International Journal of Computer Vision (2018) 126:615–636
https://doi.org/10.1007/s11263-017-1057-z

# SDF-2-SDF Registration for Real-Time 3D Reconstruction from RGB-D Data

**Miroslava Slavcheva**[1,2] · **Wadim Kehl**[1,3] · **Nassir Navab**[1] · **Slobodan Ilic**[1,2]

Received: 24 June 2017 / Accepted: 20 November 2017 / Published online: 18 December 2017


**Abstract**
We tackle the task of dense 3D reconstruction from RGB-D data. Contrary to the majority of existing methods, we focus not only on trajectory estimation accuracy, but also on reconstruction precision. The key technique is SDF-2-SDF registration, which is a correspondence-free, symmetric, dense energy minimization method, performed via the direct voxel-wise difference between a pair of signed distance fields. It has a wider convergence basin than traditional point cloud registration and cloud-to-volume alignment techniques. Furthermore, its formulation allows for straightforward incorporation of photometric and additional geometric constraints. We employ SDF-2-SDF registration in two applications. First, we perform small-to-medium scale object reconstruction entirely on the CPU. To this end, the camera is tracked frame-to-frame in real time. Then, the initial pose estimates are refined globally in a lightweight optimization framework, which does not involve a pose graph. We combine these procedures into our second, fully real-time application for larger-scale object reconstruction and SLAM. It is implemented as a hybrid system, whereby tracking is done on the GPU, while refinement runs concurrently over batches on the CPU. To bound memory and runtime footprints, registration is done over a fixed number of limited-extent volumes, anchored at geometry-rich locations. Extensive qualitative and quantitative evaluation of both trajectory accuracy and model fidelity on several public RGB-D datasets, acquired with various quality sensors, demonstrates higher precision than related techniques.

**Keywords** Signed distance field · Registration · 3D reconstruction · Camera tracking · Global optimization · RGB-D sensors

## 1 Introduction

The variety of depth sensors available on the market has brought tasks, such as three-dimensional object reconstruction and simultaneous localization and mapping (SLAM) in static scenes, closer to the general user. The goal is to determine the 6 degrees-of-freedom camera poses and subsequently fuse the respective RGB-D frames into geometrically consistent models. Applications include robotic manipulation, industrial design, interior planning and 3D content creation. Therefore, a variety of real-time solutions have been developed (Kähler et al. 2015; Kerl et al. 2013; Newcombe et al. 2011; Nießner et al. 2013), but most of them solely focus on the accuracy of the estimated trajectory. This is often not representative of the usability of the model, which might be crucial for realistic virtual and augmented reality experiences. Thus, this work aims to develop techniques which also deliver high fidelity 3D reconstructions.

One of the most seminal works capable of real-time reconstruction is KinectFusion (Izadi et al. 2011; Newcombe et al.

Communicated by Michael S. Brown.

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s11263-017-1057-z) contains supplementary material, which is available to authorized users.

✉ Miroslava Slavcheva
mira.slavcheva@tum.de

Wadim Kehl
wadim.kehl@tri.global

Nassir Navab
nassir.navab@tum.de

Slobodan Ilic
slobodan.ilic@siemens.com

[1] Technische Universität München, München, Germany

[2] Siemens CT, München, Germany

[3] Toyota Research Institute, Los Altos, CA, USA

2011). It conveniently stores the recovered geometry in an incrementally built signed distance field (SDF). However, its frame-to-model camera tracking via iterative closest points (ICP Besl and McKay 1992; Chen and Medioni 1991) limits it to objects with distinct geometry and to uniform scanning trajectories. Furthermore, the required data association step might slow performance down for data clouds larger than VGA resolution.

Point-to-implicit techniques (Bylow et al. 2013; Canelhas et al. 2013; Ren and Reid 2012) circumvent costly correspondence search by directly projecting the point clouds of incoming depth frames onto the cumulative SDF and minimizing the difference to its zero level set. Such registration has proven to be more robust than ICP, but becomes unreliable when data is sparse or once the global model starts accumulating errors.

Although the frame-to-growing-model registration used in the above techniques incorporates a form of global optimization through the cumulative SDF, it only allows for drift reduction, without a possibility to reposition incorrectly fused geometry. Existing approaches that explicitly perform optimization require all depth maps (Zhou and Koltun 2013) or meshed scene fragments (Choi et al. 2015; Fioraio et al. 2015; Henry et al. 2013; Zhou et al. 2013) to be stored and lead to lengthy posterior processing.

One of the most widespread choices (Choi et al. 2015; Dimashova et al. 2013; Henry et al. 2013; Kehl et al. 2014) for global refinement is the optimization of a pose graph, e.g. $g^2o$ (Kümmerle et al. 2011). Its cost quickly grows with the number of poses and can thus also entail long runtime. The refinement scheme proposed here relies on a weighted average SDF as a means for spreading information from all frames and, thus, works without a pose graph. Furthermore, we opt for concurrent tracking and refinement, taking inspiration in PTAM (Klein and Murray 2007), one of the most acclaimed real-time monocular SLAM techniques.

The present work employs highly accurate implicit-to-implicit registration, both for camera tracking and multi-view optimization, and applies it to real-time object reconstruction and SLAM. It builds upon two previous approaches (Slavcheva and Ilic 2016; Slavcheva et al. 2016) that address these tasks.

The first contribution, SDF-2-SDF (Slavcheva et al. 2016), is a volumetric, correspondence-free registration energy between pairs of SDFs, used for frame-to-frame camera tracking and frame-to-model pose optimization. Its dense, symmetric formulation allows for a larger convergence basin and more accurate pose estimates than previous techniques. Here we extend the framework to incorporate surface orientation and photometric constraints, achieving even higher accuracy.

Despite its precision, SDF-2-SDF is limited to object reconstruction due to its underlying regular grid structure. Memory reduction techniques such as octrees (Kehl et al. 2016; Steinbrücker et al. 2013, 2014; Zeng et al. 2013) and voxel hashing (Kähler et al. 2015; Nießner et al. 2013) are not applicable, since for accuracy reasons the SDF being aligned is re-generated on every iteration. To enable larger-scale reconstruction, the second contribution, SDF-TAR (Slavcheva and Ilic 2016), applies registration over a fixed number of limited-extent volumes (LEVs). These are partial SDFs anchored at informative, geometry-rich locations. Thus memory usage is fixed and we can profit from massively parallel GPU processing. As the CPU is idle in the meantime, the refinement task is done concurrently there, so that the whole pipeline can be completed in real time. Optimization is done in batches of fixed number of frames and lasts until a next batch is ready. Here we continue the work in Slavcheva and Ilic (2016) by analysing its applicability to large scale object reconstruction versus SLAM, i.e. we investigate its performance under predominantly inward (object-centric) versus outward camera motion.

We extend our analysis of both SDF-2-SDF and SDF-TAR using several public RGB-D benchmarks and compare to other state-of-the-art approaches. Among others, we evaluate on our *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016), which is the first to provide groundtruth CAD models and trajectories for scans acquired with commodity and industrial depth sensors, and is thus highly suited for assessing 3D reconstruction precision.

In the following, we first review related approaches in Sect. 2. We outline our volumetric implicit-to-implicit registration scheme in Sect. 3. Then we explain how it is applied to the tasks of object reconstruction and SLAM, in Sects. 4 and 5 respectively. Section 6 contains various qualitative and quantitative experiments on public datasets. Finally, we discuss strengths and limitations, and conclude in Sect. 7.

## 2 Related Work

In this section we review existing approaches for small- and large-scale object and scene reconstruction from RGB-D data. We focus particularly on the utilized ways for reducing memory and runtime requirements.

*Volumetric Registration* KinectFusion (Newcombe et al. 2011) is among the most celebrated systems for reconstruction from RGB-D data. It employs Curless and Levoy's volumetric depth map fusion (Curless and Levoy 1996) to represent scene geometry as a continuously updated SDF, which aids smoothing noise away. Registration is done by rendering the global SDF into a point cloud and applying point-to-plane ICP (Besl and McKay 1992; Chen and Medioni 1991; Rusinkiewicz and Levoy 2001), making it susceptible to drift under erratic motion or lack of discriminative geometry.

Point-to-implicit approaches (Bylow et al. 2013; Canelhas et al. 2013; Ren and Reid 2012) avoid the costly correspondence association step of ICP. They directly project an incoming point cloud onto the volume and minimize the difference to its zero-level set, achieving more precise camera motion estimation.

SDF-2-SDF registration extends this line of work into an entirely implicit-to-implicit framework by minimizing the direct voxel-wise difference. Thus it is also correspondence-free, and carries additional advantages, such as being denser and symmetric, whereby both SDFs that are being registered steer towards optimal alignment. As a result, it achieves higher accuracy.

*Additional Constraints* Several ICP variants utilize color in order to avoid registration failure when geometry is not sufficiently discriminative (color-ICP Johnson and Kang 1999, RGBD-ICP Henry et al. 2010, multi-feature ICP Schütz et al. 1998). The object reconstruction pipeline of Kehl et al. (2014) relies on dense visual odometry (DVO) (Kerl et al. 2013) for camera tracking, which employs a photoconsistency constraint to find the best alignment between two RGB-D frames. After a pose optimization step, selected keyframes are fused using a modification of Zach et al. (2007)'s TV-$L^1$ minimization scheme, which takes into account the color associated with each SDF voxel. Similarly, Bylow et al. (2014) demonstrate that a voxel grid color term improves registration accuracy, especially in the absence of rich geometric features. As the SDF-2-SDF formulation allows for straightforward incorporation of additional voxel-wise constraints, we also associate RGB values with voxels.

Another possibility to increase pose estimation robustness is through further geometric terms. Masuda (2002) uses the difference between normal vectors to this end. We instead utilize the dot product as a more accurate measure of surface orientation similarity. Although our approach works well without these color and normal constraints, they are straightforward to integrate and further boost performance.

*Memory Load Reduction* A major drawback of regular voxel grids is their high memory requirement, which limits the operational volume to medium-scale spaces. It has been tackled in various ways, including moving volumes (Roth and Vona 2012; Whelan et al. 2013a, 2012), octrees (Kehl et al. 2016; Steinbrücker et al. 2013, 2014; Zeng et al. 2013), voxel hashing (Kähler et al. 2015; Nießner et al. 2013), hierarchical (Houston et al. 2006), non-hierarchical (Nielsen and Museth 2006) and hybrid structures (Chen et al. 2013). However, they are beneficial for storing or updating values, but are not as efficient when an SDF has to be re-generated multiple times per second, e.g. when its camera pose is re-estimated.

On the other hand, methods like RGB-D SLAM (Endres et al. 2012) that detect 2D features and match them in 3D, discard a lot of useful information and require RANSAC (Fischler and Bolles 1981) and pose graph optimization (Kümmerle et al. 2011) to estimate consistent trajectories. While many authors have addressed 3D keypoint detection (Clarenz et al. 2004; Gelfand et al. 2005; Ioannou et al. 2012; Johnson and Hebert 1999; Steder et al. 2010; Tombari et al. 2013), the noise and occlusions inherent to 3D data limit applications to object detection, recognition and classification (Alexandre 2012; Bo et al. 2011; Drost et al. 2010).

Inspired by the accuracy of SDF-2-SDF registration, we aim to apply it to larger-sized objects and SLAM. KinectFusion (Newcombe et al. 2011) and point-to-implicit approaches (Bylow et al. 2013; Canelhas et al. 2013) register an amount of data equal to VGA resolution and thus require only a limited number of reduction operations and can profit from GPU parallelization. However, the number of voxels in SDF-2-SDF depends on the bounding box and the desired voxel size, and is thus not known a priori. To tackle this issue, we carry out registration over a fixed number of limited-extent volumes (LEVs), which are small SDFs with fixed side length. We anchor them at geometry-rich locations, thus ensuring similar accuracy, fixed memory requirements and suitability for GPU implementation.

*Global Optimization* Although refinement can be highly beneficial, it is often not viable for volumetric methods. Due to the high processing requirements of dense data, most existing pipelines resort to expensive posterior optimization that can take hours (Choi et al. 2015; Fioraio et al. 2015; Henry et al. 2013; Zhou and Koltun 2013; Zhou et al. 2013).

This added runtime can be avoided by running refinement concurrently to tracking, in a PTAM (Klein and Murray 2007) fashion. Optimization is applied either on all frames, or on a fixed amount of those last tracked. Pirker et al. (2011) carry out sliding window bundle adjustment, but the used sparse 2D–3D correspondences entail loop closure detection and posterior pose graph optimization. Whelan et al. (2013b) combine incremental as-rigid-as-possible space deformation and every-frame map correction, but depend on the presence of loop closure and add some minimal time latency as more frames are processed. Similarly, ElasticFusion (Whelan et al. 2015, 2016) relies on local loop closures to activate non-rigid model-to-model refinement, without further improving the estimated trajectory.

SDF-2-SDF can be used for global optimization, where an SDF generated from an existing pose estimate is aligned with the weighted average SDF of all frames. In this way drift is distributed across the trajectory without the need for a pose graph, resulting in more consistent geometry of the reconstruction. In the SLAM setting, we apply such refinement on the CPU, concurrently to tracking which runs on the GPU. Optimization is done over LEVs in a batch of the

most recently tracked frames, so that memory is bounded, and runs until the next batch becomes ready.

## 3 SDF-2-SDF Registration

Here we briefly outline our mathematical notation and explain our implicit-to-implicit registration scheme.

### 3.1 Mathematical Notation

Camera tracking entails estimating the 6 DoF pose at every time instance. We represent rigid-body transformations minimally as twist coordinates from the Lie algebra $se(3)$ of the special Euclidean group $SE(3)$ (Ma et al. 2003): $\xi = (\mathbf{u} \quad \boldsymbol{\omega})^\top = (u_1, u_2, u_3, \omega_1, \omega_2, \omega_3)^\top$, where $\boldsymbol{\omega} \in \mathbb{R}^3$ denotes the rotational component and $\mathbf{u} \in \mathbb{R}^3$ corresponds to the translation. We denote the motion of any 3D point $\mathbf{X} = (\mathbf{X}_X, \mathbf{X}_Y, \mathbf{X}_Z)^\top$ in terms of $\xi$ as $\mathbf{X}(\xi)$.

