DOI: 10.1111/cgf.14980
Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

Generating Parametric BRDFs from Natural Language Descriptions

Sean Memery **Osmar Cedron** **Kartic Subr**

The University of Edinburgh
Edinburgh, Scotland, UK

Abstract

Artistic authoring of 3D environments is a laborious enterprise that also requires skilled content creators. There have been impressive improvements in using machine learning to address different aspects of generating 3D content, such as generating meshes, arranging geometry, synthesizing textures, etc. In this paper we develop a model to generate Bidirectional Reflectance Distribution Functions (BRDFs) from descriptive textual prompts. BRDFs are four dimensional probability distributions that characterize the interaction of light with surface materials. They are either represented parametrically, or by tabulating the probability density associated with every pair of incident and outgoing angles. The former lends itself to artistic editing while the latter is used when measuring the appearance of real materials. Numerous works have focused on hypothesizing BRDF models from images of materials. We learn a mapping from textual descriptions of materials to parametric BRDFs. Our model is first trained using a semi-supervised approach before being tuned via an unsupervised scheme. Although our model is general, in this paper we specifically generate parameters for MDL materials, conditioned on natural language descriptions, within NVIDIA's Omniverse platform. This enables use cases such as real-time text prompts to change materials of objects in 3D environments such as "dull plastic" or "shiny iron". Since the output of our model is a parametric BRDF, rather than an image of the material, it may be used to render materials using any shape under arbitrarily specified viewing and lighting conditions.

CCS Concepts

• ***Computing methodologies*** → *Machine learning; Natural language processing; Computer graphics;*

1. Introduction

Crafting realistic materials for use in physically based rendering (PBR) is a difficult creative process. A highly specialised material designer is often needed for workflows that require accurate rendering of real world objects, e.g. VFX, video games, and visual learning simulations. However, there is a rising trend of user-led design through natural language, i.e. prompting. A user without technical design expertise may wish to simply describe their envisioned material to obtain a BRDF model that matches their description. Although tremendous progress has been made in implicit neural representation of scenes, explicit representation of materials by modeling BRDFs allows tuning and artistic control. In addition, the synthesized material can then be applied to general settings across a variety of geometries, viewing conditions and lighting distributions. In this paper, we address the problem of generation of traditional BRDF material shaders, specifically NVIDIA's Omniverse.

There are two typical representations for PBR materials. Highly accurate measured data can be found from real world objects, tabulating the proportion of light reflected along a set of reflected directions for each discretized incident direction. This format can be highly accurate but has many drawbacks, such as a large memory cost and being difficult to design and interpret. For usage in real-time applications, the BRDF can be estimated using parametric functions known as shaders. In NVIDIA's Omniverse, a recent 3D software platform, the Material Definition Language (MDL) format is used to write shader code specific to representing materials in real-time rendering. Within Omniverse there are multiple preset material definitions that are parametric functions defined by a series of values representing different physical properties.

Often, for a 3D scene to be created by a designer, a prototype scene is made with fewer time and resources required. Databases of pre-made materials are utilised to speed up development. Our goal is to build a tool to aid in this task by allowing designers, or non-technical users, to describe a scene in text and receive suitable materials. We do this by utilising CLIP [Radford et al.(2021)], a multi-modal image and text embedding model. CLIP was trained to encourage embeddings of texts and images to lie close to one another in latent space. In doing so, CLIP provides a correlation rating between text and image pairs, the cosine similarity between the two embeddings, allowing it to be used as a semantic loss during training. With this, we train an autoencoder model to learn a latent space that matches BRDF parameters, conditioned on text embeddings. This architecture can be trained to predict parameters for any target BRDF function in common renderers. Although we demonstrate this with NVIDIA's MDL format, and within a game

Figure 1: ***BRDF Generation from Material Prompts****: Our model allows users to assign materials to objects in real-time scenes by simply describing their desired appearance. This image shows the Kitchen Set USD scene by Pixar, running in Unreal Engine, with materials assigned by typing a description into the text box in the lower left of the screen.*

engine (Unreal Engine), it can equally be applied within the context of other material formats and PBR renderers.

2. Related Work

2.1. BRDF Representation

As outlined in the survey paper "BRDF Representation and Acquisition" [Guarnera et al.(2016)], there are many ways to represent the BRDF. This is due to the many use cases of the function, and the varying complexity of the representations. Fundamentally, a BRDF is a function that takes as input an incoming light direction $\mathbf{v_i}$ and an outgoing direction $\mathbf{v_r}$, and produces the ratio of incoming irradiance to outgoing radiance. The equation is,

$$f_r(\mathbf{v_i}, \mathbf{v_r}) = \frac{dL_r(\mathbf{v_r})}{dE_i(\mathbf{v_i})} = \frac{dL_r(\mathbf{v_r})}{L_i(\mathbf{v_i})\cos\theta_\mathbf{i} d\mathbf{v_i}} \quad (1)$$

where E_i is the incident irradiance, L_i is incident radiance and L_r is the reflected radiance. For an incident ray colliding with a surface with normal n, θ_i is the measure of the angle between v_i and n. In practice however, much of the complexity of the function can be removed as light is simulated simply by measured RGB values, as done in the MERL dataset [Matusik et al.(2003)]. MERL stores BRDF values in a tabulated grid of ratios of incoming irradiance to outgoing radiance, used as a multiplier to incoming RGB values. In real-time applications, a more memory-efficient representation is needed, so the BRDF is approximated analytically by shader functions. This is code written in a shader language, such as GLSL [Khronos(2023)], to be run in parallel on the GPU. In NVIDIA Omniverse, there are many shader functions that can be used to simulate different material types, all represented in the Material Definition Language (MDL). OmniPBR is the default material, it can describe dielectric and non-dielectric materials, but lacks features such as sub-surface scattering. OmniSurface is a more intricate BRDF implementation that can represent a wider range of materials.

