Contents

Table of Contents **1**

1 Automatic generation of coral reef islands **3**

1.1 Introduction 4

1.2 State of the art 7

1.3 Description of our method 21

1.4 Procedural island generation 21

1.5 Data generation 34

1.6 Learning-based generation 38

1.7 Results 40

1.8 Conclusion and future works 42

Chapter 1

Automatic generation of coral reef islands

Figure 1.1: *Given a free hand-drawn sketch of the different regions of an island (bottom inset of each example), our method generates a corresponding height field using neural networks. Subsidence (gradual sinking of land) is controlled by the user in the luminosity channel of the input. These examples show the gradual subsidence of the same island over time.*

Abstract

In this chapter, we propose a procedural method for generating single volcanic islands with coral reefs based on user sketches from two projections commonly used in geological and remote sensing domains: a top view defining the island's shape, and a profile view, establishing its elevation. We add a user-defined wind field that deforms the island to mimic long-term effect of wind and wave, providing finer user control over the island's shape. Next, We model the coral growth on the island following biological observations. This results in a terrain height field of the island and its surrounding reef. Our method creates a large variety of coral reef island models which we used to train a conditional Generative Adversarial Network (cGAN). By applying data augmentation, the cGAN enhances the variety in the generated islands, providing users with greater freedom and intuitive controls over the shape and structure of the final output.

Contents

1.1 Introduction 4
1.2 State of the art 7
1.2.1 Coral reef island formation theories 7
1.2.2 Traditional terrain generation methods 10
1.2.3 Sketch-based terrain modelling 13
1.2.4 Deep learning 17
1.3 Description of our method 21
1.4 Procedural island generation 21
1.4.1 Initial height field generation 23
1.4.2 Coral reef modelling 30
1.5 Data generation 34
1.5.1 Data augmentation 37
1.6 Learning-based generation 38
1.7 Results 40
1.7.1 Advantages of the approach 41
1.7.2 Limitations 42
1.8 Conclusion and future works 42

↑ **Back to summary**

1.1 Introduction

Figure 1.2: *Kauai Island, Tuvuca Island, Mascarene Island, and Cocos Island (from left to right) are islands formed through an identical geological process; however, their shapes vary greatly, from a full island with fringing coral reef (leftmost) to a complete atoll (rightmost).*

The scarcity of high-resolution data, the need for volumetric and multi-scale representations, and the strong influence of biological and hydrodynamic processes present unique modelling challenges for underwater landscapes. Among these environments, coral reef islands are of particular interest and complexity. They result from a long-term interplay of volcanic activity, coral growth, erosion, and subsidence, producing highly distinctive morphologies that procedural techniques must capture if they are to generate convincing seascapes. Incorporating numerous processes is tedious to control

for a user-centered modelling application. We propose a simplified interface to focus on the overall large-scale modelling of coral reef islands using sketching methods.

To better understand these morphologies, we begin by recalling the classical geological explanation of their formation, as proposed by Charles Darwin in the 19th century [DJ42]. His subsidence theory remains a foundational description of how volcanic islands gradually transform into barrier reefs and atolls (such as the islands presented in Figure 1.2), and it provides the conceptual backdrop for our generative model.

According to this theory, these islands begin as volcanic landmasses, created when magma from the Earth's mantle erupts through the ocean floor and builds up layers of volcanic rock, eventually rising above sea level (Figure 1.3). In tropical waters, these volcanic islands create ideal conditions for coral reefs to develop. Corals, thriving in the shallow, sunlit waters around the island, initially form fringing reefs attached to the island's coastline.

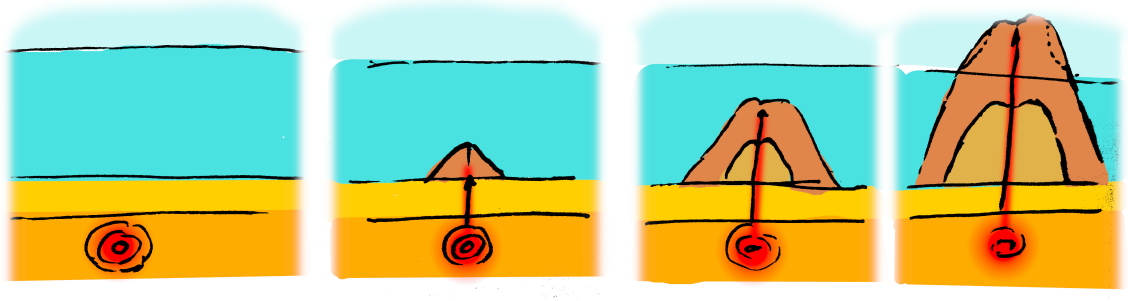


Figure 1.3: *Volcanic islands rise above hotspots. The rise of magma from the hotspot forms a seamount. Consecutive eruptions from the volcano grow the seamount until the peak emerges above sea level. The ground above and close to sea level is affected by hydraulic, aeolian, and coastal erosion.*

As time passes, the volcanic island undergoes subsidence, a slow sinking process caused by the cooling and contraction of the Earth's crust beneath the island. In response, corals continue to grow upwards to remain within the photic zone, where sunlight supports their survival. This upward growth combined with subsidence leads to the formation of barrier reefs, that become separated from the island by a lagoon as it sinks further.