An RGB-D frame is composed of a color image $I : \mathbb{N}^2 \to \mathbb{R}^3$ and an aligned depth map $D : \mathbb{N}^2 \to \mathbb{R}$. Assuming a calibrated camera and a function $\pi(\mathbf{X}) = \mathbf{x}$ that projects 3D points onto pixel coordinates $\mathbf{x} = (x, y)^\top \in \mathbb{N}^2$, $D$ stores the depth values of the points: $D(\mathbf{x}) = \mathbf{X}_Z$. The inverse relation, $\pi^{-1}$, back-projects a pixel $\mathbf{x}$ to 3D: $\mathbf{X} = \pi^{-1}(\mathbf{x}, D(\mathbf{x}))$.

### 3.2 SDF Generation

A signed distance field (SDF) in 3D space is an implicit function $\phi\colon \Omega \subseteq \mathbb{R}^3 \to \mathbb{R}$ that assigns to each point $\mathbf{X}$ its signed distance to the closest surface location (Osher and Fedkiw 2003): it is positive for points in front of objects, and negative for points inside. Thus the surface corresponds to the zeroth level-set crossing, which can be extracted via marching cubes (Lorensen and Cline 1987) or ray tracing (Curless and Levoy 1996).

A single RGB-D pair allows to generate a discrete projective truncated SDF from its corresponding viewpoint. For this purpose, first the bounding volume is determined by back-projecting all pixels. Then it is discretized into cubic voxels of predefined side length $l$.

A point $\mathbf{X}$ lies in the voxel with index $vox : \mathbb{R}^3 \to \mathbb{N}^3$:

$$vox(\mathbf{X}) = int\left(\frac{1}{l}(\mathbf{X} - \mathbf{C}) - (0.5, 0.5, 0.5)^\top\right), \tag{1}$$

where $int$ rounds to integers, and $\mathbf{C}$ is the lower-left corner of the volume. All points within the same voxel are assigned the properties of its center

$$\mathbf{V}(\mathbf{X}) = l\left(vox(\mathbf{X}) + (0.5, 0.5, 0.5)^\top\right) + \mathbf{C}, \tag{2}$$

so denote the entire voxel by $\mathbf{V} \in \mathbb{R}^3$. As a depth image only contains measurements of surface points, the projective signed distance is the difference of sensor reading for the voxel center projection $\pi(\mathbf{V})$ and its depth $\mathbf{V}_Z$:

$$\phi_{true}(\mathbf{V}) = D(\pi(\mathbf{V})) - \mathbf{V}_Z, \tag{3}$$

$$\phi(\mathbf{V}) = \begin{cases} sgn(\phi_{true}(\mathbf{V})), & if|\phi_{true}(\mathbf{V})| \geq \delta \\ \phi_{true}(\mathbf{V})/\delta, & otherwise \end{cases} \tag{4}$$

$$\omega(\mathbf{V}) = \begin{cases} 1, & if\,\phi_{true}(\mathbf{V}) > -\eta \\ 0, & otherwise \end{cases} \tag{5}$$

$$\zeta(\mathbf{V}) = I(\pi(\mathbf{V})). \tag{6}$$

As $\phi_{true}(\mathbf{V})$ is a projective distance, it depends on the viewpoint. To diminish this effect, it is scaled by a factor $\delta$ and truncated into the interval $[-1, 1]$, resulting in $\phi(\mathbf{V})$. Since only values near the surface are significant, a common speed-up practice is to execute calculations only in a narrow-band near it (Adalsteinsson and Sethian 1995; Losasso et al. 2006; Whitaker 1998). The chosen value of $\delta$ determines its thickness.

The binary weight $\omega(\mathbf{V})$ indicates whether the signed distance value for a voxel is reliable. All visible locations and a region of size $\eta$ behind the surface, reflecting the expected object thickness, are assigned weight one. Voxels with zero weight are discarded from computations.

Finally, we store the RGB triple corresponding to each voxel in another grid, $\zeta$, of the same dimensions as $\phi$. Note that color is meaningful only near the surface.

The outlined single-frame SDF generation approach creates *interface beams* where the camera rays pass the surface silhouette, because values of 1 and $-1$ are adjacent there, as shown in Fig. 1. As beams are view-point-dependent, we favour SDF re-generation over interpolation upon camera pose re-estimation. They cancel out when multiple SDFs are fused, but have faulty gradients that need to be omitted from calculations on projective SDFs. This is easily done, since the central difference gradient on a beam has at least one component with absolute value 1, and since voxels behind the surface have not been observed and have zero weight.

We will often use the SDF gradient, since the normalized 3D spatial gradient $\nabla_{\mathbf{X}}\phi$ equals the normals $\overline{\mathbf{n}}$ at surface locations (Osher and Fedkiw 2003). Similar to color, normals are valid only in the narrow band, which is composed of the voxels satisfying the quick binary check $|\phi(\mathbf{V})| < 1$.

Once a new camera pose has been estimated, its SDF is fused into the common model $\Phi$ via the rolling weighted average scheme of Curless and Levoy (1996):

$$\Phi_{t+1}(\mathbf{V}) = \frac{W_t(\mathbf{V})\Phi_t(\mathbf{V}) + \omega_{t+1}(\mathbf{V})\phi_{t+1}(\mathbf{V})}{W_t(\mathbf{V}) + \omega_{t+1}(\mathbf{V})},$$
$$W_{t+1}(\mathbf{V}) = W_t(\mathbf{V}) + \omega_{t+1}(\mathbf{V})\,. \tag{7}$$

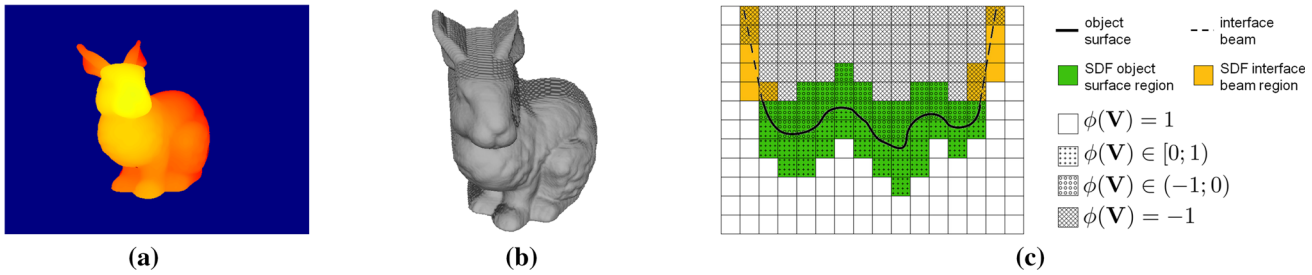



**Fig. 1** Cross section of a single-frame projective truncated SDF, identifying specialized regions, including *interface beams*. **a** Depth map, **b** SDF rendering, showing beams, **c** cross section along the x-y plane of the volume

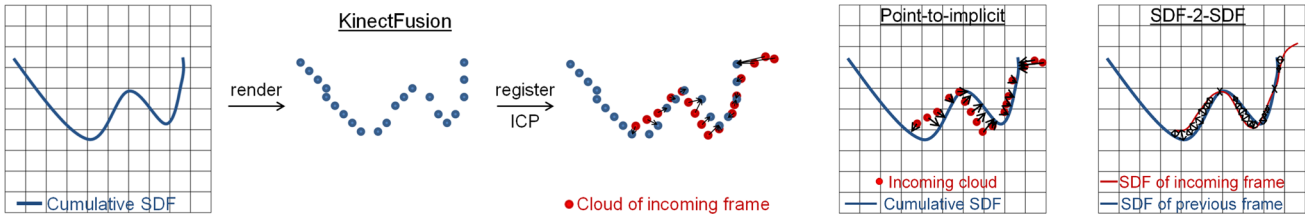



**Fig. 2** 2D analogy for comparison of the operational principles of KinectFusion (Newcombe et al. 2011), point-to-implicit (Bylow et al. 2013; Canelhas et al. 2013) and SDF-2-SDF

Each color channel is averaged similarly, but the weights are multiplied with the cosine of the viewing ray angle, giving more influence to voxels facing the camera (Bylow et al. 2013).

### 3.3 SDF-2-SDF Energy

The SDF-2-SDF registration energy directly minimizes the direct voxel-wise difference of two grids that occupy the same volume. Its main component, $E_{geom}$, is based on the geometry encoded by signed distances:

$$E_{geom}(\xi) = \sum_{voxels} \left( \phi_{ref}\omega_{ref} - \phi_{cur}(\xi)\omega_{cur}(\xi) \right)^2, \qquad (8)$$

where $\phi_{ref}$ is the reference SDF, $\phi_{cur}$ is the SDF whose optimal pose $\xi^*$ is being determined, and $\omega_{ref}$ and $\omega_{cur}$ are the respective weights. Voxel indices are omitted for ease of notation, i.e. we write $\phi_{ref}$ instead of $\phi_{ref}(\mathbf{V})$.

$E_{geom}$ is based on the intuition that, as implicit functions, SDFs densely interpolate depth measurements throughout space. Thus both SDFs that are being registered steer convergence towards the optimally aligned state. This is shown in Fig. 2, where we visualize the operational principles of KinectFusion, point-to-implicit and SDF-2-SDF in 2D. Solid lines correspond to the zero level set of an SDF, which has positive values on one side, and negative values on the other. Note that each voxel in SDF-2-SDF contributes one summand in the energy, as in Eq. 8, but we visualize it more densely to highlight the fact that, as opposed to point clouds, SDFs have values everywhere in the volume.

KinectFusion (Newcombe et al. 2011) associates each point in an incoming point cloud to a point in the cloud rendered from the cumulative SDF. This procedure is both costly and prone to errors. Point-to-implicit approaches (Bylow et al. 2013; Canelhas et al. 2013) minimize the sum of the signed distances of the incoming cloud to its zero level set. This is a more robust variant of ICP, because the points follow the natural gradient of the SDF towards alignment with its zero interface. Finally, SDF-2-SDF extends this idea by registering two implicit fields, both of which have densely sampled signed distance values, whose gradients point towards the minimal cost, i.e. where overlap is best.

Furthermore, SDF-2-SDF is a symmetric energy and will yield nearly identical results if the target and data frames are swapped. On the contrary, point-to-implicit can yield different results depending on which one is represented as a cloud. In particular, if an incoming frame is very noisy, registration can be significantly impaired. Thanks to the smoothing properties of implicit functions, an SDF is likely to be less influenced by noise.

Last but not least, we utilize the truncated $\pm 1$s, as opposed to other approaches that designate them as empty space in order to reduce storage requirements (Kähler et al. 2015; Nießner et al. 2013). While a modified version of voxel hashing or a hierarchical grid might achieve similar functionality, we use a regular voxel grid, which results in more sample points and ensures convergence from a larger deviation.

Thus we attribute our energy function design choice to the geometric benefits of SDFs, including dense sampling of space, a meaningful gradient, and lower sensitivity to noise.

We demonstrate these statements experimentally in the evaluation section.

The grid structure used in the SDF-2-SDF formulation allows for straightforward incorporation of additional constraints that can be expressed over voxels. In particular, we propose two terms on the surface voxels (which we approximate as the non-truncated voxels in an SDF grid), namely we require similar surface orientation, $E_{norm}$, and overlapping color, $E_{RGB}$:

$$E_{norm}(\xi) = \sum_{\substack{surface \\ voxels}} \left(1 - \overline{\mathbf{n}}_{ref} \cdot \overline{\mathbf{n}}_{cur}(\xi)\right), \tag{9}$$

$$E_{RGB}(\xi) = \sum_{\substack{surface \\ voxels}} \sum_{\substack{channel \\ c\in\{R,G,B\}}} \left(\zeta^c_{ref} - \zeta^c_{cur}(\xi)\right)^2. \tag{10}$$

As the SDF gradient equals the normals at surface locations, no additional computations are required. Furthermore, this means that $E_{geom} + E_{norm}$ is a higher-order approximation of the underlying continuous shape than $E_{geom}$ alone. Thus our expectation is that, given data with little to moderate noise, registration will be slightly more accurate and converge faster. On the other hand, we expect $E_{RGB}$ to be helpful in situations with low geometric detail, but richer texture.

The full energy combines all terms, with relative influence determined by the factors $w_{geom}$, $w_{norm}$, $w_{RGB}$:

$$E_{SDF}(\xi) = \frac{1}{2} w_{geom} E_{geom}(\xi) + \\ + w_{norm} E_{norm}(\xi) + \frac{1}{2} w_{RGB} E_{RGB}(\xi). \tag{11}$$

## 4 Object Reconstruction

Our system is depicted in Fig. 3. The object of interest is assumed to be placed on a flat surface, and masked as done in Kehl et al. (2014) and Rusu et al. (2009). Optionally, depth images are de-noised via anisotropic diffusion (Vijayanagar et al. 2014) or bilateral filtering (Tomasi and Manduchi 1998). As opposed to other SDF methods that require manual selection (Bylow et al. 2013; Canelhas et al. 2013; Newcombe et al. 2011), the bounding volume is automatically estimated by back-projection of all masked depth map pixels. It is then slightly padded and used for the generation of both SDFs that are to be aligned.

These steps are applied to each input image fed to our tracking method that performs frame-to-frame SDF-2-SDF registration, thus avoiding error accumulation and allowing for a moving volume of interest. Once it is complete, a predefined number of keyframes is globally SDF-2-SDF-registered to their weighted average SDF, circumventing the need for a pose graph. This refinement follows a coarse-to-fine scheme with respect to voxel size. Note that the tracking and optimization stages are entirely stand-alone, and can be used in other pipelines. Finally, a colored surface mesh is obtained via the marching cubes algorithm (Lorensen and Cline 1987).

### 4.1 Camera Tracking

Frame-to-model tracking can be detrimental in object reconstruction: errors in pose estimation can introduce incorrect geometry when fused into the global model, and consequently adversely affect the subsequent tracking. Therefore, we favor frame-to-frame camera tracking on single-frame projective SDFs.

