Terrain Design Prototyping With Diffusion Models

Hajime Fukai s1290102 Supervised by Prof. Maxim Mozgovoy

Abstract

Video games often feature a vast expanse of game world and such games are called *open-world* games. The objective of this study is to aid the prototyping of world map creation of such games. We have attempted to achieve this objective by creating terrain generation models. We have trained two distinct image-to-image translation models: pix2pix and Palette. The experiment results have shown that pix2pix requires less computing resources to train, and it is capable of generating valid and natural terrain than Palette.

1 Introduction

Today, the *open-world* games are becoming popular in the video game industry. The open-world games are the video games which feature a vast expanse of game world. One of the characteristics of open-world games is the degree of freedom; players can enjoy the game in their own way. In order to achieve this freedom, it is necessary to provide the players with a sufficiently large world map.

There are two typical approaches to the implementation of the open-world games.

The first approach is to use manually designed world maps. In this approach, the world data is designed and created by the developer, and it is stored alongside the game program. The advantage of this approach is that the developer has complete control of the game world, therefore the developer can tailor it to the desired game experience. Examples of games using this approach include *The Legend of Zelda: Breath of the Wild* (2017) and *Grand Theft Auto V* (2013).

The second approach is to generate world maps in real time using a deterministic, or non-deterministic algorithm. This approach is known as *procedural content generation* (PCG) of world maps. Practically, this approach is able to create an infinitely large world, making an endlessly explorable game world possible. However, sometimes it is difficult for the developer to predict the outcome of the algorithm if the algorithm is non-deterministic. Examples of games using this approach include *Minecraft* (2011) and *No Man's Sky* (2019).

In this study, we focus on the former approach of the open-world game implementation, namely the manually designed world maps. In this case, the developer of the game must create the world map. However, creating such a world map, realistic terrain in particular, is a difficult task for nonexperts since it is required to know the characteristics of the real terrain.

The objective of this study is to aid the prototyping of world map creation, by creating terrain generation models. These models take a hand-drawn contour sketch as input, and generate terrain corresponding to the sketch in the form of an elevation map. We train each model with real terrain dataset, so that it can generate realistic, natural terrain. Such models enable nonexperts to create a world map easily and quickly, allowing rapid prototyping of a world map.

Traditionally, this sort of procedural content generation tasks have been implemented using a Generative Adversarial Network (GAN) [2]. On the other hand, recent advancement in image synthesis has proven the advantages of diffusion models [3] [7]. Therefore, we investigate the applicability of diffusion models to the procedural content generation by comparing with a traditional GAN.

2 Background

2.1 Sketch2Map

Sketch2Map [9] is a framework for sketch-based world map design prototyping proposed by Tong Wang et al.

Sketch2Map features a two-stage generative model. The first stage is the conditional generative adversarial network (conditional GAN) model which converts a sketch into an elevation map, and the second stage is a deterministic algorithm generating a level asset from the map. The training dataset of a conditional GAN consists of data generated by a random map generator and the real world elevation data.

Our study has drawn some inspiration from their work, such as the application of conditional GANs into terrain generation.

2.2 Conditional Generative Adversarial Network

A conditional generative adversarial network (conditional GAN) [4] is a general-purpose image-to-image translation model proposed by Phillip Isola et al. in 2018. A conditional GAN is a variant of a generative adversarial network (GAN) [2] and it learns the mapping from an input image to an output image. The implementation of a conditional GAN is released under the name of *pix2pix*.

A conditional GAN is a mapping from an observed image $\boldsymbol{x}$ and a random noise vector $\boldsymbol{z}$ to an output image

$\boldsymbol{y}$, namely $G : \{\boldsymbol{x}, z\} \to \boldsymbol{y}$. The loss function of a conditional GAN is

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

where $\mathcal{L}_{cGAN}(G, D)$ is a loss function

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =&\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\log D(\boldsymbol{x}, \boldsymbol{y})] + \\ &\mathbb{E}_{\boldsymbol{x},z}[\log(1 - D(\boldsymbol{x}, G(\boldsymbol{x}, z)))]\end{aligned}$$

and $\mathcal{L}_{L1}(G)$ is an L1 distance

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y},z}[\|\boldsymbol{y} - G(\boldsymbol{x}, z)\|_1].$$

2.3 Palette

Palette [8] is an image-to-image translation model based on the diffusion models [3], proposed by Chitwan Saharia et al. in 2022.

