Journal of Visual Communication and Image Representation 25(1):137-147, Springer, January 2014, corrected Table 3 & Sec 6.2.

Multi-Resolution Surfel Maps for Efficient Dense 3D Modeling and Tracking

Jörg Stückler and Sven Behnke

Autonomous Intelligent Systems, Computer Science Institute VI, University of Bonn, Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
{*stueckler, behnke*}*@ais.uni-bonn.de*

Abstract

Building consistent models of objects and scenes from moving sensors is an important prerequisite for many recognition, manipulation, and navigation tasks. Our approach integrates color and depth measurements seamlessly in a multi-resolution map representation. We process image sequences from RGB-D cameras and consider their typical noise properties. In order to align the images, we register view-based maps efficiently on a CPU using multi-resolution strategies. For simultaneous localization and mapping (SLAM), we determine the motion of the camera by registering maps of key views and optimize the trajectory in a probabilistic framework. We create object models and map indoor scenes using our SLAM approach which includes randomized loop closing to avoid drift. Camera motion relative to the acquired models is then tracked in real-time based on our registration method. We benchmark our method on publicly available RGB-D datasets, demonstrate accuracy, efficiency, and robustness of our method, and compare it with state-of-the-art approaches. We also report on several successful public demonstrations where it was used in mobile manipulation tasks.

Keywords: RGB-D image registration, simultaneous localization and mapping, object modeling, pose tracking

1. Introduction

Robots performing tasks in unstructured environments must estimate their pose in reference to objects and parts of the environment. They require such perception to either manipulate objects, navigate to a goal, or

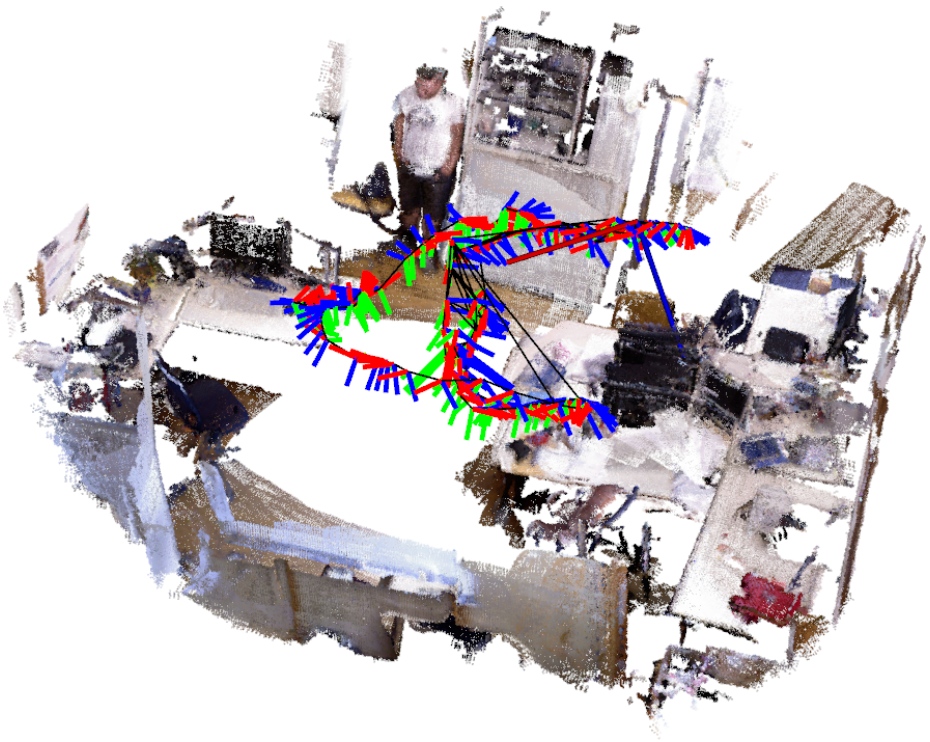

Figure 1: Map and trajectory acquired with our approach (on the freiburg1_room RGB-D sequence [1]). We represent RGB-D images by color and shape distributions within voxels (denoted as surfels) at multiple resolutions and estimate camera motion between images through efficient registration of our representation. To recover dense structure and motion, key views (poses visualized by coordinate frames) are extracted along the camera trajectory and globally aligned using pose graph optimization. The key view maps are visualized by 15 samples from each surfel at 5 cm resolution.

understand the actions of other agents. Robots should act immediately and fluently—especially dynamic tasks require fast perception.

In this article, we propose a method for efficient 3D perception of objects and indoor scenes using RGB-D cameras. We learn dense 3D models and track the pose of the camera with respect to these models in real-time at high frame rate. Our method is efficient enough to run on CPUs—in contrast to many approaches that strongly rely on massive parallel computation on GPUs that may not be available on light-weight or low-cost robot platforms.

We contribute a novel approach to 3D modeling and tracking that is based

on an efficient yet robust probabilistic registration method[1]. We represent RGB-D images in 3D multi-resolution octree maps that model shape and texture distributions within voxels. We propose a registration method that is suitable for aligning maps generated from single images as well as maps that aggregate multiple views, for instance, to create multi-view models of objects. This fast and accurate registration method is utilized to find spatial constraints between key views in a simultaneous localization and mapping (SLAM) framework to map objects and indoor scenes. Our SLAM approach constructs a graph of spatial constraints and optimizes the joint likelihood of the view poses. To this end, we also assess the uncertainty of the registration estimates from the measurement uncertainty maintained in our maps. For incremental mapping, we incorporate randomized loop-closing to find new spatial constraints efficiently. The acquired models can then be used for tracking the camera motion in real-time on a CPU. By the multi-resolution nature of our maps, our method keeps track of objects in a wide range of distances and speeds.

We evaluate the accuracy of our registration and SLAM approaches on publicly available RGB-D benchmarks and compare them with state-of-the-art methods. We also measure the robustness, accuracy, and efficiency of our object tracking approach. Finally, we report on several successful public demonstrations of our method in mobile manipulation tasks.

2. Related Work

Scene modeling has long been investigated in the computer vision and robotics communities. Early work on simultaneous localization and mapping (SLAM) in robotics has focused on acquiring 2D maps with mobile robots using range sensors such as laser scanners and sonars (e.g. [2]). Over the last decade, some approaches have been proposed that estimate the 6 degree-of-freedom (DoF) trajectory of a robot and a 3D map by means of 3D scan registration [3, 4, 5]. Scans are frequently registered by derivatives of the Iterative Closest Points (ICP) algorithm [6]. ICP methods operate on raw points directly and typically require subsampling to achieve high frame rates. Generalized-ICP [5] unifies the ICP formulation for various error metrics such as point-to-point, point-to-plane, and plane-to-plane. We compress

[1]Our framework is available open-source from http://code.google.com/p/mrsmap

image content into RGB-D statistics in voxels at multiple resolutions and register voxel contents between images.

Our registration approach bears similarities to scan representation and registration using the 3D Normal Distribution Transform (3D-NDT) [7]. The NDT discretizes point clouds in a regular grid, in which each cell contains the mean of points as well as their covariance. Accurate and robust matching of 3D scans is then achieved through minimizing the matching likelihood between NDTs of scans. We propose novel methods to increase accuracy and to enable high frame-rate operation for RGB-D images: Our approach exploits measurement principles of RGB-D sensors to rapidly extract multi-resolution surfel maps from images. To register maps efficiently, we associate surfels at multiple resolutions concurrently, realizing implicitly a coarse-to-fine registration scheme while taking advantage of the highest resolution available in both maps. While 3D-NDT also supports a multi-resolution representation of 3D scans, the 3D-NDT registration process optimizes from coarse to fine resolutions, only considering a single resolution at a time. Furthermore, our association strategy searches for matches in local neighborhoods whereas NDT only performs single direct look-ups in the grid. Thirdly, our approach utilizes shape and texture features to judge the compatibility between surfels. Our highly efficient implementation registers 640×480 RGB-D images at a frame rate of about 15 Hz on a CPU.

