Computer Vision 2
WS 2018/19

Part 13 – Visual Odometry II
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

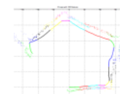


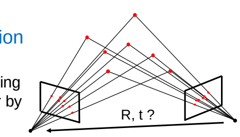
image source: [Zhang, Li, Nevatia, CVPR'08]

Topics of This Lecture

- Point-based Visual Odometry
  - Recap: 2D-to-2D Motion Estimation
  - 2D-to-3D Motion Estimation
  - 3D-to-3D Motion Estimation
  - Further Considerations
- Direct Methods
  - Direct image alignment
  - Pose parametrization
  - Lie group se(3) and the exponential map
  - Residual linearization
  - Optimization 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: Direct vs. Indirect Methods

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

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

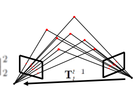
- Indirect methods
  - formulate alignment objective in terms of reprojection error of geometric primitives (e.g., points, lines)

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

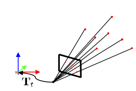
Slide credit: Jörg Stückler

Motion Estimation from Point Correspondences

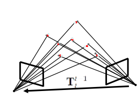
- 2D-to-2D
  - Reproj. error:
    $$E\left(\mathbf{T}_t^{t-1}, X\right) = \sum_{i=1}^{N} \|\bar{\mathbf{y}}_{t,i} - \pi(\bar{\mathbf{x}}_i)\|_2^2 + \|\bar{\mathbf{y}}_{t-1,i} - \pi\left(\mathbf{T}_t^{t-1}\bar{\mathbf{x}}_i\right)\|_2^2$$
  - Introduced linear algorithm: 8-point



- 2D-to-3D
  - Reprojection error: $E(\mathbf{T}_t) = \sum_{i=1}^{N} \|\mathbf{y}_{t,i} - \pi(\mathbf{T}_t\bar{\mathbf{x}}_i)\|_2^2$
  - Introduced linear algorithm: DLT PnP



- 3D-to-3D
  - Reprojection error: $E\left(\mathbf{T}_t^{t-1}\right) = \sum_{i=1}^{N} \|\bar{\mathbf{x}}_{t-1,i} - \mathbf{T}_t^{t-1}\bar{\mathbf{x}}_{t,i}\|_2^2$
  - Introduced linear algorithm: Arun's method



Slide credit: Jörg Stückler

Recap: Eight-Point Algorithm for Essential Matrix Est.

- First proposed by Longuet and Higgins, 1981
- Algorithm:
  1. Rewrite epipolar constraints as a linear system of equations
     $\tilde{\mathbf{y}}_i \mathbf{E} \tilde{\mathbf{y}}'_i = \mathbf{a}_i \mathbf{E}_s = 0 \longrightarrow \mathbf{A}\mathbf{E}_s = 0 \qquad \mathbf{A} = (\mathbf{a}_1^\top, \ldots, \mathbf{a}_N^\top)^\top$
     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 $\mathbf{A} = \mathbf{U}_\mathbf{A} \mathbf{S}_\mathbf{A} \mathbf{V}_\mathbf{A}^\top$ and unstack the 9th column of $\mathbf{V}_\mathbf{A}$ into $\tilde{\mathbf{E}}$.
  3. Project the approximate $\tilde{\mathbf{E}}$ into the (normalized) essential space: Determine the SVD of $\tilde{\mathbf{E}} = \mathbf{U}\,\mathrm{diag}(\sigma_1, \sigma_2, \sigma_3)\mathbf{V}^\top$ 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

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

Recap: Eight-Point Algorithm cont.

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

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

- 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

Recap: Eight-Point Algorithm cont.

- Normalized essential matrix: $\|\mathbf{E}\| = \left\|\hat{\mathbf{t}}\right\| = 1$
- Linear algorithms exist that require only 6 points for general motion
- Non-linear 5-point algorithm with up to 10 (possibly complex) solutions
- Points need to be in „general position": certain degenerate configurations exists (e.g., all points on a plane)
- No translation, ideally: $\left\|\hat{\mathbf{t}}\right\| = 0 \Rightarrow \|\mathbf{E}\| = 0$
- But: for small translations, signal-to-noise ratio of image parallax may be problematic: „spurious" pose estimate

