BAD-SLAM: Bundle Adjusted Direct RGB-D SLAM

Authors: Thomas Schöps; Torsten Sattler; Marc Pollefeys

Conference: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

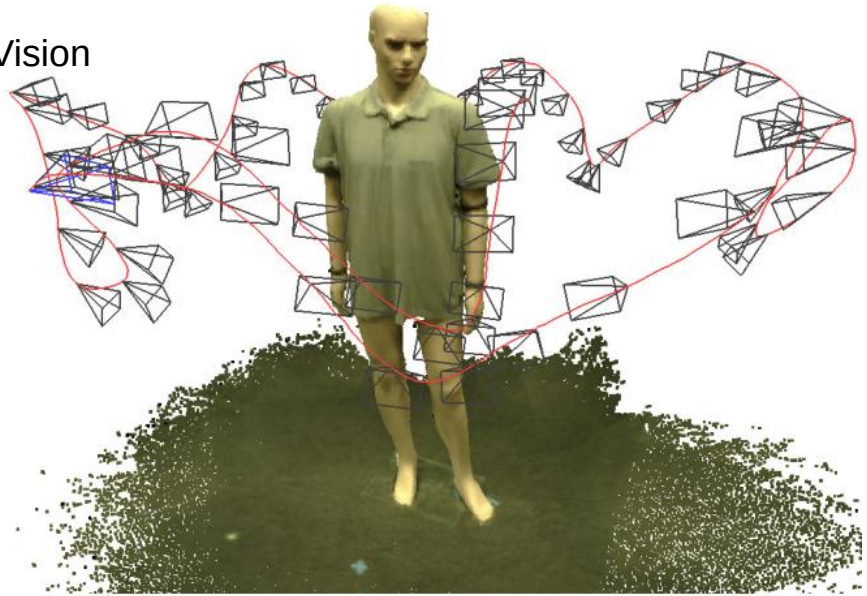
Maximilian Listl

Technical University of Munich

Faculty of Informatics

Chair of Computer Vision & Artificial Intelligence

Garching, 5th of October 2020



Contents

Motivation - Indirect vs Direct Methods

Two previous approaches for Context: Indirect Methods (ORB-SLAM 2), Direct Methods (DVO-SLAM)

1. Image Input
2. Identify Features (e.g. corners)
3. Track Features for initial Trajectory Estimates of Camera and Map estimates
4. Optimize Pose and Map Estimation

1. Image Input
2. Get initial Trajectory and Map Estimates by using entire Image Information (depth map, Intensity)
3. Optimize Pose and Map Estimation

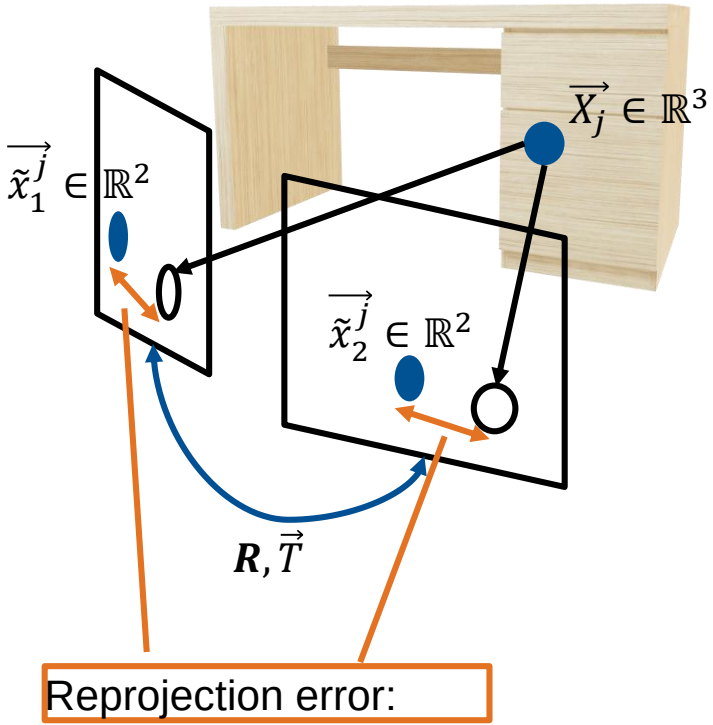
Motivation - Bundle Adjustment

- Optimization problem to jointly estimate optimal Pose(Trajectory) and optimal 3D coordinates
- e.g. Optimizing Reprojection error

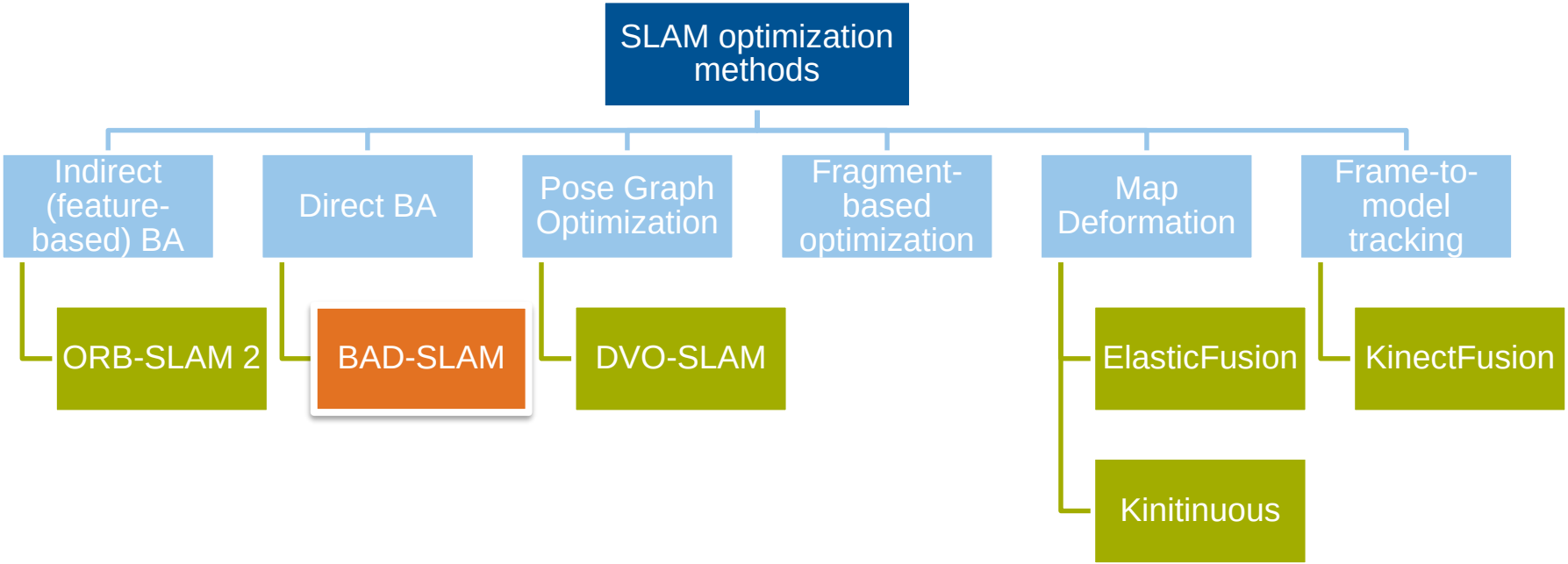
$$\min_{R,T,X_1,\dots,X_N} E(\boldsymbol{R}, T, X_1, \dots, X_N)$$

$$\min_{R,T,X_1,\dots,X_N} \sum_{j=1}^{N} \left\| \overrightarrow{\tilde{x}_1^j} - \pi\left(\overrightarrow{X_j}\right) \right\|_2^2 + \left\| \overrightarrow{\tilde{x}_2^j} - \pi\left(\boldsymbol{R}, \vec{T}, \overrightarrow{X_j}\right) \right\|_2^2$$

Noisy data from image



Motivation - Many ways to compute RGB-D SLAM



Motivation – Authors Goal

- Goal:
  - Use fast direct Bundle Adjustment
  - Use dense RGB-D measurements
  - Perform RGB-D SLAM in real-time

Contents

First Contribution: BAD-SLAM algorithm

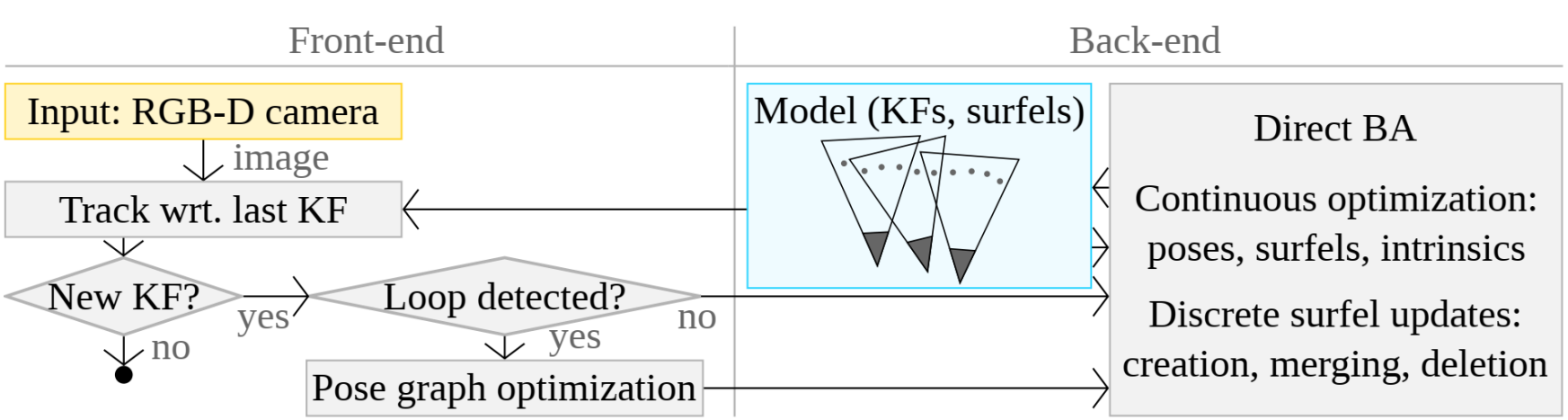
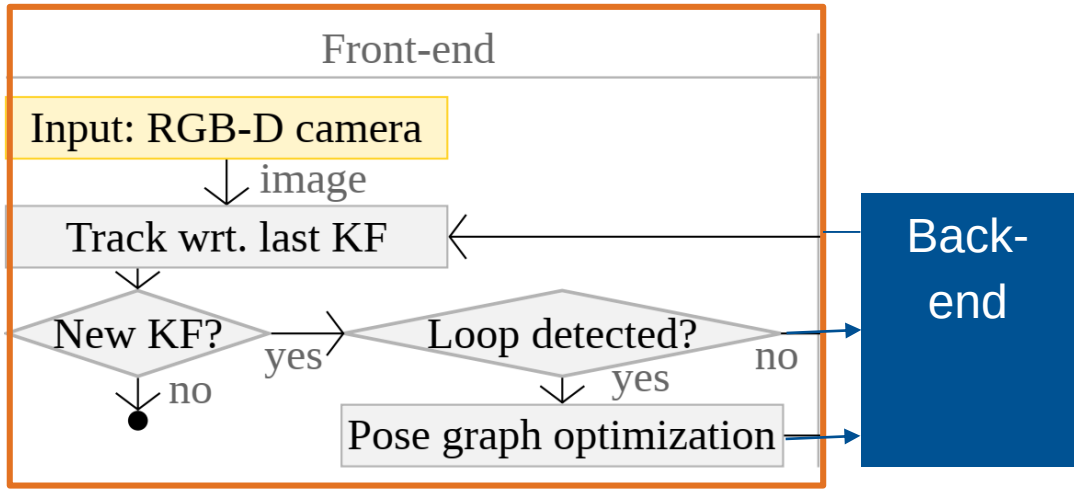


Figure 2. Approach overview. KF stands for keyframe.

