Controlled Texture Map Generation

Siddharth Narsipur
University of Rochester
snarsipu@u.rochester.edu

Suchith Hedge
University of Rochester
shedge@u.rochester.edu

Abstract

High-quality material capture and generation is difficult and requires significant expertise. ControlNet [1] is a deep learning algorithm that is used for controlling image synthesis in large diffusion models. Leveraging recent progress in such models, we explore training a ControlNet on a new open-source materials dataset to generate normal and roughness maps, utilizing only the basecolor as the conditioning input. We demonstrate the effectiveness of a ControlNet in capturing material details and producing accurate normal and roughness maps.

1. Introduction

Materials are vital for computer graphics and are widely used across domains (animation, 3D modelling, mixed reality). However, their creation remains challenging, even for experts.

Physically Based Rendering (PBR) is a modern approach to shading and rendering materials. PBR materials use texture maps that encode different stylization and details such as height, metallicness etc., that are then combined and applied to the surface of 3D models to create patterns or define visual effects (see Fig. 1). Basecolor (Albedo) maps are the input that defines the color or reflectivity of the surface. While creating Albedo maps for PBR materials is straightforward, they define just the base color and lack the complexity needed for realistic textures.

ControlNet [1] adds spatial conditioning control to large, pretrained text-to-image diffusion models. The work explored how various conditions such as edges (canny edge), depth (depth map), segmentation (ADE20k), and human pose could be used to control image synthesis. Importantly for us, training ControlNet is efficient and requires an order of magnitude less compute than training a model from scratch.

In this report, we discuss our work on training two ControlNet models, *controlnet-rough* and *controlnet-normal*, to generate roughness and normal maps, conditioned only on an input photograph of the material.

2. Related Work

Previous work in the field focused on two distinct but related problems - material capture and material generation. Material capture aims to create a closely matched representation of an existing material. Material generation aims to create novel materials, often utilizing a text or image prompt as a condition.

Vecchio et al. [2] worked on a custom latent diffusion model that generates high-resolution, tileable PBR materials that are guided by ControlNet conditioning. However, their models were trained on a proprietary materials dataset owned by Adobe, and were not released publicly. Our work differs from theirs by using an open-source dataset and adding control to a standard text-to-image model.

Vecchio et al. [3] created a separate approach for generation, using reflectance maps with diffusion models. Their models supported conditional and unconditional synthesis, allowing for the generation of materials from a sketch, image, text description or color palette. However, their model did not allow for tileable materials and was computationally intensive during inference.

Deschaintre et al. [4] describe a neural network that reconstructs complex materials given an input photograph that generalizes well to real-world images and is trained to handle a large variety of materials. Martin et al. [5] used a deep convolution neural net, consisting of two cascaded UNets, to enable material capture, focusing specifically on outdoor images.

Zhou et al. [6] created a StyleGAN-based architecture modified to generate tileable material maps through circular padding and tensor rolling, with optional conditioning on an input image. The same team furthered their work, training a generative model without any synthetic data, by using only real photographs taken with a cell phone [7].

3. Dataset

Over the past decade, most research in materials has relied on proprietary libraries owned by Adobe. One of our main motivations for this work was the release of MatSynth [8], an open-source dataset containing over 5300

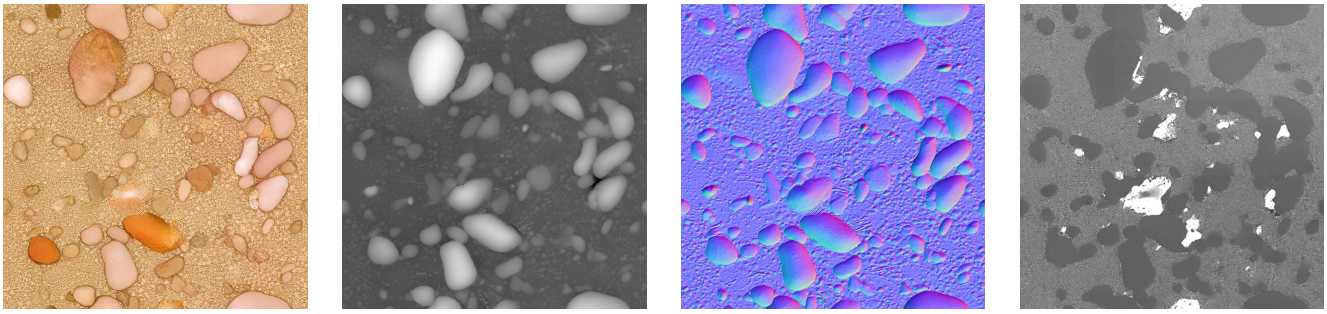

Figure 1. Basecolor, Height, Normal and Roughness Maps.
Images from ControlMat supplementary material [2]

high-resolution PBR materials.