Neural representations of measured BRDFs have become more popular in recent years for many of the same reasons analytical BRDFs are popular [Sztrajman et al.(2021)] [Rainer et al.(2019)]. Dimensionality reduction of measured BRDFs [Tongbuasirilai et al.(2022)] [Chen et al.(2020)] [Zheng et al.(2022)] is particularly useful as it brings many desirable attributes such as lower memory usage, faster parsing, and more human readable values. Similarly, learning a latent space of dimensionality reduced representations is useful for design and material interpolation [Benamira et al.(2022)] [Hu et al.(2020)].

Another longstanding challenge in graphics programming has been BRDF estimation from other modalities. The most common form of this is to estimate BRDF values from images of real world materials. These images often consists of single photographs, possibly captured with flash. To achieve a successful estimation, it is necessary to render a visually similar image by generating normal, roughness, and metallic maps. Methods face numerous obstacles, including complications arising from camera flash and the accurate representation of fine details [Rhee and Lee(2022)] [Boss and Lensch(2019)] [Otani and Komuro(2021)]. [Zhou et al.(2023)] introduces a material generator learned from single images of surfaces.

2.2. Generation from Text

With the coming of more effective language modelling methods, there has been an explosion of work on generating 3D content from text. Many of this has been directly inspired from other modalities, such as diffusion methods as used in 2D image generation, applied to 3D models [Huang et al.(2023)]. Other work has focused on general scene generation, either unconditioned [Anciukevičius et al.(2022)] or conditioned on user data, such as object positioning

for example [Po and Wetzstein(2023)]. The majority of these methods focus on implicit object generation, that is predicting objects whose shape and appearance are contained in the neural model that generates them. This is best shown by the popularity of Neural Radiance Fields [Mildenhall et al.(2020)] [Lin et al.(2023)]. NeRFs are neural representations of scenes that are entirely contained in the network, with no explicit graphics resources required. There has been a small amount of work done on explicit generation from text, where the generated content is separate from the model that created it [Khalid et al.(2022)] [Chen et al.(2023a)]. In particular there is Point-E [Nichol et al.(2022)], a 3D point cloud generation technique that is conditioned on text prompts. It utilises text to 2D image generation and image to point cloud diffusion, while being guided by text prompts. There is a focus on efficiency over accuracy in this work, showing promise for an eventual real-time application. There is also Fantasia3D [Chen et al.(2023b)], a recent work that learns to predict geometry and appearance from text prompts, that can generate multiple texture maps of a small set of microfacet BRDF parameters. The results are a highly detailed Spatially-Varying BRDF (SVBRDF) that is trained for each material prompt.

Recent years have seen a rise in work relating text and visuals, mirroring the rise in popularity of generative text models. [Wu et al.(2020)] present an early work in generating descriptions and gives a thorough study of the available methods. There has also been multiple concurrent works that generate materials conditioned on text prompts. [Hu et al.(2023)] utilise an extensive material dataset to generate high quality material node graphs using an autoregressive model conditioned on a CLIP text or image embedding. [Deschaintre et al.(2023)] use a large dataset of fabric materials and descriptions to fine-tune CLIP for material retrieval. Both works make use of CLIP, showing a growing trend within data-driven material generation.

2.3. Stylising from Text

A popular use of natural language conditioning on 3D content is stylisation. This is the editing of 3D content to match a given style, as described in a text. A popular recent method for this is the use of CLIP embeddings as a semantic loss during training. [Michel et al.(2022)] propose Text2Mesh, a model for estimating vertex colour and displacement values in a given mesh, so as to fit a style described in text. While this is effective at the chosen task, the edited meshes still lack many modern graphics components. [Chen et al.(2022)] improve on the visualisation of Text2Mesh by estimating a wider range of material properties, to achieve a more realistic rendering of the given mesh. Both of these works however represent their graphics content implicitly within their models, making their methods difficult to integrate into traditional workflows. TANGO proposes disentangling their neural representation and exporting it as explicit graphics resources, which is effective but not efficient enough to be feasible for large amounts of data.

3. Method

To develop a real-time material generation system, several components are required. First is an efficient physically based renderer that can provide accurate images of generated materials, this is NVIDIA Omniverse. Second is a method of evaluating the generated images against the input text, this is the semantic loss from CLIP. Third is a source of training data, in this case text and material pairs. Lastly is a machine learning model that can make material parameter predictions from the CLIP text embedding inputs. In our system, this is a simple fully connected autoencoder model.

3.1. Data

Training the model requires many samples of text that materials can be generated from. There were two sources of this data, the first is from several popular websites of collections of materials [FreePBR(2023)] [Textures(2023)]. We simply found the 100 most common nouns and 25 most common adjectives from the material names across each website, with table 1 displaying the 10 most common of each. We combined these these to create a dataset of material prompts with either 0 or 1 adjectives followed by a noun.