First Contribution: BAD-SLAM algorithm



Generic Front-end

Keyframe Selection: every 10th frame

Loop Closure Detection:

- Identifies keyframe m which is most similar with the last keyframe k
- If identified then Loop closure by Pose graph Optimization

First Contribution: BAD-SLAM algorithm

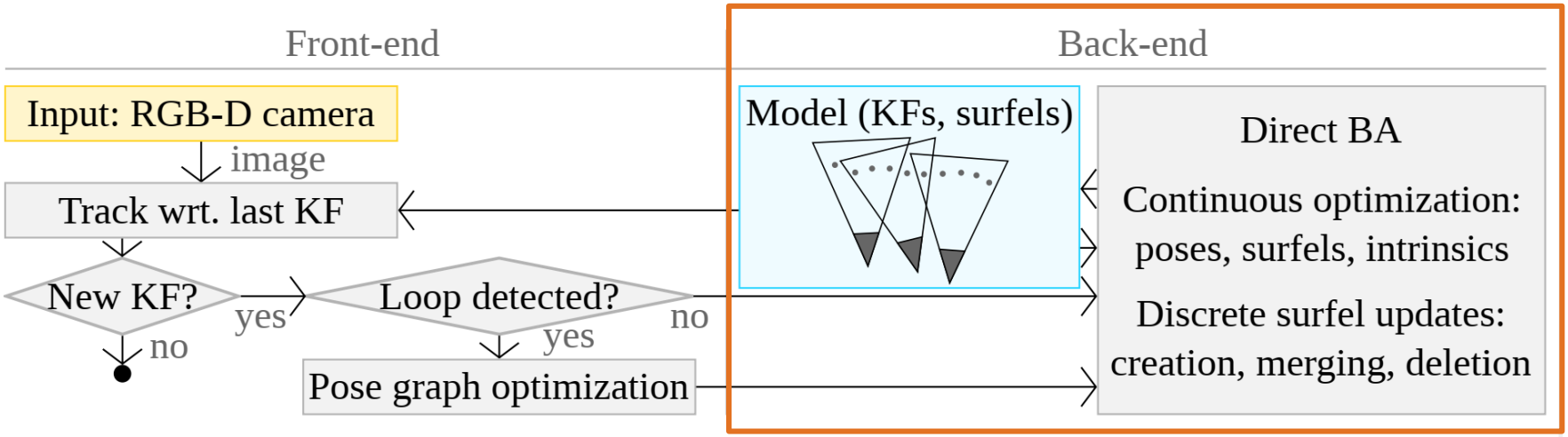


Figure 2. Approach overview. KF stands for keyframe.

Surfels - Surface Elements

Components:

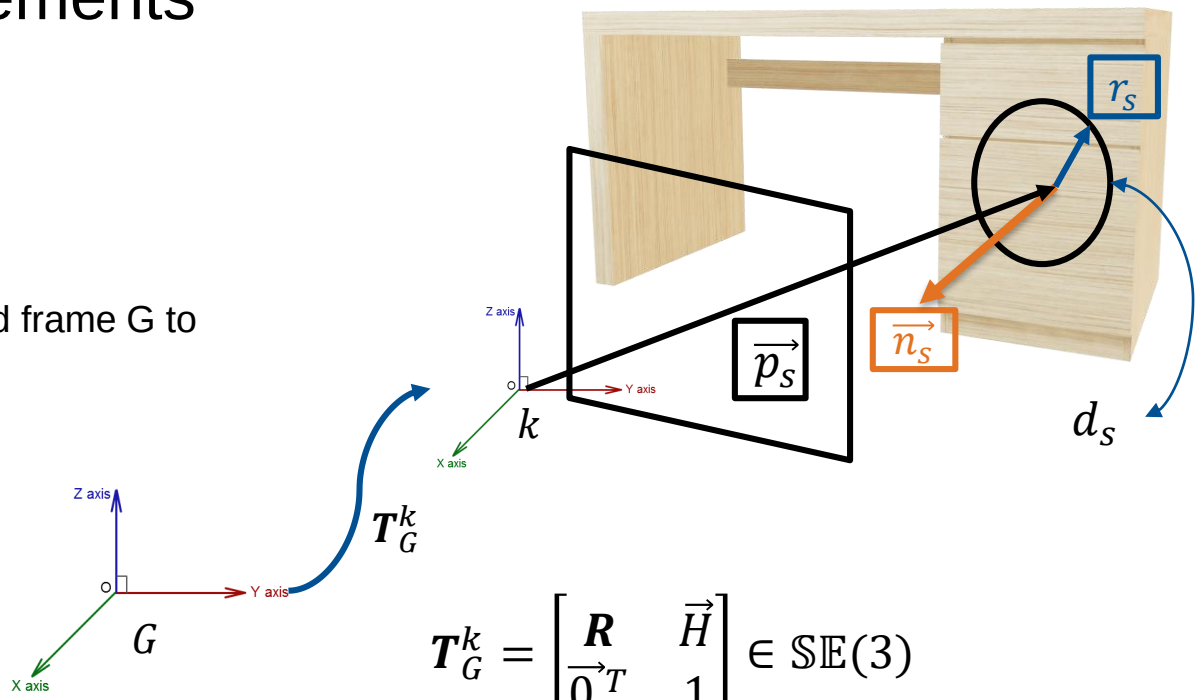
$\overrightarrow{n_s} \in \mathbb{R}^3$: Surfel normal

$r_s \in \mathbb{R}$: Surfel radius

$\overrightarrow{p_s} \in \mathbb{R}^3$: Center Point of Surfel

$\boldsymbol{T}_G^k$:Linear Transform from fixed world frame G to local keyframe k

$d_s \in \mathbb{R}$: scalar visual descriptor



$$\boldsymbol{T}_G^k = \begin{bmatrix} \boldsymbol{R} & \vec{H} \\ \overrightarrow{0^T} & 1 \end{bmatrix} \in \mathbb{SE}(3)$$

Cost function

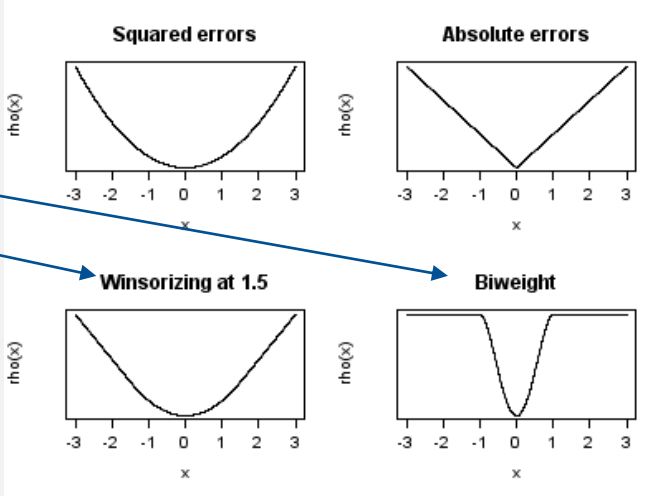
$$C(K,S) = \sum_{k \in K} \sum_{s \in S} \rho_{Tukey}\left(\sigma_D^{-1}\, r_{geom}(s,k)\right) + \omega_{photo}\, \rho_{Huber}(\sigma_p^{-1}\, r_{photo}(s,k))$$

$$\omega_{photo} = 10^{-2}$$

$$\sigma_p = \frac{1}{180}$$

ρ_{Tukey}: $Robust\ loss\ function\ (M-Estimator)$

ρ_{Huber}: $Robust\ loss\ function\ (M-Estimator)$



https://upload.wikimedia.org/wikipedia/commons/c/c1/RhoFunctions.png

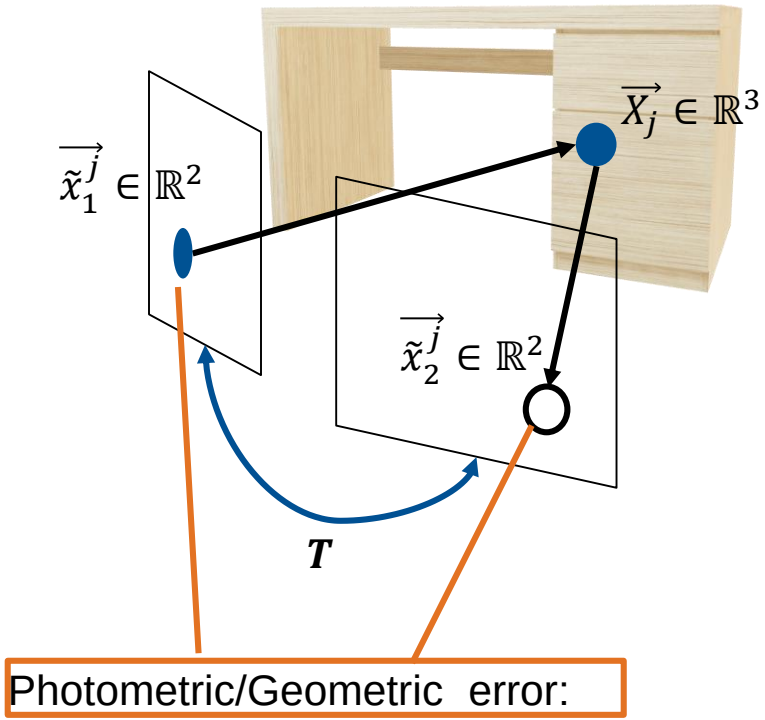
Photometric and Geometric Error

Photometric Error: (Color/Intensity)

$$r_{photo} = I_2\left(\pi\left(T\left(\overrightarrow{X_j}\right)\right)\right) - I_1(x)$$

Geometric Error: (Depth)

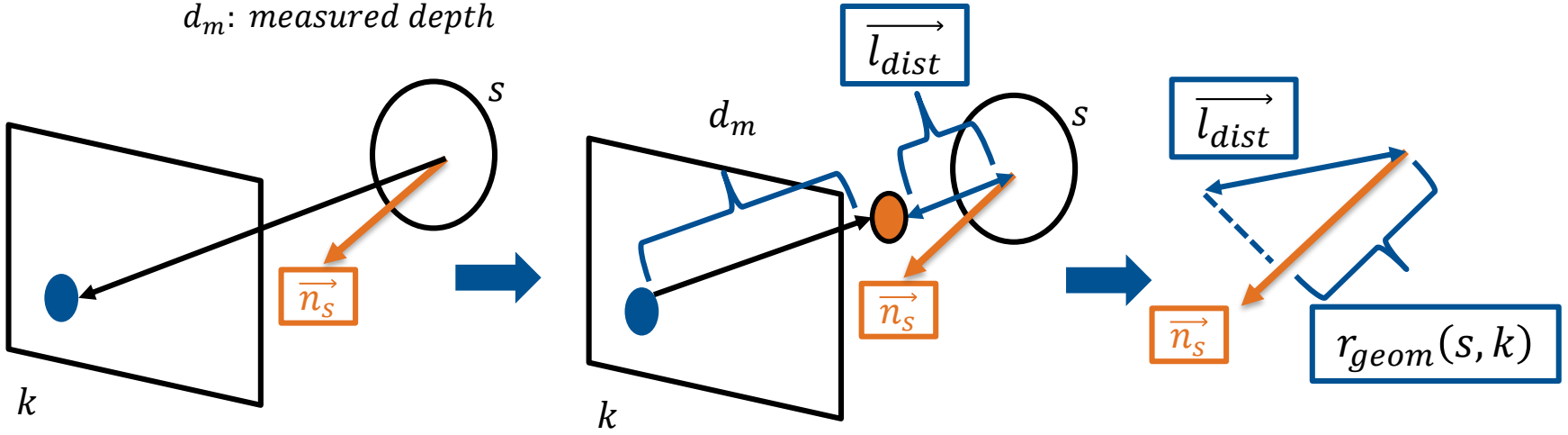
$$\mathrm{r}_{geo} = Z_2\left(\pi\left(T\left(\overrightarrow{X_j}\right)\right)\right) - Z_1(x)$$