We determine the relative transformation between two RGB-D frames by setting the pose of the first one to identity and incrementally updating the other one. The tracking minimization scheme for the geometry term is based on a first-order Taylor approximation around the current pose estimate $\xi^k$ (Eqs. 12, 13, 14). Like other rigid registration approaches, it leads to an inexpensive $6 \times 6$ linear system (Eq. 15). Weighting terms have been omitted from formulas for clarity. In order to avoid numerical instability, we take a step of size $\beta$ towards the optimal solution (Eq. 16). In each

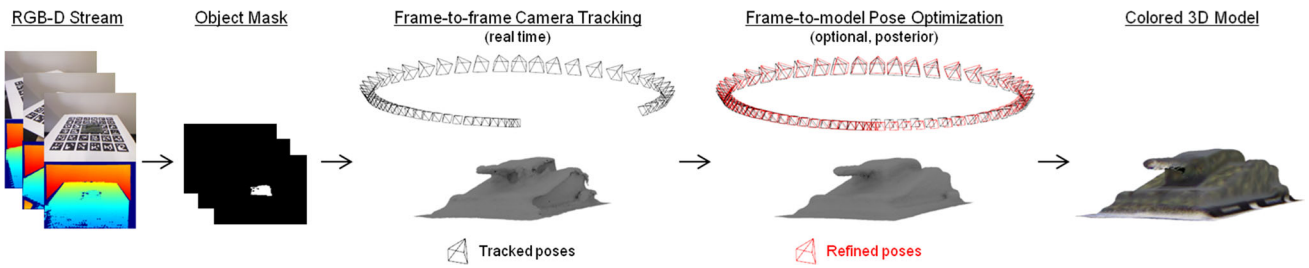



**Fig. 3** SDF-2-SDF pipeline: the bounding box of the object is automatically determined for every frame by masking and back-projection. It is then discretized into voxels and used for SDF-2-SDF registration with the next frame. Once this online tracking stage is complete, several keyframes are jointly optimized in less than a minute in SDF-2-SDF fashion using their weighted average. The system runs entirely on the CPU and finally yields a colored mesh from the SDF grid (Color figure online)

iteration $\phi_{cur}$ is generated from the current pose estimate $\xi^k$. We terminate when the translational update falls below a threshold.

$$\mathbf{A} = \sum_{voxles} \nabla_\xi^\top \phi_{cur}\left(\xi^k\right) \nabla_\xi \phi_{cur}\left(\xi^k\right) \tag{12}$$

$$\mathbf{b} = \sum_{voxles} \left(\phi_{ref} - \phi_{cur}\left(\xi^k\right) + \right. \\ \left. + \nabla_\xi \phi_{cur}\left(\xi^k\right) \xi^k\right) \nabla_\xi^\top \phi_{cur}\left(\xi^k\right) \tag{13}$$

$$\frac{\mathrm{d}E_{geom}}{\mathrm{d}\xi} = \mathbf{A}\xi - \mathbf{b} \tag{14}$$

$$\xi^* = \mathbf{A}^{-1}\mathbf{b} \tag{15}$$

$$\xi^{k+1} = \xi^k + \beta\left(\xi^* - \xi^k\right) \tag{16}$$

Above $\nabla_\xi \phi$ denotes the Jacobian of the point $\mathbf{V} \in \mathbb{R}^3$, denoting the voxel center, with respect to the pose $\xi$. It is obtained by the chain rule:

$$\nabla_\xi \phi(\mathbf{V}(\xi)) = \nabla_{\mathbf{X}} \phi(\mathbf{V}) \frac{\partial \mathbf{V}}{\partial \xi} = \\ = \nabla_{\mathbf{X}} \phi(\mathbf{V}) \left(\mathbf{I}_{3\times 3} \mid -[\mathbf{V}(\xi^{-1})]_\times\right), \tag{17}$$

where $\mathbf{I}_{3\times 3}$ is the $3 \times 3$ identity matrix, $\xi^{-1}$ denotes the inverse of the rigid pose represented by twist coordinates $\xi$, and $[\cdot]_\times$ is the skew-symmetric matrix of its argument. Thus $\nabla_\xi \phi \in \mathbb{R}^{1\times 6}$.

Each color grid channel is a scalar field, so it is treated identically to $E_{geom}$.

As the normals of the SDF equal its spatial gradient, the surface orientation term imposes curvature constraints, whose derivation is mathematically equivalent to a second-order Taylor approximation of $E_{geom}$. The derivative of $E_{norm}$ with respect to each component $j$ of the twist coordinates is:

$$\frac{\mathrm{d}E_{norm}}{\mathrm{d}\xi_j} = \sum_{\substack{surface \\ voxels}} -\overline{\mathbf{n}}_{ref} \cdot \left(\nabla_{\mathbf{x}} \overline{\mathbf{n}}_{cur}(\xi) \quad \frac{\partial \mathbf{V}}{\partial \xi} \quad \delta_j\right), \tag{18}$$

where $\delta_j$ is a 6-element vector of zeros with $j$-th component 1, and $\nabla_{\mathbf{x}} \overline{\mathbf{n}} \in \mathbb{R}^{3\times 3}$ is the spatial gradient of a normal vector, which evaluates how the orientation changes with location, i.e. it is a measure of curvature.

### 4.2 Global Pose Optimization

After tracking, a predefined number of regularly spaced keyframes are taken for generation of the final reconstruction. The weighted averaging provides a convenient way to incorporate the information from all of their viewpoints into a global model $\phi_{avg}$.

However, when using noisy data the estimated trajectory might have accumulated drift, so the keyframes' poses need to be refined to ensure optimal geometry. For this task we propose a frame-to-model scheme based on the SDF-2-SDF registration energy. Each pose $\xi_t$ is better aligned with the global weighted average model $\phi_{avg}$, which is used as the reference in Eq. 8. In effect, the optimization is interleaved with the computation of the final reconstruction, and takes less than 30 s for 24 keyframes. The linearization of the energy follows a gradient descent minimization with step $\alpha$:

$$\frac{\mathrm{d}E_{geom}}{\mathrm{d}\xi} = \sum_{voxels} \left(\phi_{cur}(\xi) - \phi_{avg}\right) \nabla_\xi \phi_{cur}(\xi), \tag{19}$$

$$\xi_t^{k+1} = \xi_t^k - \alpha \frac{\mathrm{d}E_{geom}(\xi_t^k)}{\mathrm{d}\xi}. \tag{20}$$

The pose of the first camera determines the world coordinate frame and is fixed to identity throughout the whole optimization. In each iteration, the pose updates of all other keyframes are determined using the global model, after which they are simultaneously applied. The weighted average is recomputed every couple of iterations (10 in our case), rather than on every step, so that the objective does not change in the meantime. Furthermore, this is done in a coarse-to-fine scheme over the voxel size to ensure that larger pose deviations can also be recovered.

### 4.3 Implementation

The SDF-2-SDF energy is highly parallelizable, because the contributions of each voxel are independent. However, as we estimate the bounding box on the fly, their amount is not known beforehand. Therefore, the number of reduction operations which ultimately lead to the $6 \times 6$ system of Eq. 15 is unknown. This is in contrast to KinectFusion (Newcombe et al. 2011) and point-to-implicit (Bylow et al. 2013; Canelhas et al. 2013), where a VGA-sized depth image is registered to a point cloud or an SDF, respectively. Thus the number of reduction operations in these methods is fixed and they can be implemented efficiently on the GPU. Instead, we opt for a parallelized CPU solution on an 8-core Intel i7-4900MQ CPU at 2.80 GHz.

As tracking at a voxel size finer than the sensor resolution is futile, we used 2 mm, corresponding to the expected error of our noisiest sensor, the Kinect. We used SSE instructions to make SDF generation efficient, which leaves the computation of each voxel's contribution to the $6 \times 6$ system as the bottleneck. To speed it up, we only process voxels with positive weight, and different values in the two grids, achieving real-time performance between 17 and 22 FPS on tabletop objects.

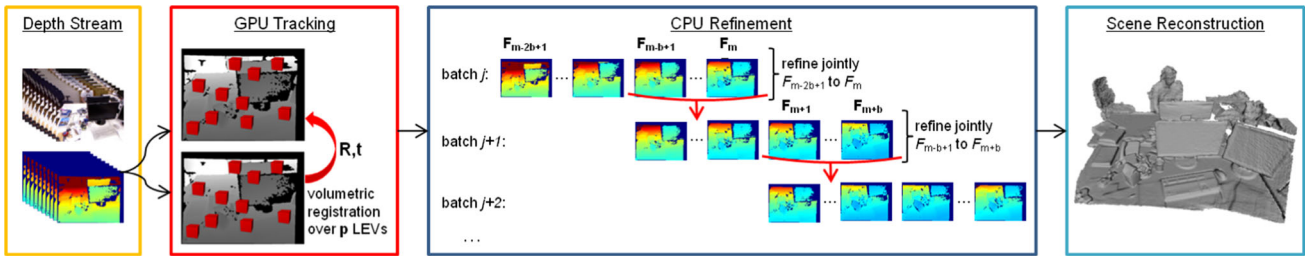



**Fig. 4** SDF-TAR pipeline: the relative motion between every two depth frames is estimated on the GPU from $p$ limited-extent volumes, anchored at locations of high curvature. As soon as frame $F_m$ is tracked, the CPU refinement module starts jointly optimizing $F_{m-2b+1}$ to $F_m$. In the meantime tracking resumes on $F_{m+1}$ to $F_{m+b}$. Once this new batch is ready, refinement is switched to $F_{m-b+1}$ to $F_{m+b}$. This strategy ensures that every pose is optimized twice for highest geometric consistency

On the other hand, the pose optimization has a simpler mathematical formulation, whereby only a 6-element vector is calculated in every gradient descent step. We apply a pyramid over voxel size (4, 2 mm, optionally 1 mm), which ensures that first larger deviations are handled, while the final model is of a high resolution. Typically the whole refinement stage takes less than half a minute, even in case of severe drift. Exact timings will be provided in the experimental section.

## 5 SLAM

Having developed an advantageous registration strategy, our next goal is to apply it for the reconstruction of larger objects and indoor spaces, i.e. to make it suitable for SLAM-like scenarios. However, regular voxel grids, as used in SDF-2-SDF object reconstruction, are extremely memory-intensive. This becomes especially problematic if a fine voxel resolution, as required for accurate reconstruction, is used for a large scene. Storing only the signed distances for $512^3$ voxels takes 0.5 GB, and for $1024^3$ - 4 GB. These figures further increase for the storage of the weight and color grids. The problem soon becomes intractable, as processing a high amount of voxels also naturally entails increased runtime. As discussed in Sect. 4.3, the involved reduction operations prevent a straightforward GPU parallelization.

To counter both the memory and speed issues, we propose a modification of the SDF-2-SDF scheme, which runs over a small, fixed amount of voxels, and consequently can be done efficiently on a GPU. We constrain registration to limited-extent volumes at geometry-rich locations, so that we can keep its accuracy, while reducing its load and increasing its speed. Furthermore, as the CPU is left mostly idle, we simultaneously use it for pose optimization. Thus we obtain a hyprid GPU/CPU scheme, visualized in Fig. 4. We call it SDF-TAR, for SDF-based parallel tracking and refinement.

### 5.1 Limited-Extent Volumes

We propose a simple to implement solution that significantly reduces the memory load. Our key idea is to set $p$ partial SDFs $\Omega_1, \ldots, \Omega_p$ of resolution $x \times y \times z$ voxels with side length $l$, and carry out SDF-2-SDF registration searching for a common rigid-body motion $\xi$ for all of these limited-extent volumes (LEVs) simultaneously. This strategy guarantees that memory will be kept constant for every pair of frames and gives an upper bound for the processing time, letting us select a maximum number of iterations that will always stay within real-time constraints.

While the choice of the LEVs' positions is obviously critical, it is also natural. Guided by the intuition that flat areas, like walls, do not contribute to and could even inhibit registration, we propose to anchor the volumes at points of high curvature. Such regions are highly distinct from their surroundings and can therefore quickly lead registration to an optimal solution.

Figure 5 illustrates the anchor point selection process. To minimize runtime, all operations are done directly on the depth map. Since the sensor error increases quadratically with distance (Khoshelham and Elberink 2012), we consider measurements further than 2 m unreliable and discard them. Furthermore, RGB-D cameras are inaccurate near depth discontinuities, thus we also mask out pixels near edges. Next, we estimate the surface normals as derivatives over the pre-processed depth map, following the method of Holzer et al. (2012). Then we calculate the curvature magnitude from the derivatives of the normal map. Finally, we apply non-maximum suppression (Neubeck and Van Gool 2006), so that only one high curvature point is selected within every window of size $w \times w$ pixels. This ensures that the LEVs centered around these locations will not overlap. Finally, we select the $p$ points with highest curvature values in the non-maximum-suppressed image. If there are less than $p$ peaks, we simply take all of them.

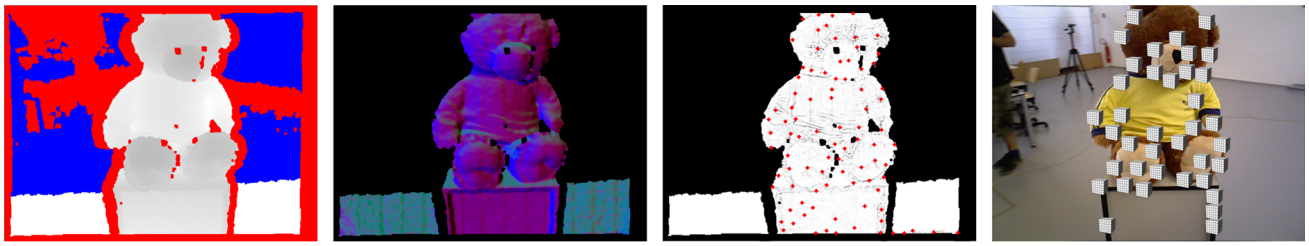


**Fig. 5** Limited-extent volume anchor point selection process: **a** locations too far away (blue) and near edges (red) are masked out from the input depth map, to discard potentially noisy values. **b** Normals are calculated as derivatives over depth. **c** Curvature is calculated as derivatives of normals. Its size is non-maximum suppressed to determine peaks separated by a minimal distance. **d** The $p$ highest peaks are used as anchor points for the LEVs, which are small SDFs of fixed size (Color figure online)

### 5.2 Limited-Extent Volume Registration

The modified geometric energy becomes:

$$E_{geom'}(\xi) = \sum_{\substack{voxels\ \Omega_i \\ i=1..p}} \left( \sum_{voxels \in \Omega_i} \left( \phi_{ref}\omega_{ref} - \phi_{cur}(\xi)\omega_{cur}(\xi) \right)^2 \right), \tag{21}$$

and similarly the sums in Eqs. 9, 10, 11, 12, 13, 18.

