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

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
  - 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.

Recap: Direct vs. Indirect Methods

- **Direct methods**
  - Formulate alignment objective in terms of photometric error
  - Introduce linear algorithm: 8-point

- **Indirect methods**
  - Formulate alignment objective in terms of reprojection error of geometric primitives (e.g., points, lines)
  - Introduce linear algorithm: DLT PnP
  - Introduce linear algorithm: Ankin's method

Motion Estimation from Point Correspondences

- **2D-to-2D**
  - Reproj. error: $E(Y, X) = \sum_{i} |\hat{y}_{ij} - \hat{y}(X, Y)|$
  - Introduced linear algorithm: 8-point

- **2D-to-3D**
  - Reprojection error: $E(Y, X) = \sum_{i} |\hat{y}_{ij} - \hat{y}(X, Y)|$
  - Introduced linear algorithm: DLT PnP

- **3D-to-3D**
  - Reprojection error: $E(Y, X) = \sum_{i} |\hat{y}_{ij} - \hat{y}(X, Y)|$
  - Introduced linear algorithm: Ankin's method
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
     \[ y_i^T x = a_i^T x = 0 \]
     \[ A = (x_1, x_2, x_3) \]
     using Kronecker product \( a_i = y_i^T \otimes x_i \)
     and \( A = (a_1, a_2, a_3)^T \)
  2. Apply singular value decomposition (SVD) on \( A = U \Sigma V^T \)
     and unstack the 9th column of \( V \) into \( \hat{E} \).
  3. Project the approximate \( \hat{E} \) into the (normalized) essential space:
     Determine the SVD of \( \hat{E} = U \text{diag}(c_1, c_2, c_3) V^T \) with \( U, V \in \text{SO}(3) \)
     and replace the singular values \( c_1, c_2, c_3 \) with \( 1, 1, 0 \) to find
     \[ E = U \text{diag}(1, 1, 0) V^T \]

Recap: Eight-Point Algorithm cont.

- Normalized essential matrix: \( ||E|| = ||\hat{E}|| = 1 \)
- 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 exist (e.g., all points on a plane)
- No translation, ideally: \( ||\hat{E}|| = 0 \Rightarrow ||E|| = 0 \)
- But: for small translations, signal-to-noise ratio of image parallax may be problematic: „spurious“ pose estimate

Recap: Triangulation

- Goal: Reconstruct 3D point \( \tilde{x} = (x, y, z, w)^T \in \mathbb{P}^3 \) from 2D image observations \( \{y_1, \ldots, y_3\} \) for known camera poses \( \{T_1, \ldots, T_3\} \)
- Linear solution: Find 3D point such that reprojections equal its projections
  \[ R_i = \pi(T_i \tilde{x}) = \sum_{j=1}^{n} x_j \pi(T_j x_i) \]
  - Each image provides one constraint \( y_i^T \pi(T_i \tilde{x}) = 0 \)
  - Leads to system of linear equations \( A \tilde{x} = 0 \), 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)
  \[ \min_{\tilde{x}} \sum \left| y_i - R_i(T_i \tilde{x}) \right|^2 \]

Normalized Eight-Point Algorithm

- Hartley, In Defense of the Eight-Point Algorithm, PAMI 1997
  - Conditioning of \( A \) 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

Relative Scale Recovery

- Problem: Each subsequent frame-pair gives another solution for the reconstruction scale
- Solution: Triangulate overlapping points \( Y_{i-2}, Y_{i-1}, Y_i \) for current and last frame pair
  \[ X_i = X_{i-2} X_{i-1} \]
  - Rescale translation of current relative pose estimate to match the reconstruction scale with the distance ratio between corresponding point pairs
    \[ r_{ij} = \frac{|X_i - X_{i-2}|}{|X_i - X_{i-1}|} \]
  - Use mean or robust median over available pair ratios
### Algorithm: 2D-to-2D Visual Odometry

