European Conference on Mobile Robotics (ECMR), Lincoln, UK, September 2015

Approximate Surface Reconstruction and Registration for RGB-D SLAM

Dirk Holz and Sven Behnke

***Abstract*—RGB-D cameras have attracted much attention in the fields of robotics and computer vision, especially for object modeling and environment mapping. A key problem in all these applications is the registration of sequences of RGB-D images. In this paper, we present an efficient yet reliable approach to align pairs and sequences of RGB-D images that makes use of local surface information. We extend previous works on 3D mapping with micro aerial vehicles to sequences of RGB-D images. The resulting alignment is based on a robust surface-to-surface error metric and uses multiple surface-to-surface patch matches between pairs of RGB-D images. Quantitative evaluations show that our approach is competitive with state-of-the-art approaches.**

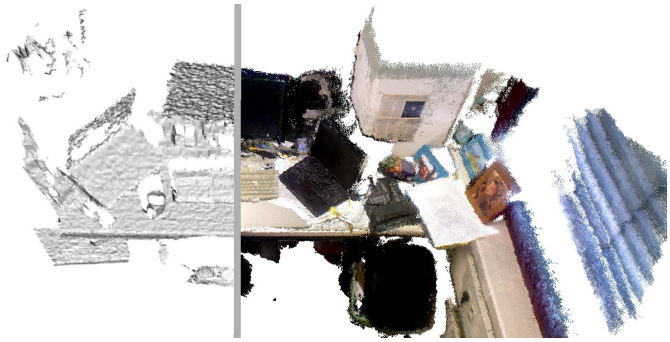


Fig. 1. Typical result of aligning a sequence of RGB-D images. Left: approximate surface reconstruction (unfiltered) of the first cloud. Right: the points of all aligned point clouds.

I. Introduction

Consumer color and depth cameras (RGB-D cameras) have huge potential in improving the perception capabilities of robots and automated vision systems in general. They acquire color (RGB) and depth (D) images both at high frame rates, e.g., 30 Hz. Intrinsic and extrinsic calibration of the two image sources yields colored 3D point clouds (see Fig. 1). Due to their comparably low cost, low weight, and small form factor, RGB-D cameras have attracted much attention in the fields of robotics and computer vision, especially for object modeling and environment mapping. What most applications have in common is that they require RGB-D images to be taken from multiple different viewpoints and that the acquired images need to be reliably registered such that the overlapping regions in the images match as well as possible. In the literature, this problem is usually referred to as Simultaneous Localization and Mapping (SLAM): building a map of the environment and localizing the information acquiring sensor(s) therein so as to consistently update and extend the map.

In recent years, many different approaches to visual odometry [1]–[3], SLAM [4]–[9], and dense mapping [10]–[13] have been proposed of which some are specifically tailored for RGB-D cameras. These methods are either based on features matched and tracked over sequences of images, or directly operate on the (semi-) dense color and depth images. Most approaches select a set of keyframes and optimize the resulting pose graph in order to obtain a globally consistent trajectory and map. State-of-the-art methods achieve globally consistent trajectories with low errors in pose estimation at high frame rates. We include some of them in a comparative evaluation.

In this paper, we state and address the problem of RGB-D SLAM in terms of multi-view 3D registration based on point correspondences between frames that encode a surface-to-surface error metric. The approach is based on previous work [14] for 3D laser scan registration and mapping with micro aerial vehicles. In order to compensate for the non-uniform point densities within and between individual scan lines of the fast rotating scanner, we approximated the underlying surface and used a generalized error metric [15] for obtaining robust registrations and accurate 3D maps of the sensed environmental structures such as buildings. In this paper, we extend the approach to be applicable to sequences of RGB-D images and make the following contributions:

1) In order to reduce the drift during the initial tracking of the camera, we register a newly acquired image against a local window of frames as opposed to the last (key) frame.
2) We integrate our previous works on range image segmentation [16] for efficiently computing local features such as surface normals on an approximate mesh representation, and for edge-aware filtering of the underlying points and the computed features to compensate for noise especially in the depth images.
3) To cope with the larger amount of data of RGB-D images in our multi-edge alignment approach, we efficiently sample both points in the images and found correspondences.

As a result, our approach of using multiple edges between views that encode surface-to-surface constraints can be applied to RGB-D video. Moreover, its performance is competitive with other state-of-the-art approaches. In fact, the proposed local window multi-edge alignment has a huge potential of contributing to other SLAM and object modeling pipelines. We present results of a thorough comparative experimental evaluation that proof these claims.

II. Related Work

Approaches to SLAM using monocular cameras, RGB-D cameras and stereo cameras can in general be split into two different categories: feature-based methods that compute and

All authors are with Autonomous Intelligent Systems Group, Computer Science Institute VI, University of Bonn, 53113 Bonn, Germany holz@ais.uni-bonn.de, behnke@cs.uni-bonn.de

track distinct repeatable key points and associate them using feature descriptors, and direct methods densely registering the acquired data.

For monocular cameras, a hybrid approach is the semi-dense visual odometry method proposed by Engel *et al.* [3]. It first computes inverse depth maps which are then used to align subsequent frames. A similar approach is followed by Forster *et al.* [7]. Engel *et al.* [9] extend their approach to build globally consistent maps even of large-scale environments.

Scherer and Zell [4] present an RGB-D SLAM approach that is efficient enough to be computed onboard an autonomous micro aerial vehicle. It is based on tracking FAST keypoints and the fast hierarchical graph optimization of Grisetti *et al.* [17]. The FAST corner detector is also used by Huang *et al.* [2] in their visual odometry method FOVIS.

A popular approach developed specifically for RGB-D images is RGB-D SLAM [8], [18]. It uses the color image to extract and match visual keypoints and descriptors (SURF, SIFT and ORB). The alignment relies on the 3D coordinates of keypoints obtained from the depth image. A similar approach is followed by Henry *et al.* [5] with FAST keypoints.