We keep tracking in a frame-to-frame manner. However, if refinement is done over all already tracked frames, its convergence time will respectively increase with their number. In addition, refinement over frames separated by a large distance is not necessarily beneficial, since they may be capturing completely non-overlapping parts of the scene. Therefore, we propose to carry out pose optimization over batches of the last few frames.

More precisely, it is done over $q \leq p$ LEVs, jointly in batches of $2b$ frames, the first half of which have already been refined once, while the second half are the lastly tracked ones. A local weighted average $\phi_{loc\ avg}$ of the $2b$ frames is generated in each LEV. As in the object reconstruction case, each $\phi_{loc\ avg}$ is re-calculated only on every $f$th iteration in order to keep the objective fixed meanwhile. For stability the first $b/2$ poses are kept fixed, while each other pose is refined following the previously introduced gradient descent scheme, resulting in one 6-element-vector pose update. Therefore, once frame number $m$ is tracked, optimization is carried out following the modified version of Eq. 19:

$$\frac{\mathrm{d}E_{geom'}}{\mathrm{d}\xi} = \sum_{\substack{voxels\ \Omega_i \\ i=1..p}} \left( \sum_{voxels \in \Omega_i} \left( \phi_d(\xi) - \phi_{loc\ avg} \right) \nabla_\xi \phi_d(\xi) \right), \quad d \in [m-2b+1, \ldots, m]. \tag{22}$$

### 5.3 Parallel Tracking and Refinement

As our objective is a fully real-time SLAM method without any posterior processing, we execute the tracking and refinement modules concurrently. We allocate a separate GPU stream responsible for tracking: an incoming depth map is transferred to device memory, pre-processed and then registered to the previous one using the limited-extent volume scheme explained above. When $b$ frames have been processed, the CPU is signalled and starts the optimization module. Refinement is done in a locally global fashion: a local batch of $2b$ frames is jointly globally optimized. The batch consists of the newly tracked $b$ poses and the $b$ previous ones, of which the first $b/2$ are kept fixed for stability and only contribute to the weighted average generation. Refinement runs until the next $b$ frames have been tracked, when the CPU is signalled to switch batches and the procedure continues. This strategy gives a broader context for optimization and ensures that every frame participates in the refinement twice, thus is geometrically consistent with frames both before and after it.

Given a trajectory estimated in this manner, a reconstruction can be generated in various ways, among which volumetric fusion (PCL 2017; Newcombe et al. 2011), carefully selected key-frame fusion (Meilland and Comport 2013), or point-based fusion (Keller et al. 2013). memory load. As the particular method is not the focus of this work, when comparing the outputs of different pipelines we will always display results generated with the same technique (PCL 2017).

### 5.4 Implementation

Our implementation was done on the previously used Intel i7-4900MQ CPU at 2.80 GHz, and an NVIDIA Quadro K2100M GPU. Pre-processing VGA-sized depth images takes 7–8 ms: transferring the image to device memory, and estimating its normals and curvature size take approximately

4.5 ms in total, while the non-max-imum suppression and sorting the peaks in order of their curvature magnitude last another 3 ms. The remaining 25 ms are entirely available for tracking, so the maximum number of iterations is set depending on the number of SDFs. Depending on frame distance, typically 20–60 iterations are sufficient for convergence. Refinement runs concurrently until the signal that a new batch is ready, when it switches to the new batch.

The tracking module requires 160 MB of GPU memory for $p = 64$ SDFs (if signed distances are stored as `float` and weights as `uchar`), totalling 322.4 MB for two frames together with their depth maps. In addition, the refinement module takes 20 MB of CPU memory for the weighted average, and another 23.4 MB for 20 depth images. These values demonstrate the real-time capabilities of SDF-TAR, combined with its low memory load. Furthermore, they show that there are enough resources for an additional thread, responsible for the data fusion in parallel.

## 6 Evaluation

Here we perform extensive qualitative and quantitative analysis of various aspects of SDF-2-SDF and SDF-TAR.

### 6.1 Test Setup and *3D-Printed RGB-D Object Dataset*

*Datasets* Our goal has been to develop solutions that are generic with respect to sensor noise characteristics, scanning motion, and object/ scene geometry and texture. Moreover, as we are interested in the usability of models, we want to assess not only tracking, but also reconstruction accuracy on real data. Therefore, we use several public datasets, acquired with different sensors.

The availability of benchmarks for object reconstruction is far more limited than for SLAM. Existing RGB-D collections of household items, such as that of Washington University (Lai et al. 2011) and Berkeley's BigBIRD (Singh et al. 2014), either lack noiseless meshes or complete 6 DoF poses (Narayan et al. 2015). Thus, we 3D-printed a selection of objects with diverse geometry, size and colors, and contribute the first, to the best of our knowledge, object dataset with original CAD models and RGB-D data from various quality sensors, acquired from externally measured poses. Our *3D-Printed RGB-D Object Dataset*, shown in Fig. 6, is available at http://campar.in.tum.de/personal/slavcheva/3d-printed-dataset/index.html.

Our five objects exhibit various richness of geometry and texture: uniformly colored (*bunny*), colored in patches (*teddy, Kenny*), densely colored (*leopard, tank*); very small (*Kenny*), very large (*teddy*); with thin structures (*leopard*'s tail, *tank*'s gun), with spherical components (*teddy, Kenny*) and symmetries (*teddy, Kenny, tank*). They were 3D-printed in color with a *3D Systems ZPrinter 650*, which reproduces details of resolution 0.1 mm (ZCorporation 2008). Thus we ensure that the textured ground-truth CAD models are at our disposal for evaluation, eliminating dependence on the precision of a stitching method or system calibration that other datasets entail.

**Fig. 6** Members of the *3D-Printed RGB-D Object Dataset*. The *bunny* is from The Stanford Repository, the *tank*—from the 3D Warehouse, and all other models—from Archive3D

To capture increasing levels of sensor noise, we used *three RGB-D cameras*: noise-free synthetic rendering in *Blender* (https://www.blender.org/ 2017), an industrial phase shift sensor of resolution 0.13 mm, and a Kinect v1. We recorded in *two scanning modes: turntable and handheld* with the Kinect. We simulated them in *Blender* as 120-pose trajectories of radius 50 cm, where the handheld one is a sine wave with frequency 5 and amplitude 15 cm. Thus the synthetic *groundtruth trajectories* are known, while the Kinect poses are obtained from a markerboard placed under the object. The industrial sensor takes 4 s to acquire a single RGB-D pair, permitting us to only record turntable sequences. Due to its limited field of view, we could not place a sufficiently large markerboard, so we will only use it for evaluation of model accuracy. In all cases the object of interest is placed on a textured support that ensures optimal conditions for visual odometry, ensuring fair comparisons.

For reconstruction of bigger objects we use the *Large Dataset of Object Scans* (Choi et al. 2016). It provides PrimeSense Carmine scans of furniture items and vehicles, acquired by non-professional users in their everyday environments. Furthermore, some sequences include a reconstruction based on KinectFusions's ICP registration combined with DVO's RGB-D photometric error. Hence this collection is suitable for qualitative comparison in realistic scenarios against well-established methods.

On the intersection between these two datasets is *CoRBS* (Wasenmüller et al. 2016b). It also contains large-scale objects: desk, human, electric cabinet and car, each captured in five different Kinect v2 trajectories. Most importantly, it provides ground-truth models and poses for them, which enable us to evaluate our approach on bigger objects.

Finally, to analyse our SLAM capabilities, we test on the *TUM RGB-D benchmark* (Sturm et al. 2017), which encompasses many Axus Xtion scans of indoor spaces, together with ground-truth trajectories.

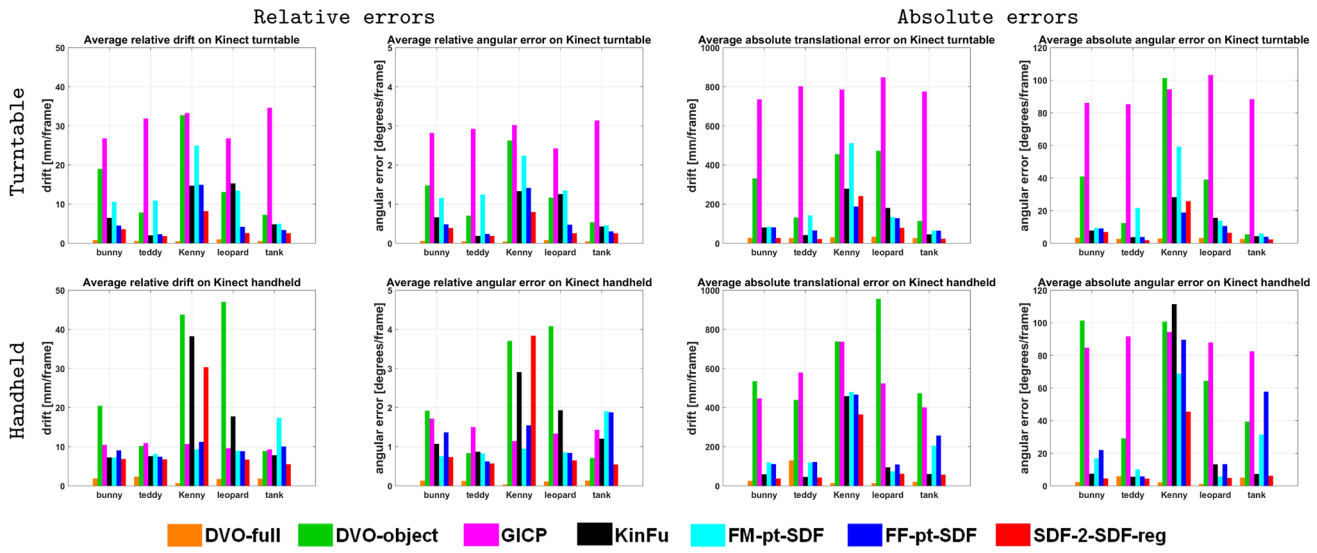



**Fig. 7** Comparison of per-frame tracking errors on Kinect sequences from the *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016)

*Methods* Different components of our techniques will be compared to the following state-of-the-art methods:

- *GICP*: PCL's (http://pointclouds.org/ 2017) frame-to-frame generalized ICP (Segal et al. 2009);
- *KinFu*: PCL's KinectFusion (PCL 2017; Newcombe et al. 2011) as a frame-to-model ICP variant;
- *FM-pt-SDF*: the frame-to-model point-to-implicit techniques of Bylow et al. (2013) and Canelhas et al. (2013) (available as a ROS package Canelhas 2017);
- *FF-pt-SDF*: our frame-to-frame modification of Canelhas (2017);
- *DVO-object*: dense visual odometry without refinement (Kerl et al. 2013) (available online (Kerl 2017) only over the object;
- *DVO-full*: DVO on the entire scene (no refinement);
- *KinFu+DVO*: the ICP/DVO combination of Choi et al. (2016);
- *DNA-SLAM*: the ToF noise-aware DVO variant in Wasenmüller et al. (2016a);
- *Kehl et al.*: the object reconstruction pipeline of Kehl e al. (2014), which tracks by DVO, detects loop closure, optimizes keyframe poses via g$^2$o (Kümmerle et al. 2011), and integrates them via TV-$L^1$ minimization (Zach et al. 2007) over colored SDFs.

As our tracking and pose optimization routines can be used stand-alone, we evaluate them separately. Thus, in addition to the *SDF-2-SDF* and *SDF-TAR* pipelines, we test the tracking-only component, denoted as *SDF-2-SDF-reg*. Unless otherwise specified, we only use $E_{geom}$ for higher speed and lower memory consumption than with the optional constraints $E_{norm}$ and $E_{RGB}$.

*Metrics* We employ the typical RGB-D benchmark metrics (Sturm et al. 2017), namely the *absolute trajectory error* (*ATE*), which quantifies the overall trajectory error, and the *relative pose error* (*RPE*), which is the drift over a fixed time interval. However, the ATE first determines the best alignment between the groundtruth and estimated trajectories. This is better suited to SLAM than object reconstruction, where every-frame fusion is done with respect to the first pose. Thus we also use an *absolute pose error*, which does not carry out an alignment step.

We evaluate the reconstruction error against the CAD model, used for 3D-printing, in CloudCompare (http://www.danielgm.net/cc/ 2017).

## 6.2 3D Object Reconstruction

We start our evaluation with the trajectory and model precision achieved on household-scale objects.

*Tracking* Figure 7 provides an overview of the tracking errors on all Kinect sensor sequences of our *3D-Printed Dataset*, without any pose refinement. We show both relative and absolute values for translational and angular deviation. The overall trend is that the angular error reflects the translational one. Furthermore, we have calculated minimum, maximum, average and root-mean squared (RMS) metrics, but note that the average ones are most indicative and only display those.

The results indicate that typically DVO-full, using the object together with its richly textured background, and our SDF-2-SDF-reg are most precise, while GICP and DVO-object are least accurate. SDF-2-SDF-reg clearly outper-

forms the remaining volumetric methods. On large objects, like *teddy*, our average relative drift is below 2 mm, which corresponds to the voxel size used for tracking, suggesting that when the data is not severely influenced by noise, only the voxel resolution is a limiting factor for accuracy. Similarly, our average relative angular error is always below 1° , and is often almost negligible, e.g. 0.19° on turntable *teddy*, and 0.26° on *tank* and *leopard*, which are challenging objects with thin structures. Likewise, SDF-2-SDF-reg has better absolute metrics even than DVO-full on turntable *bunny*, *teddy*, *tank* and handheld *teddy*, despite using only geometric constraints over the object of interest.

