Robotic 3D Vision

Lecture 5: Visual Odometry 1 – Introduction, Indirect Methods

Prof. Dr. Jörg Stückler
Computer Vision Group, TU Munich
http://vision.in.tum.de

What We Will Cover Today

- Lie algebra se(3)
- Introduction to and definition of visual odometry
- Indirect vs. direct methods
- Indirect methods
  - 2D-to-2D motion estimation
  - 2D-to-2D monocular visual odometry

Recap: 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 $\widetilde{\mathbf{x}} = \widetilde{w}\bar{\mathbf{x}}$	$\widetilde{\mathbf{x}} = \begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{w} \end{pmatrix} \in \mathbb{P}^2$	$\widetilde{\mathbf{x}} = \begin{pmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{z} \\ \widetilde{w} \end{pmatrix} \in \mathbb{P}^3$

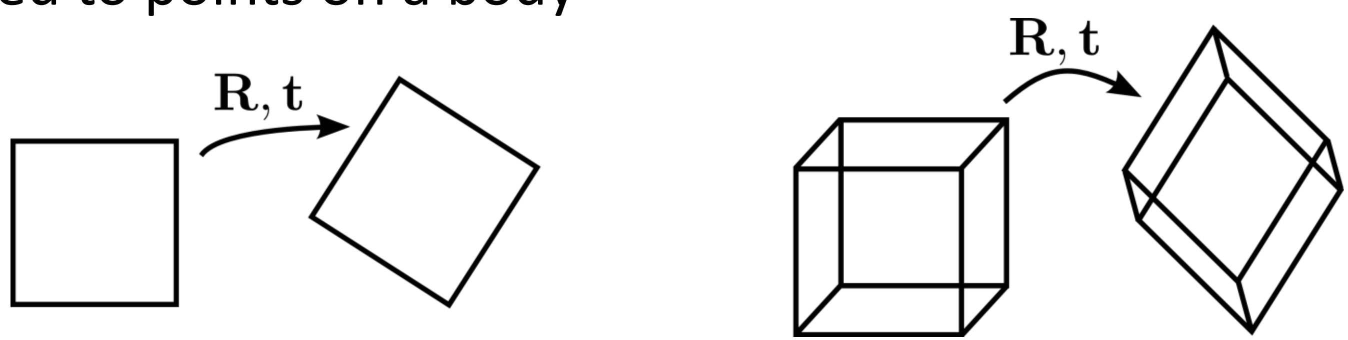
Recap: Euclidean Transformations

- Euclidean transformations apply rotation $\mathbf{R} \in \mathbf{SO}(n) \subset \mathbb{R}^{n \times n}$ and translation $\mathbf{t} \in \mathbb{R}^n$

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

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

$n = 2$



$n = 3$

Recap: 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 which we denote as Special Orthogonal Group $\mathbf{SO}(n)$
  - 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

Recap: Special Euclidean Group SE(n)

- Euclidean transformation matrices have a special structure as well:

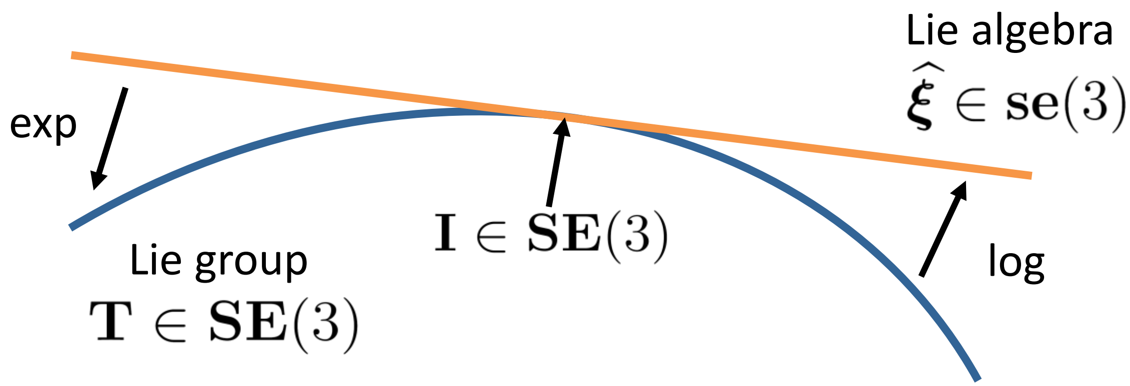
$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbf{SE}(3) \subset \mathbb{R}^{4\times 4}$$

  - 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 denote as Special Euclidean Group $\mathbf{SE}(3)$. The group operator is matrix multiplication:

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

Representing Motion using Lie Algebra se(3)



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

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

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

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

- $\mathbf{SE}(3)$ is a Lie group, i.e. a smooth manifold with compatible operator, inverse and neutral element
- Its Lie algebra $\mathbf{se}(3)$ provides an elegant way to parametrize poses for optimization
- Its elements $\hat{\boldsymbol{\xi}} \in \mathbf{se}(3)$ 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 $\widehat{\boldsymbol{\omega}}(t) = -\widehat{\boldsymbol{\omega}}^\top(t)$ such that $\dot{\mathbf{R}}(t) = \widehat{\boldsymbol{\omega}}(t)\mathbf{R}(t)$
  - Assume constant $\widehat{\boldsymbol{\omega}}(t)$ and $\mathbf{R}(0) = \mathbf{I}$
  - We write $\mathbf{R}(t+\delta t) = \mathbf{R}(t) + \delta t \widehat{\boldsymbol{\omega}}\mathbf{R}(t) = (\mathbf{I} + \delta t \widehat{\boldsymbol{\omega}})\mathbf{R}(t)$ for infinitesimal δt
  - Hence, we can write $\mathbf{R}(t) = \lim_{n\to\infty}(\mathbf{I} + \frac{t}{n}\widehat{\boldsymbol{\omega}})^n$
  - This series yields the matrix exponential $\mathbf{R}(t) = \exp(\widehat{\boldsymbol{\omega}} t)$
  - Matrix exponential has closed-form solution (Rodriguez formula)
  - $\widehat{\boldsymbol{\omega}} t$ corresponds to minimal axis-angle representation

Further Insights into se(3)