One of the first successful demonstrations of dense registration and mapping has been presented by Newcombe *et al.* and was coined KinectFusion [10]. KinectFusion uses signed distance functions in a grid-based environment representation and ICP-based registration [19] for aligning newly acquired depth images. All components are implemented on a GPU and allow (near) real-time operation. Assuming the camera movements between frames are small, this incremental registration can reliably align the data and update the environment model. By using more information than only a single frame against which KinectFusion registers considerably reduces drift usually arising in pairwise registration. In many cases, incremental registration can achieve globally consistent environment maps without the need for detecting loop closures and global optimization, e.g., as shown in a previous work on 2D laser-based mapping [20]. The drawback of incremental registration is that errors made in the update of the used environment representation cannot be corrected later. In this paper, we address the alignment of captured frames in terms of multi-view registration and do not build a particular environment representation.

Steinbruecker *et al.* [13] also use signed distance functions for dense mapping but organize the map in an octree structure. Stückler and Behnke [11] proposed a surfel-based registration method for constructing multi-resolution surfel maps (MRSMAPs) that are also represented in an octree. Kerl *et al.* [6] follow a different visual SLAM approach (DVO-SLAM) by minimizing the photometric and the depth error over all pixels. We include RGB-D SLAM [18], MRSMAP [11], DVO-SLAM [6], and an open source implementation of KinectFusion [10] in a comparative experimental evaluation.

Recently, Maier *et al.* [21] presented an efficient approach to RGB-D object modeling. They split the camera trajectory into chunks of equal size, and first optimize the alignments within the chunks before globally aligning the chunks to each other. Since the camera is moved around the object to model, these splits along the trajectory yield spatially coherent partitions. We achieve a similar behavior by using local windows in the initial alignment of newly acquired frames. Moreover, the local windows include earlier frames in case of loop closures. The local alignment can then compensate for the accumulated drift or trigger a global optimization of the trajectory in case conflicts are found. Our local alignment approach is inspired by the double window approach in the SLAM framework of Strasdat *et al.* [22].

In multi-view scan matching, poses are determined simultaneously by aligning all scans. In the 2D domain, a popular approach is the one by Lu and Milios (LUM) [23]. Borrmann *et al.* [24] extend this approach to six degrees of freedom for the alignment of 3D scans and present methods to efficiently deal with the resulting nonlinearities [24]. The resulting approach first applies the ICP algorithm to align consecutive point clouds and then builds a graph based on the determined connectivity of view poses similar to our approach. Both the determined transformations and the sets of point correspondences are represented in the edges. From both, a measurement vector and its covariance matrix are computed which are then fed as one block into a large linear system for optimization. In contrast, in our approach, every correspondence pair forms a block in the final non-linear error function. Furthermore, LUM uses a point-to-point error metric as in the original ICP algorithm. Instead, we approximate the surface and use a probabilistic surface-based error metric.

Similar to our multi-edge alignment step are the approaches of Zlot and Bosse [25] and Ruhnke *et al.* [26]. For mapping mines with a continuously spinning laser scanner, Zlot and Bosse use non-rigid surfel registration and graph optimization for aggregating point clouds and building consistent maps. Ruhnke *et al.* also use raw point matches as constraints in the graph and apply a surfel-based error metric to iteratively refine both the sensor poses and the positions of the points. Their approach can build highly accurate object models but requires a rough initial alignment of the dense RGB-D data. Moreover, by optimizing the position of every point in the resulting object model, the approach is computationally complex. In contrast, we aim at both initially aligning the acquired point clouds and building globally consistent environment models while trying to reduce the complexity of the involved processing steps, e.g., by using only descriptive subsets of the dense RGB-D data and local windows. The idea behind this paper is to apply our pipeline for 3D mapping with MAVs [14] to dense RGB-D data, making the necessary adaptions to make it both applicable and feasible, and to evaluate how the resulting system compares to state-of-the-art RGB-D SLAM approaches.

III. METHOD

Our approach is split into three stages. In the first stage, we approximate, for each frame, the underlying surface in the form of a quad mesh. The mesh serves three purposes: it allows 1) computing features such as surface normals directly on the mesh, 2) extracting neighborhoods from the mesh topology, and 3) caching values such as computed distances and normal deviations in its edges. Referring to Fig. 2, the extracted mesh is then smoothed and used for feature extraction. In the second stage, both the mesh and the computed features are fed into the local alignment so as to keep track of the camera pose. If loop closures are detected, the so far estimated trajectory is globally optimized in the third stage.

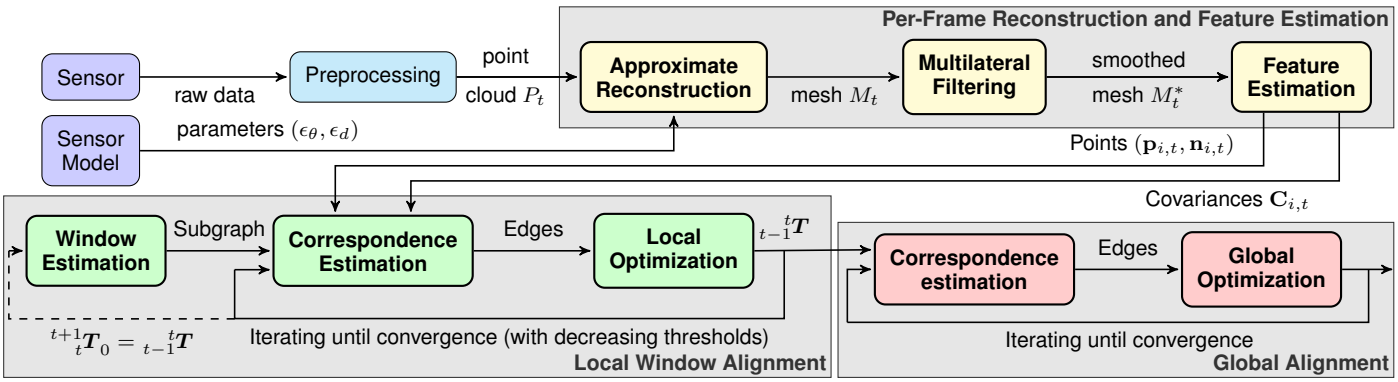


Fig. 2. System overview and data flow. For a newly acquired point cloud, we first approximate the underlying surface reconstruction. Using the mesh topology, we compute approximate local surfaces and apply a multilateral filter to smooth both points and normals. We then compute local covariances and feed the point clouds as well as the computed features into the local alignment. Global optimization then yields globally consistent trajectories.