**Input:** image sequence $I_{st}$  
**Output:** aggregated camera poses $T_{st}$

**Algorithm:**
1. Extract and match keypoints between $I_s$ and $I_t$  
2. Compute relative pose $T_{st}^{-1}$ from Essential matrix between $I_s$ and $I_t$  
3. Compute relative scale and rescale translation of $T_{st}^{-1}$ accordingly  
4. Aggregate camera pose by $T_k = T_{k-1}T_{st}^{-1}$

---

### 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 $X = \{x_1, \ldots, x_N\}$ and corresponding image observations $Y = \{y_1, \ldots, y_N\}$, determine camera pose $T$, within the local map.
- Minimize least squares geometric reprojection error $E(T) = \sum_{i=1}^{N} ||y_i - \pi(T,x_i)||^2$
- **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]

- **Goal:** determine projection matrix $P = (R|t) \in \mathbb{R}^{3 \times 4}$  
- Each 2D-to-3D point correspondence $3D$: $x_i = (x_i, y_i, z_i)^T \in \mathbb{R}^3$  
  $2D$: $y_i = (y_{i1}, y_{i2}, 1)^T \in \mathbb{R}^3$ gives two constraints
  $\begin{bmatrix} 0 & -w_i x_i & -w_i y_i & 0 \\ w_i x_i & 0 & -w_i z_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P^1 \\ P^2 \\ P^3 \end{bmatrix} = 0$
  through $y_i \times (P^k_n) = 0$
- Form linear system of equation $A p = 0$ with $p = \begin{bmatrix} P_1^1 \\ P_2^1 \\ P_3^1 \end{bmatrix} \in \mathbb{R}^3$
- Solve for $p$, determine unit singular vector of $A$ corresponding to its smallest eigenvalue.

---

### Direct Linear Transform for PnP

- **Goal:** determine projection matrix $P = (R|t) \in \mathbb{R}^{3 \times 4}$
- Each 2D-to-3D point correspondence $3D$: $x_i = (x_i, y_i, z_i)^T \in \mathbb{R}^3$  
  $2D$: $y_i = (y_{i1}, y_{i2}, 1)^T \in \mathbb{R}^3$ gives two constraints
  $\begin{bmatrix} 0 & -w_i x_i & -w_i y_i & 0 \\ w_i x_i & 0 & -w_i z_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P^1 \\ P^2 \\ P^3 \end{bmatrix} = 0$
  through $y_i \times (P^k_n) = 0$
- Form linear system of equation $A p = 0$ with $p = \begin{bmatrix} P_1^1 \\ P_2^1 \\ P_3^1 \end{bmatrix} \in \mathbb{R}^3$
- Solve for $p$, determine unit singular vector of $A$ corresponding to its smallest eigenvalue.
3D-to-3D Motion Estimation

• Given 3D point coordinates of corresponding points in two camera frames
  \[ X_i = \{ x_{i,1}, \ldots, x_{i,3} \} \]
  \[ X'_i = \{ x'_{i,1}, \ldots, x'_{i,3} \} \]
  determine relative camera pose \[ T_{i+1}^{-1} \]

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

• Geometric least squares error: \[ E (T_{i+1}^{-1}) = \sum_{i=1}^{N} \| X_{i,3} - T_{i+1}^{-1} x_{i,3} \|^2 \]

• Closed-form solutions available, e.g., [Arun et al., 1987]

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, \( N \geq 3 \)
  \[ X_i = \{ x_{i,1}, \ldots, x_{i,3} \} \]
  \[ X'_i = \{ x'_{i,1}, \ldots, x'_{i,3} \} \]

• Determine means of 3D point sets
  \[ \mu_i = \frac{1}{N} \sum_{i=1}^{N} x_{i,j} \]
  \[ \mu'_i = \frac{1}{N} \sum_{i=1}^{N} x'_{i,j} \]

