PSDF Fusion: Probabilistic Signed Distance Function for On-the-fly 3D Data Fusion and Scene Reconstruction

Wei Dong[1,2](✉), Qiuyuan Wang[1,2], Xin Wang[1,2], and Hongbin Zha[1,2](✉)

[1] Key Laboratory of Machine Perception (MOE), School of EECS, Peking University, Beijing, China
{w.dong,qiuyuanwang,xinwang_cis}@pku.edu.cn, zha@cis.pku.edu.cn
[2] Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China

Abstract. We propose a novel 3D spatial representation for data fusion and scene reconstruction. *Probabilistic Signed Distance Function* (Probabilistic SDF, PSDF) is proposed to depict uncertainties in the 3D space. It is modeled by a joint distribution describing SDF value and its inlier probability, reflecting input data quality and surface geometry. A *hybrid data structure* involving voxel, surfel, and mesh is designed to fully exploit the advantages of various prevalent 3D representations. Connected by PSDF, these components reasonably cooperate in a consistent framework. Given sequential depth measurements, PSDF can be incrementally refined with less ad hoc parametric Bayesian updating. Supported by PSDF and the efficient 3D data representation, high-quality surfaces can be extracted on-the-fly, and in return contribute to reliable data fusion using the geometry information. Experiments demonstrate that our system reconstructs scenes with higher model quality and lower redundancy, and runs faster than existing online mesh generation systems.

Keywords: Signed Distance Function · Bayesian updating

1 Introduction

In recent years, we have witnessed the appearance of consumer-level depth sensors and the increasing demand of real-time 3D geometry information in next-generation applications. Therefore, online dense scene reconstruction has been a popular research topic. The essence of the problem is to fuse noisy depth data stream into a reliable 3D representation where clean models can be extracted. It is necessary to consider uncertainty in terms of sampling density, measurement accuracy, and surface complexity so as to better understand the 3D space.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01240-3_43) contains supplementary material, which is available to authorized users.

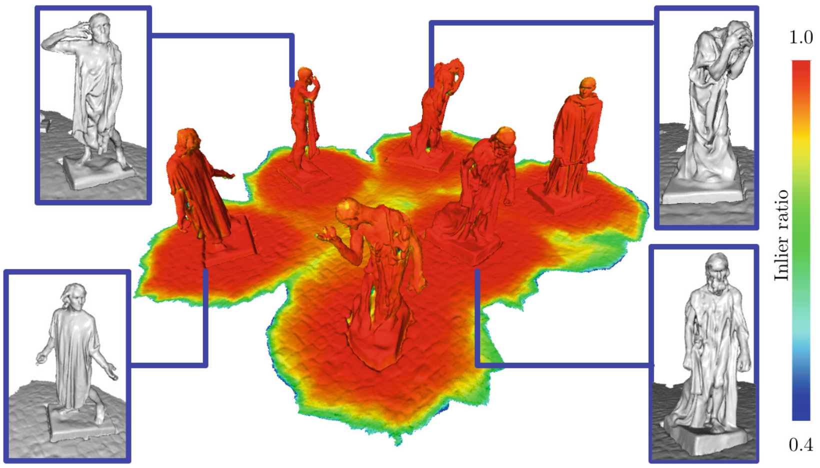


Fig. 1. Reconstructed mesh of *burghers*. The heatmap denotes the SDF inlier ratio (SDF confidence). Note details are preserved and outliers have been all removed without any post-processing. Inlier-ratio heatmaps in Figs. 4, 5 and 6 also conform to this colorbar. (Color figure online)

Many representations built upon appropriate mathematical models are designed for robust data fusion in such a context. To handle uncertainties, *surfel* and *point* based approaches [13,15,29] adopt filtering-based probabilistic models that explicitly manipulate input data. *Volume* based methods [5,12,20,27], on the other hand, maximize spatial probabilistic distributions and output discretized 3D properties such as SDF and occupancy state. With fixed topologies, *mesh* based methods [35] may also involve parametric minimization of error functions.

While such representations have been proven effective by various applications, their underlying data structures endure more or less drawbacks. Surfels and points are often loosely managed without topology connections, requiring additional modules for efficient indexing and rendering [1], and is relatively prone to noisy input. Volumetric grids lack flexibility to some extent, hence corresponding data fusion can be either oversimplified using weighted average [20], or much time-consuming in order to maximize joint distributions [27]. In addition, ray-casting based volume rendering is also non-trivial. Usually storing vertices with strong topological constraints, mesh is similarly hard to manipulate and is less applicable to many situations. There have been studies incorporating aforementioned data structures [14,18,23,24], yet most of these pipelines are loosely organized without fully taking the advantages of each representation.

In this paper, we design a novel framework to fully exploit the power of existing 3D representations, supported by PSDF-based probabilistic computations. Our framework is able to perform reliable depth data fusion and reconstruct high-quality surfaces in real-time with more details and less noise, as depicted in Fig. 1. Our contributions can be concluded as:

1. We present a novel hybrid data structure integrating voxel, surfel, and mesh;
2. The involved 3D representations are systematically incorporated in the consistent probabilistic framework linked by the proposed PSDF;
3. Incremental 3D data fusion is built upon less ad-hoc probabilistic computations in a parametric Bayesian updating fashion, contributes to online surface reconstruction, and benefits from iteratively recovered geometry in return.