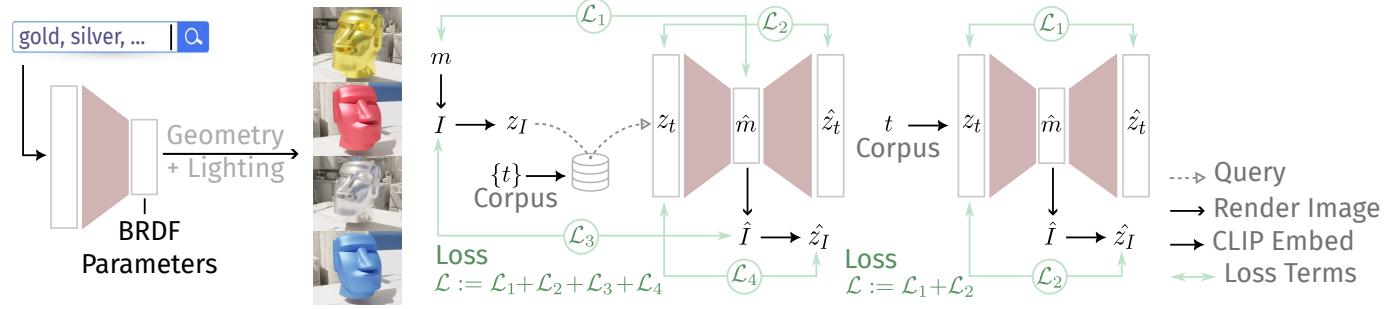
The second source of material prompts is from several publicly available novels, all nouns and associated adjectives are parsed using part of speech tagging with the NLTK library [Bird and Loper(2004)]. The parsed descriptions are then screened using concreteness ratings available online [Brysbaert et al.(2013)], where descriptions are only saved if their noun is found to have a particularly high concreteness rating (at least 4.5 from a maximum of 5). This method can provide realistic material prompts with multiple adjectives, rather than using random combinations of adjectives from source one, which may contradict each other e.g. "wet, dry rock".

Nouns	Adjectives
wood	metallic
metal	shiny
brick	smooth
concrete	polished
stone	rough
glass	matte
marble	reflective
gold	glossy
plastic	dull
leather	dark

Table 1: *10 most common material nouns and adjectives.*

3.2. Training

There are two methods for training the autoencoder model, based on the two sources of material prompts, semi-supervised and unsupervised. The semi-supervised training procedure requires pairs of material and material descriptions. To get these, we utilise CLIP once more to annotate rendered images of randomly generated materials. To do this we generate a random vector of material parameters m, and render an image I of a sphere with this material applied to it. A CLIP image embedding z_I is found from the material render, and a vector database of CLIP text embeddings is queried. This vector database is made from the material prompts created from common nouns and adjectives, described in 3.1. With this method we can find the material prompt embedding z_T that best fits the material render. These values make up a single semi-supervised input for the model.



(a) Example usage of our model (b) Initial, semi-supervised training (c) Unsupervised training

Figure 2: *Overview of our training process, both semi-supervised and unsupervised learning.*

The material prompt embedding z_T is given as input to the autoencoder model. A vector of material parameters $\hat{m}$ is found as the model's latent representation of the input embedding. The output of the model is the reconstructed text embedding $\hat{z}_T$. During training, the predicted vector of material parameters $\hat{m}$ is rendered in the same environment as the target material to get the reconstructed material image $\hat{I}$, which is then given to CLIP to calculate the final image embedding $\hat{z}_I$. Finally, all values have been calculated for the semi-supervised loss function:

$$L = ||\hat{m}, m||_1 + ||\hat{z}_T, z_T||_1 + ||\hat{z}_I, z_I||_1 + CLIP_{sim}(z_T, \hat{z}_I, z_I) \quad (2)$$

Where the $CLIP_{sim}$ function is one minus the difference between the cosine similarities of the prediction and target embeddings:

$$CLIP_{sim}(z_T, \hat{z}_I, z_I) = 1 - (sim(z_T, \hat{z}_I) - sim(z_T, z_I)) \quad (3)$$

The model is then further trained in an unsupervised manner. Using the text from novels and concreteness screening, a material prompt is chosen randomly. This prompt is embedded by CLIP, as z_T, and given to the model as input. Identical to the semi-supervised training, a vector of material parameters $\hat{m}$ and a reconstructed text embedding $\hat{z}_T$ are outputted by the model. The CLIP image embedding of the rendered material, $\hat{z}_I$, is also generated as in the semi-supervised training. These are the values used to calculate the unsupervised loss:

$$L = ||\hat{z}_T, z_T||_1 + sim(z_T, \hat{z}_I) \quad (4)$$

3.3. Material Post-Processing

Along with the material parameters found within the latent representation, there are also others which were chosen to be predicted in another way, due to nature of the visual learning task. For example, though it may seem natural to predict the diffuse colour of the material along with the other material parameters, we found that this caused the model to ignore other visual features and only attempt to predict colour. This is likely due to the larger influence colour has than other material parameters on the final rendered image. As a simple solution to this, we predict colour, along with transparency and refractive index, by querying a vector database of text and value pairs, to find the most similar text to the material prompt and assign the paired value. A vector database is a collection of text embeddings, each with saved values, that allows querying based on a nearest neighbour search i.e. the nearest embeddings to a search embedding are found and their saved values retrieved. We use the open source library Chroma [Chroma(2023)] for this project.

To collect the text-value pairs, we used multiple text-colour sources [Wikipedia contributors(2023)] [XKCD(2010)], refractive index tables [Wikipedia contributors(2022)], and transparency values from material examples [Nvidia(2023)]. Importantly, only a relatively small amount of data is needed for this task as a weighted sum of values can be calculated using the text embedding distances found during querying. This allows diversity within the queries without needing extensive databases of values.

