GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images

Adam Tlemsani

February 2025

1 Introduction

The following is a written report detailing the main contributions of the GET3D [1] paper. This paper was published by Jun Gao et al, a research team from NVIDIA [2] based in the University of Toronto. The paper was published in 2022 and accepted into NeurIPS [3] the same year. This report contains both an explanation of the technical details regrading the paper and an in depth discussion about the paper's impact and limitations. The rest of the report is structured into 2 main sections, a paper summary and a critique and discussion. We begin with a brief introduction into the problem this paper is attempting to help solve. Followed by that is a primer in some core concepts necessary in computer graphics to understand the paper. A breakdown of the method employed in the GET3D paper is presented, followed by its results. Finally, to conclude this report we discuss some of its impact, wider implications and critiques.

2 Paper summary

2.1 The problem

The main problem this paper attempts to solve is the scalable production of 3D models. Various industries are dependant on these models including the films industry who commonly need to create entire digital worlds capable of immersing an audience into believing is real, the video games industry who similarly need vast 3D worlds to captivate their players and increasingly, medical technologies are using such 3D models for tasks such as surgery planning, interactive demonstrations and prosthetic creation.

Figure 1: A 3D world created by [4] rendered with Blender [5]. This world includes various objects such as ships, icebergs, birds and clouds.

See Figure 1 for an example of one such world. Here the world is constructed of various 3D objects such as ships, icebergs, birds and water. Whilst a human could build all the models and

construct this world in a reasonable amount of time, it quickly becomes infeasible as the number of objects grows.

Figure 2: A more complex world [6] once again rendered with Blender [5] with many more objects. Worlds of this scale become increasingly difficult for humans to create.

Figure 2 illustrates a more complex world. Here there are unique buildings, walls, mountains, vegetation and much more. Traditionally, these models were meticulously developed by hand. Whilst this has served well for the past couple of decades, as the need for more complex worlds grows, a limit is reached on how quickly and efficiently humans can develop these models.

With the importance of 3D models and the challenges of their creation in mind. The GET3D [1] paper, the main focus of this report, attempts to ease the development of these 3D models through the use of generative AI. Their approach builds upon years of advancements in the field of generative AI and is an accumulation of ground breaking technologies. Several novel approaches to tackle the many challenges of 3D model creation are also introduced in this paper and will be explored in the upcoming sections.

2.2 Computer graphics background

Before we begin with a discussion of the methods presented in the GET3D paper, it will be useful to cover some fundamentals of computer graphics. An important concept critical to understanding the paper are textures. In the context of computer graphics, when discussing textures, we generally are referring to images which can be applied to a model to give it properties such as colour, roughness and surface normals. This may be different to the typical meaning of textures which could refer to an objects roughness or the sensation upon touching it. When discussing this paper [1] we typically are only referring to the colour property when mentioning textures as it is the primary focus of the paper. Some work was done in the paper to cover the other use cases we mentioned such as specular roughness, but it was not the primary focus. Figure 3 further illustrates how textures

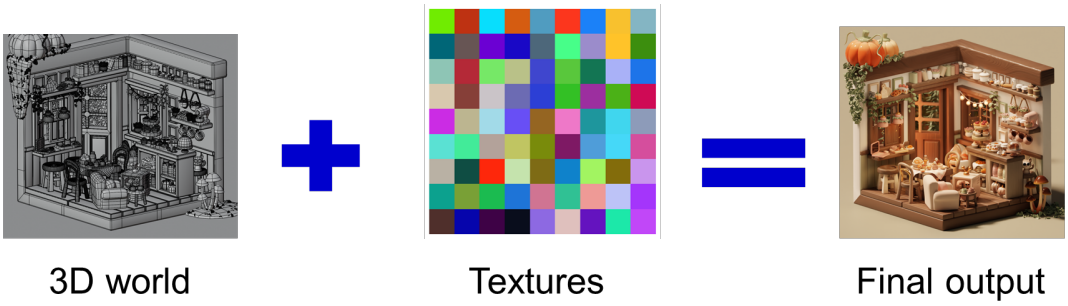

Figure 3: A visual illustration of how textures work in graphics. The 3D world [7] was rendered in Blender [5] with and without its colour. The texture image dictates the colour at every point in the 3D world.

work in computer graphics. A key subtlety here is that objects and the colours are separate and do not depend on each other. This allows for 2 important use cases, changing the colour on an already generated model, or change the model for an already chosen colour. Any system to be used for 3D model generation should ideally be able to keep the objects and colours disentangled.