Generally, the diffusion models are formulated as the process of adding Gaussian noise to the given image (forward process) and the process of denoising the given noisy image (reverse process). The diffusion models generate images from a noisy image $\boldsymbol{y}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ by removing the noise iteratively.

The loss function of Palette is

$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\mathbb{E}_{\gamma}\left\| f_\theta(\boldsymbol{x}, \sqrt{\gamma}\boldsymbol{y} + \sqrt{1-\gamma}\boldsymbol{\epsilon}, \gamma) - \boldsymbol{\epsilon}\right\|_p^p$$

where $\boldsymbol{x}$ is the conditioning image, $\boldsymbol{y}$ is the ground truth image, γ is the noise level indicator and f_θ is the neural network predicting the noise $\boldsymbol{\epsilon}$. $\tilde{\boldsymbol{y}} = \sqrt{\gamma}\boldsymbol{y} + \sqrt{1-\gamma}\boldsymbol{\epsilon}$ is a noisy image created from $\boldsymbol{y}$.

2.4 NASADEM

NASADEM [6] is a global elevation dataset published by Land Processes Distributed Active Archive Center (LP DAAC) and it was derived from the data obtained in Shuttle Radar Topography Mission (SRTM) as well as Terra Advanced Spaceborne Thermal and Reflection Radiometer (ASTER) Global Digital Elevation Model (GDEM) and Ice, Cloud, and Land Elevation Satellite (ICESat) Geoscience Laser Altimeter System (GLAS) to improve the accuracy of the data.

NASADEM_HGT is a part of NASADEM dataset and each region of the dataset represents an area of 1 degree latitude by 1 degree longitude. It has 3 channels in total. The *hgt* channel represents the elevation in meters. This elevation is relative to the EGM96 geoid. The *swb* channel represents the water body; 0 for land, 255 for water. The *num* channel indicates the source of the data.

3 Method

3.1 Dataset Structure

Our dataset consists of pairs of a height-map and a contour-map.

A height-map is a 256x256 PNG image representing terrain. The red and green channels of a height-map represent the height of the point. The green channel is positive if the point has positive height, and the red channel is positive if the point has negative height. If the height of the point is 0, then these channels are 0. The red channel and the green channel can not be positive at the same time. The blue channel is 255 if the point is water, 0 if the point is land.

A contour-map is a 256x256 PNG image representing a sketch for the corresponding terrain. A contour-map has three kinds of solid lines and the background is black. A green line signifies the enclosed area has higher elevation than the area around the line. In contrast, the red line signifies lower elevation. A blue line signifies the boundary between water and land.

3.2 Dataset Creation

We have created our dataset from NASADEM_HGT v001 [6].

Algorithm 1 is the pseudo-code for creating a height-map and Algorithm 2 is the the pseudo-code for creating a contour-map from a given NASADEM_HGT region. Examples of created dataset are shown in Figure 1.

3.2.1 Height Relativization

In order to augment the locality of the dataset, we reduce the DC component of the height from each region. This is achieved by making the height relative from the *base-height* of the region. The pseudo-code for height relativization is shown in Algorithm 3.

3.2.2 Blurring

We applied OpenCV's Gaussian blur to the height and the water body mask of each region so that we can make the contour-map smoother. The kernel size of the blur is proportional to the square root of the land area, so that the blur does not oversimplify the coast line. The detail of this process can be found in Algorithm 4.

3.2.3 Height Stratification

Height stratification is a process of rounding the height value in each region. This process is necessary for making contour lines consistent in the following step. The pseudo-code for height stratification is in Algorithm 5. In the following, we call the connected area having the exact same height a *stratum*.

