

# Procedural 3D Terrain Generation Using Generative Adversarial Networks

Emmanouil Panagiotou\*  
panagiotouemm@gmail.com

School of Electrical and Computer Engineering, National  
Technical University of Athens  
Greece

Eleni Charou

exarou@iit.demokritos.gr

Institute of Informatics and Telecommunications, National  
Centre for Scientific Research Demokritos  
Greece



Figure 1: Procedurally generated samples of satellite images.

## ABSTRACT

Procedural 3D Terrain generation has become a necessity in open world games, as it can provide unlimited content, through a functionally infinite number of different areas, for players to explore. In our approach, we use Generative Adversarial Networks (GAN) to yield realistic 3D environments based on the distribution of remotely sensed images of landscapes, captured by satellites or drones. Our task consists of synthesizing a random but plausible RGB satellite image and generating a corresponding Height Map in the form of a 3D point cloud that will serve as an appropriate mesh of the landscape. For the first step, we utilize a GAN trained with satellite images that manages to learn the distribution of the dataset, creating novel satellite images. For the second part, we need a one-to-one mapping from RGB images to Digital Elevation Models (DEM). We deploy a Conditional Generative Adversarial network (CGAN), which is the state-of-the-art approach to image-to-image

translation, to generate a plausible height map for every randomly generated image of the first model. Combining the generated DEM and RGB image, we are able to construct 3D scenery consisting of a plausible height distribution and colorization, in relation to the remotely sensed landscapes provided during training.

## CCS CONCEPTS

• **Computing methodologies** → **3D imaging**; *Neural networks*; **Adversarial learning**.

## KEYWORDS

deep learning, general adversarial networks, satellite imagery, procedural generation, gaming, 3D point cloud, digital elevation models

\*Corresponding author

## 1 INTRODUCTION

Procedural content creation has been used in the past by game developers, as it can offer increased gameplay variety and replayability for the player, as well as lower budgets for gaming companies. Renowned games of different genres such as the Borderlands [19] and Civilization [20] series, Minecraft [11] and No Man’s Sky [21], apply analogous techniques. Procedural generation is an emerging research field on Artificial Intelligence (AI) & gaming, leading to various new state-of-the-art approaches [3, 23]. In most cases, developers create procedural characters, dungeons or landscapes by using predefined templates that can randomize some aspects of the generated object. As discussed in [10], games should take advantage of real-world information available on the internet. For our approach we generate random images that follow the distribution of real remotely sensed imagery. Particularly, we require a function

$$P : Z \rightarrow X$$

where  $Z$  is random noise and  $X$  is the generated image. To add a dimension of height to each pixel of the generated image, a one-to-one mapping  $G$ , generates a 3D point cloud or Digital Elevation Model (DEM) for each input tile  $X$ . Specifically,

$$G : X \rightarrow Y$$

where  $X$  is the domain of images produced by  $P$  and  $Y$  that of DEMs. Both tasks require a rule-based approach, as the generated input images, as well as the resulting one-to-one mappings are infinite. Obviously, both systems are impossible to "hard-code", therefore AI or Machine Learning (ML) models have to be implemented, as they can learn such rules in an automated, data-driven manner. In particular, Deep Learning (DL), the data-intensive version of ML, has recently been proven to be useful for many difficult problems. Especially in image processing tasks for computer vision [5, 9, 22], specific DL algorithms are the go-to solutions. Consequently, we propose a DL method for procedural 3D scenery generation that is data driven and relies solely on real remotely sensed imagery, with no need of any input from the developer. The model succeeds in replicating the input data distribution, generating images and 3D representations of increased variation and high quality. The code for our work has been made publicly available at <https://github.com/Panagiotou/Procedural3DTerrain>.

## 2 DEEP LEARNING TECHNIQUES FOR IMAGE PROCESSING

In this section, we provide necessary context to the information discussed throughout our research paper.

### 2.1 Typical Convolutional Architecture

A Convolutional Neural Network (CNN) is a (deep) neural network consisting of an input layer, multiple hidden layers and an output layer. The first layer, expects an image as input, which is passed to the next layers. Every hidden layer is comprised of convolutional layers that convolve the input by applying a dot product with a kernel consisting of trainable weights. The resulting output is passed by a pooling layer that reduces the input dimensions for the next layer. The output layer computes the error of the predicted output in relation to the expected ground truth values and backpropagates

that error to previous layers, updating the trainable weights accordingly. Compared to standard feedforward neural networks, CNNs are able to make strong hypothesis regarding the nature of the images as they take the 2D structure into account, thus using much fewer connections and parameters, leading to faster training times [9].

