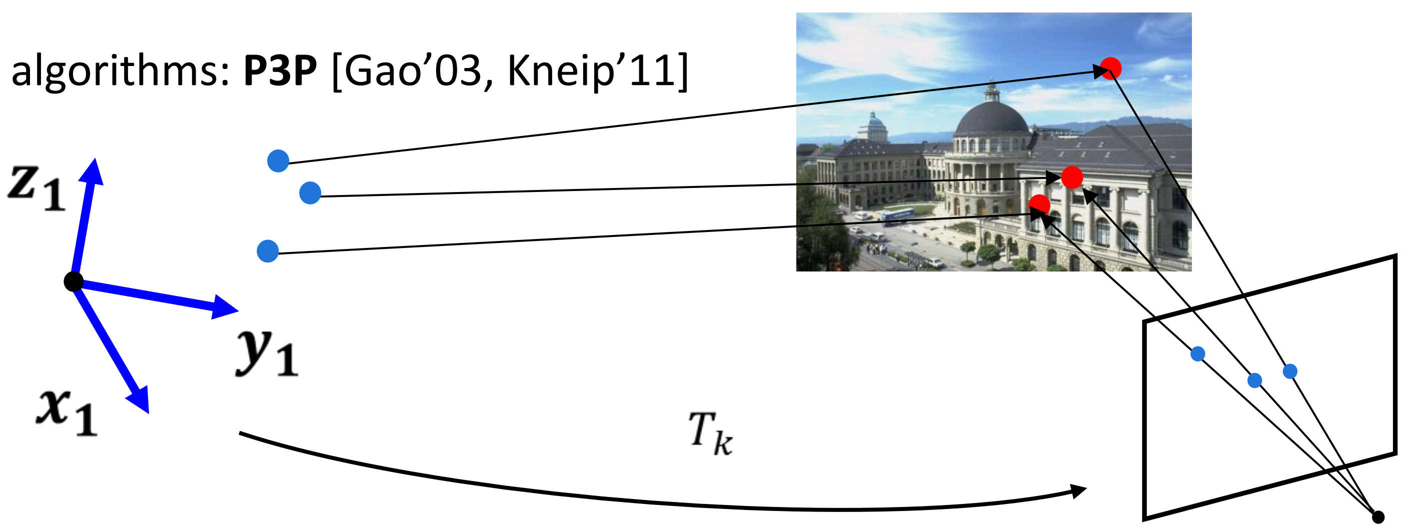
3D-to-2D

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:

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

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



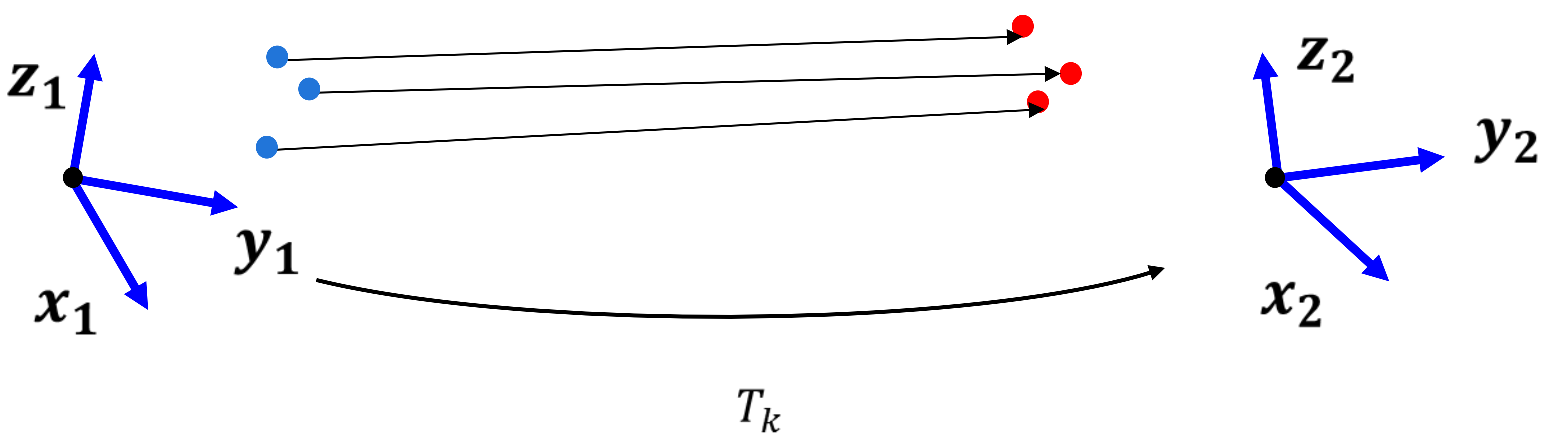
3D-to-3D

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:

