ROBOTICS & PERCEPTION GROUP

University of Zurich[UZH]

Visual Odometry and SLAM: past, present, and the robust-perception age

Davide Scaramuzza

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. 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, **Simultaneous Localization And Mapping: Present, Future, and the Robust-Perception Age**, IEEE Transactions on Robotics (cond. Accepted), 2016. PDF

Outline

- Theory
- Open Source Algorithms
- Event-based Vision

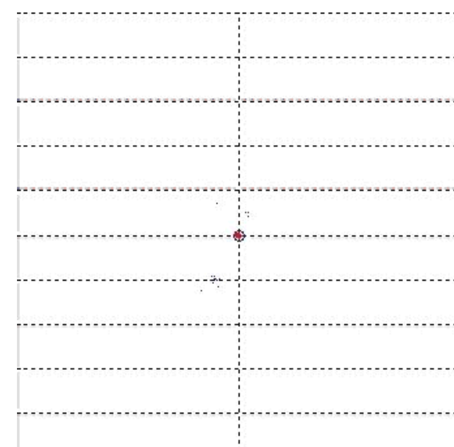
What is Visual Odometry (VO) ?

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):

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 **2004 Mars mission** (see papers from Matthies, Olson, etc. from JPL)

- **2004**: VO used on a robot on another planet: Mars rovers Spirit and Opportunity

- **2004**. VO was revived in the academic environment by David Nister «Visual Odometry» paper. The term VO became popular (and Nister became head of MS Hololens before moving to TESLA in 2014)

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

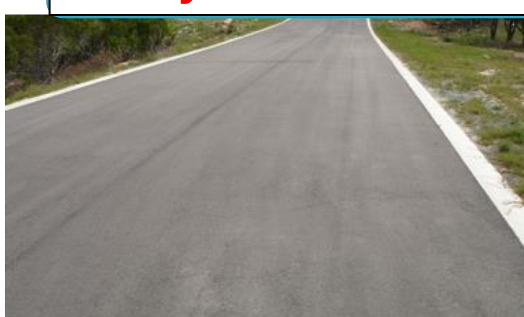
- **2004**: VO used on a robot on another planet: M

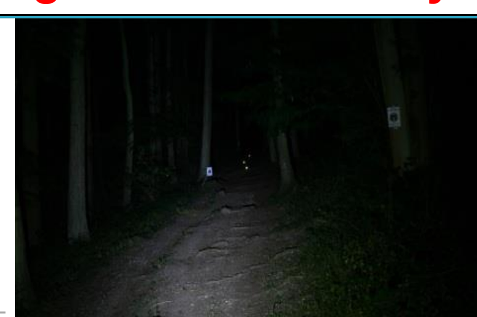
- **2004**. VO was revived in the academic environ
by David Nister «Visual Odometry» paper.
The term VO became popular (and Nister
became head of MS Hololens before moving
to TESLA in 2014)

Assumptions

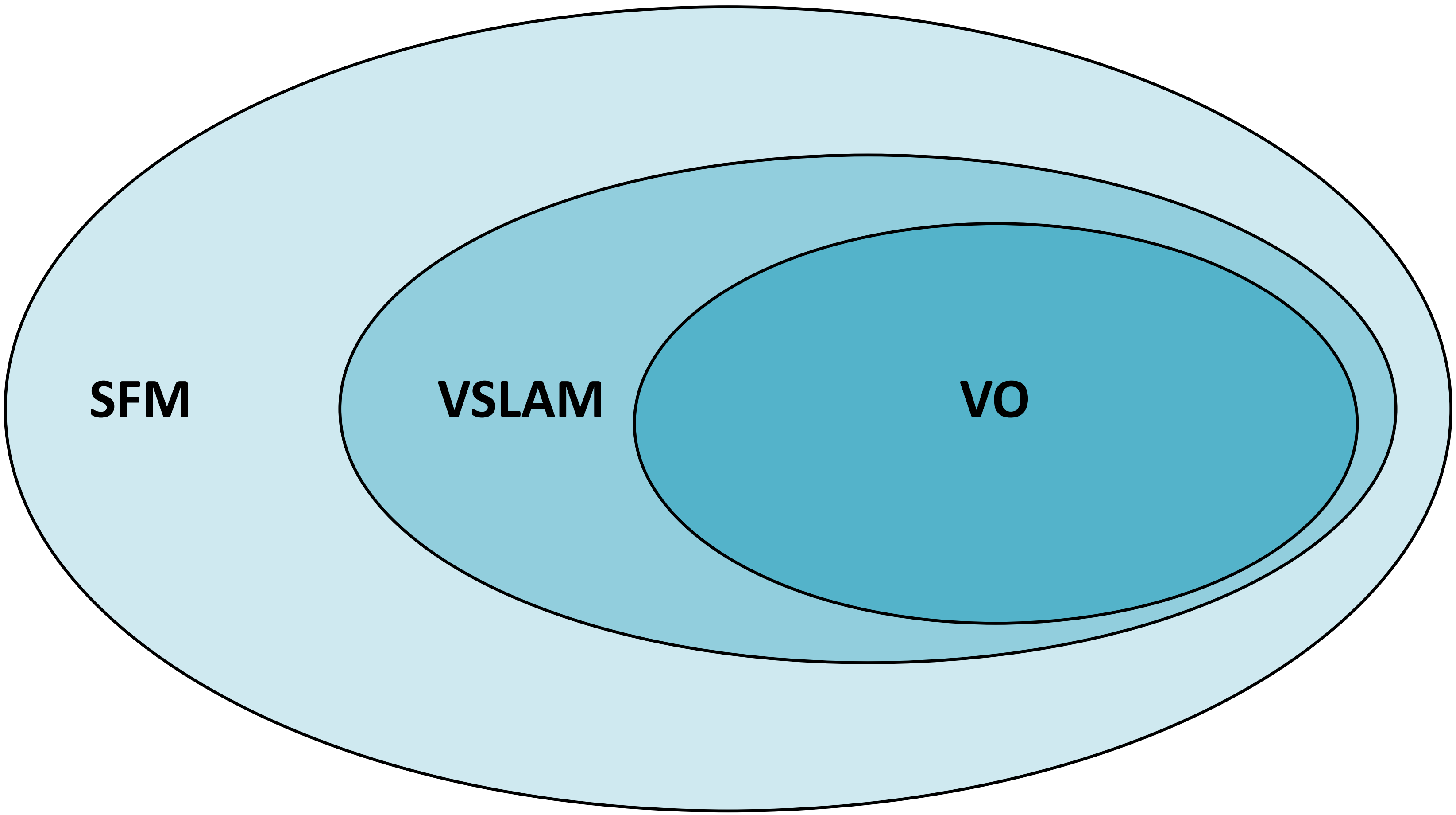
- **Sufficient illumination** in the environment
- **Dominance of static scene** over moving objects
- **Enough texture** to allow apparent motion to be extracted
- Sufficient **scene overlap** between consecutive frames

Is any of these scenes good for VO? Why?





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**.

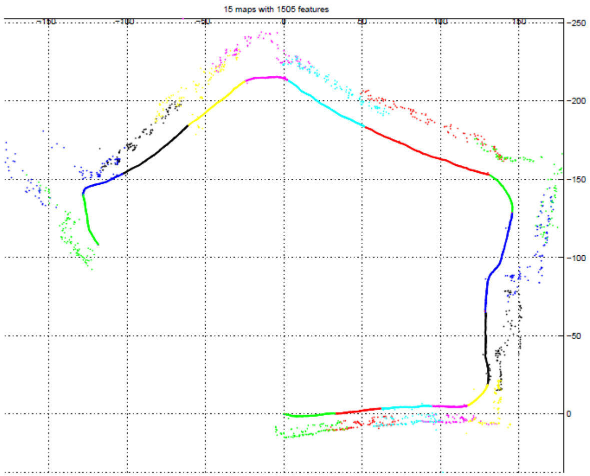
- Terminology: sometimes SFM is used as a synonym of VO

VO vs. Visual SLAM

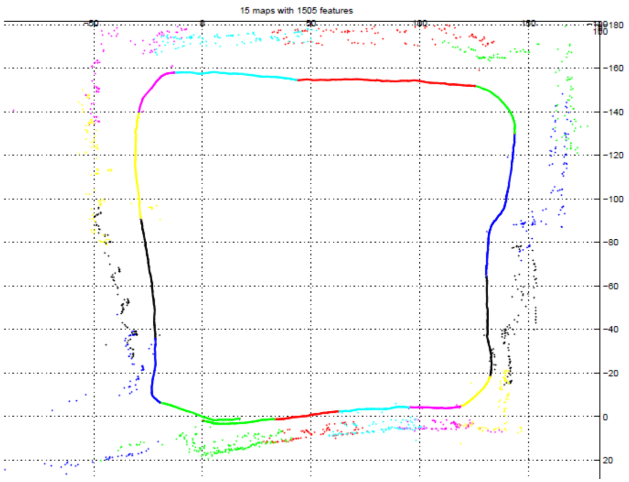
- **Visual Odometry**
  - Focus on incremental estimation/**local consistency**
- **Visual SLAM**: Simultaneous Localization And Mapping
  - Focus on **globally consistent** estimation
  - **Visual SLAM = visual odometry + loop detection + graph optimization**

- 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 et al., RSS'07]

VO Working Principle

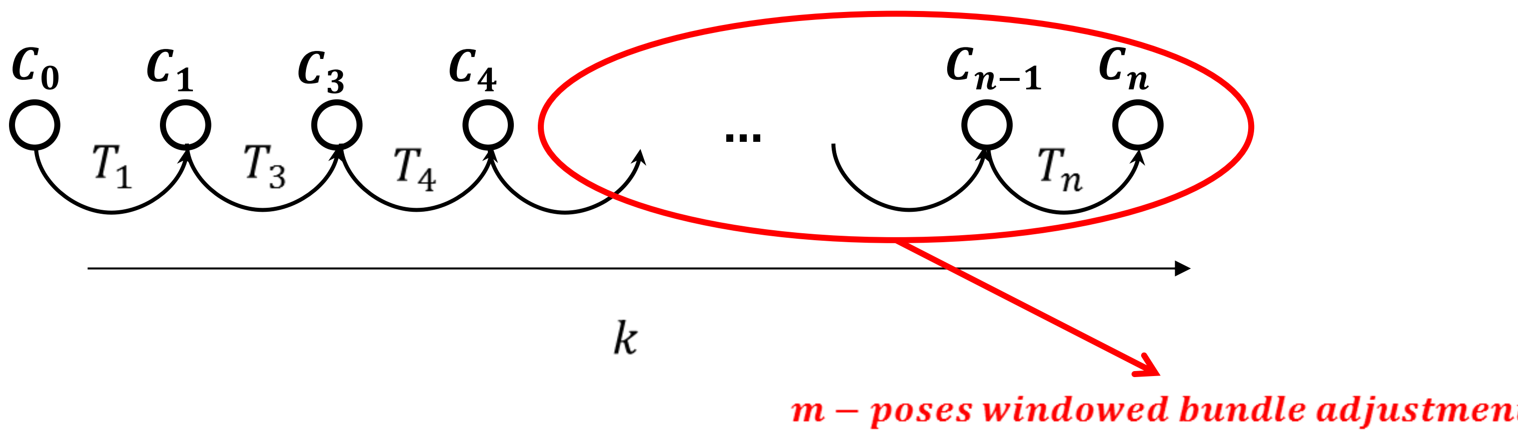
1. Compute the relative motion T_k from images I_{k-1} to image I_k

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

2. Concatenate them to recover the full trajectory

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

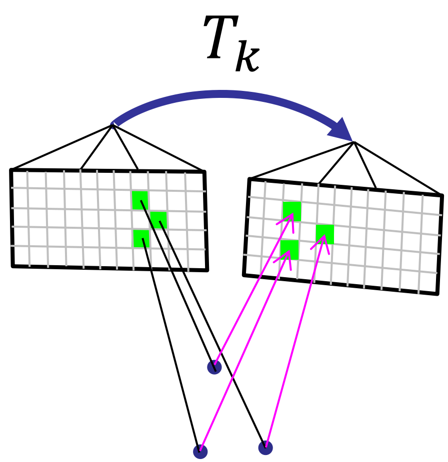
3. An optimization over the last m poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment)



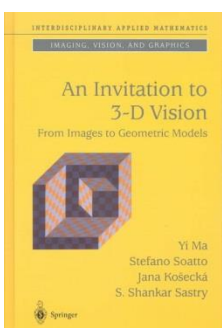
How do we estimate the relative motion T_k ?

Image I_{k-1}

Image I_k

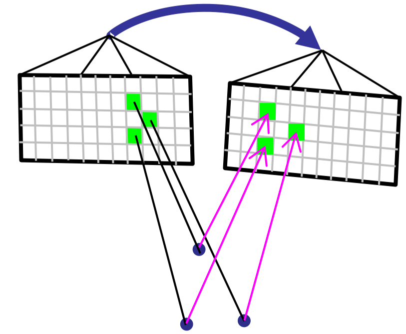


$$T_k = \arg\min_{\mathbf{T}} \iint_{\bar{\mathcal{R}}} \rho \left[I_k\Big(\pi\big(\mathbf{T} \cdot \pi^{-1}(\mathbf{u}, d_{\mathbf{u}})\big)\Big) - I_{k-1}(\mathbf{u}) \right] d\mathbf{u}$$



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

Direct Image Alignment



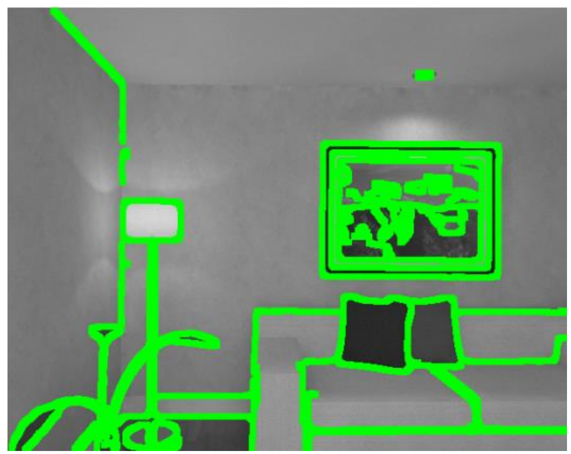
It minimizes the **per-pixel intensity difference**

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

Dense

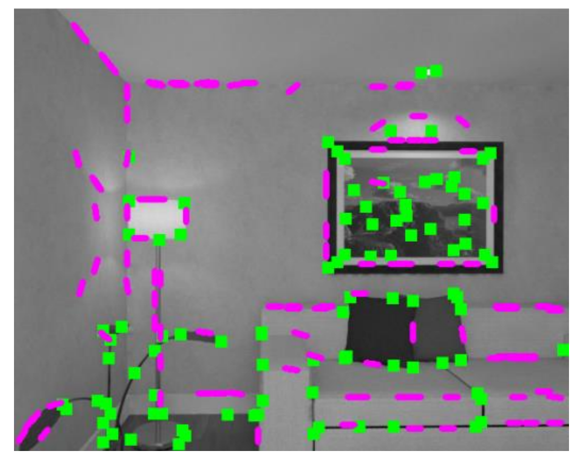
DTAM [Newcombe et al. '11]
300'000+ pixels

Semi-Dense



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

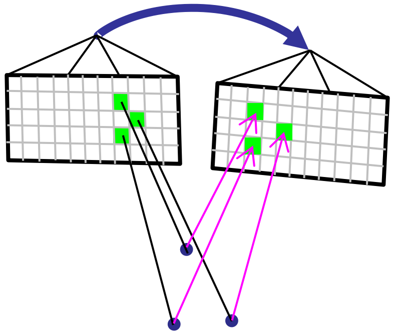
Sparse



SVO [Forster et al. 2014, TRO'16]
100-200 features x 4x4 patch
~ 2,000 pixels

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

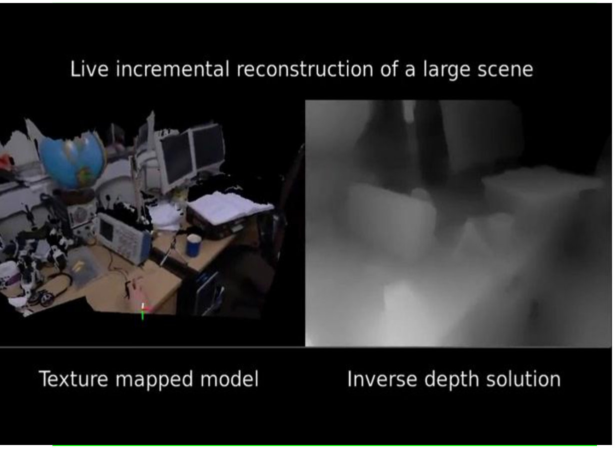
Direct Image Alignment



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

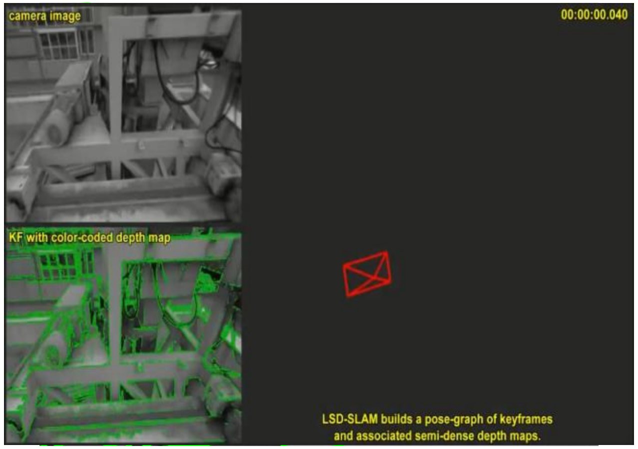
$$T_{k,k-1} = \arg\min_{T} \sum_{i} \|I_k(\boldsymbol{u}'_i) - I_{k-1}(\boldsymbol{u}_i)\|_{\sigma}^{2}$$

Dense



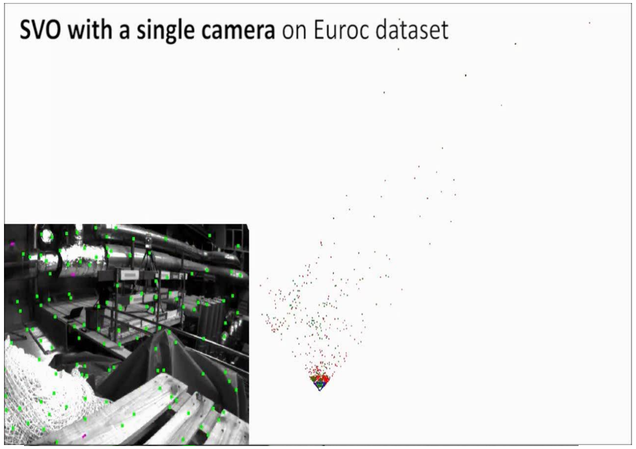
DTAM [Newcombe et al. ‘11]
300,000+ pixels

Semi-Dense



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

Sparse

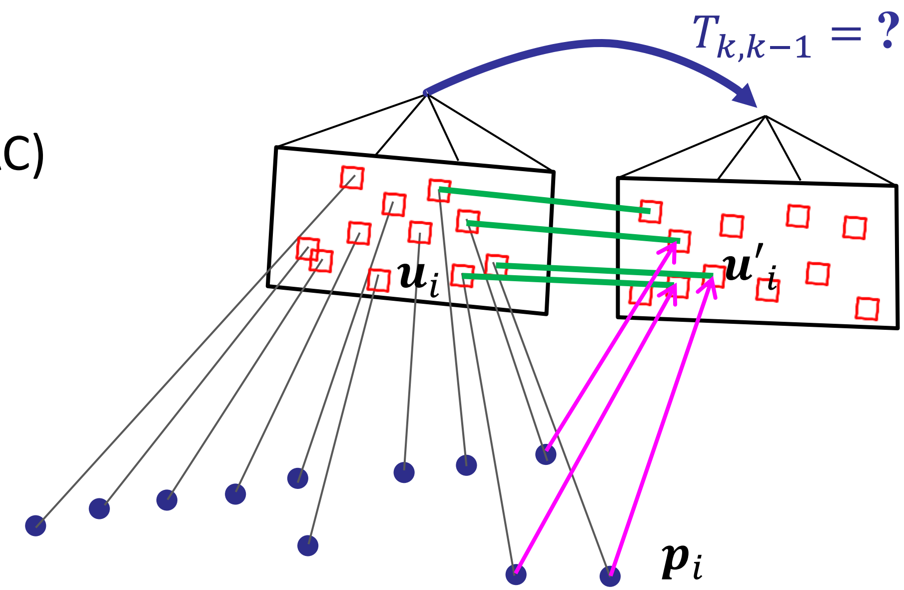


SVO [Forster et al. 2014]
100-200 features x 4x4 patch
~ 2,000 pixels

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

Feature-based methods

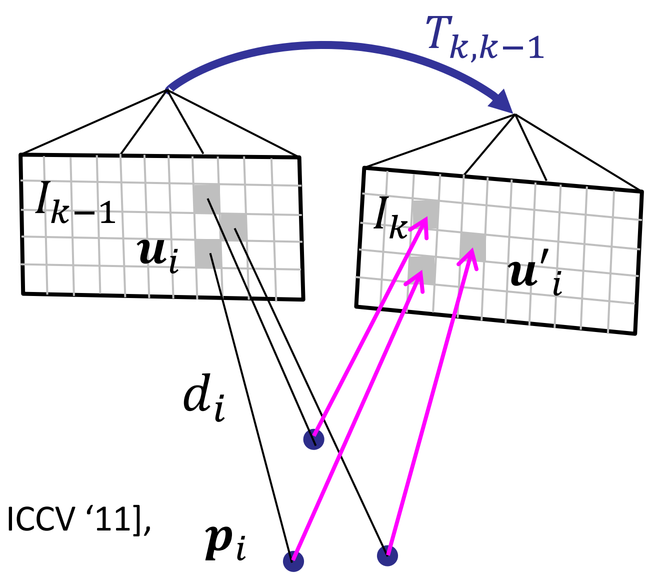
1. Extract & match features (+RANSAC)
2. Minimize **Reprojection error** minimization



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

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

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

✓ **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**

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

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**

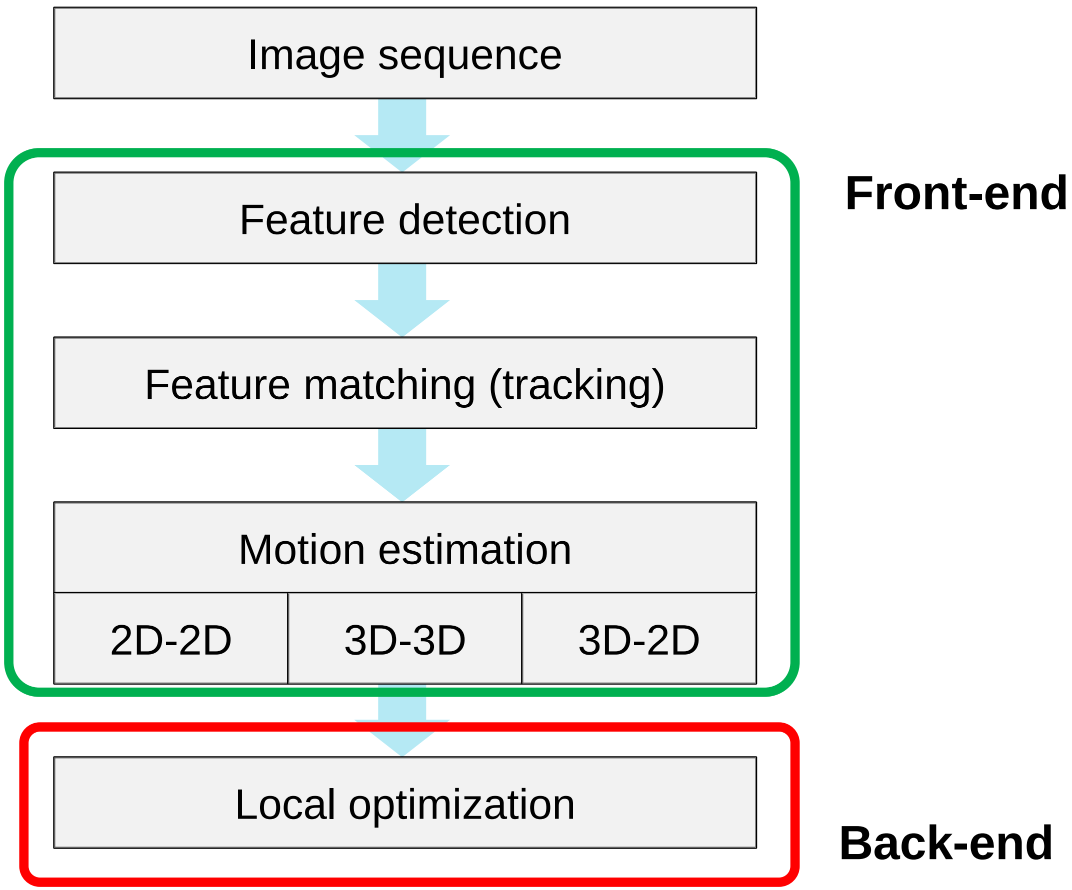
✗ **Limited frame-to-frame motion**

✗ **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]

VO Flow Chart

VO computes the camera path incrementally (pose after pose)



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)