- For continuous rigid-body motion we can write

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

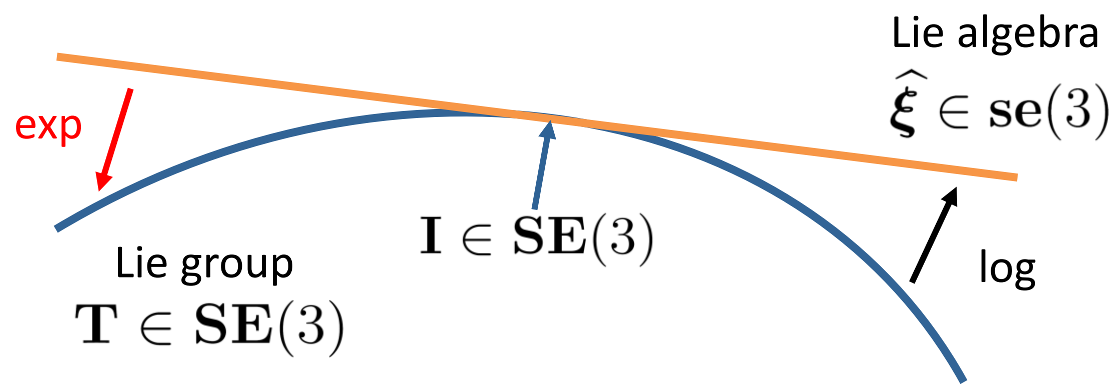
- For constant $\widehat{\boldsymbol{\xi}}(t)$ the differential equation has a unique solution:

$$\mathbf{T}(t) = \exp\left(\widehat{\boldsymbol{\xi}}t\right)\mathbf{T}(0)$$

- For initial condition $\mathbf{T}(0) = \mathbf{I}$, we have $\mathbf{T}(t) = \exp\left(\widehat{\boldsymbol{\xi}}t\right)$

- To reduce clutter in notation, we will absorb t into $\widehat{\boldsymbol{\omega}}$ and $\widehat{\boldsymbol{\xi}}$

Exponential Map of SE(3)

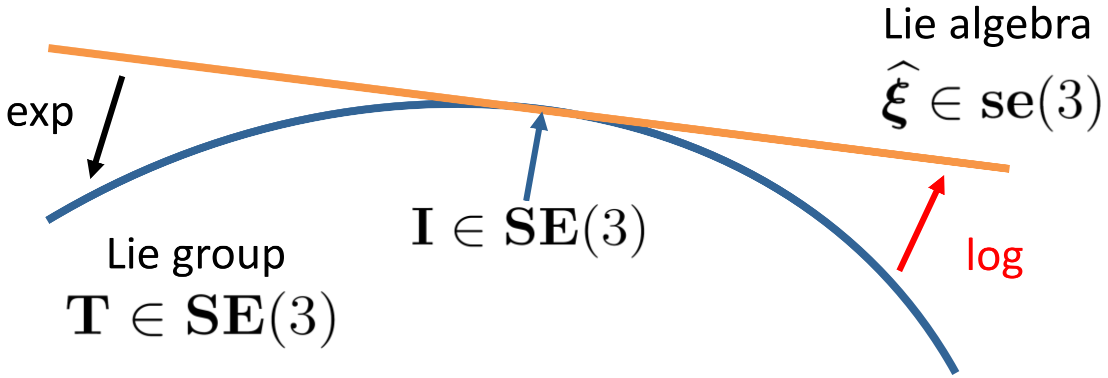


- The exponential map finds transformation matrices for twists:

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

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

Logarithm Map of SE(3)



- The logarithm map finds twists for transformation matrices:

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

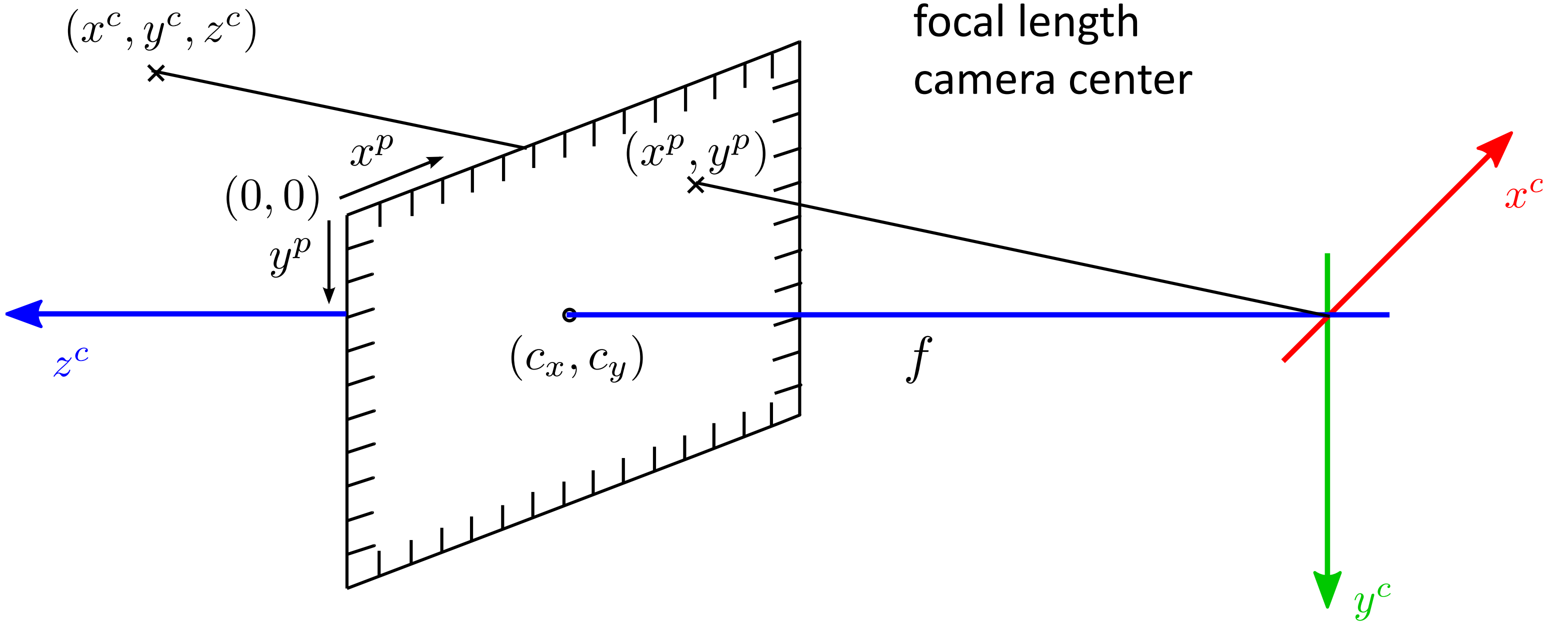
$$\log\left(\mathbf{R}\right) = \frac{|\omega|}{2\sin|\omega|}\left(\mathbf{R} - \mathbf{R}^T\right) \qquad |\omega| = \cos^{-1}\left(\frac{\operatorname{tr}\left(\mathbf{R}\right) - 1}{2}\right)$$

