# SDF-TAR: Parallel Tracking and Refinement in RGB-D Data using Volumetric Registration

Miroslava Slavcheva
mira.slavcheva@tum.de
Technische Universität München
Munich, Germany

Slobodan Ilic
slobodan.ilic@siemens.com
Siemens AG
Munich, Germany

**Abstract**

This paper introduces SDF-TAR: a real-time SLAM system based on volumetric registration in RGB-D data. While the camera is tracked online on the GPU, the most recently estimated poses are jointly refined on the CPU. We perform registration by aligning the data in limited-extent volumes anchored at salient 3D locations. This strategy permits efficient tracking on the GPU. Furthermore, the small memory load of the partial volumes allows for pose refinement to be done concurrently on the CPU. This refinement is performed over batches of a fixed number of frames, which are jointly optimized until the next batch becomes available. Thus drift is reduced during online operation, eliminating the need for any posterior processing. Evaluating on two public benchmarks, we demonstrate improved rotational motion estimation and higher reconstruction precision than related methods.

## 1 Introduction

Real-time Simultaneous Localization and Mapping (SLAM) is among the most pivotal computer vision tasks, with many commercial applications ranging from robotic navigation and scene reconstruction to augmented and virtual reality. Equipped with a hand-held camera, the goal is to explore a static environment, simultaneously determining the 6 degrees-of-freedom camera pose at every instance and reconstructing the surroundings.

A new wave of research was elicited with the advent of inexpensive RGB-D sensors, which eliminate the inherent scale problem of monocular SLAM. The earliest works [12, 18] relied on visual features to match 3D locations via variants of the Iterative Closest Points (ICP) [2, 7] algorithm. Soon after, the seminal KinectFusion system [22, 34] demonstrated the advantages of volumetric registration through the use of a continuously incremented truncated signed distance field (SDF) to represent the estimated scene geometry. Various related approaches have proposed improvements to the registration energy [4, 5, 24] and strategies to tackle the memory limitations of regular grid SDFs, such as moving volumes [40, 48, 49], octrees [43, 44, 52] and voxel hashing [36].

Although frame-to-growing-model registration 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 [53] or meshed scene fragments [8, 14, 19, 54] to be stored and lead to lengthy posterior refinement.

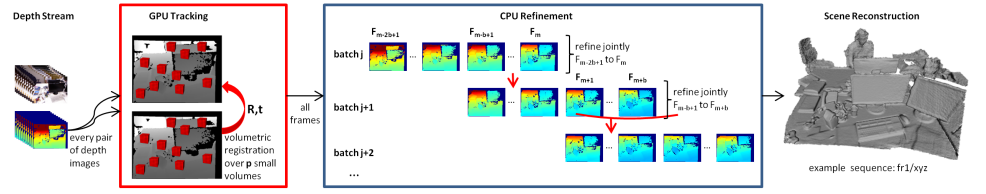



Figure 1: SDF-TAR pipeline: the relative motion between every two frames is estimated on the GPU from $p$ small volumes. As soon as frame $F_m$ is tracked, the CPU refinement module starts jointly optimizing $F_{m-2b+1}$ to $F_m$. In the meantime tracking resumes on $F_{m+1}$ to $F_{m+b}$. Once this new batch is ready, the refinement module switches to $F_{m-b+1}$ to $F_{m+b}$. This strategy ensures that every pose is optimized twice for optimal geometric consistency.

One of the most acclaimed real-time monocular SLAM techniques, PTAM [28], solves these issues by combining tracking in one thread with global map refinement in another. Inspired by this, we propose a similar framework in RGB-D settings. The key idea is to enable concurrent execution by unifying the efficiency of sparse interest point alignment with the accuracy of dense volumetric approaches. This is motivated by our recent SDF-2-SDF registration method [41] that aligns pairs of SDF grids, resulting in more precise motion estimation than ICP and point-to-implicit approaches [4, 5]. While related methods, such as KinectFusion, register over an amount of data equal to the depth map resolution, SDF-2-SDF processes all voxels. In addition to the associated high memory requirements, the atomicity of the underlying reduction operations prevents profiting from massive GPU parallelization, thus restricting operation to small spaces that are insufficient for SLAM. Guided by the intuition that geometry-poor locations impede registration, we propose to select a fixed number of the most geometry-rich locations in a range image, and anchor small SDF volumes of fixed size around them. Thus only informative data is used for registration, yielding the accuracy of fully dense techniques, at a fraction of the cost. Furthermore, this strategy is more straightforward to implement than moving volumes, octrees and voxel hashing. It enables us to apply SDF-2-SDF registration in parallel over all volumes on the GPU, seeking a common rigid-body motion, and to additionally perform concurrent pose refinement on the CPU, thus minimizing drift in real-time without the need for posterior global optimizaiton.