Cost function - Geometric Residual

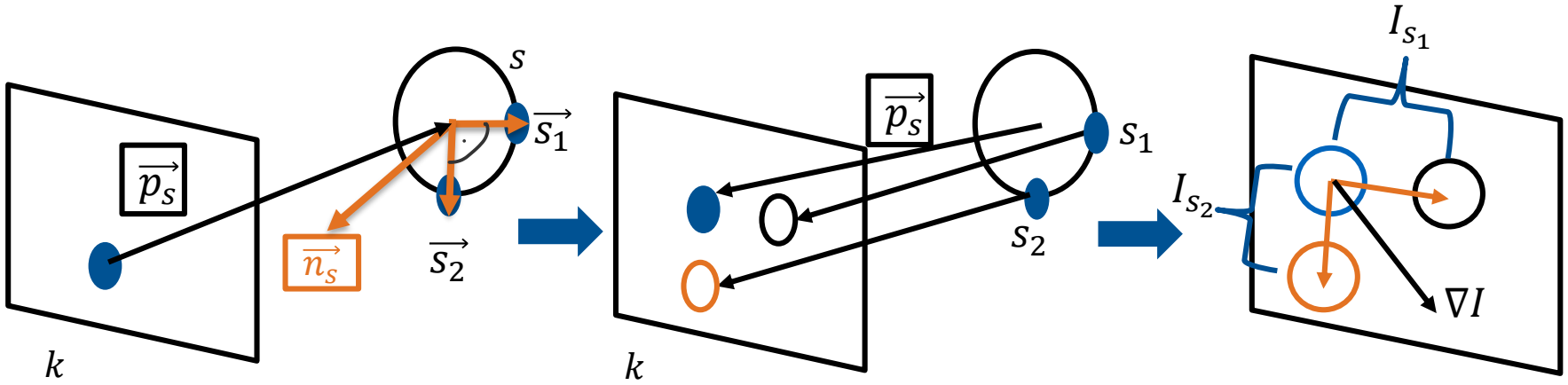
$$r_{geom}(s,k) = \left(\boldsymbol{T}_G^k \overrightarrow{n_s}\right)^T \underbrace{\left(\pi_{D,k}^{-1}\left(\hat{\pi}_{D,k}\left(\boldsymbol{T}_G^k \overrightarrow{p_s}\right)\right) - \boldsymbol{T}_G^k \overrightarrow{p_s}\right)}_{\overrightarrow{l_{dist}}}$$

d_m: *measured depth*



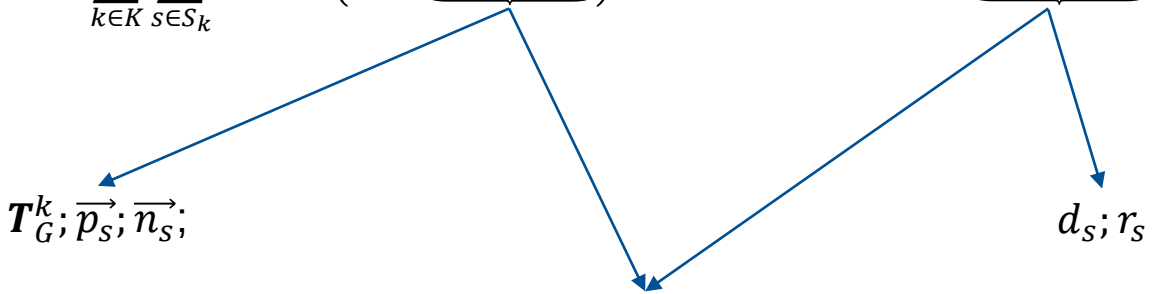
Cost function - Photometric Residual

$$r_{photo}(s,k) = \left\| \begin{matrix} I\left(\pi_{I,k}(\overrightarrow{s_1})\right) - I(\pi_{I,k}(\overrightarrow{p_s})) \\ I\left(\pi_{I,k}(\overrightarrow{s_2})\right) - I(\pi_{I,k}(\overrightarrow{p_s})) \end{matrix} \right\|_2 - d_s = \left\| \begin{matrix} I_{s_1} \\ I_{s_2} \end{matrix} \right\|_2 - d_s = \|\nabla I\|_2 - d_s$$



Too many variables to optimize!

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\ \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\ \rho_{Huber}(\sigma_p^{-1}\ \underbrace{r_{photo}(s,k)})$$



How do we optimize this Cost-function?

Solution:

- Alternating optimization
  - Most optimization variables separately updated or optimized
  - within each iteration the cost function is optimized by alternatingly fixing a set of variables

Optimization Algorithm (Bundle Adjustment)

- Divided Algorithm into 3 Parts
  - Creation of surfels ($|S|$)
  - Optimization of variables ($\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s}; d_s; \boldsymbol{K}$)
  - Cleanup and final update ($|S|; r_s$)

Algorithm 1 Surfel-based alternating direct BA scheme

1: **for all** keyframes **do** Create missing surfels
2: **for** $i \in [1, \text{max\_iteration\_count}]$ **do**
3: Update surfel normals
4: Optimize surfel positions and descriptors
5: **if** i = 1 **then** Merge similar surfels
6: Optimize keyframe poses
7: Optimize camera intrinsics (optionally)
8: **if** no keyframe moved **then break**
9: **for all** keyframes which moved in the last loop **do**
10: Merge similar surfels
11: Delete outlier surfels; Update surfel radii

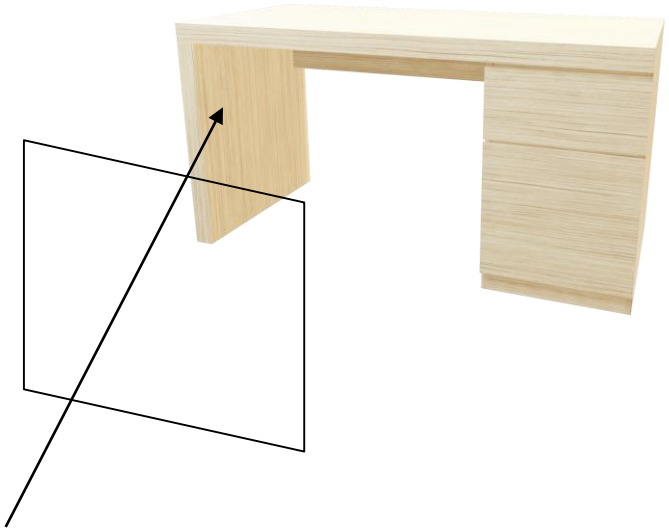
Optimization Algorithm (Bundle Adjustment)

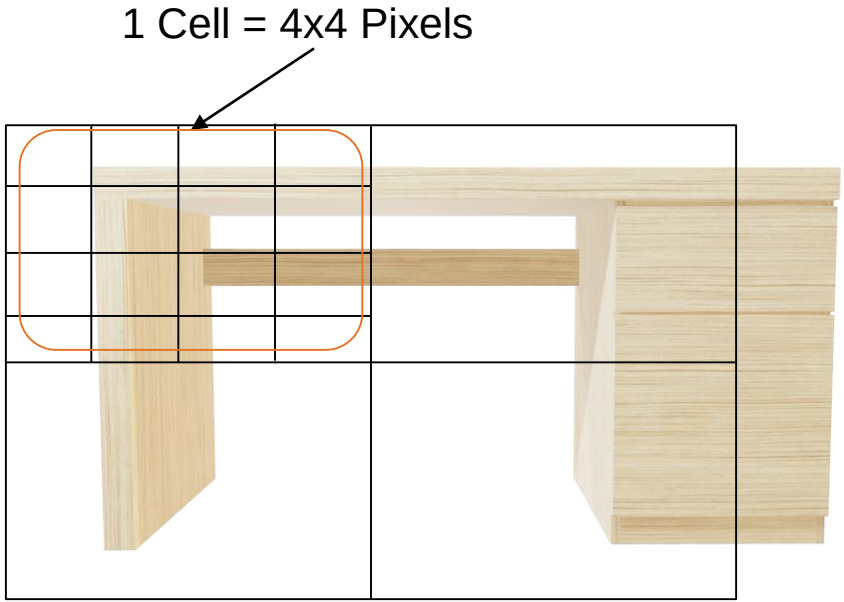
- Divided Algorithm into 3 Parts
  - Creation of surfels ($|S|$)
  - Optimization of variables ($\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s}; d_s; \boldsymbol{K}$)
  - Cleanup and final update ($|S|; r_s$)

Algorithm 1 Surfel-based alternating direct BA scheme

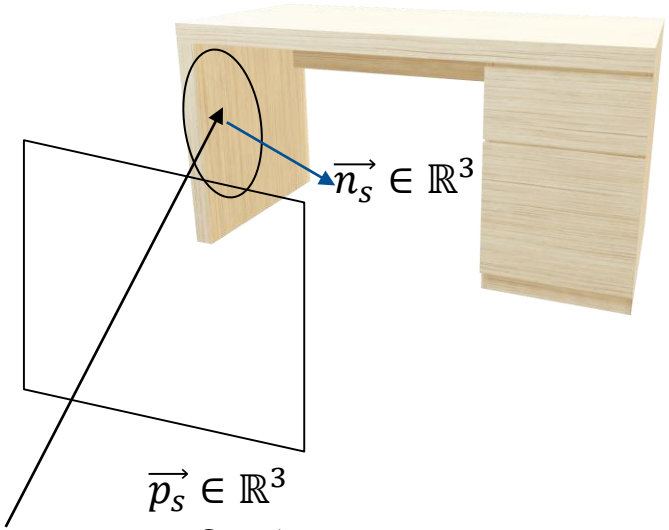
1: **for all** keyframes **do** Create missing surfels
2: **for** $i \in [1, \text{max\_iteration\_count}]$ **do**
3: Update surfel normals
4: Optimize surfel positions and descriptors
5: **if** i = 1 **then** Merge similar surfels
6: Optimize keyframe poses
7: Optimize camera intrinsics (optionally)
8: **if** no keyframe moved **then break**
9: **for all** keyframes which moved in the last loop **do**
10: Merge similar surfels
11: Delete outlier surfels; Update surfel radii

