Generating Normal Maps from Diffuse Game Textures with Machine Learning

Ong Wei Yong
Department of Computer Science and Creative Technologies
University of the West of England
Coldharbour Lane
Bristol, UK
wei11.ong@live.uwe.ac.uk
Student ID: 20022191

Figure 1. Two pairs of a Diffuse Texture and its generated Normal Map with Machine Learning.

Abstract
This project aims to use machine learning to automatically generate normal maps from diffuse game textures. By using machine learning algorithms, it is possible to generate normal maps quickly and efficiently, saving time and resources for game developers. The project will explore various techniques and evaluate their performance on a dataset of diffuse game textures. The goal is to develop a system that can generate accurate and visually pleasing normal maps with minimal input from the user.

Author Keywords
Pix2PixHD; Normal Map; Diffuse Map; Machine Learning; Image Generation;

Introduction
As games get bigger and more expansive, more and more game textures are required to build such virtual worlds and experiences. In many cases, modern 3D games include both diffuse maps and normal maps in their material to create a visually appealing world. Creating a tool that automatically generates diffuse textures from normal maps will speed up the texture creation process and thus improve the overall efficiency in any game development pipeline.

Background & Research

What is Pix2Pix?

Pix2Pix is a conditional Generative Adversarial Network (cGAN) that translates an input image into a corresponding output image (Isola *et al.*, 2016). Isola *et al.* (2016) conducted experiments to enable users to turn edges into photos and even black and white images into colour photos. A high definition version of Pix2Pix called Pix2PixHD was also later implemented (Wang *et al*., 2017).

What are cGANs?

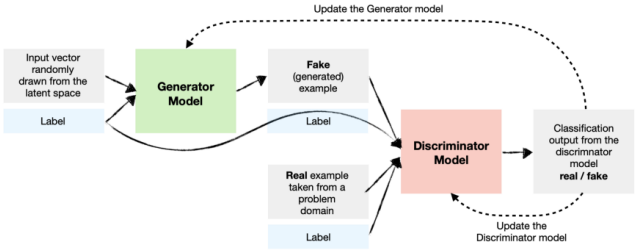

Figure 2. A cGAN model architecture (Dobilas, 2020).

cGANs are an extension of Generative Adversarial Network (GAN) that trains generative models via a machine learning framework (Frumkin *et al.*). This deep learning method applies a conditional setting where both the generator and discriminator are provided supplementary information and labels. By dealing with information from different contexts, the model can learn multi-modal mapping from inputs to outputs (Mirza *et al.,* 2014). Frumkin *et al.* lists cGANs' many possible applications such as:

- Image-to-image translation
- Text-to image synthesis
- Video Generation
- Convolutional face generation
- Generating shadow maps
- Diversity-sensitive computer vision tasks

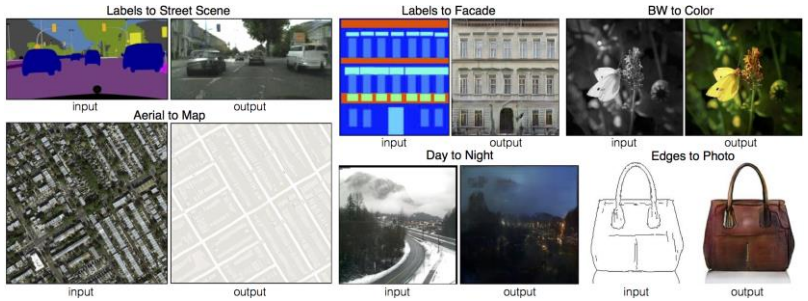

Figure 3. Example of Implementation Models of Pix2Pix (Isola et al., 2016).

What are Diffuse Maps?

Diffuse Maps are a type of texture map that is applied onto objects in a game engine. They define the colour and patterns of objects. When you map a diffuse colour, it resembles painting an image on the surface of the object (Crytek).

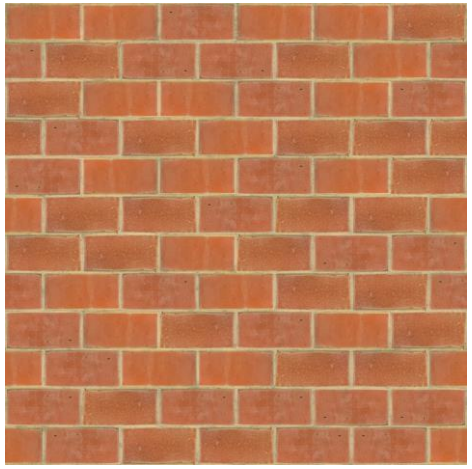

Figure 4. Example of a Brick Diffuse Map.

What are Normal Maps?

Normal Maps are also another type of texture map. They are used to fake lighting onto meshes with a lower polygon count as if it were a mesh with a higher polygon count. This enables models to look a lot more detailed without sacrificing as much performance. The RGB values represent X, Y, Z direction components of the mesh's normal (Learn OpenGL).