Normalized Eight-Point Algorithm

- Hartley, In Defense of the Eight-Point Algorithm, PAMI 1997
  - Conditioning of $\mathbf{A}$ can be improved by shifting and rescaling image coordinates
  - Normalize coordinates to zero mean and unit variance
  - Very important for estimating the fundamental matrix due to pixel coordinates

Recap: 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 $\mathbf{T}_{0:t}$

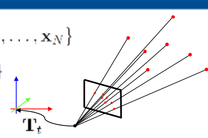
Algorithm:
For each current image I_k :
1. Extract and match keypoints between I_{k-1} and I_k
2. Compute relative pose $\mathbf{T}_k^{k-1}$ from essential matrix between I_{k-1}, I_k
3. Compute relative scale and rescale translation of $\mathbf{T}_k^{k-1}$ accordingly
4. Aggregate camera pose by $\mathbf{T}_k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

Topics of This Lecture

- Point-based Visual Odometry
  - Recap: 2D-to-2D Motion Estimation
  - 2D-to-3D Motion Estimation
  - 3D-to-3D Motion Estimation
  - Further Considerations
- Direct Methods
  - Direct image alignment
  - Pose parametrization
  - Lie group se(3) and the exponential map
  - Residual linearization
  - Optimization 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 $\mathbf{P} = (\mathbf{R}\ \mathbf{t}) \in \mathbb{R}^{3\times4} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}$
- Each 2D-to-3D point correspondence
  3D: $\widetilde{\mathbf{x}}_i = (x_i, y_i, z_i, w_i)^\top \in \mathbb{P}^3$ 2D: $\widetilde{\mathbf{y}}_i = (x'_i, y'_i, w'_i)^\top \in \mathbb{P}^2$
  gives two constraints

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

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

- Point-based Visual Odometry
  - Recap: 2D-to-2D Motion Estimation
  - 2D-to-3D Motion Estimation
  - 3D-to-3D Motion Estimation
  - Further Considerations
- Direct Methods
  - Direct image alignment
  - Pose parametrization
  - Lie group se(3) and the exponential map
  - Residual linearization
  - Optimization considerations

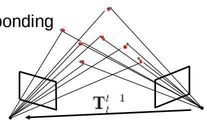
3D-to-3D Motion Estimation

- 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

  $$\mathbf{A} = \sum_{i=1}^{N} \left(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}\right) \left(\mathbf{x}_t - \boldsymbol{\mu}_t\right)^\top \qquad \mathbf{A} = \mathbf{USV}^\top \qquad \mathbf{R}_{t-1}^t = \mathbf{VU}^\top$$

- 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 $I_{0:t}^l, I_{0:t}^r$
Output: aggregated camera poses $\mathbf{T}_{0:t}$

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 $\mathbf{T}_k^{k-1}$ from 3D-to-3D point matches X_k to X_{k-1}
4. Aggregate camera poses by $\mathbf{T}_k = \mathbf{T}_{k-1} \mathbf{T}_k^{k-1}$

Topics of This Lecture

- Point-based Visual Odometry
  - Recap: 2D-to-2D Motion Estimation
  - 2D-to-3D Motion Estimation
  - 3D-to-3D Motion Estimation
  - Further Considerations
- Direct Methods
  - Direct image alignment
  - Pose parametrization
  - Lie group se(3) and the exponential map
  - Residual linearization
  - Optimization 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?
- …

Recap: Keypoint Detectors

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

Harris Corners

- 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

DoG (SIFT) Blobs

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

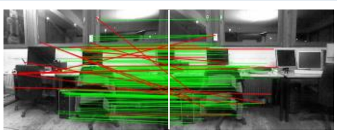
- Corners vs. blobs for visual odometry:
  - Typically corners provide higher spatial localization accuracy, but are less well localized in scale
  - Corners are typically detected in less distinctive local image regions
  - Highly run-time efficient corner detectors exist (e.g., FAST)

Harris Corners

DoG (SIFT) Blobs

Image source: Svetlana Lazebnik

Recap: Keypoint Matching

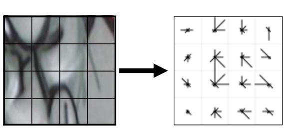


- Desirable properties for VO:
  - High recall,
  - Precision,
  - Robustness,
  - Computational efficiency