KinFu and the two point-to-implicit strategies perform similar to each other. In most cases KinFu is more accurate than pt-SDF, while frame-to-frame is slightly better than the frame-to-model pt-SDF variant. A notable failure case for FM-pt-SDF was the turntable *teddy*, where symmetry on the back caused drift from the middle of the sequence onwards, which lead to unrepairable errors in the global model and consequently flawed tracking. Similarly, FM-pt-SDF performed poorly on the turntable *Kenny* due to its fine structures, while FF-pt-SDF did not suffer from error build-up and was most accurate. These observations support our hypothesis that a frame-to-frame strategy is better for object scanning scenarios, in which, as opposed to SLAM, it is rare to repeatedly cover vast overlapping areas.

In the presence of severe Kinect-like noise, the sparse point clouds of objects in the scene tend to become too corrupted and degrade registration accuracy. On the contrary, SDF-2-SDF-reg is more precise than cloud-based KinFu and pt-SDF because the inherent smoothing properties of volumetric representations counteract noise better. Moreover, SDF-2-SDF-reg relies on a denser set of correspondences: on average, the used clouds consist of $8 \times 10^3$ data points, while the SDFs have $386 \times 10^3$ voxels. Thus the problem is constrained more strongly, making our proposed registration strategy more suitable for object reconstruction.

*Convergence Basin* To deepen our analysis of SDF-2-SDF registration, we investigated its convergence basin. We simulated initial conditions, in which the global minimum is gradually further away, by skipping frames from the original turntable Kinect sequences. The first five plots in Fig. 8 display comparisons to other techniques on each object, while the last plot is averaged over all of them. In the majority of cases, the errors of most methods grow approximately linearly, but SDF-2-SDF-reg has the slowest rate. Thus thanks to its denser formulation and the existence of meaningful values everywhere in the volume, it can determine an accurate pose from a much larger initial deviation. In particular, with the exception of two-frame distance on *Kenny*, SDF-2-SDF-reg is considerably more precise than DVO-full. The remaining results exhibit a trend similar to that of Fig. 7: GICP,

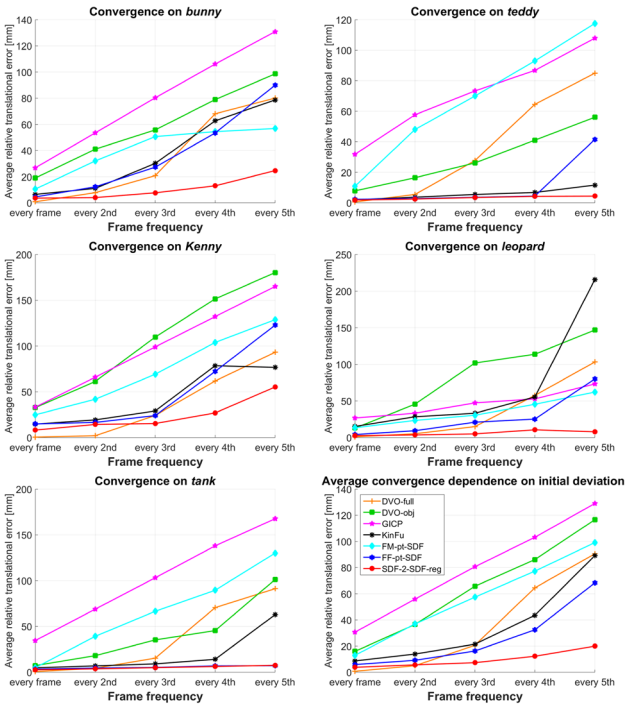



**Fig. 8** Convergence analysis of registration methods with respect to frame distance, simulating larger initial deviation

DVO-object and FM-pt-SDF have the fastest error growth rates, and are outperformed by KinFu and FF-pt-SDF, which behave alike.

Notably, on *tank* the errors of FF-pt-SDF and SDF-2-SDF-reg are nearly identical for each frame distance. This indicates that a frame-to-frame strategy is more advantageous than frame-to-model for a larger pose difference. The reason is that a model is of limited help here, because a new frame exposes more unseen parts of the object. On the other hand, the previously observed parts can steer into local minima. Moreover, as the *tank* has a relatively uncomplicated geometry, the point-to-implicit and implicit-to-implicit methods behave similarly. However, the remaining objects, where geometry is more peculiar, present a larger challenge to pt-SDF, since its point clouds are more susceptible to noise than SDFs. These observations once again confirm our design choices for the SDF-2-SDF framework.

*Additional Constraints* Next, we evaluate the effect of surface orientation and photoconsistency on the registration error, summarized in Fig. 9. We obtained similar results with weight values from the set $\{0.05, 0.1, 0.2\}$ for both $w_{norm}$ and $w_{RGB}$. Each additional constraint decreases the error of $E_{geom}$ by a certain amount, depending on the properties of the object, while all three together make $E_{SDF}$ most accurate. This does not hold only for the angular error of haldheld *Kenny*. Its error decreases with the normal term, but increases with the texture one. We suppose this is due to depth-to-color

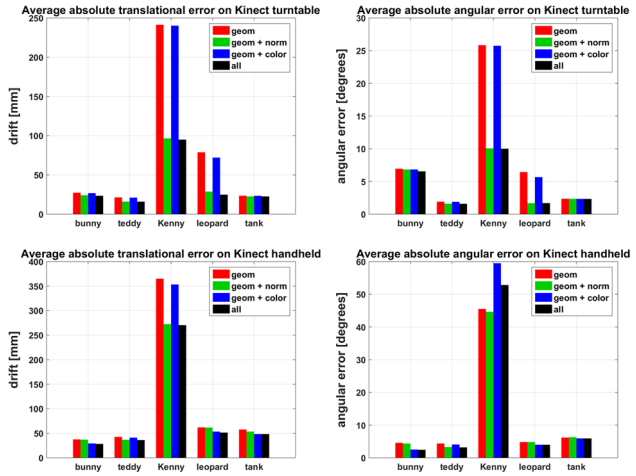



**Fig. 9** Effect of curvature and photometric constraints on the average absolute error of SDF-2-SDF registration

**Table 1** Effect of curvature constraints on convergence rate: comparison of number of iterations until convergence when tracking based on the signed distance term only (*geom*) versus combined with the surface orientation term (*geom+norm*)

| Object | Iterations to convergence | | | |
|---|---|---|---|---|
| | Turntable | | Handheld | |
| | geom | geom+norm | geom | geom+norm |
| Bunny | 42.09 | 23.37 | 41.26 | 31.18 |
| Teddy | 17.28 | 15.60 | 25.16 | 17.22 |
| Kenny | 46.01 | 28.36 | 76.44 | 41.09 |
| Leopard | 24.03 | 17.29 | 34.52 | 22.49 |
| Tank | 28.88 | 19.73 | 41.86 | 29.02 |

camera calibration, which can lead to a significant offset on a small object like this. Moreover, considering that the additional terms entail more calculations, we advocate to use them depending on the specific case. For instance, $E_{norm}$ is very beneficial on the *teddy*, since it is a large object, where normals can be estimated reliably. Color helps on richly textured objects, like *leopard* and *tank*. In several cases the final error of $E_{SDF}$ has become even lower than that of DVO-full from Fig. 7, where it previously was not.

As previously discussed, since surface normals are the derivatives of signed distances, $E_{norm}$ is a second-order term in addition to $E_{geom}$, which does not significantly change the optimum of the energy. However, assuming that the computation of normals is not heavily influenced by noise, we expect that the optimum would be reached in fewer iterations. We investigated this claim in Table 1 and Fig. 10.

First, we notice that handheld sequences typically require more iterations to convergence than turntable ones. This is due to the more erratic scanning motion, causing more noticeable effects of motion blur and rolling shutter. In addition, bigger objects, like *teddy*, require less iterations than smaller ones, like *Kenny*, since they contain more data, a smaller portion of which is influenced by noise. Finally, the expectation for less iterations with the normal term is confirmed in all cases. The plots indicate that $E_{norm}$ remedies cases when $E_{geom}$ alone did not converge, since spikes in the red lines are not present in the blue ones. There are rare cases in which the combined energy needs several more iterations than the geometric one alone, occurring on frame pairs with smaller overlap. As the standard deviation of iteration number decreases noticeably, we conclude that the second-order term regularizes the energy.

Note that high quality industrial sensor sequences typically require less iterations than Kinect ones, even based on $E_{geom}$ only. Therefore, while on Kinect data $E_{norm}$ might lead to convergence in half the iterations, its contribution is not as significant on less noisy data.

*Pose Refinement and Object Reconstruction* Figure 11 shows our results on the industrial sensor and Kinect sequences of our *3D-Printed Dataset* after global pose refinement. While the shapes are reconstructed well, the difference in device quality is apparent. The models obtained from phase shift data are very detailed, while those from Kinect are smoothed out. This is most visible on the edges of the *tank*, on the ears of the *leopard* that were not captured by the Kinect, and from the lack of details on the *bunny* body.

Furthermore, in Figs. 12 and 13 we provide qualitative comparison on both the industrial and Kinect turntable sequences of our *3D-Printed RGB-D Object Dataset* between KinFu, the DVO-full based method of Kehl et al., and SDF-2-SDF without and with refinement. These snapshots reflect the numerical reconstruction errors, listed in Table 2, where we also test Kehl et al.'s pipeline with the less accurate DVO-object.

In most cases, SDF-2-SDF-reg yields better results than KinFu even without refinement. In particular, optimization is typically not needed when using phase shift data. On the other hand, it is vital on the more challenging *Kenny*, *leopard* and *bunny* Kinect scans. Therefore, our registration technique alone outperforms related methods on high quality depth data, while it requires the posterior refinement step on noisier data.

Both figures show that the large *teddy* is the easiest object for all methods, while the tiny *Kenny* is most difficult, since it is more affected by noise. On industrial data SDF-2-SDF-reg produces slightly better reconstructions than Kehl et al., while on Kinect data Kehl et al. is superior. However, the model errors indicate that if its DVO tracking component is constrained only on the object, performance is significantly worse on Kinect data and might even fail on the more erratic handheld scans. Contrary to expectations, the table shows better results for Kehl-object than Kehl-full on industrial

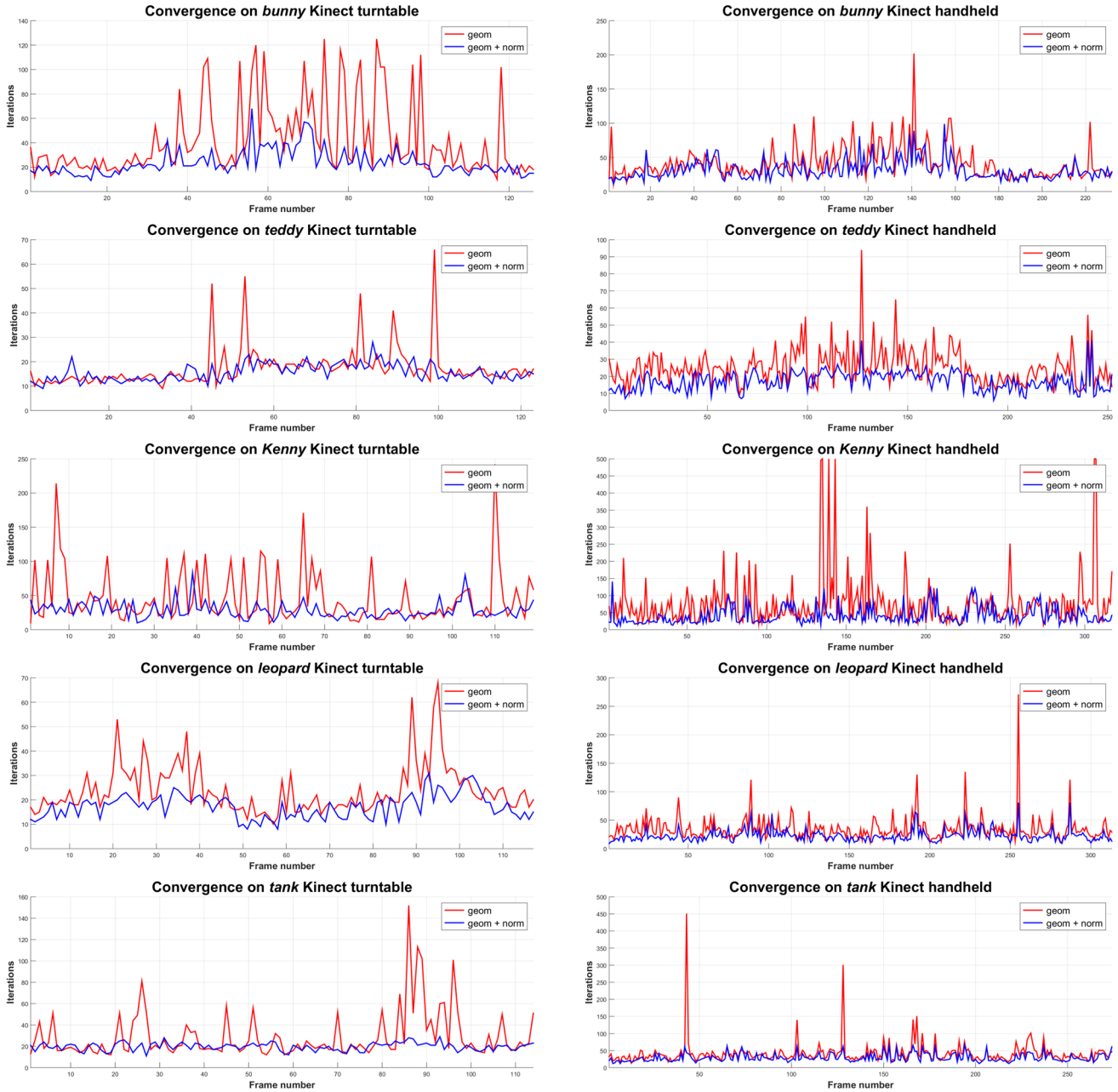



**Fig. 10** Comparisons of iterations to convergence with $E_{geom}$ versus $E_{geom} + E_{norm}$ on Kinect sequences

data. This is, however, because the provided implementation required resizing the original 2040 × 1080 images to VGA resolution, leading to increased error when processing areas near the image border, where the textured table is located. The results of KinFu and SDF-2-SDF did not change for VGA and the original size, indicating lower sensitivity of volumetric approaches to such issues. Moreover, the speed of SDF-2-SDF remained unaffected, as it only depends on the voxel resolution, and not on the image or point cloud size, while KinFu slowed down with larger image dimensions. Thus our system generalizes well not only for various object geometry, but also for any device.

