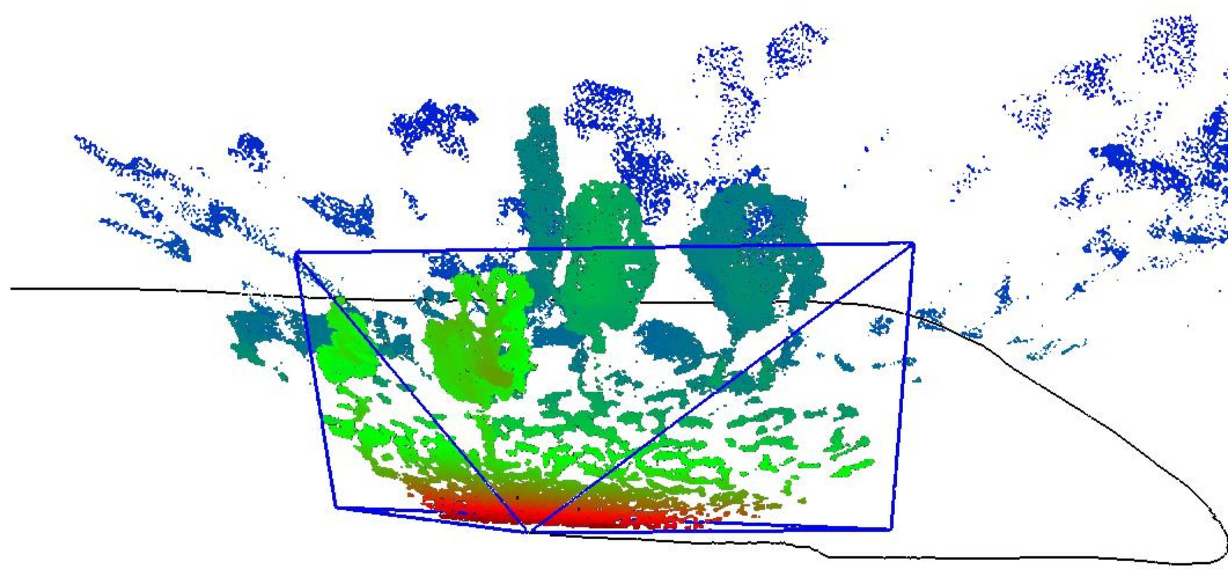
Semi-Dense Visual Odometry for a Monocular Camera

Jakob Engel, Jürgen Sturm, Daniel Cremers

Intl. Conf. on Computer Vision (ICCV) 2013



Monocular Video **Camera Motion** and **Scene Geometry**

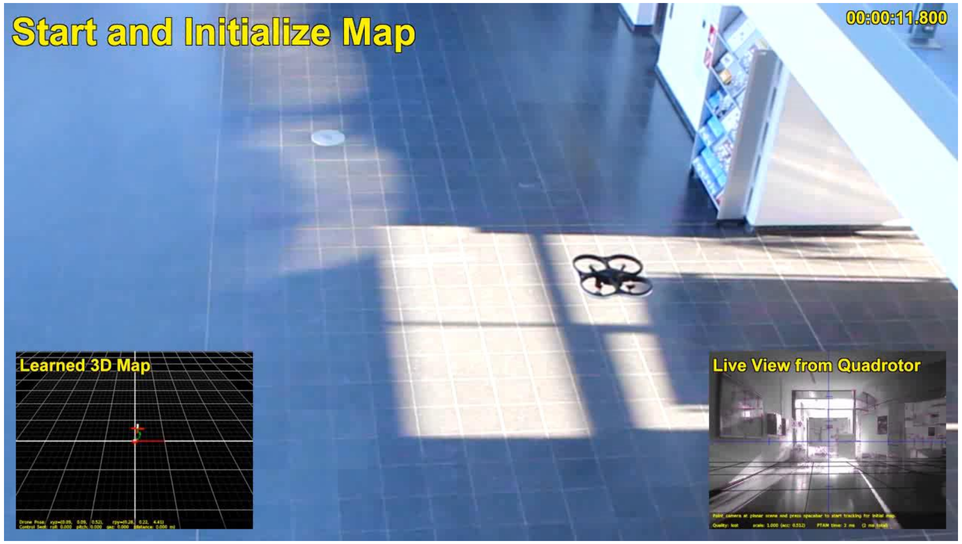
Visual Odometry

Camera: 752x480 @ 30fps, global shutter, monochrome, 130° diagonal fov

Augmented / Virtual Reality

- requires **camera pose** to render objects
- requires **scene geometry** e.g. for physical interaction