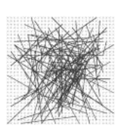
Recap: Keypoint Matching

- 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



SIFT gradient pooling



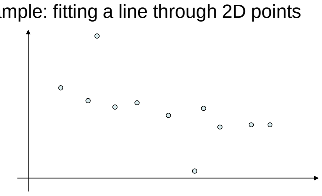
BRIEF test locations

- 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

Image source: Svetlana Lazebnik / Calonder et al., ECCV 2010

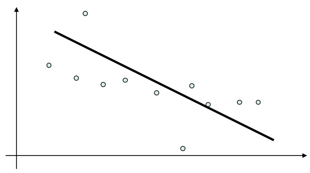
Recap: RANSAC

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



Recap: RANSAC

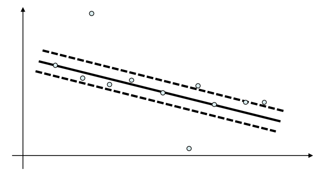
- Least-squares solution, assuming constant noise for all points



Bad!

Recap: RANSAC

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

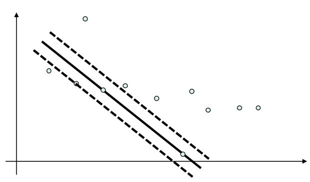


Quite ok..

7 inliers
4 outliers

Recap: RANSAC

- Let's try 2 other random points

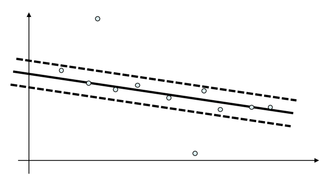


Quite bad..

3 inliers
8 outliers

Recap: RANSAC

- Let's try yet another 2 random points

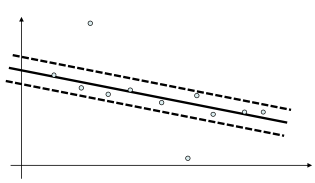


Quite good!

9 inliers
2 outliers

Recap: RANSAC

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



Even better!

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

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

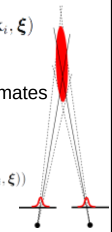
Probabilistic Modelling

- Model image point observation likelihood $p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\xi})$
  - E.g., Gaussian: $p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\xi}) \sim \mathcal{N}\left(\mathbf{y}_i; \pi\left(\mathbf{T}(\boldsymbol{\xi})\mathbf{x}_i\right), \Sigma_{\mathbf{y}_i}\right)$
- Optimize maximum a-posteriori likelihood of estimates

$$p(X, \boldsymbol{\xi} \mid Y) \propto p(Y \mid X, \boldsymbol{\xi})\, p(X, \boldsymbol{\xi}) \;= p(X, \boldsymbol{\xi}) \prod_{i=1}^{N} p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\xi})$$

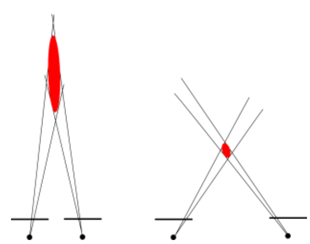
  - Neg. log-likelihood: $E(X, \boldsymbol{\xi}) = -\log(p(X, \boldsymbol{\xi})) \sum_{i=1}^{N} \log(p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\xi}))$
  - Gaussian prior and observation likelihood:

$$E(X, \boldsymbol{\xi}) = \text{const.} + \left(\boldsymbol{\xi} - \boldsymbol{\mu}_{\boldsymbol{\xi},0}\right)^{\top} \Sigma_{\boldsymbol{\xi},0}^{-1} \left(\boldsymbol{\xi} - \boldsymbol{\mu}_{\boldsymbol{\xi},0}\right) + \sum_{i=1}^{N} \left(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}_i,0}\right)^{\top} \Sigma_{\mathbf{x}_i,0}^{-1} \left(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}_i,0}\right) + \left(\mathbf{y}_i - \pi\left(\mathbf{T}(\boldsymbol{\xi})\mathbf{x}_i\right)\right)^{\top} \Sigma_{\mathbf{y}_i}^{-1} \left(\mathbf{y}_i - \pi\left(\mathbf{T}(\boldsymbol{\xi})\mathbf{x}_i\right)\right)$$



