

Computer Vision II: Multiple View Geometry

Exercise 8: Direct Image Alignment

Christiane Sommer, Rui Wang

July 06, 2017



Direct Image Alignment

- = “Direct Tracking” / “Dense Tracking” / “Dense Visual Odometry”
- = “Lucas-Kanade Tracking on SE(3)”

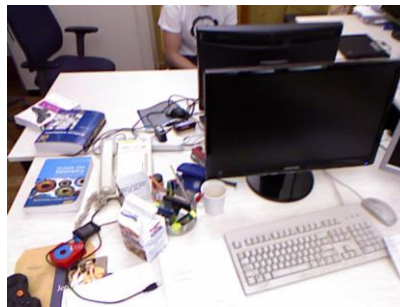
reference image



reference depth



+

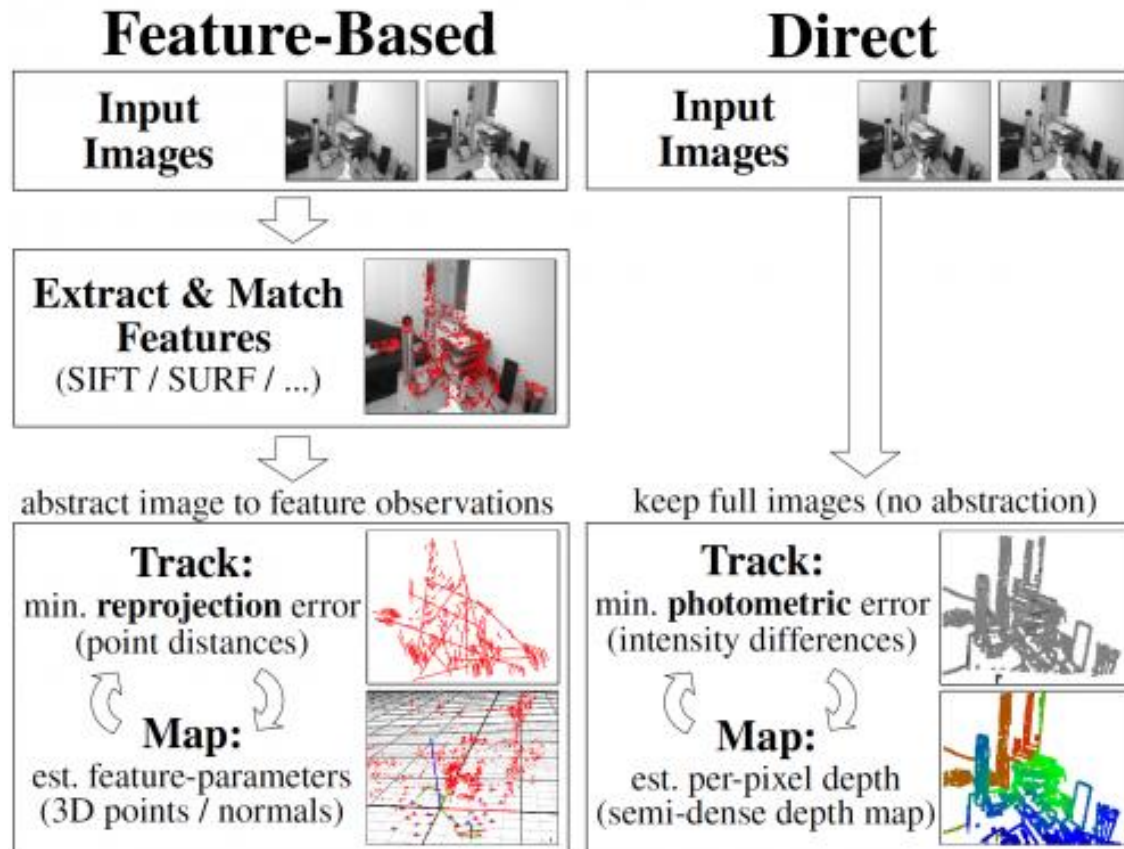


new image



