NeRF, meet differential geometry!

Thibaud Ehret **Roger Marí** **Gabriele Facciolo**
Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli
91190, Gif-sur-Yvette, France

Abstract

Neural radiance fields, or NeRF, represent a breakthrough in the field of novel view synthesis and 3D modeling of complex scenes from multi-view image collections. Numerous recent works have been focusing on making the models more robust, by means of regularization, so as to be able to train with possibly inconsistent and/or very sparse data. In this work, we scratch the surface of how differential geometry can provide regularization tools for robustly training NeRF-like models, which are modified so as to represent continuous and infinitely differentiable functions. In particular, we show how these tools yield a direct mathematical formalism of previously proposed NeRF variants aimed at improving the performance in challenging conditions (i.e. RegNeRF). Based on this, we show how the same formalism can be used to natively encourage the regularity of surfaces (by means of Gaussian and Mean Curvatures) making it possible, for example, to learn surfaces from a very limited number of views.

1 Introduction

Realistic rendering of new views of a 3D scene or a given volume is a long standing problem in computer graphics. The interest in this problem has been rekindled by the growth of augmented and virtual reality. Traditionally, such scenes were estimated from a set of images using tools such as, for example, COLMAP [26] based on the concept of structure from motion.

Recently, Mildenhall et al. [17] have shown that differentiable volume rendering operations can be plugged into a neural network to learn a neural radiance field (NeRF) volumetric representation of a scene encoding its geometry and appearance. Starting from a sparse, but nonetheless large, set of views of the scene, NeRF learns in a self-supervised manner, by maximizing the photo-consistency across the predicted renderings corresponding to the available viewpoints. After convergence, the network is able to render realistic novel views by querying the NeRF function at unseen viewpoints.

This breakthrough led to a very active research topic focusing on pushing back the limits of the initial models. Notably, Martin-Brualla et al. [14] showed that it is possible to learn scenes from unconstrained photos with, for example, moving transient objects or different lightning conditions. Other works deal with dynamic or deformable scenes [21, 31, 19, 20, 12], complex illumination models [28, 2, 36, 13] or very few training views [34, 9, 11]. In other words, the goal is to make NeRF more robust to be able to train reliably even in the most adverse conditions. For example, imposing regularity constraints on the scene seems like a promising way to reduce reliance on large datasets [18].

In this work, we introduce NeRF to differential geometry. We aim to show that differential geometry can provide tools that, straight out of the box, can help make NeRF more robust. The advantages are twofold: first, differential geometry is mathematically formalized and its literature is vast with many suitable tools already available and, second, neural representations are perfectly adapted to represent continuous infinitely differentiable volumetric functions in which the the differential geometry operators are naturally defined.

Preprint. Under review.

To this aim, we present two applications of differential geometry applied to NeRF and its extensions. The first application, in Section 3, is linked to the problem of training a NeRF model when few images (for example only three) are available. For that, we show that the newly proposed RegNeRF [18] can be expressed in terms of differentiable calculus and we show that the new expression leads to better regularization and generalization, and thus better performance. The second application, in Section 4, concerns surface regularization of volumetric renderings. We show that by adding a regularization on the curvature of the surface estimated by VolSDF [33], it is possible to estimate surfaces that are less noisy and therefore more accurate to the target surface.

2 Related Work

2.1 Fundamentals of Neural Radiance Fields

NeRF was originally introduced as a continuous volumetric function $\mathcal{F}$, learned by a multi-layer perceptron (MLP), to model the geometry and appearance of a 3D scene [17, 30]. Given a 3D point $\mathbf{x} \in \mathbb{R}^3$ of the scene and a viewing direction $\mathbf{v} \in \mathbb{R}^2$, NeRF predicts an associated RGB color $\mathbf{c} \in [0,1]^3$ and a scalar volume density $\sigma \in [0, \infty)$, i.e.

$$\mathcal{F} : (\mathbf{x}, \mathbf{v}) \mapsto (\mathbf{c}, \sigma). \tag{1}$$

The value of σ defines the geometry of the scene and is learned exclusively from the spatial coordinates $\mathbf{x}$, while the value of $\mathbf{c}$ is also dependent on the viewing direction $\mathbf{d}$, which allows to recreate non-Lambertian surface reflectance.

NeRF models are trained based on a classic differentiable volume rendering operation [15], which establishes the resulting color of any ray passing through the scene volume and projected onto a camera system. Each ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{v}$ with $t \in \mathbb{R}^+$, defined by a point of origin $\mathbf{o}$ and a direction vector $\mathbf{v}$, is discretized into N 3D points $\{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{v}$ with t_i sampled between the minimum and maximum depth. The sampling depends on the method. The rendered color $\mathbf{c}(\mathbf{r})$ of a ray $\mathbf{r}$ is obtained as

$$\mathbf{c}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i, \quad \text{where} \quad \alpha_i = 1 - \exp(-\sigma_i(t_{i+1} - t_i)) \quad \text{and} \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{2}$$

In (2), $\mathbf{c}_i$ and σ_i correspond to the color and volume density output by the MLP at the i-th point of the ray, i.e. $\mathcal{F}(\mathbf{x}_i, \mathbf{v})$ as per (1). The transmittance T_i and opacity α_i are two factors that weight the contribution of the i-th point to the rendered color according to the geometry described by σ_i and $\sigma_j : j < i$, to handle occlusions.