Robotics



- requires **camera pose** to control robot position
- requires **scene geometry** e.g. to avoid obstacles

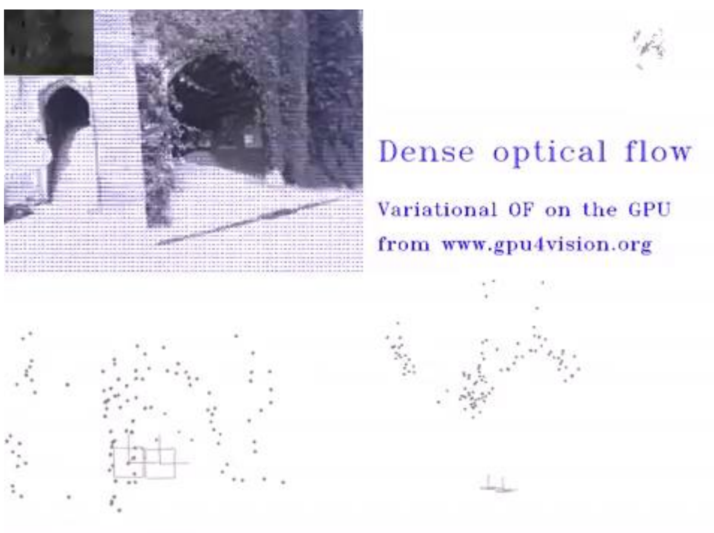
Why Monocular?

- small, light-weight
- cheap
- low power comsumption
- versatile (scale-ambivalent)
- easy(er) to calibrate

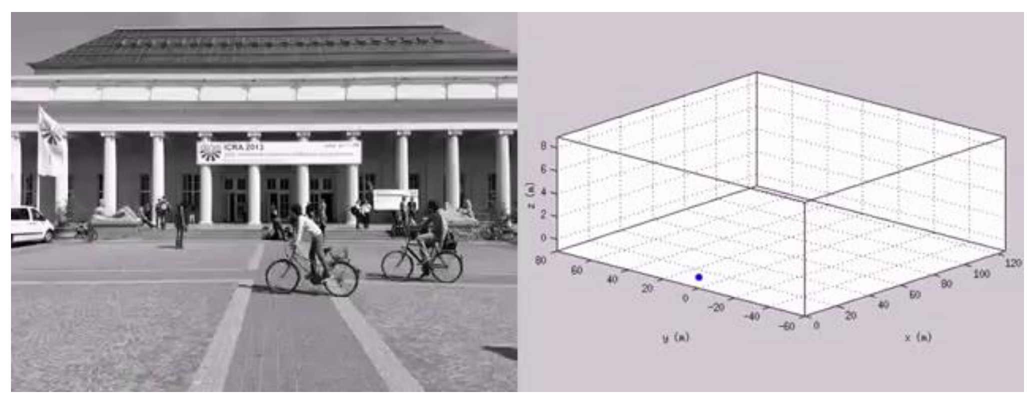
Long History...

