Computer Vision II: Multiple View Geometry

Exercise 8: Direct Image Alignment

Nikolaus Demmel, Marvin Eisenberger

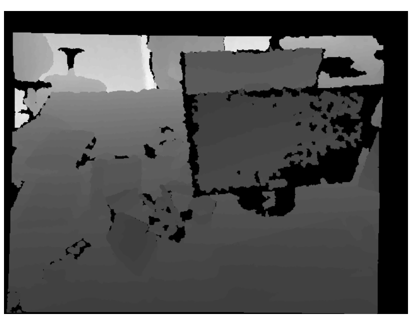
June 20, 2018

Direct Image Alignment

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

reference image

reference depth



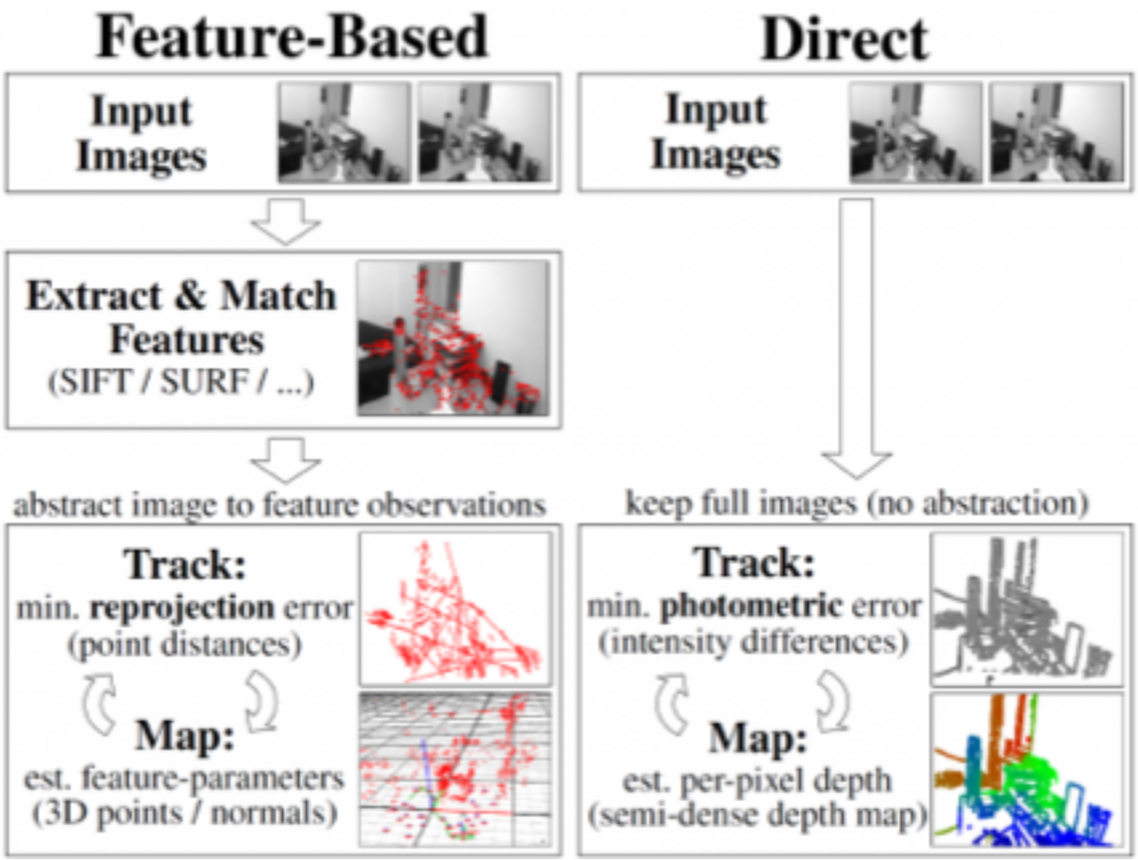
+

new image

→ **Camera pose ξ**

Slides based on slides by R. Maier 2016

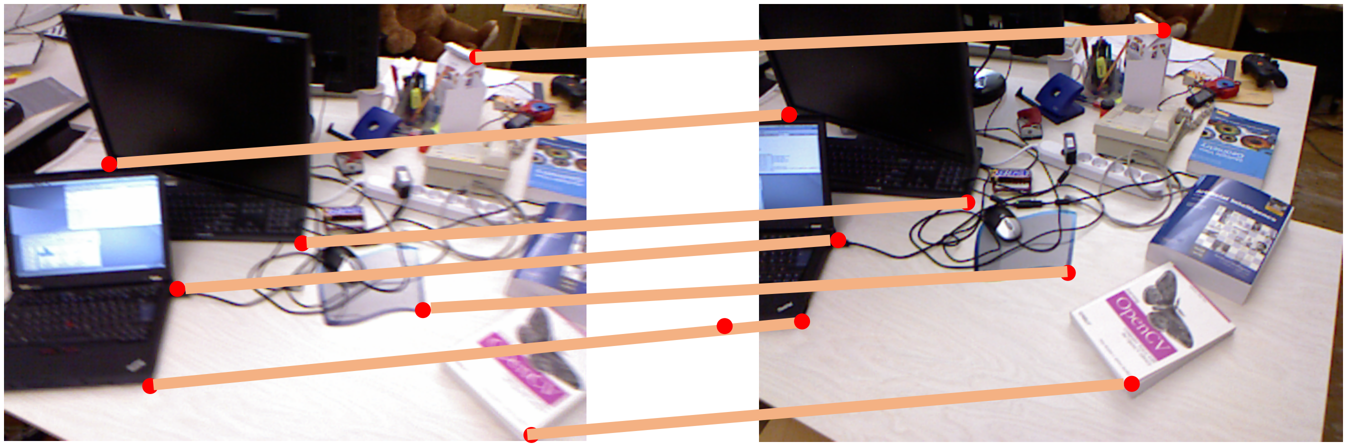
Keypoints, Direct, Sparse, Dense



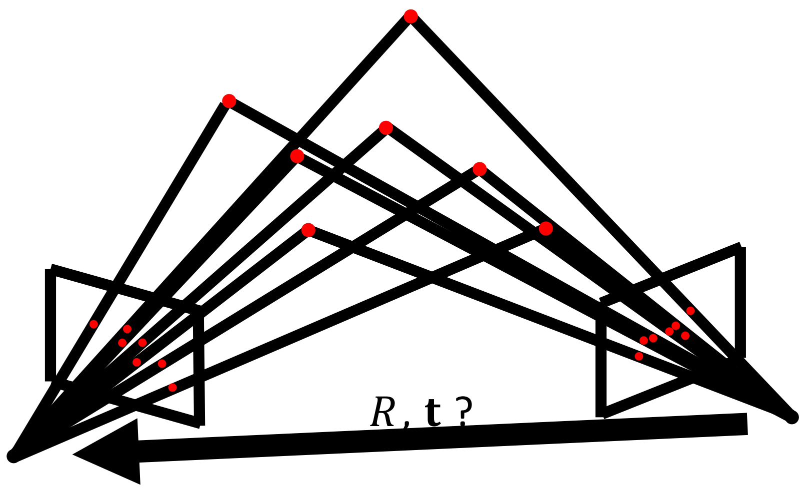
- Sparse: use a small set of selected pixels (keypoints)
- Dense: use all (valid) pixels