Using the transmittance T_i and opacity α_i, the observed depth $d(\mathbf{r})$ in the direction of a ray $\mathbf{r}$ can be rendered in a similar manner to (2) [6, 23] as

$$d(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i t_i. \tag{3}$$

Following this volume rendering logic, the NeRF function $\mathcal{F}$ is optimized by minimizing the squared error between the rendered color and the real colors of a batch of rays $\mathcal{R}$ that project onto a set of training views of the scene taken from different viewpoints:

$$L_{RGB} = \sum_{\mathbf{r} \in \mathcal{R}} \|\mathbf{c}(\mathbf{r}) - \mathbf{c}_{\text{GT}}(\mathbf{r})\|_2^2, \tag{4}$$

where $\mathbf{c}_{\text{GT}}(\mathbf{r})$ is the observed color of the pixel intersected by the ray $\mathbf{r}$, and $\mathbf{c}(\mathbf{r})$ is the NeRF prediction (2). The rays in each batch $\mathcal{R}$ are chosen randomly from the available views, encouraging gradient flow where rays casted from different viewpoints intersect with consistent scene content.

mip-NeRF: The original NeRF approach casts a single ray per pixel [17]. When the training images observe the scene at different resolutions, this strategy can lead to blurred or aliased renderings. To prevent such situation, the mip-NeRF formulation [1] can be adopted, which casts a cone per pixel instead. As a result, mip-NeRF is queried in terms of conical frustums and not discrete points, yielding a continuous and natively multiscale representation of regions of the volume space.

2.2 Regularization in Few-shot Neural Rendering

The original NeRF methodology was demonstrated using 20 to 62 views for real world static scenes, and 100 views or more for synthetic static scenes [17]. In the absence of large datasets, the MLP usually overfits to each training image when only a few are available, resulting in unrealistic interpolation for novel view synthesis and poor geometry estimates.

A number of NeRF variants have been recently proposed to address few-shot neural rendering. The use of regularization techniques is common in these variants, to achieve smoother results in unobserved areas of the scene volume or radiometrically inconsistent observations [11].

Implicit/indirect regularization methods rely on geometry and appearance priors learned by pre-trained models. PixelNeRF [34] introduced a framework that can be trained across multiple scenes, thus acquiring the ability to generalize to unseen environments. The MLP learns generic operations while keeping the output conditioned to scene-specific content thanks to an additional input feature vector, extracted by a pre-trained convolutional neural network (CNN). Similarly, DietNeRF [9] complements the NeRF loss (4) with a secondary term that encourages similarity between pre-trained CNN high-level features in renderings of known and unknown viewpoints. Other approaches like π-GAN [4], Pix2NeRF [3] or LOLNeRF[22] combine NeRF with generative models: latent codes are mapped to an instance of a radiance field of a certain pre-learned category (e.g. faces, cars), thus providing a reasonable 3D model regardless of the number of available of views.

Explicit/direct regularization methods can be divided into externally supervised and self-supervised. Self-supervised variants incorporate constraints to enforce smoothness between neighboring samples in space, such as RegNeRF [18] (see Section 3). InfoNeRF [11] is another example that prevents inconsistencies due to insufficient viewpoints by minimizing a ray entropy model and the KL-divergence between the normalized ray density obtained from neighbor viewpoints. In contrast, externally supervised regularization methods usually penalize differences with respect to predefined geometric cues. Depth-supervised NeRF [6] encourages the rendered depth (3) to be consistent with a sparse set of 3D surface points obtained by structure from motion. A similar strategy is used in [13], based on a set of 3D points refined by bundle adjustment; or [23], where a sparse point cloud is converted into dense depth priors by means of a depth completion network.

3 Differential geometry to formalize (and improve) previous methods

RegNeRF definition. In order to improve the robustness of the models when training with few data, Niemeyer et al. [18] proposed RegNeRF, which uses an additional term in the loss function to regularize the predicted depth map. Indeed, a classic hypothesis in depth and disparity estimation is that the target is smooth [25, 8]. This work also proposed a regularization of the appearance of a predicted patch using a normalizing flow network trained to estimate the likelihood of a predicted patch compared to normal patches from the JFT-300M dataset [29]. This part is however not studied here.

