Critical Regularizations for Neural Surface Reconstruction in the Wild

Jingyang Zhang* Yao Yao Shiwei Li Tian Fang
David McKinnon Yanghai Tsin Long Quan
The Hong Kong University of Science and Technology Apple

Abstract

Neural implicit functions have recently shown promising results on surface reconstructions from multiple views. However, current methods still suffer from excessive time complexity and poor robustness when reconstructing unbounded or complex scenes. In this paper, we present RegSDF, which shows that proper point cloud supervisions and geometry regularizations are sufficient to produce high-quality and robust reconstruction results. Specifically, RegSDF takes an additional oriented point cloud as input, and optimizes a signed distance field and a surface light field within a differentiable rendering framework. We also introduce the two critical regularizations for this optimization. The first one is the Hessian regularization that smoothly diffuses the signed distance values to the entire distance field given noisy and incomplete input. And the second one is the minimal surface regularization that compactly interpolates and extrapolates the missing geometry. Extensive experiments are conducted on DTU, BlendedMVS, and Tanks and Temples datasets. Compared with recent neural surface reconstruction approaches, RegSDF is able to reconstruct surfaces with fine details even for open scenes with complex topologies and unstructured camera trajectories.

1. Introduction

Surface reconstruction from multiple calibrated views is one of the key tasks in 3D computer vision. Traditionally, the task is solved by first estimating a point cloud from images by multi-view stereo (MVS) [1, 4, 23, 26], and then extracting a triangular mesh from the point cloud [8, 11, 14]. Recently, neural implicit surface reconstruction also shows comparable or even better results especially for textureless and non-Lambertian surfaces. These methods apply multi-layer perceptrons (MLP) to map a space coordinate to different geometry properties, such as density [16], occupancy [17] or signed distance to the nearest surface point [25, 28, 29, 31]. The MLP can be fit into explicit geometry representations such as contour masks [17, 29], depth maps [31] and point clouds [24], or can be further optimized with the scene appearance through differentiable rendering [16–18, 25, 28–30].

However, it is still a challenging task to conduct surface reconstruction in the wild. First, textureless or non-Lambertian surfaces, which exist in real-world scenes, are hard to be recovered even for learning-based methods. Second, camera trajectories of real-world data may be unstructured instead of object-centric. In traditional mesh reconstruction pipelines, although multi-view stereo [23, 26] has been proven to be effective for a variety of different scenes, the reconstructed point cloud inevitably suffers from missing or noisy geometries, which are difficult to be corrected in later mesh extraction steps. On the other hands, recent neural surface methods [16, 18, 25, 28] is able to generate surfaces directly from multi-view images. However, the operation of volumetric rendering [16, 18, 25, 28] with implicit functions is time-consuming, and those methods are originally designed for reconstructions of object-centric captures rather than open scenes with unstructured camera trajectories.

In this paper, we propose RegSDF, a neural framework for surface reconstruction from multi-view images for a broad variety of scenes. Implementation-wise, we choose a signed distance field (SDF) as the geometry representation and a surface light field as the appearance model to generate rendered images for network training. To take advantages of well-established MVS pipelines, we additionally apply an oriented point cloud from MVS as input to our reconstruction, where the SDF will be fit into observed data points and also normal directions. Two critical regularizations, namely the Hessian regularization of second derivatives and the minimal surface constraint, are proposed for robust neural surface reconstruction. The Hessian regularization is designed to let the signed distance value smoothly diffuse to the entire signed distance field, which is important for reconstructing complete surface from incomplete and noisy point clouds. Meanwhile, the minimal surface regularization is introduced to compactly interpolate holes and

*This work is done when Jingyang Zhang was an intern at Apple

extrapolate missing parts in the implicit surface. We show in experiments that the proposed two regularizations, along with the point cloud supervision, are sufficient to produce high-quality and robust reconstruction results from multi-view images.

The proposed method has been evaluated on *DTU* [6], *BlendedMVS* [27], and *Tanks and Temples* [10] datasets. We show by both qualitative and quantitative results that our method outperforms other neural implicit surface reconstruction systems by higher surface accuracy, stronger generalization ability to complex scenes, and shorter training time. Also, compared with traditional meshing methods, the proposed framework is robust against point cloud noise and can produce realistic rendered images.

2. Related Works

Differentiable rendering. Differentiable rendering jointly optimizes all scene parameters including the geometry by inverting the rendering process. In this section we only review neural-based methods. There are two lines of works on the neural differentiable rendering, based on respectively radiance field [16] and surface light field [29].

The radiance field representation assumes that radiance is emitted from all space points and applies the volumetric rendering [16, 18, 25, 28] to synthesize images. A soft density field is applied to represent the geometry of the scene. The density value could be further interpreted as occupancy [18] or signed distance [25, 28] for explicit geometry regularizations. There are two major drawbacks of volumetric rendering based methods. First, it is difficult to accurately extract the surface from the soft density field. Second, the volume rendering requires expensive MLP sampling along the viewing ray.

The surface light field representation follows the assumption that lights are reflected from a opaque geometry surface. For example, DVR [17] regards the object surface as the zero-crossing interval of the occupancy field, and applies root-finding such as the secant method to find the surface intersection. This process can be accelerated in SDF-based methods [29, 31] by using sphere tracing. However, such methods can only optimize the surface locally, and additional geometric supervisions like masks or depth maps are required to as inputs. As a result, the quality of the surface output will depend on the input data quality. In this paper, we show that proper regularizations, including supervision on second order derivatives and minimal surface constraint, is able to robustly optimize the geometry even for incomplete or missing point cloud inputs.

Neural implicit surface reconstruction. Recent neural surface reconstruction methods apply MLPs to represent geometries as implicit density field [16], occupancy field [15, 17, 18, 20–22] or signed distance field [9, 12, 13, 19, 24, 25, 28, 29, 31]. The neural surface can be initialized by intermediate geometry representations such as masks, depth maps and point clouds, and can be further optimized with the scene appearance by differentiable rendering.

Neural implicit surface reconstruction has several advantages over traditional pipelines. First, as the surface is directly modeled and optimized, the final result is optimal with respect to input images, while traditional pipelines produce sub-optimal results through lossy conversions from depth maps to point cloud, and then to triangular meshes. Second, the neural appearance representation can naturally model the view dependent appearance, which is suitable for modeling and reconstructing non-Lambertian surfaces. Inspired by recent works [29, 31], we apply the SDF and surface light field to represent the geometry and appearance of the scene.

Geometry supervision for neural surface. Previous works introduce intermediate geometry representations to directly guide the implicit function or provide rough geometric prior. DVR [17] and IDR [29] take object masks as inputs to supervise object silhouettes during the network optimization. However, it is difficult to automatically estimate perfect masks for input images, and the silhouette information is not adequate for recovering concave geometries in the scene. MVSDF [31] introduces depth maps from MVS and use SDF fusion to directly supervise the SDF in the whole space, but the method still suffers from incompleteness and noises, making the system unstable for unstructured inputs or complex scenes. In contrast, our methods apply oriented point cloud as inputs, but also takes advantages of input images to refine fine detail geometries in the surface.

Traditional implicit surface reconstruction. Traditional methods [2, 7, 8] use volumetric representations like octrees to store occupancy or signed distance fields in space. Function values and their derivatives are fit to the input data and the Marching Cubes [14] algorithm is applied to extract surfaces as a level set. These methods can generate triangular meshes with fine details with affordable memory consumption. However, volumetric representation has no intrinsic smoothness compared to neural representation so it is more sensitive to noisy point clouds. Also, octrees are hard to naturally expand to places without observed data points, resulting in missing surfaces when input point clouds are incomplete. Finally, the volumetric representation is hard to model the view-dependent color of the surface, making it hard to optimize for non-Lambertion surfaces. In contrast, neural representation is robust against noisy or missing data, and could be optimized with view-dependent appearance models for realistic view synthesis.

