Appendix

- In Sec. A1, we provide diffusion models from a score-based perspective following Karras et al. [20].
- In Sec. A2, we provide additional experiments on our approach, including additional ablation study, qualitative results, and video results.
- In Sec. A3, we document implementation details.

Algorithm 1 Training

repeat
  $\boldsymbol{x} \sim p_{\text{data}}$
  $\sigma \sim [\sigma_{\min}, \sigma_{\max}]$
  $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$
  Take gradient descent step on
    $\nabla_\phi \left\| D_\phi(\boldsymbol{x} + \sigma \boldsymbol{z}, \sigma) - \boldsymbol{x} \right\|^2$
until converged
$\text{score}(\boldsymbol{x}, \sigma) = \nabla_{\boldsymbol{x}} \log p_\sigma(\boldsymbol{x}) = (D_\phi(x, \sigma) - x)/\sigma^2$

Algorithm 2 Deterministic Sampling

$\{\sigma_i\}_{i=1}^T$ descending; $\sigma_0 = 0$
$\boldsymbol{x}_T = \sigma_T \boldsymbol{z},\ \boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\boldsymbol{x}_{t-1} = \boldsymbol{x}_t + (\sigma_t - \sigma_{t-1}) \cdot \sigma_t \cdot \text{score}(\boldsymbol{x}_t, \sigma_t)$
  $= \underbrace{(1 - w_t)\, \boldsymbol{x}_t + w_t D_\phi(\boldsymbol{x}_t, \sigma_t)}_{\text{weighted average}} \quad w_t = \frac{\sigma_t - \sigma_{t-1}}{\sigma_t}$
return $\boldsymbol{x}_0$

Figure A1. Training and Sampling Algorithm Card for Score-Based Methods with numerical scaling $s(t) = 1$ and $\sigma(t) = t$. Note that the inference step is analogous to DDIM [52], and simplifies to a weighted averaging between the current iterate $\boldsymbol{x}_t$ and the denoiser output $D(\boldsymbol{x}_t, \sigma_t)$. This particular scheduling allows for taking large step sizes, and a sample can be generated in as few as 80 network evaluations [20] while maintaining high image quality.

A1. Diffusion Models from Score-Based Perspective

We provide a more detailed recap of diffusion models from the score-based perspective. For a quick overview, we summarize the training and deterministic sampling algorithms in Fig. A1; the deterministic sampling algorithm can be made stochastic by adding noise and adjusting σ level after each update (details see Karras et al. [20]).

In the following analysis we assume that each dimension of the random vector $\boldsymbol{x}$ is independent, and that the variance in each dimension is 1. The general form of the forward noising step of a diffusion model can be described as scaling and adding noise, *i.e.*

$$\boldsymbol{x}_t = s(t)\boldsymbol{x}_0 + s(t)\sigma(t)\boldsymbol{z}, \tag{28}$$

where $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$ and $\boldsymbol{x}_0$ is a sample drawn from data distribution. $s(t)$ and $\sigma(t)$ are user-defined coefficients. Here the coefficient on noise $\boldsymbol{z}$ is parameterized as the product of $s(t)$ and $\sigma(t)$ so that $\sigma(t)$ represents the noise-to-signal ratio in $\boldsymbol{x}_t$.

SMLD [53, 54], DDIM [52] and Karras [20] sets scaling, *i.e.* $s(t) = 1$, and therefore adding noise by $\boldsymbol{x}_0 + \sigma(t)\boldsymbol{z}$ would cause $\boldsymbol{x}_t$ to numerically get larger as t increases. DDPM on the other hand introduced rapidly decreasing $s(t)$ to scale down the successive $\boldsymbol{x}_t$ so that at any time t, $p_t(x)$ has variance fixed at 1. This goal of maintaining a standard deviation 1 requires that

$$\text{Var}[\boldsymbol{x}_t] = \text{Var}[s(t)\boldsymbol{x}_0] + \text{Var}[s(t)\sigma(t)\boldsymbol{z}] \tag{29}$$

