Computer Vision 2
WS 2018/19

Part 12 – Visual Odometry
04.12.2018

Prof. Dr. Bastian Leibe

RWTH Aachen University, Computer Vision Group
http://www.vision.rwth-aachen.de
Course Outline

• Single-Object Tracking

• Bayesian Filtering

• Multi-Object Tracking
  – Introduction
  – MHT, (JPDAF)
  – Network Flow Optimization

• Visual Odometry
  – Sparse interest-point based methods
  – Dense direct methods

• Visual SLAM & 3D Reconstruction

• Deep Learning for Video Analysis

image source: [Zhang, Li, Nevatia, CVPR’08]
Topics of This Lecture

• **Visual Odometry**
  – Definition, Motivation

• **Geometry Background**
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• **Point-based Visual Odometry**
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
Recap: What is Visual Odometry?

Visual odometry (VO)...

• … is a variant of tracking
  – Track motion (position and orientation) of the camera from its images
  – Only considers a limited set of recent images for real-time constraints

• … also involves a data association problem
  – Motion is estimated from corresponding interest points or pixels in images, or by correspondences towards a local 3D reconstruction
Recap: What is Visual Odometry?

Visual odometry (VO)…

• … is prone to drift due to its local view

• … is primarily concerned with estimating camera motion
  – Not all approaches estimate a 3D reconstruction of the associated interest points/ pixels explicitly.
  – If so it is only locally consistent

Slide credit: Jörg Stückler

Image source: [Clemente et al., RSS 2007]
Visual Odometry Example

SVO: Fast Semi-Direct Monocular Visual Odometry

Christian Forster, Matia Pizzoli, Davide Scaramuzza

rpg.ifi.uzh.ch

University of Zurich
Department of Informatics

Slide credit: Jörg Stückler
Visual Odometry Term

• **Odometry**
  – Greek: „hodos“ – path, „metron“ – measurement
  – Motion or position estimation from measurements or controls
  – Typical example: wheel encoders

• **Visual Odometry**
  – David Nister‘s „Visual Odometry“ paper from 2004 about keypoint-based methods for monocular and stereo cameras
Why Visual Odometry?

- VO is often used to complement other motion sensors
  - GPS
  - Inertial Measurement Units (IMUs)
  - Wheel odometry
  - etc.

- VO is much more accurate than wheel odometry and not prone to wheel slippage.

- VO is important in GPS-denied environments (indoors, close to buildings, etc.)
Sensor Types for Visual Odometry

• Monocular cameras
  – Pros: Low-power, light-weight, low-cost, simple to calibrate and use
  – Cons: requires motion parallax and texture, scale not observable

• Stereo cameras
  – Pros: depth without motion, less power than active structured light
  – Cons: requires texture, accuracy depends on baseline, synchronization and extrinsic calibration of the cameras

• Active RGB-D sensors
  – Pros: no texture needed, similar to stereo processing
  – Cons: active sensing consumes power, blackbox depth estimation

Image source: IDS, PointGrey, ASUS
Topics of This Lecture

• Visual Odometry
  – Definition, Motivation

• **Geometry Background**
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• Point-based Visual Odometry
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
A Note about Notation

• This course material originated from the 2016 CV 2 lecture held together with Jörg Stückler (now Prof. @ MPI Tübingen)
  – The notation follows the MASKS textbook and is slightly different from the notation used in the CV 1 lecture.
  – We’ll stick with this notation in order to be consistent with the later lectures
  – In case you get confused by notation, please interrupt me and ask…

An Invitation to 3D Vision, Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, Springer, 2004
Geometric Point Primitives

- **Point**
  \[ \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2 \]
  \[ \mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \]

- **Augmented vector**
  \[ \mathbf{\bar{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{R}^3 \]
  \[ \mathbf{\bar{x}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4 \]

- **Homogeneous coordinates**
  \[ \mathbf{\tilde{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^2 \]
  \[ \mathbf{\tilde{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^3 \]
Euclidean Transformations

• Euclidean transformations apply rotation and translation

\[ x' = Rx + t \]

\[ \overline{x}' = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \overline{x} \]

• Rigid-body motion: preserves distances and angles when applied to points on a body

\[ n = 2 \]

\[ n = 3 \]