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

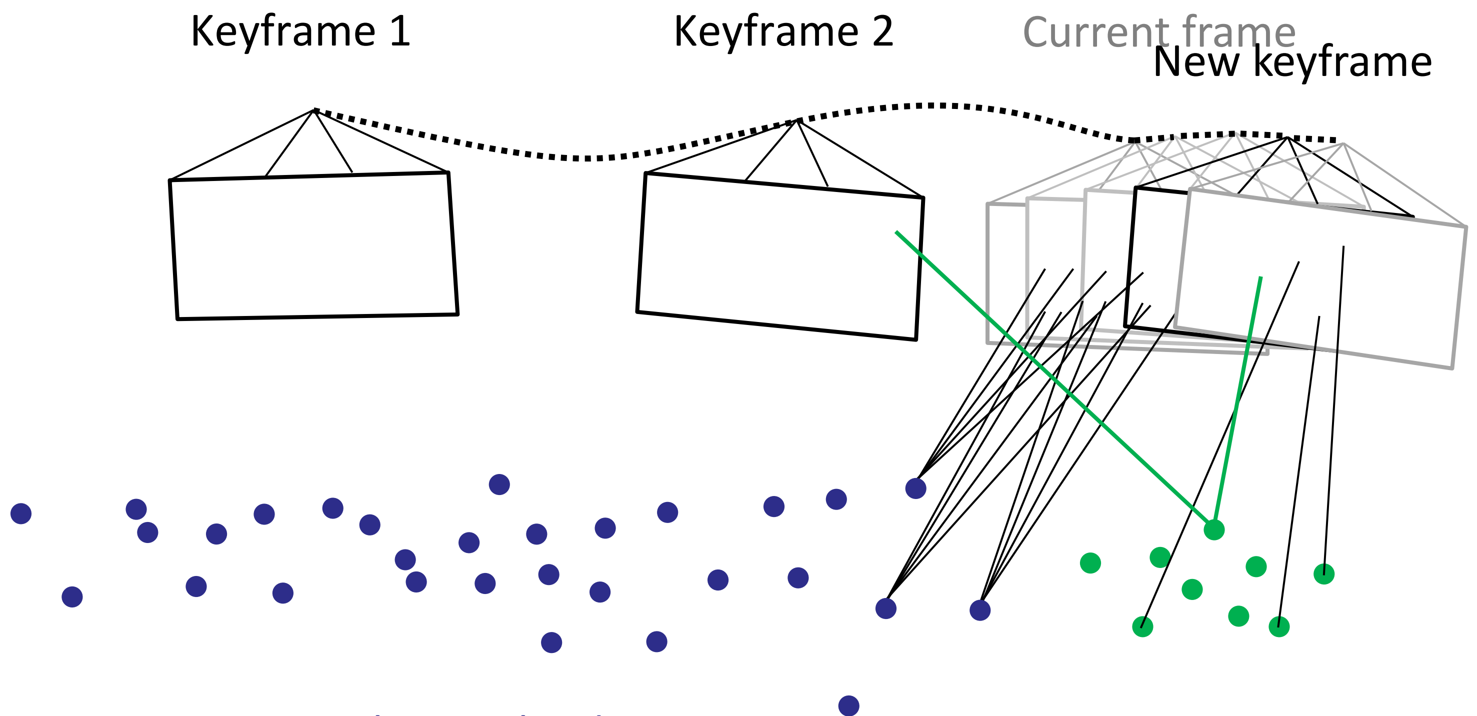
- Popular algorithm: [Arun'87] for global registration, ICP for local refinement or Bundle Adjustment (BA)



Motion Estimation: Summary

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

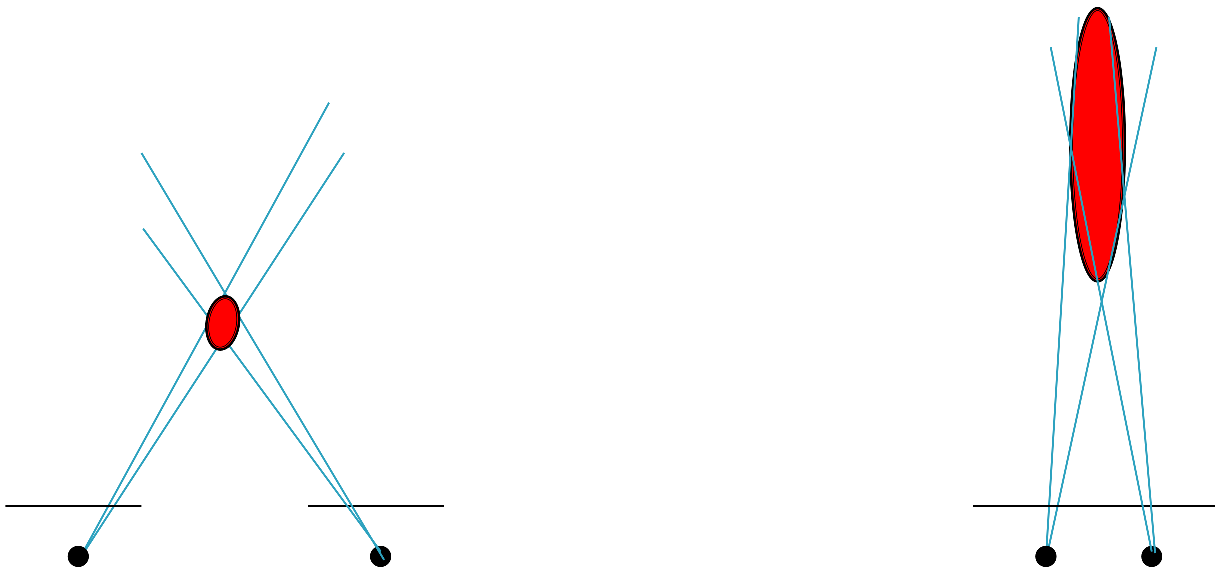
Example: Keyframe-based Monocular Visual Odometry