3. Method

3.1. Geometry and Appearance Representations

Our geometry and appearance modelling, and the construction of differentiable surface intersections closely follow the works in [29, 31]. We define the surface $\mathcal{S}$ as the zero level set of a Signed Distance Function f represented by a MLP with network of parameters θ. Let $\Omega \subset \mathbb{R}^3$ be the domain of the scene bounding box. The network function $f : \Omega \to \mathbb{R}$ takes a space location $\mathbf{x}$ as input and yields the distance from the query location to the nearest surface point. Following the previous level set method [32], we assume f is Lipschitz continuous. The scene interior $\mathcal{V}$ is represented as

$$\mathcal{V} = \{\mathbf{x} \in \Omega \mid f(\mathbf{x}; \theta) < 0\},$$

and the surface $\mathcal{S}$ as

$$\mathcal{S} = \partial\mathcal{V} = \{\mathbf{x} \in \Omega \mid f(\mathbf{x}; \theta) = 0\}. \tag{1}$$

And the appearance is modeled as a surface light field represented by another MLP g with network parameters ϕ. The network $g : \mathcal{S} \times \mathbb{R}^3 \times \mathbb{R}^3 \to [0, 1]^3$ takes an intersection point $\mathbf{x}$ on surface, the unit normal vector of the point $\mathbf{n}$, and the unit view direction vector $\mathbf{v}$ as inputs, and yields the RGB color of this surface point $\mathbf{c}$:

$$\mathbf{c} = g(\mathbf{x}, \mathbf{n}, \mathbf{v}; \phi),$$

where $\mathbf{x} \in \mathcal{S}$ and the normal can be auto-differentiated from the SDF as $\mathbf{n} = \nabla_{\mathbf{x}} f(\mathbf{x}; \theta)$.

The intersection points are obtained by sphere tracing algorithm and is not differentiable with respect to the network parameters θ. To construct a differentiable version $\mathbf{x}(\theta)$, we can differentiate both sides of $f(\mathbf{x}(\theta); \theta) = 0$ with respect to θ and rearrange the terms. Given the current SDF network parameters θ_0, unit view direction vector $\mathbf{v}$ and the intersection point $\mathbf{x}_0$, the differentiable surface intersection $\mathbf{x}(\theta)$ is derived as:

$$\mathbf{x}(\theta) = \mathbf{x}_0 - \frac{f(\mathbf{x}_0; \theta) - f(\mathbf{x}_0; \theta_0)}{\mathbf{v} \cdot \nabla_{\mathbf{x}} f(\mathbf{x}_0; \theta_0)} \mathbf{v}, \tag{2}$$

where $f(\mathbf{x}_0; \theta_0)$ and $\nabla_{\mathbf{x}} f(\mathbf{x}_0; \theta_0)$ are constants.

3.2. Data Term

The proposed method additionally takes as input the oriented point cloud from an MVS network. We first generate point clouds without normal by Vis-MVSNet [30], estimate the point cloud normals by principle component analysis for the neighborhood of each point, and align the normal according to camera positions.

The resulted point cloud can serve as a reliable geometric guidance because they are properly filtered in the MVS step, and interior/exterior of the object can be disambiguated by the normal.

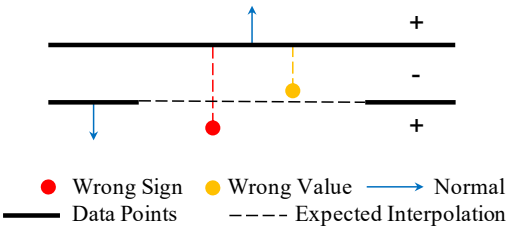

Figure 1. Illustration of wrong estimation of sign and distance value because of the incompleteness in input point clouds, which makes it unreliable to directly supervise the distance in the whole space.

Surface points. At the locations of data points, their signed distance are expected to be zero, and surface normals are expected to agree with the data points. Let $\mathbf{x}_D \in \mathcal{D}$ be all the data points, $\mathbf{n}_D$ be the unit normal vector of the data points, the data loss is given as

$$L_D = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}_D \in \mathcal{D}} \lambda_d |f(\mathbf{x})| + \lambda_n (1 - \frac{\mathbf{n}_D \cdot \nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}),$$

where λ_d and λ_n are the weights for distance term and normal term.

Boundary. We also want to obtain the distance far away from the surface as an additional boundary condition. Camera centers are good choices because cameras must be located in free space and the sign of their distances can be determined. If the camera centers are not inside the bounding box Ω, we consider the first intersection of the view lines with Ω. Denote these points as boundary points $\mathbf{x}_B \in \mathcal{B}$. For each $\mathbf{x}_B$, we find its nearest neighbor in the point cloud $\mathbf{x}_B^{nn} \in \mathcal{D}$ with unit normal vector $\mathbf{n}_B^{nn}$. The distance between $\mathbf{x}_B$ and its nearest oriented data point $\mathbf{x}_B^{nn}$ is derived as

$$d(\mathbf{x}_B) = |(\mathbf{x}_B - \mathbf{x}_B^{nn}) \cdot \mathbf{n}_B^{nn}|,$$

and the boundary loss L_B is the L1 loss between $f(\mathbf{x}_B)$ and $d(\mathbf{x}_B)$.

Although we can calculate signed distances for any $\mathbf{x} \in \Omega$ as $d(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^{nn}) \cdot \mathbf{n}^{nn}$, these distances are unreliable because point clouds from MVS may have missing points so that the real nearest point cannot be found in the input. Fig. 1 shows two examples that distances may have wrong sign or value. However, these problems are not severe for boundary points who have determined sign and are far from the surface.

3.3. Regularization

In this section, we introduce the regularization that smoothly and compactly interpolates and extrapolates distance values away from data points. In each optimization iteration, we uniformly draw samples $\mathbf{x}_R \in \mathcal{R} \subset \Omega$ from the bounding box and do the following regularization.

The gradients. The gradient magnitude is governed by the Eikonal equation $\|\nabla f(\mathbf{x})\| = 1$ in the whole space as in [29, 31]. The Eikonal loss L_E would therefore expect the gradient magnitude to be 1 in the whole space for the signed distance field without truncation, it is defined to be summed up over all the random samples and normalized by the number of samples as

$$L_E = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{x} \in \mathcal{R}} |\|\nabla f(\mathbf{x})\| - 1|.$$

The gradient directions are expected not to change rapidly. Following [2], we encourage the small second order derivatives everywhere in space and can define a loss using the second order derivatives as Hessian matrix $\mathbf{H}$:

$$L_H = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{x} \in \mathcal{R}} \|\mathbf{H} f(\mathbf{x})\|_1,$$

where $\|\cdot\|_1$ is the element-wise matrix 1-norm.

The smoothness of gradients/normals is also introduced in previous works [18] by minimizing the difference of normals between a surface point and a sample within its neighborhood. Instead of the discrete approach, we calculate analytical Hessian matrices by further back-propagating the gradient computation graph.

This regularization leads to two observations. First, the distance values that are properly supervised by the data terms can diffuse to the whole space. As results, we obtain correct distance value away from the surface, and incompleteness of the input can be interpolated. Second, the surfaces with supporting data points are smoothed to avoid overfitting the noise in the input.

The minimal surface. The surface areas without supporting data points need to be interpolated or extrapolated. The Hessian loss tends to preserve surface normals and extends the existing surfaces as planes. But when the missing surfaces are non-planar or when we want to close a cluster of single-sided point cloud, the extra surfaces may overshoot. Moreover, when the scene is not watertight, the surface between object and the boundary of bounding box is random. Inspired by an active contour method in 2D [3], we would expect the resulted surface to have minimal total area so that the interpolated and extrapolated surfaces are compact, which is illustrated in Fig. 2.