In computer vision, many approaches to Structure from Motion (SfM) are based on the extraction and matching of keypoints between images. Stereo vision is frequently used to directly obtain depth measurements for keypoints [8, 9, 10]. Efficient RANSAC methods can then be applied to estimate the motion of the camera rig. This approach similarly applies if depth measurements are available from time-of-flight or structured-light RGB-D cameras [11, 12].

MonoSLAM [13], based on Extended Kalman Filters, was one of the first methods that demonstrated feature-based on-line SLAM in real-time with a single camera. More recently, Klein and Murray [14] proposed a real-time capable bundle-adjustment method within small workspaces. Current work on SfM in computer vision also includes real-time dense surface reconstruction from monocular videos [15, 16]. Newcombe et al. [16] proposed DTAM, an impressive method for dense tracking and mapping of small workspaces that is real-time capable on GPU. It acquires dense models of key frames which could be globally aligned into a dense model of the scene using view-based dense SLAM methods such as our approach.

In recent years, affordable depth cameras such as time-of-flight or structured-light cameras (e.g. Microsoft Kinect, Asus Xtion) have become available. Paired with the developments in computer vision on real-time dense depth estimation from monocular image sequences, exploiting dense depth for robotic perception is now a viable option. The premise of exploiting dense depth is to increase the robustness of image registration over the use of sparse interest points from only textured image regions. It should be most beneficial in textureless environments that have geometric structure. Efficient means have to be developed, however, to take full advantage of the high frame rates and high-resolution images provided by such sensors. Steinbruecker et al. [17] recently proposed a method for dense real-time registration of RGB-D images. They model the perspective warp between images through view-pose changes and optimize for the best pose that explains the difference in intensity. In our approach, we construct 3D representations of the images and optimize for the relative pose between them. Note that our registration method is more general, since our representation supports data fusion from multiple view points. Hence, we also employ it for the registration of images to maps that aggregate multiple views, e.g., for tracking multi-view object models.

Closely related to our setting is KinectFusion, proposed by Newcombe et al. [18]. They incrementally register depth images to a map that is aggregated from previous images and demonstrate remarkable performance for small workspaces. The approach is applied for augmented reality user interfaces and supports the tracking of the pose of objects and the camera in real-time. Since KinectFusion is implemented on GPU, it has—due to memory restrictions—stronger workspace limitation than CPU-based implementations like ours. In order to scale to larger workspaces, KinectFusion has been extended using moving volumes [19, 20]. Due to their incremental nature, these approaches still accumulate minor drift in the map estimate over time [20] when the camera is swept into previously unseen areas. This effect could be corrected through loop-closing like in our view-based SLAM approach. For this, local submaps have to be built and eventually to be registered in a submap-based SLAM framework. Our framework supports a compact representation of local submaps and registers individual RGB-D images as well as entire local submaps that summarize many images. We detect loop closures and find a best alignment of key views by jointly optimizing spatial constraints between views. We determine the relative pose between views using our registration method and assess the uncertainty of

the pose estimate.

Some approaches have been proposed that also learn maps from depth and RGB-D images in a trajectory optimization framework [21, 22, 23]. May et al. [21] match time-of-flight depth images using ICP and apply global relaxation over all images to build a consistent 3D map. Henry et al. [22] extract textured surface patches, register them using the ICP algorithm to the model, and apply graph optimization to obtain an accurate map. Our approach provides shape-texture information in a compact representation that supports pose tracking from a wide range of distances, since the model is represented at multiple scales. Endres et al. [24] match point features from the RGB image between frames and refine the registration estimate using ICP. Our registration method incorporates shape and texture seamlessly and is also applicable to textureless shapes.

The modeling of the geometry of objects from multiple views is a traditional research topic in robotics and computer graphics. A diverse set of applications exists for such explicit geometric map representations like, for instance, object recognition or manipulation planning.

One early work of Chen and Medioni [25] registers several range images using an iterative least squares method. In order to acquire full-view object models, the authors propose to take four to eight views onto the object. Each view is then registered to a map that is aggregated from the precedingly registered views. If the content and sequence of scans is chosen carefully to include significant overlap with the already acquired map, this procedure accumulates less error than pair-wise registration of successive views. Weise et al. [26] match surface patches between range images and align them globally to reconstruct 3D object models. Krainin et al. [27] learn models of objects with an approach similar to Henry et al. [22]. Again, our map representation includes shape and texture seamlessly and inherently supports tracking from a wide range of distances due to its multi-scale structure.

3. Multi-Resolution Surfel Maps

3.1. Map Representation

We represent joint color and shape distributions at multiple resolutions in a probabilistic map. We use octrees as the natural data structure to represent 3D space at multiple resolutions. In each node of the tree, i.e., inner nodes as well as leaf nodes across all resolutions (voxel sizes), we store statistics on the joint spatial and color distribution of the points $\mathcal{P}$ within

its volume which we denote as surfel. The distribution is approximated with sample mean μ and covariance Σ of the data, i.e., we model the data as normally distributed in a node's volume. Instead of computing mean and covariance in the nodes with a two-pass algorithm, we use a single-pass update scheme with high numerical accuracy [28]. It determines the sufficient statistics $\mathcal{S}(\mathcal{P}) := \sum_{p\in\mathcal{P}} p$ and $\mathcal{S}^2(\mathcal{P}) := \sum_{p\in\mathcal{P}} pp^T$ of the normal distribution from the statistics of two point sets $\mathcal{P}^A$ and $\mathcal{P}^B$ through

$$\begin{aligned} \mathcal{S}(\mathcal{P}^{A\cup B}) &\leftarrow \mathcal{S}(\mathcal{P}^A) + \mathcal{S}(\mathcal{P}^B), \\ \mathcal{S}^2(\mathcal{P}^{A\cup B}) &\leftarrow \mathcal{S}^2(\mathcal{P}^A) + \mathcal{S}^2(\mathcal{P}^B) + \frac{\delta\delta^T}{N_A N_B (N_A + N_B)}, \end{aligned} \tag{1}$$

where $N_{(\cdot)} := |\mathcal{P}^{(\cdot)}|$ and $\delta := N_B\mathcal{S}(\mathcal{P}^A) - N_A\mathcal{S}(\mathcal{P}^B)$. From these, we obtain sample mean $\mu(\mathcal{P}) = \frac{1}{|\mathcal{P}|}\mathcal{S}(\mathcal{P})$ and covariance $\Sigma(\mathcal{P}) = \frac{1}{|\mathcal{P}|-1}\mathcal{S}^2(\mathcal{P}) - \mu\mu^T$.

Careful treatment of the numerical stability is required when utilizing one-pass schemes for calculating the sample covariance [28]. We require a minimum sample size of $|\mathcal{P}| \geq 10$ to create surfels and stop incorporating new data points if $|\mathcal{P}| \geq 10,000$[2]. The discretization of disparity and color produced by the RGB-D sensor may cause degenerate sample covariances, which we robustly detect by thresholding the determinant of the covariance at a small constant.

We not only represent the local surface geometry by the distribution of 3D point coordinates in the nodes, but also model the spatial distribution of color by maintaining the joint distribution of point coordinates and color in a 6D normal distribution. In order to separate chrominance from luminance information, we choose a variant of the HSL color space. We define the L$\alpha\beta$ color space as $L := \frac{1}{2}(\max\{R,G,B\} + \min\{R,G,B\})$, $\alpha := R - \frac{1}{2}G - \frac{1}{2}B$, and $\beta := \frac{\sqrt{3}}{2}(G-B)$. The chrominances α and β represent hue and saturation of the color and L represents its luminance.

Since we build maps of scenes and objects from all perspectives, multiple distinct surfaces may be contained within a node's volume. We model this by maintaining multiple surfels in a node that are visible from several view

[2]Using double precision (machine epsilon $2.2\cdot10^{-16}$) and assuming a minimum standard deviation of 10^{-4} in $\mathcal{P}$, and reasonable map sizes (maximal radius smaller than 10^2 m), we obtain a theoretical bound for the relative accuracy of the covariance entries in the order of 10^{-6} at 10^4 samples [29]. More accurate but slower two-pass schemes could be used for extremely large map or sample sizes, or smaller noise.

