VSGAN: 3D-aware Image Synthesis based on Volume Rendering of Neural Implicit Surfaces

Jiyu Liu, Jundong Zhou, Yongyang Pan, Zhaoyu Zeng
Shanghai Jiao Tong University
520030910342, 520030910341, 520030910354, 520030910344

Abstract

As volume rendering demonstrating advantages in 3D-reconstruction, many works based on volume rendering strive to improve the precision of reconstruction. We develop our work based on two of them and combine them together to achieve better performance. We implemented experiments to illustrate the results of our algorithm on Carla *dataset but get unsatisfied results. Some main reasons lead to the awful results are discussed after demonstration of the results.*

1. Introduction

3D-reconstruction has become a substantial problem in computer vision since it is introduced in [2]. It is widely applied in medical science, virtual reality, special effects, etc. In recent years, the development of deep neural network enhanced 3D-reconstruction problem in many aspects. NeRF [3] introduced neural network to approximate volume density and use the item to improve rendering method. Volsdf [5] replaced the density model with a more reasonable function, improving the performance of reconstruction.

Generative Adversarial Network(GAN) is a widely used algorithm to enhance the robustness of a model. GRAF [4] combined GAN and NeRF together, achieving better performance in both rotation and elevation disturbance.

Our contribution In this paper, we merge GAN and VolSDF together to pursue a powerful representation for generative image synthesis and test the performance of our algorithm on $Carla$ dataset.

2. Methodology

2.1. Density Function

Let the set $\Omega \subset \mathbf{R}^3$ represent the space occupied by some object in $\mathbf{R}^3$. Density $\sigma : \mathbf{R}^3 \to [0, 1]$ denotes the probability that light is occluded at point x. $\mathbf{1}_\Omega(x)$ is the indicator function of $x \in \Omega$. Signed distance is represented as

$$d_\Omega(x) = (-1)^{\mathbf{1}_\Omega(x)} min_{y \in \partial\Omega} ||x - y||$$

. In this work, we use the density function in VolSDF [5], i.e. $\sigma(x) = \alpha\Psi_\beta(-d_\Omega(x))$, where α, β are learnable parameters, and Ψ_{beta} is the Cumulative Distribution Function (CDF) of the Laplace distribution with zero mean and β scale.

2.2. Volume rendering

Consider a ray $\mathbf{x}$ emanating from a camera position $\mathbf{c} \in \mathbb{R}^3$ in direction $\mathbf{v} \in \mathbb{R}^3, \|\mathbf{v}\| = 1$, defined by $\mathbf{x}(t) = \mathbf{c} + t\mathbf{v}, t \geq 0$. In essence, volume rendering is all about approximating the integrated light radiance along the ray reaching the camera. Two important quantities that participate in the computation are the volume's *transperancy T*, the *radiance field L*.

The *transparency* function of the volume along a ray $\mathbf{x}$, denoted T, indicates, for each $t \geq 0$, the probability a light particle succeeds traversing the segment $[\mathbf{c}, \mathbf{x}(t)]$ without bouncing off,

$$T(t) = \exp\big(-\int_0^t \sigma(\mathbf{x}(s))ds\big), \tag{1}$$

and the *opacity* O is the complement probability,

$$O(t) = 1 - T(t), \tag{2}$$

that is monotonic increasing where $O(0) = 0, O(\infty) = 1$. In that way, think O as a CDF, and

$$\tau(t) = \frac{dO}{dt}(t) = \sigma(\mathbf{x}(t))T(t) \tag{3}$$

is the corresponding PDF. Given these, the volume rendering equation can be written as,

$$I(\mathbf{c}, \mathbf{v}) = \int_0^\infty L(\mathbf{x}(t), \mathbf{n}(t), \mathbf{v})\tau(t)dt, \tag{4}$$

where $L(\mathbf{x}, \mathbf{n}, \mathbf{v})$ is the radiance field, namely the amount of light emanating from point $\mathbf{x}$ in direction $\mathbf{v}$; The integral in