### 2.2 Generative Adversarial Networks

Generative Adversarial Networks (GAN) [1] constitute a general framework for training generative models, i.e. models that can produce samples, not only differentiate between them. GANs consist of a generator  $G$  and a discriminator  $D$ , both modeled as artificial neural networks. The generator is optimized to reproduce the true data distribution  $p_{data}$ , which can be fixed to the distribution of interest, by generating images (or any form of data) that are difficult for the discriminator to differentiate from the real images, namely the actual data distribution  $p_{data}$ . Simultaneously, the discriminator is tasked with differentiating real images from synthetic data generated by  $G$ . Their training procedure is a minimax two-player game with the following objective function:

$$\min_G \max_D V(D, G) = \int_{x \sim p_{data}(x)} [\log D(x)] + \int_{z \sim p_z(x)} [\log (1 - D(G(z)))] \quad (1)$$

where  $z$  is a noise vector sampled from a prior noise distribution of choice  $p_z$ , usually a uniform or a normal distribution, and  $x$  is a real image, from the data distribution  $p_{data}$ . [1] prove that, given enough capacity, the generator can learn to replicate the true data distribution.

### 2.3 Conditional Generative Adversarial Networks

As suggested in [1] and first examined in [12], CGANs can extend GANs by incorporating additional information, like a class label or, analogous to our case, extracted features, in effect conditioning the generator and the discriminator to it. Denoting the additional conditioning variable as  $c$ , we can substitute  $D(x)$  and  $G(z)$  from Equation 1 with  $D(x|c)$  and  $G(z|c)$ , whereas the rest of the formulation remains the same:

$$\min_G \max_D V(D, G) = \int_{x \sim p_{data}(x)} [\log D(x|c)] + \int_{z \sim p_z(x)} [\log (1 - D(G(z|c)|c))] \quad (2)$$

By conditioning on  $c$ , we can control the quintessence of the output of the generator, allowing the noise  $z$  to add background information, pose, etc [4, 17, 24, 25].

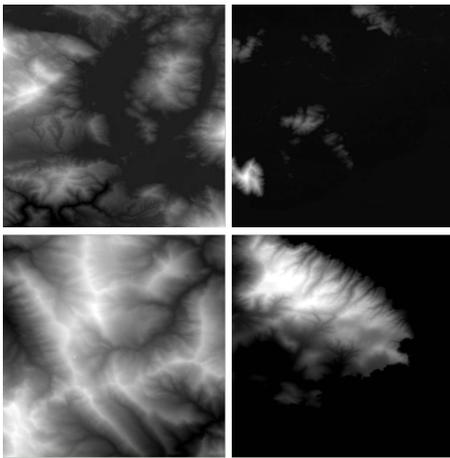
## 3 DATASET

In order for a CNN architecture to be trained, a large-scale dataset is imperative, as well as computing power to process it, preferably with the parallel processing capabilities of a Graphics Processing Unit (GPU). This especially holds in our case, where the objective is to train a GAN architecture for generating a vast variety of random images. Our second task consists of performing an image-to-image translation from those generated images to their corresponding DEMs. During this process the DEM is interpreted as a single band

(grayscale) image. Evidently, a dataset of pairs of RGB satellite images and their corresponding DEM images is needed. As we were unable to acquire data containing both RGB and DEM images, we decide to build our own. To be more precise, a large area over Greece was selected as our region of interest (ROI). The DEM images corresponding to our ROI are provided by ALOS Global Digital Surface Model "ALOS World 3D - 30m (AW3D30)" [7] and can be granted with a request to the respective owners. We then split the DEMs into smaller tiles and, for each tile, a script obtains the corresponding RGB tile. In particular, the program extracts a GeoJSON polygon from the georeferenced DEM tile and feeds it to the Google Earth Engine API [2], which is publicly available. This, then, returns the true color bands [TCI\_R, TCI\_G, TCI\_B] Sentinel-2 MSI, which, when stacked, yield the requested RGB satellite image corresponding to the input DEM. To get the final dataset we reshape our data so that all tiles are  $256 \times 256$  pixels.



(a) Satellite Images



(b) DEMs

Figure 2: Example (a) satellite images and (b) their corresponding DEM tiles over different locations of Greece.