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

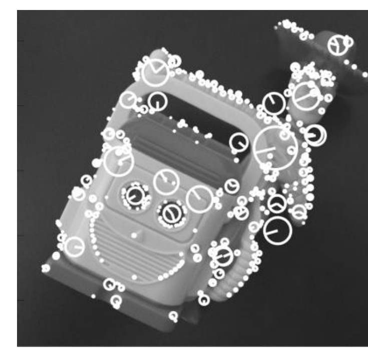
Example features tracks

Feature Extraction

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*

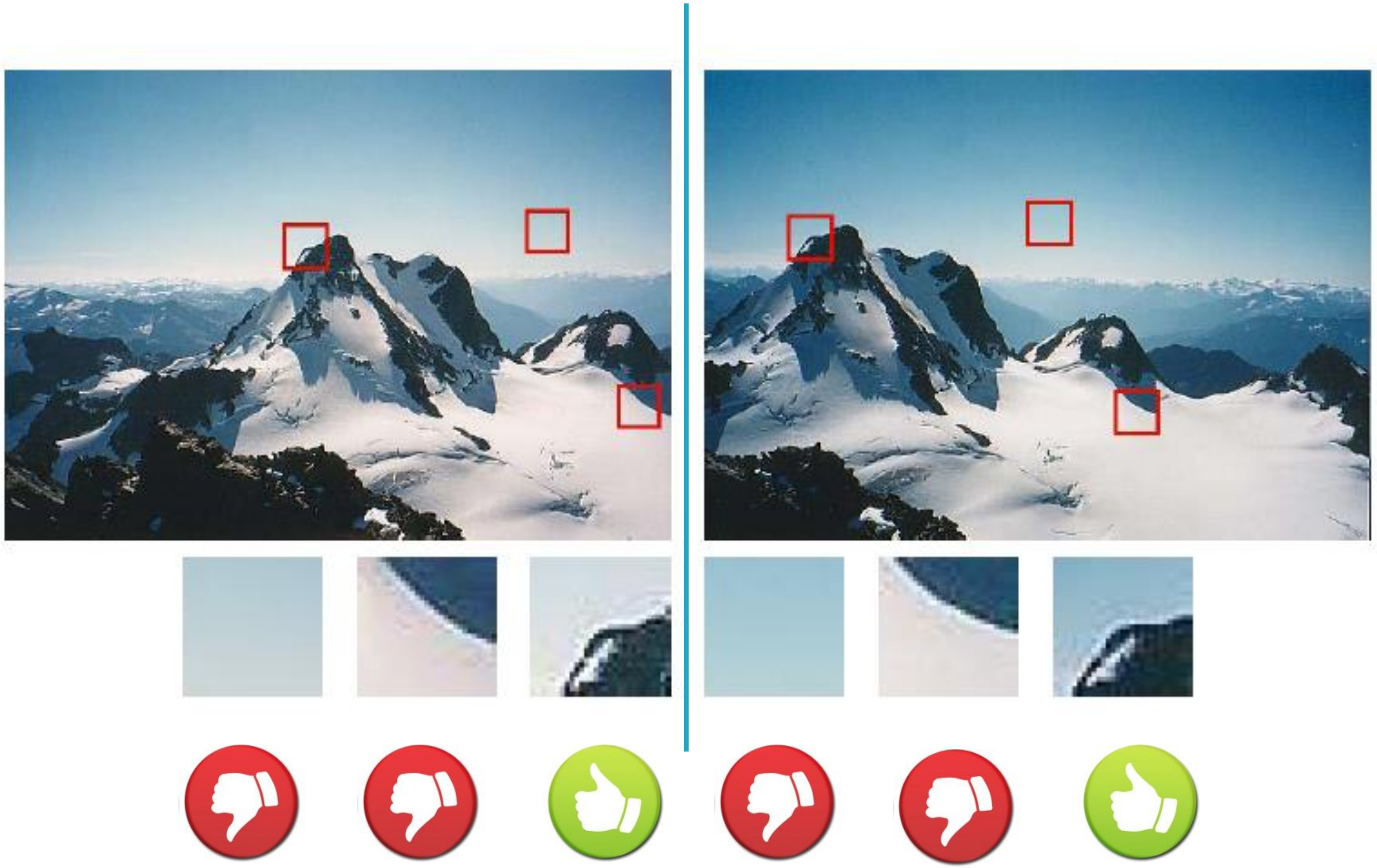
- A **blob** is any other image pattern, **which is not a corner**, that significantly differs from its neighbors in intensity and texture
  - **Has less localization accuracy than a corner**
    - **Blob detectors are better for place recognition**
  - It's **more distinctive than a corner**
  - E.g.,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



- ***Descriptor**: Distinctive feature **identifier***
  - ***Standard** descriptor: squared patch of pixel intensity values*
  - ***Gradient** or difference-based descriptors: SIFT, SURF, ORB, BRIEF, BRISK*

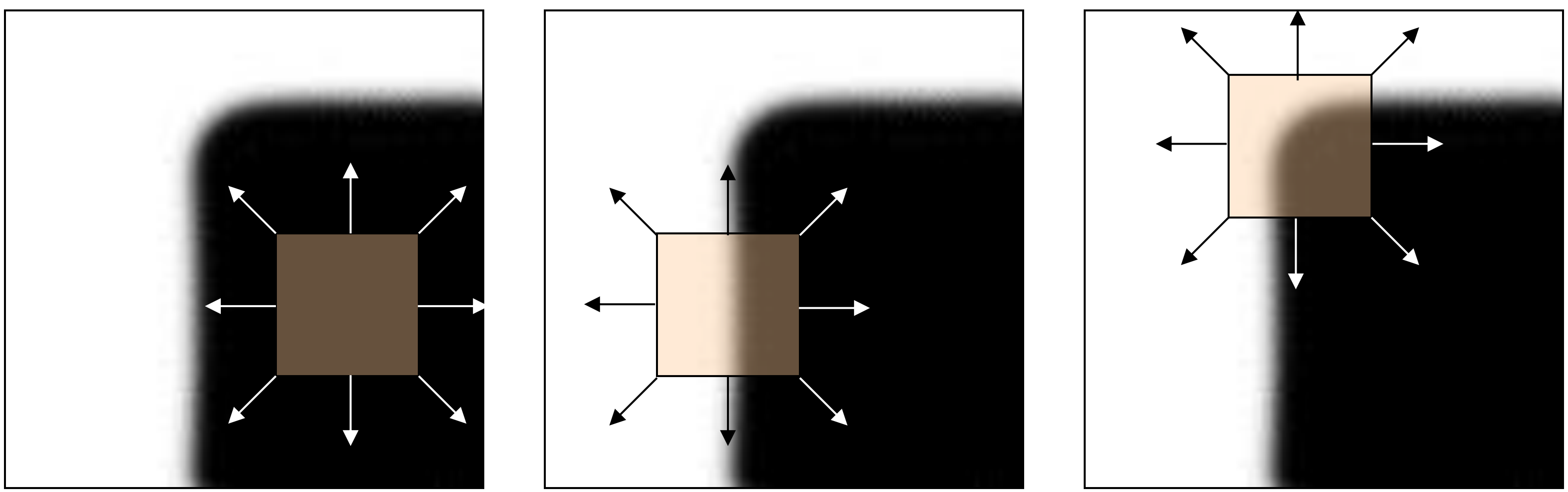
What are Good Features to Track ?

Which of the patches below can be matched reliably?



Harris Corners (1988)

- How do we identify corners?
- We can easily recognize the point by looking through a small window
- Shifting a window in **any direction** should give a **large change** in intensity in at least 2 directions



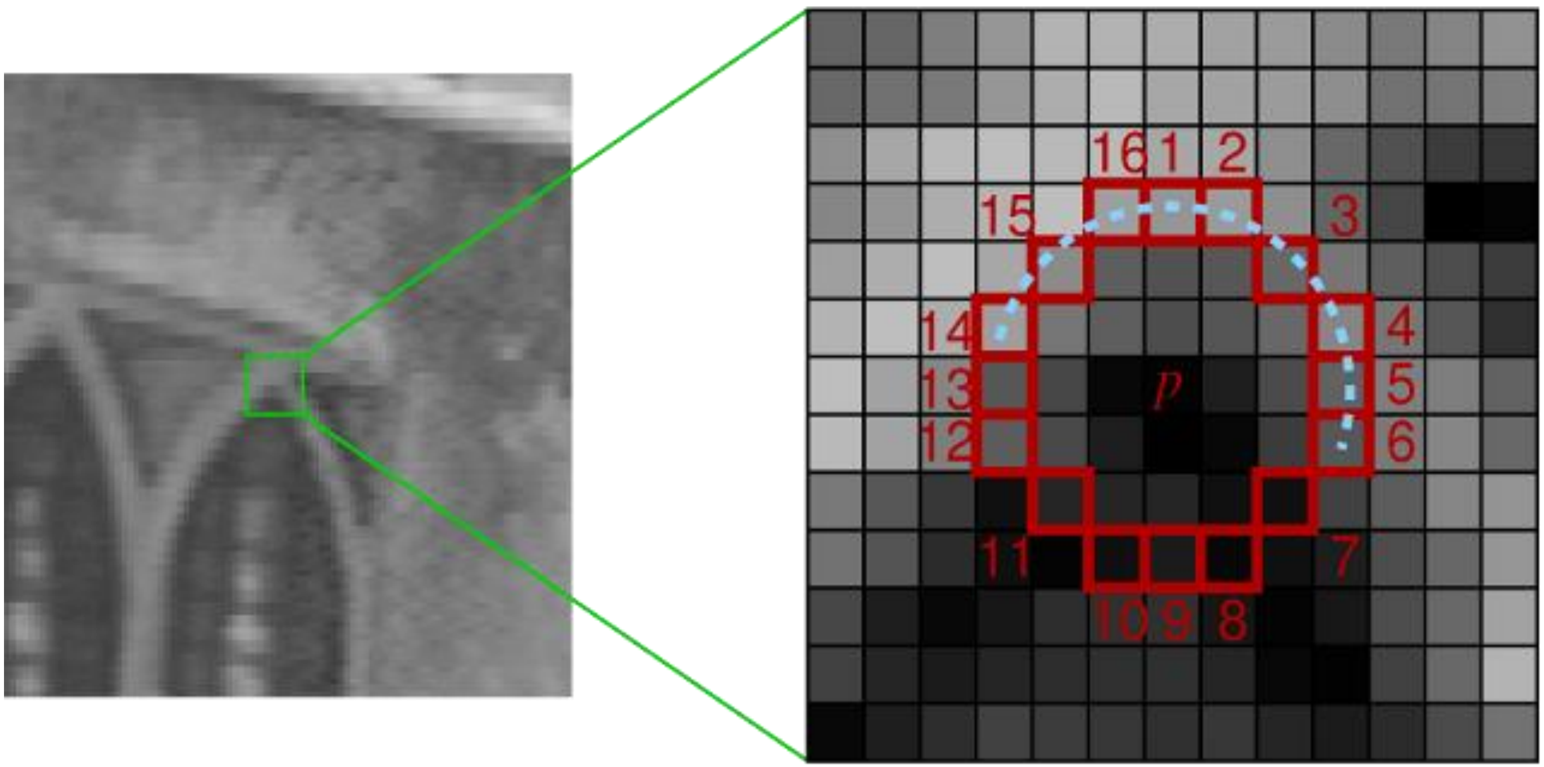
"flat" region: no intensity change

"edge": no change along the edge direction

"corner": significant change in at least 2 directions

FAST corner detector **[Rosten et al., PAMI 2010]**

- FAST: Features from Accelerated Segment Test
- Studies intensity of pixels on circle around candidate pixel *C*
- *C* is a FAST corner if a set of *N* contiguous pixels on circle are:
  - all brighter than *intensity_of(C)+theshold*, or
  - all darker than *intensity_of(C)+theshold*

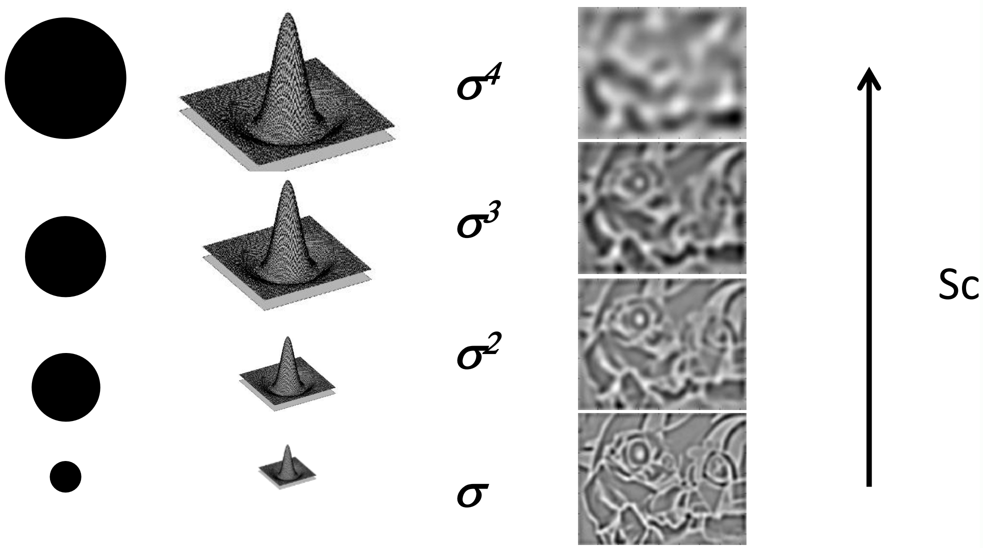


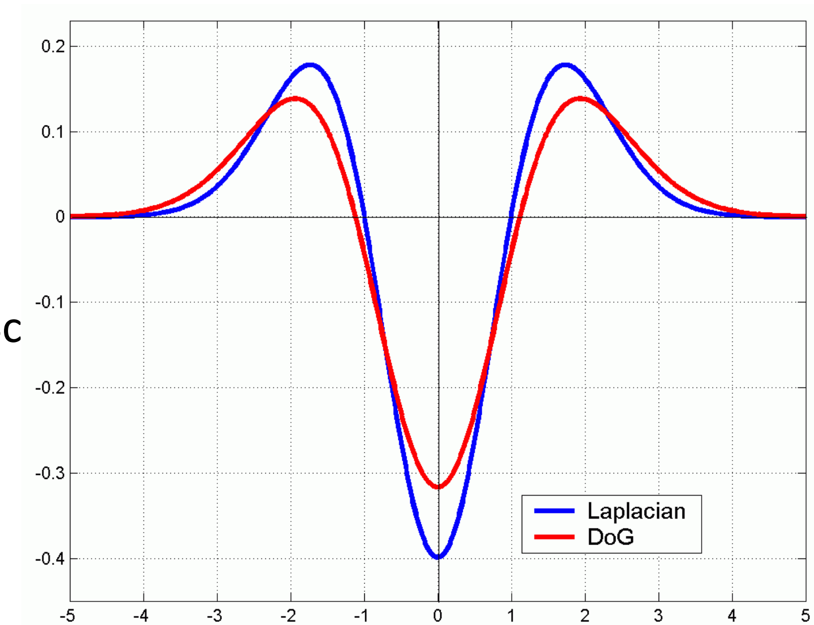
- Typical FAST mask: test for 9 contiguous pixels in a 16-pixel circle
- Very fast detector - in the order of 100 Mega-pixel/second

SIFT

SIFT responds to local regions that look like Difference of Gaussian (~Laplacian of Gaussian)

$$LOG \approx DoG = G_{k\sigma}(x,y) - G_{\sigma}(x,y)$$



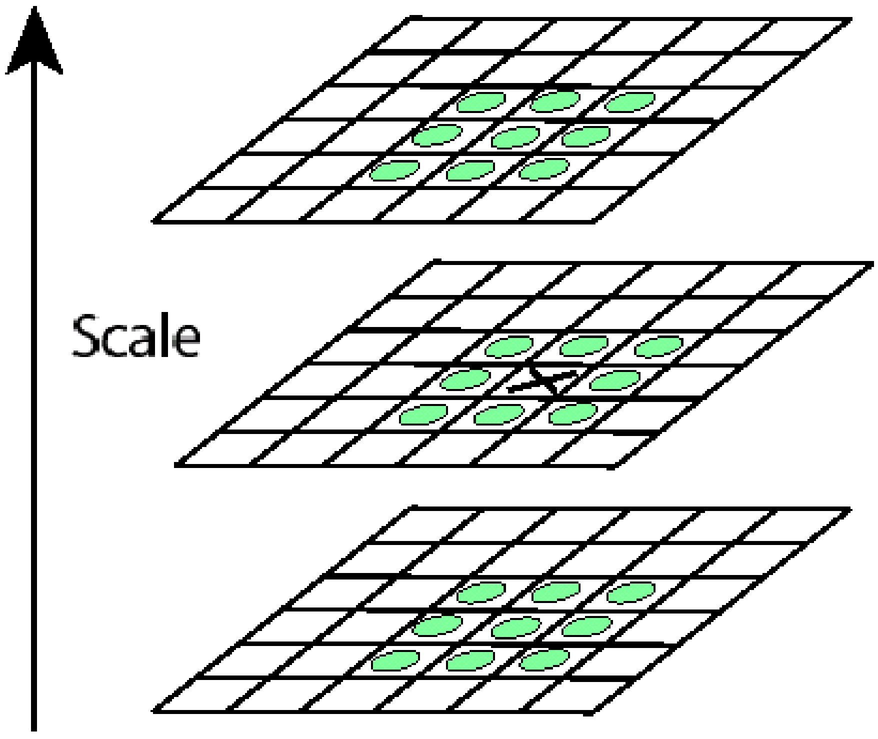


SIFT detector (location + scale)

SIFT keypoints: local extrema in both location and scale of the DoG

- Detect maxima and minima of difference-of-Gaussian in scale space

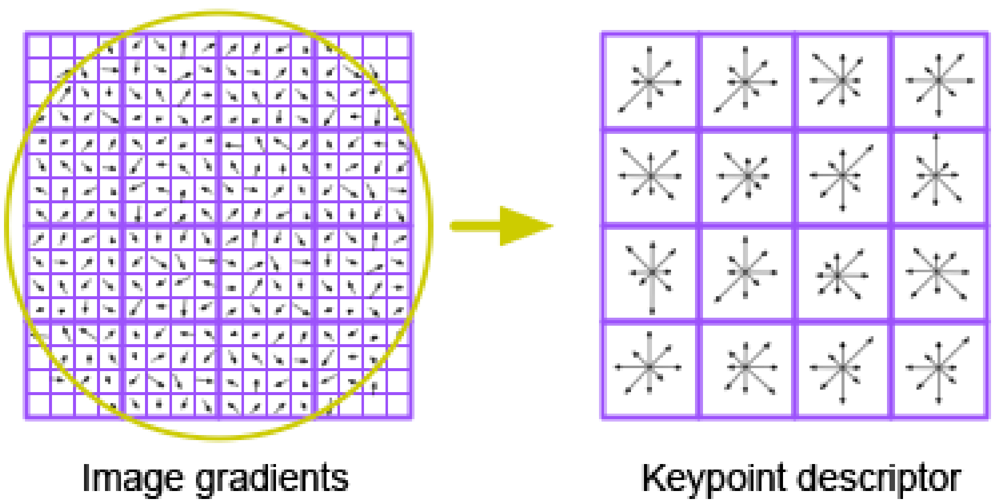
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below



For each max or min found, output is the **location** and the **scale**.

SIFT descriptor

- **S**cale **I**nvariant **F**eature **T**ransform
- Invented by David Lowe [IJCV, 2004]
- Descriptor computation:
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting SIFT descriptor: 4x4x8 = 128 values
  - Descriptor Matching: Euclidean-distance between these descriptor vectors (i.e., SSD)



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* , 2004.

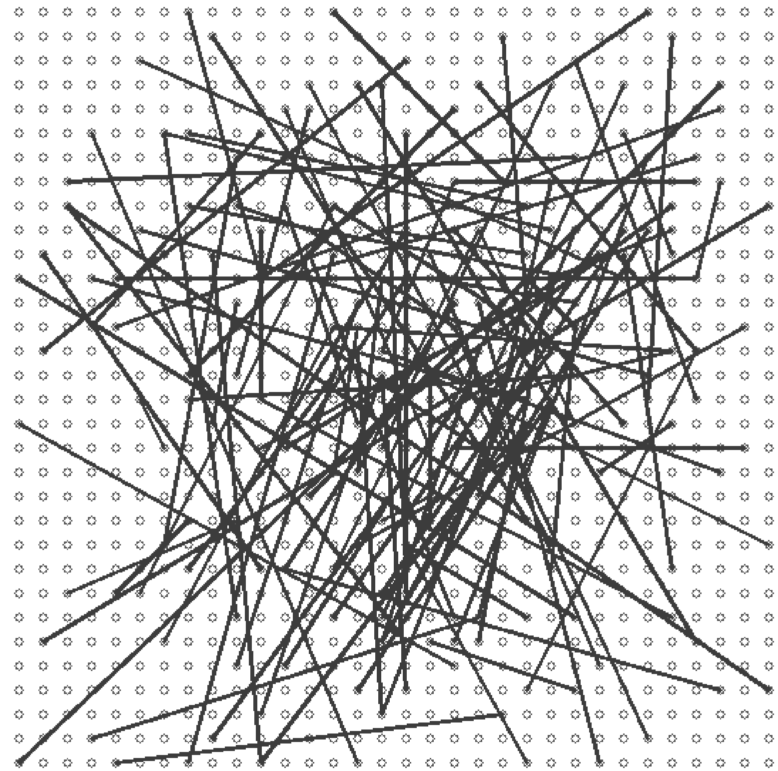
BRIEF descriptor [Calonder et. al, ECCV 2010]

- **B**inary **R**obust **I**ndependent **E**lementary **F**eatures
- Goal: high speed (in description and matching)

- **Binary** descriptor formation:
  - Smooth image
  - **for each** detected keypoint (e.g. FAST),
  - ***sample*** 256 intensity pairs **p**=(p_1, p_2) within a squared patch around the keypoint
  - **for each pair p**
    - **if** $p_1 < p_2$ **then *set*** bit **p** of descriptor to **1**
    - **else *set*** bit **p** of descriptor to **0**

- The pattern is generated randomly only once; then, the same pattern is used for all patches

- Not scale/rotation invariant
- Allows **very fast** Hamming Distance matching: count the number of bits that are different in the descriptors matched

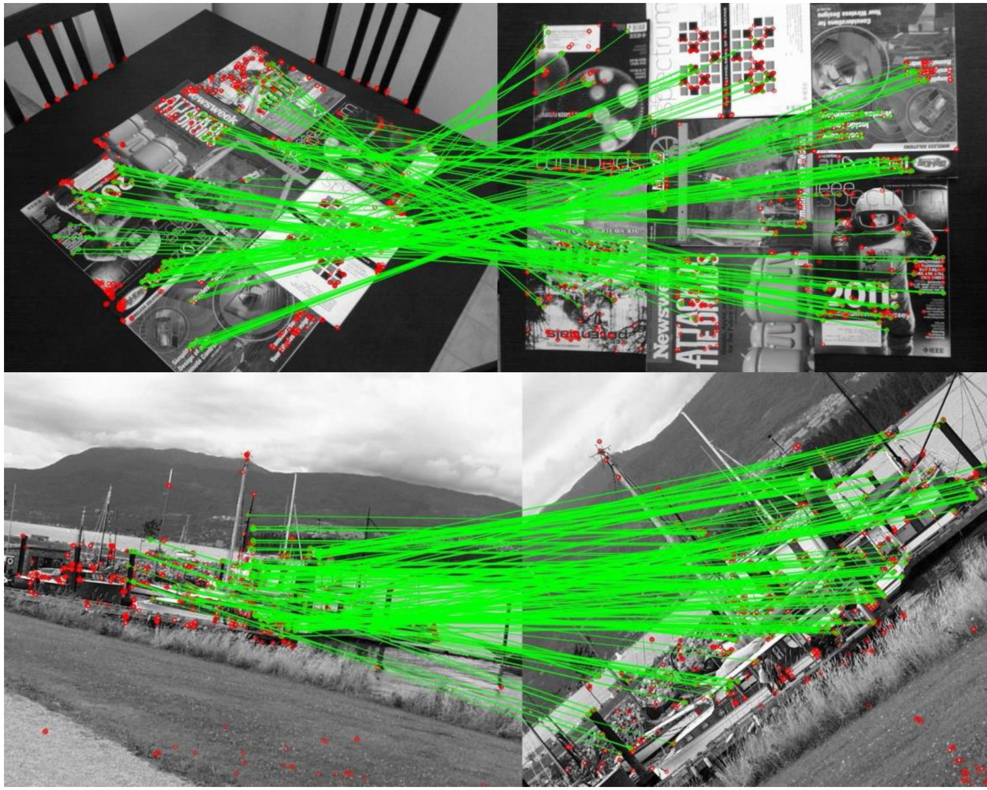


Pattern for intensity pair samples – generated randomly

Calonder, Lepetit, Strecha, Fua, BRIEF: Binary Robust Independent Elementary Features, ECCV'10]

ORB descriptor [Rublee et al., ICCV 2011]

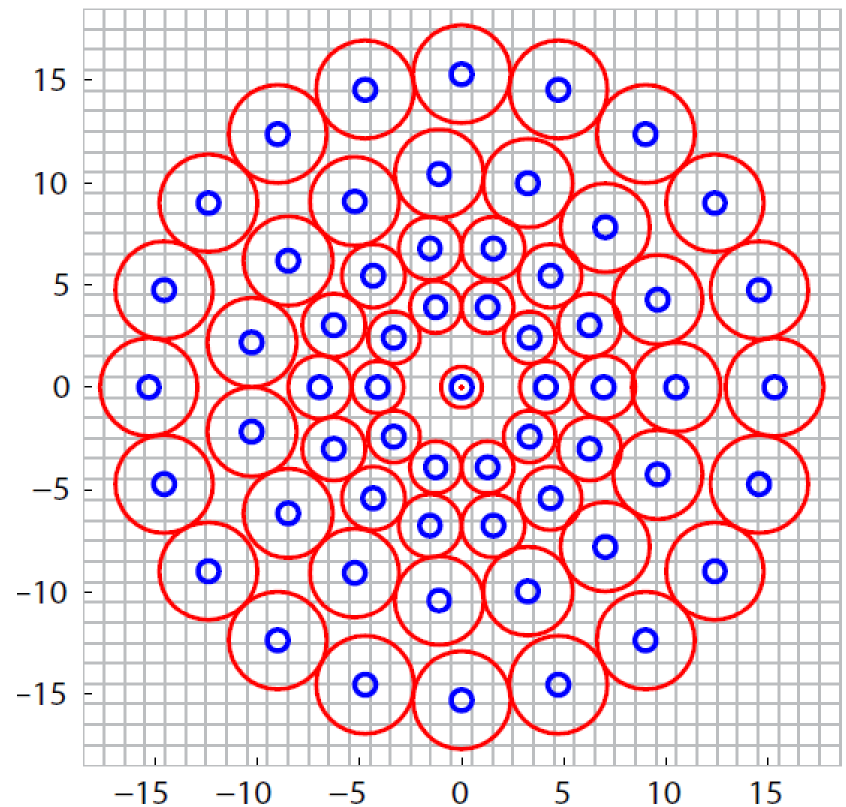
- ***O***riented FAST and ***R***otated ***B***RIEF
- Alterative to SIFT or SURF, designed for fast computation
- Keypoint detector based on **FAST**
- **BRIEF** descriptors are *steered* according to keypoint orientation (to provide rotation invariance)
- Good Binary features are learned by minimizing the correlation on a set of training patches.