Drift in Motion Estimates

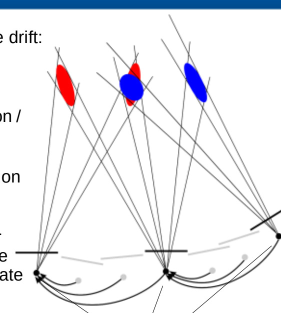
- Estimation errors accumulate: Drift
- Noisy observations in 2D image point location
- Motion estimation and triangulation accuracy depend on ratio of baseline to depth
- 3D-to-3D vs. 2D-to-3D:
  - Low 3D triangulation accuracy for small baseline
  - 3D-to-3D: 2x triangulation, typically less accurate than 2D-to-3D



baseline << depth   baseline ~ depth

Keyframes

- Popular approach to reduce drift: Keyframes
- Carefully select reference images for motion estimation / triangulation
- Incrementally estimate motion towards keyframe
- If baseline sufficient (and/or image overlap small), create next keyframe [and triangulate 3D positions of keypoints]



Motion Estimation for Input Type

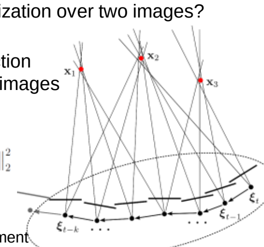
Correspondences	Monocular	Stereo	RGB-D
2D-to-2D	X	X	X
2D-to-3D	X	X	X
3D-to-3D		X	X

Local Optimization Windows

- Can we do better than optimization over two images?
- Optimize motion / reconstruction on a local current window of images

$$E(X_{t-k:t}, \boldsymbol{\xi}_{t-k:t}) = \sum_{j=0}^{k} \sum_{i=1}^{N_{t-j}} \left\| \mathbf{y}_{t-j,i} - \pi\left(\mathbf{T}(\boldsymbol{\xi}_{t-j})\mathbf{x}_{t-j,i}\right) \right\|_2^2$$

  - Local bundle adjustment
  - Local motion-only bundle adjustment (3D keypoint positions held fixed)
  - Initialize with algebraic approaches



optimization window

Summary

- Visual odometry estimates relative camera motion from image sequences
- Indirect point-based methods
  - Minimize geometric reprojection error
  - 2D-to-2D, 2D-to-3D, 3D-to-3D motion estimation
  - RANSAC for robust keypoint matching
  - Keyframes can reduce drift
  - Local optimization window can further increase accuracy
- *Next: direct methods*

Topics of This Lecture

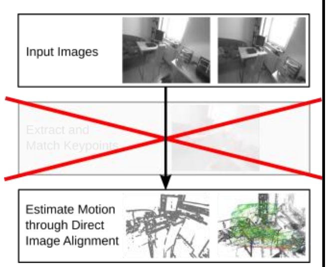
- Point-based Visual Odometry
  - Recap: 2D-to-2D Motion Estimation
  - 2D-to-3D Motion Estimation
  - 3D-to-3D Motion Estimation
  - Further Considerations
- Direct Methods
  - Direct image alignment
  - Pose parametrization
  - Lie group se(3) and the exponential map
  - Residual linearization
  - Optimization considerations

Direct Visual Odometry Pipeline

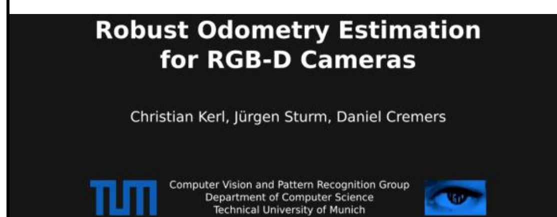
- Avoid manually designed keypoint detection and matching
- Instead: direct image alignment

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

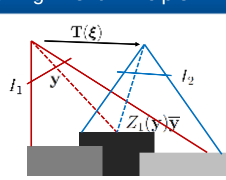
- Warping requires depth
  - RGB-D
  - Fixed-baseline stereo
  - Temporal stereo, tracking and (local) mapping



Direct Visual Odometry Example (RGB-D)