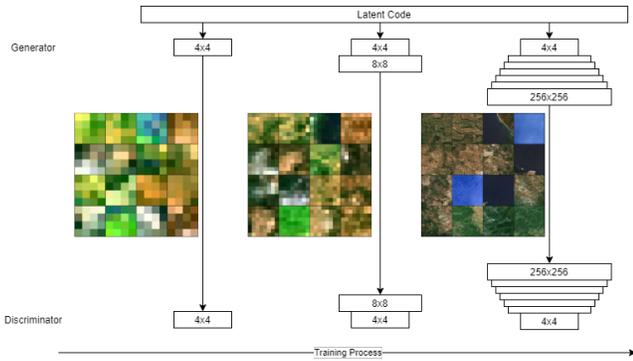
The overall process is graphically presented in Figure 3. Some pairs of the dataset can be seen at Figure 2. As a preprocessing step, we project the DEMs to the  $[-1, 1]$  range, as each tile was scaled to the  $[-1, 1]$  range according to the global minimum-maximum of the entire dataset.



Figure 3: Flow chart of the satellite-imagery dataset collection process using the Google Earth Engine API. The outermost points of the georeferenced DEM are selected as the boundary for the GeoJSON Polygon, which when fed into the API returns the corresponding RGB satellite image.

#### 4 GENERATIVE ADVERSARIAL NETWORK FOR SATELLITE IMAGE GENERATION

As aforementioned, the first step in procedurally producing random 3D landscapes, is generating random images that mimic the real satellite images of the dataset. While attempts for lower resolution image generation [1, 15] have been successful, researchers discovered a difficulty in convergence mainly for higher resolutions. This effect called "mode collapse" occurs when the discriminator, at some point, wins the minimax game resulting in non-convergence of the generator, who starts producing similar results for every input sample. In our case, we choose to construct images of size  $256 \times 256$ . Therefore, we choose to implement a technique of progressive growing GANs (ProGAN) introduced in [8]. This architecture, allows training to occur in multiple stages. Instead of training all layers of the generator and discriminator models, ProGANs are trained one layer at a time, leading to exponential growth of the generated images on every step. This method, proves to be very effective in stabilizing the training process and reducing its duration, leading to convergence and producing images of high resolution at the same time. The increase in resolution is achieved by adding new layers to both networks as seen in Figure 4.

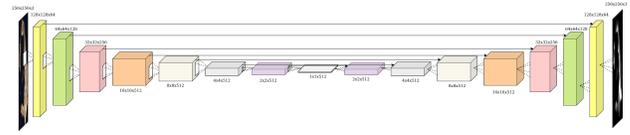


**Figure 4: Flow chart of the training process. Both the Generator and the Discriminator start with a low resolution of  $4 \times 4$ . Size is advanced exponentially, until the target distribution of  $256 \times 256$  is reached.**

The weights of all previous layers, remain trainable during this process and for the model to avoid shocks during this transition, new layers are *faded in* gradually. This process of fading in a new layer, is controlled by a parameter  $\alpha$  ranging from 0 to 1 over the course of multiple iterations, producing a weighted sum of the two last layers of the generator. The discriminator can be regarded as a symmetrical copy of the generator. Input images are either "fake" images synthesized by the generator or real images of the dataset, obviously scaled down to the current training resolution. Through a series of convolutional layers, the image is downscaled, until the last layer, where a boolean decision is returned.