BRISK descriptor [Leutenegger, Chli, Siegwart, ICCV 2011]

- **B**inary **R**obust **I**nvariant **S**calable **K**eypoints
- Detect corners in scale-space using FAST
- Rotation and scale invariant

- **Binary**, formed by pairwise intensity comparisons (like BRIEF)
- **Pattern** defines intensity comparisons in the keypoint neighborhood
- **Red circles**: size of the smoothing kernel applied
- **Blue circles**: smoothed pixel value used
- Compare short- and long-distance pairs for orientation assignment & descriptor formation
- Detection and descriptor speed: ~10 times faster than SURF
- Slower than BRIEF, but scale- and rotation- invariant



Summary of Features for VO and SLAM

Detector	Descriptor	Accuracy	Relocalization & Loop closing	Efficiency
Harris	Patch	++++	-	+++
Shi-Tomasi	Patch	++++	-	+++
SIFT	SIFT	++	++++	+
SURF	SURF	++	++++	++
FAST	BRIEF	++++	+++	++++
ORB	ORB	++++	+++	++++
FAST	BRISK	++++	+++	++++

ORB & BRISK:

- 128-to-256-bit binary descriptors
- Fast to extract and match (Hamming distance)
- Good for relocalization and Loop detection
- Multi-scale detection → same point appears on several scales

VO Flow Chart

VO computes the camera path incrementally (pose after pose)

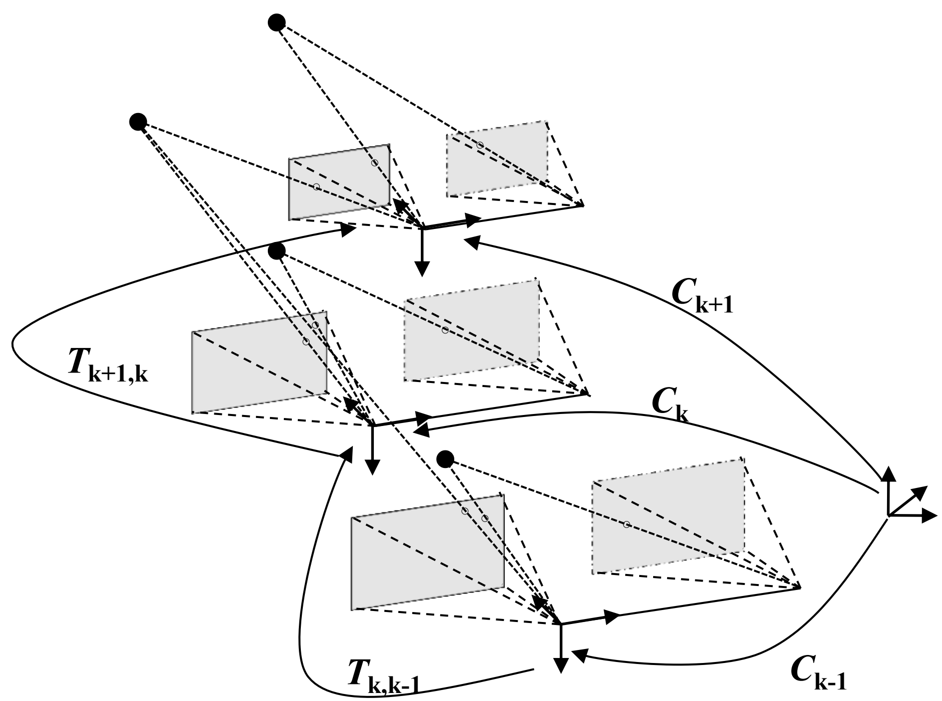
Image sequence

Feature detection

Feature matching (tracking)

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

Local optimization



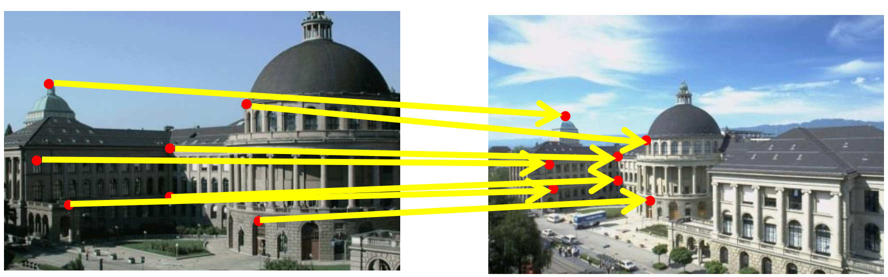
2D-to-2D

Motion from Image Feature Correspondences

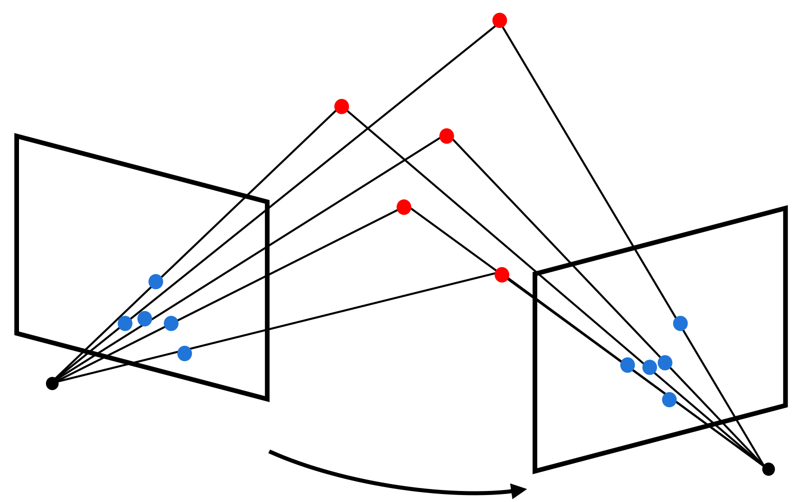
- Both feature points f_{k-1} and f_k are specified **in 2D**
- The minimal-case solution involves **5-point** correspondences
- The solution is found by minimizing the reprojection error:

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

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



I_{k-1} I_k



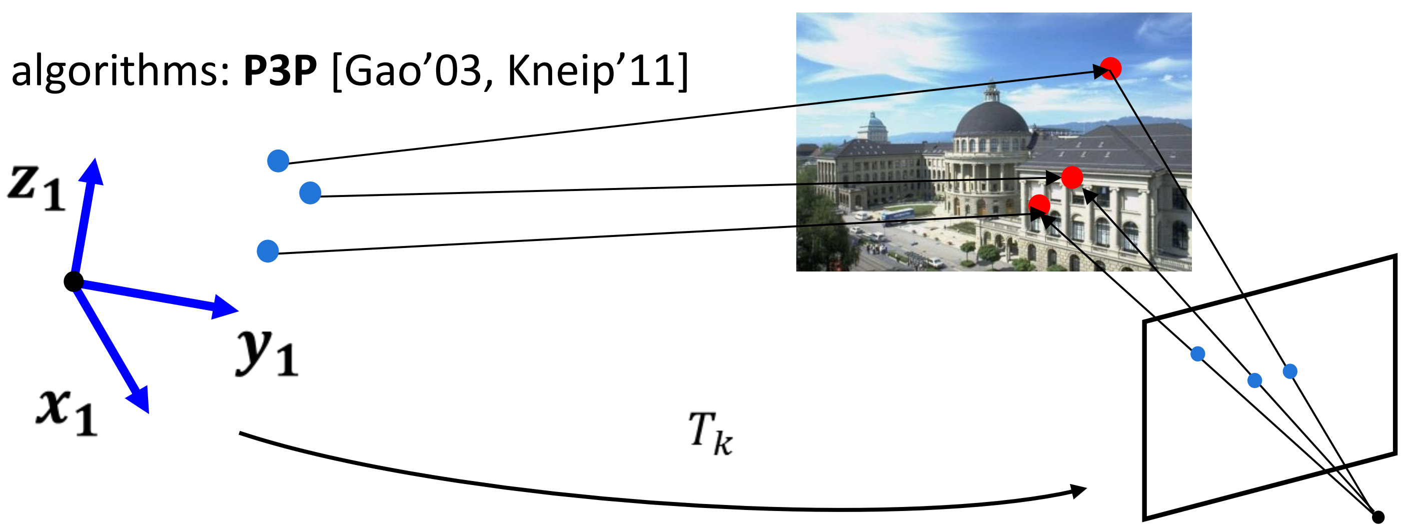
3D-to-2D

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:

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

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



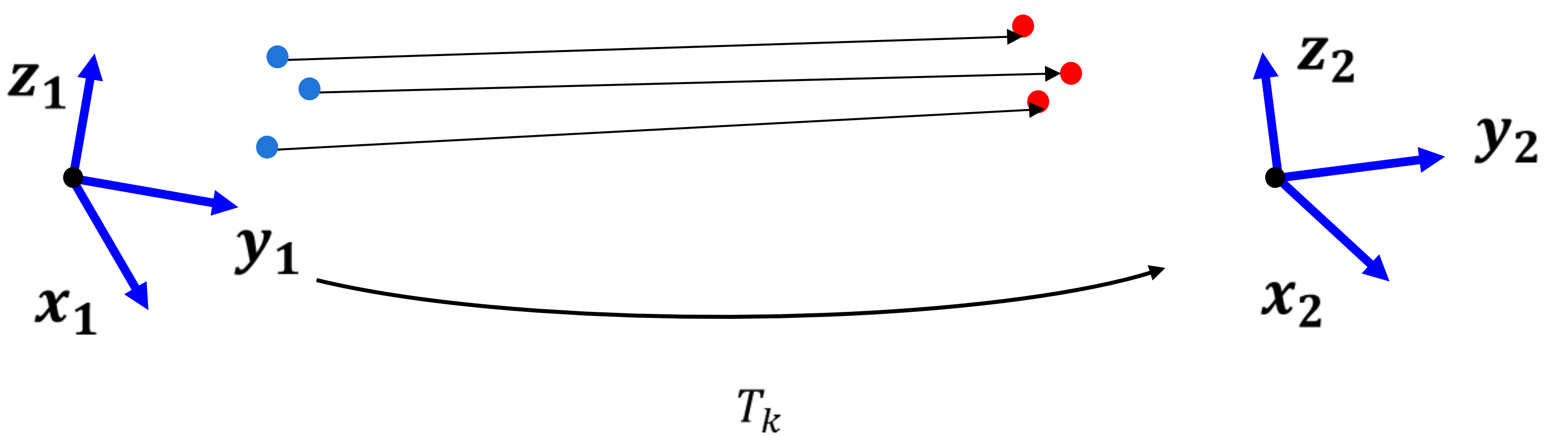
3D-to-3D

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:

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

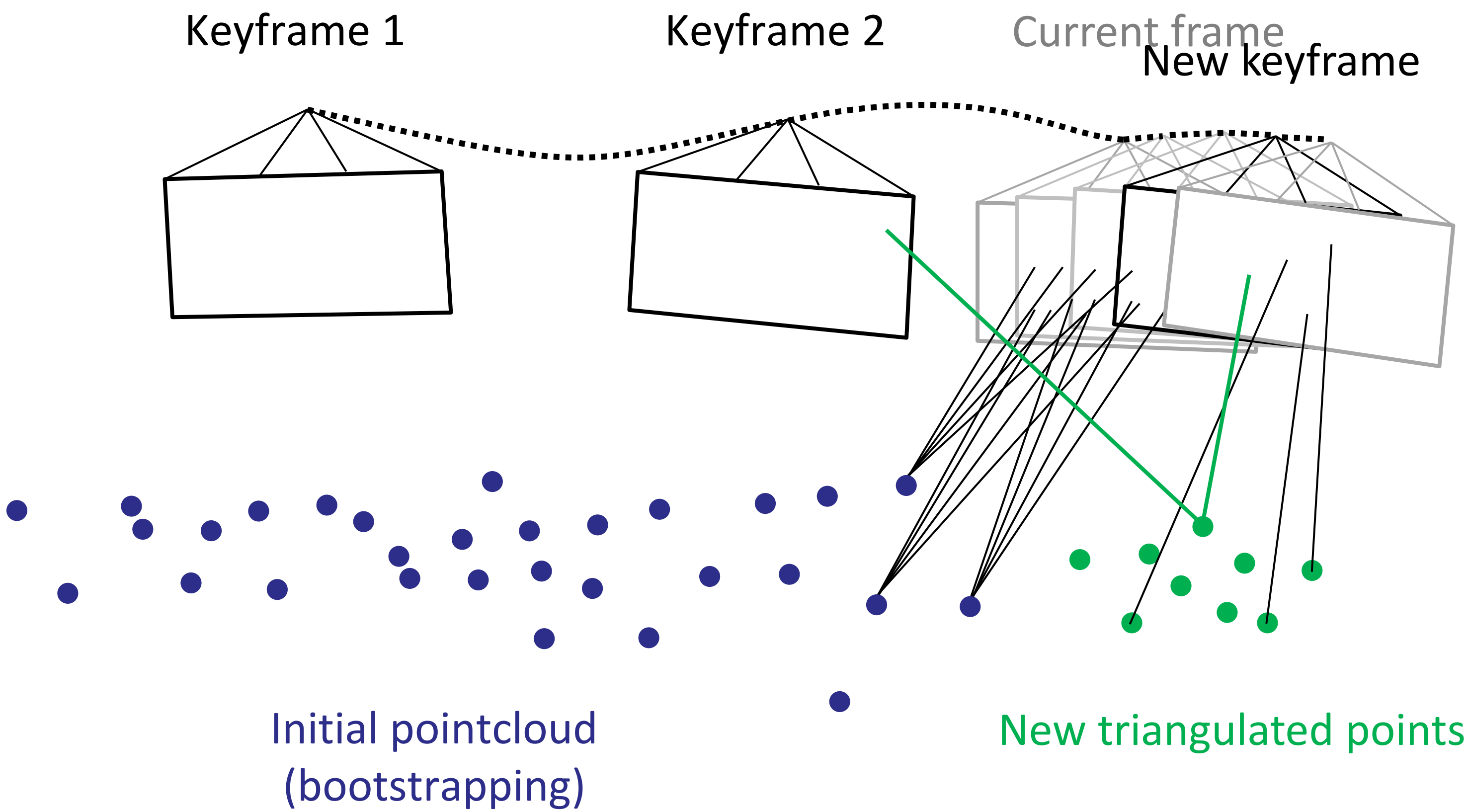
- Popular algorithm: [Arun'87] for global registration, ICP for local refinement or Bundle Adjustment (BA)



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

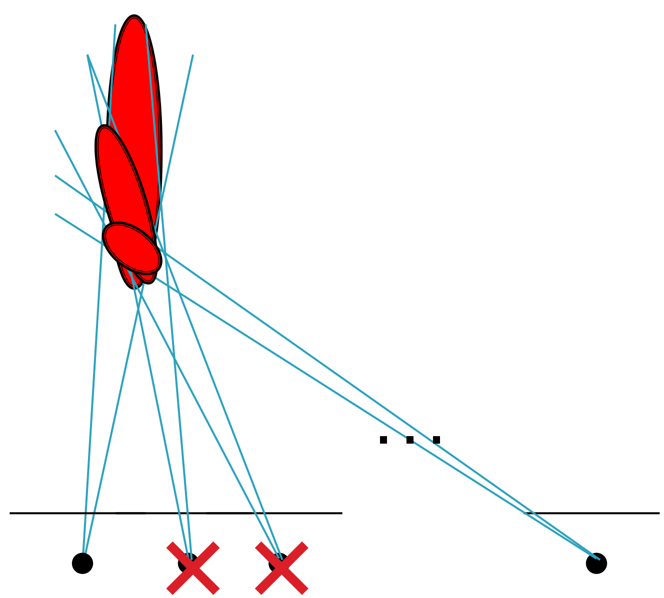
- When frames are taken at nearby positions compared to the scene distance, 3D points will exibit large uncertainty

Small baseline → large depth uncertainty

Large baseline → small depth uncertainty

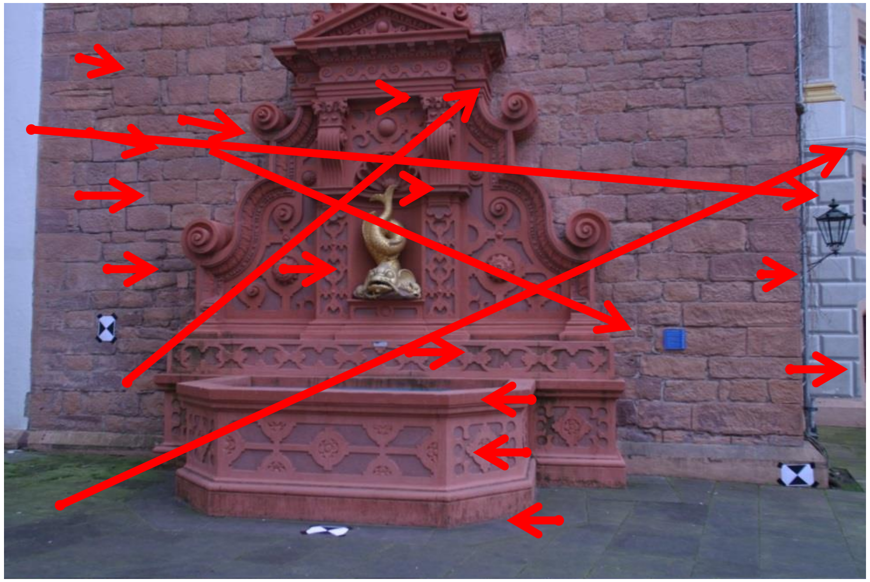
Keyframe Selection

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

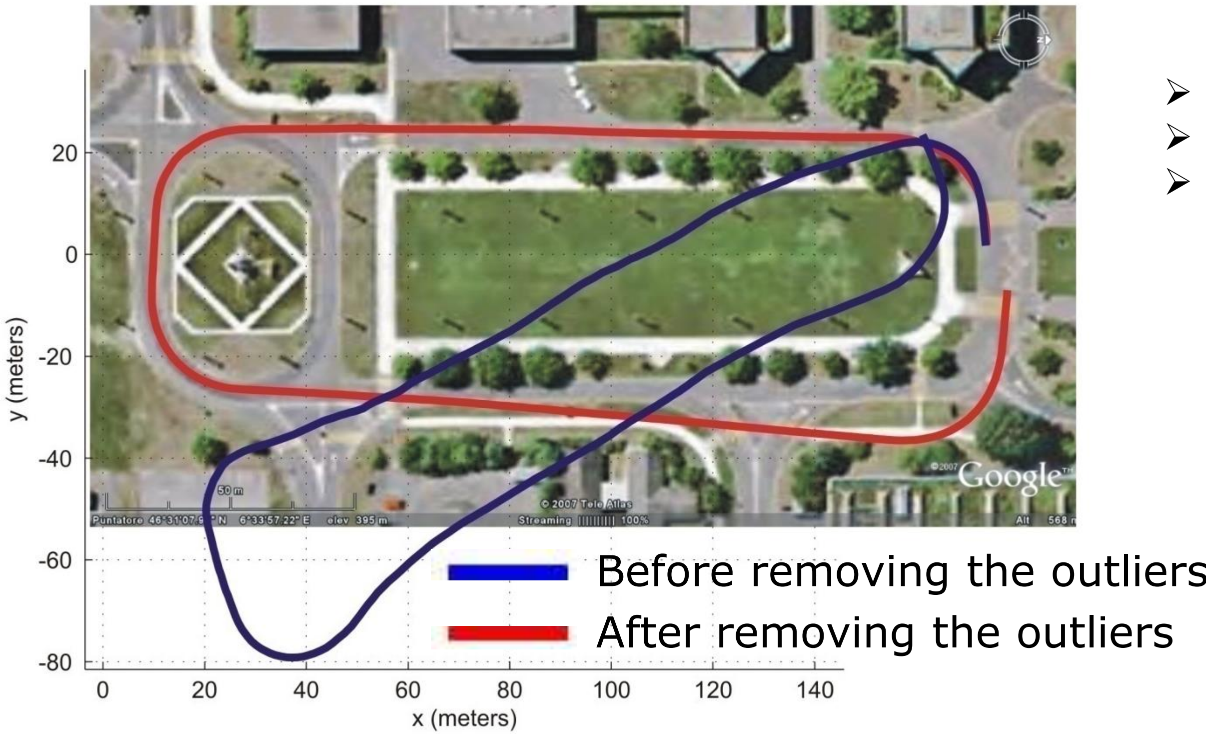


Robust Estimation

- 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

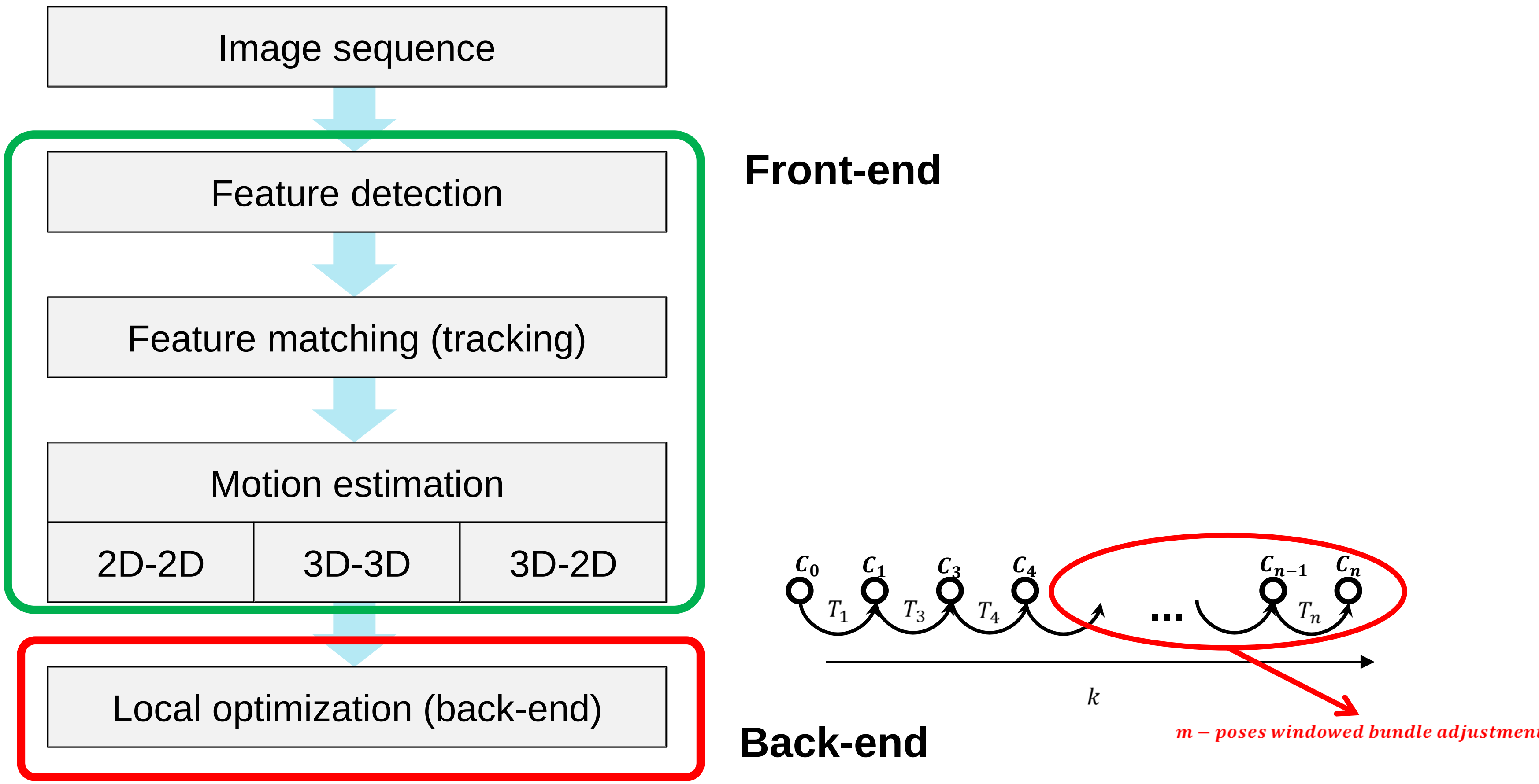


- ➢ Error at the loop closure: 6.5 m
- ➢ Error in orientation: 5 deg
- ➢ Trajectory length: 400 m

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

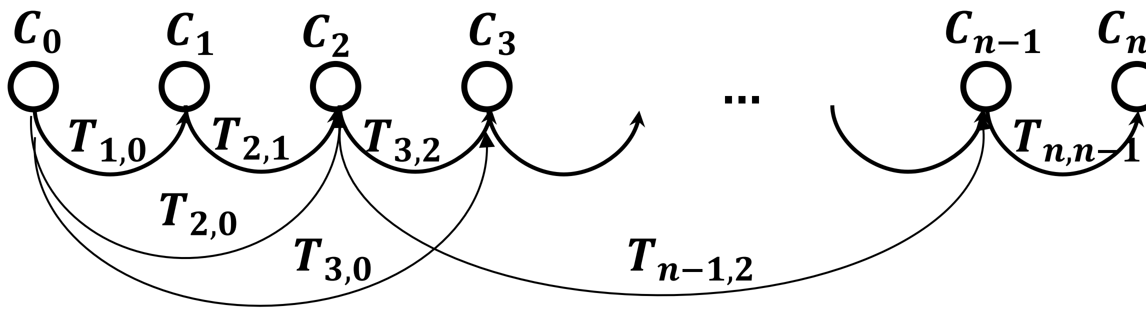
VO Flow Chart

VO computes the camera path incrementally (pose after pose)



Pose-Graph Optimization

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

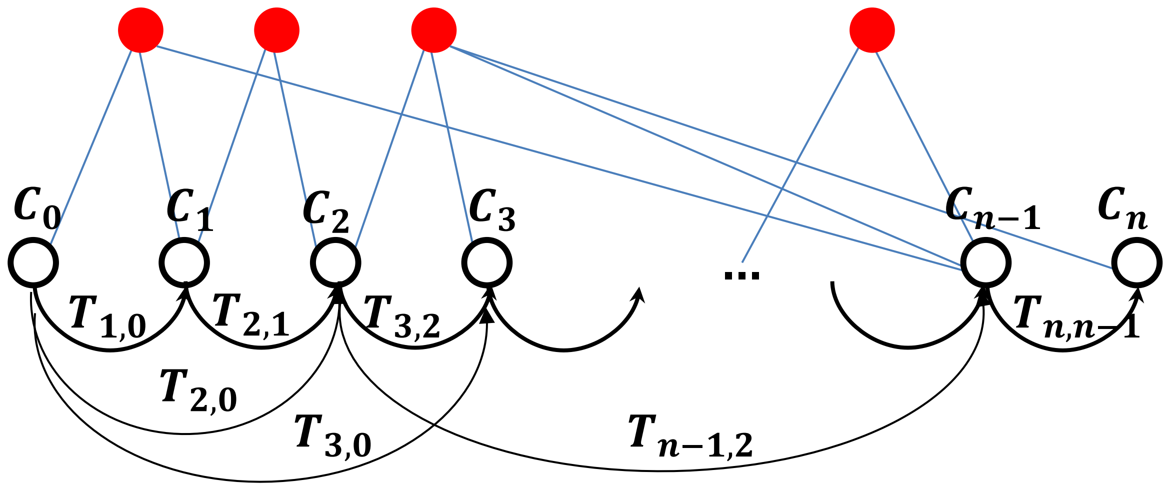


- Transformations can be computed also between non-adjacent frames T_{ij} (e.g., when features from previous keyframes are still observed). They can be used as additional constraints to improve cameras poses by minimizing the following:

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

- 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

$$X^i, C_k = argmin_{X^i, C_k,} \sum_{i,k} \rho_H\left(p_k{}^i - \pi\left(X^i, C_k\right)\right)$$

- $\rho_H()$ is a robust cost function (e.g., Huber cost) to downweight wrong matches
- 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
- Very costly: example: 1k images and 100k points, 1s per LM iteration. 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**: $O\left((qM + lN)^3\right)$ 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 3D landmarks fixed, e.g., (**motion-only BA**)

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

Visual Place Recognition

- **Goal**: find the most similar images of a **query** image in a database of $\boldsymbol{N}$ **images**
- **Complexity:** $\frac{N^2 \cdot M^2}{2}$ feature comparisons (*worst-case* scenario)
  - Each image must be compared with all other images!
  - N is the number of all images collected by a robot
    - Example: 1 image per meter of travelled distance over a $100m^2$ house with one robot and 100 feature per image → $\mathrm{M} = 100$, $N = 100 \rightarrow N^2M^2$/2= $\sim 50\ Million$ feature comparisons!

