Computer Vision Group
Prof. Daniel Cremers

Technische Universität München

Practical Course: Vision-based Navigation Winter Term 2017/2018

Lecture 2: Visual Motion Estimation

Vladyslav Usenko, Lukas von Stumberg, Prof. Dr. Jörg Stückler

What we will cover today

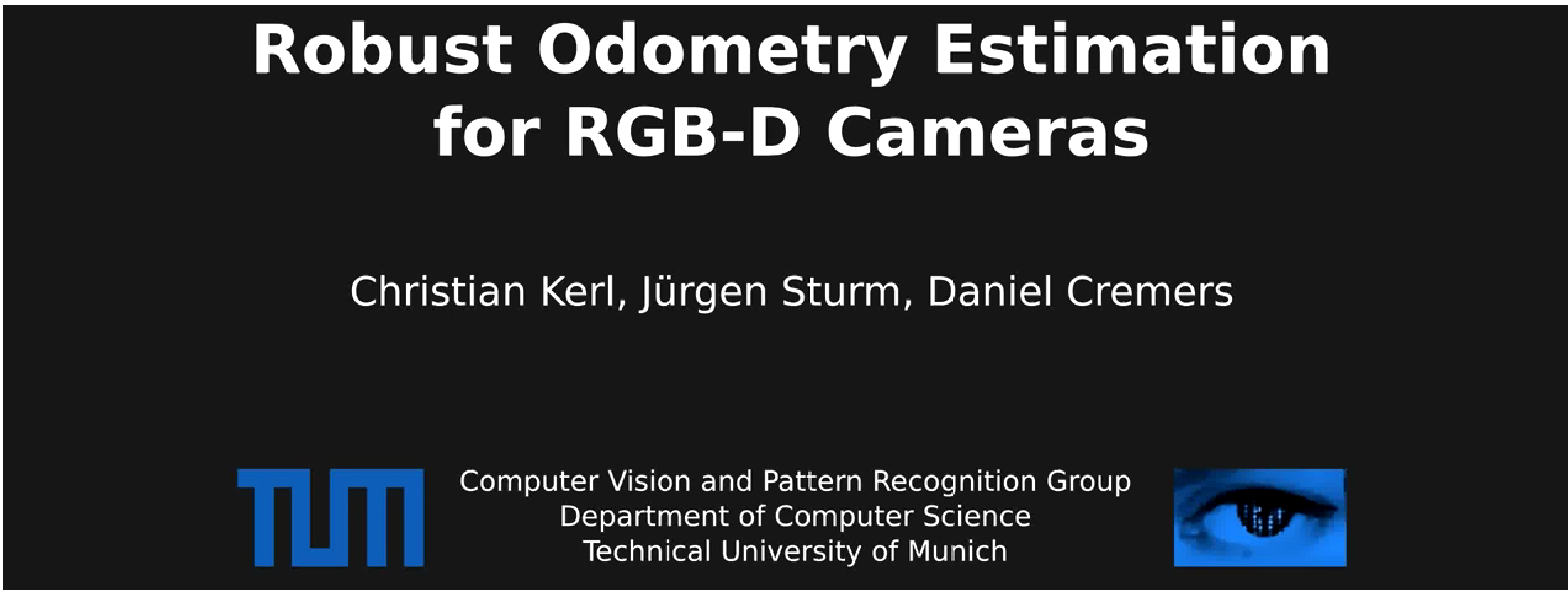
- Introduction to visual motion estimation approaches
  - Visual odometry (VO) vs. visual SLAM
  - Overview on VO approaches for monocular, stereo, RGB-D cameras
  - The notions of sparse, dense, and direct

- Sparse, keypoint-based visual odometry

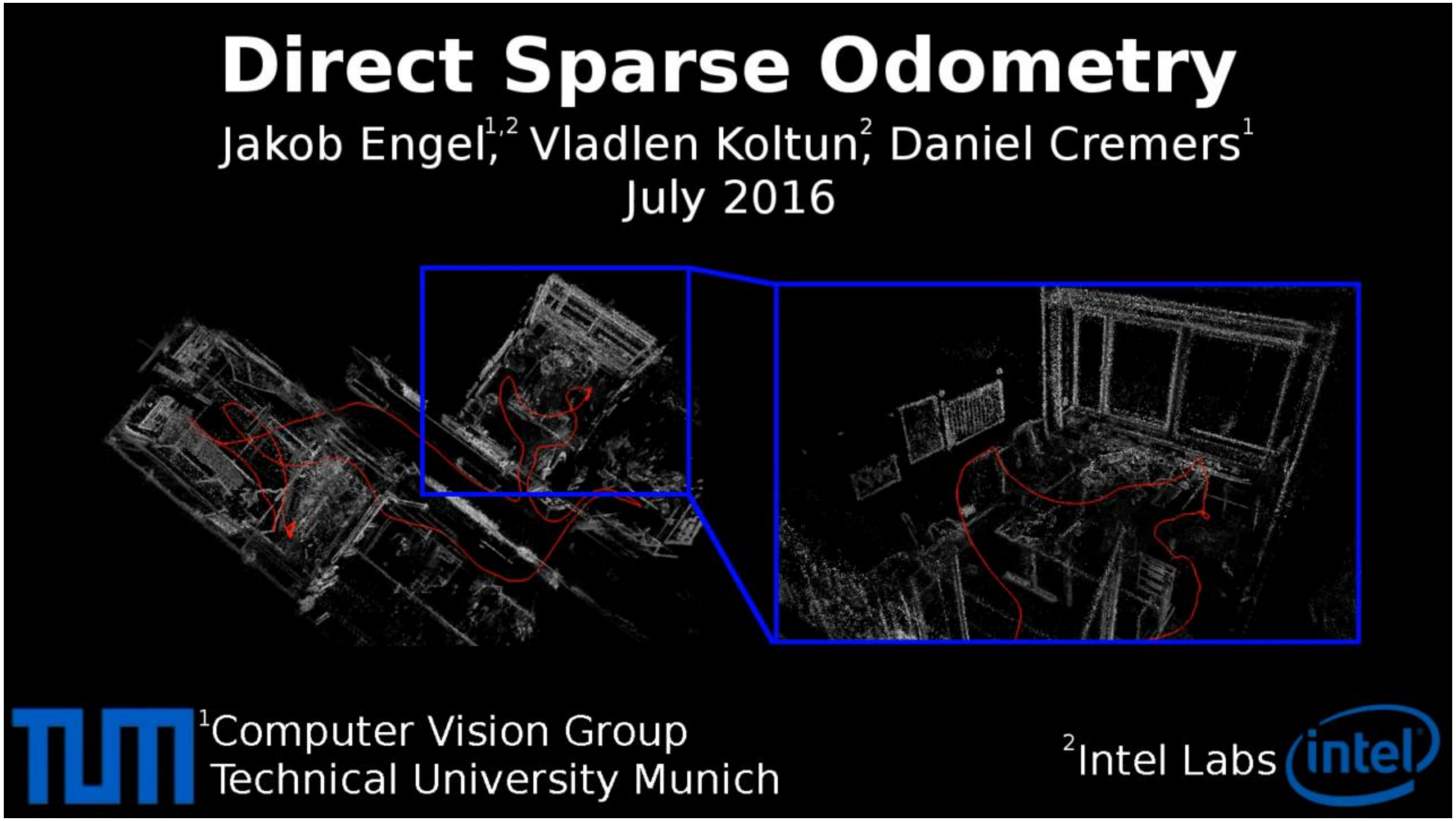
- Direct, dense motion estimation
  - Motion representation using the SE(3) Lie algebra
  - Non-linear least squares optimization
  - Direct dense RGB-D odometry

Part 1:
Introduction to Visual Odometry

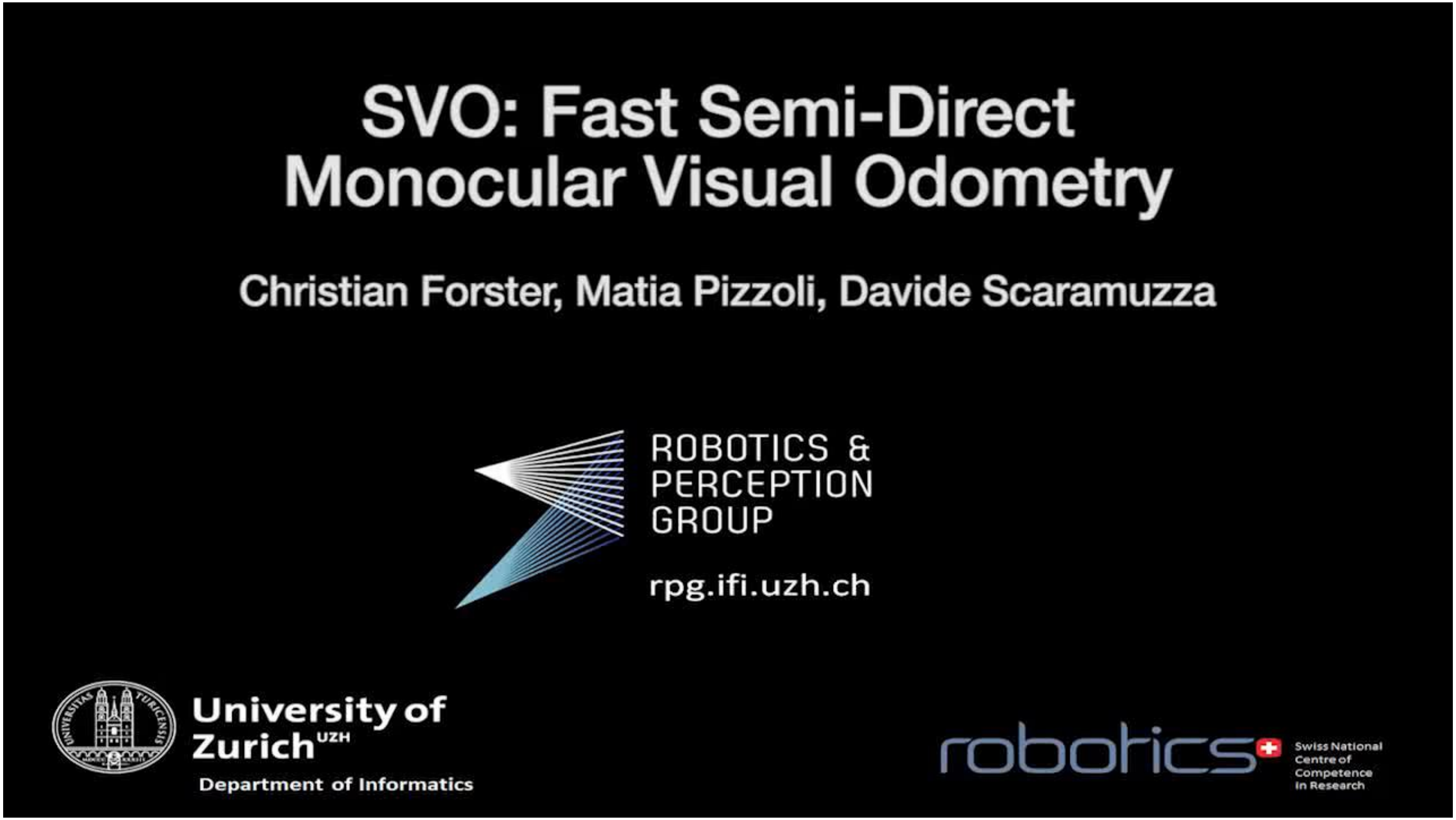
Visual Motion Estimation a.k.a. Visual Odometry