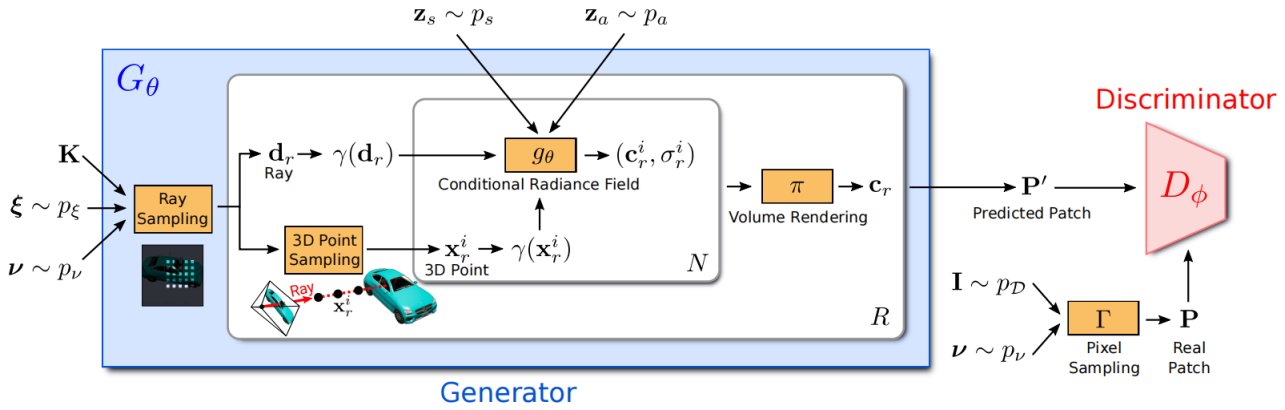

Figure 1. Generative Radiance Fields

equation 4 is approximated using a numerical quadrature at some discrete samples $\mathbf{S} = \{s_i\}_{i=1}^m, 0 = s_1 < s_2 < \cdots < s_m = M$, where M is some large constant:

$$I(\mathbf{c}, \mathbf{v}) \approx \hat{I}_\mathbb{S}(\mathbf{c}, \mathbf{v}) = \sum_{i=1}^{m-1} \tilde{\tau}_i L_i \tag{5}$$

Where $\hat{\tau}_i \approx \tau(s_i)\Delta_s$ is the approximated PDF multiplied by the interval length, $L_i = L(\mathbf{x}(s_i), \mathbf{n}(s_i), \mathbf{v})$ is the sampled radiance field.

2.3. Sampling in VolSDF

Since the PDF τ is typically extremely concentrated near the object's boundary, the choice of the sample points of $\mathbb{S}$ has a crucial effect on the approximation quality. One existing method is to use adaptive sample, e.g., $\mathbb{S}$ computed with inverse CDF. However, this fails due to T depends on the density model σ and is not given explicitly. In Nerf [3], a second, coarse network was trained specifically for the approximation of the opacity O, and was used for inverse sampling. However, the second network's density does not necessarily faithfully represents the first network's density. Here we choose the sampling algorithm based on error bound for the opacity approximation and replace the sampling in Section 2.6 with it.

2.4. Bound on opacity approximation error

For a set of samples $T = t_i{}_{i=1}^n, 0 = t_1 < t_2 < \cdots < t_n = M$, we let $\delta_i = t_{i+1} - t_i, \sigma_i = \sigma(\mathbf{x}(t_i))$. Given some $t \in (0, M]$, assume $t \in [t_k, t_{k+1}]$, and apply the rectangle rule:

$$\int_0^t \sigma * \mathbf{x}(s))ds = \hat{R}(t) + E(t), \hat{R}(t) = \sum_{i=1}^{k-1} \delta_i\sigma_i + (t - t_k)\sigma_k. \tag{6}$$

$E(t)$ denotes the error thus the corresponding approximation of the opacity function 2 is

$$\hat{O}(t) = 1 - \exp\big(-\hat{R}(t)\big). \tag{7}$$

The goal is to derive a uniform bound over $[0, M]$ to the approximation $\hat{O} \approx O$ using the theorems and lemmas proposed in Volsdf [5].