Solution: Use an inverted file index!
Complexity reduces to $\boldsymbol{N \cdot M}$

[“Video Google”, Sivic & Zisserman, ICCV’03]
[“Scalable Recognition with a Vocabulary Tree”, Nister & Stewenius, CVPR’06]
See also FABMAP and Galvez-Lopez’12’s (DBoW2)]

Indexing local features: inverted file text

- For text documents, an efficient way to find all *pages* on which a *word* occurs is to use an index

- We want to find all *images* in which a *feature* occurs

- To use this idea, we'll need to map our features to "visual words"

Index

"Along I-75," From Detroit to Florida; ***inside back cover***
"Drive I-95," From Boston to Florida; ***inside back cover***
1929 Spanish Trail Roadway; 101-102,104
511 Traffic Information; 83
A1A (Barrier Isl) - I-95 Access; 86
AAA (and CAA); 83
AAA National Office; 88
Abbreviations,
Colored 25 mile Maps; cover
Exit Services; 196
Travelogue; 85
Africa; 177
Agricultural Inspection Stns; 126
Ah-Tah-Thi-Ki Museum; 160
Air Conditioning, First; 112
Alabama; 124
Alachua; 132
County; 131
Alafia River; 143
Alapaha, Name; 126
Alfred B Maclay Gardens; 106
Alligator Alley; 154-155
Alligator Farm, St Augustine; 169
Alligator Hole (definition); 157
Alligator, Buddy; 155
Alligators; 100,135,138,147,156
Anastasia Island; 170
Anhaica; 108-109,146
Apalachicola River; 112
Appleton Mus of Art; 136
Aquifer; 102
Arabian Nights; 94
Art Museum, Ringling; 147
Aruba Beach Cafe; 183
Aucilla River Project; 106
Babcock-Web WMA; 151
Bahia Mar Marina; 184
Baker County; 99
Barefoot Mailmen; 182
Barge Canal; 137
Bee Line Expy; 80
Belz Outlet Mall; 89
Bernard Castro; 136
Big "I"; 165
Big Cypress; 155,158
Big Foot Monster; 105
Billie Swamp Safari; 160
Blackwater River SP; 117
Blue Angels
A4-C Skyhawk; 117
Atrium; 121
Blue Springs SP; 87
Blue Star Memorial Highway; 125
Boca Ciega; 189
Boca Grande; 150
Butterfly Center, McGuire; 134
CAA (see AAA)
CCC, The; 111,113,115,135,142
Ca d'Zan; 147
Caloosahatchee River; 152
Name; 150
Canaveral Natnl Seashore; 173
Cannon Creek Airpark; 130
Canopy Road; 106,169
Cape Canaveral; 174
Castillo San Marcos; 169
Cave Diving; 131
Cayo Costa, Name; 150
Celebration; 93
Charlotte County; 149
Charlotte Harbor; 150
Chautauqua; 116
Chipley; 114
Name; 115
Choctawatchee, Name; 115
Circus Museum, Ringling; 147
Citrus; 88,97,130,136,140,180
CityPlace, W Palm Beach; 180
City Maps,
Ft Lauderdale Expwys; 194-195
Jacksonville; 163
Kissimmee Expwys; 192-193
Miami Expressways; 194-195
Orlando Expressways; 192-193
Pensacola; 26
Tallahassee; 191
Tampa-St. Petersburg; 63
St. Augsutine; 191
Civil War; 100,108,127,138,141
Clearwater Marine Aquarium; 187
Collier County; 154
Collier, Barron; 152
Colonial Spanish Quarters; 168
Columbia County; 101,128
Coquina Building Material; 165
Corkscrew Swamp, Name; 154
Cowboys; 95
Crab Trap II; 144
Cracker, Florida; 88,95,132
Crosstown Expy; 11,35,98,143
Cuban Bread; 184
Dade Battlefield; 140
Dade, Maj. Francis; 139-140,161
Dania Beach Hurricane; 184
Daniel Boone, Florida Walk; 117
Daytona Beach; 172-173
De Land; 87
De Soto, Hernando,
Anhaica; 108-109,146
County; 149
Explorer; 146
Landing; 146
Napitaca; 103
Driving Lanes; 85
Duval County; 163
Eau Gallie; 175
Edison, Thomas; 152
Eglin AFB; 116-118
Eight Reale; 176
Ellenton; 144-145
Emanuel Point Wreck; 120
Emergency Callboxes; 83
Epiphytes; 142,148,157,159
Escambia Bay; 119
Bridge (I-10); 119
County; 120
Estero; 153
Everglade,90,95,139-140,154-160
Draining of; 156,181
Wildlife MA; 160
Wonder Gardens; 154
Falling Waters SP; 115
Fantasy of Flight; 95
Fayer Dykes SP; 171
Fires, Forest; 166
Fires, Prescribed ; 148
Fisherman's Village; 151
Flagler County; 171
Flagler, Henry; 97,165,167,171
Florida Aquarium; 186
Florida,
12,000 years ago; 187
Cavern SP; 114
Map of all Expressways; 2-3
Mus of Natural History; 134
National Cemetery ; 141
Part of Africa; 177
Platform; 187
Sheriff's Boys Camp; 126
Sports Hall of Fame; 130
Sun 'n Fun Museum; 97
Supreme Court; 107
Florida's Turnpike (FTP), 178,189
25 mile Strip Maps; 66
Administration; 189
Coin System; 190
Exit Services; 189
HEFT; 76,161,190
History; 189
Names; 189
Service Plazas; 190
Spur SR91; 76
Ticket System; 190
Toll Plazas; 190
Ford, Henry; 152
Fort Barrancas; 122
Buried Alive; 123
Fort Caroline; 164
Fort Clinch SP; 161
Fort De Soto & Egmont Key; 188
Fort Lauderdale; 161,182-184

Building the Visual Vocabulary

Limitations of VO-SLAM systems

- Limitations
  - Monocular (i.e., absolute scale is unknown)
  - Requires a reasonably **illuminated** area
  - **Motion blur**
  - **Needs texture**: will fail with large plain walls
  - **Map is too sparse** for interaction with the environment

- Extensions
  - **IMU** for robustness and absolute scale estimation
  - **Stereo**: real scale and more robust to quick motions
  - **Semi-dense or dense mapping** for environment interaction
  - **Event-based cameras** for high-speed motions and HDR environments
  - **Learning** for improved reliability

Visual-Inertial Fusion

Absolute Scale Determination

- The absolute pose x is known up to a scale s, thus

$$x = s\tilde{x}$$

- IMU provides accelerations, thus

$$v = v_0 + \int a(t)dt$$

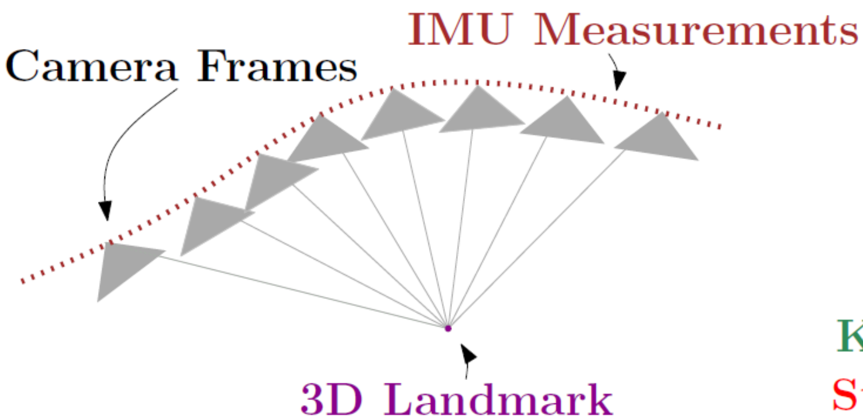
- By derivating the first one and equating them

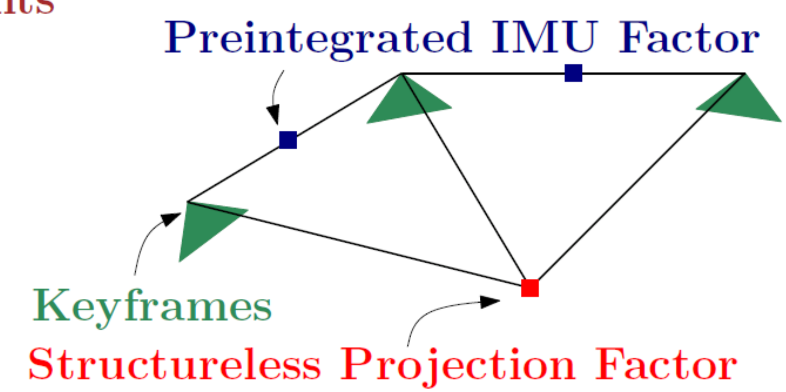
$$s\dot{\tilde{x}} = v_0 + \int a(t)dt$$

- As shown in [Martinelli, TRO'12], for 6DOF, both s and v_0 can be determined in closed form from a **single feature observation and 3 views**
- This is used to initialize the asbolute scale [Kaiser, ICRA'16]
- The scale can then be tracked with
  - EKF [Mourikis & Roumeliotis, IJRR'10], [Weiss, JFR'13]
  - Non-linear optimization methods [Leutenegger, RSS'13] [Forster, RSS'15]

Visual-Inertial Fusion [RSS'15]

- Fusion solved as a *non-linear optimization problem*
- Increased accuracy over filtering methods





$$\sum_{(i,j)\in\mathcal{K}_k} \|\mathbf{r}_{\mathcal{I}_{ij}}\|^2_{\Sigma_{ij}} + \sum_{i\in\mathcal{K}_k}\sum_{l\in\mathcal{C}_i} \|\mathbf{r}_{\mathcal{C}_{il}}\|^2_{\Sigma_{\mathcal{C}}}$$

IMU residuals *Reprojection residuals*

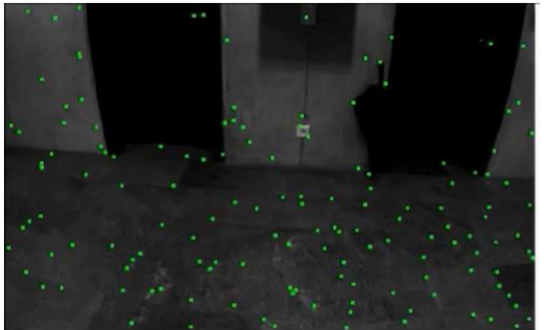
Forster, Carlone, Dellaert, Scaramuzza, IMU Preintegration on Manifold for efficient Visual-Inertial Maximum-a-Posteriori Estimation, *Robotics Science and Systens*'15, **Best Paper Award Finalist**

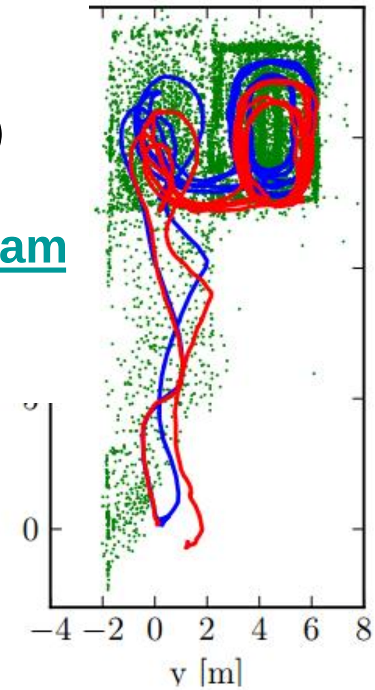
Comparison with Previous Works

Open Source

SVO + GTSAM (Forster et al. RSS'15) (optimization based, pre-integrated IMU): [**https://bitbucket.org/gtborg/gtsam**](https://bitbucket.org/gtborg/gtsam)

Instructions here: [**http://arxiv.org/pdf/1512.02363**](http://arxiv.org/pdf/1512.02363)

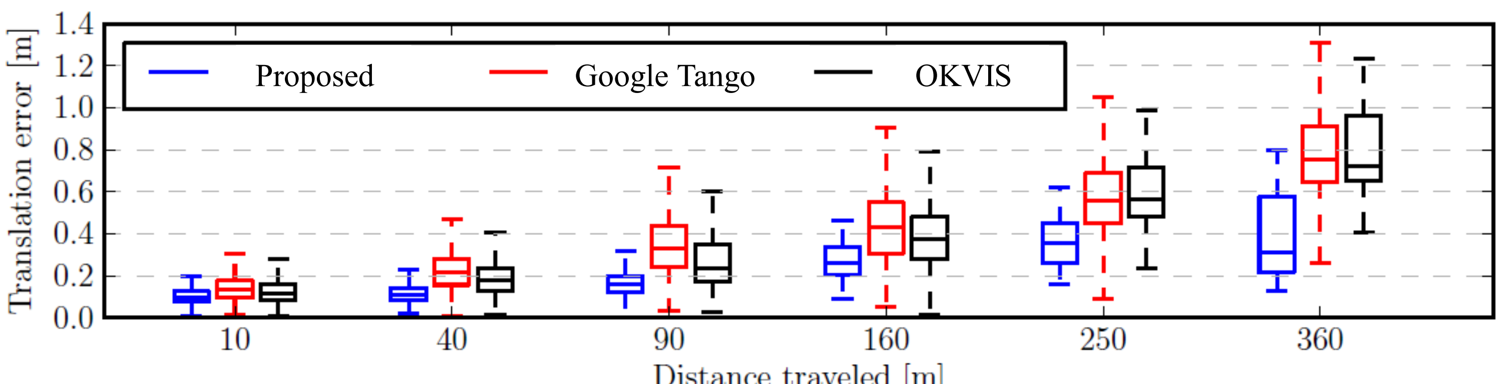




YouTube: https://youtu.be/CsJkci5lfco

5x

Accuracy: 0.1% of the travel distance



Forster, Carlone, Dellaert, Scaramuzza, IMU Preintegration on Manifold for efficient Visual-Inertial Maximum-a-Posteriori Estimation, *Robotics Science and Systens*'15, **Best Paper Award Finalist**

Open-source
VO & VSLAM algorithms

Intro: visual odometry algorithms

- Popular visual odometry and SLAM algorithms
  - ORB-SLAM (University of Zaragoza, 2015)
    - ORB-SLAM2 (2016) supports stereo and RGBD camera
  - LSD-SLAM (Technical University of Munich, 2014)
  - DSO (Technical University of Munich, 2016)
  - SVO (University of Zurich, 2014/2016)
    - SVO 2.0 (2016) supports wide angle, stereo and multiple cameras

ORB-SLAM

Large-scale Feature-based SLAM

[Mur-Artal, Montiel, Tardos, TRO’15]

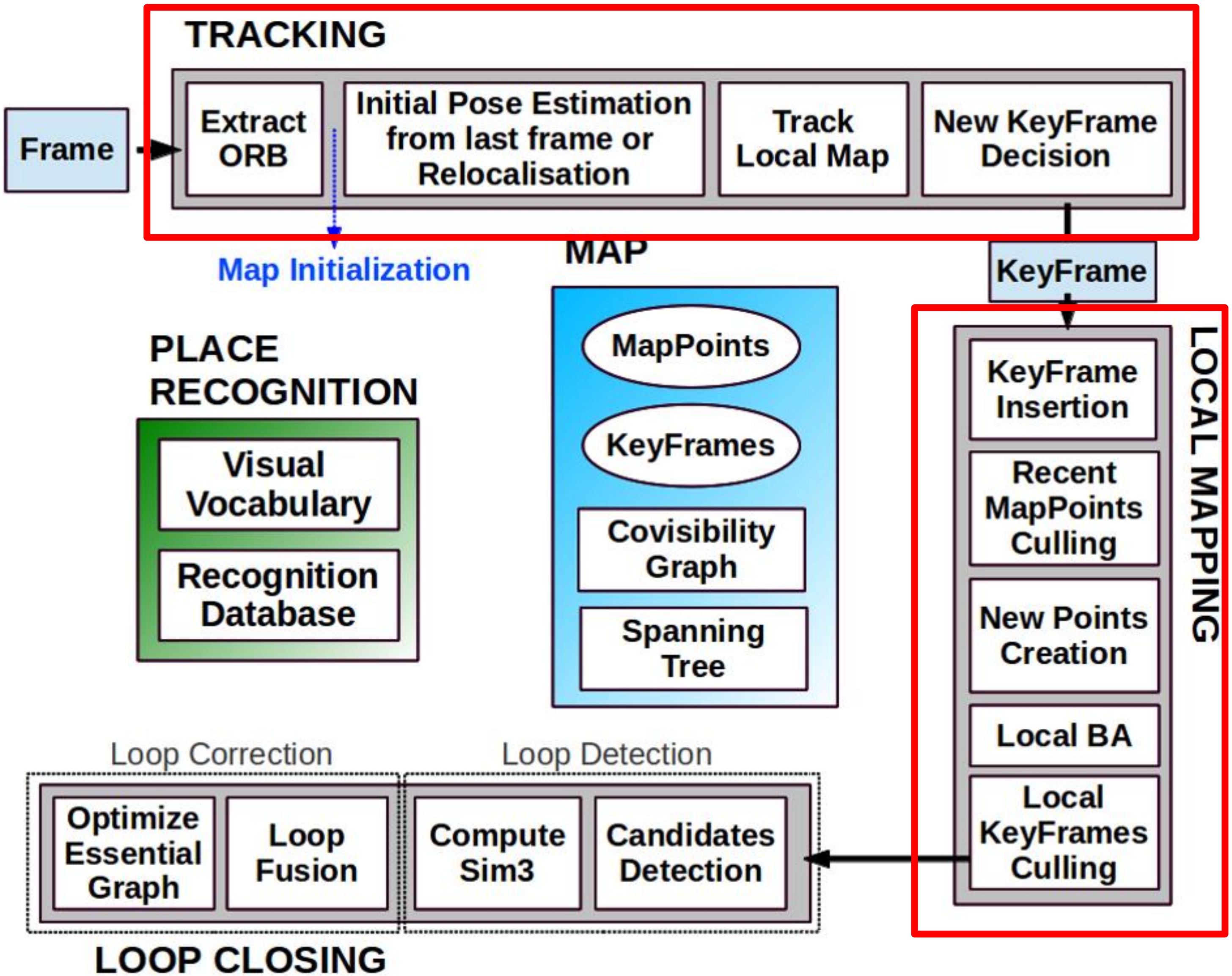
ORB-SLAM: overview

- It combines all together:
  - Tracking
  - Mapping
  - Loop closing
  - Relocalization (DBoW)
  - Final optimization

- **ORB**: FAST corner + Oriented Rotated Brief descriptor
  - Binary descriptor
  - Very fast to compute and compare

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

ORB-SLAM: overview

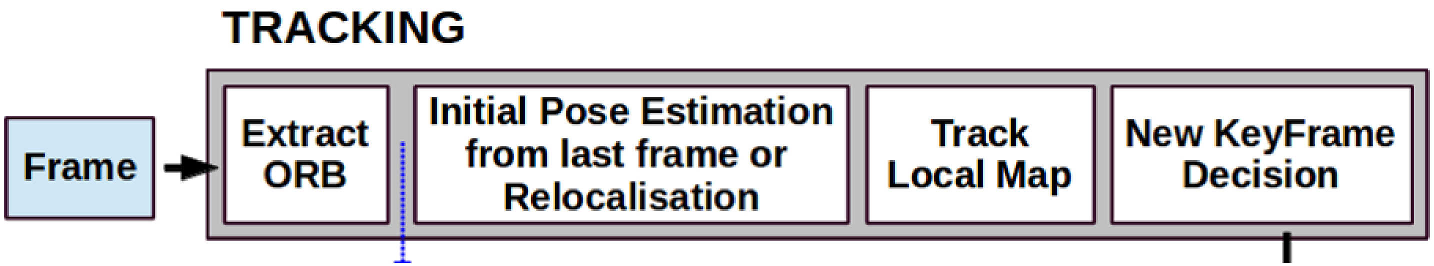


ORB-SLAM: ORB feature

- ORB: Oriented FAST and Rotated Brief
  - 256-bit binary descriptor
  - Fast to extract and match (Hamming distance)
  - Good for tracking, relocation and Loop detection
  - Multi-scale detection: same point appears on several scales

Detector	Descriptor	Rotation Invariant	Automatic Scale	Accuracy	Relocation & Loops	Efficiency
Harris	Patch	No	No	++++	-	++++
Shi-Tomasi	Patch	No	No	++++	-	++++
SIFT	SIFT	Yes	Yes	++	++++	+
SURF	SURF	Yes	Yes	++	++++	++
FAST	BRIEF	No	No	+++	+++	++++
ORB	ORB	Yes	No	+++	+++	++++

ORB-SLAM: tracking

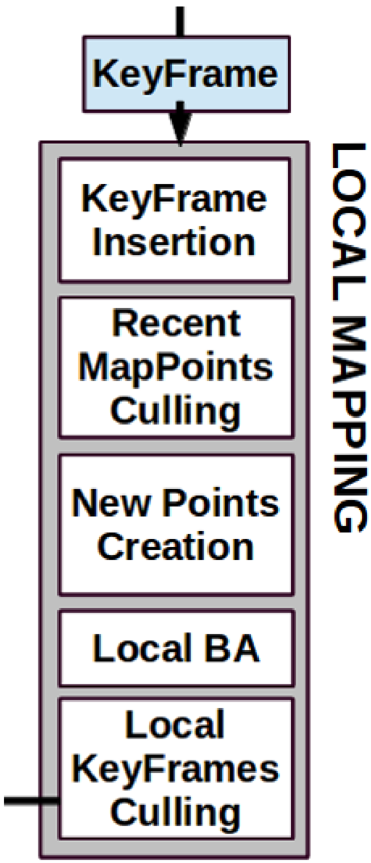


- For every new frame:
  - First track w.r.t. last frame

    Find matches from last frame in the new frame -> PnP
  - Then track w.r.t. local map

    Find matches from local keyframes in the new frame -> PnP

ORB-SLAM: mapping



- Map representation
  - Keyframe poses
  - Points
    - 3D positions
    - Descriptor
    - Observations in frames
- Functions of the mapping part
  - Triangulate new points
  - Remove redundant keyframes/points
  - Optimize poses and points

Q: why do we need keyframes instead of just using points?

ORB-SLAM: video

LSD-SLAM

Large-scale Semi-Dense SLAM

[Engel, Schoeps, Cremers, ECCV'14]

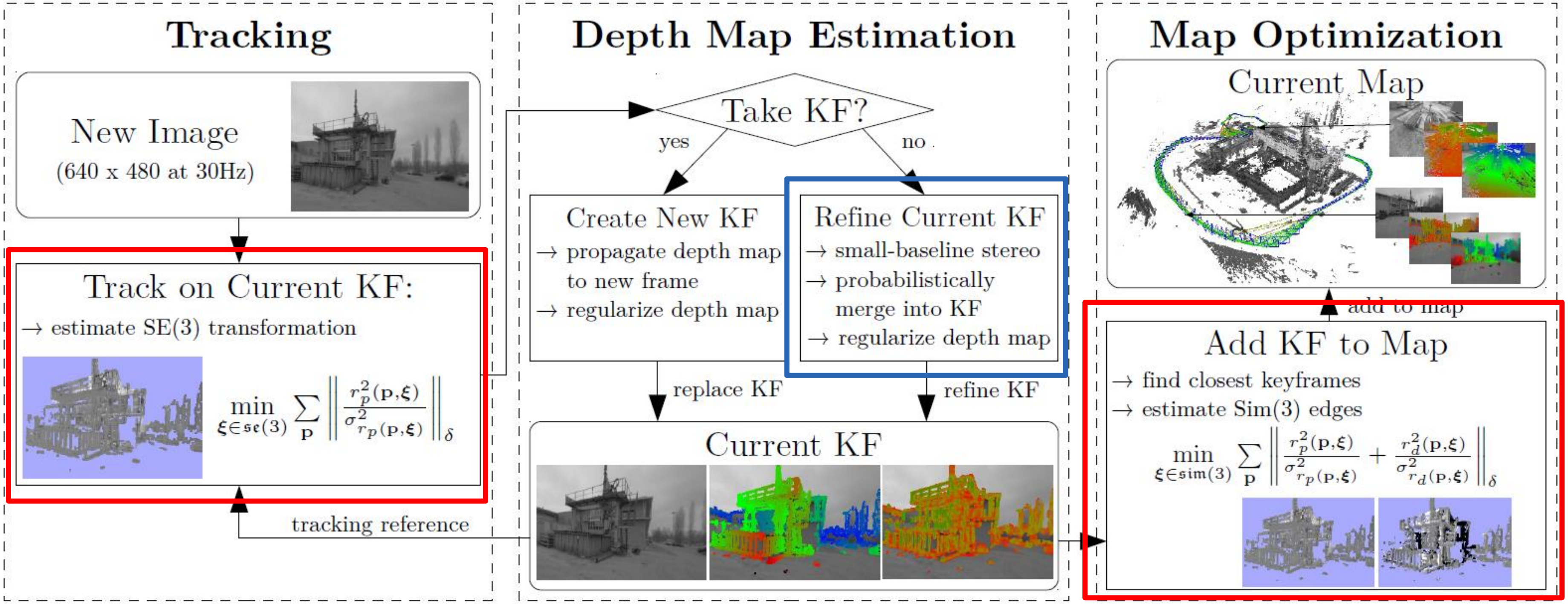
LSD-SLAM: Overview

- **Direct** (photometric error) **+ Semi-Dense** formulation
  - 3D geometry represented as semi-dense depth maps.
  - Optimizes a photometric error
  - Separateley optimizes poses (direct image alignment) & geometry (pixel-wise filtering)

- Includes:
  - Loop closing
  - Relocalization
  - Final optimization

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

LSD-SLAM: overview



- Direct image alignment
- Depth refinement and regularization

Instead of using features, LSD-SLAM uses **pixels with large gradients**.

