School of Computing
FACULTY OF ENGINEERING AND PHYSICAL SCIENCE

UNIVERSITY OF LEEDS

Learning to texture 3D models

Isaiah Jean Fergile-Leybourne

Submitted in accordance with the requirements for the degree of Computer Science with Artificial Intelligence BSc

Type of project: Exploratory Software

[illegible]

(Signature of Student) ____________________

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Three-dimensional (3D) models are composed of a mesh and a texture. A mesh is the shape - or form - of the object and is composed of interconnected vertices that construct faces. A texture is an image that is mapped to the outer surface of the mesh. Typically, texturing is performed by trained 3D artists and can be a time-consuming process. This project will be investigating image-to-image learning systems and how they can be applied to automatically generate textures, which in turn can be used to texture three-dimensional models.
Generative Adversarial Networks (commonly referred to as GANs) are an approach to machine learning that specialises in generating new data points for a given dataset. GANs aim to discover patterns in the overall input dataset and produce similar but novel outputs. This is done but dividing the model into two sub-models: a generator and a discriminator. The generator creates new instances from the given input dataset and the discriminator tries to classify each generated instance either real of fake (part of the input dataset or a newly generated instance respectively). The two sub-models are trained together and compete against one another – entering a cat and mouse game – where the generator is trying to create more 'realistic' novel outputs and the discriminator is trying a classify them with better accuracy. This overall structure results in new outputs that are similar to the input dataset but still express originality.

1.2 Aims and Objective

1.2.1 Aims

This is an Exploratory Software project meaning we will produce a piece of software that aims to demonstrate features and concepts relating to the project. This means that the software will not be is not a polished production quality-level application that will be presented to an end-user, but instead will serve the purpose of illustrating how Image-to-Image translation can be applied to automatically texture 3D models. Ideally, the textures that are generated should be novel but still maintain a sense of realism. The automated process will save time for the user and allow unskilled individuals to perform texturing of 3D models, making this area of computing more accessible.
Ultimately, we would like the user of the application to be able to paint a UV map, for a given 3D model, using a semantic layout. This semantic layout will be composed of

basic colours which will be used as an input for the GAN to identify what type of texture should correspond with each portion of the UV map. The GAN will then generate new textures, based on a dataset of textures, that will then be applied to a 3D model. We can then render the final textured model.

1.2.2 Objectives

The objective of this project is to create an application that can automatically apply textures, that are generated by a GAN, to a 3D model. The GAN will be trained on our newly formed dataset that is a standardised amalgamation of texture datasets found online (Source). The use of a GAN will mean that the texture generated textures will be novel allowing the user of the program to create new and diverse textures. Ideally, the textures generated should be consistent with the input datasets and not subject to artifacts or other irregularities that would make the textures seem unnatural. Since we do not want the process to be entirely automated, we would like the user to interact with a User Interface to draw a semantic map over the model's UV map. After generating the textures and applying them to the model the final, textured version of the model can be rendered.
These objectives can be represented as a five step process:

1) Creating a standardised dataset of images and models from other datasets found online.
2) Creating a GAN model that can create original textures based on the dataset.
3) Creating a User Interface (UI) within the application that can be interacted with to specify how the textures should be applied.
4) Implementing within the application a feature that integrates the GAN model and user's input automatically apply the textures to 3D meshes.
5) Rendering the textured 3D model.