3.4. Implementation

In our implementation, the autoencoder model has 5 encoder layers and 5 decoder layers. The first layer has a size of 512, matching the dimension of the CLIP text embedding. Each subsequent layer is half the size of the previous, until the fifth layer with a size of 32. The decoder layers mirror those in the encoder. The latent space is the size of the chosen BRDF parameters, in this case 8 for the parameters of the OmniSurface function in NVIDIA's MDL format. Every layer is fed into a ReLU connection and the final decoder layer feeds into a sigmoid function. The model was trained with the Adam optimizer and a learning rate of 1e-4, for several thousand training steps. Our implementation was trained using an RTX 4090 with 24GBs of VRAM, enabling the training of the model and the rendering of images to be done simultaneously. We rendered images using NVIDIA Omniverse's Python API which was called within a Pytorch training loop. Renders used in training were of a simple scene of a sphere with the current material applied and a bright environment map. The images were cropped to remove all visuals but the sphere, to allow CLIP to focus on the material.

4. Evaluation

4.1. Survey

We evaluated our model via a survey of 125 users on Amazon Mechanical Turk who indicate pairwise preference for a given prompt. As a comparison to images rendered with predictions made by our model, we used the Base Materials pack available on NVIDIA Omniverse. This is a database of common materials such as "Beige Carpet", "Concrete Rough", "Brick", etc. They are created using the

"OmniPBR" MDL function to estimate the BRDF, this is a default function of Omniverse and capable of realistically representing diverse materials. Our survey was presented as a series of pairs of images with a single text input which we call the material prompt. We constructed a fixed set of 250 possible prompts by combining the 25 most common nouns and 10 most common adjectives in our text corpus. The target material from the Base Materials pack was chosen as the material whose name is found to be closest to the prompt, such as "Concrete Rough" for the prompt "Concrete". This is in the spirit of what a designer might choose given a material description. The second image in the pair is a render of an identical scene, but with a material generated from our model with the material prompt as input. The survey-taker either chooses their preferred image for each prompt or may click "Unsure" if they are unable to decide. Pairs were shown in random order and sometimes repeated, after flipping, within a survey.

Figure 3 summarises the results of our survey. It shows that on average, users were unable to distinguish BRDFs generated by our model from BRDFs in the database. However, the model seemed to perform better for some materials (such as chrome) than others (such as gold). Figure 4 shows the histogram of response times by users when they chose images rendered with our materials (green) vs those rendered with images from the Base material database. The summary of these results is that materials automatically generated by our model were indistinguishable from those from the manually-crafted database associated with a specific prompt.

4.2. Example Applications

To demonstrate practical use of the model, we developed applications for two possible use cases. The first is an Unreal Engine integration allowing a player to type a material prompt and simply click to select an object in the world to apply a predicted material onto. Screenshots of this are shown in figure 5, with multiple example prompts. It's important to note that due to the size of the model, inference can be extremely fast. Encoding a text embedding to a material vector takes just 5ms in our tests, allowing our Unreal Engine demo to set materials in real-time.

Our second demo is a system for initialising a scene conditioned on a paragraph description. Our application parses the paragraph of text and extracts the nouns and their associated adjectives. These are then screened for concreteness using publicly available concreteness ratings of words [Brysbaert et al.(2013)]. The final material prompts are then inputted to the model and a collection of material vectors are generated. These vectors of parameters are then used to edit a default MDL file, with the target BRDF function, to export the materials as files. These material files can then be used within software such as NVIDIA Omniverse to initialise a scene. Figure 6 shows a visualisation of the process.

4.3. Word2Vec Comparison

To evaluate the material latent space learned by our model, we compare the distribution of samples with two metrics: LPIPS [Zhang et al.(2018)] distance metric, and Word2Vec [Mikolov et al.(2013)] distance. LPIPS is a similarity metric between images that utilises the final layers of deep neural networks, we use it to find the similarity between predicted material samples and the Word2Vec neighbours of their material prompts. To study these distributions, we get the 10 most common adjectives from our semi-supervised dataset and find the 10 nearest neighbours of their Word2Vec representations. Note that we remove the antonyms from the set of neighbours as Word2Vec is known to group words with inverse meanings. We use these neighbours, along with the 25 most common nouns from our dataset, to create material prompts for our model and get the LPIPS distance between rendered images of the neighbour prompt materials and starting prompts.

In figure 7, we plot the Word2Vec distance of each material prompt to its starting prompt compared to its LPIPS distance to the starting prompt. This plot compares distances of fully coloured materials, as described in section 3.3, which is likely over represented in the LPIPS similarity metric. To study the effects of only the material parameters on the correlation of distances, figure 8 shows the same experiment but all predicted materials use the same grey base colour. This forces LPIPS to only vary based on the material parameters and not colour.

4.4. Interpolation

Figures 9 and 10 show our study of the interpolation of our model's BRDF latent space. Both plots are generated from 1000 samples.

In figure 9, two pairs of ground truth material and text are selected, then two predicted materials are generated using the ground truth texts. The vectors of material parameters are then interpolated, between the first ground truth material and the second, and between the first predicted material and the second. Finally, the LPIPS distance between rendered images of the two interpolated materials are plotted.

In figure 10, two ground truth material and text pairs are also selected, with predicted materials generated similarly. In this case, only the predicted materials are interpolated, between the first prediction and the second. The LPIPS distance between the interpolated material and the first ground truth material is plotted in blue, and the distance to the second ground truth material is plotted in orange.