A. Davison et.al., 2004

G. Klein et.al., 2007



H. Strasdat et.al., 2010

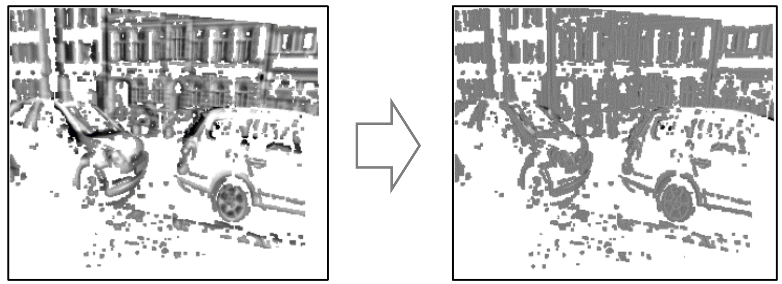


A. Mourikis et.al., 2013

Overview

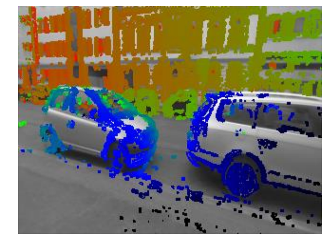
Input Video
640x480 @ 30Hz

Tracking
Estimate pose of new frame relative to depth map

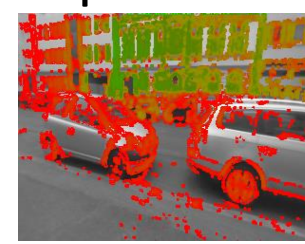


Semi-Dense Depth Map

- pixel depth

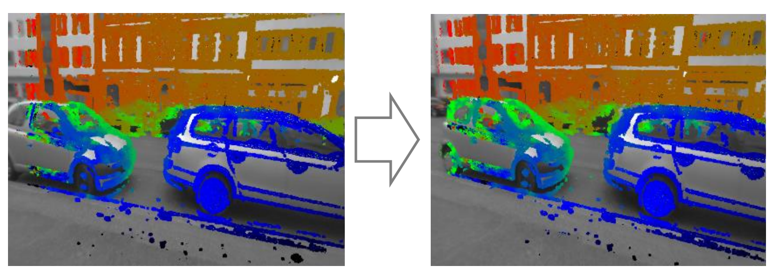


- depth variance



- image

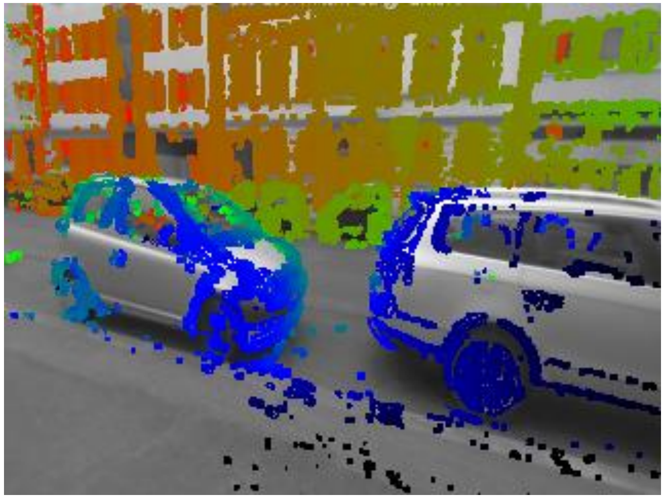
Mapping
Propagate and update depth map (stereo)