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

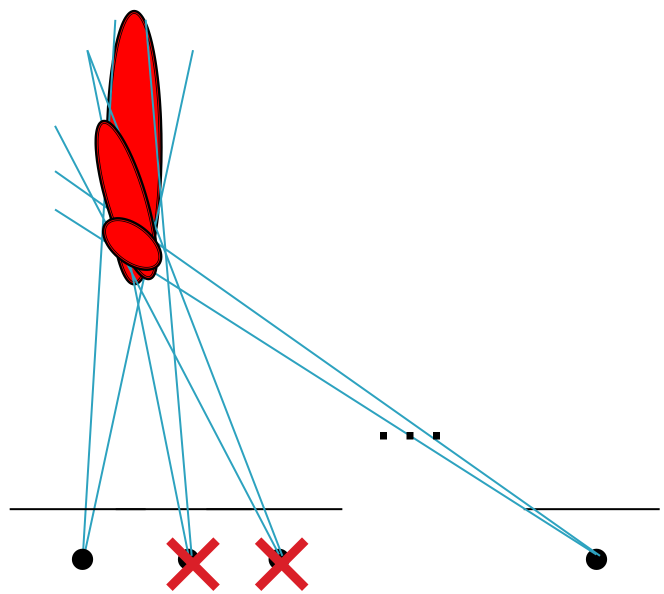
Keyframe Selection

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



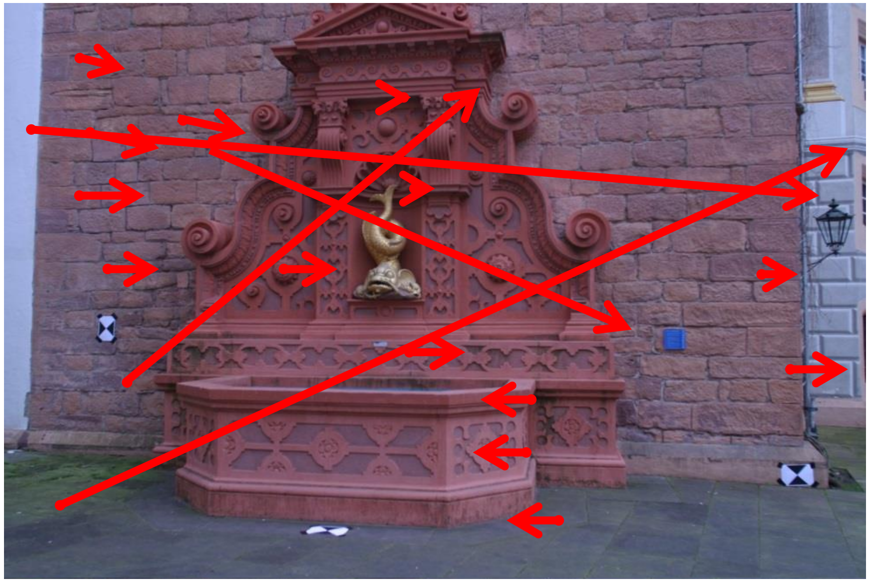
Keyframe Selection

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

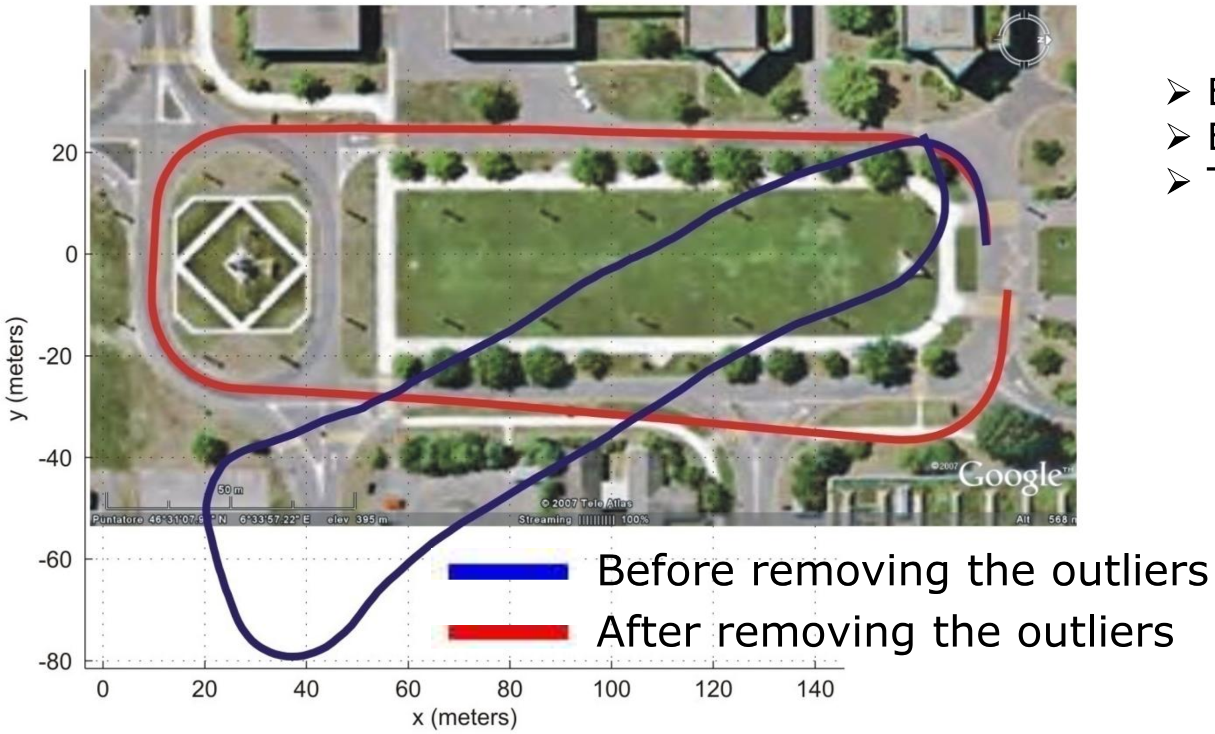


Robust Estimation

- Matched points are usually contaminated by outliers
- Causes of outliers are:
  - image noise
  - occlusions
  - blur
  - changes in view point and illumination
- For the camera motion to be estimated accurately, outliers must be removed
- This is the task of Robust Estimation



