ROBOTICS & PERCEPTION GROUP

Universität Zürich[UZH]

# Lecture 09
# Multiple View Geometry 3

Davide Scaramuzza

# Lab Exercise 6 - Today

- Room ETH HG E 33.1 from 14:15 to 16:00
- Work description:  P3P algorithm and RANSAC

# Outline

- Bundle Adjustment
- SFM with $n$ views

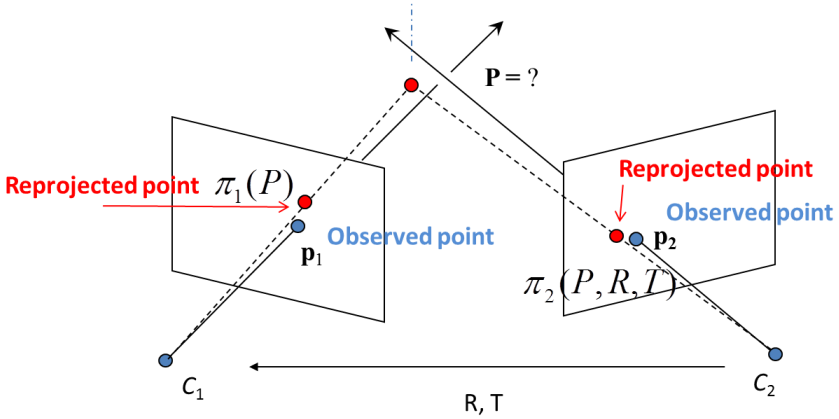
# Bundle Adjustment (BA)

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

$$(R,T,P^i) = \arg\min_{R,T,P^i} \sum_{i=1}^{N} \left\| \mathrm{p}_1^i - \pi_1(\mathrm{P}^i, \mathrm{C}_1) \right\|^2 + \left\| \mathrm{p}_2^i - \pi_2(\mathrm{P}^i, \mathrm{C}_2) \right\|^2$$

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

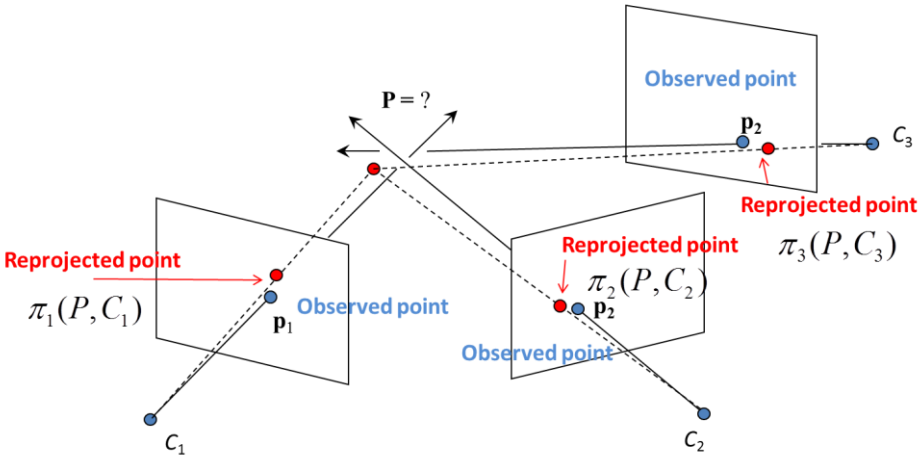
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)
- In order to not get stuck in local minima, the **initialization should be close the minimum**

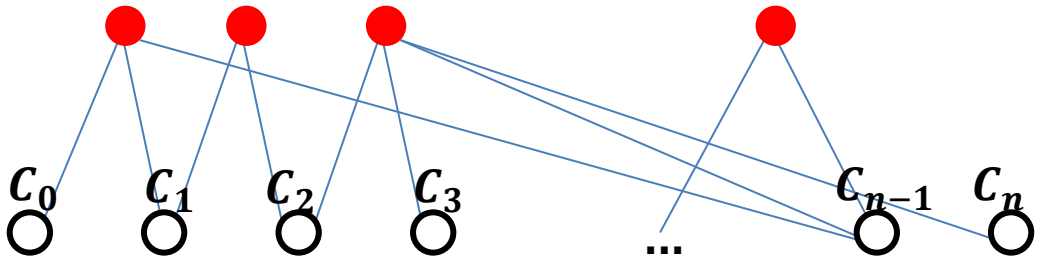


# Bundle Adjustment (BA) for $n$ Views

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

$$(X^i, C_k) = \arg\min_{X^i, C_k} \sum_{k} \sum_{i=1} \left\| \mathrm{p}_k^i - \pi_k(X^i, C_k) \right\|^2$$



# Outline

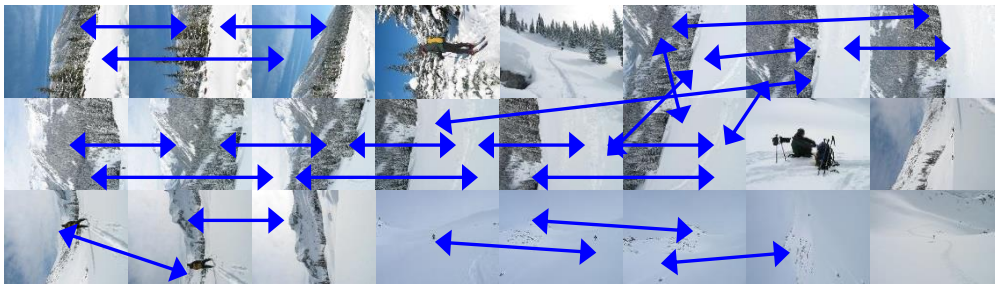
- Bundle Adjustment
- SFM with $n$ views

