SuperGaussian: Repurposing Video Models for 3D Super Resolution

Yuan Shen[1,2]*, Duygu Ceylan[2], Paul Guerrero[2], Zexiang Xu[2], Niloy J. Mitra[2,3], Shenlong Wang[1], and Anna Frühstück[2]

[1] University of Illinois at Urbana-Champaign
[2] Adobe Research
[3] University College London

Abstract. We present a simple, modular, and generic method that upsamples coarse 3D models by adding geometric and appearance details. While generative 3D models now exist, they do not yet match the quality of their counterparts in image and video domains. We demonstrate that it is possible to directly repurpose existing (pre-trained) video models for 3D super-resolution and thus sidestep the problem of the shortage of large repositories of high-quality 3D training models. We describe how to repurpose video upsampling models – which are not 3D consistent – and combine them with 3D consolidation to produce 3D-consistent results. As output, we produce high-quality Gaussian Splat models, which are object-centric and effective. Our method is category-agnostic and can be easily incorporated into existing 3D workflows. We evaluate our proposed SuperGaussian on a variety of 3D inputs, which are diverse both in terms of complexity and representation (e.g., Gaussian Splats or NeRFs), and demonstrate that our simple method significantly improves the fidelity of current generative 3D models.
Check our project website for details: supergaussian.github.io.

Keywords: 3D super resolution · video upsampling · category-agnostic · 3D generation · Gaussian splatting · 3D-consistent

1 Introduction

Generative 3D models have become a reality. Multiple methods [2, 25, 31, 46] have been developed to generate 3D models, optionally conditioned using text prompts or images. These methods use a combination of image and 3D data for supervision, are fast, and produce diverse results.

Unfortunately, the 3D models generated by the current methods still lack the level of detail and accuracy achieved by state-of-the-art generative models for images [26, 40] or videos [6]. Multiple challenges contribute to this limitation. First is the choice of 3D representation. While grid-based models are most popular as they do not need prior knowledge of the (generated) shape, their regular structure (e.g., volume grid [25, 34], triplanes [2, 5]) puts a limit on the fidelity

* This project was done during Yuan's internship at Adobe Research.

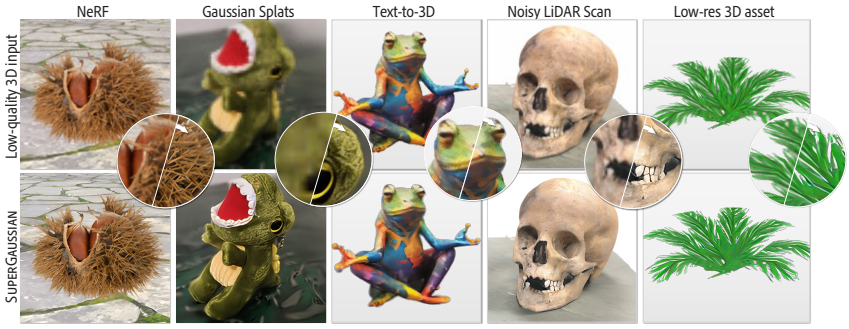

Fig. 1: We present SuperGaussian, a novel method that repurposes existing video upsampling methods for the 3D superresolution task. SuperGaussian can handle various input types such as NeRFs, Gaussian Splats, reconstructions obtained from noisy scans, models generated by recent text-to-3D methods [31], or low-poly meshes (e.g., assets used in Sim-on-Wheels [45]). SuperGaussian generates high-resolution 3D outputs with rich geometric and texture details in the form of Gaussian Splats.

of the generation results. Secondly, acquiring large volumes of high-quality yet diverse 3D data continues to be difficult. While state-of-the-art image and video models are trained on several billion training samples, the most extensive 3D training datasets, at best, contain a few million objects.

