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

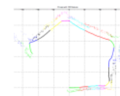


image source: [Zhang, Li, Nevatia, CVPR'08]

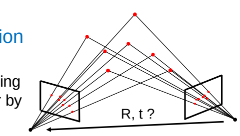
Topics of This Lecture

- 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

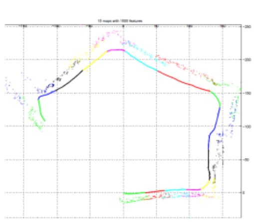


Slide credit: Jörg Stückler

Recap: What is Visual Odometry ?

Visual odometry (VO)…

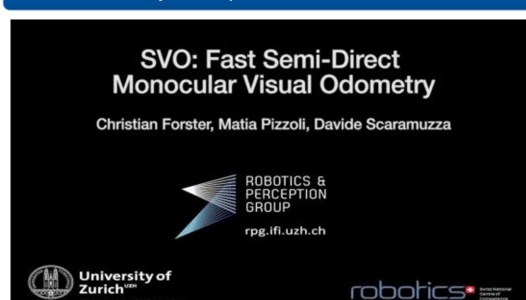
- … is prone to drift due to its local view
- … is primarily concerned with estimating camera motion
  - Not all approaches estimate a 3D reconstruction of the associated interest points/ pixels explicitly.
  - If so it is only locally consistent



Slide credit: Jörg Stückler

Image source: [Clemente et al. RSS 2007]

Visual Odometry Example

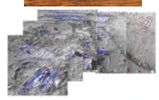


Slide credit: Jörg Stückler

Visual Odometry Term

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

- Visual Odometry
  - 1980-2004: Prominent research by NASA JPL for Mars exploration rovers (Spirit and Opportunity in 2004)
  - 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

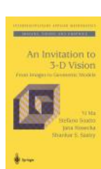
- Active RGB-D sensors
  - Pros: no texture needed, similar to stereo processing
  - Cons: active sensing consumes power, blackbox depth estimation

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

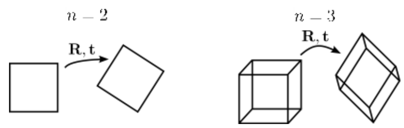
	2D	3D
Point	$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$	$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$
Augmented vector	$\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{R}^3$	$\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4$
Homogeneous coordinates $\tilde{\mathbf{x}} = \tilde{w}\bar{\mathbf{x}}$	$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^2$	$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^3$

Euclidean Transformations

- Euclidean transformations apply rotation and translation

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t} \qquad \bar{\mathbf{x}}' = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \bar{\mathbf{x}}$$

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



Special Orthogonal Group SO(n)

- Rotation matrices have a special structure

$$\mathbf{R} \in \mathbf{SO}(n) \subset \mathbb{R}^{n \times n}, \det(\mathbf{R}) = 1, \mathbf{R}\mathbf{R}^T = \mathbf{I}$$

  i.e. orthonormal matrices that preserve distance and orientation

- They form a group denoted as Special Orthogonal Group $\mathrm{SO}(n)$
  - 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

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

- Pro:
  - Easy to concatenate and invert

$$\mathbf{R}_C^A = \mathbf{R}_B^A \mathbf{R}_C^B \qquad \mathbf{R}_A^B = \left(\mathbf{R}_B^A\right)^{-1}$$

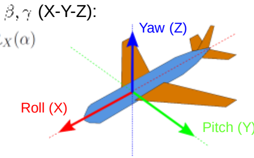
- Con:
  - Overparametrized (9 parameters for 3 DoF) – problematic for optimization

3D Rotation Representations – Euler Angles

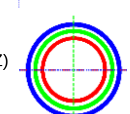
- Euler Angles: 3 consecutive rotations around coordinate axes
  Example: roll-pitch-yaw angles α, β, γ (X-Y-Z):

$$\mathbf{R}_{XYZ}(\alpha, \beta, \gamma) = \mathbf{R}_Z(\gamma)\,\mathbf{R}_Y(\beta)\,\mathbf{R}_X(\alpha)$$

with
$$\mathbf{R}_X(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$
$$\mathbf{R}_Y(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$$
$$\mathbf{R}_Z(\gamma) = \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



- 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!