# Structure From Motion with $n$ Views

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

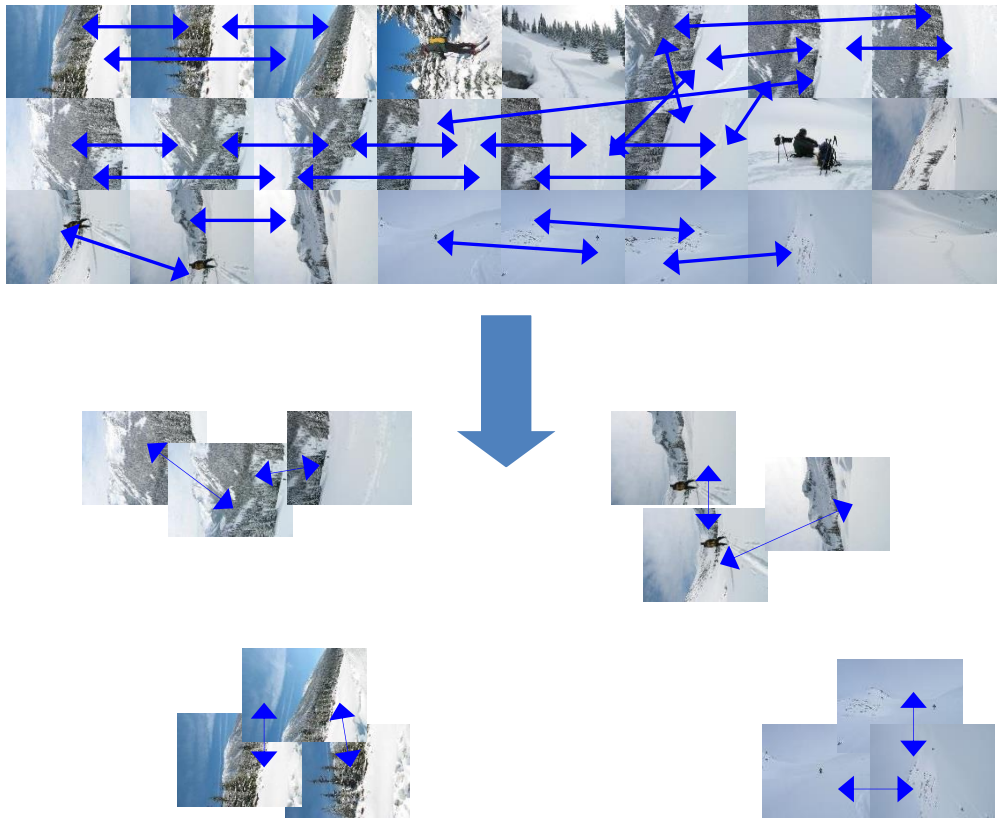
# Hierarchical SFM

1. Extract and match feature between nearby frames



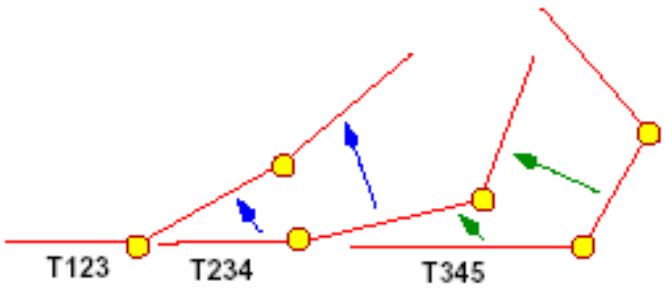
# Hierarchical SFM

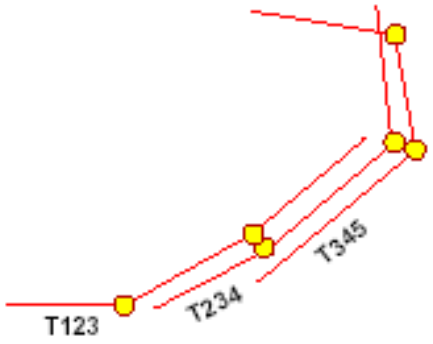
1. Extract and match feature between nearby frames
2. Identify clusters consisting of 3 nearby frames:
3. Compute SFM for 3 views
    1. Compute SFM for 2 views
    2. Add 3rd view:
        1. SFM between 1 and 2
        2. SFM between 2 and 3
        3. Then merge 1-2 with 2-3



# Hierarchical SFM

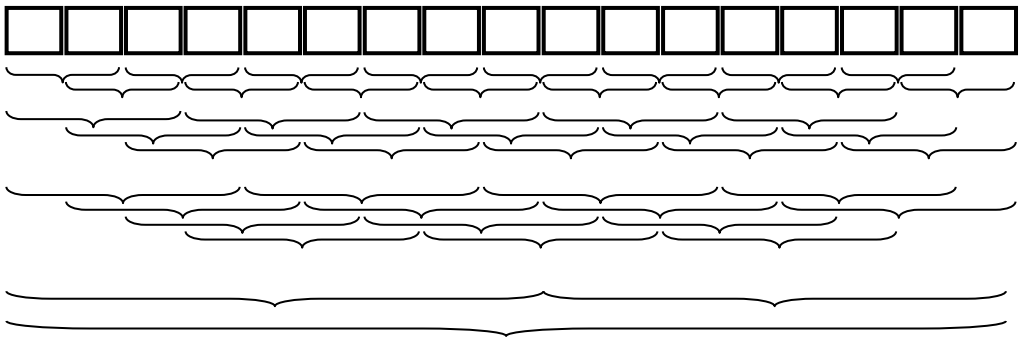
1. Extract and match feature between nearby frames
2. Identify clusters consisting of 3 nearby frames:
3. Compute SFM for 3 views
    1. Compute SFM for 2 views
    2. Add 3rd view:
        1. SFM between 1 and 2
        2. SFM between 2 and 3
        3. Then merge 1-2 with 2-3
4. Merge clusters pairwise and refine (BA) both structure and motion



# Hierarchical SFM

1. Extract and match feature between nearby frames
2. Identify clusters consisting of 3 nearby frames:
3. Compute SFM for 3 views
    1. Compute SFM for 2 views
    2. Add 3rd view:
        1. SFM between 1 and 2
        2. SFM between 2 and 3
        3. Then merge 1-2 with 2-3
4. Merge clusters pairwise and refine (BA) both structure and motion