Visual Motion Estimation a.k.a. Visual Odometry



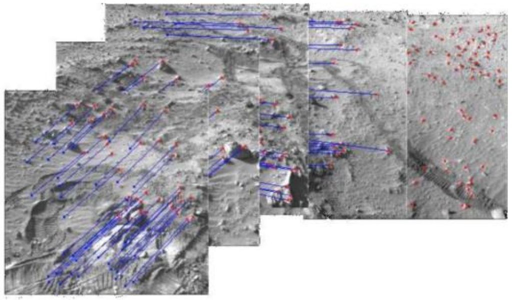
Visual Motion Estimation a.k.a. Visual Odometry



The Term “Visual Odometry”

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

- Visual Odometry (VO):
  - 1980-2004: Dominant 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



Visual Odometry

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

- Important in GPS-denied environments (indoors, underwater, etc.)

- Relation to Visual Simultaneous Localization and Mapping (SLAM):
  - Local (VO&VSLAM) vs. global (VSLAM) consistency
  - VO: 3D reconstruction only at local scale (if at all)
  - VO: Real-time requirements

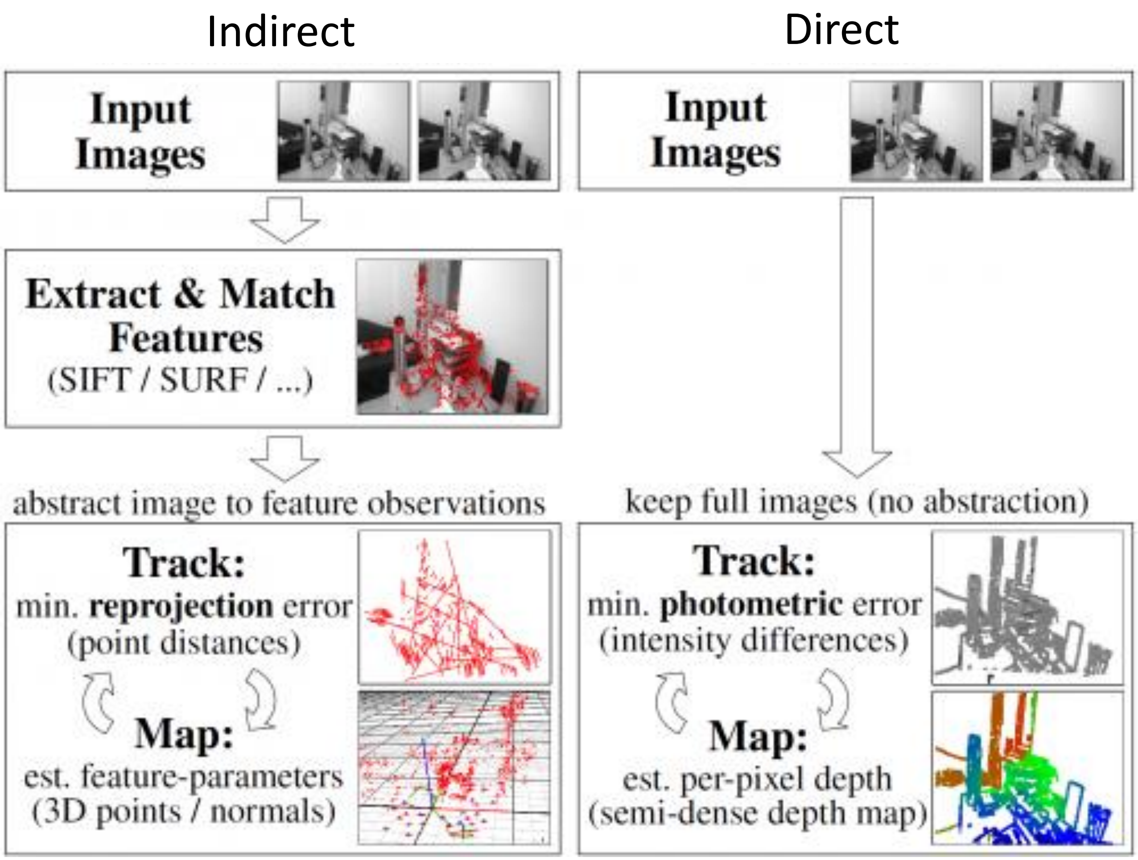
Sensors for Visual Odometry

- Monocular:
  - Pros: Low-power, light-weight, low-cost, simple to calibrate and use
  - Cons: requires motion parallax and textured scenes, scale not observable

- Stereo:
  - Pros: depth without motion, less power than active structured light
  - Cons: requires textured scenes, accuracy depends on baseline, requires extrinsic calibration of the cameras, synchronization of the cameras

- Active RGB-D sensors:
  - Pros: also work in untextured scenes, similar to stereo processing
  - Cons: active sensing consumes power, blackbox depth estimation

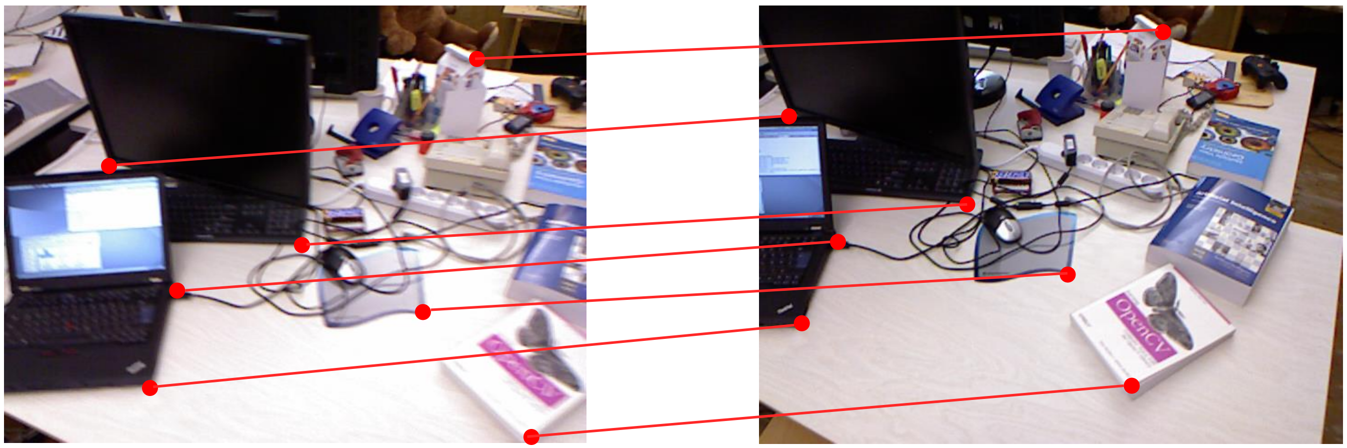
Indirect, Direct, Sparse, Dense



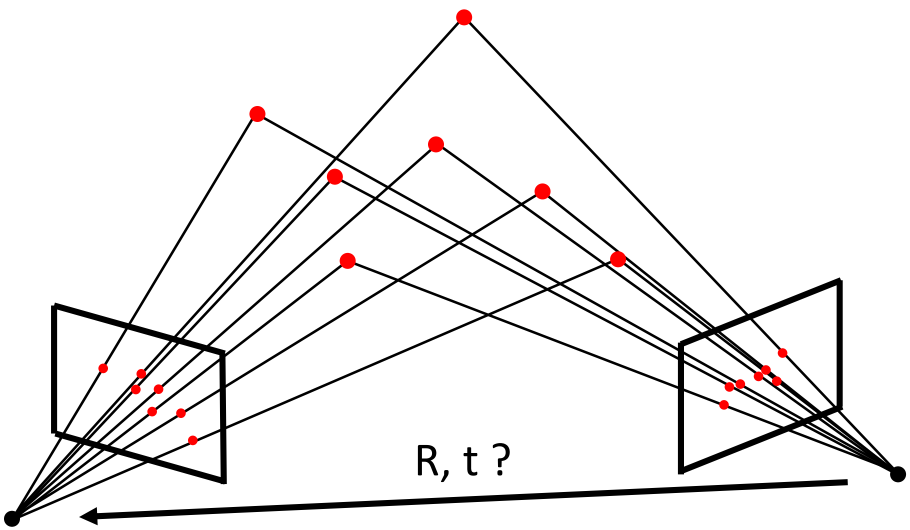
- Sparse: use a small set of selected pixels (keypoints)
- Dense: use all (valid) pixels