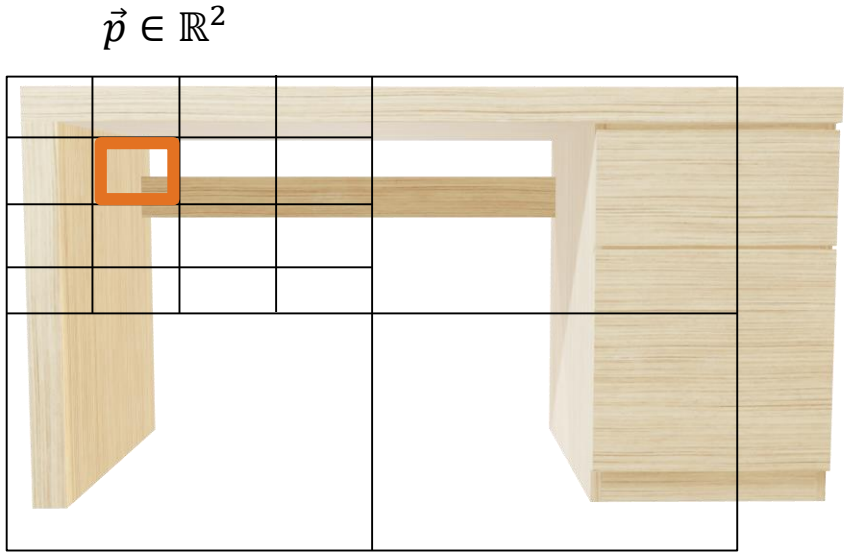
Part 1 – Surfel Creation 1



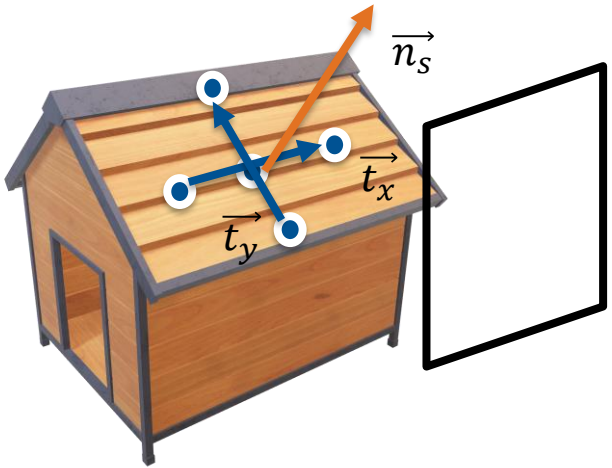


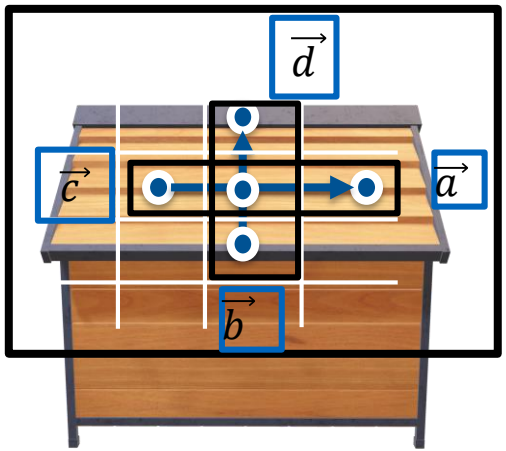
Part 1 – Surfel Creation 1





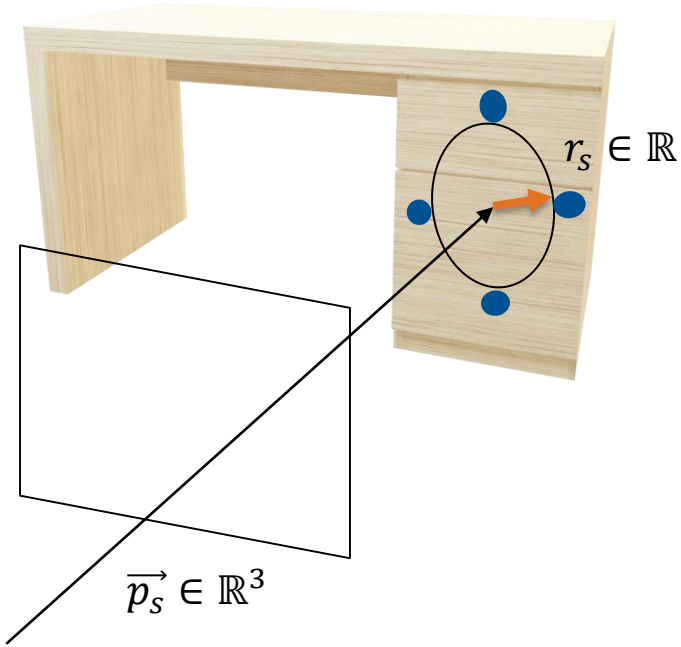
Computing initial n_s

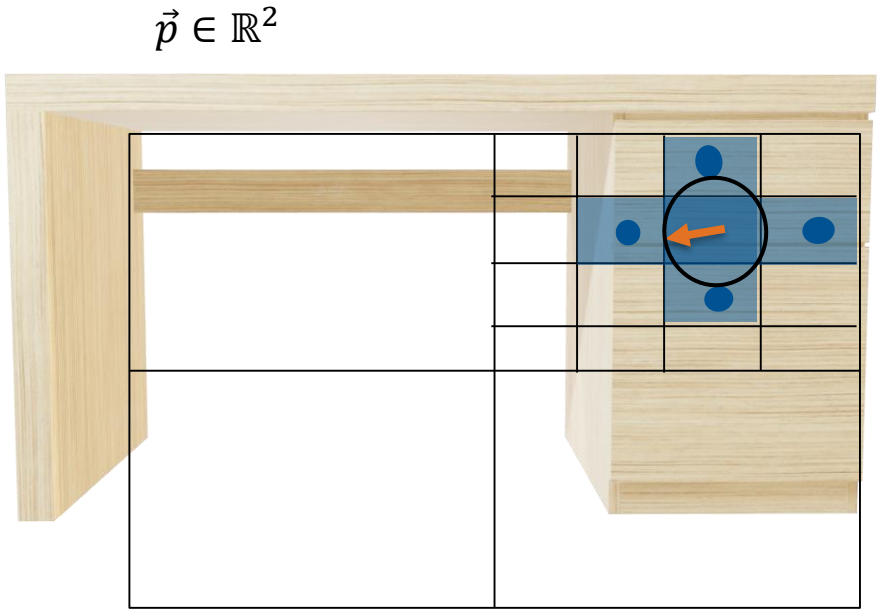




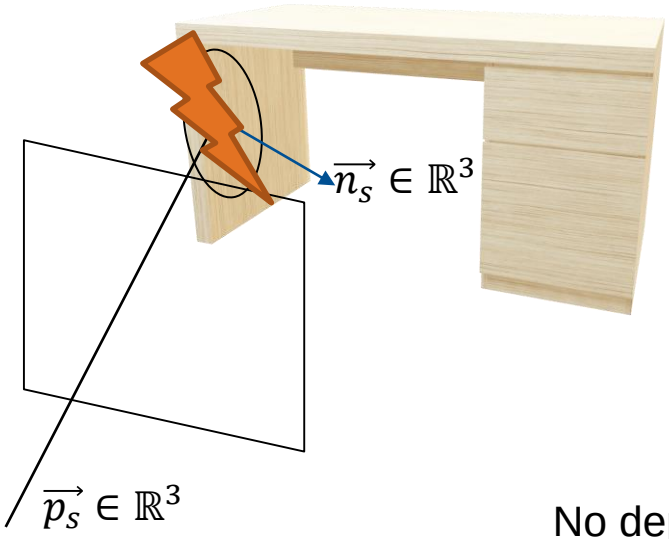
- $\overrightarrow{t_x} = (\vec{a} - \vec{c}) = \begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix}$
- $\overrightarrow{t_y} = \left(\vec{d} - \vec{b}\right) = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}$
- $due\ to\ \overrightarrow{n_s} \perp \overrightarrow{t_x}\ and\ \overrightarrow{n_s} \perp \overrightarrow{t_y}$
- $\overrightarrow{n_s} = t_y \times t_x = \begin{pmatrix} -4 \\ 0 \\ 4 \end{pmatrix}$