# Hierarchical SFM: Example

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

# Structure From Motion with $n$ Views

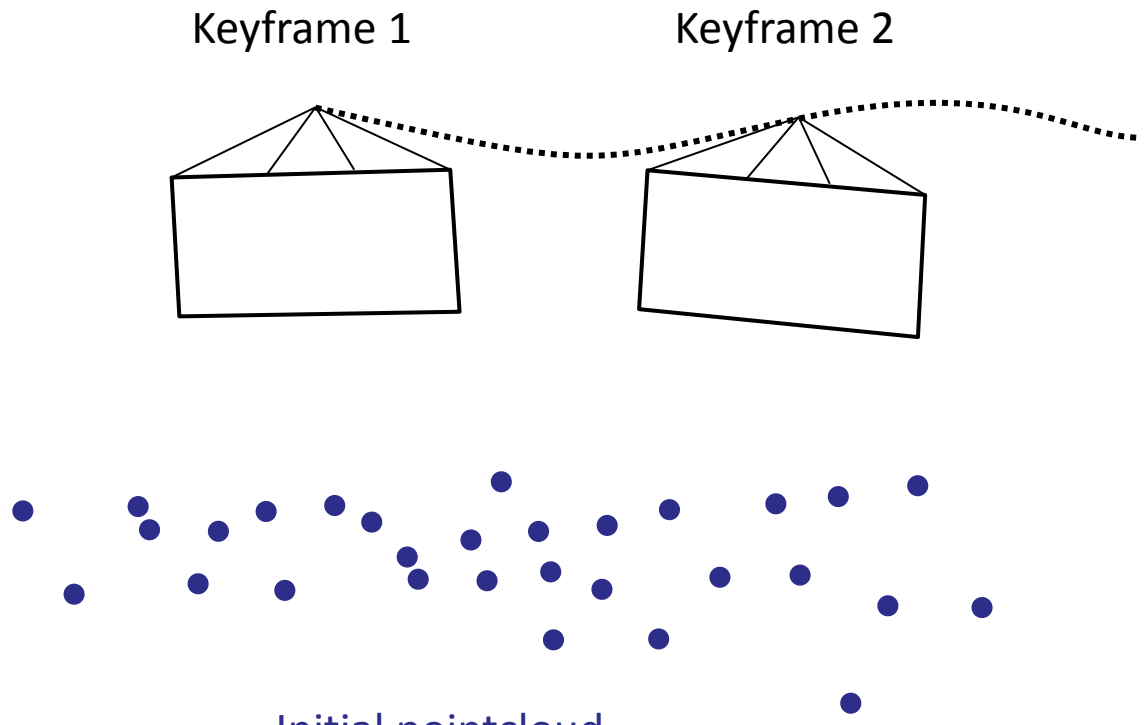
- Compute initial structure and motion
  - **Hierarchical SFM**
  - **Sequential SFM**
- Refine simultaneously structure and motion through BA

# Sequential SFM - also called Visual Odometry (VO)

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

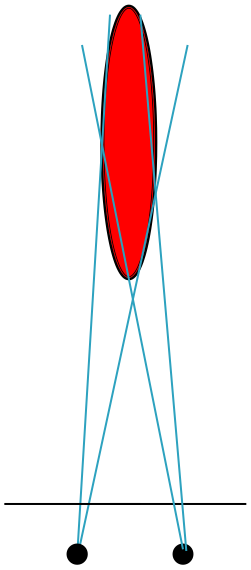
# Monocular VO (i.e., with a single camera)

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

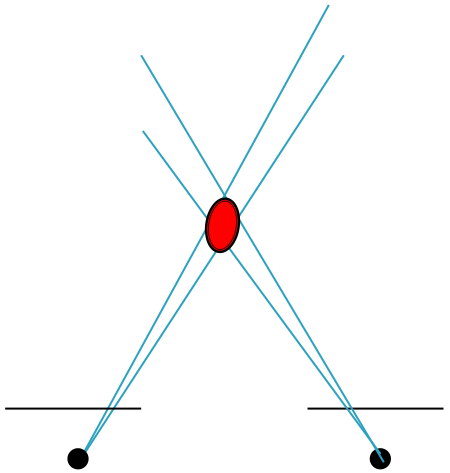


# Skipping frames (Keyframe Selection)

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



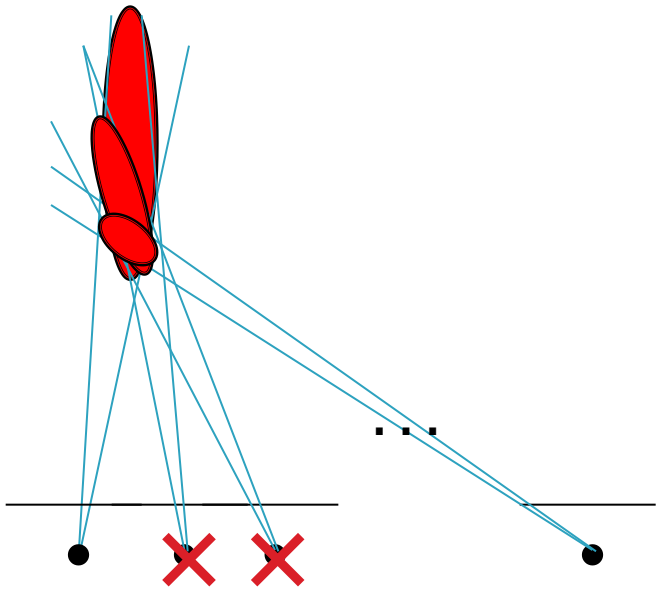
Small baseline → large depth uncertainty