A. Approximate Surface Reconstruction

Surface reconstructions are compact representations of the underlying sensed environmental structures and can become handy in a variety of pre-processing tasks such as computing point neighborhoods, local surface normals or smoothed (and upsampled or downsampled) representations [16]. In order to compute an approximate surface reconstruction, we traverse an organized point cloud P once and build a simple quad mesh by connecting every point $\boldsymbol{p} = P(u, v)$ (in the u-th row and the v-th column) to its neighbors $P(u, v+1)$, $P(u+1, v+1)$, and $P(u+1, v)$ in the next row and column. We only add a new quad to the mesh if $P(u, v)$ and its three neighbors are valid measurements, and if all connecting edges between the points are not occluded. The first check accounts for possibly missing or invalid measurements in the organized structure. For the latter occlusion checks, we examine if one of the connecting edges falls into a common line of sight with the viewpoint $\boldsymbol{v} = \boldsymbol{0}$. If so, one of the underlying surfaces occludes the other and the edge is not $valid$:

$$valid = (|\cos\theta_{i,j}| \leq \cos\epsilon_\theta) \wedge \left(d_{i,j} \leq \epsilon_d^2\right), \quad (1)$$

$$\text{with} \quad \theta_{i,j} = \frac{(\boldsymbol{p}_i - \boldsymbol{v}) \cdot (\boldsymbol{p}_i - \boldsymbol{p}_j)}{\|\boldsymbol{p}_i - \boldsymbol{v}\| \, \|\boldsymbol{p}_i - \boldsymbol{p}_j\|}, \quad (2)$$

$$\text{and} \quad d_{i,j} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2, \quad (3)$$

where ϵ_θ and ϵ_d denote maximum angular and length tolerances, respectively. The latter accounts for sensor noise, i.e., tolerable depth continuities such as quantization effects in the depth images. We use a simple isotropic noise model for Microsoft Kinect cameras that we developed for range image segmentation [16]. It depends on the measured distance z:

$$\epsilon_d(z) = n\sqrt{2}\,\sigma(z), \quad (4)$$

$$\text{with } \sigma(z) = 0.00263z^2 - 0.00519z + 0.00755, \quad (5)$$

where n is the subsampling factor applied, i.e., using only every n-th row and column for constructing the quad mesh. If both distance and occlusion checks pass, we add a new quad. Otherwise, holes arise. After construction, we simplify the resulting mesh by removing unused vertices.

B. Multilateral Filtering

Naturally, sensor measurements are affected by noise. Especially depth images suffer from distance-dependent noise and quantization effects. In order to compensate for local noise in depth measurements, we apply a filter for smoothing both the points and their normals while preserving edges in the sensed geometric structures. The formulation of our filter is motivated by the concept of multilateral filtering [27] and measures the similarity of points w.r.t. their position, surface orientation, and appearance. We filter both a point $\mathbf{p}_i$ and its normal $\mathbf{n}_i$ over its 1-ring-neighborhood N_i, i.e., all points that are directly connected to $\mathbf{p}_i$ by an edge in the mesh:

$$\mathbf{p}_i = \frac{\sum_{j\in N_i} w_{ij}\mathbf{p}_j}{\sum_{j\in N_i} w_{ij}}, \quad \text{and } \mathbf{n}_i = \frac{\sum_{j\in N_i} w_{ij}\mathbf{n}_j}{\sum_{j\in N_i} w_{ij}}, \quad (6)$$

$$\text{with} \quad w_{ij} = \underbrace{e^{-\alpha\|\mathbf{p}_i - \mathbf{p}_j\|}}_{\text{distance term}} \underbrace{e^{-\beta\|\mathbf{n}_i - \mathbf{n}_j\|_1}}_{\text{normal term}} \underbrace{e^{-\gamma(I_i - I_j)/c_I}}_{\text{intensity term}}. \quad (7)$$

The normalization constant c_I is used to scale the intensity differences to lie in the interval $[0, 1]$. Weights α, β, and γ can be used to adjust the behavior of the filter. Equally weighting distance, surface normal and color deviation term already achieves considerable smoothing while preserving edges and corners ($\alpha = \beta = \gamma = 1$). Depending on the desired smoothing level, we extend the point neighborhood to include the neighbors of neighbors and ring neighborhoods farther away from the point.

C. Approximate Normal and Covariance Estimates

In order to estimate local surface normal and covariance matrix of a point, we directly extract its local neighborhood from the topology in the mesh instead of searching for neighbors. We compute the normal $\boldsymbol{n}_i$ for a point $\boldsymbol{p}_i$ directly on the mesh as the weighted average of the plane normals of the N_T faces surrounding $\boldsymbol{p}_i$ (extracted from the topology):

$$\boldsymbol{n}_i = \frac{\sum_{j=0}^{N_T} (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,b}) \times (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,c})}{\| \sum_{j=0}^{N_T} (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,b}) \times (\boldsymbol{p}_{j,a} - \boldsymbol{p}_{j,c}) \|}, \quad (8)$$

with face vertices $\boldsymbol{p}_{j,a}$, $\boldsymbol{p}_{j,b}$ and $\boldsymbol{p}_{j,c}$. We then compute the local covariance matrix $\boldsymbol{\Sigma}_i$ as in [15]:

$$\boldsymbol{\Sigma}_i = \boldsymbol{R}_{\boldsymbol{n}_i} \begin{pmatrix} \epsilon & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{R}_{\boldsymbol{n}_i}{}^T \quad (9)$$

with a rotation matrix $\boldsymbol{R}_{\boldsymbol{n}_i}$ so that ϵ reflects the uncertainty along the approximated local surface normal $\boldsymbol{n}_i$. The intuition

behind this is that we assume the point to lie on the approximated surface while not knowing where the point is lying on the surface. The lower the uncertainty ϵ the more we assume local planarity around measured points. Consequently, with a low value ($0 < \epsilon \leq 10^{-3}$), the registration error to be minimized (introduced in the following) converges to a plane-to-plane error metric.

