This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

BAD SLAM: Bundle Adjusted Direct RGB-D SLAM

Thomas Schöps[1] Torsten Sattler[2] Marc Pollefeys[1,3]
[1]Department of Computer Science, ETH Zürich [2]Chalmers University of Technology [3]Microsoft

Abstract

A key component of Simultaneous Localization and Mapping (SLAM) systems is the joint optimization of the estimated 3D map and camera trajectory. Bundle adjustment (BA) is the gold standard for this. Due to the large number of variables in dense RGB-D SLAM, previous work has focused on approximating BA. In contrast, in this paper we present a novel, fast direct BA formulation which we implement in a real-time dense RGB-D SLAM algorithm.

In addition, we show that direct RGB-D SLAM systems are highly sensitive to rolling shutter, RGB and depth sensor synchronization, and calibration errors. In order to facilitate state-of-the-art research on direct RGB-D SLAM, we propose a novel, well-calibrated benchmark for this task that uses synchronized global shutter RGB and depth cameras. It includes a training set, a test set without public ground truth, and an online evaluation service. We observe that the ranking of methods changes on this dataset compared to existing ones, and our proposed algorithm outperforms all other evaluated SLAM methods. Our benchmark and our open source SLAM algorithm are available at: www.eth3d.net

1. Introduction

SLAM is the problem of simultaneously building a 3D model of a scene and determining the pose of a camera in the scene. This for example enables augmented reality [32] and perception for self-driving cars [64]. Modern SLAM systems consist of two parts [32]: A front-end that tracks the camera pose in real-time, and a back-end that jointly optimizes the 3D map and the previous camera poses. The gold standard for the back-end optimization is *bundle adjustment* (BA) [62]: optimization of all parameters of the reconstructed model and the cameras.

BA algorithms are widely used with sparse features, *e.g.*, [7, 44, 54]. These however discard most of the image information. *Direct* [10, 45, 46, 69] approaches can make use of all data. For a dense reconstruction, this introduces many variables that need to be optimized. Thus, a common assumption is that full BA is infeasible in real-time for direct and dense methods [27]. Instead, approximations are common such as pose graph optimization [11, 30, 34, 60] or deformable geometry [67–69] (*c.f.* Sec. 2). In this paper, we demonstrate that direct alternating BA on dense RGB-D data is feasible in real-time, using a single GPU. We propose a novel, practical BA algorithm, carefully designed to run in real-time in scenes such as the one in Fig. 1.

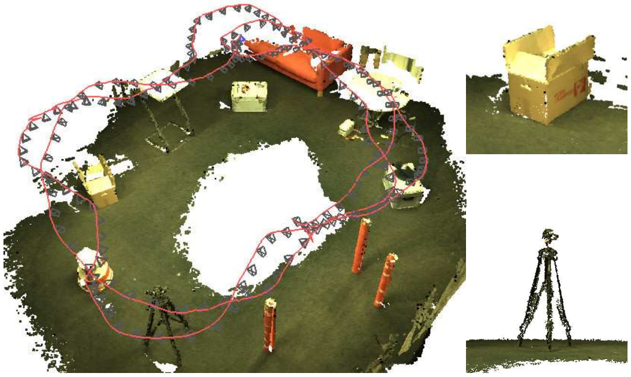

Figure 1. Scene from our benchmark reconstructed in real-time with ca. 335’000 surfels, with keyframes and estimated trajectory.

Experiments on the TUM RGB-D benchmark [61] show the benefit of direct BA over approximate methods. However, the feature-based ORB-SLAM2 [44] outperforms all direct methods. This is likely caused by effects such as rolling shutter [10], asynchronous RGB and depth frames, and depth distortion. We believe that such problems are better solved in hardware than in software. For example, handling rolling shutter is computationally complex [53, 56] and introduces degenerate cases [1]. We thus introduce a novel benchmark for RGB-D SLAM that was recorded with accurately calibrated, synchronized global shutter cameras. On this dataset, ORB-SLAM2 is outperformed by our direct BA approach, while DVO-SLAM [30] performs on a similar level as ORB-SLAM2. Our benchmark is available online with public leaderboards for a training and a test set.

In summary, our contributions are: i) A novel, fast algorithm for direct BA, implemented in an RGB-D SLAM system which outperforms existing approaches. ii) A well-calibrated benchmark dataset for RGB-D SLAM, but also monocular and stereo visual-inertial SLAM. In contrast to previous datasets, ours uses synchronized global shutter cameras, removing the need to model effects such as rolling shutter. We show that the ranking of RGB-D SLAM methods changes on this dataset compared to existing datasets. iii) A leaderboard for our benchmark, and our SLAM system as open source, available at www.eth3d.net.

2. Related Work

The main technical contribution of our work is a real-time capable direct BA strategy for RGB-D data. This section thus focuses on optimization strategies used for SLAM. We also shortly discuss benchmarks for RGB-D SLAM.

Frame-to-model tracking algorithms track the motion of the camera with respect to the 3D model built so far. Examples include voxel-based KinectFusion-style approaches [28, 45, 47], similar methods based on surfel representations [29, 35, 36], as well as some keyframe-based techniques [39–43]. However, the camera poses are never refined, leading to drift in the estimated trajectory over time.

Map deformation can be done explicitly when a loop is detected. For example, Kintinuous [68] deforms the reconstructed mesh for loop closing. Similar approaches are used by [67] and ElasticFusion [69]. However, while such a deformation can approximately improve the model, it does not accurately take into account all available information.

Pose graph optimization methods [9, 11, 30, 34, 60, 65] build a graph of simplified constraints on the relative transformation between camera pairs. This is then used to refine the camera poses. This reduces the size of the optimization problem, but also only approximates BA.

Fragment-based optimization fuses together chunks of frames into fragments and then optimizes the global (rigid) fragment alignment [6,15,27,49]. This enables efficient optimization of the map and cameras due to the low number of variables [27]. Yet, the intra-fragment structure and poses are typically fixed, *i.e.*, only approximate BA is performed.

