Lecture 09
Multiple View Geometry 3

Davide Scaramuzza
Lab Exercise 6 - Today

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

• Bundle Adjustment
• SFM with $n$ views
Bundle Adjustment (BA)

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

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

**NB:** here, by \( C_1, C_2 \) we denote the **pose of** each camera in the **world** frame
- Can be minimized using **Levenberg–Marquardt** (more robust than Gauss-Newton to local minima)
- In order to not get stuck in local minima, the **initialization should be close the minimum**
Bundle Adjustment (BA) for $n$ Views

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

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

• Bundle Adjustment
• SFM with $n$ views
Structure From Motion with $n$ Views

- Compute initial structure and motion
  - Hierarchical SFM
  - Sequential SFM

- Refine simultaneously structure and motion through BA
Hierarchical SFM

1. Extract and match features between nearby frames
Hierarchical SFM

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

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

How do you merge clusters?
Hierarchical SFM: Example

- Reconstruction from 150,000 images from Flickr.com associated with the tags “Rome” and “Roma”
- Cloud of 496 computers, 21 hours of computation!
Structure From Motion with \( n \) Views

- Compute initial structure and motion
  - Hierarchical SFM
  - Sequential SFM

- Refine simultaneously structure and motion through BA
Sequential SFM - also called Visual Odometry (VO)

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

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

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

- **1980 to 2000**: The VO research was dominated by **NASA/JPL** in preparation of the 2004 mission to Mars.

- **2004**: VO was used on a robot on another planet: Mars rovers Spirit and Opportunity (see seminal paper from **NASA/JPL, 2007**).

- **2004**: VO was revived in the academic environment by David Nister’s «Visual Odometry» paper. The term VO became popular.
More about history and tutorials


VO vs VSLAM vs SFM
Structure from Motion (SFM)

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

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

- VO is a particular case of SFM

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

- Terminology: sometimes SFM is used as a synonym of VO
VO vs. Visual SLAM

- **Visual Odometry**
  - Focus on incremental estimation/local consistency

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

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

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

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

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

- Popular algorithms: P3P [Gao’03, Kneip’11]
3D-to-3D

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

- Both $f_{k-1}$ and $f_k$ are specified in 3D. To do this, it is necessary to triangulate 3D points (e.g. use a stereo camera)
- The minimal-case solution involves 3 non-collinear correspondences
- The solution is found by minimizing the 3D-3D Euclidean distance:

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

- Popular algorithm: [Arun’87] for global registration, ICP for local refinement or Bundle Adjustment (BA)
Case Study:
Monocular Visual Odometry
Monocular VO (i.e., with a single camera)

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

![Diagram of keyframes and initial pointcloud]
Skipping frames (Keyframe Selection)

- When frames are taken at nearby positions compared to the scene distance, 3D points will exhibit 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 exhibit 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 \( \frac{\text{keyframe distance}}{\text{average-depth}} > \text{threshold (\~10-20 \%)} \)
Monocular VO (i.e., with a single camera)

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

![Diagram showing localization process in monocular VO with keyframes and initial pointcloud.](image-url)
Localization

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

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

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

- Extract and triangulate new features
  - Is it necessary to do this for every frame or can we just do it for keyframes?
  - What are the pros and cons?
Monocular Visual Odometry: putting all pieces together

- We denote the relative motion between adjacent keyframes:
  \[ T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \]

- By concatenation of all these transformations, the full trajectory of the camera can be recovered:
  \[ C_k = T_{k,k-1}C_{k-1} \]

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

\[ C_0 \quad C_1 \quad C_2 \quad C_3 \quad \cdots \quad C_{n-1} \quad C_n \]

\[ T_{1,0} \quad T_{2,1} \quad T_{3,2} \quad \cdots \quad T_{n,n-1} \]

Sliding-window bundle adjustment
Loophole Closure Detection (i.e., Place Recognition)

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

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

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

We will address place recognition in Lecture 12
Recall: VO vs. Visual SLAM

- Visual SLAM = visual odometry + loop detection + graph optimization
Open Source Monocular VO and SLAM algorithms

- **PTAM** [Klein, 2007] -> Oxford, Murray’s lab
- **ORB-SLAM** [Mur-Artal, T-RO, 15] -> Zaragoza, Tardos’ lab
- **LSD-SLAM** [Engel, ECCV’14] -> Munich, Cremers’ lab
- **DSO** [Engel’16] -> Munich, Cremers’ lab
- **SVO** [Forster, ICRA’14, TRO’17] -> Zurich, Scaramuzza’s lab
PTAM: Parallel Tracking and Mapping for Small AR Workspaces

Parallel Tracking and Mapping for Small AR Workspaces

ISMAR 2007 video results

Georg Klein and David Murray
Active Vision Laboratory
University of Oxford
Feature based
- ORB feature = FAST corner + Oriented Rotated Brief descriptor
- Binary descriptor
- Very fast to compute and compare
- Minimizes reprojection error

Includes:
- Loop closing
- Relocalization
- Final optimization

Real-time (30Hz)
Feature-based methods

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

\[ T_{k,k-1} = \arg \min_T \sum \|u'_i - \pi(p_i)\|_2^2 \]

