Robotic 3D Vision

Lecture 8: Visual Odometry 3 –Direct Methods

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

What We Will Cover Today

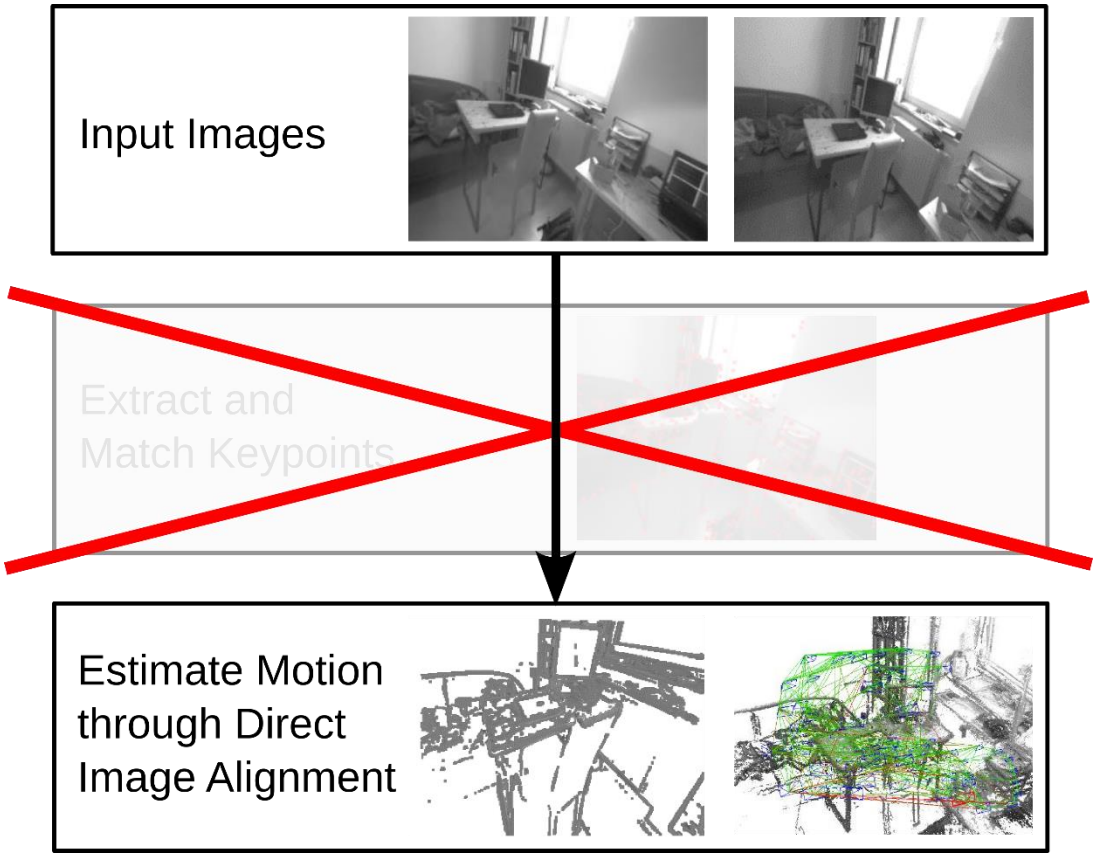
- Direct visual odometry methods
  - Principles of direct image alignment
  - Photometric alignment
  - Geometric alignment
- Direct visual odometry for RGB-D cameras
- Direct visual odometry for monocular cameras
  - Semi-dense monocular odometry
- Photometric calibration
- Stereo extensions

Direct Visual Odometry Pipeline

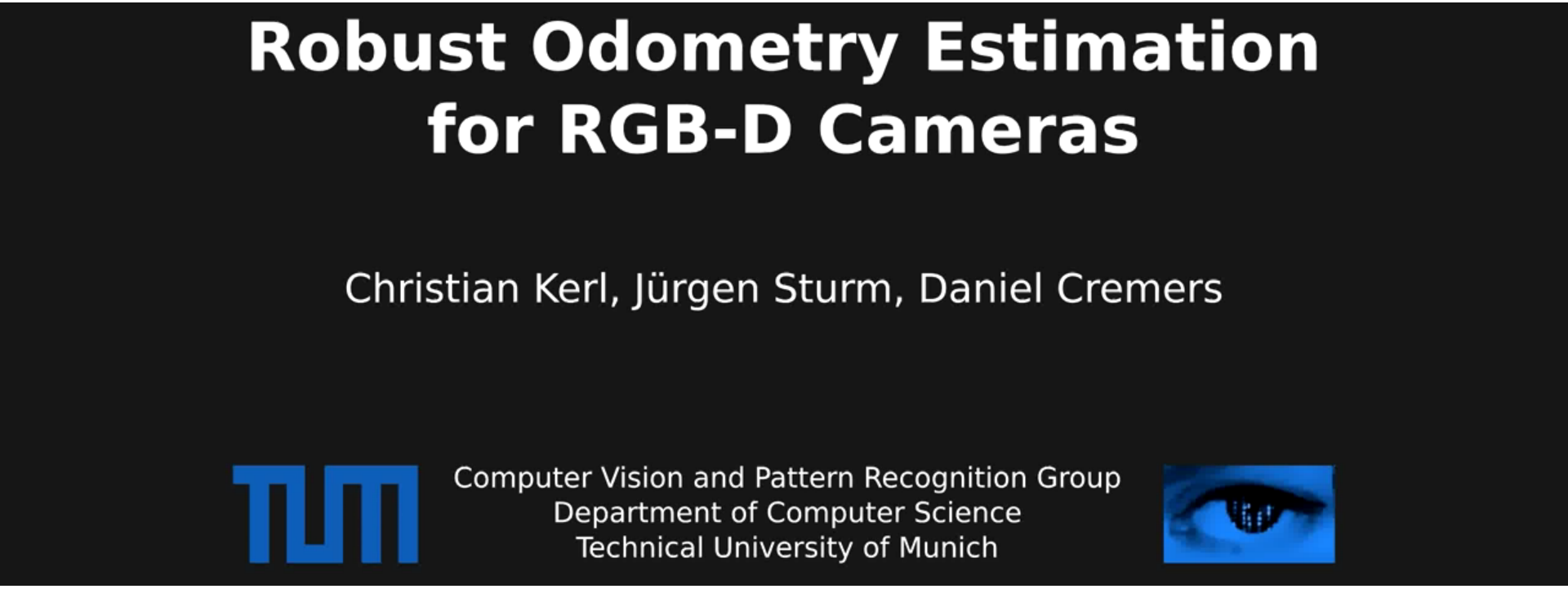
- Avoid manually designed keypoint detection and matching
- Instead: direct image alignment

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

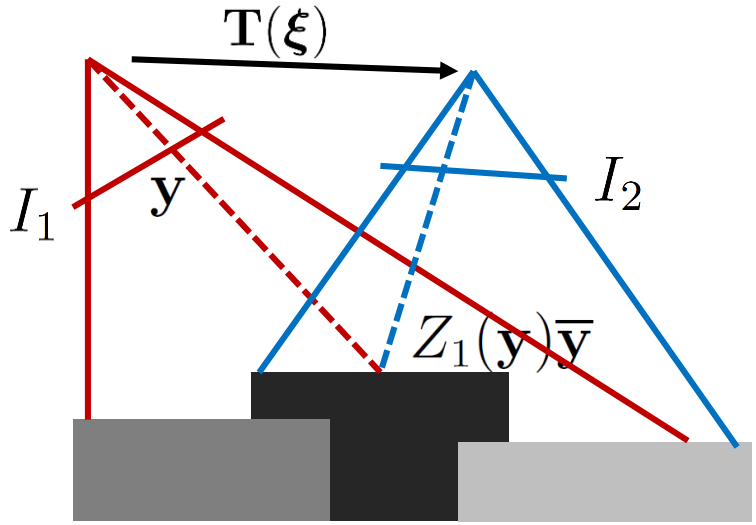
- Warping requires depth
  - RGB-D
  - Fixed-baseline stereo
  - Temporal stereo, tracking and (local) mapping



Direct Visual Odometry Example (RGB-D)