Consider the depth map $\tilde{d}$ produced by the NeRF model from a given viewpoint (using Eq. (3)). Let (x, y) be the coordinate of a given pixel of this depth map. Depth regularization can be obtained by enforcing that all neighboring pixels $(x + i, y + j)$ for $i \in \{-1, 1\}$ and $j \in \{-1, 1\}$ have the same depth as the central pixel (x, y). Based on this, the additional loss term proposed by Niemeyer et al. [18] can be expressed as

$$L_{depth} = \sum_{(x,y)} \sum_{(i,j)\in\{-1,1\}^2} (\tilde{d}(x + i, y + i) - \tilde{d}(x, y))^2. \tag{5}$$

In practice, this regularization is not done on the entire depth maps but rather by sampling patches. Note that this loss is computed not only for all patches corresponding to a view in the training dataset, but also for rendered patches whose observation is not available. Indeed, all views should verify this depth regularity property, not only those in the training data. This means that training the RegNeRF model requires the sampling of rays corresponding to unseen views.

Formal definition of RegNeRF. The loss proposed by Niemeyer et al. [18] in Eq. (5) can be seen as a regularization term

$$L_{depth} = \sum_{(x,y)} \|\nabla \tilde{d}(x, y)\|^2 \tag{6}$$

on the depth map computed using the finite difference formula. Regularization penalizing the squared L_2 norm of the gradients has the effect of removing high gradients in the depth map and as such making the resulting depth smooth as aimed. In addition, it does not penalize slanted surfaces (since they have null Laplacian) as it would happen in the case of using a total variation regularization [24].

We show now that the approximated regularization term computed using finite difference on patches can be directly computed using differential calculus and exploiting the automatic differentiation of neural networks. First, by changing the ReLU activation function to a Softplus activation function, we can transform the MLP used in NeRF into a continuous and infinitely differentiable function. This allows to train directly using the gradient of the model, or even higher order operators as it will be shown in Section 4, since the operator is defined everywhere.

Note that in Eq. (6) the depth $\tilde{d}$ map is a projected image parametrized by the image coordinates (x, y), while the depth d in Eq. (3) is defined in the space of rays, which are defined by an optical center and viewing direction $(\mathbf{o}, \mathbf{v})$. Let $C : \mathbb{R}^2 \to \mathbb{R}^3$ be the transformation that converts the image coordinates into the equivalent ray so that $\tilde{d}(x, y) = d(\mathbf{o}, C(x, y))$, then the corresponding gradients are $\nabla_{(x,y)}\tilde{d}(x, y) = \nabla d(\mathbf{o}, \mathbf{v})\mathbf{J}_C(x, y)$, where $\mathbf{v} = C(x, y)$ and $\mathbf{J}_C$ is the Jacobian matrix of C.

In order to work directly in the space of rays when defining the differential of d and speed up the computation, we drop the contribution of $\nabla d(\mathbf{o}, \mathbf{v})$ in the $\mathbf{v}$ direction. Refer to the supplementary material for more details on why this approximation does not affect the final result.

Under the above hypothesis, assume that $\tilde{d}(x, y)$ correspond to the projection of the volume density onto the plane with normal $\mathbf{v}$ and going through $\mathbf{o}$. This means that $\nabla\tilde{d}(x, y)$ corresponds to the component orthogonal to $\mathbf{v}$ of $\nabla_{\mathbf{o}} d(\mathbf{o}, \mathbf{v})$ with $\nabla_{\mathbf{o}}$ corresponding to the gradient computed with respect to $\mathbf{o}$. This means that L_{depth} can be expressed as

$$L_{depth} = \sum_{(\mathbf{o},\mathbf{v})\in\mathcal{R}} \|\nabla_{\mathbf{o}} d(\mathbf{o}, \mathbf{v}) - (\nabla_{\mathbf{o}} d(\mathbf{o}, \mathbf{v}) \cdot \mathbf{v})\, \mathbf{v}\|^2. \tag{7}$$

We shall point out that this expression of the loss function does not require patches and therefore can be applied directly with a single ray sampling like it is traditionally done with NeRF training. In practice, we also add a differentiable clipping to the depth loss to preserve sharp edges that could otherwise be over-smoothed. The original RegNerf depth loss did not require such clipping because the finite difference already acts as a smoothing of the gradient of the depth map.

Experimental results. We test the impact of the proposed differential regularization on the task of scene estimation using only three input views from the LLFF dataset [16]. This is the most extreme test case and, as such, it is highly reliant on the quality of the regularization to avoid catastrophic collapse as shown by Niemeyer et al. [18] for mip-NeRF [1]. In order to compare the proposed formalization of RegNeRF [18] to its original version, we modified the code of the authors and replace their depth loss by the one derived in this section.

In Table 1, we compare the results of the original RegNeRF (using the models trained by the authors) with the modified loss proposed in this Section, referred as *diff RegNeRF*. Since the code released by the authors does not contain the additional appearance loss, we added a third comparison that corresponds to RegNeRF without the additional appearance loss (*i.e*. the models that can be trained using the code from the authors). The proposed diff RegNeRF not only improves by 1dB the PSNR of reconstructed unseen views compared to the equivalent RegNeRF version, it also outperforms the RegNeRF with the appearance loss by almost 0.5dB. This is also the case for other metrics such as SSIM and LPIPS.