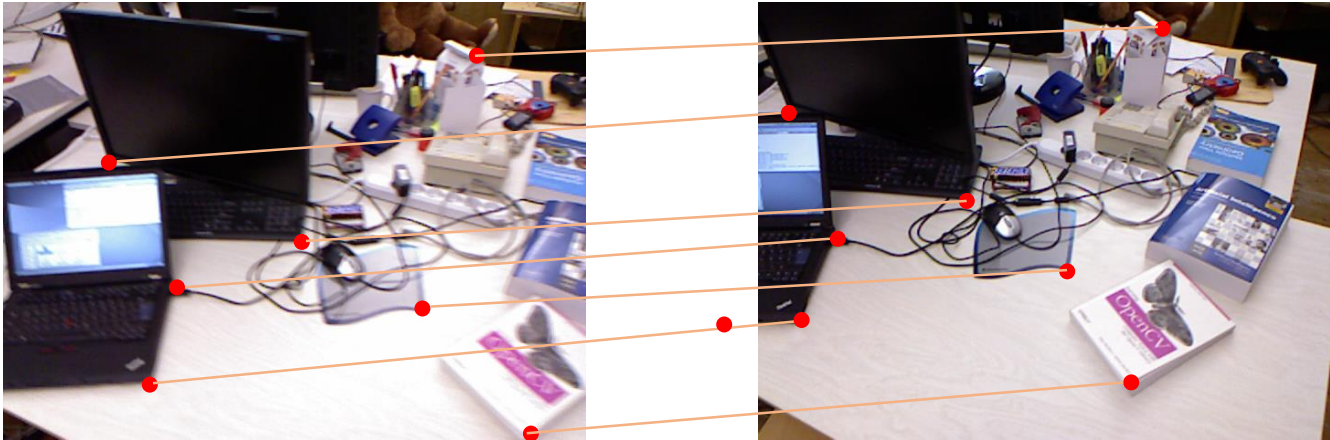
**Camera
pose ξ**

Keypoints, Direct, Sparse, Dense

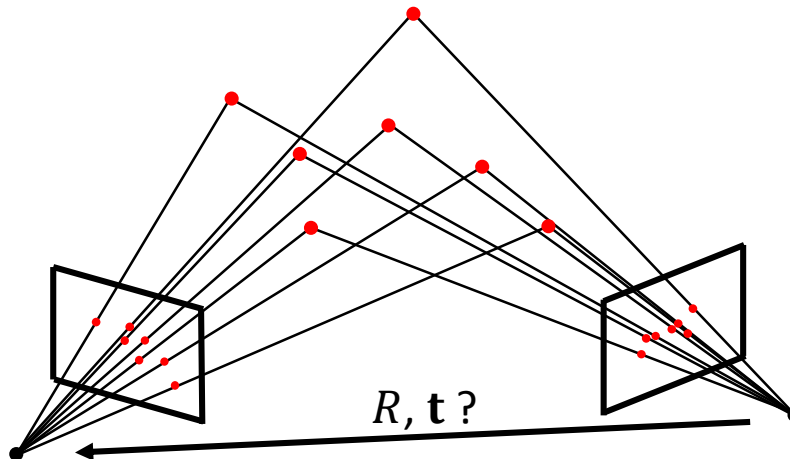


- Sparse: use a small set of selected pixels (keypoints)
- Dense: use all (valid) pixels

Sparse Keypoint-based Visual Odometry



Extract and match
keypoints



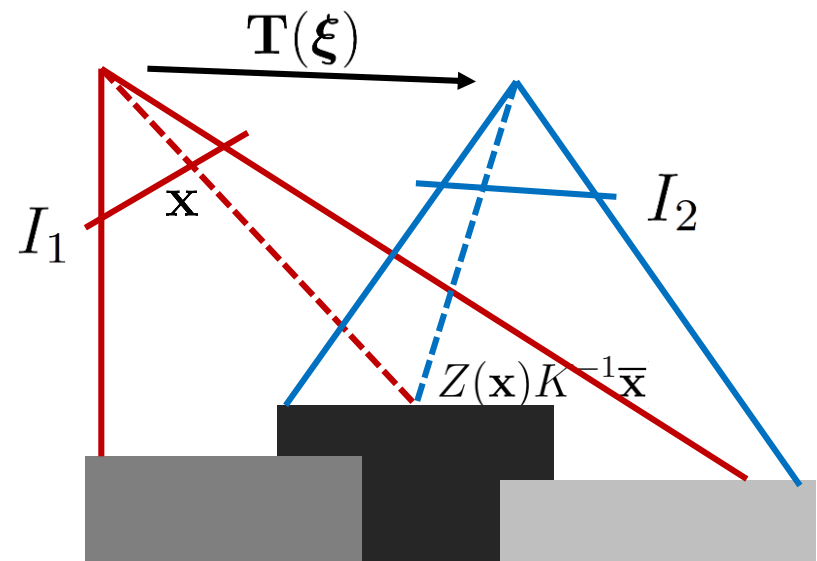
Determine relative
camera pose (R, \mathbf{t})
from keypoint matches