5. Discussion

5.1. Word2Vec Comparison

Figures 7 and 8 show the results of our study of the latent space of our model. The correlation in the first figure is near zero, with a high p-value, implying there is no relation between Word2Vec distance and LPIPS distance, however this is only for the full colour renders of materials. Figure 8 compares the two metrics on renders of materials with a grey base colour, and shows a positive correlation coefficient of 0.354 with a low p-value of 0.027. This suggests that an increase in the Word2Vec distance between two prompts relates to an increase in the LPIPS distance between the rendered images of the prompts predicted materials. As we remove antonyms from the neighbour set, this result implies that the predictions made by our model become visually dissimilar as the material prompts' meanings become more dissimilar. As this correlation is present

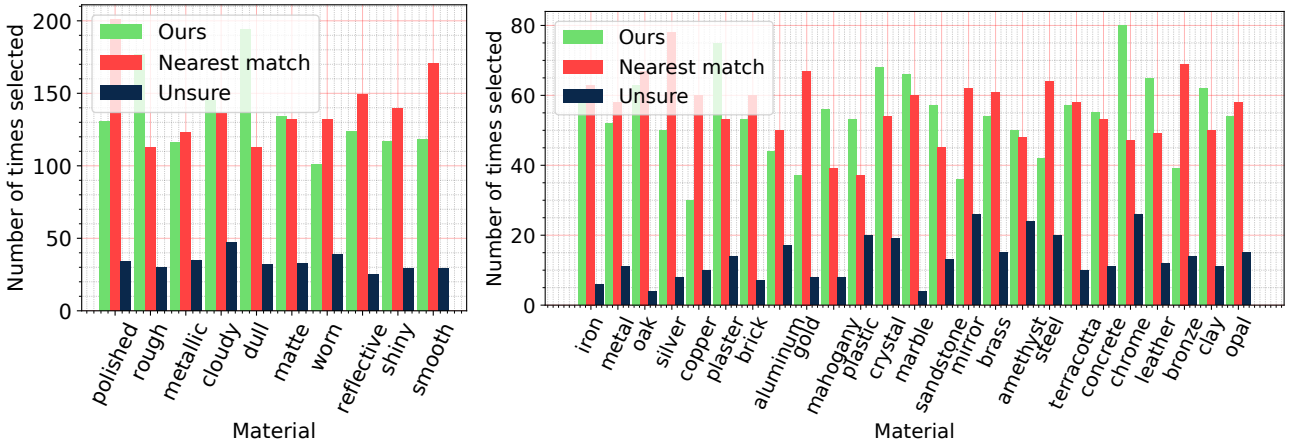


Figure 3: *Results from our survey. The plots show the number of times users chose an image rendered with our (green) BRDF against an image rendered using a BRDF from NVIDIA's Omniverse database, for a given text prompt. The plots show users' preferences across adjectives (a) and nouns (b).*

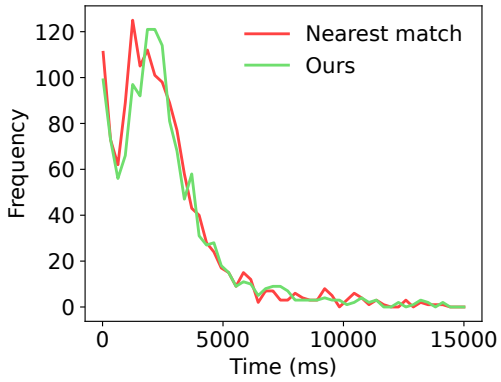


Figure 4: *Histograms of user response times in the survey when selecting our BRDF (green) vs a material from the database (red). The histograms are virtually indistinguishable. The slightly larger mode for our histogram suggests a subtle trend showing that users who took longer could have had a preference for our BRDFs.*

only when materials lack post-processing (i.e. base colour), it suggests that our model follows a similar distribution to Word2Vec, but that LPIPS is highly sensitive to material colour.

5.2. Interpolation

Our two interpolation results have two different conclusions. Figure 9 shows the interpolation of predicted and ground truth materials. We see a significant decrease in the distance between the two as interpolation occurs, before returning to the initial distance. This is to be expected within a smooth space as for both interpolations the beginning and end states are "near" each other, as our model is trained to make nearby predictions. Therefore, on average, the interpolated materials will approach each other as interpolation increases, before diverging again as interpolation finishes.

The second plot, figure 10, has a very clear interpretation. Predictions begin near their ground truth targets (the blue line begins low) before finishing far from their targets after being interpolated to another material (the blue line ends high). The inverse of this is shown in the orange line which lowers as the interpolation approaches the orange line's ground truth material.

Both of these results imply a smooth material representation that acts predictably under interpolation.

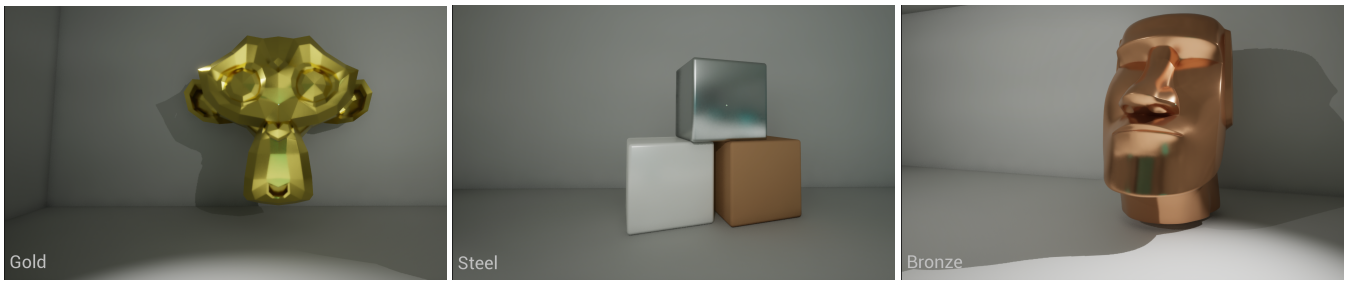
6. Limitations

6.1. Textual Prompt Space

Although the text prompts (adjective(s) + noun) used to generate materials are relatively simple, they are actually not trivial to work with. In fact, having multiple adjectives within the same text prompt can lead to incongruent results due to contradictory or even vague adjectives. Furthermore, CLIP model struggles to annotate randomly generated materials with text prompts possessing multiple adjectives. To overcome these problems, a two-phase approach was implemented, starting with a semi-supervised training with single adjectives followed by an unsupervised training on text prompts with multiple adjectives. Richer textual prompts were out of the scope of this paper due to the small amount of labelled data (material/prompt) available and the difficulty to obtain high quality materials.