2 Related Work

Dense Reconstruction from Depth Images. There is a plethora of successful off-the-shelf systems that reconstruct 3D scenes from depth scanners' data. [4] presents the volumetric 3D grids to fuse multi-view range images. [20] extends [4] and proposes KinectFusion, a real-time tracking and dense mapping system fed by depth stream from a consumer-level Kinect. KinectFusion has been improved by VoxelHashing [22], InfiniTAM [12], and BundleFusion [5] that leverage more memory-efficient volumetric representations. While these systems perform online depth fusion, they usually require offline MarchingCubes [17] to output final mesh models; [6,14,24] incorporates online meshing modules in such systems. Instead of utilizing volumetric spatial representations, [13] proposes point-based fusion that maintains light-weight dense point cloud or surfels as 3D maps. ElasticFusion [31] and InfiniTAM_v3 [23] are efficient implementations of [13] with several extensions; [16] further improves [13] by introducing surface curvatures. The point-based methods are unable to output mesh online, hence may not be suitable for physics-based applications. [2,34] split scenes into fragments and register partially reconstructed mesh to build comprehensive models, but their offline property limits their usages.

3D Scene Representations. Dense 3D map requires efficient data structures to support high resolution reconstructions. For volumetric systems, plain 3D arrays [20] are unsuitable for large scale scenes due to spatial redundancies. In view of this, moving volumes method [30] is introduced to maintain spatial properties only in active areas. Octree is used to ensure a complete yet adaptive coverage of model points [10,24,33]. As the tree might be unbalanced causing long traversing time, hierarchical spatial hashing is utilized [12,22] supporting O(1) 3D indexing, and is further extended to be adaptive to local surface complexities [11].

There are also studies that directly represent scenes as point clouds or mesh during reconstruction. In [13,31] point clouds or surfels are simply arranged in an 1D array. Considering topologies in mesh, [18] manages point clouds with inefficient KD-Tress for spatial resampling. [35] maintains a 2.5D map with fixed structured triangles which will fail to capture occlusions. Hybrid data structures are also used to combine volumes and mesh. [24] builds an octree-based structure where boundary conditions have to be carefully considered in term of mesh triangles. [14] uses spatial hashed blocks and stores mesh triangles in the block level, but ignores vertex sharing between triangles. [6] reveals the correspondences between mesh vertices and voxel edges, reducing the redundancy in the aspect of data structure. Yet improvement is required to remove false surfaces generated from noisy input data.

Uncertainty-Aware Data Fusion. Uncertainty is one of the core problems remain in 3D reconstruction, which may come from imperfect inputs or complex environments. In volumetric representations that split the space into grids, probability distributions are usually utilized to model spatial properties. Binary occupancy is an intuitive variable configuration, denoting whether a voxel is physically occupied. [32] proposes a joint distribution of point occupancy state and inlier ratio over the entire volume with visual constraints and achieves competitive results. [27,28] similarly emphasize ray-consistency to reconstruct global-consistent surfaces, whose inferences are too sophisticated to run in real-time. Although surface extraction can be performed on occupancy grids via thresholding or ray-casting, it is usually very sensitive to parameters. Instead of maintaining a $\{0, 1\}$ field, [4] introduces SDF which holds signed projective distances from voxels to their closest surfaces sampled by input depth measurements. [19,20] use weight average of Truncated SDF in the data fusion stage by considering a per-voxel Gaussian distribution regarding SDF as a random variable. While Gaussian noise can be smoothed by weight average, outliers have to be carefully filtered out with ad hoc operations. [22] uses a temporal recycling strategy by periodically subtracting weight in volumes; [14] directly carves out noisy inputs; [5] proposes a weighted subtraction to de-integrate data which are assumed to be incorrectly registered. As a non-local prior, [7] refines SDF value on-the-go using plane fitting, which performs well mainly in flat scenes with relatively low voxel resolutions. We find a lack of systematic probabilistic solution for SDF dealing both Gaussian noise and possible outliers.

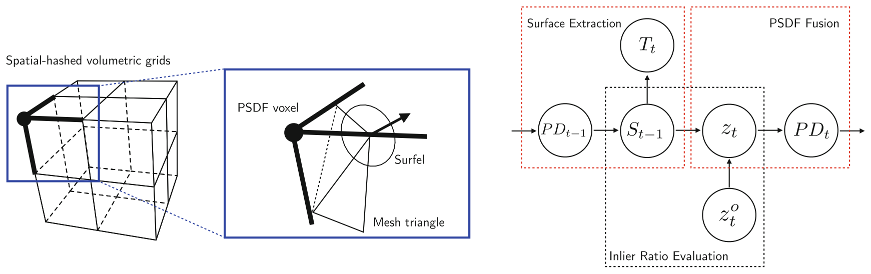
For point-based representations, [29] proposes an elegant math model treating inlier ratio and depth of a point as random variables subject to a special distribution. The parameters of such distributions are updated in a Bayesian fashion. This approach is adopted by [8] in SLAM systems for inverse depths, achieving competitive results. An alternative is to select ad hoc weights involving geometry and photometric properties of estimated points and computing weighted average [15,26]. This simple strategy shares some similarity to the fusion of SDF values, and is also used in RGB-D systems where depths are more reliable [13,23]. Despite the solid math formulations, point-based methods are comparatively prone to noise due to their discrete representations.

3 Overview

Our framework is based on the *hybrid data structure* involving three *3D representations* linked by *PSDF*. The pipeline consists of iterative operations of data fusion and surface generation.

3.1 Hybrid Data Structure

We follow [6,14,22] and use a spatial hashing based structure to efficiently manage the space. A hash entry would point to a block, which is the smallest unit to



(a) Data structure (b) Pipeline

Fig. 2. A basic hybrid unit, and the system pipeline.

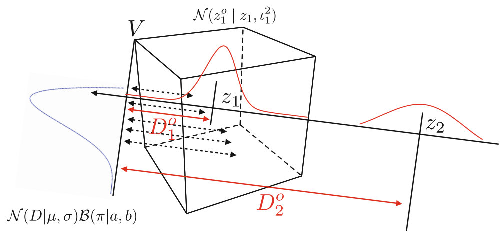
allocate and free. A block is further divided into $8 \times 8 \times 8$ small voxels. Following [6] we consider a voxel as a 3-edge structure instead of merely a cube, as depicted in Fig. 2(a), which will avoid ambiguity when we refer to shared edges. PSDF values are stored at the corners of these structures. In addition, we maintain surfels on the volumetric grids by limiting their degree of freedom on the edges of voxels; within a voxel at most 3 surfels on edges could be allocated. This constraint would regularize the distribution of surfels, guarantee easier access, and avoid duplicate allocation. Triangles are loosely organized in the level of blocks, linking adjacent surfels. In the context of mesh, a surfel could be also interpreted as a triangle vertex.