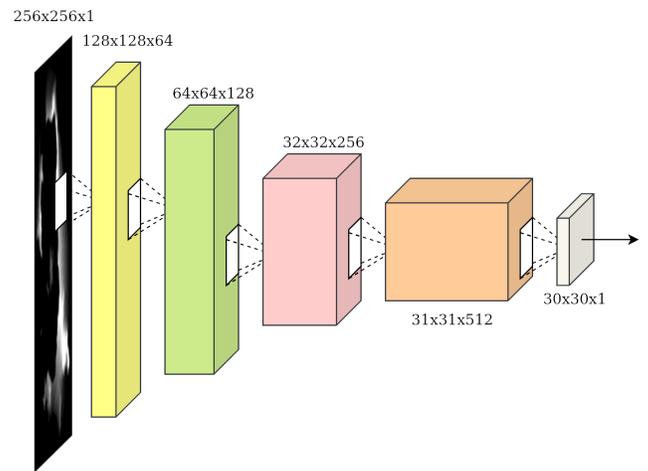
## 5 CONDITIONAL GENERATIVE ADVERSARIAL NETWORK FOR ELEVATION PREDICTION

Following [6], we use the pix2pix architecture to train the CGAN framework. In particular, we use an encoder-decoder architecture, described as U-net in [18] for the generator. This model, first downsamples the conditioning input (e.g. satellite) image down to a bottleneck layer using a series of convolutional layers. Afterwards, through a series of deconvolutions, roughly the inverse operator of the convolution, the images are upsampled, decoding the bottleneck code to the size of the output image. Every convolutional layer is connected with a skip connection to its respective deconvolutional layer, helping the model to converge during training since it skips some layers by feeding the output of one layer as the input to next layers [26], which facilitates training, provided the global, low-level structure is the same between input and output, as is the case in our task. The architecture of the U-net can be seen in Figure 5.



**Figure 5: The generator architecture of choice: U-net [18]. It consists of an encoder that downsamples the input image using convolutional blocks up until the bottleneck layer. Thereafter, deconvolutional blocks upsample the image to the desired dimensions. The skip connections, denoted by pointed arrows between corresponding layers of the encoder and the decoder, facilitate training by providing crucial lower-level information from the encoder to the decoder. Given that the input and the output have the same low-level structure, these low-level features serve as the canvas that guides the decoder in the generation of the final output.**

The discriminator model is a binary classifier, deciding whether a given image (e.g. DEM) has been produced by the generator, or belongs to the real images provided by the dataset. Deep CNNs have been heavily tested and therefore proven to work on image classification tasks [16]. In our case, a PatchGAN [6] is used. The main difference is that the traditional CNN architecture would come to a decision based on the whole input image, whereas the PatchGAN maps the  $256 \times 256$  image, in our case, to a square array of outputs. Each output "pixel" signifies whether the corresponding patch is real or fake.



**Figure 6: The discriminator architecture of choice: PatchGAN [6]. The discriminator decides whether its input is from the true data distribution based on local information by concentrating on the fidelity of individual image patches. Convolutional and pooling layers are applied to reduce the dimensions of the input images.**

The final decision for the whole image is derived by averaging over all the individual patches. Using a Patch-based approach for the Discriminator, compared to a traditional CNN architecture for binary image classification, has proven to encourage high frequency crispiness in the resulting images [6]. The PatchGAN architecture can be seen in Figure 6.

The task of predicting plausible DEMs for input remotely sensed imagery, as well as, model evaluation and accuracy have been addressed thoroughly in our previous work [13].

## 6 RESULTS

We first present our results of the ProGAN model Figure 8a. It is evident that random RGB satellite images of great resolution and variety are being generated. The DEMs produced by the CGAN model, presented in Figure 8b, render a plausible representation in relation to the input images, as well as the data distribution of DEMs provided during training. We observe that the ProGAN model, by progressively growing the size of the output image, has learned to generate sharp results that imitate images that are present in our dataset. Various basic elements like river banks, islands with greener water near the surface and snow, are present. Likewise, the CGAN model produces detailed and accurate DEMs, resulting in plausible 3D representations, Figure 7.

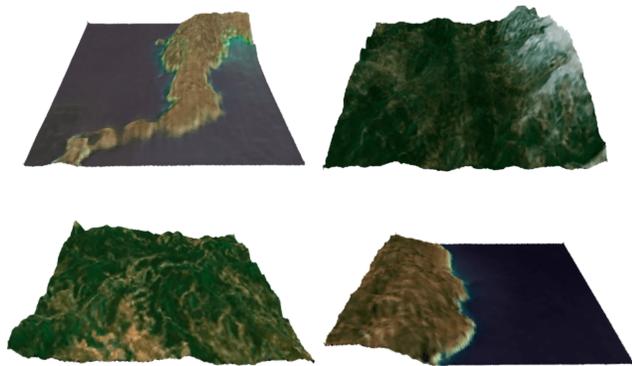
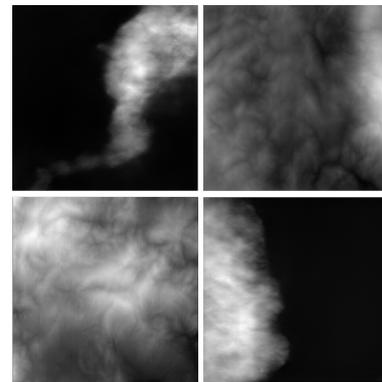


Figure 7: A 3D visualization of the generated landscapes produced by both models.