Visual results on two examples of the LLFF dataset are shown in Figs. 1 and 2. In both cases, we compare the proposed version with the models trained by Niemeyer et al. [18]. Fig. 1 shows a first example where our formalization outperforms RegNerf across all the evaluation metrics. The proposed method is able to learn a better geometry of the image, leading to a more complete reconstruction of the triceratops skull (see the horn on the right), but also of the rest of the scene, such as the sign panel in the foreground or the handrails in the background.

Fig. 2 shows a second example where the PSNR is worse with respect to the original method. However, the depth map estimated by our formalism is still much smoother without losing details. In addition, just like with the triceratops, we can see that some details are also better reconstructed, like

Table 1: Quantitative comparison of novel view synthesis for different implementations of RegNeRF on the LLFF dataset. All models were trained using only three input views. *RegNeRF (w/o app. reg.)* corresponds to the original RegNeRF without appearance regularization, while the proposed modification is *diff RegNeRF*. The results using RegNerf with appearance regularization are also provided for reference. The proposed mathematical formalization of RegNeRF almost systematically achieves the best results across all metrics without requiring any additional appearance regularization. The LPIPS metric is computed using the official implementation provided by Zhang et al. [35]. Best results are shown in bold.

		fern	flower	fortress	horns	leaves	orchids	room	trex	avg.
PSNR	RegNeRF (w/o app. reg.)	19.85	19.64	22.28	13.05	16.46	15.43	20.62	20.37	18.46
	diff RegNeRF	**20.15**	**20.27**	**23.68**	**17.80**	**16.88**	15.50	21.04	**20.58**	**19.49**
	RegNeRF [18]	19.87	19.93	23.32	15.65	16.60	**15.56**	**21.53**	20.17	19.08
SSIM	RegNeRF (w/o app. reg.)	0.694	0.677	0.706	0.486	0.599	0.483	0.843	0.774	0.658
	diff RegNeRF	**0.703**	**0.707**	**0.761**	**0.680**	**0.645**	0.487	**0.864**	**0.791**	**0.705**
	RegNeRF [18]	0.697	0.688	0.743	0.610	0.613	**0.502**	0.861	0.766	0.685
LPIPS	RegNeRF (w/o app. reg.)	0.323	0.243	0.294	0.341	0.229	0.259	0.204	0.197	0.261
	diff RegNeRF	**0.290**	**0.223**	**0.219**	**0.293**	**0.186**	**0.247**	**0.171**	**0.166**	**0.224**
	RegNeRF [18]	0.304	0.234	0.258	0.356	0.222	0.251	0.185	0.197	0.251

the audio conferencing system in the middle of the table. Note how, in both examples, the depth map is better regularized than the original RegNeRF even though we only used the input views at training time while [18] also regularizes unseen views and requires defining a patch size and patch-based dataloaders. This shows that the proposed formalism yields a much better generalization. Moreover, our implementation is only slightly slower ($1.2\times$ the computation time of the original implementation) and slightly more costly in terms of memory, because of the need to estimate additional gradients. All experiments in this section were computed using a loss weight of $1e^{-7}$ with a clipping value of 20.0 on an NVIDIA GeForce 3090ti GPU.

4 Differential geometry for surface regularization using mean and Gaussian curvatures

Learning regularized surfaces with NeRF-like models. The second example we propose to look into is the regularization of 3D surfaces. Indeed, another trend with NeRF-like models is to directly learn a surface model instead of a density function as shown in Section 2.1. In particular, IDR [32] and volSDF [33] both learn the surface thanks to a signed distance function (SDF). This SDF can then be used directly or as a guide to sample points as done in NeRF to learn the surface. Since this SDF function can be seen as an implicit function F defining the surface $\mathcal{S}$ as the set of points $\left\{\mathbf{x} \in \mathbb{R}^3 \mid F(x) = 0\right\}$, it is possible to compute other differential quantities related to surface regularity, such as the curvature. We propose in this section to look at the Gaussian curvature γ_{gauss} and the mean curvature γ_{mean}, since they both have an analytical expression that can be easily implemented using the existing deep learning frameworks.