Theorem 2.1 *The derivative of the density σ within a segment $[t_i, t_{i+1}]$ satisfies*

$$\left|\frac{d}{ds}\sigma(\boldsymbol{x}(s))\right| \le \frac{\alpha}{2\beta}\exp\big(-\frac{d_i^*}{\beta}\big), \quad where d_i^* = \min_{s\in[t_i,t_{i+1}]\boldsymbol{y}\notin B_i\cup B_{i+1}} \|\boldsymbol{x}(s) - \boldsymbol{y}\|, \tag{8}$$

and $B_i = \{\boldsymbol{x} | \|\boldsymbol{x} - \boldsymbol{x}(t_i)\| |d_i|\}, d_i = d_\Omega(\boldsymbol{x}(t_i))$.

Theorem 2.2 *For $t \in [0, M]$, the error of the approximated opacity $\hat{O}$ can be bounded as follows:*

$$\left|O(t) - \hat{O}(t)\right| \le \exp\big(-\hat{R}(t)\big)\big(\exp\big(\hat{E}(t)\big) - 1\big) \tag{9}$$

Taking the maximum over all intervals furnishes a bound $B_{T,\beta}$ as a function of T and β.

$$\max_{t\in[0,M]}\left|O(t) - \tilde{O}(t)\right| \le B_{T,\beta} = \max_{k\in[n-1]}\big\{\exp\big(-\hat{R}(t_K)\big)\big(\exp(\hat{E}(t_{k+1})) - 1\big)\big\} \tag{10}$$

Lemma 2.3 *Fix $\beta > 0$. For any $\epsilon > 0$ a sufficient dense sampling T will provide $B_{T,\beta} < \epsilon$.*

Lemma 2.4 *Fix $n > 0$. For any $\epsilon > 0$ a sufficiently large β that satisfies*

$$\beta \ge \frac{\alpha M^2}{4(n-1)\log(1+\epsilon)} \tag{11}$$

will provide $B_{T,\beta} \le \epsilon$.

Figure 2. Original car models.

2.5. Sampling algorithm

Sampling algorithm is developed for computing the samplings $\mathbb{S}$ to be used in equation 5. This is done by first utilizing the bound in equation 10 to find samples T so that $\hat{O}$ provides an ϵ approximation to the true opacity O. Second, we perform inverse CDF sampling with $\hat{O}$.

We start with a uniform sampling $T = T_0$, and use Lemma 2.4 to initially set a $\beta_+ > \beta$ that satisfies $\beta_{T,\beta_+} \leq \epsilon$. Then, we repeatedly upsample τ to reduce β_+ while maintaining $\beta_{T,\beta_+} \leq \epsilon$. β_+ usually converges to β and even in cases it does not, the algorithm provides β_+ for which the opacity approximation still maintains an ϵ error.

We initialize T with uniform sampling $T_0 = t_i{}_{i=1}^{n}$, where $t_K = (k-1)\frac{M}{n-1}, k \in [n]$. Given this sampling we next pick $\beta_+ > \beta$ according to Lemma 2.4 so that the error bound satisfies the required ϵ bound.

In order to reduce β_+ while keep $\beta_{T,\beta_+} \leq \epsilon$, n samples are added to T, where the number of points sampled from each interval is proportional to its current error bound.

Assuming T was sufficiently upsampled and satisfy $\beta_{T,\beta_+} < \epsilon$, we decrease β_+ towards β. Since the algorithm did not stop we have that $\beta_{T,\beta} > \epsilon$. Therefore the Mean Value Theorem implies the existence of $\beta_* \in (\beta, \beta_+)$ such that $\beta_{T,\beta_*} = \epsilon$. Then we use the bisection method(with maximum of 10 iterations) to efficiently search for β_* and update β_+ accordingly. The algorithm runs iteratively until $\beta_{T,\beta} \leq \epsilon$ or a maximal number of 5 iterations is reached. Either way, we use the final τ and β_+ to estimate the current opacity $\hat{O}$, finally we return a fresh set of m=64 samples $\hat{O}$ using inverse transform sampling.

2.6. Generative Radiance Fields

We use the similar structure as in [4] (See Fig 1). The generator G_θ takes camera matrix K, camera pose ζ, 2D sampling pattern ν and shape/appearance codes $z_s \in \mathbf{R}^m$ / $z_a \in \mathbf{R}^n$ as input and predicts an image patch P_0. The discriminator D_ϕ compares P_0 to a patch P extracted from real image I.

