Computer Vision Group
Prof. Daniel Cremers

Technische Universität München

Practical Course: Vision-based Navigation Summer Term 2015

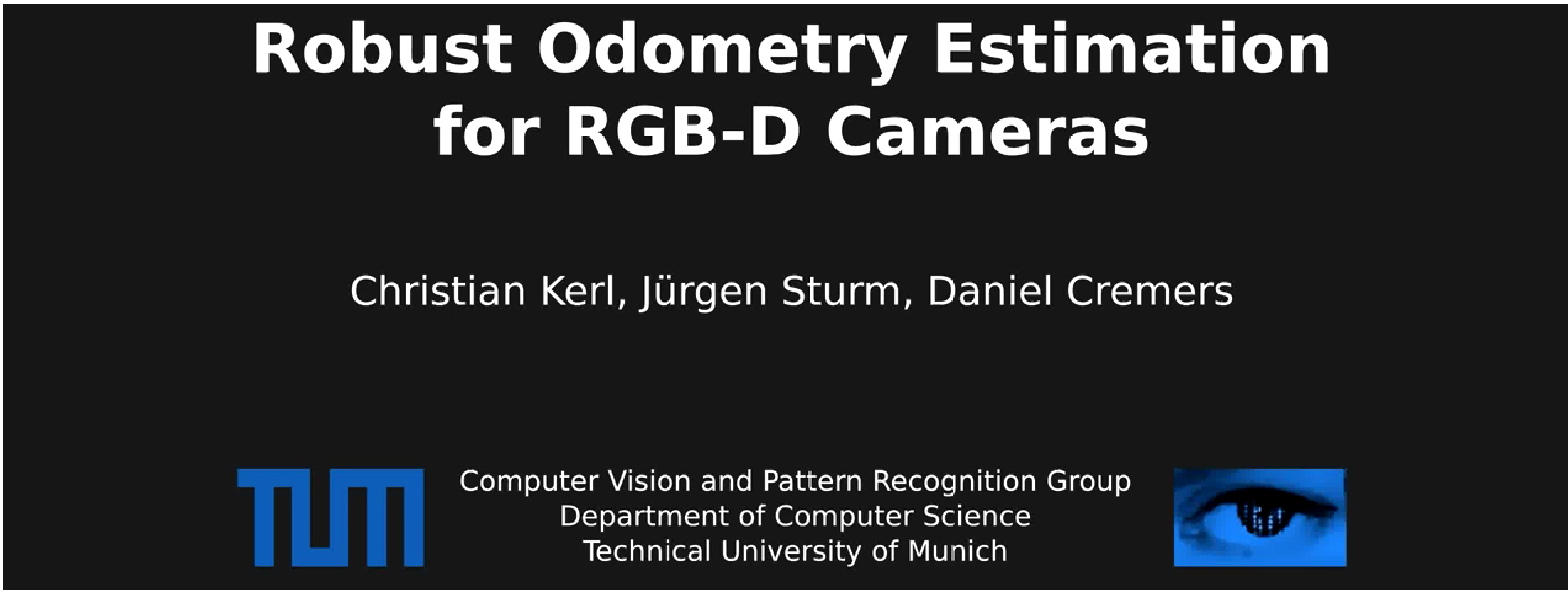
Lecture 3: Visual Motion Estimation, Direct Dense Methods