2.3 Background

Before we begin discussing the novel contributions of the main model presented in the paper, we should, as the paper does, examine previous models and attempts at 3D model generation. Table 1 shows the limitation of previous methods and how the GET3D model attempts to solve them.

Method	Application	Representation	Supervision	Textured Mesh	Arbitrary Topology
OccNet [8]	3D generation	Implicit 3D	3D	✗	✓
PointFlow [9]	3D generation	Point cloud 3D	3D	✗	✓
Texture3D [10]	3D generation	Mesh	2D	✓	✗
StyleNerf [11]	3D-aware NV	Neural field	2D	✗	✓
EG3D [12]	3D-aware NV	Neural field	2D	✗	✓
PiGAN [13]	3D-aware NV	Neural field	2D	✗	✓
GRAF [14]	3D-aware NV	Neural field	2D	✗	✓
Ours	3D generation	Mesh	2D	✓	✓

Table 1: Comparison of different 3D generation and neural volume (NV) methods from [1]. The first 7 methods listed are all previous methods with their limitations presented in the columns. The last row exemplifies the strengths of the GET3D model as previous methods struggled to meet the necessary criteria for scalable and convenient 3D model generation

In the table we can see that past models have struggled to achieve all the necessary features to be practically useful. The GET3D paper explains that models which produce neural volumes are not as useful as actual 3D object generation. For the representation, we want meshes as this is what most graphics engines such as Blender [5] are designed to utilise. If a model uses a different representation, we would need to further process the output which leads to delays in utilising the the 3D models. For the supervision we prefer 2D images as there is vastly more 2D images for training then 3D models. Textured meshes are the preferred output as manually trying to find a texture which gives the object a good appearance can be a tedious process. Finally, our model should be able to generate arbitrary topology, in other words we don't want our model to be limited in the types of 3D shapes it can be produce. From Table 1 we can see that GET3D is the only model which satisfies all these demands.

2.4 Method

The main model of the paper is presented in Figure 4.

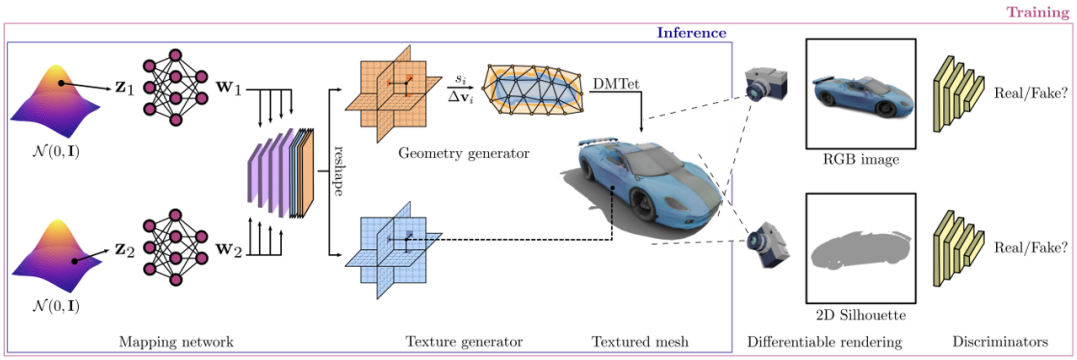

Figure 4: The model of the GET3D paper from the GET3D paper [1]

The model starts by sampling from a normal distribution to acquire 2 random vectors z_1 and z_2. These vectors are then passed through MLPs to obtain 2 new vectors of the same size, w_1 and w_2. Here both the geometry and texture generators share the same backbone network. This backbone consists of convolutions conditioned on w_1 and w_2. The result is both a geometry and texture triplane capable of producing both 3D shapes and their corresponding textures. A key objective of the paper was to achieve a disentanglement between the texture and the shape, this is why there are 2 generators instead of 1. To produce a shape we project a potential point of the object onto the geometry triplane and then use conditioned fully connected layers to obtain a signed point field function capable of building the 3D shape. The process is similar for textures, but instead we sample a point in 3D space and project it onto the texture triplane. Using the features of the triplane we can again use conditioned fully connected layers to obtain an RGB value for this part of the object, resulting in a fully textured 3D shape. For the purpose of inference we can stop now, however when training the model, a lost must be calculated to back-propagate the errors and optimise the model. To train the generators on 2D images, we have to convert the 3D shape to 2D. To do this the paper simulates virtual cameras taking pictures of the object and uses the generated 2D images to train the generators through the use of discriminators, a standard practice in GANs [15]. As the whole network is differentiable, back-propagation can be used to train the network.