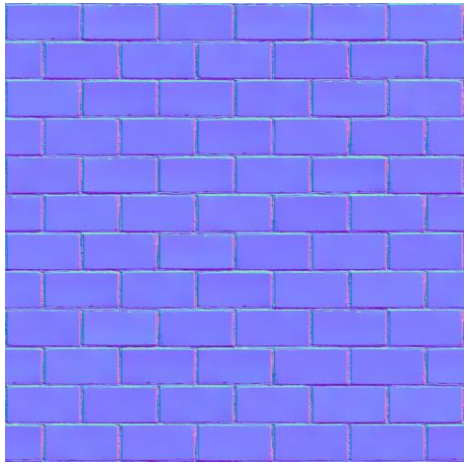

Figure 5. Example of a Brick Normal Map.

Why do we need this tool?

Quickly generating normal maps from diffuse maps will enable game developers to quickly create believable textures for their 3D spaces. This will save time and money for game development teams.

Related Works

Adobe Photoshop

One current method of generating a Normal Map from a Diffuse Map is by using traditional photo editing methods such as editing a Diffuse Map's colours using a filter in photo editing software such as Adobe Photoshop.

Here are the steps as listed by Geldart (2022):

- Open the texture you want to use in Photoshop.
- Go to the Filter Menu.
- Select '3D'.
- Then select 'Generate Normal Map'.

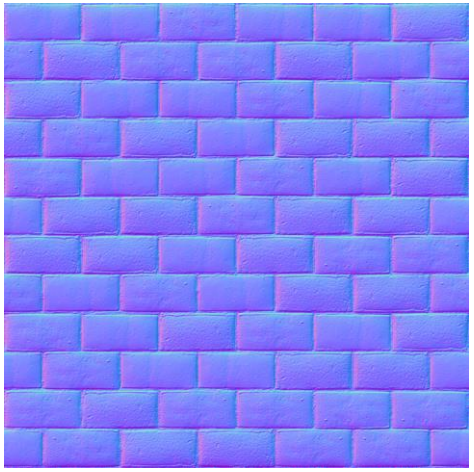

Figure 6. Example of a Brick Normal Map generated using Adobe Photoshop.

Diffuse2Normal Pix2Pix

Another method was implemented by Cuni (2020) to generate Normal Maps using Pix2Pix.

Cuni (2020) trained 790 diffuse-normal map image pairs for 32 hours. Cuni (2020) further states that this method generates consistent normal maps of any given texture.

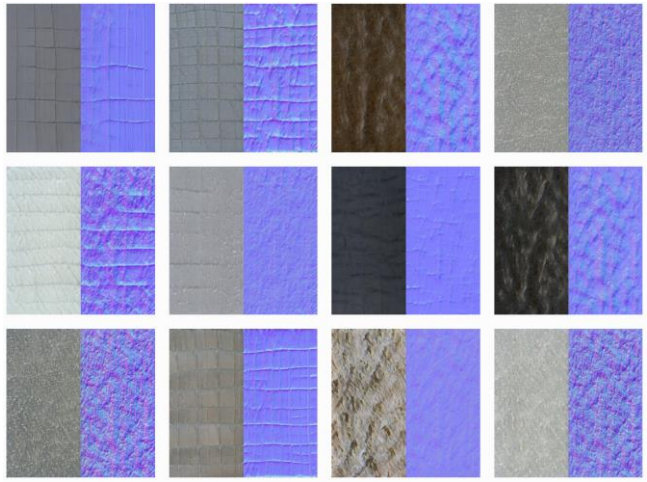

Figure 7. Generated texture maps (Cuni, 2020).

Implementation and Analysis

Dataset Collation

Dataset Collation for game textures was done in two sections:

- Testing dataset – Consists of 4 types of diffuse textures. Bark, Brick, Asphalt and Leather.
- Training dataset – Consists of diffuse-normal texture pairs.

The testing dataset was collated by using 'Stable Diffusion 2.0', a latent text-to-image diffusion model. It allows users to generate images via a text description. Further features include inpainting, outpainting and image-to-image translations via a text prompt (Cusick, 2022). The textures were also upscaled to match the size of the training dataset.

The training dataset was collated by web scraping from ambientCG, a public domain resource created by Demes (2022) for Physically Based Rendering. 1024x1024 pixel texture sets were scraped via a CSV file provided by ambientCG using the *'wget'* command from the '*pandas'* library.

The scraped texture sets initially came with other types of textures. In order to isolate just the diffuse and normal texture pairs, the following textures were removed:

- Metalness textures.
- Emission textures.
- Roughness textures.
- Ambient Occlusion textures.
- Mask ID textures.
- Alpha textures.

Implementation – Training and Testing
Previous known methods were first researched and tested to attempt to create a tool which generates more accurate normal maps.

Adobe Photoshop

Texture sets were generated using Adobe Photoshop using the steps by Geldart (2022).