Indirect (feature-based) BA. Typically, SLAM methods perform indirect BA on a sparse set of keypoints only, optimizing the reprojection error of the reconstructed points. BundleFusion [7] performs global SIFT [38] keypoint matching in real-time. Direct image alignment is used within small chunks of frames, and as a post-processing step on the whole reconstruction. [16, 17] combine direct localization of image patches with indirect BA optimizing the reprojection error. Henry *et al.* [25] use depth to extend the keypoint reprojection error. A similar formulation is also used by ORB-SLAM2 [44]. It significantly outperforms all current direct RGB-D SLAM systems on existing datasets as it is less sensitive to unmodeled geometric distortions. However, our experiments show that our dense BA strategy outperforms ORB-SLAM2 on well-calibrated data.

Direct BA optimizes the camera and scene parameters by minimizing a photometric error. Delaunoy & Pollefeys [8] propose an offline algorithm with a mesh-based object representation. This requires frequent remeshing, contributing to a high runtime. In another offline algorithm, Goldlücke *et al.* [23] optimize for (super-resolved) appearance and camera calibration, while leaving the initial geometry constant aside from creating a displacement map.

SDF-2-SDF [59] performs pairwise alignment of signed distance functions (SDFs). A final offline BA step refines keyframe poses and a fused reconstructed SDF. In contrast to our method, SDF-2-SDF considers geometry only. Furthermore, the discretization into voxels limits its accuracy.

Yan *et al.* [70] use surfels as map representation. They propose an optimization scheme similar to direct BA, which however combines pose graph optimization with some direct structure and pose updates only.

The geometric term of our direct BA approach is similar to multiview ICP [14]. However, instead of matching frames to a global reconstruction, multiview ICP builds on pairwise matching, requiring to take all overlapping pairs into account for using all information. Furthermore, multiview ICP does not refine the geometry.

Both Alismail *et al.* [2] and DSO [10] propose direct BA approaches for visual odometry. Only few keyframes which are temporally close to the current frame and the 3D points visible in them are optimized. DSO has been extended with loop closure handling, but only using pose graph optimization without BA [20]. As one key difference to these works, we propose a direct BA approach for global consistency in SLAM, instead of restricting it to odometry only. Our approach runs in real-time, which is made possible by alternating between optimizing poses and geometry to limit the size of the individual optimization problems.

Benchmarking RGB-D SLAM. Tab. 1 shows a comparison of our new benchmark with existing datasets. Ours particularly stands out in providing real RGB-D data with synchronized global-shutter cameras. In contrast to other real-world RGB-D datasets, ours also provides full IMU data. Furthermore, to our knowledge, ours is the first real-world RGB-D SLAM benchmark with a public online leaderboard on a test set for which ground truth is withheld. Aside from RGB-D SLAM, (visual-inertial) monocular and stereo SLAM methods can also be evaluated on our benchmark. We provide online leaderboards for these tasks as well.

3. Direct RGB-D Bundle Adjustment

As is common for SLAM algorithms, our method consists of a front-end and a back-end (*c.f.* Fig. 2). The front-end tracks the motion of the RGB-D camera in real-time. It thus provides initial estimates for camera poses and scene geometry. The back-end, running at a lower frequency [32], then refines both the camera trajectory and the geometry to build a consistent 3D map. The core technical contribution

Method	Real data	RGB-D	Stereo	Global shutter	Sync'ed cameras	IMU	Accurate GT	Geometry GT	Benchmark
TUM RGB-D [61]	X	X				(1)	X		(2)
TUM VI [57]	X		X	X	X	X	X		
TUM Mono [12]	X			X	-				
CoRBS [66]	X	X					X	X	
ICL-NUIM [24]		X		X	X		X	(3)	
VIPER Odometry [51]				X	-		X		X
InteriorNet [37]		X	X	X	X	X	X	X	
KITTI Odometry [21]	X	(4)	X	X	X	X			X
EuRoC MAV [3]	X		X	X	X	X	X	(5)	
Ours	X	X	X	X	X	X	(6)		X

Table 1. Comparison of selected existing datasets with our new benchmark. Notes on numbered entries: (1) Accelerometer but not gyroscope measurements are available. (2) While this dataset has a test set, it is not well suited for benchmarking since it shows the same scenes as the training set, and there is no online leaderboard. (3) Available in an extended version of the dataset [6]. (4) Sparse measurements of a spinning laser scanner are available. (5) Structure ground truth is available for some of the sequences. (6) A motion capturing system is used for all but a few training datasets, for which GT is obtained using Structure-from-Motion (*c.f.* Sec. 5).

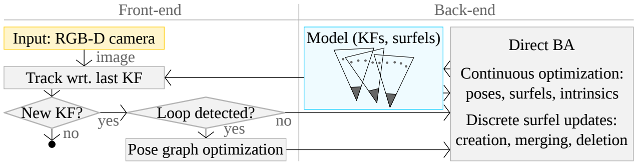

Figure 2. Approach overview. KF stands for keyframe.

of this paper is a novel Bundle Adjustment (BA) strategy for direct RGB-D SLAM, used in the back-end. In the following, we describe this strategy in detail. The front-end, which follows common practice, is briefly discussed in Sec. 4.

Our BA strategy is based on a set of key concepts: We use both geometric constraints, based on the recorded depth images, and photometric constraints. The latter use gradients rather than raw pixel intensities to be robust to photometric changes. To enable efficient optimization, our approach alternates between refining the 3D map and the camera poses to minimize the number of parameters considered at each point in time (*c.f.* Alg. 1). As a result, ours is the first dense BA approach for RGB-D SLAM that runs in real-time for smaller scenes. In the following, we first describe our data representation before detailing the cost function we optimize and the optimization procedure itself.

3.1. Data Representation

We represent the scene geometry with dense *surfels* and use *keyframes* to reduce the amount of input data for BA.

A *keyframe* is defined by an RGB-D frame and its 6DOF camera pose. We use a shared intrinsic calibration for all keyframes, where we separately model the color and depth sensors as pinhole cameras. Optionally, we also model depth image deformations, which is important to improve the depth quality of typical consumer depth cameras [72].