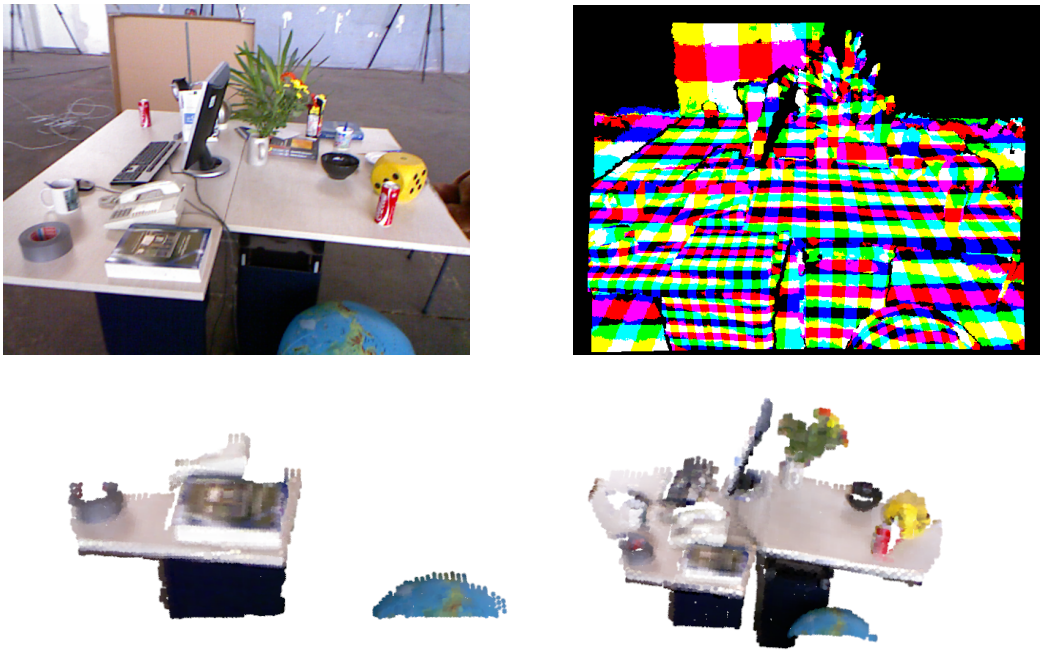

Figure 2: Multi-resolution surfel map aggregation from an RGB-D image. Top left: RGB image of the scene. Top right: Maximum node resolution coding, color codes octant of the leaf in its parent's node (max. resolution 0.0125 m, see text for details). Bottom: 15 samples per color and shape surfel at 0.025 m (left) and at 0.05 m resolution (right).

directions. We use up to six orthogonal view directions aligned with the basis vectors e_1, e_2, e_3 of the map reference frame. When adding a new point to the map, we determine the view direction onto the point and associate it with the surfel belonging to the most similar view direction.

3.2. Shape-Texture Descriptor

We construct descriptors of shape and texture in the local neighborhood of each surfel (see Fig. 3). Similar to FPFH features [30], we first build three-bin histograms $h_{sh}(s)$ of the three angular surfel-pair relations between the query surfel s and its up to 26 neighbors s' at the same resolution. The three angles are measured between the normals of both surfels $\measuredangle(n, n')$ and between each normal and the line $\Delta\mu := \mu - \mu'$ between the surfel means, i.e., $\measuredangle(n, \Delta\mu)$ and $\measuredangle(n', \Delta\mu)$. Each surfel-pair relation is weighted with the number of points in the neighboring node. We smooth the histograms to better cope with discretization effects by adding the histogram of neighboring surfels

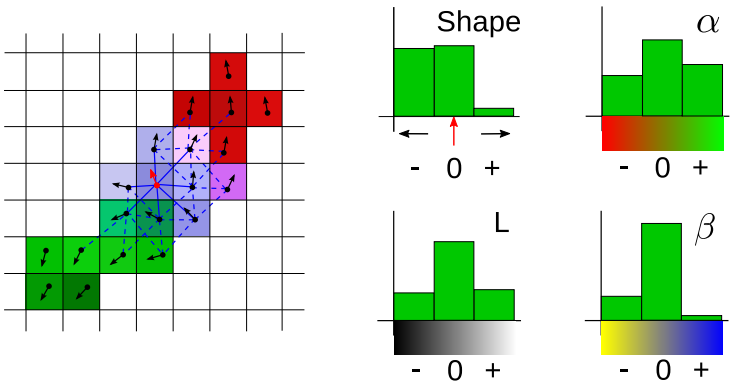

Figure 3: 2D illustration of our local shape-texture descriptor. We determine a local description of shape, chrominance (α, β), and luminance (L) contrasts to improve the association of surfels. Each node is compared to its 26 neighbors. We subsequently smooth the descriptors between neighbors. See Sec. 3.2 for further details.

with a factor $\gamma = 0.1$ and normalize the histograms by the total number of points.

Similarly, we extract local histograms of luminance $h_L(s)$ and chrominance $h_\alpha(s)$, $h_\beta(s)$ contrasts. We bin luminance and chrominance differences between neighboring surfels into positive, negative, or insignificant. Note, that neighboring voxels can efficiently be found using precalculated look-up tables [31]. We store the pointers to neighbors explicitly in each node to achieve better run-time efficiency than tracing the neighbors through the tree. The octree representation is still more memory-efficient than a multi-resolution grid, as it only represents the 3D surface observed by the sensor.

3.3. Efficient RGB-D Image Aggregation

The use of sufficient statistics allows for an efficient incremental update of the map. In the simplest implementation, the sufficient statistics of each point is added individually to the tree. Starting at the root node, the sufficient statistics is recursively added to the nodes that contain the point in their volume.

Adding each point individually is, however, not the most efficient way to generate the map. Instead, we exploit that by the projective nature of the camera, neighboring pixels in the image project to nearby points on the sampled 3D surface—up to occlusion effects. This means that neighbors in the image are likely to belong to the same octree nodes.

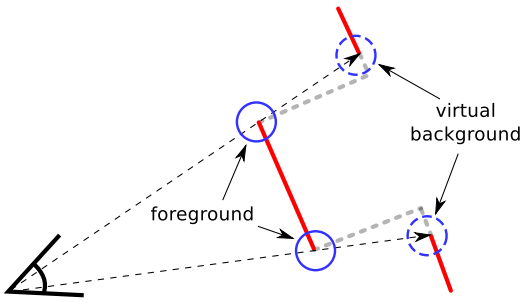

Figure 4: 2D illustration of occlusion handling. We exclude surfels from the registration process that receive background points at depth discontinuities. At such points, the visibility of the structure in the background changes with the view point, causing distorted pose estimates if not excluded.

We further consider the typical property of RGB-D sensors that noise increases quadratically with depth. We thus adapt the maximum octree node resolution at a pixel to the squared distance from the sensor. In effect, the size of the octree is significantly reduced and the leaf nodes subsume local patches in the image (see top-right of Fig. 2). We exploit these properties and scan the image to aggregate the sufficient statistics of contiguous image regions that belong to the same octree node. This measurement aggregation allows to construct the map with only several thousand insertions of node aggregates for a 640×480 image in contrast to 307,200 point insertions.

After the image content has been incorporated into the representation, we precompute mean, covariance, surface normals, and shape-texture features for later registration purposes.

3.4. Handling of Image and Virtual Borders

Special care must be taken at the borders of the image and at virtual borders where background is occluded (see Fig. 4). Nodes that receive such border points only partially observe the underlying surface structure. When updated with these partial measurements, the surfel distribution is distorted towards the visible points. In order to avoid this, we determine such nodes by scanning efficiently through the image, and neglect them.

In contrast, foreground depth edges describe the valuable geometric contour of objects. We distinguish between surfels on contours which receive foreground points at depth discontinuities, and surfels that don't receive such points. During registration, we use this property as an additional feature cue

and only match surfels that are either both on a contour or both not on a contour.

4. Robust Efficient Registration of Multi-Resolution Surfel Maps

The registration of multi-resolution surfel maps requires two main steps that are iterated multiple times and both need to be addressed efficiently: First, we robustly associate surfels between the maps. For these associations, we then determine a transformation that maximizes their matching likelihood.