Slide credit: Jörg Stückler
Special Orthogonal Group SO(n)

- Rotation matrices have a special structure
  \[ R \in SO(n) \subset \mathbb{R}^{n \times n}, \det(R) = 1, RR^T = I \]
  i.e. orthonormal matrices that preserve distance and orientation

- They form a group denoted as Special Orthogonal Group SO(n)
  - The group operator is matrix multiplication – associative, but not commutative!
  - Inverse and neutral element exist

- 2D rotations only have 1 degree of freedom (DoF), i.e. angle of rotation
- 3D rotations have 3 DoFs, several parametrizations exist such as Euler angles and quaternions
3D Rotation Representations – Matrix

• Straight-forward: Orthonormal matrix

\[
R = \begin{pmatrix}
  r_{11} & r_{12} & r_{13} \\
  r_{21} & r_{22} & r_{23} \\
  r_{31} & r_{32} & r_{33}
\end{pmatrix} \in \mathbb{R}^{3 \times 3}
\]

• Pro:
  – Easy to concatenate and invert

\[
R^A_C = R^A_B R^B_C \quad R^B_A = (R^A_B)^{-1}
\]

• Con:
  – Overparametrized (9 parameters for 3 DoF) – problematic for optimization

Slide credit: Jörg Stückler
3D Rotation Representations – Euler Angles

- **Euler Angles**: 3 consecutive rotations around coordinate axes
  Example: roll-pitch-yaw angles $\alpha, \beta, \gamma$ (X-Y-Z):

  $$R_{XYZ}(\alpha, \beta, \gamma) = R_Z(\gamma) R_Y(\beta) R_X(\alpha)$$

  with
  - $R_X(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$
  - $R_Y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$
  - $R_Z(\gamma) = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

- 12 possible orderings of rotation axes (f.e. Z-X-Z)
- **Pro**: Minimal with 3 parameters
- **Con**: Singularities (gimbal lock), concatenation/inversion via conversion from/to matrix
  - 1 DoF lost!

Slide credit: Jörg Stückler
3D Rotation Representations – Axis-Angle

• **Axis-Angle**: Rotate along axis $\mathbf{n} \in \mathbb{R}^3$ by angle $\theta \in \mathbb{R}$:

$$\mathbf{R}(\mathbf{n}, \theta) = \mathbf{I} + \sin(\theta)\hat{\mathbf{n}} + (1 - \cos(\theta))\hat{\mathbf{n}}^2 \quad \|\mathbf{n}\|_2 = 1$$

where

$$\hat{\mathbf{x}} := \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \quad \hat{\mathbf{x}} \mathbf{y} = \mathbf{x} \times \mathbf{y}$$

• Reverse: $\theta = \cos^{-1}\left(\frac{\text{tr}(\mathbf{R}) - 1}{2}\right)$

$$\mathbf{n} = \frac{1}{2 \sin(\theta)} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

• 4 parameters: $(\mathbf{n}, \theta)$

• 3 parameters: $\omega = \theta \mathbf{n}$

• Pro: minimal representation for 3 parameters

• Con: $(\mathbf{n}, \theta)$ has unit norm constraint on $\mathbf{n}$ - problematic for optimization; both parametrizations not unique; concatenation/inversion via $\text{SO}(3)$

Slide credit: Jörg Stückler
3D Rotation Representations – Quaternions

- **Unit Quaternions**: \[ q = (q_x, q_y, q_z, q_w)^\top \in \mathbb{R}^4, \quad \|q\|_2 = 1 \]

- Relation to axis-angle representation:
  - **Axis-angle to quaternion**:
    \[
    q(n, \theta) = \left( n^\top \sin \left( \frac{\theta}{2} \right), \cos \left( \frac{\theta}{2} \right) \right)
    \]
  - **Quaternion to axis-angle**:
    \[
    n(q) = \begin{cases} 
    (q_x, q_y, q_z)^\top / \sin(\theta/2), & \theta \neq 0 \\
    0, & \theta = 0
    \end{cases}
    \]
    \[
    \theta = 2 \arccos(q_w)
    \]
3D Rotation Representations – Quaternions cont.