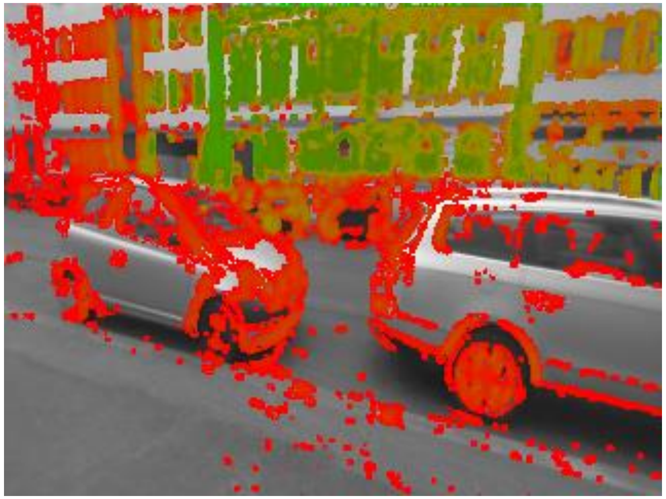
Separate Initialization

- 8 point algorithm
- random

World Representation



inv. depth mean



inv. depth variance

intensity image

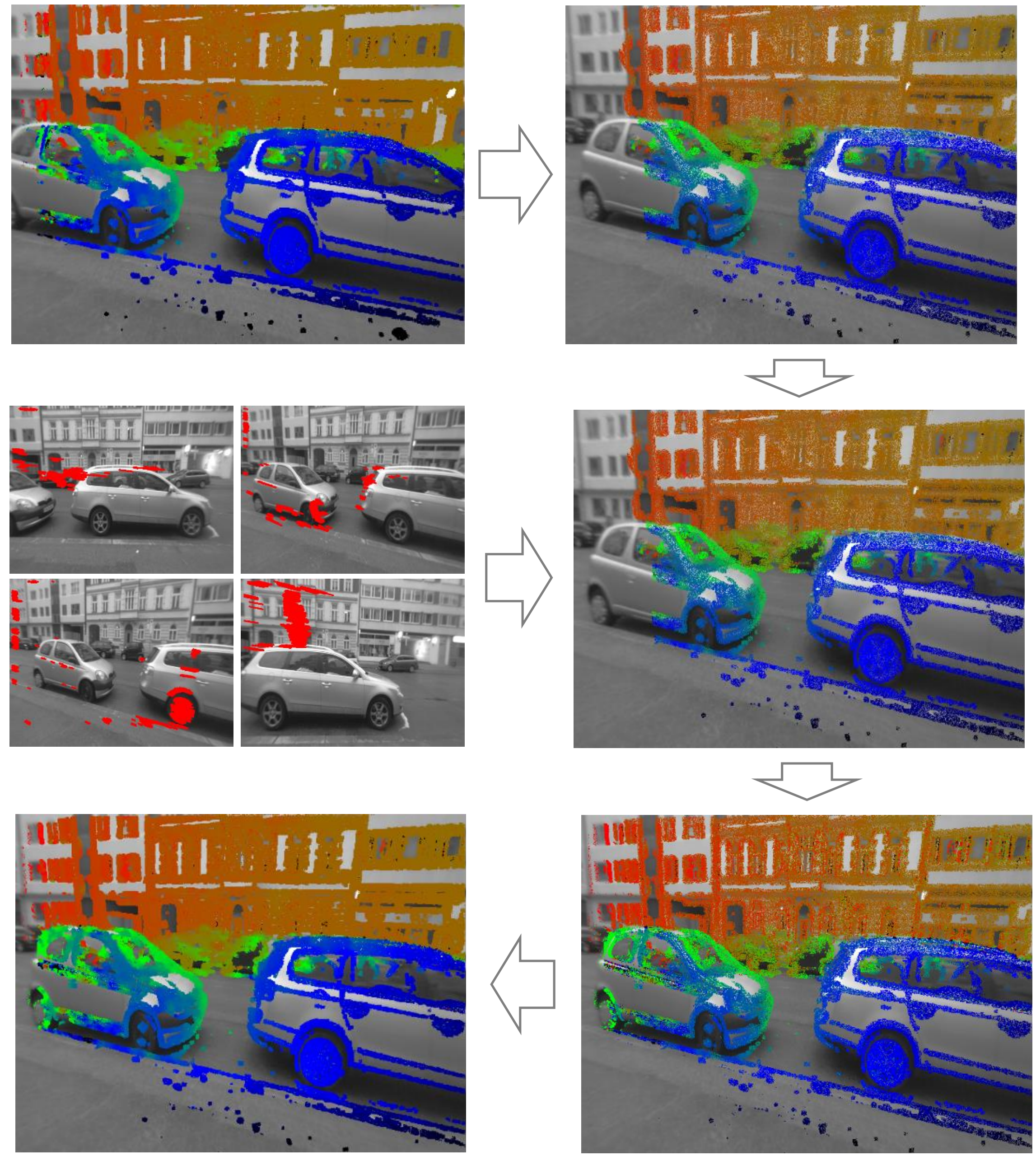
- Gaussian on inverse depth
- for all pixel with sufficient abs. image gradient
- pose-graph + keyframes