Each PBR material in the dataset included basecolor (albedo), normal map, roughness map, diffuse map, displacement, height, metallic, and secular components. However, for our specific task, we focused solely on the basecolor, normal, and roughness maps, as these elements of the PBR are arguably the most critical in generating a complete render.

3.1. Pre-Processing

First, we performed a cleaning process to identify and remove PBR materials with low correlations between their basecolor maps and corresponding roughness and normal maps. Notably, materials such as fabric, where color variations do not correlate with height changes, road signs which contain text that do not correlate with surface features, or materials where the basecolor contained all the information, were excluded from our dataset. Due to the nuanced nature of these correlations, manual removal was necessary as automation was not reliably feasible. At the end of this stage, we reduced the dataset to approximately 3,351 materials.

3.2. Data Augmentation

To expand our dataset size and enhance model robustness, we employed a series of cropping, flipping, and rotating techniques. Following previous work [5], we produced 17 variations per material by rotating materials by 90, 180, and 270 degrees, flipping materials, and generating multiple 512x512 crops from a single 1024x1024 input. As a result, we significantly increased our dataset size, ending up with 52,309 augmented images. Our dataset is publicly available on Hugging Face [9].

3.3. Test split

To evaluate our models, we used the test split of the MatSynth dataset containing approximately 90 images. We also tested our models on real-life photographs taken on cell phones to judge their performance on material capture.

4. Method

In this section we describe the models we chose, and the methodology we followed before beginning model training.

4.1. Base Model

Past ControlNet models developed by the community predominantly employed the Stable Diffusion series [10]. Consequently, we opted for Stable Diffusion Version 2 (SD v2) as our base model for training.

4.2. ControlNet

ControlNet training accepts a dataset consisting of a text prompt, an image prompt and a conditioning image. As ControlNet ”injects” conditional control into the blocks of a neural network [7], it generates a new model that is separated from the base model.

For *controlnet-rough* and *controlnet-normal*, the basecolor was the image prompt and the respective roughness or normal map was the conditioning image.

4.3. Text Prompt

One of our main challenges was the choice of a text prompt to feed the ControlNet model. Previous work [2, 11] relied on using a ”prompt drop rate,” where only a certain percentage of training steps would be shown a text prompt, with the remaining steps being replaced by an empty string. This allows the model to directly recognize semantics in the conditioning image as a replacement for the prompt.

Following this approach, we extracted descriptive material information (color, material type etc.) from our dataset and created shortened text prompts that included the map type appended at the beginning - ”Normal Map of Red Leather”, ”Roughness Map of Grey Brick Wall” etc. However, in our initial training with a 50% drop rate, the model performed poorly and generated noisy results.

We observed that as the Stable Diffusion model would be prompted on, say ”Normal Map of Christmas Tree”, it would generate wildly different images of Christmas Trees

on each epoch, and fail to be conditioned on our normal/roughness map. We then discovered that MatSynth dataset contained non-descriptive or poor-quality tags for many materials, which hindered the model performance.

We considered passing images through an image captioning model such as BLIP but decided that we would focus our conditioning completely on our image to achieve the pixel-to-pixel correlation between basecolor and normal/roughness maps.

Two ideas were then considered - passing in an empty prompt ("") or a fixed prompt, using just the name of the texture map to be generated ("Normal Map"). After experimentation, we found that the latter approach worked better and allowed the ControlNet model to learn the basic "style" of a normal/roughness map, while the actual detail was passed on from the conditioning image.

5. Experiments

The following sections contain a description of model training, our attempts at achieving better results, and our model evaluation. The training, inference and testing scripts are available at our public Github repository (sidnarsipur/TextGen) [12].

