ANKARA YILDIRIM BEYAZIT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES



LAYER-BASED ARCHITECTURE FOR SYNTHETIC DIGITAL SURFACE MODEL GENERATION

M.Sc. Thesis by

Yasemin KILIÇ

Department of Computer Engineering

June, 2023

ANKARA

LAYER-BASED ARCHITECTURE FOR SYNTHETIC DIGITAL SURFACE MODEL GENERATION

A Thesis Submitted to the

Graduate School of Natural And Applied Sciences of

Ankara Yildirim Beyazit University

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering, Department of Computer Engineering

by

Yasemin KILIÇ

June, 2023

ANKARA

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**LAYER-BASED ARCHITECTURE FOR SYNTHETIC DIGITAL SURFACE MODEL GENERATION**" completed by **YASEMİN KILIÇ** under supervision of **ASSOC. PROF. DR. FATİH NAR** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Fatih NAR

Supervisor

Assist. Prof. Dr. Murat SARAN

Jury Member

Assoc. Prof. Dr. Hilal ARSLAN

Jury Member

Prof. Dr. Sadettin ORHAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: 2023, 06 June **Signature:** _____

Name & Surname: Yasemin KILIÇ

ACKNOWLEDGEMENTS

I would like to thank my advisor, Assoc. Prof. Dr. Fatih NAR of the Faculty of Engineering and Natural Sciences at Ankara Yildirim Beyazit University for his continuous support and motivation during the study of the Author. His immense knowledge and valuable recommendations created the milestones of this study. His guidance helped the author throughout the research and writing this thesis.

I thank my colleagues at the AYBU family, who made me love the department, lessons, research, and learning until I came to this process.

June, 2023 **Yasemin KILIÇ**

LAYER-BASED ARCHITECTURE FOR SYNTHETIC DIGITAL SURFACE MODEL GENERATION

ABSTRACT

In this thesis, a novel layer-based method for generating the 2.5-dimensional digital surface model (DSM) is proposed. The suggested methodology employs Deep Learning and Convolutional Neural Networks (CNN) while utilizing a loss function specifically developed for the employed neural network architecture. A real-world digital terrain model (DTM), building footprint layer, and 3D tree models are used to create synthetic training data. The proposed neural network model is trained using this synthetic data and random input noise so it can learn to generate a DSM for giving DTM and random sketches for building and tree layers. Thus, the proposed model can generate a realistic DSM based on real-world and random user inputs. Such DSM models are useful for training other deep-learning models and creating rich game or simulation environments.

Keywords: Digital Terrain Model, Digital Surface Model, Natural Elements, Man-Made Objects, Synthetic Data, Deep Learning

Sentetik Dijital Yüzey Modeli Üretimi İçin Katman Tabanlı Mimari

ÖZ

Bu tezde, 2.5 boyutlu dijital yüzey modelinin (DSM) üretilmesi için katman tabanlı yeni bir yöntem önerilmiştir. Önerilen metodoloji, kullanılan sinir ağı mimarisi için özel olarak geliştirilmiş bir kayıp fonksiyonunu kullanırken Derin Öğrenme ve Evrişimli Sinir Ağlarını (CNN) kullanır. Sentetik eğitim verileri oluşturmak için gerçek dünyadaki bir dijital arazi modeli (DTM), bina ayak izi katmanı ve üç boyutlu ağaç modelleri kullanılır. Önerilen sinir ağı modeli, bu sentetik veriler ve rasgele giriş gürültüsü kullanılarak eğitilmiştir, böylece bina ve ağaç katmanları için DTM ve rasgele eskizler vermek için bir DSM oluşturmayı öğrenebilir. Böylece önerilen model, gerçek dünya ve rasgele kullanıcı girdilerine dayalı gerçekçi bir DSM üretebilir. Bu tür DSM modelleri, diğer derin öğrenme modellerini eğitmek ve zengin oyun veya simülasyon ortamları oluşturmak için kullanışlıdır.

Anahtar kelimeler: Sayısal Arazi Modeli, Sayısal Yüzey Modeli, Doğal Öğeler, İnsan Yapımı Nesneler, Sentetik Veri, Derin Öğrenme

CONTENTS

M.Sc. THESIS EXAMINATION RESULT FORM ii
ETHICAL DECLARATION iii
ACKNOWLEDGEMENTS iv
ABSTRACT v
ÖZ vi
NOMENCLATURE ix
LIST OF FIGURES xi

CHAPTER 1 – INTRODUCTION 1

1.1 Data Types 1
1.1.1 Digital Elevation Model 2
1.1.2 Digital Surface Model 3
1.1.3 Digital Terrain Model 4
1.1.4 Standardized Digital Surface Model 5
1.2 Contents Of DSM 6
1.3 Data Generation 6
1.3.1 Terrain Generation 8
1.3.1.1 Synthetic Terrain Generation 9
1.3.1.2 Terrain Generated DSM 12
1.3.2 Man-Made Data Generation 17
1.3.2.1 Road Networks 17
1.3.2.2 Buildings 23
1.3.3 Natural Objects Generation 29
1.3.3.1 Vegetations 29
1.3.3.2 Water Bodies 36
1.3.4 Virtual Worlds 40
1.3.4.1 Urban Settlement 44
1.4 Convolutional Neural Network 51
1.5 Our Contribution 57

CHAPTER 2 – PROPOSED METHOD 58

2.1 Model Architecture 58

2.2 Dataset 59

2.3 Model Training 63

CHAPTER 3 – RESULTS 64

CHAPTER 4 – CONCLUSIONS 69

REFERENCES 70

CURRICULUM VITAE 91

NOMENCLATURE

CAD	Computer Aided Design
CGA	Computer Generated Architecture
CHM	Canopy Height Model
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DBN	Deep Belief Network
DEM	Digital Elevation Model
DSM	Digital Surface Model
DTM	Digital Terrain Model
ESRI	The Environmental Systems Research Institute
GAN	Generative Adversarial Network
GPS	Global Positioning System
GPU	Graphics Processing Unit
InSAR	Artificial Aperture Radar Interferometry
LIDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory Network
MSE	Mean Square Error
nDSM	Normalized Digital Surface Model
NPI	Normalized Pressure Integral
PC	Red Green Blue
PCG	Procedural Content Generation
PM	Procedural Method
RGB	Red Green Blue
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SAR	Synthetic Aperture Radar
SPH	Smoothed Particle Hydrodynamics
VHR	Very High Resolution

LIST OF FIGURES

Figure 1.1 Differences in Scope between DEMs, DSMs, and DTMs 2
Figure 1.2 3D DEMs of the combined marine and land surface data 2
Figure 1.3 A Standard 2m DSM 3
Figure 1.4 A Standard DTM 4
Figure 1.5 A Standard nDSM 5
Figure 1.6 Terrain Generation in landscapes [1] 7
Figure 1.7 Terrain Generation with The Fractal Subdivision [2] 10
Figure 1.8 DTM Generation from DSM [3] 13
Figure 1.9 OpenStreetMap and Generated Road [4] 18
Figure 1.10 An innovative shape grammar called CGA shape is employed to algorithmically create 3D models and environments in computer graphics [5] .23
Figure 1.11 Making lively three-dimensional representations out of excellent two-dimensional geographical data 26
Figure 1.12 Simple Vegetation Types 30
Figure 1.13 A Vegetation Generation in Ecosystem [6] 34
Figure 1.14 Original image from Google Earth and generated river flows over [7] ..37
Figure 1.15 A Virtual City 41
Figure 1.16 An Urban View 45
Figure 1.17 An Urban Generation 46
Figure 1.18 CNN: A Basic Structure.....52
Figure 1.19 Cross Correlation 2 Channels 53
Figure 1.20 Correlation with Padding&Stride 54
Figure 1.21 Pooling Layer.....55
Figure 1.22 Alexnet 56
Figure 1.23 GoogleNet.....56
Figure 2.1 Initial CNN Model.....59
Figure 2.2 Berlin DTM Data.....59
Figure 2.3 Berlin Buildings 60
Figure 2.4 Berlin DSM Data.....61
Figure 2.5 Low-Poly Tree Models 61

Figure 2.6 Figures represent respectively DTM, DSM, Building, and Incoming Building Data. 62
Figure 3.1 Prediction Result Of Initial CNN Model 64
Figure 3.2 Loss Curve Of Initial CNN Model with Adam Optimization 65
Figure 3.3 Prediction Result Of CNN Model with Adagrad Optimization 65
Figure 3.4 Loss Curve Of CNN Model with Adagrad Optimization 66
Figure 3.5 Layers Of CNN Model having Multi Sequential Layers with Adagrad Optimization 66
Figure 3.6 Loss Curve Of CNN Model having Multi Sequential Layers with Adagrad Optimization 66
Figure 3.7 Loss Curve Of CNN Model with Adam Optimization using k-fold cross-validation with RMSE metric 67

CHAPTER 1

INTRODUCTION

Considering the research and developments that have occurred over thirty years, we obtain terrain data from sufficient data sources, with the opportunities provided by these technological developments. This resource can sometimes be real geological data, or it can also be a tool that allows us to render terrain and even edit the existing data.

Synthetic environments are an isolated application area for developing artificial intelligence techniques. They are likely to happen in the future but are not possible to test in today's conditions or unpredictable results. This field attracts our attention, especially in application areas such as flight or battlefield simulators and special effects in feature films, the game, and the simulation industry. It has also become an effective tool for creating and editing synthetic landscapes. In areas where artificial intelligence is preferred, such as deep learning, these methods hold geological data in pairs with Digital Surface Models by fusing the Digital Surface and the Digital Terrain models in fields where artificial intelligence is preferred. They have been used to generate elevation data, RGB data, and other types of synthetic data. To this end, many synthetic methods of information production have been suggested, as described in more detail below.

In the first chapter, the structure of the models holding the geological data (Section 1.1), the content of the DSM (Section 1.2), and a literature search of the production and modeling of the DSM involving the terrain, man-made or natural objects (Section 1.3) will be explained. General structure of the CNN model used in this thesis will be explained in the Section 1.4. Finally, the Section 1.5 concludes the literature research by mentioning the contribution of the thesis.

1.1 Data Types

In the literature, wide variety of elevation models are proposed for representing the geospatial data such as Digital Terrain Model, Digital Elevation Model, and Digital Surface Model.

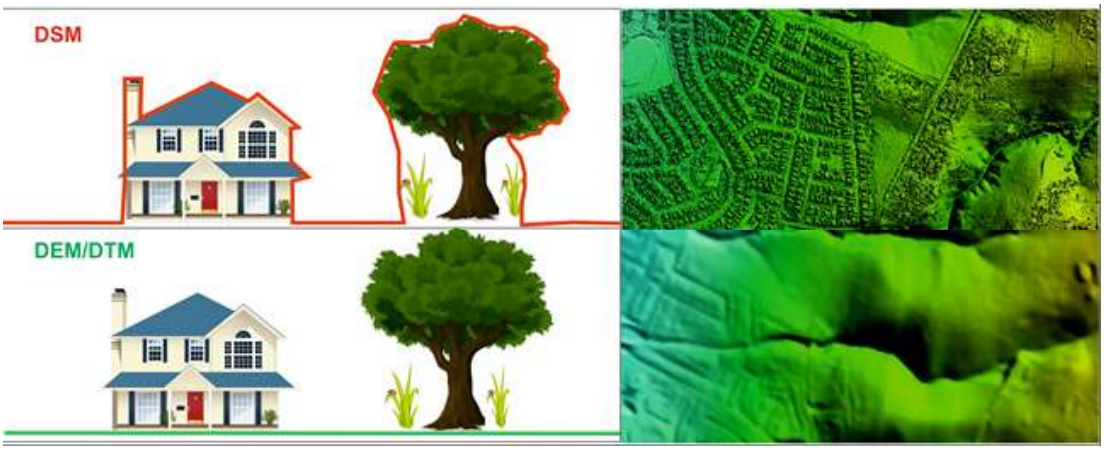

Figure 1.1 Differences in Scope between DEMs, DSMs, and DTMs

1.1.1 Digital Elevation Model

One of the preferred models for modeling the Earth is the one that contains elevation information of terrain features. These models are based on the position and altitude of the planet and the natural elements (vegetation, etc.) as well as the artificial elements (buildings, towers, power lines, etc.) deposited in them. Remote detection methods such as InSAR, Stereo Photogram Metrics, Contour, Satellite Interaction, and LiDAR may be used to collect data for the DEM. Satellites, planes, and drones are just some of them. It comprises raster grids representing the bare world referring to a vertical reference point. It is often favored for soil mapping, hydrologic modeling, and stabilization of land degradation. Enhancements can be implemented manually to create DEM data and standard tools, such as MapInfoPro.

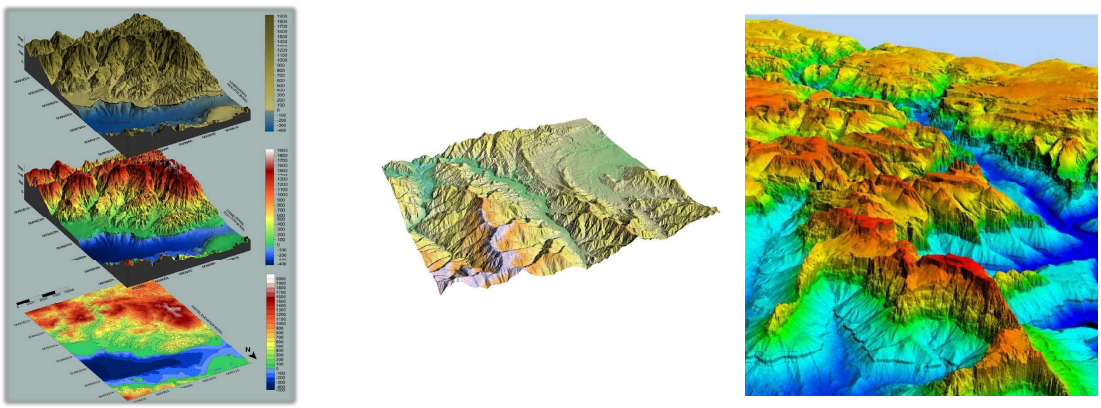

Figure 1.2 3D DEMs of the combined marine and land surface data

This digital elevation architecture has two sub-models, such as defined surface and terrain information.

1.1.2 Digital Surface Model

The surface representation provides a detailed topographical model that depicts the different heights of the Earth's terrain in three dimensions, encompassing a wide range of organic and man-made features. It represents the surface of the world by including all visible objects such as vegetation, roads, and buildings on its numerical structure, which includes a numerical terrain model. DSMs represent a valuable geographic asset encompassing precise spatial data, including positional coordinates and elevation details, capturing a comprehensive range of terrestrial elements like buildings, road networks, waterways, spatial land use, and governmental demarcations.



Figure 1.3 A Standard 2m DSM

When creating a DSM, there are also different approaches for treating stereo pairs based on automatic, semi-automatic, and interactive vectoring modes. They can be created from photogrammetry, LiDAR, SAR interferometry, a geographical vector layer, and satellite or aerial research materials. Synthetic data is required to produce flexible models with desired properties based on these expensive resources. It provides auxiliary tools developed to reduce costs.

The digital surface model is applicable in a multitude of industries, encompassing various fields such as geospatial navigational technologies, intelligent transportation systems, urban hazard assessment, cellular network infrastructure in telecommunications, flood risk assessment, execution of runway approach areas, vegetation management, and field of vision mining.

1.1.3 Digital Terrain Model

It is extremely difficult to provide an adequate explanation for Earth's digital topography model. In general, the bare Earth can be described as a topographical elevation model that can be adapted for different purposes. On of the common digital topography model for bare earth is DTM. Combined with DSM, it is used in various applications for detection of vegetation as well as man-made elements, such as structures and automobiles. The model only maintains elevation data associated with a rectangular grid for the underlying terrain.

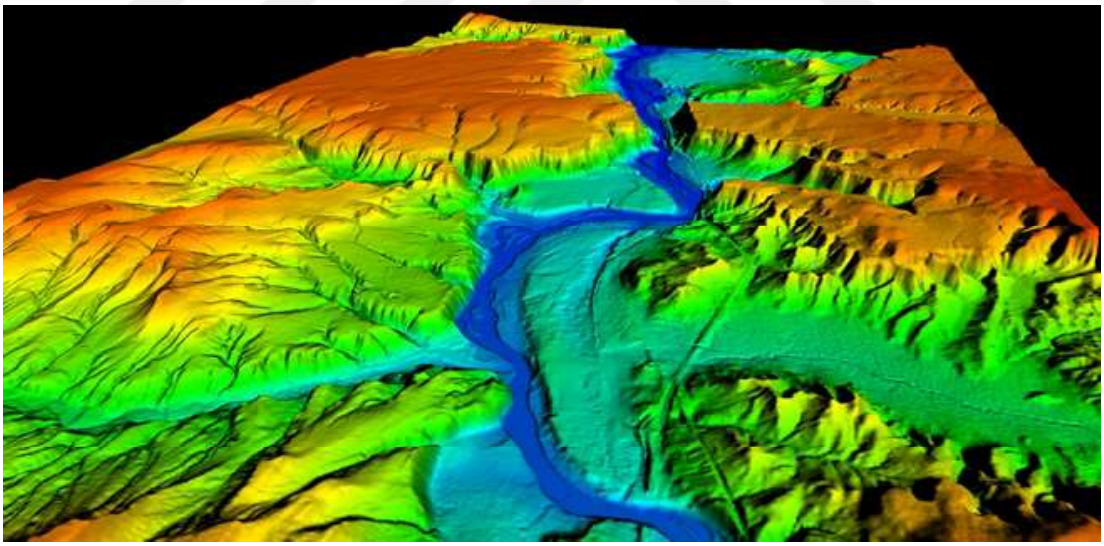

Figure 1.4 A Standard DTM

The possible usage area of DTMs can be quite diverse. These application areas include the extraction of terrestrial parameters, vegetation and artificial objects, road planning, and three-dimensional visualization and simulation. It also encompasses massive water movements, precision agriculture and forestry, geodesy and surveying, geophysics, geography, and remote sensing.

There are many approaches in the literature to generate DTM, which is very costly to produce by hand [8]. Along with automatic approaches based on linear estimation techniques with morphological filtering [9], there are also nonlinear approaches using a variational approach [10, 11]. The removal of cross-linked components, the two-dimensional empirical mode decomposition, and the derived from a DSM by interpolation are other methods used [8].

1.1.4 Standardized Digital Surface Model

Normalized surface elevation representation is a by-product of height obtained by subtracting the DSM from the DTM data, namely normalized DSM (nDSM). This model accepts bare earth have a value of 0. It depicts objects such as trees or buildings by their respective absolute heights relative to the local terrain. Although there is a risk of error in such nDSM generation height data sets, it should be noted that this risk can be higher for DSM data derived from LiDAR.

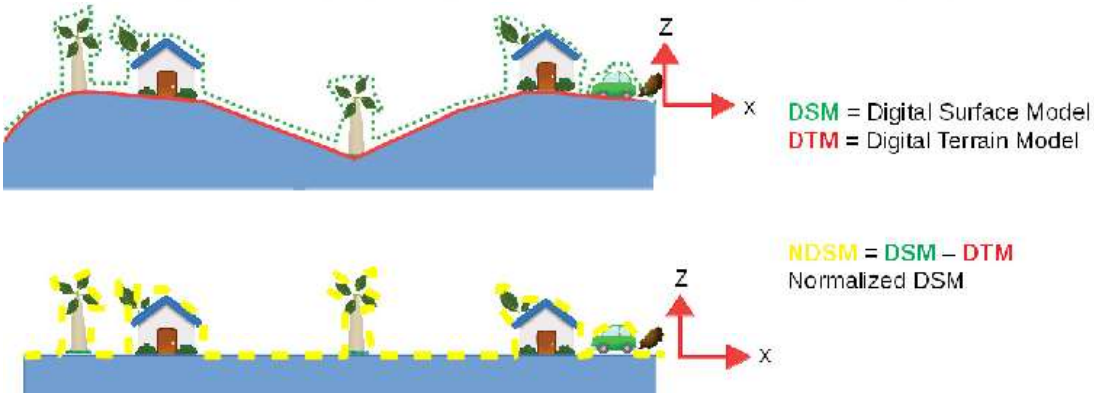

Figure 1.5 A Standard nDSM

Normalized digital surface models are preferred in studies in building height estimation, urban structure mapping, tree height estimation, above-ground biomass estimation, and forestry. The canopy height model (CHM), which is an nDSM representing only vegetation, can be said to be another form of use.

1.2 Contents Of DSM

DSM can be produced from many alternative acquisition approaches such as LiDAR and Very High Resolution (VHR) optical satellite images. Real-Time Kinematic GPS Topographic maps are also preferred for various land structures such as flat and bare ground, mountainous areas, hardwood forests, glaciers, green areas, and urban agglomerations. Yet, there may be simple differences depending on the source derived. For instance, the DSM derived from the LiDAR is created from the data of the initial pivot point. In contrast, the height values representing the highest height per pixel are derived from photogrammetry. Despite these simple differences, it can be said that the essentially produced model may have three major components in order to produce the expected consistency. They are terrain, natural objects (water bodies, trees, bush, etc.), and man-made objects (roads, buildings, poles, bridges, etc.).

1.3 Data Generation

Terrains have become an essential element that holds significant relevance in various computer-based applications such as games, visual effects, training, and simulation. Since such environments, which have a wide range of uses, are quite complex and detailed, modeling is a very challenging process. Facilitating this laborious process has been the focus of researchers' interest. Because of this search, one of the techniques that have found a place in our lives has been procedural methods.

Procedural techniques emerged as modeling and texture creation techniques, including the procedural definitions of geometry and surface color in the emergence of computer graphics. Since then, they have become actively utilized research themes as a solid paradigm of modeling, texture creation, and animation. It transcends beyond being a mere subject of research in computer science. Each procedural method had a different usage area. L-system grammars are mainly employed for producing plants. Agent-based particle systems are applied for modeling entities like fire and smoke which don't have clear borders. Perlin noise was used to simulate cloud formations and organic models. However, methods based on autonomous agents preferred to animate the collective behavior of groups of animals or objects [12].

This concept's significance lies in its relevance to a wide range of disciplines that are nature-inspired processes such as textures [13], plants [14], lands [15], buildings [5, 16], rivers [17–20], etc., human-centered processes such as urban areas and road networks [13, 21, 22], and even physics-based simulations, continue to increase in parallel with its development. It takes a great deal of effort to discover the discipline that aligns it with all these fields and to find the right formalism and structures [23].