To sum up, we propose SDF-TAR: a real-time system for parallel tracking and refinement based on direct registration between multiple limited-extent SDFs, summarized in Figure 1. Our contributions are: (1) a novel approach for reducing the memory footprint of volumetric registration, while preserving its accuracy; and (2) a fully real-time volumetric SLAM method which combines GPU tracking with concurrent CPU pose refinement on overlapping batches of RGB-D frames for online drift reduction. The limited-extent volumes (LEVs) lead to more precise tracking than state-of-the-art techniques when the dominant motion is rotational, and on-par accuracy in general settings. Furthermore, we assess the drift reduction achieved by refinement, which is also reflected in higher-fidelity reconstructions.

## 2 Related Work

KinectFusion [22, 34] is among the most celebrated reconstruction systems that work on RGB-D data. It uses Curless and Levoy's volumetric depth map fusion [10] to represent scene geometry as a continuously updated SDF, which aids smoothing noise away. Reg-

istration is done by rendering the SDF into a point cloud and applying point-to-plane ICP, making it susceptible to drift under erratic motion or lack of discriminative geometry.

Point-to-implicit approaches [4, 5] seek to overcome the limitations of ICP [39] by actively using the SDF. They directly project an incoming point cloud onto the volume and minimize the difference to its zero-level set, yielding more precise camera motion than KinectFusion. SDF-2-SDF [41] leverages this scheme to a denser, implicit-to-implicit formulation, whereby pairs of SDFs are directly aligned. It is used both for the frame-to-frame tracking and subsequent global optimization, leading to improved trajectory and reconstruction precision in the context of object scanning. We propose to transfer it to SLAM scenarios, in a fully online fashion inspired by PTAM [28]'s concurrent tracking and refinement. To this end, the camera is tracked in real-time on the GPU, while a fixed number of already tracked frames are jointly refined on the CPU. As there is no real-time constraint on the refinement, it runs for as much time as the tracking module permits, i.e. until the next batch is complete.

A major limitation of regular voxel grids is their high memory requirement. It has been tackled in multiple ways, including moving volumes [40, 48, 49], octrees [43, 44, 52], hybrid hierarchical structures [6] and voxel hashing [24, 36]. However, methods that rely on dense image alignment need robust techniques to disregard outliers [13, 26]. On the other hand, methods like RGB-D SLAM [12] that detect 2D features and match them in 3D discard a lot of useful information and require RANSAC [15] and pose graph optimization [29] to estimate consistent trajectories. While plenty of research efforts have gone in the direction of 3D keypoint detection [9, 16, 21, 23, 42, 46], the associated occlusions and noise currently limit their applications to object detection, recognition and classification [1, 3, 11].

We propose a quasi-dense technique which combines the efficiency of keypoint-based methods with the accuracy of dense schemes: we set small volumes around locations of distinct geometry and determine a common rigid-body motion for all of them. The volumes capture local geometry and thus grant flexibility with respect to their exact positions. The anchor points are chosen as the locations with highest mean curvature, which is the second-order derivative taken directly from the depth map [20], facilitating real-time performance.

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 [8, 14, 19, 53, 54]. On the contrary, our refinement is applicable online, as it also works over partial volumes, while the good initialization from tracking mitigates the increased demand of jointly optimizing several frames.

The sliding window bundle adjustment of Pirker *et al*. [38] is somewhat similar to our idea, but its use of sparse 2D-3D correspondences requires loop closure detection and posterior pose graph optimization. Whelan *et al*. [50] 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 [51] relies on local loop closures to activate non-rigid model-to-model refinement, without further improving the estimated trajectory. Therefore, we identify SDF-TAR as the first pose-graph- and loop-closure-free volumetric RGB-D SLAM method that carries out camera tracking and batch optimization in a fully online fashion.