Pix2Pix

Google Colab, an online tool that allows users to edit and run python code through a browser was used to during this implementation (Google LLC).

- The Pix2Pix repository was cloned into Google Colab and dependencies were installed.
- Folders were then split with *split-folders* into three folders labelled train, val and test.
- Files were then moved into the appropriate directory.
- Both Diffuse and Normal Maps were then combined into image pairs.
- Image pairs were then trained at first with 5 epochs to test and then at 100 epochs. Images are downscaled to 256x256 pixels during training by default.
- Normal maps were then upscaled to 1024x1024 pixels and generated.

Pix2PixHD

Implemented with Google Colab.

- The project was connected to a GPU Instance.
- Google Drive was mounted.
- Pix2PixHD repository was cloned into Google Drive.
- Dataset Folders were created (2 training folders for image translation and 1 testing folder).
- Textures Dataset was uploaded.
- Training variables were set.
- Training was run until 137 epochs.
- Normal Images were generated with training data.

Stable Diffusion

Implemented with Google Colab.

- The Stable Diffusion 2 repository was cloned into Google Colab and dependencies were installed.
- The 768x768 pixel Stable Diffusion 2 Checkpoint File was downloaded and loaded into the project.

- Text prompts and samples were set.
- The texture maps were then generated and downloaded.

The following images were generated with Stable Diffusion 2 at 768x768 pixels.

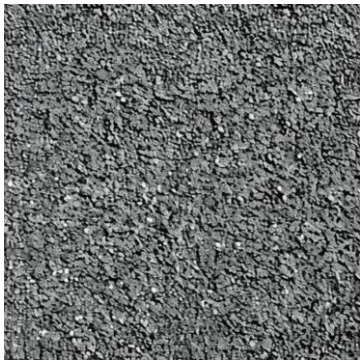

Figure 8. The generated Asphalt Texture Image. Text Prompt: 'Road Asphalt Texture'.

Figure 10. The generated Bark Texture Image. Text Prompt: 'Bark Texture'.

The images generated by Stable Diffusion 2 were then upscaled to 1024x1024 pixels with Gigapixel AI (Topaz Labs, 2022) to match the trained dataset.

Figure 9. The generated Leather Texture Image. Text Prompt: 'Leather Texture'.

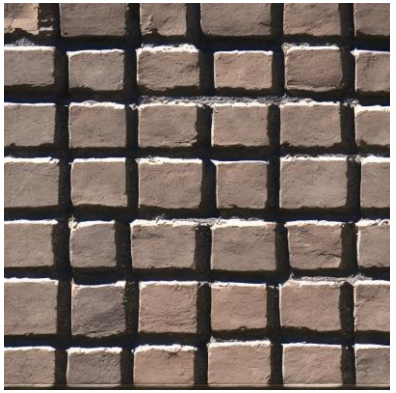

Figure 11. The generated Brick Texture Image. Text Prompt: 'Brick Texture'.

The Stable Diffusion Textures were then used to generate Normal Maps with Pix2PixHD.

Unity Engine (Unity Technologies, 2022).

The texture pairs were then implemented within Unity Game Engine.

Figure 12. The generated Stable Diffusion Textures and Pix2PixHD Generated Normals implemented within Unity Engine.

In Figure 12, the smoothness of the material was set higher to see the effect of the normal maps as normal maps drive a material's reflection more easily.

Evaluation

Pix2Pix

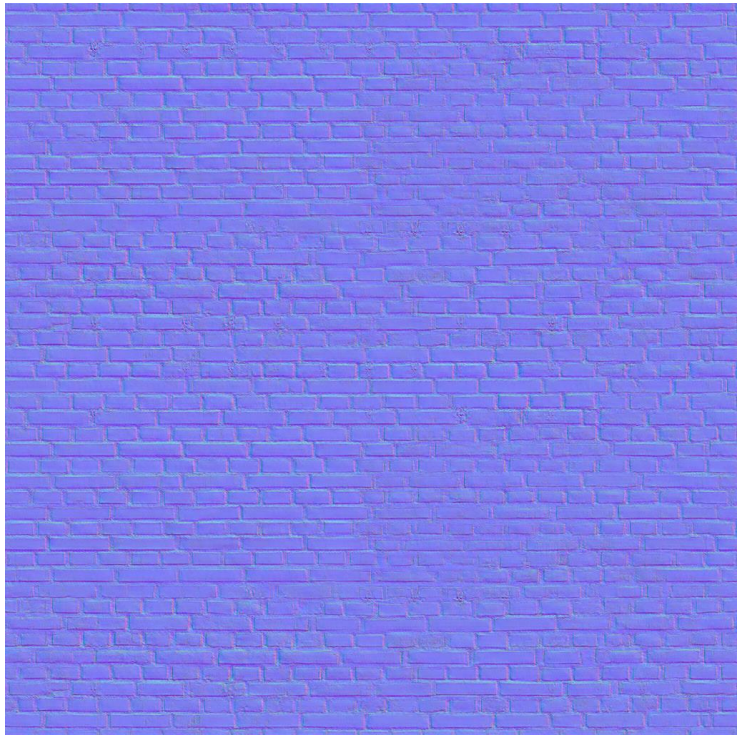

Figure 13. A Brick Normal Map generated using Pix2Pix.