This increase is an indisputable effect of robust and fast usage methods developed with increased CPU power and affordable computer. Although there is no single definition because of the variety of usage areas, it is possible to take the definition in the book into consideration [24]. Shaker et al. [25] explained what PCG is and what it is not, giving examples.

PMs have many powerful features that can be used. Abstraction is an integral part embedded in these architectures. This abstraction part allows the computer to create models with the required resolution by shifting the programming time requirements from the programmer to the computer. Due to its flexible structural design, it is suitable for creating training at low costs that are not bound by the laws of physics.

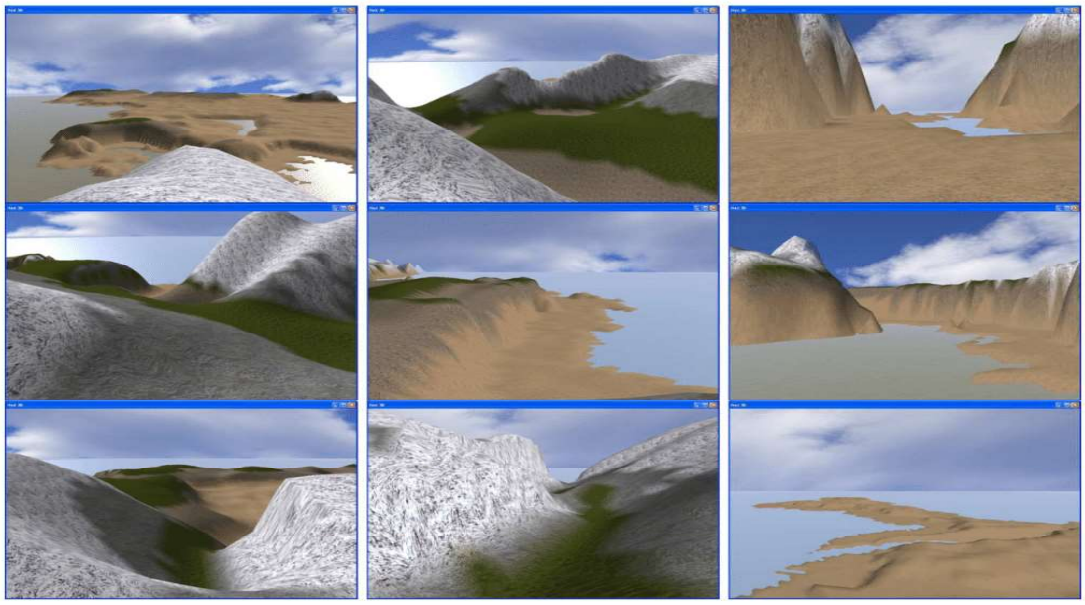

Figure 1.6 Terrain Generation in landscapes [1]

Parametric control is another powerful PM feature, known as database amplification by Smith (1984). It provides high levels of control to the modeler/animator. Despite their flexible and endless variety of productive nature, it may make them difficult to alter once constructed. They may need to manipulate complex parameters whose effects on output are hardly predictable. So, they may not be suitable for manual modeling because of such control deficiencies.

From a research perspective, Fournier et al. [26]'s 1982 frontier study , Digital Representation of Stochastic Models, was one of the first realistic approaches to object modeling publications. It explicitly discusses stochastic terrain model generation. Darwin R. Peachey [27]'s publication explicitly discusses procedural content generation. That gives us a sample of solid texture functions based on Fourier synthesis. Peachey presented us with a technique that provided a possibility for 2D textures to look three-dimensional.

In short, these methods lack spatial information based on topographic features, yet, they have been preferred frequently to establish the details of the terrain [28].

1.3.1 Terrain Generation

Many literature publications on the market can produce impressive landscapes in terrain production. Given these studies with pluses and minuses, existing field modeling techniques could be classified as procedural methods [1,20,23,25,29–36], sketch-based methods [37–40], sample-based methods [35,41] and physics-based methods [42]. The scope of use of algorithms is determined based on user needs, considering criteria such as controllability problems and tolerance to adapt to large-scale terrains. So it may be necessary to know the differences that could be critical in this selection.

One of the most basic methods of modeling a terrain is heightmap. This map expresses two-dimensional grids of elevation values. It can be generated by many procedural algorithms, such as subdivision-based methods. In the literature, many technical articles have presented new techniques for surface fabrication due to the inherent lack of height maps that model rock overhangs [43], caves [43] or waves [44].

Terrain generation algorithms before height maps have a long history. With the improvements made, the noise functions have been made capable of capturing the variable multiscale nature of the terrains in real-time, including its hybrid and salient multifractals [40]. They were first appearing in the late 1980s ([26], [45]) based on subdivision methods in which a controlled random offset is added to a rough elevation map iteratively to create elevation details. The approximations of terrain patches using stochastic processes such as fractional Brownian motion [46] has been a remarkable achievement in this field. In cases where the division steps and the initial offset range cannot be predetermined, insufficient results have been obtained in determining mountain and valley-like features with the method which can cause the roughness of the land to increase.

A procedural terrain modeling technique has various advantages such as adaptive detail, built-in anti-aliasing technique, and high supporting content. Gamito and Musgrave [29] propose a system of Earth curvature comprising distorting the initial elevation field surface along a vector field. These benefits were used in the conventional terrain format and refracted waves following Fournier's work. The technique used in Peytavie et al. [43]'s paper is based on an efficient tiling algorithm suitable for real-time editing to simulate and stabilize hundreds of rock fragments separated from bedrock and piled together. They were combining an information structure comprising discrete volume data and a model utilizing indirect encoding.

1.3.1.1 Synthetic Terrain Generation

A texture generation was developed resembling nature with fractal-inspired textures and shapes of a movie by Ken Perlin in 1982. Perlin Noise [47] was then reconstructed with low persistence. After that, a set of values was derived through a probabilistic function which was interpolated to produce a more natural aspect. Terragen software leverages the Perlin Noise technique to create terrain, mountains, ocean waves, foliage, clouds, and rocky surfaces with intricate details. It provides high controllability with its capability that starts from the landscape and continues from rivers to the sun, moon, and stars. Some prefer to use this method, which relies on noise generators for height map generation.

Mustage [48] introduced a new methodology for generating height fields of fractal terrains. It also included fractal measurement controlled locally, along with other factual characteristics. So, it provided realism to eroded landscapes due to the fractal character's local interchange factor or dimension. The work done was mainly based on thermal wear and simple hydraulic erosion. Because of the constraints in available computing capabilities, it can require serious simplification to introduce real drainage and weathering models that require the simulation of thermal and hydraulic erosion [40]. This work enhanced the credibility of the mountainous terrain with various enhancements, including corrosion [49] and 3D hydraulic erosion [50], but again the amount of control was limited in the simulation as the erosion used was typical global processes. Yet, with some articles [51], the slowness that occurs with thousands of times used has been brought to reasonable conclusions leveraging Graphics Processing Unit (GPU) technology. In erosion implementation, the experiments [51] show that the GPU can achieve a speedup of six to ten times compared to the CPU.

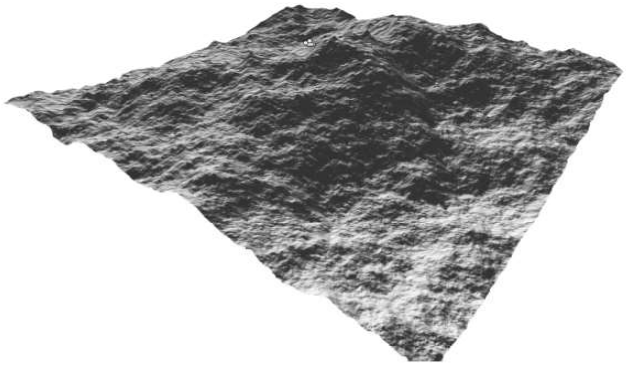

Figure 1.7 Terrain Generation with The Fractal Subdivision [2]

The fractal surfaces were mostly related to the concept of fractional Brownian motion. There exist numerous variations of this noise, commonly referred to as fBm. They are quite well-suited for rendering certain elevated land formations with raised crests or undulating slopes. Some of them also allow the production of three-dimensional formations created through procedural generation, such as clouds with realistic depth and density [34]. The performance of fBm noise algorithms is almost perfect. The degree of control over the generation of the height map is almost similar to the Midpoint displacement method by the restriction but they cannot provide local variation.

Surface can be fundamentally described Voronoi diagrams as the distance decomposition of the metric space with a separate set of objects. This method has previously become widely used in many scientific applications exist, including space analysis, urban planning, settlement analysis, geology, robotics, and ecology. It was only available without procedural approaches. The major target of the algorithm is textures found in nature, including materials like paper, tree bark, and dried mud. These materials generally have distinct patterns and irregularities that can be effectively recreated with very little input data [52].

Procedural methods are optimized methods for creating content quickly and efficiently that contribute to mitigating and dealing with this issue [52]. For instance, the subject of superb control of terrain morphology with randomness, cannot be supported by procedural methods. These methods are often effective methods for generating large-scale terrain. Likewise, by using procedural approaches that can create geometric models, textures, animations, and even sound clips, it is possible to generate artistic terrain with less time and effort and less cost. The sketch-based approach can produce terrain according to the morphology specified by the user drawing, but is inefficient in manual editing and requires user experience. Sample-based methods that create synthetic terrain with features taken from real data, on the other hand, give very difficult and inefficient results for large-scale fields that require very large data sets. Computer-generated terrains generated using physics-based techniques, can able to closely adapt the real-world terrain morphology. Yet, this methodology, which lacks precise control, is not compatible with wide-ranging terrain.

Because how the shape or structure of the terrain surface is influenced by many natural phenomena like rainfall erosion and fluctuations in temperature, it can be said that it is quite laborious to simulate the terrain formation process using physical models [28]. Physical simulations [48, 53] require the creation of the universe created at once and cause both processing and memory costs. Therefore, they are used in areas where these requirements are not so strict, such as movies and them, to manipulate this deficiency. The most well-known examples are erosion simulations that occur with the slow forming of the land resulting from forces such as hydrological and aeolian.

1.3.1.2 Terrain Generated DSM

Approaches for generating terrains typically incorporate a combination of techniques, such as fractal terrain generation, terrain synthesis based on terrain features, the utilization of sample terrain patches [54], or noise algorithms.

Due to the irregular and fragmented nature of natural shapes, we can use it with a branch of mathematics that we call fractal. This fractal is established in an iterative structure that contains a great deal of self-similarity. And it can use to exhibit complex regular geometric expressions. Fractals possess a recursive structure that allows them to be applied iteratively on progressively smaller scales. These structures result in the generation of highly intricate surfaces that faithfully abstract the natural objects they represent [36]. Yet, due to its repetitive, tedious, and limited visualization, it must be automated with computer-based applications.

Procedural modeling approaches came to the forefront with the provided by Ebert [24]'s detailed source and Fournier et al. [26]'s method of adaptive subdivision. They lack spatial information, but, accurately describe the morphological features of the terrain. Thus, approaches based on fractals have been widely employed to create intricate features of large-scale terrain.

Automating it by reducing the user's manipulation of the parametric structure also means reducing user controls. It can be said that the nature of procedural methods is a complete paradox with its pros and cons. Thus, we can say that the lack of user control in procedural methods was one matter needs to be resolved.

The fact that these currently suggested approaches aim to minimize the need for user interaction during the production process [32]. Therefore, some studies have explored explicitly the midpoint displacement technique [26] using the change-based method in terrain generation. Amburn et al. [55]'s subdivision method manages the behavior of the procedures to external events as well as to normal termination conditions. They provided a head start against this trend by providing clients with the ability for specifying restrictions, although it proved insufficient.

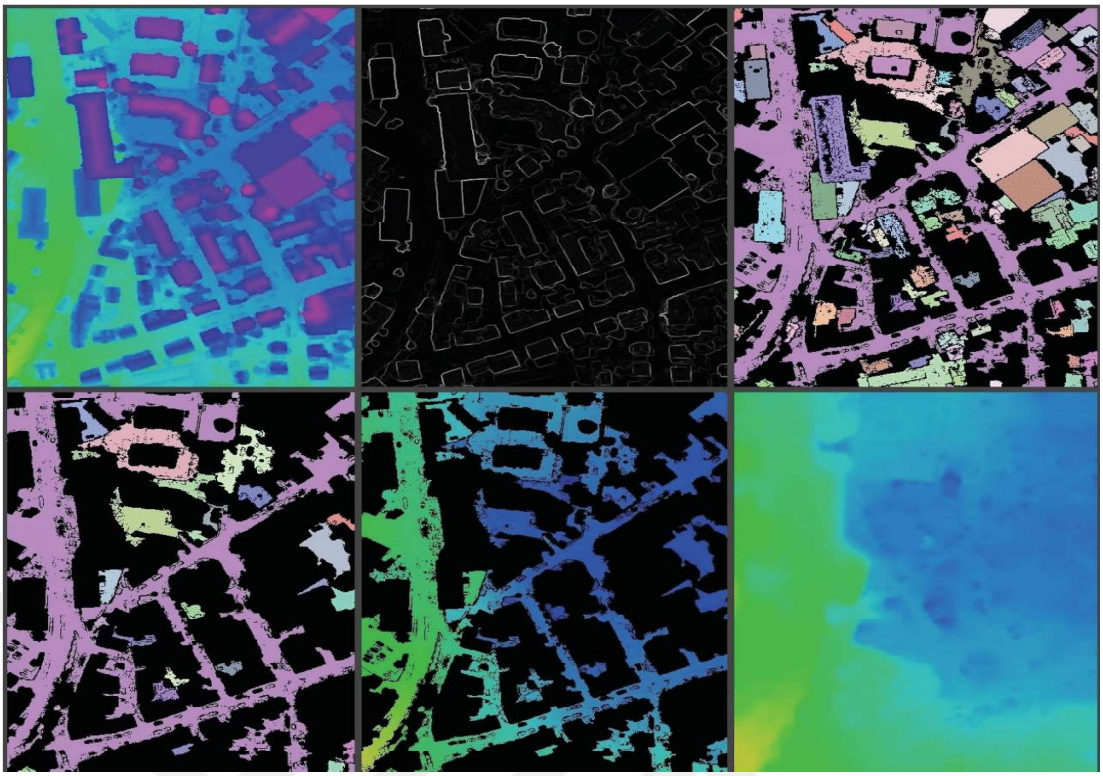

Figure 1.8 DTM Generation from DSM [3]

Rusnell et al. [56] presented a new, easy-to-control, terrain synthesis method that gives consumers direct control over the arrangement and placement of a diverse range of features, ensuring a robust generation of landscapes. This method, which relies on the principles of Dijkstra's algorithm and the blending of layers created according to individual characteristics, produced faster and higher quality results than Zhou et al. [57]'s method, although it had some disadvantages. Zhou et al.'s system was built upon an intuitive interface that utilizes sketch-based interactions that produced the ultimate terrain produced by synthesizing example terrains for specifying desired terrain characteristics. But, It failed to incorporate novel fine-grained details not found in the input fabric of the method, having a quality of the final synthesis that was dependent on the wealth of available field data.

Although sketch-based interfaces, which appear in the literature to improve both usability and control, are generally applied to geometric modeling [38, 39, 58], they have also been yielded impressive results in the algorithmic generation of elements including flowers [59], trees [60] and to a limited extent terrain [37, 61]. In general,

interactive sketch-based approaches empower users to draw intricate silhouettes, generating a land-form that corresponds to their sketches [62]. Terrain sketching was introduced by Gain et al. [40], inspired by the effectiveness of portraying interfacing and the possibilities of scene-outlining strategies in various fields of procedural modeling. They separated translating a feature into a visual representation through the creation of its silhouette and shading, unlike previous approaches.

Height map, also called height field, is among the techniques that are very frequently employed to represent terrains precisely in a procedural terrain generation [31, 63]. It is a raster image expressed as two-dimensional grids of elevation values, where each pixel stores values to show in three-dimensional computer illustrations. An elevation map is pretty easy and GPU-friendly to use, supporting more than the basic orthogonal elevation. A height map can be manually crafted using a traditional painting program with interactive user input or generated using a dedicated terrain editor capable of rendering the terrain in immersive 3D, as well as using terrain generation algorithms such as a 2D simplex noise function [64], and diffusion-limited aggregation [54, 65]. Height maps can be applied in relief maps to determine the precise positioning of 3D data that will cast a shadow, in displacement maps to alter the physical position of individual points [34].

Saunders [66] created a method using machine learning techniques to synthesize terrain height fields, built upon real-world digital elevation models of real-world terrain with a CAD-style interface he created. In this thesis, the user was expected to design delineating the land layout by sketching basic regions and specifying the desired land features of each region. This designed area was then combined with the elevation data fragments from the sample height areas using the genetic algorithm. This technique offers valuable benefits to generating several reasonably realistic outcomes with minimal user effort and expertise, providing virtually limitless possibilities while requiring relatively little user effort and expertise while allowing the algorithm to come up with new feature arrangements with the guided randomization inherent in the genetic algorithm. But unfortunately, it still depends on user-set restrictions.

A mountain generation technique that is taken mountains elevation and base propagation as a constraint is presented by Kamal and Uddin [67]. In this paper, a new constrained mid-point displacement approach to parametrically controlled artificial terrain generation is proposed, whose characteristics such as height and base span can form a single mountain at a predetermined height. Belhadj's [68] work has a more versatile approach, incorporating a predefined range of values to constrain the midpoint displacement process.

There are also some articles [1] to generate terrains that made up an intuitively controllable agent-based terrain generation system, creating specific landforms at areas, regions characterized by undulating hills and mountainous terrains adorned with valleys and mountain passes. This method, which allows frequency control on certain landforms, did not provide direct user control where they occur; and the results were not interactively obtained.

After the evolution of GPUs, height map generation with interactive user control has become possible. Gain et al. [40] presented an approach to generating a height map of a mountain in a 3D environment using sketches of its silhouette and boundaries, creating a matching mountain using noise propagation. The painting gray-scale images, developed by Schneider et al. [64], provided interactive control to edit the terrain. Bernhardt et al. [69] has presented a sketch-based modeling algorithm for terrain edition that can create and display complex and high-resolution terrains that work efficiently and quickly so that the user can see the terrain transition while editing the drawing. They achieved fast enough impressive computational efficiency in data representation and rendering through a synergistic CPU and GPU integration. Carpentier and Bidarra [30] have complemented their common synthesis algorithms, providing a flexible level of control that includes methods that enable users to brush through typical terrain features that are better intricate as well as realistic.

Bruneton and Neyret [13] have contributed to the improvement of digital elevation models [70] with their continuous data using adaptive refinement, procedural vector features to represent linear and spatial features, such as highways, rivers, railways, and lakes, as in GIS applications.

Although many of the methods explained above implement grid-based systems, the reality is that there has been a need to develop another approach (e.g. Layered data structures like patch systems, voxel data, or 3D networks) that can describe a real-world surface rather than methods for constructing a surface from existing mathematical formulas.

Kristof et al. [71]'s paper presented a new technique, also called Lagrangian, that efficiently combines fluid simulation using a smoothed interpolation approach, known as SPH, and a model for geological weathering forms grounded in real-world physics, which we are also familiar with from an Euler approach. Their results showed that this method based on particle simulation is effective for the erosion of concentrated, sizable, and delicate fluid, even for scenes containing up to 25000 particles. A groundbreaking algorithmic method for autonomously deforming fractal terrain data via the application of randomized local exploration with a certain set of constraints expressed as mask images for fractal terrain deformation is presented by Stachniak and Stuerzlinger [72]. However, it is an inescapable fact that it is a computationally expensive approach.

A hybrid approach has been provided by Douillard et al. [73] that captures the ground surface and obstacles, taking into account the minimum, average, and maximum elevations of all measurements within a single grid cell. While this map is unique in performing the jointly ground extraction, overhang representation, and 3D segmentation, it will have the inherent disadvantage of parameterization and the prohibitive memory cost of dense production data.

A diffusion-guided technique for creating terrain by leveraging a collection of parametrized curves was presented by Hnaidi et al. [54]. It featured land-forms including cliffs, riverbeds, or ridge-lines that provide natural based on vector analysis, and control based on feature analysis over landscape to produce procedurally realistic graphs. Hnaidi et al. [54], who modeled land features with the feature curve concept, constituted a remarkable resource in the literature. However, Becher et al. [62] also defined structures in three-dimensional space that are impossible to identify with an elevation field, unlike Hnaidi et al. [54]'s height maps.

Despite the encouraging results, the procedural methods, which are promising compared to manual content creation, unfortunately also have widely known disadvantages such as randomness and limited influence on the outcomes, and the lack of comprehensive integrated solutions.

Another alternative to height maps is to instead use an extension of it, a set of linked height maps called the patch system. The system's main idea is to use as many height maps as needed, which have two vertically overlapping points that belong to two separate height maps, where the connection between the pixels of the different height maps is supported. An elegant extension of the height map, this model can be used with no unnecessary overhead or complexity, according to Baudrillard [74]. Also, Zou et al. [57] presented a sample-based system that creates patches from sample a terrain characterized by prominent sinuous characteristics such as ridges and valleys, that dominate the visual landscape. Although they achieved better results inspired by the visual diversity exhibited in real-world terrain data, supporting a wide variety of user-controlled terrain synthesis styles, this method was limited by the input set provided as in the sample-based method [34].

Another usage alternative is voxels, comprising a series of volumetric pixels, which we can encounter in many applications from medical images to computer graphics. Since the voxel array is a mosaic of space, it eliminates the need for voxel coordinates, without the need for any amplification per voxel that each array defines the surface geometry with the presence/absence of voxels. However, it must be recognized that there will be cost, whether calculating poly line parameterization, constructing a patch system, or storing a voxel array [74].

1.3.2 Man-Made Data Generation

1.3.2.1 Road Networks

Terrain features must be in perfect harmony with their neighborhood to create a realistic virtual world. In case of something else, system integration may need to be done manually, with the entire load being the responsibility of the designer. This will hurt their efficiency and restrain their capacity to explore [32].