Part 2:
Sparse Visual Odometry

Sparse Keypoint-based Visual Odometry



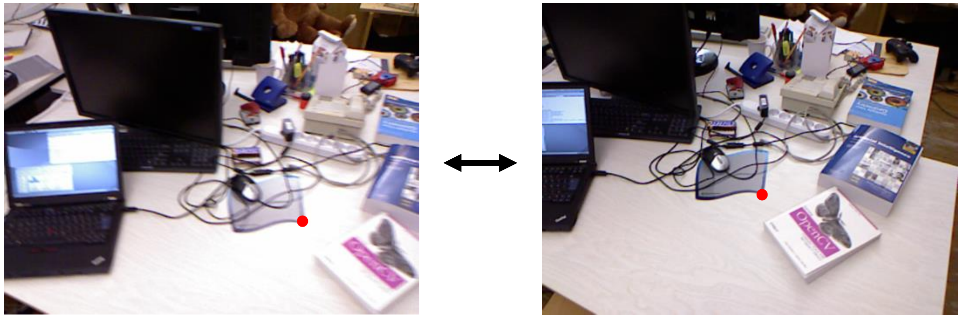
Extract and match keypoints



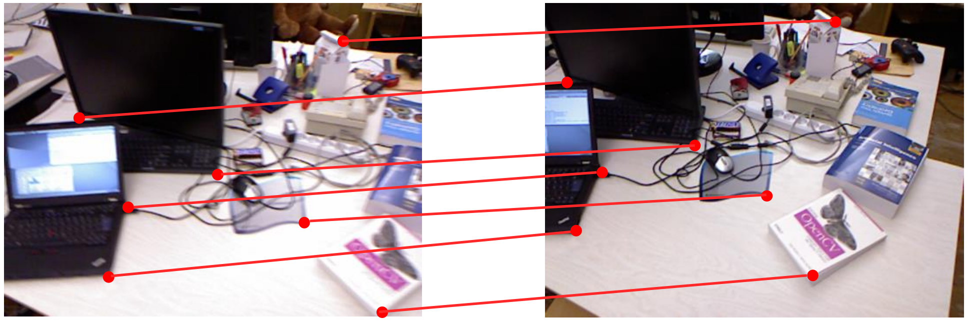
Determine relative camera pose (R, t) from keypoint matches

Keypoint Extraction

- Detection repeatability
  - We want to find the (accurate) image of the same 3D point from different view-points



- Descriptor distinctiveness
  - We want a descriptor that achieves (in the ideal case) a unique and correct association of corresponding keypoints

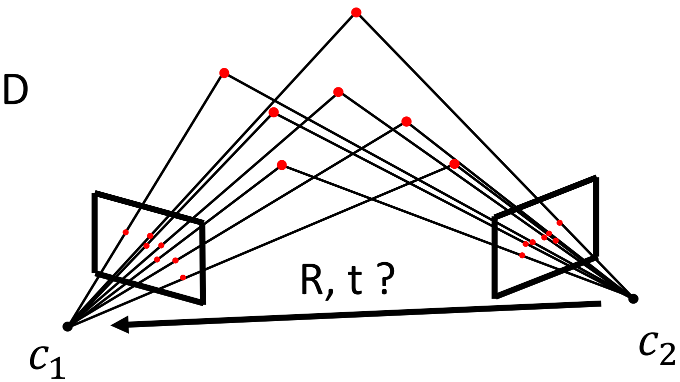


Keypoint Detectors and Descriptors

- Keypoint detection and description in images has been extensively studied
- Nowadays there is plenty of fast and repeatable detectors available, e.g.,
  - Harris corner variants
  - FAST corner variants (e.g. ORB detector)
  - DoG blob variants (SIFT, SURF)
  - Learning-based keypoints
- Many detectors come with a suitable descriptor, e.g.,
  - ORB (binary pixel comparisons locally around keypoint)
  - SIFT/SURF (grayscale gradient patterns locally around keypoint)

Monocular Keypoint-based Motion Estimation

- Monocular case: no depth available at keypoints



- If we knew the relative pose of the cameras and the 3D position of each keypoint match, we could directly compute to which pixels the keypoints should project in each camera image

- To find the unknown pose and 3D positions: minimize the reprojection error of all keypoints (optimization problem)

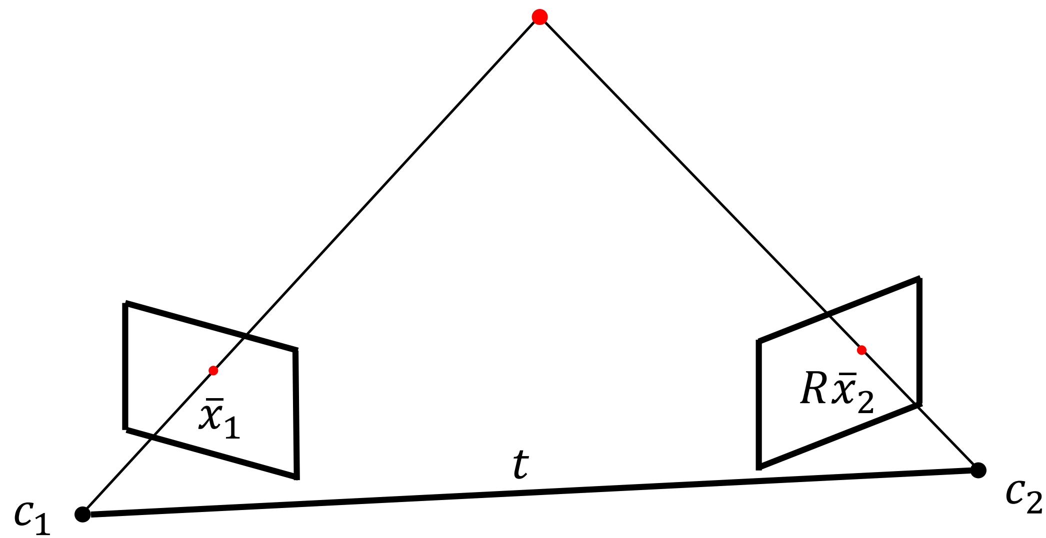
$$E(R, t, x_1, \dots, x_N) = \frac{1}{N} \sum_i \left\| z_{1,i} - \pi(x_i) \right\|_2^2 + \left\| z_{2,i} - \pi(R x_i + t) \right\|_2^2$$

- Reprojection error: difference between measured and expected pixel position of a keypoint

Uniqueness?
Optimality?

Motion from Epipolar Geometry

- Alternative: examine epipolar geometry more closely



- The rays from each camera to the keypoint and the baseline t are coplanar!

$$\bar{x}_1^T(t \times R\bar{x}_2) = 0 \;\leftrightarrow\; \bar{x}_1^T[t]_\times R\bar{x}_2 = 0$$

- The essential matrix $E = [t]_\times R$ captures the relative camera pose
- Each keypoint match provides an „epipolar constraint“
- 8 matches suffice to determine E (8-point algorithm)
- In the uncalibrated case, the camera calibration needs to be subsumed into the so-called fundamental matrix $F = K^{-T}EK^{-1}$

8-Point Algorithm (Longuet-Higgins, 1981)

- Find approximation to essential matrix:

  - Construct matrix $A = (a_1, a_2, \dots, a_N)^T$ with $a_i = \bar{x}_{1,i} \times \bar{x}_{2,i}$.

  - Apply a singular value decomposition (SVD) on $\mathrm{A} = USV^T$ and unstack the 9th column vector of V into $\tilde{E}$

  - Project the approximate $\tilde{E}$ into the (normalized) essential space:
    Determine the SVD of $\tilde{E} = U \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)\, V^T$ and replace the singular values $\sigma_1, \sigma_2, \sigma_3$ with 1,1,0 to find $E = U \operatorname{diag}(1,1,0)\, V^T$

  - Determine one of the following 4 possible solutions that intersect the points in front of both cameras:

$$R = U\, R_Z^T\left(\pm\frac{\pi}{2}\right) V^T$$

$$[t]_\times = U\, R_Z\left(\pm\frac{\pi}{2}\right) \operatorname{diag}(1,1,0) U^T$$

$$\text{with} \quad R_Z^T\left(\pm\frac{\pi}{2}\right) = \begin{pmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3D Keypoint-based Motion Estimation