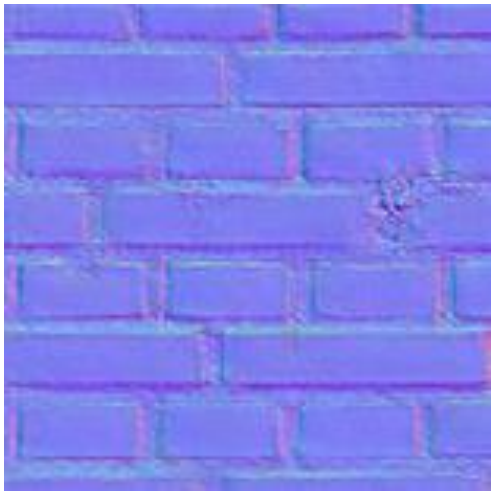

Figure 14. A cropped Brick Normal Map generated using **Pix2Pix.**

Figure 15. The cropped **original** Brick Normal Map.

Generated Normal Maps with Pix2Pix had too low a resolution as shown in Figure 14 compared to the original texture's image resolution as shown in Figure 15. This is because Pix2Pix trains images with a cropped image and then upscaled later. To bypass this issue, this method was replaced with a high-resolution implementation of Pix2Pix called Pix2PixHD.

Brick

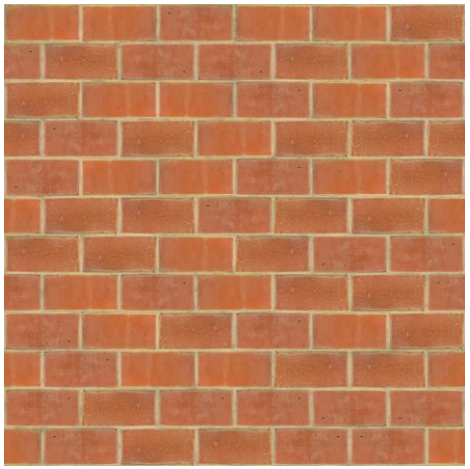

Figure 16. The **original** Brick Diffuse Texture.

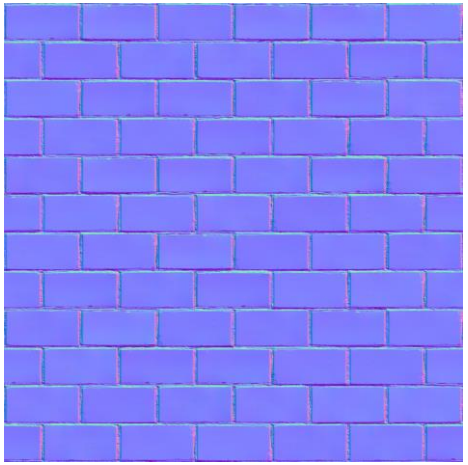

Figure 17. The **original** Brick Normal Texture.

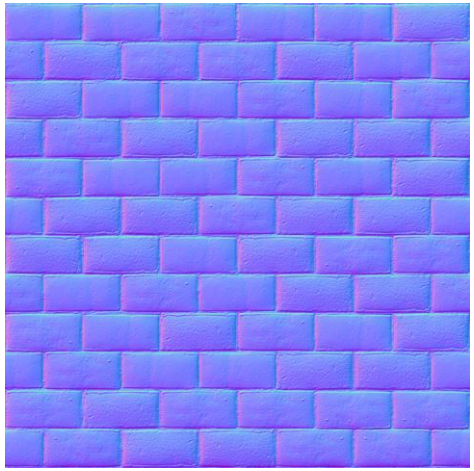

Figure 18. The Brick Normal Texture generated in Adobe Photoshop.

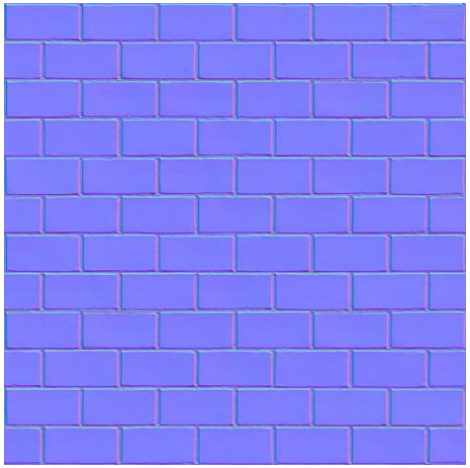

Figure 19. The Brick Normal Texture generated in Pix2PixHD.