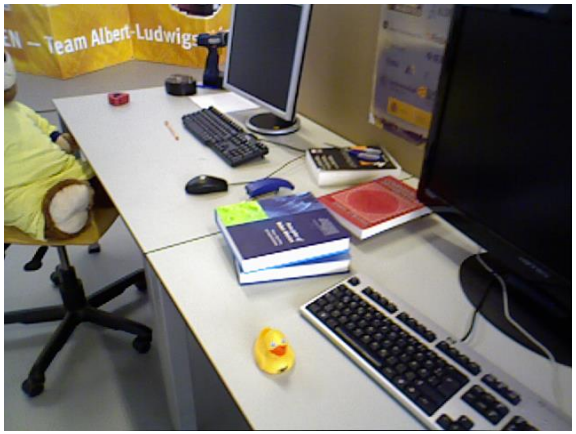
Direct Image Alignment Principle



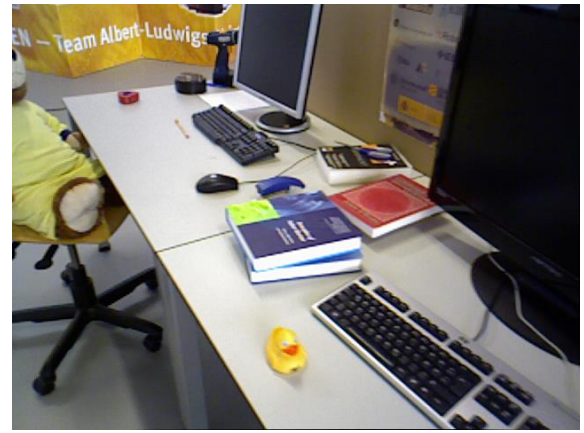
- If we know pixel depth, we can „simulate“ an image from a different view point
- Ideally, the warped image is the same as the image taken from that pose:

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

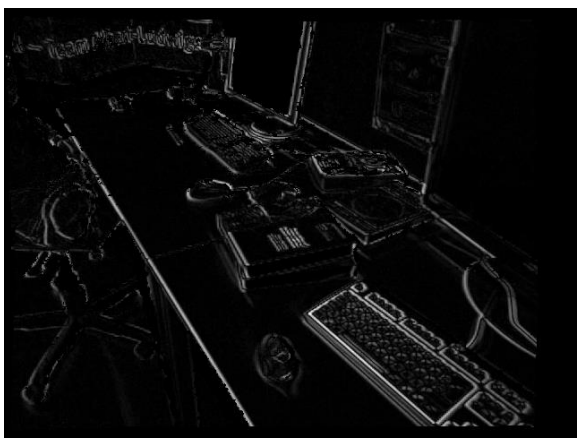
Derivative of Image Warp



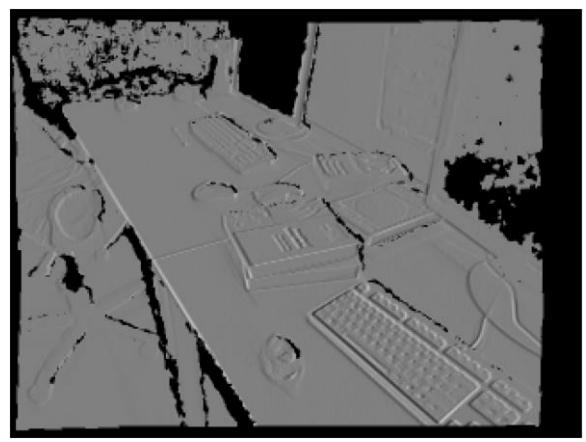
I_1



I_2



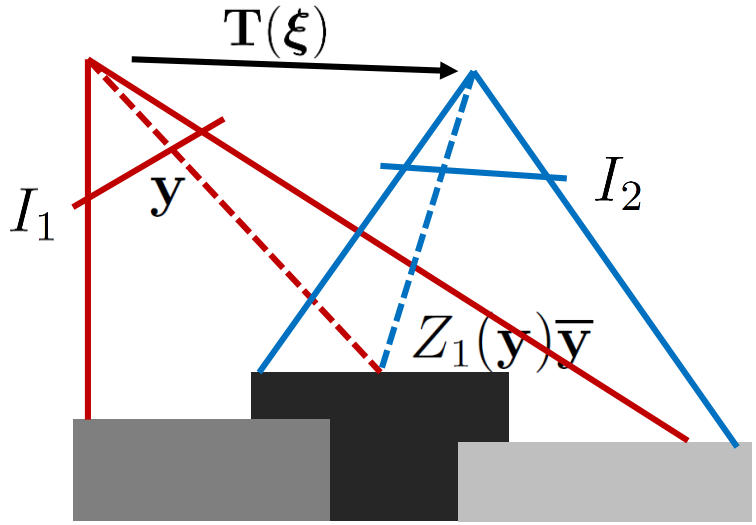
I_1 - I_2



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

Images from Kerl et al., ICRA 2013

Direct RGB-D Image Alignment



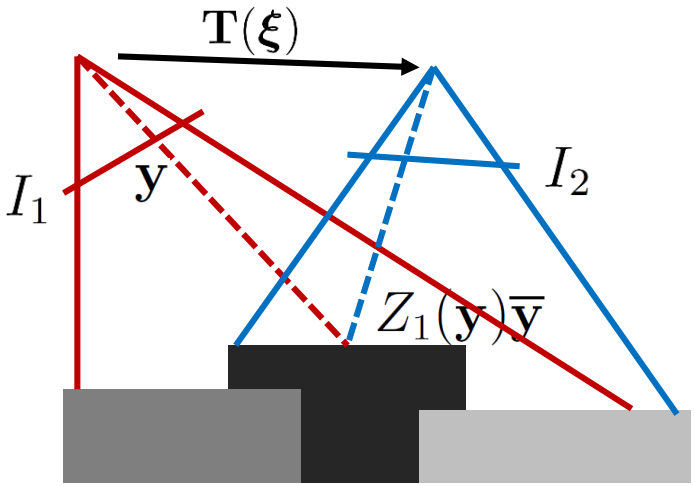
- RGB-D cameras measure depth, we only need to estimate camera motion!
- In addition to the photometric error

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

we can measure geometric error directly

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

Probabilistic Direct Image Alignment



- Measurements are affected by noise

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

- A convenient assumption is Gaussian noise

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

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

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

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

Optimization Approach

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

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

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

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

Recap: Gauss-Newton Method

- Approximate Newton's method to minimize E(x)
  - Approximate E(x) through linearization of residuals

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

$$= \frac{1}{2}\left(\mathbf{r}(\mathbf{x}_k) + \mathbf{J}_k\left(\mathbf{x}-\mathbf{x}_k\right)\right)^\top \mathbf{W}\left(\mathbf{r}(\mathbf{x}_k) + \mathbf{J}_k\left(\mathbf{x}-\mathbf{x}_k\right)\right) \qquad \mathbf{J}_k := \nabla_{\mathbf{x}}\mathbf{r}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k}$$

$$= \frac{1}{2}\mathbf{r}(\mathbf{x}_k)^\top \mathbf{W}\mathbf{r}(\mathbf{x}_k) + \underbrace{\mathbf{r}(\mathbf{x}_k)^\top \mathbf{W}\mathbf{J}_k}_{=:\mathbf{b}_k^\top}\left(\mathbf{x}-\mathbf{x}_k\right) + \frac{1}{2}\left(\mathbf{x}-\mathbf{x}_k\right)^\top \underbrace{\mathbf{J}_k^\top \mathbf{W}\mathbf{J}_k}_{=:\mathbf{H}_k}\left(\mathbf{x}-\mathbf{x}_k\right)$$

  - Find root of $\nabla_{\mathbf{x}}\widetilde{E}(\mathbf{x}) = \mathbf{b}_k^\top + \left(\mathbf{x}-\mathbf{x}_k\right)^\top \mathbf{H}_k$ using Newton's method, i.e.

$$\nabla_{\mathbf{x}}\widetilde{E}(\mathbf{x}) = \mathbf{0} \text{ iff } \mathbf{x} = \mathbf{x}_k - \mathbf{H}_k^{-1}\mathbf{b}_k$$

- Pros:
  - Faster convergence (approx. quadratic convergence rate)
- Cons:
  - Divergence if too far from local optimum (H not positive definite)
  - Solution quality depends on initial guess

Recap: Levenberg-Marquardt Method

- Gradually transition between gradient descent and Gauss-Newton
  - Augment Hessian approximation of Gauss-Newton (damping)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{H}_k + \lambda \mathbf{I}\right)^{-1} \mathbf{b}_k$$

  - Adaptive weighting: $\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{H}_k + \lambda \operatorname{diag}(\mathbf{H}_k)\right)^{-1} \mathbf{b}_k$
  - Start with $\lambda = 0.1$
  - Accept step and decrease lambda $\lambda \leftarrow \lambda/2$ if error function decreases, otherwise discard step and increase lambda $\lambda \leftarrow 2\lambda$ (akin line search)

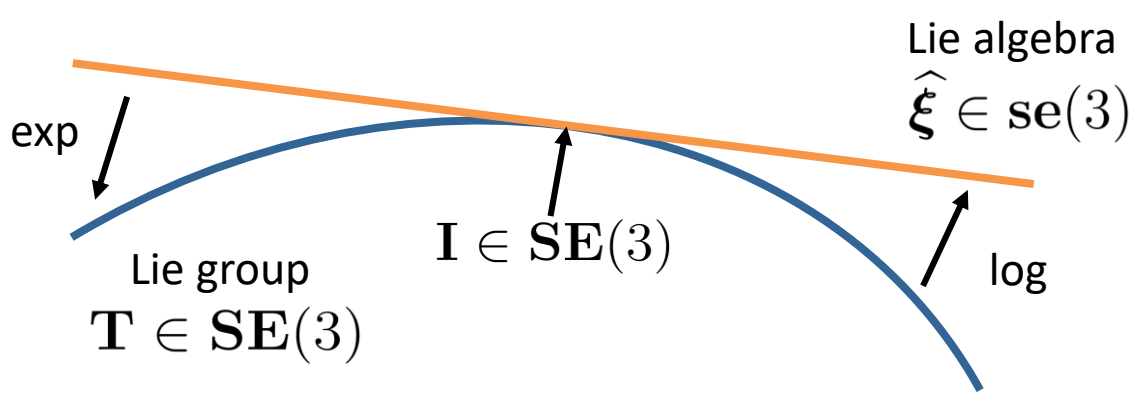
- Pros:
  - Fast convergence close to local optimum (quadratic convergence rate close to optimum)
  - More stable but slow convergence far from local optimum
- Cons:
  - Solution quality depends on initial guess