Specific implementation of generator and discriminator can be found in [4]. **We keep the main structure and replace the density function estimation and sampling method in volume rendering module π with methods in VolSDF [5].** We except the generator to output well-volume-rendered patches to enhance the overall performance of the whole model.

Algorithm 1: Sampling algorithm

Data: error threshold $\epsilon > 0$;β
Initialize $T = T_0$;
Inirialize β_+ such that $\beta_{T,\beta_+} \leq \epsilon$;
while $B_{T,\beta_+} > \epsilon$ *and not max iter* **do**
  upsample T;
  if $B_{T,\beta_+} < \epsilon$ **then**
    Find $\beta_* \in (\beta, \beta_+)$ so that;
    $\beta_{T,\beta_*} = \epsilon$;
    Update $\beta_+ \leftarrow \beta_*$;
  end
end
Estimate $\hat{O}$ using T and β_+.;
$\mathbb{S} \leftarrow$ get fresh m samples using $\hat{O}^{-1}$
Output: return $\mathbb{S}$

3. Experiments

In this section, we firstly narrate the process of code modification. After that, we briefly introduce the dataset usage and experimental setup. Finally, we show the experiment results.

3.1. Code Modification Process

At the beginning, we hope to find an interface to connect VolSDF and GAN, like [4] have done with NeRF and GAN. However, we failed because the codes of GRAF and VolSDF are quite complicated and their difference are quite large. Therefore, we decide to take a different approach which makes modifications on the code of GRAF. We find out the code blocks that generate the density, and implement SDF function and Laplace function, and other detailed processes of VolSDF. After debugging, we run the experiment.

3.2. Dataset and Setup

We use the *Carla* dataset [1] in [4] which has 10k images of 18 car models with randomly sampled colors and realistic texture and reflectance properties. We follow the original experimental setup of [4] by using `default.yaml` as the config file.

3.3. Experiment Results

As we can see in Figure 3, the experiment results look awful and are quite different with the car models showed

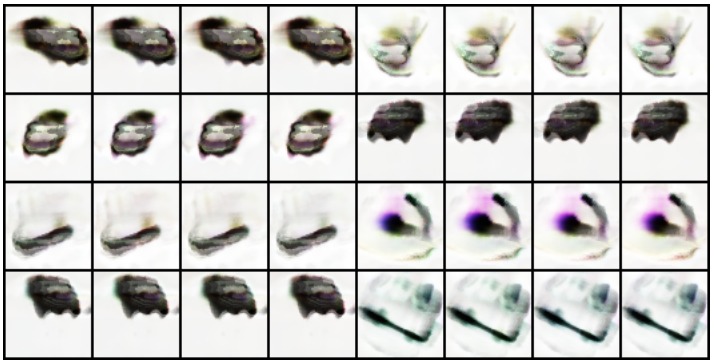

Figure 3. Experiment results.

in Figure 2. We think the reasons for these results are as follows:

- Only connect VolSDF and GRAF at the code level is not enough. We original think that if VolSDF improved NeRF then we can implement the improvement on GRAF. However, our assumption maybe logically correct but practically wrong. To complete this, we may need to find the relations on mathematics.
- Figure 3 is the results of `rgb_map`. Meanwhile, the results of `acc_map` and `depth_map` are nearly all white. That means we may missed the process of background or depth.
- Probably there are problems with codes. The implementation of `MLP` and `LaplaceDensity` may cause some mistake due to the code framework of GRAF, i.e. the framework of functions is not quite appropriate for `nn.Module` class.
- The error bound sampling have not been implemented successfully yet. This is probably the main problem.

4. Division of labor

- Jiyu Liu: debug bound sampling part and density function part
- Jundong Zhou: Report writing, debug bound sampling part
- Yongyang Pan: Report writing, implement bound sampling part
- Zhaoyu Zeng: implement density function part, presentation

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 3

[2] D. Lazaro, Z. El Bitar, V. Breton, and I. Buvat. Effect of noise and modeling errors on the reliability of fully 3d monte carlo reconstruction in SPECT. In *IEEE Symposium Conference Record Nuclear Science 2004.* IEEE. 1

[3] 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, 2020. 1, 2

[4] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 3

[5] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 1, 2, 3