3D Rotation Representations – Axis-Angle

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

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

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

- Reverse: $\theta = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right) \qquad \mathbf{n} = \frac{1}{2\sin(\theta)} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$
- 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 $\mathrm{SO}(3)$

3D Rotation Representations – Quaternions

- Unit Quaternions: $\mathbf{q} = (q_x, q_y, q_z, q_w)^\top \in \mathbb{R}^4$, $\|\mathbf{q}\|_2 = 1$
- Relation to axis-angle representation:
  - Axis-angle to quaternion:

$$\mathbf{q}(\mathbf{n}, \theta) = \left(\mathbf{n}^\top \sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right)\right)$$

  - Quaternion to axis-angle:

$$\mathbf{n}(\mathbf{q}) = \begin{cases} (q_x, q_y, q_z)^\top / \sin(\theta/2), & \theta \neq 0 \\ \mathbf{0}, & \theta = 0 \end{cases}$$
$$\theta = 2\arccos(q_w)$$

3D Rotation Representations – Quaternions cont.

- Pros:
  - Unique up to opposing sign $\mathbf{q}$ $\quad -\mathbf{q}$
  - Direct rotation of a point:

    $$\mathbf{p}' = \mathbf{R}\mathbf{p} = \mathbf{q}(\mathbf{R})\mathbf{p}\mathbf{q}(\mathbf{R})^{-1}$$

  - Direct concatenation of rotations:

    $$\mathbf{q}(\mathbf{R}_2\mathbf{R}_1) = \mathbf{q}(\mathbf{R}_2)\mathbf{q}(\mathbf{R}_1)$$

  - Direct inversion of a rotation:

    $$\mathbf{q}(\mathbf{R}^{-1}) = \mathbf{q}(\mathbf{R})^{-1}$$

    with $\quad \mathbf{q}^{-1} = (-\mathbf{q}_{xyz}^\top, q_w)^\top$,

    $$\mathbf{q}_1\mathbf{q}_2 = (q_{1,w}\mathbf{q}_{2,xyz} + q_{2,w}\mathbf{q}_{1,xyz} + \mathbf{q}_{1,xyz} \times \mathbf{q}_{2,xyz}, q_{1,w}q_{2,w} - \mathbf{q}_{1,xyz}\mathbf{q}_{2,xyz})$$

- Con: Normalization constraint is problematic for optimization

Special Euclidean Group SE(3)

- Euclidean transformation matrices have a special structure as well: $\quad \mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbf{SE}(3) \subset \mathbb{R}^{4\times 4}$
  - Translation $\mathbf{t}$ has 3 degrees of freedom
  - Rotation $\mathbf{R} \in \mathbf{SO}(3)$ has 3 degrees of freedom
- They also form a group which we call $\mathbf{SE}(3)$. The group operator is matrix multiplication:

  $$\cdot : \mathbf{SE}(3) \times \mathbf{SE}(3) \to \mathbf{SE}(3)$$
  $$\mathbf{T}_B^A \cdot \mathbf{T}_C^B \mapsto \mathbf{T}_C^A$$

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 $\mathbf{T}_t^{t-1} \in \mathbf{SE}(3)$ between frames

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

  $$\mathbf{T}_t = \mathbf{T}_0\mathbf{T}_1^0 \cdots \mathbf{T}_t^{t-1}$$

Direct vs. Indirect Methods

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

  $$p(\mathbf{I}_2 \mid \mathbf{I}_1, \boldsymbol{\xi}) \quad \longrightarrow \quad E(\boldsymbol{\xi}) = \int_{\mathbf{u}\in\Omega} |\mathbf{I}_1(\mathbf{u}) - \mathbf{I}_2(\omega(\mathbf{u}, \boldsymbol{\xi}))| \, d\mathbf{u}$$