Pose Parametrization for Optimization

- Requirements on pose parametrization
  - No singularities
  - Minimal to avoid constraints

- Various pose parametrizations available
  - Direct matrix representation => not minimal
  - Quaternion / translation => not minimal
  - Euler angles / translation => singularities
  - Twist coordinates of elements in Lie Algebra se(3) of SE(3) (axis-angle / translation)

Recap: Representing Motion using Lie Algebra se(3)



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

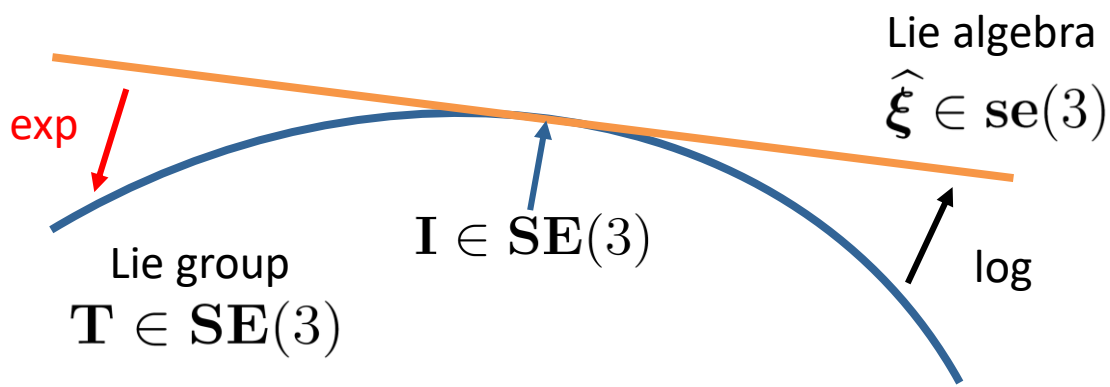
$$\boldsymbol{\omega} \in \mathbb{R}^3$$

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

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

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

Recap: Exponential Map of SE(3)

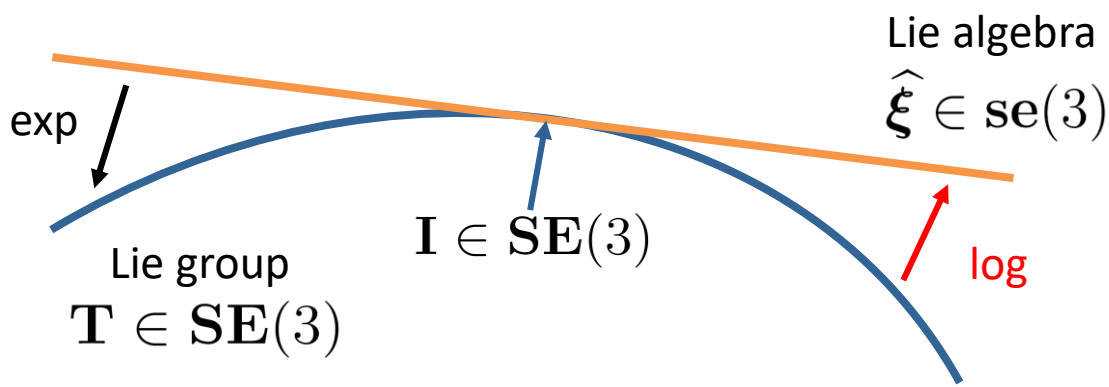


- The exponential map finds the transformation matrix for a twist:

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

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

Recap: Logarithm Map of SE(3)



- The logarithm maps transformation matrices to twists:

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

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

Recap: Some Notation for Twist Coordinates

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

Optimization with Twist Coordinates

- Twists provide a minimal local representation without singularities

- Since $\mathbf{SE}(3)$ a smooth manifold, we can decompose transformations in each optimization step into the transformation itself and an infinitesimal increment

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

But!

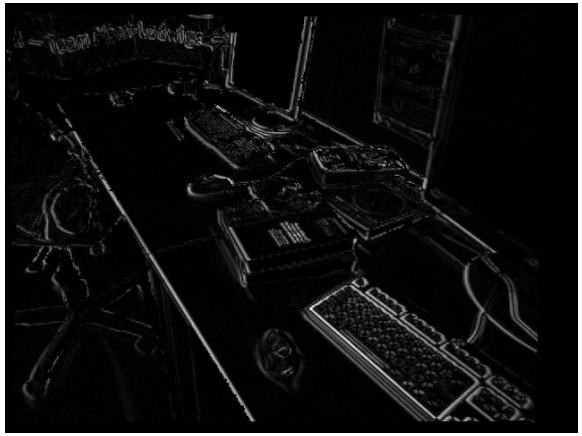
$$\mathbf{T}(\boldsymbol{\xi} + \boldsymbol{\delta\xi}) \neq \mathbf{T}(\boldsymbol{\xi})\,\mathbf{T}(\boldsymbol{\delta\xi})$$

- Example: Gradient descent on the auxiliary variable

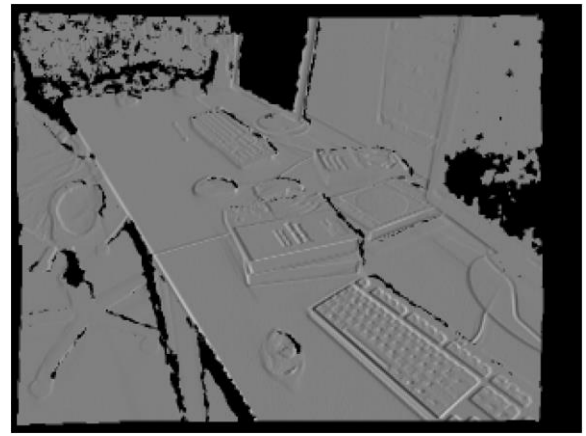
$$\boldsymbol{\delta\xi}^* = \mathbf{0} - \eta \nabla_{\boldsymbol{\delta\xi}} E(\boldsymbol{\xi}_i, \boldsymbol{\delta\xi})$$