Large baseline → small depth uncertainty

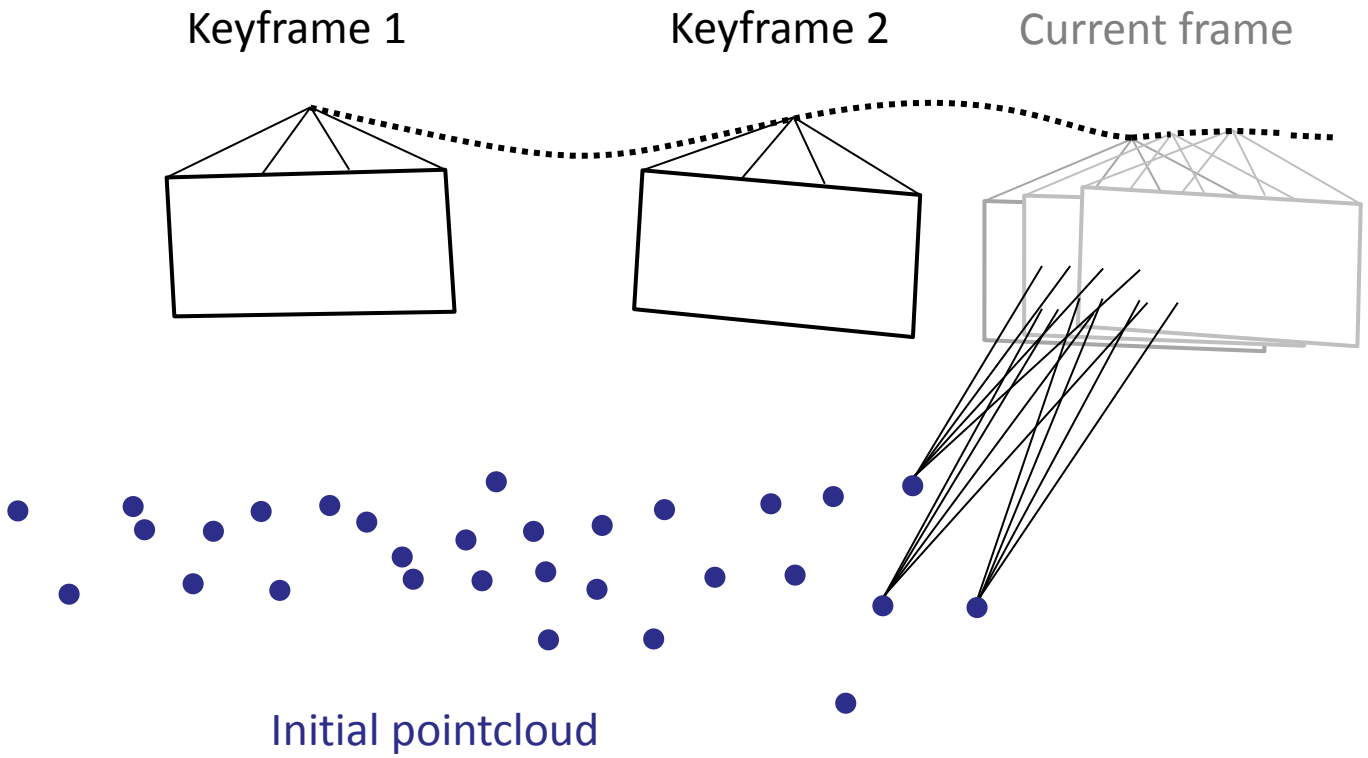
# Skipping frames (Keyframe Selection)

- When frames are taken at nearby positions compared to the scene distance, 3D points will exibit large uncertainty
- One way to avoid this consists of **skipping frames** until the average uncertainty of the 3D points decreases below a certain threshold. The selected frames are called ***keyframes***
- **Rule of the thumb:** add a keyframe when $\frac{\textit{keyframe distance}}{\textit{average-depth}} > \textit{threshold (~10-20 \%)}$



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

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



# Localization

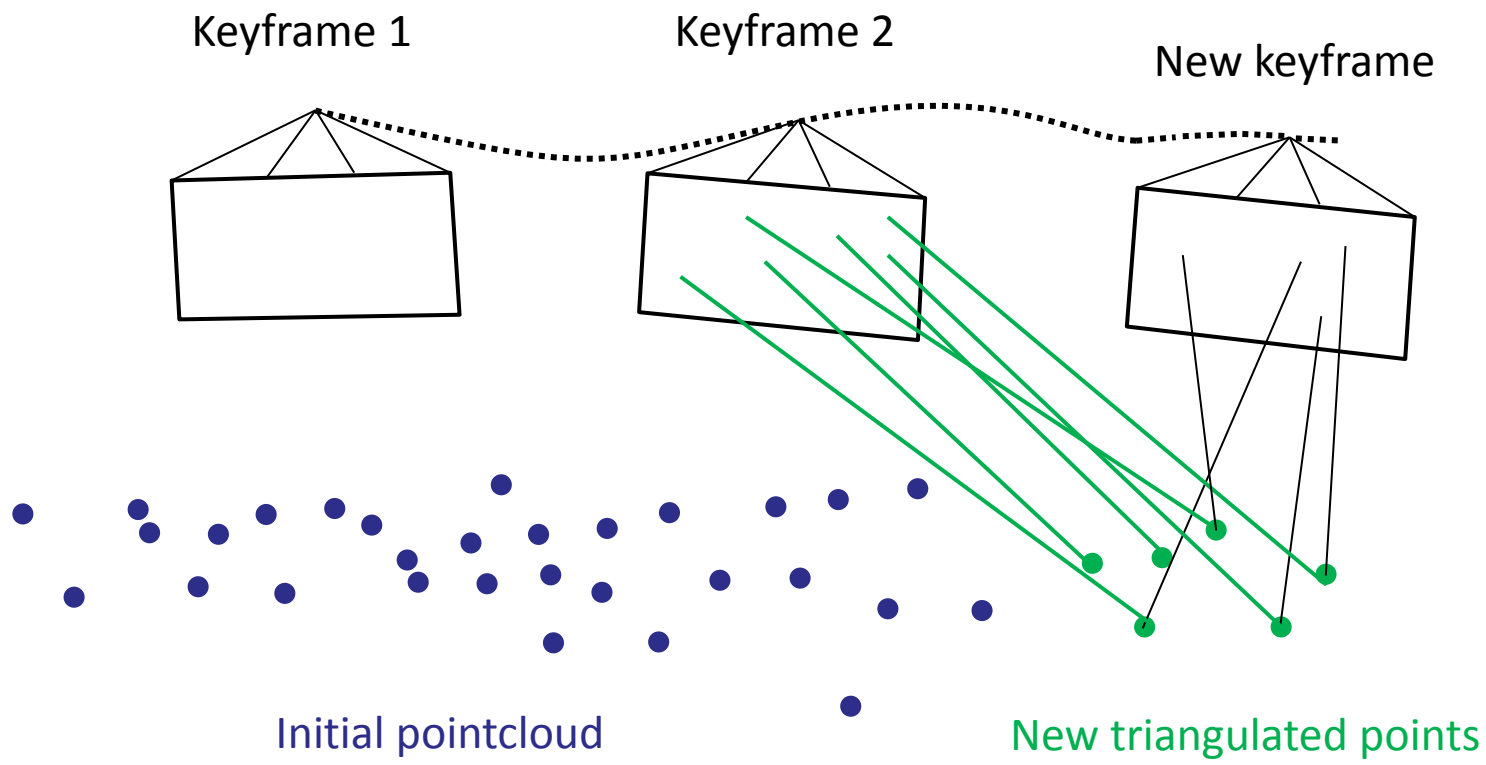
- Compute camera pose from known 3D-to-2D feature correspondences
  - Extract correspondences (how?)
  - Solve for $R$ and $t$ ($K$ is known)