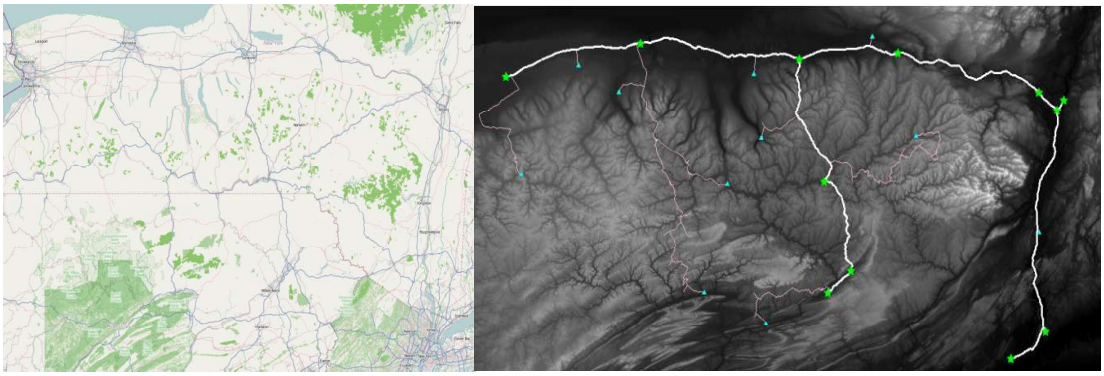

Figure 1.9 OpenStreetMap and Generated Road [4]

Developing an integrated modeling approach with a profitable and natural workflow to combine the qualities of handcrafted and rule-based modeling, Smelik et al. [33] permitted creators to focus on what they need to form.

The main methods used today when designing a road network for video games, simulations, the city, and even some other products will be procedural, model-driven strategies, system modeling, multi-dimensional arrays, and L-systems. As demonstrated by Greuter et al. [75], a straightforward road network model can be created using a thick square grid, although it is additionally conceivable to include relocation commotion to the network focuses to make a less tedious mesh and make the street organization more realistic.

The network of roads' inter-connectedness can be determined through constants that determine the density of the road and the distance between its intersections. Approaches designed to build road networks over larger terrains can be more interactive, taking important input from the user to maintain control. In the system mentioned in the article published by Smelik et al. [32], terrain and their features were first sketched by a user and then procedurally transferred to the map were designed using a colored grid, while roads and other smaller features were created using the standard drawing tool.

The L-system is better suited for branching patterns for bifurcation models and can also be adapted to road networks. In the production of road networks, methods based on the L-system [76] can use various techniques such as tensor field [77], templates [78].

In their 2002 paper, Sun et al. [78] suggested an alternative model-based approach for building cities. They have used adaptive templates by observing several patterns common in real road networks for city modeling. This system, where the user can divide the map into regions and choose which pattern to use, including population-based, raster, radial, or mixed patterns to create the roads covered for each, attempts to divide each of the regions into approximately equal segments based on the region's population density. Paths generated with short steps, in every one of which the system produces radials of unvarying size, paths created with short steps must select the radial with the least elevation change in a legal region [52]. However, the results in cities obtained using this technique only reflected defined patterns, failing to provide a realistic city road network complexity and scale.

Here are the articles that are vital within the advance of procedural strategies toward intelligently and more designer-controlled modeling. With Parish et al. [76]'s published work on automated procedural modeling of cities, he extended the underlying L-system into a seminal CGA Shape grammar. Their self-sensitive L-systems are designed as a capable apparatus for building great-looking virtual cities from scratch utilizing little factual and geographic information in a brief time. They created building allocations by expressing the road types they called "highway" and "street" for different purposes in L-systems. This system, which they called CityEngine [79], was later expanded and formed the basis for new studies like [77]. In earlier times, the application of the texture synthesis technique without considering geometric structures such as streets, parcels, and building footprints led to less consistent and unrealistic results [80]. Yet, Aliağa et al. [81]'s contribution has been a noteworthy step to creating a reasonably acceptable street network in terms of synthesizing both structure and image at the same time. Chen et al. [77] describing how to create common road models, combining the tensor fields with the high and low-level modeling, performed interactive modeling of road networks. The interactive road network design process, which starts with the user designing some basic tensor fields with a drawing interface and continues with the creation of street graphics from these areas with the help of hyper-current lines, finally ends with obtaining a three-dimensional city modeling from the graphic.

Benes et al. [82] have presented a maximally automated growth-based algorithm in which a formation that starts with an initial transportation grid is transformed into a completely developed city consisting of roadway systems, taking into account the surrounding terrain. It could generate content-rich cities on meaningful main roads that are automatically generated over time through a simulation of trade between neighboring cities, although it has its shortcomings.

Because The transportation system typically establishes city blocks and districts, Kelly and McCabe [83] created a real-time city customization and editing tool by mapping points on the three-dimensional topography. This application called Citygen is presented the model that designs the basic character of the city, which its main road network is the starting point of the production process. But in this model, the areas edged by the roads could be embellished with one of the patterns, such as orthogonal grids, and industrial roads that lead to dead ends.

Instead of typical city roads, the approach explored by Glass et al. [84] is based on a combination of a Voronoi diagram to describe the land parcel and road structure around each building in informal settlements. A combination of regular subdivisions with and without displacement noise was preferred for smaller roads in the system which has been very successful in recreating patterns. Compared to other adapted real-world examples, it seems to perform quite well as a noisy small path generation method.

Unlike other approaches, Lechner et al. [85] provided an approach based on individual entities that decided whether to add a direct road link to the network by placing two intermediaries, called the extender and the connector, checking the length of the path that leads to a randomly chosen location. In another study by Lechner et al. [86], to improve their previous work and speed up the urban connection, there were agency add-ons. The method, which gave reasonable results, is unfortunately slow.

There are even those who turn to use several modular road segments that are laid together to save time and cost when creating their entire road network [87]. They are a 3D modeling paradigm based on iterative rule sets that are derived from the book of Prusinkiewicz and Hanan [88].

There are also works like TownSim [89], inspired by Emilien et al.'s [90] work using rough terrain to create realistic countryside-like settlements. In this publication, Song et al. have presented an agent-based city evolution algorithm that can qualitatively assess the alignment, interconnection, and shape of the road networks they have developed. This work was motivated primarily driven by the aim of constructing adequate road networks for evaluating autonomous car applications. Khan et al. [91] presented an approach that uses procedural modeling to generate road scenes modeled in a real-world urban environment, with different influence factors and high-quality semantic annotations, and to rapidly-produce a synthetic replica of the real-world. Since the process of creating content in vast virtual worlds requires a lot of work and equipment to support the infrastructure, road networks must be randomly created by increasing the diversity in the environment to ensure sustainability. It will support this diversity with the use of procedural techniques that can assist road network models with or without detailed content.

Traditional procedural methods are also used to interactively identify road networks and create cities. Able to simulate the distribution of land-use in urban landscapes by using agent-based simulation instead of L-systems, Lechner et al. [92]'s agent-based approach greatly improves diversity and realism by automating the layout of buildings and roads in digital cities.

Another way to create a road is to render a high-quality aerial image by removing roads [93–95], buildings, and other components such as zebra stripes. However, these approaches, which produce editable results, were not exceptionally appropriate for creating a virtual street organization. Andersson et al. [96] has presented a procedural method to produce a believable three-dimensional terrain model based on roads. This method was designed to develop terrain-to-road fitting software that installs drainage channels alongside it, emulates the landscape model surrounding the roadway to rough terrain, and specifies the terrain type for each network segment. They leveraged procedural noise algorithms such as Perlin, Simplex, and Worley to alter the terrain morphology, while its compatibility is ensured by the radial basis function interpolation technique.

One of the studies carried out to meet this need is that of Güner [87]'s lightweight 2D road network model with low detail and light load. This method made it possible to change the road lanes and lines that contain basic elements such as intersections and pavements, which are included in traditional road network modeling applications, and straight and circular road segments consisting of a low number of polygons. This study, which aims to produce models even with limited hardware at low cost, can produce user-centered results. Their compared results with CityEngine show better performance in regard to texture and particularly polygon complexity.

Martek [4] presented a procedural approach to construct a plausible hierarchical road network in large virtual worlds with a terrain map created using basic information about cities. In this approach, a duo of parameters are utilized to rapidly generate the randomly discovered tree, which gives the user significant control over the output without much tweaking, but can cause big changes to the output of small paths.

Collecting and handling the preparation set for directed learning may be a resource-intensive and expensive errand. To avoid this loss Tremblay et al. [97] presented a system for object detection and preparing profound neural systems utilizing manufactured pictures. This approach, which utilized space randomization to produce manufactured information to train this network, not only generates images much faster but also contains plausable variations.

The two-dimensional road network model [87], which includes basic elements such as intersection and pavement, has low detail and better performance in light load, and uses straight and circular road segments consisting of a low number of polygons. Although the raw output of the model, giving a modeling framework based on street lines isn't palatable sufficient, it can be preferred to save laborious working hours.

Another study [98] using domain deep learning, which has an incredible impact on segmentation, used the GE-GAN. The study used the dual street systems of neighboring cities to predict interregional road traffic. It can be said that it constitutes an introduction to subsequent studies to detect and classify traffic incidents [99].

1.3.2.2 Buildings

Today, cities containing dystopian alternative realities with remarkably detailed descriptions such as Assassin's Creed [100] and The Witcher [101], which are developed animatedly, are frequently encountered. However, these cities, which are exceptionally exorbitant and time-consuming to develop, contain an expansive number of geometric plans counting thousands of structures and lanes.



Figure 1.10 An innovative shape grammar called CGA shape is employed to algorithmically create 3D models and environments in computer graphics [5]

Digital rendering, which can be encountered in many examples in life, such as artificial reality, computer games, movies, and simulations, can be a very challenging process in building modeling. Buildings are primary components with a geometric description basically consisting of standard parts such as facade, roof, and unknown, and attribute table data containing a set of key/value pairs [102]. It would be almost impracticable to manually represent each building by manipulating its materials and combining it them into appropriate formats in a unique view. This design process, which can be done by hand or using special programs, has become a necessity to be done without requiring much time and effort, given the problem with content creation, considering the increasing demand for products in these environments and today's rapidly changing conditions.

Utilizing pictures, procedural modeling, or linguistic use organizing is the as it were alternatives accessible to creators in the literature for generating buildings especially. Be that as it may, including more significance and genuineness might require extra exertion. Subsequently, the integration of additional procedures that use real data to automatically enrich the grammar with real-world data, and technologies that collect accurate exact spatial data approximately from building outside surfaces using images [103, 104], recordings [105], and remote scanning taken from a separate [106], increase accuracy and create a more realistic impression. However, it's critical to keep in mind that depending exclusively on these strategies might not be sufficient to

generate very extremely dense meshed building scenes using these methods alone, or may not produce effective results in detailed metropolitan with intricate characteristics. Notwithstanding, it's pivotal to mind that, even though it offers up-and-coming modeling, it can be seriously a significant roadblock for the method that needs reproduction. Therefore, it ought to in this manner be used mostly for more thorough simulations of already-existing settlements and cities extensively. With the capabilities of the LandXplorer [102] software architecture, efforts have been made to meet the demands of virtual three-dimensional town models that can manage, integrate, and distribute geographic information. Conversely, procedural administration [5] allows the creation of new structures and commendable, totally iconic virtual towns that incorporate more than fair individual development of the urban setting, allowing for much more thorough exploration, and greater-scale production at a faster pace than ever before.

A procedural approach that will make it possible to create a wide variety of scenes consists of a series of instructions that can be reproduced or modified according to user needs using simpler controls, rather than explicitly stating or hiding all their intricate details [107]. Procedural modeling, which works particularly well for the natural phenomena of plant resources, has reached a consistency where more complex objects can be defined, with the studies in [108], [14], [109], [110], [111], [76] and [112] which started with the definition of the L-system by Artistied Lindenmayer [113].

Another method used as an alternative is Stiny [114]'s shape grammars, which provide ease of use in architectural designs and analysis [115, 116]. In Merrel's system, [116], the user-supplied sample model is manually created with building blocks, which they refer to as model pieces that occupy every position in the 3D mesh, and then assembled by the user. On the other hand, some studies can find patterns for many curved surfaces by synthesizing beam patterns with polygonal meshes such as quadrilateral, pentagon, and hexagon [12]. But because they only synthesized mass conceptual architectures of various construction houses and other structures and were often built on manual and human decision-making, they remained largely conceptual tools and did not yield effective results when used in very large detailed areas.

L-system is one of the most promising design grammars used in plant modeling. It is an indisputable fact that building data, designed by a series of splitting steps rather than a growth-like process, is fundamentally different from that of plants. Because of this, there are some researchers [114, 115] that use alternative solutions considering that this method cannot give effective results in building design.

The studies have caused the proliferation of geographical data and have been published to the world in open sources such as OpenStreetMap [117]. However, these data, which have a flexible structure due to the flexible additive rules, cannot guarantee accuracy. Although various tools have been developed to resolve this inconsistency and improve data quality, data in urban areas requires technical expertise [118]. The article of Groenewegen et al. [119] is one of the methods created to do away with the requirement that external experts participate in the creation operation. This approach creates highly intuitive city layouts, allowing large city plans to be created in seconds, making it significantly faster than comparable agent-based software.

Although some think that L-systems are deemed unsuitable for the builds, in the literature, there are many successful studies [81, 116, 120] that focus on the problems of procedural modeling of structures, most of them focusing on grammar-based solutions and solid model creation process structured as the union, scaling, rotation, and translation of volumetric shapes. Addressing the modeling problem of urban architecture that can automatically model and create realistic buildings with different styles, Wonka and his team [115] have derived their building designs using a modern type of grammar that splits using the idea of shape as a foundation called CGA Shape. They designed an interactive town planning application with excellent detailed visual points, including feature-building models flexibly using a relationship matcher and a different control syntax. They have progressed in their designs by taking the L-systems in [111, 121] to the grammatical formations and by taking in [76] to the street modeling. Although very realistic, complex, and highly efficient buildings can be created using the split grammar technique, it also requires a certain competence and degree of experience, as a restricted quantity of buildings can be created in real-time use.



Figure 1.11 Making lively three-dimensional representations out of excellent two-dimensional geographical data

Split-grammar and converse methods for two-dimensional shapes have typically been used to create exterior schemes [31]. But the creation of builds involves a variety of interactive multi-tools. In city designs such as CityBuilder [122], the terrain is often produced to identify various building types once the road network has been established but does not create the actual physical characteristics of buildings, including their form and outward appearance. The patches arranged in a rectangular grid can be made to represent the world, or a patch can be added to create each building or road.

Greuter et al. [75]'s approach to procedural city generation is essentially an iterative process of expanding building floors generated by combining quadrilaterals that were generated at arbitrary to obtain the virtually limitless towns in actual-period. This system, which produces good buildings with simpler methods, has created buildings constructed from sections, each created from a unique floor plan, and extruded these to different heights. By achieving intuitively outline rates on hardware suitable for consumers, this approach minimizes the amount of procedural recreation of the build to effectively use memory and support a constraints frame value.

The walls, windows, and doors of the buildings, which are typically created by adding volumetric polygons divided into floors, are the resulting split facades. The resulting variation can be added with shape parameters, conditional or stochastic rules, or randomly generated numbers. We can say that the most advanced compact methods used for this are shape grammars.

Müller [5] introduced a novel geometry syntax, CGA shape, for creating huge-scale cityscape representations that have excellent visual appeal and geometrical features although initially built on division rules based on the simple L-systems technique. Offering a grammar-based solution for procedurally modeling the detailed structure of the building envelope this system, used a combination of volumetric shapes and transformation processes to produce mass models of buildings. Although the building offers realistic building models, it may lose the semantic information of the shapes inside. Finkenzeller and Bender [123], who present facades and roofs in more detail, propose to compensate for the loss of semantic information with a graph.

There are also studies on architectural modeling using cellular textures [124] and texture synthesis [125]. But they could not provide the desired performance.

Although the use of the L-system has not been as successful in building modeling as in plant modeling approaches, it is a fact that it is less complex. Toriah et al. [120] were able to generate procedural automated and comprehensive 3D solid building models using several modular construction methods such as offset, ellipse, L-system, smooth, polygon reduction, polygon division, transformation, and extrusion.

There are also many studies developed using this procedural approach. The Parish and Müller [76] are the first to create a series of models by applying the CGA rule technique for converting the 2D building footprint into a 3D model. In this first but highly valuable study, it is explained how a complete city that starts with a rectangular floor plan, can be built and modeled separately in a brief period without requiring a model for each building. In 2007, Müller et al. [126] continued their work and published an informative article on how to model building facades with the help of procedural models using photographs. They designed a model containing a facade view pattern,

along with the form grammar rules they created to reconstruct windows or doors using 3D objects. However, although these grammars provide versatile and often successful automatic control to the user, they are complex to use and require experience.

Weber et al. [127]. developed a three-dimensional interactive urban simulation system that changes over time by incorporating detailed urban geometry such as exact parcel borders, randomly placed roadways, avenue breadths, three-dimensional road topology, construction footprints, and three-dimensional building casings. The system, which offers rapid computation time and user engagement at an interaction rate of around one second for every temporal increment, has the disadvantage that it requires about a full day of reviewing the simulation and its parameters to produce high-visual-quality results. This technique optimizes the use of the building cache, resulting in much faster processing and lowering costs [52].

The industry, which has recently developed with the techniques used in recent studies, has turned to the use of artificial intelligence to create alternative city variations by using large data sets obtained from static 3D models. While most focus on city modeling for optimization of city configurations, a few have made improvements that facilitate previously developed simulations [128]. One of the most influential works recorded on this subject is the first version of CityEngine which was published towards the end of 2008, specializing in 3D model development [129]. Initially launched by Procedural Inc [76], The Environmental Systems Research Institute (ESRI) is presently investing the system into place [79]. It is one of the latest applications [130] creating extensive three-dimensional models to feed urban environment architecture, preparing, and simulation for modeling 3D geospatial data. Based on this 3D-GIS model, which enables the environmental assessment of large plazas and streets to be controlled according to the visual assessment at a small or large scale of the building shape, height, and color, it is possible to successfully regulate and optimize the quality of the space. The system allows the creation of several distinct building styles ranging from skyscrapers to commercial and residential, using the L-system building kit, each allowing for the addition of different products and providing an expandable solution [52].

There are some model-based studies [131] that adopt a "top-down" strategy that aims to best match the new image using an earlier model of the expected image. But it is unquestionably unsuitable for objects that exhibit major changes in their current form or allow changes in their topology.

While some can be derived automatically to create a building, such as the building's purpose, its assembly time, the count of floors over land, and its net interior space, attributes, which can also be in voluntary geographic information, can be quite useful. Combining different types of attributes and different usability-contained situations that occur in the actual planet where the data's fullness varies, can be very important to use predictive models and validate results. Biljecki [132] stated that the number of floors is a very useful estimator for estimating the elevation of the structure in the training-based model he created with the point cloud cluster they obtained from the City of Rotterdam. Using an index called the normalized perimeter, known as NPI, which expresses the proportion between the shape's circumference and the circumference of a circle of equal area, they concluded that shorter buildings have more neighboring buildings.

Although we generally focus on general modeling techniques, there may be detailed modeling such as polygonal formations that can be calculated patterns to structure faces using calculus (See [80]).

1.3.3 Natural Objects Generation

1.3.3.1 Vegetations

The generating and modeling of the plant kingdom, which is of interest to computer graphics research, have been a subject of extensive study and great interest in visual computing for over thirty years and is still an improving subject. Especially planting vegetation on existing terrain is becoming a widely preferred process for displaying natural formations in 3D. However, its frequent use cannot change the fact that this type of large-scale vegetation modeling is very costly in terms of time and effort.

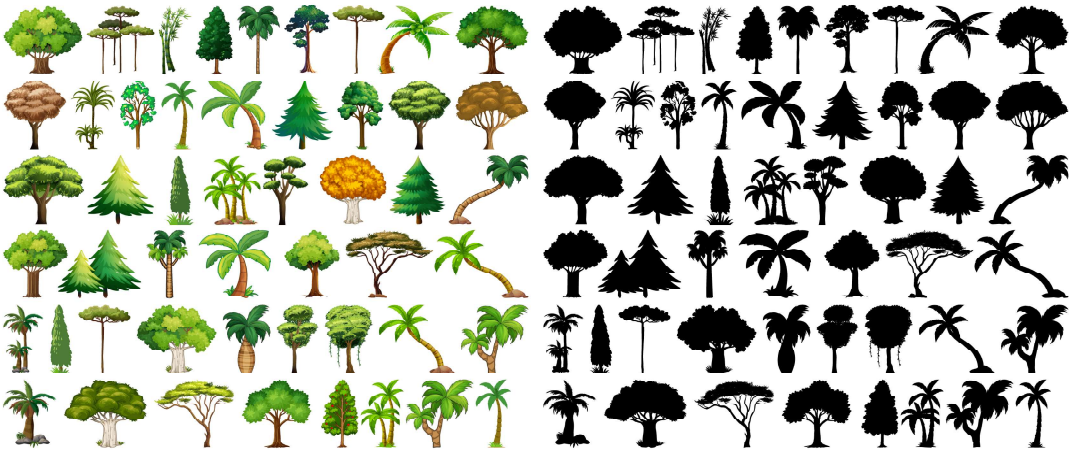

Figure 1.12 Simple Vegetation Types

It is incredibly time-consuming and challenging to simulate real ecological systems of plants, not only because of the complexity implied by geometrical details but also because of the complexity of highly detailed interactive biological phenomena such as the interplay between the growth and correlations of vegetation. The usage areas of plant modeling range from urban and environmental planning applications to the forest areas in 3D game simulations, from the research on ecosystems in forestry to virtual environments that are frequently mentioned today [133]. To perform many operations such as procedural models, growth simulations, environmental response modeling, and movement of plant models to create a realistic plant model, sufficient biology knowledge is required [6]. To make highly detailed realistic modeling, current approaches in the literature generally simulate ecological communities by abstracting the pertinent geometry to increasingly interconnected and layered representations including volumetric textures, voxels, and branch templates to overcome this complexity [121, 134–136], but many applications make use of detailed models that preserve the structure of a plant while remaining true to the descriptive characteristics of plants [137, 138].

Visual modeling of tree architecture which starts with Honda et al. [139]'s recursive model that was developed a tree by characterizing it with parameters such as branching angles and mode ratio, grows exponentially, is a part of the majority of producers' tree patterns that have been put forth so far. Later, Aono and Kunii [140] contributed by adapting this model to computer graphics, while Oppenheimer [141]

was instrumental in the development of the model. Other studies that contribute to visual realism with recursive tree models presented parametrically using the random and organized variation, the position of the affected tree branches, are by Lintermann and Deussen [142], Weber and Penn [143], Prusinkiewicz et al. [110], and Reeves and Blau [144].

Procedural plant modeling, which is created by starting from the root and progressing to gradually smaller branches or even leaves, may be obtained in numerous ways, containing the rule-based algorithm [14], the iterative operations [145], the cellular automata algorithm [146] and the combination of approaches [142].

Introduced in the 1960s to create a two-dimensional grid-based automaton exemplar of abstract branching trees, these modeling topologies went through many improvements until the 2000s, by reinforcing them with geometrical elements imaginative by biology. Extending the open leaf venation model to three dimensions to obtain realistic tree structures, Runions et al. [107]'s approach was an option in which competition between branches rather than the iterative branching process plays a considerable responsibility in decisive the shape of woodland and plants.