4.1. Multi-Resolution Surfel Association

Starting at the finest resolution, we iterate through each node in the current resolution and establish associations between the surfels in each view direction. In order to choose the finest resolution possible, we do not associate a node, if one of its children already has been associated, saving redundant matches on coarse resolutions. We have to iterate data association and pose optimization multiple times, hence we gain efficiency by bootstrapping the association process from previous iterations. If a surfel has not been associated in the previous iteration, we search for a matching surfel in the target map. The surfel mean is transformed with the current pose estimate x and an efficient cubic volume query is performed in the target map for which the side length of the cube is set to twice the resolution of the query surfel's node. In this way, the resolution at which misalignments are corrected is adapted from coarse to fine resolutions inherently during the registration process. The view direction of the query surfel is also rotated according to x to select the surfel for the corresponding view direction in the target map.

If an association from a previous iteration exists, we associate the surfel with the best matching surfel within the direct neighbors of the last associated surfel's node. Since we precalculate the 26-neighborhood of each octree node, this look-up amounts to only constant time. We accept associations only, if the shape-texture descriptors of both surfels match which we evaluate by thresholding on the Euclidean distance $d_f(s_i, s_j) \leq \tau$ with $\tau = 0.1$ between the shape and texture descriptor, where $d_f(s_i, s_j) := \sum_{c \in \{sh, L, \alpha, \beta\}} d_c(s_i, s_j)$. No weighting between the descriptor components has been necessary in our implementation. Within a resolution, surfels are processed independently which allows the load within a resolution to be distributed over multiple CPU cores.

4.2. Observation Model

Our goal is to register an RGB-D image z, from which we construct the source map m_s, towards a target map m_m. We formulate our problem as finding the most likely pose x that maximizes the likelihood $p(z \mid x, m_m)$ of observing the current image z in the target map. We choose to represent poses $x = (q, t)^T$ by translations $t \in \mathbb{R}^3$ and by unit quaternions q for a compact representation of the rotational part without singularities.

We construct the source map m_s from the image z and determine the observation likelihood between source and target map,

$$p(m_s \mid x, m_m) = \prod_{(i,j)\in\mathcal{A}} p(s_{s,i} \mid x, s_{m,j}), \tag{2}$$

where $\mathcal{A}$ is the set of surfel associations between the maps, and $s_{s,i} = (\mu_{s,i}, \Sigma_{s,i})$, $s_{m,j} = (\mu_{m,j}, \Sigma_{m,j})$ are associated surfels. The observation likelihood of a surfel match is the difference of the surfels under their normal distributions,

$$\begin{aligned} p(s_{s,i} \mid x, s_{m,j}) &= \mathcal{N}\left(d_{i,j}(x); 0, \Sigma_{i,j}(x)\right), \\ d_{i,j}(x) &:= \mu_{m,j} - T(x)\mu_{s,i}, \\ \Sigma_{i,j}(x) &:= \Sigma_{m,j} + R(x)\Sigma_{s,i}R(x)^T, \end{aligned} \tag{3}$$

where $T(x)$ is the homogeneous transformation matrix for the pose estimate x and $R(x)$ is its rotation matrix. We marginalize the surfel distributions for the spatial dimensions.

Note that due to the difference in view poses between the images, the scene content is discretized differently between the maps. We compensate for inaccuracies due to discretization effects by trilinear interpolation. This is possible, if a scene surfel $s_{s,i}$ is directly associated with the model surfel $s_{m,j}$ in the octree node at the projected position of the scene surfel $T(x)\mu_{s,i}$. Instead of directly using the associated model surfel $s_{m,j}$ in the observation likelihood (Eq. (3)), we determine mean and covariance of the model surfel at the projected position $T(x)\mu_{s,i}$ through trilinear interpolation of neighboring surfels in the model map.

4.3. Pose Optimization

We optimize the logarithm of the observation likelihood (Eq. (3))

$$L(x) = \sum_{a\in\mathcal{A}} \log(|\Sigma_a(x)|) + d_a^T(x)\Sigma_a^{-1}(x)d_a(x) \tag{4}$$

for the pose x in two stages: We apply fast approximate Levenberg-Marquardt (LM) optimization to initialize fine registration using Newton's method.

The LM method is suitable for weighted non-linear least squares problems of the form $\arg\max_x e^T(x)We(x)$, where $e(x) = y - f(x)$ is a vector of residuals and W is a weighting matrix. In our case, we stack the residuals between associated surfels $e(x) = (d_a(x))_{a\in\mathcal{A}}$ and neglect the effect of the pose on the covariance to obtain a constant block-diagonal weighting matrix $W = \operatorname{diag}\left(\{w_a\Sigma_a^{-1}(x)\}_{a\in\mathcal{A}}\right)$. We weight each surfel match also with the similarity $w_a := \tau - d_f(a)$ of the shape-texture descriptors. LM optimization now performs damped Gauss-Newton steps

$$x' \leftarrow x + (J^TWJ + \lambda I)^{-1}J^TWe(x), \tag{5}$$

where $J := (J_a(x))$ is the Jacobian stacked from individual Jacobians $J_a = \frac{dT}{dx}(x)\mu_{s,a}$ per surfel association. Note that due to the block-diagonal structure of W, this update decomposes into simple sums over terms per association, i.e., $J^TWJ = \sum_{a\in\mathcal{A}} J_a^TW_aJ_a$ and $J^TWe(x) = \sum_{a\in\mathcal{A}} J_a^TW_ad_a$. During the LM optimization, we do not use trilinear interpolation of surfels and update the association of surfels only, if the pose changes of the LM method converged below a threshold. We stop iterating the LM method, if the pose still not changes after surfel association, or a maximum number of iterations is reached. Afterwards, we fine-tune the registration result by Newton's method directly on the observation log-likelihood (Eq. (4)) with trilinear interpolation and surfel association in each step. While Newton's method requires second-order derivatives, they can be efficiently calculated analytically due to the simple form of the observation log likelihood in Eq. (4). Our approach typically converges within 10-20 iterations of LM and 5 iterations of Newton's method to a precise estimate. We parallelize the evaluation of the first- and second-order derivatives over surfels which yields a significant speed-up on multi-core CPUs.

The normalization constraint on the quaternion part of our pose representation requires special handling during the optimization. We incorporate the normalization by only optimizing for a compact 3-dimensional quaternion representation that consists of the coefficients of the imaginary parts. The real part of the quaternion can be recovered from the normalization constraint and its initial sign before the optimization. This approach alone would only be valid for angular misalignments below 180°. To allow for arbitrary angular misalignments, we compose the current pose estimate x from a

constant part x' and a subsequent pose change Δx, i.e., $T(x) = T(\Delta x)T(x')$. In each iteration of LM or Newton's method, we set $x' = x$ and optimize for Δx instead.

4.4. Estimation of Pose Uncertainty

We obtain an estimate of the observation covariance using a closed-form approximation [32]:

$$\Sigma(x) \approx \left(\frac{\partial^2 L}{\partial x^2}\right)^{-1} \frac{\partial^2 L}{\partial s \partial x} \Sigma(s) \frac{\partial^2 L}{\partial s \partial x}^T \left(\frac{\partial^2 L}{\partial x^2}\right)^{-1}, \tag{6}$$

where x is the pose estimate, s denotes the associated surfels in both maps, and $\Sigma(s)$ is the covariance of the surfels. The covariance estimate of the relative pose between the maps captures uncertainty along unobservable dimensions, for instance, if the maps view a planar surface.

5. Model Learning and Tracking

Our image representation and registration techniques now provide tools for learning models of objects or indoor scenes and for tracking them. Models are acquired by moving an RGB-D camera around an object or through a scene. Our goal is then to estimate the motion of the camera in order to overlay images into a consistent 3D model of the object or the scene. Once the model is available, it can be tracked in the live images of the camera using our registration approach.

5.1. Model Learning

Naïve sequential registration of images, i.e., visual odometry, would be prone to drift. Instead, we register images towards a reference key view to keep track of the camera motion. Since registration quality degrades with the view pose difference between images, a new key view is generated from the current image if the camera moved sufficiently far, and the new key view is set as the reference for further tracking. The registration result x_i^j between a new key view v_i and its reference v_j is a spatial constraint that we maintain as values of edges $e_{ij} \in \mathcal{E}$ in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of key views (see Fig. 5).