## 3 SDF-TAR Pipeline

In the following we describe our partial volume scheme for reducing the memory requirements of regular voxel grid registration. Then we explain how the implicit-to-implicit energy

that we introduced in SDF-2-SDF [21] is applied over these small volumes, both for tracking and refinement, which we combine in parallel into our hybrid GPU/CPU SLAM system.

## 3.1 Background

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 $SE(3)$ [31]: $\xi = (\mathbf{u} \;\; \boldsymbol{\omega})^\top = (u_1, u_2, u_3, \omega_1, \omega_2, \omega_3)^\top$, where $\boldsymbol{\omega} \in \mathbb{R}^3$ denotes the rotational component and $\mathbf{u} \in \mathbb{R}^3$ corresponds to the translation. We denote applying this transformation to a 3D point $\mathbf{X} = (\mathbf{X}_X, \mathbf{X}_Y, \mathbf{X}_Z)^\top \in \mathbb{R}^3$ as $\mathbf{X}(\xi)$.

To estimate camera motion we register pairs of RGB-D frames, where the depth map is denoted by $D : \mathbb{N}^2 \to \mathbb{R}$. It consists of the projections $\pi(\mathbf{X}) = \mathbf{x}$ of 3D points onto the image plane, where $\mathbf{x} = (x, y)^\top \in \mathbb{N}^2$ is the pixel and $D(\mathbf{x}) = \mathbf{X}_Z$ is the value stored in the map. The inverse relation $\pi^{-1}$, back-projects a pixel $\mathbf{x}$ to 3D coordinates $\mathbf{X} = \pi^{-1}(\mathbf{x}, D(\mathbf{x}))$.

A signed distance field (SDF) in 3D space is an implicit function $\phi : \Omega \subseteq \mathbb{R}^3 \to \mathbb{R}$ that assigns to each point $\mathbf{X}$ its signed distance to the closest surface location [37]: 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 or ray tracing [10, 30].

A single depth image allows to generate a discrete projective SDF from its corresponding viewpoint. For this purpose, first the extents of the occupied volume are determined by back-projecting all pixels. Then it is discretized into cubic voxels of predefined side length $l$. Any 3D point within a given voxel is assigned the same properties as its center, so we will denote the whole voxel, and any point in it, by $\mathbf{V} \in \mathbb{R}^3$. These properties are:

$$\phi_{true}(\mathbf{V}) = D(\pi(\mathbf{V})) - \mathbf{V}_Z \,, \tag{1}$$

$$\phi(\mathbf{V}) = \begin{cases} sgn(\phi_{true}(\mathbf{V})) & \text{, if } |\phi_{true}(\mathbf{V})| \geq \delta \\ \phi_{true}(\mathbf{V})/\delta & \text{, otherwise} \end{cases} \tag{2}$$

$$\omega(\mathbf{V}) = \begin{cases} 1 & \text{, if } \phi_{true}(\mathbf{V}) > -\eta \\ 0 & \text{, otherwise.} \end{cases} \tag{3}$$

We store the value $\phi(\mathbf{V})$, obtained from the true signed distance $\phi_{true}$, scaled by a factor $\delta$ and truncated into the interval $[-1, 1]$. The binary weight $\omega(\mathbf{V})$ indicates whether the value for a voxel is reliable, i.e. when the voxel has already been observed, or if it is in the range of the expected object thickness $\eta$. Voxels with zero weight are discarded from computations.

For registration we use the SDF-2-SDF energy [21], which directly minimizes the per-voxel difference of two SDFs that occupy the same volume:

$$E_{SDF}(\xi) = \frac{1}{2} \sum_{voxels} \left( \phi_{ref}\omega_{ref} - \phi_{cur}(\xi)\omega_{cur}(\xi) \right)^2 , \tag{4}$$

where $\phi_{ref}$ is the reference SDF, generated from the identity pose, and $\phi_{cur}$ is the SDF whose optimal camera pose $\xi$ is currently being estimated. Voxel indices are omitted for ease of notation, i.e. we write $\phi_{ref}$ instead of $\phi_{ref}(\mathbf{V})$.