Influence of Outliers on Motion Estimation

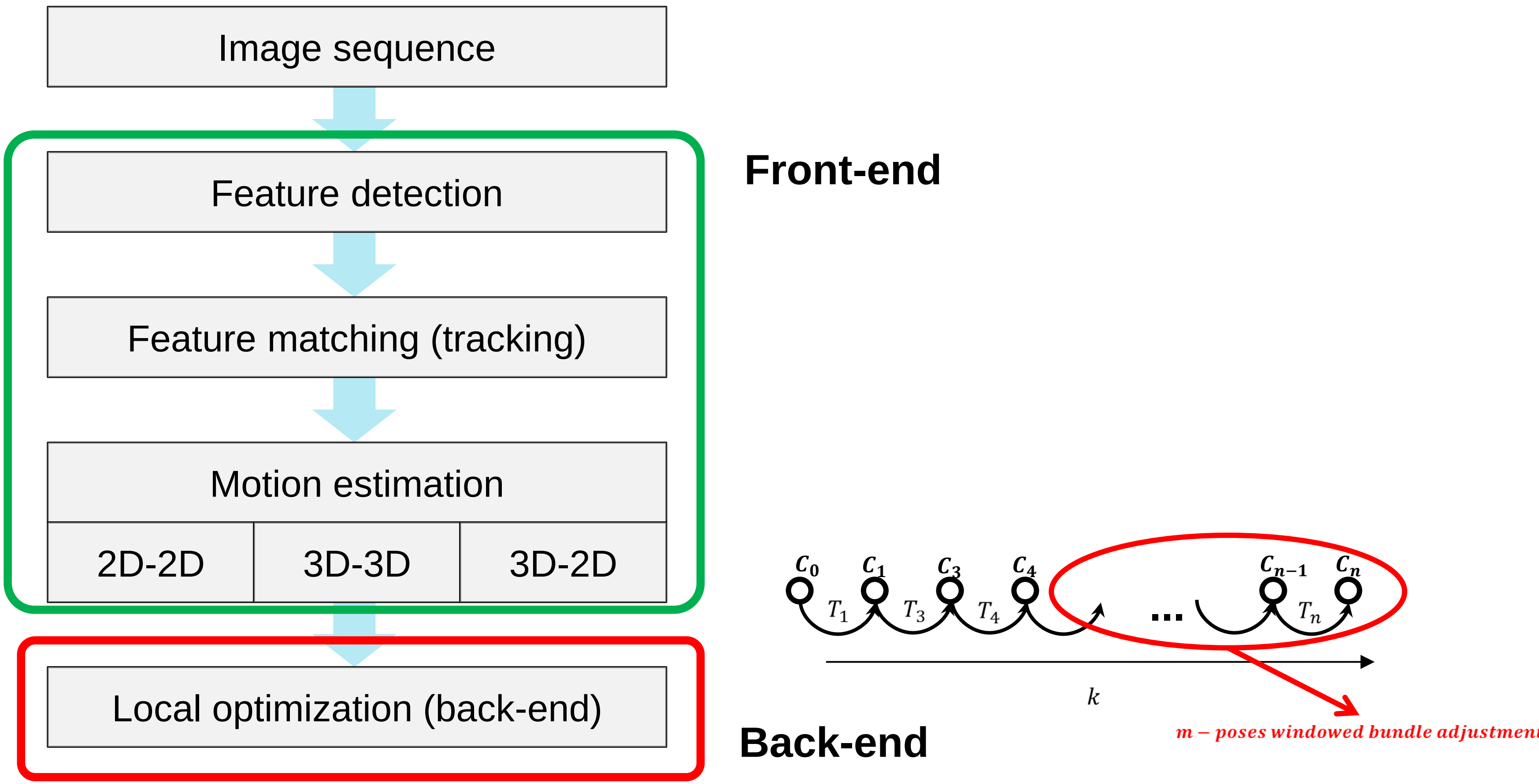


- Error at the loop closure: 6.5 m
- Error in orientation: 5 deg
- Trajectory length: 400 m

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

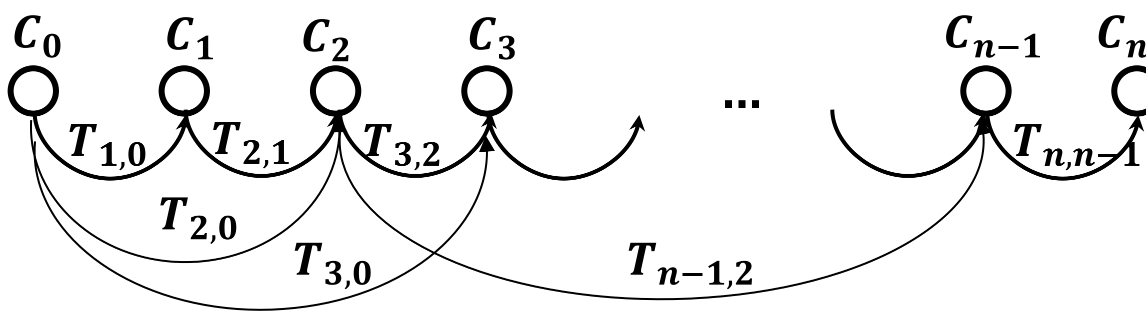
VO Flow Chart

VO computes the camera path incrementally (pose after pose)



Pose-Graph Optimization

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

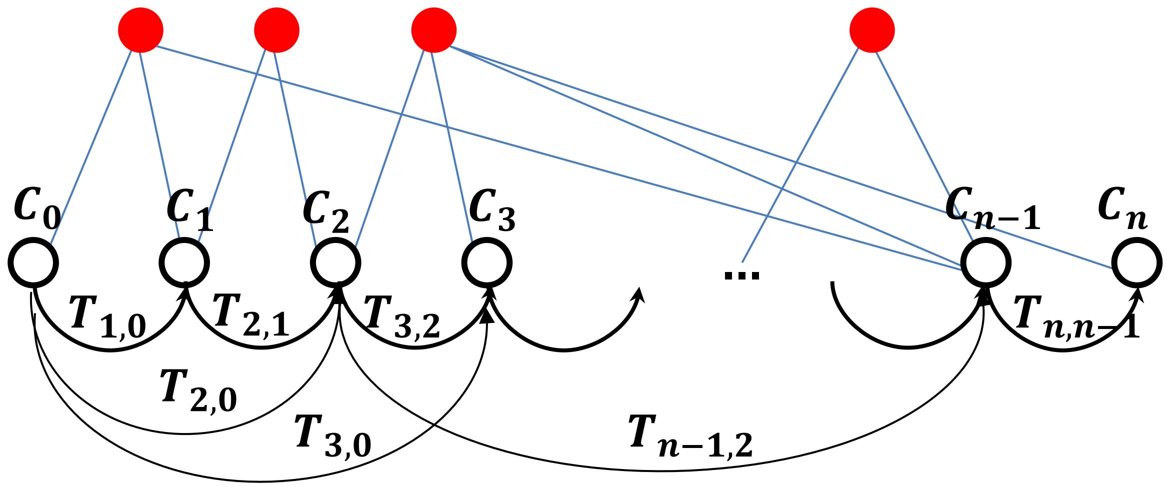


- Transformations can be computed also between non-adjacent frames T_{ij} (e.g., when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following:

$$\sum_i \sum_j \left\| C_i - T_{ij} C_j \right\|^2$$

- 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

Bundle Adjustment (BA)



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

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

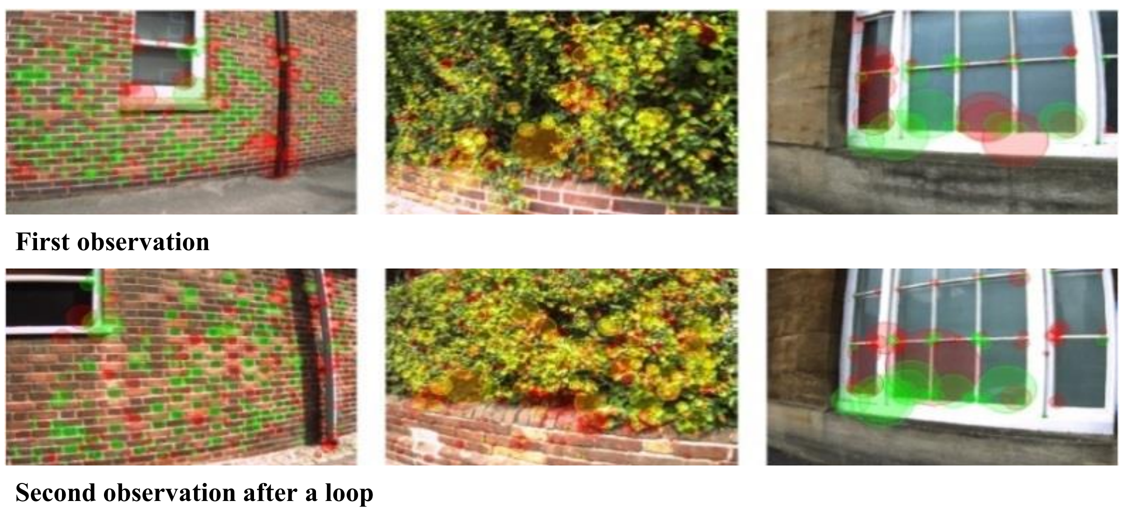
- In order to not get stuck in local minima, the initialization should be close to the minimum
- Gauss-Newton or Levenberg-Marquadt can be used. For large graphs, efficient open-source software exists: GTSAM, g2o, Google Ceres can be used.

