International Conference on Intelligent Robots and Systems (IROS), Macau, China, November 2019.

Directional TSDF: Modeling Surface Orientation for Coherent Meshes

Malte Splietker and Sven Behnke

***Abstract*— Real-time 3D reconstruction from RGB-D sensor data plays an important role in many robotic applications, such as object modeling and mapping. The popular method of fusing depth information into a truncated signed distance function (TSDF) and applying the marching cubes algorithm for mesh extraction has severe issues with thin structures: not only does it lead to loss of accuracy, but it can generate completely wrong surfaces. To address this, we propose the directional TSDF—a novel representation that stores opposite surfaces separate from each other. The marching cubes algorithm is modified accordingly to retrieve a coherent mesh representation. We further increase the accuracy by using surface gradient-based ray casting for fusing new measurements. We show that our method outperforms state-of-the-art TSDF reconstruction algorithms in mesh accuracy.**

I. Introduction

3D models are a useful way to describe objects or whole environments, which can be used in a variety of robotic applications like scene understanding, manipulation, and navigation. Since the publication of KinectFusion [1] in 2011, TSDF fusion has turned into a de facto standard for fast registration and reconstruction using low-cost RGB-D sensors. TSDF fusion divides the modeled volume into a discretized grid of voxels and fuses distance information into it. Even though alternative approaches based on surfel predictions [2], [3] or direct meshing [4], [5] have emerged, TDSF fusion still remains the most popular choice.

Since the publication of KinectFusion, much work has been done to improve reconstruction speed and quality. There are, however, some fundamental limitations within the method itself. Firstly, it cannot represent anything thinner than the voxel size. While loss of fine details would be acceptable, the extracted surfaces can become completely wrong as illustrated in Fig. 1. Secondly, the state-of-the-art method of iteratively integrating new measurements is erroneous for steep observation angles and different observation directions as it overwrites and, in this way, destroys the representation.

A common way to mitigate these issues is to decrease the voxel size, which increases details and makes the problem less noticeable. However, for embedded hardware or large-scale mapping tasks a coarser voxel resolution might be required due to computation and memory restrictions. Moreover, the effect is not only affected by the voxel resolution, but also by the often depth-dependent truncation distance which is required to deal with measurement noise and usually spans multiple voxels. The problem lies in the representation

This research has been supported by MBZIRC 2017 price money.

All authors are with the Autonomous Intelligent Systems Group, University of Bonn, Germany. splietke@ais.uni-bonn.de

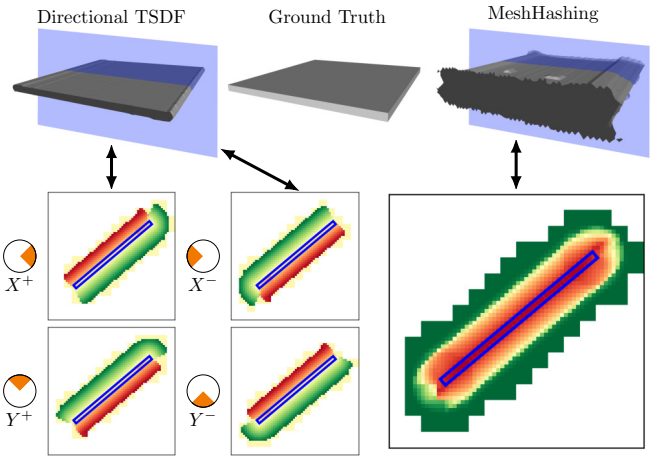
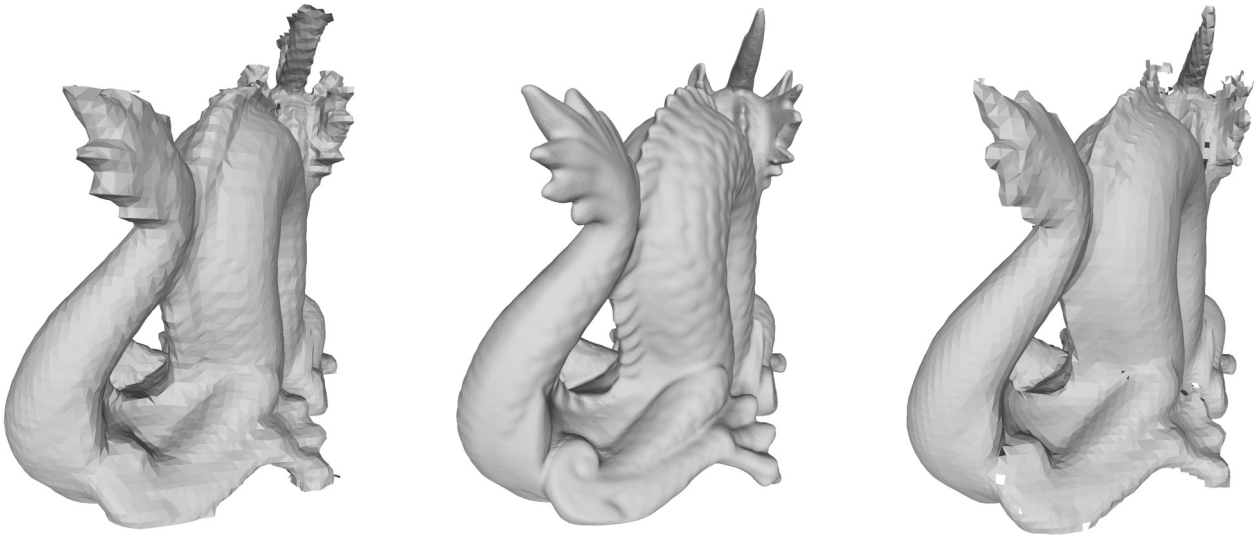