According to Eq. 1, the volume of the interior can be calculated by

$$\text{volume}(\mathcal{V}) = \int_{\Omega} H(f(\mathbf{x};\theta)) d\mathbf{x},$$

area of the implicit surface can be calculated by:

$$\begin{aligned} \text{area}(\mathcal{S}) &= \int_{\Omega} \|\nabla H(f(\mathbf{x};\theta))\| d\mathbf{x} \\ &= \int_{\Omega} \delta(f(\mathbf{x};\theta)) \|\nabla f(\mathbf{x};\theta)\| d\mathbf{x}, \end{aligned}$$

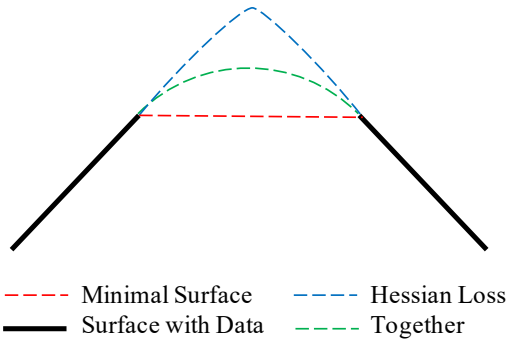

Figure 2. Effects of the regularization. The Hessian loss tends to preserve normals, and the minimal surface constraint closes the surface as planes. We can achieve a natural interpolation by the combination of these two regularizations.

where H is the Heaviside function and δ is the Dirac function. Because f is an SDF, $\|\nabla f(\mathbf{x};\theta)\| = 1$ and can be omitted. In practice, we use a regularized Dirac function δ_ϵ and calculate the integration in a Monte Carlo manner. The minimal surface loss is given by:

$$L_M = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{x} \in \mathcal{R}} \delta_\epsilon(f(\mathbf{x})), \text{ where } \delta_\epsilon(z) = \frac{\epsilon \pi^{-1}}{\epsilon^2 + z^2}.$$

We set the parameter ϵ controlling the sharpness of the peak to $\epsilon = 10$ in practice.

3.4. Differentiable Rendering

Similar to recent differentiable rendering methods, we re-render the input images and minimize the difference. In each iteration, we randomly sample pixels $\mathbf{p} \in \mathcal{I}$ with ground truth color $\hat{\mathbf{c}}_\mathbf{p}$. For each pixel, we first find the surface intersection $\mathbf{x}_\mathbf{p}^{rt} \in \mathcal{S}$ along the view direction $\mathbf{v}_\mathbf{p}$ by sphere tracing. Then we calculate the normal $\mathbf{n}_\mathbf{p}^{rt} = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_\mathbf{p}^{rt}}$ and construct a differentiable intersection $\mathbf{x}_\mathbf{p}^{rt}(\theta)$ by Eq. 2. Now the color of $\mathbf{p}$ can be determined by evaluating the surface light field g with the above inputs and the render loss is given as

$$L_R = \frac{1}{|\mathcal{I}|} \sum_{\mathbf{p} \in \mathcal{I}} \|\mathbf{c}_\mathbf{p} - \hat{\mathbf{c}}_\mathbf{p}\|_1,$$
$$\text{where} \quad \mathbf{c}_\mathbf{p} = g(\mathbf{x}_\mathbf{p}^{rt}(\theta), \mathbf{n}_\mathbf{p}^{rt}, \mathbf{v}_\mathbf{p}).$$

The render loss jointly optimizes the geometry and the appearance. It can recover surface details and refine the interpolated surfaces.

4. Experiments

4.1. Implementation

Point cloud generation. We use the pre-trained Vis-MVSNet [30] to generate the point cloud. Hyper-parameters, including number of source views N_v, proba-

	Chamfer (mm)									PSNR						
	sPSR	SSD	NeRF	UNISURF	NeuS	VolSDF	IDR	MVSDF	RegSDF (Ours)	sPSR	SSD	NeRF	VolSDF	IDR	MVSDF	RegSDF (Ours)
24	0.628	0.761	1.920	1.320	1.370	1.140	1.630	0.826	**0.597**	19.30	19.23	26.24	**26.28**	23.29	25.02	24.78
37	1.335	1.657	1.730	1.360	**1.210**	1.260	1.870	1.763	1.410	15.63	14.65	**25.74**	25.61	21.36	19.47	23.06
40	0.639	0.684	1.920	1.720	0.730	0.810	**0.630**	0.883	0.637	19.51	18.49	**26.79**	26.55	24.39	25.96	23.47
55	0.373	0.406	0.800	0.440	**0.400**	0.490	0.480	0.440	0.428	19.27	19.33	**27.57**	26.76	22.96	24.14	22.21
63	1.061	0.888	3.410	1.350	1.200	1.250	**1.040**	1.105	1.342	20.85	20.26	**31.96**	31.57	23.22	22.16	28.57
65	0.591	0.519	1.390	0.790	0.700	0.700	0.790	0.904	**0.623**	17.72	17.83	**31.50**	**31.50**	23.94	26.89	25.53
69	0.675	0.621	1.510	0.800	0.720	0.720	0.770	0.748	**0.599**	21.65	21.81	**29.58**	29.38	20.34	26.38	21.81
83	0.888	0.946	5.440	1.490	1.010	1.290	1.330	1.259	**0.895**	23.32	23.03	32.78	**33.23**	21.87	25.79	28.89
97	0.862	0.694	2.040	1.370	1.160	1.180	1.160	1.018	**0.919**	18.78	18.78	**28.35**	28.03	22.95	26.22	26.81
105	0.851	0.793	1.100	0.890	0.820	**0.700**	0.760	1.347	1.020	21.65	21.60	32.08	**32.13**	22.71	27.29	27.91
106	0.534	0.481	1.010	**0.590**	0.660	0.660	0.670	0.868	0.600	21.27	21.34	**33.49**	33.16	22.81	27.78	24.71
110	0.811	0.744	2.880	1.470	1.690	1.080	0.900	0.844	**0.594**	18.41	18.45	**31.54**	31.49	21.26	23.82	25.13
114	0.289	0.290	0.910	0.460	0.390	0.420	0.420	0.340	**0.297**	19.56	19.58	**31.00**	30.33	25.35	27.79	26.84
118	0.379	0.353	1.000	0.590	0.490	0.610	0.510	0.467	**0.406**	23.47	23.82	**35.59**	34.90	23.54	28.60	21.67
122	0.413	0.503	0.790	0.620	0.510	0.550	0.530	0.465	**0.389**	24.30	24.45	**35.51**	34.75	27.98	31.49	28.25
Mean	0.688	0.689	1.857	1.017	0.871	0.857	0.899	0.885	**0.717**	20.31	20.18	**30.65**	30.38	23.20	25.92	25.31

Table 1. Quantitative results on DTU [6] dataset. Our method achieves the lowest Chamfer distance among the neural surface reconstruction methods and a balance between geometry accuracy and rendering fidelity. *Bold values are the best among the neural methods only.

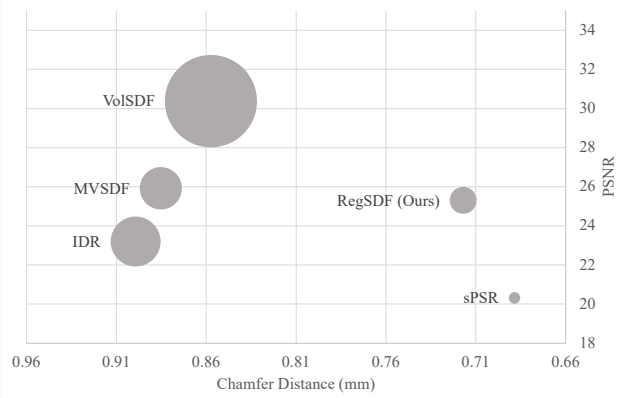

Figure 3. Trade off between geometry accuracy and rendering fidelity. The systems in the upper right corner performs better. The radius of the circles represent the time consumption of each methods.

bility threshold $\mathbf{p}_t$ and number of views for geometric consistency N_f, are set according to different datasets. We use $N_v = 5$, $N_f = 3$, $\mathbf{p}_t = (0.6, 0.7, 0.8)$ for *DTU*, $N_v = 7, N_f = 3, \mathbf{p}_t = (0.6, 0.7, 0.8)$ for *BlendedMVS*, and $N_v = 20$, $N_f = 3$, $\mathbf{p}_t = (0.3, 0.4, 0)$ for *Tanks and Temples*. Normals of point clouds are calculated by PCA for local point clusters by Open3D [33]. Finally point clouds are downsampled to have roughly uniform density, where the target density is selected as the 90% percentile of the inter-point distance of the original point cloud.

Network architecture. Following [29, 31], we use an 8-layer MLP with 512 hidden units and a skip connection in the middle to represent the SDF and a similar 4-layer MLP as the surface light field. Point locations and view directions are enhanced by Positional encoding [16] with 6 and 4 octaves respectively before fed into the networks. The SDF network additionally outputs a 256-channel location descriptor which is then fed into the surface light field.