Brick Texture Evaluation

When compared against the original Brick Normal Texture as shown in Figure 17, the Brick Normal Texture generated in Adobe Photoshop as shown in Figure 18 was observed to have the following inaccuracies:

- The Bricks had a rounded surface when they were supposed to be flat.
- The Bricks had a lot noisier details. This makes the normal map over detailed.

The same comparison was done with the Brick Normal Texture generated in Pix2PixHD as shown in Figure 19.

The texture was observed to have the following inaccuracies:

- The Brick's edges lack slight detail in its edge warping.
- The texture has some slight artifacts and smudging.

Based on my observations, the Pix2PixHD implementation provided the more accurate normal map generation method for the Brick Texture.

Asphalt

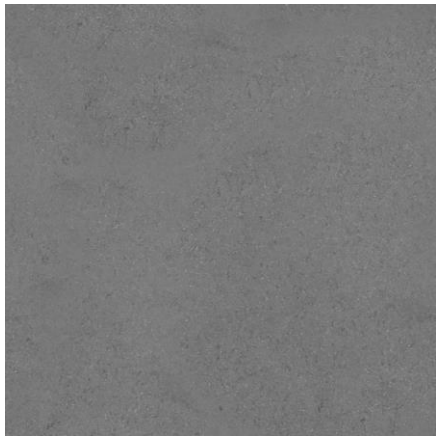

Figure 20. The **original** Asphalt Diffuse Texture.

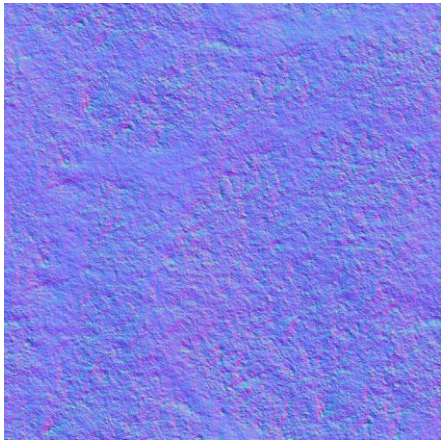

Figure 21. The **original** Asphalt Normal Texture.

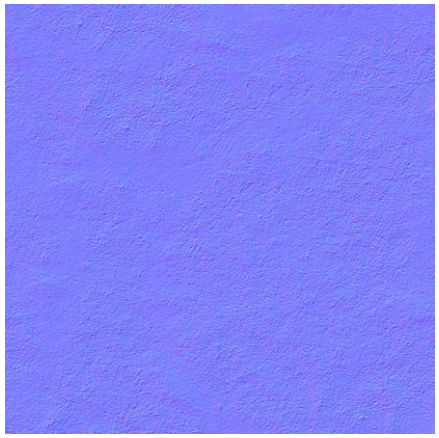

Figure 22. The Asphalt Normal Texture Generated in Adobe Photoshop.

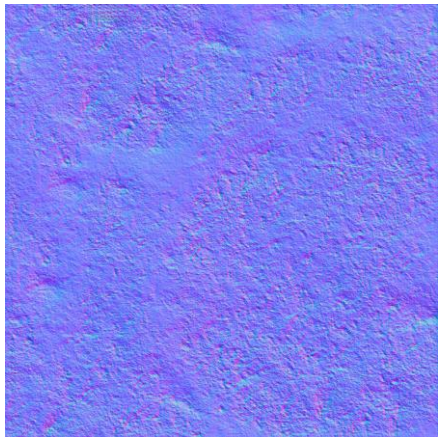

Figure 23. The Asphalt Normal Texture Generated in Pix2PixHD

Asphalt Texture Evaluation

When compared against the original Asphalt Normal Texture as shown in Figure 21, the Asphalt Normal Texture generated in Adobe Photoshop as shown in Figure 22 was observed to have the following inaccuracies:

- The texture was much less detailed.
- The texture contrast was not as strong. This means that the material would not have as strong a bump compared to its original counterpart.

The same comparison was done with the Asphalt Normal Texture generated in Pix2PixHD as shown in Figure 23.

The texture was observed to have the following inaccuracies:

- The texture has some slight artifacts and smudging.

Based on my observations, the Pix2PixHD implementation provided the more accurate normal map generation method for the Asphalt Texture.

Leather

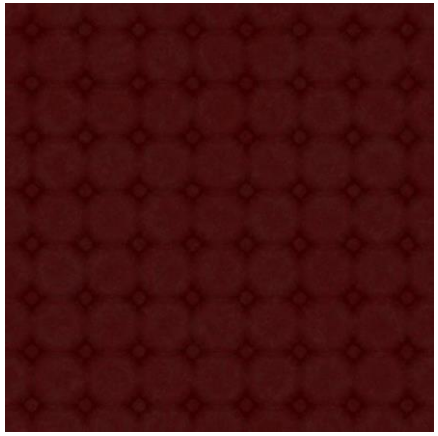

Figure 24. The **original** Leather Diffuse Texture.