3.2 3D Representations

Voxel and PSDF. In most volumetric reconstruction systems SDF or truncated SDF (TSDF) (denoted by D) of a voxel is updated when observed 3D points fall in its neighbor region. Projective signed distances from measurements, which could be explained as SDF observations, are integrated by computing weight average. Newcombe [19] suggests that it can be regarded as the solution of a maximum likelihood estimate of a joint Gaussian distribution taking SDF as a random variable. While Gaussian distribution could depict the uncertainty of data noise, it might fail to handle outlier inputs which are common in reconstruction tasks using consumer-level sensors. Moreover, SDF should depict the projective distance from a voxel to its *closest* surface. During integration, however, it is likely that *non-closest* surface points are taken into account, which should also be regarded as outlier SDF observations. In view of this, we introduce another random variable π to denote the inlier ratio of SDF, initially used in [29] to model the inlier ratio of 3D points:

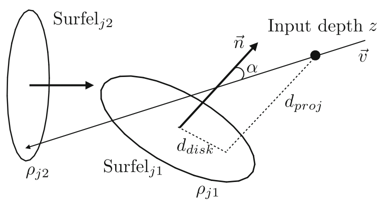
$$p(D_i^o \mid D, \tau_i, \pi) = \pi \mathcal{N}(D_i^o; D, \tau_i^2) + (1 - \pi)\mathcal{U}(D_i^o; D_{min}, D_{max}), \tag{1}$$

where D_i^o reads an SDF observation computed with depth measurements, τ_i is the variance of the SDF observation, $\mathcal{N}$ and $\mathcal{U}$ are Gaussian and Uniform distributions.



V
$\mathcal{N}(z_1^o \mid z_1, \iota_1^2)$
z_1
D_1^o
z_2
D_2^o
$\mathcal{N}(D|\mu,\sigma)\mathcal{B}(\pi|a,b)$

(a) PSDF update



(b) Inlier ratio estimation

Fig. 3. Left, illustration of PSDF distribution with multiple SDF observations (D_1^o is likely to be an inlier observation, D_2^o is possibly an outlier). The red curves show Gaussian distributions per observation. The blue curve depicts a Beta distribution which intuitively suggests that inlier observations should be around D_1^o. Right, estimation of SDF inlier ratio ρ involving observed input 3D point and already known surfels. (Color figure online)

Following [29], the posterior of PSDF can be parameterized by a Beta distribution multiplying a Gaussian distribution $\mathcal{B}(a,b)\mathcal{N}(\mu;\sigma^2)$, given a series of observed input SDF measurements. The details will be discussed in Sect. 4.1. The parameters a, b, μ, σ of the parameterized distribution are maintained per voxel.

Surfel. A surfel in our pipeline is formally defined by a position $\boldsymbol{x}$, a normal $\boldsymbol{n}$, and a radius r. Since a certain surfel is constrained on an edge in the volume, $\boldsymbol{x}$ is generally an interpolation of 2 adjacent voxel corners.

Triangle. A triangle consists of 3 edges, each linking two adjacent surfels. These surfels can be located in different voxels, even different blocks. In our framework triangles are mainly extracted for rendering; the contained topology information may be further utilized in extensions.

Depth Input. We receive depth measurements from sensors as input, while sensor poses are assumed known. Each observed input depth z^o is modeled as a random variable subject to a simple Gaussian distribution:

$$p(z^o \mid z, \iota) = \mathcal{N}(z^o; z, \iota^2), \tag{2}$$

where ι can be estimated from a precomputed sensor error model.

3.3 Pipeline

In general, our system will first generate a set of surfels S_{t-1} in the volumetric PSDF field PD_{t-1}. Meanwhile, mesh triangle set T_t is also determined by linking reliable surfels in S_{t-1}. S_{t-1} explicitly defines the surfaces of the scene, hence can be treated as a trustworthy geometry cue to estimate outlier ratio of the input depth data z_t^o. PD_{t-1} is then updated to PD_t by fusing evaluated depth data distribution z_t via Bayesian updating. The process will be performed iteratively every time input data come, as depicted in Fig. 2(b). We assume the poses of the sensors are known and all the computations are in the world coordinate system.

4 PSDF Fusion and Surface Reconstruction

4.1 PSDF Fusion

Similar to [20], in order to get SDF observations of a voxel V given input 3D points from depth images, we first project V to the depth image to find the projective closest depth measurement z_i. Signed distance from V to the input 3D data is defined by

$$D_i = z_i - z^V, \tag{3}$$

where z^V is a constant value, the projective depth of V along the scanning ray.

The observed D_i^o is affected by the variance ι_i of z_i in Eq. 2 contributing to the Gaussian distribution component in Eq. 1, provided D_i is an inlier. Otherwise, D_i would be counted in the uniform distribution part. Figure 3(a) illustrates the possible observations of SDF in one voxel.

Variance ι_i can be directly estimated by pre-measured sensor priors such as proposed in [21]. In this case, due to the simple linear form in Eq. 3, we can directly set $D_i^o = z_i^o - z^V$ and $\tau_i = \iota_i$ in Eq. 1.