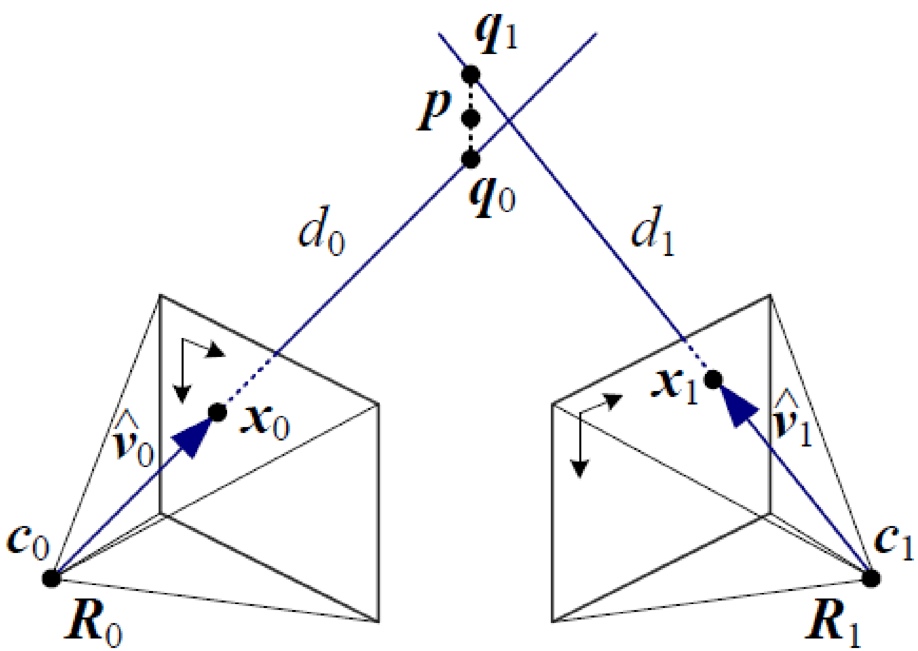
- Stereo case: rotation and translation known between the left and right image
- Match keypoints between left and right image, triangulate their 3D positions
- To estimate motion between two stereo image pairs we could:
  - use 8-point algorithm on keypoints in the left images
  - recover scale from triangulated stereo depth
- Alternatively, since 3D positions of the keypoints known: simpler least-squares optimization of the reprojection error:

$$E(R,t) = \frac{1}{N}\sum_i \left\| z_{1,i} - \pi(R^T x_i - R^T t) \right\|_2^2 + \left\| z_{2,i} - \pi(R x_i + t) \right\|_2^2$$

Triangulation

- **Given:** n cameras $\{M_j = K_j(R_j\ \mathbf{t}_j)\}$

  Point correspondence $\mathbf{x}_0, \mathbf{x}_1$
- **Wanted:** Corresponding 3D point $\mathbf{p}$



Triangulation

- Where do we expect to see $\mathbf{p} = (X\ Y\ Z\ W)^\top$?

$$\hat{x} = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}W}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}W} \qquad \hat{y} = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}W}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}W}$$

- Minimize the residuals

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \sum_j d(\mathbf{x}_j, \hat{\mathbf{x}}_j)^2$$

Triangulation

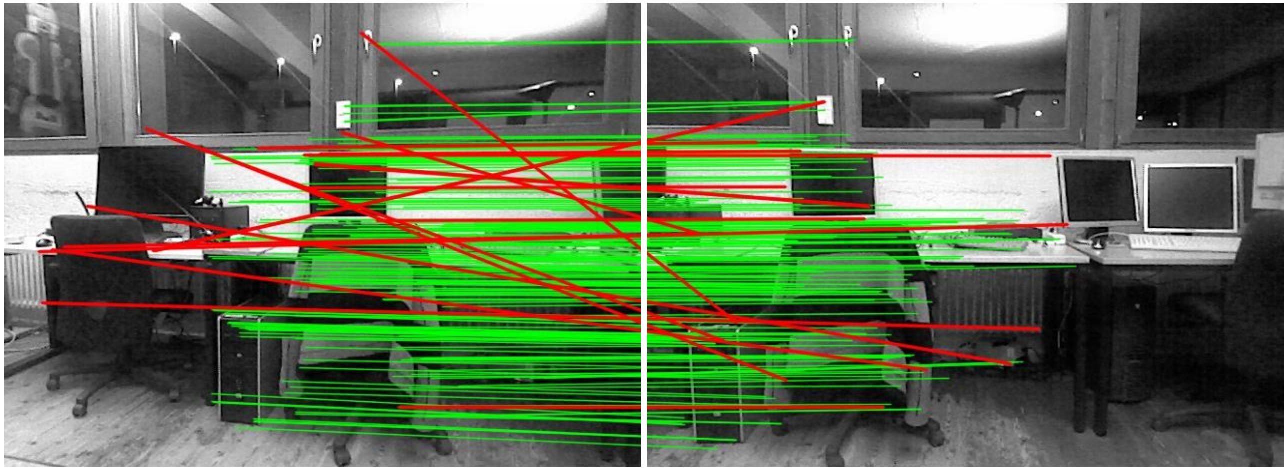
- Multiply with denominator gives

$$0 = (x_j m_{31} - m_{11})X + (x_j m_{32} - m_{12})Y + (x_j m_{33} - m_{13})Z + (x_j m_{34} - m_{14})W$$
$$0 = (y_j m_{31} - m_{21})X + (y_j m_{32} - m_{22})Y + (y_j m_{33} - m_{23})Z + (y_j m_{34} - m_{24})W$$

Solve for $\mathbf{p} = (X \ Y \ Z \ W)^\top$ using:

- Linear least squares with W=1
- Linear least squares using SVD
- Non-linear least squares of the residuals (most accurate)

Robust Keypoint Matching



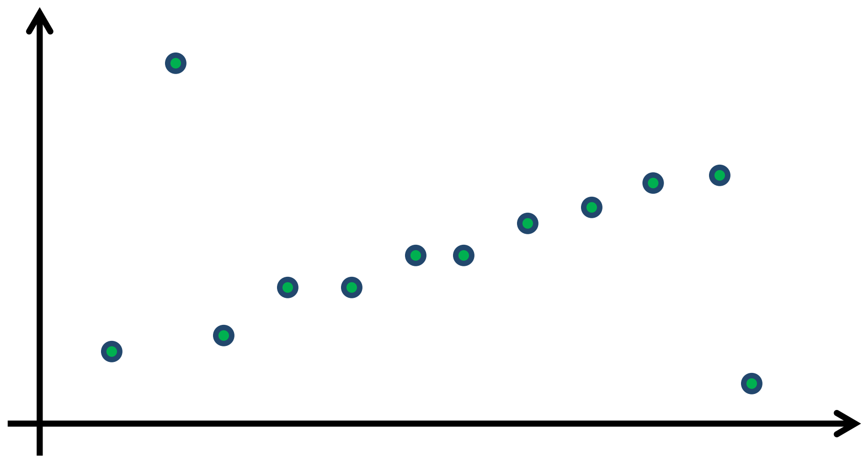
- Keypoint detectors and descriptors not perfect
- Pose estimation very sensitive to wrong correspondences (especially when using the 8-point algorithm)

- Idea: try out different combinations of 8 matches until we find a good fit for most of the overall keypoints

- Random Sample Consensus (RANSAC) algorithm

Robust Estimation

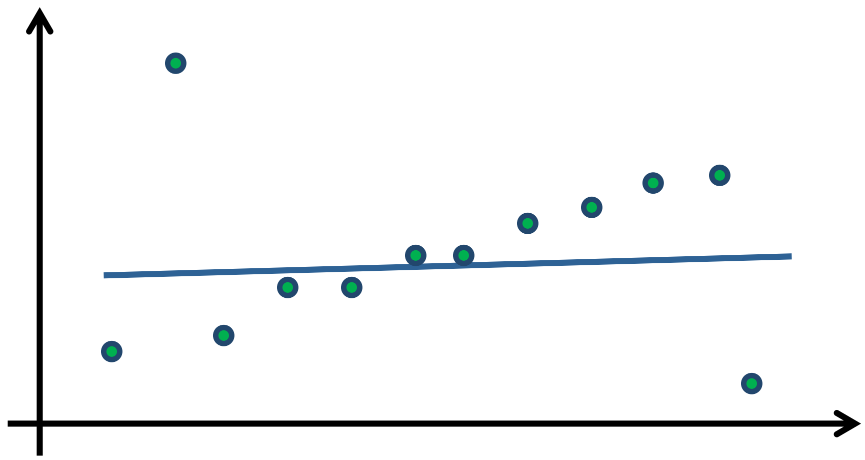
Example: Fit a line to 2D data containing outliers



- Input data is a mixture of
  - Inliers (perturbed by Gaussian noise)
  - Outliers (unknown distribution)
- Let's fit a line using least squares…

Robust Estimation

Example: Fit a line to 2D data containing outliers



- Input data is a mixture of
  - Inliers (perturbed by Gaussian noise)
  - Outliers (unknown distribution)
- Least squares fit gives poor results!