$E_{SDF}$ is based on the intuition that, as implicit functions, SDFs densely interpolate depth measurements throughout space. Thus both of the SDFs that are being registered steer convergence towards the optimally aligned state, demonstrated by higher accuracy than ICP

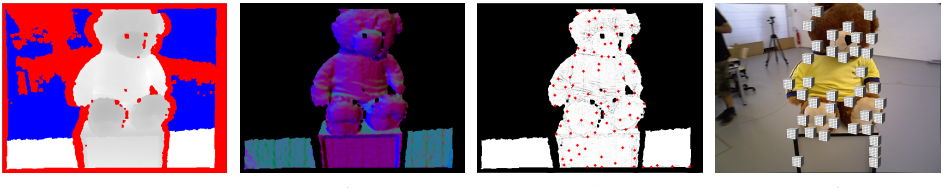
and point-to-implicit registration [41]. However, regular voxel grids are extremely memory-intensive when used to represent large scenes. This becomes especially problematic if a fine voxel resolution is used, as required for accurate reconstruction. Storing only the signed distances for $512^3$ voxels takes 0.5 GB, and for $1024^3$ - 4 GB. These figures further increase by 25% with the storage of the weight grid. The problem soon becomes intractable as processing a high amount of voxels also naturally entails increased runtime.

## 3.2 Selection of Limited-Extent Volume Locations

To circumvent this we propose an easy to implement solution that significantly reduces the memory load. Our key idea is to select $p$ limited-extent volumes (LEVs) $\Omega_1, ..., \Omega_p$ 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 small volumes simultaneously. In case of SDF-2-SDF alignment of volumes occupying the full frame, the high number of voxels either do not fit into device memory, or slow processing down due to the atomic reduction operations for accumulating each voxel's contribution. In our strategy the limited, fixed number of voxels permits fully exploiting the computational capabilities of the GPU. It guarantees that memory usage will be kept constant 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 positions of the LEVs is obviously critical, it is also natural. Guided by the intuition that flat areas, like walls, do not contribute and even inhibit registration, we propose to anchor the SDFs at points of high curvature. Such regions are highly distinct from their surroundings and therefore quickly lead registration to an optimal solution. We demonstrate the effectiveness of this choice in the experimental section.

Figure 2 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 [27], 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 preprocessed depth map, following the method of Holzer *et al.* [20]. Then we calculate the curvature magnitude from the derivatives of the normal map. Finally, we apply non-maximum suppression [33], so that only one high curvature point is selected within a window of size $w \times w$ pixels. This guarantees that the volumes of the SDFs centered around these locations will be non-overlapping. 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.



(a) (b) (c) (d)

Figure 2: LEV anchor point selection process. (a) Input depth map with overlayed validity mask: locations too far away (blue) and near edges (red) are discarded. (b) Normal map. (c) Curvature size is calculated and non-maximum suppressed to yield peaks that are sufficiently far apart. (d) Local volumes set on the $p$ anchor points with maximum curvature.

### 3.3 Limited-Extent Volume Registration

**Tracking** To register a depth frame $F_{m+1}$ to frame $F_m$, the anchor points for the small volumes $\Omega_i$ are selected on the reference frame $F_m$. Then the SDFs of $F_m$ are generated from the identity pose and $E_{SDF}$ is applied as a sum over all voxels in all LEVs. The pose $\xi$ of $F_{m+1}$ relative to $F_m$ is estimated iteratively by applying a first-order Taylor approximation around the current guess $\xi^k$, yielding a $6 \times 6$ linear system:

$$\mathbf{A} = \sum_{\substack{volume\ \Omega_i \\ i=1..p}} \left( \sum_{voxels \in \Omega_i} \nabla_\xi^\top \phi_{m+1}\left(\xi^k\right) \nabla_\xi \phi_{m+1}\left(\xi^k\right) \right), \tag{5}$$

$$\mathbf{b} = \sum_{\substack{volume\ \Omega_i \\ i=1..p}} \left( \sum_{voxels \in \Omega_i} \left( \phi_m - \phi_{m+1}\left(\xi^k\right) + \nabla_\xi \phi_{m+1}\left(\xi^k\right) \xi^k \right) \nabla_\xi^\top \phi_{m+1}\left(\xi^k\right) \right), \tag{6}$$

$$\frac{\mathrm{d}E_{SDF}}{\mathrm{d}\xi} = \mathbf{A}\xi - \mathbf{b}\,, \quad \xi^* = \mathbf{A}^{-1}\mathbf{b}\,, \quad \xi^{k+1} = \xi^k + \beta\left(\xi^* - \xi^k\right). \tag{7}$$

Here $\nabla_\xi \phi$ is the Jacobian of a voxel center point with respect to the pose $\xi$, and $\beta$ is the size of the step taken towards the current optimal solution estimate $\xi^*$. The binary weight factors have been omitted in order to simplify notation. They determine whether a voxel contributes to the sums in $\mathbf{A}$ and $\mathbf{b}$: it does only if both $\omega_{ref}$ and $\omega_{cur}$ for this voxel are set to valid.