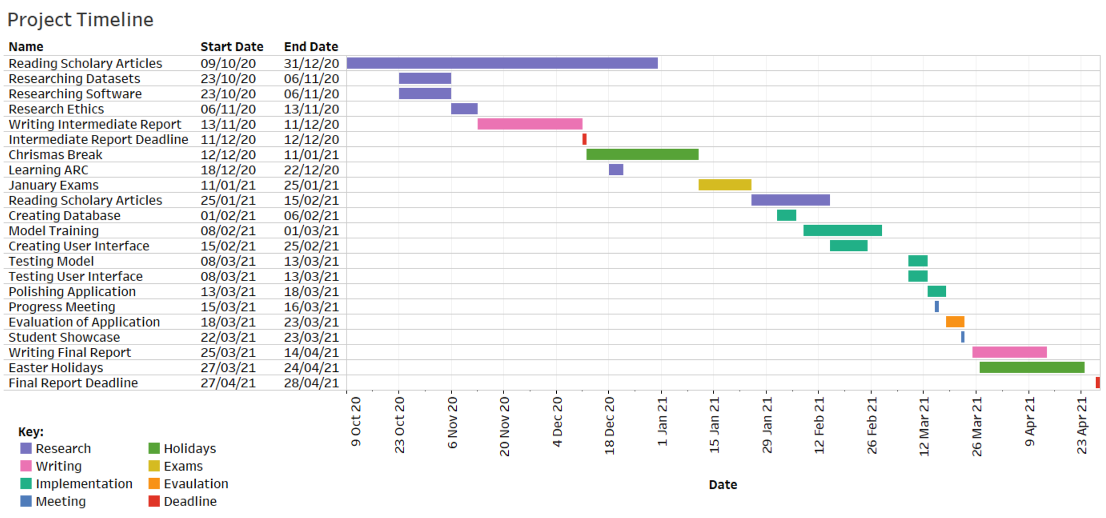
1.3 Deliverables

The final deliverables can be separated into six main parts:

- An executable application that satisfies the objectives listed in Objectives.
- The source code that was used to construct the user interface.
- The source code that was used to train the model.
- The source code that was used to build the database.
- A 'README' file with instruction on how to use the program.
- A final report that documents the planning, management, research and delivery for the project. The end of the report will also contain an evaluation via a self-appraisal.

The project can be separated into two main parts. The GAN model which generates new textures and the User Interface which the end-user interacts with to apply the textures to 3D meshes.

1.4 Initial Plan


Project Timeline

Name	Start Date	End Date
Reading Scholary Articles	09/10/20	31/12/20
Researching Datasets	23/10/20	06/11/20
Researching Software	23/10/20	06/11/20
Research Ethics	06/11/20	13/11/20
Writing Intermediate Report	13/11/20	11/12/20
Intermediate Report Deadline	11/12/20	12/12/20
Chrismas Break	12/12/20	11/01/21
Learning ARC	18/12/20	22/12/20
January Exams	11/01/21	25/01/21
Reading Scholary Articles	25/01/21	15/02/21
Creating Database	01/02/21	06/02/21
Model Training	08/02/21	01/03/21
Creating User Interface	15/02/21	25/02/21
Testing Model	08/03/21	13/03/21
Testing User Interface	08/03/21	13/03/21
Polishing Application	13/03/21	18/03/21
Progress Meeting	15/03/21	16/03/21
Evaluation of Application	18/03/21	23/03/21
Student Showcase	22/03/21	23/03/21
Writing Final Report	25/03/21	14/04/21
Easter Holidays	27/03/21	24/04/21
Final Report Deadline	27/04/21	28/04/21

Figure 1.1: A Gantt chart showing the initial plan for the project.

After conducting background research, an initial plan was devised. Some time will be allocated to learning ARC, Leeds' High Performance Graphics cluster over the Christmas period. After, a database will be created using a JetBrains' DataGrip. Multiple textures will be downloaded from the "Describable Textures Dataset" (DTD) [5] and the "Flickr Material Database" [20]. These textures will be inspected, sorted and then labelled with information about the textures. A table within the database containing references to these images files and their metadata will be created. Additionally, models will be downloaded from Princeton's ModelNet [22] and standardised so that they are triangulated. Another table will be created in the database with information regarding the files, their UV maps, and metadata for the models. Qt will be used to construct the User Interface and OpenGL will be used to render the 3D models within the User Interface. The pix2pix (source) deep learning framework, implemented in PyTorch, will be used to train and test the model. The this will take place using ARC. Once the model has been train, a User Interface can be created and integrated with the model.

1.4.1 Project Scope

One challenging aspect of this project is finding appropriate texture and model resources to use. The UV maps must be accessible beforehand, and a considerable amount of time may have to be spent pre-processing the resources so they are fit to be used on this project. For this reason and the fact that this project is an implementation of Exploratory software, the set of models that can be used by the application will be predefined instead of end-users being able to import their own model files.