Another option used is procedural methods that allow efficient production and can be controlled parametrically in its most general definition. These kinds of vegetation generation can be quite diverse given its wide range of applications and techniques. Some studies emphasize authentic geometric depictions with a focus on specific plant components [147, 148], as well as approaches that incorporate biological processes to describe larger-scale ecosystems, often using interactive authoring techniques [149] that describe trees in more abstract terms. Therefore, it can be standard procedure to classify procedural vegetation formation according to user interaction [150, 151], plant reconstruction, procedural modeling level [152], and also the amount of information represented [34] to make more understandable the literature studies.

As another one of the basic procedural methods, tiling is structuring that allows large and detailed landscapes to be created from small texture sets and stochastic information while minimizing storage and memory requirements. They can be used with mapping

methods or selecting them with a number of parameters such as height and slope in the formation of more detailed natural elements such as rock, grass, sand, and snow. However, the majority of reviews have intense on placement branching buildings based on fractals [140, 141] or L-Methods [14, 108, 113].

The fractal models have received a quite little consideration while there are other procedural forms of native phenomena that have existed for lengthy but are less well familiar. It can be an effective mathematical method to present a realistic, vivid plant animation effect under natural conditions (see The fractal System [65]) for nonlinear science in scientific research. Oppenheimer [141] presented a fractal computer model controlling the geometry and topology with numerical parameters similar to the organism's DNA of branching objects, resulting in the diversity and realism.

Plant modeling, which holds long been a significant area of study for calculator graphs has received a considerable amount of attention. Since the introduction of the ecological arrangement prepared by Lindenmayer, also known as L-systems [113] in 1968 as a mathematical theory for biological development, it has been the subject of much thought and development in many areas of the natural sciences. The system designed by the biologist Lindenmayer as a scientific idea for biological development is actually an official syntax when it comes to computer artwork, particularly for the creation of fractal-inspired and lifelike modeling of plants. This concept which defines a linear partitioning mechanism as a syntax transformation system with generator sets consisting of only 0 and 1 in biology, became a useful tool of realistic modeling that later spread rapidly pertaining to hypothetical computer scientific disciplines, especially in graphics for calculators, where many objects. Aono and Kunii [140] and Smith [145] became the first company researchers to generate plants and trees with L-systems on the basis. After such upgrades, L systems which can be used to simulate the individual plants [153], trees [109] or entire ecosystems [154], was expanded by two major extensions: differential dL-systems [155] and open L-systems [111]. The point of difference between these fundamentally similar algorithms is that one supports the continuity of plant development in the simulation, while the other cares about the integration of exogenous flow and environmental

sensitivity. Prusinkiewicz [108] commented on [113] extended article with a geometric interpretation and recursion, whose system allows the creation of branching structures and three-dimensional depiction of the generated module sequence.

In order to prototype and render scenes with many plants, Deussen et al. [134] submitted the scheme and innovative implementation of a method that is described as a frame. This system produces a vegetation denseness graph that is resolute by customer revision or ecological system model. In this system, plant distribution could be added manually, using ecosystem simulation, or both, to lands created with an interactive graphic editor. Their procedural models included ecological parameters such as L-system and shade tolerance. Their methods have gained a degree of success that scales well to larger environments [156].

While sketch-based techniques developed so far can provide artistic freedom to model trees [157], plants, or even flowers [59] according to requirements, data-driven methods have focused on obtaining plants with more convincing and acceptable levels of detail [6]. The use of user-defined sketches, which can combine both in one spot, can be a powerful tool for creating complex tree geometry by transforming freehand sketches through probability, optimization, using user-defined branch plots as a tool to direct particle streaming when defining or rebuilding tree diagram forms or defining sketchy-based envelope shapes [158].

Several methods attempt to restore vegetation architecture from various information styles, including photographs [159–161], videos [162] or laser scanned dot clusters [163]. Systems have also been developed that allow them to more precisely reconstruct branching structures using multiple view images to meaningfully reconstruct the holistic 3D structure of a tree [160, 164].

With advancing technological advances, we see commercial developments applying L-systems to create rich land-forms containing detailed flora and trees that encompass a wide variety of different species. Approaches to modeling flora so far have generally been based on draw-based and other hands-on networks that are well-liked for their usability, biologically inspired L-systems or procedural models that use rules to

automatically construct the botanic forms of vegetation. Bradley et al. [159]'s paper presented an automated approach of picture-based restoration, teaching a mathematical exemplar suitable for immense-scale details-driven modeling from an iterative three-dimensional recovery process that creates an intense reconstruction.

Every natural feature in an ecosystem somehow interacts with each other and creates a response accordingly. Due to this interaction of the vegetation with the environment [111], maintaining manual control [165], creating or modeling the natural environment can be a difficult and unmanageable responsibility for large-scale environments. Many methods have been put forth for the implementation of adaptive plants due to the requirement to obtain vegetation representations that adapt interactively to varying situations and react to the presence of additional plants, shifting lighting circumstances, and the surroundings themselves. Some of them simulate facility development through parallel string rewrite systems using environmentally friendly query modules, while others address the necessity for vegetation to simulate adaptation using inversely procedural algorithms [107] or simulated adaptation [166] using inverse procedural models [167].

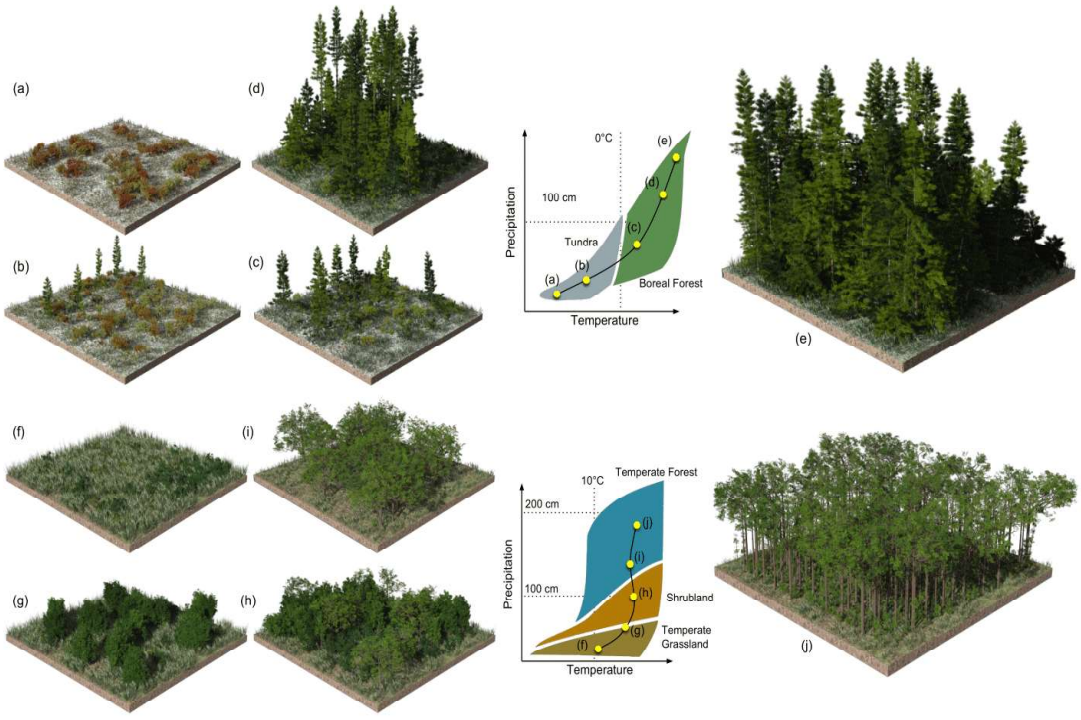

Figure 1.13 A Vegetation Generation in Ecosystem [6]

There are various methods focused on this interaction using algorithms such as random-walk [168], inverse modeling of the growth response [166], inverse procedural modeling [41], competing for sources and organizing oneself [165] or space colonization [107]. These interactions range from the simulation of dynamic plant life development and climbing plants to plant-liquid interactions such as wind and fire, which increases the realism of plant geometries by providing dynamic movements. While the vast majority of existing methods focus on interacting with support structures that do not support physics, providing limited control, Hädrich et al. [168] have presented dynamic growing development of plant models that simulates physical surroundings interactions, such as bending by combining plants with Lagrange fluid dynamics, branch splitting in addition to the plant's reaction to the storm.

Makowski et al. [6] used a multi-scale approach that is built upon the similarity between and within plant species and can faithfully display biological properties. To depict the branching structures characteristic of species for both single plants and entire forests, they combined multiple branch patches representing the topology of branching structures that took advantage of their similarity.

Zhang et al. [41], whose work demonstrates the close ties with procedural generation and procedural reconstruction approaches aimed at generating vegetation, chose to use a pre-existing deep neural architecture to capture abundant local details about vegetation distribution extracted from the landscape. While doing this, they were also able to add a mesh that creates a patch-wise matching-based density map of the target vegetation distribution and easily verify the connection between the image features and terrain they obtained from it by adding a procedural method. Their created method has shown that it can quickly match new realistic scenes by associating the vegetation distribution with the terrain. However, due to utilizing low-level attributes derived from pre-existing deep learning models, there are some requirements of memory of the CNN training model.

Other rule-based procedural builders, utilizing simple criteria that consider the terrain's elevation and gradient, for instance, are intended to yield outcomes that are,

by explanation, ecologically reasonable [156]. By Greene [146] a novel stochastic modeling was presented in which the beam data obtained by the evaluation of response to light, including illumination, and heliotropism, are represented using voxel space. Argudo et al. implemented the technique of image segmentation, which preserves the general shape of just one tree in the imagery and doesn't require modeling expertise, to build a compact representation of dense tree crowns using a circular distance map form, and is easy to compress the associated image-based representation.

Recently, literature research has focused on the adaptability to surfaces, the reaction towards realistic materials, and the behavior of plants that changed with dynamic states based on realistic physics [158].

1.3.3.2 Water Bodies

Procedural modeling methods used to obtain automatically acceptable 3D digital content when the grammar is properly defined, which has emerged to facilitate user interaction, have been highly preferred methods for a natural resource such as a river that provides natural adaptation to complex topology, various shape geometry, and terrain constraints.

Traditionally implemented legacy solutions were built around creating manual waterway landscapes using specialized software for creating models, often supplemented by mesh extension, particle systems, and animated textures, but where it was the user's responsibility for maintaining consistency between rivers and terrains. A shift in the creation of models of lakes and streams has occurred in the past few decades, including rivers, rather than the existing ready-made tools. These procedural techniques that produce fast and reasonable results have been used quite frequently, although the results are difficult to control for incorporating water resources into terrain modeling, such as river modeling in the field.

In the first study, an algorithm with recurrence was utilized to split and divide a single river course to construct a network of rivers with a geographical landscape.

Here, Kelley et al. have presented a strategy to mimic surface deterioration of stream networks modified utilizing triangulate and fractured interpolation techniques, based on empirical erosion models used in fractal-enhanced geomorphology. In later work, some have sought to create rivers in the mountains [18] and o create an elevation map with mountain ridges and river networks [169], while others have generated a planetary-scale river network [170]. The quick and efficient method [169] for densely interconnected river systems marked by an intricate river system flowing through its valleys preferred Gaussian plotting on the elevation map.

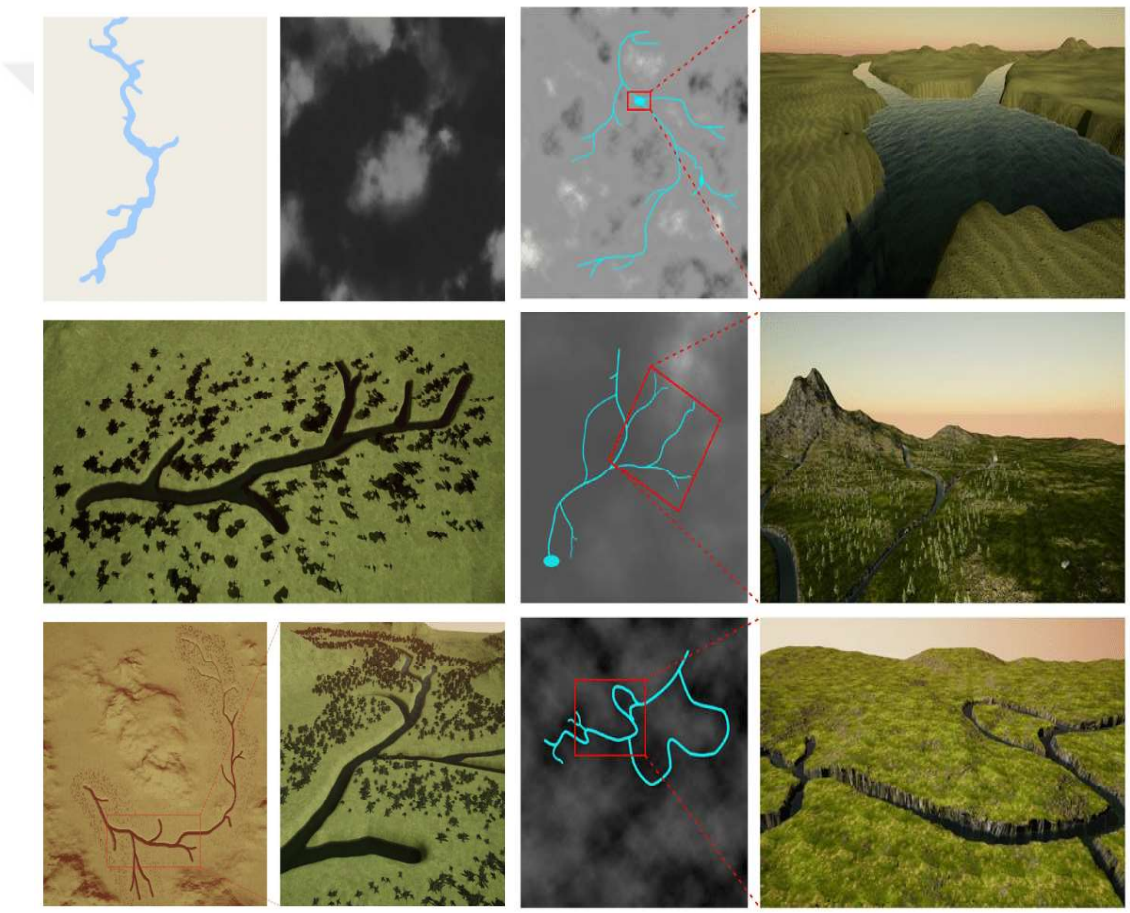

Figure 1.14 Original image from Google Earth and generated river flows over [7]

In the literature pool, procedural bodies of water, such as seas and ponds and their linkages, river systems, estuaries, and cascades have not received as much concentration as rivers, and are typically not considered [31]. Drawing attention to the lack of work in this field, Teon [19] proposed fast and simple algorithms that require further work to increase the realism of coastal features such as river bends, deltas, and beach formations.

Projected results may require iterative refinement because of the randomness that procedural processes can cause. Zhang et al. [7] have also developed a solution to this optimization problem with the model they developed on the problem of similarity calculation of two river views with the inverse modeling approach of rivers, which has not been discussed in the literature before. Zhang et al.'s reverse procedural exploration of the synthesizing of water instances, which extracts derived characteristics from samples to enable quick rendering of 3D content while maintaining a seamless relationship between water network and landscape, focuses on how to strike an ideal harmony among customization and interaction. However, there was also lacking controllability in this model, which was created parametrically due to the stochastic method they used and the effect of the existing terrain on the river network.

Genevaux et al. [171] has provided a terrain created by blending procedural elevation features and watercourse segments using shaping techniques, using a novel hierarchical hybrid terrain data representation. A workable way to create authentic terrain was to incorporate the understanding of the waterways into the landscape modeling process. Zhang et al. [28] discuss a procedural modeling method that uses principles of hydrological to simulate the natural processes of water flow and its inverse process to generate virtual terrains by the Midpoint Displacement method. In this article it was also discussed some challenges and limitations of this displacement technique such as uncontrollable terrain shape and terrain tending to a 'blocky' appearance. Their system which was adapted to the river network designed by Tokunaga [172] by the theory of hydrology, blended it with the procedural modeling to controllability and the Poisson equation to add realism.

In hydrology, research [28, 42] has shown that although procedural methods can often produce river views quickly, they lack user control. To better control the land structure, many researchers have turned to models that provide interactive editing with sketches [20, 33] that are preferred for modeling water resources but do not always guarantee physical fitness, which is another preferred method.

Using vector-based control curves as the general modeling principle to represent

geometric characteristics of landforms and bodies of water, Hnaidi et al. [54] suggested a collaborative approach that provides control using 3D curves based on the diffusion equation. Despite the grid structure based on differential properties capable of generating Spacious high-detail terrain, the challenge of physiognomy visualization still exists in river modeling [28]. By Emilien et al. [173] it was presented with an interactive procedural design, providing user freedom to shape flowing streams and waterfalls to model coherent complex waterfall scenes that automatically meet physical fitness in terms of the flow network. Samavati and Runions [174] proposed an interactive method for adding man-made structures, terrain features as well as the body of water region with a flexible triangle subdividing made using a multiple scales minimum-square surface structure, to Digital Earth.

While today's advanced interactive water production methods [35, 173] have not enough been successful in maintaining consistency between rivers and landscapes, they have produced more acceptable results than procedural methods [28, 175] for effective user control. Sample-based model synthesis preferred for the rendering of texture synthesis [176], generation of 3D geometric objects, trees [41, 60], road networks [177], landscapes [57], and buildings [81, 178], is one of the approaches that need to be explored in the creation of water bodies [7]. The new road network that Nishida et al. [177] created by extracting patches from samples and combining it with procedural approaches, inspired Zhang et al. [7] to create the river scene.

Recently, as a result of increasing applications of artificial intelligence, example-driven procedural modeling has also been combined with machine learning [179] Advances in computer graphics techniques and hardware have revealed that it will be possible to create more realistic terrains with deep-learning models. The insufficiency of the river delta configuration alone to generate an authentic terrain has prompted researchers to use the incorporation of creative methodologies including conditional generator adversary networks (cGANs) and L-systems.

Offering a new framework for modeling plots from input plots, Guérin et al. [180] suggest using a conditional adaptive adversarial network, known as cGAN, that has been educated on the realistic landscape to create new terrain. His approach combines

procedural modeling and interactive freehand sketching for usage in a variety of applications, including simulating erosion, developing terrain representations, and filling in absent sections within a coherent framework, overcoming the disparity between the ease of use and adaptability of interactive writing workflows, while harnessing the descriptive capability of real-world or procedural examples to create new terrain. To build convincing topography representations of river deltas and coastal regions from the water coverage map, Valencia et al. [112] offered a modular approach that combines several cGANs sub-modules. This approach allows for the creation of water valleys using polar regions, tropical, or both climates. However, the method's need for user-drawn sketches limits its capabilities as much as it needs a natural water map. With subsequent development, Valencia et al. [181] presents an automated adaptable technique that can produce plots with lifelike rivers bed formations with the integration of rule-forming cGAN with stochastic L-systems design to create original and realistic river deltas.

1.3.4 Virtual Worlds

The above-mentioned eye-catching aspects of the PM in use, often increasing the variety of uses, make it remarkably appealing for computer-generated world creation [34]. Virtual worlds, which are frequently used by the video games and movies industry to present the dystopian universe to the audience, are also frequently encountered in literature studies in many studies that are difficult and laborious to do in the real world [182]. Existing usage of realistic virtual worlds can show the varieties in many areas such as generalizes to a real robot scenario [183], analyzing Stereo Vision by the hazardous regions in real images [184], simulation of extreme weather conditions [185], smart cities and reality systems [182, 186, 187].

Virtual world scenes that use techniques that load different levels of detail require significant scale to encompass diverse elements, ranging from the landscape itself to vegetation and water features, and that comes with an incredible amount of geometry, and footprint challenges.



Figure 1.15 A Virtual City

It can be claimed that both of the most exciting potential uses of virtual environments are education and gaming, given the advantages provided to domains including training, learning, entertainment, online communities, online shopping, research, telework, teleconferencing, as well as public sector knowledge [188].

Virtual worlds are gaining more and more importance in the game and simulation industry. One of them is Minecraft [189] which has provided some pretty good virtual experiences in detailed and randomly generated worlds with deep-scale forests, high mountains, and vast oceans. The user can build anything he wants, from one-of-a-kind tree houses to hiding places in caves under the sea. Another example would be Second Life [190], which showcases a virtual reality that allows one to interact with real-life players from all over the world.

By Dunn [36], has been made a game engine development to generate terrain that can procedurally generate terrain, vegetation, water, and atmospheric effects at multiple levels of detail in a low-poly, no-texture art style using noise algorithms. His system, which uses the height map-based algorithm in distant terrain as well as a voxel-based algorithm in the nearby terrain using volumetric representation, gave the opportunity to display terrain features that transcend the capabilities of the height map portraying subterranean caverns, ledges, and cliffs. With a viewing distance of 25 miles at 180 frames per second in high resolution, it performed impressively in many scenes, including forests and sky, and water.

The desire to create serious games that focus on learning besides the aim of entertaining entertainment necessitated procedural content generation. Intense interest in PCG in many research areas is encountered not only in computer science but also in biology, psychology, architecture, and urban studies (e.g., the Esri CityEngine [79]). The area diversity can occur, from producing very specific content such as rivers [17–20], vegetation [41], roads [13, 21, 22], or buildings [5, 16], to more complex methodologies that produce different worlds that describe the selected terrain type or different features [33, 118]. In this subject, Smith's article on design, modularity, and algorithm content information and its usability to assist creative technology can be considered a basis. It can be described as the automated generation of digital resources for scenarios, films, video games, and other media using predetermined algorithmic principles for minimum users based on this article.

The games that cause these competitive developments in the game industry have led to the development of textures, 3D models, and many methods to obtain a unique-looking environment with little or no change, which can adapt to changing conditions when necessary. The necessity to show that the virtual world is worth seeing again by introducing new stories and impressions of mission and level changes has also led to an increased interest in procedural methods [23]. Due to factors like expense, physical accessibility, or potentially hazardous efficiency, a few educational tasks are very challenging to carry out in the actual world. Digital worlds can become an excellent opportunity to perform all kinds of simulations and activities avoiding learning barriers.

In the virtual world designed in The Polytechnic University at Madrid for practice laboratories, students were able to practice courses such as electronics, biotechnology, and chemistry designed without going to the real laboratory, even if the hardware or infrastructures were not available in real facilities at the university. The virtual environment created in a beautiful, comfortable, spacious, and learning-friendly way not only provides motivation but also saves time as it does not require physical participation [191].