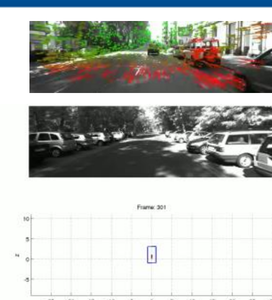
- Indirect methods
  - formulate alignment objective in terms of reprojection error of geometric primitives (e.g. points, lines)

  $$p(\mathbf{Y}_2 \mid \mathbf{Y}_1, \boldsymbol{\xi}) \quad \longrightarrow \quad E(\boldsymbol{\xi}) = \sum_i |\mathbf{y}_{1,i} - \omega(\mathbf{y}_{2,i}, \boldsymbol{\xi})|$$

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

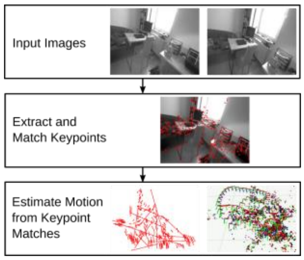
Point-based (Indirect) Visual Odometry Example



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)

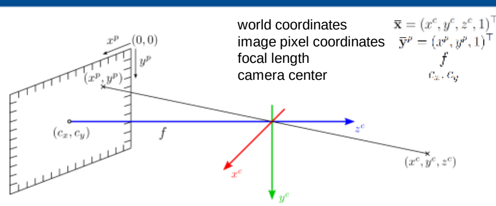


Topics of This Lecture

- 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 $\bar{\mathbf{x}} = (x^c, y^c, z^c, 1)^\top$
image pixel coordinates $\bar{\mathbf{y}}^p = (x^p, y^p, 1)^\top$
focal length f
camera center c_x, c_y

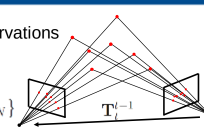


$$\begin{pmatrix} x^p \\ y^p \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{camera matrix } C} \underbrace{\begin{pmatrix} x^c/z^c \\ y^c/z^c \\ 1 \end{pmatrix}}_{=:\pi(\bar{\mathbf{x}}) = \bar{\mathbf{y}}}$$

camera matrix C $\quad =: \pi(\bar{\mathbf{x}}) = \bar{\mathbf{y}}$ (normalized image coordinates)

2D-to-2D Motion Estimation

- Given corresponding image point observations
  $Y_t = \{\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,N}\}$
  $Y_{t-1} = \{\mathbf{y}_{t-1,1}, \ldots, \mathbf{y}_{t-1,N}\}$
  of unknown 3D points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  determine relative motion $\mathbf{T}_t^{t-1}$ between frames

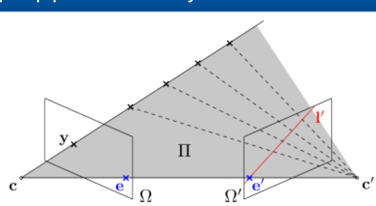


- Obvious try: minimize reprojection error using least squares

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

- Convexity? Uniqueness (scale-ambiguity)?
- Alternative algebraic approaches: 8-point / 5-point algorithm

Recap: Epipolar Geometry

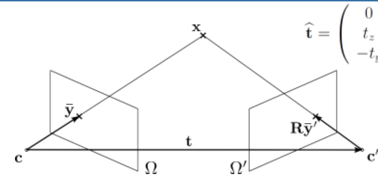


- Camera centers $\mathbf{c}, \mathbf{c}'$ and image point $\mathbf{y} \in \Omega$ span the epipolar plane Π
- The ray from camera center $\mathbf{c}$ through point $\mathbf{y}$ projects as the epipolar line $\mathbf{l}'$ in image plane Ω'
- The intersections of the line through the camera centers with the image planes are called epipoles $\mathbf{c}, \mathbf{c}'$

Essential Matrix

$$\hat{\mathbf{t}} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}$$

$$\hat{\mathbf{t}} = [\mathbf{t}]_\times$$



- The rays to the 3D point and the baseline $\mathbf{t}$ are coplanar

$$\bar{\mathbf{y}}^\top \left(\mathbf{t} \times \mathbf{R}\bar{\mathbf{y}}'\right) = 0 \Leftrightarrow \bar{\mathbf{y}}^\top \hat{\mathbf{t}}\mathbf{R}\bar{\mathbf{y}}' = 0$$