RANdom SAmple Consensus (RANSAC) [Fischler and Bolles, 1981]

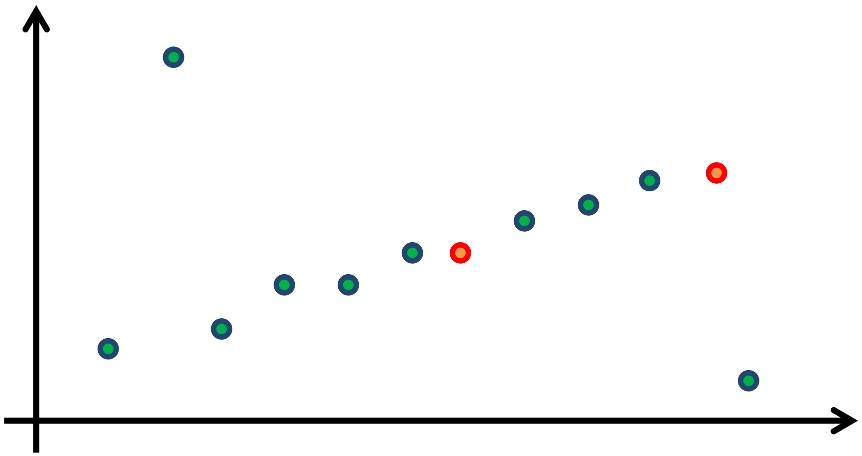
Goal: Robustly fit a model to a data set S which contains outliers

Algorithm:

1. Randomly select a (minimal) subset
2. Instantiate the model from it
3. Using this model, classify all data points as inliers or outliers
4. Repeat 1-3 for N iterations
5. Select the largest inlier set, and re-estimate the model from all points in this set

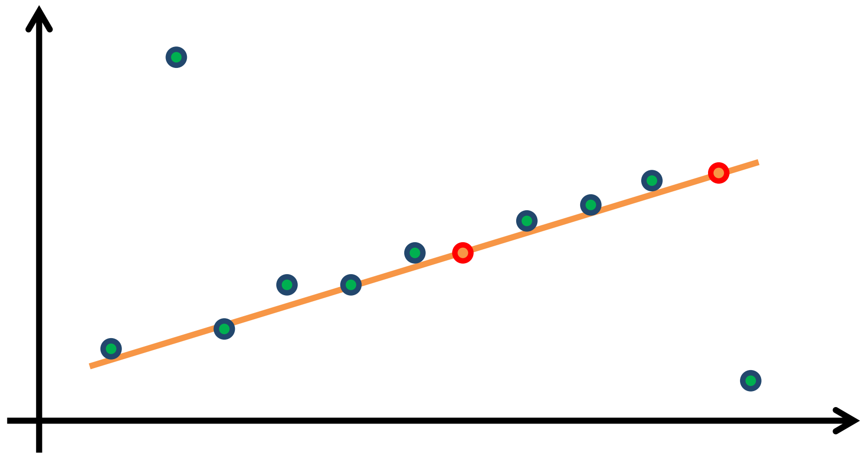
Example

- Step 1: Sample a random subset



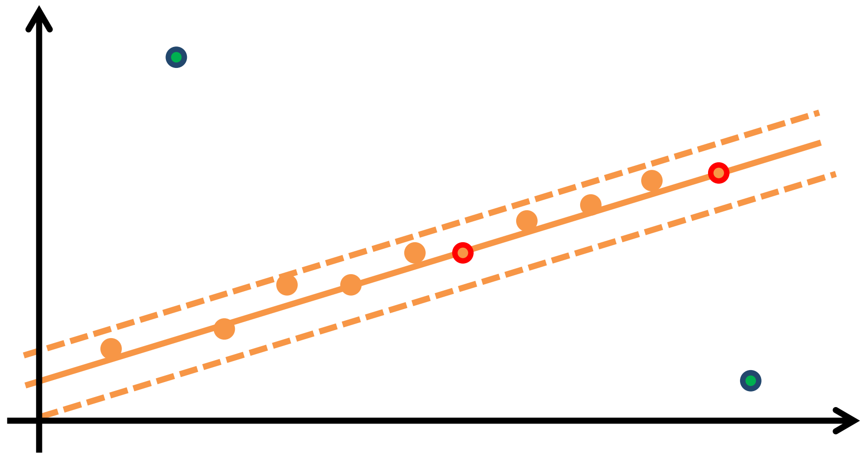
Example

- Step 2: Fit a model to this subset



Example

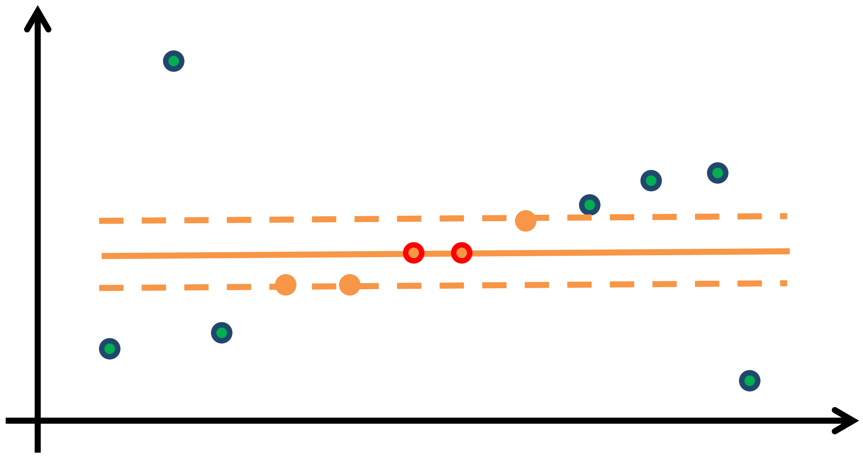
- Step 3: Classify points as inliers and outliers (e.g., using a threshold distance)



→ 10 inliers, 2 outliers

Example

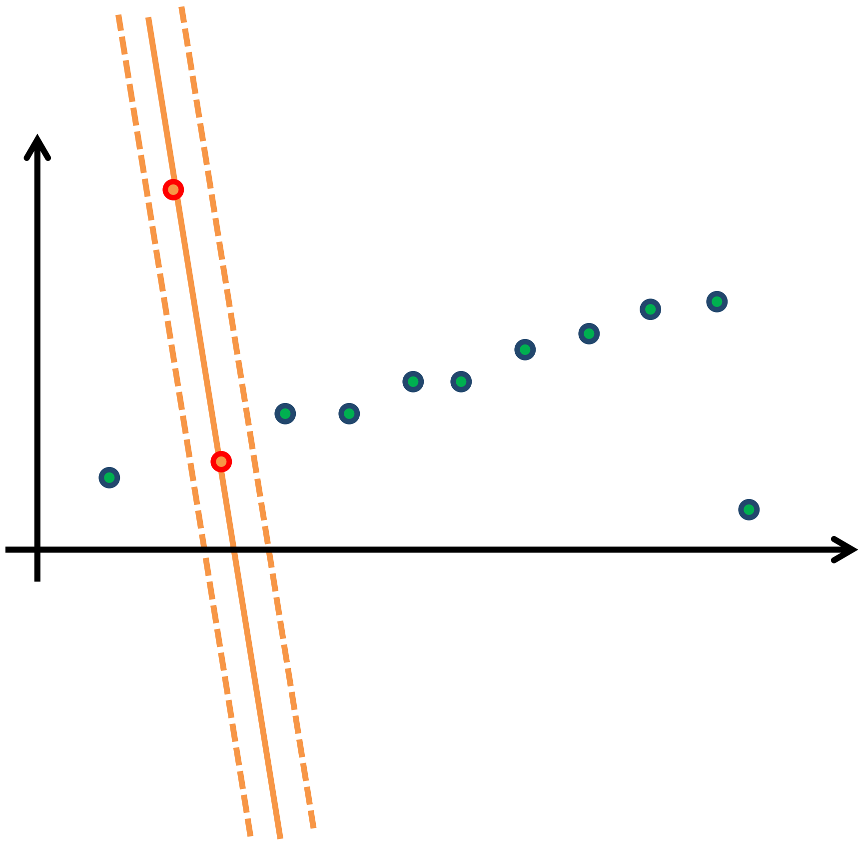
- Step 4: Repeat steps 1-3 for N iterations



Iteration 2:
→ 5 inliers, 7 outliers

Example

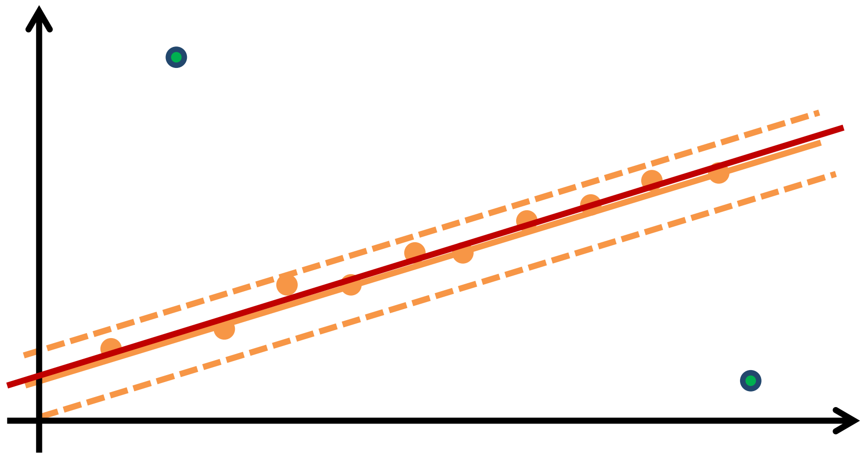
- Step 4: Repeat steps 1-3 for N iterations



Iteration 3:
→ 2 inliers, 10 outliers

Example

- Step 5: Select the best model (most inliers), then re-fit model using all inliers



Best model:
Iteration 1
(10 inliers, 2 outliers)

How Many Iterations Do We Need?

