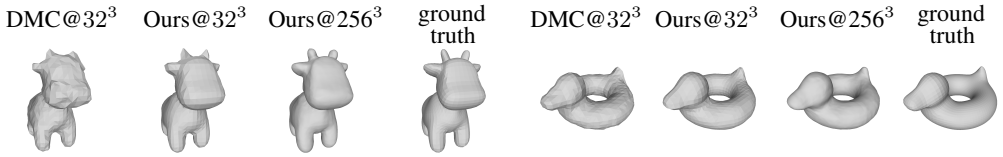
We thank all reviewers for their thoughtful comments. Below, we address their concerns individually.

[R1, closest point] We don't impose any constraints on sampling point $\mathbf{v}$ and our theorem may be proven for any mapping $\mathbf{v}'$ such that $\lim_{\Delta\mathbf{v}\to 0}(\mathbf{v}') = \mathbf{v}$. We chose the closest point one because it seemed the most natural to us.

[R1, effective update] The updated point indeed won't correspond to $\mathbf{v}'$ as described by the theorem because, when back-propagating gradients to the latent code for refinement, we constrain surface change through a learned prior. This is desirable, as we actually want surface deformations to be regularized by our learned shape-space.

[R1, differentiable rasterization] Rasterization is indeed not differentiable. We use the continuous relaxation of [Kato18], approximating the discrete binary operation by a linear function to back-propagate gradients. We will clarify.

[R2, comparison to DMC] Deep Marching Cubes (DMC) is designed to convert point clouds into a surface mesh probability distribution. It can handle topological changes but is limited to low resolution surfaces for the reasons discussed in related work. In the figure below, we compare our approach to DMC. We fit both representations to a toy dataset consisting of two shapes: a genus-0 cow, and a genus-1 rubber duck. We use a latent space of size 2. Our metric is Chamfer l_2 distance evaluated on 5000 samples for unit sphere normalized shapes and shown at the bottom of the figure. As reported in the original paper, we found DMC to be unable to handle grids larger than 32^3 because it has to keep track of all possible mesh topologies defined within the grid. By contrast, our approach is unlimited in resolution and can capture high frequency details, such as the ears of the cow.



CD-$l_2 \cdot 10^2 \downarrow$	1.87	1.84	**1.80**		1.98	1.94	**1.90**	

[R2, different categories] In the main paper, we followed the Pix3D benchmark and reported qualitative results for chairs only. However, we show results for several other ShapeNet categories in Fig. 8 of Supplementary.

[R2, failure cases and limitations]. Failure cases for SVR are presented in Fig. 10 of the Supplementary material. Furthermore, an important limitation is that the training loss of Equation 1 is insufficiently sensitive to topological errors and this is something we are working on.

[R2, car constraints]. The simplest way to restrict changes would be to either limit training data to a specific car type or to increase the regularization weight discussed in section 1.6.4 of the Supplementary material. A more difficult but more powerful approach would be to design constraints directly in terms of the mesh surface. We believe the differentiability of MeshSDF makes this a practical proposition and this will be a topic for future research.

[R2, performance] In section 3 of the Supplementary material, we analyze the computational performance of our differentiable iso-surface extraction pipeline. We did not report performance for other components of the full pipeline (e.g. DeepSDF network, differentiable rasterization) because they are discussed in the original papers.

[R2, exposition] Following DeepSDF, we set $\lambda_{reg} = 10^{-3}$. We will add this to a revised version of the manuscript.

[R3, Marching Cubes discretization]. This is indeed a valid concern, as our differentiation result only holds for samples on the zero-crossing surface. In our experiments, we extract surface samples at 256^3 resolution. This yields an average SDF value of 10^{-5} for the samples. In practice, this is small enough to safely apply our differentiation result.

[R3, combining explicit and implicit losses] Our parameterization enables us to jointly exploit the advantages of explicit and implicit representations: in experiment 4.2, we train our network by supervising for the implicit field while, at inference time, we use explicit surface mesh losses to preform refinement.

[R3, DISN] Unlike DISN, which uses camera information to perform perceptual feature pooling, our baseline (MeshSDF Raw) does not exploit camera information. We speculate that this is the reason for the performance gap.

[R3, Equation 5] We sample points uniformly with respect to surface area when computing point-to-surface distance.

[R4, comparison to differentiable rendering] Indeed, recent advances in differentiable rendering [Liu20] have shown that is possible to render continuous SDFs differentiably by carefully designing a differentiable version of the sphere tracing algorithm. By contrast, we simply use MeshSDF end-to-end differentiability to exploit an *off-the-shelf* differentiable rasterizer and achieve the same result. To highlight the advantages of doing so, we take the generative model in figure above, initialize latent code so that to generate the cow, and then minimize silhouette distance with respect to the duck. In the table below we compare our approach to [Liu20]. Sphere tracing requires to query the network along each camera ray in a sequential fashion, resulting in longer computational time with respect to our approach, which projects surface triangles to image space and then rasterizes them in parallel. Furthermore, our approach requires less function evaluation, as we do not need to sample densely the volume around the field zero-crossing. We refer the reader to Section 3 of the Supplementary section for additional information on how we query our network.

Method	l_2 silhouette distance $\downarrow$	# network queries $\downarrow$	run time [s] $\downarrow$
Liu20 [most efficient settings, 512^2 renders]	0.005973	898k	1.24
MeshSDF [isosurface extraction at 256^3, 512^2 renders]	**0.004625**	**266k**	**0.29**