Initialization. Previous methods usually initialize the SDF to roughly form a sphere. However, this initialization is not appropriate for geometries with arbitrary topologies. Instead, we follow [5] and apply a more general initialization to preserve a uniform distribution $\mathcal{U}(-1, 1)$ from inputs.

Loss weights. The final loss is a weighted sum of the above mentioned losses: $L = \lambda_D L_D + \lambda_B L_B + \lambda_E L_E + \lambda_H L_H + \lambda_M L_M + \lambda_R L_R$. Also, there are also additional weights λ_d and λ_n inside L_D. If not otherwise specified, we empirically set $\lambda_d = \lambda_n = \lambda_D = \lambda_B = \lambda_R = 1$ and $L_E = 0.1$, $L_H = L_M = 0.01$ in our experiments.

Training. The network is trained with a batch size of 8 for 1800 epochs for each *DTU* and *BlendedMVS* scene, or 600 epochs for each *Tanks and Temples* scene. The initial learning rate is set to 10^{-3}. Starting from 1/3 of the whole training process, the learning rate is scaled down by $\sqrt{10}$ every 1/6 of the total epochs. The sample numbers are $|\mathcal{D}| = 32768$, $|\mathcal{R}| = 16384$ and $|\mathcal{I}| = 4096 \times 8$. We use the first 1/6 of training as warm up stage where we disable the differentiable intersection to avoid instabilities.

Mesh extraction. After training, we extract a triangular mesh from the SDF function by Marching Cube [14] algorithm. The space resolution is set to 1024^3 for *Tanks and Temples* dataset and open scenes in *BlendedMVS* dataset. For other scenes, the resolution is set to 512^3.

Evaluation metrics. Chamfer distance is a commonly used metric for mesh evaluation. However, we observe that lower Chamfer distance may not necessarily reflects visually better results as the metric only considers the distance between isolated 3D points. To better measure the similarity between meshes, we additionally introduce the surface normal consistency into the metric. Specifically, we first sample oriented point clouds from both meshes where point cloud normals are inherited from mesh triangles. Similar to pure distance metric, we find the nearest neighbor for each point, but additionally consider the normal apart from the distance. A point will be counted as an inlier if and only if the distance and the angular difference between two normals are smaller than a certain threshold. We calculate percentages

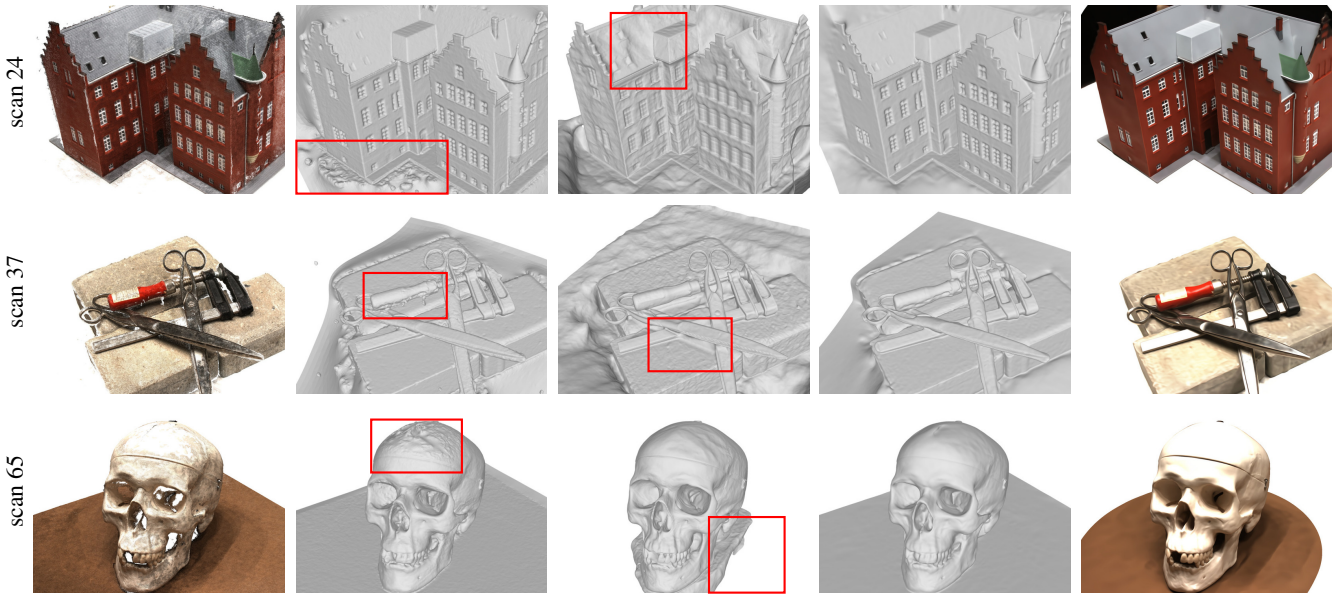

Figure 4. Qualitative results on DTU [6] dataset.

of inliers in both the estimation (accuracy) and the ground truth (completeness), and report the F-score for evaluating *BlendedMVS* and *Tanks and Temples* datasets.

4.2. Baseline Methods

We compare our method with 1) sPSR [8] and SSD [2], which are classical mesh reconstructions from point cloud inputs; 2) NeRF [16], UNISURF [18], NeuS [25] and VolSDF [28], which are neural surface reconstructions by volumetric neural rendering without additional inputs; 3) IDR [29] and MVSDF [31], which are neural surface reconstructions by ray-tracing with additional supervisions. Also, we further compare our method with SIREN [24], which is also a neural surface reconstruction method from point clouds, on *BlendedMVS* and *Tanks and Temples* datasets.

4.3. Benchmark on Dataset in Laboratory

We first benchmark our method on *DTU* dataset [6]. The dataset contains 128 laboratory-captured scenes each with 49 views. All scenes are object-centric and cameras are located in front of the upper bounding sphere of the object. To be consistent with previous methods, we report both surface accuracy as Chamfer distance and render fidelity as PSNR on the same subset of scenes as in [18, 25, 28, 29, 31].

Comparison on extracted meshes are shown in Fig. 4. All methods can reconstruct correct topologies of the scenes. It is shown that NeuS produces bumpy surfaces in the rooftop area. We believe it is caused by the ambiguity of disentangling appearance and geometry, which demonstrates the importance of introducing the geometry supervision for neural surface reconstructions. Also, unrestricted surfaces exist in ground areas away from the object, which may be problematic for the reconstruction of non-object-centric scenes.

Quantitative results are shown in Tab. 1. Our method produce high-quality results similar to classical SSD and sPSR, and is significantly better than recent neural-based methods. Meanwhile, NeRF and VolSDF achieve highest image fidelity scores. In fact, there is a trade off between the geometry accuracy and the rendering fidelity. As shown in Fig. 3, our method keeps a good balance between the geometry and the appearance qualities. In addition, our method takes less time ($\sim$ 3.5 hours) to optimize the scene compared with other neural surface reconstruction methods ($\sim$12 hours for VolSDF).

4.4. Benchmark on Datasets in the Wild

BlendedMVS. Next, we evaluate our method on *BlendedMVS* [27] dataset. We choose to test 4 object-centric scenes and 4 UAV-captured open scenes in the dataset. The distance-only and the normal-aware F-scores described in Sec. 4.1 are used for evaluation. As previous methods have not been evaluated on this dataset, we manually run sPSR, SSD, NeuS and SIREN by provided open-sourced codes.