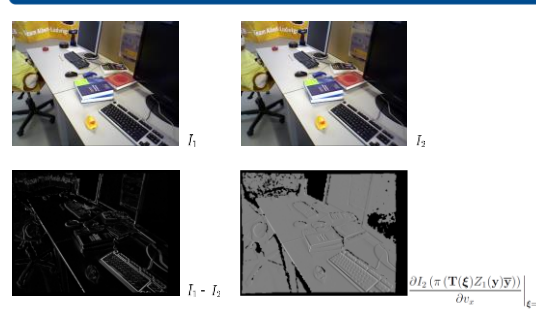
Direct Image Alignment Principle



- If we know pixel depth, we can „simulate" an image from a different view point
- Ideally, the warped image is the same as the image taken from that pose:

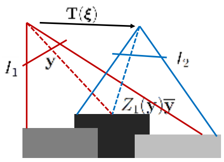
$$I_1(\mathbf{y}) = I_2(\pi(\mathbf{T}(\xi) Z_1(\mathbf{y})\bar{\mathbf{y}}))$$

Derivative of Image Warp



Images from Kerl et al. ICRA 2013

Direct RGB-D Image Alignment



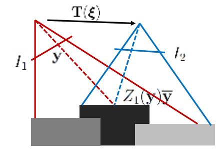
- RGB-D cameras measure depth, we only need to estimate camera motion!
- In addition to the photometric error

$$I_1(\mathbf{y}) = I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}}))$$

we can measure geometric error directly

$$[\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}}]_z = Z_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}}))$$

Probabilistic Direct Image Alignment



- Measurements are affected by noise

$$I_1(\mathbf{y}) = I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}})) + \epsilon$$

- A convenient assumption is Gaussian noise

$$\epsilon \sim \mathcal{N}(0, \sigma_I^2)$$

- If we further assume that pixel measurements are stochastically independent, we can formulate the a-posteriori probability

$$p(\boldsymbol{\xi} \mid I_1, I_2) \propto p(I_1 \mid \boldsymbol{\xi}, I_2)p(\boldsymbol{\xi})$$

$$\propto p(\boldsymbol{\xi}) \prod_{\mathbf{y}\in\Omega} \mathcal{N}\left(I_1(\mathbf{y}) - I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}})); 0, \sigma_I^2\right)$$

Optimization Approach

- Optimize negative log-likelihood
  - Product of exponentials becomes a summation over quadratic terms
  - Normalizers are independent of the pose

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{y}\in\Omega} \frac{r(\mathbf{y},\boldsymbol{\xi})^2}{\sigma_I^2} \text{, stacked residuals: } E(\boldsymbol{\xi}) = \mathbf{r}(\boldsymbol{\xi})^\top \mathbf{W}\mathbf{r}(\boldsymbol{\xi})$$

$$r(\mathbf{y},\boldsymbol{\xi}) = I_1(\mathbf{y}) - I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}}))$$

- Non-linear least squares problem can be efficiently optimized using standard second-order tools (Gauss-Newton, Levenberg-Marquardt)

Gauss-Newton for Non-Linear Least Squares

- Gauss-Newton method, iterate:
  - Linearize residuals:

$$\widetilde{\mathbf{r}}(\boldsymbol{\xi}) = \mathbf{r}(\boldsymbol{\xi}_i) + \nabla_{\boldsymbol{\xi}}\mathbf{r}(\boldsymbol{\xi}_i)(\boldsymbol{\xi} - \boldsymbol{\xi}_i) \qquad \mathbf{J}_i := \nabla_{\boldsymbol{\xi}}\mathbf{r}(\boldsymbol{\xi}_i) \in \mathbb{R}^{\dim(\mathbf{r})\times\dim(\boldsymbol{\xi})}$$

$$\widetilde{E}(\boldsymbol{\xi}) = \frac{1}{2}\widetilde{\mathbf{r}}(\boldsymbol{\xi})^\top \mathbf{W}\widetilde{\mathbf{r}}(\boldsymbol{\xi})$$

$$\nabla_{\boldsymbol{\xi}}\widetilde{E}(\boldsymbol{\xi}) = \mathbf{J}_i^\top \mathbf{W}\widetilde{\mathbf{r}}(\boldsymbol{\xi})$$

$$\nabla_{\boldsymbol{\xi}}^2\widetilde{E}(\boldsymbol{\xi}) = \mathbf{J}_i^\top \mathbf{W}\mathbf{J}_i =: \mathbf{H}_i \in \mathbb{R}^{\dim(\boldsymbol{\xi})\times\dim(\boldsymbol{\xi})}$$