2.5 Results from the paper

Category	**Method**	**COV (%, ↑)**	**MMD (↓)**	**FID (↓)**			
		LFD	CD	LFD	CD	Ori 3D	
Car	PointFlow	51.91	57.16	1971	0.82	–	–
	OccNet	27.29	42.63	1717	**0.61**	–	–
	Pi-GAN	0.82	0.55	6626	25.54	52.82	104.29
	GRAF	1.57	1.57	6012	10.63	49.95	52.85
	EG3D	60.16	49.52	1527	0.72	15.52	21.89
	Ours	**66.78**	**58.39**	**1491**	0.71	**10.25**	**10.25**
	Ours+Subdiv.	62.48	55.93	1553	0.72	12.14	12.14
	Ours (improved G)	59.00	47.95	1473	0.81	10.60	10.60

Table 2: Performance comparison of different methods for 3D car generation from [1]. Coverage (COV) represents the diversity of generated shape. Minimum Matching Distance (MMD) measures the quality of individual shapes. Fréchet Inception Distance (FID) measures the quality of generated textures Higher COV is better (↑), while lower MMD and FID are better (↓).

Table 2 presents the quantitative results of the paper for a specific type of object, 3D car shapes. In bold we can see the best performing model for each metric. From the table we can see that the GET3D model generally performs better than previous methods. In a visual field such

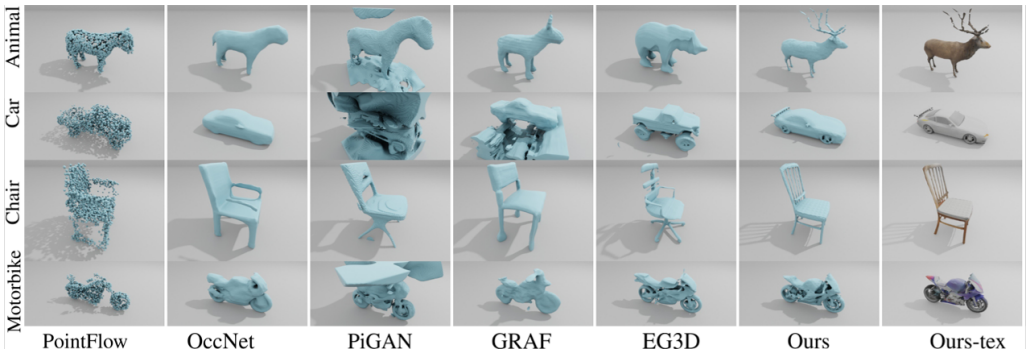

Figure 5: The qualitative results of the GET3D model in comparison to previous methods for a variety of selected objects from [1]. The second last columns presents the GET3D model's geometry output without any texture whilst the last column adds the generated texture.

as computer graphics it is often beneficial to examine the qualitative results of the model to get a better grasp of the performance of the model in comparison to previous methods. This qualitative illustration is presented in Figure 5.

Whilst these quantitative results are important in confirming the superior performance of the GET3D model they are difficult to interpret. We can see that the GET3D model generally performs better than previous methods both quantitively and qualitatively. Looking at the generated shapes we can see some of the downsides of previous methods. The PointFlow [9] method, for example, generates point clouds which do not properly capture the structure of these 3D objects and has numerous gaps. The other previous methods suffer from a lack of geometric detail which in comparison to the GET3D method is evidently inferior. This superior geometric detail combined with its generated texture, which the other methods lack, make the GET3D model a more efficient and convenient model to use for anyone wishing to create 3D worlds populated with 3D models.

While the GET3D model can generate both models and textures, a key strength of the GET3D model is the fact that the geometry and texture are separate and not intrinsically linked. This allows for adjusting the geometry without modifying the texture and vice versa.

Figure 6: A visualisation of the separation of the texture and geometry generated from [1]. Here the rows show a constant geometry and varying texture whilst the columns show a constant texture applied to varied geometry

This strength is shown in Figure 6. Here we can see various objects and textures being adjusted whilst keeping the other constant. From a users perspective this is handy as if, for example, a generated shape was a little bit off the desired outcome but the texture was correct, we could simply adjust the shape whilst leaving the texture unchanged and vice versa.