(a) Satellite Images



(b) DEMs

Figure 8: Procedurally generated (a) satellite images and (b) respective DEM tiles produced by the CGAN. Images produced are diverse and at the target resolution of  $256 \times 256$ .

## 7 DISCUSSION-FUTURE WORK

While individual results of our approach presented in Figures 8 and 7, are remarkable, an emerging problem is choosing neighboring tiles. In particular, while game content is generated and if the game is infinite-world, every tile needs to have 8 neighboring tiles. This process of choosing appropriate tiles, is left for future research, but one approach could be using images produced by latent codes close to the one which produced the center tile. Close latent points, in our case, are similar noise vectors, which therefore produce similar images. One can then create a linear interpolation between a starting and a target image, like the one presented in Figure 9.

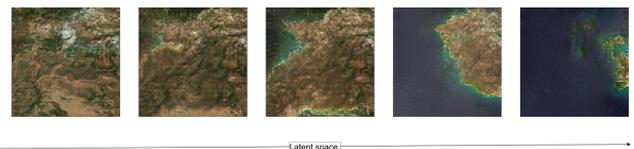


Figure 9: A sparse interpolation in Latent Space.

A more lightweight solution for producing 3D landscapes introduced in [13] is to use a single CGAN model, solving the inverse problem i.e., train the inverse operator,  $G^{-1}$ , to predict the surface coloration, meaning the RGB image, conditioned on a DEM. In this case, a random  $256 \times 256$  DEM tile is sampled from a Perlin noise distribution [14], which is especially suited for generating plausible landscapes with peaks and valleys.



**Figure 10: The well known technique of generating DEMs with random Perlin noise, is enhanced by adding plausible colors to the random DEM, using the trained inverse CGAN model.**

In conclusion, an idea left for future work, is to implement a global model combining the scopes of both models, e.g. generating random satellite imagery while producing a plausible DEM representation. This model will have to minimize a combined loss for both problems, probably leading to difficulty in convergence, but would likely yield more realistic and robust results.

## REFERENCES

- [1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.
- [2] Noel Gorelick, Matt Hancer, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment* 202 (2017), 18–27.
- [3] Daniele Gravina, Ahmed Khalifa, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. 2019. Procedural content generation through quality diversity. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–8.
- [4] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. 2016. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579* (2016).
- [5] Forrest N Iandola, Song Han, Matthew W Moskevich, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).
- [6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.
- [7] JAXA-EORC. 2016. ALOS Global Digital Surface Model "ALOS World 3D - 30m" (AW3D30). <http://www.eorc.jaxa.jp/ALOS/en/aw3d30/index.htm> (2016).
- [8] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *arXiv:1710.10196 [cs.NE]*
- [9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
- [10] Antonios Liapis. 2018. Real-world data as a seed, The Procjam Zine., 3 (2018), 35–40.
- [11] Minecraft. 2009. Xbox 360. (2009).
- [12] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
- [13] Emmanouil Panagiotou, Georgios Chochlakis, Lazaros Grammatikopoulos, and Eleni Charou. 2020. Generating Elevation Surface from a Single RGB Remotely Sensed Image Using Deep Learning. *Remote Sensing* 12, 12 (2020), 2002.
- [14] Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.
- [15] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
- [16] Waseem Rawat and Zenghui Wang. 2017. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation* 29, 9 (2017), 2352–2449.
- [17] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396* (2016).
- [18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.
- [19] Borderlands (series). 2009–2019. PlayStation 3. (2009–2019).
- [20] Civilization (series). 2009. Microsoft Windows. (2009).
- [21] No Man's sky. 2018. PlayStation 4. (2018).
- [22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
- [23] Julian Togelius, Georgios N Yannakakis, Kenneth O Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 3 (2011), 172–186.
- [24] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiao lei Huang, and Dimitris N Metaxas. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 5907–5915.
- [25] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiao lei Huang, and Dimitris N Metaxas. 2018. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1947–1962.
- [26] Xiaobin Zhu, Zhuangzi Li, Xiaoyu Zhang, Haisheng Li, Ziyu Xue, and Lei Wang. 2018. Generative Adversarial Image Super-Resolution Through Deep Dense Skip Connections. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 289–300.