• Determine rotation from
  \[ A = \sum_{i=1}^{N} (x_{i,1} - \mu_{i,1}) (x_{i,1} - \mu'_{i,1})^\top \]
  \[ A = USV^\top \]
  \[ R_{i+1} = VU^\top \]

• Determine translation as \( t_{i+1} = \mu_i - R_{i+1} \mu'_{i+1} \)

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

Input: stereo image sequence \( I_{crf}, I'_{crf} \)
Output: aggregated camera poses \( T_{crf} \)

Algorithm:
For each current stereo image \( I'_i, I_{i-1} \):
1. Extract and match keypoints between \( I'_i \) and \( I_{i-1} \)
2. Triangulate 3D points \( X'_i \) between \( I'_i \) and \( I_{i-1} \)
3. Compute camera pose \( T^{-1}_{i} \) from 3D-to-3D point matches \( X'_i \) to \( X_{i-1} \)
4. Aggregate camera poses by \( T_i = T_{i-1} T^{-1}_{i} \)

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
• Blobs
  – Image regions that stick out from their surrounding in intensity/texture
  – Circular high-contrast regions
• Examples: Harris, FAST
• Scale-selection: Harris-Laplace

• Harris Corners
  – Image source: Svetlana Lazebnik

• Sten (SIFT) Blobs
  – Image source: Svetlana Lazebnik
Recap: Keypoint Detectors

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

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)

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

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

Recap: RANSAC

• Let's try 2 other random points

Recap: RANSAC

• Let's try yet another 2 random points

Recap: RANSAC

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

Recap: RANSAC

• RANdom SAmple Consensus algorithm formalizes this idea

### Algorithm:
- Input: data \( \mathcal{D} \), required data points for fitting, success probability \( p \), outlier ratio \( \rho \)
- Output: inlier set
  1. Compute required number of iterations
  2. For \( \mathcal{L} \) iterations do:
     1. Randomly select a subset of \( \mathcal{D} \) 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
  \[ N \text{ for } p = 0.99 \]

<table>
<thead>
<tr>
<th>Line</th>
<th>Required #points</th>
<th>Outlier ratio ( \epsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( 2 )</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>( 3 )</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>( 4 )</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>( 7 )</td>
<td>40%</td>
</tr>
<tr>
<td></td>
<td>( 11 )</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>( 17 )</td>
<td>60%</td>
</tr>
<tr>
<td></td>
<td>( 27 )</td>
<td>70%</td>
</tr>
</tbody>
</table>

- Plane
  \[ N \text{ for } p = 0.99 \]

<table>
<thead>
<tr>
<th>Essential matrix</th>
<th>Required #points</th>
<th>Outlier ratio ( \epsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( 8 )</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>( 9 )</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>( 26 )</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>( 3 )</td>
<td>40%</td>
</tr>
<tr>
<td></td>
<td>( 7 )</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>( 11 )</td>
<td>60%</td>
</tr>
<tr>
<td></td>
<td>( 17 )</td>
<td>70%</td>
</tr>
</tbody>
</table>

Probabilistic Modelling

- Model image point observation likelihood
  \[ p(y_i | x_i, \xi) \]
  - E.g., Gaussian:
  \[ p(y_i | x_i, \xi) \sim N(y; \pi(\xi(x_i)), \Sigma_y) \]

- Optimize maximum a-posteriori likelihood of estimates
  \[ p(x_i, \xi | Y) \propto p(Y | x_i, \xi) p(x_i | \xi) = p(x_i | \xi) \prod_{i} p(y_i | x_i, \xi) \]

- Neg. log-likelihood:
  \[ L(X, \xi, Y) = - \log(p(x_i | \xi)) \sum_{i} \log(p(y_i | x_i, \xi)) \]

- Gaussian prior and observation likelihood:
  \[ L(X, \xi, Y) = \text{const.} + \sum_{i} \left[ (x_i - \mu_x) \Sigma_x^{-1} (x_i - \mu_x) + (y_i - \pi(\xi(x_i))) \Sigma_y^{-1} (y_i - \pi(\xi(x_i))) \right] \]

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-vs. 2D: Low 3D triangulation accuracy for small baseline
- 2D-to-2D: 2x triangulation, typically less accurate than 2D-to-3D

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

<table>
<thead>
<tr>
<th>Correspondences</th>
<th>Monocular</th>
<th>Stereo</th>
<th>RGB-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D-to-2D</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2D-to-3D</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3D-to-3D</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Local Optimization Windows

- Can we do better than optimization over two images?