6.2. Learning BRDF via Abstract Material Descriptions

Since the focus of the encoder is not to learn a neural representation of a BRDF but to learn the parameters of various materials instead, using an abstract material description such as MDL for describing properties (surface, physically based, glass-with-volume, etc.) was a straightforward choice. Moreover, MDL is a shading language that does not produce programs for a particular renderer but rather it defines the behaviour of light at a higher level, which makes it

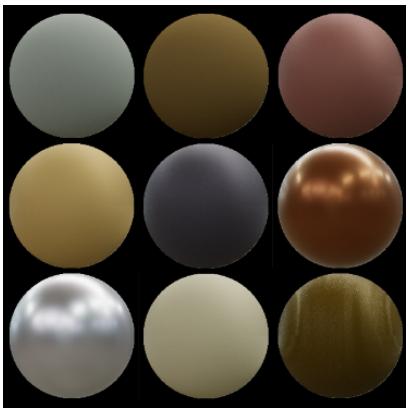


(a) *Material Prompt:* ***Gold*** **(b)** *Material Prompt:* ***Steel, Plastic, Brick*** **(c)** *Material Prompt:* ***Bronze***

Figure 5: *Examples from the Unreal Engine demo of our model.*

"The castle stood upon the hill. Its walls were of rough, grey stone, sturdy and strong. Big wooden doors, bound with iron bands, stood at the entrance. Up above, granite statues decorated the rooftops. Inside, there was more wood and stone, along with metals like bronze and silver. The castle was a mix of textures, from the cool, hard stone to the warm, grainy wood."

"rough, grey stone"
"wooden doors"
"granite statues"
"wood"
"stone"
"bronze"
"silver"
"cool, hard stone"
"warm, grainy wood"



(a) *Scene description.* **(b)** *Parsed material prompts.* **(c)** *Collection of generated materials.*

Figure 6: *An example scene initialisation utilising our model.*

a suitable material description for working with different renderers and under different settings (end-user applications or within more complex workflows). However, it is important to take into account the practical implications of the chosen material description, e.g. rendering time, complexity and portability, as they can easily hinder training and applications of the generated materials. As an anecdote, we tried to use our method with PBRT material descriptions, but the high rendering time became a bottleneck while training and ultimately had to be abandoned.

6.3. Material Post-Processing

Material generation involves learning complex, orthogonal concepts/spaces such as material properties, colour, texture map, normal map, displacement maps, etc. Whether we should frame this problem as a group of independent learning tasks or rather as a combined single task is unclear, the literature has adopted different approaches based on the scope, resources and limitations of the project. Besides, since the project is aimed to end-users, it is important to take into account how many different interpretations of a prompt are possible and therefore, it is quite likely that properties such as colour, transparency and refractive index of the generated materials will be further tuned to fit user needs.

7. Conclusion

In this paper, we presented a method for training an autoencoder model to learn a latent space of BRDF parameters. The model can be implemented to predict parameters for any parametric BRDF implementation, with our work utilising the MDL functions available in NVIDIA Omniverse for training data and evaluation. Our demos show the practical use of such a model, and the capability of real-time generation in Unreal Engine. We studied how the latent space of our model represents BRDF parameters through interpolation experiments and comparisons with the word distribution in Word2Vec. We believe the results of our evaluations show an effectiveness in our training method, that a semantic loss for text and rendered images is capable of learning subtle parameters such as material values. However, our survey shows the extent of this method, as it was found to be indistinguishable but not surpassing database retrieval.

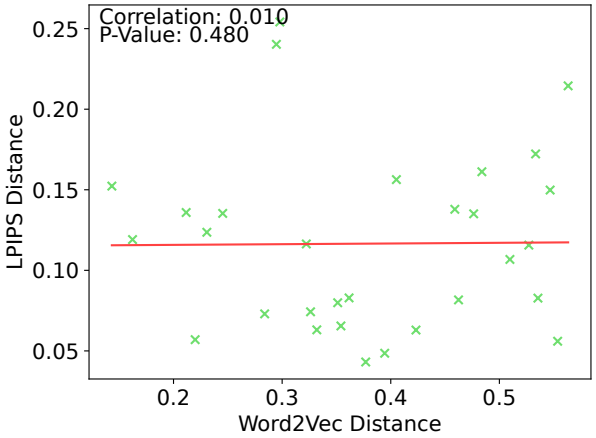


Figure 7: *Correlation of LPIPS distance and Word2Vec distance, for full colour images.*

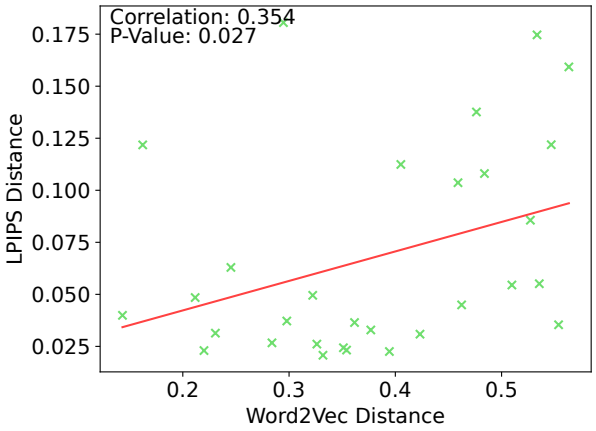


Figure 8: *Correlation of LPIPS distance and Word2Vec distance, for grey images.*