These curvatures are respectively defined as

$$\gamma_{mean} = -\mathbf{div}\left(\frac{\nabla F}{\|\nabla F\|}\right) \qquad \text{and} \qquad \gamma_{gauss} = \frac{\nabla F \times H^*(F) \times \nabla F^t}{\|\nabla F\|^4}, \tag{8}$$

where H^* is the adjoint of the Hessian of F. Derivation details of these two curvatures can be found in [7]. Using (8), we can define a regularization loss as

$$L_{curv}(\kappa_{curv}) = \mathbb{E}_{x \in \mathcal{S}}\left[\min\left(|\gamma(x)|, \kappa_{curv}\right)\right], \tag{9}$$

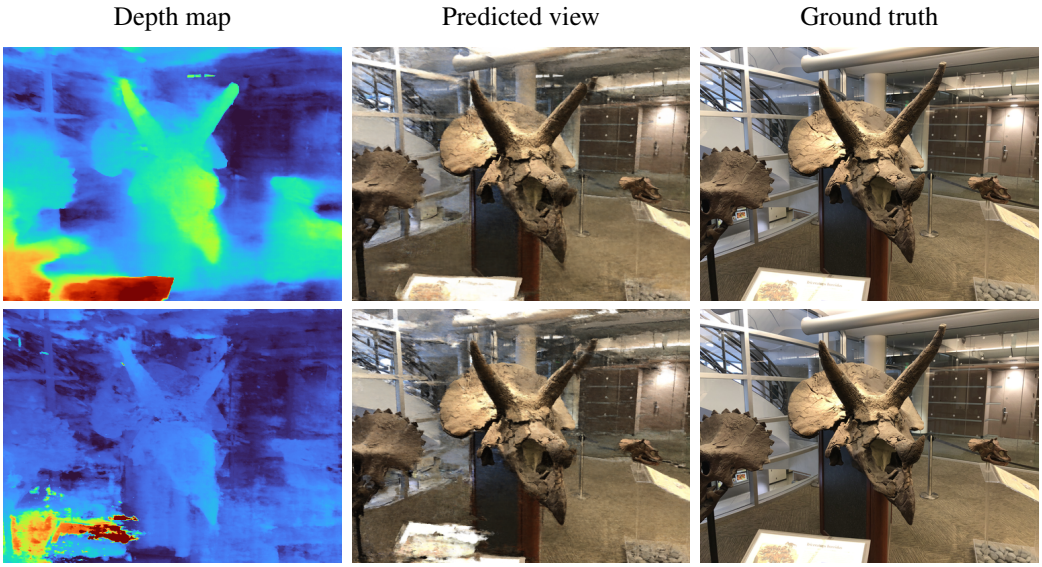

Figure 1: Visual example of novel view synthesis for the *horns* sequence of the LLFF dataset after training with three views. The depth map produced by our definition of the loss (top) is more regular than the one produced by the original RegNeRF with appearance regularization (bottom). It also recovers more details both in the foreground (see the sign panel on the left or the triceratops' left horn) but also in the background (see the glass panels and the handrails). Results best seen zoomed.

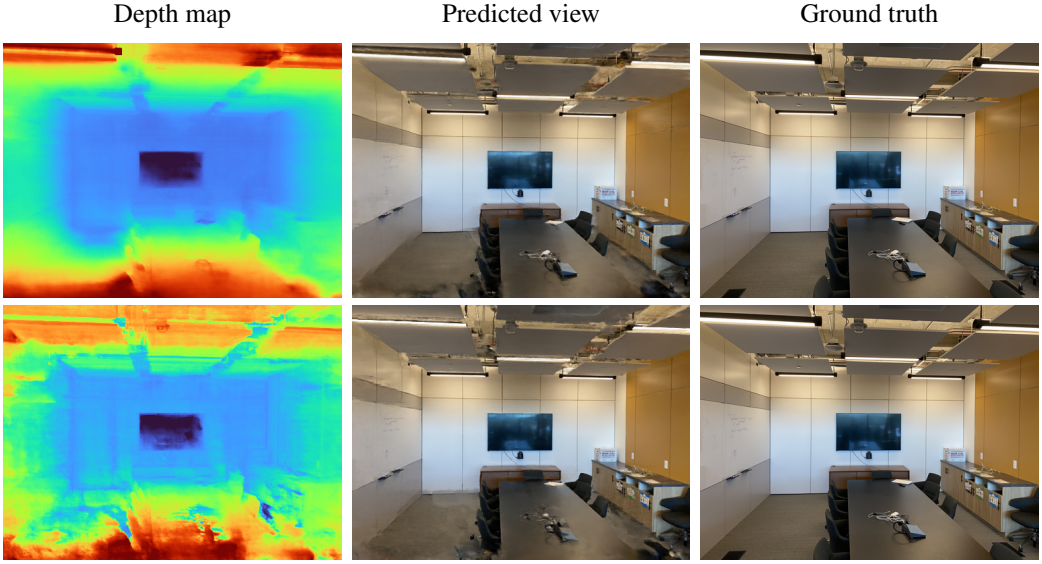

Figure 2: Visual example of novel view synthesis for the *room* sequence of the LLFF dataset after training with three views. Even for a scene where the proposed redefinition of the loss (top) produces worse PSNR, the depth map corresponding to the novel view is more regular than the one of the original RegNeRF (bottom). Some details, such as the audio conference system, are also better reconstructed. Results best seen zoomed.