Dr. Jörg Stückler

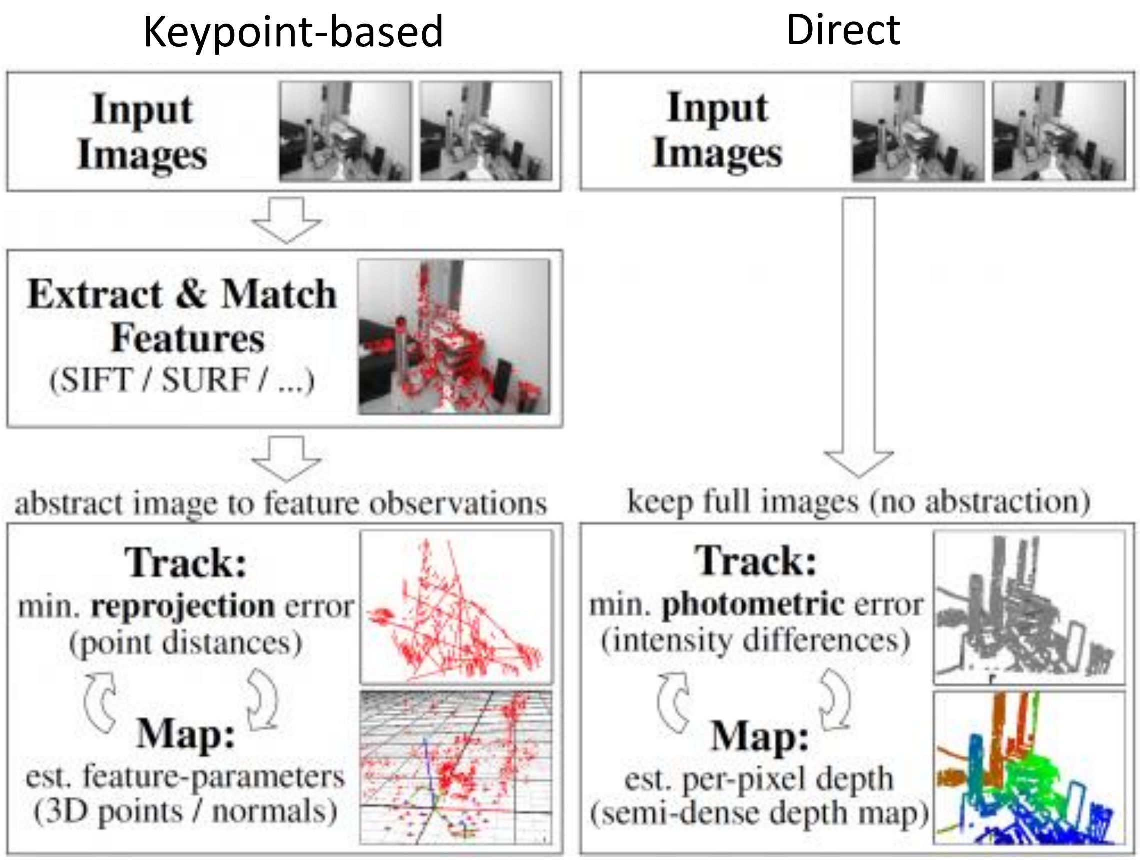
What we will cover today

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

Direct Visual Odometry with RGB-D Cameras

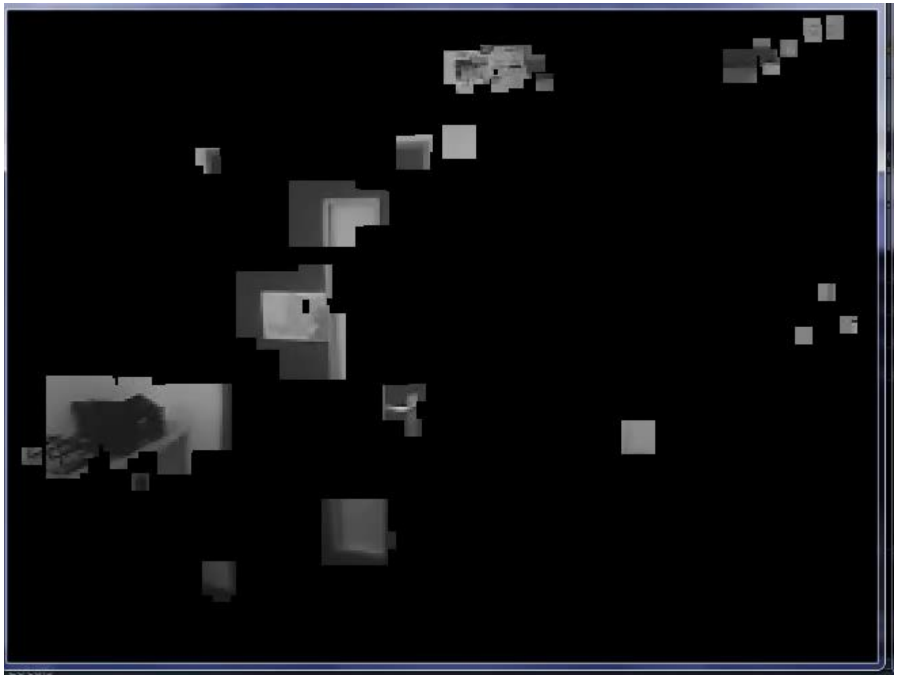


Keypoint-based vs. Direct VO Methods



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

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

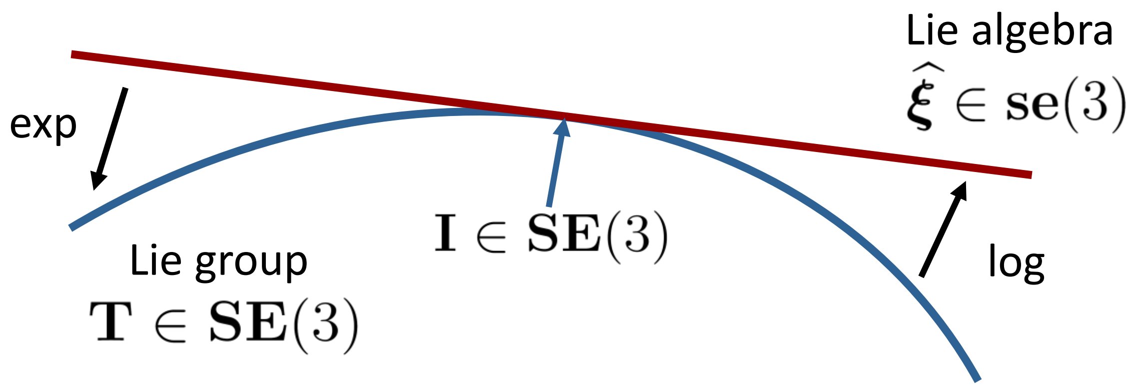
- Let's say we want to optimize a cost function $E(\theta)$ for the pose θ in some parametrization
- We need to set $\nabla_\theta E(\theta) = 0$