LSD-SLAM: Direct Image Alignment

- New frame w.r.t. last keyframe

$$E_p(\boldsymbol{\xi}_{ji}) = \sum_{\mathbf{p} \in \Omega_{D_i}} \left\| \boxed{\frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}}} \right\|_\delta$$

- Finding pose that Minimizes photometric error r_p over all selected pixels
- Weighted by the photometric covariance

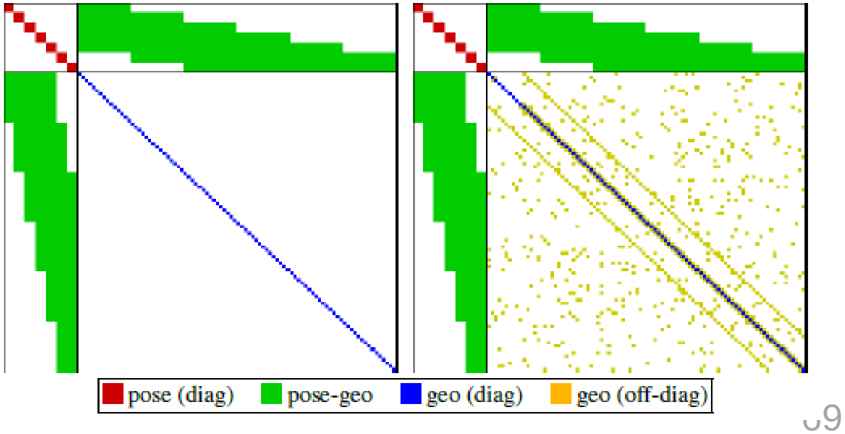
- Keyframe w.r.t. global map

$$E(\boldsymbol{\xi}_{ji}) := \sum_{\mathbf{p} \in \Omega_{D_i}} \left\| \boxed{\frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}}} + \boxed{\frac{r_d^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma^2_{r_d(\mathbf{p}, \boldsymbol{\xi}_{ji})}}} \right\|_\delta$$

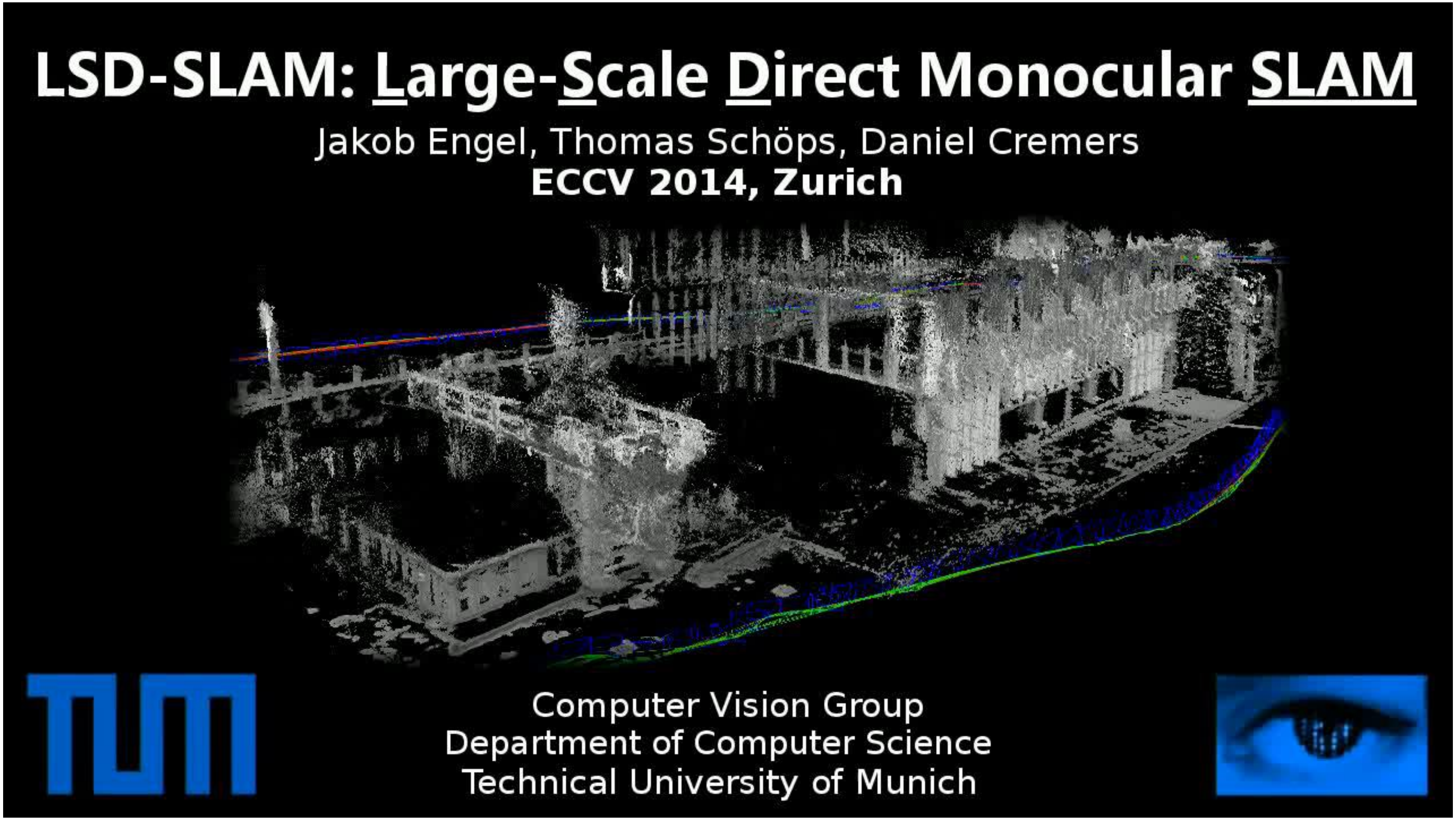
- Also minimizing geometric error: distance between the points in the current keyframe and the points in the global map.

LSD-SLAM: Depth Refinement/Regularization

- Depth estimation: per pixel stereo:
  - Using the estimated pose from image alignment, we can perform stereo matching for each pixel.
  - Using the stereo matching result to refine the depth.
- Regularization
  - Average using adjacent depth
  - Remove outliers and spurious estimations: visually appealing



LSD-SLAM: video



DSO

Direct Sparse Odometry

[Engel, Koltun, Cremers, Arxiv'16]

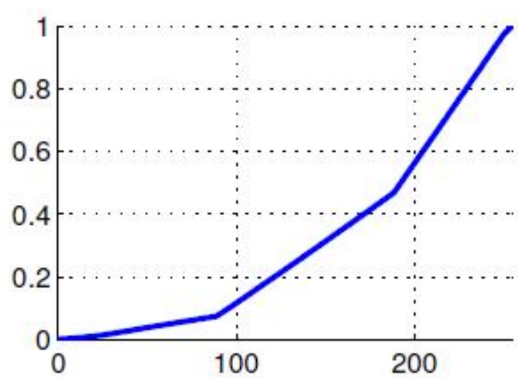
DSO: Tracking frontend

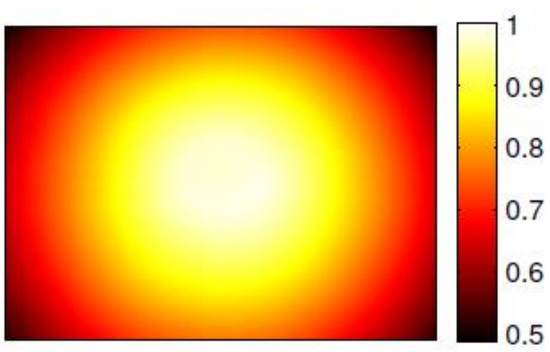
- Direct Image Alignment

$$E_{\text{photo}} := \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \sum_{j\in\text{obs}(\mathbf{p})} E_{\mathbf{p}j}$$

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

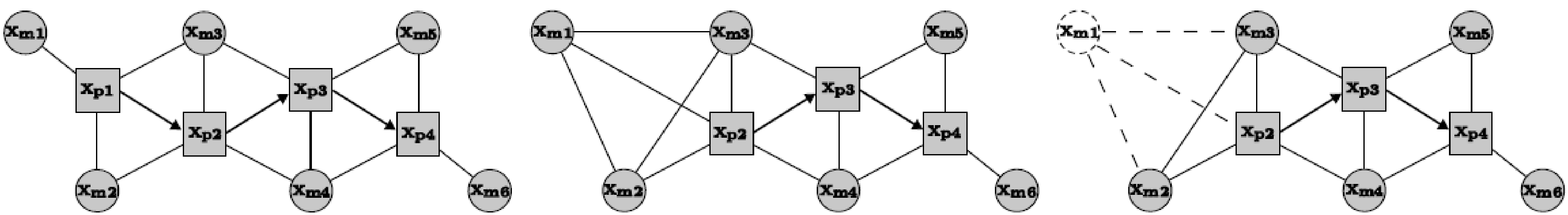
- Using points of large gradients
- Incorporate photometric correction: robust to exposure time change
  - Using exposure time t_i t_j to compensate exposure time change
  - Using affine transformation if no exposure time is known





DSO: Optimization backend

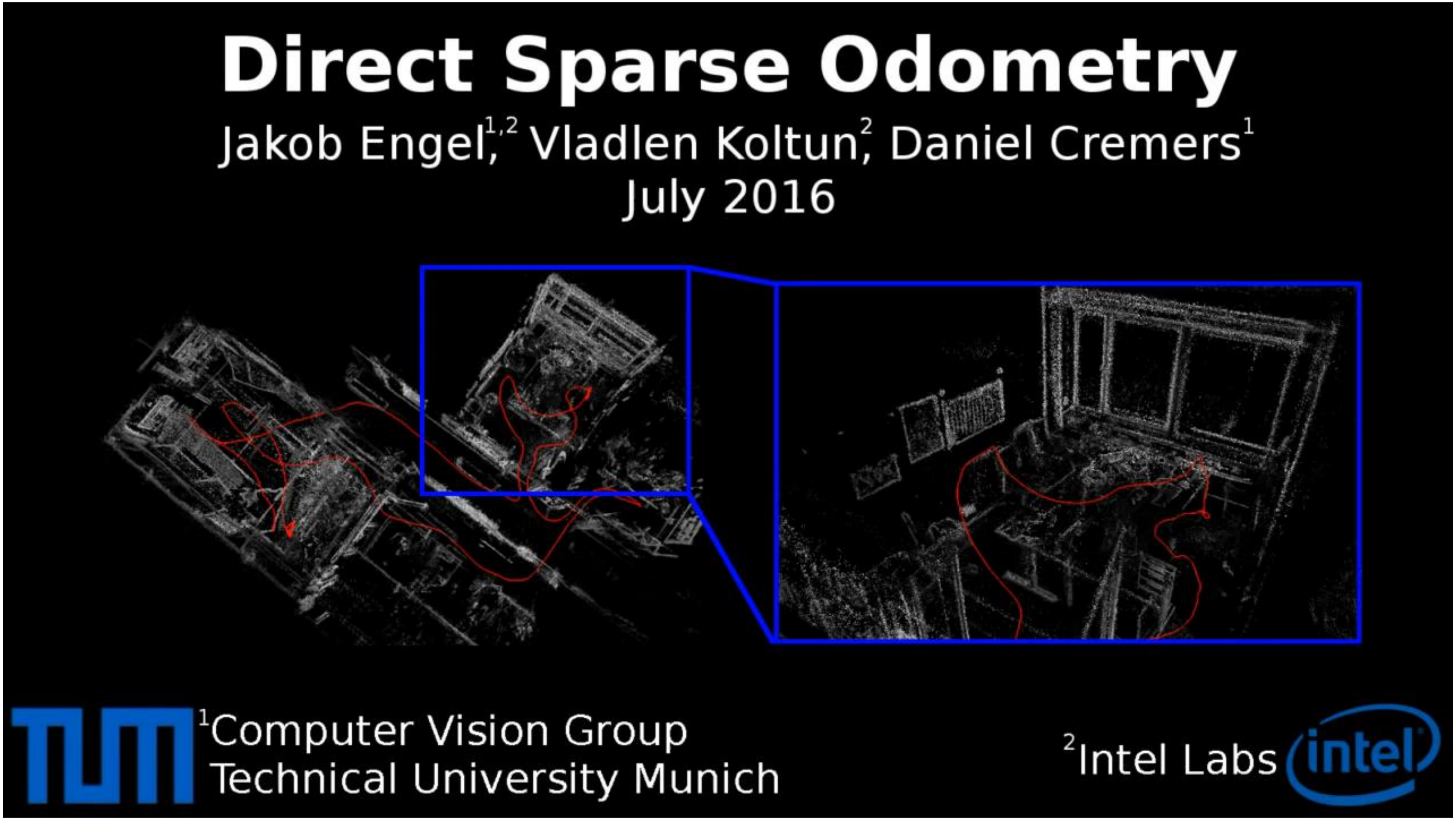
- Sliding window estimator
  - Not full bundle adjustment
  - Only keep a fixed length window (e.g., 3 keyframes) of past frames
  - Instead of simply dropping the states out of the window, marginalizing the states:



Advantage:

- Help improve accuracy
- Still able to operate in real-time

DSO: Video

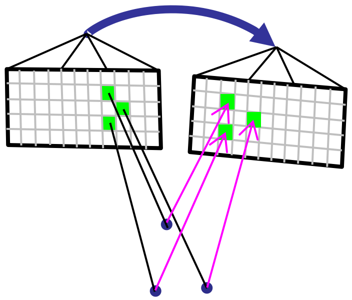

Direct Sparse Odometry
Jakob Engel[1,2] Vladlen Koltun[2] Daniel Cremers[1]
July 2016
[1]Computer Vision Group
Technical University Munich
[2]Intel Labs

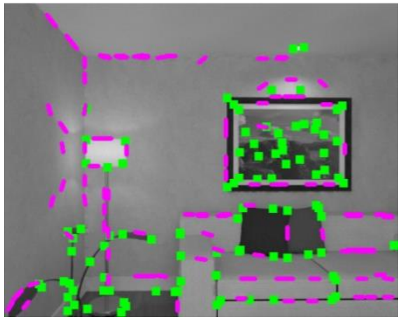
SVO

Fast, Semi-Direct Visual Odometry

[Forster, Pizzoli, Scaramuzza, ICRA'14, TRO'16]

SVO: overview





Direct (minimizes photometric error)

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

Feature-based (minimizes photometric error)

- Frame-to-Keyframe pose refinement

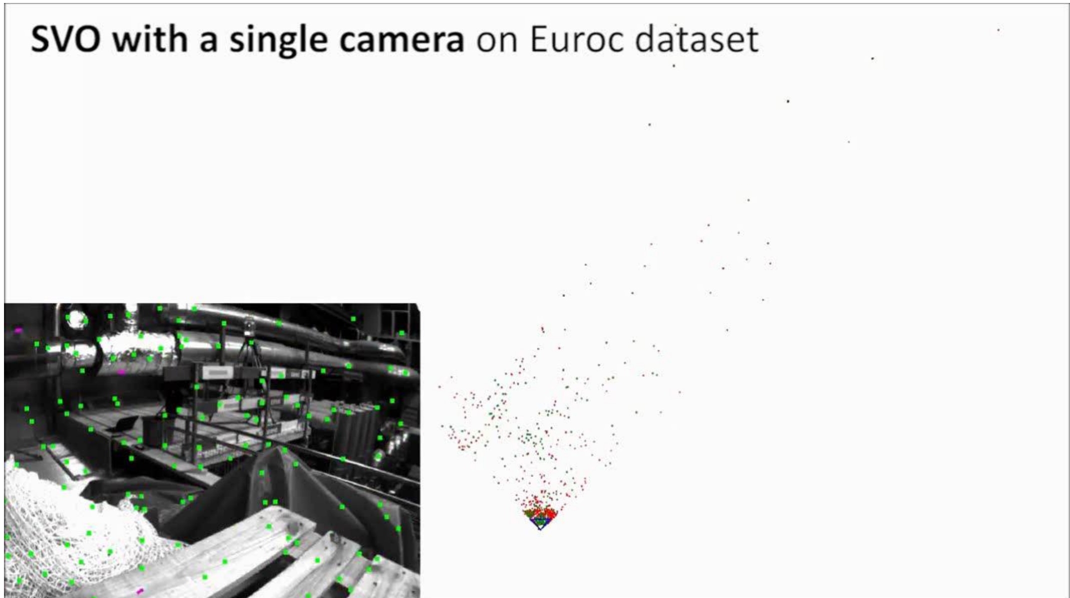
Edgelet **Corner**

Mapping

- **Probabilistic depth** estimation

Extensions

- Omni-cameras
- Multi-camera systems
- IMU pre-integration
- Dense → REMODE



SVO: Semi-Direct Visual Odometry [ICRA'14]

Direct

- Frame-to-frame motion estimation

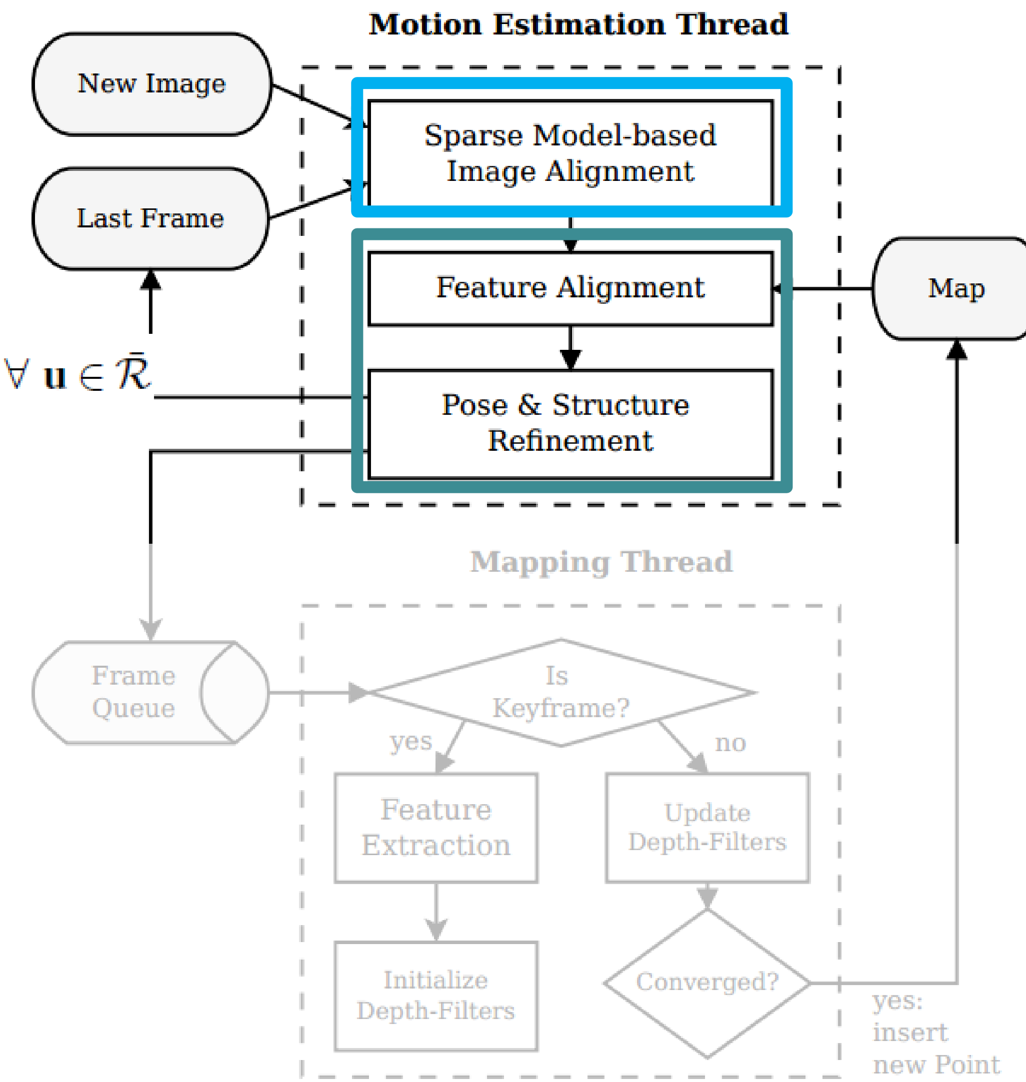
$$\mathbf{T}_{k,k-1} = \arg\min_{\mathbf{T}} \iint_{\bar{\mathcal{R}}} \rho\left[\delta I(\mathbf{T},\mathbf{u})\right] d\mathbf{u}.$$

$$\delta I(\mathbf{T},\mathbf{u}) = I_k\left(\pi\left(\mathbf{T}\cdot\pi^{-1}(\mathbf{u},d_{\mathbf{u}})\right)\right) - I_{k-1}(\mathbf{u}) \quad \forall\ \mathbf{u}\in\bar{\mathcal{R}}$$

Feature-based

- Frame-to-Keyframe pose refinement

$$\mathbf{T}_{k,w} = \arg\min_{\mathbf{T}_{k,w}} \frac{1}{2}\sum_i \| \mathbf{u}_i - \pi(\mathbf{T}_{k,w}\ {}_w\mathbf{p}_i) \|^2$$



SVO: Semi-Direct Visual Odometry [ICRA'14]

Direct

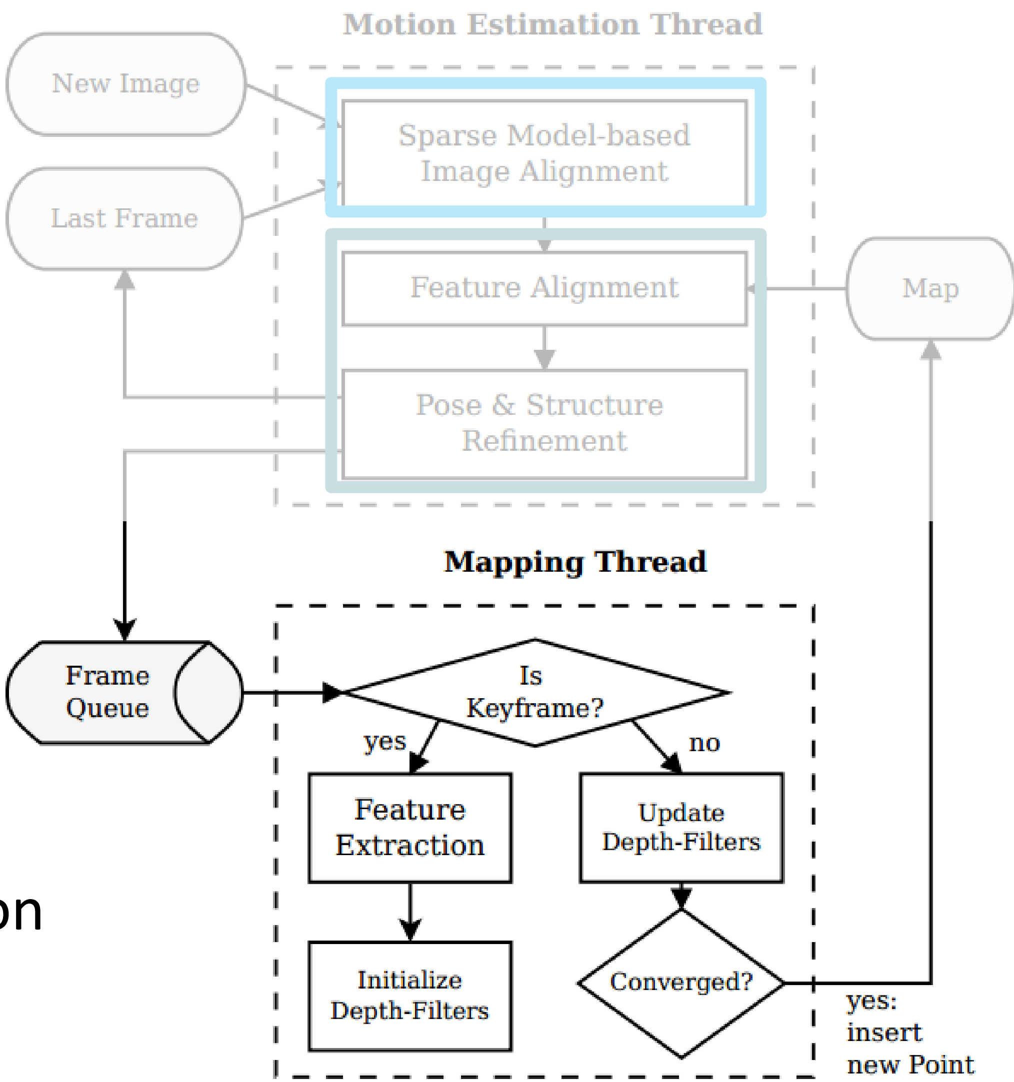
- Frame-to-frame motion estimation

Feature-based

- Frame-to-Kreyframe pose refinement

Mapping

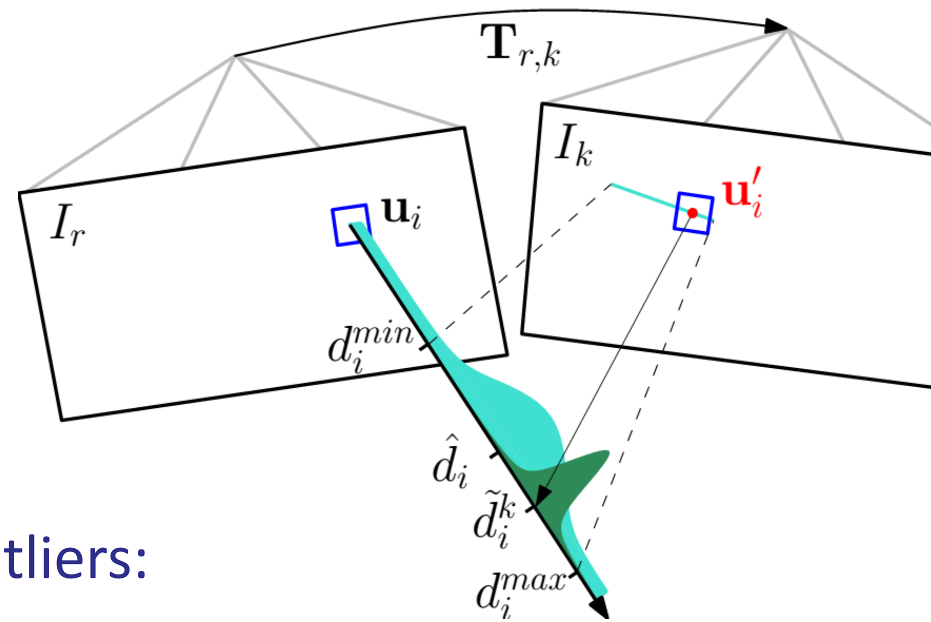
- Feature extraction only for every keyframe
- **Probabilistic depth** estimation of 3D points