**Pose Refinement** Optimization 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 weighted average $\phi_{avg}$ is generated in each partial volume over the $2b$ frames and serves as reference for alignment. It is calculated following Curless and Levoy's scheme [10], but this is done only every $f$ iterations 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 a gradient descent scheme, resulting in a 6-element vector pose update:

$$\frac{\mathrm{d}E_{SDF}}{\mathrm{d}\xi} = \sum_{\substack{volume\ \Omega_i \\ i=1..q}} \left( \sum_{voxels \in \Omega_i} \left(\phi_d(\xi) - \phi_{avg}\right) \nabla_\xi \phi_d(\xi) \right), \quad d \in [m-2b+1, \ldots, m]\,, \tag{8}$$

$$\xi_d^{k+1} = \xi_d^k - \alpha \frac{\mathrm{d}E_{SDF}(\xi_d^k)}{\mathrm{d}\xi}\,. \tag{9}$$

### 3.4 Parallel Tracking and Refinement

Our goal is a fully real-time SLAM method that does not entail any posterior processing. Therefore we execute the tracking and refinement modules concurrently, as outlined in Figure 1. 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. 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 [35], carefully selected keyframe fusion [32], or point-based fusion [25], which we favour due to its low memory load. As the particular method is not the focus of this paper, when comparing the outputs of different pipelines we always display results generated with the same technique, namely the incremental volumetric depth map fusion used in PCL's KinFu implementation[1].

# 4 Evaluation

In this section we investigate the dependence of the performance of our system on its parameters, compare it to related techniques and analyze its advantages.

## 4.1 Computational Performance

We carried out our experiments on a PC with an Intel i7-4900MQ CPU at 2.80 GHz and an NVIDIA Quadro K2100M GPU. Pre-processing the depth images takes 7-8 ms: transferring the depth image to device memory, estimating the normals and calculating the curvature size take approximately 4.5 ms in total, while the non-maximum 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. Typically 40-60 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 KB of GPU memory for $p = 64$ LEVs of $8^3$ voxels (if signed distances are stored as `float` and weights as `uchar`), totalling 2.66 MB for two frames together with their depth maps. In addition, the refinement module over $q = 8$ LEVs takes 20 KB of CPU memory for the weighted averages, and another 23.4 MB for 20 range images. These values demonstrate the real-time capabilities of SDF-TAR, combined with its low memory load. Note that the reported memory consumption only includes the depth maps and SDFs used for tracking and pose refinement, but not the reconstruction, as, depending on the goal of the application, any method of choice can be used for depth map fusion.

## 4.2 Parameter Study

The parameters in SDF-TAR reflect the inherent properties of the environment. While the majority of them are fixed, some depend on the richness of the scanned geometry.

The resolution of a single 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 refinement, using an even smaller one is not beneficial because it becomes 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 SDFs are used for tracking, only $q = 8$ 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 $f = 5^{\text{th}}$ iteration.

[1] KinectFusion Implementation in the Point Cloud Library (PCL), https://github.com/PointCloudLibrary/pcl/tree/master/gpu/kinfu.

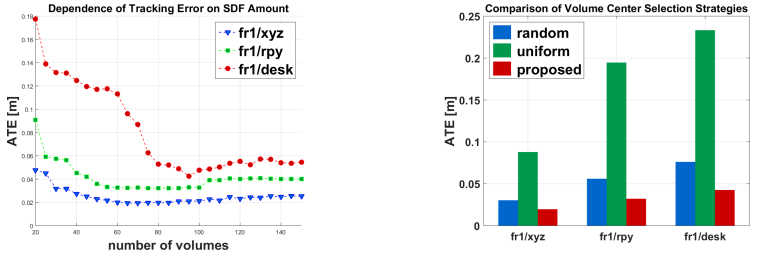



Figure 3: SDF-TAR parameter evaluation: influence of number of small SDFs (left) and their anchor point selection strategy (right) on the absolute trajectory error (without refinement).