Given a series of independent observations D_i^o, we can derive the posterior

$$p(D, \pi \mid D_1^o, \tau_1 \cdots D_n^o, \tau_n) \propto p(D, \pi) \prod_i p(D_i^o \mid D, \tau_i^2, \pi), \tag{4}$$

where $p(D, \pi)$ is a prior and $p(D_i^o \mid D, \tau_i^2, \pi)$ is defined by Eq. 1. It would be intractable to evaluate the production of such distributions with additions. Fortunately, [29] proved that the posterior could be approximated by a parametric joint distribution:

$$p(D, \pi \mid D_1^o, \tau_1 \cdots D_n^o, \tau_n) \propto \mathcal{B}(\pi \mid a_n, b_n)\mathcal{N}(D \mid \mu_n, \sigma_n^2), \tag{5}$$

therefore the problem could be simplified as a parameter estimation in an incremental fashion:

$$\mathcal{B}(\pi \mid a_n, b_n)\mathcal{N}(D \mid \mu_n, \sigma_n^2) \propto p(D_n^o | D, \tau_n^2, \pi)\mathcal{B}(\pi \mid a_{n-1}, b_{n-1})\mathcal{N}(D \mid \mu_{n-1}, \sigma_{n-1}^2). \tag{6}$$

In [29] by equating first and second moments of the random variables π and D, the parameters could be easily updated, in our case evoking the change of SDF distribution and its inlier probability:

$$\mu_n = \frac{C_1}{C_1 + C_2} m + \frac{C_2}{C_1 + C_2} \mu_{n-1}, \tag{7}$$

$$\mu_n^2 + \sigma_n^2 = \frac{C_1}{C_1 + C_2}(s^2 + m^2) + \frac{C_2}{C_1 + C_2}(\sigma_{n-1}^2 + \mu_{n-1}^2), \tag{8}$$

$$s^2 = 1/(\frac{1}{\sigma_{n-1}^2} + \frac{1}{\tau_n^2}), \tag{9}$$

$$m = s^2(\frac{\mu_{n-1}}{\sigma_{n-1}^2} + \frac{D_i^o}{\tau_n^2}), \tag{10}$$

$$C_1 = \frac{a_{n-1}}{a_{n-1} + b_{n-1}} \mathcal{N}(D_i^o; \mu_{n-1}, \sigma_{n-1}^2 + \tau_n^2), \tag{11}$$

$$C_2 = \frac{b_{n-1}}{a_{n-1} + b_{n-1}} \mathcal{U}(D_i^o; D_{min}, D_{max}), \tag{12}$$

the computation of a and b are the same as [29] hence ignored here. In our experiments we find that a truncated D_i^o leads to better results, as it directly rejects distant outliers. SDF observations from non-closest surfaces are left to be handled by PSDF.

4.2 Inlier Ratio Evaluation

In Eqs. 11–12, the expectation of $\mathcal{B}(\pi \mid a, b)$ is used to update the coefficients, failing to make full use of known geometry properties in scenes. In our pipeline, available surface geometry is considered to evaluate the inlier ratio ρ_n of D_n^o, replacing the simple $\frac{a_{n-1}}{a_{n-1}+b_{n-1}}$. Note ρ_n is computed per-frame in order to update C_1, C_2; π is still parameterized by a and b.

ρ_n can be determined by whether an input point z is near the closest surface of a voxel and results in an inlier SDF observation. We first cast the scanning ray into the volume and collect the surfels maintained on the voxels hit by the ray. Given the surfels, 3 heuristics are used, as illustrated in Fig. 3(b).

Projective Distance. This factor is used to measure whether a sampled point is close enough to a surfel which is assumed the nearest surface to the voxel:

$$w_{dist} = \exp\left(\frac{-|\boldsymbol{n}^T(\boldsymbol{x} - z\boldsymbol{v})|^2}{2\theta^2}\right), \tag{13}$$

where $\boldsymbol{v}$ is the normalized direction of the ray in world coordinate system and θ is a preset parameter proportional to the voxel resolution.

Angle. Apart from projective distance, we consider angle as another factor, delineating the possibility that a scanning ray will hit a surfel. We use the empirical angle weight in [15]:

$$w_{angle} = \begin{cases} \frac{\cos(\alpha) - \cos(\alpha_{max})}{1 - \cos(\alpha_{max})}, & \text{if } \alpha < \alpha_{max}, \\ w_{angle}^0, & \text{else}, \end{cases} \tag{14}$$

where $\alpha = \langle \boldsymbol{n}, \boldsymbol{v} \rangle$, α_{max} is set to 80° and w_{angle}^0 assigned to 0.1.

Radius. The area that surfels could influence vary, due to the local shape of the surface. The further a point is away from the center of a surfel, the less possible it would be supported. A sigmoid-like function is used to encourage a smooth transition of the weight:

$$w_{radius} = \gamma + \frac{2(1-\gamma)}{1 + \exp(\frac{-d_{disk}}{r})}, \tag{15}$$

$$d_{disk} = \sqrt{(z\boldsymbol{v} - \boldsymbol{x})^T (I - \boldsymbol{n}\boldsymbol{n}^T)(z\boldsymbol{v} - \boldsymbol{x})}, \tag{16}$$

where parameter $\gamma \in [0, 1)$ and is set to 0.5 in our case.

Putting all the factors together, we now have

$$\rho = w_{dist} \cdot w_{radius} \cdot w_{angle}. \tag{17}$$

To compute the ρ predicted by all the surfels, one may consider either summations or multiplications. However, we choose the highest ρ instead – intuitively a depth measurement is a sample on a surface, corresponding to exactly one surfel. A more sophisticated selection might include a ray consistency evaluation [27,28,32] where occlusion is handled. When a new area is explored where no surfels have been extracted, we use a constant value $\rho_{pr} = 0.1$ to represent a simple occupancy prior in space, hence we have

$$\rho = \max_j \{\rho_{pr}, \rho_1, \cdots, \rho_j, \cdots\}. \tag{18}$$

4.3 Surface Extraction

PSDF implicitly defines zero crossing surfaces and decides whether they are true surfaces. The surface extraction is divided into two steps.

