Synthetic Illusions using Pix2Pix and CycleGAN

Senai Leniston-Kahsai

Abstract

For this project a neural network was trained on a synthetic dataset rendered using Unity game engine software. Training using a synthetic data set allows for endless possibilities and the ability to selectively refine the final creative output. In this project a neural network was trained on image pairs of patterned 3D objects rendered using a game engine (target output) and a depth map of the rendered image as the translation layer (or input). The network was able to output images of 3D models with a pattern applied as a 3D effect when provided with a depth map of the model.

Introduction

Neural networks require a large amount of high-quality labelled data. Without the infrastructure and the data collection methods of large technology companies it can be difficult to source a large enough dataset to train a neural network effectively. Therefore, synthetic datasets are a powerful tool to disrupt this disadvantage. Training using a synthetic data set allows for endless possibilities and the ability to selectively refine the final creative output. In the literature synthetic data has been used to train a robotic arm to improve its capability pick up objects (Tobin et al., 2017) and used to improve bounding box detection of cars (Tremblay et al., 2018).

My initial inspiration for this project was to be able to create pieces of art based on optical illusion art such as below.

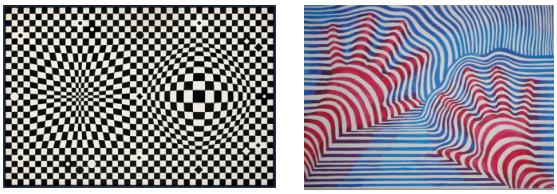

Figure 1: Optical art. VEGA III (left) (Vaserely, 1957) and Hand Op Art (right) (Chan, 2012).

A project called Neural Cities by Jasper van Loenen (2017) explored generative cityscapes using pix2pix image pairs of Google StreetView scenes and associated depth maps (Figure 2). Using new depth maps such as below right the cityscape was applied onto the object. A depth map is a grayscale image that contains information about the distance between the surface of objects from a given perspective.

Figure 2: Neural Cities was able to generate new images of the city of Arnhem given any depth map (van Loenen, 2017).

By utilising depth maps as an image translation layer it is hoped that the neural network learns information about depth and that affects the warping of a repeating pattern applied on a 3D model. This project will also showcase the promise of using synthetic data to train neural networks without the need for manually hand-labelling data.

Method

Dataset Generation

The generation of synthetic data was achieved using an open source package by Unity Game Engine, called 'Image Synthesis for Machine Learning' which was modified to suit the purposes of this project (Unity Technologies, n.d.). It is a sample project that can be used to build a synthetic image generator using the Unity camera to render images at runtime (Rodriguez, 2019). This tool is capable of rapidly generating large datasets of rendered images of 3D scenes as well as accompanying image depth (depth maps), segmentation maps (categorisation), optical flow (motion) and more as shown below. This allows simple generation of image pairs to use with Pix2Pix.

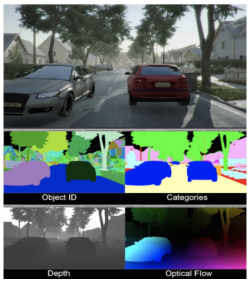

Figure 3: The repository provided by Unity can be customised to create a variety of image translation layers (Unity Technologies, n.d.).

Pix2pix was utilised as the implementation of image-to-image translation using a conditional adversarial network called CycleGAN (cGAN). Pix2pix works by training the neural network on a set of related input-output image pairs to generate the target output from the input (Isola et al., 2016). The aim for this project is that a neural network will be trained on image pairs of patterned 3D objects rendered using a game engine (target output) and a depth map of the rendered image as the translation layer (or input). The final aim of this project is that the network will be able to take input depth maps of any image or model and output an image with the pattern applied to create the optical effect above.

Results

Experiment 1: Randomly generated objects

Initial experiments were conducted using randomly generated spheres with a checkerboard pattern applied to both the spheres and the background. The following experiment was trained on 1000 image pairs for 200 epochs with the output shown in Figure 4 below.

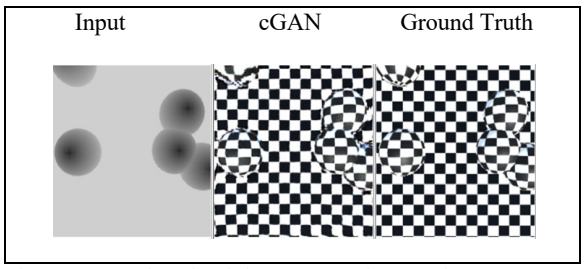

Figure 4: Results of training on 1000 images for 200 epochs.