Fig. 1: Directional TSDF (proposed) solves the problems of reconstructing thin objects, which state-of-the-art TSDF fusion methods have issues with (here MeshHashing [6]). The bottom row shows the TSDFs (left: directional, right: undirected) in cross section, where the blue rectangle indicates the ground truth. The green and red colors denote areas that are in front or behind the surface, respectively. Color gradients indicate the signed distances to the object and the surface is extracted at the transition between the colors.

itself, as the TSDF implicitly encodes surfaces as zero crossings and the density of these transitions is bounded by the voxel resolution and the truncation distance. The direction from which surfaces have been observed is only encoded indirectly by the gradient normal. This is especially problematic during fusion, because information from different directions (at corners or on opposite sides of a wall) might be contradictory within the truncation range. This leaves the TSDF in an inconsistent state with false information. Furthermore, the state-of-the-art method for data integration, voxel projection, where each voxel is projected into the camera image and associated with the nearest pixel, has disadvantages. It causes serious aliasing, incorrectly handles steep surfaces and neglects large amounts of input data, especially for larger voxel sizes. To address these issues, we introduce the directional TSDF—a novel data structure that encodes the surface orientations by dividing the modeled volume into six directions according to the positive and negative coordinate axes. This representation can handle observations of thin structures from different, opposing directions, without introducing aliasing. Data integration is done in a ray-casting fashion along the surface normals for every input point. The advantages are that all data is utilized and that the resulting representation is more accurate with respect to steep angle



(a) MeshHashing [6] (b) Ground truth (c) Proposed

Fig. 2: Qualitative reconstruction comparison on the Dragon model from the Stanford 3D scanning repository [7]. Voxel size in both cases is $10\,\mathrm{mm}$.

observations. To extract surfaces from the directional TSDF, a modified marching cubes algorithm is proposed, which can also model opposite faces while remaining computationally inexpensive.

In summary, the contributions of this paper are a novel representation which is better suited for mapping scenes from different viewing directions. We also present an improved data integration scheme which considers the actual surface gradient for determining the correct voxels for fusion. Finally, a mesh extraction method for this new representation is proposed. We thoroughly evaluate our methods on standard data sets.

II. Related Work

Surface reconstruction from range data has been an active research topic for a long time. It gained in popularity through the availability of affordable depth cameras and parallel computing hardware. Zollhöfer et al. [8] give a comprehensive overview on modern 3D reconstruction from RGB-D data. The two main streams or research are TSDF fusion [1], [6] and surfel extraction [2], [3]. TSDF-based methods make up the majority, due to their simplicity and mesh output. Surfels, however, maintain the surface and observation direction in form of a normal per surfel. Thus they can distinguish observations from different sides. Another interesting approach is presented by Schöps et al. [9], who triangulate surfels to create a mesh representation.

An important step in data keeping was the switch from statically allocated voxel arrays to hash tables, allocating only required areas as proposed by Niesner et al. [10]. This enables scanning of large areas with limited memory and is considered state-of-the-art [6], [11], [12]. We base our work on Dong et al. [6], who further improve the data structure by tightly coupling voxel and meshing data. Signed distance data is stored on the corners of mesh cubes, which makes interpolation superfluous. Also the allocation, storage and access of vertex information is coupled to the structure, which decreases memory consumption and computation time.

As stated earlier, a major drawback of the TSDF representation is the voxel resolution, because the maximum object resolution is proportional to the voxel size. While decreasing the voxel size is one option, it is also wasteful in many areas with little detail. Steinbrücker et al. [13] address this by dynamically adjusting the voxel resolution at the cost of additional octree nesting depth and a very complex surface extraction algorithm. This does, however, not solve the problem completely as depth-dependent noise needs to be considered in choosing the truncation range. The undirected TSDF of Fig. 1 shows, how the truncation range from the opposite direction pushes the zero crossing away from the ground truth. Henry et al. [14] dynamically create new TSDF volumes whenever the angle of the surface changes too much. This is similar to our approach, but relies on managing a huge number of separate volumes. Also the volume separation decision relies on larger surfaces; therefore it does not deal with small details. Moreover, their approach lacks a mesh extraction method and renderings are created by ray casting.

The de facto standard method for integrating measurements, voxel projection [1], [6], [13], [14], suffers from aliasing effects especially for large voxel sizes and steep observation angles [11]. Curless et al. [15] perform voxel projection onto an intermediate mesh generated from the input, thereby using the interpolated values of multiple input points to update a voxel. Many approaches have tried to overcome the issues of voxel projection TSDF fusion. Commonly data integration is weighted according to the quality of measurements. Stotko and Golla [16] evaluated different weighting options for fusion. This helps to compensate distance- and angle-dependent noise. To reduce the

effects of integrating false information from surfaces with high observation angles, the point-to-plane distance metric can be applied [17]. As an alternative to voxel projection, ray casting [11] shoots a ray from the camera through every observed point and all voxels within the truncation range are updated. A sped up version using grouped ray casting was presented by Oleynikova et al. [12]. While for larger voxels the computational overhead is higher, the advantage is that there are no aliasing effects and that all information is utilized. Fossel et al. [18] address the issue that—especially for wide-angle sensors like LIDARs—the line-of-sight ray casting direction does not always comply with the surface direction. They estimate the surface gradient and choose the truncation range along the normal in a 2D SLAM system.