5.1. Hyperparameters

The choice of hyperparameters was crucial to model performance and slight differences created huge changes between training runs. The main parameters to consider were prompt drop rate (mentioned above), batch size, learning rate, and the number of epochs (training steps).

Batch Size was set to 15 on controlnet-rough and 12 on controlnet-normal to maximise GPU utilization. The lower batch size for controlnet-normal was due to the GPU utilization spiking in the middle of training, which caused the program to crash. Performance was also improved by using xFormers memory efficient attention. We experimented with gradient acculumation and mixed precision (fp16) to increase batch size but it proved ineffective compared to just setting the batch size.

To experiment, we investigated a range of learning rates from 1e-4 to 1e-5 to optimize the training process of our neural network models. Our decision to explore lower learning rates was informed by the expectation of achieving more stable training updates. However, when using lower learning rates, such as 1e-5, we observed that the models failed to converge even after 10,000 training steps, resulting in poor inference quality. In contrast, a learning rate of 1e-4 converged and yielded the most optimal results in our experiments.

The performance of ControlNet models is known to decay over time with smaller datasets [13], so it was important to choose enough epochs to allow the model to learn but not overshoot. We trained *controlnet-rough* and *controlnet-normal* for 8 and 11 epochs respectively. For *controlnet-normal*, we found a performance decay after 8 epochs and therefore used the 8-epoch model for evaluation over the 11-epoch model.

Model	Learning Rate	Epochs	Batch Size
controlnet-rough	1e-4	8	15
controlnet-normal	1e-4	11	12

Table 1. Training Hyperparameters

5.2. Training

Both models were trained on one A100 40GB GPU on the BlueHive cluster.

We followed the official Hugging Face diffusers training script [14], a powerful open-source library for training or fine-tuning large diffusion models, with modifications made by us to modify text prompting and improve performance reporting. The training lasted 19 hours and 26 hours for *controlnet-rough* and *controlnet-normal* respectively.

Both models generated poor, noisy results for the first 8k steps, but suddenly converged between 10k and 15k steps, which was consistent with community experience [13].

(a) 1k steps (b) 6k steps (c) 10k steps (d) 14k steps

Figure 2. Sudden convergence at 14k steps.

5.3. Results

After testing, we find that our models succeed in producing consistent, high-quality normal and roughness maps for a variety of materials. Our models are particularly good at generating materials containing repeated patterns and discernible shapes, such as ground, brick, cloth and stone (see Fig. 3, 4).

Inference takes 3-5 seconds and 12GB VRAM for 512*512 images and 10-15 seconds with 25GB VRAM for 1024*1024 images. We use 50 inference steps and experiment with different seeds to find the best result.

Although our models were trained only on basecolor maps, they still perform well on receiving flash photographs, due to the minor differences between basecolor maps and photographs for many classes of materials, highlighting our model's ability to generalize beyond inputs that resemble its training distribution.

However, due to our limited training time and use of a general base model, our approach does have limitations.

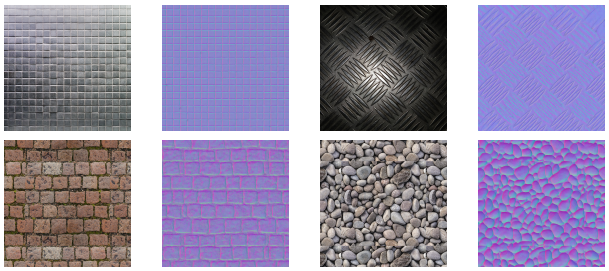

Figure 3. Input Photograph and Generated Normal Maps

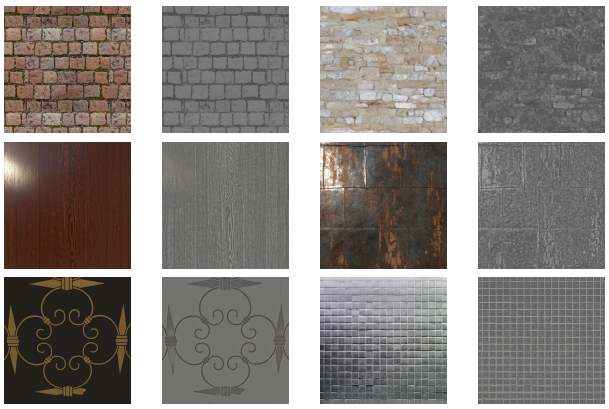

Figure 4. Input Photograph and Generated Roughness Maps

First, while it accepts tileable input, the generated normal map often has visible seams. Second, while shapes, textures, and depth are captured well in our generated normal maps, the model often mispredicts the color of our Blue Channel (corresponding to the Z-Axis).