Algorithm 1 Create height-map from *dem_region*

```
function CREATE_HEIGHT_MAP($dem\_region$)
    Resize $dem\_region$ into 256x256
    Relativize height of $dem\_region$
    Render height-map of $dem\_region$ as PNG image
end function
```

Algorithm 2 Create contour-map from *dem_region*

```
function CREATE_CONTOUR_MAP($dem\_region$)
    Resize $dem\_region$ into 1800x1800
    Blur $dem\_region$
    Relativize height of $dem\_region$
    Stratify $dem\_region$
    $contour\_graph$ := the result of contouring $dem\_region$
    Color edges of $contour\_graph$ using stratified $dem\_region$
    Render edges of $contour\_graph$ in 256x256 PNG image
end function
```

Algorithm 3 Relativize height of *dem_region*

```
function MEASURE_SIDE_RATIO($dem\_region$,$upper\_bound$)
    return the ratio of pixels on the sides of $dem\_region$ where its height is lower or equal to $upper\_bound$
end function
function DETERMINE_BASE_HEIGHT($dem\_region$)
    if $dem\_region$ has water then
        return minimum point of water
    else
        return minimum $h$ which satisfies MEASURE_SIDE_RATIO($dem\_region, h$) $\geq 0.5$
    end if
end function
function RELATIVIZE_HEIGHT($dem\_region$)
    $h$ := DETERMINE_BASE_HEIGHT($dem\_region$)
    for all y do
        for all x do
            $dem\_region.hgt[y][x] = dem\_region.hgt[y][x] - h$
        end for
    end for
end function
```

Algorithm 4 Blur *dem_region*

```
function ODDIFY($n$)
    return n + (n % 2) + 1
end function
function BLUR($dem\_region$)
    a := the area of the land in $dem\_region$
    $land\_kernel$ := ODDIFY(floor($\sqrt{a}/4$))        ▷ OpenCV's GaussianBlur requires odd kernel size
    Apply Gaussian blur to $dem\_region.hgt$ with kernel size ($land\_kernel, land\_kernel$)
    $water\_kernel$ := ODDIFY(floor($\sqrt{a} \times 3/32$))
    Apply Gaussian blur to $dem\_region.swb$ with kernel size ($water\_kernel, water\_kernel$)
    Binarize $dem\_region.swb$ with threshold $4/16 \times 255$
end function
```

Algorithm 5 Stratify *dem_region*

```
1: function STRATIFY($dem\_region$)
2:     $levels := \{\dots, -e^6, -e^4, -e^2, 0, e^2, e^4, e^6, \dots\}$ where $e$ is the base of the natural logarithms
3:     for all y do
4:         for all x do
5:             $dem\_region.hgt[y][x]$ = the greatest element in $levels$ less than or equal to $dem\_region.hgt[y][x]$
6:         end for
7:     end for
8: end function
```

3.2.4 Contouring

We used CONREC [1] to contour the given height-map. $\{\dots, -e^6, -e^4, -e^2, e^2, e^4, e^6, \dots\}$ was used as contour levels parameter of CONREC. The result of CONREC is a list of edges on the boundary between contour levels.

The boundaries between land and water are also identified by CONREC, and those edges are colored blue.

3.2.5 Contour Edge Coloring

The contour edges obtained in the previous step are colored by performing breadth-first search on stratified height-map. The pseudo-code is for contour edge coloring is in Algorithm 6.

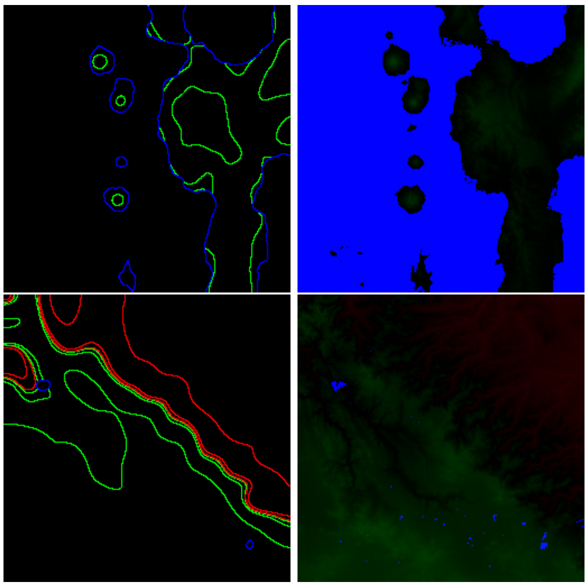

Figure 1: Created dataset

4 Experiments

4.1 Generative Models

We have trained two distinct models for performing translation between a contour-map and a height-map: Palette and pix2pix. These two models are trained with the aforementioned dataset to generate a corresponding height-map conditioned on a contour-map. We have trained both models for 1200 epochs.