  - Find minimum of linearized system, linearize and set $\nabla_{\boldsymbol{\xi}}\widetilde{E}(\boldsymbol{\xi}) = \mathbf{0}$:

$$\nabla_{\boldsymbol{\xi}}\widetilde{E}(\boldsymbol{\xi}) \approx \nabla_{\boldsymbol{\xi}}\widetilde{E}(\boldsymbol{\xi}_i) + \nabla_{\boldsymbol{\xi}}^2\widetilde{E}(\boldsymbol{\xi}_i)(\boldsymbol{\xi} - \boldsymbol{\xi}_i)$$

$$\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i - \left(\nabla_{\boldsymbol{\xi}}^2\widetilde{E}(\boldsymbol{\xi}_i)\right)^{-1}\nabla_{\boldsymbol{\xi}}\widetilde{E}(\boldsymbol{\xi}_i) = \boldsymbol{\xi}_i - \mathbf{H}_i^{-1}\mathbf{J}_i^\top \mathbf{W}\mathbf{r}(\boldsymbol{\xi}_i)$$

Levenberg-Marquardt Method

- Due to linearization, $\mathbf{H}_i$ may not be a good approximation of the Hessian far from the optimum (could even be degenerate)
- Idea: „damping" of step-length trades-off between Gauss-Newton and gradient descent

$$\boldsymbol{\xi}_{i+1} = \boldsymbol{\xi}_i - (\mathbf{H}_i + \lambda\mathbf{I})^{-1}\mathbf{J}_i^\top \mathbf{W}\mathbf{r}(\boldsymbol{\xi}_i)$$

  - If error decreases, decrease λ to shift towards Gauss-Newton
  - If error increases, reject update and increase λ to rather perform gradient descent
  - Can converge from worse starting conditions than Gauss-Newton, but requires more iterations

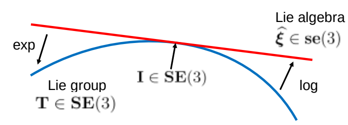
Pose Parametrization for Optimization

- Requirements on pose parametrization
  - No singularities
  - Minimal to avoid constraints
- Various pose parametrizations available
  - Direct matrix representation => not minimal
  - Quaternion / translation => not minimal
  - Euler angles / translation => singularities
  - Twist coordinates of elements in Lie Algebra se(3) of SE(3) (axis-angle / translation)

Topics of This Lecture

- Point-based Visual Odometry
  - Recap: 2D-to-2D Motion Estimation
  - 2D-to-3D Motion Estimation
  - 3D-to-3D Motion Estimation
  - Further Considerations
- Direct Methods
  - Direct image alignment
  - Pose parametrization
  - Lie group se(3) and the exponential map
  - Residual linearization
  - Optimization considerations

Representing Motion using Lie Algebra se(3)



$$\xi := \begin{pmatrix} \omega \\ \mathbf{v} \end{pmatrix} \in \mathbb{R}^6$$

$\omega \in \mathbb{R}^3$

$\mathbf{v} \in \mathbb{R}^3$

$$\hat{\xi} := \begin{pmatrix} \hat{\omega} & \mathbf{v} \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

- $\mathbf{SE}(3)$ is a smooth manifold, i.e. a Lie group
- Its Lie algebra $\mathbf{se}(3)$ provides an elegant way to parametrize poses for optimization
- Its elements $\hat{\xi} \in \mathbf{se}(3)$ form the tangent space of $\mathbf{SE}(3)$ at identity
- The $\mathbf{se}(3)$ elements can be interpreted as rotational and translational velocities (twists)

Insights into se(3)

- Let's look at rotations first and assume time-continuous motion
  - We know that $\mathbf{R}(t)\mathbf{R}^\top(t) = \mathbf{I}$
  - Taking the derivative for time yields $\dot{\mathbf{R}}(t)\mathbf{R}^\top(t) = -\mathbf{R}(t)\dot{\mathbf{R}}^\top(t)$
  - This means there exists a skew-symmetric matrix $\hat{\omega}(t) = -\hat{\omega}^\top(t)$ such that $\dot{\mathbf{R}}(t) = \hat{\omega}(t)\mathbf{R}(t)$
  - Assume constant $\hat{\omega}(t)$ and solve linear ordinary differential equation (ODE):
    $$\mathbf{R}(t) = \exp(\hat{\omega}t)\mathbf{R}(0)$$
  - Further assuming $\mathbf{R}(0) = \mathbf{I}$, we obtain
  - Matrix exponential has a closed-form solution; $\hat{\omega}t$ corresponds to minimal axis-angle representation