Validation of the trained model (Figure 5) with unseen depth maps created using the synthetic image generator yielded promising results as the pattern was applied effectively. However, more detailed depth maps generated using Blender 3D modelling software were less promising. The model was only trained on spheres at a specific depth and therefore, more variation in depth information will likely provide more flexibility in the model.

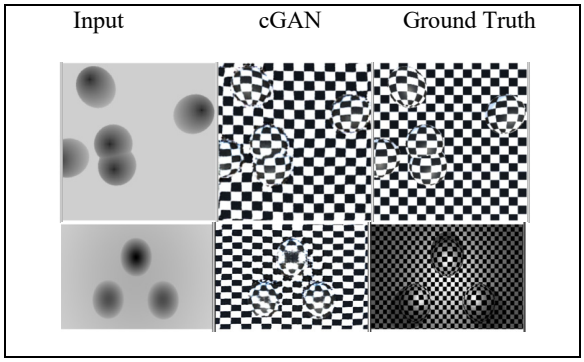

Figure 5: Validation of Experiment 1 model. Top row is validation data while bottom row is a detailed depth map rendered using 3D modelling software.

Experiment 2: Deforming mesh

The second major experiment involved warping a plane GameObject mesh with a checkerboard pattern using a ripple deform to create a depth map with more varied depth information as shown in Figure . This was to create more varied depth maps than simple spheres.

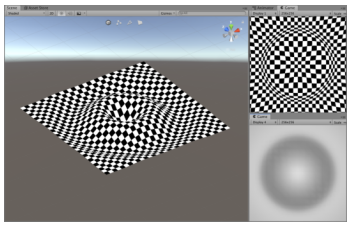

Figure 6: A dynamic ripple deformation was applied to a plane to provide more varied depth information.

The model was trained on 1000 images for 200 epochs (Figure 7).

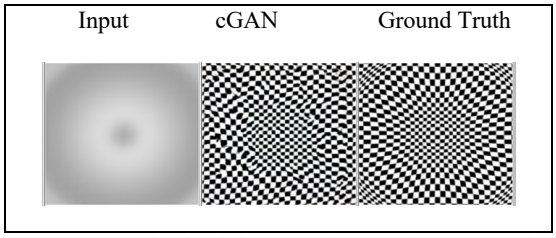

Figure 7: Result of training 1000 images for 200 epochs.

Testing on validation data created using the synthetic image generator yielded the results shown in Figure 8. The ripple effect allowed for more varied depth information. However, the depth maps were still too uniform and spherical. Therefore, in the next experiment the mesh was deformed in a more random fashion to create a more varied dataset.

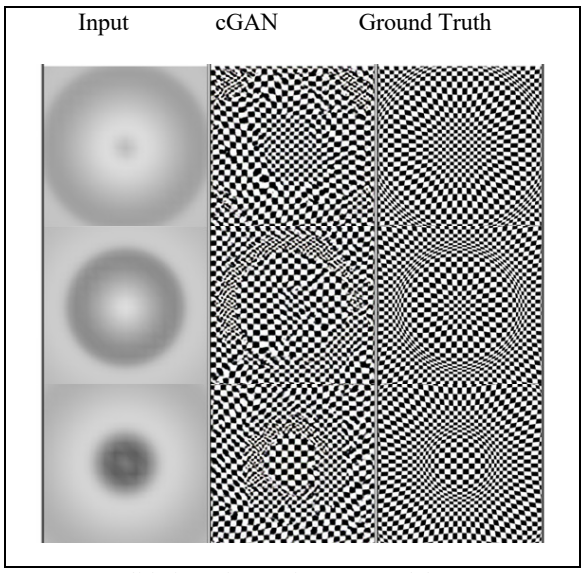

Figure 8: Validation of Experiment 2 model

Experiment 3: Pattern choice

For this experiment two different patterns (checkerboards and horizontal lines) were tested in both black and white and coloured formats with 1000 images in each dataset trained for 200 epochs each. The mesh deformations were varied by deforming the mesh using a sine wave deformation as well as introducing multiple ripple deformations that could be moved around at runtime using input from the user with the directional keys (Figure 9).