We explore how to increase the fidelity of generated 3D objects. Starting from any generic coarse 3D representation, our goal is to 'upsample' the (coarse) 3D input model *without* requiring category-specific training. Our main observation is that any 3D representation can be rendered from multiple viewpoints along a smooth trajectory and mapped to an intermediate generic video representation. Hence, it is possible to repurpose existing video models [9, 69] to perform the 3D upsampling or super-resolution task. Such models are trained on large sets of video data, therefore providing strong priors that can be applied in general scenarios. The critical challenge is to ensure 3D consistency; while being temporally smooth, video models, however, are not guaranteed to be 3D consistent. Note that our approach of using video, unlike those using image-based priors and treating each frame independently, significantly improves the (initial) consistency across time.

We address this challenge with a simple, modular, and generic approach that can be integrated into existing workflows. We proceed as follows: First, we render a video of the scene from the coarse 3D input given sampled view trajectories; second, we upsample the rendered video using a pretrained *video-based upsampler* that is optionally fine-tuned to handle domain-specific artifacts of the input modality. For 3D consolidation, we adopt Gaussian Splatting [27] as our output representation. Being an object-centric representation, Gaussian splats are ideally suited for encoding individual objects and are capable of capturing local details. Gaussian splats also strike a good balance between simplicity, fidelity of encoded models, and efficiency of rendering. Fig. 1 shows that our proposed

video-based 3D super-resolution framework can be successfully adapted to upscale diverse sets of coarse 3D modalities.

We evaluate our algorithm on various low-resolution assets and report the quality and diversity of the generated high-resolution Gaussian splats. We compare our algorithm against possible alternatives and report an ablation study of our design choices. In summary, our main contributions include:

(i) re-purposing pretrained video upsampling models for 3D super-resolution;
(ii) finetuning video upsampling prior using generic 3D data to handle artifacts characteristic to low-res Gaussian splats; and
(iii) proposing SuperGaussian as an efficient, class-agnostic, and modular 3D super-resolution procedure that successfully upsamples diverse 3D scene representation, including Gaussian splats, NeRF, and others.

2 Related Work

Image Super-Resolution. Enhancing image details and resolution is a long-standing problem in computer vision. This is inherently an ill-posed problem, so generative methods are well suited to tackle the hallucination of missing information. For conciseness, we concentrate on recent generative approaches and refer the reader to a comprehensive review [11] of this subject area.

Early CNN-based approaches [14, 15, 28, 66] recognize the applicability of deep networks for super-resolution. Several approaches leverage powerful priors learned by image synthesis models to infer image details. Generative Adversarial Networks [17] have been extensively used for this task, either through optimization-based techniques [36] or conditional feed-forward network designs [3, 24,29,53–55]. Other approaches have explored normalizing flows [35] or transformer-based methods [12,13]. The immensely popular denoising diffusion probabilistic models [22,41] have also been recently explored for image upsampling [30,42,63].

Video Super-Resolution. Following the success of learning-based single-image super-resolution methods, there have been various efforts to extend this success to the video domain. Several strategies have been explored to make this extension. While some work [33] introduces temporal aggregation modules to aggregate information across time, recurrent neural networks [8, 60] with uni- and bi-directional information sharing have also been a choice of architecture. Deformable convolutions [48,52] have been utilized to align per-frame features both for video upsampling and restoration tasks. More recently, diffusion-based video upsamplers have been proposed by injecting temporal layers into base diffusion model [69]. In our work, we leverage VideoGigaGAN [57], a state-of-the-art generative video upsampler. However, our method is agnostic to the video model we use and can further be improved in the future with advances in video upsampling techniques, e.g., diffusion-based video upsampler.

Image-Based 3D Super-Resolution. Several approaches have recently aimed at increasing the resolution of implicit radiance fields, in particular Neural Radiance Fields (NeRFs). This is an important and practical application for both capture and synthesis scenarios where higher detail and quality are desirable. Start-

ing with a low-res radiance field, at each iteration, this thread of work [19,23,49] renders images from the current representation. Next, a pre-trained image upsampler is used to upsample the renderings. Then the gradient flow coming from the difference between the current renderings and upsampled images, is used to optimize the 3D representation. Similarly, CROP [61] also leverages a single image upsampler but introduces an uncertainty volume to improve the multi-view consistency. Unlike these methods, however, we adopt a video upsampler instead of a single-image upsampling network. Thanks to better handling of temporal consistency, we show that using the video upsampler improves the visual quality of the final results. Moreover, unlike previous work mainly focusing on upsampling NeRF representations, SuperGaussian is a generic approach, which we demonstrate on various 3D input types.