Qualitative results are shown in Fig. 5. For NeuS, surfaces near the bounding box boundary are missing. Meanwhile, SIREN tends to produce watertight surfaces, which leads to extra surfaces above or below the scene (see *building* and *church*). Quantitative results are shown in Tab. 2, where our method significantly outperforms all other neural-based methods

Tanks and Temples. Lastly, we test our method on the training set of *Tanks and Temples* dataset [10]. There are

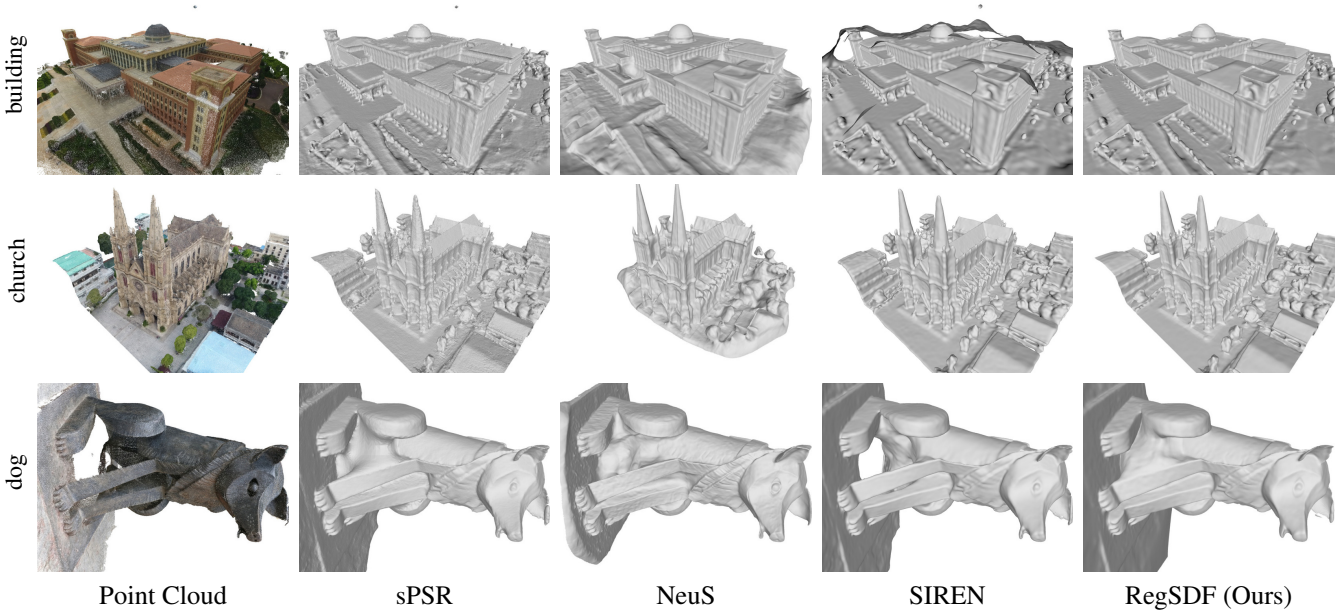

Figure 5. Qualitative results on BlendedMVS [27] dataset.

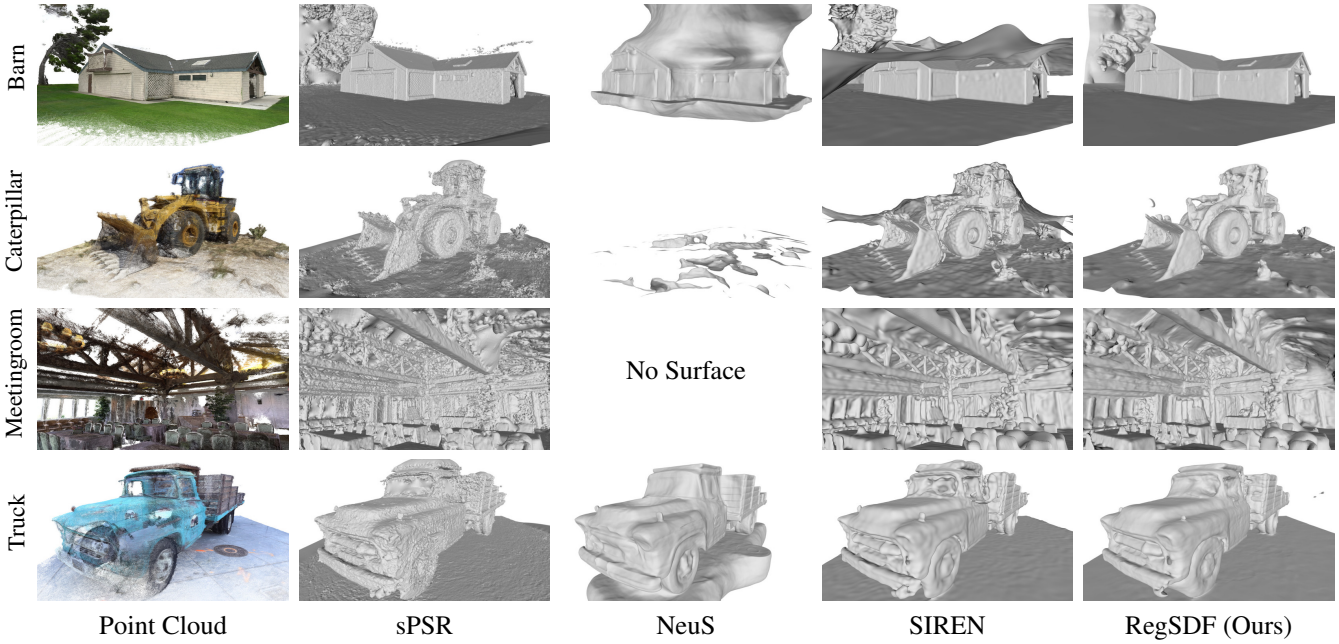

Figure 6. Qualitative results on Tanks and Temples [10] dataset. SPSR [8] produces noisy surface, NeuS [25] is not robust for all the scenarios, and SIREN [24] creates incorrect surface closure.

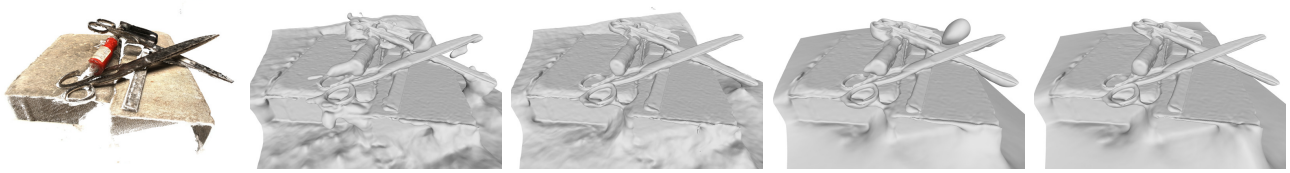

Figure 7. Qualitative results of ablation studies on DTU [6] dataset. The Hessian loss produces smooth surface, and the minimal surface constraint creates compact closure of single-sided point cloud (the scissors).

	Distance					Distance and Normal				
	sPSR	SSD	NeuS	SIREN	RegSDF (Ours)	sPSR	SSD	NeuS	SIREN	RegSDF (Ours)
Caterpillar	23.60%	19.99%	-	20.94%	**21.54%**	13.43%	13.16%	-	15.71%	**16.75%**
Truck	42.64%	37.68%	24.08%	41.93%	**46.32%**	33.25%	31.26%	20.19%	37.70%	**42.03%**
block	82.04%	81.01%	48.91%	60.43%	**77.45%**	71.98%	71.47%	42.03%	52.79%	**68.06%**
building	75.86%	75.80%	46.89%	58.26%	**74.50%**	61.30%	61.78%	37.78%	46.69%	**61.14%**
church	70.09%	68.01%	28.20%	56.48%	**62.25%**	46.66%	46.07%	16.38%	38.97%	**43.12%**
dog	72.02%	72.68%	56.68%	68.85%	**69.36%**	67.88%	68.65%	52.88%	64.53%	**65.38%**
doll	83.48%	82.36%	29.13%	59.74%	**81.45%**	75.71%	74.43%	22.53%	54.20%	**74.41%**
jade	53.00%	52.83%	29.19%	**46.47%**	43.73%	44.58%	44.73%	22.75%	**39.61%**	37.21%
robot	77.65%	68.62%	61.33%	62.13%	**75.04%**	70.16%	61.99%	54.87%	56.06%	**68.31%**
ruin	73.81%	70.06%	32.23%	58.32%	**70.12%**	64.59%	61.56%	27.87%	50.81%	**62.02%**
Mean	73.49%	71.42%	41.57%	58.84%	**69.24%**	62.86%	61.33%	34.64%	50.46%	**59.96%**

Table 2. Quantitative results on BlendedMVS [27] and Tanks and Temples [10] dataset. The right side contains the evaluation where both distance and normal similarity are considered as criteria for inliers.