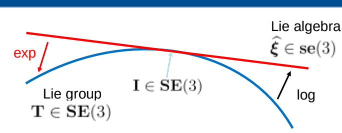
Further Insights into se(3)

- For continuous rigid-body motion we can write

$$\dot{\mathbf{T}}(t) = \left(\dot{\mathbf{T}}(t)\mathbf{T}^{-1}(t)\right)\mathbf{T}(t) = \hat{\xi}(t)\mathbf{T}(t) \qquad \hat{\xi}(t) := \begin{pmatrix} \hat{\omega}(t) & \mathbf{v}(t) \\ 0 & 0 \end{pmatrix}$$

- Interpretation: tangent vector along curve of $\mathbf{T}(t)$
- Again, for constant $\hat{\xi}(t)$ this linear ODE has a unique solution:
  $$\mathbf{T}(t) = \exp\left(\hat{\xi}t\right)\mathbf{T}(0)$$
- For initial condition $\mathbf{T}(0) = \mathbf{I}$, we have $\mathbf{T}(t) = \exp\left(\hat{\xi}t\right)$
- To reduce clutter in notation, we will absorb t into $\hat{\omega}$ and $\hat{\xi}$

Exponential Map of SE(3)

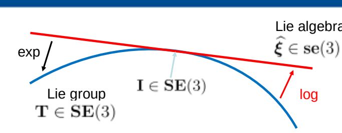


- The exponential map finds the transformation matrix for a twist:

$$\exp\left(\hat{\xi}\right) = \begin{pmatrix} \exp\left(\hat{\omega}\right) & \mathbf{A}\mathbf{v} \\ \mathbf{0} & 1 \end{pmatrix}$$

$$\exp\left(\hat{\omega}\right) = \mathbf{I} + \frac{\sin|\omega|}{|\omega|}\hat{\omega} + \frac{1-\cos|\omega|}{|\omega|^2}\hat{\omega}^2 \qquad \mathbf{A} = \mathbf{I} + \frac{1-\cos|\omega|}{|\omega|^2}\hat{\omega} + \frac{|\omega|-\sin|\omega|}{|\omega|^3}\hat{\omega}^2$$

Logarithm Map of SE(3)



- The logarithm maps twists to transformation matrices:

$$\log\left(\mathbf{T}\right) = \begin{pmatrix} \log\left(\mathbf{R}\right) & \mathbf{A}^{-1}\mathbf{t} \\ \mathbf{0} & 0 \end{pmatrix}$$

$$\log\left(\mathbf{R}\right) = \frac{|\omega|}{2\sin|\omega|}\left(\mathbf{R} - \mathbf{R}^T\right) \qquad |\omega| = \cos^{-1}\left(\frac{\operatorname{tr}\left(\mathbf{R}\right) - 1}{2}\right)$$

Some Notation for Twist Coordinates

- Let's define the following notation:
  - Inversion of hat operator: $\begin{pmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}^\vee = (\omega_1\ \omega_2\ \omega_3\ v_1\ v_2\ v_3)^\top$
  - Conversion: $\boldsymbol{\xi}(\mathbf{T}) = (\log(\mathbf{T}))^\vee, \quad \mathbf{T}(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}})$
  - Pose inversion: $\boldsymbol{\xi}^{-1} = \log(\mathbf{T}(\boldsymbol{\xi})^{-1}) = -\boldsymbol{\xi}$
  - Pose concatenation: $\boldsymbol{\xi}_1 \oplus \boldsymbol{\xi}_2 = (\log(\mathbf{T}(\boldsymbol{\xi}_2)\,\mathbf{T}(\boldsymbol{\xi}_1)))^\vee$
  - Pose difference: $\boldsymbol{\xi}_1 \ominus \boldsymbol{\xi}_2 = \left(\log\left(\mathbf{T}(\boldsymbol{\xi}_2)^{-1}\,\mathbf{T}(\boldsymbol{\xi}_1)\right)\right)^\vee$

Optimization with Twist Coordinates

- Twists provide a minimal local representation without singularities
- Since $\mathbf{SE}(3)$ is a smooth manifold, we can decompose transformations in each optimization step into the transformation itself and an infinitesimal increment

$$\mathbf{T}(\boldsymbol{\xi}) = \mathbf{T}(\boldsymbol{\xi})\exp\left(\widehat{\delta\boldsymbol{\xi}}\right) = \mathbf{T}(\delta\boldsymbol{\xi} \oplus \boldsymbol{\xi})$$

But!