$$s(t)^2 \underbrace{\text{Var}[\boldsymbol{x}_0]}_{=\mathbf{I}} + s(t)^2\sigma(t)^2 \underbrace{\text{Var}[\boldsymbol{z}]}_{=\mathbf{I}} = \mathbf{I} \tag{30}$$

$$s(t)^2 + s(t)^2\sigma(t)^2 = 1 \tag{31}$$

$$\sigma(t) = \sqrt{\frac{1 - s(t)^2}{s(t)^2}} \tag{32}$$

DDPM specifies the $s(t)$ by a set of β_t, *i.e.*, $s(t) = \sqrt{\bar{\alpha}_t} = \sqrt{\prod_{i \le t} \alpha_i} = \sqrt{\prod_{i \le t}(1 - \beta_i)}$, and therefore $\sigma(t) = \sqrt{\frac{1 - \bar{\alpha}_t}{\bar{\alpha}_t}}$.

The noising step (28) describes the marginal distribution at $p_t(\boldsymbol{x})$. The infinitesimal time evolution of this process can be written as the following stochastic differential equation [55]:

$$\mathrm{d}\boldsymbol{x} = f(t)\boldsymbol{x}\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t \quad \text{where} \quad f(t) = \frac{\dot{s}(t)}{s(t)} \quad g(t) = s(t)\sqrt{2\dot{\sigma}(t)\sigma(t)}. \tag{33}$$

Fokker-Planck equation [32] states that a stochastic differential equation of the form (33) is identified with a partial differential equation describing the marginal probability density distribution $p_t(\boldsymbol{x})$

$$\mathrm{d}\boldsymbol{x} = f(x,t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t \;\longleftrightarrow\; \frac{\partial p_t(\boldsymbol{x})}{\partial t} = -\boldsymbol{\nabla}\cdot\left[f(x,t)\,p_t(\boldsymbol{x}) - \frac{g(t)^2}{2}\boldsymbol{\nabla}_x p_t(\boldsymbol{x})\right]. \tag{34}$$

Applying this identity tells us that a stochastic differential equation like (33) implies a deterministic, ordinary differential equation. Here we illustrate the proof schematically:

$$\begin{array}{ccc}
\underbrace{\mathrm{d}\boldsymbol{x} = f(t)\,\boldsymbol{x}\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{stochastic}} & \xRightarrow{\text{FP}} & \frac{\partial p_t(\boldsymbol{x})}{\partial t} = -\boldsymbol{\nabla}\cdot\left[f(t)\,\boldsymbol{x}\,p_t(\boldsymbol{x}) - \frac{g(t)^2}{2}\boldsymbol{\nabla}_x p_t(\boldsymbol{x})\right] \\
\Big\downarrow \text{implies} & & \Big\Downarrow \text{equal (by log derivative trick; expanded below)} \\
\underbrace{\mathrm{d}\boldsymbol{x} = \left(f(t)\,\boldsymbol{x} - \frac{g(t)^2}{2}\boldsymbol{\nabla}_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)\mathrm{d}t + 0\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{deterministic}} & \xLeftarrow[\text{FP}]{} & \frac{\partial p_t(\boldsymbol{x})}{\partial t} = -\boldsymbol{\nabla}\cdot\left[\left(f(t)\,\boldsymbol{x} - \frac{g(t)^2}{2}\boldsymbol{\nabla}_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)p_t(\boldsymbol{x}) - 0\right]
\end{array} \tag{35}$$

The application of the log derivative trick is expanded below:

$$\frac{\partial p_t(\boldsymbol{x})}{\partial t} = -\boldsymbol{\nabla}\cdot\left[f(t)\,\boldsymbol{x}\,p_t(\boldsymbol{x}) - \frac{g(t)^2}{2}\boldsymbol{\nabla}_x p_t(\boldsymbol{x})\right] \tag{36}$$

$$= -\boldsymbol{\nabla}\cdot\left[\frac{f(t)\,\boldsymbol{x}\,p_t(\boldsymbol{x}) - \frac{g(t)^2}{2}\boldsymbol{\nabla}_x p_t(\boldsymbol{x})}{p_t(\boldsymbol{x})}p_t(\boldsymbol{x})\right] \tag{37}$$