- For a probability of success p, we need

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$$ iterations

  for subset size s and outlier ratio ϵ
- E.g., for p=0.99:

	Required 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

Summary on RANSAC

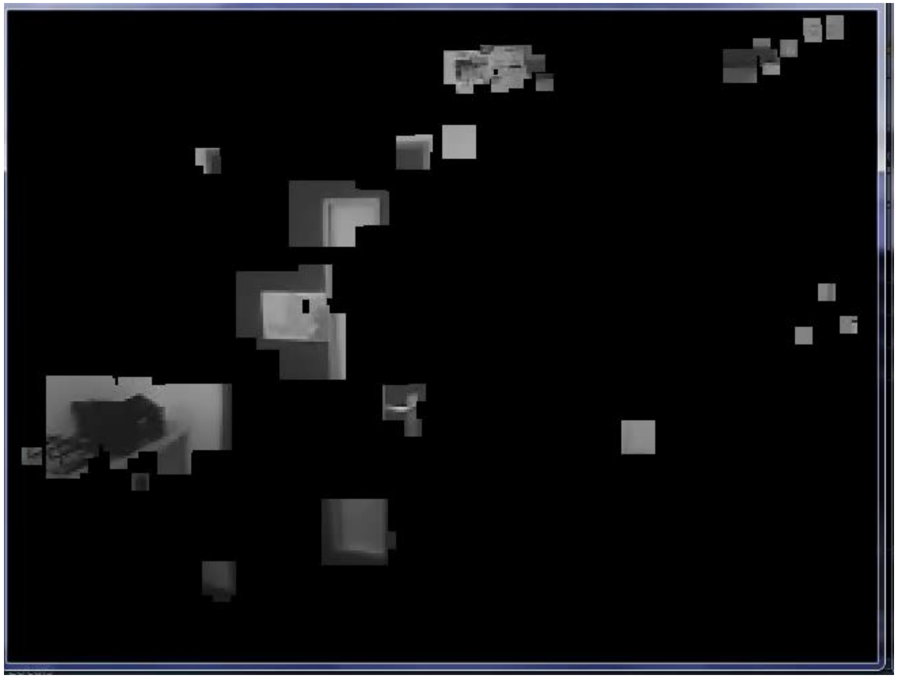
- Efficient algorithm to estimate a model from noisy and outlier-contaminated data
- RANSAC is used today very widely
- Often used in feature matching / visual motion estimation
- Many improvements/variants (e.g., PROSAC, MLESAC, …)

Part 2: Lessons Learned

- How to estimate motion from keypoints from monocular images using the 8-point algorithm
- How to use the 8-point algorithm for stereo and RGB-D
- How to triangulate keypoint matches given the camera pose
- How to separate inliers from outliers using RANSAC

Part 3:
Direct Dense Visual Odometry

Problem with Keypoint-based Methods



Special Euclidean Group SE(3)

- Not all matrices are transformation matrices: Transformation matrices have a special structure

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

  - Translation $\mathbf{t}$ has 3 degrees of freedom
  - Rotation $\mathbf{R}$ has 3 degrees of freedom
- They form a group which we call SE(3). The group operator is matrix multiplication:

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

- The operator is associative, but not commutative!
- There is also an inverse and a neutral element

Parametrizations of SE(3)

- Translation $\mathbf{t}$ has 3 degrees of freedom
- Rotation $\mathbf{R}$ has 3 degrees of freedom

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

- Different parametrizations θ of $\mathbf{T}(\theta)$
  - Direct matrix representation
  - Quaternion / translation
  - Axis,angle / translation
  - Later: Twist coordinates in Lie Algebra se(3) of SE(3)

Pose Parametrization for Optimization

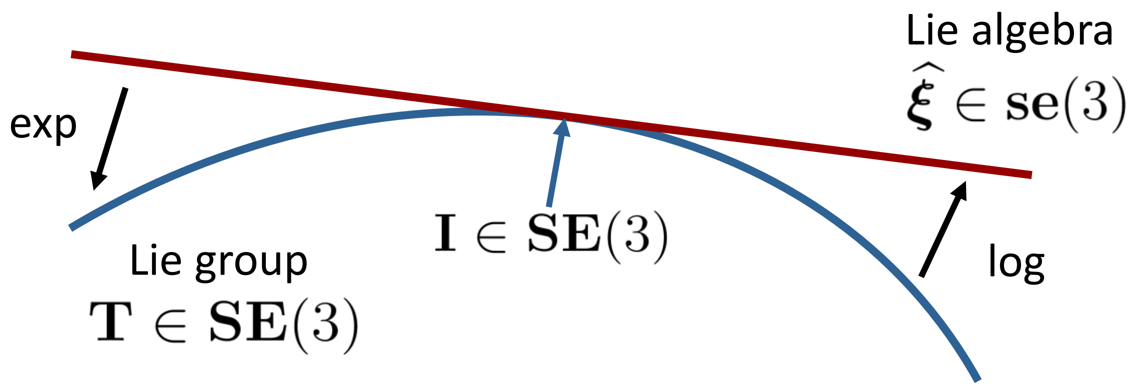
- Let's say we want to optimize a cost function $E(\theta)$ for the pose in some θ parametrization
- We need to set $\nabla_\theta E(\theta) = 0$

  which we can tackle using gradient descent (or higher-order methods) by making steps on θ

$$\theta \leftarrow \theta - \lambda \nabla_\theta E(\theta)$$

- When we determine the derivative of $E(\theta)$, we will require the derivative of $\mathbf{T}(\theta)$ for θ, which should have no singularities
- We also update the pose parametrization, which requires a minimal representation

SE(3) Lie Algebra for Representing Motion



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

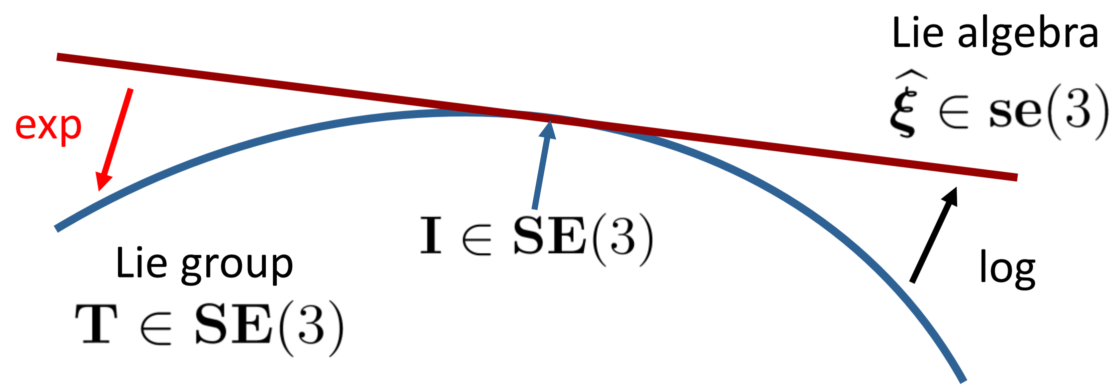
$$\boldsymbol{\omega} \in \mathbb{R}^3 \qquad \widehat{\boldsymbol{\omega}} := [\boldsymbol{\omega}]_\times$$

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

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

- SE(3) is also a smooth manifold which makes it a Lie group
- The SE(3) Lie algebra se(3) provides an elegant way to parametrize poses for optimization
- Its elements $\widehat{\boldsymbol{\xi}} \in \mathbf{se}(3)$ form the tangent space of SE(3) at its identity $\mathbf{I} \in \mathbf{SE}(3)$
- The se(3) elements can be interpreted as rotational and translational velocities applied for some duration (twist) that explain the infinitesimal motion away from the identity transformation

Exponential Map of SE(3)

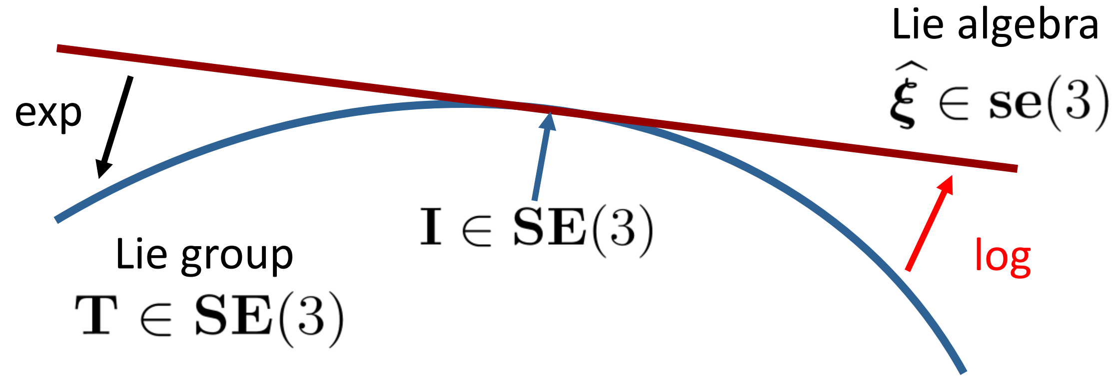


- The exponential map finds the transformation matrix for a twist:

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

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

Logarithm Map of SE(3)



- The logarithm maps twists to transformation matrices:

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

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

Optimization with Twist Coordinates

- How are twists useful in optimization?
- They provide a minimal representation without singularities close to identity
- Since SE(3) is a smooth manifold, we can decompose $\mathbf{T}(\boldsymbol{\xi})$ in each optimization step into the transformation itself and a small increment (could be left or right-multiplied) $\boldsymbol{\delta\xi}$:

$$\mathbf{T}(\boldsymbol{\xi}) := \mathbf{T}(\boldsymbol{\xi})\mathbf{T}(\boldsymbol{\delta\xi})$$

- Gradient descent operates on the auxiliary variable

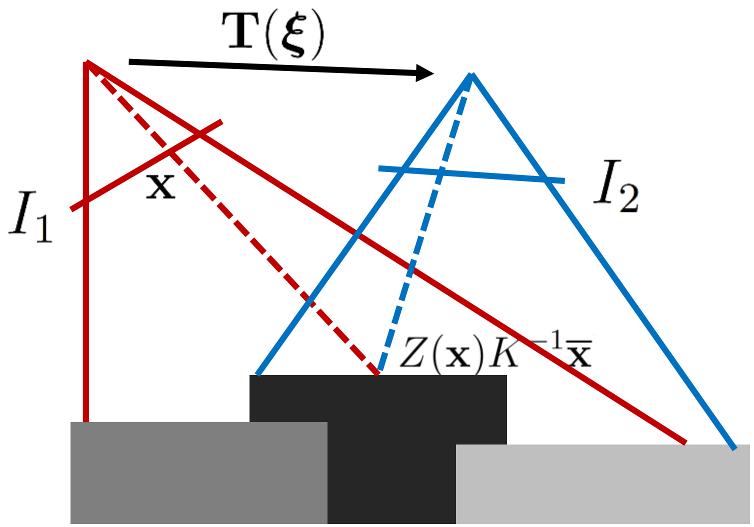
$$\boldsymbol{\delta\xi} \leftarrow \mathbf{0} - \nabla_{\boldsymbol{\delta\xi}} E(\boldsymbol{\delta\xi})$$

$$\widehat{\boldsymbol{\xi}} \leftarrow \log\left(\exp\left(\widehat{\boldsymbol{\xi}}\right)\exp\left(\widehat{\boldsymbol{\delta\xi}}\right)\right)$$