  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



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

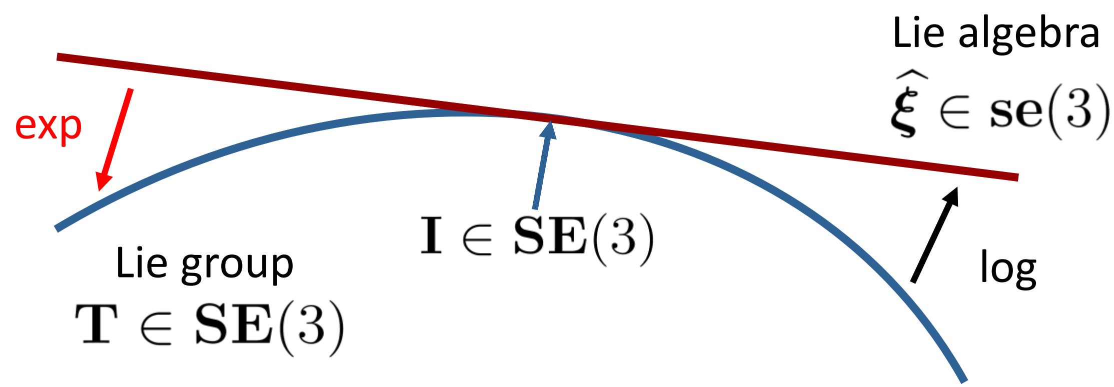
$$\boldsymbol{\omega} \in \mathbb{R}^3 \qquad \widehat{\boldsymbol{\omega}} := [\boldsymbol{\omega}]_\times$$

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

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

- 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)

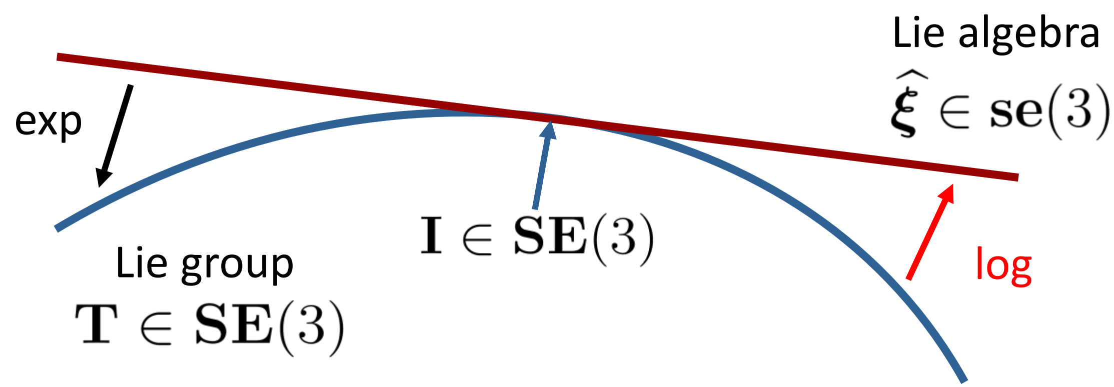


- The exponential map finds the transformation matrix for a twist:

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

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

Logarithm Map of SE(3)



- The logarithm maps twists to transformation matrices:

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

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

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):

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

- Gradient descent operates on the auxiliary variable $\boldsymbol{\delta\xi}$

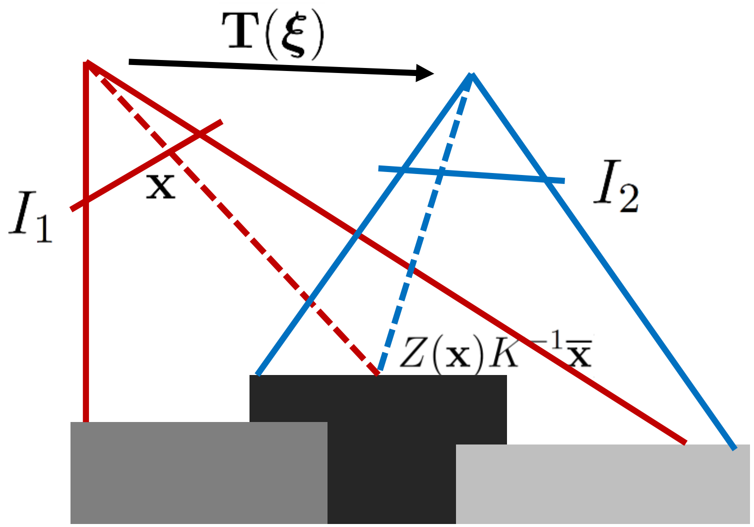
$$\boldsymbol{\delta\xi} \leftarrow \mathbf{0} - \nabla_{\boldsymbol{\delta\xi}} E(\boldsymbol{\delta\xi})$$

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

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 $\boldsymbol{\xi}$, 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