Probabilistic Depth Estimation in SVO

Depth-Filter:

- Depth-filter for every new feature
- Recursive Bayesian depth estimation
- Epipolar search using ZMSSD



Measurement Likelihood models outliers:

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

- 2-Dimensional distribution: Depth $\boldsymbol{d}$ and inliner ratio $\boldsymbol{\rho}$
- Mixture of Gaussian + Uniform
- Inverse depth

Probabilistic Depth Estimation in SVO

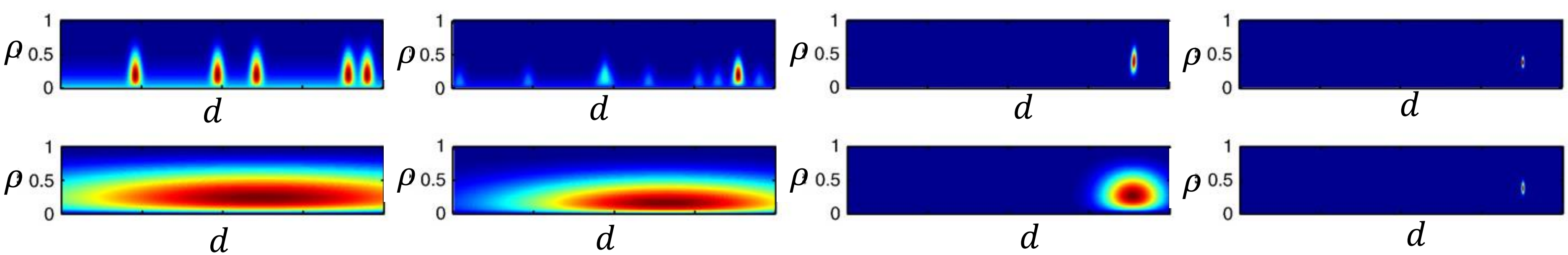
- Based on the model by (Vogiatzis &Hernandez, 2011) but with inverse depth

$$p(\hat{d}, \rho | d_{r+1}, \ldots, d_k) \propto p(\hat{d}, \rho) \prod_k p(d_k | \hat{d}, \rho) \quad (1)$$

$$p(d_k | \hat{d}, \rho) = \rho \mathcal{N}(d_k | \hat{d}, \tau_k^2) + (1 - \rho) \mathcal{U}(d_k | d_{min}, d_{max}) \quad (2)$$

- The posterior in (1) can be approximated by

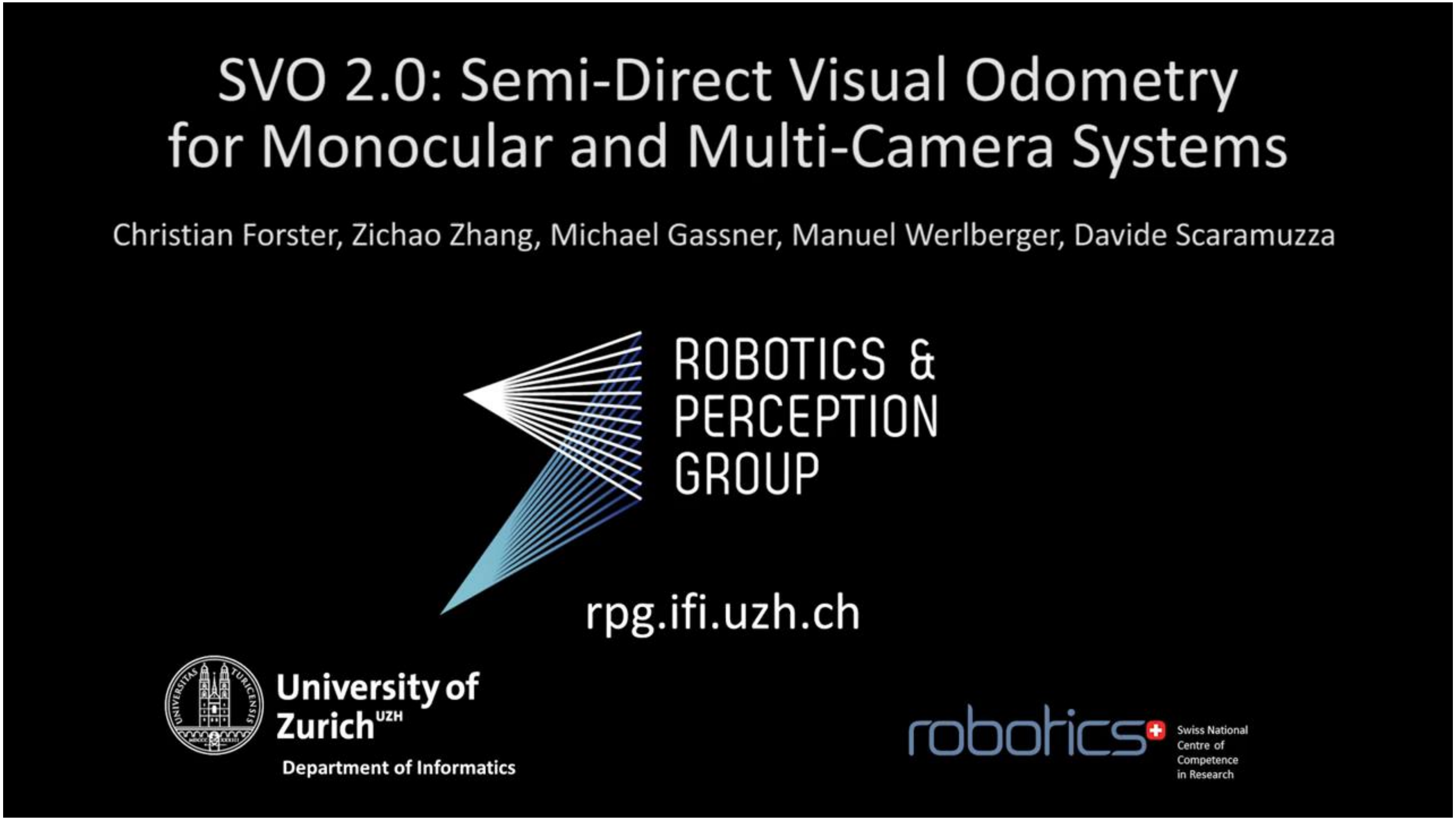
$$q(\hat{d}, \rho | a_k, b_k, \mu_k, \sigma_k^2) = Beta(\rho | a_k, b_k) \mathcal{N}(\hat{d} | \mu_k, \sigma_k^2) \quad (3)$$



The parametric model $\{a_k, b_k, \mu_k, \sigma_k^2\}$ describes the pixel depth at time *k*.

SVO: Video

https://youtu.be/hR8uq1RTUfA

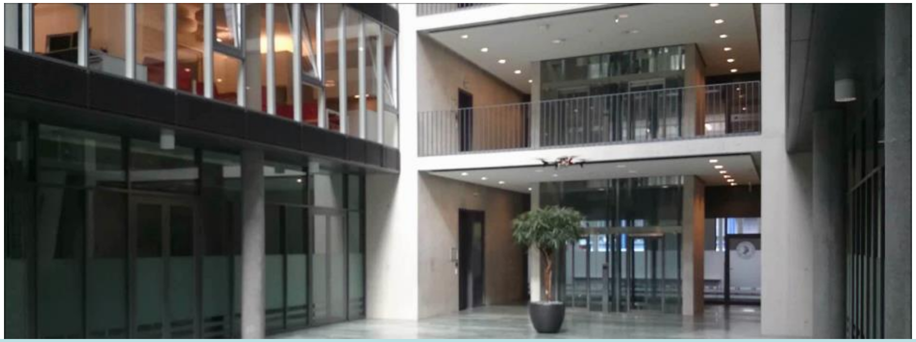


SVO for Autonomous Drone Navigation

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

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

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

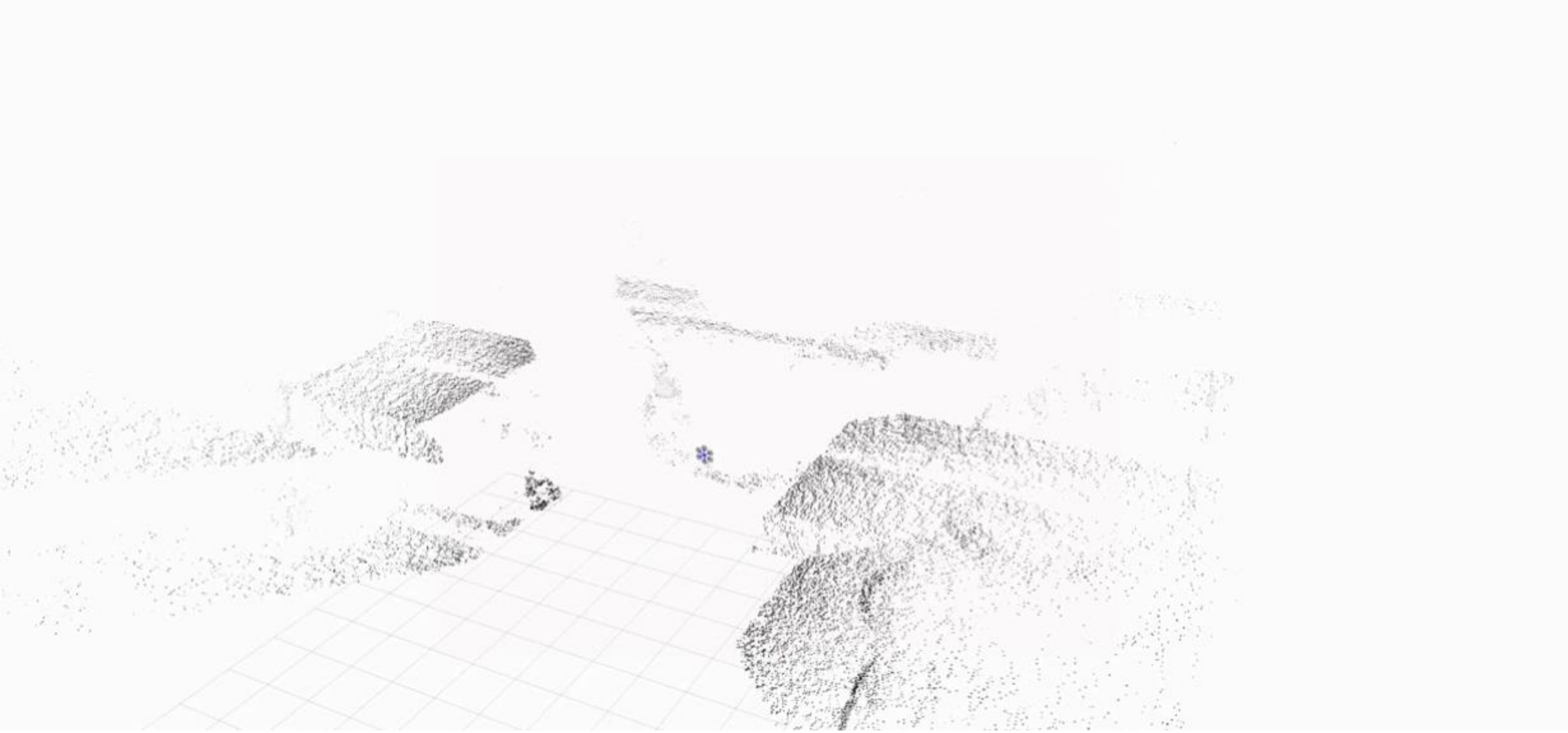


Video: https://youtu.be/fXy4P3nvxHQ

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**.

SVO on 4 fisheye Cameras from AUDI dataset

Video: https://www.youtube.com/watch?v=gr00Bf0AP1k



[Forster, et al., «SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems», TRO’16]

Processing Times of SVO

	Thread	Intel i7 [ms]	Jetson TX1 [ms]
Sparse image alignment	1	0.66	2.54
Feature alignment	1	1.04	1.40
Optimize pose & landmarks	1	0.42	0.88
Extract features	2	1.64	5.48
Update depth filters	2	1.80	2.97

TABLE III: Mean time consumption in milliseconds by individual components of SVO Mono on the EUROC Machine Hall 1 dataset. We report timing results on a laptop with Intel Core i7 (2.80 GHz) processor and on the NVIDIA Jetson TX1 ARM processor.

[Forster, et al., «SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems», TRO'16]

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

Summary: Feature-based vs. direct

Feature-based (ORB-SLAM, part of SVO/DSO)

1. Feature extraction
2. Feature matching
3. RANSAC + P3P
4. **Reprojection error** minimization

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

✓ Large frame-to-frame motions

✗ Slow (20-30 Hz) due to costly feature extraction and matching

✗ Not robust to high-frequency and repetive texture

✗ Outliers

Direct approaches (LSD, DSO, SVO)

1. Minimize photometric error

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

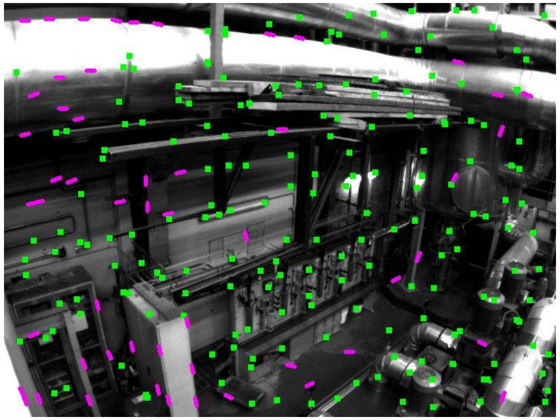
✓ Every pixel in the image can be exploited (precision, robustness)

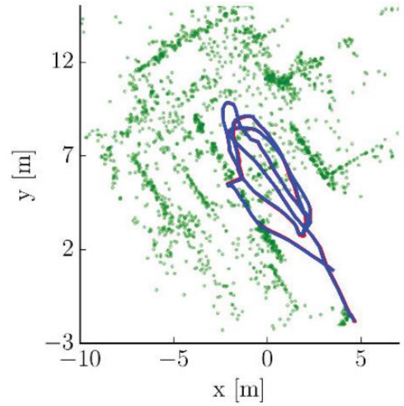
✓ Increasing camera frame-rate reduces computational cost per frame

✗ Limited to small frame-to-frame motion

Comparison among SVO, DSO, ORB-SLAM, LSD-SLAM [Forster, TRO'16]

- See next two slides
- For a thorough evaluation please refer to [Forster, TRO'16] paper, where all these algorithms are evaluated in terms of accuracy against ground truth and timing on several datasets: EUROC, TUM-RGB-D, ICL-NUIM





[Forster, et al., «SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems», TRO'16]

Accuracy (EUROC Dataset) [Forster, TRO’16]

	Monocular						
	SVO (edgelets + prior)	SVO (bundle adjustment)	ORB-SLAM (no loop-closure)	ORB-SLAM (no loop, real-time)	DSO	DSO (real-time)	LSD-SLAM (no loop-closure)
Machine Hall 01	0.10	0.06	**0.02**	0.61	0.05	0.05	0.18
Machine Hall 02	0.12	0.07	**0.03**	0.72	0.05	0.05	0.56
Machine Hall 03	0.41	×	**0.03**	1.70	0.18	0.26	2.69
Machine Hall 04	0.43	0.40	0.22	6.32	2.50	**0.24**	2.13
Machine Hall 05	0.30	×	0.71	5.66	**0.11**	0.15	0.85
Vicon Room 1 01	0.07	**0.05**	0.16	1.35	0.12	0.47	1.24
Vicon Room 1 02	0.21	×	0.18	0.58	0.11	**0.10**	1.11
Vicon Room 1 03	×	×	0.78	0.63	0.93	**0.66**	×
Vicon Room 2 01	0.11	×	**0.02**	0.53	0.04	0.05	×
Vicon Room 2 02	**0.11**	×	0.21	0.68	0.13	0.19	×
Vicon Room 2 03	1.08	×	1.25	**1.06**	1.16	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’16]

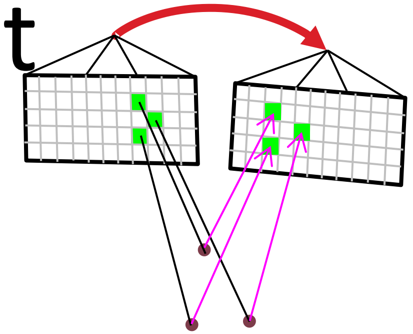
Timing (EUROC Dataset) [Forster, TRO'16]

	Mean	St.D.	CPU@20 fps
SVO Mono	2.53	0.42	55 ±10%
SVO Mono + Prior	**2.32**	**0.40**	**70 ± 8%**
SVO Mono + Prior + Edgelet	2.51	0.52	73 ± 7%
SVO Mono + Bundle Adjustment	5.25	10.89	72 ±13%
SVO Stereo	4.70	1.31	90 ± 6%
SVO Stereo + Prior	**3.86**	**0.86**	**90 ± 7%**
SVO Stereo + Prior + Edgelet	4.12	1.11	91 ± 7%
SVO Stereo + Bundle Adjustment	7.61	19.03	96 ±13%
ORB Mono SLAM (No loop closure)	29.81	5.67	187 ±32%
LSD Mono SLAM (No loop closure)	23.23	5.87	236 ±37%

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'16]

Review of Direct Image Alignment



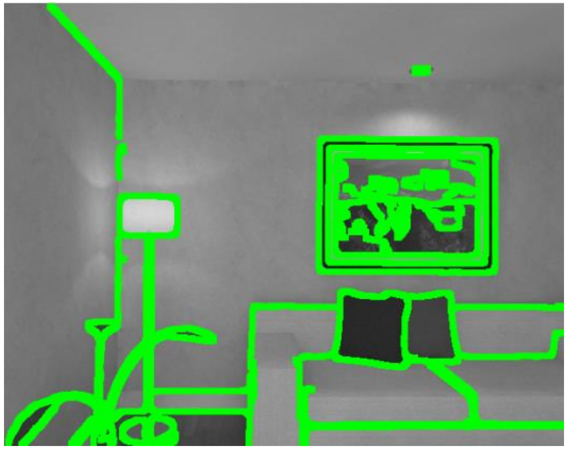
It minimizes the **per-pixel intensity difference**

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

Dense

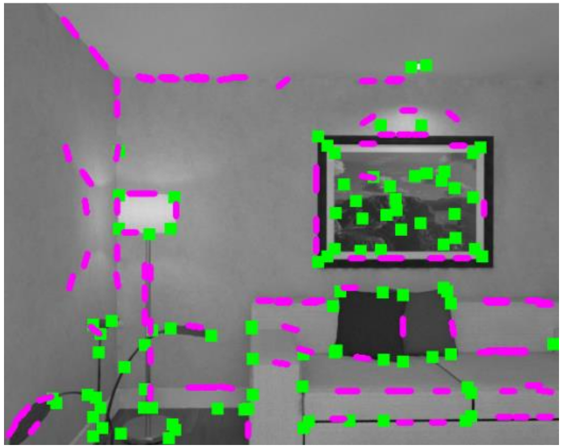
DTAM [Newcombe et al. ‘11]
300’000+ pixels

Semi-Dense



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

Sparse

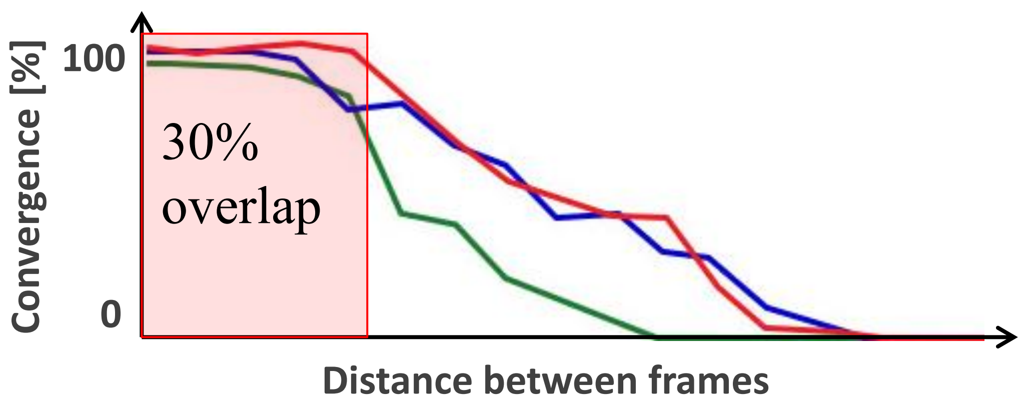


SVO [Forster et al. 2014, TRO’16]
100-200 features x 4x4 patch
~ 2,000 pixels

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

Dense vs Semi-dense vs Sparse: what's best? [TRO'16]

- Goal: study the magnitude of the **perturbation for which image-to-model alignment is capable to converge** as a function of the distance to the reference image
- The performance in this experiment is a measure of robustness: successful pose estimation from large initial perturbations shows that the algorithm is capable of dealing with rapid camera motions
- 1000 Blender simulations
- Alignment considered converged when the estimated relative pose is closer than 0.1 meters from ground-truth
- Result: difference between semi-dense image alignment and dense image alignment is marginal. This is because pixels that exhibit no intensity gradient are not informative for the optimization (their Jacobians are zero).
  - Using all pixels becomes only useful when considering motion blur and image defocus



[Forster, et al., «SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems», TRO'16]

Summary: keyframe and filter-based method

- Why the parallel structure in all these algorithms?
  - Mapping is often expensive
    - Local BA
    - Loop detection and graph optimization
    - Depth filter per feature
  - Using the best map available for **real-time** tracking [1]
- Why not filter-based method?
  - Keyframe-based: more accuracy per unit computing time [2]
  - Still useful in visual-inertial fusion
    - MSCKF
    - ROVIO

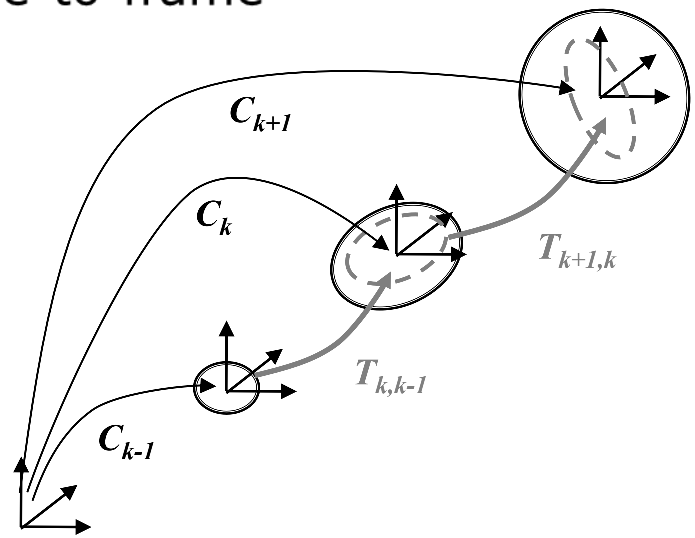
[1] Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces.
[2] Strasdat, Hauke, José MM Montiel, and Andrew J. Davison. "Visual SLAM: why filter?."