4.2 Hand-drawn Contour Sketches

We have created hand-drawn contour sketch dataset for testing the trained models. This dataset consists of 141 contour-maps drawn by hand. The number of data is augmented by flipping and rotating these hand-drawn contour-maps, resulting in a total of 1128 contour maps. Examples of this dataset are shown in Figure 2. We have used this dataset for conducting the following evaluations.

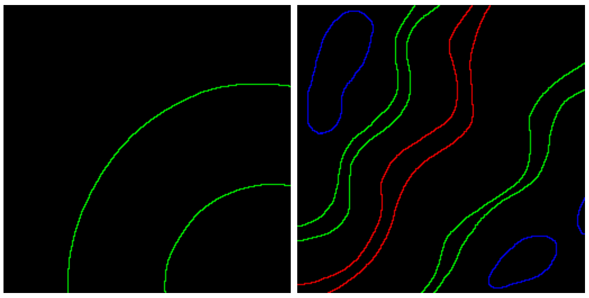

Figure 2: Hand-drawn contour sketches

4.3 Validity Evaluation

In the validity evaluation, we test if the output images are semantically valid as height-maps defined in Section 3.1.

Height-maps in our dataset always satisfy

$$\forall y \forall x \min(r_{x,y}, g_{x,y}) = 0$$

where $r_{x,y}$ and $g_{x,y}$ are the values of red and green channels at (x, y) of the image. Note that $\min(r_{x,y}, g_{x,y}) = 0$ implies that at least one of $r_{x,y}$ and $g_{x,y}$ must be 0.

However, we expect the output images to have generation noise. In order to ignore minute generation noise, we define the validity of an output image by the following formula:

Algorithm 6 Color edges of $contour\_graph$ using stratified dem_region

```
1: function COLOR_CONTOUR_GRAPH($contour\_graph, dem\_region$)
2:     Perform breadth-first search on $dem\_region$ starting from the strata where the height is 0, and obtain the shortest distance to each stratum
3:     for all $edge$ in $contour\_graph$ do
4:         $s1, s2$ := two strata adjacent to $edge$ where $s1.height < s2.height$
5:         if $s1.distance < s2.distance$ then        ▷ $s.distance$ is the distance to $s$ obtained in line 2
6:             $edge.color = GREEN$
7:         else
8:             $edge.color = RED$
9:         end if
10:    end for
11: end function
```

$$\forall y \forall x \min(r_{x,y}, g_{x,y}) \leq t$$

where t is the tolerance of the test, and we use $t = 16$. This test tolerates the error cases where both channels are positive, unless the error $\min(r_{x,y}, g_{x,y})$ is greater than tolerance t.

4.4 Score-based Evaluation

In the score-based evaluation, we assess the valid output of each model by calculating the fractal dimension of resultant height-maps. The rationale behind this is that natural terrain is expected to have fractal structures. We use the box-counting method described by Wen-Li Lee et al. [5] for the calculation. For this evaluation, we only take into account the height value of the terrain, and thus the water body is ignored, due to the limitation of the algorithm.

4.5 Environment

We have conducted our experiments using a computer with the following equipment:

- CPU: Intel Core i7 6700K
- RAM: DDR4 32GB
- GPU: NVIDIA GeForce RTX 4060 Ti 16GB

5 Results

5.1 Training Time

The training time for pix2pix was roughly 93.33 hours, while the training time for Palette was roughly 480 hours.

5.2 Generation Time

The generation time is the required time to generate one height-map from a contour-map. The average generation time of pix2pix was 0.005182 seconds, while the average generation time of Palette was 46.33 seconds.

5.3 Generated Height-maps

Eight samples of generated height-maps by each model are shown in Figure 3.

5.4 Validity Evaluation

The results of the validity evaluation are shown in Table 1. Each column is showing the result for the corresponding generative model. The error rate is the number of invalid output divided by the number of all output.

	pix2pix	Palette
Number of Valid Output	901	400
Number of Invalid Output	227	728
Error Rate [%]	20.12	64.54

Table 1: Results of Validity Evaluation

5.5 Score-based Evaluation

The results of the score-based evaluation are shown in Table 2. In addition to the scores for each model, we have calculated the score of the dataset created in Section 3.2 for reference. The score distributions for pix2pix, Palette and the dataset are shown in Figure 4, 5 and 6.