We find additional spatial constraints between similar key views using our registration method in a randomized hypothesis-and-test scheme. This scheme tests one spatial constraint per frame to enable on-line operation

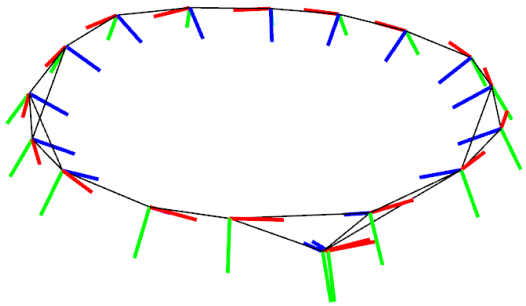

Figure 5: Exemplary key view graph. Key views (poses visualized by coordinate frames) are extracted along the camera trajectory after sufficient motion and spatial constraints between key views (black lines) are established using our registration method.

and also detects loop-closures. The poses of the key views are optimized in a probabilistic pose graph optimization framework, which also supports on-line operation by iterating graph optimization once per frame.

Finally, we fuse the key views by overlaying them in one multi-view map from their optimized pose estimates.

5.1.1. Constraint Detection

On each frame, we check for one new constraint between the current reference v_{ref} and other key views v_{cmp}. We determine a probability

$$p_{\text{chk}}(v_{\text{cmp}}) = \mathcal{N}\left(d(x_{\text{ref}}, x_{\text{cmp}}); 0, \sigma_d^2\right) \cdot \mathcal{N}\left(|\alpha(x_{\text{ref}}, x_{\text{cmp}})|\,; 0, \sigma_\alpha^2\right) \qquad (7)$$

that depends on the linear and rotational distances $d(x_{\text{ref}}, x_{\text{cmp}})$ and $|\alpha(x_{\text{ref}}, x_{\text{cmp}})|$ of the key view poses x_{ref} and x_{cmp}, respectively. We sample a key view v according to $p_{\text{chk}}(v)$ and determine a spatial constraint between the key views using our registration method.

It is important to validate the matching of key views, since if the key views barely or even not overlap our registration method may find suboptimal solutions that—if included—could let the pose graph optimization diverge. We examine the matching likelihood of the registration estimate for validation. For each surfel in one of the key views, we find the best matching surfel in the second view. Here, we directly take into account the consistency of surface normals and therefore determine the matching likelihood of the surfels as the product of the likelihood under their distributions and under a normal distribution in the angle between their normals. We assign a minimum

likelihood to all surfels with a worse match or without a match. In this way, the matching likelihood accounts for the overlap between the views. This likelihood is directional and, hence, we evaluate it in both directions.

We cannot use a global threshold for deciding if a constraint should be added, as the matching likelihood depends on the image content. Instead we require the matching likelihood of a new constraint to be at least a fraction of the matching likelihood for the initial constraint of the key view. This constraint has been established through tracking of the reference key view and is thus assumed to be consistent.

5.1.2. Pose Graph Optimization

Our probabilistic registration method provides a mean x_i^j and covariance estimate $\Sigma(x_i^j)$ for each spatial constraint e_{ij}. We obtain the likelihood of the relative pose observation $z = (x_i^j, \Sigma(x_i^j))$ of the key view j from view i by

$$p(x_i^j \mid x_i, x_j) = \mathcal{N}\left(x_i^j; \Delta(x_i, x_j), \Sigma(x_i^j)\right), \tag{8}$$

where $\Delta(x_i, x_j)$ denotes the relative pose between the key views for their current estimates x_i and x_j.

From the graph of spatial constraints, we infer the probability of the trajectory estimate given all relative pose observations

$$p(\mathcal{V} \mid \mathcal{E}) \propto \prod_{e_{ij} \in \mathcal{E}} p(x_i^j \mid x_i, x_j). \tag{9}$$

We solve this graph optimization problem by sparse Cholesky decomposition within the g^{2}o framework [33].

5.1.3. Key View Fusion to Maps or Object Models

We fuse the pose-optimized key views by overlaying them in one multi-view multi-resolution surfel map.

For creating object models, we include only measurements within a volume of interest which is defined by a user or may also be provided by image segmentation approaches. In our implementation, the user selects points in one image of the sequence to form a convex hull on the support plane for the footprint of the object. The user then specifies the height above the support of the object.

5.2. Pose Tracking

We apply our registration method to estimate the pose of the camera with respect to a model. We aggregate the current RGB-D image in a multi-resolution surfel map and register it to the map. During object tracking we save unnecessary computations by processing the image in a volume of interest close to the last pose estimate. We only process image points that are likely under the spatial distribution of the object model. Mean and covariance of this distribution are readily obtained from the sum of surfel statistics $|\mathcal{P}|$, $\mathcal{S}(\mathcal{P})$, and $\mathcal{S}^2(\mathcal{P})$ over all view directions in the root node of the tree.

6. Experiments

We evaluate our approaches on publicly available RGB-D datasets[3]. The RGB-D benchmark dataset by Sturm et al. [1] is used to assess image registration and RGB-D SLAM in indoor scenes, for which we restrict our evaluation to sequences of static scenes. In addition, we generated a complementary RGB-D dataset for the evaluation of object tracking and modeling.

Both datasets contain RGB-D image sequences with ground truth information for the camera pose which is measured using external optical motion capture systems. Our object dataset includes sequences of three objects of different sizes (a chair, a textured box, and a small humanoid robot). For model reconstruction, we recorded each object from a 360° trajectory. Three test sets are provided for evaluating tracking performance with slow, medium, and fast camera motion for each object. Each sequence consists of 1,000 frames recorded at 30 Hz and VGA (640×480) resolution. We set the maximum resolution of our maps to 0.0125 m throughout the experiments which is a reasonable lower limit with respect to the minimum measurement range of the sensor (ca. 0.4 m). We evaluate timings of all methods on a notebook PC with an Intel Core i7 3610QM 2.3 GHz (max. 3.3 GHz) QuadCore CPU using full resolution (VGA) images. Resulting trajectories are assessed using the absolute trajectory error (ATE) and relative pose error (RPE) metrics as proposed in [1].

Table 1: Comparison of median (max.) relative pose error (RPE) in mm for incremental registration on RGB-D sequences of the Freiburg benchmark dataset [1].