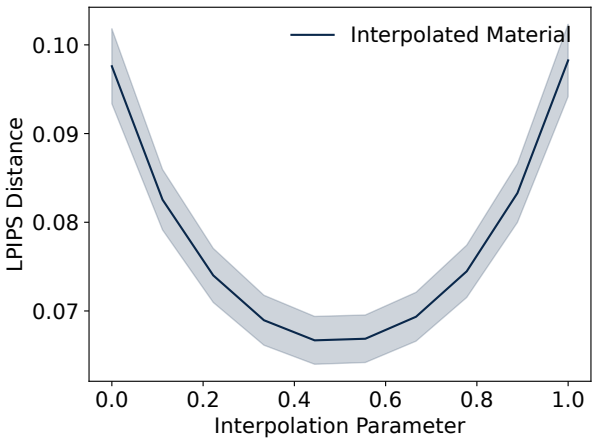


Figure 9: *LPIPS distance between the interpolations of predicted and ground truth materials.*

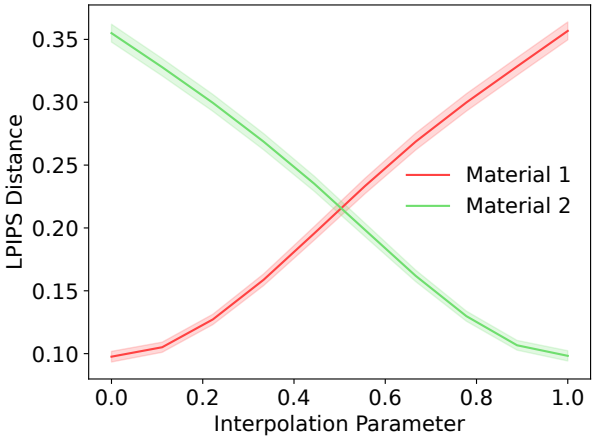


Figure 10: *Interpolation of predicted materials and LPIPS distance to fixed ground truths.*

References

[Anciukevičius et al.(2022)] Titas Anciukevičius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J. Mitra, and Paul Guerrero. 2022. RenderDiffusion: Image Diffusion for 3D Reconstruction, Inpainting and Generation. arXiv:2211.09869 [cs.CV] 2

[Benamira et al.(2022)] Alexis Benamira, Sachin Shah, and Sumanta Pattanaik. 2022. Interpretable Disentangled Parametrization of Measured BRDF with β-VAE. arXiv:2208.03914 [cs] http://arxiv.org/abs/2208.03914 version: 1. 2

[Bird and Loper(2004)] Steven Bird and Edward Loper. 2004. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Barcelona, Spain, 214–217. https://aclanthology.org/P04-3031 3

[Boss and Lensch(2019)] Mark Boss and Hendrik P. A. Lensch. 2019. Single Image BRDF Parameter Estimation with a Conditional Adversarial Network. arXiv:1910.05148 [cs, eess] http://arxiv.org/abs/1910.05148 version: 1. 2

[Brysbaert et al.(2013)] Marc Brysbaert, Amy Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods* 46 (10 2013). https://doi.org/10.3758/s13428-013-0403-5 3, 5

[Chen et al.(2023a)] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023a. Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation. arXiv:2303.13873 [cs] http://arxiv.org/abs/2303.13873 3

[Chen et al.(2023b)] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023b. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873* (2023). 3

[Chen et al.(2022)] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. 2022. TANGO: Text-driven Photorealistic and Robust 3D Stylization via Lighting Decomposition. arXiv:2210.11277 [cs] http://arxiv.org/abs/2210.11277 3

[Chen et al.(2020)] Zhe Chen, Shohei Nobuhara, and Ko Nishino. 2020. Invertible Neural BRDF for Object Inverse Rendering. (2020), 17. 2

[Chroma(2023)] Chroma. 2023. Chroma. https://github.com/chroma-core/chroma. 4

[Deschaintre et al.(2023)] Valentin Deschaintre, Julia Guerrero-Viu, Diego Gutierrez, Tamy Boubekeur, and Belen Masia. 2023. The Visual Language of Fabrics. arXiv:2307.13681 [cs.GR] 3

[FreePBR(2023)] FreePBR. 2023. FreePBR. https://freepbr.com/. 3

[Guarnera et al.(2016)] D. Guarnera, G.C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. 2016. BRDF Representation and Acquisition. *Computer Graphics Forum* 35, 2 (2016), 625–650. https://doi.org/10.1111/cgf.12867 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12867 2

[Hu et al.(2020)] Bingyang Hu, Jie Guo, Yanjun Chen, Mengtian Li, and Yanwen Guo. 2020. DeepBRDF: A Deep Representation for Manipulating Measured BRDF. 39, 2 (2020), 157–166. https://doi.org/10.1111/cgf.13920 2

[Hu et al.(2023)] Yiwei Hu, Paul Guerrero, Milos Hasan, Holly Rushmeier, and Valentin Deschaintre. 2023. Generating Procedural Materials from Text or Image Prompts. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. ACM. https://doi.org/10.1145/3588432.3591520 3