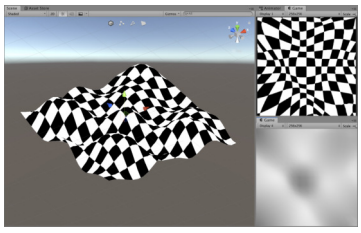

Figure 9: The plane mesh with multiple dynamic deformation effects applied in the Unity editor. The user can control the deformations using the arrow keys at runtime. The right of the image shows the output target image and associated depth map.

The results of training for the four datasets is shown below in Figure 10. The results are similar for both patterns regardless of whether there is colour or not. This is likely due to the simplicity and repetitiveness of the patterns.

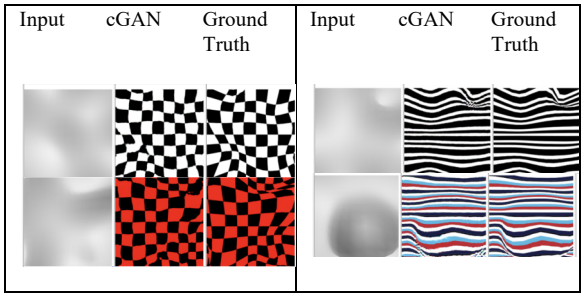

Figure 10: Results of training 1000 images for 200 epochs.

The goal of the project was to be able to apply a pattern to any depth map. Therefore, in addition to the validation data the model was also run on depth maps generated in Unity of a model of a human hand and a human head (Figure 11).

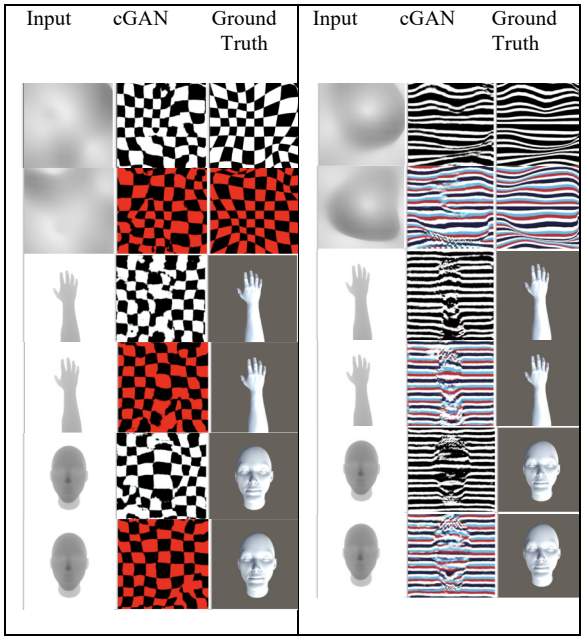

Figure 11: Validation of four datasets (top two rows) and models run on depth maps of hand and head 3D models.

The results of this experiment highlight that the horizontal lines are the optimal pattern for this dataset. The models trained on horizontal lines are able to capture the general shape of the models and apply the texture more accurately than the models trained on the checkerboard pattern. Therefore, the final step was to increase the dataset dramatically using one chosen pattern.

Experiment 4: Final results

The final experiment involved choosing the pattern that worked the best in Experiment 3 and training the model on 10,000 images. Two models were trained on 10,000 images: a locally trained model that ran for 8 epochs and a remotely trained model that ran for 200 epochs.

Although there are minor differences in the trained models at 8 and 200 epochs (Figure 12), the validation tests show that increased training led to a loss in the model's ability to generalise from input compared to the model trained for 8 epochs (Figure 13). This is likely due to the 200 epoch model memorising the data.

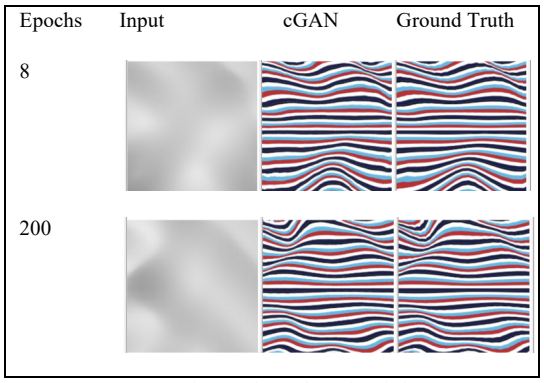

Figure 12: Comparison of results of training on 10,000 images for 8 and 200 epochs show little differences in outputs.