• Pros:
  – Unique up to opposing sign \( q = -q \)
  – Direct rotation of a point:
    \[
    p' = R p = q(R)pq(R)^{-1}
    \]
  – Direct concatenation of rotations:
    \[
    q(R_2R_1) = q(R_2)q(R_1)
    \]
  – Direct inversion of a rotation:
    \[
    q(R^{-1}) = q(R)^{-1}
    \]
    with \( q^{-1} = (-q_{xyz}, q_w)^\top \),
    \[
    q_1q_2 = \left(q_1,wq_2,xyz + q_2,wq_1,xyz + q_1,xyz \times q_2,xyz, q_1,wq_2,w - q_1,xyzq_2,xyz\right)
    \]

• Con: Normalization constraint is problematic for optimization
Special Euclidean Group SE(3)

• Euclidean transformation matrices have a special structure as well:

\[ T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \text{SE}(3) \subset \mathbb{R}^{4\times4} \]

  – Translation \( t \) has 3 degrees of freedom
  – Rotation \( R \in \text{SO}(3) \) has 3 degrees of freedom

• They also form a group which we call \( \text{SE}(3) \). The group operator is matrix multiplication:

\[ \cdot : \text{SE}(3) \times \text{SE}(3) \rightarrow \text{SE}(3) \]

\[ T^A_B \cdot T^B_C \mapsto T^A_C \]
Definition of Visual Odometry

- Visual odometry is the process of estimating the egomotion of an object using only inputs from visual sensors on the object.

- **Inputs:** images at discrete time steps $t$,
  - Monocular case: Set of images $I_{0:t} = \{I_0, \ldots, I_t\}$
  - Stereo case: Left/right images $I_{0:t}^l = \{I_0^l, \ldots, I_t^l\}$, $I_{0:t}^r = \{I_0^r, \ldots, I_t^r\}$
  - RGB-D case: Color/depth images $I_{0:t} = \{I_0, \ldots, I_t\}$, $Z_{0:t} = \{Z_0, \ldots, Z_t\}$

- **Output:** relative transformation estimates $T_{t-1}^t \in SE(3)$ between frames.

Conventions:
- Let $T_t \in SE(3)$ be the camera pose at time $t$ in the world frame.
- $T_{t-1}^t$ transforms points from camera frame at time $t$ to $t-1$, i.e.

$$T_t = T_0 T_1^0 \cdots T_{t-1}^t$$
Direct vs. Indirect Methods

- **Direct methods**
  - formulate alignment objective in terms of photometric error (e.g. intensities)

\[
p(I_2 \mid I_1, \xi) \quad \Rightarrow \quad E(\xi) = \int_{u \in \Omega} |I_1(u) - I_2(\omega(u, \xi))| \, du
\]

- **Indirect methods**
  - formulate alignment objective in terms of reprojection error of geometric primitives (e.g. points, lines)

\[
p(Y_2 \mid Y_1, \xi) \quad \Rightarrow \quad E(\xi) = \sum_i |y_{1,i} - \omega(y_{2,i}, \xi)|
\]
Topics of This Lecture

• Visual Odometry
  – Definition, Motivation

• Geometry Background
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• Point-based Visual Odometry
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
Point-based (Indirect) Visual Odometry Example

LibVISO2, Geiger et al., StereoScan: Dense 3D Reconstruction in Real-time, IV 2011
Point-based Visual Odometry Pipeline

- Keypoint detection and local description (CV I)
- Robust keypoint matching (CV I)
- Motion estimation
  - 2D-to-2D: motion from 2D point correspondences
  - 2D-to-3D: motion from 2D points to local 3D map
  - 3D-to-3D: motion from 3D point correspondences (e.g., stereo, RGB-D)

Images from Jakob Engel
Slide credit: Jörg Stückler
Topics of This Lecture

• Visual Odometry
  – Definition, Motivation

• Geometry Background
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• Point-based Visual Odometry
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
Recap: Pinhole Projection Camera Model

world coordinates
image pixel coordinates
focal length
camera center

\[
\begin{pmatrix}
\bar{x}^c \\
\bar{y}^c \\
1
\end{pmatrix}
= \begin{pmatrix}
\bar{x}^p \\
\bar{y}^p \\
1
\end{pmatrix} = \pi(\bar{x}) = \bar{y}
\]
(normalized image coordinates)

\[
\bar{x}^p = (x^c, y^c, z^c, 1)^T
\]
\[
\bar{y}^p = (x^p, y^p, 1)^T
\]