Virtual worlds are a kind of metadata store with key features such as personality, physical nature, and persistence [188]. Furthermore, online environments offer dynamic and

realistic playing role modeling for teaching scenarios [188]. A lesson filled in the three-dimensional environment with scalar equations, such as linear algebra, has been explained with a virtual game with 3D graphics [187]. The challenge resulted in the victors obtaining learning points in the system calculated using a scalar representation of three-dimensional vectors, with each student represented as a person having graphic iconography and characterizing vector. In this way, they increased their experience of using 3D graphic objects using vectors. It is possible to create a virtual reality study for the visualization of cultural heritage objects that can show the user the difference between the real state of the object and its reconstructed state [186], as well as a virtual driving school established with the road network created in an unknown area in the real world.

In the discipline of artificial intelligence, there are many conceivable areas of perspective, such as level of popularity, mode of user engagement, computational intricacy, expressiveness, and quality structure. In the understanding of creating a natural virtual world, the development process, which basically starts with natural elements such as landscape, trees, and plants, continues with buildings and road networks, which are revealed by the interaction of non-natural human beings such as trees and vegetation.

As a result of evaluating the simplified nature-like creation in terms of reality, progress can continue with steps such as forest destruction and roadway expansion, re-orienting trees and water resources according to them by placing cities. As virtual worlds grow in size and detail, designers are encouraged to use procedural methods that are often unintuitive, difficult to integrate, and far from simple, providing little user control [32] (see, e.g. the article by [72] and [57]). Such large and extensive scenes that require a large amount of geometry to represent may require resource constraints based on GPU, processor speed, and a number of processors [36]. Freiknecht et al. [23] believe that it is essential to a balance between performance and reality fidelity. Also, researchers predict that geometric modeling techniques such as L-systems can develop better modeling/animation when used with PMs in virtual reality applications that do not have such usage limitations [24].

Unsupervised learning is still a relatively unresolved research topic, unlike supervised learning, which has recently gained considerable prominence with deep learning techniques. Generative Adversarial Networks (GAN), of them, are frequently preferred because they offer superior data generation capacity. Although there are some issues with this technique, which is notable for being a highly popular method in both semi-annotated and unsupervised exploration, including Nash's balance the inside correlate shift, manner breakdown, disappearing gradient, and the absence of suitable assessment criteria, these issues are still being successfully resolved. With the use of models such as 3D-GAN and Tree-GAN, real and manufactured 3D objects have also begun to be created [192]. The recent development of GAN, which has important effects on human society such as the world and climate changes, has started to emerge fast and practical applications that can reduce damage with the early measures it will provide [112, 181, 193].

Buildings, DTMs, bodies of water, vegetation, and urban furniture, CityGML [194] provides class definitions, arrangements, and semantic descriptions for geographic features in ever-expanding and expensive 3 dimensional artificial town modeling. It makes sense to use this CityGML format, which can transform them into a central information hub, to come to a uniform specification of important entities, properties, and connections inside a 3-dimensional metropolitan simulation that can be accessed by multiple users in various programs. Language, algebra, structure, and presentation are four distinct components of simulated three-dimensional city models that can be represented by this freely accessible data paradigm and XML-derived structure. The semantic information obtained by clearly associating CityGML objects, which are complementary to geographic visualization standards such as KML is likely to be used in the thesis. It can also be used to create thematic classes such as buildings, doors, and plants, and provide thematic attributes, allowing filtering of the objects and accordingly 3D graphic shapes, appearance properties, and materials.

1.3.4.1 Urban Settlement

The city structure, which has very different multi-pattern structures in its different regions or neighborhoods, expresses an extremely complex system, a more complex

process than building layout and shape when considered with the geographically interconnected structure. The 3D city model can be defined as an image of the outermost part of the planet that includes elements like trees, buildings, plants, and artificially constructed ones in urban areas [102].



Figure 1.16 An Urban View

Man-made phenomena such as dense urban settlements, which have recently attracted the attention of researchers, are very difficult to automate in realistic, detailed, and suitable for real-time use due to the human factor. These models, which are visually and functionally very complex and as a result of hundreds of years of development and evolution under the influence of numerous factors, are affected by many factors such as population, transportation, environment, altitude, vegetation, geology, and culture. Its creation cannot be accomplished in a single step, and a number of techniques must be applied, such as roads, plots, building structures, and building facades [52]. Since creating a virtual city is a very demanding process, a road layout must be designed and many buildings added. Generally, the generating cycle begins with traditional methods such as linear map information, digital elevation model, and panoramas that are based on laser-scanned excellent quality satellite imagery, or by using DSM and close-range imagery with texture modeling. [102].

As traditional methods consume a lot of resources and energy and cannot meet the increasing quantity and quality requirements, it has had to leave their place for fast,

semi-automatic urban modeling. When it comes to modeling an urban structure, the main studies we encounter in the literature have been various synthesis techniques, procedural simulation, and other partially automated development processes are some of the methodologies used.

Until nowadays, procedural methods, which were preferred only because there were no other options, are now one of the first choices that come to mind, over other expensive options, to reduce costs. They are ones of the best and low-cost 3D modeling methods which is preferred not only for simulation in the fields of virtual reality [76, 188] in the film and game [195] industries, but also for research and education [191] in urban planning, archeology, architecture and engineering [1, 4, 5, 7, 12, 20–22, 25, 29, 30, 32, 35, 36, 75, 77, 82–87, 91, 92, 96, 118, 126, 167, 170, 171, 173, 177, 179]. SimCity [195] is an uncommon instance of a constructed city combining urban research, even if it is a game constructed using simpler designs and game functionality that reflect practically all possibilities in city construction. SimCity became one of the initial towns to prepare videogames to offer the idea of an easy-to-use interface.

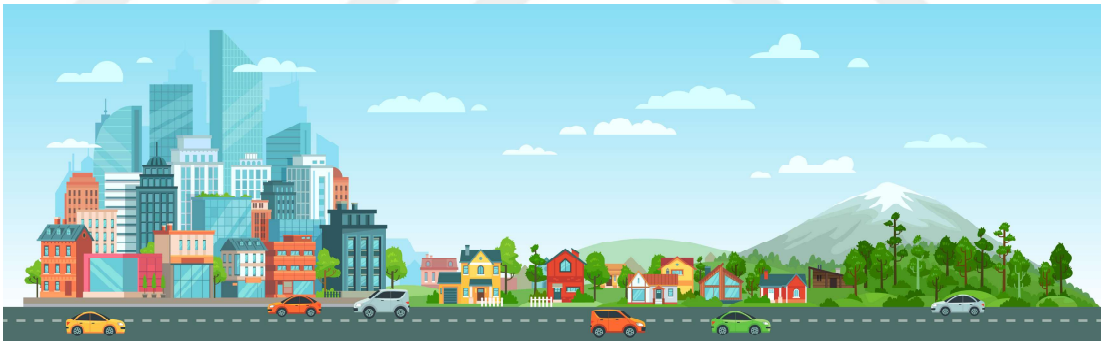

Figure 1.17 An Urban Generation

There are many models for urban settlement that allow modifications and explorations so that the user can more freely configure the existing design system. In urban design, which is only one of these areas, many methods have been discovered that allow changes and discoveries so that the user can configure the existing system more freely.

With Stiny [114]'s pioneering the idea of shape grammars, the design process based on line and point arrangement has started to be preferred in many studies. However, due to its iterative nature, it could not keep up with the necessity of automation [80]. The

work of Wonka et al. [115] has started a good development by making this grammatical structure more compatible with computer graphics.

In the field of urban planning, architecture firms like Houseal Lavigne and Woodbridge It prefer to utilize ESRI's CityEngine to depict designs for home development in the state of Illinois roadways landscaping bumpers in Tulsa, Oklahoma, and an additional entertainment zone in Oshkosh, Wisconsin [89].

In an academic manuscript during the SIGGRAPH 2001, a system called CityEngine that offers a high level of user control and uses a procedural approach based on L systems to model an entirety of town with an acceptable quantity of statistical and geographic input data was shown by Parish and Müller in their paper [76] on the procedural demonstrating of urban areas. As there are parcels defined by polygons surrounded by streets and roads, it is also possible to subdivide these polygonal regions with different partitioning methods. CityEngine is made up of a number of elements such as generating roads and constructing buildings and building facing, which divides the land into parcels and creates the appropriate geometry to the relevant data to form a highway and street system for city production. This system, which does not support dynamic texture creation, used a number of traditional facade textures and bridges were added manually to the city model. In such systems, highways or major roads originally drawn according to the Global Goals are then regulated to fit within a legal space, using locality-based restraints reflecting the current state of the existing road network [52]. Although procedural city building methods were initially created by combining typical road models on the basis of the road network, in recent years they can also deal with issues such as urban land use, traffic flow models, and even simulations involving agents [34]. Kelly et al. designed Citygen [83] created using procedural generation methods in real-time and user-interactive to quickly generate the urban geometry of a typical city which includes modeling a large number of spatial intricacies, such as topography, roads, structures, and related elements.

In order to model this complexity in a simpler way, a number of solutions have been proposed as used in statistical analyzes. While there are works for buildings and other structures, such as Legakis et al. [124]'s synthesis of modular patterns of bricks stone,

and walls, weathering and erosion are crucial components to depict urban reality. Some of them are Dorsey et al. [196]'s oxidation and erosion modeling on stones and Chen et al. [197]'s simulation of the effect of air on materials such as algae, rust, and dirt accumulation. To simulate road networks using tensor fields, Chen et al. presented an interactive visual simulation technique that uses aging particles to simulate changes in the material caused by the deterioration of various weather conditions that result in rusty and dusty appearances.

As evidenced by the research of Greuter et al. [198], the dense fixed block size grid layout is the simplest pattern-based technique that can be used. They have designed a real-time, free of repetitive processes, that looks different each time and continues almost endlessly, a virtual world they call the Undiscovered City, which enables the procedural generation of plants, trees, terrain, and other natural characters such as forests, mountain ranges or tropical islands. Using a concept that works in real-time at interactive frame rates, this work used a simple, uniformly regular, fixed block size grid layout, reminiscent of the center of a modern city, that creates a road network while placing the buildings they created using a combination of simple geometric primitives. This technique, which has limited realism, has been enhanced by incorporating an elaborate population-based, raster-based, or hybrid stencil technique offered by Sun et al. [78] It is created from highways obtained using pattern templates that form the skeleton of the road network. Providing a high level of user control for the city designs using various painting tools, determining city population, and planning primary roadways., the city's design provided better control than other studies but failed to manipulate minor features [199]. The sample-based method presented by Aliağa et al. [81] presented stochastic procedural modeling. It was suitable for expansion and interpolation as well as allowing manual additions. In the next study, Vanegas et al. [200] used a method based on subsequence building modeling by building user properties, which they created using a flat skeleton algorithm inside each parcel.

Neise et al. [201] have presented a new earning-based framework that can simulate the variation in vegetation shape and plant location among municipal districts to adequately replace artificial and actual urban landscapes with plant life. Kelly et

al. [202] presented a CNN technique, together with BigSUR, generated street-level pictures, Geographic Information Systems footsteps knowledge, and coarse geometrical networks that globally balance the sources of inaccuracy while creating weight representations that are partitioned conceptually and have corresponding exterior features. One of the shortcomings was that this mass model could not accurately capture Freeform buildings, using straight line segments for PE representative footprints-polygons and profiles. Gao et al. [203] have proposed a novel method consisting of models processed by CityEngine, tools for customizable urban simulation [79], and UnrealEngine4, a high-quality game rendering engine [204], to expand The range of morphological and lexical metropolitan sceneries in cyber-physical-social networks (CPSS) seen from above, to create a synthetic large-scale data set. They used an approach to the contextual texture that requires no humans to participate and produces rich contextual features, consisting of rules-based partitioned buildings with random parameter efficiency as high-rise buildings, advertisements, and residents. The real-time processing and randomization strategy seems to be a promising study in 3D large-scale data sets.

Shen et al. [205] have developed a workflow that can produce a three-dimensional urban representation from a two-dimensional image input. Tremblay et al. [97] presented a system that trains deep neural networks where the simulation parameters, such as lighting, pose, object textures, etc., are selected by the field randomization technique, detecting the object employing artificial images to account for the fluctuation in the system's real-world data. Utilizing the method of field stochasticity, to train deep neural networks has been a promising approach to harnessing the power of synthetic data to be photorealistic results. Lechner et al. [122] created procedural city models that use agent-based modeling to automate the creation of very large virtual two-dimensional urban landscapes that take a terrain map as input and generate geometric and scene graphics with roads and building models as output. This city designer not only generates a virtual representation of the road network and buildings but also simulates cityscape evolution, implementing an agent-oriented technique for building cities.

Designed for building large and structured cities, these algorithms are not well suited for

creating settlements such as small villages or farmlands [31]. In a study [84] to construct the street infrastructure in South African impoverished neighborhoods, a tessellation diagram for primary roads was created using stochastic grammars or secondary road subdivisions. Emilien et al. [90], on the other hand, presented an iterative technique to create a village structure with the desired characteristics, starting from the first road network, considering the constraints in the local terrain. Several building layouts were made according to weight criteria such as terrain height, slope, accessibility to road location, and distance.

Using the 2D digital map and aerial imagery, laser profiler data has been processed and converted into automatically created 3D city models. For example, MapCube, which is one of the 3D city models, has started to be used to cover all the big cities of Japan including Tokyo, Osaka, Kyoto, Hiroshima, Sendai and Sapporo [206]. There are also ready-to-use parametric procedural modeling techniques that offer a potent set of tools for visualizing and designing three-dimensional cities (also see SketchUp [207], Rhinoceros [208] and Autodesk [209]) suitable for 3D modeling, enabling rapid processes for urban design, model creation, and projection of urban simulations. CityEngine [79], an advanced procedural design tool, one of the best ones accessible, may require familiarity with the architecture to ensure effective use. In the era of smart cities, the urban design plan allows the user to clearly show, organize, direct, and project past and even future developments in cities. The desire of ordinary people such as archaeologists to visualize the existence of cultural heritage has led to the proliferation of visualizations of the past in CityEngine. Watson et al. [12] presented an analysis to reconstruct the design of a city at any time using the GIS database on CityEngine, which is used by laymen like archaeologists to virtually visualize the presence of cultural heritage. The results concluded that the system provides a user-friendly, satisfactory, high-level interface. Such visualizations not only simplify production but can also produce compelling visualizations that add imagination to design long before a building or even a city is built [12].

Creating 3D city models may require geospatial information that is widely available as open data but often unsuitable because of their poor accuracy and rough resolution (such

as 30 m). Therefore, data problems are frequently encountered in studies conducted on this subject. Biljecki et al. [132], who explored alternative ways of creating 3D city models to train and test predictive models in the absence of data, have demonstrated the computer-aided generation of three dimensions urban representations without terrain information. Given that the findings from Support Vector Machines and Multiple Linear Regression are inferior to RF, the present research selected to use Random Forests, a kind of supervised learning for classification as well as a regression that works by building a series of decision trees based on randomly selected portions of data. It has also been invaluable as it minimizes the number of predictive models by selecting only those estimators that are important, providing a reasonable solution to any problem encountered in the use of height measurements with data obtained with approximate heights until the problem is resolved. Today's research has turned to the use of machine learning and artificial intelligence to provide a diversity of 3D cityscape variations using the data sets they have obtained. Some focused on the generation of city models to enable the optimization of city configurations, while a few worked on the development of platforms to facilitate existing simulations.

1.4 Convolutional Neural Network

Deep learning is a fast-developing area of computational intelligence that makes use of multiple-layered artificial neural networks to learn and retrieve complicated patterns and high-level characteristics from massive volumes of information. It takes cues from the mechanisms of human cognition or cognitive architectures and is developed using some intelligent algorithms that aim to mimic the structure and function of the brain. It can handle large amounts of data and make satisfactory predictions, classify objects, recognize speech and images, and perform other complex decisions with well-suited remarkable accuracy.

Neural networks come in a wide range of varieties and are frequently employed in deep learning processes. Although CNNs, RNNs, LSTMs, and are some of the most well-known varieties, additional artificial neural network types include autoencoders, deep belief networks (DBNs), generative adversarial networks (GANs), and others to use for unsupervised learning, picture, and speech synthesis, and data compression,

among other applications. Understanding these different types and deciding which is the right choice is quite crucial for developing effective solutions to complex problems. Because of the CNN model's great accuracy in areas like machine vision and analysis of pictures, we are concentrating on it in the present paper.

A Convolutional Neural Network, or ConvNet, is a particular kind of deep neural network model that provides a lot of conveniences to automate processes by minimizing human effort as well as providing high efficiency in sampling. It refers to a special type of linear operation in which both image matrices are multiplied to extract their properties, with the Convolution theory at its core. The model works by applying convolutional filters to the input image, extracting relevant features, and reducing the dimensionality of the data. The output of these filters is then processed by multiple layers of the network to make a final prediction. Therefore, it generally has often found use in areas such as object detection in images, face recognition and classification of handwritten digits, computer vision and natural language processing, and segmentation of tumors and lesions for medical imaging purposes. When developing a CNN model structure because it contains many types of network layers, each with a different structure and basic mathematical operations, there is a mental progression to not only what each layer works for and how it works alone, but also how different layers interact together, affecting data transformation. One of the main challenges for CNNs in learning is the complex interplay between high-level integration of such transactions within the network. Despite this difficulty of use, the structure of the model, which finds a lot of use, is typically created by stacking three types of layer structures: an input layer, convolutional layers to extract features, pooling layers to reduce the spatial size, fully connected layers to perform classification, and an output layer to produce the final prediction.

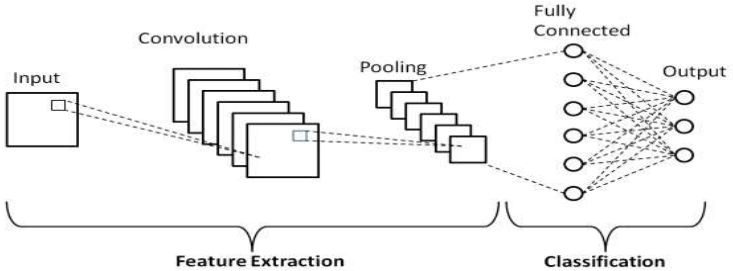

Figure 1.18 CNN: A Basic Structure

The input layer in the model is the first layer that receives the input data. It is responsible for accepting the image or other data and passing it on to the next layer for processing. The input layer does not perform any computation but simply acts as a placeholder for the input data. The number of characteristics in the data being input, like the total amount of words of a phrase and the individual pixel counting of an image, is equal to the number of synapses in the layer that receives it.

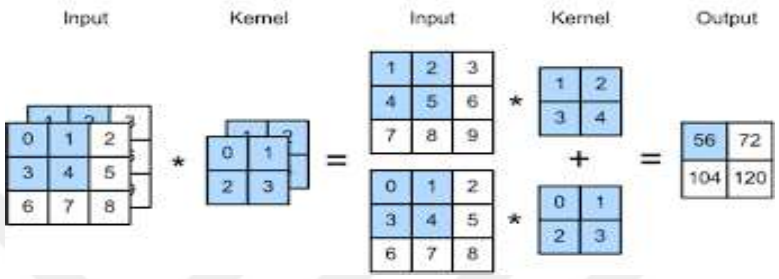

Figure 1.19 Cross Correlation 2 Channels

The convolution layer which is a key component of the model, performs a mathematical operation called convolution on the input data, applying filters to extract relevant features. It can be said that it is responsible for detecting patterns and features that are important in the intended operations, such as edges and shapes in images, thanks to its structure that allows the network to learn hierarchical representations of input data. To create a two-dimensional character mapping, every single filter in the structure scans over the initial information and computes the dot product between the filter weights and the data. This cross-correlation procedure typically begins with the convolution window in the upper left corner of the input tensor and proceeds by sliding along the input tensor, both left-to-right and top-down, to extract pertinent features from the information being supplied. Each of these input sub-tensors is multiplied by a core tensor on an element-by-element basis, and the resulting tensor is then added to produce a single vector value that represents the value of the final tensor at the relevant point. Using multiple filters with different weight sets to generate multiple feature maps that give us information about the image by using the mathematical operation, can improve performance by enabling the network to learn multiple levels of abstraction from the input data.

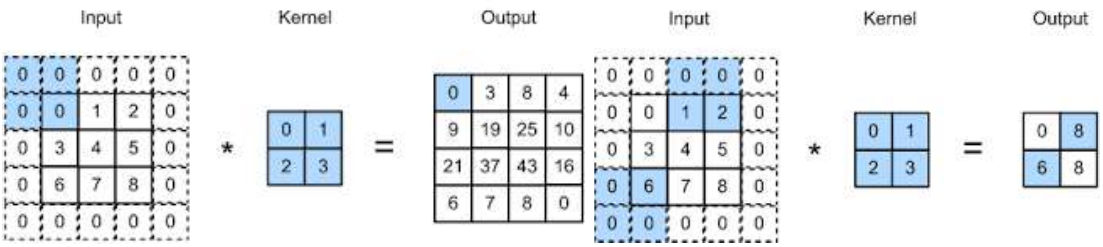

Figure 1.20 Correlation with Padding&Stride

The convolution layer, which allows the network to learn hierarchical representations of input data without disturbing the spatial relationship between pixels, feeds the other layers, most of which are the Pooling Layer, with its acquired features. This convolutional layer output can also be called a feature map.

It is a fact that with the application of these operations, the tensor input tends to achieve significantly smaller outputs. Among the layers, it can be said that the most popular tool that provides solutions for such image reduction problems to pixels is Padding **??**. Using Padding, it can be ensured that the values of unused extra pixels in the 2D Cross-Correlation Process can be set to zero, which causes problems with the use of small kernels. However, if the convolution kernel is large, dimensionality can be greatly reduced by simultaneously displacing multiple elements, bypassing intermediate positions 1.20.

The Pooling Layer is the intermediate layer that acts as a bridge between the convolutional layer and the fully connected layer. It is responsible for down-sampling the feature maps produced by the convolutional layer by reducing the spatial size of the feature maps while retaining important information. The pooling helps to prevent overfitting and provides more computational efficiency. In this layer structure typically there are two main types of pooling layers: max pooling and average pooling. Maximum pooling generates the output value by selecting the maximum value from a set of adjacent pixels, while the average pooling method averages the values instead.