Some Notation for Twist Coordinates

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

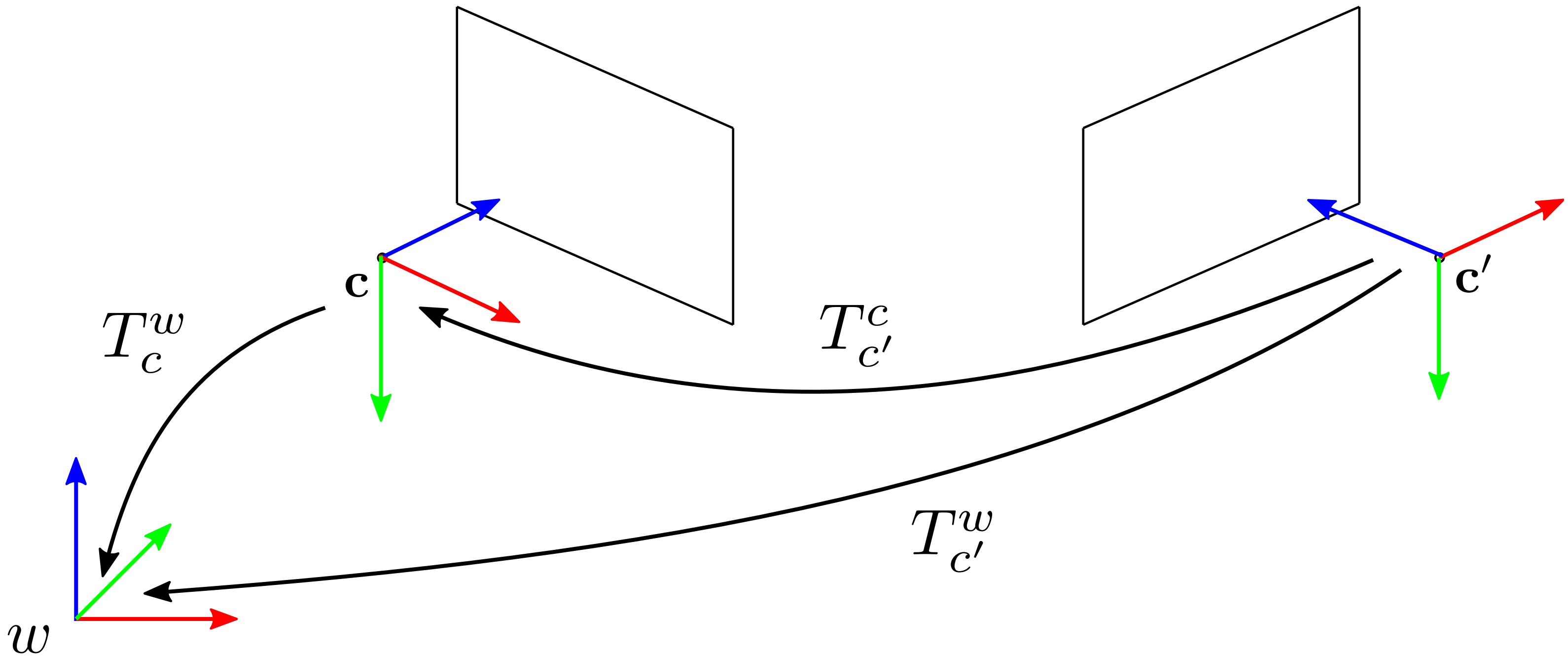
Recap: Pinhole Camera Model

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



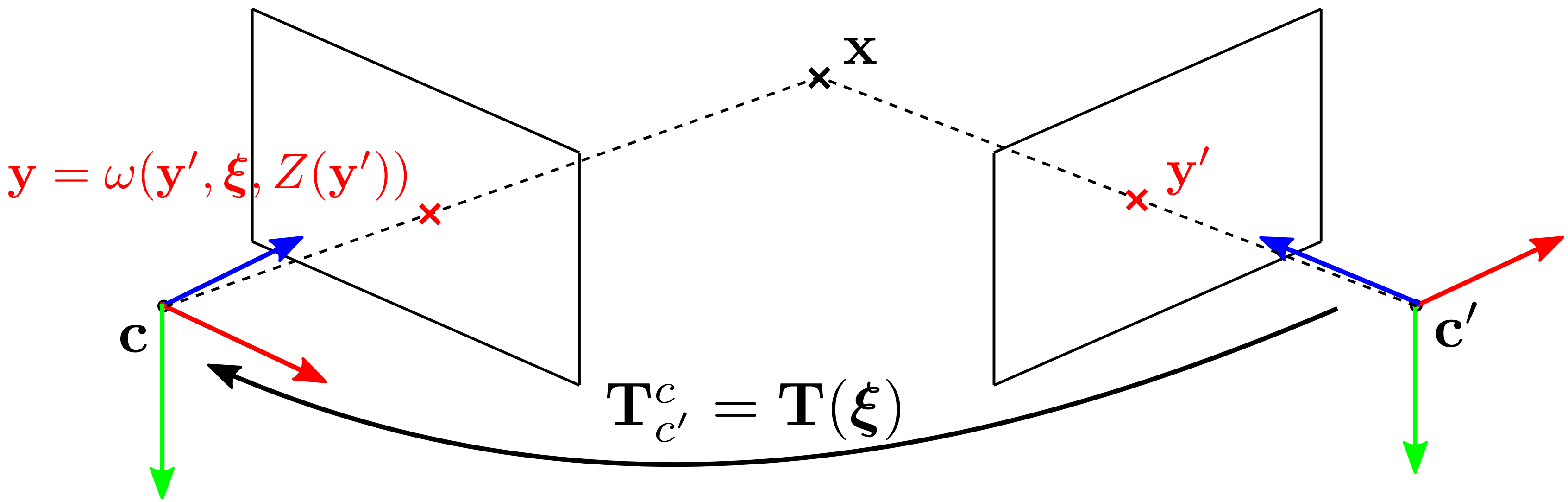
$$\begin{pmatrix} x^p \\ y^p \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{camera matrix } C} \underbrace{\begin{pmatrix} x^c/z^c \\ y^c/z^c \\ 1 \end{pmatrix}}_{=: \pi(\bar{\mathbf{x}}) = \bar{\mathbf{y}} \text{ (normalized image coordinates)}}$$

Recap: Camera Extrinsics



- Euclidean transformations ($T_c^w, T_{c'}^w, T_{c'}^c$) between camera view poses and world frame

Warping Function



- Normalized image coordinates:

$$\omega(\mathbf{y}', \boldsymbol{\xi}, Z(\mathbf{y}')) = \pi(\mathbf{T}(\boldsymbol{\xi})\overline{Z(\mathbf{y}')\overline{\mathbf{y}}'})$$

- Pixel coordinates:

$$\omega(\mathbf{y}'_p, \boldsymbol{\xi}, Z(\mathbf{y}'_p)) = \mathbf{C}\pi(\mathbf{T}(\boldsymbol{\xi})\overline{Z(\mathbf{y}'_p)\mathbf{C}^{-1}\overline{\mathbf{y}}'_p})$$

Recap: What is Visual Odometry?