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

Properties of Residual Linearization



I_1 - I_2



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

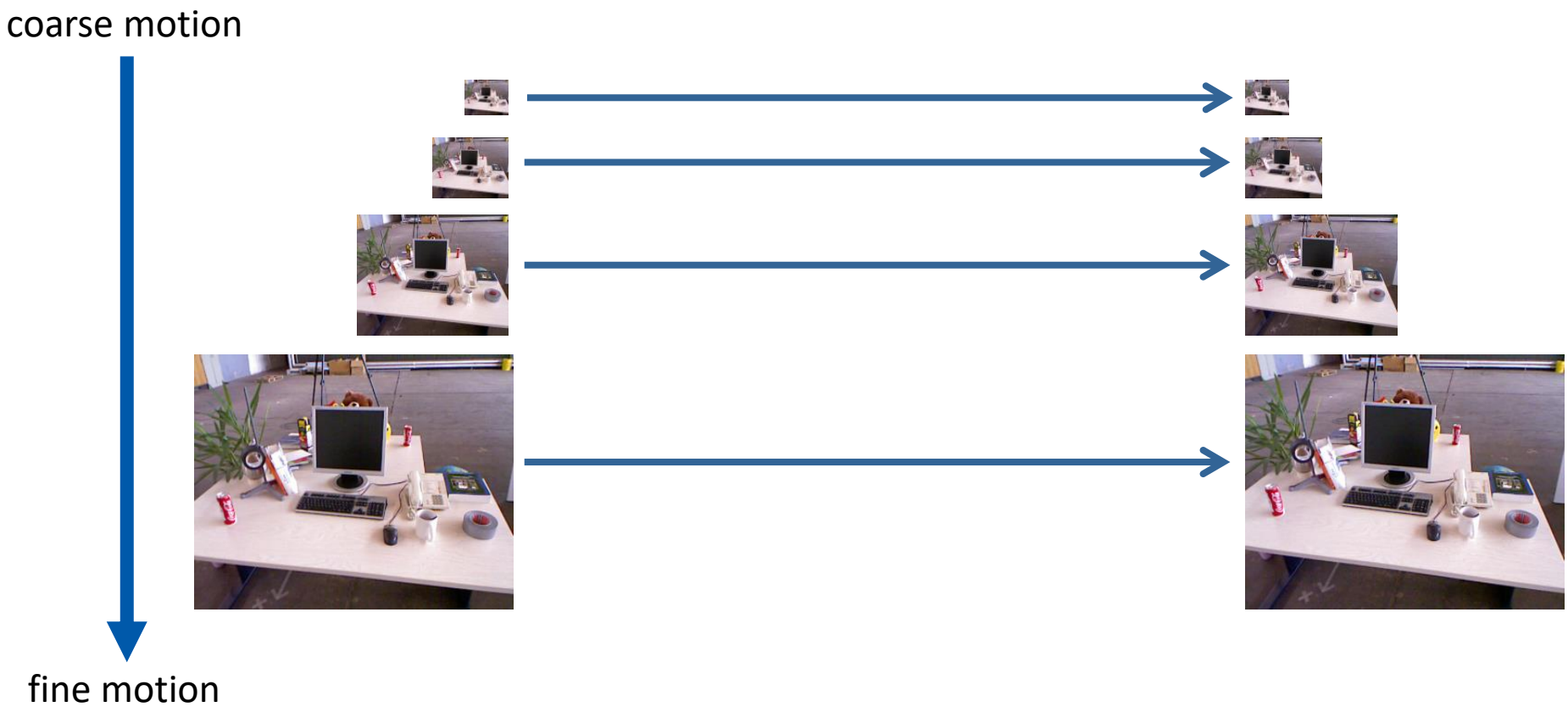
- Gradient of residuals wrt pose

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

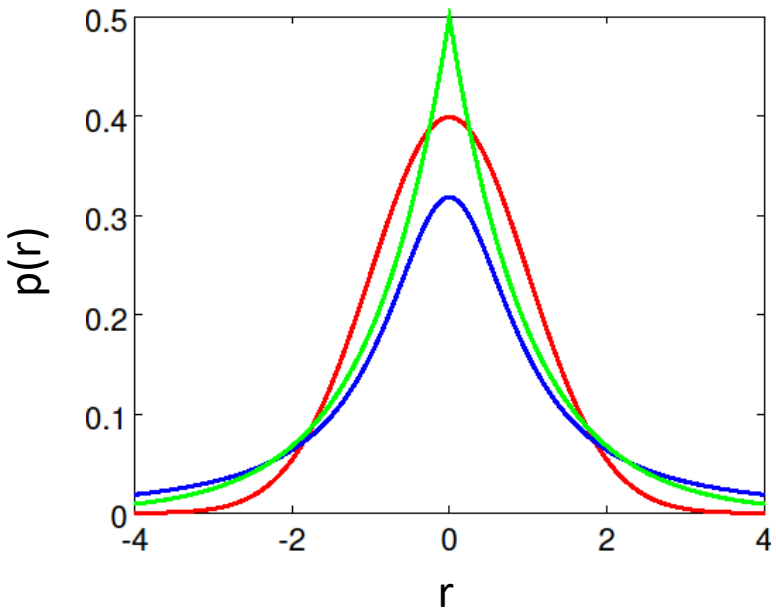
with $\omega(\mathbf{y},\boldsymbol{\xi}) := \pi(\mathbf{T}(\boldsymbol{\xi})Z_1(\mathbf{y})\overline{\mathbf{y}})$

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

Coarse-To-Fine Optimization



Residual Distributions

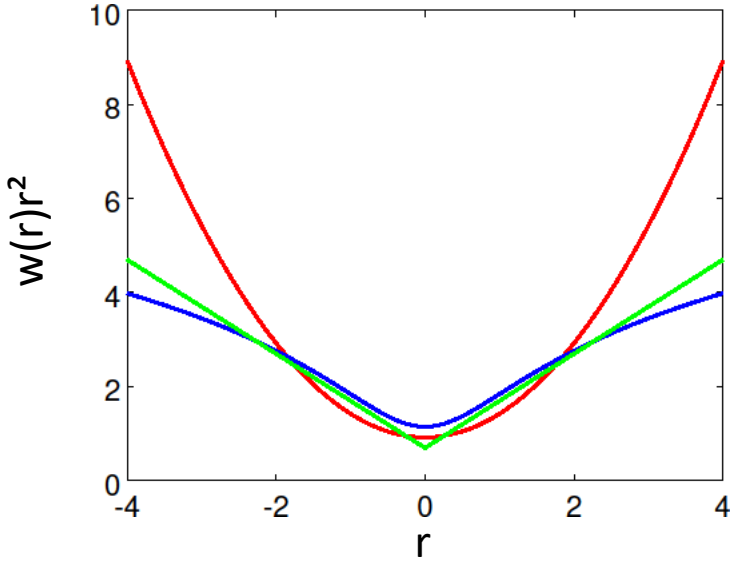


- Normal distribution
- Laplace distribution
- Student-t distribution

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

Images from Kerl et al., ICRA 2013

Optimizing Non-Gaussian Measurement Noise



- Normal distribution
- Laplace distribution
- Student-t distribution

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

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

Laplace distribution:

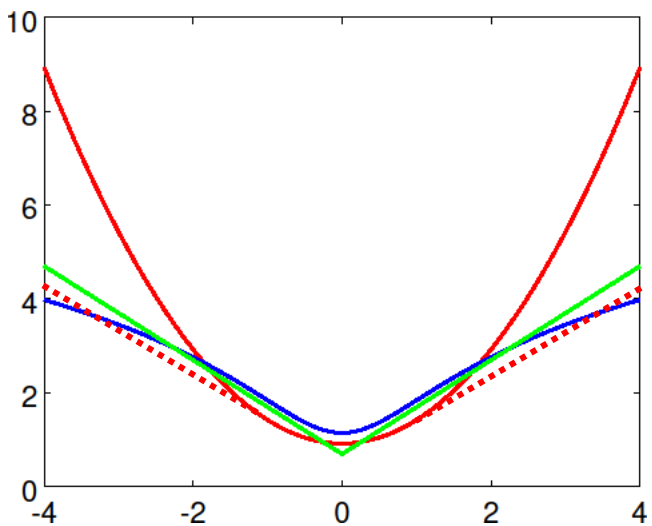
$$w\left(r(\mathbf{y},\boldsymbol{\xi})\right) = \left|r(\mathbf{y},\boldsymbol{\xi})\right|^{-1}$$

- Keep weights constant in each Gauss-Newton iteration

Huber Loss

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

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



- Normal distribution
- Laplace distribution
- Student-t distribution

Huber-loss for δ = 1

Efficient Non-Linear Least Squares

- Gauss-Newton / Levenberg-Marquardt can be applied very efficiently to direct image alignment:
  - $\mathbf{H}_i$ is only a 6x6 matrix
  - $\mathbf{b}_i = \mathbf{J}_i^\top \mathbf{W}\mathbf{r}(\boldsymbol{\xi}_i)$ is a 6x1 vector
  - Since we treat each pixel stochastically independent from neighboring pixels, $\mathbf{H}_i$ and $\mathbf{b}_i$ are summed over individual pixels

$$\mathbf{H}_i = \sum_{\mathbf{y}\in\Omega} \frac{w(\mathbf{y}, \boldsymbol{\xi}_i)}{\sigma_I^2} \mathbf{J}_{i,\mathbf{y}}^\top \mathbf{J}_{i,\mathbf{y}} \qquad \mathbf{b}_i = \sum_{\mathbf{y}\in\Omega} \mathbf{J}_{i,\mathbf{y}}^\top \frac{w(\mathbf{y}, \boldsymbol{\xi}_i)}{\sigma_I^2} r(\mathbf{y}, \boldsymbol{\xi}_i)$$

$$\mathbf{J}_{i,\mathbf{y}} := \nabla_{\delta\boldsymbol{\xi}} r(\mathbf{y}, \boldsymbol{\delta\xi} \oplus \boldsymbol{\xi}_i)$$

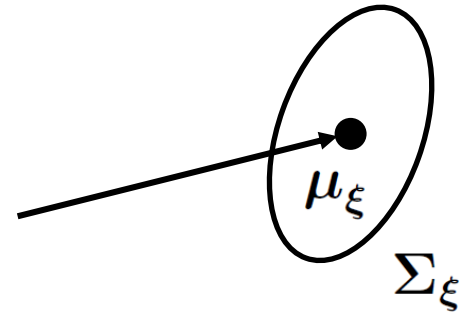
  - This allows for highly efficient parallel processing, e.g. using a GPU

Distribution of the Pose Estimate

- Non-linear least squares determines a Gaussian estimate

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

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



- Due to right-multiplication of pose increment $\boldsymbol{\delta\xi}$, covariance from Hessian is expressed in camera frame of I_1
- Pose covariance in frame of I_2 can be obtained using the adjoint in $\mathbf{SE}(3)$

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

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

$$\mathrm{ad}_{\mathbf{T}(\boldsymbol{\xi})} = \begin{pmatrix} \mathbf{R}(\boldsymbol{\xi}) & \mathbf{0} \\ \widehat{\boldsymbol{t}}\mathbf{R}(\boldsymbol{\xi}) & \mathbf{R}(\boldsymbol{\xi}) \end{pmatrix}$$