Eventually, the volcanic island may submerge completely, leaving only the coral structure visible above water. This process results in an atoll: a ring-shaped reef encircling a central lagoon. Over geological time, the island's shape evolves from a prominent volcanic peak into a coral-dominated reef system shaped by subsidence, coral growth, and erosion (Figure 1.4).

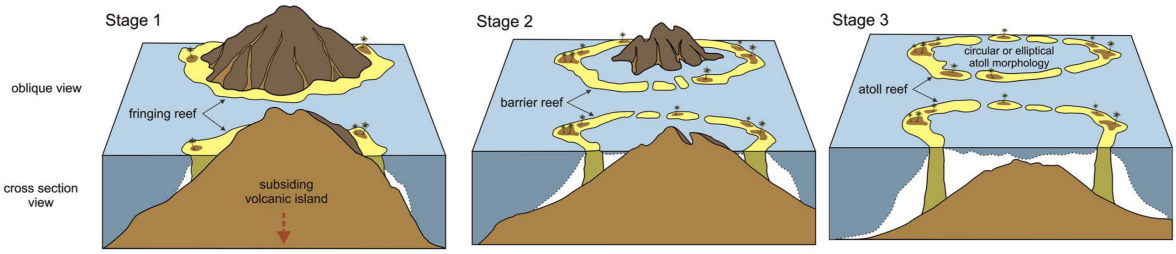


Figure 1.4: *Coral colonies grow near sea level, forming a fringing reef (Stage 1). The island sinks slowly (subsidence); the coral reef continues its growth to keep living coral colonies in the photic zone. The sinking island increases the size of the lagoon, forming a barrier reef island (Stage 2). When the peak of the island is below the surface of the lagoon, an atoll is formed (Stage 3) [TG13].*

Simulating the formation of coral reef islands presents significant challenges due to the complex interplay of geological, environmental, and biological factors [Hop14]. One major difficulty lies in capturing the long-term subsidence of volcanic islands, occuring over millions of years, and the concurrent upward growth of coral reefs that rely on environmental conditions (e.g., water depth, temperature, sunlight, ...). This combination of slow geological processes and dynamic biological growth is inherently challenging to replicate in a computational model.

Additionally, the biological aspects of coral growth are inherently dependant on environmental factors. Coral reefs grow only within a specific range of water depth and sunlight, and their growth patterns are influenced by reef ecosystem's health and resources availability. Accurately modelling these biological dependencies in a procedural system is challenging, as these factors are numerous and difficult to generalise. Moreover, the scarcity of data available obstructs global understanding of these biomes. In a recent high-resolution mapping of shallow coral reefs [LMK*24], researchers estimated the total surface area of this biome to cover less than 0.7% of Earth's area, and more specifically that coral habitat represents less than 0.2%.

Existing terrain generation methods, such as Perlin noise-based algorithms or uplift-erosion models, are often ill-suited for these multi-disciplinary processes. Although they generate natural-looking landscapes (such as alpine landscapes, representing about a quarter of land area [KSBZ14]), they do not account for the unique geological and biological interactions critical for coral reef island development, resulting in a lack of coherence. Capturing these dynamics, while also providing user control during the modelling of a terrain, requires a balance between realism and procedural flexibility, allowing for both accurate computationally expensive simulation of natural processes and intuitive user control in interactive time.

The formation of these islands involves processes at multiple scales, from the growth patterns of coral colonies to large-scale sediment transport, which are difficult to simulate directly. As a result, purely procedural or physics-based simulations fail to produce convincing or diverse coral reef island landscapes. On the other hand, deep learning methods are inoperable due to the extremely small amount of data, and the scarcity of high-resolution DEMs of these regions.

Despite advances in terrain generation, current methods struggle to support user-controlled design of specific island shapes and achieving realism without real data. Coral reef islands exemplify this gap: we lack datasets to directly train deep models, and purely procedural methods require expert tuning to mimic their features.