Bundle Adjustment vs Pose-graph Optimization

- BA is **more precise** than pose-graph optimization because it adds additional constraints (*landmark constraints*)
- But **more costly**: $O\big((qM + lN)^3\big)$ 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**)

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

- 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)]



Open Source SFM for MAVs (i.e., (offline))

- MAVMAP: https://github.com/mavmap/mavmap

Closed Source SFM for MAVs (i.e., (offline))

- Pix4D: https://pix4d.com/

Open Source VO, VIO, VSLAM

VO (i.e., no loop closing)

- **Modified PTAM**: (feature-based, mono): http://wiki.ros.org/ethzasl_ptam
- **LIBVISO2** (feature-based, mono and stereo): http://www.cvlibs.net/software/libviso
- **SVO** (semi-direct, mono, stereo, multi-cameras): https://github.com/uzh-rpg/rpg_svo

VIO

- **ROVIO** (tightly coupled EKF): https://github.com/ethz-asl/rovio
- **OKVIS** (non-linear optimization): https://github.com/ethz-asl/okvis

VSLAM

- **ORB-SLAM** (feature based, mono and stereo): https://github.com/raulmur/ORB_SLAM
- **LSD-SLAM** (semi-dense, direct, mono): https://github.com/tum-vision/lsd_slam

Open Source VO, VIO **for MAVs**

VO (i.e., no loop closing)

- **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

IMU-Vision fusion:

- **Multi-Sensor Fusion Package (MSF)** (Weiss et al.) - EKF, loosely-coupled: http://wiki.ros.org/ethzasl_sensor_fusion
- **SVO + GTSAM (Forster et al. RSS'15**) (optimization based, pre-integrated IMU): https://bitbucket.org/gtborg/gtsam
  - Instructions here: http://arxiv.org/pdf/1512.02363

Open Source Optimization Tools

- GTSAM: https://collab.cc.gatech.edu/borg/gtsam?destination=node%2F299
- G2o: https://openslam.org/g2o.html
- Google Ceres Solver: http://ceres-solver.org/

Place Recognition

- DBoW2: https://github.com/dorian3d/DBoW2
- FABMAP: http://mrg.robots.ox.ac.uk/fabmap/

MAV Datasets

These datasets include ground-truthed 6-DOF poses from Vicon and synchronized IMU and images:

- EUROC MAV Dataset (forward-facing stereo): http://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets
- RPG-UZH dataset (downward-facing monocular) http://rpg.ifi.uzh.ch/datasets/dalidation.bag

Other Older Software and Datasets

SOFTWARE AND DATASETS

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

Other Older Software and Datasets

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

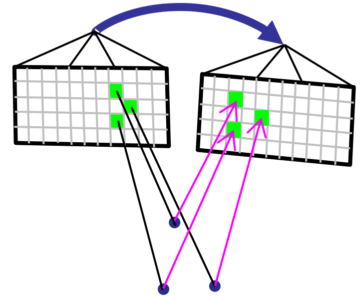
SVO: Fast, Semi-Direct Visual Odometry

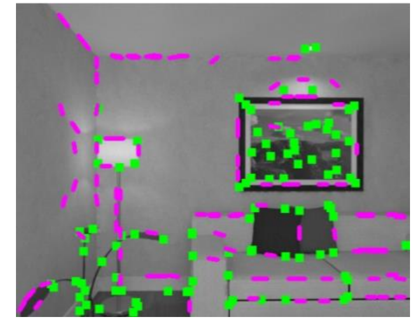
[Forster, Pizzoli, Scaramuzza, ICRA’14]