A *surfel* s in our method is an oriented disc defined by a 3D center point $\mathbf{p}_s$, a surface normal vector $\mathbf{n}_s$, a radius r_s, and a scalar visual descriptor d_s. We chose surfels as a scene representation as they can be efficiently fused and updated by BA, and can be quickly deformed to adapt to loop closures [69]. In contrast to voxel-based scene representations (*e.g.*, [45]), surfels can easily represent thin surfaces and scene detail of arbitrary scale. In contrast to meshes [8], no expensive topology updating is required during BA.

Naively optimizing 3D surfel positions during BA can lead to problems. RGB-D sensors provide measurements in weakly textured regions with little variation in geometry, *e.g.*, white walls. In these regions, neither photometric nor geometric descriptors constrain all degrees of freedom, allowing some surfels to arbitrarily move within a surface.

To prevent potential artefacts, *e.g.*, holes, we thus restrict the movement of the surfels during optimization. [10] proposes to fix a surfel's pixel position in its keyframe and to only optimize its (inverse) depth. In practice, we encountered problems when using this parameterization in combination with the point-to-plane residuals typically used for RGB-D SLAM [73]. For silhouettes and object boundaries in particular, the parameterization leads to intersecting the plane and a nearly parallel viewing ray, which is ill-conditioned. Correspondingly, we observed that the optimization moved some surfels to wrong locations.

Our solution is moving surfels along the surfel normal directions only, and only using measurements having similar normals to optimize them. This provides a sensible direction of optimization while avoiding ill-defined residuals. As position and normal optimization is alternated, surfels are not restricted to move along their initial normal only.

3.2. Cost Function

Our goal is to optimize the parameters of the model defined in Sec. 3.1 to maximize consistency. These parameters are the surfel attributes, keyframe poses, and optionally camera intrinsics, while measurements are the (mostly) raw RGB-D input data stored in the keyframes. The cost could be measured per element of the model (*i.e.*, surfel), or of the input (*i.e.*, pixel). Since one pixel can correspond to multiple surfels, a non-diagonal Hessian matrix may result in the latter case during optimization. For efficiency reasons, we thus use the model-based approach in this work.

The cost is thus computed by projecting each surfel into each keyframe to establish correspondences with pixel measurements. Geometric and photometric residuals measure how well the surfel models the depth and color at its corresponding image positions. The goal of the optimization is to maximize photometric and geometric consistency.

Mathematically, we define the cost as follows. Let K be the set of all keyframes and let S_k be the set of all surfels that have a corresponding measurement in keyframe $k \in K$. To robustly handle outliers, the geometric and photometric residuals r_{geom} and r_{photo} are weighted with Tukey's biweight and the Huber robust loss function, respectively, with weighting parameter 10 for both functions. The photometric residuals are furthermore weighted by $w_{\text{photo}} = 10^{-2}$ to prefer depth information. The resulting cost function is a sum of geometric and photometric terms:

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

Here, σ_D and σ_p are standard deviations of the geometric and photometric measurements. They are used to normalize the individual residuals, which we explain in the following.

Geometric residuals. Point-to-plane data associations are commonly used for RGB-D SLAM [73] and are known to work well for ICP [5]. We thus define r_{geom} analogously:

$$r_{\text{geom}}(s,k) = \left(\mathbf{T}_G^k \mathbf{n}_s \right)^T \left(\pi_{D,k}^{-1}(\hat{\pi}_{D,k}(\mathbf{T}_G^k \mathbf{p}_s)) - \mathbf{T}_G^k \mathbf{p}_s \right) \quad (2)$$

Here, $\hat{\pi}_{D,k}$ maps a 3D point in the local coordinate system of keyframe k to the center of the corresponding depth image pixel. In return, $\pi_{D,k}^{-1}$ maps a depth pixel to its 3D point in the keyframe's local coordinate system via its measured depth d_m. The transformation $\mathbf{T}_G^k$ maps the position $\mathbf{p}_s$ and normal $\mathbf{n}_s$ of a surfel s from the global map coordinate system G to the local coordinate system of keyframe k. Intuitively, Eq. 2 computes the distance between the surfel and measurement positions along the surfel's normal direction.

The resulting residual is normalized using an estimate σ_D of its standard deviation. To derive this, we apply uncertainty propagation of the measurement uncertainties. For a well-calibrated camera, it is reasonable to assume that the measurement uncertainty is purely in the depth direction. Many depth cameras use stereo matching on infrared images to compute depth maps. In this case, the depth error grows quadratically with the depth d_m and can be shown to be $\delta d_m^2 (bf)^{-1}$ [18], which we use as a model for the standard deviation σ_{d_m}. Here, b is the baseline in meters, *i.e.*, the camera-camera distance for active stereo, or the camera-projector distance for structured light. f is the focal length in pixels, and δ is the expected stereo matching error in pixels, which we set to 0.1. Applying uncertainty propagation (in this case: $\sigma_D^2 = (\frac{\partial r_{\text{geom}}}{\partial d_m})^2 \sigma_{d_m}^2$) then leads to:

$$\sigma_D = \delta \frac{d_m^2}{bf} \left| \left(\mathbf{T}_G^k \mathbf{n}_s \right)^T \pi_k(\mathbf{T}_G^k \mathbf{p}_s) \right| \ . \quad (3)$$

The depth uncertainty depends on the sensor type; a different model should be used for *e.g.*, time-of-flight sensors.

Photometric residuals. For r_{photo}, we compare the surfel's descriptor d_s to the surfel's projection into the image. The descriptor should be robust against photometric changes while offering a wide basin of convergence and enabling efficient optimization. We use simple intensity gradient magnitudes, calculated in a geometrically consistent way. This approach is similar in principle to the pose refinement in [55]. However, [55] uses several points to compute one residual, which prevents fast geometry optimization since the resulting Hessian for surfel optimization is not diagonal. Instead, for one residual we sample the image at the projections of the surfel center $\mathbf{p}_s$ and of two fixed points $\mathbf{s}_1$, $\mathbf{s}_2$ on the boundary of the same surfel's disc. The surfel disc radius r_s resembles the pixel sampling of the highest-resolution image observing the surfel, *c.f.* the radius update in Sec. 3.3. $\mathbf{s}_1$ and $\mathbf{s}_2$ are chosen such that the directions $\mathbf{s}_1 - \mathbf{p}_s$ and $\mathbf{s}_2 - \mathbf{p}_s$ are orthogonal. Then the gradient magnitude is computed and compared to d_s to obtain:

$$r_{\text{photo}}(s,k) = \left\| \begin{pmatrix} I(\pi_{I,k}(\mathbf{s}_1)) - I(\pi_{I,k}(\mathbf{p}_s)) \\ I(\pi_{I,k}(\mathbf{s}_2)) - I(\pi_{I,k}(\mathbf{p}_s)) \end{pmatrix} \right\|_2 - d_s \ . \quad (4)$$

Here, $\pi_{I,k}$ is the projection function for the RGB image and $I(\cdot)$ the bilinearly interpolated image intensity.