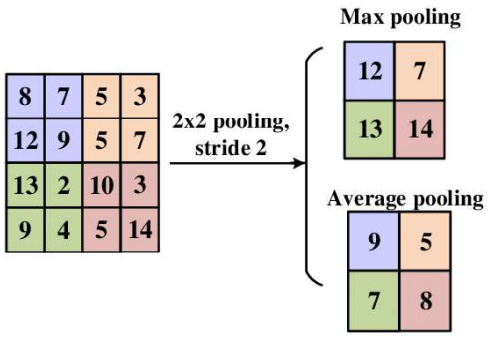

Figure 1.21 Pooling Layer

Depending on the method used, they reduce the size of the evolved feature map by reducing the connections between the layers. Thereby they allow the network to focus on the most important features and ignore less important details. It plays an important role to reduce computational costs in the overall architecture of a CNN. The layer typically operates on a small region of the feature map, and slides over the entire feature map by producing a smaller down-sampled.

A fully connected Layer used to connect between two different layers is usually placed as the last few layers before the output layer. These layers, which reduce human control and thus provide a more automatic structure to the CNN, flatten the inputs from the previous layers and put them in vector form. It is usually continued mathematical function operations in these few layers, and the classification process begins. The layer that drops out and the activation function are two additional crucial factors that influence the accuracy of the model in addition to the layers that should be included in a CNN model's structure. By removing neurons from neural networks during training, the dropout parameter reduces overfitting and enhances the efficiency of a machine learning model. The functionality of activation, which gives CNN a greater knowledge of a variety of continuous and complex interactions between the network's parameters and details on which neurons will be turned on in which layer, is another crucial feature. Today's preferred examples involve methods with unique applications like ReLU, Softmax, tanH, and Sigmoid.

With the emergence of modern CNN architectures, larger-scale networks such as AlexNet, GoogLeNet, ResNet, and DenseNet have been used in the industry.

Considering these developments, we examined the existing modern CNN models before creating the CNN architecture that we will use in the thesis.

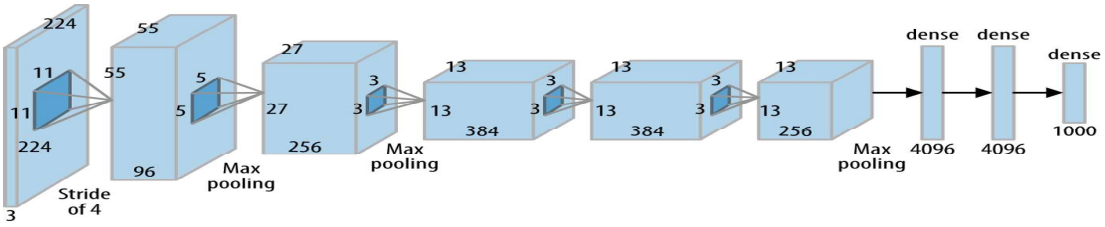

Figure 1.22 Alexnet

Although they are strikingly similar, AlexNet which has a much deeper structure than LeNet, preferred to use eight layers, including two completely linked concealed layers, one fully interconnected final layer, and five layers with convolution. However, using the sigmoid activation function as a simple ReLU activation function created the risk of not successfully training the model due to the risk of obtaining a gradient of almost 0 in case the model parameters are not selected properly.

The GoogleNet architecture, on the other hand, was created a little differently from other technology architectures. This architecture uses a stack of 9 starting blocks arranged in 3 groups, which includes many different types of methods that allow it to create a deeper architecture. Maximum pooling in the starting blocks reduced dimensionality while weights and convolutions were added to reduce bias. The number of channels selected, the number of blocks before size reduction, the allocation of capacity between channels, etc. Considering the processing load, it can be said that its use is computationally complex and requires experience.

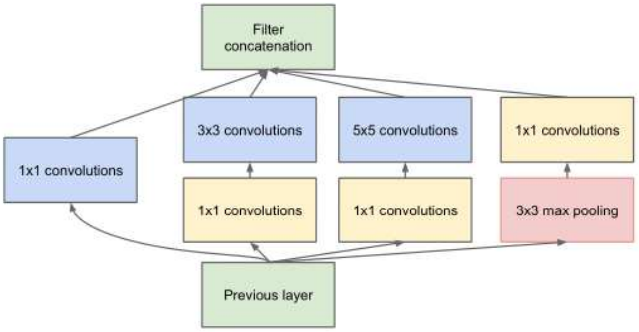

Figure 1.23 GoogleNet

As the model starts to contain more hidden layers and the structure starts to get deeper and deeper, things can become inextricable. For this, it has created a ResNet architecture similar to GoogLeNet in layers where all neurons within the layer share common weight values in the hidden layers.

1.5 Our Contribution

While the majority of studies in the literature build a model by producing man-made or natural objects separately, another part is in the approach of producing only selected objects based on fixation. These approaches can be quite diverse. Although there are studies that produce narrowly centered on one data type, there are also models that raise the level of producing more realistic models by combining various object combinations such as tree and building data. Based on these approaches in the literature, our motivation has been to develop a tool that can meet the needs of the user more flexibly and produce a model. This tool, in which we define each type of object to be added to the Terrain as a layer, was established as an apparatus for producing a synthetic DSM that can generate layers randomly according to user demand or according to the layer constants given to the system. Thus, a flexible tool has been obtained that can generate the desired layers randomly according to the user's request, and the desired layers depending on the given input constraints. In order to achieve such a flexible structure, we used the deep learning technique, which is very popular today and produces very good results. In the system defined as DSM = DTM + layer1 + layer2 + layer3 + layer4, where layer1 was the road, layer2 was the building, layer3 was the water and layer4 was the trees. The original architecture we created is inspired by the autoencoder architecture. Our results show that our method has potential as a successful technique.

CHAPTER 2

PROPOSED METHOD

The primary research question of this thesis is whether a simple CNN model, which is trained using DTM, building, and tree data, can effectively generate a DSM from a given DTM data, even though DTM data does not contain information on the length and height of objects such as buildings and trees. In other words, the goal of this thesis is to investigate whether the CNN model can successfully extract the necessary information from building and tree data to create a DSM that includes the height information of these objects. This is an important question to answer because DSMs are widely used in fields such as urban planning, cartography, and environmental science, and having an automated method for generating DSMs would save time and resources compared to manually collecting height data for each object. Therefore, the results of this study could have practical implications for these fields and could lead to new applications of CNN models in remote sensing and geospatial analysis.

2.1 Model Architecture

The objective of the thesis is to train a CNN model that generates a DSM model for given DTM and object data as layers. Since DTM data does not contain information on the height and length of objects, such as buildings and trees, the CNN model needs to learn how to generate the missing data from the available building and tree data. 2.1 illustrates our approach using the CNN model to generate DSM data using the DTM.

The proposed CNN model consists of multiple layers designed to process the input data and produce an output DSM model. With 5 convolutional layers, each followed by a ReLU activation function, the generated initial model passes a 3-channel input image (DTM, building, and tree) through convolutional layers with kernel size 5 and padding 2, producing 16, 32, 64, 32, and 16 output channels, respectively, before obtaining a single output channel. The final layer uses a linear activation function to produce the output DSM model. Using a linear activation function in the last layer is a common approach for regression problems where in our case we are doing regression on a per-pixel basis (see 2.1).

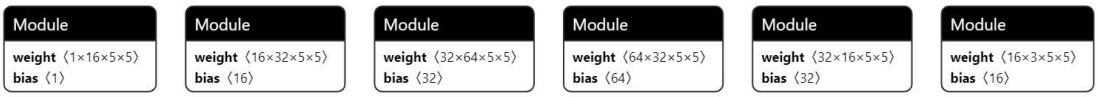

Figure 2.1 Initial CNN Model

2.2 Dataset

The DTM data for the city of Berlin was obtained from a public data source provided by the ArcGIS Pro software and downloaded in a standard format. The resolution of DTM is 2.4861 meters per pixel where the image has a size of 1024×1024 ($2.5\ \text{km}^2 \times 2.5\ \text{km}^2$). Center coordinate is 52.524886, 13.562331 in WGS84 (EPSG:4326). We then processed the data to ensure that it was in a suitable format for use in our model. Specifically, we converted the data into tiff format, which is a widely used format for geospatial data. Obtained data in Tiff format has 32-bit floating format where each pixel represents terrain elevation at the location of that pixel. Note that, since DTM is generally smooth, it can be up-sampled to higher resolutions using bicubic interpolation if higher resolution DSM is required.



Figure 2.2 Berlin DTM Data

In addition to the DTM data, we also obtained building data for the city of Berlin from the OSM-Buildings website (https://osmbuildings.org/). Layer data downloaded from the OSM-Buildings website contains the building's footprints, roofs, and heights. Building height tiff data are constructed by rasterizing the OSM-Buildings building layer. The resolution of Building tiff is 0.1 meter per pixel where the image has a size of 25458×25458. Tiff data are in a 32-bit floating format where each pixel represents the building height at the location of that pixel.



Figure 2.3 Berlin Buildings

We then integrated this building data with the DTM data to create a comprehensive dataset for use in training our CNN model. This involved merging the building data with the DTM data by aligning the coordinates of the two data sets and assigning the building heights to the appropriate locations in the DTM data. Since data is quite large we divided the data into smaller patches.

DSM of Berlin can be constructed by combining DTM, buildings, and other objects such as trees. We used a 3D tree library to populate a large set of trees.

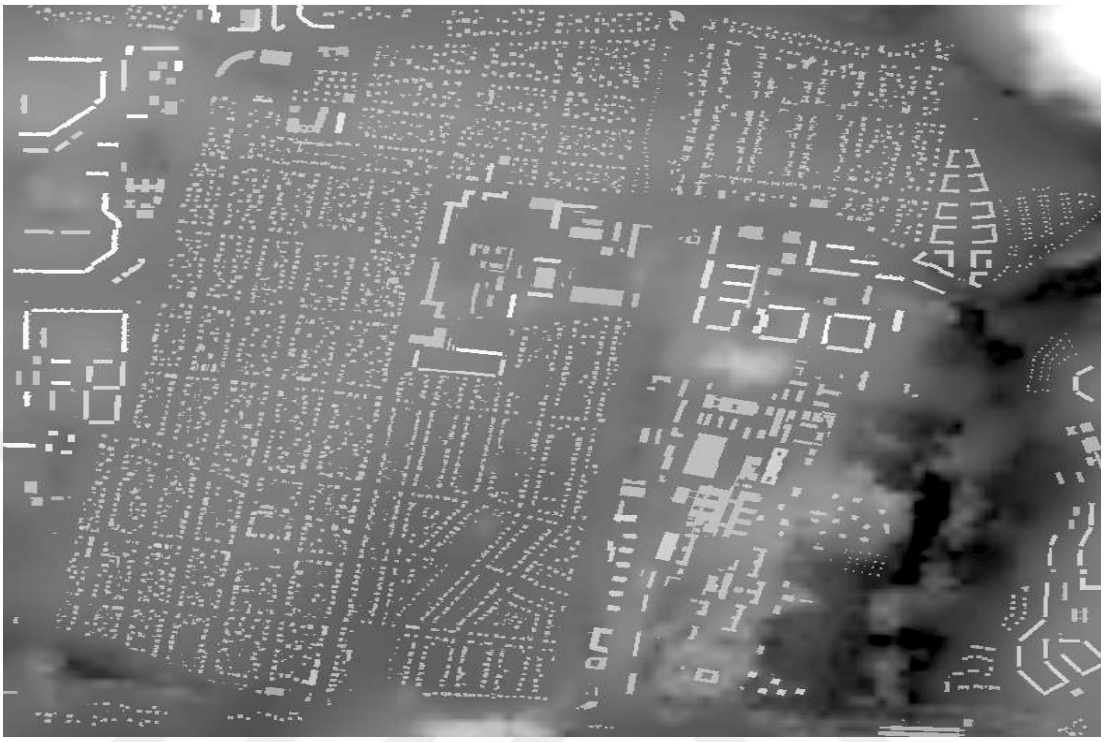

Figure 2.4 Berlin DSM Data

The resulting dataset consisted of three channels as input data: the first channel contained the DTM data, the second channel contained the building data, and the third channel contained tree data. Constructed DSM is the desired output (target) data.

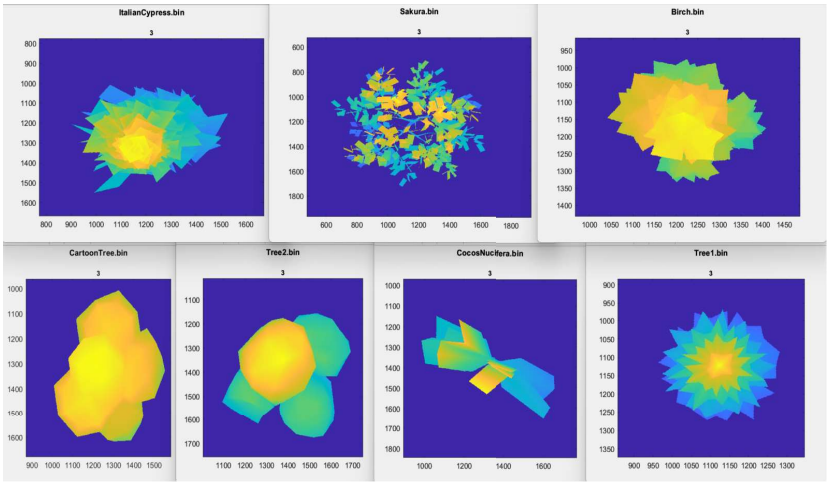

Figure 2.5 Low-Poly Tree Models

When the tree channels were added to the dataset, we included low-poly tree objects that are simplified 3D models of trees with much less detail than their high-poly counterparts. While it can generally be said that including low-poly trees provides enough information to create a DSM model, it has been acknowledged that this approach has the potential to reduce the computational resources required to train a CNN model. During this process, the Birch, Cartoon Tree, Cocos Nucifera, Italian Cypress, and Sakura low-poly tree models downloaded from the website [210] were converted into height maps, and DSM data were combined to produce DSM patches that we will use for training our CNN model. So, our model gets DTM, building layer, and tree layer as 3-channels input and tries to construct a DSM as an output. Of course, if the building layer and tree layer are high-quality then this neural network model is not useful. Here we expect the neural network model learning to create a high-quality DSM using a low-quality building and tree layers. Therefore, we smoothed and add noise to the building and tree layers for creating the training data.

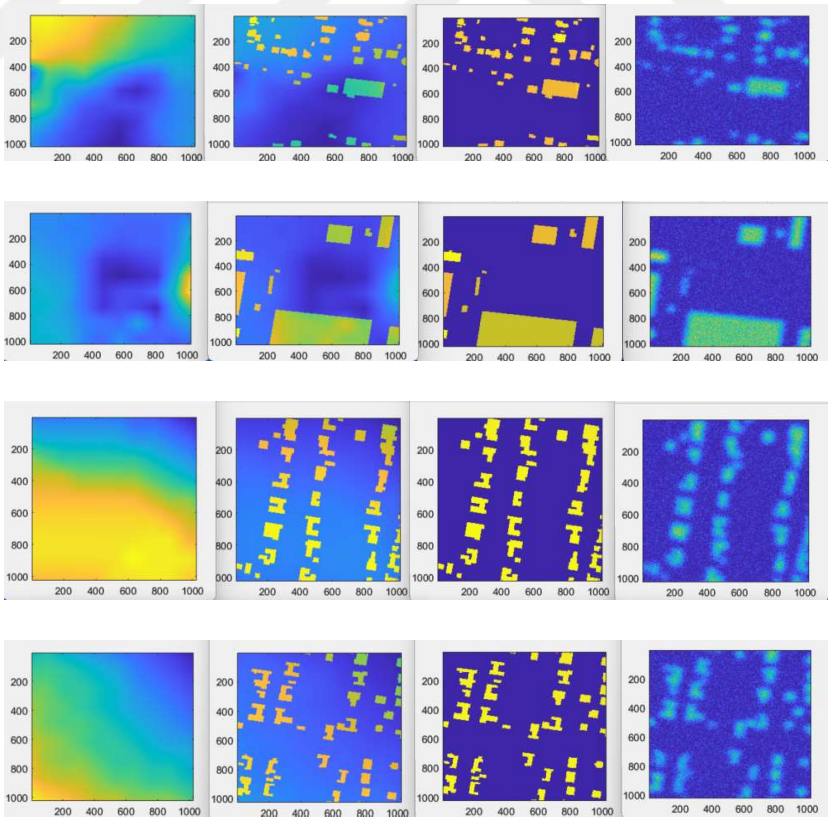

Figure 2.6 Figures represent respectively DTM, DSM, Building, and Incoming Building Data.

The data set used to train and evaluate the CNN model consisted of 2349 samples, which were subsequently split into 65% for training, 20% for validation, and 15% for testing. However, due to time constraints, the sample size was reduced to 200 for some evaluations.

2.3 Model Training

Initially, the performance of the Convolutional Neural Networks (CNNs) model for the DSM prediction task was evaluated without employing k-fold cross-validation. The Mean Squared Error (MSE) metric was used to assess the quality of the model predictions between the estimated and actual DSM values. The optimizer used was the Adam optimizer, which is a Self-adjusting learning rate optimization algorithm. This initial model was trained using varying epoch lengths and batch sizes between 10 to 100. While undergoing the training phase, we used a knowledge acquisition rate of 1×10^{-3} to optimize the model's performance.

Following the initial evaluation, the k-fold cross-validation technique was applied to obtain a more accurate estimate of the model's performance. A value of k=10 was chosen for the cross-validation, dividing the dataset into ten folds. During training, a batch size of 128 was prefered, allowing for efficient weight updates after processing each batch of 128 samples. Although MSE is in the squared form, it was noted that MSE alone failed to give a clear assessment of the prediction error. After finishing the cross-validation with k-folds and training procedure, it was decided to move to the Root Mean Squared Error (RMSE) measurement for an interpretable evaluation to address this problem. The RMSE scores collected for each fold were averaged to obtain the general efficiency metric for the CNN models.

CHAPTER 3

RESULTS

In our initial model testing, the model using a batch size of 16 and a learning rate of 1×10^{-3} on a data set of 200 images for 40 epochs with 5 Convolutional Layers performed with acceptable accuracy on the training set. The performance of the model using the mean squared error (MSE) loss function and the Adam optimizer on the first 8 examples of the test data set was visualized by using the results obtained from the test data set and the corresponding ground truth labels 3.1.

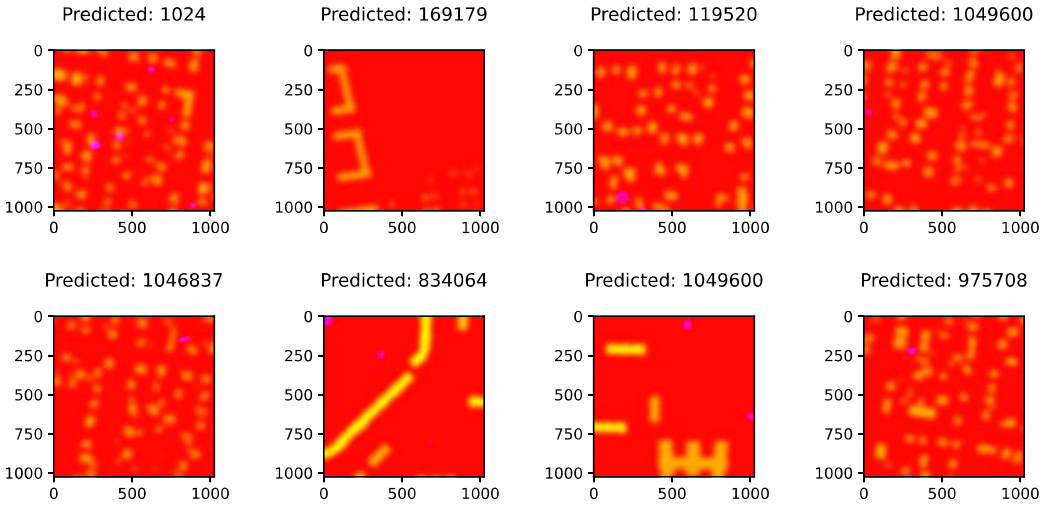

Figure 3.1 Prediction Result Of Initial CNN Model

When using 40 images for validation, we achieved a validation loss of 3.333758. Although our model showed promising results on the validation set, further research is needed to improve the model's performance on data.

To improve the learning performance of the model, we increased the epoch number to 50. As a result, the test loss decreased from 5.104754 to 3.767403, indicating an improvement in model performance. However, since increasing epoch numbers can lead to problems such as overfitting, further testing is needed.

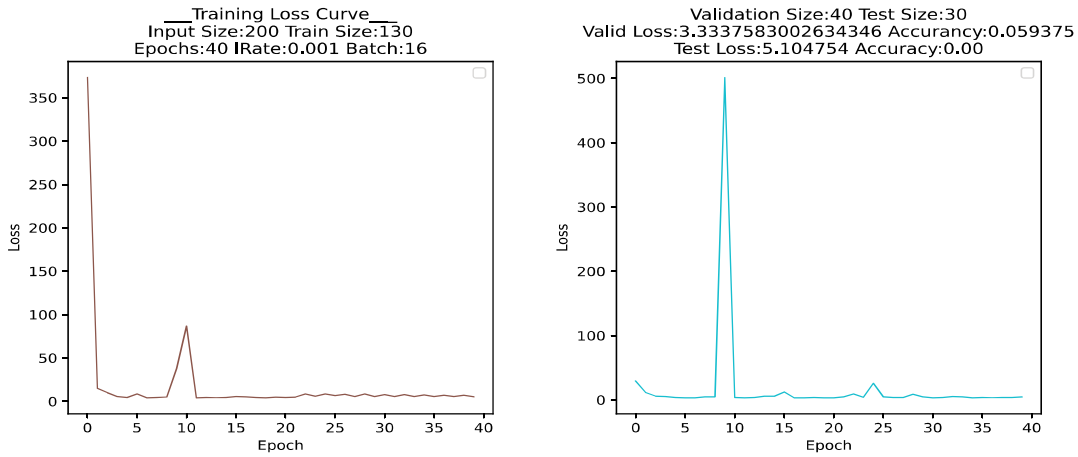

Figure 3.2 Loss Curve Of Initial CNN Model with Adam Optimization

We attempted to prevent overfitting and encourage the network to learn more robust and generalizable features by using a dropout rate of 0.5 in the CNN model. We closely monitored the training progress and evaluated the balance between performance and training time, but unfortunately, we were not able to achieve the desired result.

To find the best performance, we also tried different batch sizes, including 16, 32, and 64 on an average GPU in the Google Colab environment. However, due to memory limitations, the efficiency did not improve beyond a certain point and we obtained the best result with a batch size of 16.