SE(3) Lie Algebra for Representing Motion

- C++ implementation: Sophus extension library for Eigen, by Hauke Strasdat, https://github.com/strasdat/Sophus
- Further reading on motion representation using the SE(3) Lie algebra:
  - Yi Ma, Stefano Soatto, Jana Kosecka, Shankar S. Sastry. An Invitation to 3-D Vision, Chapter 2: http://vision.ucla.edu/MASKS/
  - http://ingmec.ual.es/~jlblanco/papers/jlblanco2010geometry3D_techrep.pdf
  - http://ethaneade.com/lie.pdf

Dense Direct Image Alignment



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

$$I_1(\mathbf{x}) = I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z(\mathbf{x})K^{-1}\overline{\mathbf{x}}))$$

- For RGB-D, we have the depth, but want to find the camera motion!

Dense Direct Image Alignment

- Given a camera motion, we can find and compare corresponding pixels through projection.
- We measure in one image a noisy version of the intensity in the other image:

$$I_1(\mathbf{x}) = I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z(\mathbf{x})K^{-1}\overline{\mathbf{x}})) + \epsilon$$

- A simple assumption is Gaussian noise, e.g. if the noise only comes from pixel noise on the chip $\epsilon \sim \mathcal{N}(0, \sigma_I^2)$
- If we further assume that the measurements are stochastically independent at each pixel, we can formulate the joint probability

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

$$p(\boldsymbol{\xi} \mid I_1, I_2) \propto \prod_{\mathbf{x}\in\Omega} \mathcal{N}\left(I_1(\mathbf{x}) - I_2(\pi(\mathbf{T}(\boldsymbol{\xi})Z(\mathbf{x})K^{-1}\overline{\mathbf{x}})); 0, \sigma_I^2\right)$$

Dense Direct Image Alignment

- Maximum-likelihood estimation problem
- Optimize negative log-likelihood
  - Product becomes a summation
  - Exponentials disappear
  - Normalizers are independent of the pose

$$E(\boldsymbol{\xi}) = \text{const.} + \frac{1}{2} \sum_{\mathbf{x} \in \Omega} \frac{r(\mathbf{x}, \boldsymbol{\xi})^2}{\sigma_I^2}$$

$$r(\mathbf{x}, \boldsymbol{\xi}) = I_1(\mathbf{x}) - I_2(\pi(\mathbf{T}(\boldsymbol{\xi}) Z(\mathbf{x}) K^{-1} \overline{\mathbf{x}}))$$

- This non-linear least squares error function can be efficiently optimized using standard methods (Gauss-Newton, Levenberg-Marquardt)

Least Squares Optimization

- If the residuals would be linear in $\boldsymbol{\xi}$, i.e., $r(\boldsymbol{\xi}) = \mathbf{A}\boldsymbol{\xi} + \mathbf{b}$ optimization would be simple, has a closed-form solution
- In this case, the error function and its derivatives are

$$
\begin{aligned}
E(\boldsymbol{\xi}) &= \frac{1}{2} r(\boldsymbol{\xi})^T \mathbf{W} r(\boldsymbol{\xi}) \\
\nabla_{\boldsymbol{\xi}} E(\boldsymbol{\xi}) &= \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} r(\boldsymbol{\xi}) = \mathbf{A}^T \mathbf{W} r(\boldsymbol{\xi}) \\
\nabla^2_{\boldsymbol{\xi}} E(\boldsymbol{\xi}) &= \mathbf{A}^T \mathbf{W} \mathbf{A}
\end{aligned}
$$

- „Linearizing“ and setting the first derivative to zero yields

$$
\begin{aligned}
\nabla_{\boldsymbol{\xi}} E(\boldsymbol{\xi}) = \nabla_{\boldsymbol{\xi}} E(\boldsymbol{\xi}_0) + \nabla^2_{\boldsymbol{\xi}} E(\boldsymbol{\xi}_0)(\boldsymbol{\xi} - \boldsymbol{\xi}_0) &= 0 \\
\boldsymbol{\xi} &= \boldsymbol{\xi}_0 - \nabla^2_{\boldsymbol{\xi}} E(\boldsymbol{\xi}_0)^{-1} \nabla_{\boldsymbol{\xi}} E(\boldsymbol{\xi}_0) \\
\boldsymbol{\xi} &= \boldsymbol{\xi}_0 - \left(\mathbf{A}^T \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{W} r(\boldsymbol{\xi}_0)
\end{aligned}
$$

Non-linear Least Squares Optimization

- In direct image alignment, the residuals are non-linear in $\boldsymbol{\xi}$
- Gauss-Newton method, iterate:
  - Linearize residuals

$$\widetilde{r}(\boldsymbol{\xi}) = r(\boldsymbol{\xi}_0) + \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})(\boldsymbol{\xi} - \boldsymbol{\xi}_0)$$

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

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

$$\nabla^2_{\boldsymbol{\xi}} \widetilde{E}(\boldsymbol{\xi}) = \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})$$

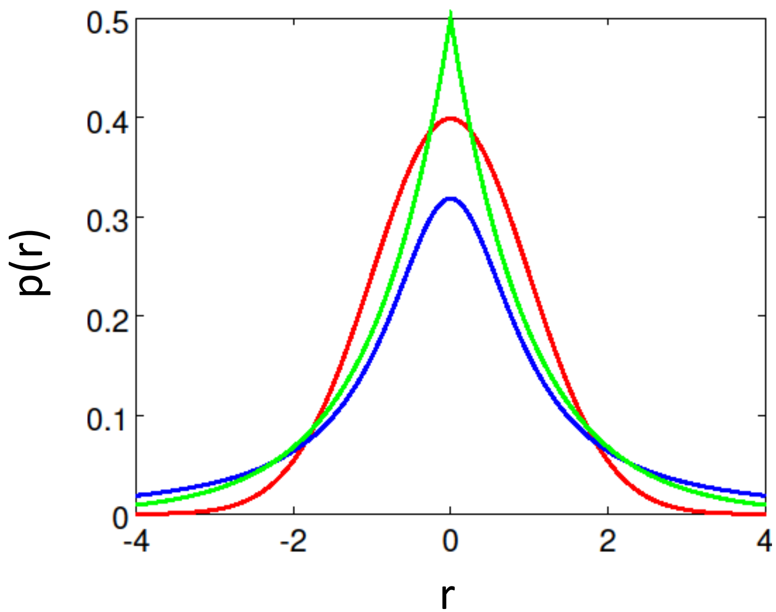
  - Solve linearized system

$$\nabla_{\boldsymbol{\xi}} \widetilde{E}(\boldsymbol{\xi}) = \nabla_{\boldsymbol{\xi}} \widetilde{E}(\boldsymbol{\xi}_0) + \nabla^2_{\boldsymbol{\xi}} E(\boldsymbol{\xi}_0)(\boldsymbol{\xi} - \boldsymbol{\xi}_0) = 0$$

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \nabla^2_{\boldsymbol{\xi}} \widetilde{E}(\boldsymbol{\xi})^{-1} \nabla_{\boldsymbol{\xi}} \widetilde{E}(\boldsymbol{\xi})$$

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \left(\nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})\right)^{-1} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} r(\boldsymbol{\xi})$$

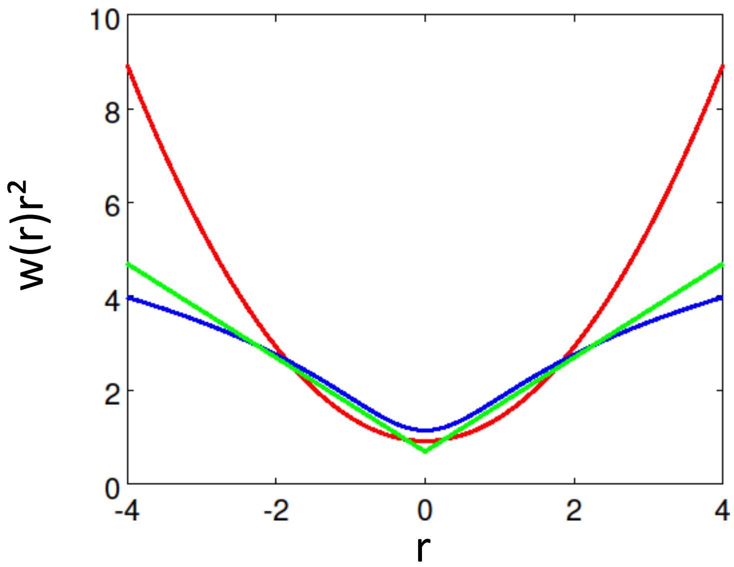
Actual Residual Distribution



- Normal distribution
- Laplace distribution
- Student-t distribution

- The Gaussian noise assumption is not valid
- Many outliers (occlusions, motion, etc.)
- Residuals are distributed with more mass on the larger values