To address these challenges, we use procedural generation as an initial step in our approach. Procedural generation employs algorithmic rules to synthesise terrain features, allowing us to encode basic patterns of coral reef island formation. However, such algorithms are inherently rigid: they are tailored to a narrow family of shapes (in our case, radially organised and centred islands) and often rely on mathematical controls that are unintuitive for non-expert users. In our work, we use a procedural model not as the final solution, but as a means to efficiently create a large and diverse set of training examples for a learning-based model. Specifically, by adjusting procedural parameters, the procedural pipeline produces varied island scenarios. Each synthetic example is represented by a detailed terrain height field and a corresponding semantic label map that marks different regions, providing structured input-output pairs for the learning stage as presented in Figure 1.1.

We then train and deploy a conditional Generative Adversarial Network (cGAN) as the core of our approach. A cGAN is a type of deep learning model that learns to generate realistic data based on an input condition or context. In our case, the cGAN takes as input the semantic label map of an island (a label layout indicating regions such as ocean, reef, beach, and mountain) generated by the procedural step and learns to produce a plausible island height field that matches this layout. By training on the many examples from the procedural generator, the cGAN captures subtle terrain features and variations unique to coral reef islands, going beyond what hard-coded procedural rules can achieve thanks to the application of data augmentation.

Once the cGAN is trained on a sufficiently large and varied set of synthetic islands, it can be used on its

own to generate new island terrains. At this stage, our procedural generation module, only necessary to provide training data, is not required for creating new islands. Instead, a user can provide a semantic map using digital drawing or another simple algorithm, and the cGAN will generate a plausible island terrain accordingly. In short, our pipeline leverages procedural modelling to create a training dataset, and then relies on the learned cGAN model for the final generation of coral reef islands. In this way, we overcome the scarcity of real data by procedurally generating training examples, and overcome the rigidity of procedural rules through cGAN-based learning.

In summary, the key contributions of this chapter are:

- a novel sketch-based procedural algorithm for shaping island terrains from top and profile views,
- the use of a deep learning model trained on synthetic data derived from procedural rules, acting as an abstraction layer that hides underlying complexity,
- a demonstration that the cGAN approach tolerates imprecise, low-detail user input sketches, broadening usability, without the need for cutting-edge network architectures,
- and an insight that procedural generation remains essential to produce training data in data-sparse domains, illustrated by the example of coral reef islands.

These contributions collectively show a pathway for blending user-driven design with learning-based generation in terrain modelling.

1.2 State of the art

Procedural terrain generation and sketch-based modelling have each seen significant advances over the past decades, yet neither alone fully addresses the particular challenges of coral reef island synthesis. On the one hand, classic noise-based and physically driven methods (Perlin noise, hydraulic erosion, uplift-erosion models) excel at producing broad, natural-looking landscapes but lack the biologically inspired reef geometries and dynamic island evolution governed by subsidence and coral growth. On the other hand, sketch-based tools give users intuitive control over silhouettes and terrain profiles, but typically require expert parameter tuning to achieve realism and do not model long-term geological or ecological processes. More recently, deep learning, especially GANs and cGANs, has emerged as a powerful way to learn terrain features and geological rules from data, yet it relies on large, labelled datasets that are scarce for coral reef islands.

In this section, we first review the key geological theories that explain reef formation, then examine traditional procedural terrain generators, followed by an overview of sketch-based terrain editing frameworks, and finally discuss recent advances in GAN-based terrain synthesis.

1.2.1 Coral reef island formation theories

Since Darwin first proposed his subsidence theory, presented in Section 1.1, geologists and biologists have debated how exactly coral reefs have formed around land masses. In this section we will present three major alternatives to Darwin's theory, before explaining why Darwin's unified subsidence model provides the most direct foundation for our procedural generation.

Murray's stand-still theory [Mur80]: Murray argued that reefs could develop on stable, non-sinking platforms, with coral growth keeping pace with modest sea-level changes rather than underlying land subsidence. He proposed that as long as water depth remained within the photic zone, reefs would accrete upward solely in response to environmental sea-level fluctuations, and continuously seaward, forming a gradual structure from a fringing reef to barrier reef (Figure 1.5). Atolls in this theory are mainly created by the degradation of the middle island through aeolian and coastal erosion until it becomes completely submeged. Murray's emphasis on environmental stability and gradual sea-level change was supported by early observations of island terraces and reef growth patterns.

However, later drilling and seismic data revealed volcanic foundations buried beneath reef limestones on many atolls, which are evidence of true subsidence that Murray's theory cannot explain.