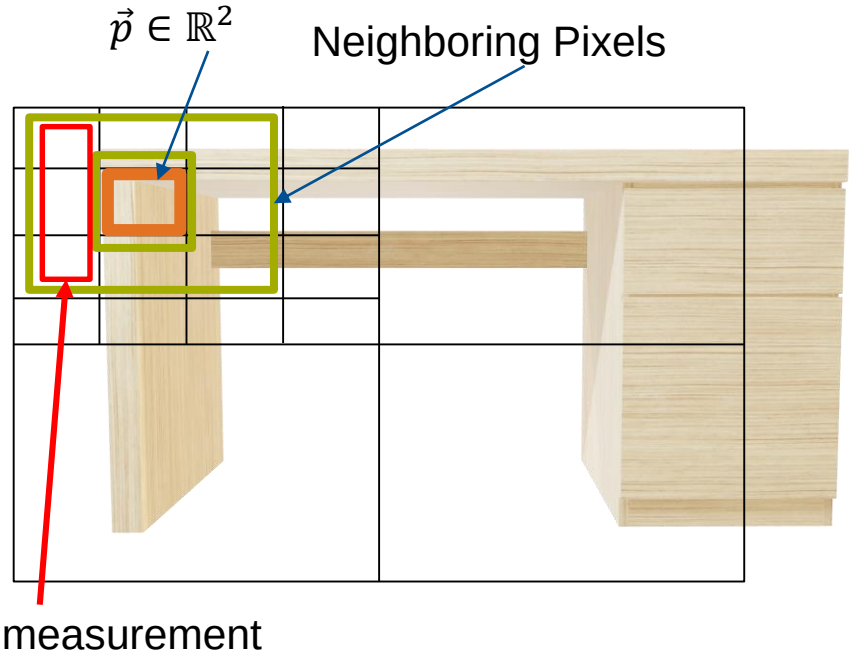
Part 1 – Computing r_s



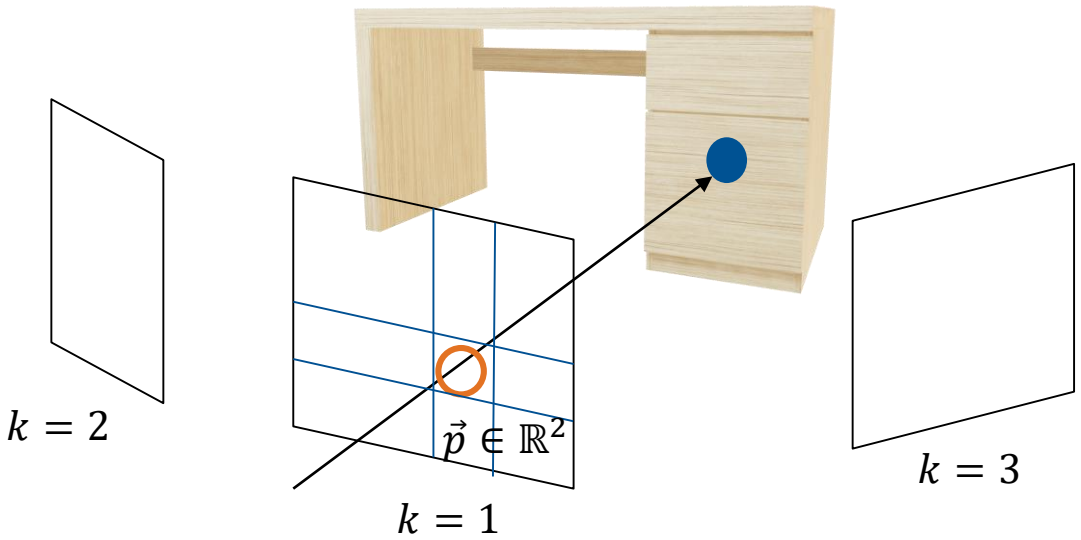


Part 1 – Surfel Creation 1

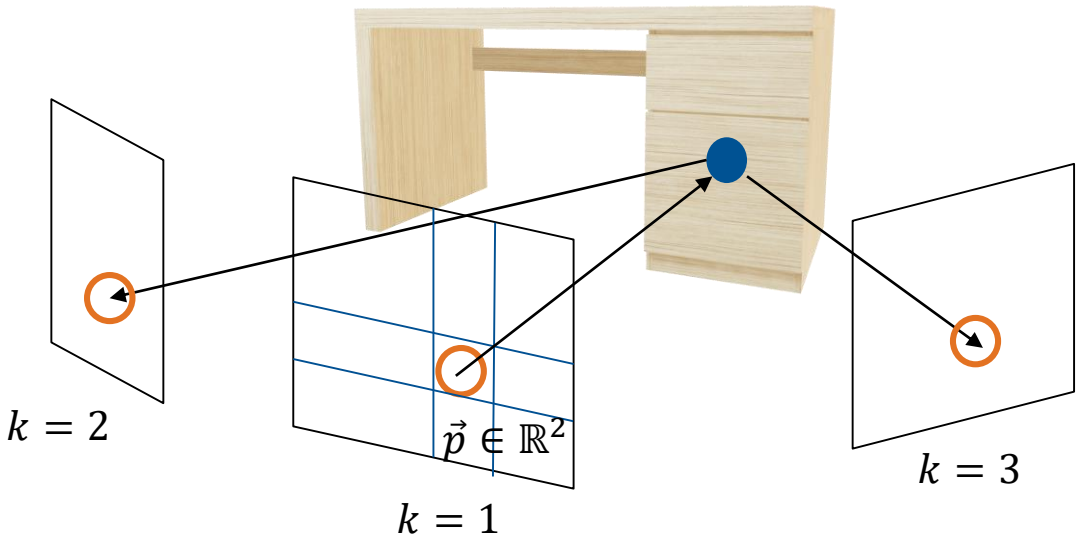




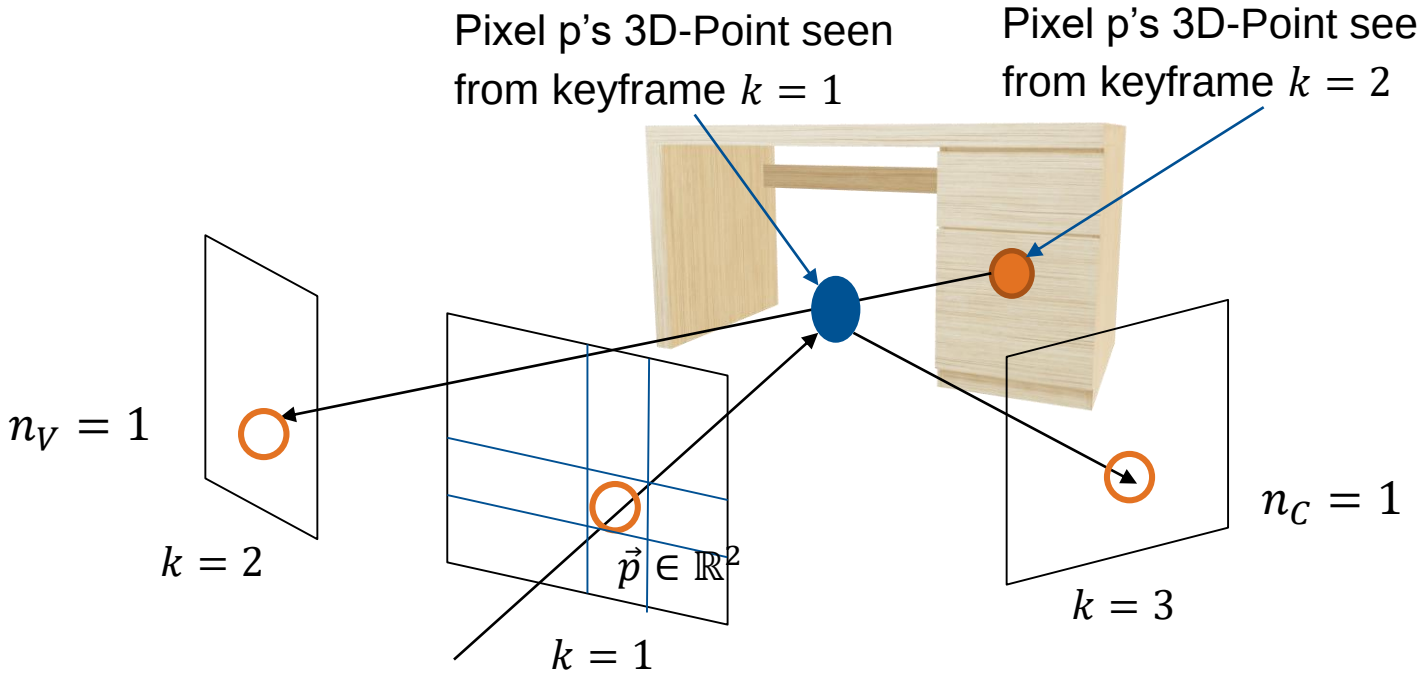
Part 1 – Outlier Filter



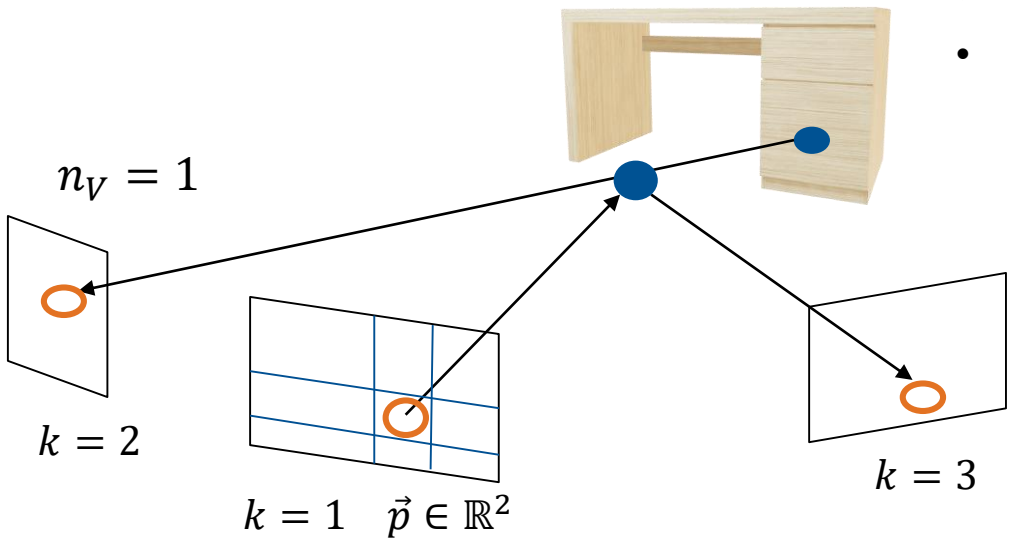
Part 1 – Outlier Filter



Part 1 – Outlier Filter



Part 1 – Outlier Filter



Outlier if :

- $n_C < n_{min}$ *or*
- $n_V > n_C$
  - $with\ n_{min} = \min(3, 1 + \lfloor 0.2|K| \rfloor)$
    $|K| \coloneqq number\ of\ Keyframes$

$n_C = 1$

Optimization Algorithm (Bundle Adjustment)

- Divided Algorithm into 3 Parts
  - Creation of surfels ($|S|$)
  - Optimization of variables ($\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s}; d_s; \boldsymbol{K}$)
  - Cleanup and final update ($|S|; r_s$)

Algorithm 1 Surfel-based alternating direct BA scheme

1: **for all** keyframes **do** Create missing surfels
2: **for** $i \in [1, \text{max\_iteration\_count}]$ **do**
3: Update surfel normals
4: Optimize surfel positions and descriptors
5: **if** i = 1 **then** Merge similar surfels
6: Optimize keyframe poses
7: Optimize camera intrinsics (optionally)
8: **if** no keyframe moved **then break**
9: **for all** keyframes which moved in the last loop **do**
10: Merge similar surfels
11: Delete outlier surfels; Update surfel radii

Part 2 – n_s Update

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\ \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\ \rho_{Huber}(\sigma_p^{-1}\ \underbrace{r_{photo}(s,k)})$$

$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

$d_s; r_s$

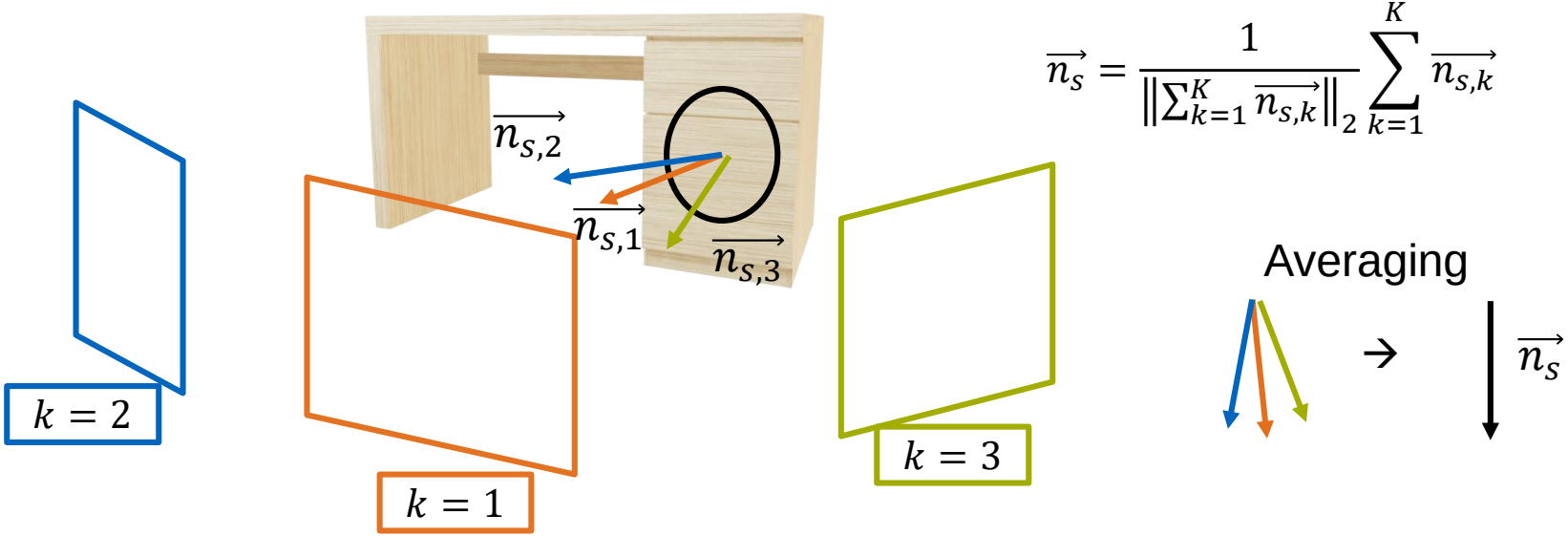
Optionally Intrinsics:

$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

Part 2 – n_s Update

Average all surfel normals and normalize them

$$\overrightarrow{n_s} = \frac{1}{\left\| \sum_{k=1}^{K} \overrightarrow{n_{s,k}} \right\|_2} \sum_{k=1}^{K} \overrightarrow{n_{s,k}}$$



Part 2 – Surfel position $\overrightarrow{p_s}$ and descriptor d_s

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\ \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\ \rho_{Huber}(\sigma_p^{-1}\ \underbrace{r_{photo}(s,k)})$$

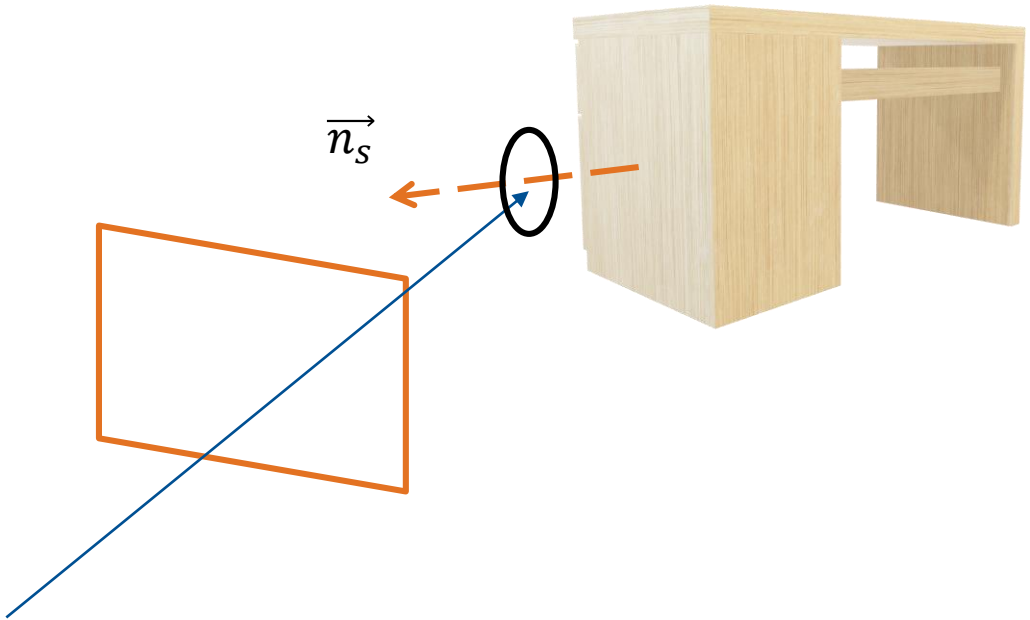
$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

$d_s; r_s$

Optionally Intrinsics:

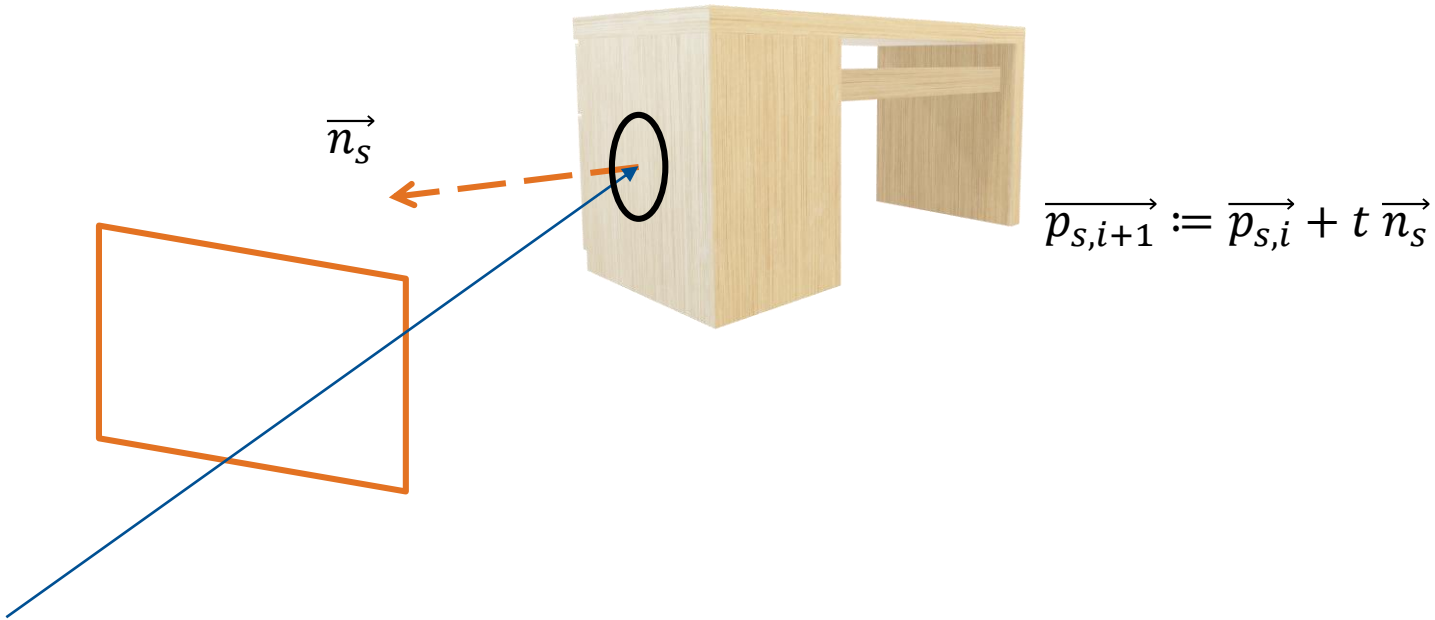
$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

Part 2 – Surfel position $\overrightarrow{p_s}$ and descriptor d_s



$$\overrightarrow{p_{s,i+1}} := \overrightarrow{p_{s,i}} + t\,\overrightarrow{n_s}$$