- Optimize motion / reconstruction on a local current window of images
  \[ E(N_{1:k}, \xi_{1:k}) = \sum_{k} \sum_{i} |y_{i,j} - \pi(\xi_{i,j})|_2^2 \]
  - Local bundle adjustment
  - Local motion-only bundle adjustment
    (3D keypoint positions held fixed)
  - Initialize with algebraic approaches
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
- Warping requires depth
  - RGB-D
  - Fixed-baseline stereo
  - Temporal stereo, tracking and (local) mapping

Direct Visual Odometry Example (RGB-D)

Robust Odometry Estimation for RGB-D Cameras

Christian Kiel, Jürgen Sturm, Daniel Cremers

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:

  \[ I_1(y) = I_2(\pi(\mathbf{T}(\xi)z, y)) \]
• RGB-D cameras measure depth, we only need to estimate camera motion!
• In addition to the photometric error
\[ I_1(y) = I_2(\pi(T(\xi)Z(y)y)) \]
we can measure geometric error directly
\[ |T(\xi)|Z(y)y| = Z_2(\pi(T(\xi)Z(y)y)) \]

Optimization Approach

• Optimize negative log-likelihood
  – Product of exponentials becomes a summation over quadratic terms
  – Normalizers are independent of the pose
\[ E(\xi) = \sum_{y \in \Omega} \frac{r(y, \xi)^2}{\sigma^2} \]
  stacked residuals: \[ r(\xi) = r(\xi)^{\top} W r(\xi) \]
\[ r(y, \xi) = I_1(y) - I_2(\pi(T(\xi)Z(y)y)) \]

• 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:
\[ \frac{r(\xi)}{\sigma} = \frac{1}{2} r(\xi)^{\top} W r(\xi) \]
\[ \nabla_r(\xi) = J^\top W r(\xi) \]
\[ \nabla^2_r(\xi) = J^\top W J \]
  – Find minimum of linearized system, linearize and set \[ \nabla^2 r(\xi) = 0 \]:
\[ \nabla^2 r(\xi) = \nabla^2_r(\xi) = \nabla^2_r(\xi) = \xi - H^\top W J \]

Pose Parametrization for Optimization

• Requirements on pose parametrization
  – No singularities
  – Minimal to avoid constraints
• Various pose parametrizations available
  – Direct matrix representation \( \Rightarrow \) not minimal
  – Quaternion / translation \( \Rightarrow \) not minimal
  – Euler angles / translation \( \Rightarrow \) 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)

- $\xi := \begin{pmatrix} \dot{\omega} \\ v \end{pmatrix} \in \mathbb{R}^d$
- $\omega \in \mathbb{R}^3$
- $v \in \mathbb{R}^3$
- $\xi := \begin{pmatrix} \dot{\omega} \\ 0 \end{pmatrix} \in \mathbb{R}^{d+1}$

- $SE(3)$ is a smooth manifold, i.e. a Lie group
- Its Lie algebra $se(3)$ provides an elegant way to parametrize poses for optimization
- Its elements $\xi \in se(3)$ form the tangent space of $SE(3)$ at identity
- The $se(3)$ 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 $R(t)R'(t) = I$
  - Taking the derivative for time yields $R(t)R'(t) = -R(t)R'(t)$
  - This means there exists a skew-symmetric matrix $\Omega(t) = -\dot{R}(t)$
  - Assume constant $\Omega(t)$ and solve linear ordinary differential equation (ODE): $R(t) = \exp(\Omega(t))$
  - Further assuming $R(0) = I$, we obtain
  - Matrix exponential has a closed-form solution; $\Omega(t)$ corresponds to minimal axis-angle representation