Direct methods

1. Minimize photometric error

\[ T_{k,k-1} = \arg \min_T \sum \|I_k(u'_i) - I_{k-1}(u_i)\|_2^2 \]

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

[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

\[ T_{k,k-1} = \arg \min_T \sum_i \| \mathbf{u}'_i - \pi(\mathbf{p}_i) \|_2^2 \]

Direct methods

1. Minimize photometric error

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

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

✓ 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)

✓ 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], [Newcombe [Engel et al., ECCV’14], [Forster et al., ICRA’14]
LSD-SLAM [Engel, ECCV’14]

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

- Includes:
  - Loop closing
  - Relocalization
  - Final optimization

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

Download from [https://vision.in.tum.de/research/vslam/lsdslam](https://vision.in.tum.de/research/vslam/lsdslam)

[Engel, Schoeps, Cremers, LSD-SLAM: Large-scale Semi-Dense SLAM, ECCV’14]
**DSO** [Engel, PAMI’17]

- **Direct** (photometric error) + **Sparse** formulation
  - 3D geometry represented as sparse large gradients
  - Minimizes **photometric error**
  - **Jointly** optimizes poses & structure (sliding window)
  - Incorporate photometric correction to compensate exposure time change

\[
E_{pj} := \sum_{p \in N_p} w_p \left\| (I_j[p'] - b_j) - \frac{t_j e_{aj}}{t_i e_{ai}} (I_i[p] - b_i) \right\|_\gamma
\]

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

Download from [https://vision.in.tum.de/research/vslam/dso](https://vision.in.tum.de/research/vslam/dso)

[Engel, Koltun, Cremers, DSO: Direct Sparse Odometry, PAMI’17]
SVO [Forster, ICRA’14, TRO’17]

- **Direct** (minimizes photometric error)
  - Corners and edgelets
  - Frame-to-frame motion estimation

- **Feature-based** (minimizes reprojection error)
  - Frame-to-Keyframe pose refinement

- **Mapping**
  - Probabilistic depth estimation

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

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

Download from [http://rpg.ifi.uzh.ch/svo2.html](http://rpg.ifi.uzh.ch/svo2.html)
Comparison SVO, DSO, ORB-SLAM, LSD-SLAM [Forster, TRO’17]

Comparison is done on public benchmarks:

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

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].

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

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

**TABLE II**: The first and second column report mean and standard deviation 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.

Processing Times of SVO

• **Laptop** (Intel i7, 2.8 GHz): up to 400 fps

• **Smartphone**, ARM Cortex-A9, 1.7 GHz (Odroid): Up to 100 fps

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

<table>
<thead>
<tr>
<th>Task</th>
<th>Thread</th>
<th>Intel i7 [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse image alignment</td>
<td>1</td>
<td>0.66</td>
</tr>
<tr>
<td>Feature alignment</td>
<td>1</td>
<td>1.04</td>
</tr>
<tr>
<td>Optimize pose &amp; landmarks</td>
<td>1</td>
<td>0.42</td>
</tr>
<tr>
<td>Extract features</td>
<td>2</td>
<td>1.64</td>
</tr>
<tr>
<td>Update depth filters</td>
<td>2</td>
<td>1.80</td>
</tr>
</tbody>
</table>

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

Dense

Live incremental reconstruction of a large scene

Texture mapped model

Inverse depth solution

Semi-Dense

LSD-SLAM [Engel’14]

~10,000 pixels

Inverse depth solution

LSD builds a pose-graph of keyframes and associated semi-dense depth maps

Sparse

SVO [Forster’14]

100-200 x 4x4 patches ≈ 2,000 pixels

SVO with a single camera on Euroc dataset

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

300’000+ pixels

LSD-SLAM [Engel’14]

~10,000 pixels

[SVO: Semi Direct Visual Odometry for Monocular and Multi-Camera Systems, TRO’17]
Direct Methods: Dense vs Semi-dense vs Sparse [TRO’16]

Dense

Semi-Dense

Sparse

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

LSD-SLAM [Engel’14]
~10,000 pixels

SVO [Forster’14]
100-200 x 4x4 patches \(\approx\) 2,000 pixels

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

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

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

- Multi-FOV Zurich Urban Dataset: [http://rpg.ifi.uzh.ch/fov.html](http://rpg.ifi.uzh.ch/fov.html)

Images from the synthetic Multi-FOV Zurich Urban Dataset
Position error: 5 mm, height: 1.5 m – Down-looking camera

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

Robustness to dynamic scenes (down-looking camera)

Automatic recovery from aggressive flight [ICRA’15]

[ICRA’10-17, AURO’12, RAM’14, JFR’15, RAL’17]
Tech Transfer activities
Parrot: Autonomous Inspection of Bridges and Power Masts

Albris drone

Automated take off, self-check & calibration
Dacuda 3D (now Magic Leap)

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

Dacuda's 3D division
Zurich-Eye, first Wyss Zurich project

Vision-based Localization and Mapping Solutions for Mobile Robots
Created in Sep. 2015, became Facebook-Oculus Zurich in Sep. 2016

The Zurich Eye team is behind the new Oculus Santa Cruz
Zurich-Eye, first Wyss Zurich project

Vision-based Localization and Mapping Solutions for Mobile Robots

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

The Zurich Eye team is behind the new Oculus Santa Cruz