Part 2 – Surfel position $\overrightarrow{p_s}$ and descriptor d_s



Part 2 – Surfel position $\overrightarrow{p_s}$ and descriptor d_s

$$New\ Parametrization:$$
$$\overrightarrow{p_{s,i+1}} \coloneqq \overrightarrow{p_{s,i}} + t\,\overrightarrow{n_s}$$

$$\min_{t,\,d_s} C(K,S)$$

$$\begin{pmatrix} t \\ d_s \end{pmatrix}_{i+1} = \begin{pmatrix} t \\ d_s \end{pmatrix}_{i} - \underbrace{(\boldsymbol{J}^T\boldsymbol{J})^{-1}\,\boldsymbol{J}^T}_{\boldsymbol{M}}\overrightarrow{r}$$
$$with\ M \in \mathbb{R}^{2\times 2}$$

- Optimization of Cost-function:
  - Gauss-Newton method
  - Fix all variables but $\overrightarrow{p_s}$ & d_s
  - Each surfel is independent therefore for each surfel only need to compute 2x2-Matrix

Part 2 – Surfel Merging

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\ \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\ \rho_{Huber}(\sigma_p^{-1}\ \underbrace{r_{photo}(s,k)})$$

$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

$d_s; r_s$

Optionally Intrinsics:

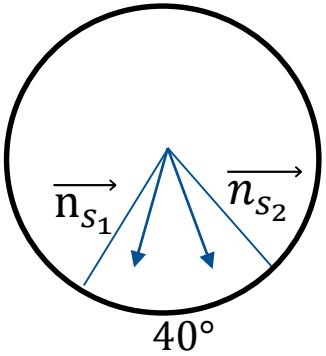
$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

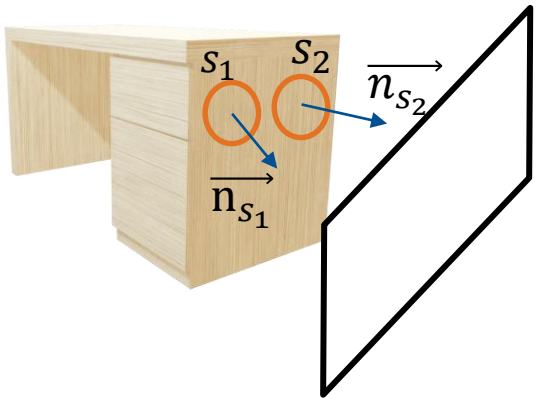
Part 2 – Surfel Merging

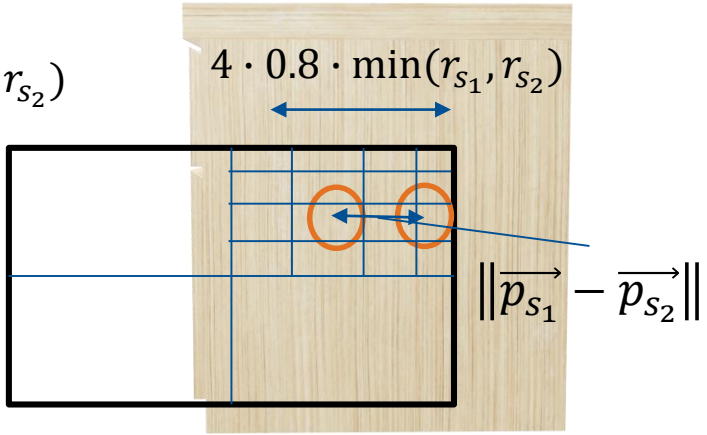
- Due to Noisy measurements we have created unnecessary surfels
- How do we eliminate these?

Part 2 – Surfel Merging

- Solution:
  - Merge similar surfels at first iteration of BA scheme
  - Conditions: (given 2 Surfels: s_1 & s_2)
    - Surfel normal: $\measuredangle(\overrightarrow{n_{s_1}}, \overrightarrow{n_{s_2}}) < 40°$
    - Surfel position: $\left\|\overrightarrow{p_{s_1}} - \overrightarrow{p_{s_2}}\right\| < 4 \cdot 0.8 \cdot \min(r_{s_1}, r_{s_2})$







Part 2 – Keyframe pose

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\ \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\ \rho_{Huber}(\sigma_p^{-1}\ \underbrace{r_{photo}(s,k)})$$

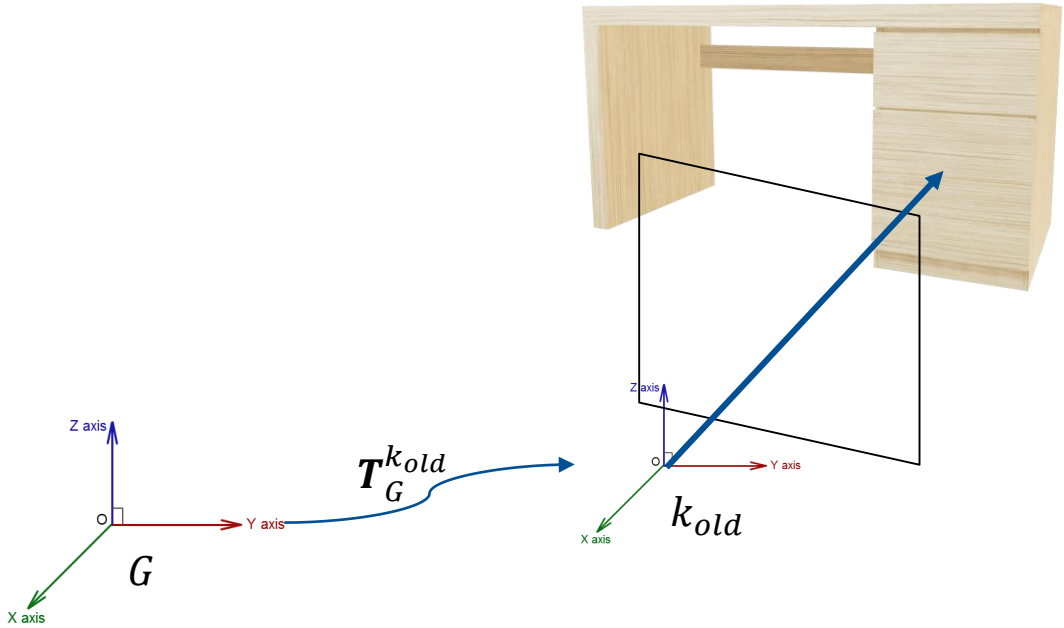
$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

$d_s; r_s$

Optionally Intrinsics:

$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

Part 2 – Keyframe pose



Part 2 – Keyframe pose

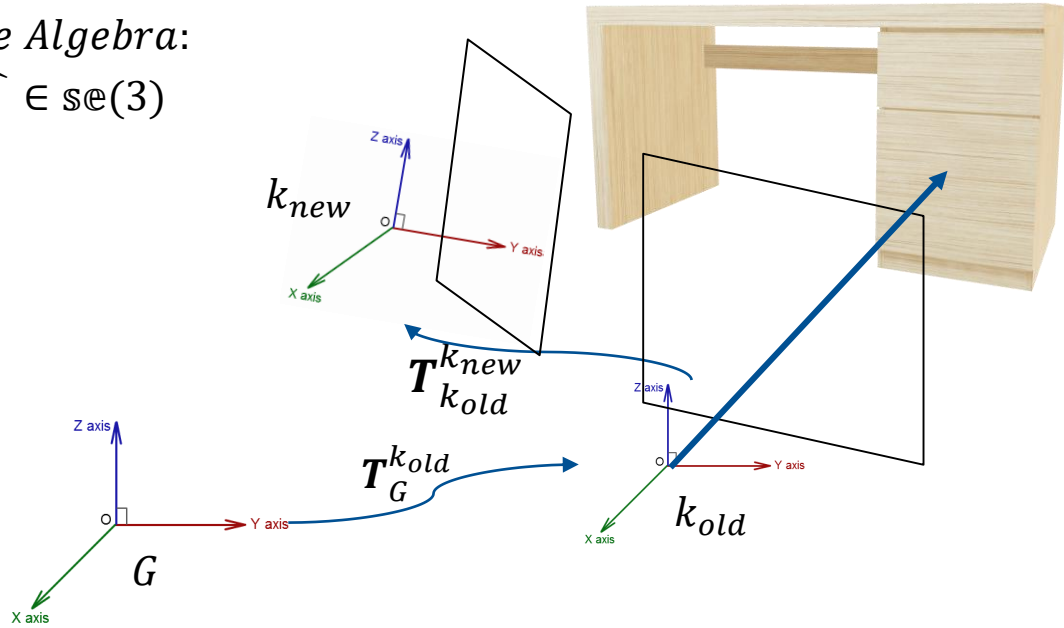
$New\ Parametrization$:

$$\boldsymbol{T}_G^{k_{new}} = \boldsymbol{T}_G^{k_{old}}\, \boldsymbol{T}_{k_{old}}^{k_{new}}$$

$$with\ \boldsymbol{T}_{k_{old}}^{k_{new}} = \exp(\hat{\xi})$$

$Lie\ Algebra$:

$$\hat{\xi} \in \mathbb{se}(3)$$



Part 2 – Keyframe pose

New Parametrization:

$$\boldsymbol{T}_G^{k_{new}} = \boldsymbol{T}_G^{k_{old}}\, \boldsymbol{T}_{k_{old}}^{k_{new}}$$

$$with\ \boldsymbol{T}_{k_{old}}^{k_{new}} = \exp(\hat{\xi})$$

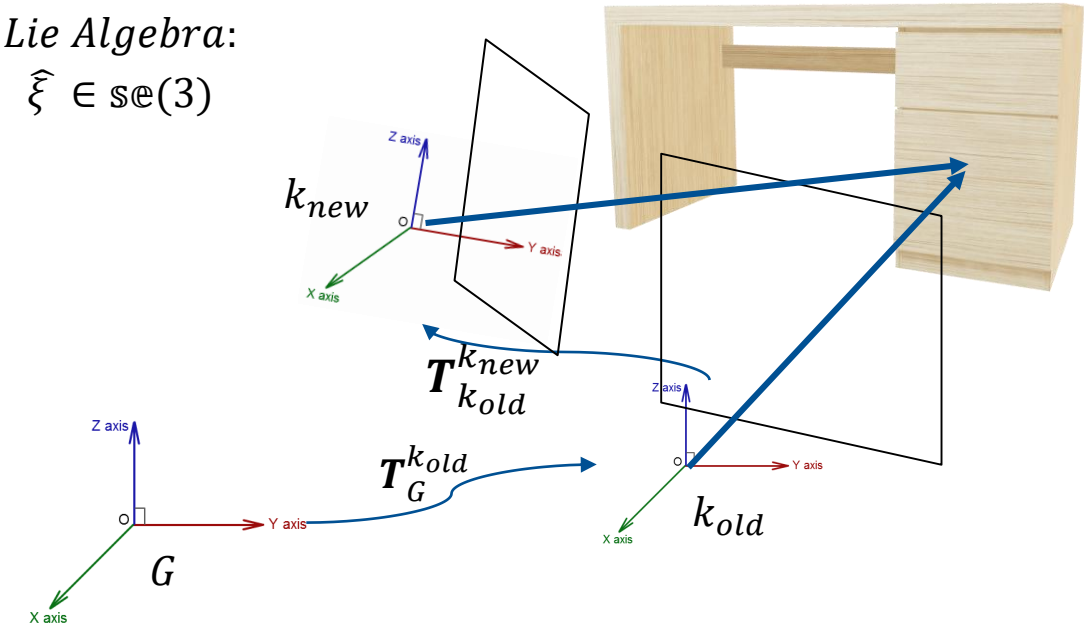
$$\min_{\overrightarrow{\xi}} C(K, S)$$

Gauss-Newton:

$$\overrightarrow{\xi_{i+1}} = \overrightarrow{\xi_i} - (\boldsymbol{J}^T\boldsymbol{J})^{-1}\,\boldsymbol{J}^T\overrightarrow{r}$$