Further Insights into se(3)

- For continuous rigid-body motion we can write $T(t) = (T(t)T^{-1}(t))T(0) = \xi(t) T(t)$
  - Interpretation: tangent vector along curve of $T(t)$
  - Again, for constant $\xi(t)$ this linear ODE has a unique solution: $T(t) = \exp(\xi(t))$
  - For initial condition $T(0) = I$, we have $T(t) = \exp(\xi(t))$
  - To reduce clutter in notation, we will absorb $t$ into $\xi$ and $\dot{\xi}$

Exponential Map of SE(3)

- The exponential map finds the transformation matrix for a twist:
  - $\exp(\xi) = \begin{pmatrix} \exp(\omega)A & \omega \times \text{A} \\ 0 & 1 \end{pmatrix}$
  - $\exp(\xi) = I + \frac{1}{2} \sin |\xi| |R - R^T|$
  - $A = I + \frac{1}{2} \cos |\xi| - \frac{1}{2} |\xi| - \sin |\xi| \omega \times R$

Logarithm Map of SE(3)

- The logarithm maps twists to transformation matrices:
  - $\log(T) = \begin{pmatrix} \log(R) & -A^{-1} \omega \\ 0 & 0 \end{pmatrix}$
  - $\log(R) = \frac{|\omega|}{2 \sin |\omega|} (R - R^T)$
  - $|\omega| = \cos^{-1}(\frac{\text{tr}(R) - 1}{2})$
Some Notation for Twist Coordinates

- Let’s define the following notation:
  - Inversion of hat operator:
    \[ \hat{O}(T) = \log(T) = (\omega_1 \omega_2 \omega_3 \mathbf{v})^T \]
  - Conversion:
    \[ \xi(T) = (\log(T))^T T(\xi) = \exp(\tilde{\xi}) \]
  - Pose inversion:
    \[ \xi^{-1} = \log(T(\xi)^{-1}) = -\xi \]
  - Pose concatenation:
    \[ \xi \otimes \xi_2 = (\log(T(\xi_2)) T(\xi_1)))^T \]
  - Pose difference:
    \[ \xi_1 \otimes \xi_2 = (\log(T(\xi_2)^{-1} T(\xi_1)))^T \]

Optimization with Twist Coordinates

- Twists provide a minimal local representation without singularities
- Since \( \text{SE}(3) \) is a smooth manifold, we can decompose transformations in each optimization step into the transformation itself and an infinitesimal increment:
  \[ T(\xi) = T(\xi) \exp(\delta\xi) = T(\xi + \delta\xi) \]
- Example: Gradient descent on the auxiliary variable:
  \[ \delta\xi^* = -\gamma \nabla_{\xi} E(\xi, \delta\xi) \]
  \[ T(\xi_{i+1}) = T(\xi_i) \exp(\delta\xi^*) \]

Properties of Residual Linearization

- Linearizing residuals yields:
  \[ \nabla E(\xi, y) = -\nabla_y f_2(\omega(y, \xi)) \nabla_c \omega(y, \xi) \]
  with \( \omega(y, \xi) := \pi(T(\xi) Z(y, y)) \)
  - 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

- 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

Coarse-To-Fine Optimization

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

Residual Distributions

- 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

\[ E(\xi) = \sum_{y \in \mathcal{Y}} w(\gamma(y, \xi)) \frac{|r(y, \xi)|^2}{\sigma^2} \]

Laplace distribution:
- \[ w(\gamma(y, \xi)) = |r(y, \xi)|^{-1} \]

Huber Loss
- Huber-loss "switches" between Gaussian (locally at mean) and Laplace distribution

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

References and Further Reading
- MASKS and MVG textbooks