Visual odometry (VO)…

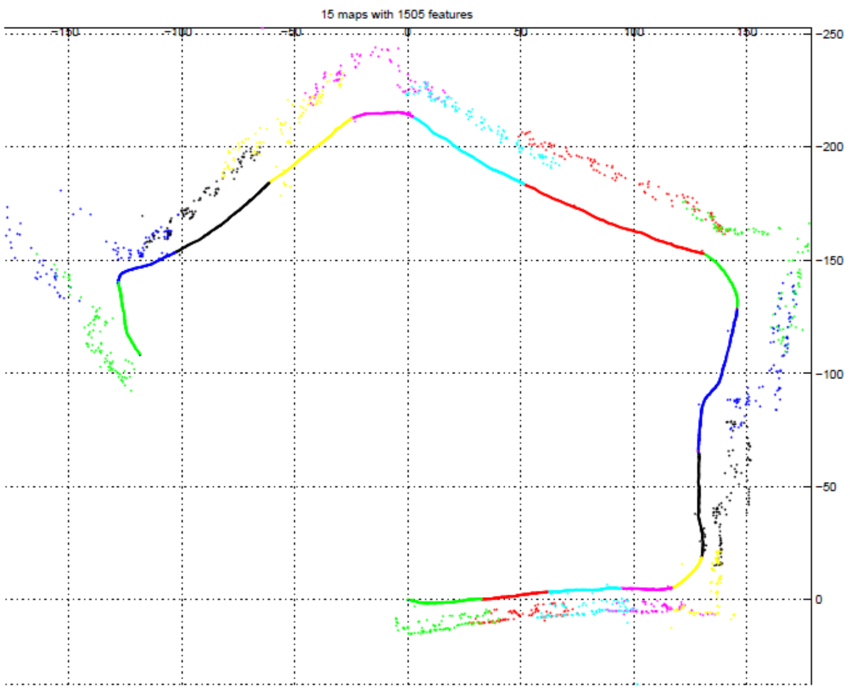
- … is a variant of tracking
  - Track the current pose, i.e. position and orientation, of the camera with respect to the environment from its images
  - Only considers a limited set of recent images for real-time constraints

- … involves a data association problem
  - Motion is estimated from corresponding interest points or pixels in images, or by correspondences towards a local 3D reconstruction

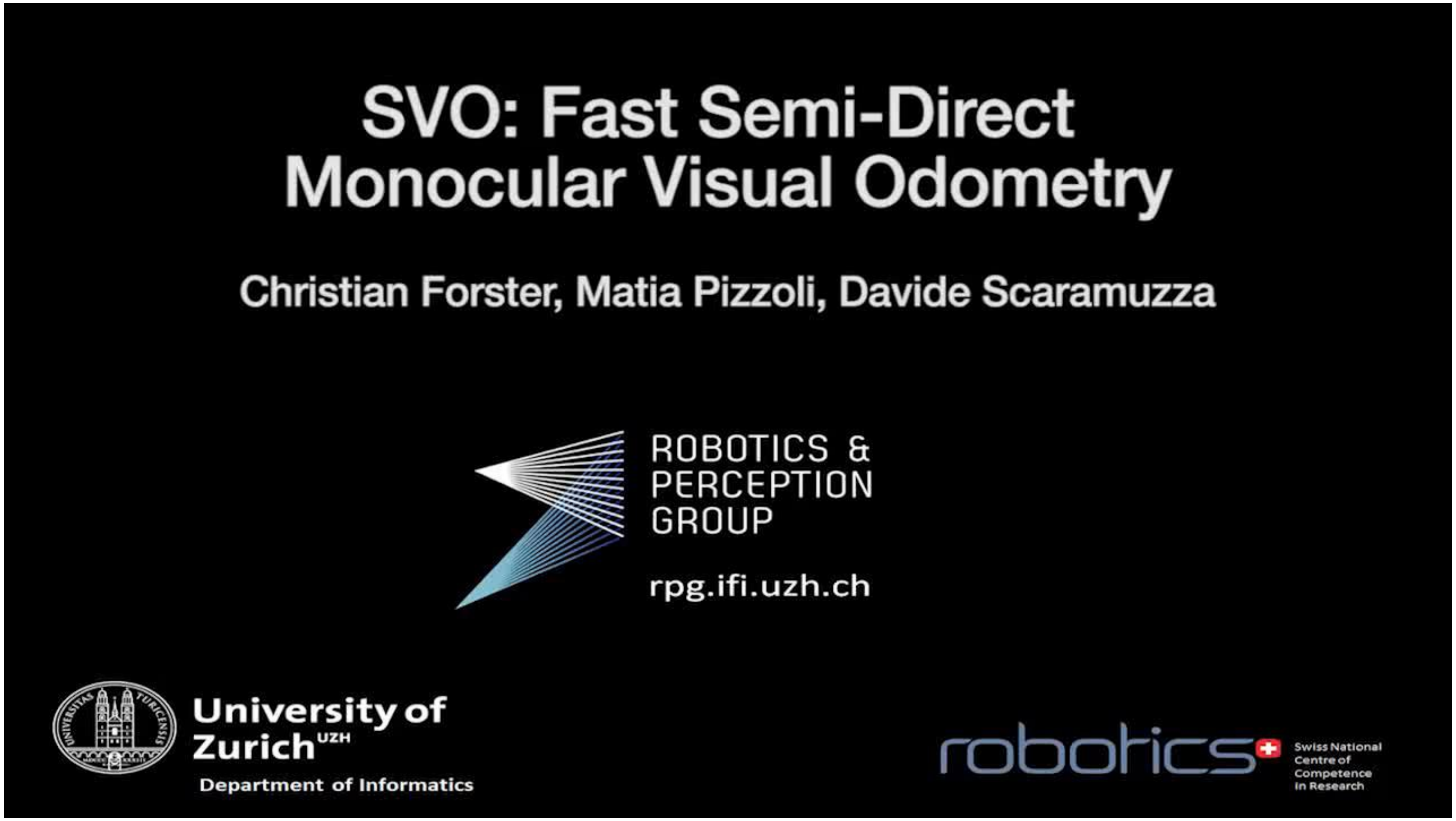
Recap: What is Visual Odometry?