The uncertainty σ_p (*c.f.* Eq. 1) of the photometric measurements depends on many factors. Some of them, *e.g.*, reflections or illumination changes during the capture process, are hard to model mathematically. For simplicity, we thus empirically set σ_p to $\frac{1}{180}$ (for intensities in $[0,1]$).

Surfel-measurement correspondence estimation. Eq. 1 compares the surfels in the map to their corresponding RGB-D measurements in the keyframes. We establish these correspondences by projecting the surfel centers into the keyframes. To filter outliers, we only establish a correspondence between a surfel s and its pixel projection in keyframe k if: i) $\mathbf{p}_s$ projects to a pixel that has a depth measurement. In practice this also discards observations at very oblique angles, since depth cameras will usually discard measurements there. ii) The depth of the measurement and the projected surfel is similar enough. Similarity is measured via the weight of the depth residual in the optimization, *i.e.*, we only establish a correspondence if $\frac{1}{r} \frac{\partial \rho_{\text{Tukey}}(r)}{\partial r} > 0$ with $r := r_{\text{geom}}(s,k)$. iii) The surfel normal is similar to the depth measurement's normal (estimated by finite differences in the depth image) and points towards the camera.

3.3. Optimization

We perform BA by optimizing for the cost in Eq. 1. Similar to sparse SfM [54], where additional points are triangulated and existing points are merged after BA, we interleave the cost optimization with several discrete surfel update steps. Changing keyframe poses can otherwise result in scene parts not being covered by surfels.

Both the numbers of surfels and keyframes quickly become large. Jointly optimizing all parameters using a second-order method such as Gauss-Newton thus quickly becomes slow, even when using the Schur complement. We thus use alternating optimization, which can still perform competitively for strongly connected problems [62].

Our optimization scheme, stated in Alg. 1, performs a number of iterations up to a maximum or until convergence. Within each iteration, alternating steps optimize the cost in Eq. 1 and update the surfels. Each step is detailed below.

Algorithm 1 Surfel-based alternating direct BA scheme

```
for all keyframes do Create missing surfels
for $i \in [1, \text{max\_iteration\_count}]$ do
    Update surfel normals
    Optimize surfel positions and descriptors
    if i = 1 then Merge similar surfels
    Optimize keyframe poses
    Optimize camera intrinsics (optionally)
    if no keyframe moved then break
for all keyframes which moved in the last loop do
    Merge similar surfels
Delete outlier surfels; Update surfel radii
```

Surfel creation. In a first step, we attempt to create new surfels for all keyframes. We partition the keyframes into 4×4 pixel cells. If no pixel in a cell corresponds to an existing surfel, we randomly choose one depth measurement within the cell to create a new surfel s. Its attributes are computed from the pixel p it is created from: $\mathbf{p}_s$ is set to $\mathbf{T}_k^G \pi_{D,k}^{-1}(p)$. $\mathbf{n}_s$ is computed via centered finite differences on the depth image. r_s is defined as the minimum distance between $\mathbf{p}_s$ and the 3D points of the 4-neighborhood of p. d_s is initialized to the first term in Eq. 4. Only pixels for which all neighboring pixels have a depth measurement are considered when creating new surfels.

Further, we only use pixels that pass the following outlier filter: We project the pixel's 3D point into each other keyframe. We count the number of times n_C the point projects to a corresponding measurement, and the number of free-space violations n_V, *i.e.*, how often the point lies in front of a keyframe's depth map. A pixel is considered an outlier if $n_C < n_\text{min}$ or $n_V > n_C$. n_min is set to $\min(3, 1 + \lfloor 0.2 |K| \rfloor)$, avoiding optimization on incomplete surfel models while the number of keyframes $|K|$ is small.

Surfel normal updates. For efficiency reasons, we treat surfel normals (which mainly only impact the directions in which surfels are allowed to move) as auxiliary variables. Instead of deriving an update step from the cost function (which may require adding a normal residual in addition to r_geom and r_photo), we thus use an update which is designed to be efficient: We average the normals of all corresponding measurements, followed by re-normalization to unit length.

Surfel position and descriptor optimization. After updating normals $\mathbf{n}_s$, surfel positions $\mathbf{p}_s$ and descriptors d_s are jointly optimized by applying a Gauss-Newton iteration to Eq. 1. We only allow surfels to move along their normal directions. A surfel position is thus parametrized as $\mathbf{p}_s + t \cdot \mathbf{n}_s$ and we optimize for t. Joint optimization of positions and descriptors is very helpful for improved convergence speed. Since different surfels are independent, this only involves solving a 2×2 matrix for each surfel.

Surfel merging. Surfel creation often generates unnecessary surfels for noisy measurements, which get denoised by their first position optimization. Thus, after position optimization in the first iteration of the BA scheme, we merge surfels with similar attributes. Two surfels s_1 and s_2 are merged if their normals are within $40°$ of each other, and their positions are closer than $4 \cdot 0.8 \cdot \min(r_{s_1}, r_{s_2})$. The factor 4 corresponds to the cell size for surfel creation and 0.8 is the merge threshold. For finding merge candidates quickly, we project the surfels into all keyframes and consider surfels projecting to the same cell for merging.

Keyframe pose optimization. We optimize the poses of all keyframes by applying the Gauss-Newton method to Eq. 1. Pose updates ϵ are parametrized as local updates in the Lie algebra $\mathfrak{se}(3)$. Thus, the transformation $\mathbf{T}_G^k$ from the global coordinates to the local coordinates of keyframe k is updated as $\mathbf{T}_k^g \cdot \exp(\hat{\epsilon})$. Local updates ensure that rotation updates are well-defined [13]. Since keyframes are independent, this results in a standard direct pose refinement for each keyframe (similar to *e.g.* [29, 30, 41]).