Mapping

1. Propagation

2. Stereo-update

3. Regularization



Direct Image Alignment

= „Direct Tracking“ / „Dense Tracking“

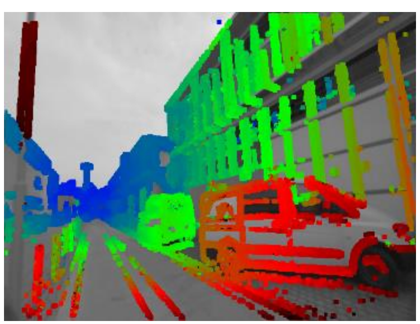
= „Lucas-Kanade Tracking on SE(3)“

→ Maximum-Likelihood Estimator

→ often used for RGB-D tracking (Kinect)

(Kerl et.al. @ ICRA ´13; Steinbruecker et.al. @ ICCV ´11; and many more)

image



per-pixel depth

+

new image

→

Camera pose ξ

Direct minimization of photometric error

$$E(\xi) = \sum_{\mathbf{p}_i \in \Omega_{\text{ref}}} \left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \xi)) \right)^2$$

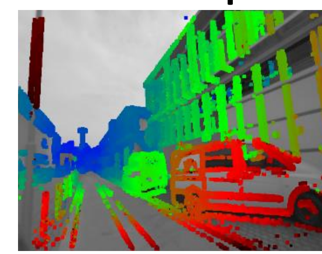
camera pose

sum over valid ref. pixel

ref. image

new image

ref. depth



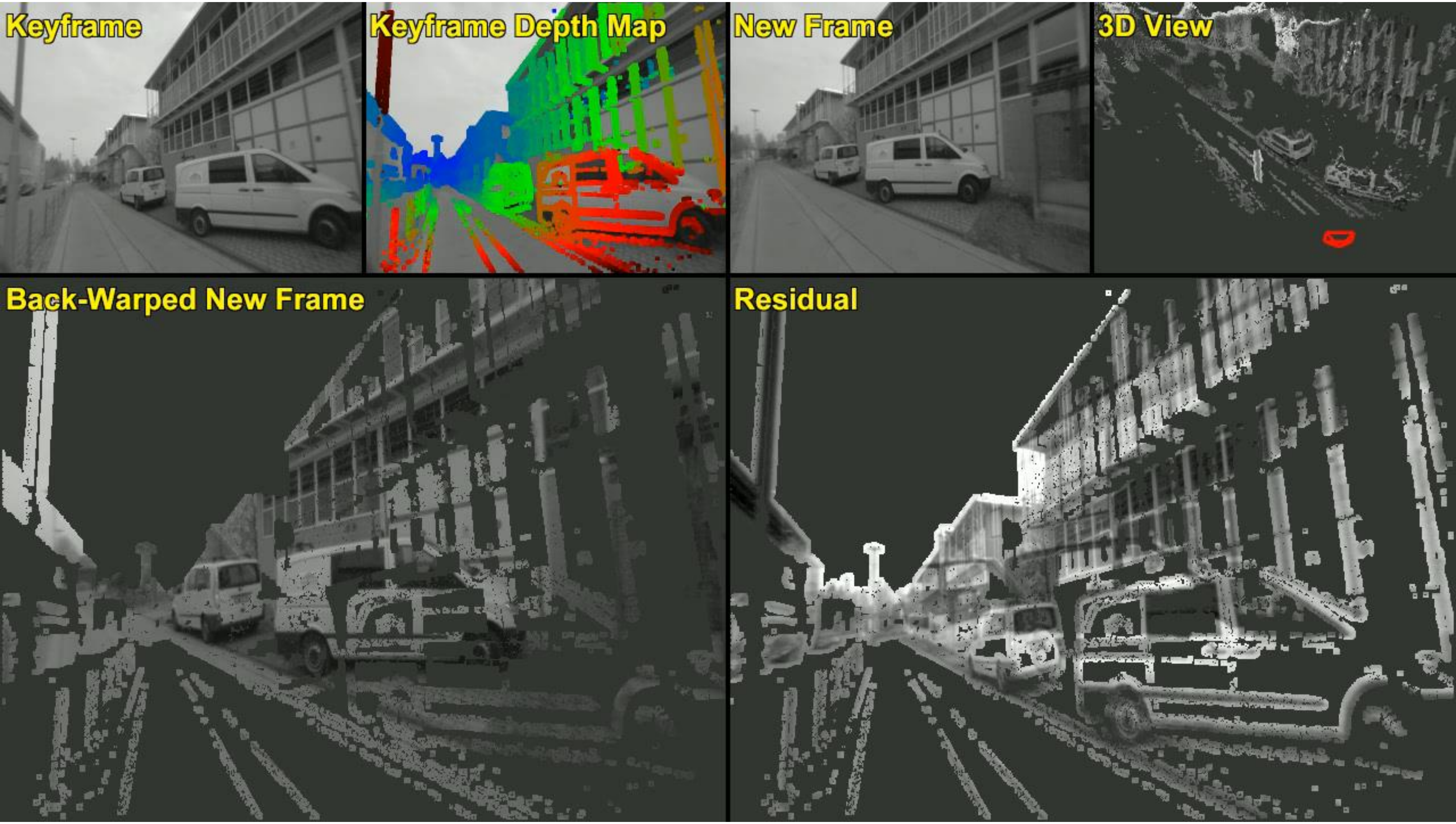
$$\omega(\mathbf{p}_i, d, \xi) := \pi\left(K\left(R_\xi K^{-1} \begin{pmatrix} d p_{i,x} \\ d p_{i,y} \\ d \end{pmatrix} + \mathbf{t}_\xi\right)\right)$$