D. Surface-to-Surface Alignment

In order to align, respectively, two approximated surfaces and two organized point clouds A and B, we search for closest neighbors in B for points $\boldsymbol{a}_i \in A$ and iteratively minimize the distances between the found matches. Instead of minimizing the point-to-point distances $\boldsymbol{d}_{ij}^{(\boldsymbol{T})} = \boldsymbol{b}_j - \boldsymbol{T}\boldsymbol{a}_i$ of the set of found correspondences C to determine the transformation $\boldsymbol{T}$ as in the original Iterative Closest Point algorithm [19], we use the generalized error metric introduced by Segal *et al.* [15]. It generalizes over the different available error metrics (point-to-point, point-to-plane, plane-to-plane) and thus takes into account information about the underlying surface. Instead of minimizing the distances $\boldsymbol{d}_{ij}^{(\boldsymbol{T})}$ between corresponding points $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$, it models the distribution

$$\boldsymbol{d}_{ij}^{(\boldsymbol{T})} \sim \mathcal{N}\left(\boldsymbol{b}_j - \boldsymbol{T}\boldsymbol{a}_i,\ \boldsymbol{\Sigma}_j^B + \boldsymbol{R}\boldsymbol{\Sigma}_i^A\boldsymbol{R}^T\right) \tag{10}$$

where $\boldsymbol{R}$ is the rotation matrix of $\boldsymbol{T}$ under the assumption that both points in A and points in B are itself drawn from independent normal distributions, i.e., $\boldsymbol{a}_i \sim \mathcal{N}(\hat{\boldsymbol{a}}_i, \boldsymbol{\Sigma}_i^A)$ and $\boldsymbol{b}_j \sim \mathcal{N}(\hat{\boldsymbol{b}}_j, \boldsymbol{\Sigma}_j^B)$. Given the correspondences $ij \in C$, the optimal transformation $\boldsymbol{T}^\star$ best aligning A to B can then be found using maximum likelihood estimation (MLE):

$$\begin{aligned}\boldsymbol{T}^\star &= \arg\max_{\boldsymbol{T}} \prod_{ij\in C} p\left(d_{ij}^{(\boldsymbol{T})}\right) = \arg\max_{\boldsymbol{T}} \sum_{ij\in C} \log\left(p\left(d_{ij}^{(\boldsymbol{T})}\right)\right) \\ &\simeq \arg\min_{\boldsymbol{T}} \underbrace{\sum_{ij\in C} d_{ij}^{(\boldsymbol{T})^T}\left(\boldsymbol{\Sigma}_j^B + \boldsymbol{R}\boldsymbol{\Sigma}_i^A\boldsymbol{R}^T\right)^{-1} d_{ij}^{(\boldsymbol{T})}}_{\text{= simplified Likelihood } L(\boldsymbol{T})}. \end{aligned} \tag{11}$$

The effect of minimizing (11) is that corresponding points are not directly dragged onto another, but the underlying surfaces represented by the covariance matrices $\boldsymbol{\Sigma}_i^A$ and $\boldsymbol{\Sigma}_j^B$ are aligned.

In previous work [14], we have used this approach for 3D SLAM with a light-weight continuously rotating 3D laser scanner carried by a micro aerial vehicle. In order to compensate for smaller inaccuracies in pair-wise registration, we used multiple edges between neighboring poses in the final trajectory optimization, where every single edge encoded a surface-to-surface error correspondence using the error metric in (11). In this paper we no longer distinguish between pairwise initial alignment and subsequent global optimization. Instead, both the initial alignment in the local windows and the global trajectory optimization in case of loop closures are formulated in exactly the same multi-edge graph optimization approach.

E. Multi-Edge Graph Optimization

In a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, neighboring poses in the trajectory form the vertices $\boldsymbol{v}_i \in \mathcal{V}$ and spatial constraints (transformations) between two vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are represented by edges $e_{ij} \in \mathcal{E}$. Instead of adding only a single edge between two vertices that encodes a transformation and the corresponding covariance matrix, we search for corresponding points between the respective point clouds and add multiple edges, one for each found correspondence (see Fig. 3).

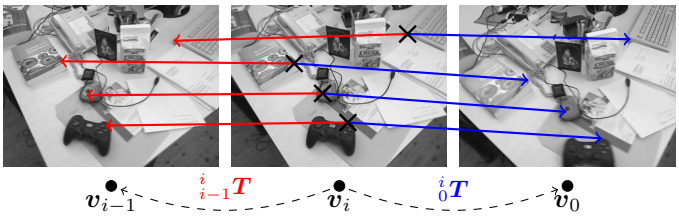


Fig. 3. Example of connecting a frame at vertex $\boldsymbol{v}_i$ to the first frame $\boldsymbol{v}_0$ and the last frame $\boldsymbol{v}_{i-1}$. Instead of using a single edge encoding the transformations (dashed lines), we use one edge per point correspondence. Instead of repeatable features, we use raw points and iteratively refine the matching.

Each edge in the graph encodes two entities: a local contribution to the measurement error $\boldsymbol{e}$ and an information matrix $\boldsymbol{H}$ which represents the uncertainty of the measurement error. The information matrix is defined as the inverse of the covariance matrix, i.e., it is symmetric and positive semi-definite. For the error measurement between, respectively, two vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ and the correspondence pair $(\boldsymbol{p}_{i,m}, \boldsymbol{p}_{j,n})$, we use the point-to-point difference vector and approximate its information matrix using the error generalized error metric (11):

$$\text{mean} \quad \boldsymbol{e}_{ij,mn}({}^{i}_{j}\boldsymbol{T}) = \boldsymbol{p}_{j,n} - {}^{i}_{j}\boldsymbol{T}\boldsymbol{p}_{i,m}, \tag{12}$$

$$\text{and} \quad \boldsymbol{H}_{ij,mn}({}^{i}_{j}\boldsymbol{T}) = \left(\boldsymbol{\Sigma}_m^{P_j} + \boldsymbol{R}\boldsymbol{\Sigma}_n^{P_i}\boldsymbol{R}^T\right)^{-1}. \tag{13}$$

The effect is that every edge contributes its approximate surface-to-surface error term to the system information matrix—thus automatically giving lower influence on incompatible or false correspondences and quickly leading to alignment even in case of larger initial displacements.