sequence	ours	warp [17] (OpenCV)	GICP [5]	3D-NDT [7]	fovis [12]
fr1 360	**5.1** (46.4)	5.9 (75.4)	18.8 (88.3)	7.8 (140.8)	7.1 **(43.1)**
fr1 desk	**4.4 (20.1)**	5.8 (131.8)	10.2 (54.9)	7.9 (26.6)	6.3 (34.2)
fr1 desk2	**4.5 (24.1)**	6.2 (147.4)	10.4 (261.2)	8.2 (46.3)	6.6 (49.9)
fr1 floor	4.9 (355.9)	**2.1** (3167)	5.0 **(193.6)**	6.3 (1021)	2.6 (412.4)
fr1 plant	**3.5 (27.7)**	4.2 (300.8)	16.1 (831.4)	7.4 (62.3)	4.6 (61.6)
fr1 room	**3.5 (33.3)**	4.6 (167.8)	10.2 (212.6)	6.1 (51.2)	5.4 (55.1)
fr1 rpy	**3.0 (24.8)**	5.1 (41.8)	10.4 (636.6)	6.8 (41.9)	5.4 (38.7)
fr1 teddy	**4.2 (77.4)**	6.1 (381.1)	21.3 (356.6)	8.8 (126.6)	7.1 (82.4)
fr1 xyz	**2.6 (9.8)**	4.1 (18.1)	3.9 (42.0)	5.2 (36.7)	4.6 (25.8)
fr2 desk	2.1 (15.1)	**2.1 (14.1)**	6.7 (64.4)	4.3 (31.7)	2.5 (15.5)
fr2 large no loop	21.8 **(167.4)**	20.5 (7.6e5)	21.3 (1289)	32.1 (2512)	**11.0** (173.0)
fr2 rpy	1.6 (29.9)	1.7 (189.5)	**1.3** (28.7)	4.2 (55.1)	1.7 **(11.0)**
fr2 xyz	**1.4** (33.1)	2.0 **(8.8)**	1.7 (26.8)	4.0 (18.0)	1.9 (9.9)
fr3 long office household	**2.6 (16.9)**	3.2 (34.0)	7.8 (209.1)	4.2 (40.2)	3.7 (35.6)
fr3 nostruct. notext. far	9.7 **(48.4)**	40.4 (6.0e4)	**8.6** (66.1)	13.8 (77.8)	11.3 (108.4)
fr3 nostruct. notext. near	15.2 **(56.9)**	28.2 (3.2e4)	12.5 (182.1)	17.1 (144.6)	**11.2** (79.3)
fr3 nostruct. text. far	18.5 **(57.9)**	19.2 (1230)	**10.9** (58.6)	18.6 (74.6)	20.8 (101.5)
fr3 nostruct. text. near	11.5 (52.4)	**7.0** (100.5)	8.9 (67.1)	10.6 (80.4)	7.3 **(41.6)**
fr3 struct. notext. far	**2.2 (16.6)**	8.6 (2579)	4.5 (23.2)	2.9 (19.2)	9.1 (62.4)
fr3 struct. notext. near	**2.1** (13.5)	8.6 (1108)	2.9 **(12.2)**	2.4 (44.5)	9.3 (86.9)
fr3 struct. text. far	5.5 (20.7)	8.1 (39.0)	7.1 (23.3)	**5.4 (20.1)**	8.8 (45.2)
fr3 struct. text. near	**3.2** (14.7)	5.9 (34.8)	5.6 **(34.2)**	5.5 (82.3)	6.5 (38.2)

Table 2: Comparison of average (std. dev.) runtime in milliseconds for incremental registration on RGB-D benchmark sequences.

sequence	ours	warp [17] (OpenCV)	GICP [5]	3D-NDT [7]	fovis [12]
fr1 desk	75.15 (9.66)	108.64 (18.41)	4015.4 (1891.6)	414.87 (255.57)	**15.98 (7.00)**
fr2 desk	61.47 (7.68)	99.29 (10.61)	3147.3 (852.98)	892.59 (253.54)	**12.98 (2.27)**

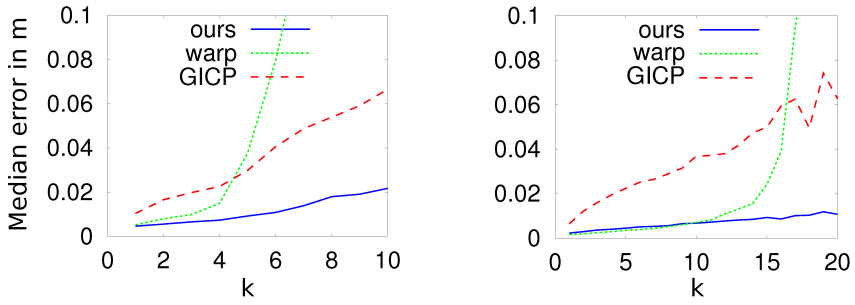

Figure 6: Median translational error of the pose estimate for different frame skips k on the freiburg1_desk (left) and freiburg2_desk (right) sequences (warp [17], GICP [5])

6.1. Incremental Registration

We first evaluate the properties of our registration method that underlies our modeling and tracking approaches. Table 1 compares median translational drift of our approach with several other registration methods. On many sequences, our approach outperforms the other methods, while also yielding only moderate maximum drift in most sequences. Our method is very competitive to other dense registration approaches such as warp [17], GICP [5], and 3D-NDT [7]. For these experiments, we used a reimplementation of warp contained in the OpenCV library with default settings but 14 iterations in the coarsest resolution and a maximum depth difference of 0.5 m. The maximum distance for matches in GICP has been chosen as $d_{\max} = 0.2\,m$. NDT has been configured to use the four scales 0.05 m, 0.1 m, 0.2 m, and 0.4 m. Higher resolutions were not possible due to memory limitations.

Difficult scenes contain only little geometric structure but fine-grained texture such as the freiburg1_floor or the freiburg3_nostructure sequences. In the freiburg2_large_no_loop sequence, the camera measures mostly distant parts in the environment, for which geometric features are barely measurable due to sensor noise and discretization of disparity. In three cases, fovis performs better if texture is available, which indicates that point feature-based registration would complement our approach well. Our approach achieves about 15 Hz on the sequences, is much more efficient than GICP or 3D-NDT, and demonstrates slightly faster run-time than warp (see Table 2).

In Fig. 6, we evaluate the robustness of our approach for skipping frames

[3]http://cvpr.in.tum.de/data/datasets/rgbd-dataset,
http://www.ais.uni-bonn.de/download/objecttracking.html

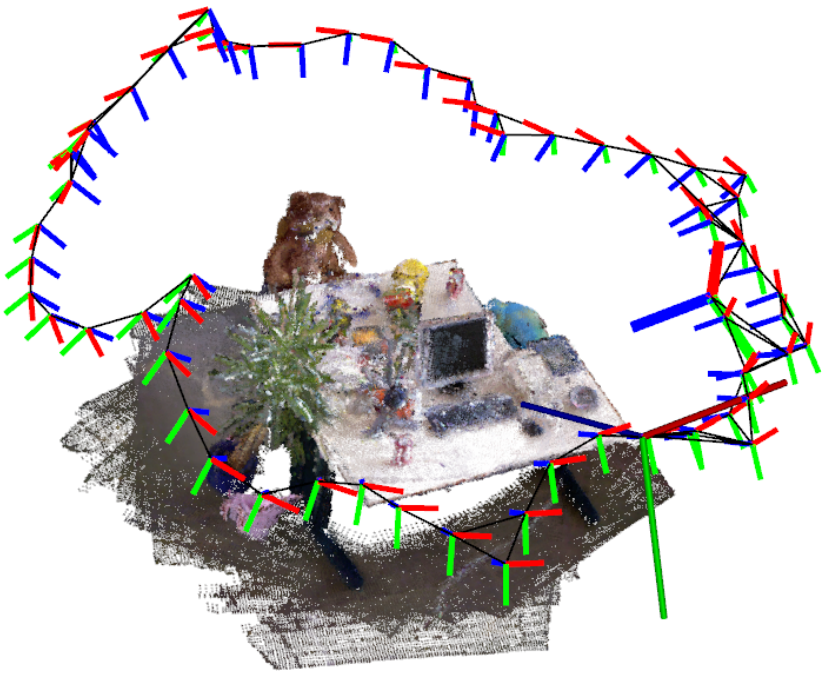

Figure 7: 3D map (5 cm resolution) visualized by samples from the color and shape surfel distributions of the aligned key frames and camera trajectory result of our approach on the freiburg2_desk sequence.

on the freiburg1_desk and freiburg2_desk sequences[4]. Our approach achieves similar accuracy than warp for small displacements, but retains the robustness of ICP methods for larger displacements when warp fails. This property is very important for real-time operation, if frames need to be dropped eventually.

6.2. Scene Reconstruction

We evaluate our SLAM approach on eleven sequences of the RGB-D benchmark dataset and compare our approach to RGB-D SLAM ([24, 34]) by measuring average RPE over all numbers of frame differences (see Table 3[5]). Fig. 7 show a typical result obtained with our approach on the freiburg2_desk sequence. The sequence contains moderately fast camera motion in a loop

[4]Results for warp and GICP taken from [17].

[5]Results have been corrected. Instead of the RMSE error, the original version of the article accidentally included the median error, which was slightly lower.