Camera intrinsics optimization. If the RGB-D camera is not accurately calibrated, we can optionally optimize the camera intrinsics. We only do this for evaluation on existing datasets; our new dataset does not require this step. Again, we use the Gauss-Newton method to minimize Eq. 1, now optimizing the intrinsic parameters while fixing all others. We use separate pinhole models for the color and depth camera and also model depth deformation. We use the depth offset model from [26], which relates true inverse depths d_true to distorted inverse depths d_dist for every pixel (x, y) as

$$d_\text{true}(x, y) = d_\text{dist}(x, y) + D_\delta(x, y) \cdot \mathrm{e}^{\alpha_0 - \alpha_1 d_\text{dist}(x, y)} \quad . \tag{5}$$

We drop α_0 as changing it has the same effect as changing D_δ for all pixels correspondingly. The remaining parameters are α_1 and a parameter image D_δ. The part of the Hessian corresponding to D_δ is a diagonal matrix. The Schur complement can thus be used to very efficiently solve this matrix despite its large size. We estimate D_δ at a quarter of the image resolution, corresponding to the surfel creation cell size, and use nearest-neighbor access to keep the Hessian diagonal. This approach allows us to quickly optimize for camera intrinsics and especially depth deformation.

Surfel cleanup and radius update. As is common in sparse SfM [54], we filter outlier surfels based on the same criteria used to detect outliers during surfel creation. In addition, the radius of each surfel is updated to the minimum radius of all its corresponding measurements. Intuitively, this corresponds to using the highest resolution under which the point is observed when computing its descriptor.

4. RGB-D SLAM Front-End

The main technical contribution of this paper is the dense direct BA approach presented in the previous section, which forms the back-end of our RGB-D SLAM approach. The following describes our front-end responsible for real-time camera pose tracking and loop closure detection.

Preprocessing. As is common, a bilateral filter is used to smooth the depth map, and large depth measurements are removed. The filter parameters depend on the camera used.

Odometry. Once a new RGB-D frame becomes available, we first estimate its pose relative to the last keyframe via standard direct photometric and geometric image alignment in SE(3) [29, 30, 41]. For robustness against illumination changes, we use intensity gradients rather than pixel intensities for tracking. The odometry's purpose is to provide good initial keyframe poses and it can be exchanged easily.

Interaction with BA. We do not address keyframe selection in this work and thus simply select every 10th frame as a keyframe. After a new keyframe is created, we test for loop closures with previous parts of the trajectory, as detailed below, before passing it to the BA back-end. If a new keyframe is created before these iterations finish, we skip the remaining ones to keep up real-time operation.

Loop closure detection. We use a standard bag-of-words approach [19] based on binary features [4] to identify the keyframe m most similar to the latest keyframe k. We get an initial estimate of the keyframes' relative pose from the resulting keypoint matches. This pose is then refined using direct alignment. We also use direct alignment to align keyframe k to the keyframes $m - 1$ and $m + 1$. If these estimates are sufficiently consistent based on thresholding their translation and angle differences, we accept the loop closure. We use the averaged relative pose in a pose graph optimization step to obtain an initial correction for the trajectory, followed by applying our BA strategy.

It should be noted that as a general limitation of direct image alignment and thus direct BA, the convergence region is small [50]. Thus it is for example conceivable that a pose graph optimization step upon a loop detection pushes old keyframes out of their convergence region. However, we did not observe this in our final system. If it became an issue, we think that future work might try to couple direct BA with a well-converging but inaccurate method. This way, it could stay close to the solution of the other method and is thus likely to converge, while acting as a refinement.

5. Benchmark Dataset

Motivation. As motivation for recording a new RGB-D SLAM benchmark, we discuss results on the popular TUM RGB-D dataset [61]. Tab. 2 shows absolute trajectory error (ATE) results (as used in [61]; smaller is better) for different SLAM methods on some commonly used sequences. As a first observation, our optional intrinsics and depth distortion optimization strongly improves our results on these datasets. This shows that the depth camera's inaccurate (internal) calibration affects SLAM methods and should be calibrated for good results. Furthermore, we observe that ORB-SLAM2 [44] as an indirect method clearly outperforms all direct methods on these datasets, including ours, which shares the second average rank with BundleFusion [7]. As part of the reason for this, [10] shows that ORB-SLAM2 is less affected by rolling shutter, as exhibited by the camera used in this dataset, than direct methods. Further, the camera's depth and color streams are not synchronized. These effects, together with the depth distortion observed above, introduce further geometric distortions which may strongly affect direct methods. We aim to evaluate the effect of distortions in the following.

	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.

	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.

Impact of distortions. While the geometric distortions in the cameras may be hard to model accurately, we can isolate the effect of asynchronous RGB-D frames and rolling shutter in synthetic datasets. We create such datasets by making dense 3D reconstructions of scenes from the TUM RGB-D dataset [61] and rendering them with their original (continuously interpolated) trajectory. We render each dataset in four variations: a 'clean' variant and variants with rolling shutter and asynchronous frames, both individually and combined. For rolling shutter of both the color and depth camera, we use the estimated shutter times (time offset between first and last scanline readout) for the Kinect v1 from [52]: ca. 30.5 ms for the depth camera and ca. 26.1

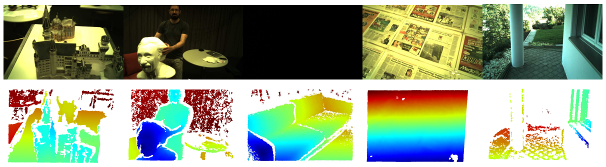

Figure 3. Example images (top) and depth maps (bottom) from our dataset, showing some of its variety including still scenes, moving objects, darkness, a planar scene requiring photometric tracking, and outdoor scenes. No white balancing is used.

ms for the color camera. For asynchronous frames, we use the worst case in the sense that a color image is rendered at the middle point in time between two successive depth images. All other offsets would yield depth/color pairs which are closer in time. Results are shown in Tab. 3. In contrast to the TUM RGB-D results, our method outperforms all others on these datasets, suggesting that more effects, such as depth distortion, need to be simulated for obtaining realistic results. However, we can still make the observation that both evaluated effects significantly degrade the SLAM results when not modeled. Since modeling all of these effects in a direct SLAM system is laborious and may lead to high runtimes [53, 56] and degeneracies [1], we think that they are better addressed in hardware. Thus, we recorded a novel RGB-D SLAM benchmark using better hardware.