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

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

# Extend Structure

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



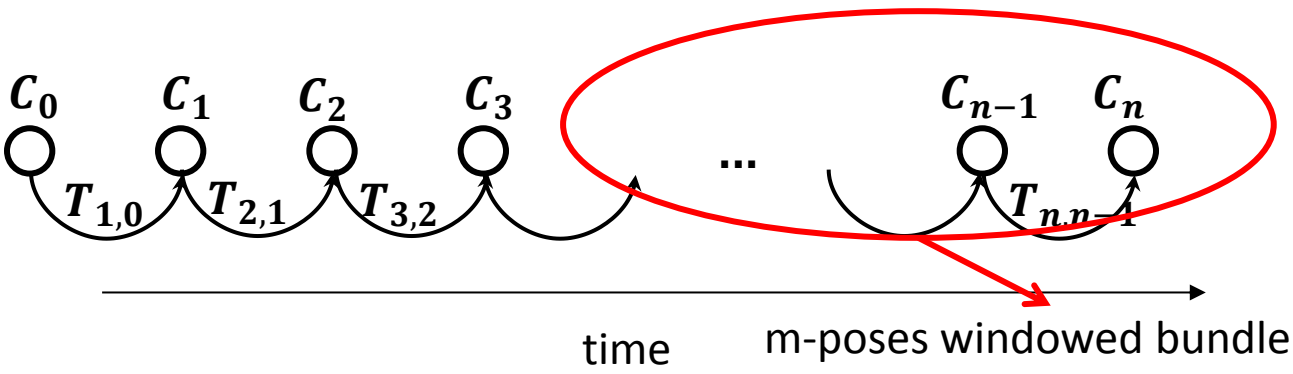
# Monocular Visual Odometry: putting all pieces together

- After bootstrapping, determine the pose $T_k$ of each additional view (keyframe) by Localization (feature matching and P3P or DLT + RANSAC)

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

- By concatenation of all these single movements, the full trajectory of the camera can be recovered , i.e.: $C_k = T_{k,k-1} C_{k-1}$

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



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

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

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

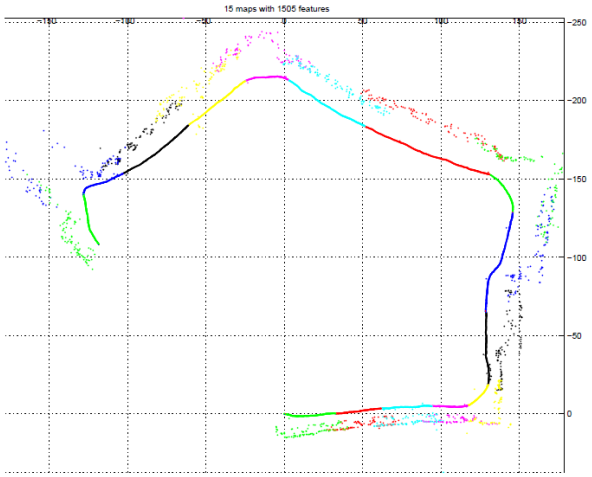
- **Loop closing problem**
  - When you go back to a previously mapped area:
    - **Loop detection**: to avoid map duplication
    - **Loop correction**: to compensate the accumulated drift
  - In both cases you need a place recognition technique

We will address place recognition in a later lecture

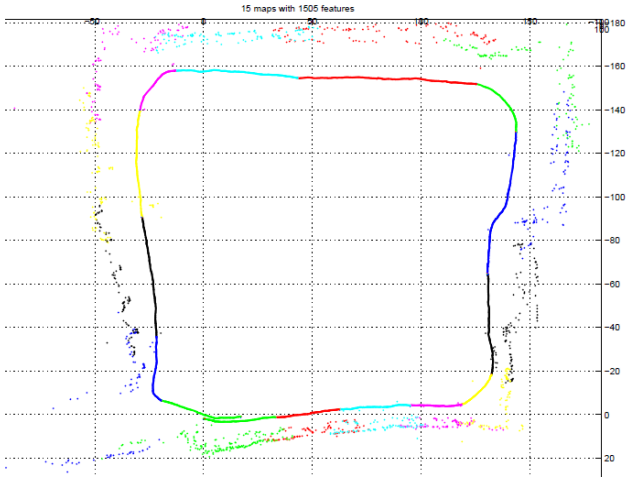
# VO vs. Visual SLAM

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

- VO **trades off consistency for real-time performance**, without the need to keep track of all the previous history of the camera.