SVO Workflow

Direct

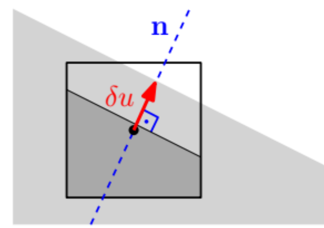
- Frame-to-frame motion estimation



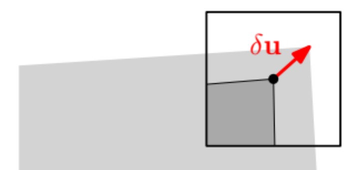


Feature-based

- Frame-to-Keyframe pose refinement



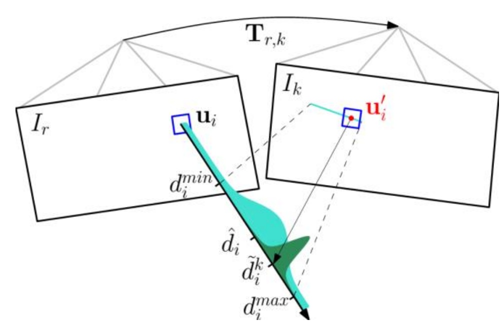
Edgelet



Corner

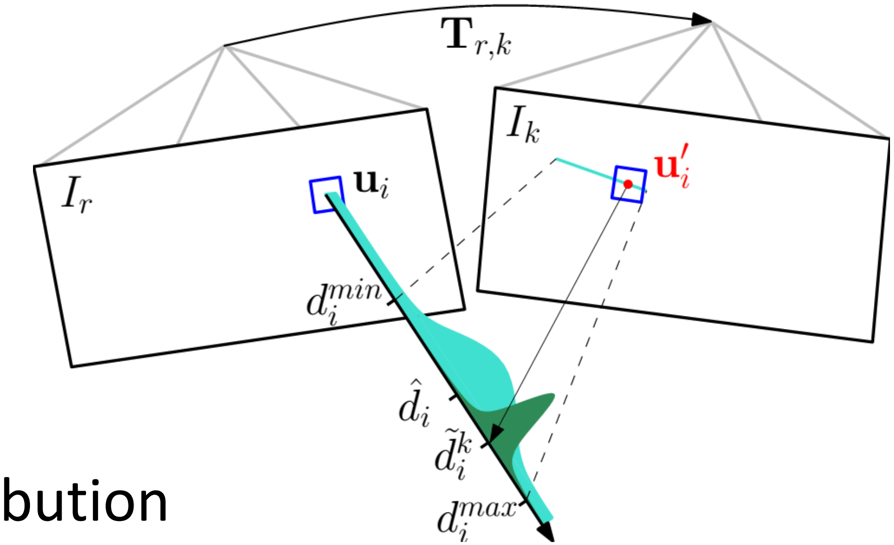
Mapping

➢ **Probabilistic depth** estimation of 3D points



[Forster, Pizzoli, Scaramuzza, «SVO: Semi Direct Visual Odometry», ICRA’14]

Probabilistic Depth Estimation

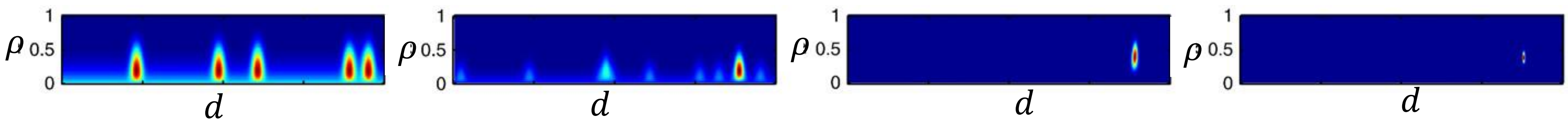


Depth-Filter:

- **Depth Filter** for every feature
- **Recursive Bayesian** depth estimation

Mixture of Gaussian + Uniform distribution

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



[Forster, Pizzoli, Scaramuzza, SVO: Semi Direct Visual Odometry, IEEE ICRA'14]

Processing Times of SVO

Laptop (Intel i7, 2.8 GHz)

400 frames per second

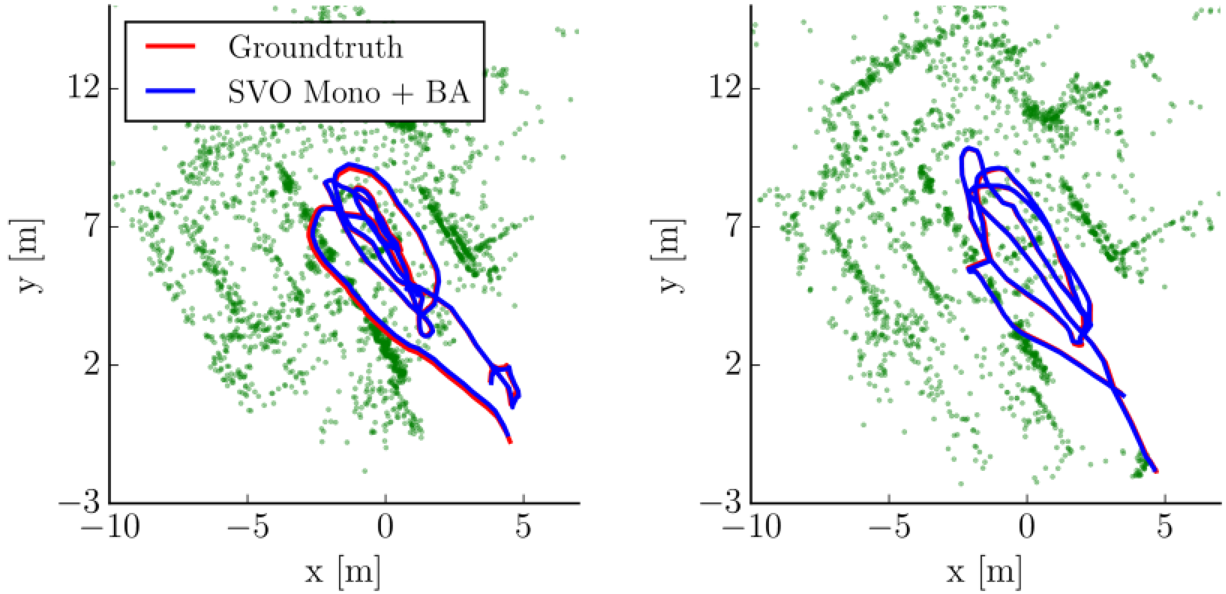
Embedded ARM Cortex-A9, 1.7 GHz

Up to 70 frames per second

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

Accuracy and Timing



	Euroc 1 RMS Error	Euroc 2 RMS Error	Timing	CPU @ 20 fps
SVO	0.26 m	0.65 m	**2.53 ms**	**55 %**
SVO + BA	**0.06 m**	**0.07 m**	5.25 ms	72 %
ORB SLAM	0.11 m	0.19 m	29.81 ms	187 %
LSD SLAM	0.13 m	0.43 m	23.23 ms	236 %