SDF-2-SDF's error is clearly below 1 mm on all phase shift sequences, and stays below 2 mm on the Kinect ones. As this corresponds to the device uncertainty, we once again confirm that our approach is only limited by the sensor resolution and the voxel size.

Finally, to demonstrate the generality of our approach, we test it on bigger objects from the *Large Dataset of Object Scans* (Choi et al. 2016), and compare to the provided

reconstructions in Fig. 14. Note that in order to stay within real-time constraints, we used a voxel size of 8 mm for tracking. Even though the dataset reconstructions are obtained via a combination of ICP's geometric error with the photoconsistency of DVO-full, SDF-2-SDF manages to better recover challenging details such as chair legs and support beams over long sequences with thousands of frames.


**Fig. 11** SDF-2-SDF reconstructions of industrial and Kinect scans from the *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016)

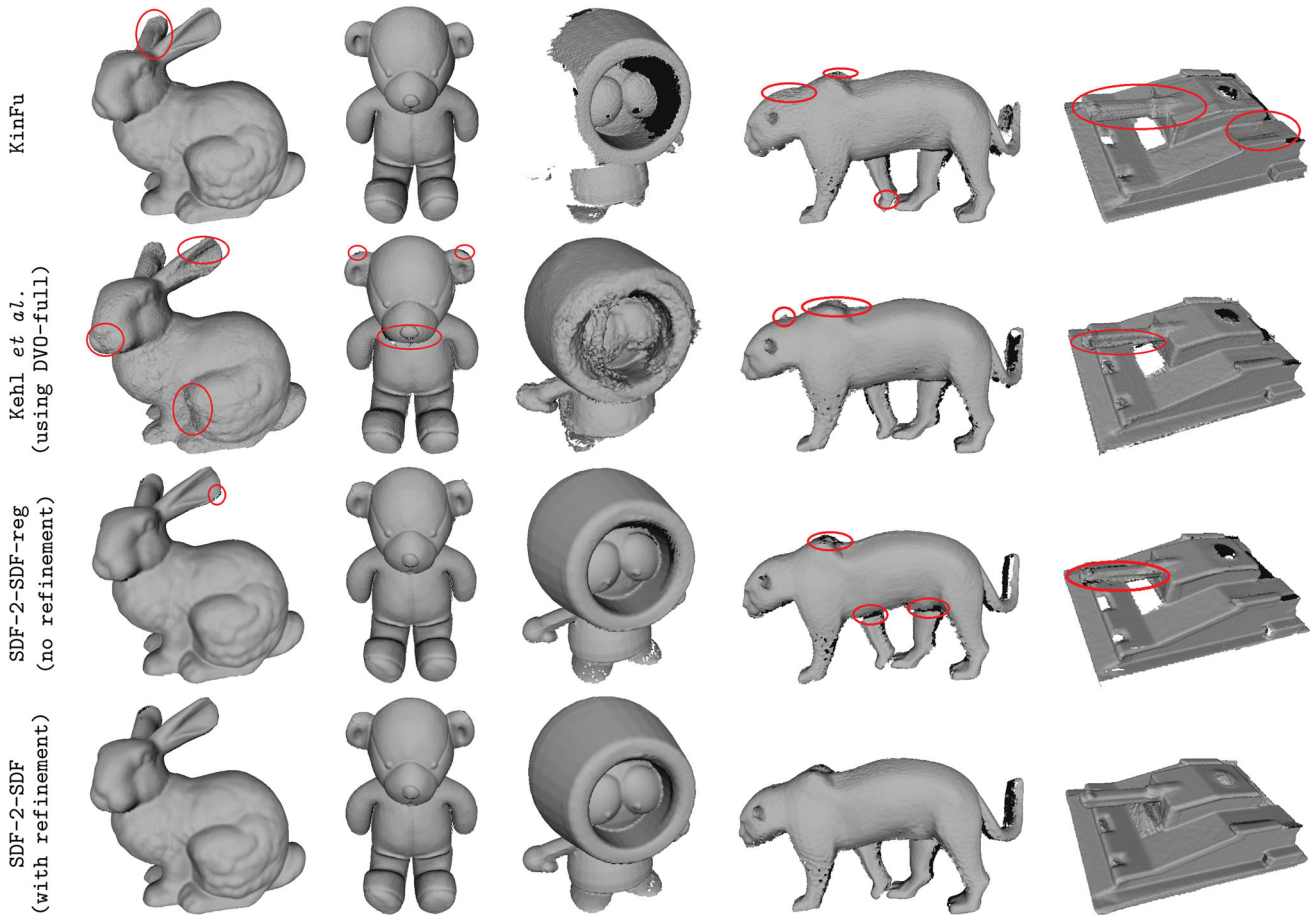



**Fig. 12** Qualitative comparison of untextured reconstructions from scans with the high quality industrial sensor of the *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016). Object poses might appear slightly different, since models yielded by different methods are non-identical. Fine structures cause related approaches to fail (on *Kenny*) or to exhibit misalignment errors (on *bunny*'s ears, *tank*'s gun, connection of *leopard*'s halves)

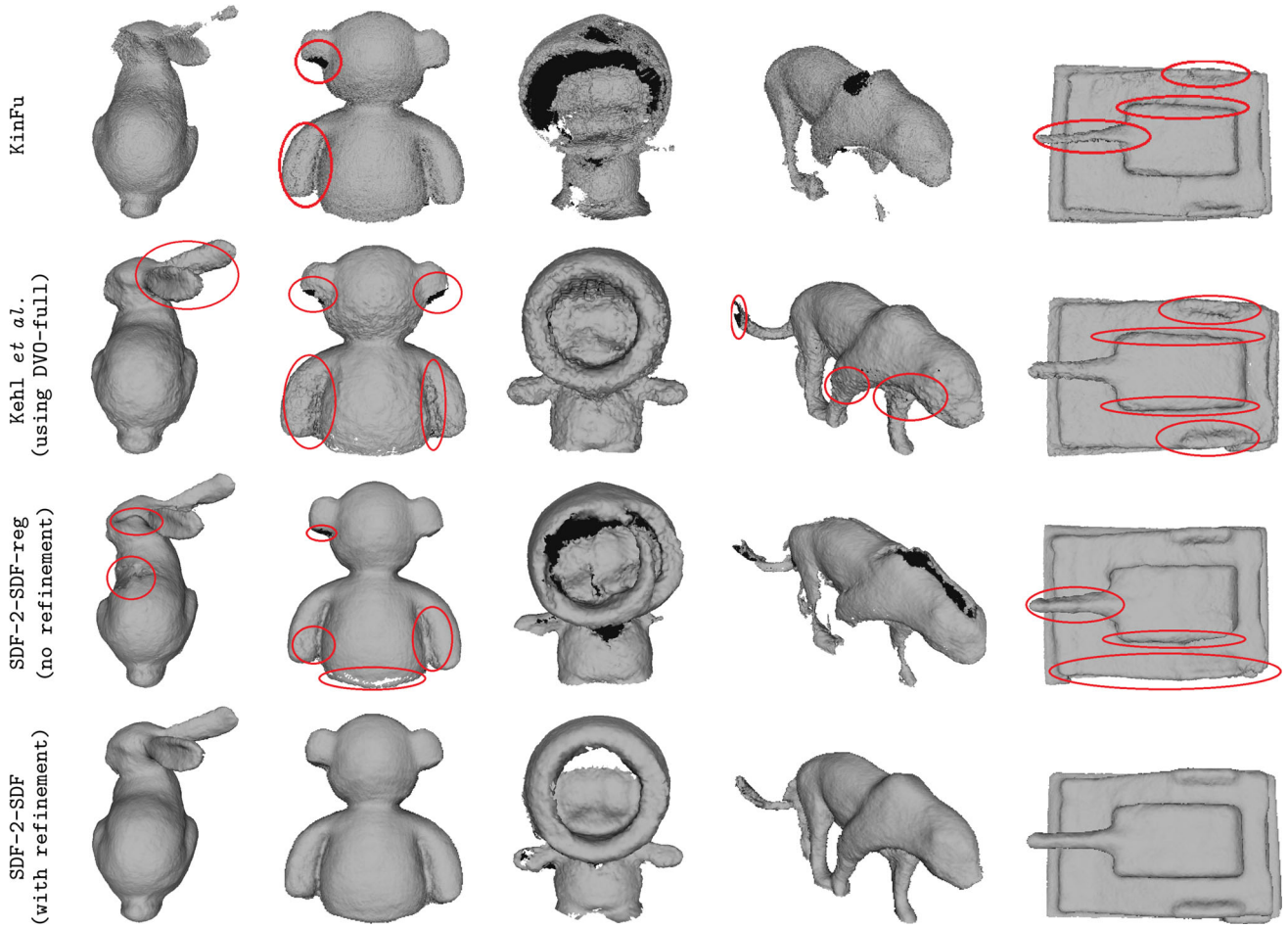



**Fig. 13** Qualitative comparison of untextured reconstructions from Kinect scans of the *3D-Printed Object Dataset* (Slavcheva et al. 2016)

*Runtime* As previously mentioned, thanks to SSE instructions and 8-core parallel processing over the narrow band, SDF-2-SDF-reg tracking runs at 17–22 FPS on household objects when a voxel size of 2 mm is used. Tables 3 and 4 list the time taken for each major step of our pipeline as average over all sequences, as well as the fastest (achieved on *synthetic Kenny*) and slowest (on *Kinect leopard*) runs.

Refinement requires at most 40 iterations on each voxel resolution level, taking up to 30 s to deliver the reconstruction, which is generated via marching cubes from the final field. In comparison, Kehl et al.'s pose graph optimization took 196.4 s on average (minimum 53 s, maximum 902 s) for the same amount of keyframes. Table 4 shows that our pose-graph-free refinement is considerably faster.

### 6.3 Large Objects, Scenes and SLAM

We now switch our focus to evaluating the modification of our method, SDF-TAR, that permits it to track the camera entirely online when scanning bigger objects and scenes. As the SDF-2-SDF energy was designed for object reconstruction, our main objective is to adapt it to larger-scale objects, such as furniture items and industrial parts. Nevertheless, we also investigate its applicability to SLAM scenarios and compare to state-of-the art volumetric approaches.

*Modification Effect* Our expectation is that the numerical accuracy of SDF-TAR is slightly inferior to SDF-2-SDF due to the decreased density, while by design it runs in real time executing both tracking and refinement concurrently. Therefore, as a proof of concept, we examine the error on all Kinect sequences of the *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016) and compare the two versions of our approach, as shown in Fig. 15.

We used a standard parameter setting for SDF-TAR (details will follow in the next section) with a voxel size of 2 mm, equal to that used in SDF-2-SDF. The outcome confirms our expectation that performance is degraded slightly. Nevertheless, the errors remain lower than the majority of other methods examined in Fig. 7. Moreover, the error on the challenging handheld *Kenny* sequence is significantly decreased. Therefore, SDF-TAR is a promising modification of SDF-2-SDF, which will make our dense

**Table 2** *CloudCompare* evaluation of the absolute cloud-to-model reconstruction error on the *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016)

| Object | Method | Mean error (mm) | | |
|---|---|---|---|---|
| | | Industr. turntable | Kinect turntable | Kinect handheld |
| Bunny | KinFu | 0.664 | 3.800 | 4.101 |
| | SDF-2-SDF-reg | 0.656 | 2.586 | 1.770 |
| | Kehl-object | 2.149 | 5.156 | 8.274 |
| | Kehl-full | 0.838 | 1.134 | 1.124 |
| | SDF-2-SDF | **0.541** | **0.953** | **0.996** |
| Teddy | KinFu | 0.998 | 1.271 | 2.355 |
| | SDF-2-SDF-reg | 0.930 | 1.078 | 1.589 |
| | Kehl-object | 1.028 | 2.306 | 2.287 |
| | Kehl-full | 4.828 | 1.221 | 3.066 |
| | SDF-2-SDF | **0.910** | **0.722** | **0.990** |
| Kenny | KinFu | 1.650 | 1.511 | 2.874 |
| | SDF-2-SDF-reg | 0.363 | 1.295 | 2.415 |
| | Kehl-object | 1.816 | 3.181 | failed |
| | Kehl-full | 2.553 | **1.263** | **2.282** |
| | SDF-2-SDF | **0.315** | 1.276 | 2.358 |
| Leopard | KinFu | 1.785 | 4.445 | 1.886 |
| | SDF-2-SDF-reg | 0.760 | 2.692 | 1.321 |
| | Kehl-object | 1.018 | 5.693 | failed |
| | Kehl-full | 3.626 | 1.907 | 1.281 |
| | SDF-2-SDF | **0.652** | **1.308** | **1.263** |
| Tank | KinFu | 1.390 | 1.561 | 2.579 |
| | SDF-2-SDF-reg | 0.953 | 1.336 | 2.042 |
| | Kehl-object | 1.573 | 1.192 | 2.340 |
| | Kehl-full | 2.617 | 1.064 | **0.946** |
| | SDF-2-SDF | **0.466** | **0.911** | 1.508 |

SDF-2-SDF-reg refers to our method without refinement, while SDF-2-SDF includes refinement. The variants of Kehl et al.'s pipeline (2014) indicate whether DVO-object or DVO-full was used for tracking
Bold values indicate the lowest error

implicit-to-implicit energy applicable to large spaces and SLAM scenarios.