In the context of 3D-aware GANs, several works have been proposed [7, 16] that jointly learn to generate a triplane-based 3D representation as well as a 2D image upsampling module. The upsampling module is applicable only to the specific triplane features learned by the generator and the final output is not fully 3D consistent since each view is upsampled independently.

Feed-forward 3D Super-Resolution. Several 3D generation and reconstruction methods have proposed direct 3D superresolution modules. Smith et al. [47] represent a 3D shape as multi-view orthographic depth maps which are then upsampled. The recent diffusion-based 3D generation method LASDiffusion [68] presents a method to predict signed distance fields from an occupancy map to improve the resolution of the 3D shapes. Both methods, however, focus only on geometry refinement while our work jointly performs appearance and geometry upsampling. Rodin [51], a triplane-based diffusion method, presents a triplane upsampling strategy. This approach is applicable only to the specific triplane features learned. Finally, our approach does not need large-scale 3D datasets and instead leverages existing video models trained on video data.

Deep Implicit Representations. Recently, neural implicit representations such as NeRF [38], VolSDF [59], and NeuS [50] have been popularized as an efficient and compact representation for 3D shapes. Efficiently capturing high-resolution details, however, still remains a challenge for such representations due to the low-dimensional volumetric structure and the computationally intensive volumetric rendering process. Recently, Gaussian Splatting [27] has been proposed as an alternative that provides an object-centric representation with fast rendering capabilities. The scene is represented by a large number of 3D Gaussian blobs whose properties are optimized through differentiable rendering. In our work, we adapt Gaussian splats as the final 3D representation output due to the efficient optimization and rendering capabilities they offer.

3 Our Approach

3.1 Overview

Given a coarse 3D representation, our goal is to perform super-resolution to improve the fidelity of the 3D representation and capture more local details. Based

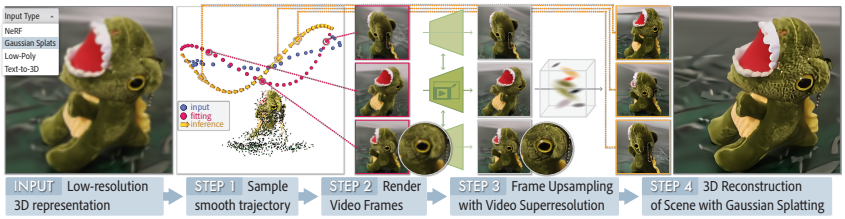

Fig. 2: SuperGaussian pipeline. Given an input low-res 3D representation, which can be in various formats, we first sample a smooth camera trajectory and render an intermediate low-resolution video. We then upsample this video using existing video upsamplers and obtain a higher resolution 3D representation that has sharper and more vivid details. Our method, SuperGaussian, produces a final 3D representation in the form of high-resolution Gaussian Splats.

on the observation that 3D content can be represented as a video depicting the 3D scene from multiple viewpoints, the main premise of our work is to leverage existing video upsampling priors for the task of 3D upsampling. As illustrated in Fig. 2, our method SuperGaussian consists of two main steps. First, we run upsampling on the video rendered from the coarse 3D representation to gain resolution and obtain sharp results. Then, we perform 3D reconstruction to generate a consistent 3D representation. In addition to leveraging the prior of the video upsampler, which is trained on a large set of video data, we perform fine-tuning over domain-specific low-res videos (i.e., videos rendered from low-res 3D representations). Hence, SuperGaussian is able to handle complicated degradation caused by various 3D capture and generation processes. Note that each component in our framework is highly modularized and could easily be replaced with other state-of-the-art video methods.

Next, we unfold this section by formulating the input and output of our method (Sec. 3.2), introducing the key video upsampling (Sec. 3.3) in our framework, and discussing the details of the 3D optimization (Sec. 3.4).