$$= -\boldsymbol{\nabla}\cdot\left[\left(f(t)\,\boldsymbol{x} - \frac{g(t)^2}{2}\underbrace{\frac{\boldsymbol{\nabla}_x p_t(\boldsymbol{x})}{p_t(\boldsymbol{x})}}_{\text{log derivative}}\right)p_t(\boldsymbol{x})\right] \tag{38}$$

$$= -\boldsymbol{\nabla}\cdot\left[\left(f(t)\,\boldsymbol{x} - \frac{g(t)^2}{2}\boldsymbol{\nabla}_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\right)p_t(\boldsymbol{x})\right]. \tag{39}$$

Substituting the expression for $f(t)$ and $g(t)$ from (33), we obtain an ODE from which we can sample the data by applying the score function with a step schedule that theoretically guarantees to take us back to initial, clean data distribution

$$\mathrm{d}\boldsymbol{x} = \frac{\dot{s}(t)}{s(t)}\boldsymbol{x} - \frac{1}{2}\left(s(t)\sqrt{2\dot{\sigma}(t)\sigma(t)}\right)^2\boldsymbol{\nabla}_{\boldsymbol{x}}\log p_t(\boldsymbol{x})\,\mathrm{d}t \tag{40}$$

$$\mathrm{d}\boldsymbol{x} = \frac{\dot{s}(t)}{s(t)}\boldsymbol{x} - s(t)\frac{\dot{\sigma}(t)}{\sigma(t)}\Big(D(\boldsymbol{x}/s(t);\sigma(t)) - \boldsymbol{x}/s(t)\Big)\,\mathrm{d}t\,. \tag{41}$$

When $s(t) = 1$, $\sigma(t) = t$, the above simplifies to

$$\mathrm{d}\boldsymbol{x} = -\sigma_t\cdot\frac{D(\boldsymbol{x};\sigma_t) - \boldsymbol{x}}{\sigma_t^2}\,\mathrm{d}t \tag{42}$$

$$\mathrm{d}\boldsymbol{x} = -\sigma_t\cdot\mathrm{score}(\boldsymbol{x},\sigma_t)\,\mathrm{d}t \tag{43}$$

Note that this schedule with $s(t) = 1$, $\sigma(t) = t$ allows for taking large step sizes during inference since it introduces no extra curvature in the trajectory beyond what's induced by the score function itself. The discretized sampling algorithm of equation (43) is described in Fig. A1.