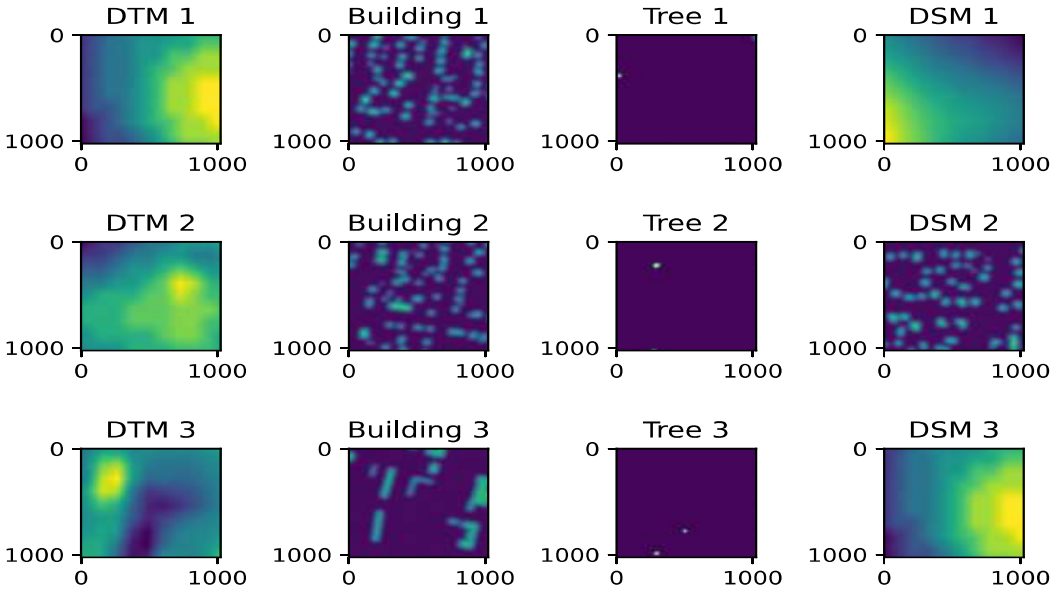

Figure 3.3 Prediction Result Of CNN Model with Adagrad Optimization

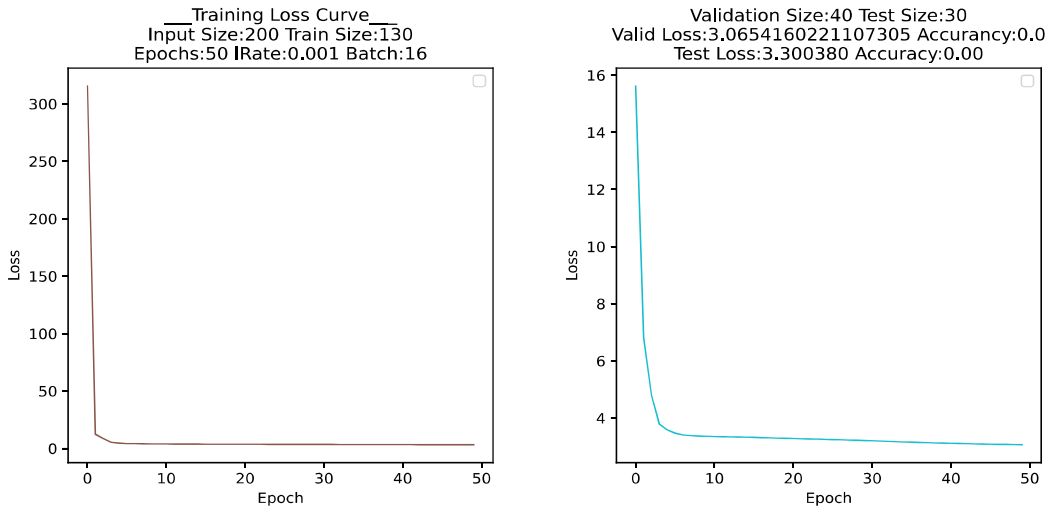

Figure 3.4 Loss Curve Of CNN Model with Adagrad Optimization

The optimizer of the model was changed from Adam to Adagrad 3.3. As a result of this modification, a significant improvement in accuracy on the test data set and a reduction (from 3.767403 to 3.30038) in the loss function were observed in the performance of the model. Specifically, the Adagrad optimizer enabled faster convergence of the model and better results compared to Adam.



Figure 3.5 Layers Of CNN Model having Multi Sequential Layers with Adagrad Optimization

To determine the impact of adding a second layer on the performance of the model 3.5, we updated the existing CNN model by adding a layer, resulting in a total of two layers.

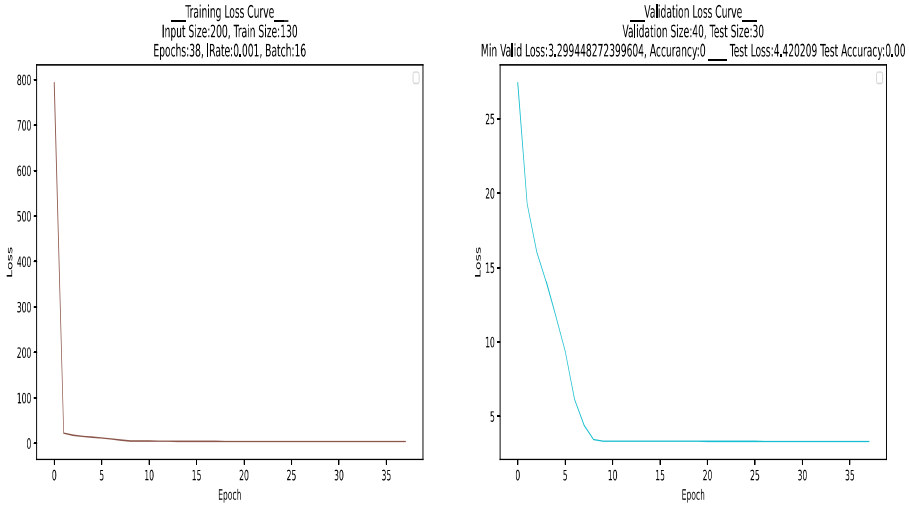

Figure 3.6 Loss Curve Of CNN Model having Multi Sequential Layers with Adagrad Optimization

The first Sequential part has convolutional layers with increasing depth from 16 to 64, after starting with 3 channels, and outputs a single channel. The second Sequential has a similar structure, starting from 32 channels and increasing in depth up to 128 and then decreasing to 64, 32, and finally a single channel. Each convolutional layer is followed by a ReLU activation function to introduce non-linearity into the model. The new architecture was trained on the same data set as the original model, and the results unfortunately didn't show a significant improvement in the accuracy of the test data set. There wasn't any significant improvement in the loss curve.

Following the completion of the k-fold cross-validation and training process 3.7, the results indicated that the CNN models achieved an average RMSE of 0.345, demonstrating their improved predictive performance compared to the initial evaluation using MSE. These findings underscore the effectiveness of employing k-fold cross-validation in assessing the performance of CNN models for the given classification task using a more interpretable metric like RMSE. The incorporation of RMSE and k-fold cross-validation provided a more robust and reliable assessment of the CNN models in the given classification task.

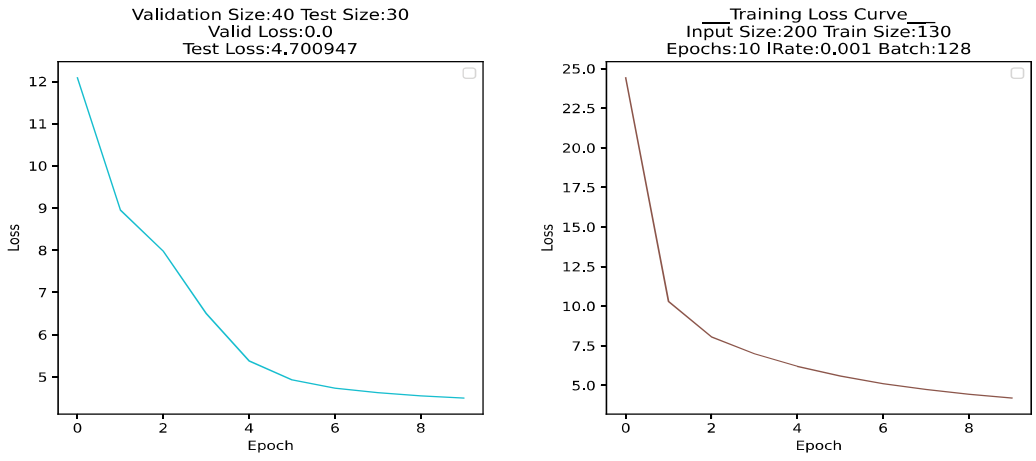

Figure 3.7 Loss Curve Of CNN Model with Adam Optimization using k-fold cross-validation with RMSE metric

We expect the CNN-based model to successfully learn the patterns in the training data and generate a DSM model with reasonable accuracy. While we have not been able to accomplish it yet due to limited computational resources, we aim to analyze the performance of the model on different subsets of the test data, such as building-only and

tree-only samples, to understand the model's strengths and weaknesses. Overall, these hyperparameters and techniques led to a well-regularized and accurate DSM model to generate DSM models from DTM data using only building and tree information. Based on preliminary results and performance improved consistently, we expect the model to be highly encouraging, and we believe that our model has the potential to provide significant advancements in this field with proper enhancements.

Overall, the combination of DTM, building, and tree data provided a comprehensive data set that allowed us to train a CNN model to generate a surface, DSM, from a terrain, DTM, data.

CHAPTER 4

CONCLUSIONS

The proposed synthetic DSM generation model employs deep learning-based architecture which can learn the texture patterns and structures of the soil, buildings, and tree surfaces. Such patterns are learned through training using a hybrid data set that contains real-world and synthetically generated data. Generation of the synthetic data contains randomness so a large number of data can be generated which reduced the extra need for data augmentation. Therefore, experiments also showed that a rich set of synthetic DSM can be generated for given DTM and real-world or random sketches of the building and tree as separate layers. Thus, the proposed model can generate a realistic DSM based on real-world and random user inputs.

In the literature, there is currently no approach based on deep learning architecture that can generate (i.e., hallucinate) buildings and trees on top of the given DTM and random sketches of buildings and trees. These sketches guide the synthetic data generation, so obtained results are not completely random. Additionally, there is no existing method that can learn the texture models and structures of soil, buildings, and tree surfaces. Therefore, it is not possible to make any comparisons with the literature or draw conclusions in this regard.

REFERENCES

[1] Doran J. and Parberry I., "Controlled procedural terrain generation using software agents," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 2, pp. 111–119, 2010.

[2] Raffe W. L., Zambetta F., and Li X., "A survey of procedural terrain generation techniques using evolutionary algorithms," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[3] "Dense digital surface model," https://www.l3harrisgeospatial.com/portals/0/images/3-19_SatelliteSurvey_Pic4.jpg, 2022.

[4] Martek C., "Procedural generation of road networks for large virtual environments," 2012.

[5] Müller P., Wonka P., Haegler S., Ulmer A., and Van Gool L., "Procedural modeling of buildings," in *ACM SIGGRAPH 2006 Papers*, 2006, pp. 614–623.

[6] Makowski M., Hädrich T., Scheffczyk J., Michels D. L., Pirk S., and Pałubicki W., "Synthetic silviculture: multi-scale modeling of plant ecosystems," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–14, 2019.

[7] Zhang J., Wang C.-b., Qin H., Chen Y., and Gao Y., "Procedural modeling of rivers from single image toward natural scene production," *The Visual Computer*, vol. 35, no. 2, pp. 223–237, 2019.

[8] Lee H. S. and Younan N. H., "Dtm extraction of lidar returns via adaptive processing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 9, pp. 2063–2069, 2003.

[9] Sithole G. and Vosselman G., "Experimental comparison of filter algorithms for bare-earth extraction from airborne laser scanning point clouds," *ISPRS journal of photogrammetry and remote sensing*, vol. 59, no. 1-2, pp. 85–101, 2004.

[10] Unger M., Pock T., Grabner M., Klaus A., and Bischof H., "A variational approach to semiautomatic generation of digital terrain models," in *International Symposium on Visual Computing*. Springer, 2009, pp. 1119–1130.

[11] Nar F., Yilmaz E., and Camps-Valls G., "Sparsity-driven digital terrain model extraction," in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2018, pp. 1316–1319.

[12] Watson B., Müller P., Veryovka O., Fuller A., Wonka P., and Sexton C., "Procedural urban modeling in practice," *IEEE computer graphics and applications*, vol. 28, no. 3, pp. 18–26, 2008.

[13] Bruneton E. and Neyret F., "Real-time rendering and editing of vector-based terrains," in *Computer Graphics Forum*, vol. 27, no. 2. Wiley Online Library, 2008, pp. 311–320.

[14] Prusinkiewicz P. and Lindenmayer A., "Graphical modeling using l-systems," in *The Algorithmic Beauty of Plants*. Springer, 1990, pp. 1–50.

[15] Musgrave F., "Methods for realistic landscape imaging," 01 1993.

[16] Finkenzeller D., "Detailed building facades," *IEEE Computer Graphics and Applications*, vol. 28, no. 3, pp. 58–66, 2008.

[17] Kelley A. D., Malin M. C., and Nielson G. M., "Terrain simulation using a model of stream erosion," in *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 1988, pp. 263–268.

[18] Prusinkiewicz P. and Hammel M., "A fractal model of mountains and rivers," in *Graphics Interface*, vol. 93. Canadian Information Processing Society, 1993, pp. 174–180.

[19] Teoh S. T., "River and coastal action in automatic terrain generation." in *CGVR*. Citeseer, 2008, pp. 3–9.

[20] Huijser R., Dobbe J., Bronsvoort W. F., and Bidarra R., "Procedural natural systems for game level design," in *2010 Brazilian Symposium on Games and Digital Entertainment*. IEEE, 2010, pp. 189–198.

[21] Galin E., Peytavie A., Maréchal N., and Guérin E., "Procedural generation of roads," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 429–438.

[22] Cura R., Perret J., and Paparoditis N., "Streetgen: In base city scale procedural generation of streets: road network, road surface and street objects," *arXiv preprint arXiv:1801.05741*, 2018.

[23] Freiknecht J. and Effelsberg W., "A survey on the procedural generation of virtual worlds," *Multimodal Technologies and Interaction*, vol. 1, no. 4, p. 27, 2017.

[24] Ebert D. S., Musgrave F. K., Peachey D., Perlin K., and Worley S., *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[25] Togelius J., Whitehead J., and Bidarra R., "Guest editorial: Procedural content generation in games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 03, pp. 169–171, 2011.

[26] Fournier A., Fussell D., and Carpenter L., "Computer rendering of stochastic models," *Communications of the ACM*, vol. 25, no. 6, pp. 371–384, 1982.

[27] Peachey D. R., "Solid texturing of complex surfaces," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 279–286.

[28] Zhang H., Qu D., Hou Y., Gao F., and Huang F., "Synthetic modeling method for large scale terrain based on hydrology," *IEEE Access*, vol. 4, pp. 6238–6249, 2016.

[29] Gamito M. N. and Musgrave F. K., "Procedural landscapes with overhangs," in *10th Portuguese Computer Graphics Meeting*, vol. 2, no. 3, 2001.

[30] De Carpentier G. J. and Bidarra R., "Interactive gpu-based procedural heightfield brushes," in *Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009, pp. 55–62.

[31] Smelik R. M., De Kraker K. J., Tutenel T., Bidarra R., and Groenewegen S. A., "A survey of procedural methods for terrain modelling," in *Proceedings of the*

CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS), vol. 2009. sn, 2009, pp. 25–34.

[32] Smelik R., Tutenel T., De Kraker K. J., and Bidarra R., "Integrating procedural generation and manual editing of virtual worlds," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010, pp. 1–8.

[33] Smelik R. M., Tutenel T., de Kraker K. J., and Bidarra R., "A declarative approach to procedural modeling of virtual worlds," *Computers & Graphics*, vol. 35, no. 2, pp. 352–363, 2011.

[34] Smelik R. M., Tutenel T., Bidarra R., and Benes B., "A survey on procedural modelling for virtual worlds," in *Computer Graphics Forum*, vol. 33, no. 6. Wiley Online Library, 2014, pp. 31–50.

[35] Emilien A., Vimont U., Cani M.-P., Poulin P., and Benes B., "Worldbrush: Interactive example-based synthesis of procedural virtual worlds," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–11, 2015.

[36] Dunn I. T., "Procedural generation and rendering of large-scale open-world environments," 2016.

[37] Watanabe N. and Igarashi T., "A sketching interface for terrain modeling," in *ACM SIGGRAPH 2004 Posters*, 2004, p. 73.

[38] Karpenko O. A. and Hughes J. F., "Smoothsketch: 3d free-form shapes from complex sketches," in *ACM SIGGRAPH 2006 Papers*, 2006, pp. 589–598.

[39] Zeleznik R. C., Herndon K. P., and Hughes J. F., "Sketch: An interface for sketching 3d scenes," in *ACM SIGGRAPH 2006 Courses*, 2006, pp. 9–es.

[40] Gain J., Marais P., and Straßer W., "Terrain sketching," in *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, 2009, pp. 31–38.

[41] Zhang J., Wang C., Li C., and Qin H., "Example-based rapid generation of vegetation on terrain via cnn-based distribution learning," *The Visual Computer*, vol. 35, no. 6, pp. 1181–1191, 2019.

[42] Šťava O., Beneš B., Brisbin M., and Křivánek J., "Interactive terrain modeling using hydraulic erosion," in *Proceedings of the 2008 acm siggraph/eurographics symposium on computer animation*, 2008, pp. 201–210.

[43] Peytavie A., Galin E., Grosjean J., and Mérillou S., "Arches: a framework for modeling complex terrains," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 457–467.

[44] Fournier A. and Reeves W. T., "A simple model of ocean waves," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 75–84.

[45] Miller G. S., "The definition and rendering of terrain maps," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 39–48.

[46] Peitgen H.-O., Jürgens H., Saupe D., and Feigenbaum M. J., *Chaos and fractals: new frontiers of science*. Springer, 1992, vol. 7.

[47] Perlin K., "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[48] Musgrave F. K., Kolb C. E., and Mace R. S., "The synthesis and rendering of eroded fractal terrains," *ACM Siggraph Computer Graphics*, vol. 23, no. 3, pp. 41–50, 1989.

[49] Wojtan C., Carlson M., Mucha P. J., and Turk G., "Animating corrosion and erosion." in *NPH*, 2007, pp. 15–22.

[50] Beneš B., Těšínskỳ V., Hornyš J., and Bhatia S. K., "Hydraulic erosion," *Computer Animation and Virtual Worlds*, vol. 17, no. 2, pp. 99–108, 2006.

[51] Anh N. H., Sourin A., and Aswani P., "Physically based hydraulic erosion simulation on graphics processing unit," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, 2007, pp. 257–264.

[52] Kelly G. and McCabe H., "A survey of procedural techniques for city generation," *ITB Journal*, vol. 14, no. 3, pp. 342–351, 2006.

[53] Chiba N., Muraoka K., and Fujita K., "An erosion model based on velocity fields for the visual simulation of mountain scenery," *The Journal of Visualization and Computer Animation*, vol. 9, no. 4, pp. 185–194, 1998.

[54] Hnaidi H., Guérin E., Akkouche S., Peytavie A., and Galin E., "Feature based terrain generation using diffusion equation," in *Computer Graphics Forum*, vol. 29, no. 7. Wiley Online Library, 2010, pp. 2179–2186.

[55] Amburn P., Grant E., and Whitted T., "Managing geometric complexity with enhanced procedural models," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986, pp. 189–195.

[56] Rusnell B., Mould D., and Eramian M., "Feature-rich distance-based terrain synthesis," *The Visual Computer*, vol. 25, no. 5, pp. 573–579, 2009.

[57] Zhou H., Sun J., Turk G., and Rehg J. M., "Terrain synthesis from digital elevation models," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 4, pp. 834–848, 2007.

[58] Nealen A., Igarashi T., Sorkine O., and Alexa M., "Fibermesh: designing freeform surfaces with 3d curves," in *ACM SIGGRAPH 2007 papers*, 2007, pp. 41–es.

[59] Ijiri T., Owada S., Okabe M., and Igarashi T., "Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints," in *ACM SIGGRAPH 2005 Papers*, 2005, pp. 720–726.

[60] Okabe M., Owada S., and Igarashi T., "Interactive design of botanical trees using freehand sketches and example-based editing," in *ACM SIGGRAPH 2006 Courses*, 2006, pp. 18–es.

[61] Cohen J. M., Hughes J. F., and Zeleznik R. C., "Harold: A world made of drawings," in *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, 2000, pp. 83–90.

[62] Becher M., Krone M., Reina G., and Ertl T., "Feature-based volumetric terrain generation and decoration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 2, pp. 1283–1296, 2017.

[63] Triebel R., Pfaff P., and Burgard W., "Multi-level surface maps for outdoor terrain mapping and loop closing," in *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2006, pp. 2276–2282.

[64] Schneider J., Boldte T., and Westermann R., "Real-time editing, synthesis, and rendering of infinite landscapes on gpus," in *Vision, modeling and visualization*, vol. 2006, 2006, pp. 145–152.

[65] Wang S., Ji H., Li P., Li H., and Zhan Y., "Growth diffusion-limited aggregation for basin fractal river network evolution model," *AIP Advances*, vol. 10, no. 7, p. 075317, 2020.

[66] Saunders R. L., "Terrainosaurus: realistic terrain synthesis using genetic algorithms," Ph.D. dissertation, Texas A&M University, 2007.

[67] Kamal K. R. and Uddin Y. S., "Parametrically controlled terrain generation," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, 2007, pp. 17–23.

[68] Belhadj F., "Terrain modeling: a constrained fractal model," in *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, 2007, pp. 197–204.

[69] Bernhardt A., Maximo A., Velho L., Hnaidi H., and Cani M.-P., "Real-time terrain modeling using cpu-gpu coupled computation," in *2011 24th SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE, 2011, pp. 64–71.

[70] Li Z., Zhu C., and Gold C., *Digital terrain modeling: principles and methodology*. CRC press, 2004.

[71] Krištof P., Beneš B., Křivánek J., and Šťava O., "Hydraulic erosion using smoothed particle hydrodynamics," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 219–228.

[72] Stachniak S. and Stuerzlinger W., "An algorithm for automated fractal terrain deformation," *Computer Graphics and Artificial Intelligence*, vol. 1, pp. 64–76, 2005.

[73] Douillard B., Underwood J., Melkumyan N., Singh S., Vasudevan S., Brunner C., and Quadros A., "Hybrid elevation maps: 3d surface models for segmentation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1532–1538.

[74] Baudrillard J., "Managing multi-valued horizons within a structural interpretation framework," Ph.D. dissertation, Université Grenoble Alpes, 2018.