Algorithm: Direct RGB-D Visual Odometry

Input: RGB-D image sequence $I_{0:t}, Z_{0:t}$

Output: aggregated camera poses $\mathbf{T}_{0:t}$

Algorithm:

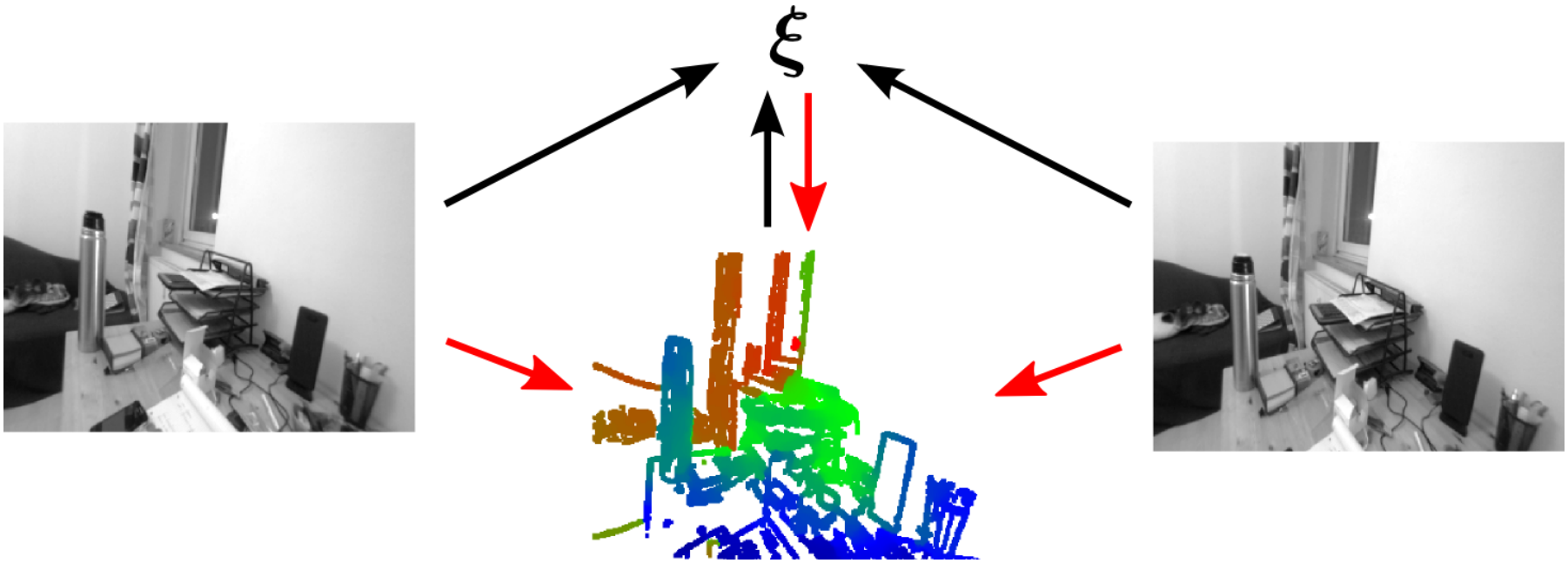
For each current RGB-D image I_k, Z_k :

1. Estimate relative camera motion $\mathbf{T}_k^{k-1}$ towards the previous RGB-D frame using direct image alignment
2. Concatenate estimated camera motion with previous frame camera pose to obtain current camera pose estimate

$$\mathbf{T}_k = \mathbf{T}_{k-1}\mathbf{T}_k^{k-1}$$

Monocular Direct Visual Odometry

- Estimate motion and depth concurrently

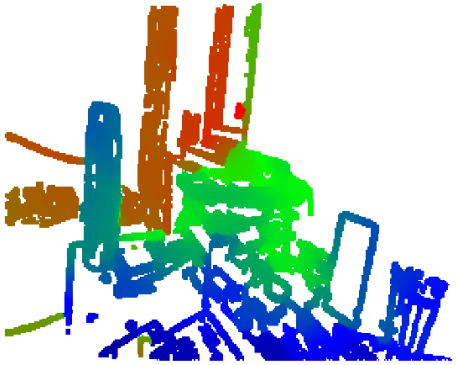


- Alternating optimization: **Tracking** and **Mapping**

Images from: Engel et al., ICCV 2013

Semi-Dense Mapping

- Estimate inverse depth and variance at high gradient pixels
- Correspondence search along epipolar line (5-pixel intensity SSD)

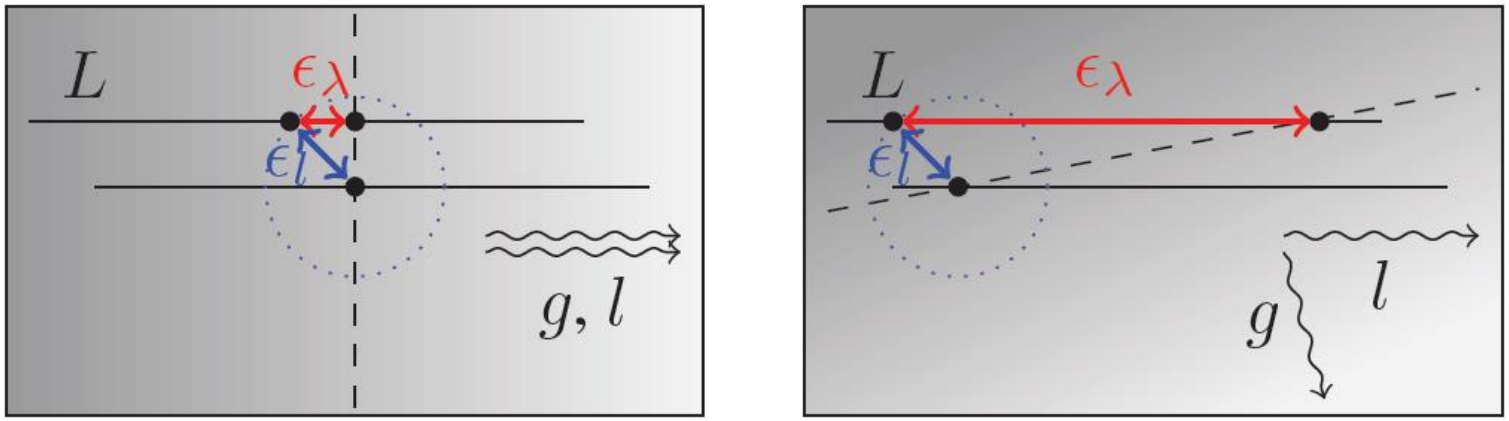


- Kalman-filtering of depth map:
  - Propagate depth map & variance from previous frame
  - Update depth map & variance with new depth observations