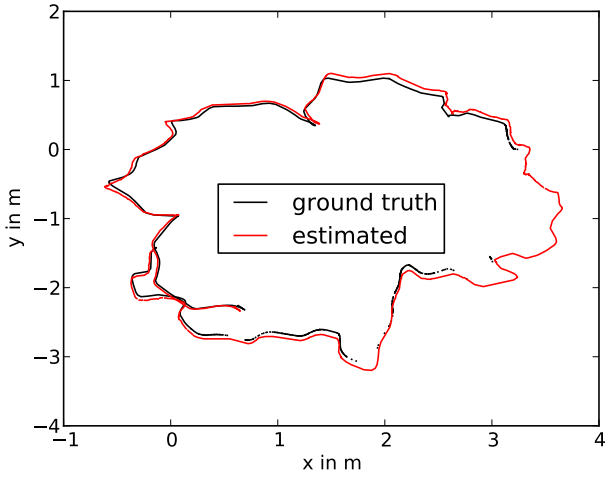

Figure 8: Projection onto the x-y-plane of ground truth and trajectory estimate of our approach on the freiburg2_desk sequence.

Table 3: Comparison of our SLAM approach with RGB-D SLAM in relative pose error (RPE).

sequence	RMSE RPE in m		
	ours all frames[5]	ours real-time[5]	RGB-D SLAM
freiburg1_360	0.123	0.126	**0.103**
freiburg1_desk2	**0.098**	0.107	0.102
freiburg1_desk	0.054	0.088	**0.049**
freiburg1_plant	**0.038**	0.062	0.142
freiburg1_room	**0.111**	0.145	0.219
freiburg1_rpy	**0.041**	0.045	0.042
freiburg1_teddy	**0.066**	0.092	0.138
freiburg1_xyz	**0.020**	0.025	0.021
freiburg2_desk	**0.100**	0.115	0.143
freiburg2_rpy	0.043	0.040	**0.026**
freiburg2_xyz	**0.032**	0.033	0.037

around a table-top setting. In Fig. 8 it can be seen that the camera trajectory recovered with our approach well aligns with the ground truth. The freiburg1_room sequence contains a trajectory loop through an office (see Fig. 1). The camera moves much faster than in the freiburg2_desk sequence. On eight[5] sequences, our method achieves better results than RGB-D SLAM if all frames are processed. For real-time operation, our SLAM approach needs to drop frames and needs to be limited to a maximum resolution of 0.05 m, but still performs very well.

Note, that our method did not succeed on sequences such as freiburg1_floor or freiburg2_large_loop. On freiburg1_floor the camera sweeps over a floor with only little texture that could be captured by the local descriptors of the surfels. Our method also cannot keep track of the camera pose, if large parts of the image contain invalid or highly uncertain depth at large distances.

The processing time for on-line operation is mainly governed by our registration method. At each image update, we have to register two pairs of views. First, we keep track of the current sensor pose by aligning the image to the closest key view in the map. Our randomized constraint detection method invokes a second registration at each image update. While the run time required for graph optimization depends on the size of the graph, one iteration of graph optimization takes up to a few milliseconds in the experiments (freiburg2_desk: median 0.79 ms, max. 4.01 ms at max. 64 key views and 138 edges).

6.3. Object Model Learning

We evaluate the accuracy of our object modeling approach by comparing trajectory estimates with ground truth using the ATE measure. It can be seen from Fig. 9 that our approach is well capable of recovering the trajectory of the camera. We provide the minimum, median, and maximum ATE of our trajectory estimates in Table 4. The median accuracy is about 1-2 cm for all sequences. It can also be seen that graph optimization significantly improves the trajectory estimate. Fig. 10 shows models learned with our approach. Typical sizes of models are ca. 54 MB for the chair and ca. 19 MB for the humanoid.

6.4. Object Tracking

In the following, we evaluate our object tracking method. The results in Table 5 demonstrate that our approach tracks the learned models with

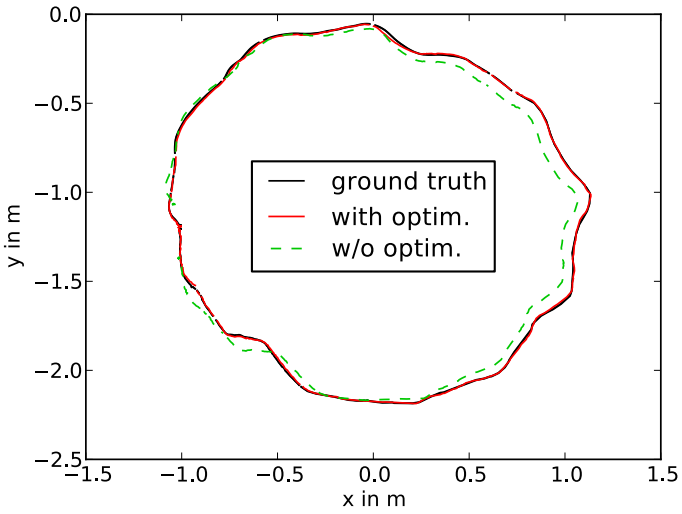

Figure 9: Ground truth (black) and trajectory estimates obtained without graph optimization (dashed green) and with graph optimization (solid red) on the box sequence.

	w/o graph optimization			with graph optimization		
sequence	min	median	max	min	median	max
humanoid	18.5	65.8	320.5	2.5	12.8	38.2
box	17.4	59.3	149.8	9.0	21.9	37.3
chair	39.1	147.5	345.5	1.6	23.8	70.8

Table 4: Absolute trajectory error in mm obtained by incremental mapping without graph optimization and with our object modeling approach (with graph optimization).

good accuracy in real-time. The tracking performance depends on distance, relative angle, and speed towards the object (see Fig. 11). For far view poses, the measured points on the object correspond to coarse resolutions in the multi-resolution representation of the image. Thus, our approach registers the image on coarse scales to the object model and the accuracy decreases while the frame rate increases compared to closer distances. The shape of the object may also influence the accuracy, such that it varies with view angle.

We have demonstrated our real-time tracking method publicly at RoboCup@Home competitions in 2011 and 2012[6]. In the Final of RoboCup 2011, our robot Cosero carried a table with a human [35] and baked omelet.

[6]Videos of the demonstrations can be found at http://www.nimbro.net/@Home.

Figure 10: Learned object models at a resolution of 2.5 cm visualized by samples from the color and shape surfel distributions. Left: humanoid, middle: box, right: chair (best viewed in color).

Cosero's main computer has been a quadcore notebook with an Intel i7-Q720 CPU during these demonstrations. To achieve high performance tracking on this CPU in parallel with the robot control processes, we subsampled the RGB-D images to QVGA (320×240) resolution. For carrying the table, we trained a model of the table. Cosero registered RGB-D images to the model to approach the table and to grasp it. It detected the lifting and lowering of the table by estimating its pitch rotation. Similarly, Cosero approached the pan on a cooking plate by tracking the object with our registration method. At RoboCup 2012, Cosero moved a chair and watered a plant. It perceived chair and watering can with the proposed method despite partial occlusions of the objects by the robot itself. The demonstrations have been well received by juries from science and media. Paired with the highest score from the previous stages, we could win both competitions.

7. Conclusion

We proposed a novel approach to scene and object modeling and pose tracking using RGB-D cameras. Central to our approach is the representation of spatial and color measurements in multi-resolution surfel maps. We exploit measurement principles of RGB-D cameras to efficiently acquire maps from images. The transformation between maps representing overlapping parts of the environment (e.g. created from key views) is estimated with an efficient

sequence	all frames		real-time	
	ATE (mm)	time (msec)	ATE (mm)	frames used (%)
humanoid slow	19.99	31.93 ± 5.89	21.14	73.3
humanoid med.	26.12	32.50 ± 5.19	26.97	78.5
humanoid fast	31.64	33.55 ± 6.66	32.20	78.8
box slow	23.55	39.01 ± 4.89	24.33	56.3
box med.	37.83	41.15 ± 14.20	46.58	61.3
box fast	24.05	33.87 ± 9.19	28.64	76.0
chair slow	22.87	49.09 ± 7.13	22.75	46.9
chair med.	15.78	49.57 ± 9.93	16.11	49.5
chair fast	27.67	48.78 ± 10.24	29.10	50.5

Table 5: Median absolute trajectory error, avg. time ± std. deviation, and percentage of frames used in real-time mode for our tracking approach.

yet robust registration method in real-time. Our approach utilizes multiple resolutions to align the maps on coarse scales and to register them accurately on fine resolutions. We demonstrate state-of-the-art registration results w.r.t. accuracy and robustness.