**Visual odometry**



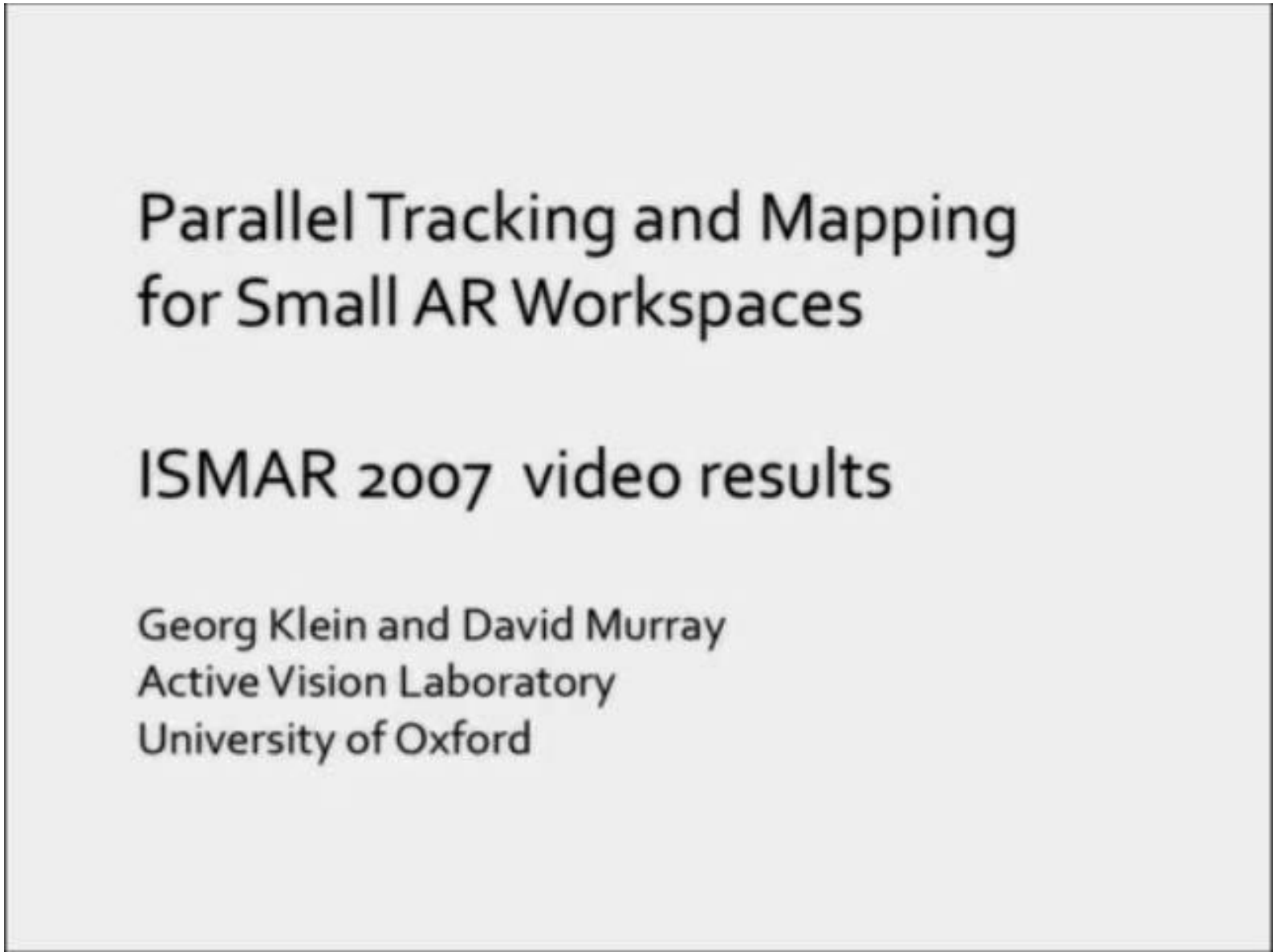

**Visual SLAM**

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

# Open Source Monocular VO and SLAM algorithms

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

# PTAM: Parallel Tracking and Mapping for Small AR Workspaces



# ORB-SLAM: Large-scale Feature-based SLAM

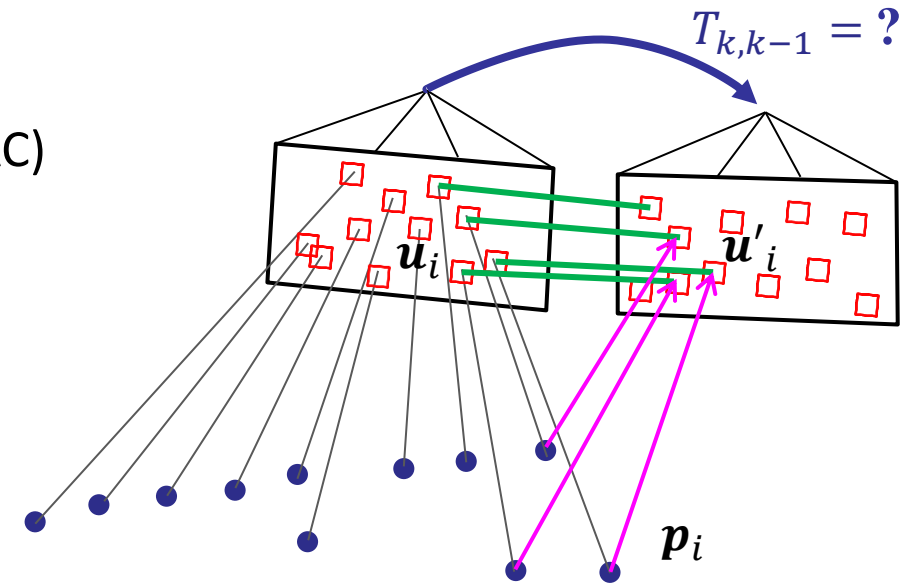
- Monocular Visual Odometry plus:
  - **Loop closing**
  - **Relocalization** (DBoW)
  - Final optimization (BA)

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

- Real-time (30Hz)

# Feature-based methods

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



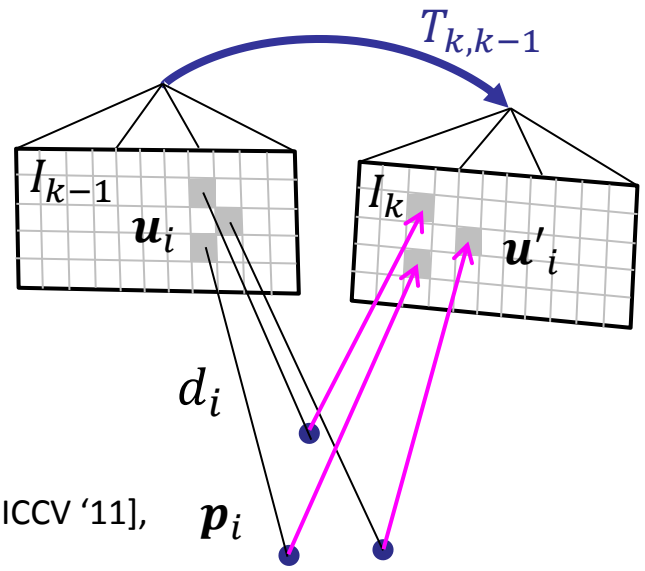

$$T_{k,k-1} = \arg\min_T \sum_i \|\boldsymbol{u}'_i - \pi(\boldsymbol{p}_i)\|^2_\Sigma$$