In contrast to the related works, we are proposing an improved representation based on the TSDF that utilizes the idea from Henry [14] to represent surfaces with different orientations separate from each other. The implementation is based on the work of Dong et al. [6], which also serves as a baseline for state-of-the-art methods using voxel projection and the marching cubes algorithm. Also we apply the gradient-based ray casting concept from Fossel et al. [18]. The key features of our method are:

- the directional TSDF representation that divides the modeled volume into six directions, thereby separately representing surfaces with different orientations,
- a gradient-based ray casting fusion for improved results,
- a thread-safe parallelization of ray casting fusion, and
- a modified marching cubes algorithm for mesh extraction from this representation.

III. Directional TSDF

A Signed Distance Function (SDF) denotes a function that for every 3D point yields the shortest distance to any surface. The sign denotes, whether the point is in front or behind the surface (inside an object). Let $\Omega \subset \mathbb{R}^3$ be a subset of space, e.g. a number of objects. In surface reconstruction, the points of interest lie on the boundary $\partial\Omega$. For a distance function d and any point $\mathbf{p} \in \mathbb{R}^3$, the SDF Φ defines the signed distance to the surface:

$$\Phi : \mathbb{R}^3 \longrightarrow \mathbb{R},\ \Phi(\mathbf{p}) = \begin{cases} -d(\mathbf{p}, \partial\Omega) & \text{if } \mathbf{p} \in \Omega, \\ d(\mathbf{p}, \partial\Omega) & \text{if } \mathbf{p} \in \Omega^c. \end{cases} \quad (1)$$

That is, points that lie inside of the object have a negative value and the surface lies exactly at the zero crossing between positive and negative values. Consequently, most regions of the SDF are superfluous for determining the surface. The Truncated Signed Distance Function (TSDF) cuts of all values above a *truncation threshold* τ, so everything outside the truncation range can be omitted. Typically TSDFs are estimated by a discretized grid of voxels and interpolation between the grid points. The truncation range is required to cope with aliasing effects and sensor noise and typically spans multiple voxels. While the method has proven to work in many scenarios, it has limitations, especially regarding thin objects, because the voxel resolution and truncation distance dictate the minimum thickness of objects. This effect occurs when observing small structures from opposite sides, as shown on the right hand side of Fig. 1 where the object "blows up" as the truncation range pushes the contour further out. The cross section of the TSDF shows how far the estimated contour (transition between red and green) is away from the ground truth (blue rectangle).

The problem obviously lies in the representation itself, since a zero transition cannot be represented by fewer than two voxels (one with for each positive and negative value). We propose a new representation, called Directional TSDF (DTSDF), which stores the signed distance information in different volumes according to the surface gradient

$$\Phi^d : \mathbb{R}^3 \longrightarrow \mathbb{R}^6,\ \Phi^d(p) = (\Phi_D(p))_{D \in \text{Directions}}. \quad (2)$$

The $\text{Directions} = \{X^+, X^-, Y^+, Y^-, Z^+, Z^-\}$ are defined by the positive and negative coordinate axes $\mathbf{v} = \{(1,0,0)^\intercal, (-1,0,0)^\intercal, \cdots\}$. This way, each voxel can encode up to six surfaces, which is beneficial for modeling orthogonal or opposite surfaces and arbitrarily thin objects. Each direction spans a sector of applicable surface normals to determine which information belongs where. While the number of sectors can be arbitrarily high, six is an obvious choice due to the cuboid voxel shape. Fewer sectors are problematic, as the covered angle increases, though in principle four sectors spanning a tetrahedron would be sufficient.

Given a measurement point's surface normal $\mathbf{n}$ and a direction vector $\mathbf{v}_D$, the direction-correspondence weight is defined as

$$w_D(\mathbf{n}) = \langle \mathbf{n}, \mathbf{v}_D \rangle . \quad (3)$$

A measurement is integrated into all directions, whose weight is above a threshold of $\sin(\pi/8)$. This allows for a smooth transition between the sectors, which is important for the creation of coherent meshes. Note, that each measurement point is integrated into at most three directions.

A. Data Structure

Given the directional correspondence computation, it is easy to spot that not all voxels need to store information for all directions. In order to save memory, the classical voxel hashing data structure [6], [10] is extended to hold varying numbers of voxel arrays, each of which represents a certain direction. Fig. 3 shows the connections: The block coordinates from the world are hashed. Then a conflict-resolving hash table maps to a dynamically allocated block. For every block the up to six voxel arrays are allocated as needed. For clearer visualization, the modeled volumes in all example images are depicted in 2D, but all arguments easily extend to 3D.