- 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
     $\tilde{\mathbf{y}}_i \mathbf{E} \tilde{\mathbf{y}}'_i = \mathbf{a}_i \mathbf{E}_s = 0 \longrightarrow \mathbf{A}\mathbf{E}_s = \mathbf{0} \qquad \mathbf{A} = \left(\mathbf{a}_1^\top, \ldots, \mathbf{a}_N^\top\right)^\top$
     using Kronecker product $\mathbf{a}_i = \tilde{\mathbf{y}}_i \otimes \tilde{\mathbf{y}}'_i$ and $\mathbf{E}_s = (e_{11}, e_{12}, e_{13}, \ldots, e_{33})^\top$
  2. Apply singular value decomposition (SVD) on $\mathbf{A} = \mathbf{U_A S_A V_A^\top}$ and unstack the 9th column of $\mathbf{V_A}$ into $\tilde{\mathbf{E}}$
  3. Project the approximate $\tilde{\mathbf{E}}$ into the (normalized) essential space: Determine the SVD of $\tilde{\mathbf{E}} = \mathbf{U} \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3) \mathbf{V}^\top$ with $\mathbf{U}, \mathbf{V} \in \mathbf{SO}(3)$ and replace the singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3$ with $1, 1, 0$ to find

$$\mathbf{E} = \mathbf{U} \operatorname{diag}(1, 1, 0) \mathbf{V}^\top$$

Eight-Point Algorithm cont.

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

$$\mathbf{R} = \mathbf{U}\mathbf{R}_Z^\top\left(\pm\frac{\pi}{2}\right)\mathbf{V}^\top \qquad \hat{\mathbf{t}} = \mathbf{U}\mathbf{R}_Z\left(\pm\frac{\pi}{2}\right)\operatorname{diag}(1, 1, 0)\mathbf{U}^\top$$

- 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 $\tilde{\mathbf{x}} = (x, y, z, w)^\top \in \mathbb{P}^3$ from 2D image observations $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ for known camera poses $\{\mathbf{T}_1, \ldots, \mathbf{T}_N\}$
- Linear solution: Find 3D point such that reprojections equal its projections

$$\mathbf{y}'_i = \pi(\mathbf{T}_i \tilde{\mathbf{x}}) = \begin{pmatrix} \frac{r_{11}x + r_{12}y + r_{13}z + t_x w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \\ \frac{r_{21}x + r_{22}y + r_{23}z + t_y w}{r_{31}x + r_{32}y + r_{33}z + t_z w} \end{pmatrix}$$

  – Each image provides one constraint $\mathbf{y}_i - \mathbf{y}'_i = 0$
  – Leads to system of linear equations $\mathbf{A}\tilde{\mathbf{x}} = 0$, two approaches:
    - Set $w = 1$ and solve nonhomogeneous system
    - Find nullspace of $\mathbf{A}$ using SVD (this is what we did in CV I)
- Non-linear solution: Minimize least squares reprojection error (more accurate)

$$\min_{\mathbf{x}} \left\{ \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{y}'_i\|_2^2 \right\}$$

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

$$r_{i,j} = \frac{\|\mathbf{x}_{t-2,t-1,i} - \mathbf{x}_{t-2,t-1,j}\|_2}{\|\mathbf{x}_{t-1,t,i} - \mathbf{x}_{t-1,t,j}\|_2}$$

  – 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 $\mathbf{T}_{0:t}$

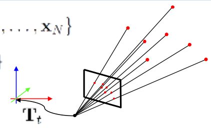
Algorithm:
For each current image I_k :
1. Extract and match keypoints between I_{k-1} and I_k
2. Compute relative pose $\mathbf{T}_k^{k-1}$ from essential matrix between I_{k-1}, I_k
3. Compute relative scale and rescale translation of $\mathbf{T}_k^{k-1}$ accordingly
4. Aggregate camera pose by $\mathbf{T}_k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

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 $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and corresponding image observations
  $Y_t = \{\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,N}\}$
  determine camera pose $\mathbf{T}_t$ within the local map



- Minimize least squares geometric reprojection error

$$E(\mathbf{T}_t) = \sum_{i=1}^{N} \|\mathbf{y}_{t,i} - \pi(\mathbf{T}_t \mathbf{x}_i)\|_2^2$$