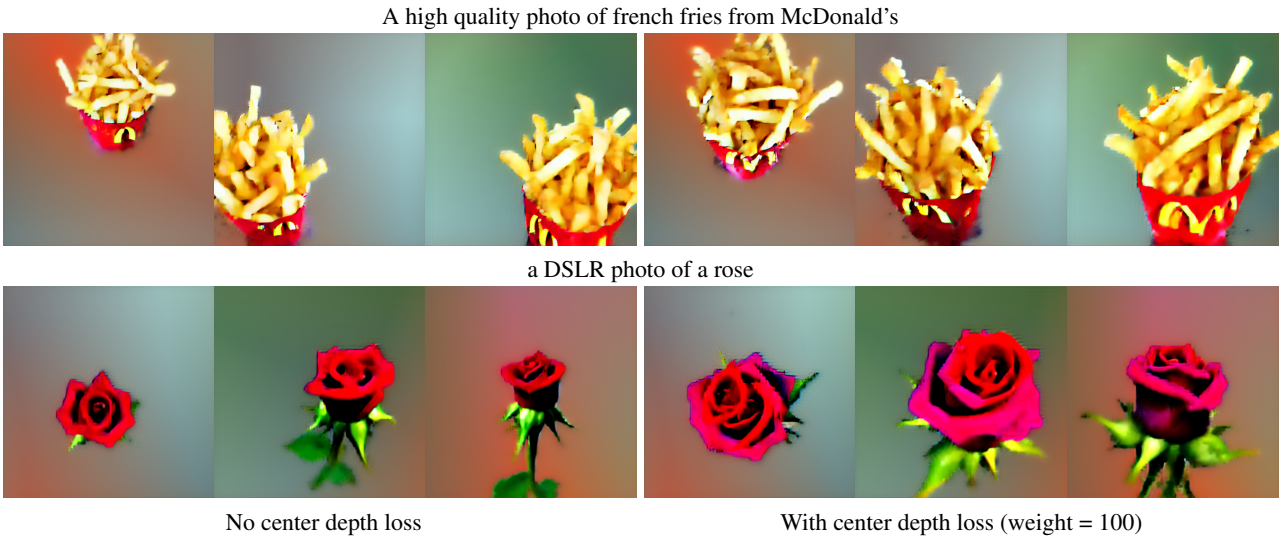

Figure A2. Ablation experiments on the proposed center depth loss. Each pair of corresponding columns of the same prompt are visualized from the same camera angle.

A2. Additional Experiments

Ablation on center depth loss. In Fig. A2, we illustrate the effect of the center depth loss proposed in Eq. (27). Without the center depth loss, we observe that some objects, *e.g.*, French Fries, are placed far from the center of the scene box and tend to drift around when the camera viewpoints are changed. This effect is more pronounced in the provided video result. In contrast, a moderate center depth loss forces the object to be placed at the scene box center. Additionally, we observe that the objects tend to be enlarged to occupy more of the visible screen space without wasting model capacity.

Additional qualitative results. We provide additional qualitative results from SJC in Fig. A3. Note that we increase the resolution of the depth maps beyond the 64×64 resolution of the image latents by rendering subpixel rays. In general, we observe that the volumetric renderer is powerful enough to hallucinate shadows (horse), water surfaces (Sydney opera house, duck), grasslands (zebra) and even a traffic lane (school bus), using the volume densities.

Video results. We have attached numerous video results in the project website, and named each file after the text prompt used to generate the 3D asset. In addition, we included the videos for the ablation experiments in Fig. 7 and Fig. A2.

A3. Implementation Details

3D scene setup. Our voxel grids are of size 100^3, and placed at world origin with a normalized side length $[-1, 1]^3$. We sample cameras uniformly on a hemisphere that covers the voxel cube with a radius of 1.5, with look-at directions pointing at the origin. The camera field of view is randomly sampled from 40 degrees to 70 degrees during optimization, and fixed to 60 degrees at test time. We found the jittering on FoV to help with 3D optimization in some cases, and this data augmentation technique is reported in DreamFusion [43]. Our scene background consists of an optimizable image of size 4×4 environment-mapped to the spherical surface by azimuth and elevation angles of the incoming ray. The small image size with constrained capacity helps to avoid confounding visual artifacts accumulating in the background during optimization.

Optimization. We use Adamax [24] optimizer and perform gradient descent at a learning rate of 0.05 for $10,000$ steps, with some prompts running at a longer schedule for better quality. Note that when performing gradient descent with PAAS, we implicitly rely on the optimizer's momentum state to perform the averaging. We have tried explicitly averaging the scores at multiple noise perturbations, but observed no clear benefits or degradation. The language-guidance scale is set to 100. Our system consumes 9GB of GPU memory during optimization, and takes approximately 25 minutes on an A6000 GPU including the time spent on miscellaneous tasks like visualization.

View-dependent prompting. An influence of DreamFusion [43] on our work is the use of view-dependent prompting. Language prompts are prepended with one of the following: "overhead view of", "front view of", "backside view of", "side view of" depending on the camera location. More specifically, when camera elevation is above 30 degrees we use the "overhead

view" prompt. Otherwise, the prompts are assigned based on the azimuth quadrant the camera falls into. This technique helps to alleviate the degeneracy of multiple frontal faces being painted around an object during optimization. We hope as part of our future work to develop a more general solution to induce the optimization towards more plausible geometry without using language as guidance.

Figure A3. Additional results of text-prompted generation of 3D models with SJC.