	No	Hessian L_H			Minimal Surface L_M			Sharpness of L_M		Default
	L_H and L_M	λ_H=1e-1	λ_H=1e-3	No L_H	λ_M=1e-1	λ_M=1e-3	No L_M	ϵ=1	ϵ=100	Config.
DTU	0.841	0.939	**0.685**	0.686	0.894	0.777	0.802	0.801	0.718	0.717
T&T.	23.90%	22.97%	28.72%	28.51%	25.33%	29.17%	29.25%	**29.45%**	29.43%	29.39%

Table 3. Ablation and sensitivity study on DTU and Tanks and Temples datasets. The value reported for DTU is the overall Chamfer distance (the lower the better), and for Tanks and Temples is the inlier percentage with normal criterion (the higher the better). The default configuration achieves overall good results.

totally 7 scenes with 2 indoor scenes which have long been considered challenging for surface reconstruction. We qualitatively compare the 7 scenes and quantitatively evaluate the 2 scenes with ground truth normals (*Caterpillar* and *Truck*). Similar to the evaluation on *BlendedMVS*, we compare our method with sPSR, SSD, NeuS and SIREN using both the distance-only and the normal-aware F-scores. Also, to better adapt NeuS to indoor reconstructions, we modify the SDF geometric initialization by negating the MLP parameters in the last layer.

Qualitative results are shown in Fig. 6. Our method shows the visually best result among all. NeuS fails to reconstruct the indoor scene, and contains inaccurate surfaces for other scenes (*Barn* and *Truck*). SPSR is not robust against noises in input point clouds, which produce noisy and bumpy surfaces in final mesh models. Similar to in BlendedMVS, SIREN produces extra surfaces at scene boundaries, and tends to aggressively close the surface (*Barn* and *Caterpillar*).

Quantitative results are shown in Tab. 2. For *Caterpillar*, sPSR produces noisy surface, but quantitatively outperforms our method in terms of Chamfer distance. However, if normal consistency is considered, our method achieves higher score than sPSR among all, which is consistent with qualitative results in Fig. 6.

4.5. Ablation Study

We conduct ablation studies to discuss the effectiveness of the Hessian regularization and the minimal surface constraint. We test the following four configurations on DTU and Tanks and Temples datasets: 1) remove both losses; 2) remove Hessian regularization; 3) remove minimal surface constraint; 4) default configuration. Qualitative results of scan 37 in DTU are shown in Fig. 7. We find that the Hessian regularization can lead to smooth surface, while the minimal surface constraint can avoid extra surface and produce compact closure for single-sided point clouds. Each losses can improve the quantitative evaluation, and the full setting achieves the lowest Chamfer distance among all.

In addition, we test the system's sensitivity to the loss weights. The quantitative results of ablation and sensitivity study is in Tab. 3. The Hessian loss can be down-weighted for DTU whose point cloud quality is relatively high, but is essential for Tanks and Temples. Overall, the default configuration achieves good results in all the datasets.

5. Conclusion

In this work, we proposed RegSDF, which is a novel neural framework for multi-view surface reconstruction. We introduced the MVS point cloud as additional input, and fit the implicit neural surface to the observed 3D points. For rest of the space, we apply the minimal surface constraint and the Hessian constraint on derivatives to regularize the implicit surface. Our method can naturally interpolate the surface where the input point cloud is missing, and is able to mitigate noises in the input point cloud. Extensive evaluations on *DTU*, *BlendedMVS*, and *Tanks and Temples* datasets have shown that the proposed method achieves both accurate surface reconstruction and high-fidelity image rendering for a variety of scenes.

References

[1] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *BMVC*, 2011. 1

[2] Fatih Calakli and Gabriel Taubin. Ssd: Smooth signed distance surface reconstruction. In *Computer Graphics Forum*, volume 30, pages 1993–2002. Wiley Online Library, 2011. 2, 4, 6, 1

[3] Tony F Chan and Luminita A Vese. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277, 2001. 4

[4] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2009. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 5

[6] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 2, 5, 6, 7, 4

[7] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 2

[8] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 1, 2, 6, 7

[9] Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *CVPR*, 2021. 2

[10] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):78, 2017. 2, 6, 7, 8, 1

[11] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *ICCV*, 2007. 1

[12] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *NeurIPS*, 2019. 2

[13] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, 2020. 2

[14] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 1, 2, 5

[15] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2

[16] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 5, 6

[17] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 1, 2

[18] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021. 1, 2, 4, 6

[19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[20] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 2

[21] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 2

[22] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *CVPR*, 2020. 2

[23] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 1

[24] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. 1, 2, 6, 7

[25] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. 2021. 1, 2, 6, 7

[26] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 1

[27] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *CVPR*, 2020. 2, 6, 7, 8, 1

[28] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv preprint arXiv:2106.12052*, 2021. 1, 2, 6

[29] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 1, 2, 3, 4, 5, 6

[30] Jingyang Zhang, Yao Yao, Shiwei Li, Zixin Luo, and Tian Fang. Visibility-aware multi-view stereo network. In *BMVC*, 2020. 1, 3, 4

[31] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. In *ICCV*, 2021. 1, 2, 3, 4, 5, 6

[32] Hong-Kai Zhao, Tony Chan, Barry Merriman, and Stanley Osher. A variational level set approach to multiphase motion. *Journal of computational physics*, 127(1):179–195, 1996. 3

[33] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 5

Supplemental Materials

1. Additional Implementation Details

Distance of Boundary points. In Sec. 3.2, we calculate nearest neighbor distances $d(\mathbf{x}_B) = |(\mathbf{x}_B - \mathbf{x}_B^{nn}) \cdot \mathbf{n}_B^{nn}|$ for the boundary points $\mathbf{x}_B \in \mathcal{B}$, which may suffer from missing geometry or inaccurate estimation of normals. In practice, we draw k nearest neighbors and take the average of each distances. Also, we consider the angle between $(\mathbf{x}_B - \mathbf{x}_B^{nn})$ and $\mathbf{n}_B^{nn}$, and exclude the neighbors with large angle because they are very likely not the nearest neighbors. These two techniques enhance the robustness against missing points and noisy normal.

Computation of Hessian. In Sec. 3.3, we introduced that the Hessian matrices can be computed by auto-differentiation, which is similar to computing gradient of the SDF. This process is done by the service provided by Pytorch that a scalar function f can be differentiated with respect to each input (x, y, z), and the calculated gradient $(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$ can be added as a part of the computation graph. By further differentiating each entry of the gradient as $(\nabla\frac{\partial f}{\partial x}, \nabla\frac{\partial f}{\partial y}, \nabla\frac{\partial f}{\partial z})$ and concatenate the vectors together, we can get the Hessian matrix $\mathbf{H}f(x, y, z)$.

Parameter Tuning. In Sec. 4.1, we introduced the weights of the losses. This configuration can handle most of the cases when the input point cloud does not contain severe noise or incompleteness. Otherwise, we can tune the weights to get better reconstruction. If the geometry is complex and the render loss is not stable, we scale the gradient back-propagated from the differentiable intersection by 0.1. The three scenes *Church*, *Courthouse* and *Meetingroom* of Tanks and Temples dataset [10] are reconstructed under this setting. If the input point cloud is noisy, we change the weight of the normal data term λ_n to 0.1 in the second half of the training process, and let the render loss correct the surface normal. The four scenes *Barn*, *Caterpillar*, *Ignatius* and *Truck* of Tanks and Temples dataset are reconstructed under this setting.

2. Baselines

In this section, we provide more details on how to conduct experiments for the baseline methods on BlendedMVS and Tanks and Temples datasets.

SPSR and SSD. We test sPSR [8] and SSD [2] by their official implementation[1] and default setting. The resolution is the same as other experiments, which is described in Sec. 4.1.

NeuS. We use an unofficial implementation[2] for NeuS [25]. The system is trained for a fix number of 300k iterations, which takes 20 hours for each scene.