$$\pi(x, y, z) := \begin{pmatrix} x/z \\ y/z \end{pmatrix}$$

$$\begin{pmatrix} R_\xi & \mathbf{t}_\xi \\ \mathbf{0} & 1 \end{pmatrix} := \exp(\widehat{\xi})$$

$\omega(\mathbf{p}_i, d, \xi)$ „warps“ a pixel from ref. image to new image

Direct Image Alignment



$$E(\xi) = \sum_{\mathbf{p}_i \in \Omega_{\text{ref}}} \left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \xi))\right)^2$$

Direct Image Alignment

$$E(\xi) = \sum_{\mathbf{p}_i \in \Omega_{\text{ref}}} \left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \xi)) \right)^2$$

- solved using the **Gauss-Newton** algorithm using left-multiplicative increments on SE(3):

$$\xi_1 \circ \xi_2 := \log(\exp(\widehat{\xi_1}) \cdot \exp(\widehat{\xi_2}))^\vee \neq \xi_1 + \xi_2$$
$$\neq \xi_2 \circ \xi_1$$

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

Direct Image Alignment

$$E(\xi) = \sum_{\mathbf{p}_i \in \Omega_{\text{ref}}} \underbrace{\left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \xi))\right)^2}_{=:\, r_i^2(\xi)}$$

1. „Linearize" **r** on Manifold around current pose $\xi^{(n)}$:

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

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

$$E(\xi) = ||\mathbf{r}_0 + J_{\mathbf{r}} \cdot \delta_\xi||_2^2 = \mathbf{r}_0^T \mathbf{r}_0 + 2\delta_\xi^T J_{\mathbf{r}}^T \mathbf{r}_0 + \delta_\xi^T J_{\mathbf{r}}^T J_{\mathbf{r}} \delta_\xi$$

$$\nabla E(\xi) = 2 J_{\mathbf{r}}^T \mathbf{r}_0 + 2 J_{\mathbf{r}}^T J_{\mathbf{r}} \delta_\xi = 0 \quad \Rightarrow \quad \delta_\xi = -(J_{\mathbf{r}}^T J_{\mathbf{r}})^{-1} J_{\mathbf{r}}^T \mathbf{r}_0$$

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

Direct Image Alignment

$$E(\xi) = \sum_{\mathbf{p}_i \in \Omega_{\text{ref}}} \underbrace{\left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \xi))\right)^2}_{=:\, r_i^2(\xi)}$$