[Huang et al.(2023)] Rui Huang, Xuran Pan, Henry Zheng, Haojun Jiang, Zhifeng Xie, Shiji Song, and Gao Huang. 2023. Joint Representation Learning for Text and 3D Point Cloud. arXiv:2301.07584 [cs] http://arxiv.org/abs/2301.07584 version: 1. 2

[Khalid et al.(2022)] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. 2022. CLIP-Mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 Conference Papers* (2022-11-29). 1–8. https://doi.org/10.1145/3550469.3555392 arXiv:2203.13333 [cs] 3

[Khronos(2023)] Khronos. 2023. Core Language (GLSL). https://www.khronos.org/opengl/wiki/Core_Language_(GLSL) 2

[Lin et al.(2023)] Yiqi Lin, Haotian Bai, Sijia Li, Haonan Lu, Xiaodong Lin, Hui Xiong, and Lin Wang. 2023. CompoNeRF: Text-guided Multi-object Compositional NeRF with Editable 3D Scene Layout. arXiv:2303.13843 [cs] http://arxiv.org/abs/2303.13843 3

[Matusik et al.(2003)] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. 2003. A Data-Driven Reflectance Model. *ACM Transactions on Graphics* 22, 3 (July 2003), 759–769. 2

[Michel et al.(2022)] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. 2022. Text2Mesh: Text-Driven Neural Stylization for Meshes. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (New Orleans, LA, USA, 2022-06). IEEE, 13482–13492. https://doi.org/10.1109/CVPR52688.2022.01313 3

[Mikolov et al.(2013)] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013). 5

[Mildenhall et al.(2020)] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *CoRR* abs/2003.08934 (2020). arXiv:2003.08934 https://arxiv.org/abs/2003.08934 3

[Nichol et al.(2022)] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-E: A System for Generating 3D Point Clouds from Complex Prompts. arXiv:2212.08751 [cs.CV] 3

[Nvidia(2023)] Nvidia. 2023. Vmaterials 2 Pack. https://enterprise.launcher.omniverse.nvidia.com/exchange/content-pack/ov-vmats2pack-01. 4

[Otani and Komuro(2021)] Haru Otani and Takashi Komuro. 2021. BRDF Measurement of Real Materials Using Handheld Cameras. In *Advances in Visual Computing* (Cham, 2021) *(Lecture Notes in Computer Science)*, George Bebis, Vassilis Athitsos, Tong Yan, Manfred Lau, Frederick Li, Conglei Shi, Xiaoru Yuan, Christos Mousas, and Gerd Bruder (Eds.). Springer International Publishing, 65–77. https://doi.org/10.1007/978-3-030-90439-5_6 2

[Po and Wetzstein(2023)] Ryan Po and Gordon Wetzstein. 2023. Compositional 3D Scene Generation using Locally Conditioned Diffusion. arXiv:2303.12218 [cs] http://arxiv.org/abs/2303.12218 3

[Radford et al.(2021)] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. *CoRR* abs/2103.00020 (2021). arXiv:2103.00020 https://arxiv.org/abs/2103.00020 1

[Rainer et al.(2019)] Gilles Rainer, Wenzel Jakob, Abhijeet Ghosh, and Tim Weyrich. 2019. Neural BTF Compression and Interpolation. 38, 2 (2019), 235–244. https://doi.org/10.1111/cgf.13633 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13633. 2

[Rhee and Lee(2022)] Chi-Hyoung Rhee and Chang Ha Lee. 2022. Estimating Physically-Based Reflectance Parameters From a Single Image With GAN-Guided CNN. 10 (2022), 13259–13269. https://doi.org/10.1109/ACCESS.2022.3147483 Conference Name: IEEE Access. 2

[Sztrajman et al.(2021)] Alejandro Sztrajman, Gilles Rainer, Tobias Ritschel, and Tim Weyrich. 2021. Neural BRDF Representation and Importance Sampling. arXiv:2102.05963 [cs] http://arxiv.org/abs/2102.05963 2

[Textures(2023)] Textures. 2023. Textures. https://www.textures.com/browse/pbr-materials/114558. 3

[Tongbuasirilai et al.(2022)] Tanaboon Tongbuasirilai, Jonas Unger, Christine Guillemot, and Ehsan Miandji. 2022. A Sparse Non-parametric BRDF Model. 41, 5 (2022), 1–18. https://doi.org/10.1145/3533427 2

[Wikipedia contributors(2022)] Wikipedia contributors. 2022. List of refractive indices — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=List_of_refractive_indices&oldid=1107192412 4

[Wikipedia contributors(2023)] Wikipedia contributors. 2023. Lists of colors — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Lists_of_colors&oldid=1137295479 4

[Wu et al.(2020)] Chenyun Wu, Mikayla Timm, and Subhransu Maji. 2020. Describing Textures using Natural Language. arXiv:2008.01180 [cs.CV] 3

[XKCD(2010)] XKCD. 2010. XKCD Colors. https://xkcd.com/color/rgb/. 4

[Zhang et al.(2018)] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *CoRR* abs/1801.03924 (2018). arXiv:1801.03924 http://arxiv.org/abs/1801.03924 5

[Zheng et al.(2022)] Chuankun Zheng, Ruzhang Zheng, Rui Wang, Shuang Zhao, and Hujun Bao. 2022. A Compact Representation of Measured BRDFs Using Neural Processes. 41, 2 (2022), 1–15. https://doi.org/10.1145/3490385 2

[Zhou et al.(2023)] Xilong Zhou, Miloš Hašan, Valentin Deschaintre, Paul Guerrero, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Nima Khademi Kalantar. 2023. PhotoMat: A Material Generator Learned from Single Flash Photos. *arXiv preprint arXiv:2305.12296* (2023). 2