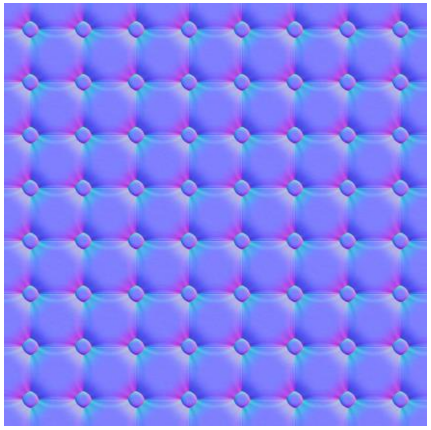

Figure 25. The **original** Leather Normal Texture.

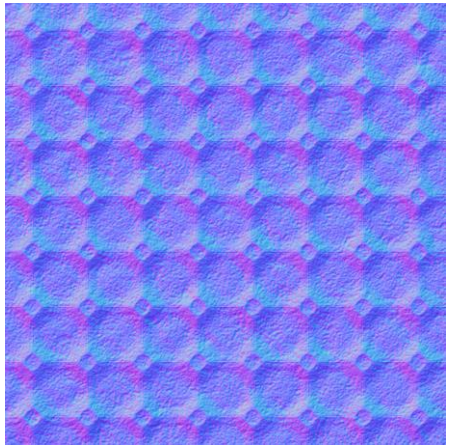

Figure 26. The Leather Normal Texture Generated in Adobe Photoshop.

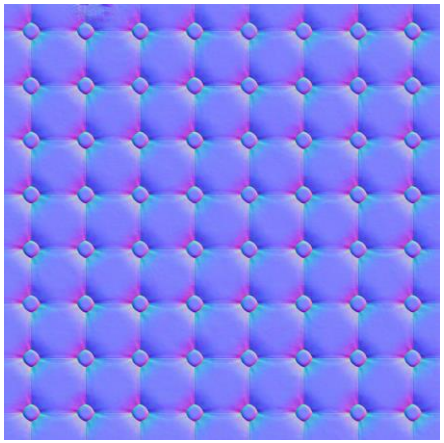

Figure 27. The Leather Normal Texture Generated in Pix2PixHD.

Leather Texture Evaluation

When compared against the original Leather Normal Texture as shown in Figure 25, the Leather Normal Texture generated in Adobe Photoshop as shown in Figure 26 was observed to have the following inaccuracies:

- The texture had a lot messy and noisier details.
- The Leather stiches went in the opposite direction.
- The Leather creases were not present.

The same comparison was done with the Leather Normal Texture generated in Pix2PixHD as shown in Figure 27.

The texture was observed to have the following inaccuracies:

- The texture has some slight artifacts and smudging.

Based on my observations, the Pix2PixHD implementation provided the more accurate normal map generation method for the Leather Texture.

Bark

Figure 28. The **original** Bark Texture.

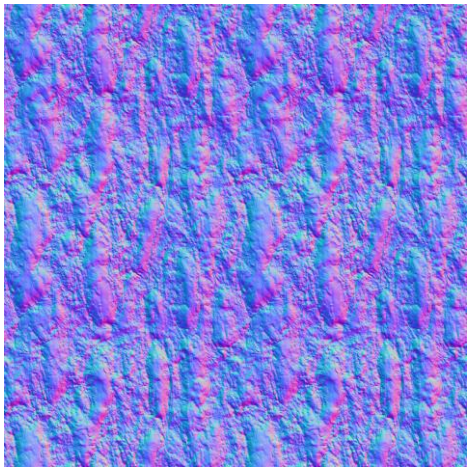

Figure 29. The **original** Bark Normal Texture.

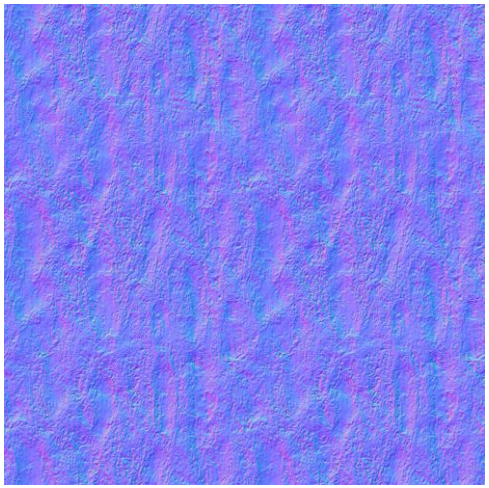

Figure 30. The Bark Normal Texture Generated in Adobe Photoshop.