3 Discussion and Critique

We have introduced and examined the utility of the GET3D model in this report and will now discuss both the strengths and weaknesses of the paper. From a practical stand point, the model is successful in achieving many of its aims in building a scalable, functional model capable of seamlessly integrating into many industries with the least amount of overhead possible whilst still achieving impressive results. This is accomplished through the clever use of several generative AI approaches refining the shortcomings of previous methods. The paper itself is well written with lots of detail allowing readers to delve into the technical aspects of their model. To this end, there is a large appendix covering the architecture of the model, the datasets used and metrics used to evaluate the models. The main paper provided the essential information needed to grasp the fundamentals of the GET3D model. Separating the main bulk of the contributions and the excessive technical appendix kept the structure and flow of the paper easy to follow.

The paper also provides their own insights into the limitations and potential downsides of their model. The model could only be trained on synthetic datasets as the process of converting 3D shapes into 2D ones requires knowing the camera distribution, something which would be much more difficult for real objects captured with a real camera. This limit could be an interesting future direction of work to take the model. The GET3D model is also currently trained for a relatively small selection of categories (for example cars, motorcycles, humans etc...). An extension of the GET3D model to be trained with more categories could be another direction of future work.

The paper also examines how their model could be impacted by biases as most machine learning models are. The authors mention potential problems with bias in generating human objects due to such biases existing in the training datasets. The paper claims that GET3D should not be used in applications with a potential harmful impact on people and encourages users of the model to take the time to de-bias their datasets before using them on GET3D.

A criticism of the paper is that after the original paper was submitted to NeurIPS [3], the authors made some adjustments to the architecture of the model. This change was significant and explored in depth in the appendix of the paper. However, the figures used at times in the original paper were of the new architecture whilst they were describing the old architecture. This mix up led to the paper being confusing at times to follow and difficult to understand fully.

Another potential downside of the paper is its broader impact on society. Whilst we described how labour intensive the creation of 3D worlds were earlier in the beginning of this report, this also means that many jobs and livelihoods are dependant on this process. If models such as GET3D become more widespread we could see this entire industry being wiped out along with the many jobs dependant on it. Whilst this applies to many of the downstream applications of machine learning technologies, it is still worthwhile to consider the impact these models we create have on society.

To conclude, the paper is an impressive advancement in the field of 3D model generation which meets many of the practical needs of end users, allowing easy deployment and wide spread usage. From this perspective the paper was very successful in its aim of furthering this field. Whilst there were some issues with the paper such as a rather confusing post submission update it is still an excellent example of world class research which should inspire other researchers by epitomizing how science is done.

4 Disclosure of Funding

Adam Tlemsani is supported by UK Research and Innovation [UKRI AI Centre for Doctoral Training in Digital Healthcare grant number EP/Y030974/1].

References

[1] Jun Gao et al. "Get3d: A generative model of high quality 3d textured shapes learned from images". In: *Advances In Neural Information Processing Systems* 35 (2022), pp. 31841–31854.

[2] NVIDIA Corporation. *NVIDIA Corporation*. `https://www.nvidia.com`. Accessed: 2025-04-03. 2025.

[3] *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*. New Orleans, USA: NeurIPS, 2022. URL: `https://neurips.cc/`.

[4] Oksana Dobrovolska. *White Lands Blender Demo File*. `https://download.blender.org/archive/gallery/blender-splash-screens/blender-3-2/`. Accessed: 2025-04-05. 2022.

[5] Blender Foundation. *Blender: 3D Creation Suite*. Accessed: 2025-02-28. 2025. URL: `https://www.blender.org/`.

[6] Piotr Krynski. *Blender 3.3 Splash Screen Demo File*. `https://www.blender.org/download/demo/splash/blender-3.3-splash.blend`. Accessed: 2025-04-05. 2022.

[7] Nicole Morena. *Blender 3.5 Splash Screen Demo File*. `https://www.blender.org/download/demo/splash/blender-3.5-splash.blend`. Accessed: 2025-04-05. 2023.

[8] Lars Mescheder et al. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4460–4470.

[9] Taesung Park et al. "Semantic image synthesis with spatially-adaptive normalization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2337–2346.

[10] Dario Pavllo et al. "Learning generative models of textured 3d meshes from real-world images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13879–13889.

[11] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[12] Eric R Chan et al. "Efficient geometry-aware 3d generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16123–16133.

[13] Brent Burley and Walt Disney Animation Studios. "Physically-based shading at disney". In: *Acm Siggraph*. Vol. 2012. 2012. vol. 2012. 2012, pp. 1–7.

[14] Katja Schwarz et al. "Graf: Generative radiance fields for 3d-aware image synthesis". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20154–20166.

[15] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.