SIREN. We use the official implementation[3] for SIREN [24] The number of training steps is proportional to the number of points in the input point cloud. We add an upper bound of 50k steps. The whole training process takes at most 8 hours for each scene.

3. Details of Evaluation Metrics

In Sec. 4.1, we introduced the metric for evaluating the similarity between meshes, which is used in the comparison on BlendedMVS [27] and Tanks and Temples [10] datasets. Specifically, we first sample points from both meshes with uniform density and preserve their normal. The target density depends on the size of each scene. Then for each point in one point cloud, we find its nearest neighbor in the other one, and check the distance and the normal consistency. The threshold for distances is set to be three times as the downsample density, and the threshold for angular differences is 30°.

We run the same evaluation script, which is self-implemented, on both datasets. For Tanks and Temples, our script is different from the official one. First, we only run ICP between the camera trajectories to align the predicted mesh and the ground truth, so the alignments for all the methods are the same. Second, instead of voxel downsampling, we calculate nearest neighbor distances within the point cloud and discard one of the points in the pair with small distance.

4. Sensitivity to the Quality of Point Clouds

We conduct an experiment to examine the sensitivity to the quality of the input point clouds and the results are shown in Fig. 1. Assuming the point cloud from vis-MVSNet is state-of-the-art, we degrade its accuracy by randomly jittering the point positions within a certain radius.

As can be seen, the sPSR method is sensitive as the noise level goes up, but our method is robust against such noises and provides more consistent results both quantitatively and qualitatively. This again supports the argument that proposed regularizations are robust to random noises.

[1] https://github.com/mkazhdan/PoissonRecon

[2] https://github.com/ventusff/neurecon

[3] https://github.com/vsitzmann/siren

5. Ablation and Sensitivity Study of Regularizations

In Tab. 1, we provide a more comprehensive quantitative ablation and sensitivity study on DTU and Tanks and Temples datasets. Generally, different hyperparameters do have influence on the result. For DTU, we can lower the weight of Hessian or even disable it because the point cloud quality is relatively high. For Tanks and Temples, however, changing the weight of Hessian loss do have notable influence on the results. Apart from the Hessian loss, other configurations are worse than or comparable with the default setting. Overall, the default setting achieves good results in all the datasets (which is shown in the main paper). And we believe the default setting will perform well for other new data.

6. More Results

We provide more results of DTU [6], BlendedMVS [27] and Tanks and Temples [10] datasets in Fig. 2, 3, 4.

In the supplementary video, we show novel view rendering results, and the geometry with respect to training steps.

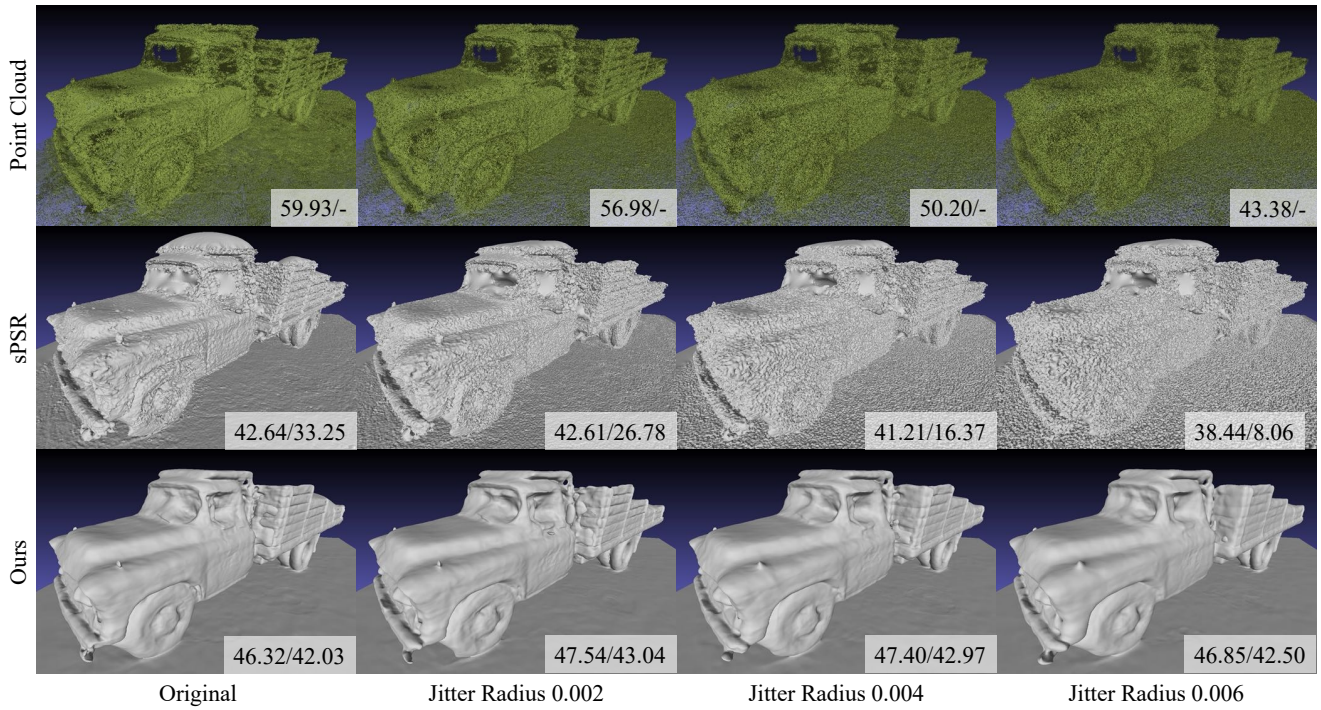

Figure 1. Influence of point cloud quality on mesh accuracy. The two numbers are the evaluation results with and without normal (details are in Sec. 4.1).

	No	Hessian L_H			Minimal Surface L_M			Sharpness of L_M		Default
	L_H and L_M	λ_H=1e-1	λ_H=1e-3	No L_H	λ_M=1e-1	λ_M=1e-3	No L_M	ϵ=1	ϵ=100	Config.
24	0.808	0.777	0.586	**0.573**	0.587	0.631	0.682	0.687	0.595	0.597
37	1.780	1.818	**1.326**	1.516	1.584	1.572	1.600	1.696	1.441	1.410
40	0.721	0.865	0.623	**0.610**	1.132	0.659	0.675	0.671	0.633	0.637
55	0.480	0.483	**0.390**	0.401	0.412	0.487	0.452	0.462	0.410	0.428
63	1.287	1.703	1.143	**1.108**	1.440	1.144	1.201	1.371	1.431	1.342
65	0.615	1.005	**0.611**	0.620	1.575	0.742	0.742	0.673	0.632	0.623
69	0.765	0.771	0.576	**0.551**	0.783	0.674	0.636	0.708	0.566	0.599
83	1.247	1.042	0.904	0.900	**0.890**	1.154	1.229	1.145	0.897	0.895
97	0.986	1.372	**0.848**	0.866	1.206	0.909	0.895	0.948	0.942	0.919
105	1.324	1.196	0.976	**0.940**	1.205	1.172	1.230	1.223	0.970	1.020
106	0.645	0.947	**0.535**	0.564	0.844	0.686	0.726	0.698	0.611	0.600
110	0.748	0.635	0.677	0.572	**0.537**	0.720	0.786	0.673	0.580	0.594
114	0.335	0.360	**0.293**	0.302	0.315	0.297	0.300	0.299	0.302	0.297
118	0.438	0.591	0.398	0.392	0.513	0.373	0.406	**0.370**	0.376	0.406
122	0.443	0.525	0.383	0.379	0.387	0.438	0.465	0.394	**0.379**	0.389
Mean	0.841	0.939	**0.685**	0.686	0.894	0.777	0.802	0.801	0.718	0.717
Caterpillar	14.33%	9.93%	16.67%	16.47%	14.26%	16.92%	16.64%	16.92%	**17.62%**	16.75%
Truck	33.47%	36.01%	40.78%	40.55%	36.40%	41.42%	41.86%	41.99%	41.24%	**42.03%**
Mean	23.90%	22.97%	28.72%	28.51%	25.33%	29.17%	29.25%	**29.45%**	29.43%	29.39%

Table 1. Ablation and sensitivity study on DTU and Tanks and Temples datasets. The value reported for DTU is the overall Chamfer distance (the lower the better), and for Tanks and Temples is the inlier percentage with normal criterion (the higher the better). The default configuration achieves overall good results.