Surfel Generation. In this stage we enumerate zero-crossing points upon 3 edges of each voxel and generate surfels when condition

$$\begin{gathered} \mu^{i1} \cdot \mu^{i2} < 0, \\ \frac{a^{i1}}{a^{i1} + b^{i1}} > \pi_{thr} \text{ and } \frac{a^{i2}}{a^{i2} + b^{i2}} > \pi_{thr}, \end{gathered} \tag{19}$$

are satisfied, where $i1$ and $i2$ are indices of adjacent voxels and π_{thr} is a confidence threshold. Supported by the reliable update of the PSDF, false surfaces could be rejected and duplicates could be removed. According to our experiments, our framework is not sensitive to π_{thr}; 0.4 would work for all the testing scenes. A surfel's position $\boldsymbol{x}$ would be the linear interpolation of corresponding voxels' positions $\boldsymbol{x}^i$ indexed by i, and the radius would be determined by σ of adjacent voxels, simulating its affecting area. Normal is set to normalized gradient of the SDF field, as mentioned in [20].

$$\boldsymbol{x} = \frac{|\mu^{i2}|}{|\mu^{i1}| + |\mu^{i2}|}\boldsymbol{x}^{\boldsymbol{i1}} + \frac{|\mu^{i1}|}{|\mu^{i1}| + |\mu^{i2}|}\boldsymbol{x}^{\boldsymbol{i2}}, \tag{20}$$

$$r = \frac{|\mu^{i2}|}{|\mu^{i1}| + |\mu^{i2}|}\sigma^{i1} + \frac{|\mu^{i1}|}{|\mu^{i1}| + |\mu^{i2}|}\sigma^{i2}, \tag{21}$$

$$\boldsymbol{n} = \nabla\mu / ||\nabla\mu||. \tag{22}$$

Triangle Generation. Having sufficient geometry information within surfels, there is only one more step to go for rendering-ready mesh. The connections between adjacent surfels are determined by the classical MarchingCubes [17] method. As a simple modification, we reject edges in the voxel whose σ is larger than a preset parameter σ_{thr}. This operation will improve the visual quality of reconstructed model while preserving surfels for the prediction stage.

5 Experiments

We test our framework (denoted by *PSDF*) on three RGB-D datasets: TUM [25], ICL-NUIM [9], and dataset from Zhou and Koltun [34]. Our method is compared against [6] (denoted by *TSDF*) which incrementally extracts mesh in spatial-hashed TSDF volumes. The sensors' poses are assumed known for these datasets, therefore the results of *TSDF* should be similar to other state-of-the-art methods such as [22,23] where TSDF integration strategies are the same. We demonstrate that our method reconstructs high quality surfaces by both qualitative and quantitative results. Details are preserved while noise is removed in the output models. The running speed for online mesh extraction is also improved by avoiding computations on false surfel candidates.

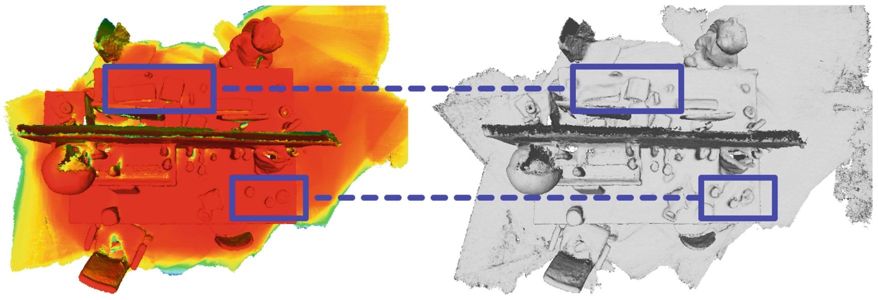


Fig. 4. Comparison of output mesh of *frei3_long_office* scene. Left, *PSDF*. Right, *TSDF*. Our method generates smooth surfaces and clean edges of objects, especially in blue boxes. (Color figure online)

For [25,34] we choose a voxel size of 8 mm and $\sigma_{thr} = 16$ mm; for [9] voxel size is set to 12 mm and $\sigma_{thr} = 48$ mm. The truncation distance is set to $3\times$ voxel size plus $3 \times \tau$; with a smaller truncation distance we found strides and holes in meshes. Kinect's error model [21] was adopted to get ι where the factor of angle was removed, which we think might cause double counting considering w_{angle} in the inlier prediction stage. The program is written in C++/CUDA 8.0 and runs on a laptop with an Intel i7-6700 CPU and an NVIDIA 1070 graphics card.

5.1 Qualitative Results

We first show that *PSDF* accompanied by the related mesh extraction algorithm produces higher quality surfaces than *TSDF*. Our results are displayed with shaded heatmap whose color indicates the inlier ratio of related SDF. Both geometry and probability properties can be viewed in such a representation.