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

- 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

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

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

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

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

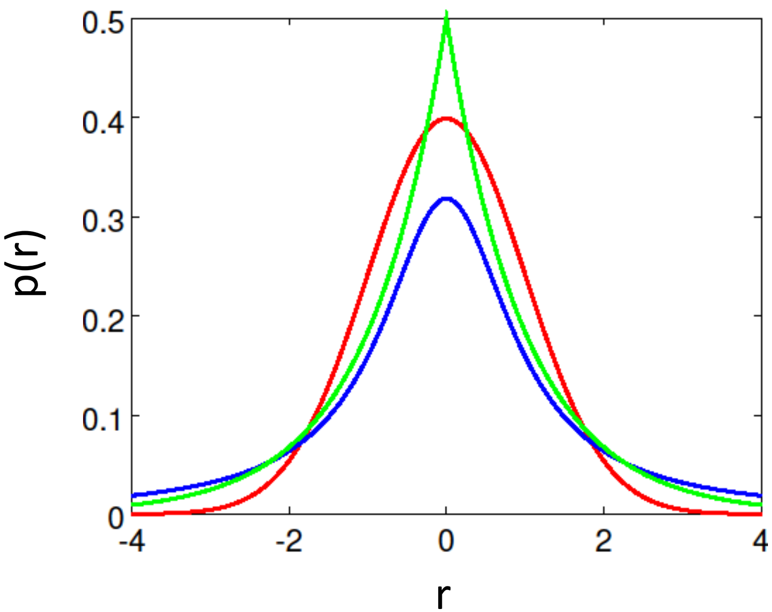
  - Solve linearized system

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

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

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

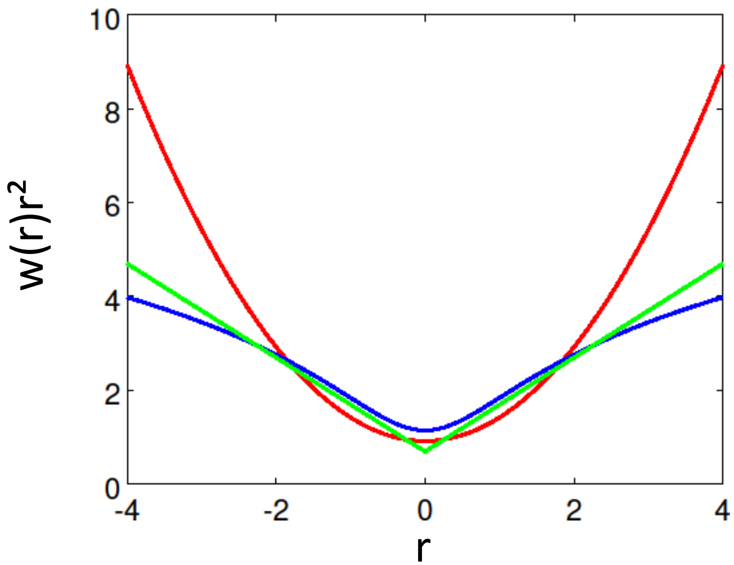
Actual Residual Distribution



- 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

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

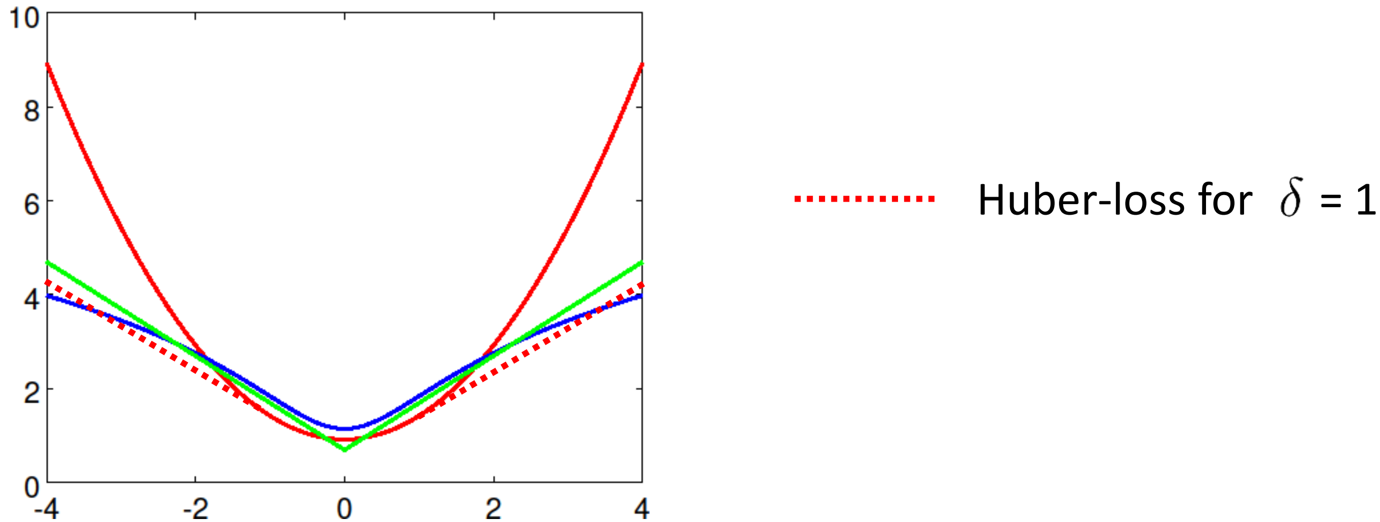
E.g., for Laplace distribution:

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

Huber-Loss

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

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



Linearization of Image Alignment Residuals

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

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

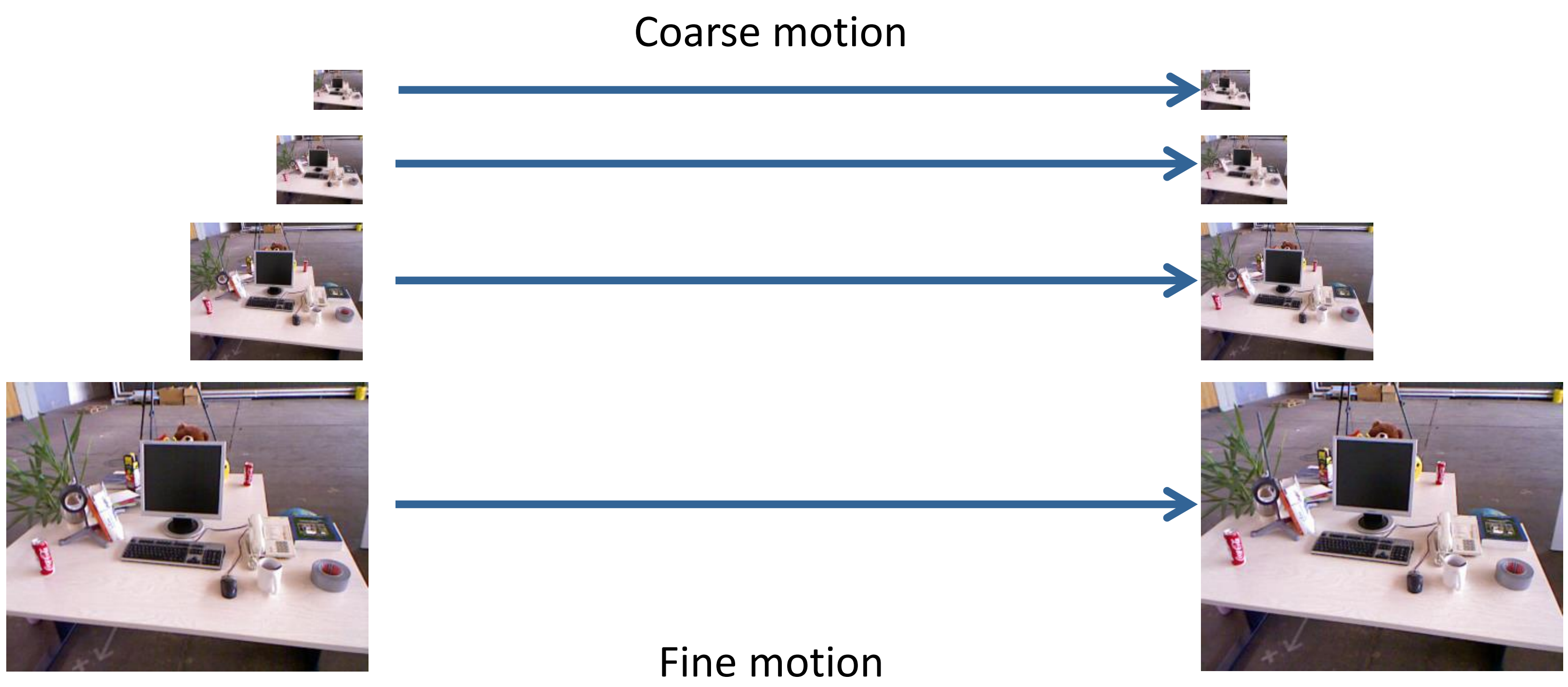
with $\mathbf{p}(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{T}(\boldsymbol{\xi}) Z(\mathbf{x}) K^{-1} \overline{\mathbf{x}}$

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

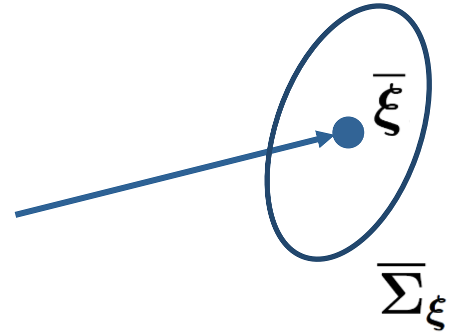
- Linearization is only valid for motions that change the projection in a small image neighborhood (where the gradient hints into the direction)

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)

$$p(\boldsymbol{\xi} \mid I_1, I_2) = \mathcal{N}\left(\overline{\boldsymbol{\xi}}, \mathrm{ad}_{\mathbf{T}(\overline{\boldsymbol{\xi}})} \overline{\boldsymbol{\Sigma}}_{\boldsymbol{\delta\xi}} \mathrm{ad}^T_{\mathbf{T}(\overline{\boldsymbol{\xi}})}\right)$$
$$\overline{\boldsymbol{\Sigma}}_{\boldsymbol{\delta\xi}} = \left(\nabla_{\boldsymbol{\delta\xi}} r(\boldsymbol{\delta\xi} = 0, \overline{\boldsymbol{\xi}})^T \mathbf{W} \nabla_{\boldsymbol{\delta\xi}} r(\boldsymbol{\delta\xi} = 0, \overline{\boldsymbol{\xi}})\right)^{-1}$$
$$\mathrm{ad}_{\mathbf{T}} = \begin{pmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} \in \mathbb{R}^{6 \times 6}$$

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 ?