1.5 Risk mitigation

1.5.1 GAN Model Creation

The GAN model will be trained on Leeds' High Performance Graphic Clusters (ARC). ARC is a shared system designed for batch processing of high throughput tasks [16]. Jobs are scheduled between users and are allocated based on availability, number of resources needed and user priority. This means that some time may lie between when the model is ready to train and when the training will actually take place This means that some forethought must be applied concerning how and when to train the model because the system is not readily available.
There are several ways to mitigate these problems. It would be best to train the model using the ARC system as few times as possible since the progress of the project is dependant on being allocated a job by the scheduler. Additionally, I have planned to train the model in early February, before starting any development on the UI. This way if any problems with the model are detected, then there is still time to fix them later on in the project.

1.5.2 Illness and Coronavirus

Due to the Coronavirus epidemic, the last year has been an unprecedented time. The University of Leeds has been impacted in a number of ways by this event. Some services within the University have been slower to respond due to changes in how staff and students have had to adapt to the current climate. This means that some resources may not be as readily available as in the past and response times may be longer than usual. In addition, there is an increased risk of illness within the academic community, that could serve as an obstacle to this project.
To mitigate this, additional time was added to each task in the initial plan. This means that if something unexpected happens to me or people that I am dependant upon in supporting me in this project, there will still be additional time to spend on this project to make up for it.

1.5.3 General Time Management

When devising the initial plan care was taken to ensure that it was a realistic plan and would not conflict with obligations to other modules and exams.

1.5.4 Version Control

There is always the possibility of some unforeseen circumstance (such as a corrupted hard drive or a fire) interfering with this project. For this reason, the version control software 'Git' will be used to manage the source code relating to this project. It will serve as a backup and will also allow for easy accessibility to prior versions. A link to the repository can be found here.

Chapter 2

Relevant Materials

2.1 Relevant Literature

2.1.1 Generative Adversarial Networks (GANs)

Generative Adversarial Networks were formally introduced in 2014. Unlike most other forms of machine learning, that are normally posed as optimisation problems, GANs are heavily based on game theory. A GAN is composed of two models which can be thought of as 'players', a generator and a discriminator. The generator aims to generate new data point based on the given input dataset, while the discriminator seeks to accurately identify the authenticity of the generated data points. This gives rise to an adversarial game between the two networks whereby the generator tries to generate new outputs that will deceive the discriminator.

The two networks are trained together, with each of the two players experiencing a cost indicated by a cost function relating to its performance. The generator tries to minimise cost by creating data points that are similar enough to the dataset that they are classified as genuine by the discriminator. All the while, the discriminator is trying to minimise its cost by correctly classifying the newly created data points. [7]

This method enables newly generated data points to be increasingly more authentic, while still being novel and distinct from the original dataset.

GANs are capable of imitating complex datasets such as dataset containing image files or audio waveforms. GANs have been demonstrated to be able to create realistic photographs [3] and new artworks [3] [11] [21]

2.1.2 Conditional Generative Adversarial Networks (cGANs)

Later in 2014, Conditional Generative Adversarial Networks were formally introduced. They work in a similar way to conventional, unconditional GANs but also allow for extra conditional data to be introduced to the model. This additional data is used to condition the GAN to output specific traits based on the conditions. The conditions are then inputted to both the generator and discriminator [15].

Pix2Pix [10] is a popular framework that uses CGAN technology to solve general-purpose image-to-image translation problems. In their paper, Isola et al. (2018) acknowledge how "a large number of internet users (many of them artists) have posted their own experiments with our system, further demonstrating its wide applicability and ease of adoption without the need for parameter tweaking" and demonstrate how GANs can make the creation of digital art more accessible. An interactive demo using four pre-trained models is available online [8].

2.1.3 Image to Image Translation

Image to image translation is a type of graphics problem that involves mapping an input image to an output image. CycleGAN [23] is a model that specialises in unpaired image translation. This means that we do not need a paired database, so this model can perform well even when a matched database does not exist. This is demonstrated by CycleGAN in an example whereby they transferred images of horses onto zebras and zebras onto horses from two separate datasets. Liu et al. (2018) [13] showed in their paper, titled "Unsupervised Image-to-Image Translation Networks", that scenes could be transformed from winter to summer scenes and vice versa.

Another use of image to image translation models is applying style transfers. CycleGAN showed that photographs could be rendered using the styles of various artists such as Monet, Van Gogh, Cezanne and Ukiyo-e [23].