Figure 4 shows that *PSDF* outperforms *TSDF* by generating clean boundaries of small objects and rejecting noisy areas on the ground. In Fig. 5, in addition to the results of *TSDF*, we also display the reconstructed mesh from offline

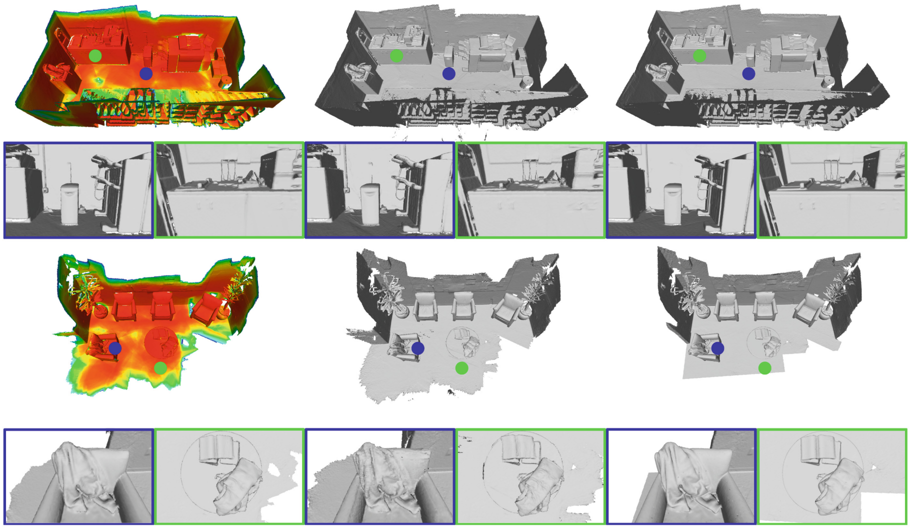


Fig. 5. Output mesh of the *copyroom* and *lounge* scenes. From left to right, *PSDF*, *TSDF*, mesh provided by [34]. Zoomed in regions show that our method is able to filter outliers while maintaining complete models and preserving details. Best viewed in color. (Color figure online)

methods provided by [34] as references. It appears that our method produces results very similar to [34]. While guaranteeing well-covered reconstruction of scenes, we filter outliers and preserve details. In *copyroom*, the wires are completely reconstructed, one of which above PC is smoothed out in [34] and only partially recovered by *TSDF*. In *lounge*, we can observe a complete shape of table, and de-noised details in the clothes.

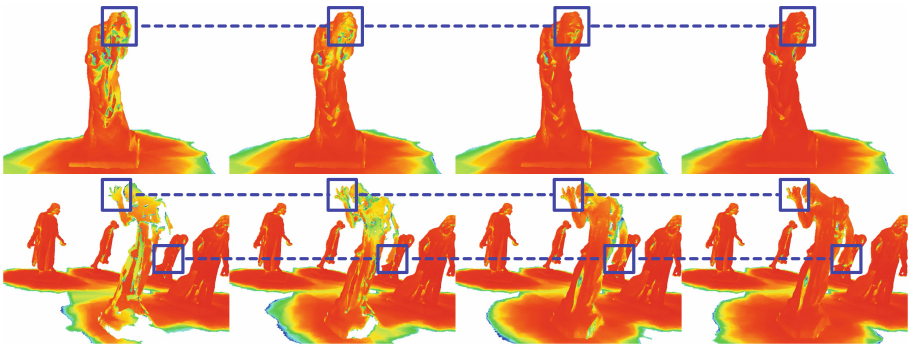


Fig. 6. Incremental reconstruction of the *burghers* dataset. Fluctuation of probability could be observed at error-prone regions as an indication of uncertainty propagation. (Color figure online)

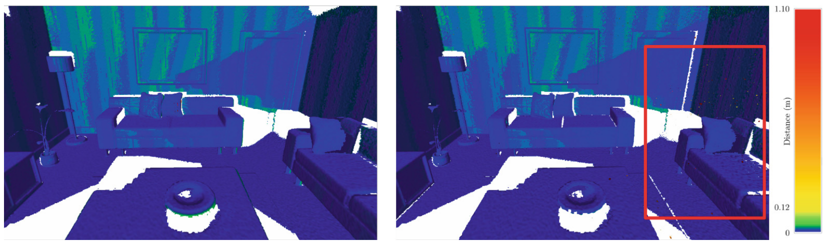
We also visualize the incremental update of π by rendering $\mathbb{E}(\beta(\pi|a,b)) = a/(a+b)$ as the inlier ratio of reconstructed mesh in sequence using colored heatmap. Figure 6 shows the fluctuation of confidence around surfaces. The complex regions such as fingers and wrinkles on statues are more prone to noise, therefore apparent change of shape along with color can be observed.

5.2 Quantitative Results

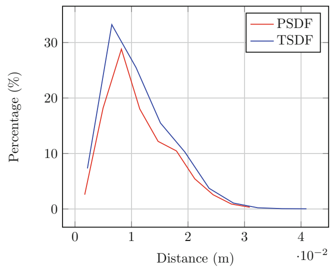
Reconstruction Accuracy. We reconstruct mesh of the synthetic dataset *livingroom2* with added noise whose error model is presented in [9]. Gaussian noise

Table 1. Statistics of the point-to-point distances from the reconstructed model vertices to the ground truth point cloud. *PSDF* yields better reconstruction accuracy.

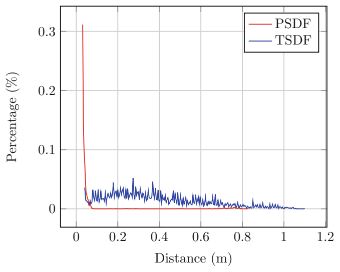
Method	MEAN (m)	STD (m)
PSDF	**0.011692**	**0.015702**
TSDF	0.022556	0.076120



(a) Point-to-point distance heatmap. Left, *PSDF*. Right, *TSDF*.