Our benchmark. For data recording, we mounted synchronized global shutter cameras [22] together with an Asus Xtion Live Pro, using the Xtion's infrared emitter together with our cameras. Color and depth images are recorded at exactly the same points in time, such that no temporal smoothness assumption is needed to use both for estimating one camera pose. Like *e.g.* [31, 33, 48, 71], we use active stereo: By doing stereo depth estimation on two IR cameras, the stereo algorithm takes advantage of both the active illumination and ambient infrared light. Ground truth poses are mostly recorded by a motion capturing system. A few of the training datasets are recorded outside of this system for more diversity. Ground truth for those is given by Structure-from-Motion on the benchmark cameras as well as additional cameras on the rig, on videos that cover the dataset sequence multiple times. These datasets are kept in a separate category since they are likely to be less accurate.

Our SLAM benchmark consists of 61 training and 35 test datasets. Fig. 3 shows example images from our datasets. All sequences are shown in the supplementary video.

Currently, it is common practice in RGB-D SLAM evaluation to subselect (and thus potentially cherry-pick) a small number of datasets, *e.g.*, from the *training* set of the TUM RGB-D dataset [61] (see *e.g.*, [7, 58, 69]). Since these datasets come with ground truth, it is furthermore unclear to what extent the methods overfit on them. By providing a benchmark with non-public ground truth and an online leaderboard (analogous to [21]), we hope to improve this.

In contrast to datasets with typical consumer depth cameras our camera is well-calibrated, which includes the calibration used internally for depth estimation. We refer to the supplementary material for additional information and experiments on the dataset's recording and calibration.

6. Evaluation

Test environment. We used a PC with an Intel Core i7 6700K and an MSI Geforce GTX 1080 Gaming X 8G. Our BA scheme was implemented on the GPU using CUDA 8.0.

For quantitative evaluation, we focus on the absolute trajectory RMSE with SE(3) alignment (SE(3) ATE RMSE, *c.f*. [61]), since this focuses on the SLAM (instead of odometry) performance. Additional results for other metrics are provided in the supplementary material. SE(3) ATE RMSE is computed by first aligning the estimated trajectory to the ground truth with a transformation in SE(3) by matching poses with the same timestamp and applying the Umeyama method [63]. Then, the RMSE of the translational differences between all matched poses is computed.

Ablation study. We evaluate the contributions of different components of our algorithm to the final results in Fig. 5 (top). Clearly, the geometric residuals r_{geom} strongly help, since the results for using photometric residuals r_{photo} only are much worse. In addition, the plots show that BA clearly improves upon running only the front-end of our method. While the remaining differences are smaller, it can be observed that using both types of residuals (”Default”) performs better than using only depth residuals. Furthermore, we evaluate an offline (but close to real-time) variant of our algorithm for which we never skip BA iterations and perform 25 BA iterations after dataset playback finished. As expected, this performs slightly better than the real-time settings. Finally, we also evaluate a preconditioned conjugate gradient (PCG) solver on the Gauss-Newton update equation in place of our alternating optimization scheme. It performs very similar, but slightly worse than the alternating optimization. Since individual iterations take longer, more BA iterations are skipped in this variant (*c.f*. Sec. 4).

Parameter values. We evaluate the keyframe creation frequency in Fig. 5 (bottom left), where the number of frames per keyframe is stated for each graph. In this evaluation, selecting keyframes more frequently always performs better than selecting less keyframes, despite possibly reducing the number of BA iterations in the real-time setting. However, the memory use also rises linearly with more keyframes.

The cell size for surfel creation is evaluated in Fig. 5 (bottom right). A smaller cell size yields denser scene reconstructions and thus higher memory use. It can improve accuracy since more geometry is considered, but also increases computational demands. This can slow down convergence of BA. According to the plotted results, in prac-

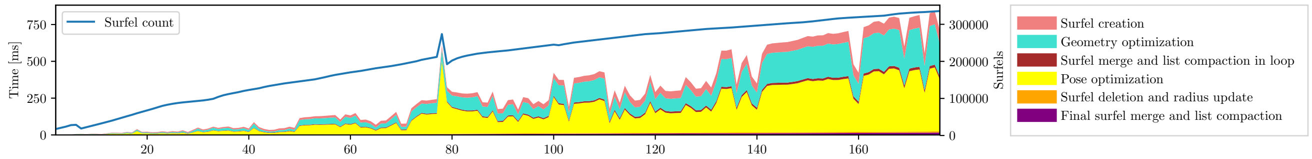

Figure 4. Runtime of our BA scheme in ms for the dataset shown in Fig. 1 (without skipping any BA iterations). The number of keyframes is shown on the x-axis. Since we create one keyframe every 10 frames for ∼27 Hz input, 370 ms of processing time are available for each keyframe; if BA takes longer, iterations are skipped in real-time mode. The spike in the surfel count corresponds to a loop closure.

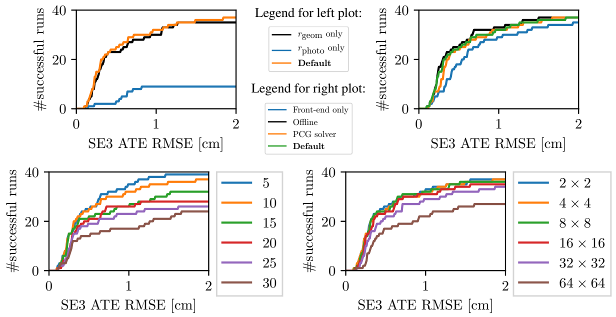

Figure 5. Ablation study (top row), keyframe interval evaluation (bottom left), and surfel sparsity evaluation (bottom right). In each plot, for a given threshold on the ATE RMSE (x-axis), the graphs show the number of *training* datasets from our benchmark for which the evaluated variant has a smaller error.