- 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]
  - OPnP [Zheng et al., Revisiting the PnP Problem: A Fast, General and Optimal Solution, ICCV 2013]

Direct Linear Transform for PnP

- Goal: determine projection matrix $\mathbf{P} = (\mathbf{R}\ \mathbf{t}) \in \mathbb{R}^{3\times 4} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}$
- Each 2D-to-3D point correspondence
  3D: $\tilde{\mathbf{x}}_i = (x_i, y_i, z_i, w_i)^\top \in \mathbb{P}^3$ 2D: $\tilde{\mathbf{y}}_i = (x'_i, y'_i, w'_i)^\top \in \mathbb{P}^2$
  gives two constraints

$$\begin{pmatrix} \mathbf{0} & -w'_i \tilde{\mathbf{x}}_i^\top & y'_i \tilde{\mathbf{x}}_i^\top \\ w'_i \tilde{\mathbf{x}}_i^\top & \mathbf{0} & -x'_i \tilde{\mathbf{x}}_i^\top \end{pmatrix} \begin{pmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{pmatrix} = \mathbf{0}$$

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

- Form linear system of equation $\mathbf{A}\mathbf{p} - \mathbf{0}$ with $\mathbf{p} := \begin{pmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{pmatrix} \in \mathbb{R}^9$ from $N \geq 6$ correspondences
- Solve for $\mathbf{p}$: determine unit singular vector of $\mathbf{A}$ corresponding to its smallest eigenvalue

Algorithm: 2D-to-3D Visual Odometry

Input: image sequence $I_{0:t}$
Output: aggregated camera poses $\mathbf{T}_{0:t}$

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

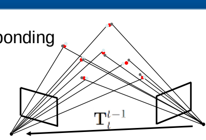
For each current image I_k :
1. Extract and match keypoints between I_{k-1} and I_k
2. Compute camera pose $\mathbf{T}_k$ using PnP from 2D-to-3D matches
3. Triangulate all new keypoint matches between I_{k-1} and I_k and add them to the local map X_k

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
  $X_{t-1} = \{\mathbf{x}_{t-1,1}, \ldots, \mathbf{x}_{t-1,N}\}$
  $X_t = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,N}\}$
  determine relative camera pose $\mathbf{T}_t^{t-1}$



- Idea: determine rigid transformation that aligns the 3D points
- Geometric least squares error: $E\left(\mathbf{T}_t^{t-1}\right) = \sum_{i=1}^{N} \left\|\overline{\mathbf{x}}_{t-1,i} - \mathbf{T}_t^{t-1}\overline{\mathbf{x}}_{t,i}\right\|_2^2$
- 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)

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, $N \geq 3$

$$X_{t-1} = \{\mathbf{x}_{t-1,1}, \ldots, \mathbf{x}_{t-1,N}\} \qquad X_t = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,N}\}$$

- Determine means of 3D point sets

$$\boldsymbol{\mu}_{t-1} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{t-1,i} \qquad \boldsymbol{\mu}_t = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_{t,i}$$

- Determine rotation from

$$\mathbf{A} = \sum_{i=1}^{N} (\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_t - \boldsymbol{\mu}_t)^\top \qquad \mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \qquad \mathbf{R}_{t-1}^t = \mathbf{V}\mathbf{U}^\top$$

- Determine translation as $\mathbf{t}_{t-1}^t = \boldsymbol{\mu}_t - \mathbf{R}_{t-1}^t \boldsymbol{\mu}_{t-1}$

Algorithm: 3D-to-3D Stereo Visual Odometry

Input: stereo image sequence $I^l_{0:t}, I^r_{0:t}$
Output: aggregated camera poses $\mathbf{T}_{0:t}$

Algorithm:
For each current stereo image I^l_k, I^r_k:
1. Extract and match keypoints between I^l_k and I^l_{k-1}
2. Triangulate 3D points X_k between I^l_k and I^r_k
3. Compute camera pose $\mathbf{T}^{k-1}_k$ from 3D-to-3D point matches X_k to X_{k-1}
4. Aggregate camera poses by $\mathbf{T}_k = \mathbf{T}_{k-1}\mathbf{T}^{k-1}_k$