(b) Histogram head: 0 to 4*cm*



(c) Histogram tail: 4*cm* to ∞

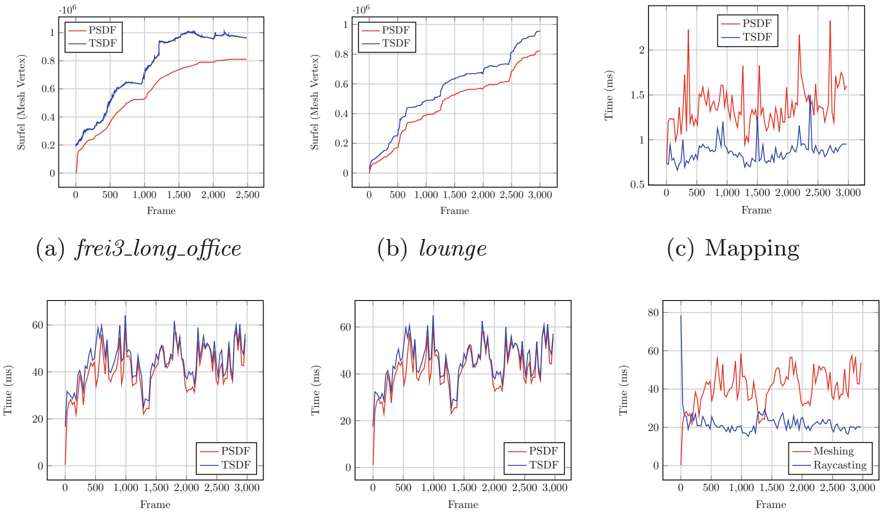
Fig. 7. Comparison of output mesh quality of *PSDF* and *TSDF*. First row: heatmap of the point-to-point distance from the reconstructed model to the ground truth. Notice the outliers in the red box. Second row, the distance histogram divided into two parts to emphasize the existence of outliers. As shown in the histogram tail, *TSDF* generates many outliers, while *PSDF* avoids such problems. Best viewed in color. (Color figure online)

on inverse depth plus local offsets is too complicated for our error model, therefore we simplify it by assigning inverse sigma at certain inverse depths to ι_i. The mesh vertices are compared with the ground truth point cloud using the free software *CloudCompare* [3].

Table 1 indicates that *PSDF* reconstructs better models than *TSDF*. Further details in Fig. 7 suggest that less outliers appear in the model reconstructed by *PSDF*, leading to cleaner surfaces.

Mesh Size. Our method maintains the simplicity of mesh by reducing false surface candidates caused by noise and outliers. As shown in Fig. 8 and Table 2, *PSDF* in most cases generates less vertices (% 20) than *TSDF*, most of which are outliers and boundaries with low confidence. Figure 8 shows that the vertex count remains approximately constant when a loop closure occurred in *frei3_long_office*, while the increasing rate is strictly constrained in the *lounge* sequence where there is no loop closure.

Time. To make a running time analysis, we take real world *lounge* as a typical sequence where noise is common while the camera trajectory fits scanning behavior of humans. As we have discussed, evaluation of inlier ratio was performed, increasing total time of the fusion stage. However we find on a GPU,



(a) *frei3_long_office* (b) *lounge* (c) Mapping

(d) Meshing (e) All (f) Meshing vs raycasting

Fig. 8. (a)–(b), increasing trend of mesh vertices (surfels). (a), *frei3_long_office* scene in which a loop closure occurs at around 1500 frames. (b), *lounge* scene without loop closures. Mesh generated by *PSDF* consumes approximately 80% the memory of *TSDF*. (c)–(f), running time analysis of our approaches and compared methods on *lounge* scene. *PSDF* is 1ms slower on GPU due to additional inlier prediction stage, but will save more time for surface extraction, causing a faster speed in general. The speed of online meshing is slower than raycasting but comparable.

Table 2. Evaluation of memory and time cost on various datasets. PSDF reduces model's redundancy by rejecting false surfaces and noise. The mapping stage of TSDF is faster, but in general PSDF spends less time considering both mapping and meshing stages.

Dataset	Frames	MEMORY (vertex count)		TIME (ms)				
		PSDF	TSDF	PSDF		TSDF		Ray-casting
				Mapping	Meshing	Mapping	Meshing	
burghers	11230	**1362303**	1438818	1.053	39.691	**0.771**	**38.907**	27.761
copyroom	5490	**1222196**	1328397	1.329	**39.090**	**0.909**	41.595	21.292
garden	6152	**922534**	978206	1.473	68.224	**0.907**	**66.680**	25.479
lounge	3000	**821360**	955218	1.331	**41.270**	**0.881**	45.248	22.097
livingroom1	965	529305	**518885**	1.255	33.090	**0.743**	**32.865**	27.248
livingroom2	880	**609421**	683008	1.407	**33.446**	**0.759**	40.230	29.069
office1	965	**667614**	674034	1.264	**24.933**	**0.799**	27.654	26.957
office2	880	**712685**	883138	1.322	**35.029**	**0.767**	45.923	29.981
frei1_xyz	790	**212840**	352444	1.193	**45.163**	**1.149**	71.670	19.796
frei3_long_office	2486	**811092**	963875	2.424	**159.417**	**1.375**	161.485	26.545

even with a relatively high resolution, the average increased time is at the scale of ms (see Fig. 8(c) and Table 2) and can be accepted.

When we come to meshing, we find that by taking the advantage of PSDF fusion and inlier ratio evaluation, unnecessary computations can be avoided and *PSDF* method runs faster than *TSDF*, as plotted in Fig. 8(d). The meshing stage is the runtime bottleneck of the approach, in general the saved time compensate for the cost in fusion stage, see Fig. 8(e) and Table 2.

We also compare the time of meshing to the widely used ray-casting that renders surfaces in real-time. According to Table 2, in some scenes where sensor is close to the surfaces performing scanning, less blocks are allocated in viewing frustum and the meshing speed could be comparative to ray-casting, as illustrated in Fig. 8(f). As for other scenes requiring a large scanning range, especially *frei3_long_office* where more blocks in frustum have to be processed, ray-casting shows its advantage. We argue that in applications that only require visualization, ray-casting can be adopted; otherwise meshing offers more information and is still preferable.

6 Conclusions

We propose PSDF, a joint probabilistic distribution to model the 3D geometries and spatial uncertainties. With the help of Bayesian updating, parameters of the distribution could be incrementally estimated. Built upon a hybrid data structure, our framework can iteratively generate surfaces from the volumetric PSDF field and update PSDF values through reliable probabilistic data fusion supported by reconstructed surfaces. As an output, high-quality mesh can be generated in real-time with duplicates removed and noise cleared.