Visual odometry (VO)…

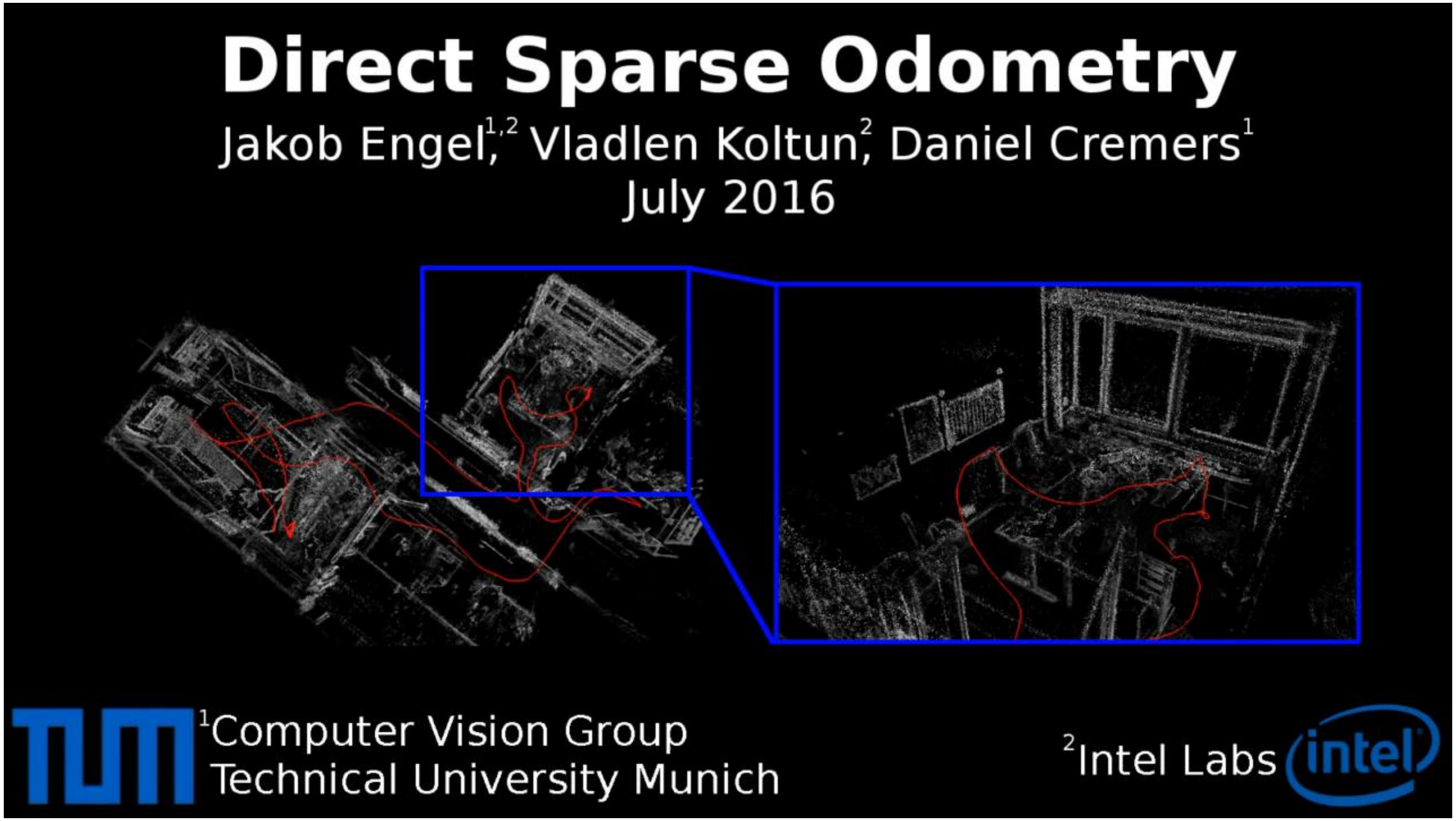
- … is prone to drift due to its local view
- … is primarily concerned with estimating camera motion
  - 3D reconstruction often a "side product". If estimated, it is only locally consistent



Visual Odometry Example



Visual Odometry Example



Notion of Visual Odometry

- **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

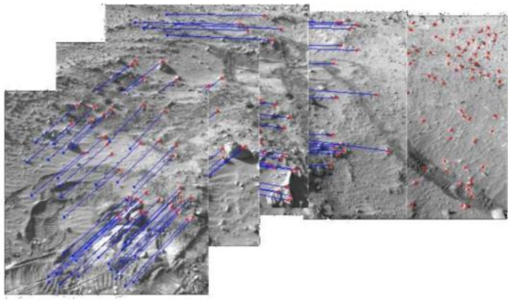


Image source: NASA, [Cheng et al., RAM, 2006]

Why Visual Odometry?

- VO is often used to complement other motion sensors
  - GPS
  - Inertial Measurement Units (IMUs)
  - Wheel odometry
  - etc.
- VO typically is more accurate than wheel odometry and not prone to wheel slippage
- VO is important in GPS-denied environments (indoors, close to buildings, etc.)

Sensors 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, resolution, synchronization and extrinsic calibration of the cameras

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

Image source: IDS, PointGrey, ASUS

Definition of Visual Odometry

- Visual odometry is the process of estimating the egomotion of an object (robot) using visual inputs from cameras on the object (robot)

- **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 $I^l_{0:t} = \{I^l_0, \ldots, I^l_t\}$ / $I^r_{0:t} = \{I^r_0, \ldots, I^r_t\}$
  - RGB-D case: Color/depth images $I_{0:t} = \{I_0, \ldots, I_t\}$ / $Z_{0:t} = \{Z_0, \ldots, Z_t\}$

- **Output**: Transformation estimate $\mathbf{T}_t \in \mathbf{SE(3)}$ of camera frame to world frame

- Camera pose integrated up from relative pose estimates
- Example: camera pose $\mathbf{T}_t = \mathbf{T}_0 \mathbf{T}_1^0 \cdots \mathbf{T}_t^{t-1}$ from frame-to-frame transformations $\mathbf{T}_t^{t-1}$

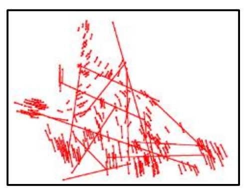
Recap: Indirect vs. Direct Methods

Indirect

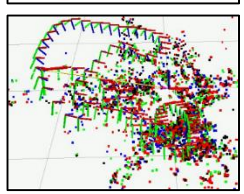
Input images

Extract and match keypoints (SIFT,BRIEF,…)

Track: min. reprojection error (point distances)



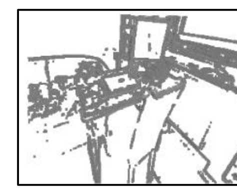
Map: estimate keypoint parameters (f.e. 3D coordinates)



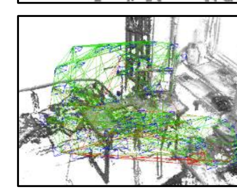
Direct

Input images

Track: min. photometric/ geometric error pixel-wise



Map: estimate per-pixel depth from photoconsistency



Indirect vs. Direct VO Methods

- Direct visual odometry methods formulate alignment objective in terms of pixel-wise error (e.g. photometric or geometric error)
  - Two-view case with known depth:

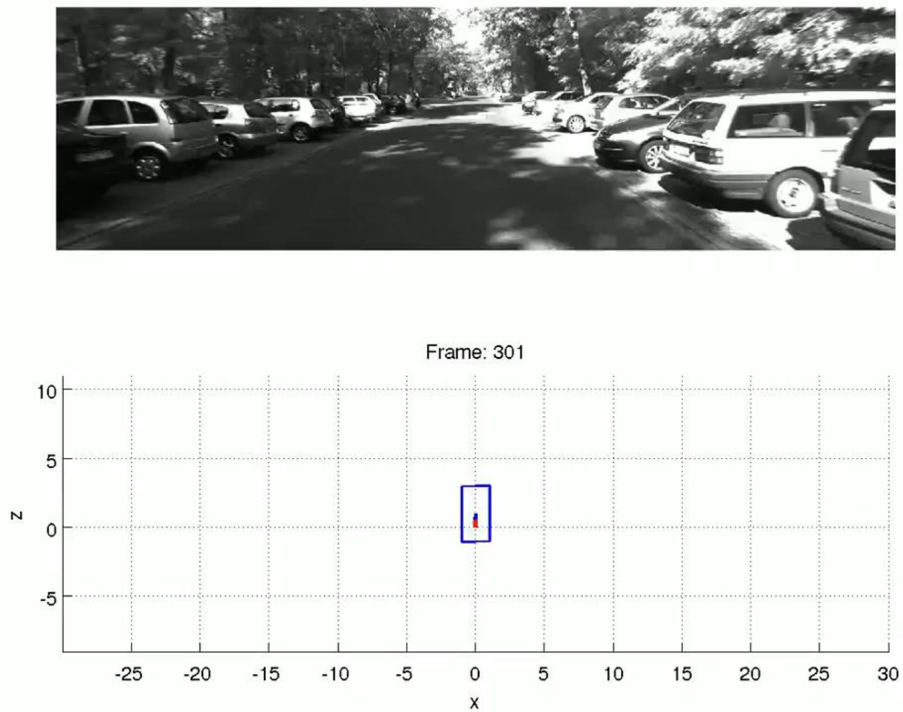
$$p(I_2 \mid I_1, Z_1, \boldsymbol{\xi}) \longrightarrow E(\boldsymbol{\xi}) = \int_{\mathbf{y} \in \boldsymbol{\Omega}} |I_1(\mathbf{y}) - I_2(\omega(\mathbf{y}, \boldsymbol{\xi}, Z_1(\mathbf{y})))| d\mathbf{y}$$

- Indirect visual odometry methods formulate alignment objective in terms of reprojection error of geometric primitives (e.g. points, lines)
  - Two-view case with known depth:

$$p(\mathcal{Y}_2 \mid \mathcal{Y}_1, Z_1, \boldsymbol{\xi}) \longrightarrow E(\boldsymbol{\xi}) = \sum_i |\mathbf{y}_{2,i} - \omega(\mathbf{y}_{1,i}, \boldsymbol{\xi}, Z_1(\mathbf{y}_{1,i}))|$$

  - $\mathcal{Y}_1, \mathcal{Y}_2$: sets of primitives (e.g. keypoints) in image 1 and 2

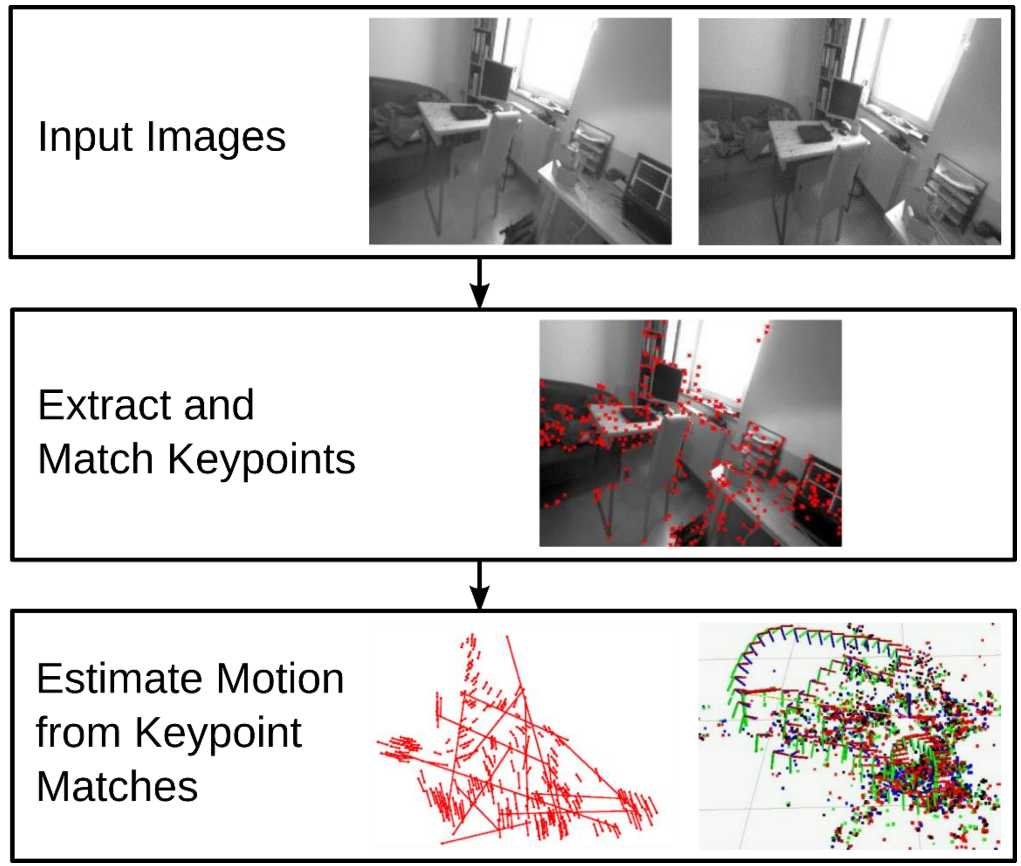
Indirect Visual Odometry Example



LibVISO2, Geiger et al., StereoScan: Dense 3D Reconstruction in Real-time, IV 2011

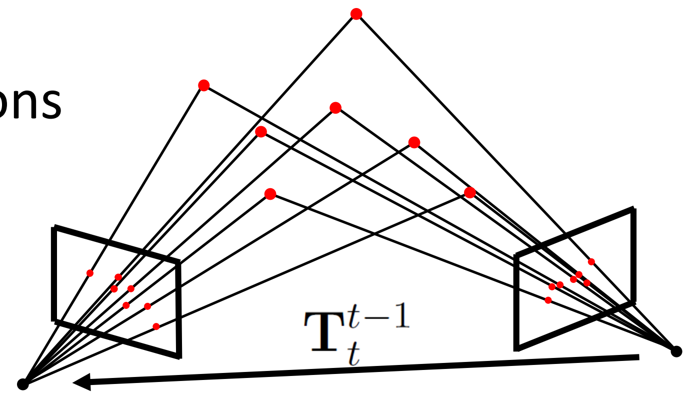
Indirect Visual Odometry Pipeline

- Keypoint detection and local description
- Robust keypoint matching
- Motion estimation
  - 2D-to-2D: motion from image correspondences
  - 2D-to-3D: motion from image to local 3D correspondences
  - 3D-to-3D: motion from local 3D correspondences (f.e. stereo, RGB-D)