For the actual optimization, we follow an iterative procedure by 1) estimating correspondence pairs for all (or a subset of) points $\boldsymbol{p}_{i,m} \in P_i$ in P_j for every two vertices $(\boldsymbol{v}_i, \boldsymbol{v}_j)$ that are to be connected and 2) optimizing the resulting linearized system for a maximum of ten inner iterations. We repeat these two steps for a maximum of ten outer iterations. For a fast initial coarse alignment in early and an accurate refinement in later outer iterations, we use a linearly decreasing distance threshold for correspondence pairs, starting with $2\,\mathrm{m}$ (∞ in the first iteration) and going to two times the expected local noise (4). In every outer iteration step, the graph is optimized using dense Cholesky decomposition and Levenberg Marquardt within the g2o framework [28]. For both inner and outer iterations, we stop when the system has converged. Convergence in graph optimization (inner iterations) can be detected based on the changes in both view poses and system error as well as the damping factor applied by Levenberg Marquardt. For detecting convergence in the overall graph refinement in the outer iterations, we check whether the view pose connectivity and the correspondences between connected view poses have changed. When no more changes are found and the inner optimization has converged, we stop optimizing the trajectory.

F. Local Window Alignment

In order to estimate a rough initial pose estimate for a newly acquired frame, standard SLAM procedures would first register

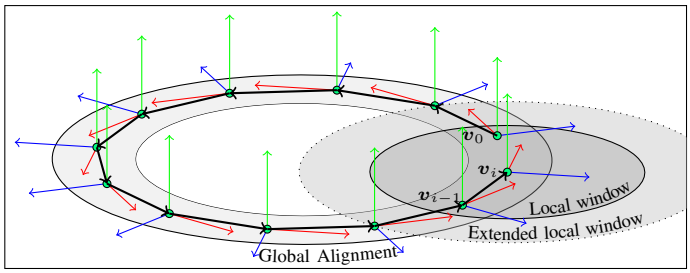


Fig. 4. Alignment in local windows as opposed to pairwise alignments. Aligning a frame (x-axis red, y-axis green, and z-axis red) to multiple other frames tends to be more stable and to drift less. In this example of a camera moving along a circular trajectory, the local window around the vertex $\boldsymbol{v}_i$ includes the last frame (at $\boldsymbol{v}_{i-1}$) and the starting pose $\boldsymbol{v}_0$ upon loop closure.

the new frame against the last (key) frame, and then search for possible loop closures for a subsequent global optimization. Instead, we determine a local window of neighboring poses and simultaneously align the new frame against all frames acquired at poses within the local window. Referring to Fig. 4, for a pose at $\boldsymbol{v}_i$ we search for closest poses (i.e., camera origins) in 3D. The found candidates are checked for a similar viewing direction by means of the angle between the camera z axes. This rough initial check suffices since the following alignment accurately deals with overlapping and non-overlapping measurement volumes. We define the local window to contain 1) the last acquired frame and the last acquired key frame, respectively, as well as 2) all poses within a radius r around the camera origin w.r.t. the current pose estimate that have a similar orientation. In order to obtain constant-time initial alignments, we additionally use an upper limit of w for the number of neighboring poses in the local window and sample the matches in between neighboring frames to keep both the number of vertices and the number of edges in the constructed subgraph constant.

Once the local window is determined, the newly acquired frame is aligned to all frames in the local window by estimating only the pose of the frame being aligned. The other poses are fixed during optimization. In contrast to pairwise registration, this one-to-many alignment is more stable and tends to drift less (see the experimental evaluation in Sec. IV). Moreover, it allows for efficiently detecting loop closures and possible inconsistencies in the trajectory estimate as described next.

G. Loop Closure Detection and Global Optimization

Our primary mean for detecting loop closures is to inspect the poses found in the local window. Naturally, if a similar pose is found that has been acquired long ago (in terms of time and frame index), a loop closure is detected. Since our local initial alignment is quite stable without considerable drifts, loop closures can be easily detected as long as the camera trajectory is bound to a single room. In larger environments, drifts in the local alignments accumulate and more sophisticated means for recognizing previously visited places are needed [29].

Once, the local window contains an earlier pose along the trajectory, we compare the transformations obtained from the alignment in the local window with those in the so far built global graph. In case of conflicts, e.g., larger jumps in the estimated pose or no convergence in the optimization, we trigger an alignment in an extended local window. This extended window includes the 1-ring neighborhood of the local window. For the alignment, all poses in the local window are now optimized, and only the poses in the extended border (i.e., in the 1-ring neighborhood) are fixed. If the extended local window contains earlier poses or shows conflicts after local alignment, global trajectory optimization is triggered.

The global optimization of the trajectory follows the same principle as the local alignment. In order to save processing time, however, the global graph does not contain all poses but only a limited set of keyframes.

H. Keyframe Selection

Several strategies exist to select whether or not to add a new key frame, e.g., adding every n-th frame, applying rotational and translational thresholds, or applying thresholds on registration error variances or the number of matched features. We apply a rotational threshold and a translational threshold as fixed upper limits in order to avoid larger distances between key frames even if the alignments in between are good. In addition, we use a measure based on matching quality and uncertainty along the dimensions of the transformation. After the local alignment of a newly acquired frame, we inspect the determined transformation $\boldsymbol{T}$ to the last keyframe and compute an estimate of the uncertainty. We use the approximation by Censi [30] to compute the covariance matrix:

$$\boldsymbol{\Sigma_T} \approx \left(\frac{\partial^2 L}{\partial \boldsymbol{x}^2}\right)^{-1} \frac{\partial^2 L}{\partial c \partial \boldsymbol{x}} \boldsymbol{\Sigma}(c) \frac{\partial^2 L}{\partial c \partial \boldsymbol{x}}^T \left(\frac{\partial^2 L}{\partial \boldsymbol{x}^2}\right)^{-1}, \quad (14)$$

where L is the simplified likelihood function in (11), c denotes the individual found correspondences C between the two point clouds P_i and P_j, and $\boldsymbol{\Sigma}(c)$ is the covariance of the correspondence pairs. Note that in (14), the relative transformation between two view poses is not represented as a homogeneous transformation matrix $\boldsymbol{T}$, but in a parameterized form $\boldsymbol{x} = (\boldsymbol{t}, \boldsymbol{q})^T$ with translation $\boldsymbol{t}$ and rotation by the unit quaternion $\boldsymbol{q}$. We then follow the approach of Kerl *et al.* [6] to compute an entropy-based measure

$$H\left(\boldsymbol{T}, \boldsymbol{\Sigma_T}\right) \propto \ln\left(|\boldsymbol{\Sigma_T}|\right), \quad (15)$$

using the determinant of $\boldsymbol{\Sigma_T}$. The entropy of the current transformation (against the last key frame) is then compared to the stored entropy of the last key frame (when it was added). If the ratio between the two entropy measures falls below a predefined threshold, the last (not the currently aligned) frame is added as a key frame, and the local alignment is repeated.