Our models also struggle with materials that show a weak correlation between the input photograph and their respective normal/roughness maps (leather, complex patterns, metals). The main reason for this is the removal of such images from our training dataset as mentioned above. Training on a larger dataset and for longer with such images will improve performance.

5.4. Evaluation

In this evaluation, we focus on comparing *controlnet-rough* and *controlnet-normal* with MatForger, another generative diffusion model trained on the same publicly available dataset [15]. Due to the closed-source nature of ControlMat and TileGen, a direct comparison with these models wasn't possible.

We evaluated MatForger, *controlnet-rough*, and *controlnet-normal* on the 89 images from the MatSynth test set, ensuring that none of the models were trained on any one of these images. To assess normal map quality, we employed cosine error, which measures the angular difference between predicted normals and the true values. For roughness maps, we used Root Mean Square (RMS) error, which indicates the overall magnitude difference between predicted and ground truth roughness. These metrics were chosen based on previous literature in the field [2].

All images were conducted using images generated at 512*512. We provided a fixed prompt ("Roughness/Normal Map") to our models with the conditioning image, while MatForger received only image conditioning since it cannot handle dual conditioning.

Model	Cosine Error	RMS Error
controlnet-rough	-	0.249
controlnet-normal	0.028	-
MatSynth	0.010	0.276

Table 2. Model Error (Lower is Better)

Our findings indicate that our model performs better on roughness generation while MatSynth performs better on normal generation. As previously noted, the frequent misprediction of the the Blue Channel in our normal maps explains the inferior performance of *controlnet-normal*.

Broadly, we notice that maps produced by MatForger have colors that are more accurate to the ground truth, while our models preserve alignment, shape and pattern better, due to the effectiveness of our ControlNet conditioning (see Fig. 5).

6. Conclusion

In this work, we train a diffusion model to create roughness and normal maps for PBR materials, conditioned only on the basecolor. We demonstrate that our method can generate realistic, high-quality texture maps for a large class of materials, and is even effective at handling flash photographs taken by a cell phone.

Training a ControlNet on top of a model such as Stable Diffusion is easier to train, quicker to converge and requires less compute. The model achieves good performance with only 20 hours of training. We explored different hyperparameters and prompting strategies, finding that a fixed prompt achieves the best results during both training and inference. Nonetheless, the model's performance declines, particularly in indoor categories, when it encounters materials whose texture maps have a low correlation with basecolor. More experimentation is needed to determine. whether this is due to model architecture or the size of the training set. Using a diffusion model specifically trained to represent and generate materials will offer broader image compatibility and improved results but requires more compute for training and inference.

Future work would be aimed at developing a robust version of this model that will focus on refining the prediction accuracy of the Blue Channel in generated normal maps.