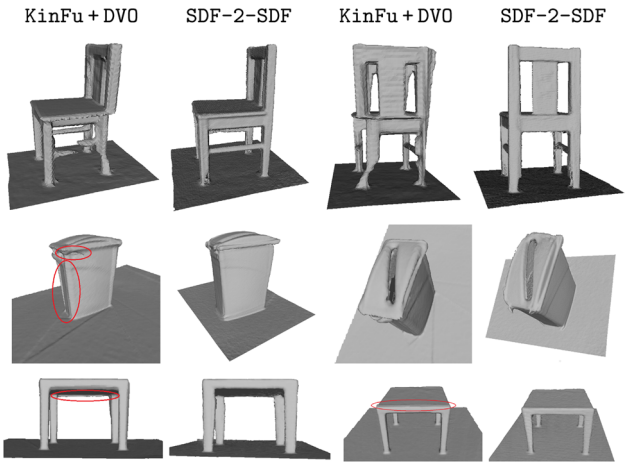



**Fig. 14** Qualitative comparison versus reconstructions in the *Large Dataset of Object Scans* (Choi et al. 2016)

**Table 3** SDF-2-SDF-reg tracking module runtime statistics on the *3D-Printed Dataset* (Slavcheva et al. 2016): average/ fastest/ *slowest*

| Tracking (milliseconds per frame) | | |
|---|---|---|
| Pre-process | Reference SDF generation | Minimization iterations |
| 1.7/ 1.6/ *1.8* | 2.6/ 1.7/ *3.8* | 45.3/ 41.4/ *54.9* |
| Overall 49.6/ 44.7/ *60.5* ms = 20/ 22/ 17 FPS | | |

**Table 4** SDF-2-SDF refinement module runtime statistics on the *3D-Printed Dataset* (Slavcheva et al. 2016): average/ fastest/ *slowest*

| Refinement (total seconds) | | |
|---|---|---|
| Weighted averages | Optimizing poses | Marching cubes |
| 1.9/ 0.4/ *6.8* | 6.1/ 0.3/ *20.2* | 0.6/ 0.2/ *1.3* |
| Overall 8.6/ 0.9/ *28.3* s | | |

*Method Parameters* The resolution of a single LEV SDF is $8^3$ voxels, with side 8 mm for tracking and 4 mm for refinement. While this finer voxel size is advantageous for more accurate

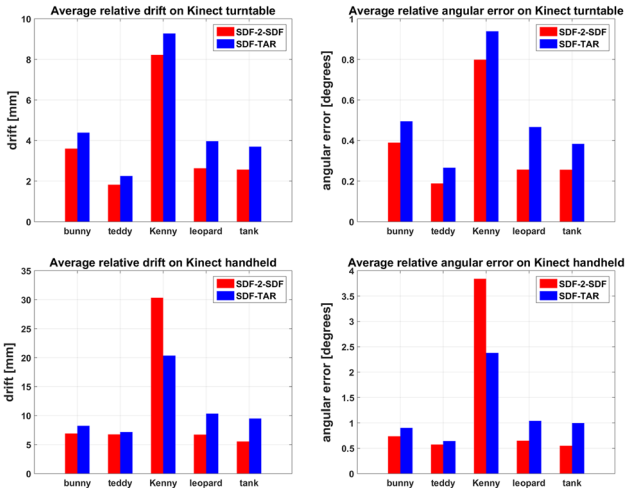



**Fig. 15** Comparison of SDF-2-SDF and SDF-TAR on Kinect data from the *3D-Printed RGB-D Object Dataset* (Slavcheva et al. 2016). SDF-TAR decreases accuracy only slightly, while ensuring tracking and refinement are accomplished within real time constraints

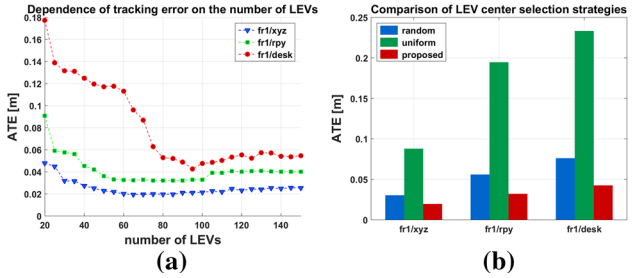



**Fig. 16** Parameter analysis of SDF-TAR: influence of **a** the number of LEVs used and **b** the LEV anchor point selection strategy on the absolute trajectory error on sequences from the TUM RGB-D benchmark (Sturm et al. 2017)

refinement, an even smaller one is not beneficial because it would become corrupted by sensor noise. The $\delta$ parameter equals the voxel size, while $\eta$ is twice the voxel size, as they control the represented surface region. Independent of how many LEVs are used for tracking, only $n = 8$ are used for refinement, since a good initialization is available and since generating them for a whole batch of frames on the CPU would otherwise take too much time. The batch size is 20 frames ($b = 10$), while the weighted average is generated on every $f = 5$th iteration.

While all parameters of SDF-TAR reflect the inherent properties of the environment, some of the remaining ones depend on the richness of the scanned geometry. In the following we assess the susceptibility of trajectory estimation accuracy to them on three sequences of the TUM RGB-D benchmark (Sturm et al. 2017): *fr1/xyz* and *fr1/rpy*, which are designed for evaluating translational and rotational motion estimation respectively, and *fr1/desk* which is a typical SLAM scenario combining both kinds of motion. We evaluate the root mean squared absolute trajectory error (ATE) proposed in the benchmark. In order to isolate the effect of the parameters on the partial volume registration, we disable the refinement module for this test.

To judge the dependence of the tracking error on the *number of volumes*, we take from 20 to 150 LEVs per frame. The results in Fig. 16a show that the error is rather large with a small number of volumes, and gradually decreases with more LEVs. There is quite a broad range of values which lead to near-optimal results, typically around 60–90 volumes. When the LEV number becomes too high, the error slightly increases again. This means that the volumes have become so many that they also cover flat regions, which inhibit registration. Naturally, in order to keep runtime as low as possible, we advocate taking the smallest amount that guarantees stable results, e.g. 80 LEVs per frame.

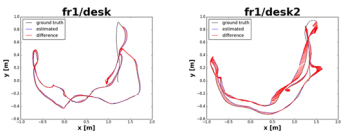
We compare our *LEV anchor point selection strategy*, which determines where the partial SDFs are centered, to two other approaches that can be applied directly over a depth map. In them the image is split into non-overlapping windows of $w \times w$ pixels, one pixel is selected per window and back-projected to 3D to give the anchor point. The *uniform* approach takes the center of each window, while the *random* strategy selects a pixel at random. For all approaches we first pre-process the depth map, as explained (cf. Fig. 4), to discard invalid regions, and then take the same number of LEVs (the amount that gave optimal results in the experiment above for the respective sequence). Figure 16b shows that the uniform strategy leads to a 4–6 times higher error than our proposal, while the random sampling is nearly twice worse than ours. Thus our strategy clearly selects more discriminative regions that, combined with its high speed, are more advantageous for registration.

Finally, we switch the *refinement module* on, observing a decrease in ATE error on *fr1/xyz* by only 19%, while on *fr1/rpy* it reduced more than 50%. Not surprisingly, on the combined motion sequence *fr1/desk* the improvement was in between: 41%. We, therefore, conclude that our refinement strategy is highly beneficial for reducing the rotational error in tracking. We attribute this to the small volumes that only encapsulate informative context around salient locations. On the contrary, motion between flat regions can only be estimated as sliding against each other, which would inhibit accurate rotation estimation.

Furthermore, we tried an every-frame *refinement strategy*, whereby we used the same frame to weighted average registration, but only optimizing the last tracked pose. This refinement lead to a very slight improvement over the non-optimized trajectory. The reason is that the energy for every-frame refinement is too similar to the tracking one, so it cannot significantly improve the pose, while the batch refinement has multiple frames influencing each other, resulting in

**Table 5** Absolute trajectory error (ATE) (meters) on sequences from the *TUM RGB-D benchmark* (Sturm et al. 2017)

| Method | fr1/xyz | fr1/rpy | fr1/desk | fr1/desk2 | fr1/360 | fr1/floor |
|---|---|---|---|---|---|---|
| KinFu (PCL 2017) | 0.023 | 0.081 | 0.057 | 0.102 | 0.591 | 0.918 |
| FM-pt-SDF (Bylow et al. 2013) | 0.021 | 0.042 | 0.035 | **0.061** | 0.119 | 0.567 |
| FM-pt-SDF (Canelhas et al. 2013) | **0.014** | – | 0.033 | 0.230 | – | 0.984 |
| SDF-TAR | 0.015 | **0.021** | 0.030 | 0.091 | **0.113** | **0.279** |




Our method achieves a considerably smaller error when the dominant motion is rotational (e.g. *rpy*, *360*), and demonstrates comparable accuracy under general motion. Qualitative examples of estimated trajectories are shown on the right
Bold values indicate the lowest error

**Table 6** Relative pose error (RPE) translational (meters/frame) and rotational (°/frame) root-mean squared values per frame on TUM RGB-D benchmark (Sturm et al. 2017) sequences

| Method | fr1/xyz | | fr1/rpy | | fr1/desk | | fr1/desk2 | | fr1/360 | | fr1/floor | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | tr. | rot. | tr. | rot. | tr. | rot. | tr. | rot. | tr. | rot. | tr. | rot. |
| KinFu (PCL 2017) | 0.004 | 0.474 | – | – | 0.020 | 2.003 | 0.020 | 1.795 | – | – | 0.035 | 1.718 |
| FM-pt-SDF (Canelhas et al. 2013) | **0.003** | 0.472 | – | – | 0.007 | **0.759** | 0.019 | 1.080 | – | – | 0.050 | 2.085 |
| SDF-TAR | **0.003** | **0.442** | 0.004 | 1.042 | **0.006** | 0.768 | **0.009** | **0.993** | **0.011** | **1.514** | **0.020** | **0.844** |

SDF-TAR outperforms the other methods on nearly all examples
Bold values indicate the lowest error

**Table 7** Relative pose error (RPE) translational (m/s) and rotational (°/s) root-mean squared values per second on CoRBS dataset (Wasenmüller et al. 2016b) sequences

| Method | Desk D1 | | Cabinet E1 | | Human H1 | |
|---|---|---|---|---|---|---|
| | tr. | rot. | tr. | rot. | tr. | rot. |
| DNA-SLAM (Wasenmüller et al. 2016a) | 0.027 | 0.970 | 0.035 | 1.426 | **0.020** | **0.725** |
| KinFu (PCL 2017) | **0.026** | 1.739 | 0.045 | 1.047 | 0.034 | 1.626 |
| FM-pt-SDF (Canelhas 2017) | 0.032 | 1.753 | 0.033 | 1.731 | 0.041 | 1.891 |
| SDF-TAR | 0.030 | **0.964** | **0.032** | **0.990** | 0.037 | 1.456 |

Bold values indicate the lowest error

**Table 8** *CloudCompare* absolute cloud-to-model error (centimeters) on objects from the *CoRBS dataset* (Wasenmüller et al. 2016b)

| Method | Desk D1 | Cabinet E1 | Human H1 |
|---|---|---|---|
| KinFu PCL (2017) | 1.5686 | 1.2504 | 0.7105 |
| FM-pt-SDF Canelhas (2017) | 1.3266 | 1.1599 | **0.6583** |
| SDF-TAR | **0.9856** | **1.0552** | 0.7258 |

Bold values indicate the lowest error

better estimates. Thus we have developed a powerful strategy that can be applied in parallel with the tracking module and significantly reduces rotational drift.

*SLAM* We continue our quantitative evaluation on one of the most widely used publicly available datasets, the TUM RGB-D benchmark (Sturm et al. 2017), and therefore now assess the SLAM capabilities of SDF-TAR. We compare against state-of-the-art systems that rely on SDFs for registration: KinFu (PCL 2017) and point-to-implicit methods (Bylow et al. 2013; Canelhas et al. 2013). We cite the values reported in the respective papers, and run our SDF-TAR with the standard setting outlined in the previous section.

The absolute and relative tracking errors are summarized in Tables 5 and 6 respectively. The ATE testifies that SDF-TAR considerably outperforms related works on sequences with dominant rotational motion, and achieve on-par or better accuracy on general types of motion. Moreover, our relative rotational drift is well below 1° even on the challenging *fr1/floor* sequence. We, therefore, conclude that the LEVs reduce the negative influences of noise, blur and rolling shutter effect by constraining registration to the most discriminative local geometry, and effectively avoiding regions that typically impede accuracy, such as flat surfaces.

*Large Objects* Finally, we assess the performance of SDF-TAR on the task of reconstructing large-scale objects. To this end, we make use of the CoRBS dataset (Wasenmüller et al. 2016b), since it contains real Kinect v2 captures of items with externally recovered CAD models, enabling both tracking

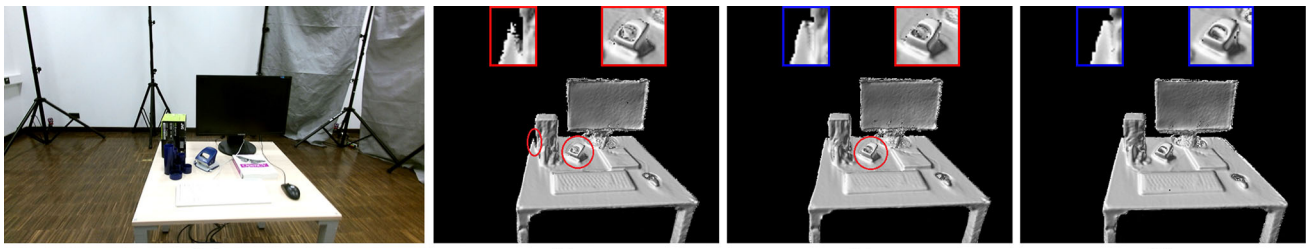



**Fig. 17** Qualitative comparison on Desk1 from the CoRBS dataset (Wasenmüller et al. 2016b): related approaches wash out fine structures due to drift (marked in red), while the concurrent refinement in SDF-TAR reduces it, yielding more detailed, higher fidelity results. **a** Color image, **b** KinFu (PCL 2017), **c** FM-pt-SDF (Canelhas 2017), **d** SDF-TAR (Color figure online)

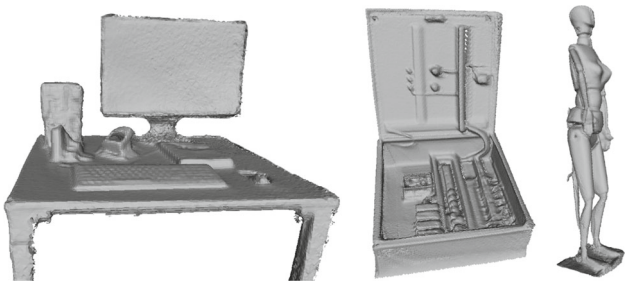


**Fig. 18** SDF-TAR results on *CoRBS dataset* (Wasenmüller et al. 2016b) sequences

and reconstruction evaluation. Since the dataset is relatively new, we run KinFu (PCL 2017) and the ROS version of FM-pt-SDF (Canelhas 2017) ourselves. For all tests we used a voxel size of 8 mm, while other parameters were set to the most advantageous ones defined by the respective authors of each approach. In addition, we include results from DNA-SLAM (Wasenmüller et al. 2016a), which is a SLAM system from the authors of the CoRBS dataset, specifically designed for time-of-flight cameras, but, unfortunately, reporting only RPE values and no model errors.