I. Point Subsampling and Correspondence Filtering

An important aspect in the alignment is determining correspondences between frames. Every such correspondence will contribute in the form of an edge to both subgraph and global graph optimization. In order to use only a small number of correspondences without loosing much information, we follow a multi-stage strategy: we first subsample query points from the frame to be aligned and then reject found correspondences that are unlikely to contribute to the alignment.

We sample query points from two distributions: uniformly over the rows and columns of the image and uniformly in normal space. The intuition behind the latter is that we want to

TABLE I. RELATIVE POSE ERROR (RPE) IN INITIAL ALIGNMENTS (WITHOUT GLOBAL OPTIMIZATION), RMSE of RPE(Δ) in $\mathrm{m/s}$ with $\Delta = 1\,\mathrm{s}$

	Pairwise Mesh Registration*			Mesh Registration + Filtering** ($n=4$)				Local Alignment*** + Filtering ($n=4, k=4$)				
Dataset	$n=1$	$n=2$	$n=4$	$k=1$	$k=2$	$k=3$	$k=4$	$w=2$	$w=3$	$w=4$	$w=5$	$w=10$
fr1_xyz	0.204	0.216	0.199	0.192	0.172	0.159	0.156	0.119	0.087	0.068	0.052	0.041
fr1_rpy	0.197	0.205	0.189	0.181	0.175	0.160	0.162	0.127	0.101	0.099	0.081	0.059
fr1_desk	0.303	0.313	0.324	0.235	0.218	0.207	0.201	0.165	0.138	0.108	0.094	0.084
fr1_desk2	0.457	0.436	0.429	0.324	0.301	0.281	0.256	0.152	0.129	0.117	0.102	0.096
fr1_room	0.383	0.375	0.390	0.281	0.262	0.239	0.241	0.189	0.148	0.120	0.098	0.084
fr1_360	0.831	0.802	0.806	0.651	0.612	0.495	0.492	0.271	0.229	0.191	0.171	0.162
fr1_teddy	0.102	0.148	0.114	0.101	0.099	0.098	0.098	0.089	0.085	0.081	0.078	0.078
fr1_plant	0.143	0.152	0.141	0.135	0.136	0.131	0.131	0.111	0.083	0.074	0.059	0.052
fr2_desk	0.198	0.192	0.201	0.152	0.148	0.138	0.139	0.091	0.072	0.051	0.039	0.030
fr3_long_office_...	0.103	0.124	0.101	0.093	0.093	0.091	0.089	0.071	0.06	0.055	0.041	0.034
Avg. improvement	—	-1.5%	1%	20%	24%	31%	32%	52%	61%	66%	72%	75%

* Mesh Registration: the input images are subsampled by using only every n-th row and column, e.g., $n = 4$ corresponds to a 160×120 image.
** Filtering: the local neighborhood used by the multilateral filter is sequentially expanded to include the 1 to k-ring neighrborhoods.
*** Local alignment: the window size w determines the number of (closest) vertices used for optimization of the subgraph.

draw samples from all surface orientations in the scene so as to robustify the alignment along all dimensions. Note that the sampled set of query points is stored for later correspondence searches if the frame is added as a keyframe.

In order to search for corresponding points in the other frames, we first project each query point into the camera coordinate frame of the target frame and check whether it is visible by frustum culling [31]. If the query point lies within the view frustum of the target frame, we use the closest point in the smoothed mesh of the target point cloud. The resulting set of correspondences is then filtered again 1) to remove false correspondences that can negatively affect the alignment, and 2) to further reduce the number of correspondences. We remove correspondences that include surface boundary points (e.g., introduced by occlusions), and apply filters on the residual correspondence pairs that remove 1) pairs whose point-to-point distance exceeds the median point-to-point distance over all correspondences, 2) pairs whose local surface normal orientations considerably deviate, and 3) pairs that contain the same matching point in the target frame. In the latter case, only the pair with the smallest point-to-point distance is kept. The local surface normals are considered to avoid that points with normals pointing in opposite directions form a correspondence.

IV. EVALUATION

In order to assess the performance of our approach, we use the datasets and error metrics from the publicly available RGB-D SLAM Benchmark[1] by Sturm *et al.* [32].

A. Accuracy of Local Alignments

For measuring the drift in initial alignments (without global optimization), we use the relative pose error (RPE) [32]. It computes the root mean square error (RMSE) of the translational errors between the estimated poses and the corresponding ground truth poses in an interval Δ. We use $\Delta = 1\,\mathrm{s}$ to measure the drift in meters per second ($\mathrm{m/s}$). For different datasets, we compare the results in terms of the RPE for different processing steps and parameters. In particular, we focus on the effect of subsampling the input image (in order to reduce processing time), filtering (to smooth the underlying data), and the size of the local window. We report all results in Table I. Note that in these experiments, global trajectory optimization is disabled. For all processing steps and parameter sets, we computed the average translational drifts and compared them with the average drift of the plain pairwise registration as a baseline to obtain an average improvement. The subsampling experiments indicate that, since both the query points and the found correspondences are already drastically sampled, the effect of subsampling the input image in the course of approximate surface reconstruction is only minor. For this reason, we have chosen to process 160×120 images ($n = 4$) to reduce computations in the pre-processing steps which are conducted for every single frame.

In contrast, smoothing the underlying data (and thus also the surface normals used in the alignment) significantly improves the alignments and reduces the translational drift. In most of the datasets, a larger portion of the data is not sensed on the object of interest (e.g., the teddy, the plant, or the desks), but on environmental structures such as walls farther away from the sensor. Therefore, the respective depth measurements are more affected by quantization effects. With an increasing smoothing factor (the included k-ring neighborhoods), the multilateral filter can effectively smooth over the emerging depth discontinuities while preserving edges. We achieved the best results with $k = 4$ and suggest to not use ring neighborhoods farther away (i.e., $k \geq 5$), since especially the local surface normals become too inaccurate and details will be smoothed away. Overall, an improvement of roughly $30\,\%$ can be achieved for pairwise registration if the data is smoothed before alignment.