\[
\begin{pmatrix}
f_x & 0 & c_x \\
0 & f_y & c_y \\
0 & 0 & 1
\end{pmatrix}
\]
camera matrix \( C \)

\[
x^c / z^c
\]
\[
y^c / z^c
\]
\[
1
\]
2D-to-2D Motion Estimation

- Given corresponding image point observations

\[ Y_t = \{y_{t,1}, \ldots, y_{t,N}\} \]

\[ Y_{t-1} = \{y_{t-1,1}, \ldots, y_{t-1,N}\} \]

of unknown 3D points \( X = \{x_1, \ldots, x_N\} \)

-determine relative motion \( T_{t-1}^{t} \) between frames

- Obvious try: minimize reprojection error using least squares

\[
E \left( T_{t-1}^{t}, X \right) = \sum_{i=1}^{N} \left\| \bar{y}_{t,i} - \pi \left( \bar{x}_i \right) \right\|^2_2 + \left\| \bar{y}_{t-1,i} - \pi \left( T_{t}^{t-1} \bar{x}_i \right) \right\|^2_2
\]

- Convexity? Uniqueness (scale-ambiguity)?

- Alternative algebraic approaches: 8-point / 5-point algorithm
Recap: Epipolar Geometry

- Camera centers \( c, c' \) and image point \( y \in \Omega \) span the epipolar plane \( \Pi \)
- The ray from camera center \( c \) through point \( y \) projects as the epipolar line \( 1' \) in image plane \( \Omega' \)
- The intersections of the line through the camera centers with the image planes are called epipoles \( e, e' \)
Essential Matrix

- The rays to the 3D point and the baseline $\mathbf{t}$ are coplanar
  \[ \tilde{\mathbf{y}}^\top (\mathbf{t} \times \mathbf{R}\tilde{\mathbf{y}}') = 0 \iff \tilde{\mathbf{y}}^\top \mathbf{tR}\tilde{\mathbf{y}}' = 0 \]

- The Essential matrix $\mathbf{E} := \hat{\mathbf{t}}\mathbf{R}$ captures the relative camera pose
- Each point correspondence provides an "epipolar constraint"
- 5 correspondences suffice to determine $\mathbf{E}$
- (Simpler: 8-point algorithm)
Eight-Point Algorithm for Essential Matrix Estimation

- First proposed by Longuet and Higgins, 1981
- Algorithm:
  1. Rewrite epipolar constraints as a linear system of equations
     \[ \tilde{y}_i E \tilde{y}_i' = a_i E_s = 0 \]
     using Kronecker product
     \[ a_i = \tilde{y}_i \otimes \tilde{y}_i' \]
     and
     \[ E_s = (e_{11}, e_{12}, e_{13}, \ldots, e_{33})^T \]
  2. Apply singular value decomposition (SVD) on
     \[ A = U_A S_A V_A^T \]
     and unstack the 9th column of \( V_A \) into \( \tilde{E} \)
  3. Project the approximate \( \tilde{E} \) into the (normalized) essential space:
     Determine the SVD of
     \[ \tilde{E} = U \text{diag}(\sigma_1, \sigma_2, \sigma_3) V^T \]
     with \( U, V \in \text{SO}(3) \)
     and replace the singular values \( \sigma_1 \geq \sigma_2 \geq \sigma_3 \) with \( 1, 1, 0 \) to find
     \[ E = U \text{diag}(1, 1, 0) V^T \]
Eight-Point Algorithm cont.

• Algorithm (cont.):
  – Determine one of the following 2 possible solutions that intersects the points in front of both cameras:

\[
R = UR_Z^T \left( \pm \frac{\pi}{2} \right) V^T \\
\hat{t} = UR_Z \left( \pm \frac{\pi}{2} \right) \text{diag}(1, 1, 0) U^T
\]

• A derivation can be found in the MASKS textbook, Ch. 5

• Remarks
  – Algebraic solution does not minimize geometric error
  – Refine using non-linear least-squares of reprojection error
  – Alternative: formulate epipolar constraints as „distance from epipolar line“ and minimize this non-linear least-squares problem
Triangulation