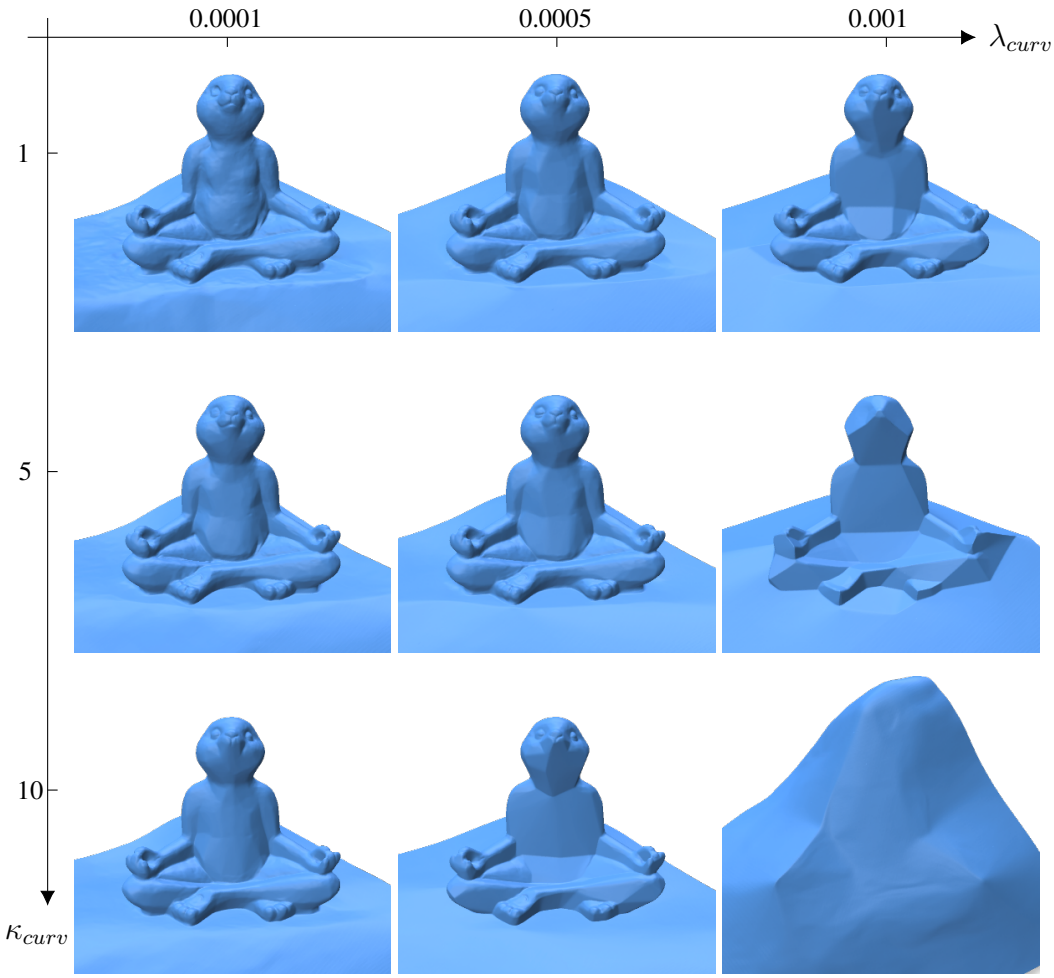

Figure 3: Study of the impact of the two parameters controlling the strength of the regularization, i.e. the weight of regularization loss λ_{curv} and the clipping value κ_{curv}, on the scene 110 of the DTU dataset. While intermediate parameters, such as $\lambda_{curv} = 0.0005$ and $\kappa_{curv} = 5$, provide a good balance between detail preservation and surface smoothness, strong regularization can produce surprising surfaces such as the Cubist-like representation given by $\lambda_{curv} = 0.001$ and $\kappa_{curv} = 5$. These results were computed using the Gaussian curvature. Results best seen zoomed.

where γ can be either γ_{mean} or γ_{gauss}, depending on the preferred behavior, and κ_{curv} is the clipping value. The final loss to train such a regularized volSDF model then becomes

$$L = L_{RGB} + \lambda_{SDF} L_{SDF} + \lambda_{curv} L_{curv}(\kappa_{curv}) \quad \text{with} \quad L_{SDF} = \mathbb{E}_{x \in \mathbb{R}^3} \left[(\|\nabla F(x)\| - 1)^2 \right]. \quad (10)$$

As in [33], the L_{SDF} term enforces the Eikonal constraint on the implicit function F, thus learning a signed distance function. In our experiments, the parameter λ_{SDF} is fixed and set to 0.1.