B. Gradient-Directed Ray Casting Fusion

Another novelty of our work is the surface gradient based ray casting fusion which, to our knowledge, has not been applied in 3D TSDF fusion before. The default method for fusing new measurements into the TSDF is voxel projection (VP), where voxels in view range and inside the camera frustum are projected onto the camera plane and are then

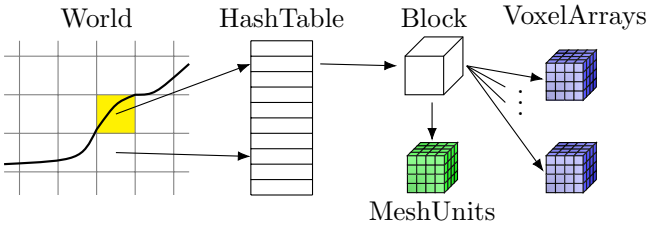
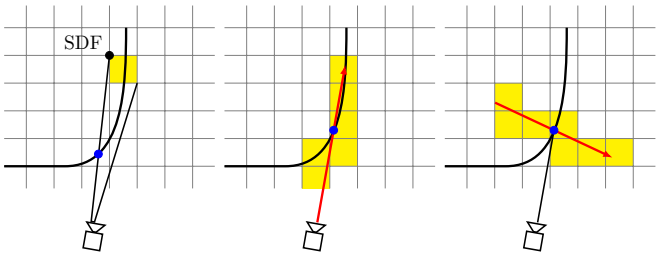


Fig. 3: Data structure for dynamically allocating blocks and per-direction voxel arrays.

associated with a single pixel in the depth image. Fig. 4a depicts an example of the drawbacks pointed out before, where the projected SDF point (black) is far away from the projected measurement (red dot), thus gets a high SDF value, even though it is very close to a surface. Handling misassociations like those can be partially mended by using the point-to-plane metric for updating the SDF value [17]. Fig. 4b shows ray casting fusion, where the algorithm casts a ray through every depth pixel and all intersected voxels within truncation range around the surface point are updated [11], [12]. While improving the association problem and better utilizing the available data, steep observation angles remain problematic [18]. Our experiments have shown that this method, especially using the standard TSDF, worsens corners when rays shoot through them from different directions. Instead we extend the idea from Fossel et al. [18] to our method and use the surface normals as shown in Fig. 4c. Starting from the projected surface point, a ray is cast along the normal (in both directions) and all intersecting voxels in truncation range are updated. To circumvent expensive voxel-ray intersection computations, the algorithm applies voxel traversal as proposed in [19].



(a) Voxel projection (b) Ray casting (c) RC with normal

Fig. 4: Fusion mode comparison. A measured surface point (blue dot) of the ground truth surface (black curve) is used to update voxels (yellow squares) along the fusion ray (red).

As the SDF corners of traversed voxels stray left and right from the ray, we furthermore apply the point-to-plane metric to increase the accuracy of the SDF. For a given surface point $\mathbf{p}$ and corresponding normal $\mathbf{n_p}$ the distance function is

$$d_{\text{p2pl}}(\mathbf{x}) = (\mathbf{p} - \mathbf{x})^\intercal \mathbf{n_p}. \tag{4}$$

The normals are computed using simple neighborhood estimation on the depth image. Due to noise and discretization in the depth image the estimated normals can be inaccurate, so a bilateral filter is applied. The depth image remains unfiltered to preserve reconstruction detail. While voxel projection updates every voxel exactly once, ray casting requires multiple updates per iteration. Details on the thread-safe fusion implementation are explained in Sec. V.

For the SDF update a combined weighting scheme is applied. The noise of RGB-D cameras depends on the measured distance, which is accounted for in w_{depth}. A high surface to view direction angle also increases inaccuracy, so it is down-weighted by w_{angle}. The factor w_D, defined above, works the same way as w_{angle}, but down-weights measurements that do not comply with the current fusion direction D. The combined weight is

$$w = w_{\text{depth}} \cdot w_{\text{angle}} \cdot w_{\text{D}}. \tag{5}$$

Stotko [16] gives a detailed overview on weighting factors.

IV. Directional Marching Cubes

A straight-forward and efficient method for mesh extraction from the TSDF representation is the marching cubes (MC) algorithm [20]. It can be easily parallelized. The world is again divided into a regular grid of *mesh units*, which in this implementation is identical to the voxel grid. For every mesh unit, the SDF values at the corners are checked and for all edges which contain a zero transition, an appropriate set of triangles is generated. Since there are only 256 configurations for positive and negative corners, called *MC index*, a lookup table is used for efficiency.

For the directional TSDF, finding those zero transitions becomes more complicated as it is not immediately clear how to combine the information collected for different directions. Also there can be opposite faces within the same mesh unit, which the original algorithm cannot handle. Every edge must now be able to hold up to one zero transition per direction. Therefore, the data structure from [6] is extended as depicted in Fig. 5. The data structure makes SDF interpolation superfluous as the relevant SDF values can be directly fetched. Every mesh unit handles three edges, each of which can

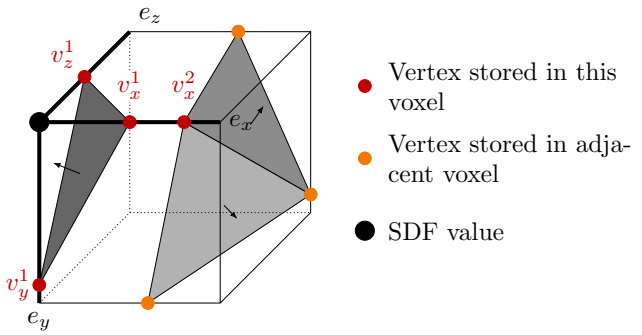


Fig. 5: Every mesh unit is responsible for storing vertices (red dots) on the thick edges e_x, e_y, e_z. If triangles have vertices on other edges, these are stored in adjacent voxels (yellow dots).

have up to two vertices for opposite surfaces. Triangles may have vertices on other edges, which are stored in adjacent mesh units. The outline of directional marching cubes is described in Algorithm 1. The steps are explained throughout this section.