Iteratively Reweighted Least Squares



- Normal distribution
- Laplace distribution
- Student-t distribution

- Can we change the residual distribution in the least squares optimization?
- We can reweight the residuals in each iteration to adapt residual distribution

$$E(\boldsymbol{\xi}) = \frac{1}{2} \sum_{\mathbf{x} \in \Omega} w(r(\mathbf{x}, \boldsymbol{\xi})) \frac{r(\mathbf{x}, \boldsymbol{\xi})^2}{\sigma_I^2}$$

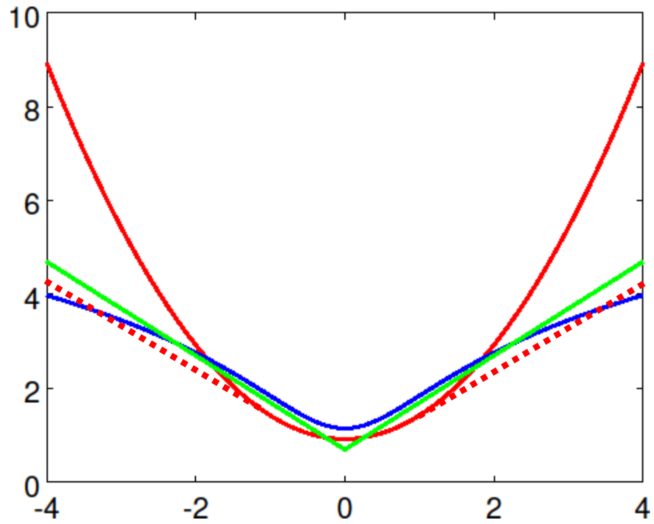
E.g., for Laplace distribution:

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

Huber-Loss

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

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



Huber-loss for δ = 1

Linearization of Image Alignment Residuals

- In our direct image alignment case, the linearized residuals are

$$\nabla_{\boldsymbol{\xi}} r(\mathbf{x}, \boldsymbol{\xi}) = -\nabla_{\pi} I_2(\pi(\mathbf{p}(\mathbf{x}, \boldsymbol{\xi}))) \;\cdot\; \nabla_{\boldsymbol{\xi}} \pi(\mathbf{p}(\mathbf{x}, \boldsymbol{\xi}))$$

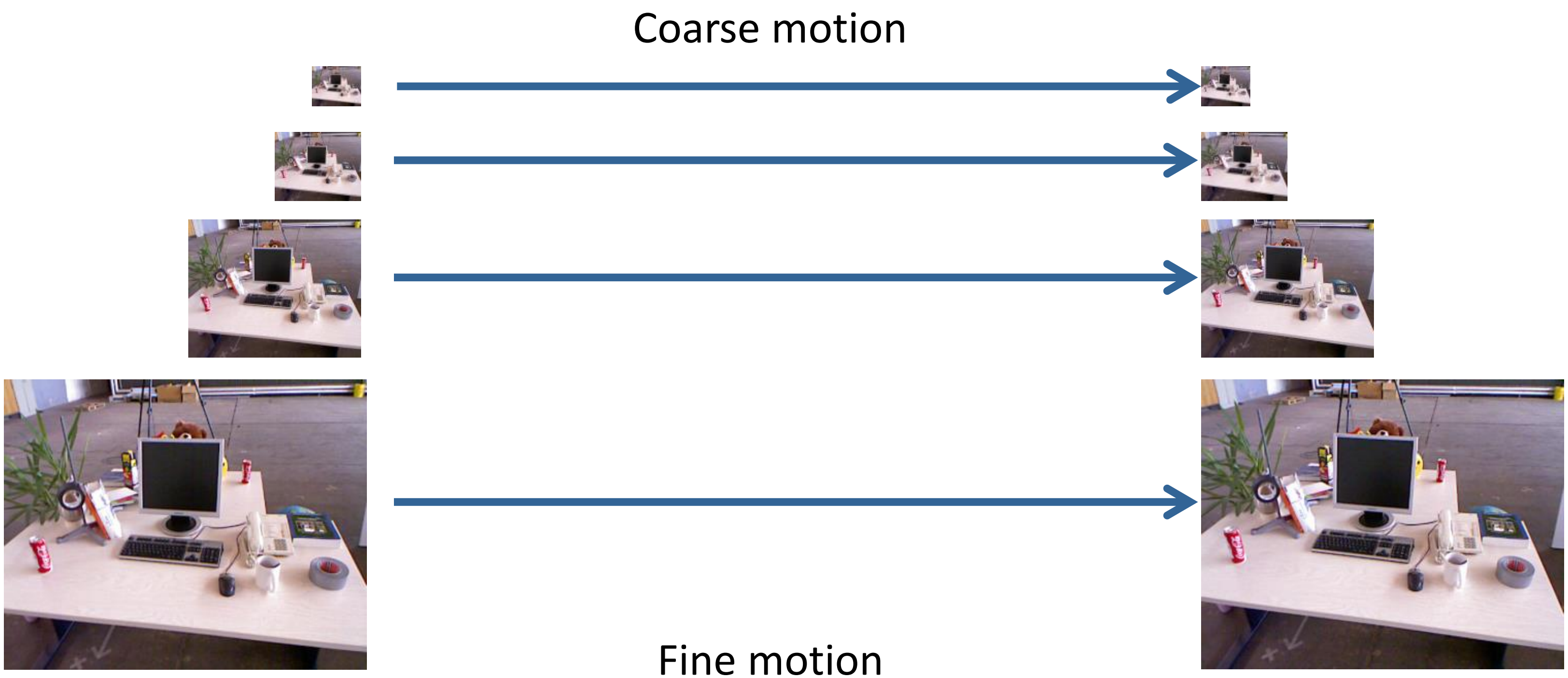
with $\mathbf{p}(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{T}(\boldsymbol{\xi}) Z(\mathbf{x}) K^{-1} \overline{\mathbf{x}}$

$$r(\mathbf{x}, \boldsymbol{\xi}) = I_1(\mathbf{x}) - I_2(\pi(\mathbf{p}(\mathbf{x}, \boldsymbol{\xi})))$$

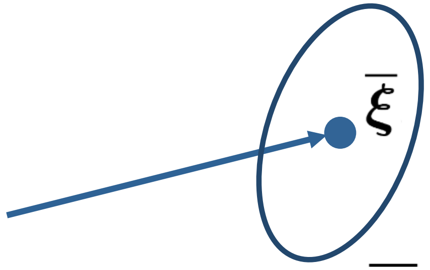
- Linearization is only valid for motions that change the projection in a small image neighborhood (where the gradient hints into the direction)

Coarse-To-Fine

- Adapt size of the neighborhood from coarse to fine



Covariance of the Pose Estimate



- Non-linear least squares determines a Gaussian estimate

$$p(\boldsymbol{\xi} \mid I_1, I_2) = \mathcal{N}\left(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\Sigma}}_{\boldsymbol{\xi}}\right)$$

$$\overline{\boldsymbol{\Sigma}}_{\boldsymbol{\xi}} = \left(\nabla_{\boldsymbol{\xi}} r(\overline{\boldsymbol{\xi}})^T \mathbf{W} \nabla_{\boldsymbol{\xi}} r(\overline{\boldsymbol{\xi}})\right)^{-1}$$

- Due to pose decomposition, we have to change the coordinate frame of the covariance using the adjoint in SE(3)

$$p(\boldsymbol{\xi} \mid I_1, I_2) = \mathcal{N}\left(\overline{\boldsymbol{\xi}}, \mathrm{ad}_{\mathbf{T}(\overline{\boldsymbol{\xi}})} \overline{\boldsymbol{\Sigma}}_{\boldsymbol{\delta\xi}} \mathrm{ad}_{\mathbf{T}(\overline{\boldsymbol{\xi}})}^T\right)$$

$$\overline{\boldsymbol{\Sigma}}_{\boldsymbol{\delta\xi}} = \left(\nabla_{\boldsymbol{\delta\xi}} r(\boldsymbol{\delta\xi} = 0, \overline{\boldsymbol{\xi}})^T \mathbf{W} \nabla_{\boldsymbol{\delta\xi}} r(\boldsymbol{\delta\xi} = 0, \overline{\boldsymbol{\xi}})\right)^{-1}$$

$$\mathrm{ad}_{\mathbf{T}} = \begin{pmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} \in \mathbb{R}^{6 \times 6}$$

Levenberg-Marquardt

- Idea: damp Gauss-Newton algorithm

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \left(\nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi}) + \lambda \, \mathbf{I}\right)^{-1} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} r(\boldsymbol{\xi})$$

- More adaptive component-wise damping:

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \Big(\nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi}) + \lambda \operatorname{diag}(\nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi}))\Big)^{-1} \nabla_{\boldsymbol{\xi}} r(\boldsymbol{\xi})^T \mathbf{W} r(\boldsymbol{\xi})$$

- Hybrid between Newton method ($\lambda = 0$) and gradient descent with step size $1/\lambda$ (for $\lambda \to \infty$)

- Start with e.g. $\lambda = 0.1$ and update λ in each iteration
- decrease λ in case of successful update (decreased error)
- increase λ in case of unsuccessful update (increased error)

Part 3: Lessons Learned

- The SE(3) Lie algebra is an elegant way of motion representation, especially for gradient-based optimization of motion parameters

- Non-linear least squares optimization is a versatile tool that can be applied for direct image alignment

- Iteratively Reweighted Least Squares allows for overcoming the limitation of basic least squares on the Gaussian residual distribution/L2 loss on the residuals

- Dense RGB-D odometry through direct image alignment can be implemented in a non-linear least squares framework.
  - The linear approximation of the residuals requires a coarse-to-fine optimization scheme
  - Non-linear least squares also provides the pose covariance

Questions ?