Images from Jakob Engel

2D-to-2D Motion Estimation



- Given corresponding image point observations

  $$\mathcal{Y}_t = \{\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,N}\}$$
  $$\mathcal{Y}_{t-1} = \{\mathbf{y}_{t-1,1}, \ldots, \mathbf{y}_{t-1,N}\}$$

  of unknown 3D points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  (expressed in camera frame at time t)
  determine relative motion $\mathbf{T}_t^{t-1}$ between frames

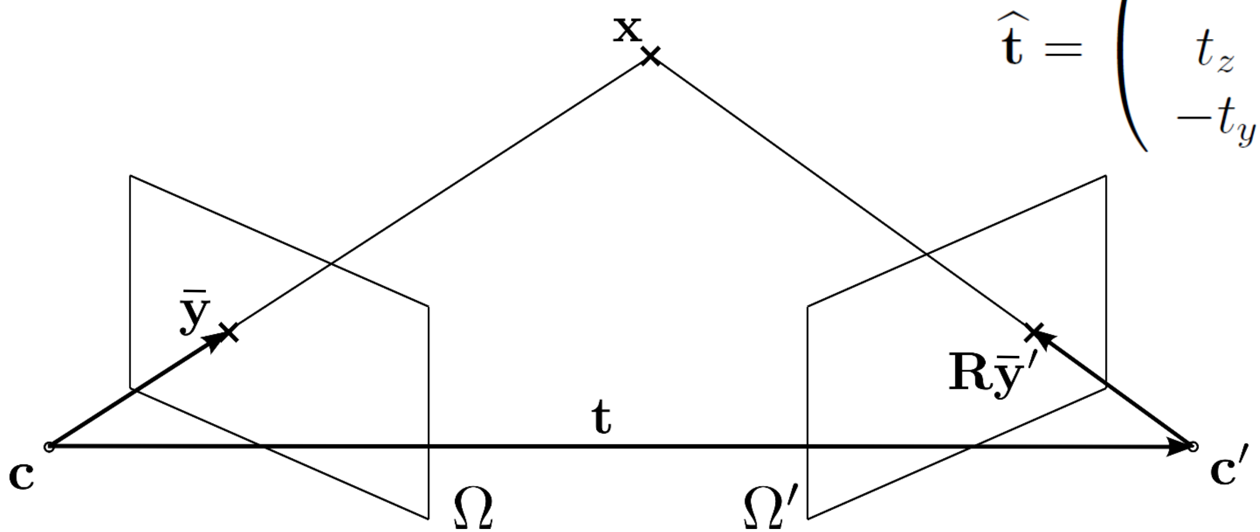
- Naive try: minimize reprojection error using least squares

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

- Convexity? Uniqueness (scale-ambiguity)?

- Alternative algebraic approach

Recap: Essential Matrix



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

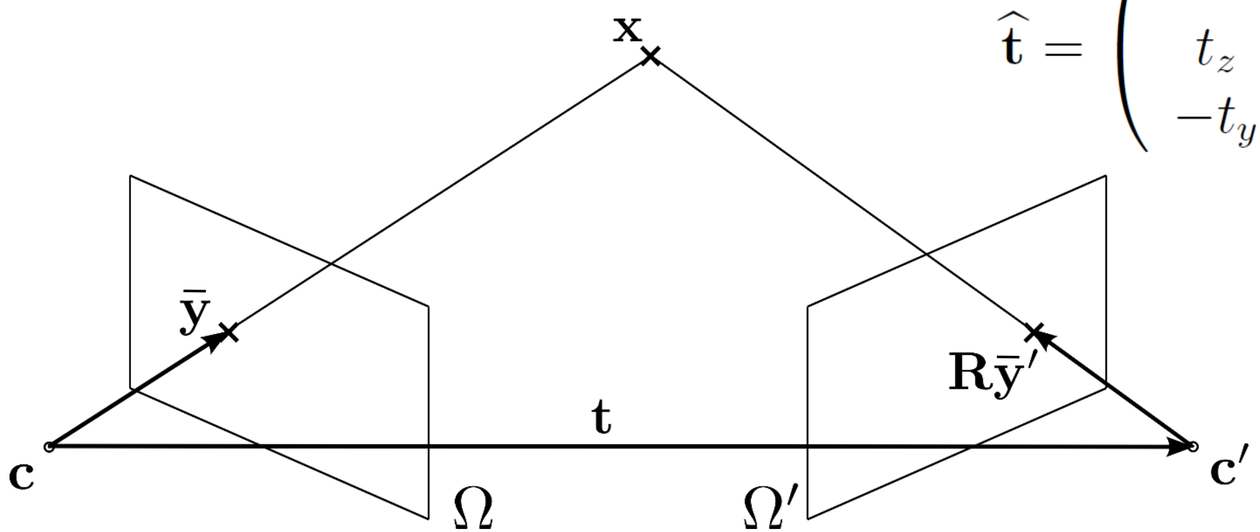
$$\widehat{\mathbf{t}} = [\mathbf{t}]_\times$$

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

$$\widetilde{\mathbf{y}}^\top (\mathbf{t} \times \mathbf{R}\widetilde{\mathbf{y}}') = 0 \Leftrightarrow \widetilde{\mathbf{y}}^\top \widehat{\mathbf{t}}\mathbf{R}\widetilde{\mathbf{y}}' = 0$$

- The essential matrix $\mathbf{E} := \widehat{\mathbf{t}}\mathbf{R}$ captures the relative camera pose
- Each point correspondence provides an „epipolar constraint“
- 5 correspondences suffice to determine $\mathbf{E}$ (simpler: 8-point algorithm)

Fundamental Matrix



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

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

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

$$\widetilde{\mathbf{y}}_p^\top \mathbf{C}^{-\top} \widehat{\mathbf{t}} \mathbf{R} \mathbf{C}^{-1} \widetilde{\mathbf{y}}_p' = \widetilde{\mathbf{y}}_p^\top \mathbf{F} \widetilde{\mathbf{y}}_p' = 0$$

- The fundamental matrix $\mathbf{F} := \mathbf{C}^{-\top} \widehat{\mathbf{t}} \mathbf{R} \mathbf{C}^{-1}$ captures the relative camera pose and camera intrinsics
- Each point correspondence provides an „epipolar constraint“
- Can be estimated from at least 7 point correspondences

Some Properties of E and F

- $\mathbf{F} \in \mathbb{R}^{3\times3}$ is a fundamental matrix iff $\mathrm{rank}(\mathbf{F}) = 2$

- $\mathbf{E} \in \mathbb{R}^{3\times3}$ is an essential matrix iff $\mathrm{rank}(\mathbf{E}) = 2$ and its non-zero singular values are equal