- **Goal**: Reconstruct 3D point \( \tilde{x} = (x, y, z, w)^\top \in \mathbb{P}^3 \) from 2D image observations \( \{y_1, \ldots, y_N\} \) for known camera poses \( \{T_1, \ldots, T_N\} \)

- **Linear solution**: Find 3D point such that reprojections equal its projections
  \[
  y'_i = \pi(T_i\tilde{x}) = \left( \frac{r_{11}x + r_{12}y + r_{13}z + t_x w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \quad \frac{r_{21}x + r_{22}y + r_{23}z + t_y w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \right)
  \]
  - Each image provides one constraint \( y_i - y'_i = 0 \)
  - Leads to system of linear equations \( Ax = 0 \), two approaches:
    - Set \( w = 1 \) and solve nonhomogeneous system
    - Find nullspace of \( A \) using SVD (this is what we did in CV I)

- **Non-linear solution**: Minimize least squares reprojection error (more accurate)
  \[
  \min_x \left\{ \sum_{i=1}^{N} \| y_i - y'_i \|_2^2 \right\}
  \]
Relative Scale Recovery

• Problem:
  – Each subsequent frame-pair gives another solution for the reconstruction scale

• Solution:
  – Triangulate overlapping points $Y_{t-2}, Y_{t-1}, Y_t$ for current and last frame pair

  \[ \Rightarrow X_{t-2, t-1}, X_{t-1, t} \]

  – Rescale translation of current relative pose estimate to match the reconstruction scale with the distance ratio between corresponding point pairs

  \[ r_{i,j} = \frac{||x_{t-2, t-1, i} - x_{t-2, t-1, j}||_2}{||x_{t-1, t, i} - x_{t-1, t, j}||_2} \]

  – Use mean or robust median over available pair ratios
Algorithm: 2D-to-2D Visual Odometry

Input: image sequence $I_{0:t}$
Output: aggregated camera poses $T_{0:t}$

Algorithm:
For each current image $I_k$:
1. Extract and match keypoints between $I_{k-1}$ and $I_k$
2. Compute relative pose $T_{k-1}^k$ from essential matrix between $I_{k-1}$, $I_k$
3. Compute relative scale and rescale translation of $T_{k-1}^k$ accordingly
4. Aggregate camera pose by $T_k = T_{k-1}T_{k-1}^k$
Topics of This Lecture

• Visual Odometry
  – Definition, Motivation

• Geometry Background
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• Point-based Visual Odometry
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
2D-to-3D Motion Estimation

• Given a local set of 3D points \( X = \{x_1, \ldots, x_N\} \) and corresponding image observations \( Y_t = \{y_{t,1}, \ldots, y_{t,N}\} \), determine camera pose \( T_t \) within the local map.

• Minimize least squares geometric reprojection error

\[
E(T_t) = \sum_{i=1}^{N} \|y_{t,i} - \pi(T_t x_i)\|_2^2
\]

• **Perspective-n-Points (PnP) problem**, many approaches exist, e.g.,
  - Direct linear transform (DLT)
  - EPnP [Lepetit et al., An accurate O(n) Solution to the PnP problem, IJCV 2009]
Direct Linear Transform for PnP

• Goal: determine projection matrix \( P = (R \ t) \in R^{3 \times 4} = \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} \)

• Each 2D-to-3D point correspondence
  3D: \( \tilde{x}_i = (x_i, y_i, z_i, w_i)^T \in \mathbb{P}^3 \)
  2D: \( \tilde{y}_i = (x'_i, y'_i, w'_i)^T \in \mathbb{P}^2 \)
  gives two constraints

\[
\begin{pmatrix}
0 & -w'_i \tilde{x}_i^T \\
w'_i \tilde{x}_i^T & 0 \\
0 & -x'_i \tilde{x}_i^T
\end{pmatrix}
\begin{pmatrix}
P_1^T \\
P_2^T \\
P_3^T
\end{pmatrix} = 0
\]

through \( \tilde{y}_i \times (P\tilde{x}_i) = 0 \)

• Form linear system of equation \( A_p = 0 \) with
  from \( N \geq 6 \) correspondences

\[
p := \begin{pmatrix} P_1^T \\ P_2^T \\ P_3^T \end{pmatrix} \in R^9
\]

• Solve for \( p \): determine unit singular vector of \( A \) corresponding to its smallest eigenvalue
Algorithm: 2D-to-3D Visual Odometry