We assess the remaining parameters of SDF-TAR on three sequences of the RGB-D benchmark [45]: *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, unless otherwise stated.

**Number of Volumes** We take from 20 to 150 LEVs per frame to judge the dependence of the tracking error on their amount. The results in Figure 3 (left) show that the error is large with a small number of volumes, and gradually decreases with more SDFs. There is quite a broad range of values which lead to near-optimal results, typically around 60-90 SDFs. When the amount of volumes 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 of volumes that guarantees stable results, *e.g.* 80 SDFs per frame.

**Anchor Point Selection Strategy** We compare our strategy for selecting the points around which the SDFs are centered (*cf.* Sec. 3.2) to two other approaches that can be applied directly on 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 projected in 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 preprocess the depth map, as explained, to discard invalid regions, and then take the same number of small volumes (the amount that gave optimal results in the experiment above for the respective sequence). Fig. 3 (right) shows that the uniform strategy leads to a 4-6 times higher error than our proposal, while the random sampling is nearly two times worse than ours. Thus our strategy clearly selects more discriminative regions that, combined with its high speed, are more advantageous for registration.

**Refinement Effect** Enabling the refinement module decreased the 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, the motion between flat regions can only be sliding against each other, which would inhibit the good rotational estimation.

| Method | fr1/xyz | fr1/rpy | fr1/desk | fr1/desk2 | fr1/360 | fr1/floor |
|---|---|---|---|---|---|---|
| KinFu | 0.023 | 0.081 | 0.057 | 0.102 | 0.591 | 0.918 |
| Bylow [4] | 0.021 | 0.042 | 0.035 | **0.061** | 0.119 | 0.567 |
| Canelhas [5] | **0.014** | - | 0.033 | 0.230 | - | 0.984 |
| SDF-TAR | 0.015 | **0.021** | **0.030** | 0.091 | **0.113** | **0.279** |

Table 1: Absolute trajectory error (ATE) comparison on RGB-D benchmark [45] sequences. Our method achieves a considerably smaller error when the dominant motion is rotational (*e.g. rpy*, *360*), while demonstrating comparable performance under translational motion.

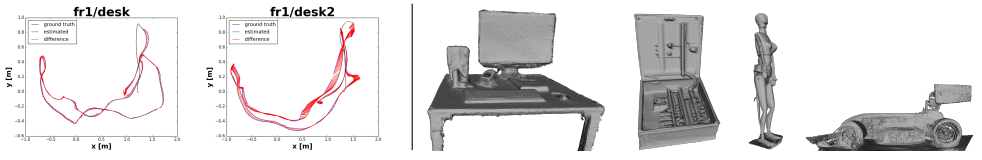



Figure 4: Qualitative results of SDF-TAR: (left) examples of estimated trajectories on the RGB-D benchmark [45]; (right) reconstructions of the objects from the CoRBS dataset [47].

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

## 4.3 Evaluation on Public Datasets

For quantitative evaluation we use the TUM RGB-D benchmark [45], which contains multiple Kinect v1 sequences with externally recorded trajectories, and the CoRBS dataset [47], which features various Kinect v2 scans of 4 large objects, together with their sub-millimeter precise CAD models from an external 3D scanner. We selected these real-world datasets over synthetic ones, such as ICL-NUIM [17], as they guarantee realistic handheld scanning scenarios and noise properties typical for RGB-D sensors.

We compare our approach to the most related systems that rely on SDFs for registration: PCL's KinFu and point-to-implicit methods [4, 5]. Since the CoRBS dataset was introduced after the point-to-implicit publications, we obtain the respective results with the implementation available in ROS[2] and specify it as pt-SDF-ROS.

The absolute trajectory errors in Table 1 testify that we considerably outperform related works on sequences with dominant rotational motion, and achieve on-par accuracy on other types of motion. We conclude that the LEVs reduce the negative influences of noise, blur and rolling shutter effect by constraining registration to the most discriminative local regions.