Images from: Engel et al., ICCV 2013

Semi-Dense Mapping

- Estimate for inverse depth uncertainty from geometric and intensity noise



Geometric noise

$$\sigma^2_{\lambda(\xi,\pi)} = \frac{\sigma_l^2}{\langle g, l \rangle^2}$$

pos. variance of epipolar line

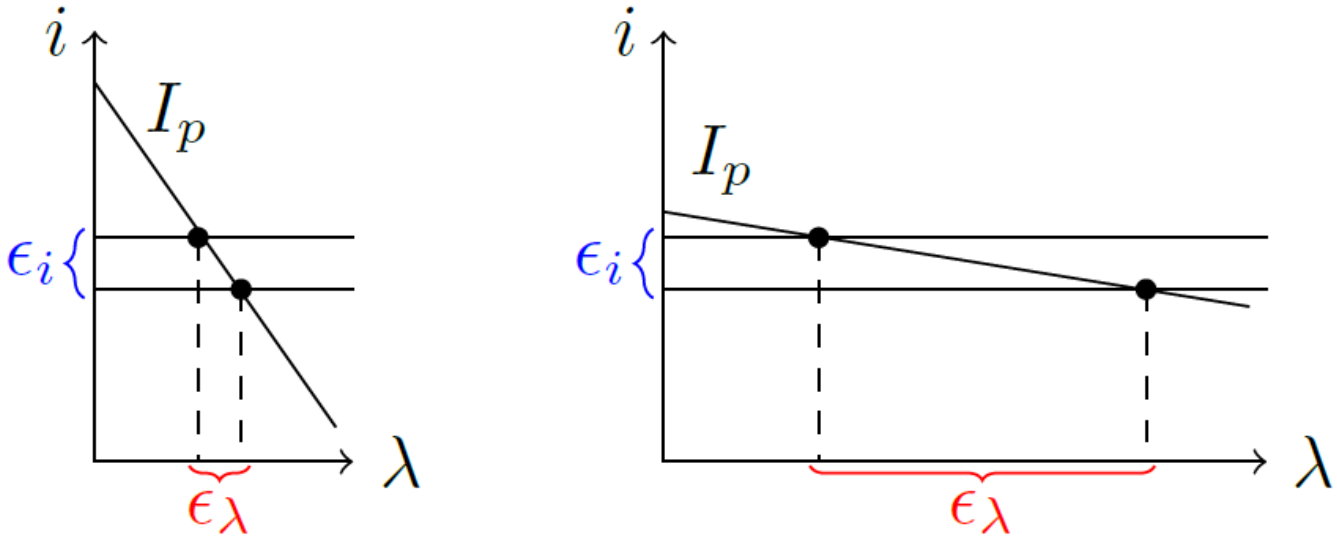
gradient direction

epipolar line direction

Images from: Engel et al., ICCV 2013

Semi-Dense Mapping

- Estimate for inverse depth uncertainty from geometric and intensity noise



Intensity noise

$$\sigma^2_{\lambda(I)} = \frac{2\sigma_i^2}{g_p^2}$$

intensity noise variance

image gradient magnitude at epipolar line

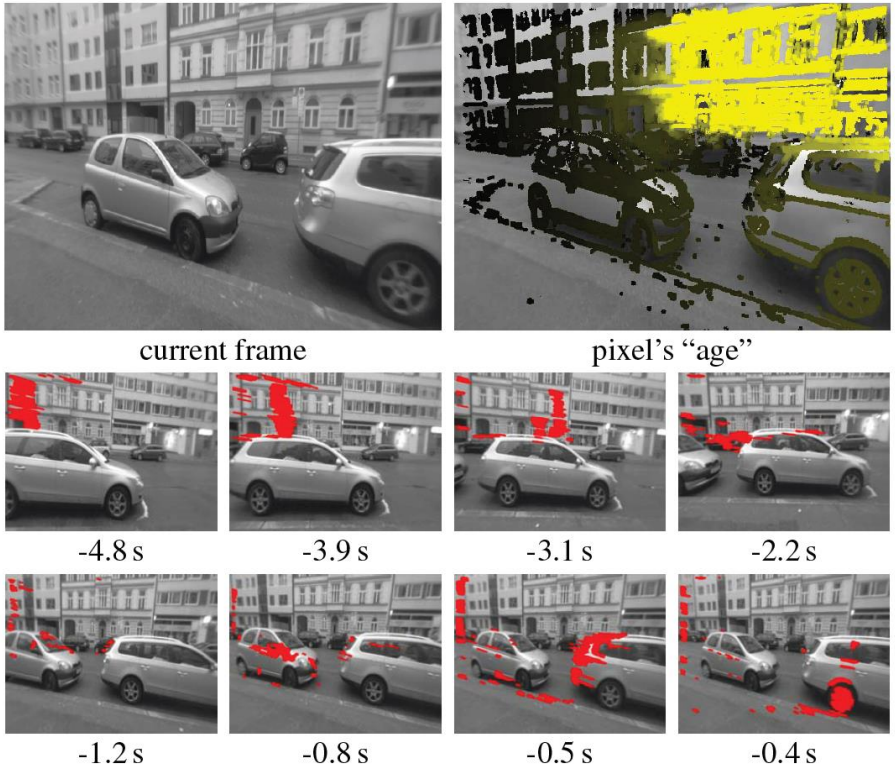
Images from: Engel et al., ICCV 2013

Choosing the Stereo Reference Frame

- Naive: use one specific reference frame (f.e. the previous frame or a keyframe)

- We can also select the reference frame for stereo comparisons for each pixel individually in order to achieve a trade-off between accuracy and computation time

Images from: Engel et al., ICCV 2013



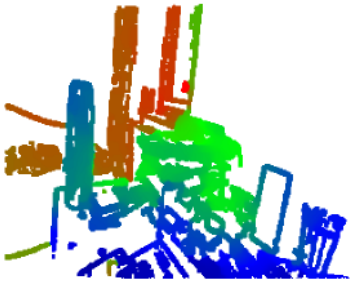
Heuristics from Engel et al., ICCV 2013:
Use oldest frame in which pixel still visible but disparity search range and observation angle below threshold

Semi-Dense Direct Image Alignment

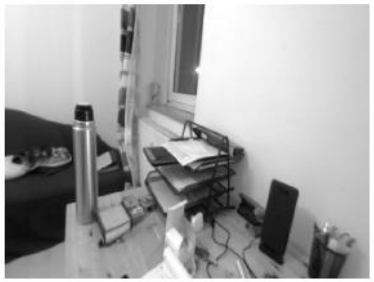
I_1

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

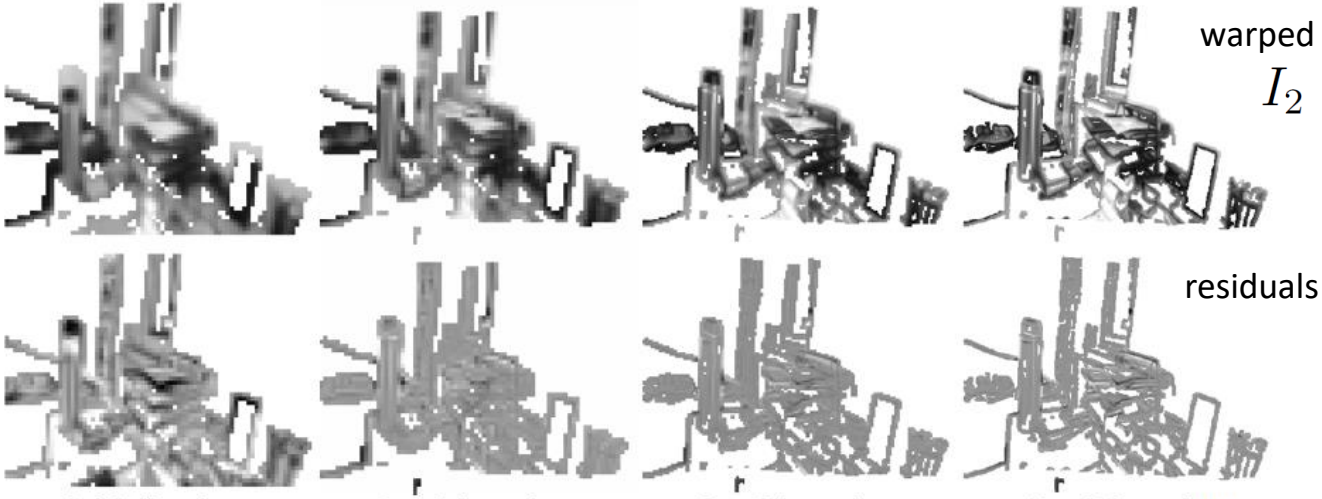
$$r(\mathbf{y}, \boldsymbol{\xi}) = I_1(\mathbf{y}) - I_2\left(\pi\left(\mathbf{T}(\boldsymbol{\xi}) Z_1(\mathbf{y}) \overline{\mathbf{y}}\right)\right)$$



Z_1



I_2



initialization on lvl 3 (80 × 60) | after 8 iterations on lvl 3 (80 × 60) | after 3 iterations on lvl 2 (160 × 120) | after 3 iterations on lvl 1 (320 × 240)

Images from: Engel et al., ICCV 2013

Algorithm: Direct Monocular Visual Odometry

Input: Monocular image sequence $I_{0:t}$
Output: aggregated camera poses $\mathbf{T}_{0:t}$