Algorithm 1 MC Index Combining

```
for every mesh unit do
    for every direction D do
        get MC index mc_D, SDF weights w_D
    end for
    Directional MC index filtering
    Inter-directional filtering
    Compute surface offsets for each edge
    Determine combined MC indices (up to 2)
    Allocate Vertices and Edges
end for
```

A. *Filtering*

It is necessary to filter the MC indices, as due to the nature of the representation some degree of false information occurs. Especially at the edge of the TSDF or at spots where different geometry collides, deviation and overhangs appear. Filtering is done in two stages, intra-directional and inter-directional.

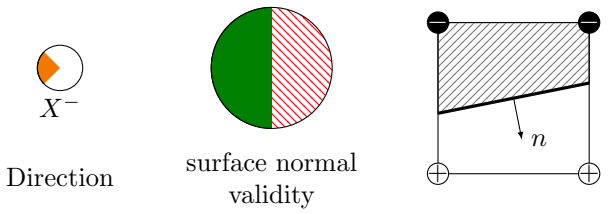


Fig. 6: Filtering an implausible MC index by surface normal. The normal $\mathbf{n}$ exceeds the validity range (green semicircle) for direction X^-. The white and black corners of the mesh unit indicate whether the value is in front of or behind the surface, respectively.

Firstly, all surfaces that would contradict the respective directions are discarded. This is performed using a pre-computed lookup table and comparison in cases where the orientation of the surface decides. Fig. 6 shows an example, where the normal of the potential surface is outside the valid range for direction X^-, so it is discarded. In this case the simple table lookup is insufficient, because there are configurations with the same MC index, where the normal is inside the valid range. The SDF gradient is utilized to identify these outliers.

In the second step, the different directions are weighted against each other in order to decide, whether a hypothesized surface is correct. The blue mesh unit in Fig. 7 shows a typical example, how the overhanging edge of direction Y^+ is identified as a false positive by X^+. To reduce false cancellations, the decision is made using a voting scheme. The credibility of a direction's information is accounted for by the SDF weight and surface gradient w.r.t. this direction. For SDF weight w_D^{sdf}, voxel center gradient $\nabla\Phi_D$, direction vector $\mathbf{v}_D$ and vote a_D,

$$a = \sum_D w_D^{\text{sdf}} \langle \nabla\Phi_D, \mathbf{v}_D \rangle a_D, \quad a_D \in \{-1, 1\} \tag{6}$$

yields the consensus. If a is smaller than 0, the MC index is set to 0 and no surface is extracted.

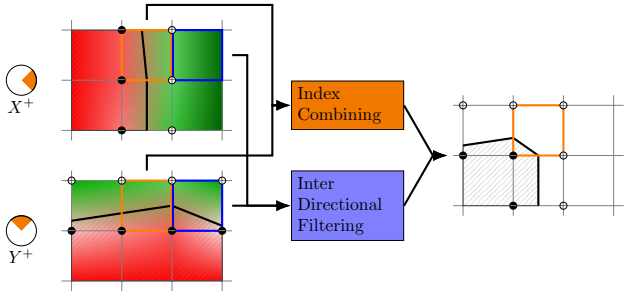


Fig. 7: MC index combining (orange) and inter-directional filtering (blue) is applied to TSDFs of directions X^+ and Y^- to retrieve the combined surface on the right side.

B. *Surface Offset Estimation*

From the remaining directions, the combined vertex positions are computed similar to what is described in [6]. For every edge the MC indices of all directions are checked for potential zero transitions. The offset is added to a weighted average with the same combined weight used in Eq. (6). Since there can be opposite surfaces sharing the same edge, two offsets are stored per edge and the data structure is updated accordingly (c.f. Fig. 5).

C. *Combined MC Index*

After filtering false positives there might still be a mesh unit with multiple surface hypotheses from different directions. There are multiple ways for combining these, but intersection has shown good results. The orange encircled voxels in Fig. 7 are combined, such that a connection between the surfaces of the two directions is established. This combination is computationally cheap, as it can be done entirely on level of MC indices. Algorithm 2 shows how the index-wise intersection is performed. The MC index is split into it's up to four unconnected components. For each of these components the compatibility to the already combined indices is checked. The intersection of the indices is equivalent to the binary *and* operation.

Before extracting the final mesh, regularization between MC indices of neighboring voxels is performed for all modified blocks. This helps to reduce slits and overhangs induced by previous steps and close the surface. Fig. 8 shows a mesh before and after the regularization. The procedure works solely on the MC indices by minimizing irregularities across voxel borders.

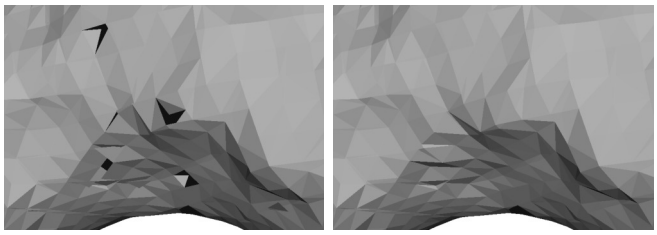
V. Thread-Safe Ray Casting Fusion

The integration of new measurements into the TSDF is done by a weighted cumulative moving average. Let D_t and W_t be a voxel's signed distance and weight values as time t. d_t and w_t are signed distance and weight update factors

Algorithm 2 MC Index Combining

Input: MCIndex[6]
Output: combined[2]

```
combined = (0, 0)
for D = 0 to 5 do
    for component in MCIndex[D] do
        if compatible(component, combined[0]) then
            combined[0] &= component
            break
        else if compatible(component, combined[1]) then
            combined[1] &= component
        end if
    end for
end for
return combined
```