3.2 Problem Formulation

Our framework can handle a diverse set of coarse 3D representations of a static scene, which we denote as ψ_{low}. For example, ψ_{low} can be Gaussian Splats, NeRF, a low-poly mesh, a low-quality captured video, or a generated 3D object with recent text-to-3D methods [31]. Any such 3D representation, ψ_{low}, can be rendered from multiple viewpoints to yield a video, a common intermediate representation. In practice, we render each 3D input from N smooth trajectories with viewpoints $\xi_{1\ldots T}^{1\cdots N}$ resulting in a sequence of RGB images $I_{1\ldots T}^{1\cdots N}$. Here, the subscript indexes the viewpoint or the pose along each trajectory, and the superscript denotes the trajectory ID. We assume the camera movement between adjacent frames is sufficiently small such that a standard video upsampler can leverage enough temporal alignment. After we perform a sequence of video upsampling, as output, we generate a high-fidelity 3D representation ψ_{high} in the form of Gaussian Splats (note that camera views are known in our setup and do not need to be estimated). This final 3D optimization produces a true 3D

output and, in the process, removes any remaining temporal inconsistencies in the refined video representations.

3.3 Initial Upsampling

First, we manually sample trajectories near the target scene in the empty region. Given the trajectory $I_{1...T} \in \{\mathbb{R}^{W \times H \times 3}\}$ describing the camera path for each individual video, the video upsampler outputs a trajectory with $\times r$ upsampling ($r = 4$ in our experiments). Mathematically,

$$\hat{I}_{1...T} = f(I_{1...T}) \in \mathbb{R}^{rW \times rH \times 3}, \tag{1}$$

where f denotes the video upsampler and $\hat{I}_{1...T}$ is the upsampled video. We assume the initial rendering resolution should be sufficiently high such that the rendered fidelity is bottlenecked by the coarse level of the input 3D representation.

Our framework can easily integrate any of the state-of-the-art pre-trained video upsamplers. For our case, we use VideoGigaGAN [57], a generative video upsampler. Please refer to Sec. 4.1 to see how we set up the video upsampler to have fair comparisons with baselines. In cases of handling input representations with severe domain bias (see Fig. 6), additional fine-tuning is desired. For example, with stripy or blob-like artifacts after zooming in, renderings from Gaussian Splats follow different degradation from standard augmentations deployed in state-of-the-art video upsamplers. Hence, to finetune the video upsampler, we need pairs of low and high-resolution videos that depict the specific degradation we would like to model. For this purpose, we use the multi-view dataset MVImgNet [62], which depicts a variety of 3D objects and scenes. First, we bilinearly downsample the original images in the dataset by a factor of 8, i.e., to 64×64px resolution, to obtain a set of low-resolution images. We then fit low-resolution Gaussian Splats to these images [27] as described in more detail in Section 3.4. We render the optimized low-res Gaussians in the original camera trajectory provided by the dataset as input to the video upsampler. As the target ground truth, we use the original videos from the dataset, which are resized to be 4 times the resolution of the input. We finetune the model using the Charbonnier regression loss [4] for its robustness to outlier pixels, the LPIPS loss [64] for its perceptual level improvements, and GAN loss [18] for its generative behavior.

3.4 3D Optimization via Gaussian Splats

We use the official Gaussian Splatting codebase[4] to perform 3D optimization to fit Gaussians to the upsampled videos. In particular, for our experiments, we follow the standard Gaussian Splatting optimization process by running 2k steps. Note that as we have perfect camera information, we directly provide them to the optimization, rather than estimating them using SfM. As for our loss function, following the practice of the original paper [27], we use L_1 and L_{SSIM} losses when

[4] https://github.com/graphdeco-inria/gaussian-splatting

optimizing the Gaussians. The advantages of adopting 3D Gaussian Splatting lie in its being an object-centric representation and its efficiency in terms of training and rendering. Besides, it is able to capture view-dependent effects well for the upsampled frames. However, SuperGaussian can also be easily integrated with other types of 3D representations, e.g., Neural Radiance Fields, which we use in our experiments on the Blender-synthetic dataset [38].