# Direct methods

1. Minimize **photometric error**

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

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




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

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

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

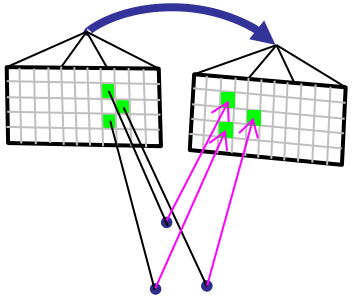
- ✓ All information in the image can be exploited (precision, robustness)
- ✓ Increasing camera frame-rate reduces computational cost per frame
- × Limited frame-to-frame motion
- × Joint optimization of dense structure and motion too expensive

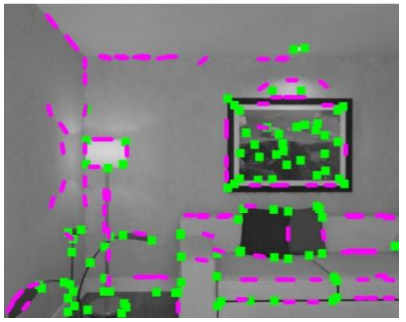
# LSD-SLAM: Large-scale Semi-Dense SLAM

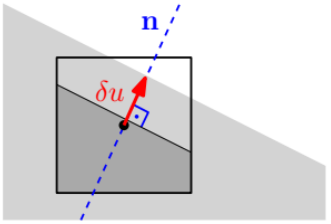
- Monocular Visual Odometry plus:
  - **Loop closing**
  - **Relocalization** (DBoW)
  - Final optimization (BA)
- **Semi-dense**: tracks all pixels on strong edges
- **Direct:** minimizes photometric error
- Real-time (30Hz)

# SVO: Semi-direct Visual Odometry

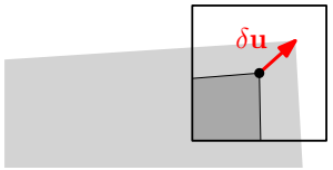
- Monocular Visual Odometry **only**:
  - No loop closing
- **Sparse**: corners and edgelets
- **Semi-Direct**
  - **Direct**: for Frame-to-Frame motion estimation
  - **Features**: for Frame-to-Keyframe pose refinement
- **Mapping**
  - **Probabilistic depth estimation**
    - depth uncertainty and
    - inlier probability)
- **Super fast** (Hz)







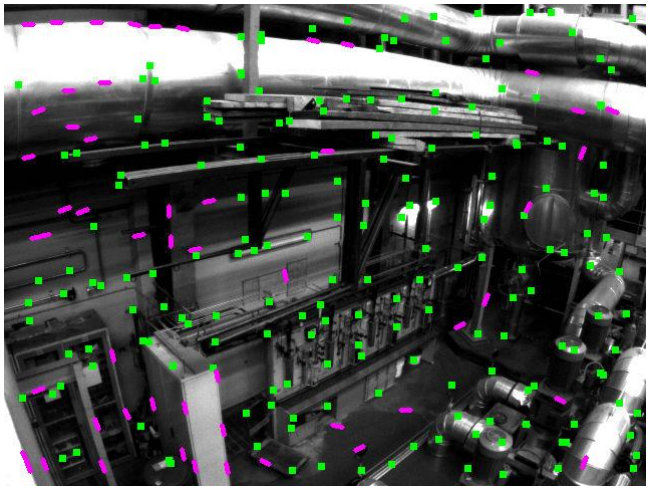

**Edgelet**

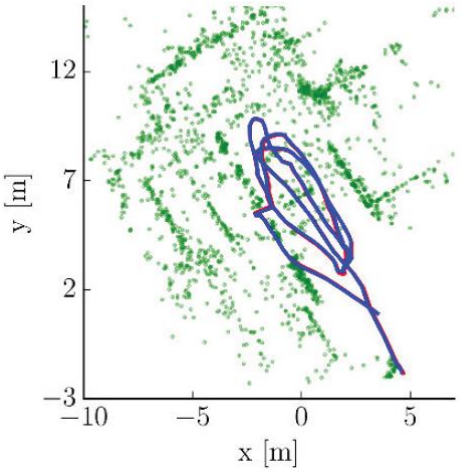



**Corner**

**SVO with a single camera** on Euroc dataset

# Accuracy and Timing






Intel i7, 2.80 GHz

| | Euroc RMS Error | Timing | CPU @ 20 fps |
|---|---|---|---|
| SVO | **0.07 m** | **5.25 ms** | **72 %** |
| ORB SLAM | 0.19 m | 29.81 ms | 187 % |
| LSD SLAM | 0.43 m | 23.23 ms | 236 % |

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

# Processing Times of SVO

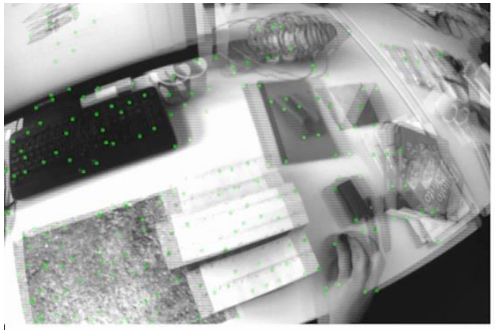
Laptop (Intel i7, 2.8 GHz)

400 frames per second

Embedded ARM Cortex-A9, 1.7 GHz

Up to 70 frames per second

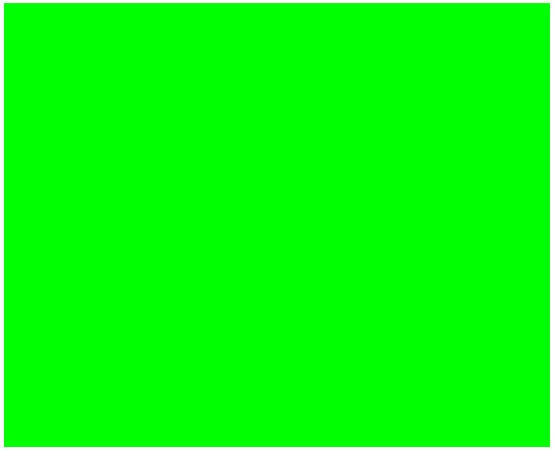
Why so fast?