(a) before (b) after

Fig. 8: Voxel neighborhood MC index regularization.

which are to be integrated. Then the update is

$$D_t = \frac{W_{t-1}D_{t-1} + w_t d_t}{W_{t-1} + w_t} = \frac{\sum_{i=1}^{t} w_i d_i}{\sum_{i=1}^{t} w_i}, \tag{7}$$

$$W_t = W_{t-1} + w_t. \tag{8}$$

In contrast to voxel-projection fusion, ray casting leads to multiple SDF updates per voxel within the same iteration. While being mathematically sound, it is problematic for a parallel implementation, as Eq. (7) and Eq. (8) cannot be performed atomically by most hardware. Eq. (7) implies, that instead of applying an incremental update step for every pixel affecting a voxel, the following equivalent and thread-safe operation can be performed. Let S_d, S_w be per-voxel summation values for signed distance and weight, respectively. They are initialized with zero at the beginning of each update iteration.

$$S_d \overset{atomic}{+=} w_i d_i, \qquad S_w \overset{atomic}{+=} w_i \tag{9}$$

In the second step all modified voxels are iterated and the final update is computed as follows:

$$D_t = \frac{W_{t-1}D_{t-1} + S_d}{W_{t-1} + S_w}, \qquad W_t = W_{t-1} + S_w. \tag{10}$$

VI. Evaluation

MeshHashing [6] serves as a baseline for recent TSDF RGB-D reconstruction algorithms with voxel hashing and marching cubes. Here, it is sometimes denoted as state-of-the-art (SOTA). Our proposed method is referred to as DTSDF. Registration is currently not implemented, so we compare the reconstruction quality and computation time.

For comparability, the well-known datasets by Zhou et al. [21] and the Stanford Computer Graphics Laboratory [7] are used. The Zhou dataset already provides trajectory and scans. For the Stanford dataset the 3D models are scaled, such that the longest side equals one meter. The camera performs an even circular motion with a two meter radius around the object, acquiring 1000 depth images with ground truth poses. The renderer uses the standard Kinect model $(f, c_x, c_y) = (525, 319.5, 239.5)$ with resolution 640×480 pixels. All experiments were run on a notebook with an Intel i7-4710HQ CPU (2.50GHz) and a GTX960M.

The truncation distance is fixed to four times the voxel size, as suggested by Oleynikova et al. [12]. Smaller factors tend to create holes in some spots, especially for voxel projection. No other parameters were changed.

Fig. 2 shows a qualitative comparison between SOTA, DTSDF and the ground truth at $10\,\mathrm{mm}$ voxel size. While at the given voxel resolution the amount of detail is limited, our method maintains a better surface (tail) and preserves geometry thinner than the voxel size (ridge, tail tip). The SOTA tends to enlarge the overall object. To quantify these findings we conducted a number of experiments at different voxel resolutions and measured the RMSE against the ground truth model. Tab. I and Tab. II show the findings for different resolutions. Most notably, our method outperforms the SOTA in almost all cases by a good margin. Note, that we chose larger voxel sizes for the Zhou dataset due to the larger overall size of the scenes.

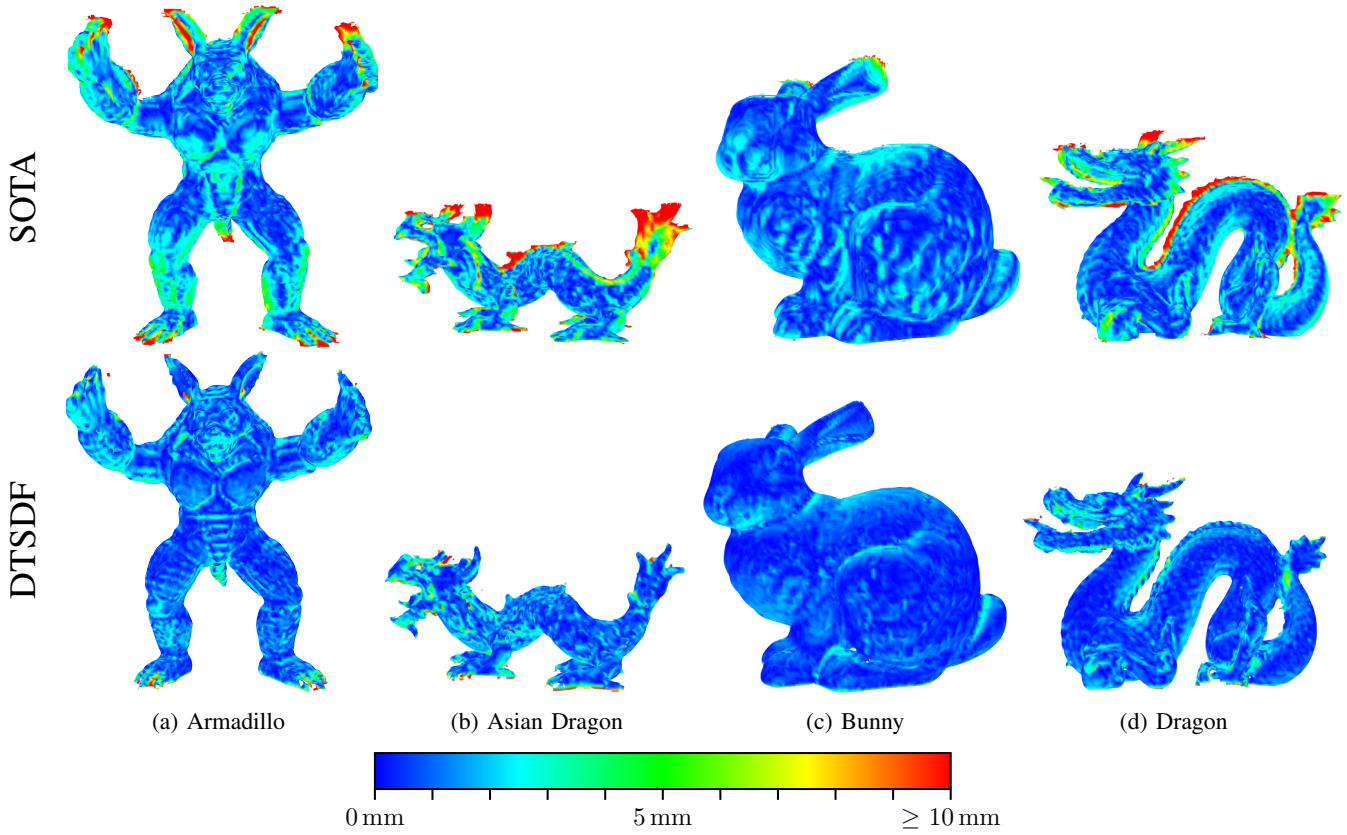
TABLE I: RMSE (in mm) of state-of-the-art and proposed method under different voxel sizes on the Stanford dataset.