4 Experiments

4.1 Setup

Dataset: We finetune the video upsampler component of our method on the MV-ImageNet dataset [62], which consists of a rich number of real object-centric videos with high quality. This dataset is processed to fit low-resolution Gaussian Splats reconstructed from 64×64px posed images as described in the previous section. In total, 200k scenes are used for training and 5k scenes for validation. A separate set of 500 scenes are used for quantitative evaluation. We obtain the ground-truth high-resolution images by center-cropping and resizing them to square images at 256×256px resolution on MVImgNet. To demonstrate the generalizability of our approach, we also show results on the Blender-synthetic dataset, consisting of synthetic 3D objects. We use the original data split as prior work [32, 49] for evaluation.

Metrics: On the MVImgNet dataset, we utilize metrics focused on perceptual quality when performing quantitative comparisons, i.e., LPIPS [65], NIQE [39], Inception score [43] and FID [21]. For the FID computation, we use the real images provided by MVImgNet as the real distribution. On the NeRF Synthetic dataset, we report the same metrics, i.e., LPIPS, PSNR, and SSIM, analogous to FastSR-NeRF [32] and NeRF-SR [49]. Aside from quantitative metrics, we strongly encourage the readers to check our qualitative results and videos in the supplementary.

Baselines: While there are a recent few domain-specific 3D upsampling solutions [23, 49, 58], we focus on comparing against baselines that are applicable to generic 3D representations. Previous work for generic 3D upsampling follows a similar paradigm by leveraging upsampling priors to iteratively optimize the chosen 3D representation. The popular choice of the prior is image-based upsampling models, instead of our proposed video-based upsampler, one seemingly trivial design choice that profoundly impacts the upsampled quality. With that, we narrow down to three state-of-the-art methods: (1) Instruct-G2G (Instruct-NeRF2NeRF [20] with Gaussian Splats as 3D format), (2) Super-NeRF [19] with Gaussian Splats as 3D format, and (3) FastSR-NeRF [67]. For completeness, we include one domain-specific baseline, NeRF-SR [49]. Besides, we introduce a customized baseline, *Pre-hoc image*, as discussed below.

For a fair comparison, we use the same state-of-the-art image upsampler, GigaGAN [24] for all the baselines. Specifically for the GigaGAN checkpoint,

Table 1: Benchmark of video and image upsampler priors. We report the performance of the upsampled frames for pre-hoc image and ours after applying the corresponding upsampler. Note: reconstruction is not performed for these results.

Model	LPIPS ↓	NIQE ↓	FID ↓	IS ↑
image upsampler	0.1533	7.06	14.67	11.67 ± 1.14
video upsampler	**0.1511**	**6.04**	**14.61**	**12.45** ± **1.19**

we follow the same strategy proposed in the original paper to train a 4× super-resolution model on LAION [44] that learns to upsample images from 64×64 to 256×256px with great details. In SuperGaussian, we use VideoGigaGAN [57] as the video upsampler, which mostly re-uses the former GigaGAN architecture except for additional BasicVSR++ layers for temporal feature extraction and propogation [9] that processes the low-res video input frames. The model size of our video prior turns out to be slightly smaller than the image prior after adjusting feature dimensions. Finally, we finetune both the image and video upsampler on MVImgNet until convergence. Benchmark of both priors are reported in Tab. 1. Both priors show generative behaviors after upsampling, and the video prior shows great temporal consistency across upsampled frames. Note SuperGaussian can be easily integrated with a stronger video prior, e.g., Upscale-a-video [69] or Sora from OpenAI [1] by finetuning over low-res video as conditions. For Instruct-G2G, we first optimize 2k steps for the warm-up stage, which prepares the low-res representation, and then optimize another 2k steps with upsampling by upsampling every 20 steps. For SuperNeRF, since the authors did not release their codebase, we self-implemented their method with our best knowledge. We wrap our GigaGAN with the consistency enforcement module [3]. Similarly, we optimize 2k step for low-res representation and then optimize for another 2k steps. We set the learning rate of the CCLC embedding to be 5×10^{-5}, which is critical to be small to avoid blurry results. In our experiments, we observe that running an iterative optimization with an image upsampler, i.e., SuperNeRF [19] and Instruct-G2G, often results in more blurry results compared to upsampling all the frames individually first, in a *pre-hoc* manner, and then performing an optimization. Hence, we define another baseline, *Pre-hoc image*, where we individually upsample all the frames once and then optimize the 3D representation.