Topics of This Lecture

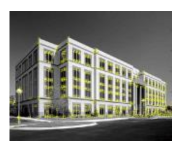
Further Considerations

- 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?
- …

Recap: Keypoint Detectors

- Corners
  - Image locations with locally prominent intensity variation
  - Intersections of edges
- Examples: Harris, FAST
- Scale-selection: Harris-Laplace

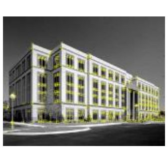
- Blobs
  - Image regions that stick out from their surrounding in intensity/texture
  - Circular high-contrast regions
- E.g.: LoG, DoG (SIFT), SURF
- Scale-space extrema in LoG/DoG


Harris Corners


DoG (SIFT) Blobs

Recap: Keypoint Detectors

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


Harris Corners


DoG (SIFT) Blobs

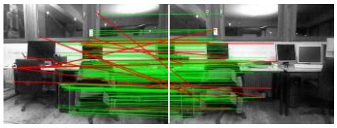
Recap: Keypoint Detectors

- 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

Recap: Keypoint Matching

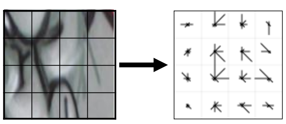


- Desirable properties for VO:
  - High recall,
  - Precision,
  - Robustness,
  - Computational efficiency

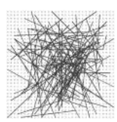
Recap: Keypoint Matching

- 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



SIFT gradient pooling

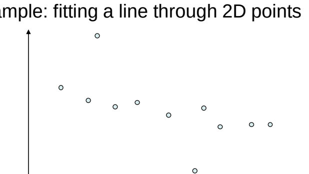


BRIEF test locations

- 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

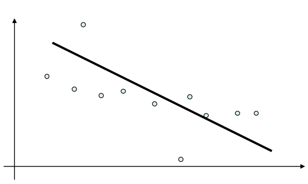
Recap: RANSAC

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



Recap: RANSAC

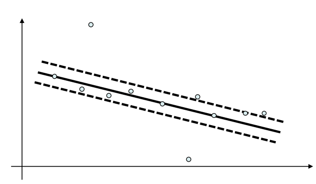
- Least-squares solution, assuming constant noise for all points



Bad!

Recap: RANSAC

- We only need 2 points to fit a line. Let's try 2 random points

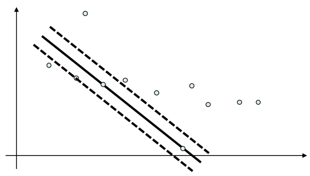


Quite ok..

7 inliers
4 outliers

Recap: RANSAC

- Let's try 2 other random points

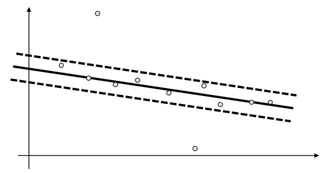


Quite bad..

3 inliers
8 outliers

Recap: RANSAC

- Let's try yet another 2 random points



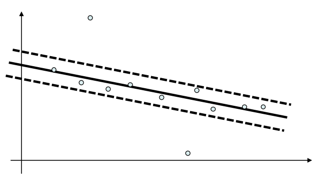
Quite good!

9 inliers
2 outliers

Slide credit: Jörg Stückler

Recap: RANSAC

- Let's use the inliers of the best trial to perform least squares fitting



Even better!

Slide credit: Jörg Stückler

Recap: RANSAC

- RANdom SAmple Consensus algorithm formalizes this idea
- Algorithm:
  Input: data D, s required data points for fitting, success probability p, outlier ratio ϵ
  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

Slide credit: Jörg Stückler

Recap: RANSAC

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

	Req. #points s	Outlier ratio ϵ 10%	20%	30%	40%	50%	60%	70%
Line	2	3	5	7	11	17	27	49
Plane	3	4	7	11	19	35	70	169
Essential matrix	8	9	26	78	272	1177	7025	70188

Slide credit: Jörg Stückler

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