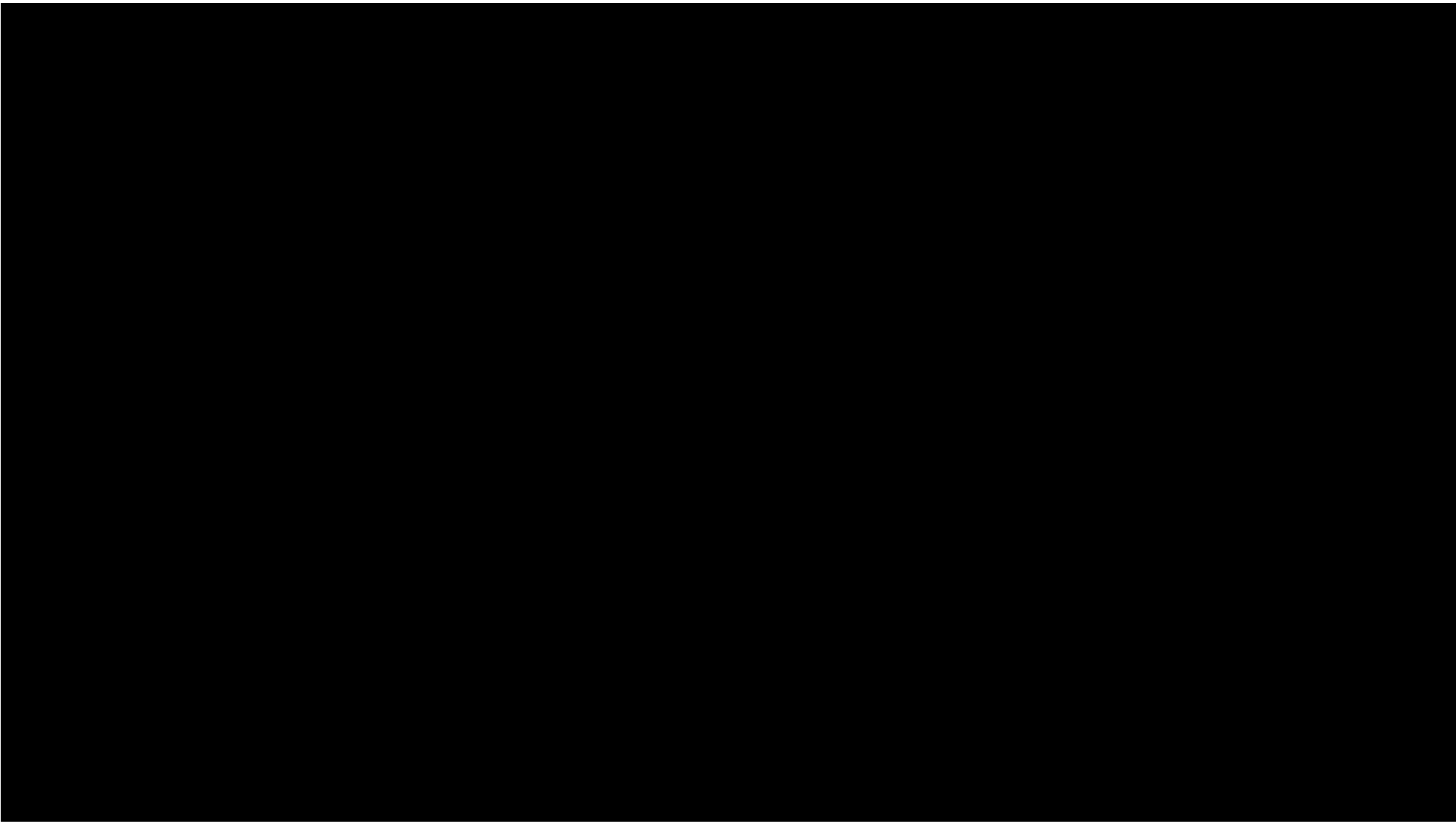
Algorithm:

Initialize depth map Z_0 f.e. from first two frames with a point-based method

For each current image I_k:

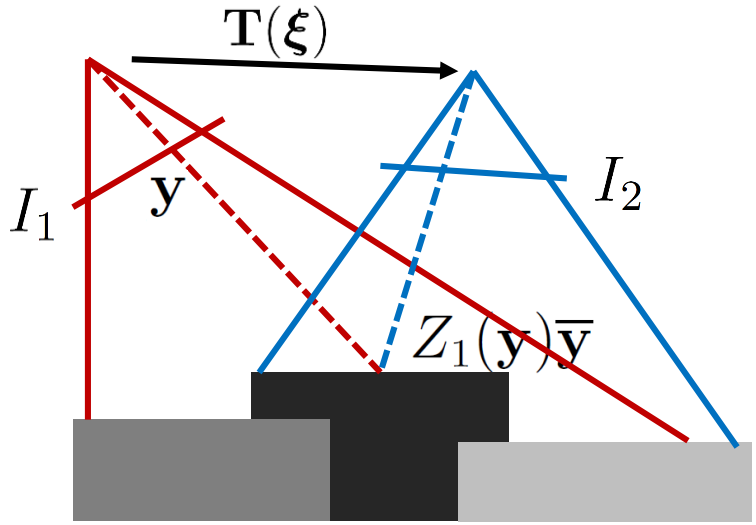
1. Estimate relative camera motion $\mathbf{T}_k^{k-1}$ towards the previous image with estimated semi-dense depth map Z_{k-1} using direct image alignment
2. Concatenate estimated camera motion with previous frame camera pose to obtain current camera pose estimate $\mathbf{T}_k = \mathbf{T}_{k-1}\mathbf{T}_k^{k-1}$
3. Propagate semi-dense depth map Z_{k-1} from previous frame to current frame to obtain $\widetilde{Z}_k$
4. Update propagated semi-dense depth map $\widetilde{Z}_k$ with temporal stereo depth measurements to obtain Z_k

Direct Visual Odometry Example (Monocular)



Engel et al., Semi-Dense Visual Odometry for a Monocular Camera, ICCV 2013

Direct Image Alignment Revisited



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

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

- What do we mean with „ideally“ ?

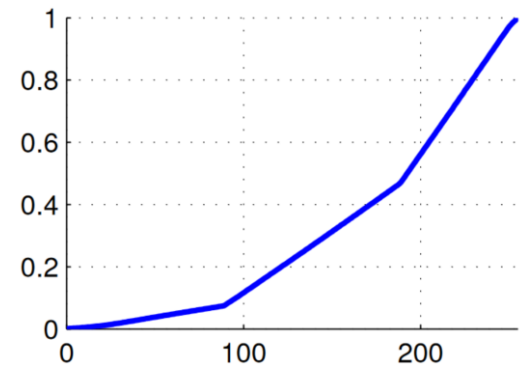
Recap: Camera Response Function

- The objects in the scene radiate light which is focused by the lens onto the image sensor

- The pixels of the sensor observe an irradiance $B : \Omega \to \mathbb{R}$ for an exposure time t

- The camera electronics translates the accumulated irradiance into intensity values according to a non-linear camera response function $G : \mathbb{R} \to [0, 255]$

example inv. G



- The measured intensity is $I(\mathbf{x}) = G(tB(\mathbf{x}))$

Recap: Vignetting

uncorrected

corrected

- Lenses gradually focus more light at the center of the image than at the image borders
- The image appears darker towards the borders
- Also called “lens attenuation”
- Lense vignetting can be modelled as a map $V : \Omega \rightarrow [0, 1]$

- Intensity measurement model

$$I(\mathbf{x}) = G(tV(\mathbf{x})B(\mathbf{x}))$$

$V(\mathbf{x})$

Brightness Constancy Assumption Revisited

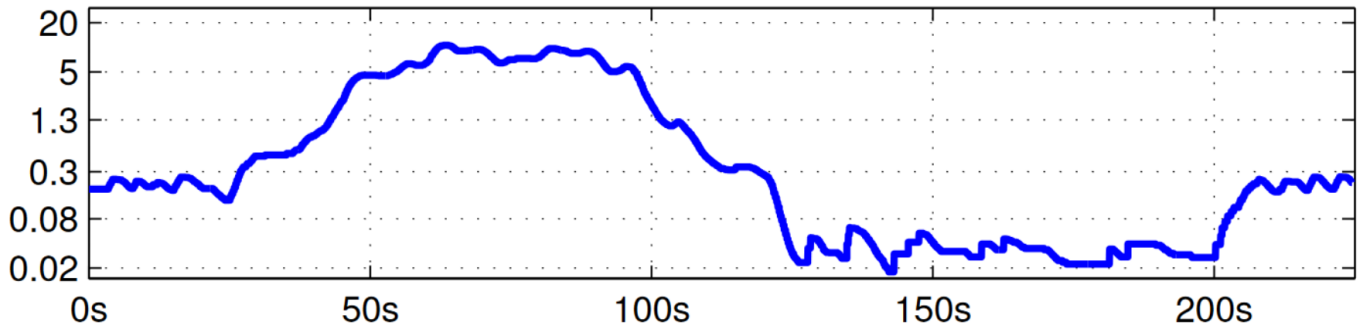
- Camera images include vignetting effects and non-linear camera response function

- Idea: invert vignetting and camera response function using a known calibration

- Perform direct image alignment on irradiance images:

$$I'(\mathbf{y}) = tB(\mathbf{y}) = \frac{G^{-1}(I(\mathbf{y}))}{V(\mathbf{y})}$$