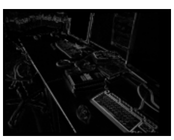
$$\mathbf{T}(\boldsymbol{\xi} + \delta\boldsymbol{\xi}) \neq \mathbf{T}(\boldsymbol{\xi})\,\mathbf{T}(\delta\boldsymbol{\xi})$$

- Example: Gradient descent on the auxiliary variable

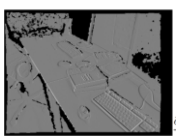
$$\delta\boldsymbol{\xi}^* = \mathbf{0} - \eta\nabla_{\delta\boldsymbol{\xi}} E(\boldsymbol{\xi}_i, \delta\boldsymbol{\xi})$$

$$\mathbf{T}(\boldsymbol{\xi}_{i+1}) = \mathbf{T}(\boldsymbol{\xi}_i)\exp\left(\widehat{\delta\boldsymbol{\xi}^*}\right)$$

Properties of Residual Linearization



I_1 - I_2



$$\left.\frac{\partial I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}}))}{\partial v_x}\right|_{\boldsymbol{\xi}=0}$$

- Linearizing residuals yields

$$\nabla_{\boldsymbol{\xi}} r(\mathbf{y}, \boldsymbol{\xi}) = -\nabla_\pi I_2(\omega(\mathbf{y}, \boldsymbol{\xi}))\ \nabla_{\boldsymbol{\xi}}\omega(\mathbf{y}, \boldsymbol{\xi})$$

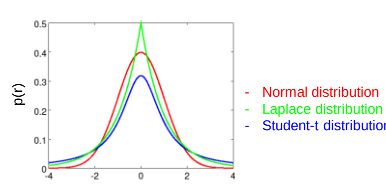
with $\omega(\mathbf{y}, \boldsymbol{\xi}) := \pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}})$

- Linearization is only valid for motions that change the projection in a small image neighborhood that is captured by the local gradient

Topics of This Lecture

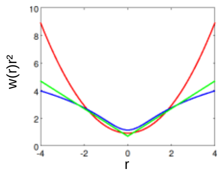
Coarse-To-Fine Optimization

Residual Distributions



- Gaussian noise assumption on photometric residuals oversimplifies
- Outliers (occlusions, motion, etc.):
  Residuals are distributed with more mass on the larger values

Optimizing Non-Gaussian Measurement Noise



- Normal distribution
- Laplace distribution
- Student-t distribution

- Can we change the residual distribution in least squares optimization?
- For specific types of distributions: yes!
- Iteratively reweighted least squares: Reweight residuals in each iteration

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{y}\in\Omega} w\left(r(\mathbf{y},\boldsymbol{\xi})\right)\frac{r(\mathbf{y},\boldsymbol{\xi})^2}{\sigma_I^2}$$

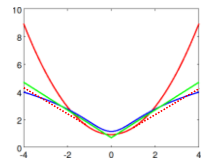
Laplace distribution:

$$w\left(r(\mathbf{y},\boldsymbol{\xi})\right) = |r(\mathbf{y},\boldsymbol{\xi})|^{-1}$$

Huber Loss

- Huber-loss „switches“ between Gaussian (locally at mean) and Laplace distribution

$$\|r\|_\delta = \begin{cases} \frac{1}{2}\|r\|_2^2 & \text{if } \|r\|_2 \leq \delta \\ \delta\left(\|r\|_1 - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases}$$



- Normal distribution
- Laplace distribution
- Student-t distribution

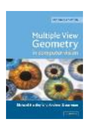
Huber-loss for δ = 1

References and Further Reading

- MASKS and MVG textbooks

MASKS

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



MVG

Multiple View Geometry in Computer Vision, R. Hartley and A. Zisserman, Cambridge University Press, 2004

- D. Scaramuzza / F. Fraundorfer, Visual Odometry: Part I & II, IEEE Robotics and Automation Magazine, 2011/2012