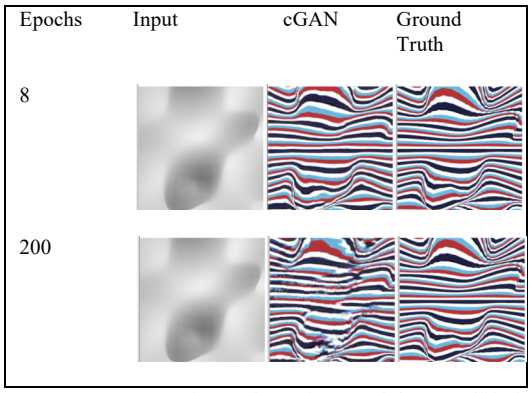

Figure 13: Comparison of running models on validation data highlights that the model run for 200 epochs is likely over-fit.

The model run for 8 epochs was used as the final model. Figure 14 shows the results of running the model on depth maps of models of a human hand and head in different orientations. The resulting outputs images were stitched together to create an animation. Further exploration of depth maps is shown in Figure 15 of the Appendix.

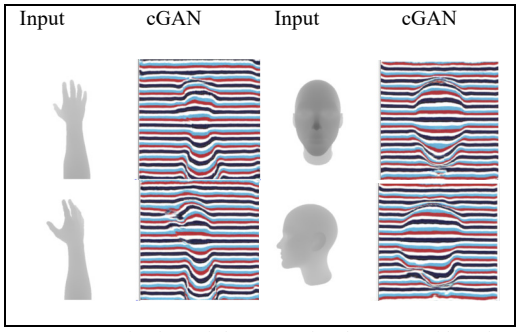

Figure 14: Final model run on depth maps of hand and head 3D models.

The final results are encouraging as the illusion has been applied seamlessly between the object and background as in the optical art. Since the mesh deformations are quite large in area it may be difficult for the network to discern dense detail in depth maps, for example the nose on the head model or the fingers of the hand. Despite this the patterns have been applied over the general shape of both models.

Discussion

The project was a success in proving that it is possible to train a model on depth information to apply patterns with a 3D effect to grayscale depth maps. This project could have unique applications in creating artworks using different patterns or optical illusions applied in 3D. A tool could be created where anyone could create a depth map of a 2D image using image editing software and apply a 3D texturing effect to that 2D image.

Using Unity's synthetic data generator tool has strengths and limitations. The depth maps rendered in Unity at runtime are not as detailed as depth maps that can be generated in 3D modelling software. However, the ability to create a large dataset rapidly is the major advantage of using this method. The amount of depth information in Unity can be increased by carefully setting the near and far clipping/focal planes of your scene effectively to capture the depth of surfaces in the scene.

The project has areas for improvement. This project was created using 256x256 resolution

images due to the limitations of the pix2pix implementation that was used. Outputs could be enhanced by using pix2pixHD which can use 2048x1024 high resolution images (Wang et al., 2018). It would be interesting to experiment further with creating variation in the patterns used in the training dataset to give the model more flexibility and usability. It would be a much more useful tool if different patterns could be applied. As identified the model run for 200 epochs was likely over-fitted to the data. Early stopping is a practice in machine learning used to prevent overfitting and reduce generalisation error i.e. errors in predicting an output from unseen data (Brownlee, 2019). Early stopping was inadvertently carried out in this project. However, it is possible to do this more systematically by monitoring training and stopping training based on a specific metric such as the loss function. Finally, in the future it would be interesting to have a video tool where the outputs could be generated in real time from a video of depth maps or even a black-and-white video (see Figure 16 in the Appendix).

Conclusion

This project showed a proof of concept of using Pix2Pix and cGAN that a pattern with 3D effects can be applied to 2D images when a neural network is trained on depth information in the form of depth maps. The project highlights the advantages of using synthetic datasets to conduct rapid experiments and create customised datasets.

References

Brownlee, J. (2019). A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks. Retrieved from https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

Chan, L. (2012). Hand Op Art. Retrieved from https://www.deviantart.com/x-luna-chan-x/art/Hand-Op-Art-305612272

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).

Rodriguez, P. (2019). Generating Synthetic Data for Image Segmentation with Unity and PyTorch/fastai. Retrieved from https://blog.stratospark.com/generating-synthetic-data-image-segmentation-unity-pytorch-fastai.html

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017, September). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 23-30). IEEE.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., ... & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 969-977).