- $\mathbf{E} \in \mathbb{R}^{3\times3}$ is a normalized esssential matrix iff $\mathrm{rank}(\mathbf{E}) = 2$ and its non-zero singular values are 1

$$\|\mathbf{E}\| = \left\|\widehat{\mathbf{t}}\right\| = 1$$

- (Normalized) essential space: set of all (normalized) essential matrices

Eight-Point Algorithm

- First proposed by Longuet and Higgins, Nature 1981
- Algorithm:

1. Rewrite epipolar constraints as a linear system of equations

$$\widetilde{\mathbf{y}}_i^\top \mathbf{E} \widetilde{\mathbf{y}}_i' = \mathbf{a}_i \mathbf{E}_s = 0 \quad \longrightarrow \quad \mathbf{A}\mathbf{E}_s = \mathbf{0} \qquad \mathbf{A} = \left(\mathbf{a}_1^\top, \ldots, \mathbf{a}_N^\top\right)^\top$$

   using Kronecker product $\mathbf{a}_i = \widetilde{\mathbf{y}}_i \otimes \widetilde{\mathbf{y}}_i'$ 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 $\widetilde{\mathbf{E}}$

3. Project the approximate $\widetilde{\mathbf{E}}$ into the (normalized) essential space: Determine the SVD of $\widetilde{\mathbf{E}} = \mathbf{U} \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3) \mathbf{V}^\top$ 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 4 possible solutions that intersects the points in front of both cameras:

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

- A derivation of the eight-point algorithm can be found in the MASKS textbook, Ch. 5
- Algebraic solution does not minimize reprojection error
- Refine using non-linear least-squares of reprojection error

Error Metric of the Eight-Point Algorithm

- What is the physical meaning of the error minimized by the eight-point algorithm?
- The eight-point algorithm finds E that minimizes

$$\operatorname{argmin}_{\mathbf{E}_s} \|\mathbf{A}\mathbf{E}_s\|_2^2$$

  subject to $\|\mathbf{E}_s\|_2^2 = 1$ through the SVD on A
- We find a least squares fit to the epipolar constraints
- Each epipolar constraint measures the volume spanned by y, t, and Ry‘

Notes on Eight-Point Algorithm

- Points need to be in „general position“ to recover unique E: certain degenerate configurations exists (f.e. points on a plane, specific quadratic surfaces)

- No translation, ideally: $\left\|\widehat{\mathbf{t}}\right\| = 0 \Rightarrow \|\mathbf{E}\| = 0$

- But: for small translations, signal-to-noise ratio of image parallax may be problematic: „spurious“ pose estimate

- Non-linear 5-point algorithm with up to 10 (possibly complex) solutions (D. Nister, An Efficient Solution to the Five-Point Relative Pose Problem, CVPR 2004)

Normalized Eight-Point Algorithm

- Hartley, In Defense of the 8-Point Algorithm, IEEE PAMI 1997
- $\mathbf{A}$ can be numerically ill-conditioned when estimating the fundamental matrix with the eight-point algorithm naively

250906.36	183269.57	921.81	200931.10	146766.13	738.21	272.19	198.81	1.00
2692.28	131633.03	176.27	6196.73	302975.59	405.71	15.27	746.79	1.00
416374.23	871684.30	935.47	408110.89	854384.92	916.90	445.10	931.81	1.00
191183.60	171759.40	410.27	416435.62	374125.90	893.65	465.99	418.65	1.00
48988.86	30401.76	57.89	298604.57	185309.58	352.87	846.22	525.15	1.00
164786.04	546559.67	813.17	1998.37	6628.15	9.86	202.65	672.14	1.00
116407.01	2727.75	138.89	169941.27	3982.21	202.77	838.12	19.64	1.00
135384.58	75411.13	198.72	411350.03	229127.78	603.79	681.28	379.48	1.00

  - Noise attenuates stronger in large pixel coordinates (quadratic dependency)
  - Least squares (SVD) more sensitive to noise in large coordinates
  - „Imbalanced“ since pixel coordinates start at (0,0)

Normalized Eight-Point Algorithm

- Popular approach: Normalize coordinates to zero mean and standard deviation $\sqrt{2}$ in each image separately

$$\overline{\mathbf{z}} = \frac{\sqrt{2}}{\sigma}(\overline{\mathbf{y}} - \boldsymbol{\mu})$$

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \overline{\mathbf{y}} \qquad \boldsymbol{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} \left\| \overline{\mathbf{y}} - \boldsymbol{\mu} \right\|_2^2$$

- Find $\mathbf{B}$ and $\mathbf{B}'$ to normalize pixel coordinates

$$\overline{\mathbf{z}} = \mathbf{B}\overline{\mathbf{y}}$$

$$\overline{\mathbf{z}}' = \mathbf{B}'\overline{\mathbf{y}}'$$

$$\mathbf{B} = \begin{pmatrix} \frac{\sqrt{2}}{\sigma} & 0 & -\frac{\sqrt{2}}{\sigma}\mu_x \\ 0 & \frac{\sqrt{2}}{\sigma} & -\frac{\sqrt{2}}{\sigma}\mu_y \\ 0 & 0 & 1 \end{pmatrix}$$

Normalized Eight-Point Algorithm

- Apply eight-point algorithm on normalized coordinates with epipolar constraints

$$\overline{\mathbf{y}}^\top \mathbf{B}^\top \mathbf{F}' \mathbf{B}' \overline{\mathbf{y}}' = 0$$

- Recover $\mathbf{F}$ from $\mathbf{F}'$

$$\mathbf{F} = \mathbf{B}^\top \mathbf{F}' \mathbf{B}'$$

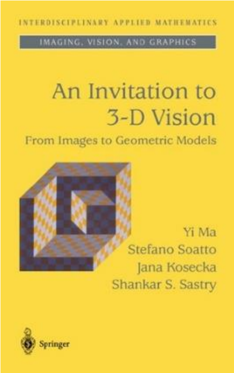
Eight-Point Algorithm for F

- Calibrated case: we know camera intrinsics, we can estimate E
- Uncalibrated case: we do not know camera intrinsics, we can only estimate F

- In the uncalibrated case, rotation and translation can not be recovered from F due to the unknown camera intrinsics

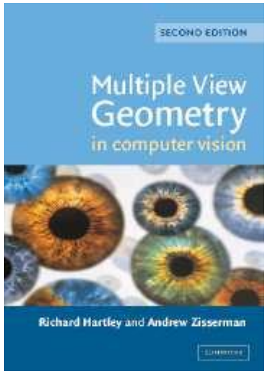
Further Reading

- 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

Lessons Learned Today

- Visual odometry is the process of estimating ego-motion using onboard visual sensors
  - Indirect methods extract and match geometric primitives such as keypoints
  - Direct methods directly operate on the pixel level
- Motion estimation from 2D-to-2D image correspondences
  - (Normalized) eight-point algorithm for estimating the essential and fundamental matrix

Thanks for your attention!