Slides based on slides by R. Maier 2016

Sparse Keypoint-based Visual Odometry



Extract and match keypoints



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

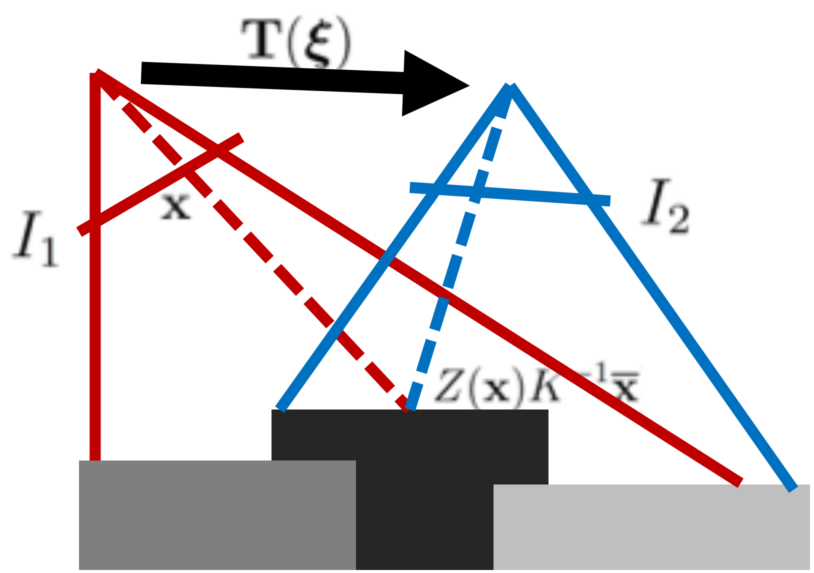
Slides based on slides by R. Maier 2016

Dense Direct Image Alignment

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

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

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



Slides based on slides by R. Maier 2016

Minimization of photometric error: Normally distributed residuals

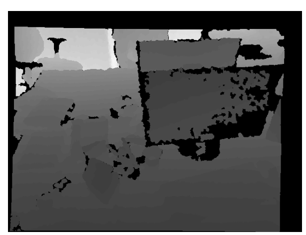
$$E(\xi) = \underbrace{\sum_{i}}_{\text{sum over valid pixels}} r_i(\xi)^2 = \sum_{i} \Big(\underbrace{I_2\big(\tau(\xi, \mathbf{x}_i)\big)}_{\text{new image}} - \underbrace{I_1(\mathbf{x}_i)}_{\text{reference image}} \Big)^2$$

camera pose

new image

reference image

reference depth Z_1



$$\tau(\xi, \mathbf{x}_i) = \pi\left(T\left(g(\xi), \pi^{-1}(\mathbf{x}_i, \underbrace{Z_1(\mathbf{x}_i)})\right)\right)$$

$$\pi\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{f_x x}{z} + c_x & \frac{f_y y}{z} + c_y \end{pmatrix}^{\top}$$

$$\pi^{-1}\left(\begin{pmatrix} x \\ y \end{pmatrix}, z\right) = \begin{pmatrix} \frac{z(x - c_x)}{f_x} & \frac{z(y - c_y)}{f_y} & z \end{pmatrix}^{\top}$$

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

Slides based on slides by R. Maier 2016

Gauss-Newton optimization

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

- Solved with **Gauss-Newton** algorithm using left-multiplicative increments on SE(3):

  $\xi_1 \circ \xi_2 := \log\big(\exp(\hat{\xi}_1) \cdot \exp(\hat{\xi}_2)\big)^{\vee} \neq \xi_2 \circ \xi_1 \neq \xi_1 + \xi_2$

- **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 optimization

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

1. In every iteration $\mathrm{k}+1$ linearize $\mathbf{r}$ on manifold around current pose $\xi^{(k)}$:

$$\mathbf{r}(\xi) \approx \underbrace{\mathbf{r}\big(\xi^{(k)}\big)}_{\mathbf{r}_0 \in \mathbb{R}^n} + \underbrace{\frac{\partial \mathbf{r}\big(\epsilon \circ \xi^{(k)}\big)}{\partial \epsilon}\Big|_{\epsilon=0}}_{J_\mathbf{r} \in \mathbb{R}^{n\times 6}} \cdot \underbrace{\big(\xi \circ (\xi^{(k)})^{-1}\big)}_{\delta_\xi}$$

2. Solve for $\nabla E(\xi) = 0$

$$E(\xi) = \left\|\mathbf{r}_0 + J_\mathbf{r} \cdot \delta_\xi\right\|_2^2 = \mathbf{r}_0^\top \mathbf{r}_0 + 2\delta_\xi^\top J_\mathbf{r}^\top \mathbf{r}_0 + \delta_\xi^\top J_\mathbf{r}^\top J_\mathbf{r} \delta_\xi$$
$$\nabla E(\xi) = 2J_\mathbf{r}^\top \mathbf{r}_0 + 2J_\mathbf{r}^\top J_\mathbf{r} \delta_\xi = 0 \quad \Rightarrow \quad \delta_\xi = -(J_\mathbf{r}^\top J_\mathbf{r})^{-1} J_\mathbf{r}^\top \mathbf{r}_0$$

3. Apply $\xi^{(k+1)} = \delta_\xi \circ \xi^{(k)}$

4. Iterate (until convergence)

Slides based on slides by R. Maier 2016

Gauss-Newton optimization

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

Jacobian J_r: partial derivatives

Gradient of residual (1x6 row of J_r):