Practical details. In our experiments, we build on the code of volSDF [33] and modify it by adding the curvature regularization loss (9). Since volSDF samples many points (in the order of 128) close to the surface, it is important to regularize only those directly on the surface (like mentioned in (9)) to keep the additional computational cost small. Doing this means that the implementation has an equivalent memory footprint and computation time as the original code. Otherwise, trying to regularize all sampled points indistinctly would be very expensive both in terms of time and memory required. All results presented in this section were computed using an NVIDIA V100 GPU. At training time, we observed instabilities in some of the experiments. The convergence would sometimes fail when using the mean curvature regularization because of the appearance of NaNs, requiring to restart the training from the last checkpoint before failure to finish the training. This

problem was never encountered when using the Gaussian curvature regularization. We do not think that this unstable behavior is is related to the definition of the mean curvature but to the current implementation instead.

Parameters study. The mean and Gaussian curvature can also be expressed in terms of the minimum curvature γ_{min} and maximum curvature γ_{max} of the surface at a given point

$$\gamma_{mean} = \frac{\gamma_{min} + \gamma_{max}}{2} \qquad \text{and} \qquad \gamma_{gauss} = \gamma_{min}\gamma_{max}. \tag{11}$$

While this is not a practical definition of the curvature since it doesn't allow for straightforward computation, it helps understanding the impact of a regularization based on either of them. Minimizing the mean curvature leads to surface smoothing [5], while minimizing the Gaussian curvature forces the minimum curvature to be zero, resulting in flat surfaces with sharp straight edges. In Fig. 3, we show the impact of the two parameters controlling the loss, namely the weight of the loss λ_{curv} and the value of the clipping κ_{curv}, on a specific example of the DTU dataset [10] using Gaussian curvature. A very small weight and a strong clipping ($\lambda_{curv} = 0.0001$ and $\kappa_{curv} = 1$) leads to barely no regularization as expected. On the contrary, a large weight with little clipping ($\lambda_{curv} = 0.001$ and $\kappa_{curv} = 10$) learns a sort of envelope of the surface. The most interesting results, however, are obtained with intermediate parameters. For example, $\lambda_{curv} = 0.0005$ and $\kappa_{curv} = 5$ produces a surface that is both strongly regularized while keeping most of the details. Another interesting result is that of $\lambda_{curv} = 0.001$ and $\kappa_{curv} = 5$, which leads to a Cubist-style surface, with little details and exaggerated edges. Such models might be interesting to learn a surface that can be represented by a mesh with few triangles or for deriving a piece-wise developable surface [27]. Note also how the regularization has a tendency to fill small gaps (for example the mouth of the bunny is completely regularized with parameters $\lambda_{curv} = 0.0001$ and $\kappa_{curv} = 10$). This shows the importance of clipping to preserve small details and the limits of such type of regularization.

Experimental results. The proposed regularization can have multiple practical applications. The first, and most straight forward, is to improve the quality of the surface models learned by using an *a priori* on the shape of the object or scene that is learned. For example, when trying to reconstruct buildings, it is expected that the surface would be very flat and with sharp straight edges. This is then a perfect application for the Gaussian curvature regularization as shown in Fig. 3. In Fig. 4, we provide two examples that are improved by using a curvature regularization. In both cases, the learned model is less noisy when using the regularization than otherwise. Another example of application is, as in Section 3, to learn a model with few views (for example, only three views). Indeed, as shown in Fig. 5, few-shot scenarios can cause volSDF [33] to produce collapsed results with holes, while the equivalent model learned using a curvature regularization is able to learn a closed and more accurate model.

5 Conclusions

We demonstrate with two introductory, yet key, experiments the usefulness of differential geometry to improve the robustness of NeRF-like models. The vast literature on differential geometry opens up many exciting opportunities, from different definitions of curvature appropriate for different types of surfaces to mathematical formalism of previously proposed methods, which we hope pushes the limits of neural rendering even further.

Modern deep learning frameworks already provide the necessary tools to implement these operators, which facilitates their use in practice. However, this is still subject to certain limitations. Higher-order operators can be costly both in memory and in computation time so a careful implementation and avoidance of bruteforce regularization is required. The latter can be done, for example, by restricting regularization costs to a carefully chosen subset of sampled points. We observed during our experiments that computation and memory requirements are not a bottleneck in that case, making the regularized variants almost as efficient as the original methods we built upon. Additional care must also be taken to avoid instabilities.