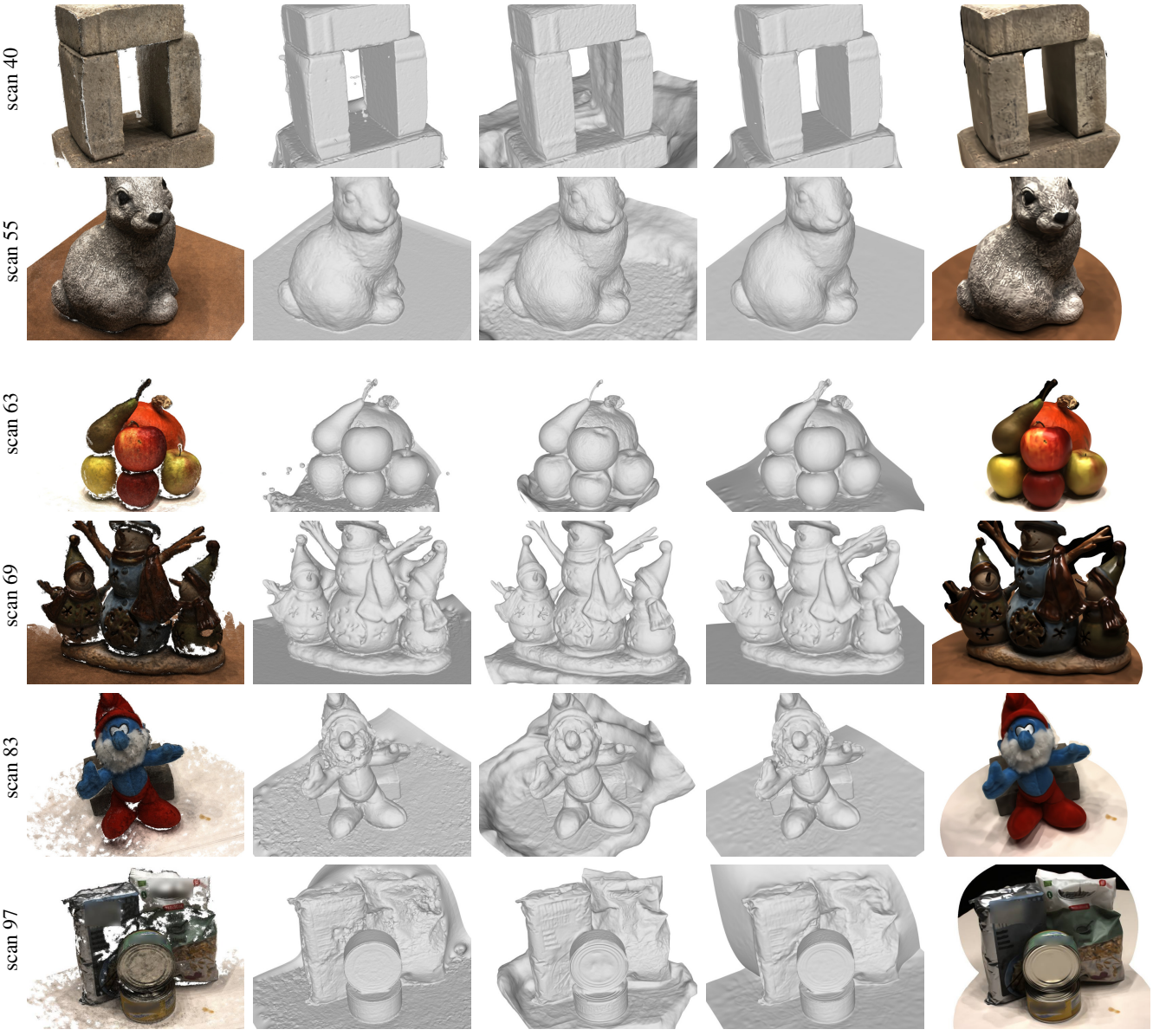

Figure 2. Qualitative results on DTU [6] dataset. Brand names are blurred.

Figure 3. Qualitative results on DTU [6] dataset (cont.).

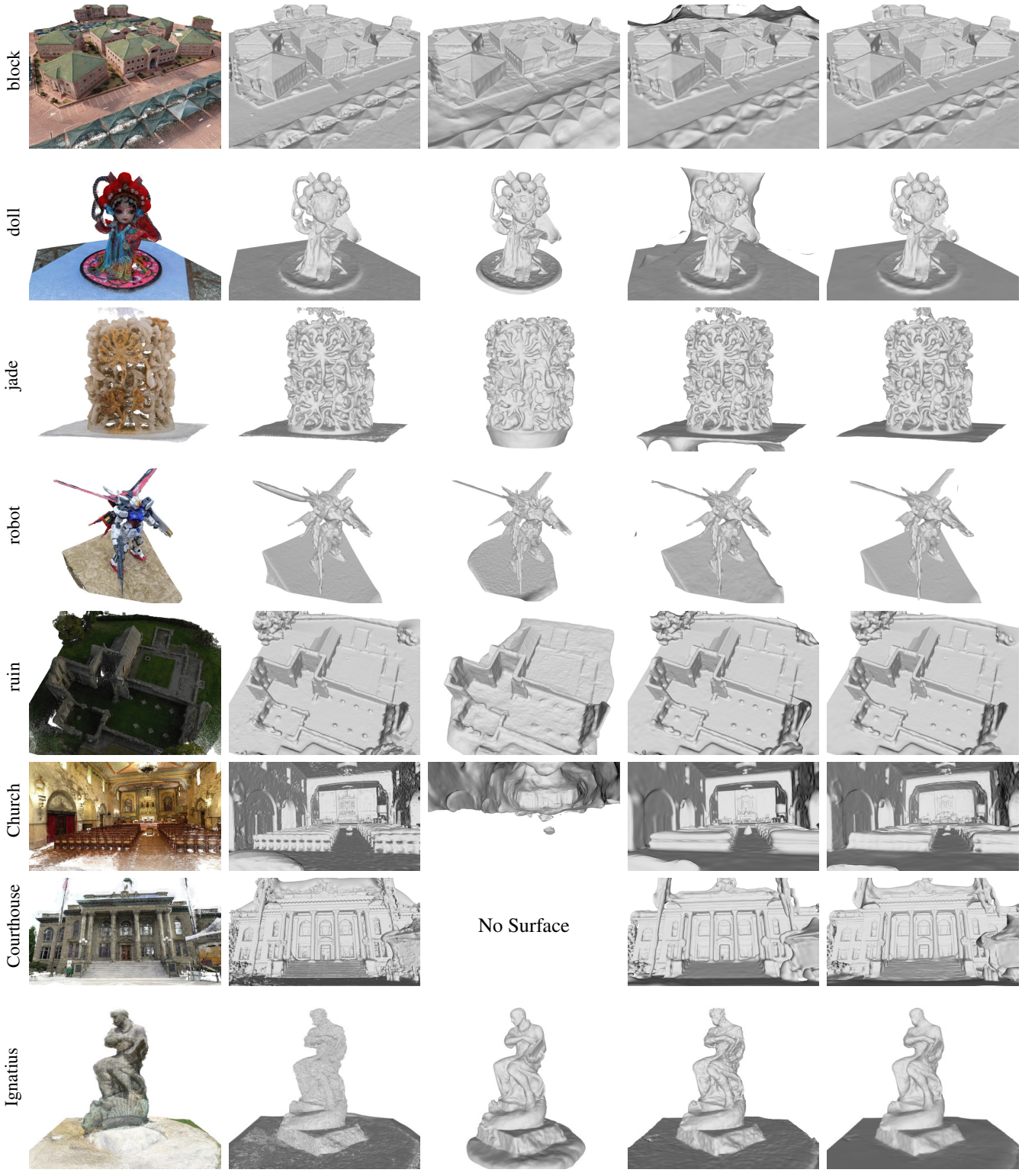

Figure 4. Qualitative results on BlendedMVS [27] and Tanks and Temples [10] dataset.