Dense Direct Image Alignment

- Known pixel depth \rightarrow "simulate" RGB-D image from different view point
- Ideally: warped image = image taken from that pose:

$$I_2(\tau(\xi, \mathbf{x}_i)) = I_1(\mathbf{x}_i)$$

- RGB-D: depth available \rightarrow find camera motion!
- Motion representation using the SE(3) Lie algebra
- Non-linear least squares optimization

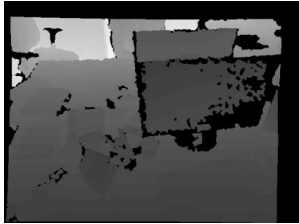


Minimization of photometric error: Normally distributed residuals

$$E(\xi) = \sum_i r_i(\xi)^2 = \sum_i \left(\underbrace{I_2(\tau(\xi, \mathbf{x}_i))}_{\text{new image}} - \underbrace{I_1(\mathbf{x}_i)}_{\text{reference image}} \right)^2$$

$E(\xi)$ camera pose
 \sum_i sum over valid pixels
 $I_2(\tau(\xi, \mathbf{x}_i))$ new image
 $I_1(\mathbf{x}_i)$ reference image

reference depth



$\tau(\xi, \mathbf{x}_i)$ warps a pixel from reference image to new image

Gauss-Newton optimization

$$E(\xi) = \sum_i r_i(\xi)^2 = \sum_i \left(I_2(\tau(\xi, \mathbf{x}_i)) - I_1(\mathbf{x}_i) \right)^2$$

- Solved with **Gauss-Newton** algorithm using left-multiplicative increments on SE(3):
 $\xi_1 \circ \xi_2 := \log(\exp(\xi_1) \cdot \exp(\xi_2)) \neq \xi_2 \circ \xi_1 \neq \xi_1 + \xi_2$
- **Intuition:** iteratively solve for $\nabla E(\xi) = 0$ by approximating $\nabla E(\xi)$ linearly (i.e. by approximating $E(\xi)$ quadratically)
- Using **coarse-to-fine** pyramid approach

Gauss-Newton: gradients

- Vector of residuals: \mathbf{r} with $r_i = r_i(0)$ i.e. evaluated at 0
- Matrix of gradients: J with rows $J^{(i)} = \frac{\partial r_i(\xi)}{\partial \xi} \Big|_{\xi=0}$
- Gradients: $\frac{\partial r_i(\xi)}{\partial \xi} \Big|_{\xi=0} =$

$$= \frac{1}{z} \begin{pmatrix} I_x f_x & I_y f_y \end{pmatrix} \begin{pmatrix} 1 & 0 & -\frac{x}{z} & -\frac{xy}{z} & z + \frac{x^2}{z} & -y \\ 0 & 1 & -\frac{y}{z} & -z - \frac{y^2}{z} & \frac{xy}{z} & x \end{pmatrix} \Big|_{(x,y,z)^\top = \pi^{-1}(\mathbf{x}_i, Z_1(\mathbf{x}_i))}$$
- Update step: $\delta_\xi = (J^\top J)^{-1} J^\top \mathbf{r}$ and $\xi^{(k+1)} = \delta_\xi \circ \xi^{(k)}$