2.1.4 Periodic Spatial GAN (PSGAN)

In a paper called, "Learning Texture Manifolds with the Periodic Spatial GAN", Bergmann et al. (2017) [2] showed how their PSGAN performed highly on periodic repeating textures found within the Oxford Describable Textures Dataset. The PSGAN was capable of learning the overall patterns of the texture images and could reproduce realistic outputs with much more authenticity than similar models.

As we want similar results that generate material textures that we can use to texture our UV maps. This paper may come in useful in helping to solve objective 2 declared in Objectives.

2.1.5 Semantic Layouts

In a groundbreaking paper, Park et al. (2019), revealed how photo-realistic images could be generated from basic semantic layouts [17]. The semantic layout was used as an input for the model. A label map was used to identify what type of object should be rendered, and its pixel position in the input image was used to acknowledge where the object should be rendered in the output image. This paper was a massive inspiration to this project and motivated us to use a semantic layout method of integrating multiple textures onto a UV map.

2.1.6 UV Mapping

A UV map is a representation of a 3D model's faces in a flat, 2D form. Texture mapping is performed so that the faces can be textured with 2D images. The relative parts of the map then correspond with the parts of the images. There are many different ways of generating UV maps, with some maps being preferable over others [14].

This is one of the more challenging aspects of the project and which is why we have decided to manually generate the UV beforehand and then implement them into the application. UV maps could automatically be generated but unfortunately with the time and resources allocated this is unfeasible.

2.2 APIs

2.2.1 Pix2Pix

The Pix2Pix model is a cGAN capable of performing a wide range of image to image translation tasks. It has demonstrated that it can generate images of street scenes or buildings from semantic layouts, accurately colour greyscale images, generate maps from satellite images, applying style transfers, and generate realistic images of objects from input sketches [10].
It is a powerful and relevant framework to use for a project such as this one.

2.2.2 PyTorch

PyTorch is a popular open-source machine learning framework that can be used for computer vision tasks. Pix2Pix has a version implemented with Pytorch so we have opted to use it for the sake on consistency [18].

2.2.3 NumPy

NumPy is a Python-based library used for performing numerical computation with multi-dimensional arrays. It is a useful library to use with machine learning projects and will simplify the project [6].

2.2.4 Qt

Qt is a Graphical User Interface (GUI) framework that allows for the creation of interactive GUIs. The application will require a GUI for the end-user to interact with. Qt was chosen for its simplicity and flexibility. Documentation and more information can be found on their website.

2.2.5 OpenGL

OpenGL (Open Graphics Library) cross-platform API for developing 2D and 3D graphical applications [9]. It is an environment we have experience using and was chosen since it is free, easy to use, well documented and is supported by Qt - via the Qt OpenGL module [19].

2.3 Resources

2.3.1 Textures

For the texture datasets, I will be using Oxford's "Describable Textures Dataset" (DTD) [4] since it is a well-labelled dataset "made available to the computer vision community for research purposes" [5]. It is a diverse dataset of 5640 images, separated into 47 categories. In addition I will be using the "Flickr Material Database" [20] a material dataset of 1000 images with 10 different categories. The size of these datasets are manageable but still large enough to obtain diverse output images via the GAN model.

2.3.2 3D Models

Models used from this project will be sourced from Princeton's ModelNet [22]. The dataset contains 127,915 3D models stored in Object File Format (OFF). The models are split into 662 categories. It is a particularly large dataset so it will be pruned to reduce the amount of hard drive space need to store the models. Additional information about ModelNet can be found on their website.

Chapter 3

Ethics

3.1 Ethical concerns

3.1.1 Offensive Images

There is a wide range of ethical concerns to consider for this project. As the image generation process is automated in this project by the model, it is important to be mindful of how the output images generated could be considered distasteful or offensive to some individuals. Earlier this year Antonio Torralba, Rob Fergus and Bill Freeman, researchers at the Massachusetts Institute of Technology, announced that they had decided to remove a dataset named TinyImages due to its usage of vulgar labels and offensive images. The dataset was a large dataset with 80 million images, each with the resolution of 32 by 32 pixels, that was automatically generated by web scraping. This aspect made it extremely difficult to manually remove offensive material leading to the team decided it was better to remove the dataset in its entirety [1]. This leads us to believe that we should exercise caution when choosing the datasets that will be used to train and test my model, given that some creators of particular datasets may have not exercised the best judgement when constructing their datasets from the data they had at their disposal. As it is not always possible to manually inspect a dataset (due to restrictions of resources such as time and manpower) it is best to take use dataset that come from reliable sources and have some understanding of how they were created. Failure to do this can lead to distasteful or obscene materials used to create the model, which in turn could lead to the model outputting obscene images.