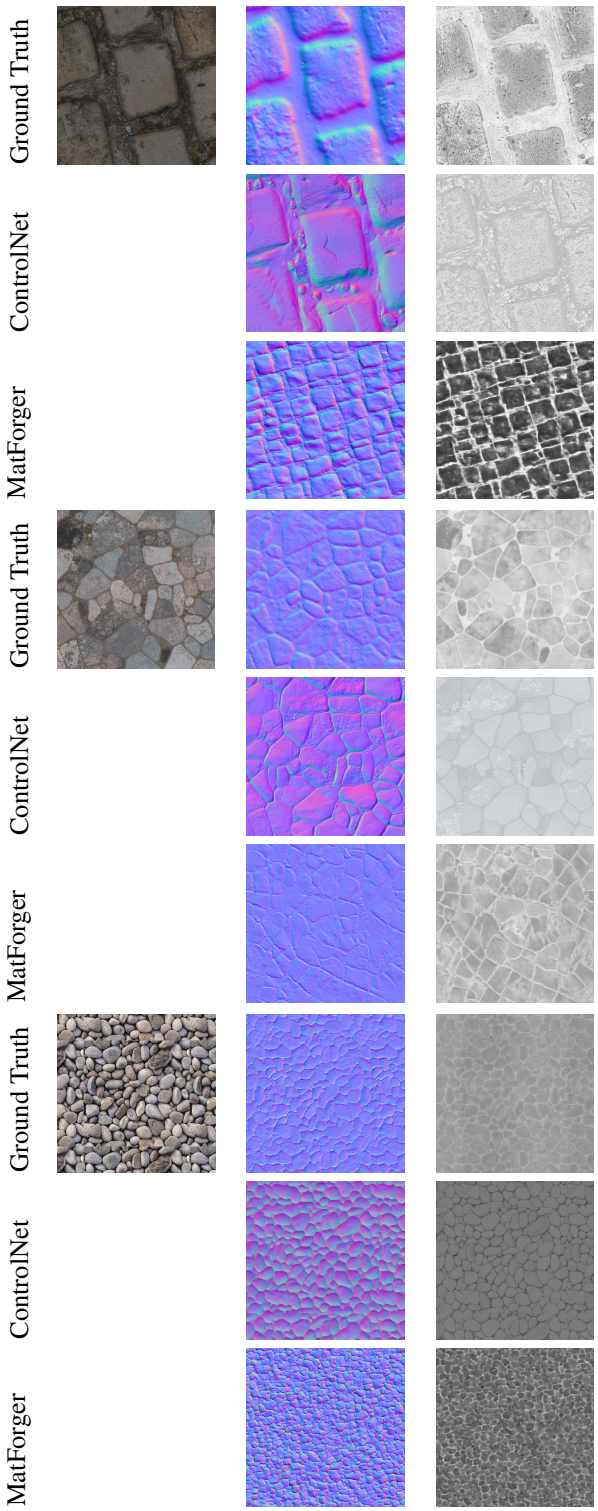

Figure 5. Comparison with Ground Truth - Material Input, Normal Map and Roughness Map.

7. Acknowledgement

The authors thank the Center for Integrated Research Computing (CIRC) at the University of Rochester for providing computational resources and technical support. We also thank our Professor Chenliang Xu and the teaching assistants of CS 249 for their advice and guidance throughout the project.

References

[1] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," 2023. 1

[2] G. Vecchio, R. Martin, A. Roullier, A. Kaiser, R. Rouffet, V. Deschaintre, and T. Boubekeur, "Controlmat: A controlled generative approach to material capture," 2023. 1, 2, 4

[3] G. Vecchio, R. Sortino, S. Palazzo, and C. Spampinato, "Matfuse: Controllable material generation with diffusion models," 2024. 1

[4] V. Deschaintre, M. Aittala, F. Durand, G. Drettakis, and A. Bousseau, "Single-image svbrdf capture with a rendering-aware deep network," *ACM Transactions on Graphics*, vol. 37, p. 1–15, July 2018. 1

[5] R. Martin, A. Roullier, R. Rouffet, A. Kaiser, and T. Boubekeur, "Materia: Single image high-resolution material capture in the wild," *Computer Graphics Forum*, vol. 41, no. 2, pp. 163–177, 2022. 1, 2

[6] X. Zhou, M. Hašan, V. Deschaintre, P. Guerrero, K. Sunkavalli, and N. Kalantari, "Tilegen: Tileable, controllable material generation and capture," 2022. 1

[7] X. Zhou, M. Hasan, V. Deschaintre, P. Guerrero, Y. Hold-Geoffroy, K. Sunkavalli, and N. K. Kalantari, "Photomat: A material generator learned from single flash photos," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23, ACM, July 2023. 1, 2

[8] G. Vecchio and V. Deschaintre, "Matsynth: A modern pbr materials dataset," 2024. 1

[9] www.huggingface.co/datasets/sidnarsipur/controlnet_data. 2

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022. 2

[11] Y. Yang, D. Gui, Y. Yuan, W. Liang, H. Ding, H. Hu, and K. Chen, "Glyphcontrol: Glyph conditional control for visual text generation," 2023. 2

[12] www.github.com/sidnarsipur/TextGen. 3

[13] www.civitai.com/articles/2078. 3

[14] www.github.com/huggingface/diffusers. 3

[15] www.huggingface.co/gvecchio/MatForger. 4