We incorporate our registration method into a probabilistic trajectory optimization framework which performs SLAM in indoor scenes in real-time and allows for learning full-view object models with good precision. Our approach compares very well to other image registration and SLAM approaches using RGB-D cameras in terms of run time and accuracy. Finally, we use the learned models to track the 6 DoF pose of objects in camera images accurately in real-time. By the multi-resolution nature of our image and object maps, our method inherently adapts its registration scale to the distance-dependent measurement noise.

In future work, we will investigate approaches to additional object perception problems such as object detection and initial pose estimation based on multi-resolution surfel maps. We also will investigate the modeling of larger scenes that requires appearance-based techniques for the detection of loop-closures. We currently work on integrating our approach with point-feature-based matching to cope with situations with far and imprecise depth or to make use of fine-grained texture in scenes with little variety in geometric structure. While our approach is efficient enough to run at high frame rates on CPUs, an implementation of our method on GPUs could further improve run time and reduce computational load on the CPU.

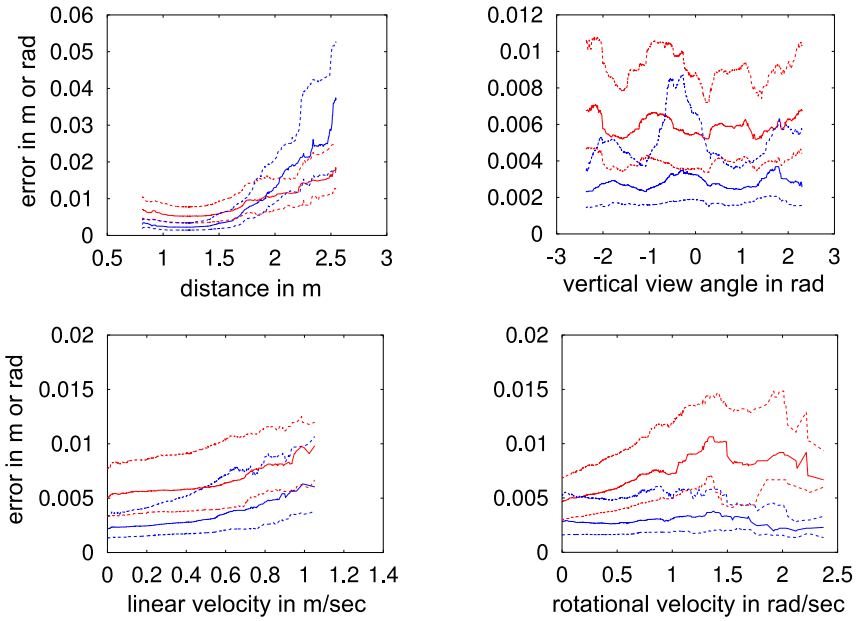

Figure 11: Tracking performance: Median translational (blue) and rotational error (red) and their quartiles (dashed lines) w.r.t. distance, vertical view angle, linear velocity, and rotational velocity on the box tracking sequences.

Acknowledgments

We thank Maria Dimashova, Aleksandr Segal, Todor Stoyanov, and Albert S. Huang for providing their implementations of warp [17] (OpenCV), GICP [5], 3D-NDT [7], and fovis [12] which have been used for the evaluation.

References

[1] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of RGB-D SLAM systems, in: Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012.

[2] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with Rao-Blackwellized particle filters, IEEE Transactions on Robotics 23 (1) (2007) 34–46.

[3] A. Nuechter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM with approximate data association, in: Proc. of the Int. Conf. on Advanced Robotics (ICAR), 2005, pp. 242–249.

[4] M. Magnusson, T. Duckett, A. J. Lilienthal, Scan registration for autonomous mining vehicles using 3D-NDT, Journal of Field Robotics 24 (10) (2007) 803–827.

[5] A. Segal, D. Haehnel, S. Thrun, Generalized-ICP, in: Proc. of Robotics: Science and Systems (RSS), 2009.

[6] P. J. Besl, N. D. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14 (2) (1992) 239–256.

[7] T. Stoyanov, M. Magnusson, H. Andreasson, A. J. Lilienthal, Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations, The International Journal of Robotics Research 31 (12) (2012) 1377–1393.

[8] S. Se, D. Lowe, J. Little, Vision-based mobile robot localization and mapping using scale-invariant features, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), Vol. 2, 2001, pp. 2051–2058.

[9] D. Nister, O. Naroditsky, J. Bergen, Visual odometry, in: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 1, 2004, pp. 652–659.

[10] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, P. Fua, View-based maps, The International Journal of Robotics Research 29 (8) (2010) 941–957.

[11] D. Droeschel, S. May, D. Holz, P. Ploeger, S. Behnke, Robust ego-motion estimation with ToF cameras (2009).

[12] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, 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.

[13] A. J. Davison, I. D. Reid, N. D. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 29 (6) (2007) 1052–1067.

[14] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: Proc. of IEEE/ACM Int. Symp. on Mixed and Augmented Reality (ISMAR), 2007, pp. 225–234.

[15] J. Stuehmer, S. Gumhold, D. Cremers, Real-time dense geometry from a handheld camera, in: Proc. of the 32nd DAGM Symposium, 2010, pp. 11–20.

[16] R. Newcombe, S. Lovegrove, A. Davison, DTAM: Dense tracking and mapping in real-time, in: Proc. of the Int. Conf. on Computer Vision (ICCV), 2011, pp. 2320–2327.

[17] F. Steinbruecker, J. Sturm, D. Cremers, Real-time visual odometry from dense RGB-D images, in: Workshop on Live Dense Reconstruction with Moving Cameras at ICCV, 2011, pp. 719–722.

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

[19] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, J. McDonald, Robust tracking for real-time dense RGB-D mapping with Kintinuous, Tech. Rep. MIT-CSAIL-TR-2012-031, Computer Science and Artificial Intelligence Laboratory, MIT (Sep 2012).

[20] H. Roth, M. Vona, Moving volume KinectFusion, in: Proceedings of the British Machine Vision Conference (BMVC), 2012.

[21] S. May, S. Fuchs, D. Droeschel, D. Holz, A. Nüchter, Robust 3D-mapping with time-of-flight cameras, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009, pp. 1673–1678.

[22] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments, The International Journal of Robotics Research 31 (5) (2012) 647–663.

[23] N. Engelhard, F. Endres, J. Hess, J. Sturm, W. Burgard, Real-time 3D visual SLAM with a hand-held camera, in: Proc. of RGB-D Workshop on 3D Perception in Robotics at European Robotics Forum, 2011.

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

[25] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, Image Vision Comput. 10 (3) (1992) 145–155.

[26] T. Weise, T. Wismer, B. Leibe, L. V. Gool, Online loop closure for real-time interactive 3D scanning, Computer Vision and Image Understanding 115 (5) (2011) 635–648.

[27] M. Krainin, P. Henry, X. Ren, D. Fox, Manipulator and object tracking for in-hand 3D object modeling, The International Journal of Robotics Research 30 (11).

[28] T. F. Chan, G. H. Golub, R. J. LeVeque, Updating formulae and a pairwise algorithm for computing sample variances, Tech. rep., Stanford, CA, USA (1979).

[29] T. F. Chan, J. G. Lewis, Computing standard deviations: accuracy, Communication of the ACM 22 (9) (1979) 526–531.

[30] R. B. Rusu, N. Blodow, M. Beetz, Fast Point Feature Histograms (FPFH) for 3D Registration, in: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), 2009, pp. 3212–3217.

[31] K. Zhou, M. Gong, X. Huang, B. Guo, Data-parallel octrees for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics 17 (5) (2011) 669–681.

[32] 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.

[33] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, 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.

[34] F. Endres, J. Hess, N. Engelhard, J. Sturm, , W. Burgard, 6D visual SLAM for RGB-D sensors, at - Automatisierungstechnik.

[35] J. Stückler, S. Behnke, Following human guidance to cooperatively carry a large object, in: Proc. of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2011, pp. 218–223.