		Voxel Size [mm]					
dataset	mode	5	10	20	30	40	50
Armadillo	SOTA	1.55	3.67	11.83	25.27	41.02	58.36
	DTSDF	1.14	1.74	3.58	6.53	14.56	21.39
Asian Dragon	SOTA	2.34	7.11	19.94	38.70	55.17	67.86
	DTSDF	1.56	2.87	6.81	11.90	19.00	30.74
Bunny	SOTA	1.90	3.82	9.01	17.70	27.97	37.98
	DTSDF	0.98	1.23	2.18	4.17	8.59	18.37
Dragon	SOTA	1.73	4.44	12.09	22.40	35.93	52.67
	DTSDF	1.23	2.15	5.35	8.99	11.70	18.70

TABLE II: RMSE (in mm) of state-of-the-art and proposed method under different voxel sizes on the Zhou dataset.

		Voxel Size [mm]			
dataset	mode	25	50	75	100
Burghers	SOTA	12.80	27.28	57.69	102.46
	DTSDF	6.86	13.38	19.60	26.53
Cactus-garden	SOTA	12.78	32.40	59.35	85.96
	DTSDF	13.19	16.69	33.00	41.68
Lounge	SOTA	14.96	33.91	57.85	85.52
	DTSDF	12.41	25.11	41.95	51.86
Copyroom	SOTA	23.17	26.25	30.66	39.82
	DTSDF	14.08	27.89	36.62	45.90
Stonewall	SOTA	7.23	13.84	26.22	39.71
	DTSDF	6.89	11.47	18.85	31.20
Totempole	SOTA	10.41	15.19	27.65	43.00
	DTSDF	7.86	9.80	14.88	24.93

Fig. 9 shows a distance error heatmap visualization which matches this observation. The models reconstructed by the SOTA have hotspots, wherever there is thin geometry (ears,



(a) Armadillo (b) Asian Dragon (c) Bunny (d) Dragon

Fig. 9: Distance error heatmap comparison between state-of-the-art (top) and proposed (bottom) on the Stanford dataset. Voxel size is 10 mm.

tail, and ridge). But other areas benefit from the DTSDF, as well.

To examine the impact of the surface gradient ray casting fusion we conducted another experiment. Tab. III shows the RMSE at 10 mm voxel resolution for different algorithm modes, which are encoded as follows: *Def* and *Dir* (TSDF or DTSDF), voxel projection (VP), point-to-plane (P2PL), ray casting (RC) and ray casting along normals (RCN). Using ray casting on the default TSDF in most cases actually worsens the results, except for the bunny. This matches our earlier observation, because standard ray casting causes overshooting at corners while ray casting along normals inflates the thin parts to the maximum extent of the truncation distance. The DTSDF outperforms the SOTA even using voxel projection, but adding ray casting along normals further widens this margin. Standard ray casting, however, leads to the same problems. Only the bunny model benefits from it, because of its round shape and lack of thin, sharp edges.

TABLE III: RMSE (in mm) for different fusion modes, voxel size = 10 mm.

model \ mode	Def +VP	Def +RC +P2PL	Def +RCN +P2PL	Dir +VP	Dir +RC +P2PL	Dir +RCN +P2PL
Armadillo	3.670	4.915	4.839	1.931	2.478	**1.741**
Asian Dragon	7.106	10.148	9.211	3.016	3.545	**2.865**
Bunny	3.816	2.792	2.958	1.625	1.674	**1.229**
Dragon	4.438	6.570	6.169	2.534	2.930	**2.146**

The runtime of the algorithm can be split into two main components: data integration and meshing. Tab. IV shows the total update time and proportion used for meshing on the Asian Dragon dataset. As expected, the amount of time spent on meshing increases with smaller voxel sizes. The data integration, however, remains fast even for higher resolutions: Fig. 10 breaks down the integration step for SOTA and our proposed method. Preprocessing and allocating blocks and voxel arrays make up a significant constant factor. Fusion and recycling times are affected by the number of blocks in the view frustum. The ray casting has an additional (constant) proportion determined by the number of depth pixels, but also increases with the number of voxels due to the update accumulation step (c.f. Sec. V).