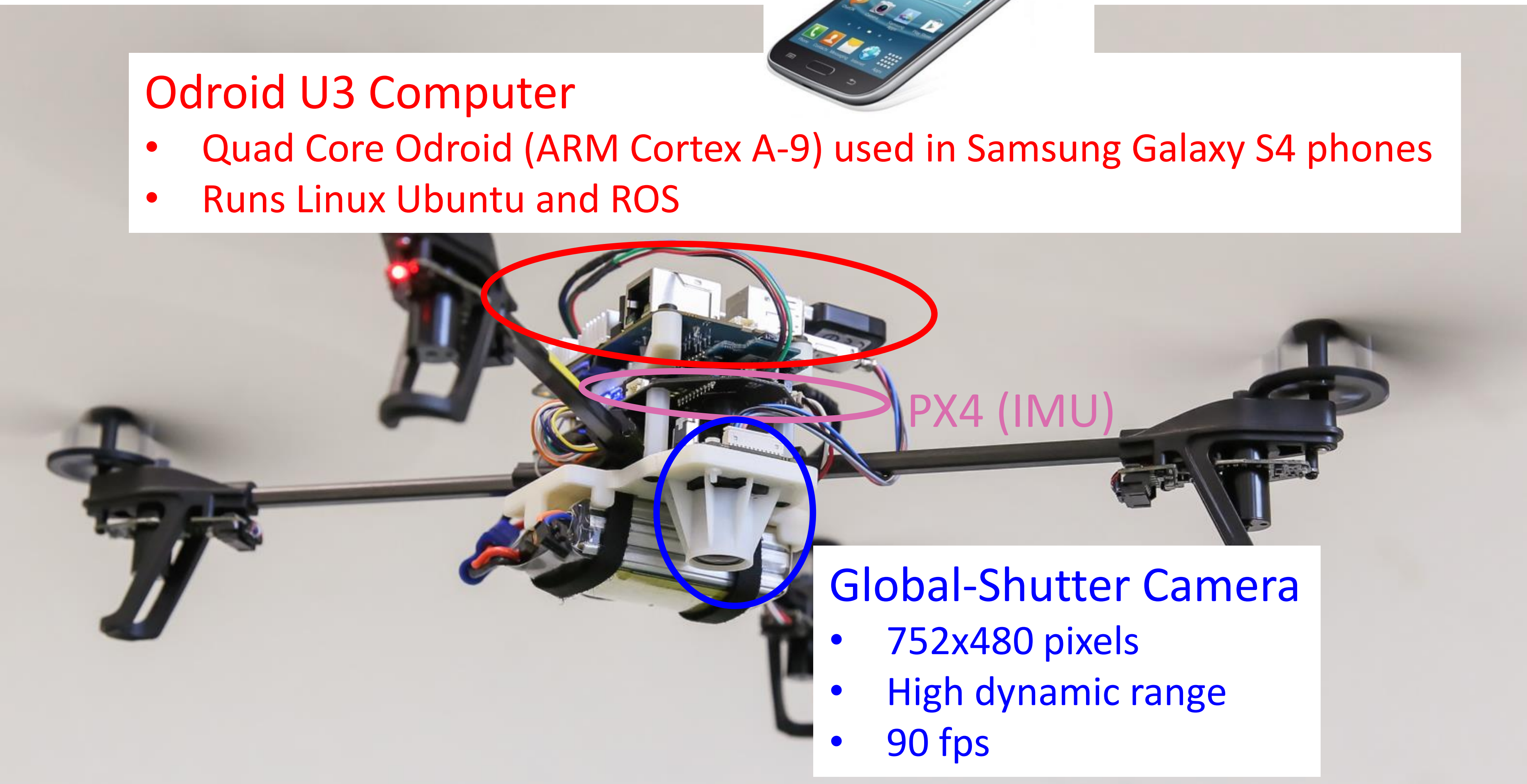
Intel i7, 2.80 GHz

Integration on a Quadrotor Platform

Quadrotor System

Odroid U3 Computer

- Quad Core Odroid (ARM Cortex A-9) used in Samsung Galaxy S4 phones
- Runs Linux Ubuntu and ROS

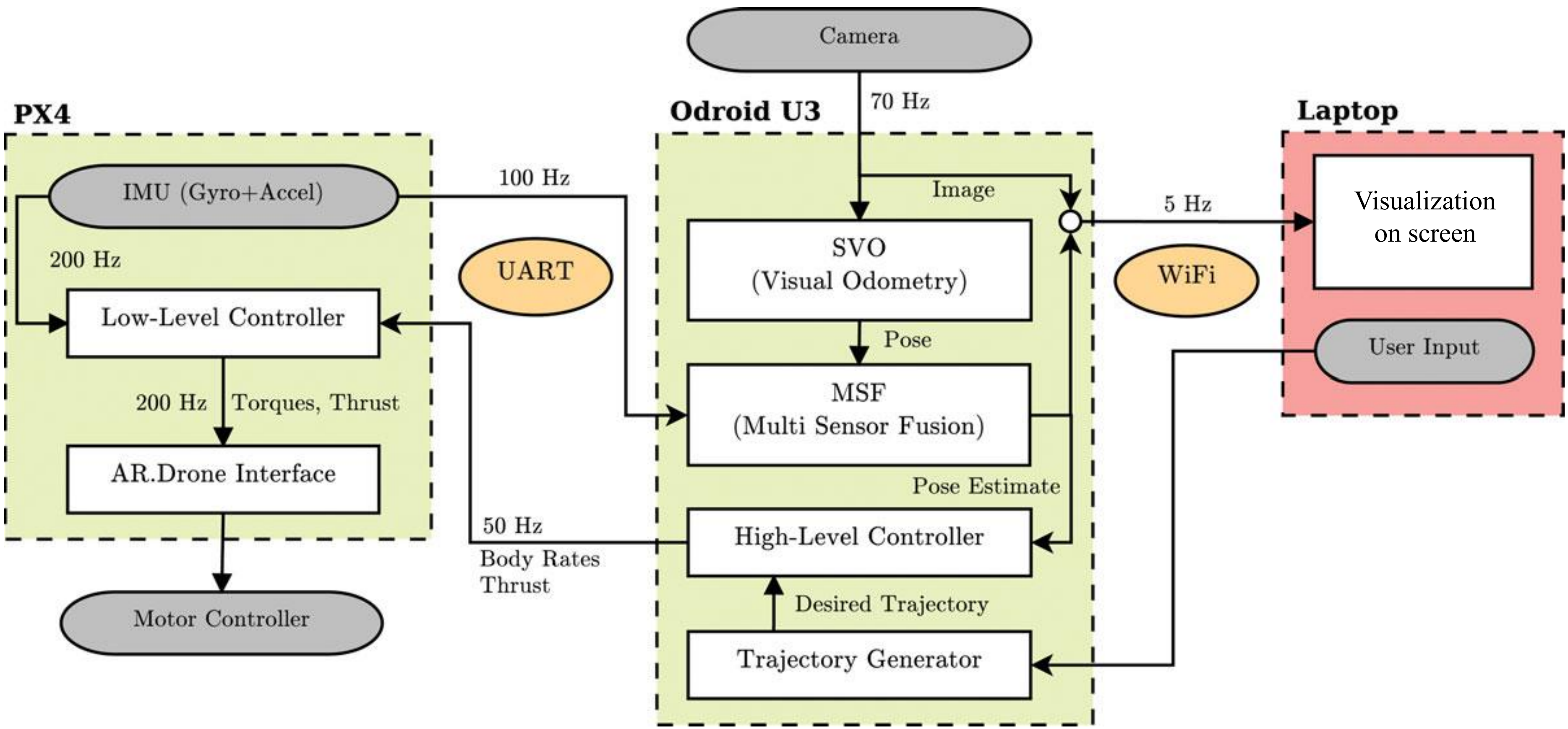


Global-Shutter Camera

- 752x480 pixels
- High dynamic range
- 90 fps

450 grams!