Finally, to remove the effect of representation on the experiment results, we modify our baselines to all use 3D Gaussian Splattings as their global representation for comparisons on MVImgNet. On the Blender synthetic dataset, Neural Radiance field is chosen as the global representation across methods to faily compare with other baselines represented in NeRF.

Implementation details: During finetuning, all model parameters are finetuned with a learning rate of 5×10^{-5}. We weigh the LPIPS, Charbonnier regression losses, and adversarial loss with weights 15, 10, and 0.05, respectively. Following GigaGAN [24], we apply R_1 regularization [37] on the discriminator at a weight of 0.02048 and interval of 16. Batch size is 64 with 12 video frames in each. Our models are trained on 64 A100 GPUs, each with 80GB VRAM.

Table 2: Quantitative Comparisons on MVImgNet to upsample low-res Gaussian Splattings. We compare our methods against baselines using perceptual metrics. Our method, even though it is generic, consistently produces the best quantitative results. Instruct-G2G stands for Instruct-N2N with 3D Gaussian Splats as its 3D representation. We encourage the reader to inspect the visual results in our website, which highlights that the visual quality of our method surpasses the baselines.

Method	LPIPS ↓	NIQE ↓	FID ↓	IS ↑
Instruct-G2G [20]	0.1867	8.33	32.56	10.52 ± 1.06
Super-NeRF [19]	0.2204	8.84	37.54	10.40 ± 1.03
Pre-hoc image	0.1524	7.65	27.04	11.27 ± 0.99
Ours	**0.1290**	**6.80**	**24.32**	**11.69 ± 1.08**

Table 3: Quantitative Comparison on Blender synthetic dataset to upsample low-res posed RGB images. Here, we compare on ×4 upsampling from 200×200 to 800×800px. We compare our methods against baselines on the official test set using metrics reported in prior work. Our method produces on-par quantitative results. Besides, our results yield more generative details, which are not captured by the reference metrics. For a fair comparison, we use Neural Radiance Field, i.e., TensoRF [10], as our 3D representation. Other baseline results are directly taken from their paper.

Method	LPIPS ↓	PSNR ↑	SSIM ↑
FastSR-NeRF [32]	0.075	**30.47**	**0.944**
NeRF-SR [49]	0.076	28.46	0.921
Ours	**0.067**	28.44	0.923

For a set with 30 frames, the video upsampling takes ∼1 second to upsample 64 × 64 inputs. High-resolution Gaussian splat optimization takes around 30 seconds for 2k steps to fit images of resolution 256×256. On average, our pipeline takes 141.33 sec to complete, the most efficient compared with baselines. We note that our codebase can be further optimized for efficiency as there is substantial room for improvement in I/O and pipelining.

4.2 Comparison Studies

Experiment protocol: On MVImgNet, for quantitative evaluations, given a low-res 3D representation, we prepare a smooth camera trajectory that passes through a set of camera poses perturbed from the original camera poses provided by the corresponding datasets. We use a video rendered from the resulting camera trajectory as input to the video upsampler. To perform pose perturbation, we randomly sample camera centers along the line segments between the scene center and each pose in the original camera trajectory. We fit a B-Spline curve to such sampled camera centers and resample camera positions on the resulting curve; finally, we add sinusoidal perturbation along the up direction. With the camera positions settled, we set the orientation of each camera by defining a look-at vector towards the scene center. The resulting perturbed camera trajectories we obtain differ from the original camera trajectories we use for finetuning the video upsampler on the MVImgNet dataset. Once we obtain a high-res 3D representation, to evaluate the result quantitatively, we use the same perturbation strategy to sample a novel evaluation trajectory that is different than the