**Input:** image sequence \( I_{0:t} \)

**Output:** aggregated camera poses \( T_{0:t} \)

Algorithm:

Initialize:
1. Extract and match keypoints between \( I_0 \) and \( I_1 \)
2. Determine camera pose (essential matrix) and triangulate 3D keypoints \( X_1 \)

For each current image \( I_k \):
1. Extract and match keypoints between \( I_{k-1} \) and \( I_k \)
2. Compute camera pose \( T_k \) using PnP from 2D-to-3D matches
3. Triangulate all new keypoint matches between \( I_{k-1} \) and \( I_k \) and add them to the local map \( X_k \)
Topics of This Lecture

• Visual Odometry
  – Definition, Motivation

• Geometry Background
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• Point-based Visual Odometry
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
3D-to-3D Motion Estimation

• Given 3D point coordinates of corresponding points in two camera frames

\[ X_{t-1} = \{x_{t-1,1}, \ldots, x_{t-1,N}\} \]

\[ X_t = \{x_{t,1}, \ldots, x_{t,N}\} \]

determine relative camera pose \( T_{t}^{t-1} \)

• Idea: determine rigid transformation that aligns the 3D points

• Geometric least squares error:

\[ E \left( T_{t}^{t-1} \right) = \sum_{i=1}^{N} \left\| \overline{x}_{t-1,i} - T_{t}^{t-1} \overline{x}_{t,i} \right\|_2^2 \]

• Closed-form solutions available, e.g., [Arun et al., 1987]
• Applicable, e.g., for calibrated stereo cameras (triangulation of 3D points) or RGB-D cameras (measured depth)

Slide credit: Jörg Stückler
3D Rigid-Body Motion from 3D-to-3D Matches

- [Arun et al., Least-squares fitting of two 3-d point sets, IEEE PAMI, 1987]
- Corresponding 3D points, \( N \geq 3 \)

\[
X_{t-1} = \{x_{t-1,1}, \ldots, x_{t-1,N}\} \quad X_t = \{x_{t,1}, \ldots, x_{t,N}\}
\]

- Determine means of 3D point sets

\[
\mu_{t-1} = \frac{1}{N} \sum_{i=1}^{N} x_{t-1,i} \quad \mu_t = \frac{1}{N} \sum_{i=1}^{N} x_{t,i}
\]

- Determine rotation from

\[
A = \sum_{i=1}^{N} (x_{t-1} - \mu_{t-1}) (x_t - \mu_t)^\top \\
A = USV^\top \quad R_{t-1}^t = VU^\top
\]

- Determine translation as

\[
t_{t-1}^t = \mu_t - R_{t-1}^t \mu_{t-1}
\]
Algorithm: 3D-to-3D Stereo Visual Odometry

Input: stereo image sequence $I_{0:t}^l, I_{0:t}^r$  
Output: aggregated camera poses $T_{0:t}$

Algorithm:
For each current stereo image $I_k^l, I_k^r$:
1. Extract and match keypoints between $I_k^l$ and $I_{k-1}^l$  
2. Triangulate 3D points $X_k$ between $I_k^l$ and $I_k^r$  
3. Compute camera pose $T_{k-1}^k$ from 3D-to-3D point matches $X_k$ to $X_{k-1}$  
4. Aggregate camera poses by $T_k = T_{k-1}T_{k-1}^k$
Topics of This Lecture

• Visual Odometry
  – Definition, Motivation

• Geometry Background
  – Euclidean Transformations
  – 3D Rotation representations
  – Definition of Visual Odometry
  – Direct vs. Indirect methods

• Point-based Visual Odometry
  – 2D-to-2D Motion Estimation
  – 2D-to-3D Motion Estimation
  – 3D-to-3D Motion Estimation
  – Further Considerations
Further Considerations

- How to detect keypoints?
- How to match keypoints?
- How to cope with outliers among keypoint matches?
- How to cope with noisy observations?
- When to create new 3D keypoints? Which keypoints to use?
- 2D-to-2D, 2D-to-3D or 3D-to-3D?
- Optimize over more than two frames?
- ...