The main problem is then to find the most appropriate tool for the problem at hand. Indeed, we have seen that the operator should be chosen differently based on the problem. For example, a Gaussian curvature regularization is perfectly appropriate for flat surfaces with strong edges, such as buildings,

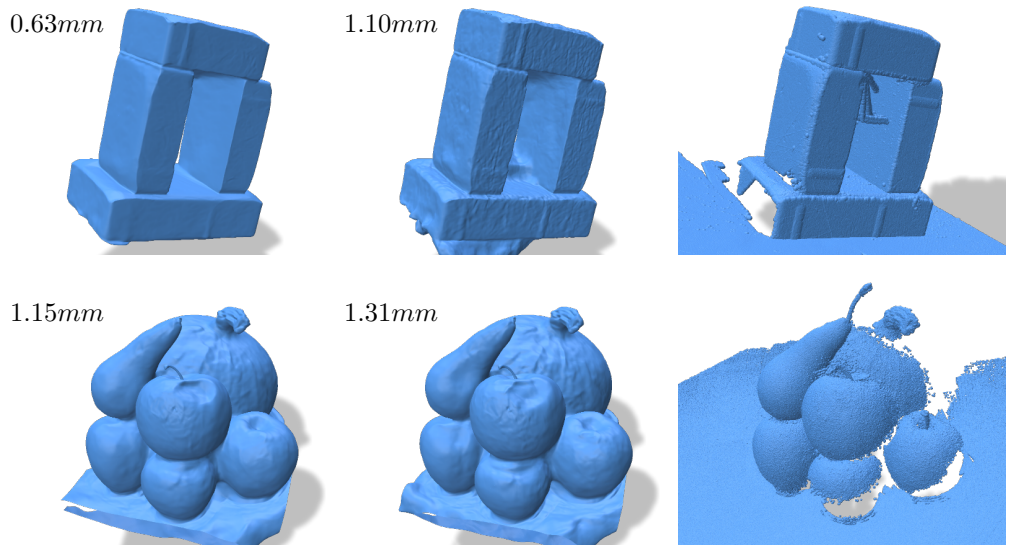

Figure 4: Visual examples of regularized reconstruction on the scene 40 (top) and scene 63 (bottom) of the DTU datset. From left to right: regularized reconstructions, original volSDF results and ground truth. The Chamfer ℓ_1 distance (in mm) is also provided in the top left of each reconstruction (smaller is better). As it can be seen, regularization improves the reconstruction for these two examples with less noise in the learned surface. The example on the top is regularized using Gaussian curvature while the example at the bottom is regularized using mean curvature. Results best seen zoomed.

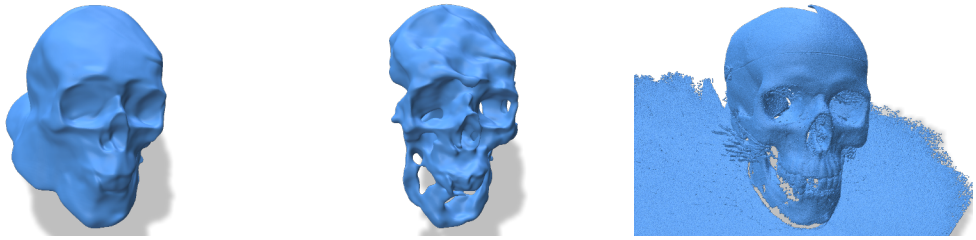

Figure 5: Example of reconstruction when using only three input views of the scene 65 of the DTU dataset using the original volSDF [33] (in the middle), with Gaussian curvature regularization (on the left) and the ground truth (on the right). The regularization allows to reconstruct a more regular surface without holes. This shows that the addition of such a regularization allows for a more robust model even when using few input views. Results best seen zoomed.

but might fill holes, such as the mouth of the bunny, in irregular surfaces. We believe that this offers new and interesting research perspectives.

Lastly, we do not foresee any other negative societal impacts than the ones already covered by NeRF [17] and related methods.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021.

[2] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. NeRD: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.

[3] Shengqu Cai, Anton Obukhov, Dengxin Dai, and Luc Van Gool. Pix2NeRF: Unsupervised conditional π-GAN for single image to neural radiance fields translation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5799–5809, 2021.

[5] Ulrich Clarenz, Udo Diewald, and Martin Rumpf. Anisotropic geometric diffusion in surface processing. In *Proceedings of the IEEE Visualization Conference*, 2000. doi: 10.1109/visual.2000.885721.

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] Ron Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005.

[8] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[9] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5885–5894, 2021.

[10] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[11] Mijeong Kim, Seonguk Seo, and Bohyung Han. InfoNeRF: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

[13] Roger Marí, Gabriele Facciolo, and Thibaud Ehret. Sat-NeRF: Learning multi-view satellite photogrammetry with transient objects and shadow modeling using RPC cameras. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022.

[14] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7219, 2021.

[15] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[16] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[17] 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 *European Conference on Computer Vision*, pages 405–421, 2020.

[18] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[19] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5865–5874, 2021.

[20] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021.

[21] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327, 2021.

[22] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. LOLNeRF: Learn from one look. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[23] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[24] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[25] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[26] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[27] Silvia Sellán, Noam Aigerman, and Alec Jacobson. Developability of heightfields via rank minimization. *ACM Trans. Graph.*, 39(4):109, 2020.

[28] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7495–7504, 2021.

[29] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[30] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. *Computer Graphics Forum*, 39(2):701–727, 2020.

[31] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12959–12970, 2021.

[32] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34, 2021.

[34] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2021.

[35] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[36] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. NeRFactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.