Requires gradient of residual:

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

= 1x6 row of $J_{\mathbf{r}}$

with

- $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} := R_{\xi^{(k)}} K^{-1} \begin{pmatrix} dp_{i,x} \\ dp_{i,y} \\ d \end{pmatrix} + \mathbf{t}_{\xi^{(k)}}$ = warped point (before projection)
- f_x, f_y, K = intrinsic camera calibration
- $\nabla I_x, \nabla I_y$ = image gradients

$$E(\xi) = \sum_{\mathbf{p}_i \in \Omega_{\text{ref}}} \underbrace{\left(I_{\text{ref}}(\mathbf{p}_i) - I(\omega(\mathbf{p}_i, D_{\text{ref}}(\mathbf{p}_i), \xi))\right)^2}_{=: r_i^2(\xi)}$$

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* correspondingly:
  - Downscale by factor of 2 (e.g. 640x480 -> 320->240)
  - $K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$ -> $K_{\frac{1}{2}} = \begin{pmatrix} \frac{f_x}{2} & 0 & \frac{c_x+0.5}{2} - 0.5 \\ 0 & \frac{f_y}{2} & \frac{c_y+0.5}{2} - 0.5 \\ 0 & 0 & 1 \end{pmatrix}$
  - (assuming discrete pixel (x,y) contains continuous value at (x,y))

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

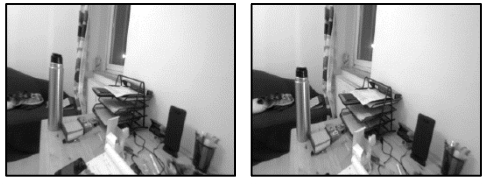
+ robust weights (e.g. Huber), see *iteratively reweighted least squares*

+ *Levenberg-Marquad (LM)* Algorithm

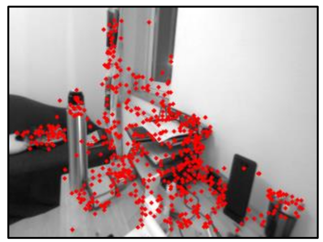
So, what‘s new?

Keypoint-Based

Input Images



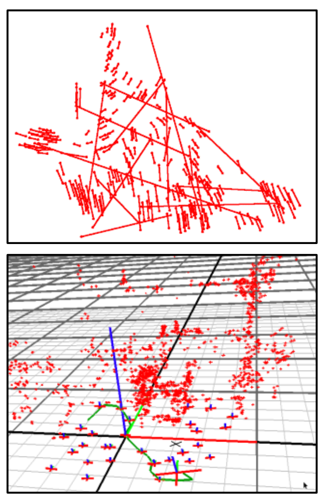
Extract & Match Features
(SIFT / SURF / ...)



abstract images to feature observations

Track:
min. **reprojection** error
(point distances)

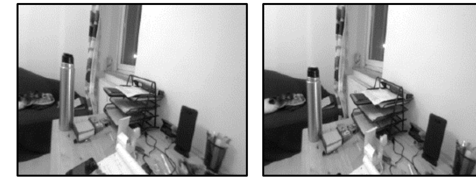
Map:
est. **feature-parameters**
(3D points / normals)



can only use & reconstruct corners

Semi-Dense (direct)

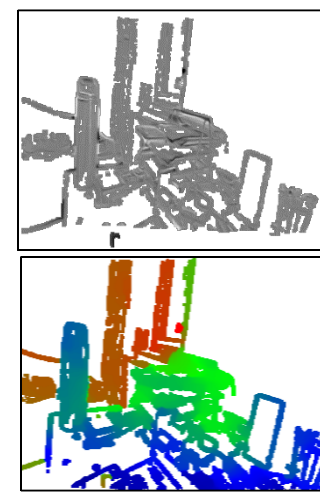
Input Images



keep full image

Track:
min. **photometric** error
(intensity difference)

Map:
est. **per-pixel depth**
(semi-dense depth map)



can use & reconstruct whole image