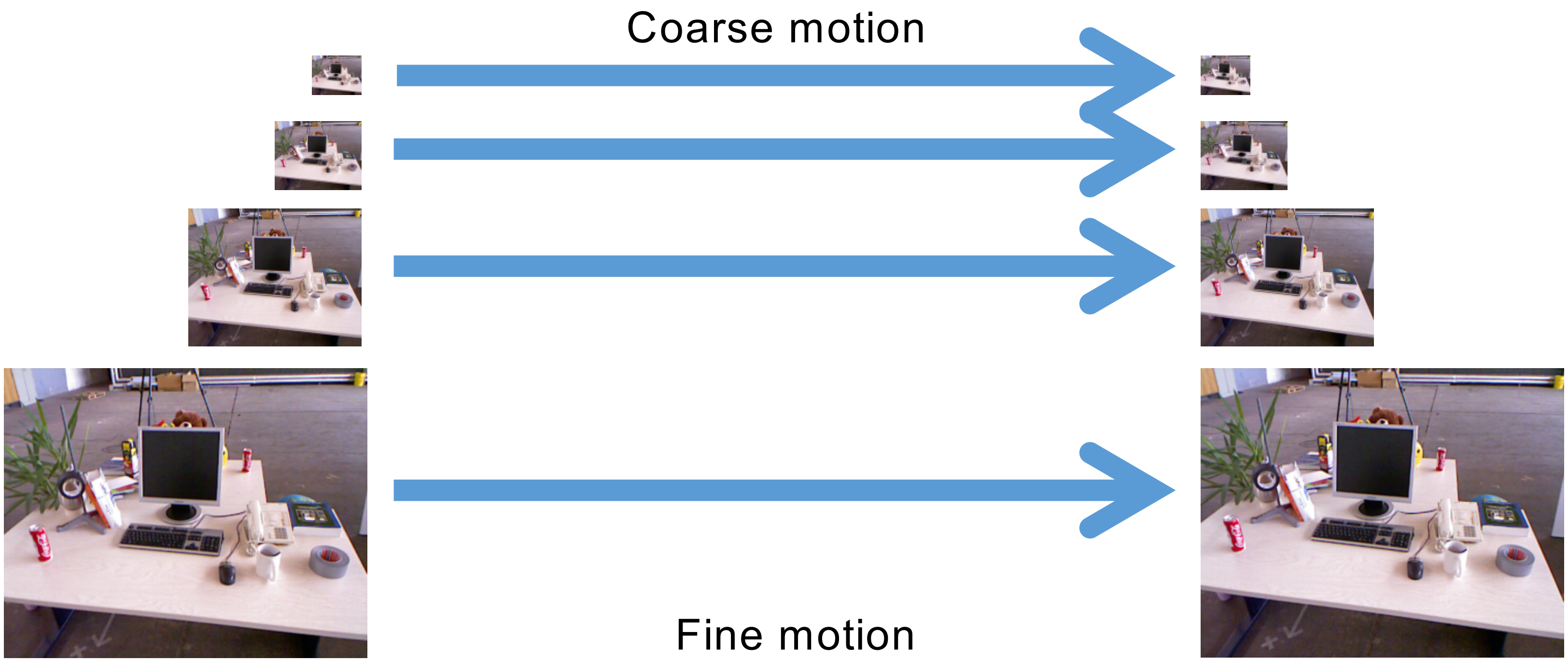
$$\frac{\partial r_i\big(\epsilon \circ \xi^{(k)}\big)}{\partial \epsilon}\bigg|_{\epsilon=0} = \frac{1}{z'}(\nabla I_x f_x \quad \nabla I_y f_y)\begin{pmatrix} 1 & 0 & -\frac{x'}{z'} & -\frac{x'y'}{z'} & z' + \frac{x'^2}{z'} & -y' \\ 0 & 1 & -\frac{y'}{z'} & -z' - \frac{y'^2}{z'} & \frac{x'y'}{z'} & x' \end{pmatrix}$$

with

- transformed 3d point $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \coloneqq T\left(g\big(\xi^{(k)}\big), \pi^{-1}\big(\mathbf{x}_i, Z_i(\mathbf{x}_i)\big)\right)$
- the image gradient $(\nabla I_x \quad \nabla I_y)^\top$ of I_2 evaluated at warped point $\mathbf{x}_i' \coloneqq \tau\big(\xi^{(k)}, \mathbf{x}_i\big)$

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:

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

  - Assumes continuous coordinate of a discrete pixel is at its center, i.e. the top-left pixel-center has continuous coordinates (0,0)

Final Algorithm

$\xi^{(0)} = \mathbf{0}$
k = 0
for *level* = 3 ... 1

  compute down-scaled images & depthmaps (factor = 2^{level})
  compute down-scaled K (factor = 2^{level})
  for *i* = 1..20

    compute Jacobian $J_{\mathbf{r}} \in R^{n \times 6}$
    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-Marquad (LM)* Algorithm

Slides based on slides by R. Maier 2016