Lie Algebra:

$$\hat{\xi} \in \mathfrak{se}(3)$$



Part 2 – Keyframe pose

New Parametrization:

$$\boldsymbol{T}_G^{k_{new}} = \boldsymbol{T}_G^{k_{old}}\, \boldsymbol{T}_{k_{old}}^{k_{new}}$$

$$with\ \boldsymbol{T}_{k_{old}}^{k_{new}} = \exp(\hat{\xi})$$

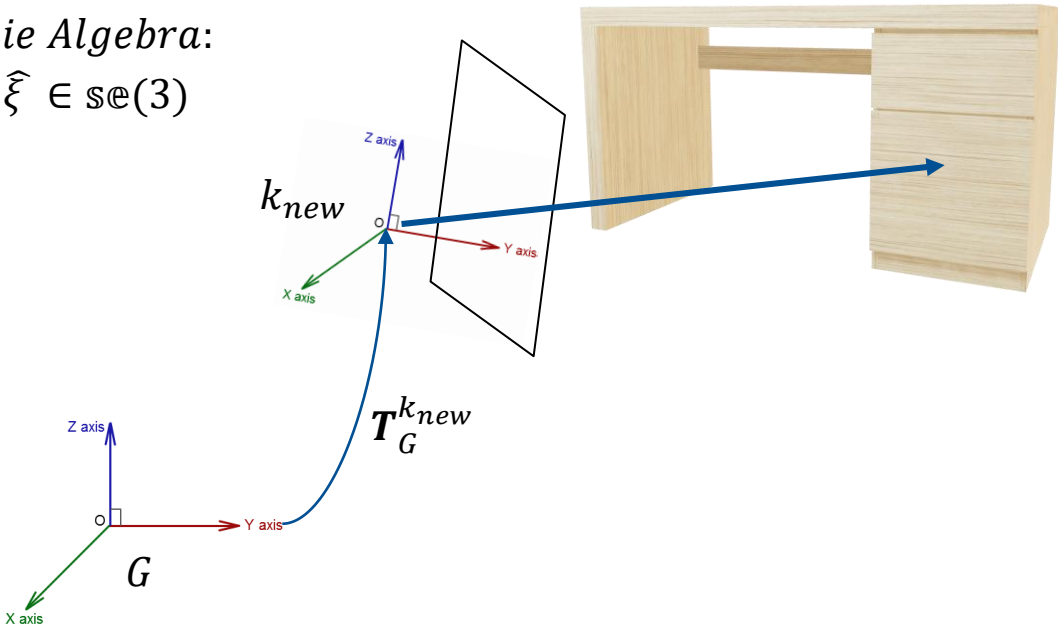
$$\min_{\overrightarrow{\xi}} C(K, S)$$

Gauss-Newton:

$$\overrightarrow{\xi_{i+1}} = \overrightarrow{\xi_i} - (\boldsymbol{J}^T\boldsymbol{J})^{-1}\,\boldsymbol{J}^T\overrightarrow{r}$$

Lie Algebra:

$$\hat{\xi} \in \mathfrak{se}(3)$$



Part 2 – Camera Intrinsics $\boldsymbol{K}$

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\, \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\, \rho_{Huber}(\sigma_p^{-1}\, \underbrace{r_{photo}(s,k)})$$

$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

$d_s; r_s$

Optionally Intrinsics:

$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

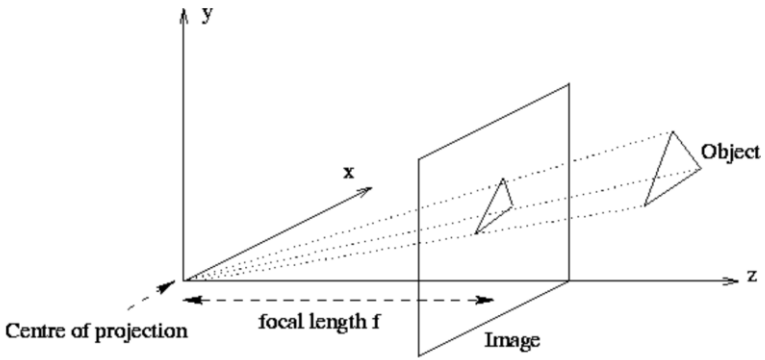
Part 2 – Camera Intrinsics $\boldsymbol{K}$

Assumption:

- Separate pinhole camera models for RGB and depth measurements
  - $\hat{\pi}_{D,k}(\dots) \rightarrow K_D$; $\pi_{I,k}(\dots) \rightarrow K_I$

- Additional model for depth deformation
  - $d_{true}(x,y) = d_{dist}(x,y) + D_\delta(x,y)\exp\left(-\alpha_1\, d_{dist}(x,y)\right)$
  - Only optimize parameters $D_\delta(x,y)$ and α_1

- Optimization fast due to Hessian diagonal at parts corresponding to $D_\delta(x,y)$



$$\frac{1}{d_{true}}\,\vec{p} = K\,\Pi_0\,\overrightarrow{p_s}$$

$$with\ d_{true}\ as\ inverse\ depth$$

Optimization Algorithm (Bundle Adjustment)

- Divided Algorithm into 3 Parts
  - Creation of surfels ($|S|$)
  - Optimization of variables ($\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s}; d_s; \boldsymbol{K}$)
  - Cleanup and final update ($|S|; r_s$)

Algorithm 1 Surfel-based alternating direct BA scheme

1: **for all** keyframes **do** Create missing surfels
2: **for** $i \in [1, \text{max\_iteration\_count}]$ **do**
3: Update surfel normals
4: Optimize surfel positions and descriptors
5: **if** i = 1 **then** Merge similar surfels
6: Optimize keyframe poses
7: Optimize camera intrinsics (optionally)
8: **if** no keyframe moved **then break**
9: **for all** keyframes which moved in the last loop **do**
10: Merge similar surfels
11: Delete outlier surfels; Update surfel radii

Part 3 – Delete Outlier Surfels

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\, r_{geom}(s,k)\right) + \omega_{photo}\, \rho_{Huber}(\sigma_p^{-1}\, r_{photo}(s,k))$$

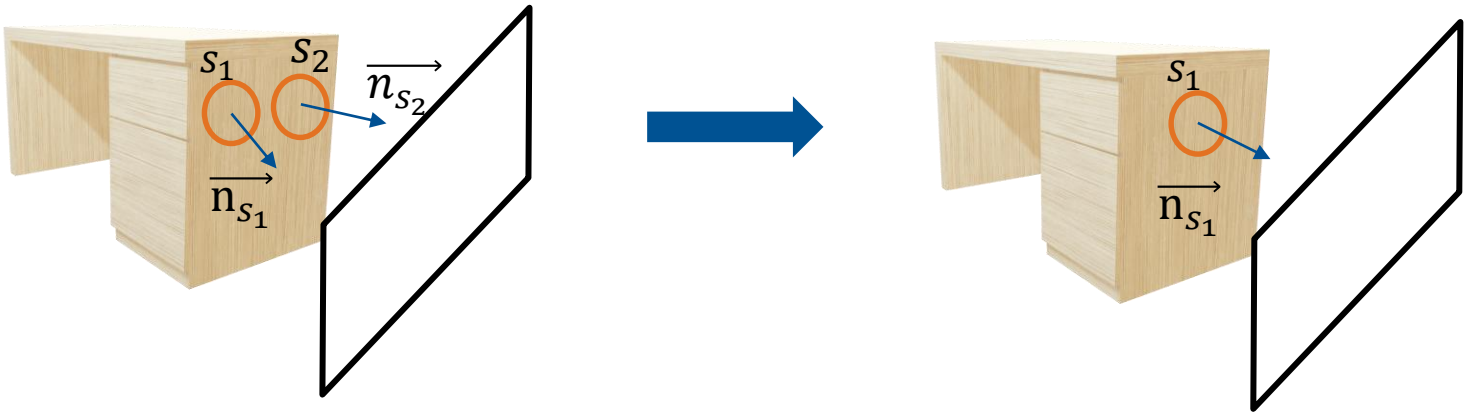
$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

$d_s; r_s$

Optionally Intrinsics:

$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

Part 2 – Surfel Merging



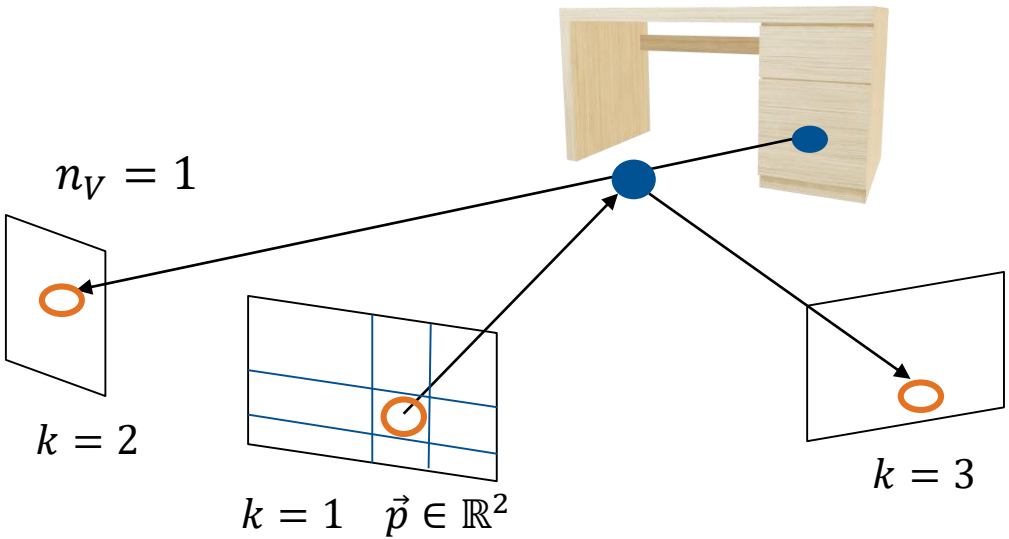
Part 3 – Delete Outlier Surfels

Outlier if :

$$n_C < n_{min} \; or \; n_V > n_C$$

$$with \;\; n_{min} = \min(3, 1 + \lfloor 0.2|K| \rfloor)$$

$$|K| \coloneqq number \; of \; Keyframes$$



$n_C = 1$

Part 3 – Radius Update

$$C(K,S) = \sum_{k \in K} \sum_{s \in S_k} \rho_{Tukey}\left(\sigma_D^{-1}\ \underbrace{r_{geom}(s,k)}\right) + \omega_{photo}\ \rho_{Huber}(\sigma_p^{-1}\ \underbrace{r_{photo}(s,k)})$$

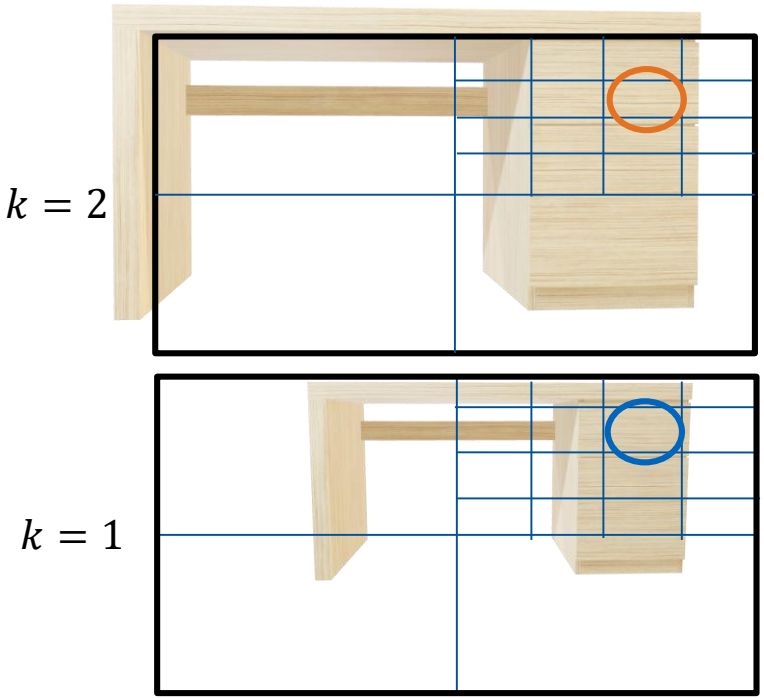
$\boldsymbol{T}_G^k; \overrightarrow{p_s}; \overrightarrow{n_s};$