Error Propagation

VO Drift

- The errors introduced by each new frame-to-frame motion accumulate over time

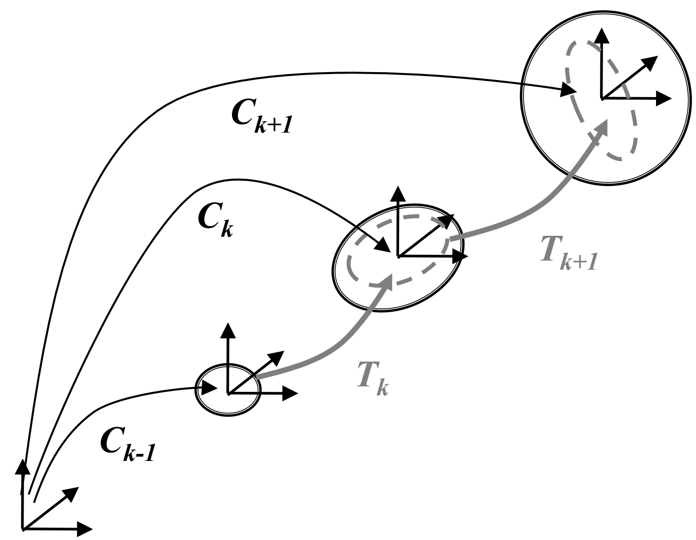
- This generates a drift of the estimated trajectory from the real path



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

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

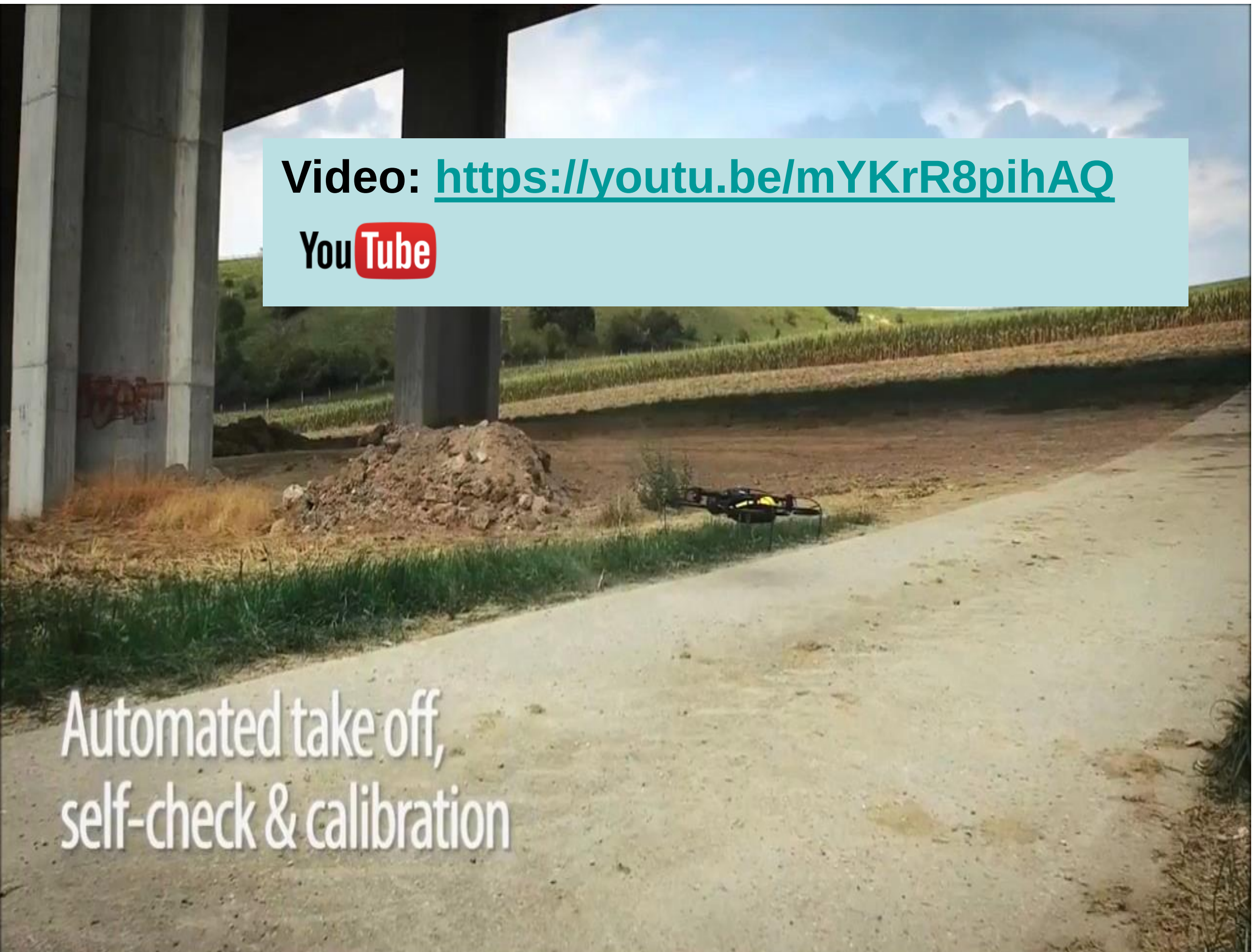
$$\begin{aligned}\Sigma_k &= J \begin{bmatrix} \Sigma_{k-1} & 0 \\ 0 & \Sigma_{k,k-1} \end{bmatrix} J^\top \\ &= J_{\vec{C}_{k-1}} \Sigma_{k-1} J_{\vec{C}_{k-1}}{}^\top + J_{\vec{T}_{k,k-1}} \Sigma_{k,k-1} J_{\vec{T}_{k,k-1}}{}^\top\end{aligned}$$

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

Commercial Applications of SVO

Application: Autonomous Inspection of Bridges and Power Masts

Project with Parrot: Autonomous vision-based navigation



Video: https://youtu.be/mYKrR8pihAQ

YouTube

Parrot senseFly

Albris drone

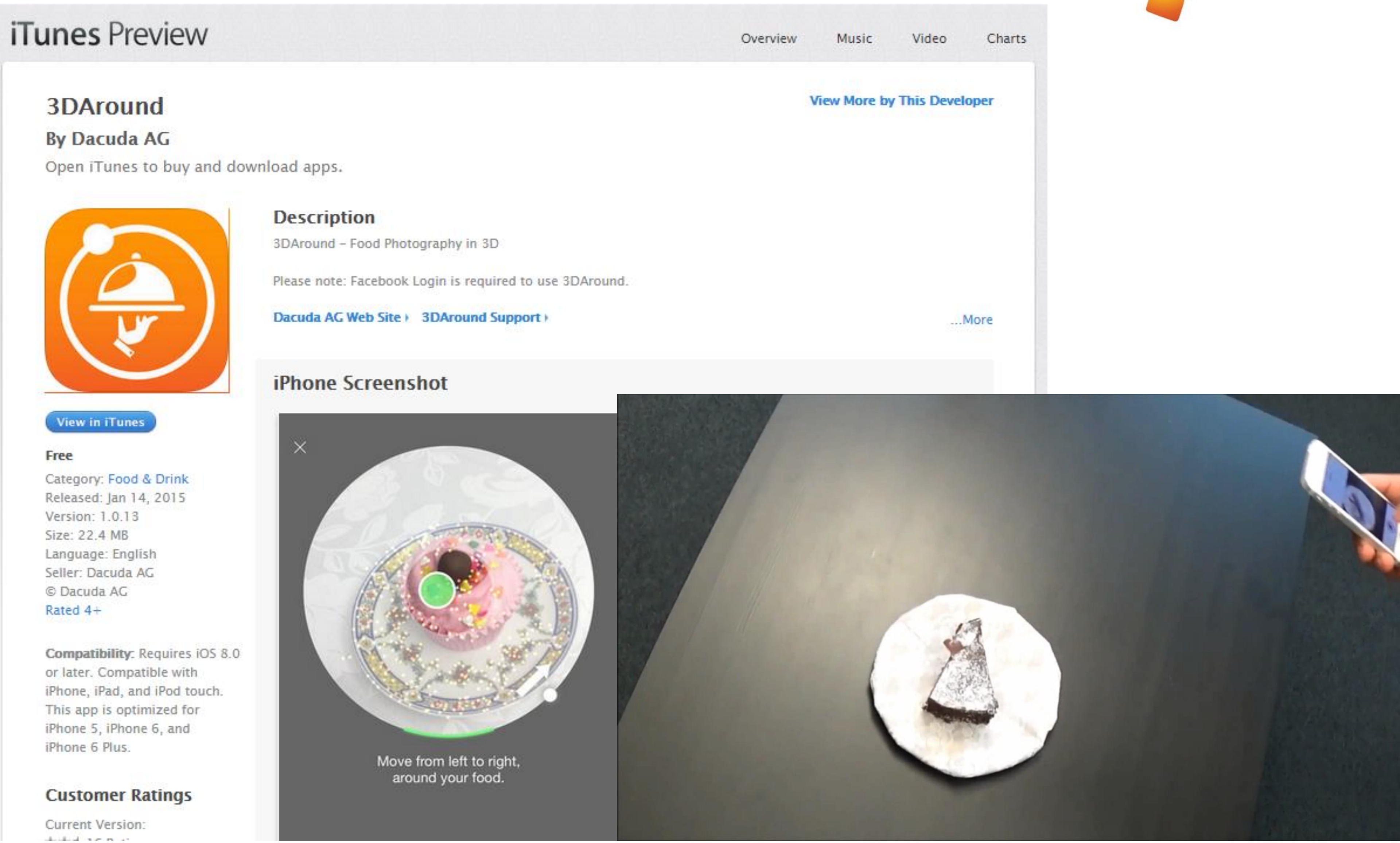
Dacuda VR solutions

Dacuda

- Fully immersive virtual reality with 6-DoF for VR and AR content (running on iPhone): https://www.youtube.com/watch?v=k0MLs5mqRNo
- Powered by SVO

3DAround iPhone App

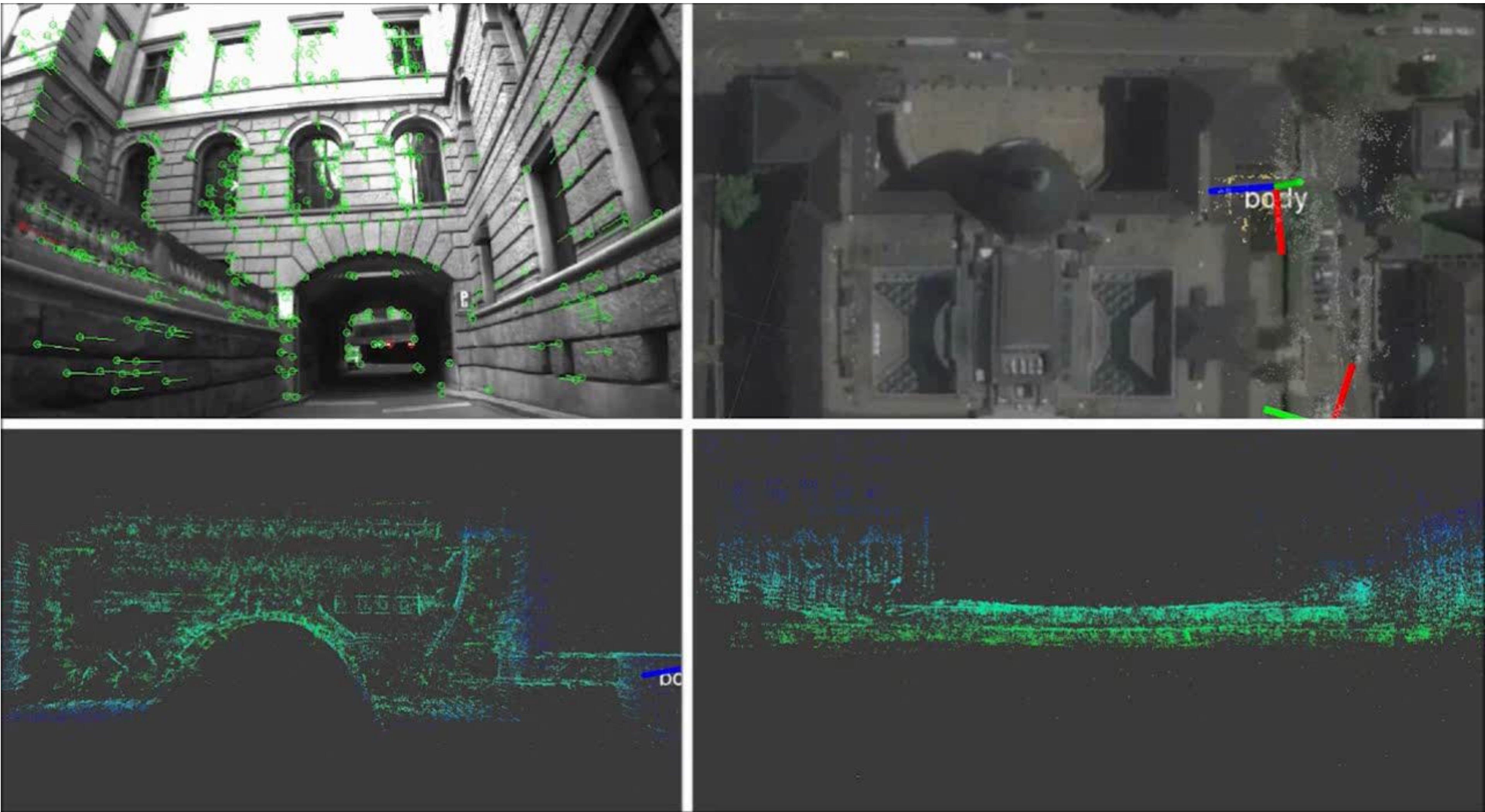
Dacuda



Zurich-Eye – www.zurich-eye.com

Vision-based Localization and Mapping Solutions for Mobile Robots

Started in Sep. 2015, **became Facebook-Oculus R&D Zurich in Sep. 2016**



Event-based Vision

Open Problems and Challenges in Agile Robotics

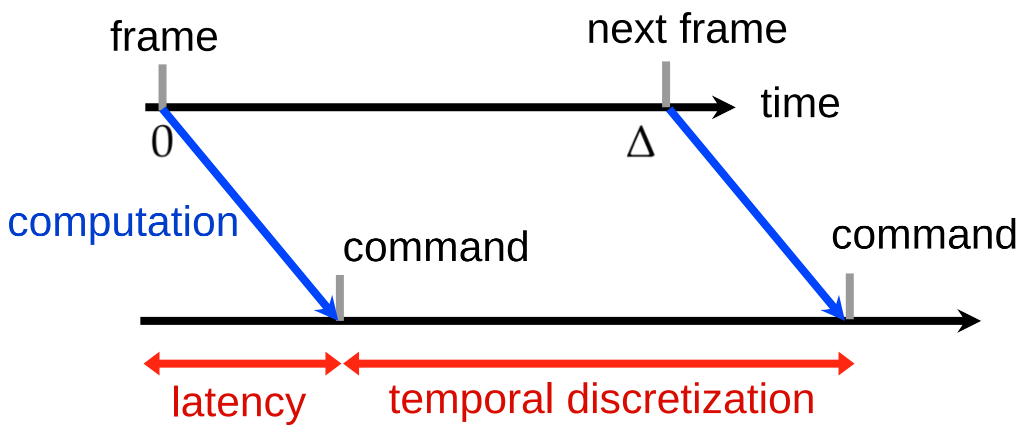
Current flight maneuvers achieved with onboard cameras are still to slow compared with those attainable by birds or FPV pilots

[FPV-Drone race]

To go faster, we need faster sensors!

- At the current state, the agility of a robot is limited by the latency and temporal discretization of its sensing pipeline.

- Currently, the average robot-vision algorithms have latencies of 50-200 ms. This puts a hard bound on the agility of the platform.

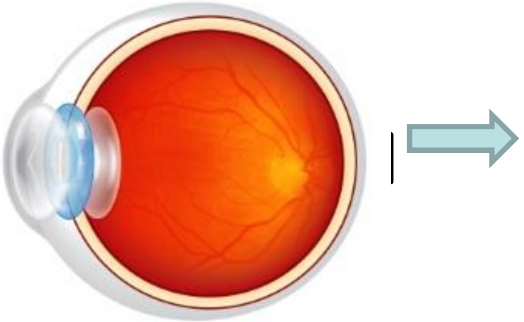


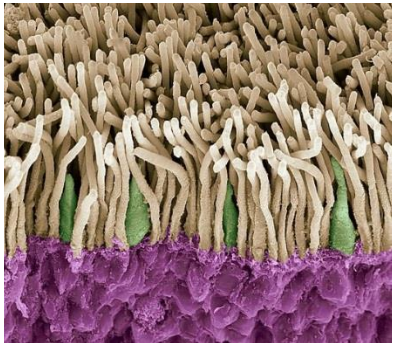
- **Can we create a low-latency, low-discretization perception pipeline?**
    - Yes, if we combine **cameras with event-based** sensors

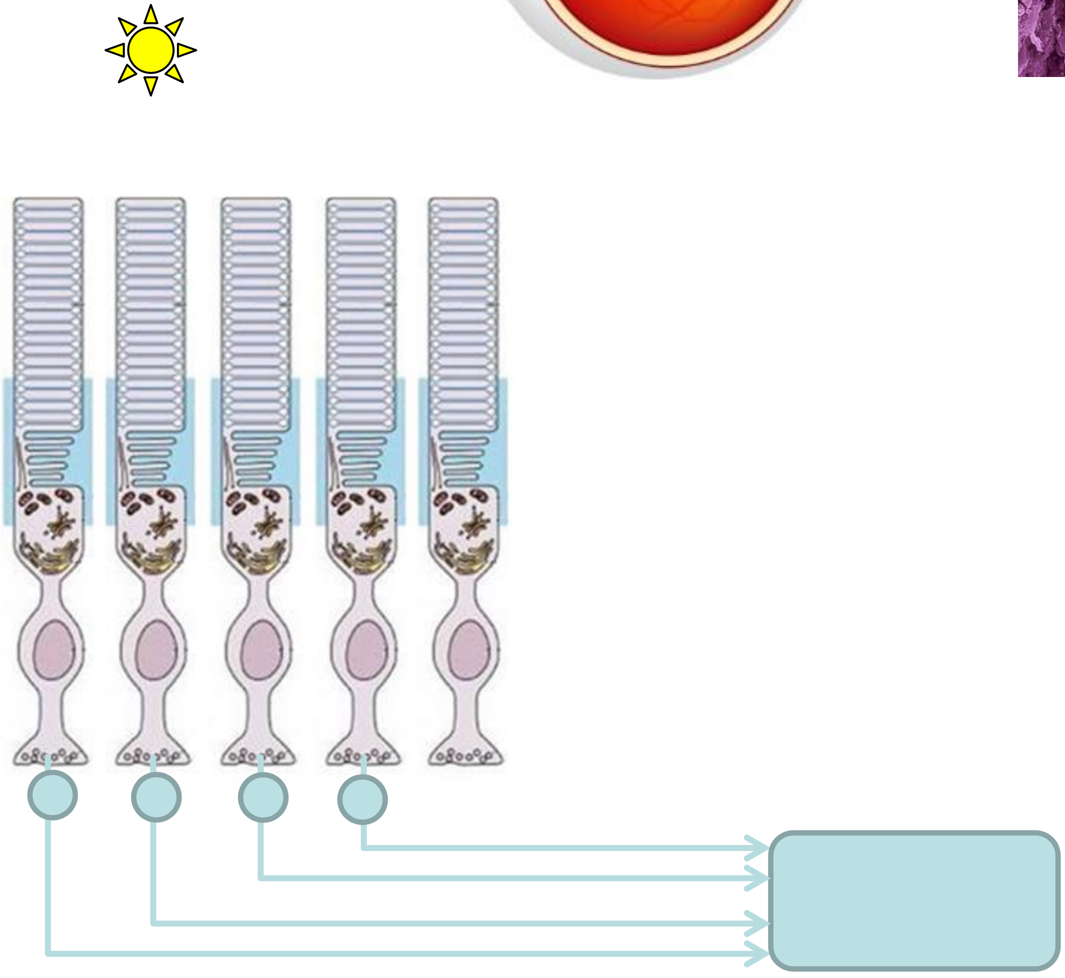
[Censi & Scaramuzza, «Low Latency, Event-based Visual Odometry», ICRA’14]

Human Vision System

- 130 million **photoreceptors**
- But only 2 million **axons**!







Dynamic Vision Sensor (DVS)

- **Event-based camera** developed by Tobi Delbruck's group (ETH & UZH).
- Temporal resolution: **1 μs**
- High dynamic range: **120 dB**
- Low power: **20 mW**
- Cost: 2,500 EUR

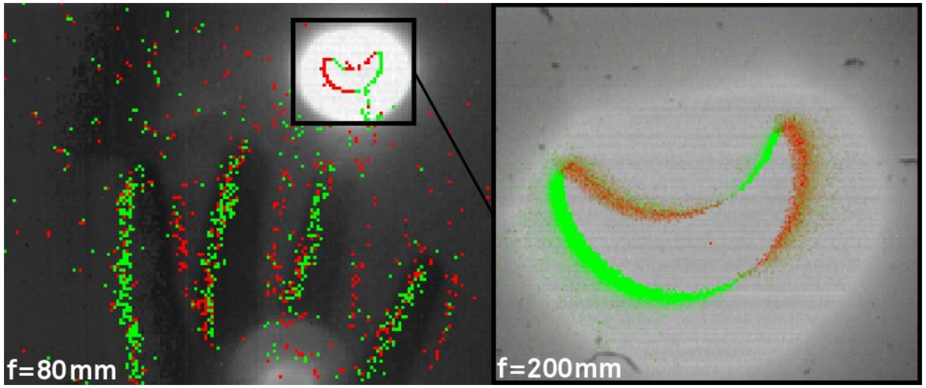
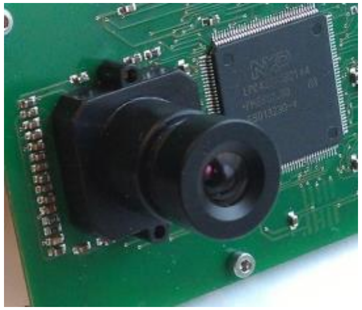


Image of the solar eclipse (March'15) captured by a DVS (courtesy of IniLabs)

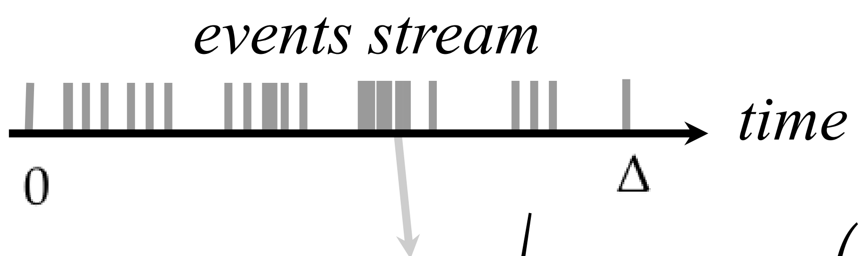


[Lichtsteiner, Posch, Delbruck. A 128x128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor. 2008]

Camera vs DVS

- A **traditional camera** outputs frames at **fixed time intervals**:

- By contrast, a **DVS** outputs **asynchronous events** at ***microsecond*** **resolution**. An event is generated each time a single pixel detects an intensity changes value

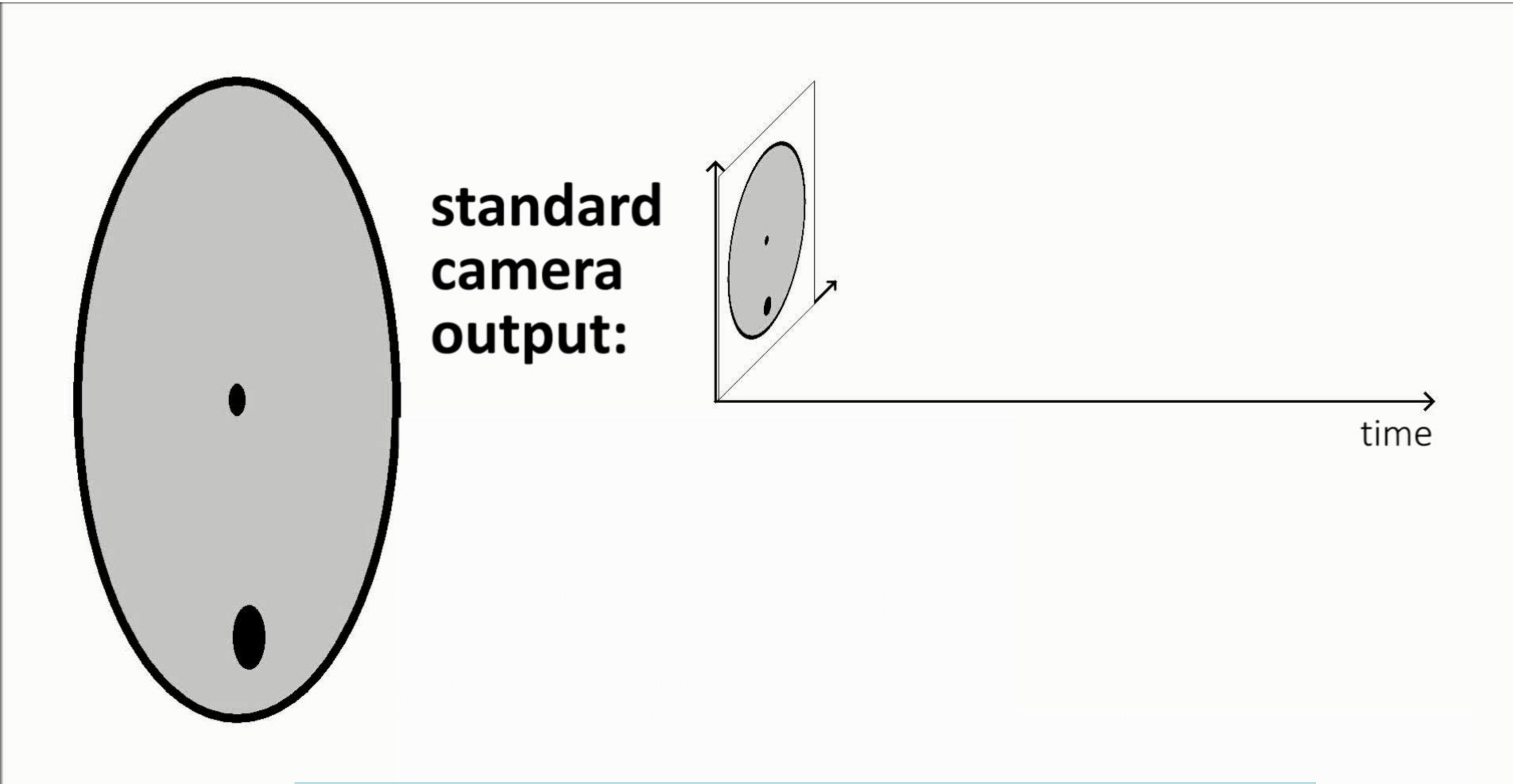


$$\textit{event: } \left\langle t, \langle x, y \rangle, sign\left(\frac{dI(x,y)}{dt}\right)\right\rangle$$

sign (+1 or -1)