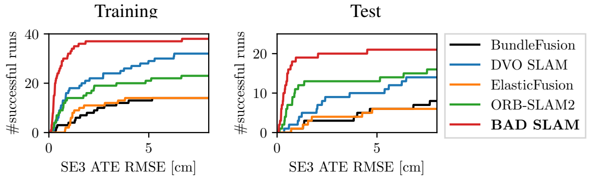

Figure 6. Evaluation on our benchmark's training and test datasets. For a given threshold on the ATE RMSE (x-axis), the graphs show the number of datasets for which the method has a smaller error.

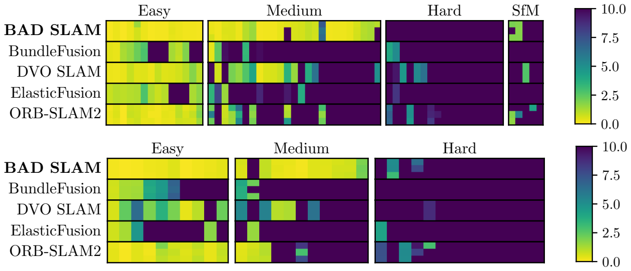

Figure 7. Complete SE(3) ATE RMSE evaluation results on the training (top) and test (bottom) datasets of our benchmark in cm. Each column visualizes the results for one dataset. We show three runs per dataset for non-deterministic methods.

tice, varying the cell size between 2 × 2 and 8 × 8 shows little effect overall. The results degrade for larger cell sizes.

Evaluation on the proposed benchmark. We compare against state-of-the-art (non-inertial) RGB-D SLAM methods for which source code is available. For all methods, parameters are tuned on the training datasets only. Fig. 6 shows cumulative results, while Fig. 7 visualizes individual results on all datasets. We qualitatively classified the datasets as easy, medium, or hard: Easy datasets are solved by most algorithms, while hard datasets are not solved well by any algorithm. The remaining ones are of medium difficulty. In addition, the 'SfM' category for training datasets contains datasets with SfM ground truth (*c.f*. Sec. 5). Our method significantly outperforms the others, yielding superior results on medium datasets and being very reliable on easy datasets. While no direct algorithm was able to beat ORB-SLAM2 on the TUM RGB-D dataset, here it is outperformed by our method. DVO SLAM performs better than ORB-SLAM2 on the training set, but worse on the test set. The hard datasets provide open challenges for future work. Common reasons for failure include textureless scenes with ambiguous structure, fast camera motion, and moving objects. Thus, while we believe that our algorithm can obtain very accurate results, it would still require for example the use of an IMU or of larger field of view to increase the whole system's robustness in these cases. It should also be noted that scalability is not considered in the evaluation above; some methods do not require a GPU.

Runtime. Fig. 4 shows the runtime of the different parts of our BA scheme on an example dataset of our benchmark. The time used by odometry is negligible. Keyframe pose and geometry optimization take up the most time.

7. Conclusion

We presented a novel RGB-D SLAM method with a real-time direct BA back-end using surfels. This allows to use rich information during global optimization, resulting in very accurate trajectories. To avoid unmodeled geometric distortion, we present an RGB-D SLAM benchmark with synchronized global shutter cameras. On this benchmark, direct methods like ours and DVO SLAM [30] perform significantly better compared to the indirect ORB-SLAM2 [44] than on existing datasets. We believe this to be very interesting for the community since it shows that existing datasets only give a partial picture of the SLAM algorithms' performance. The benchmark also contains hard sequences as open challenges for (visual-only) RGB-D SLAM. For example, using silhouettes or initializing depth measurements from RGB images might help solving them. As future work, one might also apply standard techniques such as windowed BA to keep up real-time BA for longer sequences.

Acknowledgements. Thomas Schöps was supported by a Google PhD Fellowship.

References

[1] Cenek Albl, Akihiro Sugimoto, and Tomas Pajdla. Degeneracies in Rolling Shutter SfM. In *ECCV*, 2016. 1, 7

[2] Hatem Alismail, Brett Browning, and Simon Lucey. Photometric bundle adjustment for vision-based SLAM. In *ACCV*, 2016. 2

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *IJRR*, 2016. 3

[4] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In *ECCV*, 2010. 6

[5] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 4

[6] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 2, 3

[7] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics (TOG)*, 36(3):24, 2017. 1, 2, 6, 7

[8] Amaël Delaunoy and Marc Pollefeys. Photometric bundle adjustment for dense multi-view 3D modeling. In *CVPR*, 2014. 2, 3

[9] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, 2012. 2, 6

[10] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *PAMI*, 40(3):611–625, 2018. 1, 2, 3, 6

[11] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014. 1, 2

[12] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*, 2016. 3

[13] Chris Engels, Henrik Stewénius, and David Nistér. Bundle Adjustment Rules. In *Photogrammetric Computer Vision*, 2006. 5

[14] Simone Fantoni, Umberto Castellani, and Andrea Fusiello. Accurate and automatic alignment of range surfaces. In *3DIMPVT*, 2012. 2

[15] Nicola Fioraio, Jonathan Taylor, Andrew Fitzgibbon, Luigi Di Stefano, and Shahram Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. In *CVPR*, 2015. 2

[16] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017. 2

[17] Yasutaka Furukawa and Jean Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *IJCV*, 84(3):257–268, 2009. 2

[18] David Gallup, Jan-Michael Frahm, Philippos Mordohai, and Marc Pollefeys. Variable Baseline/Resolution Stereo. In *CVPR*, 2008. 4

[19] Dorian Gálvez-López and Juan D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012. 6

[20] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. LDSO: Direct sparse odometry with loop closure. In *IROS*, 2018. 2

[21] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 32(11):1231–1237, 2013. 3, 7

[22] Pascal Gohl, Dominik Honegger, Sammy Omari, Markus Achtelik, Marc Pollefeys, and Roland Siegwart. Omnidirectional visual obstacle detection using embedded FPGA. In *IROS*, 2015. 7