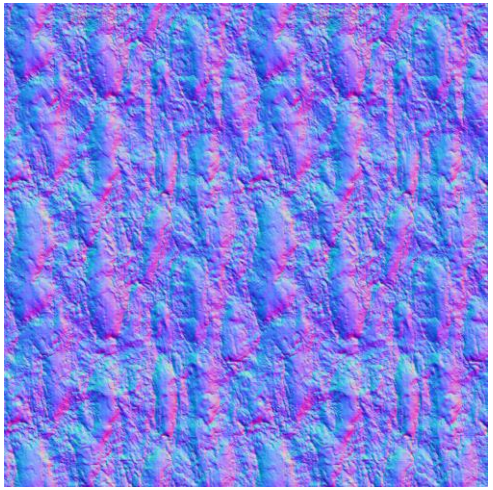

Figure 31. The Bark Normal Texture Generated in Pix2PixHD.

Bark Texture Evaluation

When compared against the original Bark Normal Texture as shown in Figure 29, the Bark Normal Texture generated in Adobe Photoshop as shown in Figure 30 was observed to have the following inaccuracies:

- The texture was much less detailed.
- The texture contrast was not as strong. The material would not have as strong a bump as the original.

The same comparison was done with the Bark Normal Texture generated in Pix2PixHD as shown in Figure 31.

The texture was observed to have the following inaccuracies:

- The texture has some slight artifacts and smudging.

Based on my observations, the Pix2PixHD implementation provided the more accurate normal map generation method for the Bark Texture.

Evaluation Summary:

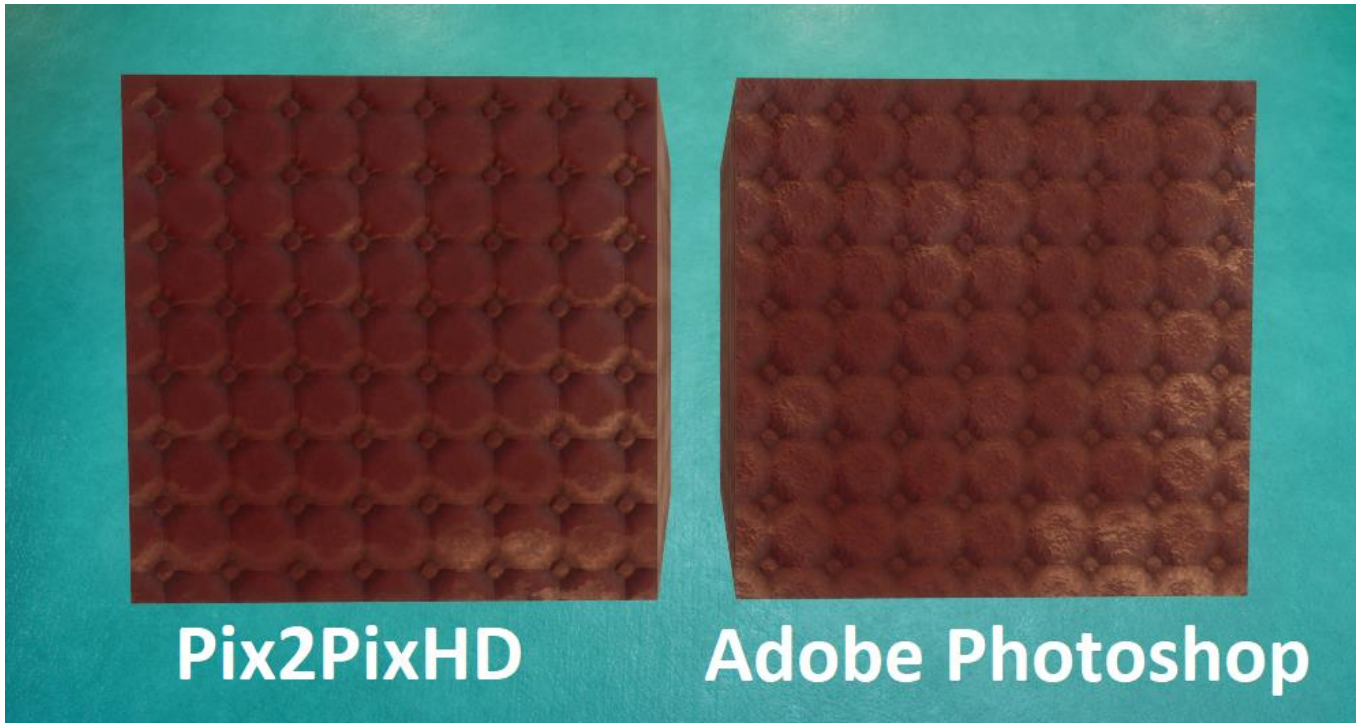

Figure 32. A comparison between the Pix2PixHD Implementation in Unity Engine compared to Adobe Photoshop for the Leather Texture. The Adobe Implementation showed much noisier textures and inaccurate details compared to the Pix2PixHD implementation.

Pix2Pix

- Resolution of normal maps were too low as images were downscaled during training.
- This resulted in the method being discarded.

Adobe Photoshop

- Some Normal Maps were overdetailed and noisy.
- Some Normal Maps details were missing.
- Some Normal Maps details were inaccurate. (e.g., Having curved surfaces instead of flat surfaces.)

Pix2PixHD

- Smudging and Artifacts were present.