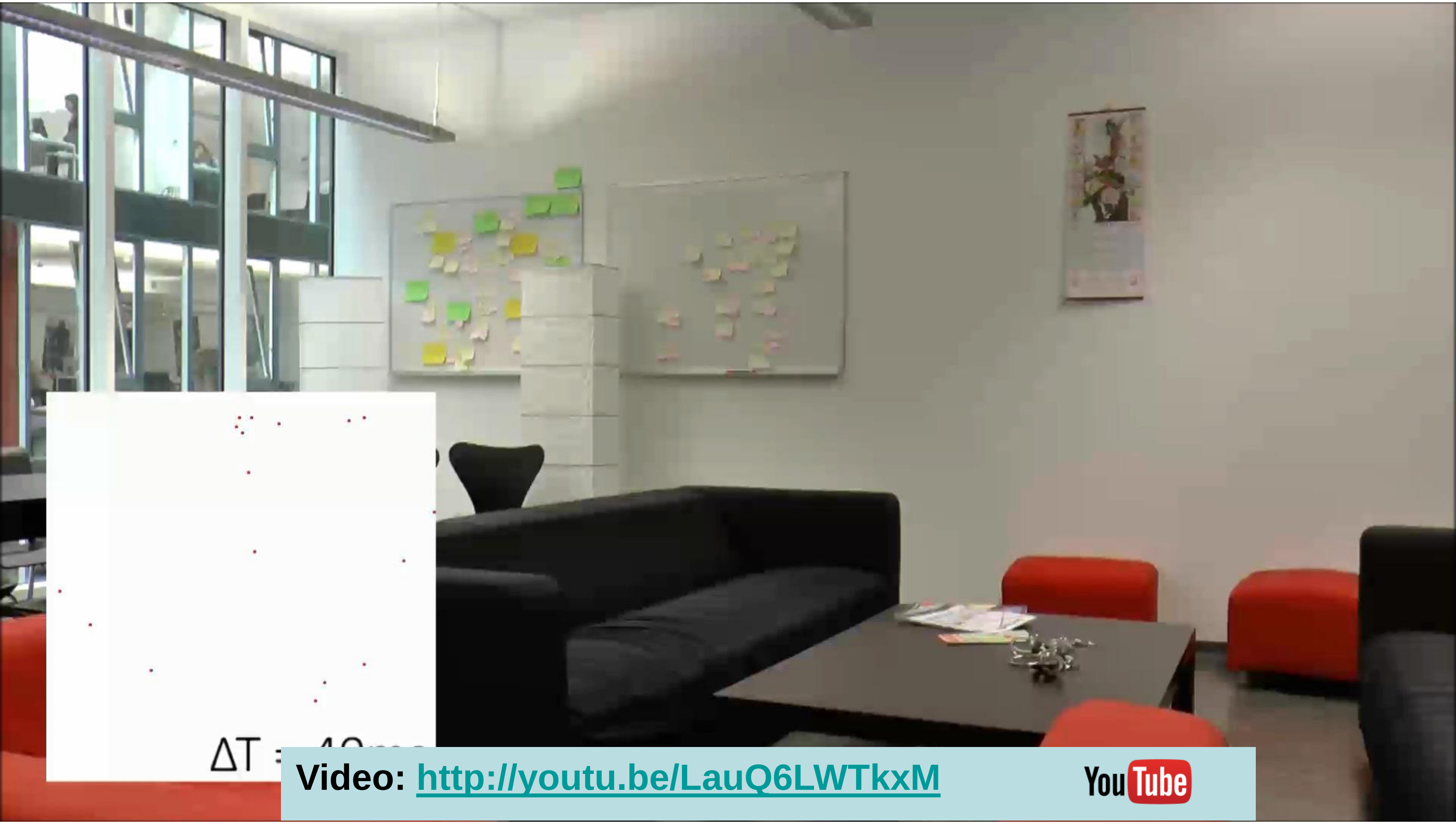
Lichtsteiner, Posch, Delbruck. *A 128x128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor.* 2008

Camera vs Dynamic Vision Sensor



Video: http://youtu.be/LauQ6LWTkxM YouTube

Camera vs Dynamic Vision Sensor

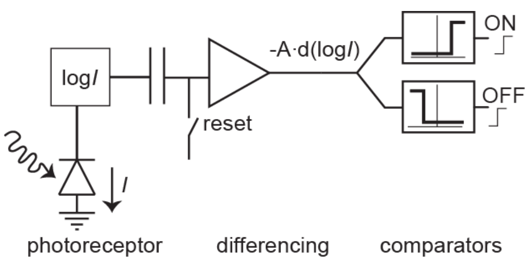


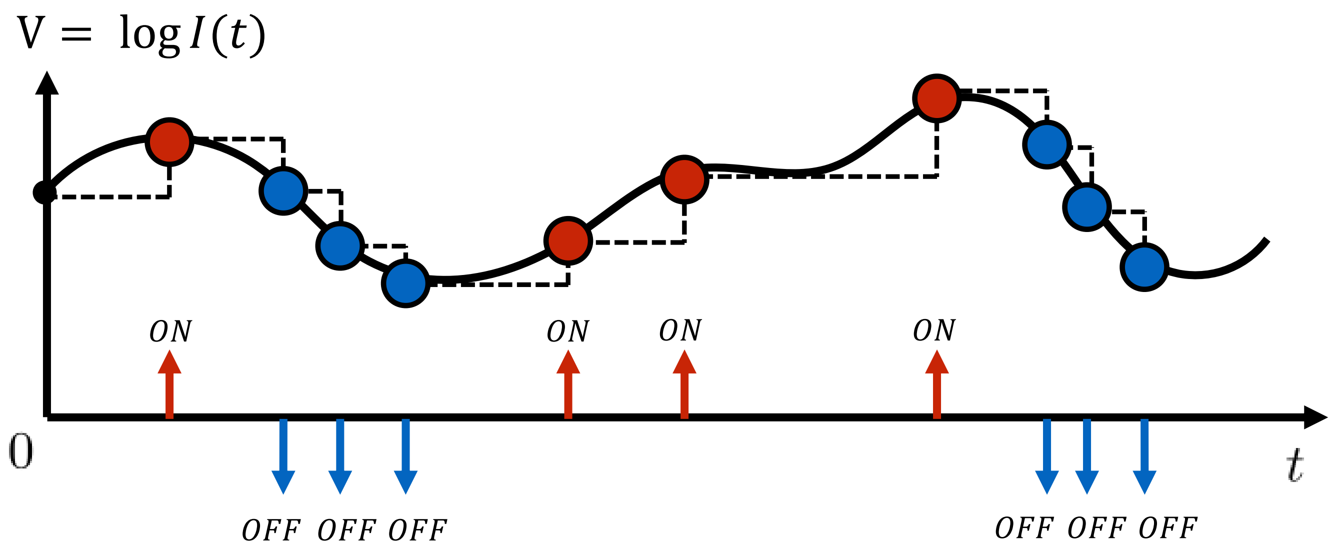
Video: http://youtu.be/LauQ6LWTkxM YouTube

DVS Operating Principle [Lichtsteiner, ISCAS'09]

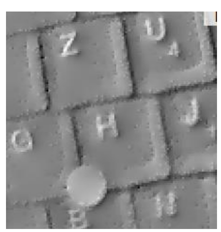
Events are generated any time a single pixel sees a change in brightness larger than C

$$\Delta \log I \geq C$$





The intensity signal at the event time can be reconstructed by integration of $\pm C$

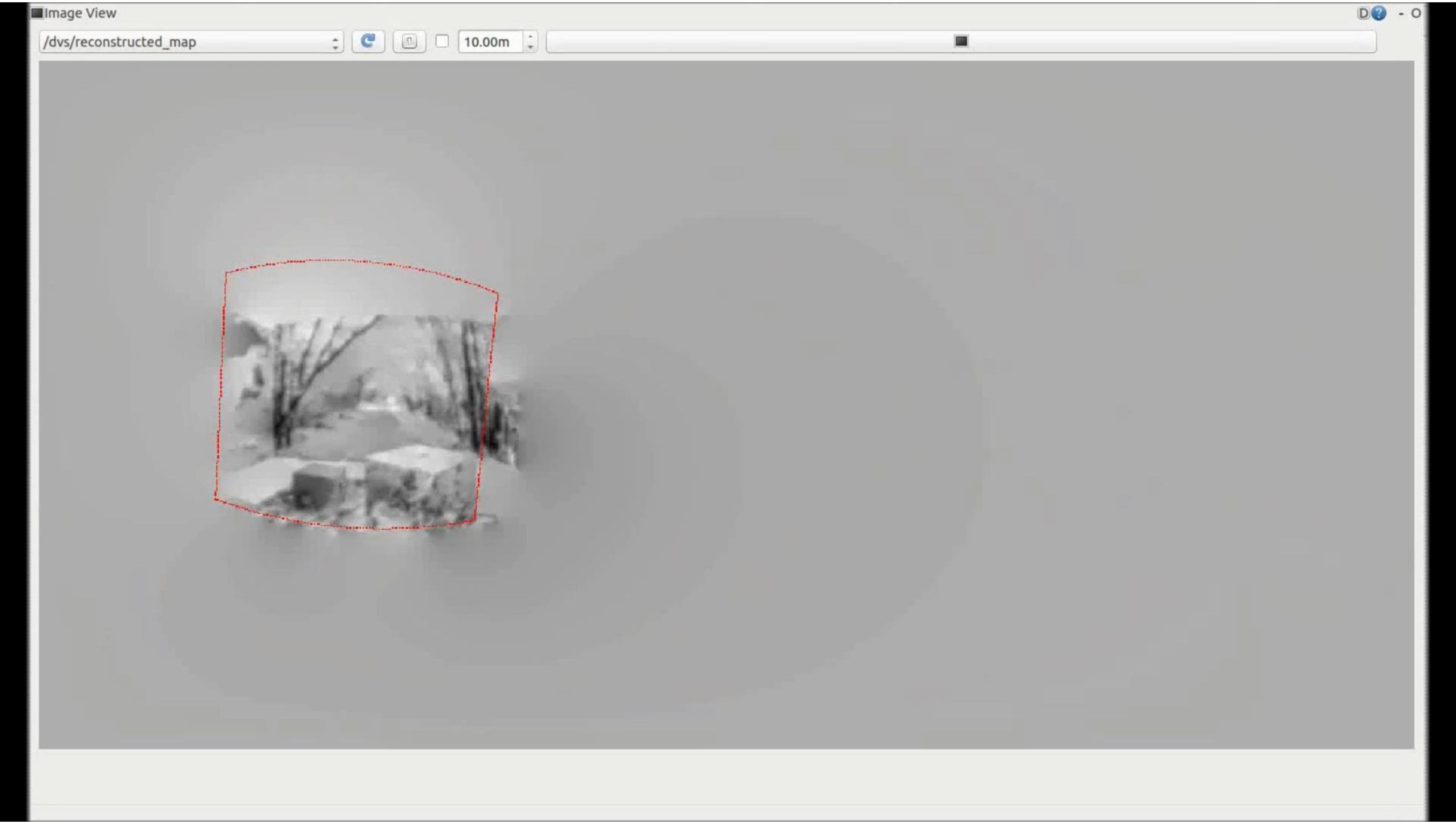


[Cook et al., IJCNN'11]

[Kim et al., BMVC'15]

[Lichtsteiner, Posch, Delbruck. A 128x128 120 dB 15µs Latency Asynchronous Temporal Contrast Vision Sensor. 2008]

Pose Tracking and Intensity Reconstruction from a DVS



Dynamic Vision Sensor (DVS)

Advantages

- **low-latency** (~1 micro-second)
- **high-dynamic range** (120 dB instead 60 dB)
- Very **low bandwidth** (only intensity changes are transmitted): ~200Kb/s
- **Low storage capacity, processing time, and power**

Disadvantages

- Require totally **new vision algorithms**
- **No intensity information** (only binary intensity changes)

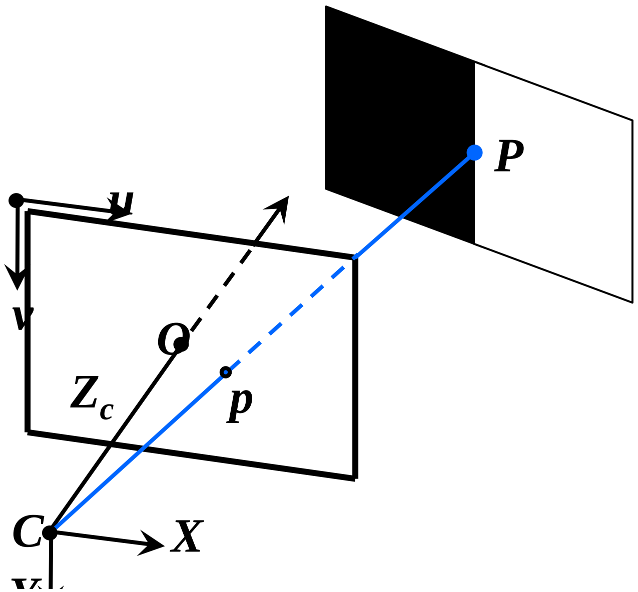
Lichtsteiner, Posch, Delbruck. A 128x128 120 dB 15µs Latency Asynchronous Temporal Contrast Vision Sensor. 2008

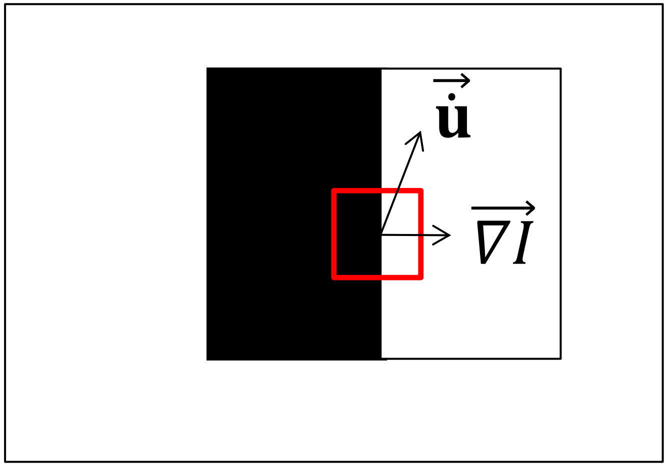
Generative Model [Censi & Scaramuzza, ICRA'14]

The generative model tells us that the **probability** that an event is generated depends on the **scalar product** between the gradient ∇I and the apparent motion $\dot{\mathbf{u}}\Delta t$

$$\mathrm{P(e)} \propto |\langle \nabla I, \dot{\mathbf{u}}\Delta t \rangle|$$

Generative event model: $\langle \nabla \Delta \log I, \dot{\mathbf{u}}\Delta t \rangle = C$





[Event-based Camera Pose Tracking using a Generative Event Model, Arxiv]

[Censi & Scaramuzza, *Low Latency, Event-based Visual Odometry*, ICRA'14]

Event-based 6DoF Pose Estimation Results

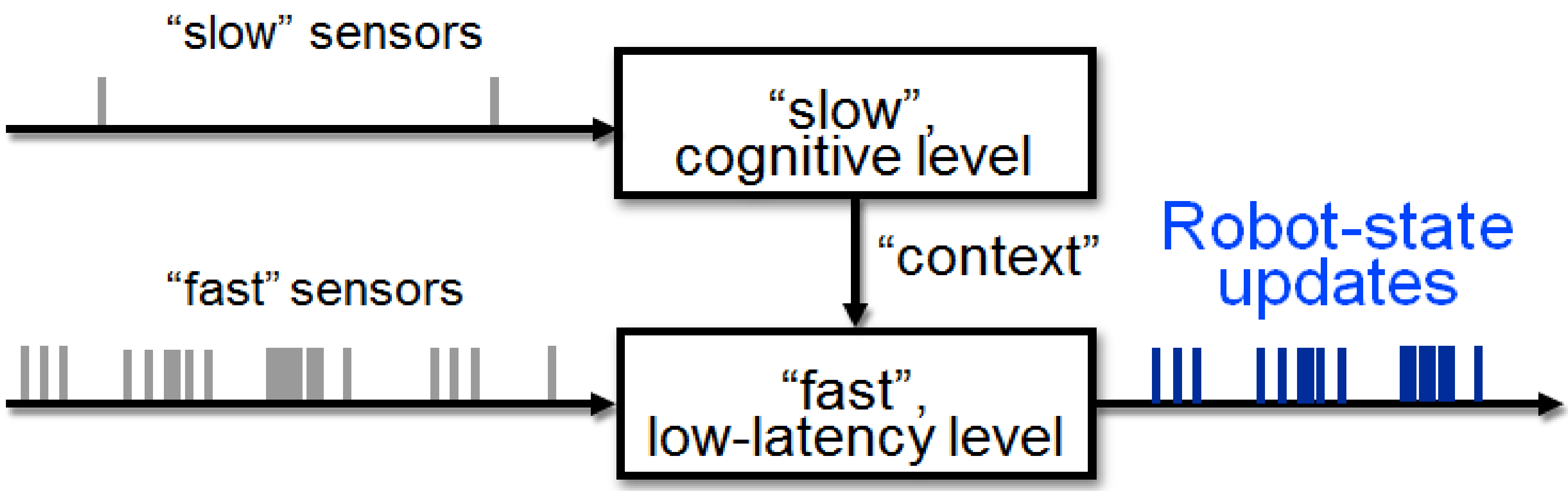
[Event-based Camera Pose Tracking using a Generative Event Model, Arxiv]
[Censi & Scaramuzza, *Low Latency, Event-based Visual Odometry*, ICRA'14]

Robustness to Illumination Changes and High-speed Motion

Video: https://www.youtube.com/watch?v=EUX3Tfx0KKE

[BMVC'16 : EMVS: Event-based Multi-View Stereo, **Best Industry Paper Award**]()

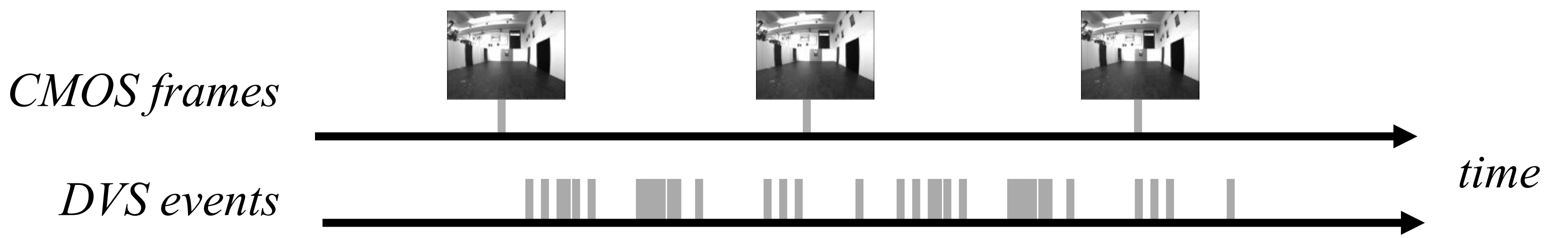
Possible future sensing architecture



[Censi & Scaramuzza, Low Latency, Event-based Visual Odometry, ICRA’14]

DAVIS: Dynamic and Active-pixel Vision Sensor [Brandli'14]

Combines an event camera with a frame-based camera in the same pixel array!

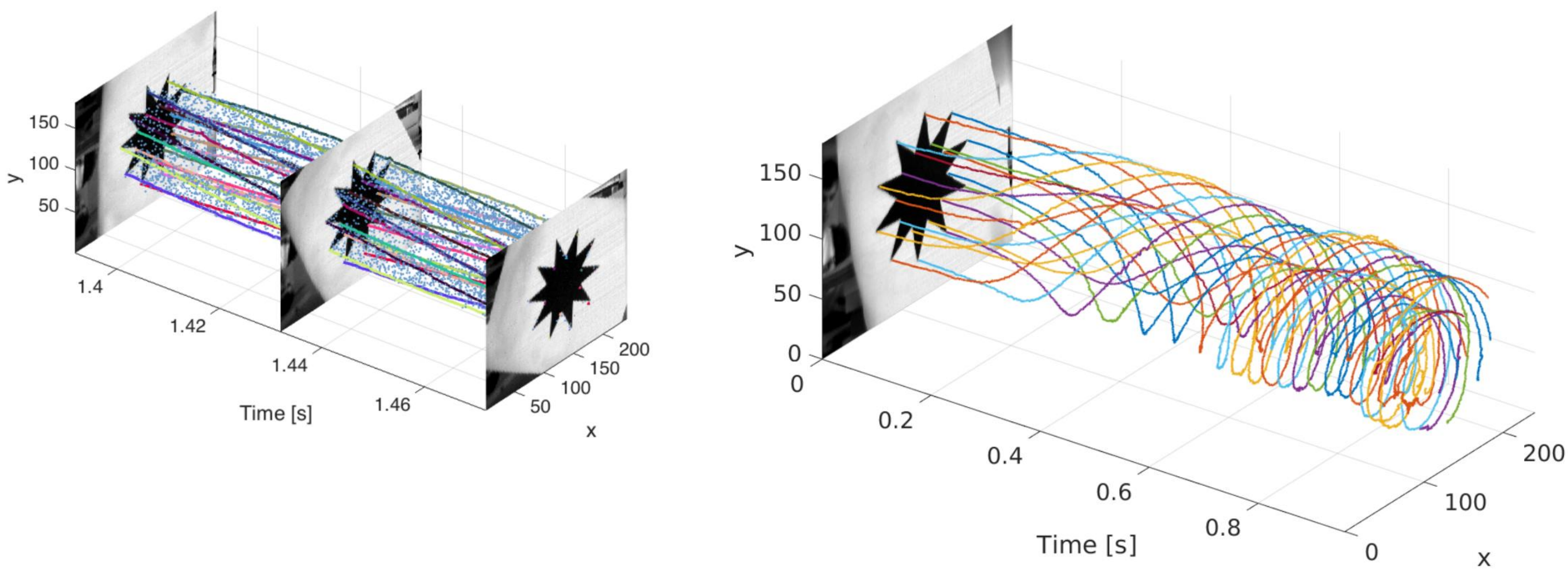


Brandli, Berner, Yang, Liu, Delbruck, "A 240× 180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor." IEEE Journal of Solid-State Circuits, 2014.

Event-based Feature Tracking [IROS'16]

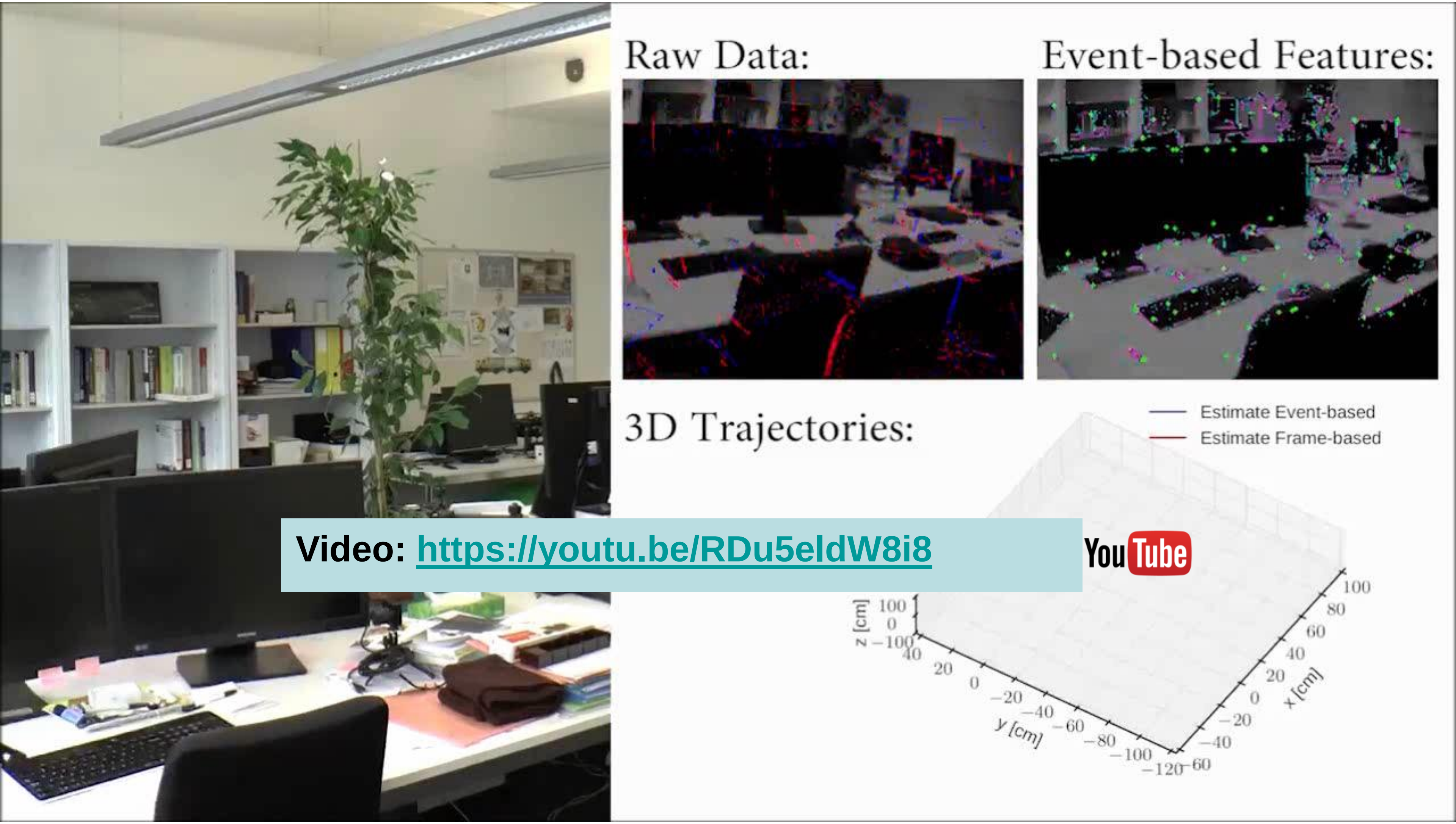
- Extract Harris corners on images
- Track corners using event-based Iterative Closest Points (ICP)

$$\arg\min_{\mathtt{A}} \sum_{(\mathbf{p}_i, \mathbf{m}_i) \in \text{Matches}} \|\mathtt{A}(\mathbf{p}_i) - \mathbf{m}_i\|^2$$



IROS'16 : Low-Latency Visual Odometry using Event-based Feature Tracks, **Best application paper award finalist**

Event-based, Sparse Visual Odometry [IROS'16]



Video: https://youtu.be/RDu5eldW8i8

IROS'16 : Event-based Feature Tracking for Low-latency Visual Odometry, **Best application paper award finalist**

Conclusions

- VO & SLAM theory **well established**
- Biggest challenges today are **reliability and robustness**
  - **HDR scenes**
  - **High-speed motion**
  - **Low-texture scenes**
- **Which VO/SLAM is best?**
  - Depends on the task and how you measure the performance!
    - E.g., VR/AR/MR vs Robotics
- **99% of SLAM algorithms are passive: need active SLAM!**
- **Event cameras** open enormous possibilities! Standard cameras have been studied for 50 years!
  - Ideal for **high speed motion** estimation and robustness to **HDR illumination changes**

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
- **DSO** (direct sparse odometry): https://github.com/JakobEngel/dso

VIO

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

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 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/

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
- **OKVIS** (non-linear optimization): https://github.com/ethz-asl/okvis

Dense SFM **for MAVs** (i.e., (offline))

- Open source:
  - MAVMAP: https://github.com/mavmap/mavmap
- Closed source:
  - Pix4D: https://pix4d.com/

Place Recognition

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

VO and VIO Datasets

VO Datasets

- **Malaga dataset**: http://www.mrpt.org/malaga_dataset_2009
- **KITTI Dataset**: http://www.cvlibs.net/datasets/kitti/

VIO Datasets

These datasets include ground-truth 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

More

- Check out also this: http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm

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