Compared to pairwise registration that only uses the last keyframe (i.e., $w = 1$), using a local window for the alignment drastically reduces the drift. The more other frames are used in the alignment, the more stable and accurate the estimated pose becomes. However, this improvement comes at the price of optimizing a larger subgraph (see Fig. 5). Since the average improvement does not considerably increase for larger local windows, we use a window size of $w = 5$ as it allows locally aligning new frames at roughly $10\,\mathrm{Hz}$.

B. Trajectory Optimization and Global Alignment

In a final series of experiments, we used our complete pipeline with global trajectory optimization enabled. We subsample the point clouds with $n = 4$ in the approximate surface reconstruction, include all local neighbors for smoothing up to ring $k = 4$, and initially align newly acquired point clouds in a local window of size of $w = 5$. For measuring the errors in the final optimized trajectory, we use the absolute trajectory

[1] http://vision.in.tum.de/data/datasets/rgbd-dataset

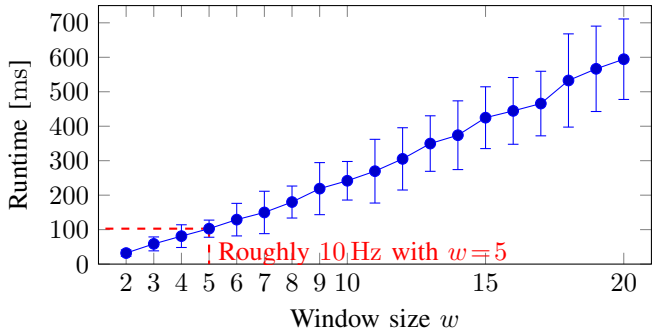


Fig. 5. Average runtimes of the local alignment per frame, measured over all datasets (with 10 complete runs per dataset) on a single core of an Intel Core i7-3740QM CPU (2.70GHz).

error (ATE) [32]. It determines a transformation best aligning the estimated trajectory and the ground truth trajectory in order to compute errors between individual frames regardless of the used base coordinate frame. We compare both our local and our global alignment approaches to DVO-SLAM by Kerl *et al.* [6], MRSMAP by Stückler and Behnke [11], RGB-D SLAM by Endres *et al.* [8], [18], and KinFu, the PCL open source implementation of KinectFusion by Newcombe *et al.* [10]. Since we have not been able to produce better results than Kerl *et al.* in our experiments, we compare against the values reported in [6]. As in the case of the translational drift, we report the RMSE in Table II with example results in Fig. 6.

Naturally, our multi-view registration approach cannot outperform sophisticated state-of-the-art dense visual SLAM methods specifically designed for RGB-D data. For example, both DVO-SLAM [6] and MRSMAP [11] apply a coarse-to-fine registration on different resolutions. In contrast, we use only the initially sampled subsets of points and the found matches during the alignment. Hence, our alignment neglects details such as smaller objects on the tables or texture details in general. This is reflected in a slightly higher ATE for our approach(es). The higher error stems from minor local inaccuracies in the overall globally consistent trajectory estimates. Still, we get a better ATE in two datasets (fr1_desk2 and fr1_plant).

While, on average, DVO-SLAM [6] and MRSMAP [11] outperform the other approaches, our global alignment comes in third and achieves a lower average ATE than both RGB-D SLAM [18] and KinFu. Most notably, however, is that our local alignment approach (without global optimization), achieves very good initial trajectory estimates without the need of optimizing more than a local window. Hence, we believe, that our approach has a large potential for inspiring related approaches and for contributing to other SLAM pipelines.

V. Conclusions

We have presented a complete pipeline for aligning pairs and sequences of RGB-D images. Our approach is based on approximating the underlying surface in the form of a quad mesh, and using the mesh for fast feature estimation (normals, covariances, etc.) and edge-aware smoothing. The alignment makes use of graph optimization with multiple edges between vertices, where every edge encodes a surface-to-surface error constraint. In order to reduce computation time, we efficiently subsample both points and used correspondences between frames for almost constant-time local alignments. Our experiments show that aligning newly acquired images in a local window as opposed to pair-wise alignment with the last frame reduces drift and achieves low relative pose estimation errors. Optimizing the complete graph after loop closures and as a last processing step yields globally consistent alignments and trajectory estimates.

TABLE II. Absolute Trajectory Errors (ATE), RMSE in m

Dataset	Ours (local)	Ours (global)	DVO-SLAM*	MRS-MAP*	RGB-D SLAM*	KinFu*
fr1_xyz	0.051	0.013	**0.011**	0.013	0.014	0.026
fr1_rpy	0.131	0.028	**0.020**	0.027	0.026	0.133
fr1_desk	0.052	0.028	**0.021**	0.043	0.023	0.057
fr1_desk2	0.081	**0.039**	0.046	0.049	0.043	0.420
fr1_room	0.142	0.073	**0.053**	0.069	0.084	0.313
fr1_360	0.254	0.082	0.083	**0.069**	0.079	0.913
fr1_teddy	0.082	0.090	**0.034**	0.039	0.079	0.154
fr1_plant	0.051	**0.025**	0.028	0.026	0.091	0.598
fr2_desk	0.090	0.046	**0.017**	0.052	—	—
fr3_long_...	0.043	0.037	**0.035**	—	—	0.064
Average	0.097	0.046	**0.034**	0.043	0.054	0.297

— *As reported by Kerl *et al.* [6]. —

In experiments, we could show that our approach is competitive with state-of-the-art approaches in terms of pose estimation accuracy. Naturally, our approach cannot be as good as sophisticated (RGB-D) SLAM approaches in terms of detail and speed (e.g., compared to the multi-resolution surfel maps [11]). However, our approach, and especially the surface-based alignment in local windows, have a huge potential for contributing to other SLAM frameworks. Moreover, the multi-edge approach can be easily extended to include point-to-point correspondences, e.g., from matching visual or 3D features.