Control Structure



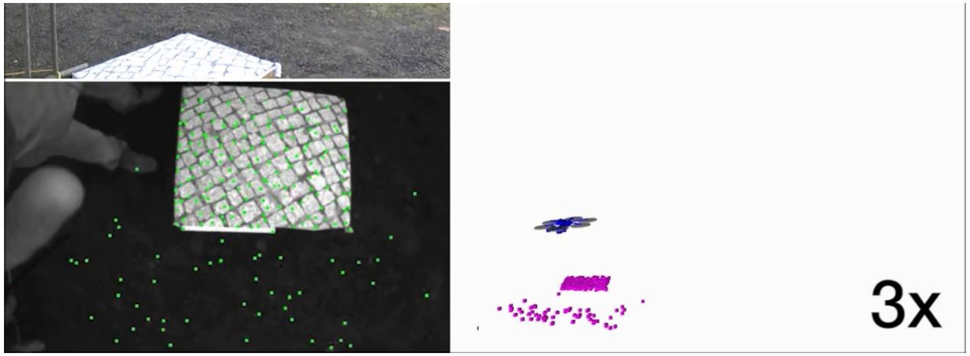
Indoors and outdoors experiments

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

https://www.youtube.com/watch?v=4X6Voft4Z_0

Speed: 4 m/s, height: 1.5 m – Down-looking camera

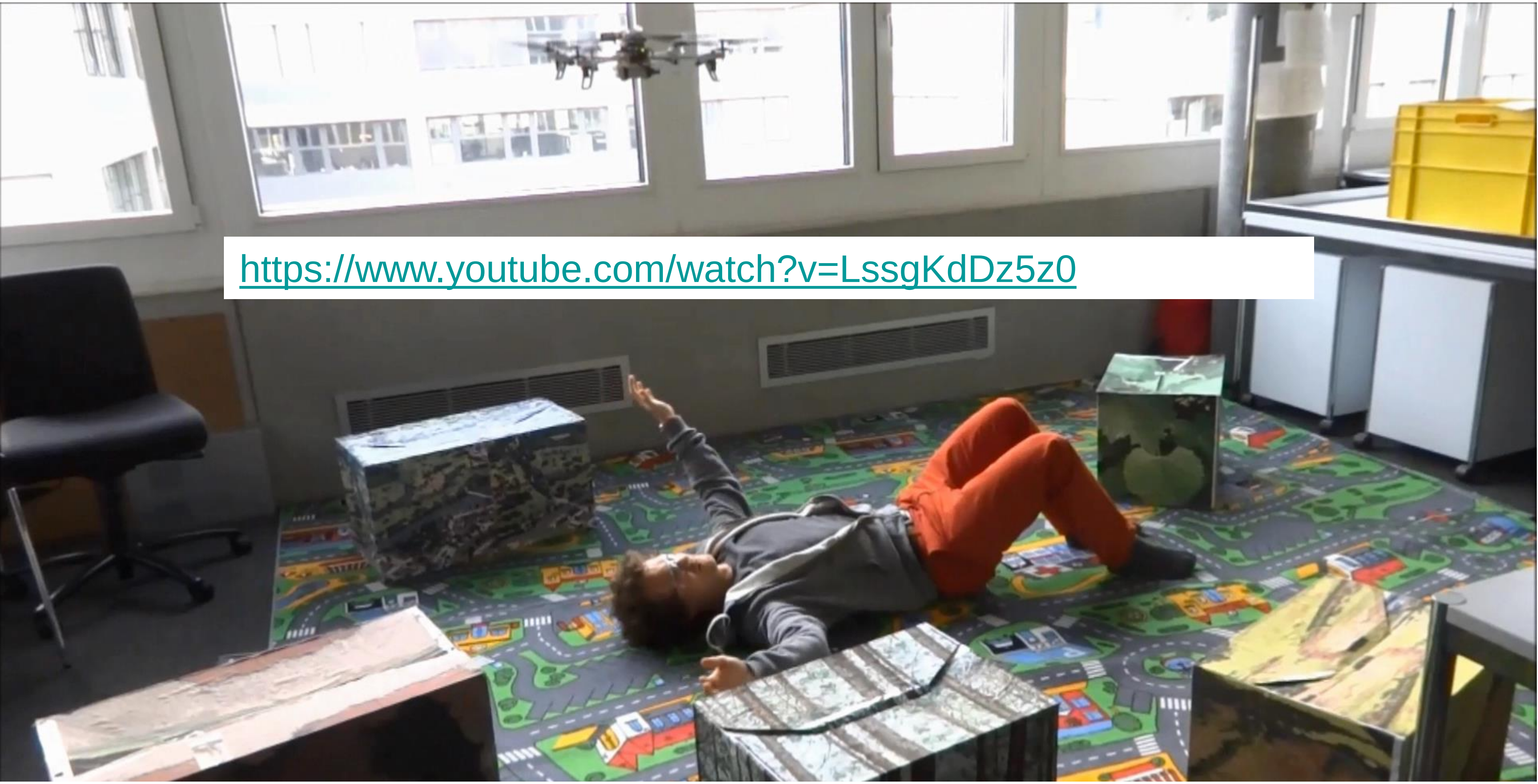
https://www.youtube.com/watch?v=3mNY9-DSUDk



Faessler, Fontana, Forster, Mueggler, Pizzoli, Scaramuzza, Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle, **Journal of Field Robotics, 2015**.

Robustness to Dynamic Objects and Occlusions

- Depth uncertainty is crucial for safety and robustness
- Outliers are caused by wrong data association (e.g., moving objects, distortions)
- Probabilistic depth estimation models outliers



https://www.youtube.com/watch?v=LssgKdDz5z0

Faessler, Fontana, Forster, Mueggler, Pizzoli, Scaramuzza, Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle, **Journal of Field Robotics, 2015**.

Robustness: Adaptiveness and Reconfigurability [ICRA'15]

Automatic recovery from aggressive flight; fully onboard, single camera, no GPS

https://www.youtube.com/watch?v=pGU1s6Y55JI

Faessler, Fontana, Forster, Scaramuzza, Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor, ICRA'15. **Demonstrated at ICRA'15** and featured on **BBC News.**

Autonomous Flight, Minimum-Snap, Speed: 4 m/s