Realtime
Camera at 70fps

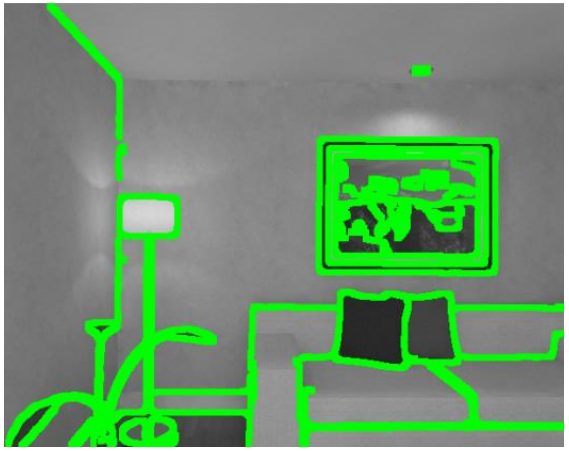
# Dense vs Semi-Dense vs Sparse

## Dense



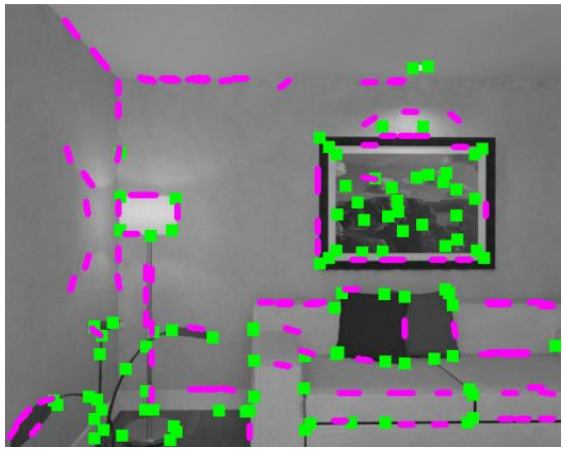
DTAM [Newcombe et al. ‘11]
300’000+ pixels

## Semi-Dense



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

## Sparse

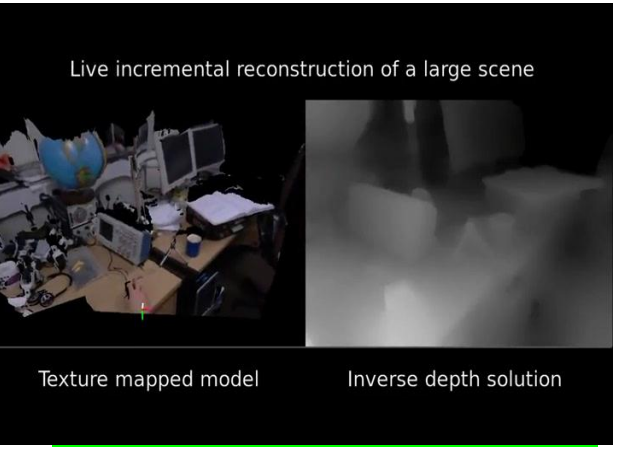


SVO [Forster et al. 2014, TRO’16]
100-200 features x 4x4 patch
~ 2,000 pixels

We will see this in the next lecture

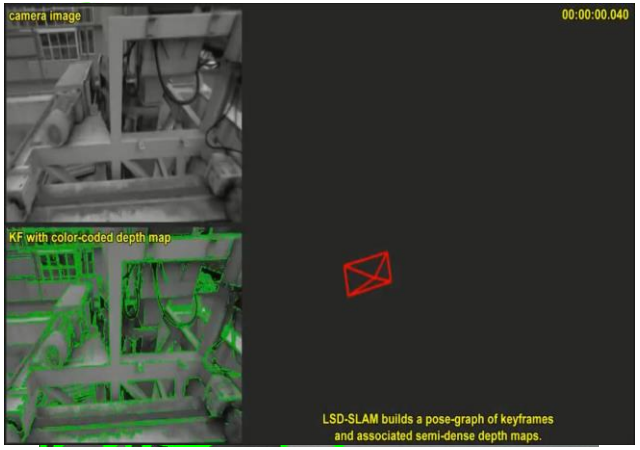
# Dense vs Semi-Dense vs Sparse

## Dense



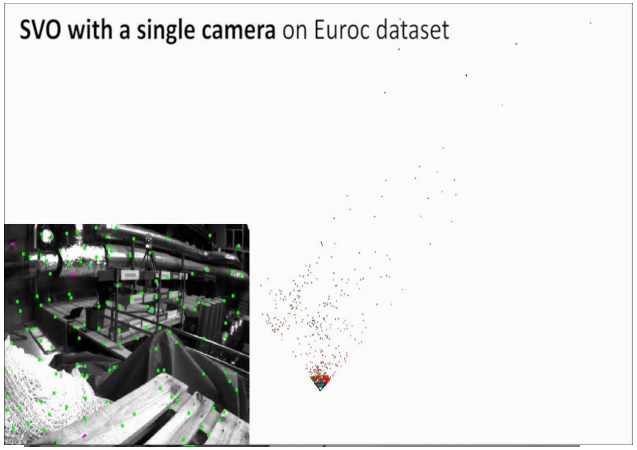

DTAM [Newcombe et al. ‘11]
300,000+ pixels

## 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

We will see this in the next lecture

# SVO for Autonomous Drone Navigation

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

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

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**.

# Tech Transfer activities

# Application: Autonomous Inspection of Bridges and Power Masts

Project with Parrot: Autonomous vision-based navigation


Parrot senseFly

Albris drone

# Dacuda VR solutions

Dacuda

- Fully immersive virtual reality with 6-DoF for VR and AR content (running on iPhone)
- Powered by SVO

# Spinoff: Zurich-Eye – *www.zurich-eye.com*

Vision-based Localization and Mapping Solutions for Mobile Robots

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