In the future, we seek to improve our framework by employing more priors to enrich the PSDF distribution. Localization modules will also be integrated in the probabilistic framework for a complete SLAM system.

Acknowledgments. This work is supported by the National Natural Science Foundation of China (61632003, 61771026), and National Key Research and Development Program of China (2017YFB1002601).

References

1. Botsch, M., Kobbelt, L.: High-quality point-based rendering on modern GPUs. In: Proceedings of Pacific Conference on Computer Graphics and Applications, pp. 335–343 (2003)
2. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2015)
3. CloudCompare-project: CloudCompare. http://www.cloudcompare.org/
4. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of ACM SIGGRAPH, pp. 303–312 (1996)
5. Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., Theobalt, C.: BundleFusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. ACM Trans. Graph. **36**(3), 24 (2017)
6. Dong, W., Shi, J., Tang, W., Wang, X., Zha, H.: An efficient volumetric mesh representation for real-time scene reconstruction using spatial hashing. In: Proceedings of IEEE International Conference on Robotics and Automation (2018)
7. Dzitsiuk, M., Sturm, J., Maier, R., Ma, L., Cremers, D.: De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3976–3983 (2017)
8. Forster, C., Pizzoli, M., Scaramuzza, D.: SVO: fast semi-direct monocular visual odometry. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 15–22 (2014)
9. Handa, A., Whelan, T., Mcdonald, J., Davison, A.J.: A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 1524–1531 (2014)
10. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: an efficient probabilistic 3D mapping framework based on octrees. Auton. Rob. **34**(3), 189–206 (2013)
11. Kähler, O., Prisacariu, V., Valentin, J., Murray, D.: Hierarchical voxel block hashing for efficient integration of depth images. IEEE Rob. Autom. Lett. **1**(1), 192–197 (2016)
12. Kähler, O., Prisacariu, V.A., Ren, C.Y., Sun, X., Torr, P., Murray, D.: Very high frame rate volumetric integration of depth images on mobile devices. IEEE Trans. Vis. Comput. Graph. **21**(11), 1241–1250 (2015)
13. Keller, M., Lefloch, D., Lambers, M., Weyrich, T., Kolb, A.: Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: Proceedings of International Conference on 3DTV, pp. 1–8 (2013)
14. Klingensmith, M., Dryanovski, I., Srinivasa, S.S., Xiao, J.: CHISEL: real time large scale 3D reconstruction onboard a mobile device using spatially-hashed signed distance fields. In: Proceedings of Robotics: Science and Systems, pp. 1–8 (2015)

15. Kolev, K., Tanskanen, P., Speciale, P., Pollefeys, M.: Turning mobile phones into 3D scanners. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3946–3953 (2014)
16. Lefloch, D., Kluge, M., Sarbolandi, H., Weyrich, T., Kolb, A.: Comprehensive use of curvature for robust and accurate online surface reconstruction. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2349–2365 (2017)
17. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of ACM SIGGRAPH, vol. 6, pp. 7–9 (1987)
18. Marton, Z.C., Rusu, R.B., Beetz, M.: On fast surface reconstruction methods for large and noisy point clouds. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3218–3223 (2009)
19. Newcombe, R.: Dense visual SLAM. Ph.D. thesis, Imperial College London, UK (2012)
20. Newcombe, R.A., et al.: KinectFusion: real-time dense surface mapping and tracking. In: Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 127–136 (2011)
21. Nguyen, C.V., Izadi, S., Lovell, D.: Modeling kinect sensor noise for improved 3D reconstruction and tracking. In: Proceedings of IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, pp. 524–530 (2012)
22. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. **32**(6), 169 (2013)
23. Prisacariu, V.A., et al.: InfiniTAM v3: a framework for large-scale 3D reconstruction with loop closure. arXiv e-prints, August 2017
24. Steinbrücker, F., Sturm, J., Cremers, D.: Volumetric 3D mapping in real-time on a CPU. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 2021–2028 (2014)
25. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proceedings of International Conference on Intelligent Robot Systems, pp. 573–580 (2012)
26. Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., Pollefeys, M.: Live metric 3D reconstruction on mobile phones. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 65–72 (2013)
27. Ulusoy, A.O., Black, M.J., Geiger, A.: Patches, planes and probabilities: a non-local prior for volumetric 3D reconstruction. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3280–3289 (2016)
28. Ulusoy, A.O., Geiger, A., Black, M.J.: Towards probabilistic volumetric reconstruction using ray potentials. In: Proceedings of International Conference on 3D Vision, pp. 10–18 (2015)
29. Vogiatzis, G., Hernández, C.: Video-based, real-time multi-view stereo. Image Vis. Comput. **29**(7), 434–441 (2011)
30. Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., Mcdonald, J.: Kintinuous: spatially extended KinectFusion. Rob. Auton. Syst. **69**(C), 3–14 (2012)
31. Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J.J., McDonald, J.: Real-time large-scale dense RGB-D SLAM with volumetric fusion. Int. J. Robot. Res. **34**(4–5), 598–626 (2015)
32. Woodford, O.J., Vogiatzis, G.: A generative model for online depth fusion. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 144–157. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_11

33. Zeng, M., Zhao, F., Zheng, J., Liu, X.: Octree-based fusion for realtime 3D reconstruction. Graph. Models **75**(3), 126–136 (2013)
34. Zhou, Q., Koltun, V.: Dense scene reconstruction with points of interest. ACM Trans. Graph. **32**(4), 112 (2013)
35. Zienkiewicz, J., Tsiotsios, A., Davison, A., Leutenegger, S.: Monocular, real-time surface reconstruction using dynamic level of detail. In: Proceedings of International Conference on 3D Vision, pp. 37–46 (2016)