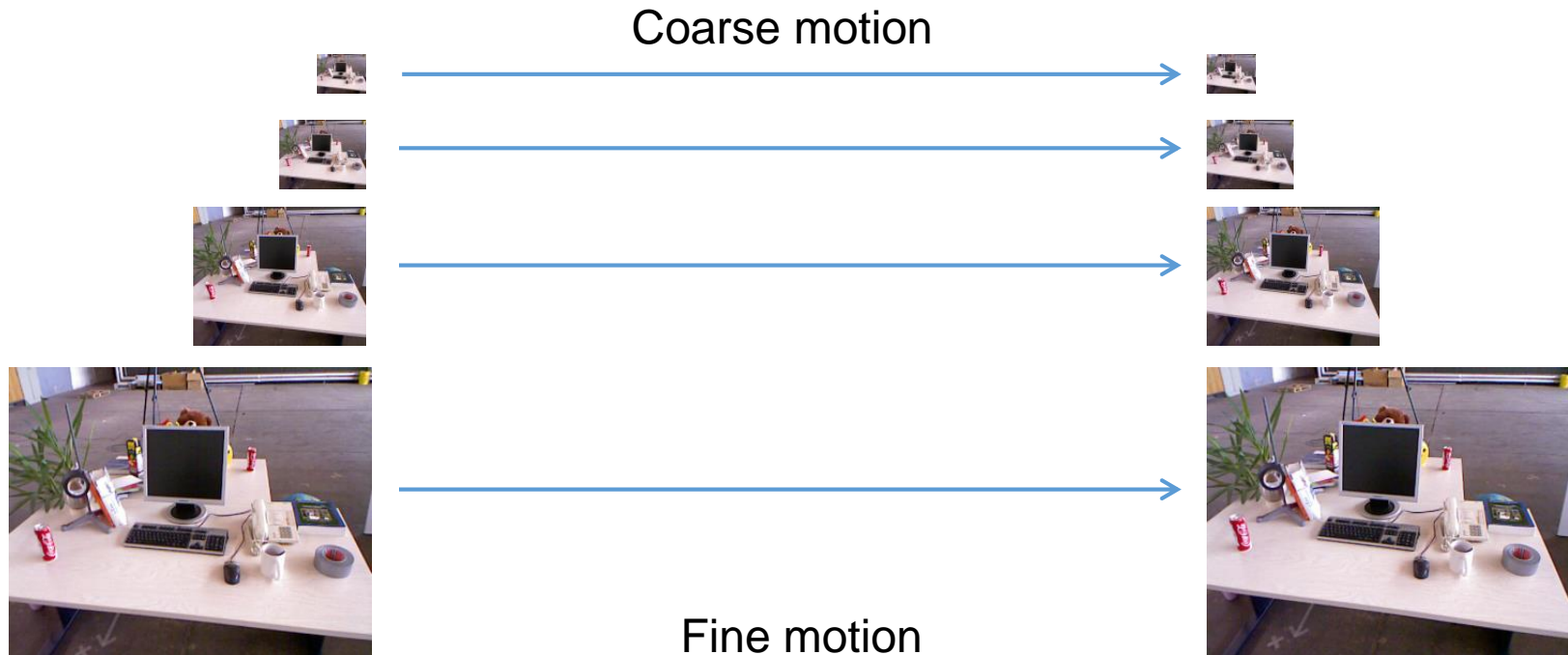
Gauss-Newton optimization

$$E(\xi) = \sum_i r_i(\xi)^2 = \sum_i \left(I_2(\tau(\xi, \mathbf{x}_i)) - I_1(\mathbf{x}_i) \right)^2$$

- Solved with **Gauss-Newton** algorithm using left-multiplicative increments on SE(3):
 $\xi_1 \circ \xi_2 := \log(\exp(\xi_1) \cdot \exp(\xi_2)) \neq \xi_2 \circ \xi_1 \neq \xi_1 + \xi_2$
- **Intuition:** iteratively solve for $\nabla E(\xi) = 0$ by approximating $\nabla E(\xi)$ linearly (i.e. by approximating $E(\xi)$ quadratically)
- Using **coarse-to-fine** pyramid approach

Coarse-to-Fine

- Adapt size of the neighborhood from coarse to fine



Coarse-to-Fine

- Minimize for down-scaled image (e.g. factor 8, 4, 2, 1) and use result as initialization for next finer level
- Elegant formulation: Downscale image and adjust K accordingly
 - Downscale by factor of 2 (e.g. 640x480 -> 320x240)
 - Adjust camera matrix elements f_x , f_y , c_x and c_y :

$$K^{(l+1)} = \begin{pmatrix} \frac{1}{2}f_x^{(l)} & 0 & \frac{1}{2}c_x^{(l)} & -\frac{1}{4} \\ 0 & \frac{1}{2}f_y^{(l)} & \frac{1}{2}c_y^{(l)} & -\frac{1}{4} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Final Algorithm

$\xi^{(0)} = 0$

$k = 0$

for $level = 3 \dots 1$

 compute down-scaled images & depthmaps (factor = 2^{level})

 compute down-scaled K (factor = 2^{level})

for $i = 1..20$

 compute Jacobian $J_r \in R^{n \times 6}$

 compute update δ_ξ

 apply update $\xi^{(k+1)} = \delta_\xi \circ \xi^{(k)}$

$k++$; maybe break early if δ_ξ too small or if residual increased

done

done

+ robust weights (e.g. Huber), see *iteratively reweighted least squares*

+ *Levenberg-Marquadt (LM) Algorithm*