6 Discussion

As for the training time, it has turned out that Palette requires much longer training time than pix2pix. The same goes for generation time. This implies that pix2pix is more affordable to train, compared to Palette.

From Figure 3, we can observe Palette produce some random noise in the output, while the output of pix2pix is more saturated. In some cases, the output is so noisy that it is considered invalid. The problem with pix2pix is that the output tends to have unnatural, repetitive patterns. This phenomenon is especially noticeable in watered areas.

	pix2pix	Palette	Dataset
Average of Fractal Dimension	2.004	1.643	2.096
Standard Deviation of Fractal Dimension	0.2941	0.3462	0.3230
Minimum of Fractal Dimension	1.083	0.9999	1.398
Maximum of Fractal Dimension	2.610	2.510	2.844

Table 2: Results of Score-based Evaluation

The possible cause of these problems is the lack of training epochs, or the insufficiency of the training data. Longer training time may yield better results.

The validity evaluation has shown that pix2pix is less likely to produce semantically invalid height-maps than Palette. In contrast, Palette has frequently produced invalid height-maps. This seems to be due to the noise we mentioned earlier.

The score-based evaluation has revealed that the average score of pix2pix is close to that of the dataset, while the score of Palette has resulted in much lower score. This implies that pix2pix is capable of generating more natural terrain than Palette. However, it should be noted that this score could have been affected by the repetitive patterns.

7 Conclusion

We have attempted to create terrain generation models and investigated the applicability of diffusion models to the procedural content generation. Our experiment results have shown that pix2pix takes less time to train, and it is capable of generating semantically valid and natural terrain, though it occasionally produces visually unnatural patterns. In contrast, training Palette requires more than five times as long as pix2pix, and Palette frequently produced invalid height-maps under our environment.

From these results, we can conclude that pix2pix is a reasonable option for performing terrain generation, since it can be trained from scratch with less computation resources than Palette and has the capacity to replicate the features of the original dataset.

Acknowledgments

I would like to thank my supervisor Maxim Mozgovoy for reviewing my research project.

References

[1] P. Bourke, "Conrec - a contouring subroutine," 1987.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018.

[5] W.-L. Lee and K.-S. Hsieh, "A robust algorithm for the fractal dimension of images and its applications to the classification of natural images and ultrasonic liver images," Signal Processing, vol. 90, no. 6, pp. 1894–1904, 2010.

[6] NASA JPL, "NASADEM Merged DEM Global 1 arc second V001," 2020, [Data set].

[7] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[8] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," 2022.

[9] T. Wang and S. Kurabayashi, "Sketch2map: A game map design support system allowing quick hand sketch prototyping," 2020 IEEE Conference on Games (CoG), pp. 596–599, 2020.