Table 7 provides an overview of the relative trajectory errors per second. KinFu (PCL 2017) and FM-pt-SDF (Canelhas 2017) perform similarly, as was often the case for small-scale objects, while our SDF-TAR and DNA-SLAM achieve higher precision. In some cases DNA-SLAM outperforms us, since it is specifically designed for this kind of depth sensor. Nevertheless, SDF-TAR still demonstrates excellent rotational motion estimation.

The *CloudCompare* results in Table 8 exhibit a similar trend. We achieve the smallest model error on most objects, which we attribute to the smaller rotational drift, combined with the benefit of online refinement. This proves that SDF-TAR has successfully leveraged SDF-2-SDF registration to large volumes of interest.

Finally, we assess our overall outcomes on the RGB-D benchmark (Sturm et al. 2017) and CoRBS (Wasenmüller et al. 2016b) in order to judge performance under outward-facing SLAM trajectories versus inward-facing object scanning ones (Chen et al. 2013) (Figs. 17, 18). We observe that our energy is well suited both for small- and large-scale object reconstruction, where the motion is object-centered. It performs on-par with other methods under more challenging SLAM motion, but is nevertheless very accurate under rotational motion.

## 7 Conclusion

We have presented a dense, correspondence-free, symmetric, volumetric energy for registering pairs of implicit representations. We have applied it for highly accurate object reconstruction in SDF-2-SDF, where online frame-to-frame tracking is followed by swift global frame-to-model optimization. Then, we have demonstrated how to modify it into a fully real-time, hybrid CPU/GPU system, SDF-TAR, which handles larger volumes. Through multiple qualitative and quantitative evaluations of various aspects of our methods, we have shown their advantages over state-of-the-art techniques, such as wider convergence basin, better rotational motion estimation and reconstruction fidelity.

## References

Adalsteinsson, D., & Sethian, J. A. (1995). A fast level set method for propagating interfaces. *Journal of Computational Physics*, *118*(2), 269–277.

Alexandre, L. A. (2012). 3D descriptors for object and category recognition: A comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

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

Blender Project: Free and open 3D creation software. https://www.blender.org/. Last Accessed March 30, 2017.

Bo, L., Ren, X., & Fox, D. (2011). Depth Kernel descriptors for object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

Bylow, E., Olsson, C., & Kahl, F. (2014). Robust camera tracking by combining color and depth measurements. In *International Conference on Pattern Recognition* (*ICPR*).

Bylow, E., Sturm, J., Kerl, C., Kahl, F., & Cremers, D. (2013). Real-time camera tracking and 3D reconstruction using signed distance functions. In *Robotics: Science and Systems Conference* (*RSS*).

Canelhas, D. (2017). sdf_tracker - ROS Wiki. http://wiki.ros.org/sdf_tracker. Last Accessed March 30, 2017.

Canelhas, D. R., Stoyanov, T., & Lilienthal, A. J. (2013). SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

Chen, Y., & Medioni, G. (1991). Object modeling by registration of multiple range images. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Chen, J., Bautembach, D., & Izadi, S. (2013). Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics*, *32*(4), 113.

Choi, S., Zhou, Q. Y., & Koltun, V. (2015) Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*).

Choi, S., Zhou, Q., Miller, S., & Koltun, V. (2016). A large dataset of object scans. arXiv:1602.02481.

Clarenz, U., Rumpf, M., & Telea, A. (2004). Robust feature detection and local classification for surfaces based on moment analysis. *IEEE Transactions on Visualization and Computer Graphics*, *10*(5), 516–524.

CloudCompare: 3D point cloud and mesh processing software. http://www.danielgm.net/cc/. Last Accessed March 30, 2017.

Curless, B., & Levoy, M. (1996). A volumetric method for building complex models from range images. In *23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96*, pp. 303–312.

Dimashova, M., Lysenkov, I., Rabaud, V., & Eruhimov, V. (2013). Tabletop object scanning with an RGB-D sensor. In *Third Workshop on Semantic Perception, Mapping and Exploration (SPME) at the 2013 IEEE International Conference on Robotics and Automation* (*ICRA*).

Drost, B., Ulrich, M., Navab, N., & Ilic, S. (2010) Model globally, match locally: Efficient and robust 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*).

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., & Burgard, W. (2012). An evaluation of the RGB-D SLAM system. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Fioraio, N., Taylor, J., Fitzgibbon, A., Di Stefano, L., & Izadi, S. (2015). Large-scale and drift-free surface reconstruction using online subvolume registration. In *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*).

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Gelfand, N., Mitra, N. J., Guibas, L. J., & Pottmann, H. (2005). Robust global registration. In *Third Eurographics Symposium on Geometry Processing* (*SGP*).

Henry, P., Fox, D., Bhowmik, A., & Mongia, R. (2013). Patch volumes: Segmentation-based consistent mapping with RGB-D cameras. In *International Conference on 3D Vision* (3DV).

Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2010). RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *International Symposium on Experimental Robotics*.

Holzer, S., Shotton, J., & Kohli, P. (2012). Learning to efficiently detect repeatable interest points in depth data. In *European Conference on Computer Vision* (*ECCV*).

Houston, B., Nielsen, M. B., Batty, C., Nilsson, O., & Museth, K. (2006). Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Transactions on Graphics (TOG)*, *25*(1), 151–175.

Ioannou, Y., Taati, B., Harrap, R., & Greenspan, M. A. (2012). Difference of normals as a multi-scale operator in unorganized point clouds. In *Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission* (*3DIMPVT*).

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., & Fitzgibbon, A. (2011). KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology* (*UIST*).

Johnson, A. E., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *21*(5), 433–449.

Johnson, A., & Kang, S. B. (1999). Registration and integration of textured 3D Data. *Image and Vision Computing*, *17*, 135–147.

Kähler, O., Prisacariu, V. A., Ren, C. Y., Sun, X., Torr, P., & Murray, D. (2015). Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, *21*(11), 1241–1250.

Kehl, W., Holl, T., Tombari, F., Ilic, S., & Navab, N. (2016). An Octree-based approach towards efficient variational range data fusion. In *British Machine Vision Conference* (*BMVC*).

Kehl, W., Navab, N., & Ilic, S. (2014). Coloured signed distance fields for full 3D object reconstruction. In *Proceedings of the British Machine Vision Conference* (*BMVC*).

Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., & Kolb, A. (2013). Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision* (*3DV*).

Kerl, C. (2017). GitHub—tum-vision/dvo: Dense Visual Odometry. https://github.com/tum-vision/dvo. Last accessed March 30, 2017.

Kerl, C., Sturm, J., & Cremers, D. (2013). Robust odometry estimation for RGB-D cameras. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Khoshelham, K., & Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, *12*(2), 1437–1454.

KinectFusion implementation in the point cloud library (PCL). https://github.com/PointCloudLibrary/pcl/tree/master/gpu/kinfu. Last Accessed March 30, 2017.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality* (*ISMAR*).

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Lai, K., Bo, L., Ren, X., & Fox, D. (2011). A large-scale hierarchical multi-view RGB-D object dataset. In *International Conference on Robotics and Automation* (*ICRA*).

Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: a high resolution 3D surface construction algorithm. In *14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87*.

Losasso, F., Fedkiw, R., & Osher, S. (2006). Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, *35*(10), 995–1010.

Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. S. (2003). *An invitation to 3-D vision: From images to geometric models*. Berlin: Springer.

Masuda, T. (2002). Registration and integration of multiple range images by matching signed distance fields for object shape modeling. *Computer Vision and Image Understanding (CVIU)*, *87*(1–3), 51–65.

Meilland, M., & Comport, A. I. (2013). On unifying key-frame and Voxel-based dense visual SLAM at large scales. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

Narayan, K. S., Sha, J., Singh, A., & Abbeel, P. (2015). Range sensor and silhouette fusion for high-quality 3D scanning. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Neubeck, A., & Van Gool, L. (2006). Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition* (*ICPR*).

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., & Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *10th International Symposium on Mixed and Augmented Reality* (*ISMAR*).

Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision* (*ICCV*).

Nielsen, M. B., & Museth, K. (2006). Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *Journal of Scientific Computing*, *26*(3), 261–299.

Nießner, M., Zollhöfer, M., Izadi, S., & Stamminger, M. (2013). Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, *32*, 169.

Osher, S., & Fedkiw, R. (2003). *Level set methods and dynamic implicit surfaces. Applied mathematical science* (Vol. 153). Springer.

Pirker, K., Rüther, M., Schweighofer, G., & Bischof, H. (2011). GPSlam: Marrying sparse geometric and dense probabilistic visual mapping. In *Proceedings of the British Machine Vision Conference* (*BMVC*).

Point Cloud Library. http://pointclouds.org/. Last Accessed March 30, 2017.

Ren, C. Y., & Reid, I. (2012). A unified energy minimization framework for model fitting in depth. In *European Conference on Computer Vision 2nd Workshop on Consumer Depth Cameras* (*ECCVW*).

Roth, H., & Vona, M. (2012). Moving volume KinectFusion. In *British Machine Vision Conference* (*BMVC*).

Rusinkiewicz, S., & Levoy, M. (2001). Efficient variants of the ICP algorithm. In *3rd International Conference on 3D Digital Imaging and Modeling* (*3DIM*).

Rusu, R. B., Holzbach, A., Blodow, N., & Beetz, M. (2009). Fast geometric point labeling using conditional random fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

Schütz, C., Jost, T., & Hugli, H. (1998). Multi-feature matching algorithm for free-form 3D surface registration. In *International Conference on Pattern Recognition* (*ICPR*).

Segal, A., Haehnel, D., & Thrun, S. (2009). Generalized-ICP. In *Robotics: Science and Systems* (*RSS*).

Singh, A., Sha, J., Narayan, K., Achim, T., & Abbeel, P. (2014) BigBIRD: A large-scale 3D database of object instances. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Slavcheva, M., & Ilic, S. (2016). SDF-TAR: Parallel tracking and refinement in RGB-D data using volumetric registration. In *British Machine Vision Conference* (*BMVC*).

Slavcheva, M., Kehl, W., Navab, N., & Ilic, S. (2016). SDF-2-SDF: Highly accurate 3D object reconstruction. In *European Conference on Computer Vision* (*ECCV*).

Steder, B., Rusu, R. B., Konolige, K., & Burgard, W. (2010). NARF: 3D range image features for object recognition. In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

Steinbrücker, F., Kerl, C., Sturm, J., & Cremers, D. (2013). Large-scale multi-resolution surface reconstruction from RGB-D sequences. In *IEEE International Conference on Computer Vision* (*ICCV*).

Steinbrücker, F., Sturm, J., & Cremers, D. (2014). Volumetric 3D mapping in real-time on a CPU. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2017). A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of the International Conference on Intelligent Robot Systems* (*IROS*).

Tomasi, C., & Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Sixth IEEE International Conference on Computer Vision(ICCV)*, pp. 839–846.

Tombari, F., Salti, S., & Di Stefano, L. (2013). Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision (IJCV)*, *102*(1), 198–220.

Vijayanagar, K. R., Loghman, M., & Kim, J. (2014). Real-time refinement of Kinect depth maps using multi-resolution anisotropic diffusion. *Mobile Networks and Applications*, *19*(3), 414–425.

Wasenmüller, O., Ansari, M., & Stricker, D. (2016) DNA-SLAM: Dense noise aware SLAM for ToF RGB-D cameras. In *Asian Conference on Computer Vision (ACCV) International Workshops*.

Wasenmüller, O., Meyer, M., & Stricker, D. (2016). CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2. In *IEEE Winter Conference on Applications of Computer Vision* (*WACV*).

Whelan, T., Johannsson, H., Kaess, M., Leonard, J. J., & McDonald, J. B. (2013). Robust real-time visual odometry for dense RGB-D mapping. In *IEEE International Conference on Robotics and Automation* (*ICRA*).

Whelan, T., Kaess, M., Leonard, J. J., & McDonald, J. (2013). Deformation-based loop closure for large scale dense rgb-d slam. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*).

Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., & Davison, A.J. (2015). ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems* (*RSS*).

Whelan, T., McDonald, J. B., Kaess, M., Fallon, M. F., Johannsson, H., & Leonard, J. J. (2012). Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*.

Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., & Leutenegger, S. (2016). ElasticFusion: Real-time dense SLAM and light source estimation. *International Journal of Robotics Research (IJRR)*, *35*(14), 1697–1716.

Whitaker, R. T. (1998). A level-set approach to 3D reconstruction from range data. *International Journal of Computer Vision (IJCV)*, *29*(3), 203–231.

Zach, C., Pock, T., & Bischof, H. (2007). A globally optimal algorithm for robust TV-$L^1$ range image integration. In *Proceedings of the 11th IEEE International Conference on Computer Vision* (*ICCV*), pp. 1–8.

ZCorporation: ZPrinter 650. Hardware manual (2008).

Zeng, M., Zhao, F., Zheng, J., & Liu, X. (2013). Octree-based fusion for realtime 3D reconstruction. *Graphical Models*, *75*(3), 126–136.

Zhou, Q., Miller, S., & Koltun, V. (2013). Elastic fragments for dense scene reconstruction. In *IEEE International Conference on Computer Vision* (*ICCV*).

Zhou, Q., & Koltun, V. (2013). Dense scene reconstruction with points of interest. *ACM Transactions on Graphics*, *32*(4), 112.