[75] Greuter S., Parker J., Stewart N., and Leach G., "Real-time procedural generation ofpseudo infinite'cities," in *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, 2003, pp. 87–ff.

[76] Parish Y. I. and Müller P., "Procedural modeling of cities," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 301–308.

[77] Chen G., Esch G., Wonka P., Müller P., and Zhang E., "Interactive procedural street modeling," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 1–10.

[78] Sun J., Yu X., Baciu G., and Green M., "Template-based generation of road networks for virtual city modeling," in *Proceedings of the ACM symposium on Virtual reality software and technology*, 2002, pp. 33–40.

[79] van Rees E., "Esri cityengine 2013," *GeoInformatics*, vol. 17, no. 2, p. 6, 2014.

[80] Vanegas C. A., Aliaga D. G., Wonka P., Müller P., Waddell P., and Watson B., "Modelling the appearance and behaviour of urban spaces," in *Computer Graphics Forum*, vol. 29, no. 1. Wiley Online Library, 2010, pp. 25–42.

[81] Aliaga D. G., Vanegas C. A., and Benes B., "Interactive example-based urban layout synthesis," *ACM transactions on graphics (TOG)*, vol. 27, no. 5, pp. 1–10, 2008.

[82] Beneš J., Wilkie A., and Křivánek J., "Procedural modelling of urban road networks," in *Computer Graphics Forum*, vol. 33, no. 6. Wiley Online Library, 2014, pp. 132–142.

[83] Kelly G. and McCabe H., "Citygen: An interactive system for procedural city generation," in *Fifth International Conference on Game Design and Technology*, 2007, pp. 8–16.

[84] Glass K. R., Morkel C., and Bangay S. D., "Duplicating road patterns in south african informal settlements using procedural techniques," in *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, 2006, pp. 161–169.

[85] Lechner T., Watson B., and Wilensky U., "Procedural city modeling," in *In 1st Midwestern Graphics Conference*. Citeseer, 2003.

[86] Lechner T., Ren P., Watson B., Brozefski C., and Wilenski U., "Procedural modeling of urban land use," in *ACM SIGGRAPH 2006 Research posters*, 2006, pp. 135–es.

[87] Güner N. G., "A procedural 2d road network generation approach," Master's thesis, Middle East Technical University, 2018.

[88] Prusinkiewicz P. and Hanan J., *Lindenmayer systems, fractals, and plants*. Springer Science & Business Media, 2013, vol. 79.

[89] Song A. and Whitehead J., "Townsim: Agent-based city evolution for naturalistic road network generation," in *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019, pp. 1–9.

[90] Emilien A., Bernhardt A., Peytavie A., Cani M.-P., and Galin E., "Procedural generation of villages on arbitrary terrains," *The Visual Computer*, vol. 28, no. 6, pp. 809–818, 2012.

[91] Khan S., Phan B., Salay R., and Czarnecki K., "Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks." in *CVPR workshops*, 2019, pp. 88–96.

[92] Lechner T., Watson B., Ren P., Wilensky U., Tisue S., and Felsen M., "Procedural modeling of land use in cities," 2004.

[93] Gruen A. and Wang X., "Cybercity modeler, a tool for interactive 3-d city model generation," in *Photogrammetric Week*, vol. 99, 1999, pp. 317–327.

[94] Zhang C., Baltsavias E. P., and Gruen A., "Knowledge-based image analysis for 3d road reconstruction," ETH Zurich, Tech. Rep., 2001.

[95] Doucette P., Agouris P., and Stefanidis A., "Automated road extraction from high resolution multispectral imagery," *Photogrammetric Engineering & Remote Sensing*, vol. 70, no. 12, pp. 1405–1416, 2004.

[96] Andersson M. B. B., Gelotte F., Bjarne Graul Sagdahl J., Berger K., and Kvarnström S., "Procedural generation of a 3d terrain model based on a predefined road mesh," 2017.

[97] Tremblay J., Prakash A., Acuna D., Brophy M., Jampani V., Anil C., To T., Cameracci E., Boochoon S., and Birchfield S., "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*, 2018, pp. 969–977.

[98] Xu D., Wei C., Peng P., Xuan Q., and Guo H., "Ge-gan: A novel deep learning framework for road traffic state estimation," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102635, 2020.

[99] Aggarwal A., Mittal M., and Battineni G., "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021.

[100] "Assassins creed," https://www.ubisoft.com/en-us/game/assassins-creed.

[101] "The witcher," https://www.thewitcher.com/en.

[102] Singh S. P., Jain K., and Mandla V. R., "Virtual 3d city modeling: techniques and applications," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, pp. 73–91, 2013.

[103] Debevec P. E., Taylor C. J., and Malik J., "Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 11–20.

[104] Jepson W., Liggett R., and Friedman S., "Virtual modeling of urban environments," *Presence: Teleoperators & Virtual Environments*, vol. 5, no. 1, pp. 72–86, 1996.

[105] Dick A. R., Torr P. H., Ruffle S. J., and Cipolla R., "Combining single view recognition and multiple view stereo for architectural scenes," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1. IEEE, 2001, pp. 268–274.

[106] Teller S. *et al.*, "Mit city scanning project: Fully automated model acquisition in urban areas," 2001.

[107] Runions A., Lane B., and Prusinkiewicz P., "Modeling trees with a space colonization algorithm." *NPH*, vol. 7, no. 63-70, p. 6, 2007.

[108] Prusinkiewicz P., "Graphical applications of l-systems," in *Proceedings of graphics interface*, vol. 86, no. 86, 1986, pp. 247–253.

[109] ——, "A look at the visual modeling of plants using l-systems," in *German Conference on Bioinformatics*. Springer, 1996, pp. 11–29.

[110] Prusinkiewicz P., Hammel M., Hanan J., and Mech R., "L-systems: from the theory to visual models of plants," in *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, vol. 3. Citeseer, 1996, pp. 1–32.

[111] Měch R. and Prusinkiewicz P., "Visual models of plants interacting with their environment," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 397–410.

[112] Valencia-Rosado L. O., Guzman-Zavaleta Z. J., and Starostenko O., "Generation of synthetic elevation models and realistic surface images of river deltas and coastal terrains using cgans," *IEEE Access*, vol. 9, pp. 2975–2985, 2020.

[113] Aristid L., "Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs," *Journal of Theoretical Biology*, vol. 18, no. 3, pp. 300–315, 1968.

[114] Stiny G. N., *Pictorial and formal aspects of shape and shape grammars and aesthetic systems.* University of California, Los Angeles, 1975.

[115] Wonka P., Wimmer M., Sillion F., and Ribarsky W., "Instant architecture," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 669–677, 2003.

[116] Merrell P., "Example-based model synthesis," in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007, pp. 105–112.

[117] "Openstreetmap," https://www.openstreetmap.org/about.

[118] Kim J.-S., Kavak H., and Crooks A., "Procedural city generation beyond game development," *SIGSPATIAL Special*, vol. 10, no. 2, pp. 34–41, 2018.

[119] Groenewegen S. A., Smelik R. M., de Kraker K. J., and Bidarra R., "Procedural city layout generation based on urban land use models," *Short Paper Proceedings of Eurographics 2009*, 2009.

[120] Toriah M. S. A. K. S., "3d automatic building footprints generation," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2009.

[121] Prusinkiewicz P. and Lindenmayer A., *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[122] Lechner T., Watson B., Ren P., Wilensky U., Tisue S., and Felsen M., "Computer science department," 2004.

[123] Finkenzeller D. and Bender J., "Semantic representation of complex building structures," in *Computer Graphics and Visualization (CGV 2008)-IADIS Multi Conference on Computer Science and Information Systems, Amsterdam, The Netherlands*, 2008.

[124] Legakis J., Dorsey J., and Gortler S., "Feature-based cellular texturing for architectural models," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 309–316.

[125] Wei L.-Y. and Levoy M., "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 479–488.

[126] Müller P., Zeng G., Wonka P., and Van Gool L., "Image-based procedural modeling of facades," *ACM Trans. Graph.*, vol. 26, no. 3, p. 85, 2007.

[127] Weber B., Müller P., Wonka P., and Gross M., "Interactive geometric simulation of 4d cities," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 481–492.

[128] Roumpani F., "Procedural cities as active simulators for planning," *Urban Planning*, vol. 7, no. 2, pp. 321–329, 2022.

[129] Badwi I. M., Ellaithy H. M., and Youssef H. E., "3d-gis parametric modelling for virtual urban simulation using cityengine," *Annals of GIS*, pp. 1–17, 2022.

[130] Zhang X., Zhu Q., and Wang J., "3d city models based spatial analysis to urban design," *Geographic Information Sciences*, vol. 10, no. 1, pp. 82–86, 2004.

[131] Cootes T., Baldock E., and Graham J., "An introduction to active shape models," *Image processing and analysis*, vol. 328, pp. 223–248, 2000.

[132] Biljecki F., Ledoux H., and Stoter J., "Generating 3d city models without elevation data," *Computers, Environment and Urban Systems*, vol. 64, pp. 1–18, 2017.

[133] Müller M., Casser V., Lahoud J., Smith N., and Ghanem B., "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 902–919, 2018.

[134] Deussen O., Hanrahan P., Lintermann B., Měch R., Pharr M., and Prusinkiewicz P., "Realistic modeling and rendering of plant ecosystems," in *Proceedings of*

the 25th annual conference on Computer graphics and interactive techniques, 1998, pp. 275–286.

[135] Livny Y., Pirk S., Cheng Z., Yan F., Deussen O., Cohen-Or D., and Chen B., "Texture-lobes for tree modelling," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, pp. 1–10, 2011.

[136] Bruneton E. and Neyret F., "Real-time realistic rendering and lighting of forests," in *Computer graphics forum*, vol. 31, no. 2pt1. Wiley Online Library, 2012, pp. 373–382.

[137] Gumbau J., Chover M., Remolar I., and Rebollo C., "View-dependent pruning for real-time rendering of trees," *Computers & Graphics*, vol. 35, no. 2, pp. 364–374, 2011.

[138] Neubert B., Pirk S., Deussen O., and Dachsbacher C., "Improved model-and view-dependent pruning of large botanical scenes," in *Computer Graphics Forum*, vol. 30, no. 6. Wiley Online Library, 2011, pp. 1708–1718.

[139] Honda H., "Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body," *Journal of theoretical biology*, vol. 31, no. 2, pp. 331–338, 1971.

[140] Aono M. and Kunii T. L., "Botanical tree image generation," *IEEE computer graphics and applications*, vol. 4, no. 5, pp. 10–34, 1984.

[141] Oppenheimer P. E., "Real time design and animation of fractal plants and trees," *ACM SiGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 55–64, 1986.

[142] Lintermann B. and Deussen O., "Interactive modeling of plants," *IEEE Computer Graphics and Applications*, vol. 19, no. 1, pp. 56–65, 1999.

[143] Weber J. and Penn J., "Creation and rendering of realistic trees," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 119–128.

[144] Reeves W. T. and Blau R., "Approximate and probabilistic algorithms for shading and rendering structured particle systems," *ACM siggraph computer graphics*, vol. 19, no. 3, pp. 313–322, 1985.

[145] Smith A. R., "Plants, fractals, and formal languages," *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 1–10, 1984.

[146] Greene N., "Voxel space automata: Modeling with stochastic growth processes in voxel space," in *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, 1989, pp. 175–184.

[147] Xie K., Yan F., Sharf A., Deussen O., Huang H., and Chen B., "Tree modeling with real tree-parts examples," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 12, pp. 2608–2618, 2015.

[148] Zhang X., Bao G., Meng W., Jaeger M., Li H., Deussen O., and Chen B., "Tree branch level of detail models for forest navigation," in *Computer Graphics Forum*, vol. 36, no. 8. Wiley Online Library, 2017, pp. 402–417.

[149] Gain J., Long H., Cordonnier G., and Cani M.-P., "Ecobrush: Interactive control of visually consistent large-scale ecosystems," in *Computer Graphics Forum*, vol. 36, no. 2. Wiley Online Library, 2017, pp. 63–73.

[150] "3d vegetation modeling and middleware," http://www.speedtree.com/.

[151] "3d plants: Xfrogplants," http://www.xfrog.com/.

[152] Livny Y., Yan F., Olson M., Chen B., Zhang H., and El-Sana J., "Automatic reconstruction of tree skeletal structures from point clouds," in *ACM SIGGRAPH Asia 2010 papers*, 2010, pp. 1–8.

[153] Prusinkiewicz P., Lindenmayer A., and Hanan J., "Development models of herbaceous plants for computer imagery purposes," in *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 1988, pp. 141–150.

[154] Prusinkiewicz P., "Simulation modeling of plants and plant ecosystems," *Communications of the ACM*, vol. 43, no. 7, pp. 84–93, 2000.

[155] Prusinkiewicz P., Hammel M. S., and Mjolsness E., "Animation of plant development," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 1993, pp. 351–360.

[156] Bradbury G. A., Subr K., Koniaris C., Mitchell K., and Weyrich T., "Guided ecological simulation for artistic editing of plant distributions in natural scenes," *Journal of Computer Graphics Techniques Vol*, vol. 4, no. 4, 2015.

[157] Okabe M., Owada S., and Igarashi T., "Interactive design of botanical trees using freehand sketches and example-based editing," in *SIGGRAPH '07*, 2007.

[158] Li B., Kałużny J., Klein J., Michels D. L., Pałubicki W., Benes B., and Pirk S., "Learning to reconstruct botanical trees from single images," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–15, 2021.

[159] Bradley D., Nowrouzezahrai D., and Beardsley P., "Image-based reconstruction and synthesis of dense foliage," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–10, 2013.

[160] Neubert B., Franken T., and Deussen O., "Approximate image-based tree-modeling using particle flows," in *ACM SIGGRAPH 2007 papers*, 2007, pp. 88–es.

[161] Argudo O., Chica A., and Andujar C., "Single-picture reconstruction and rendering of trees for plausible vegetation synthesis," *Computers & Graphics*, vol. 57, pp. 55–67, 2016.

[162] Li C., Deussen O., Song Y.-Z., Willis P., and Hall P., "Modeling and generating moving trees from video," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, pp. 1–12, 2011.

[163] Xu H., Gossett N., and Chen B., "Knowledge and heuristic-based modeling of laser-scanned trees," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 4, pp. 19–es, 2007.

[164] Reche-Martinez A., Martin I., and Drettakis G., "Volumetric reconstruction and interactive rendering of trees from photographs," in *ACM SIGGRAPH 2004 Papers*, 2004, pp. 720–727.

[165] Palubicki W., Horel K., Longay S., Runions A., Lane B., Měch R., and Prusinkiewicz P., "Self-organizing tree models for image synthesis," *ACM Transactions On Graphics (TOG)*, vol. 28, no. 3, pp. 1–10, 2009.

[166] Pirk S., Stava O., Kratt J., Said M. A. M., Neubert B., Měch R., Benes B., and Deussen O., "Plastic trees: interactive self-adapting botanical tree models," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–10, 2012.

[167] Stava O., Pirk S., Kratt J., Chen B., Měch R., Deussen O., and Benes B., "Inverse procedural modelling of trees," in *Computer Graphics Forum*, vol. 33, no. 6. Wiley Online Library, 2014, pp. 118–131.

[168] Hädrich T., Benes B., Deussen O., and Pirk S., "Interactive modeling and authoring of climbing plants," in *Computer Graphics Forum*, vol. 36, no. 2. Wiley Online Library, 2017, pp. 49–61.

[169] Belhadj F. and Audibert P., "Modeling landscapes with ridges and rivers: bottom up approach," in *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, 2005, pp. 447–450.

[170] Derzapf E., Ganster B., Guthe M., and Klein R., "River networks for instant procedural planets," in *Computer Graphics Forum*, vol. 30, no. 7. Wiley Online Library, 2011, pp. 2031–2040.

[171] Génevaux J.-D., Galin É., Guérin E., Peytavie A., and Benes B., "Terrain generation using procedural models based on hydrology," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–13, 2013.

[172] Zhang L., Dai B., Wang G., Li T., and Wang H., "The quantization of river network morphology based on the tokunaga network," *Science in China Series D: Earth Sciences*, vol. 52, no. 11, pp. 1724–1731, 2009.

[173] Emilien A., Poulin P., Cani M.-P., and Vimont U., "Interactive procedural modelling of coherent waterfall scenes," in *Computer Graphics Forum*, vol. 34, no. 6. Wiley Online Library, 2015, pp. 22–35.

[174] Samavati F. and Runions A., "Interactive 3d content modeling for digital earth," *The Visual Computer*, vol. 32, no. 10, pp. 1293–1309, 2016.

[175] Cordonnier G., Braun J., Cani M.-P., Benes B., Galin E., Peytavie A., and Guérin E., "Large scale terrain generation from tectonic uplift and fluvial erosion," in *Computer Graphics Forum*, vol. 35, no. 2. Wiley Online Library, 2016, pp. 165–175.

[176] Han J., Zhou K., Wei L.-Y., Gong M., Bao H., Zhang X., and Guo B., "Fast example-based surface texture synthesis via discrete optimization," *The Visual Computer*, vol. 22, no. 9, pp. 918–925, 2006.

[177] Nishida G., Garcia-Dorado I., and Aliaga D. G., "Example-driven procedural urban roads," in *Computer Graphics Forum*, vol. 35, no. 6. Wiley Online Library, 2016, pp. 5–17.

[178] Guerrero P., Jeschke S., Wimmer M., and Wonka P., "Learning shape placements by example," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–13, 2015.

[179] Nishida G., Garcia-Dorado I., Aliaga D. G., Benes B., and Bousseau A., "Interactive sketching of urban procedural models," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[180] Guérin É., Digne J., Galin E., Peytavie A., Wolf C., Benes B., and Martinez B., "Interactive example-based terrain authoring with conditional generative adversarial networks," *Acm Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–13, 2017.

[181] Valencia-Rosado L. O., Guzman-Zavaleta Z. J., and Starostenko O., "A modular generative approach for realistic river deltas: When l-systems and cgans meet," *IEEE Access*, vol. 10, pp. 5753–5767, 2022.

[182] Qiu W., Zhong F., Zhang Y., Qiao S., Xiao Z., Kim T. S., and Wang Y., "Unrealcv: Virtual worlds for computer vision," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1221–1224.

[183] Fang Q., Xu X., Wang X., and Zeng Y., "Target-driven visual navigation in indoor scenes using reinforcement learning and imitation learning," *CAAI Transactions on Intelligence Technology*, vol. 7, no. 2, pp. 167–176, 2022.

[184] Zhang Y., Qiu W., Chen Q., Hu X., and Yuille A., "Unrealstereo: Controlling hazardous factors to analyze stereo vision," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 228–237.

[185] Ros G., Sellart L., Materzynska J., Vazquez D., and Lopez A. M., "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[186] Novotnỳ M., Lacko J., and Samuelčík M., "Applications of multi-touch augmented reality system in education and presentation of virtual heritage," *Procedia Computer Science*, vol. 25, pp. 231–235, 2013.

[187] Nishizawa H., Shimada K., Ohno W., and Yoshioka T., "Increasing reality and educational merits of a virtual game," *Procedia Computer Science*, vol. 25, pp. 32–40, 2013.

[188] González M. A., Santos B. S. N., Vargas A. R., Martín-Gutiérrez J., and Orihuela A. R., "Virtual worlds. opportunities and challenges in the 21st century," *Procedia Computer Science*, vol. 25, pp. 330–337, 2013.

[189] Studios M., "Minecraft," https://www.minecraft.net/, 2009, accessed: 2022-10-10.

[190] "Second Life," https://secondlife.com/, 2003, accessed: 2022-10-10.

[191] de Madrid U. P., "Laboratorios Virtuales," https://serviciosgate.upm.es/gate/laboratorios-virtuales/, 2021, accessed: 2022-10-10.

[192] Jabbar A., Li X., and Omar B., "A survey on generative adversarial networks: Variants, applications, and training," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–49, 2021.

[193] Shu D. W., Park S. W., and Kwon J., "3d point cloud generative adversarial network based on tree structured graph convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3859–3868.

[194] Kolbe T. H., "Representing and exchanging 3d city models with citygml," in *3D geo-information sciences*. Springer, 2009, pp. 15–31.

[195] "Simcity," https://en.wikipedia.org/wiki/SimCity.

[196] Dorsey J., Edelman A., Jensen H. W., Legakis J., and Pedersen H. K., "Modeling and rendering of weathered stone," in *ACM SIGGRAPH 2005 Courses*, 2005, pp. 4–es.

[197] Chen Y., Tong X., Guo B., and Shum H.-Y., "Visual simulation of weathering by y-ton tracing," Jul. 7 2009, uS Patent 7,557,807.

[198] Greuter S., Parker J., Stewart N., and Leach G., "Undiscovered worlds–towards a framework for real-time procedural world generation," in *Fifth International Digital Arts and Culture Conference, Melbourne, Australia*, vol. 5, 2003, p. 5.

[199] Vanegas C. A., Aliaga D. G., Benes B., and Waddell P. A., "Interactive design of urban spaces using geometrical and behavioral modeling," *ACM transactions on graphics (TOG)*, vol. 28, no. 5, pp. 1–10, 2009.

[200] Vanegas C. A., Kelly T., Weber B., Halatsch J., Aliaga D. G., and Müller P., "Procedural generation of parcels in urban modeling," in *Computer graphics forum*, vol. 31, no. 2pt3. Wiley Online Library, 2012, pp. 681–690.

[201] Niese T., Pirk S., Albrecht M., Benes B., and Deussen O., "Procedural urban forestry," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 2, pp. 1–18, 2022.

[202] Kelly T., Femiani J., Wonka P., and Mitra N. J., "Bigsur: large-scale structured urban reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 6, 2017.

[203] Gao Q., Shen X., and Niu W., "Large-scale synthetic urban dataset for aerial scene understanding," *IEEE Access*, vol. 8, pp. 42 131–42 140, 2020.

[204] "The most powerful real-time 3d creation tool-unreal engine," https://www.unrealengine.com/en-US.

[205] Shen B., Li B., and Scheirer W. J., "Automatic virtual 3d city generation for synthetic data collection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 161–170.

[206] Takase Y., Sho N., Sone A., and Shimiya K., "Automatic generation of 3d city models and related applications," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. 5, 2003.

[207] "3d design software | 3d modeling on the web | sketchup," https://www.sketchup.com/.

[208] "Rhino - rhinoceros 3d," https://www.rhino3d.com/.

[209] "Autodesk | 3d design, engineering & construction software," https://www.autodesk.com/.

[210] "Cgtrader," https://www.cgtrader.com/.