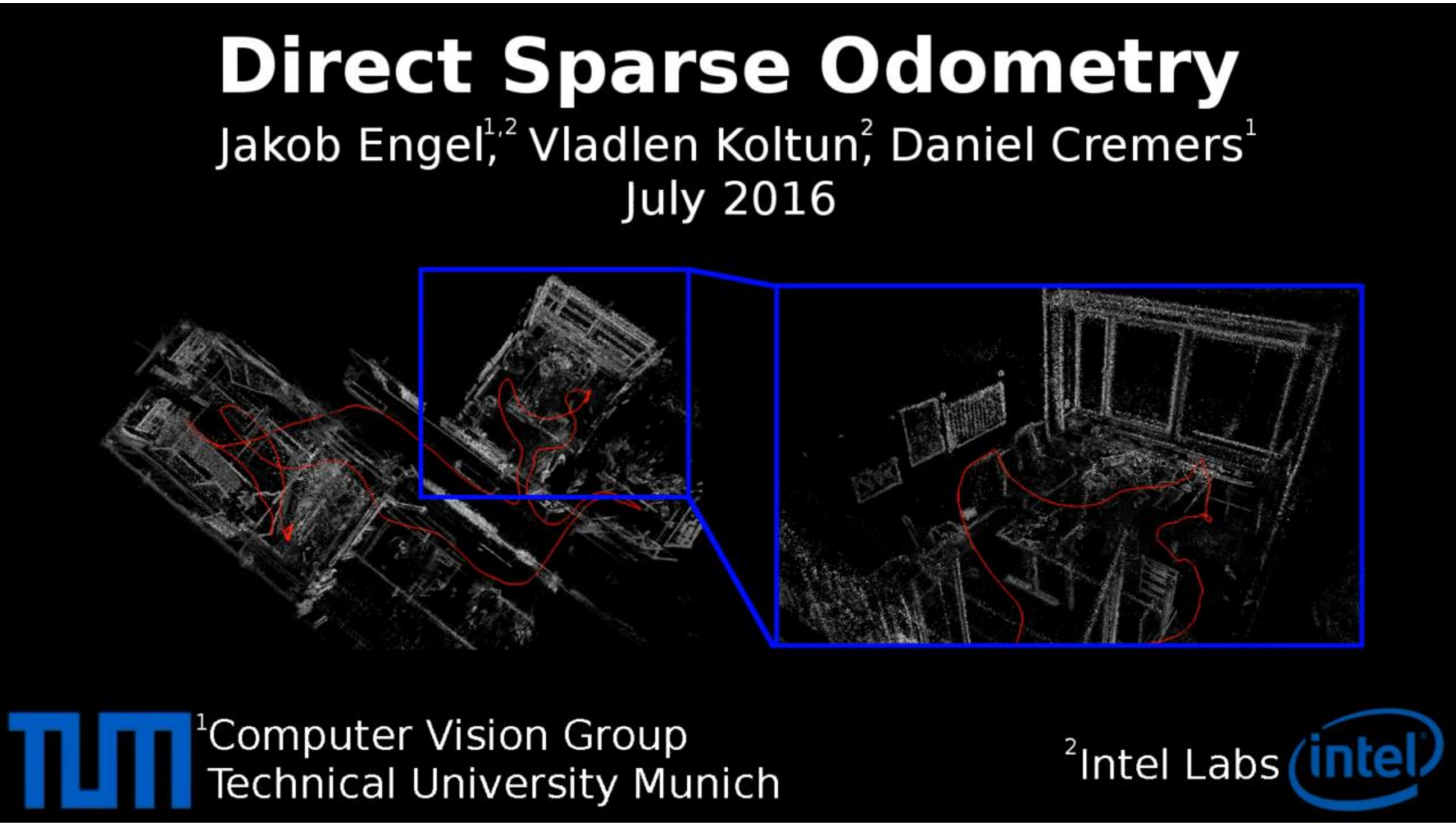
Brightness Constancy Assumption Revisited



- Automatic exposure adjustment needed in realistic environments
- Add exposure parameters explicitly to objective function:

$$(I_2(\omega(\mathbf{y}, \boldsymbol{\xi}, Z_1(\mathbf{y}))) - b_2) - \frac{t_2 \exp(a_2)}{t_1 \exp(a_1)}(I_1(\mathbf{y}) - b_1)$$

Direct Sparse Visual Odometry (Monocular)



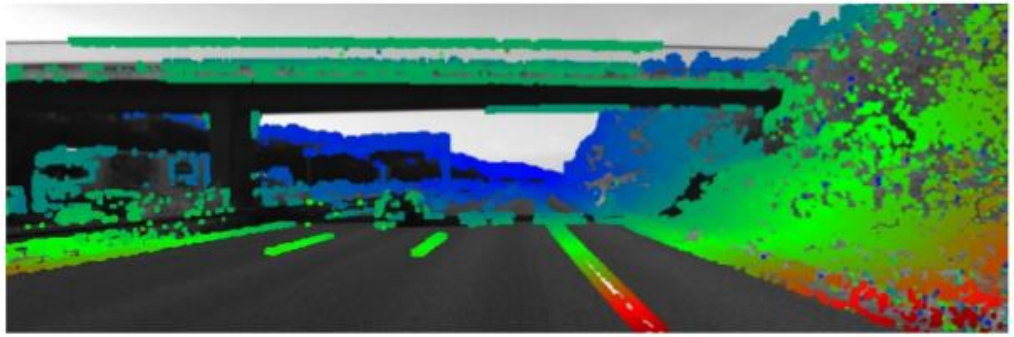
Engel et al., Direct Sparse Odometry, TPAMI 2017

Direct Mapping with Stereo Cameras

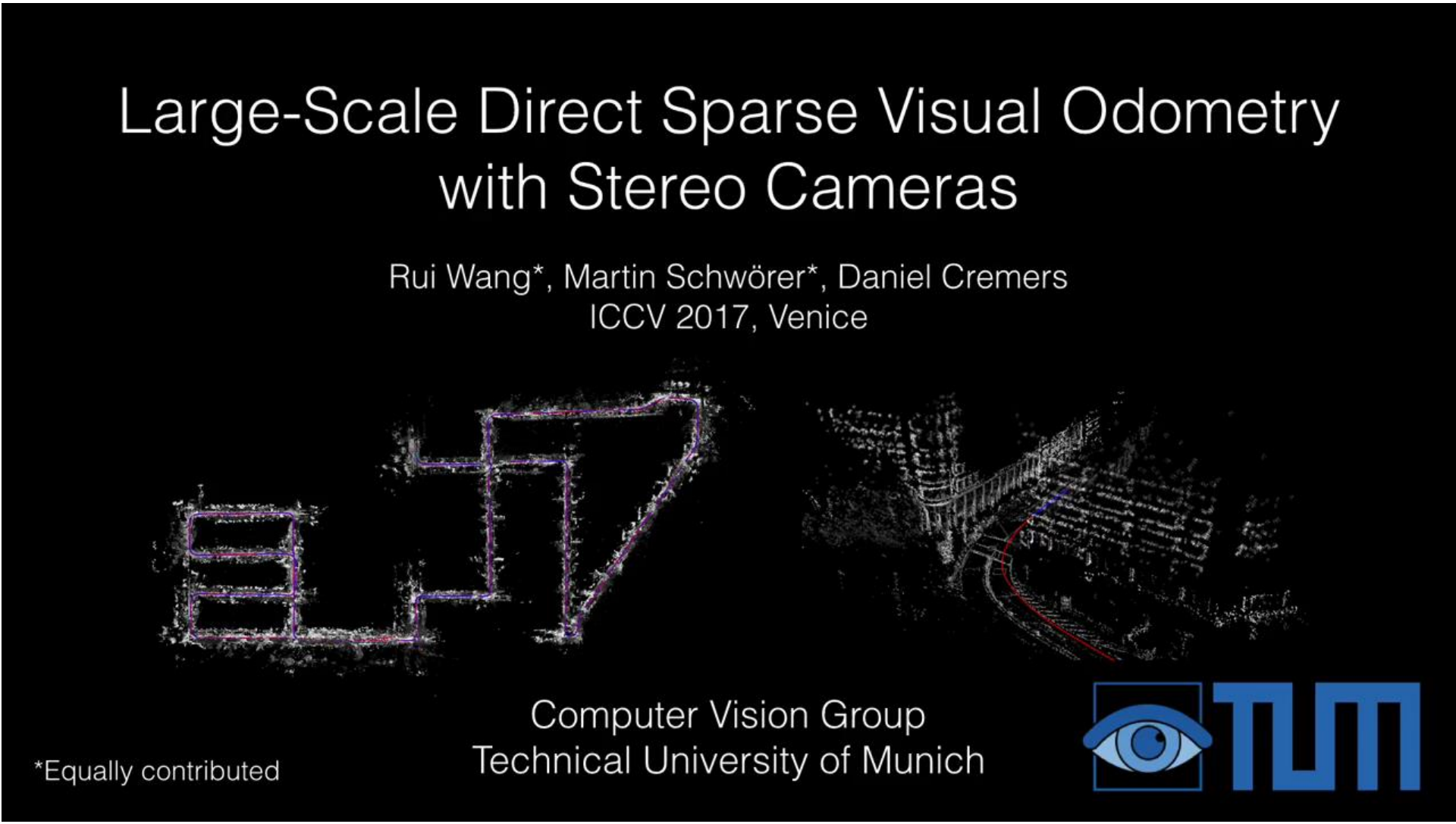
- For stereo cameras, we can exploit the known camera extrinsics to estimate depth from static stereo (left-right images) in addition to temporal stereo (successive left or right images)

no information from static stereo

no information from temporal stereo



Direct Sparse Visual Odometry (Stereo)



Wang et al., Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras, ICCV 2017

Lessons Learned Today

- Direct image alignment avoids manually designed keypoints and can use all available image information
- Direct visual odometry
  - Dense RGB-D odometry by direct image alignment with measured depth
  - Direct image alignment for monocular cameras requires depth estimation from temporal stereo
  - Stereo cameras: Direct depth estimation using static and temporal stereo
- Direct image alignment as non-linear least squares problem
  - Linearization of the residuals requires a coarse-to-fine optimization scheme
  - SE(3) Lie algebra provides an elegant way of motion representation for gradient-based optimization
  - Iteratively reweighted least squares allows for wider set of residual distributions than Gaussians
- Photometric calibration and exposure parameter estimation

Thanks for your attention!