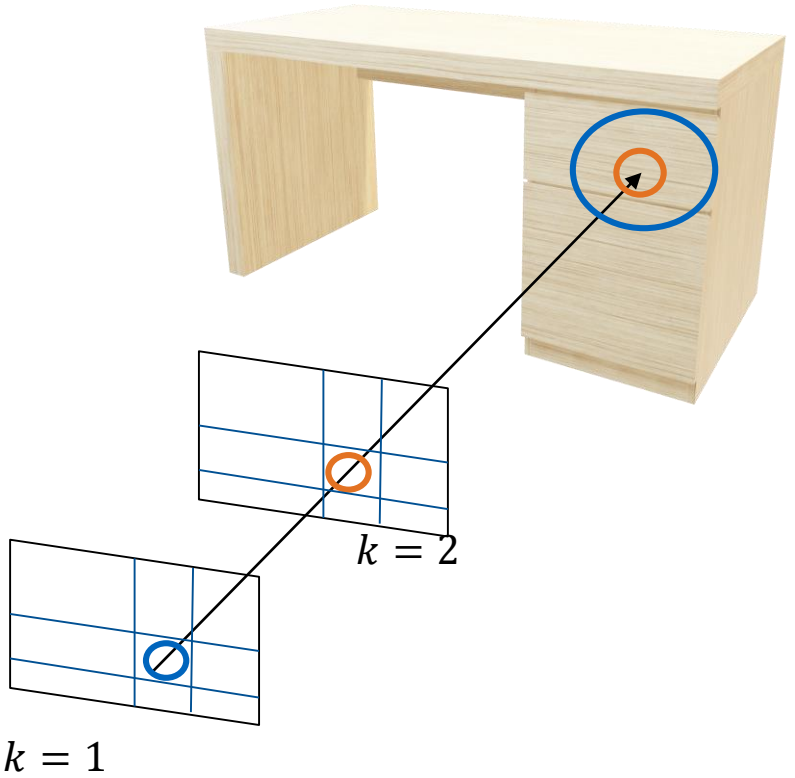
$d_s; r_s$

Optionally Intrinsics:

$\hat{\pi}_{D,k}(\dots); \pi_{I,k}(\dots) \rightarrow \boldsymbol{K}$

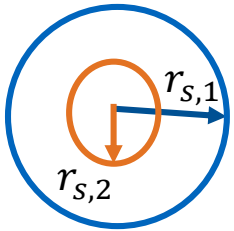
Part 3 – Radius Update

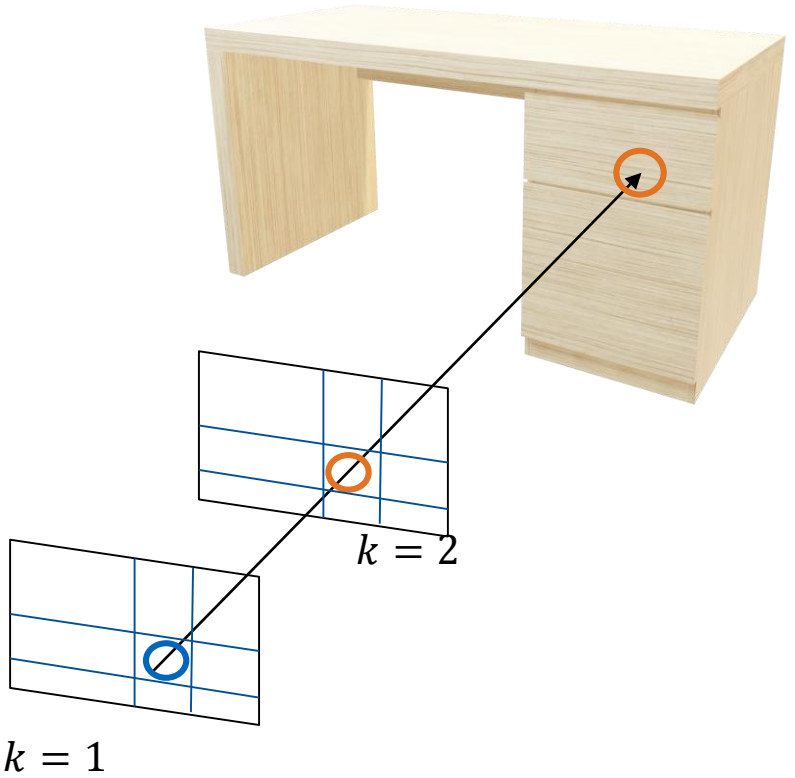




Part 3 – Radius Update

$$r_{s,new} = \min_i r_{s,i}$$





Contents

Interim report: Works but not good enough

Benchmark with TUM RGB-D dataset:

- Direct Methods (DVO-SLAM, BAD-SLAM) surpassed by indirect methods (ORB-SLAM 2)
- Assumption: Worse performance due to:
  - Rolling shutter
  - Asynchronous Frames (RGB images and Depth map)
  - Badly calibrated camera intrinsics
  - => To what extend is BAD-SLAM affected by distortions?

	fr1/desk	fr2/xyz	fr3/office	avg. rank
BundleFusion [7]	**1.6 (1)**	1.1 (3)	2.2 (4)	2.7 (2)
DVO SLAM [30]	2.1 (5)	1.8 (6)	3.5 (8)	6.3 (6)
ElasticFusion [69]	2.0 (4)	1.1 (3)	1.7 (2)	3.0 (4)
Kintinuous [68]	3.7 (8)	2.9 (9)	3.0 (6)	7.7 (8)
MRSMap [60]	4.3 (9)	2.0 (7)	4.2 (9)	8.3 (9)
ORB-SLAM2 [44]	**1.6 (1)**	**0.4 (1)**	**1.0 (1)**	**1.0 (1)**
PSM SLAM [70]	**1.6**	-	3.1	-
RGB-D SLAM [9]	2.3 (6)	0.8 (2)	3.2 (7)	5.0 (5)
VoxelHashing [47]	2.3 (6)	2.2 (8)	2.3 (5)	6.3 (6)
Ours (fixed intr.)	3.6	1.2	2.5	-
Ours	1.7 (3)	1.1 (3)	1.7 (2)	2.7 (2)

Table 2. ATE RMSE results in cm on TUM RGB-D datasets (rank in brackets). Ours achieves the second best average rank after ORB-SLAM2 and alongside BundleFusion. Results for other methods are as reported in [7], [70] and [44]. Our results without intrinsics and depth deformation optimization are clearly worse, showing that this is necessary for these datasets.

Interim report: How badly is BAD-SLAM affected?

Benchmark with synthetic Dataset:

- 4 Datasets out of TUM RGB-D dataset
  - Clean
  - Asynchronous
  - Rolling Shutter
  - Asynchronous + Rolling Shutter

- Observation:
  - effects degrade SLAM results
  - Not all effects have been modelled

- => Do we solve this issue by modelling every effect in Software?

	clean		async		rs		async & rs	
	avg.	med.	avg.	med.	avg.	med.	avg.	med.
BundleFusion [7]	0.34	0.22	1.10	1.14	1.10	1.02	1.48	1.40
DVO SLAM [30]	0.32	0.23	2.33	0.72	5.10	1.37	4.94	1.39
ElasticFusion [69]	1.11	0.90	1.98	1.17	2.70	1.77	3.19	2.52
ORB-SLAM2 [44]	0.47	0.30	0.60	0.40	3.25	1.57	3.49	1.55
Ours	**0.15**	**0.02**	**0.40**	**0.21**	**0.99**	**0.87**	**1.01**	**0.98**

Table 3. ATE RMSE [cm] averages and medians for seven synthetic datasets per category. Asynchronous RGB-D frames (async) and rolling shutter (rs) both worsen the results.

Contents

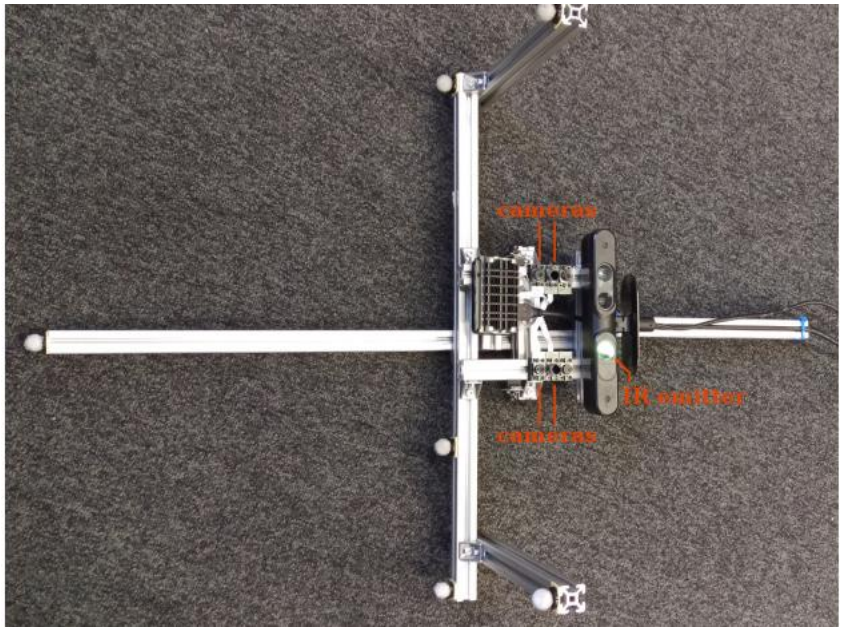
Second Contribution: Better Datasets

No, due to:
High implementation effort
High computational demand => high runtimes
Possibility of degenerate cases

Solution:
Solve issue with better hardware.

Second Contribution: Better Datasets

- Synchronized, Global shutter RGB and depth sensors
- Infrared Emitter
- Vicon Markers and Vicon System for trajectory tracking (ground truth)
- Well calibrated System
  - Optional camera intrinsics optimization not necessary



Second Contribution: New Benchmark

- 61 training sets (with ground truth)
- 35 test sets (non-public ground truth)
  - Online Leaderboard

- Ground truth:
  - Majority with Vicon System
  - Minority with Structure-from-Motion
    - E.g. outdoor scenes (Bench)

Figure 1. Scene from our benchmark reconstructed in real-time with ca. 335'000 surfels, with keyframes and estimated trajectory.

Contents

Conclusion: Final Results

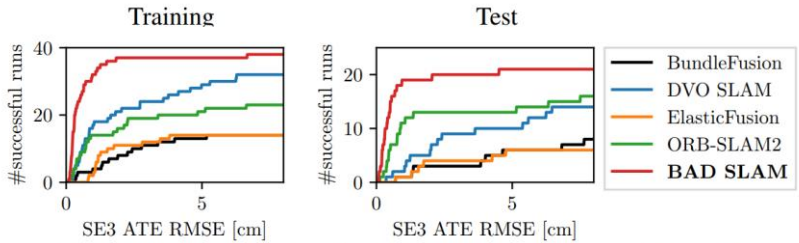


Figure 6. Evaluation on our benchmark's training and test datasets. For a given threshold on the ATE RMSE (x-axis), the graphs show the number of datasets for which the method has a smaller error.

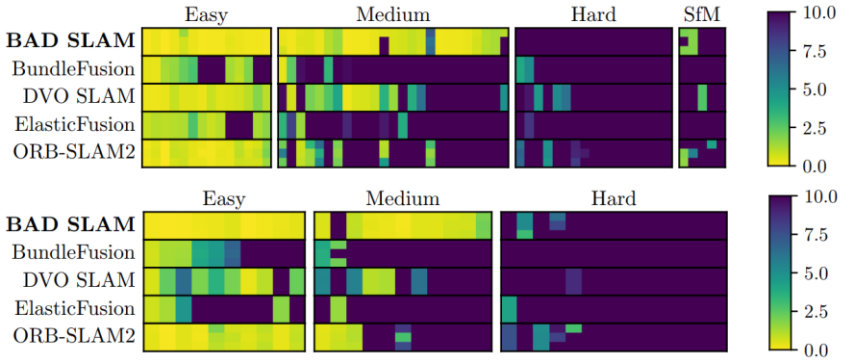


Figure 7. Complete SE(3) ATE RMSE evaluation results on the training (top) and test (bottom) datasets of our benchmark in cm. Each column visualizes the results for one dataset. We show three runs per dataset for non-deterministic methods.

Conclusion: What does BAD-SLAM show us?

1. A new fast direct bundle adjustment formulation
2. Direct methods are negatively affected by rolling shutter, asynchronous RGB and Depth measurements, calibration errors
3. Direct methods may perform significantly better on well calibrated Dataset than popularly chosen indirect methods (ORB-SLAM2)
4. Existing datasets only give partial picture of SLAM algorithm performance

References

[1] Thomas Schops,Torsten Sattler, Marc Pollefeys; BAD SLAM: Bundle Adjusted Direct RGB-D SLAM, In CVPR 2019

[2] Thomas Schops,Torsten Sattler, Marc Pollefeys; Supplementary Document for:
BAD SLAM: Bundle Adjusted Direct RGB-D SLAM, In CVPR 2019

Thank you for your attention! ☺

Cost function – Depth error

$$\sigma_D = \delta \frac{d_m^2}{b\, f} \left| \left(T_g^k \overrightarrow{n_s}\right)^T \pi_k \left(T_g^k \overrightarrow{p_s}\right) \right|$$