[23] Bastian Goldlücke, Mathieu Aubry, Kalin Kolev, and Daniel Cremers. A super-resolution framework for high-accuracy multiview reconstruction. *IJCV*, 106(2):172–191, 2014. 2

[24] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *ICRA*, 2014. 3

[25] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *IJRR*, 31(5):647–663, 2012. 2

[26] Daniel Herrera, Juho Kannala, and Janne Heikkilä. Joint depth and color camera calibration with distortion correction. *PAMI*, 34(10):2058–2064, 2012. 5

[27] Olaf Kähler, Victor A. Prisacariu, and David W. Murray. Real-time large-scale dense 3D reconstruction with loop closure. In *ECCV*, 2016. 1, 2

[28] Olaf Kähler, Victor A. Prisacariu, Julien Valentin, and David Murray. Hierarchical voxel block hashing for efficient integration of depth images. *RA-L*, 1(1):192 – 197, 2016. 2

[29] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *3DV*, 2013. 2, 5, 6

[30] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual SLAM for RGB-D cameras. In *IROS*, 2013. 1, 2, 5, 6, 8

[31] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel RealSense stereoscopic depth cameras. *arXiv preprint arXiv:1705.05548*, 2017. 7

[32] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, 2007. 1, 2

[33] Kurt Konolige. Projected texture stereo. In *ICRA*, 2010. 7

[34] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Structureless Pose-Graph Loop-Closure with a Multi-Camera System on a Self-Driving Car. In *IROS*, 2013. 1, 2

[35] Damien Lefloch, Markus Kluge, Hamed Sarbolandi, Tim Weyrich, and Andreas Kolb. Comprehensive use of curvature for robust and accurate online surface reconstruction. *PAMI*, 39(12):2349–2365, December 2017. 2

[36] Damien Lefloch, Tim Weyrich, and Andreas Kolb. Anisotropic point-based fusion. In *International Conference*

on Information Fusion (FUSION), pages 2121–2128, July 2015. 2

[37] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. InteriorNet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *BMVC*, 2018. 3

[38] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2

[39] Maxime Meilland, Christian Barat, and Andrew Comport. 3D high dynamic range dense visual SLAM and its application to real-time object re-lighting. In *ISMAR*, 2013. 2

[40] Maxime Meilland, Andrew Comport, and Patrick Rives. Real-time dense visual tracking under large lighting variations. In *BMVC*, 2011. 2

[41] Maxime Meilland and Andrew I. Comport. On unifying key-frame and voxel-based dense visual SLAM at large scales. In *IROS*, 2013. 2, 5, 6

[42] Maxime Meilland and Andrew I. Comport. Super-resolution 3D tracking and mapping. In *ICRA*, 2013. 2

[43] Maxime Meilland, Tom Drummond, and Andrew I. Comport. A unified rolling shutter and motion blur model for 3D visual registration. In *ICCV*, 2013. 2

[44] Raul Mur-Artal and Juan D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 1, 2, 6, 8

[45] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 1, 2, 3

[46] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011. 1

[47] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D reconstruction at scale using voxel hashing. In *SIGGRAPH Asia*, 2013. 2, 6

[48] H. K. Nishihara. PRISM: A practical real-time imaging stereo matcher. In *Technical Report A.I. Memo 780, MIT, Cambridge, MA*, 1984. 7

[49] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *ICCV*, 2017. 2

[50] Seonwook Park, Thomas Schöps, and Marc Pollefeys. Illumination change robustness in direct visual SLAM. In *ICRA*, 2017. 6

[51] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *ICCV*, 2017. 3

[52] Erik Ringaby and Per-Erik Forssén. Scan rectification for structured light range sensors with rolling shutters. In *ICCV*, 2011. 6

[53] Olivier Saurer, Marc Pollefeys, and Gim Hee Lee. Sparse to Dense 3D Reconstruction from Rolling Shutter Images. In *CVPR*, 2016. 1, 7

[54] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016. 1, 4, 5

[55] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. 4

[56] David Schubert, Nikolaus Demmel, Vladyslav Usenko, Jörg Stuckler, and Daniel Cremers. Direct sparse odometry with rolling shutter. In *ECCV*, 2018. 1, 7

[57] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stueckler, and Daniel Cremers. The TUM VI benchmark for evaluating visual-inertial odometry. In *IROS*, 2018. 3

[58] Yifei Shi, Kai Xu, Matthias Niessner, Szymon Rusinkiewicz, and Thomas Funkhouser. PlaneMatch: Patch coplanarity prediction for robust RGB-D registration. In *ECCV*, 2018. 7

[59] Miroslava Slavcheva, Wadim Kehl, Nassir Navab, and Slobodan Ilic. SDF-2-SDF: Highly accurate 3D object reconstruction. In *ECCV*, 2016. 2

[60] Jörg Stückler and Sven Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1):137–147, 2014. 1, 2, 6

[61] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, 2012. 1, 3, 6, 7

[62] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 1, 5

[63] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *PAMI*, 13(4):376–380, 1991. 7

[64] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018. 1

[65] Hao Wang, Jun Wang, and Liang Wang. Online reconstruction of indoor scenes from RGB-D streams. In *CVPR*, 2016. 2

[66] Oliver Wasenmüller, Marcel Meyer, and Didier Stricker. CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2. In *WACV*, 2016. 3

[67] Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. Online loop closure for real-time interactive 3D scanning. *CVIU*, 115(5):635–648, 2011. 1, 2

[68] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J. Leonard, and John McDonald. Real-time large scale dense RGB-D SLAM with volumetric fusion. *IJRR*, 2014. 1, 2, 6

[69] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *RSS*, 2015. 1, 2, 3, 6, 7

[70] Zhixin Yan, Mao Ye, and Liu Ren. Dense visual SLAM with probabilistic surfel map. *TVCG*, 23(11):2389–2398, 2017. 2, 6

[71] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser, and Sean Fanello. ActiveStereoNet: End-to-end self-supervised learning for active stereo systems. In *ECCV*, 2018. 7

[72] Qian-Yi Zhou and Vladlen Koltun. Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In *CVPR*, 2014. 3

[73] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3D reconstruction with RGB-D cameras. In *Eurographics*, 2018. 3, 4