To further investigate rotation, in Table 2 we evaluate the error per frame. We test on the same sequences as [5], which is the only related work that reports RPE. As the translational errors reflect the rotational ones [45], and as expected by our lower ATE, both error components are typically lower for us. In particular, our rotational drift is well below 1° even on the

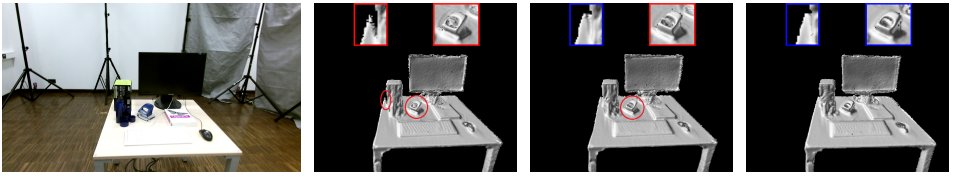
[2] sdf_tracker - ROS Wiki, http://wiki.ros.org/sdf_tracker.

| Method | fr1/xyz | | fr1/desk | | fr1/desk2 | | fr1/floor | |
|---|---|---|---|---|---|---|---|---|
| | tr. [m] | rot. [°] | tr. [m] | rot. [°] | tr. [m] | rot. [°] | tr. [m] | rot. [°] |
| Canelhas [5] | **0.003** | 0.472 | 0.007 | **0.759** | 0.019 | 1.080 | 0.050 | 2.085 |
| SDF-TAR | **0.003** | **0.442** | **0.006** | 0.768 | **0.009** | **0.993** | **0.020** | **0.844** |

Table 2: Relative pose error (RPE) translational and rotational root-mean squared values per frame on TUM RGB-D benchmark [45] sequences.

| Method | Desk | Cabinet | Human | Car |
|---|---|---|---|---|
| KinFu | 1.5686 | 1.2504 | 0.7105 | 2.9072 |
| pt-SDF-ROS | 1.3266 | 1.1599 | **0.6583** | 4.3870 |
| SDF-2-SDF | 1.1981 | 1.9836 | 1.7968 | 2.8947 |
| SDF-TAR | **0.9856** | **1.0552** | 0.7258 | **2.5470** |

Table 3: CloudCompare absolute cloud-to-model error [centimeters] on CoRBS objects [47].



color image KinFu pt-SDF-ROS SDF-TAR

Figure 5: Qualitative comparison (CoRBS/Desk): related approaches wash out fine structures due to tracking drift (marked in red regions), while the concurrent refinement of our SDF-TAR successfully minimizes drift, yielding more detailed, higher fidelity meshes.

challenging *fr1/floor*, indicating again that SDF-TAR is more powerful in handling rotation.

In the CoRBS dataset we used the first sequence for each object. We noticed that the currently provided groundtruth trajectories do not always yield geometrically consistent meshes, so, after personal communication with the authors, we decided to only evaluate the reconstruction error. Moreover, due to synchronization issues in the *Car* sequences the camera appears to jump back and forth between depth frames, causing all methods to perform poorly. By limiting the volume to the object of interest, we also test full regular grid SDF-2-SDF [41], although this is not its intended small-scale application. The CloudCompare[3] results in Table 3 prove that SDF-TAR has successfully leveraged SDF-2-SDF registration to SLAM scenarios. We achieve the smallest model error on most objects, which we attribute to the smaller rotational drift, combined with the benefit of online refinement.

## 5 Conclusions

We have presented a hybrid GPU/CPU system for concurrent tracking and batch refinement. SDF-TAR uses a volumetric registration scheme based on a novel memory reduction scheme, which aligns multiple voxel grids representing partial SDFs anchored at locations of distinctive geometry. These limited-extent volumes not only provide an easy to implement way for keeping memory load and runtime fixed, but also lead to considerably more accurate rotational motion estimation than related methods, as demonstrated on public datasets.

[3] CloudCompare - 3D Point Cloud and Mesh Processing Software, http://www.danielgm.net/cc/.

# References


[1] L. A. Alexandre. 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)*, 2012.

[2] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992.

[3] L. Bo, X. Ren, and D. Fox. Depth Kernel Descriptors for Object Recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[4] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions. In *Robotics: Science and Systems Conference (RSS)*, 2013.

[5] D. R. Canelhas, T. Stoyanov, and A. J. Lilienthal. SDF Tracker: A Parallel Algorithm for On-line Pose Estimation and Scene Reconstruction from Depth Images. In *2013 IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*, 2013.

[6] J. Chen, D. Bautembach, and S. Izadi. Scalable Real-time Volumetric Surface Reconstruction. *ACM Transactions on Graphics*, 32(4), 2013.