Slide credit: Jörg Stückler
Recap: Keypoint Detectors

- **Corners**
  - Image locations with locally prominent intensity variation
  - Intersections of edges

- **Examples:** Harris, FAST
  - **Scale-selection:** Harris-Laplace

- **Blobs**
  - Image regions that stick out from their surrounding in intensity/texture
  - Circular high-contrast regions

- **E.g.:** LoG, DoG (SIFT), SURF
  - **Scale-space extrema** in LoG/DoG

*Image source: Svetlana Lazebnik*
Recap: Keypoint Detectors

• Desirable properties of keypoint detectors for VO:
  – High repeatability,
  – Localization accuracy,
  – Robustness,
  – Invariance,
  – Computational efficiency

Harris Corners

DoG (SIFT) Blobs

Image source: Svetlana Lazebnik
Recap: Keypoint Detectors

• Corners vs. blobs for visual odometry:
  – Typically corners provide higher spatial localization accuracy, but are less well localized in scale
  – Corners are typically detected in less distinctive local image regions
  – Highly run-time efficient corner detectors exist (e.g., FAST)

Harris Corners

DoG (SIFT) Blobs

Image source: Svetlana Lazebnik
Recap: Keypoint Matching

- Desirable properties for VO:
  - High recall,
  - Precision,
  - Robustness,
  - Computational efficiency
Recap: Keypoint Matching

- Several **data association** principles:
  - Matching by **reprojection error / distance to epipolar line**
    - Assumes an initial guess for camera motion
    - *(e.g., Kalman filter prediction, IMU, or wheel odometry)*
  - **Detect-then-track** *(e.g., KLT-tracker):*
    - Correspondence search by local image alignment
    - Assumes incremental small (but unknown) motion between images
  - **Matching by descriptor:**
    - Scale-/viewpoint-invariant local descriptors allow for wider image baselines
  - Robustness through **RANSAC** for motion estimation
Recap: Local Feature Descriptors

- Extract signatures that describe local image regions:
  - Histograms over image gradients (SIFT)
  - Histograms over Haar-wavelet responses (SURF)
  - Binary patterns (BRIEF, BRISK, FREAK, etc.)
  - Learning-based descriptors (e.g., Calonder et al., ECCV 2008)
- Rotation-invariance: Align with dominant orientation
- Scale-invariance: Adapt local region extent to keypoint scale

Slide credit: Jörg Stückler
Image source: Svetlana Lazebnik / Calonder et al., ECCV 2010
Recap: RANSAC

- Model fitting in presence of noise and outliers
- Example: fitting a line through 2D points
Recap: RANSAC

- Least-squares solution, assuming constant noise for all points
Recap: RANSAC

• We only need 2 points to fit a line. Let’s try 2 random points

Quite ok..
7 inliers
4 outliers
Recap: RANSAC

• Let’s try 2 other random points

Quite bad..
3 inliers
8 outliers
Recap: RANSAC

- Let’s try yet another 2 random points

Quite good!

9 inliers
2 outliers
Recap: RANSAC

• Let’s use the inliers of the best trial to perform least squares fitting
Recap: RANSAC

• **RAN**dom **SA**mple **C**onsensus algorithm formalizes this idea

• Algorithm:
  Input: data $D$, $s$ required data points for fitting, success probability $p$, outlier ratio $\epsilon$
  Output: inlier set
  1. Compute required number of iterations $N = \frac{\log (1 - p)}{\log (1 - (1 - \epsilon)^s)}$
  2. For $N$ iterations do:
     1. Randomly select a subset of $s$ data points
     2. Fit model on the subset
     3. Count inliers and keep model/subset with largest number of inliers
  3. Refit model using found inlier set
Recap: RANSAC

- Required number of iterations
  - $N$ for $p = 0.99$

<table>
<thead>
<tr>
<th>Req. #points (s)</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>17</td>
<td>27</td>
</tr>
<tr>
<td>Plane</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>11</td>
<td>19</td>
<td>35</td>
<td>70</td>
</tr>
<tr>
<td>Essential matrix</td>
<td>8</td>
<td>9</td>
<td>26</td>
<td>78</td>
<td>272</td>
<td>1177</td>
<td>7025</td>
</tr>
</tbody>
</table>
Textbooks

• More background on Algebraic Geometry and Visual Odometry can be found in

An Invitation to 3D Vision, Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, Springer, 2004