TABLE IV: Mean total update time (in ms) and percentage that is used for meshing for dataset Asian Dragon.

		Voxel Size [mm]					
	mode	5	10	20	30	40	50
total time	SOTA	112.4	30.9	10.7	7.5	6.4	5.2
	Proposed	199.0	74.0	31.7	19.9	15.1	13.0
total time	SOTA	96.8	90.3	73.5	61.5	48.0	45.5
	Proposed	96.8	92.8	84.5	75.9	68.8	63.5

Dong et al. [6] provide an accurate memory analysis for

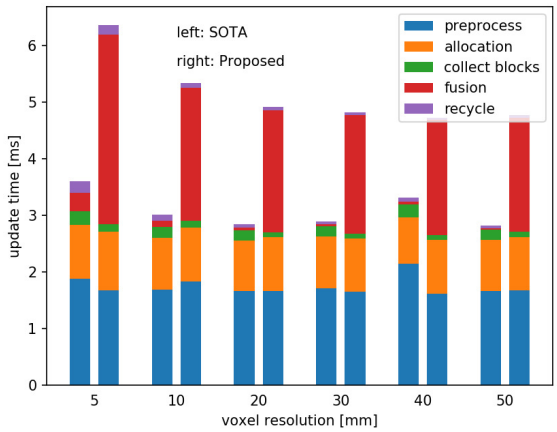


Fig. 10: Mean data integration time for dataset Asian Dragon and different voxel resolutions. The left bars show the SOTA, the right side corresponds to DTSDF.

their MeshHashing implementation. The memory requirements for our algorithm differ from the original algorithm only by the number of voxel arrays allocated per block. Fig. 11 shows the mean number of voxel arrays per block for different voxel resolutions and models. With smaller voxels the number of allocations decreases, as the number of integrations from opposite directions decreases within individual blocks. For the same reason the bunny, which is round and thicker in volume, requires fewer voxel arrays.

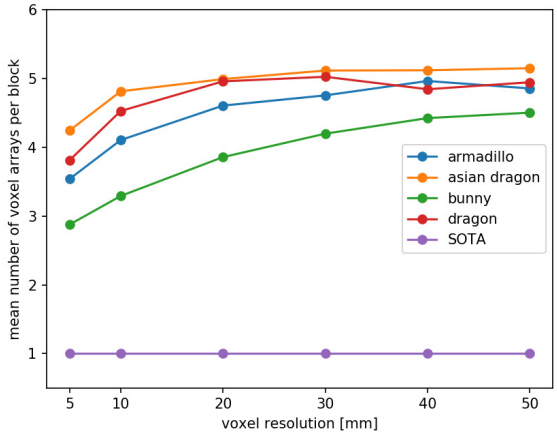


Fig. 11: Mean number of voxel arrays per block for different datasets and voxel resolutions.

VII. Conclusions

The proposed Directional TSDF representation and the matching modified marching cubes algorithm overcome limitations of the state-of-the-art method at the price of memory consumption and computation time. Less resolution is required for dealing with corners and thin objects which makes our method interesting for large-scale applications. Its robustness against different observation angles makes DTSDF also attractive for frame-to-model registration.

References

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

[2] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3D modeling and tracking," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 137–147, 2014.

[3] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[4] W. N. Greene and N. Roy, "FLaME: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4696–4704.

[5] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time cpu-based large-scale 3D mesh reconstruction," 2018.

[6] W. Dong, J. Shi, W. Tang, X. Wang, and H. Zha, "An efficient volumetric mesh representation for real-time scene reconstruction using spatial hashing," 2018.

[7] S. C. G. Laboratory, "The Stanford 3D scanning repository," http://graphics.stanford.edu/data/3Dscanrep/, accessed Feb 25, 2019.

[8] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3D reconstruction with RGB-D cameras," *Computer Graphics Forum (Eurographics State of the Art Reports)*, vol. 37, pp. 625–652, 2018.

[9] T. Schöps, T. Sattler, and M. Pollefeys, "SurfelMeshing: Online surfel-based mesh reconstruction," 2018.

[10] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 6, p. 169, 2013.

[11] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3D reconstruction onboard a mobile device using spatially hashed signed distance fields." in *Proceedings of Robotics: Science and Systems (RSS)*, vol. 4, 2015.

[12] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1366–1373.

[13] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a cpu," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2021–2028.

[14] P. Henry, D. Fox, A. Bhowmik, and R. Mongia, "Patch volumes: Multiple fusion volumes for consistent rgb-d modeling," in *RSS workshop on RGB-D: Advanced reasoning with depth cameras, Berlin, Germany*, 2013.

[15] 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*. ACM, 1996, pp. 303–312.

[16] P. Stotko and T. Golla, "Improved 3D reconstruction using combined weighting strategies," in *Proceedings of the Central European Seminar on Computer Graphics (CESCG)*, 2015, pp. 135–142.

[17] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, "Real-time camera tracking and 3D reconstruction using signed distance functions." in *Proceedings of Robotics: Science and Systems (RSS)*, vol. 2, 2013.

[18] J.-D. Fossel, K. Tuyls, and J. Sturm, "2D-SDF-SLAM: A signed distance function based SLAM frontend for laser scanners," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1949–1955.

[19] J. Amanatides, A. Woo *et al.*, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.

[20] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, no. 4, 1987, pp. 163–169.

[21] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 112, 2013.