Pix2PixHD generated better normal maps compared to Adobe Photoshop and Pix2Pix. This can be also observed in the Unity Engine Implementation shown in Figure 32.

How can this tool be improved?

- Train with own dataset style.

Game studios could train the tool with their own art style. For example, a game with a stylised art direction instead of a realistic art direction might find the tool generating more accurate normal maps for stylised textures.

- Train with separated labels for diffuse-normal texture map pairs.

Some textures may be confused with other types of similar looking textures. Training with more confined labels such as training all the brick textures as their own dataset may mitigate this issue.

- Train dataset for longer. (Higher Epochs)

This will mitigate smudging and artifacts.

Future Work

The following are some references to potential expansions of this project:

SurfaceNet: Adversarial SVBRDF Estimation from a Single Image

SurfaceNet enables users to estimate spatially-varying bidirectional reflectance distribution function (SVBRDF) material properties from a single image by utilizing GANs and image translation. This means that users will be able to generate Diffuse, Normal, Roughness and Specular Maps from just a single source image (Vecchio, Palazzo and Spampinato, 2021).

Armorlab

ArmorLab is a software designed for AI-powered texture authoring. Users can extract base color, height, normal map, occlusion, and roughness textures from a photo. Users can also generate seamless materials with the same set of texture maps with just a text prompt. The software also enables users to implement tiling, upscale and variance as well as inpainting to their materials (Armorlab).

OpenAI

Using the OpenAI API, users are able to implement text to image generation straight into game engines such as Unity or Unreal Engine (OpenUPM).

Conclusion

The use of neural networks as a texture map generation tool has a place in any game development team whether big or small, allowing for better work efficiency by saving on time costs while still maintaining visual quality.

References

Adobe Inc. (2022) *Adobe Photoshop* (2022) [computer program]. Available from: https://www.adobe.com/uk/creativecloud/plans.html [Accessed 10 December 2022].

ArmorLab (no date) *Armorlab* (no date) [computer program]. Available from: https://armorlab.org/ [Accessed 10 December 2022].

Crytek (no date) *Diffuse Maps*. Available from: https://docs.cryengine.com/display/SDKDOC2/Diffuse+Maps [Accessed 10 December 2022].

Cuni, B. (2020) *Deep Textures*. Available from: https://www.cunicode.com/works/deep-textures [Accessed 10 December 2022].

Cusick, B. (2022) Stable Diffusion 2.0 release. *Stability.ai* [blog]. Available from: https://stability.ai/blog/stable-diffusion-v2-release [Accessed 10 December 2022].

Demes, L. (2022) *AmbientCG*. [online]. Available from: https://ambientcg.com/ [Accessed 10 December 2022].

Dobilas, S. (2022) *CGAN: Conditional Generative Adversarial Network — How to Gain Control Over GAN Outputs*. Available from: https://towardsdatascience.com/cgan-conditional-generative-adversarial-network-how-to-gain-control-over-gan-outputs-b30620bd0cc8 [Accessed 10 December 2022].

Frumkin, D., Manipula, M., Kachai, K., Fehr, A. and Sewell, D. (no date) *Conditional Generative Adversarial Network (cGAN)*. [online]. Available from: https://golden.com/wiki/Conditional_generative_adversarial_network_(cGAN)-99B85NK [Accessed 10 December 2022].

Geldart, M. (2022) *Can You Create a Normal Map in Photoshop?*. Available from: https://www.websitebuilderinsider.com/can-you-create-a-normal-map-in-photoshop/ [Accessed 10 December 2022].

Google LLC (no date) *Colaboratory*. Available from: https://research.google.com/colaboratory/faq.html [Accessed 10 December 2022].

Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A.A. (2016) Image-to-image translation with conditional adversarial networks. *arXiv* [online]. [Accessed 10 December 2022].

Learn OpenGL (no date) *Normal Mapping*. Available from: https://learnopengl.com/Advanced-Lighting/Normal-Mapping [Accessed 10 December 2022].

Mirza, M. and Osindero, S. (2014) Conditional generative adversarial nets. *arXiv* [online]. [Accessed 10 December 2022].

OpenUPM (no date) *OpenAI*. Available from: https://openupm.com/packages/com.openai.unity/ [Accessed 10 December 2022].

Topaz Labs (2022) *Gigapixel AI* (2022) [computer program]. Available from: https://www.topazlabs.com/gigapixel-ai [Accessed 10 December 2022].

Unity Technologies. (2022) *Unity* (2022) [computer program]. Available from: https://unity.com/ [Accessed 10 December 2022].

Vecchio, G., Palazzo, S. and Spampinato, C. (2021) SurfaceNet: Adversarial SVBRDF Estimation from a Single Image. *arXiv* [online]. [Accessed 10 December 2022].

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J. and Catanzaro, B. (2017) High-resolution image synthesis and semantic manipulation with conditional GANs. *arXiv* [online]. [Accessed 10 December 2022].