Unity Technologies. (n.d.). Image Synthesis for Machine Learning. Retrieved from https://bitbucket.org/Unity-Technologies/ml-imagesynthesis/src.

van Loenen, J. (2017). Neural Cities. Retrieved from https://jaspervanloenen.com/neural-cities/

Vaserely, V. (1957). *VEGA III* [Painting, Kinetic art, Oil on canvas]. Guggenheim, New York, NY.

Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A., Kautz, J., & Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8798-8807).

Appendix

The final model was trained on images where surfaces closer to the near clipping/focal plane (the camera in Unity) are darker and surfaces further away are lighter. It is also possible to have depth maps with the opposite configuration (lighter surfaces at the near clipping plane and darker surfaces further away) as highlighted in Figure 15 below.

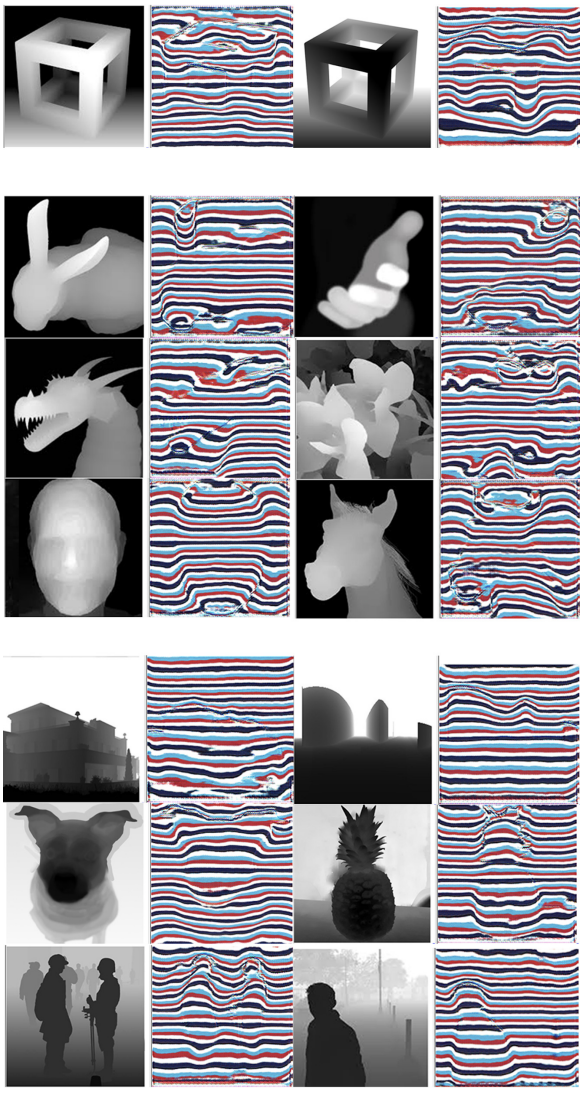

Figure 15: Final model run on a variety of depth maps

These results highlight that running the model on images using the opposite depth maps can yield unique results where the nearer surfaces have an inverted 3D illusion effect applied.

Another experiment was to run the model on black-and-white images. These black-and-white images contain no information about depth. The trained model will pick up on differences in lighting and shades of grey to create unique outputs such as in Figure 16 below.

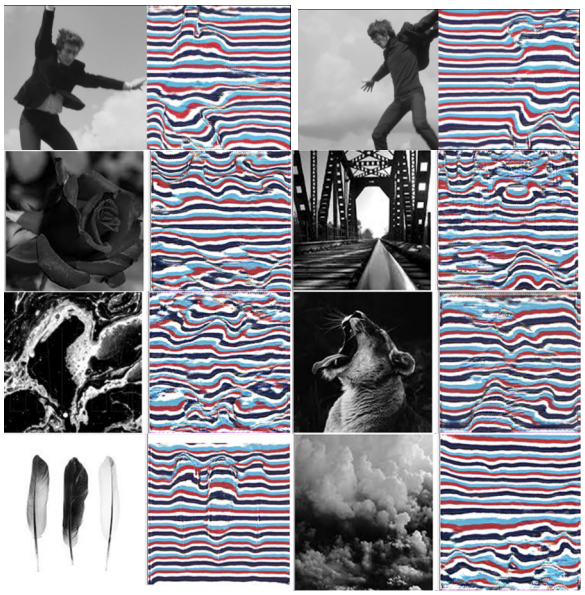

Figure 16: Final model run on a variety of black and white images

It is possible to run the model on images from a black-and-white video and then to stitch the output images together to recreate black-and-white videos with 3D effects applied.