In its current implementation, our approach distinguishes only local and global alignment. A logical next step would be to extend it to a hierarchical approach with the complete trajectory and subgraphs on higher levels, and sub-regions and single points in images on lower levels so as to allow the alignment to correct individual measurements and inaccuracies within 3D point clouds.

Acknowledgements

This project has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 610917 (STAMINA).

References

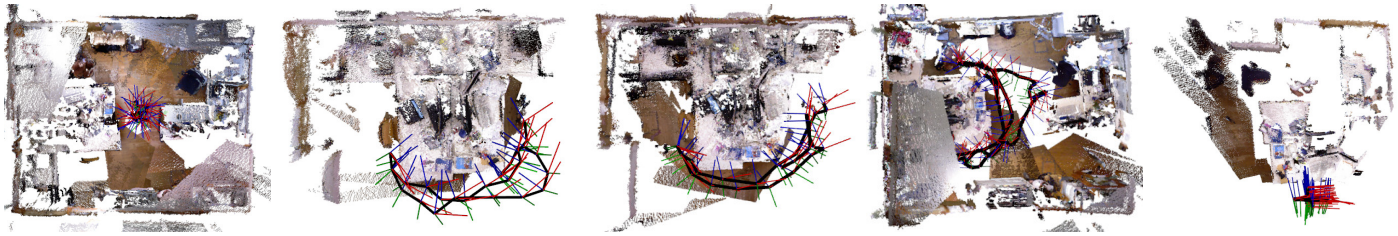
[1] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *Proc. of the Intelligent Vehicles Symposium (IV)*, 2010.

[2] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Int. Symp. on Robotics Research (ISRR)*, 2011.

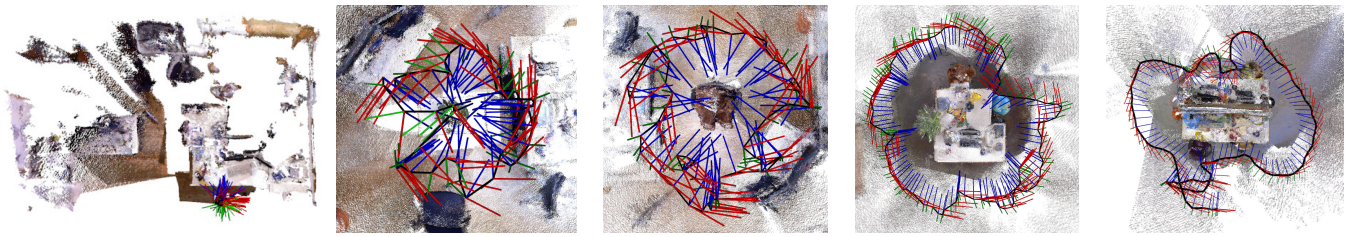
[3] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2013, pp. 1449–1456.

[4] S. A. Scherer and A. Zell, "Efficient onbard RGBD-SLAM for autonomous MAVs," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 1062–1068.

[5] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The Int. Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.



(a) fr1_360 (b) fr1_desk (c) fr1_desk2 (d) fr1_room (e) fr1_xyz



(f) fr1_rpy (g) fr1_plant (h) fr1_teddy (i) fr2_desk (j) fr3_office

Fig. 6. RGB-D SLAM results. Shown are the acquired RGB-D point clouds aligned using the determined poses (top view). The poses are visualized using the respective coordinate frames (x-axis red, y-axis green, and z-axis blue). In addition, we visualize the estimated camera trajectory as a black line connecting consecutive poses. All maps and trajectories are globally consistent.

[6] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 2100–2106.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[8] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[9] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: large-scale direct monocular SLAM," in *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014, pp. 834–849.

[10] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: real-time dense surface mapping and tracking," in *Proc. of the IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[11] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3D modeling and tracking," *J. Vis. Comun. Image Represent.*, vol. 25, no. 1, pp. 137–147, 2014.

[12] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013, pp. 5724–5731.

[13] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers, "Large-scale multi-resolution surface reconstruction from rgb-d sequences," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2013, pp. 3264–3271.

[14] D. Holz and S. Behnke, "Mapping with micro aerial vehicles by registration of sparse 3D laser scans," in *Proc. of the Int. Conf. on Intelligent Autonomous Systems (IAS)*, 2014.

[15] A. Segal, D. Hähnel, and S. Thrun, "Generalized-ICP," in *Proc. of Robotics: Science and Systems*, 2009.

[16] D. Holz and S. Behnke, "Approximate triangulation and region growing for efficient segmentation and smoothing of range images," *Robotics and Autonomous Systems*, vol. 62, no. 9, pp. 1282–1293, 2014.

[17] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 273–278.

[18] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[19] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[20] D. Holz and S. Behnke, "Sancta Simplicitas – On the efficiency and achievable results of SLAM using ICP-Based Incremental Registration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 1380–1387.

[21] R. Maier, J. Sturm, and D. Cremers, "Submap-based bundle adjustment for 3D reconstruction from RGB-D data," in *Proc. of the German Conf. on Pattern Recognition (GCPR)*, 2014.

[22] H. Strasdat, A. Davison, J. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2011, pp. 2352–2359.

[23] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[24] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 130–142, 2008.

[25] R. Zlot and M. Bosse, "Efficient large-scale three-dimensional mobile mapping for underground mines," *Journal of Field Robotics*, vol. 31, no. 5, pp. 758–779, 2014.

[26] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard, "Highly accurate 3D surface models by sparse surface adjustment," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 751–757.

[27] I. T. Butt and N. M. Rajpoot, "Multilateral filtering: a novel framework for generic similarity-based image denoising," in *Proc. of the IEEE Int. Conf. on Image Processing (ICIP)*, 2009, pp. 2945–2948.

[28] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: a general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 3607–3613.

[29] M. Cummins and P. Newman, "FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model," in *27th Intl Conf. on Machine Learning (ICML2010)*, 2010.

[30] A. Censi, "An accurate closed-form estimate of ICP's covariance," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 3167–3172.

[31] S. May, D. Droeschel, D. Holz, S. Fuchs, E. Malis, A. Nüchter, and J. Hertzberg, "Three-dimensional mapping with time-of-flight cameras," *Journal of Field Robotics, Special Issue on Three-Dimensional Mapping, Part 2*, vol. 26, no. 11-12, pp. 934–965, 2009.

[32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.