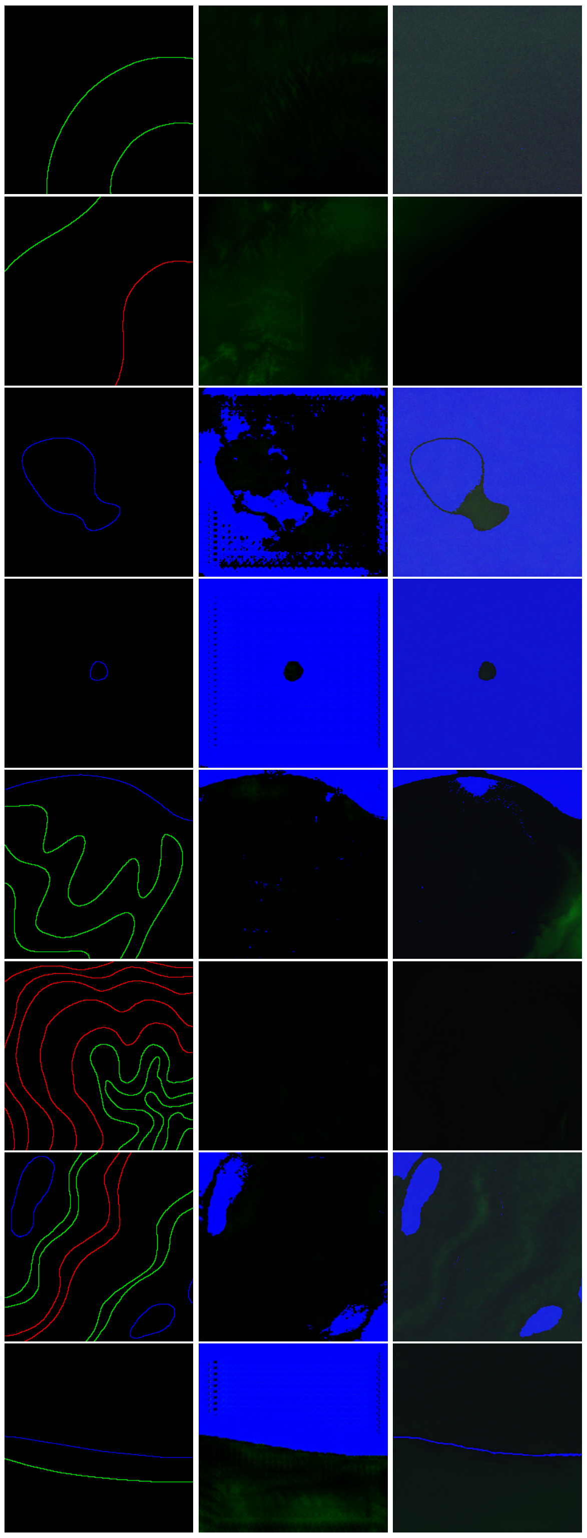
Figure 3: Generated height-maps (left: input contour-map, middle: output of pix2pix, right: output of Palette)

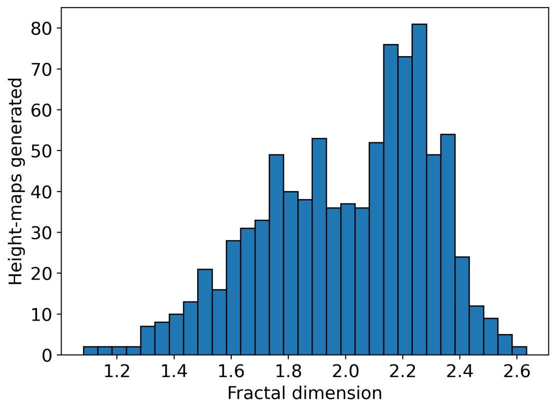

Figure 4: Score distribution of pix2pix

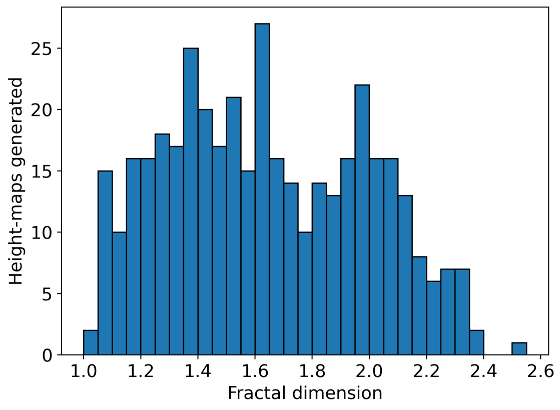

Figure 5: Score distribution of Palette

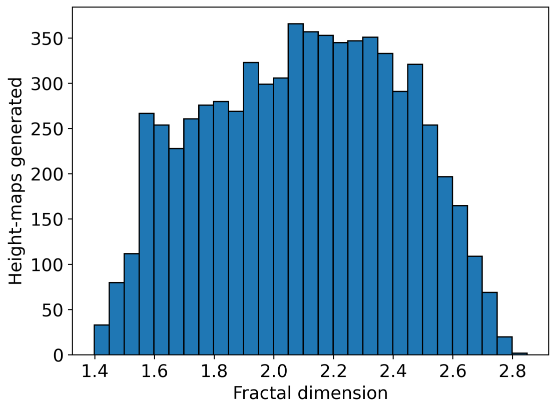

Figure 6: Score distribution of the Dataset