[7] Y. Chen and G. Medioni. Object Modeling by Registration of Multiple Range Images. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2724–2729, vol. 3, 1991.

[8] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust Reconstruction of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] U. Clarenz, M. Rumpf, and A. Telea. Robust Feature Detection and Local Classification for Surfaces Based on Moment Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):516–524, 2004.

[10] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, 1996.

[11] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model Globally, Match Locally: Efficient and Robust 3D Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[12] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An Evaluation of the RGB-D SLAM System. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[13] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision (ECCV)*, 2014.

[14] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi. Large-Scale and Drift-Free Surface Reconstruction Using Online Subvolume Registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[15] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[16] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust Global Registration. In *Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP)*, 2005.

[17] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[18] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *International Symposium on Experimental Robotics*, 2010.

[19] P. Henry, D. Fox, A. Bhowmik, and R. Mongia. Patch Volumes: Segmentation-Based Consistent Mapping with RGB-D Cameras. In *International Conference on 3D Vision (3DV)*, 2013.

[20] S. Holzer, J. Shotton, and P. Kohli. Learning to Efficiently Detect Repeatable Interest Points in Depth Data. In *12th European Conference on Computer Vision (ECCV)*, 2012.

[21] Y. Ioannou, B. Taati, R. Harrap, and M. A. Greenspan. Difference of Normals as a Multi-scale Operator in Unorganized Point Clouds. In *Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission (3DIMPVT)*, 2012.

[22] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology (UIST)*, 2011.

[23] A. E. Johnson and M. Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5):433–449, 1999.

[24] O. Kähler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. Torr, and D. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(11):1241–1250, 2015.

[25] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *2013 International Conference on 3D Vision (3DV)*, 2013.

[26] C. Kerl, J. Sturm, and D. Cremers. Robust Odometry Estimation for RGB-D Cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[27] K. Khoshelham and S. O. Elberink. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2):1437–1454, 2012.

[28] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[29] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, 2011.

[30] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, 1987.

[31] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

[32] M. Meilland and A. I. Comport. 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)*, 2013.

[33] A. Neubeck and L. Van Gool. Efficient Non-Maximum Suppression. In *18th International Conference on Pattern Recognition (ICPR)*, 2006.

[34] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *10th International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[35] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2320–2327, 2011.

[36] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. *ACM Transactions on Graphics (TOG)*, 2013.

[37] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153 of *Applied Mathematical Science*. Springer, 2003.

[38] K. Pirker, M. Rüther, G. Schweighofer, and H. Bischof. GPSlam: Marrying Sparse Geometric and Dense Probabilistic Visual Mapping. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2011.

[39] H. Pottmann, S. Leopoldseder, and M. Hofer. Registration Without ICP. *Computer Vision and Image Understanding (CVIU)*, 95(1):54–71, 2004.

[40] H. Roth and M. Vona. Moving Volume KinectFusion. In *British Machine Vision Conference (BMVC)*, 2012.

[41] M. Slavcheva, W. Kehl, N. Navab, and S. Ilic. SDF-2-SDF: Highly Accurate 3D Object Reconstruction. In *European Conference on Computer Vision (ECCV)*, 2016.

[42] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. 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)*, 2010.

[43] F. Steinbrücker, C. Kerl, J. Sturm, and D. Cremers. Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[44] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D Mapping in Real-time on a CPU. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[45] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

[46] F. Tombari, S. Salti, and L. Di Stefano. Performance Evaluation of 3D Keypoint Detectors. *International Journal of Computer Vision (IJCV)*, 102(1):198–220, 2013.

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

[48] T. Whelan, J. B. McDonald, M. Kaess, M. F. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[49] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. B. McDonald. Robust Real-Time Visual Odometry for Dense RGB-D Mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

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

[51] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems (RSS)*, 2015.

[52] M. Zeng, F. Zhao, J. Zheng, and X. Liu. Octree-based Fusion for Realtime 3D Reconstruction. *Graphical Models*, 75(3):126–136, 2013.

[53] Q.-Y. Zhou and V. Koltun. Dense Scene Reconstruction with Points of Interest. *ACM Transactions on Graphics*, 32(4), 2013.

[54] Q.-Y. Zhou and V. Miller, S. Koltun. Elastic Fragments for Dense Scene Reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.