3.1.2 Unintentional Outputs

Although not explicitly relevant to image generation, Microsoft's Tay – an AI launched on Twitter in 2016 marketed as a chat-bot – received mass criticism from the public and journalists alike due Tay's usage of racist and sexist slurs and various other insulting remarks made on the social media network. After two days Microsoft removed the AI from the platform and released an apology stating that "a coordinated attack by a subset of people exploited a vulnerability in Tay"[12]. Although this project is not designed to be a user on such an open platform like Twitter, this example still serves as a demonstration that not all end users will use the software presented in this project as intended and may seek to exploit bugs within the program, which would lead to a poor reflection on my work and possible perceived affiliations with insulting material.

3.1.3 Copyright and Intellectual Property

When planning the implementation of this project care was exercised in ensure no Copyright infringement took place with regard to software, APIs, and resources used. In addition, plagiarism was avoided at all cost and all usages of authors work has been cited see 'References'.

3.1.4 Social Implications

Automation can lead to rapid change in job markets and otherwise change people's lives significantly. The scope of this project is too small to drastically make any meaningful impact on a societal level.

3.1.5 Privacy Concerns

This project will be using images of textures using under the creative commons or with the permission of the creators. No infringement on people's human right to privacy will take place as no image of people will be used.

Glossary

Generative Adversarial Networks A machine learning approach that can generate new data points. For a more in depth description see Generative Adversarial Networks (GANs). 12, 14

User Interface A way in which a user and computer can interact with one another. 12, 14

Acronyms

GAN	Generative Adversarial Networks. 12
UI	User Interface. 12

References

[1] B. F. Antonio Torralba, Rob Fergus. TinyImages Apology, 2020. Accessed: 30/12/2020.

[2] U. Bergmann, N. Jetchev, and R. Vollgraf. Learning Texture Manifolds with the Periodic Spatial GAN, 2017. Accessed: 30/12/2020.

[3] A. Brock, J. Donahue, and K. Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis, 2019. Accessed: 30/12/2020.

[4] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. Accessed: 30/12/2020.

[5] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing Textures in the Wild Website, 2014. Accessed: 30/12/2020.

[6] T. S. community. NumPy Documentation, 2020. Accessed: 30/12/2020.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. Accessed: 30/12/2020.

[8] C. Hesse. Image-to-Image Demo, 2017. Accessed: 30/12/2020.

[9] T. K. G. Inc. OpenGL Overview, 2020. Accessed: 30/12/2020.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks, 2018. Accessed: 30/12/2020.

[11] T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks, 2019. Accessed: 30/12/2020.

[12] P. Lee. Learning from Tay's introduction, 2016. Accessed: 30/12/2020.

[13] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised Image-to-Image Translation Networks, 2018. Accessed: 30/12/2020.

[14] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping, 08 1993. Accessed: 30/12/2020.

[15] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014. Accessed: 30/12/2020.

[16] T. U. of Leeds. Leeds Arc, 2020. Accessed: 30/12/2020.

[17] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization, 2019. Accessed: 30/12/2020.

[18] PyTorch. Pytorch Documentation, 2020. Accessed: 30/12/2020.

[19] Qt. Qt OpenGL, 2020. Accessed: 30/12/2020.

[20] L. Sharan, R. Rosenholtz, and E. Adelson. Material perception: What can you see in a brief glance? *Journal of Vision - J VISION*, 14(9), 2014. Accessed: 30/12/2020.

[21] J. Simon. Artbreeder, 2019. Accessed: 30/12/2020.

[22] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes, 2014. Accessed: 30/12/2020.

[23] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2020. Accessed: 30/12/2020.