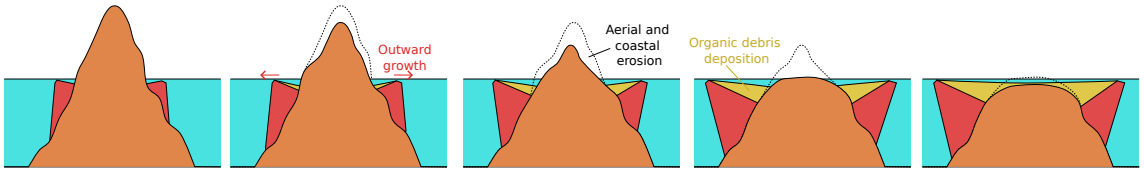


Figure 1.5: *In Murray's theory, with the island fixed in place but reducing in size due to erosion, reefs simply remains at sea level and grows outwards (producing fringing reefs, then barrier reefs, and eventually atolls) without any subsidence.*

Daly's glacial-control theory [Dal15]: Daly shifted the focus from steady subsidence to global sea-level oscillations driven by glacial–interglacial cycles. He argued that reefs thrive during high sea levels in interglacials but are exposed and eroded during glacial lowstands. As sea levels rise again, the wave-cut benches left behind provide ideal habitats for new coral colonies, which grow upward with the water level and can develop into barrier reefs or atolls once sea level overtops the island (Figure 1.6). He support this model by pointing to multiple reef terraces at different elevations and well-documented 100 m sea-level drops during ice ages as support for this cycle-driven reef development. The plausibility of Daly's model lies in its direct link with climatic data and the clear geomorphic signatures of past sea-level stands. Yet, core samples often show uninterrupted reef accretion atop subsiding volcanic bases, indicating that glacial cycles alone cannot account for continuous reef "keep-up" growth.

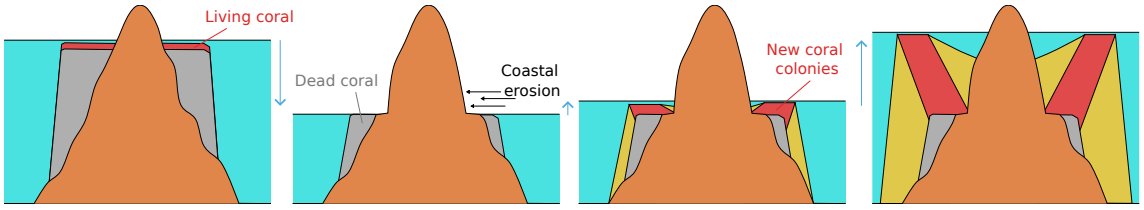


Figure 1.6: *In Daly's theory, repeated sea-level drops expose the reef to wave planation and erosion into a flat bench, and subsequent high stands see new coral rims accrete on that outer terrace to form a lagoon-separated barrier reef.*

Droxler's karstification theory [DJ21]: In contrast to models based on volcanic subsidence, Droxler et Jorry attribute atoll formation to repeated karst dissolution of a broad carbonate platform during low-sea-level glacials, followed by renewed coral accretion in interglacials (Figure 1.7). High-resolution bathymetry and drill cores reveal karst-etched depressions beneath certain atoll crests, supporting this cyclic exposure-dissolution mechanism. While compelling for extensive carbonate shelves, this theory hinges on pre-existing flat platforms and meteoric water circulation (water originating from precipitation), which are conditions uncommon on volcanic seamounts. As a result, Droxler's model explains atoll rings on continental carbonate foundations but does not readily apply to volcanic island reef systems.

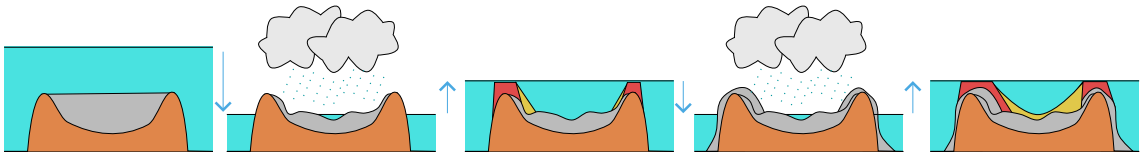


Figure 1.7: *A broad carbonate platform (grey) is exposed and karstified during low-sea-level glacials, then drowns and is rimmed by reef growth at its outer edge during high stands, yielding an atoll-style rings.*

Of all the competing hypotheses, Charles Darwin's subsidence model offers several compelling advantages for our procedural island generation due to its simplicity, historical significance, and effectiveness [TMNM97]. This theory provides a straightforward framework outlining a clear progression from fringing reefs to barrier reefs to atolls as a volcanic island subsides, which can be easily modelled, making it practical for generating plausible island landscapes. This simplicity not only ensures predictability in simulation outcomes but also reduces computational demands, which is particularly beneficial for interactive edition applications. Additionally, Darwin's model provides a foundational basis upon which more complex phenomena (sea-level changes and climatic effects), considered as secondary factors in the geological formation of islands, could be layered. This allows us to start with a basic model and, in the future, add complexity as needed, offering flexibility in simulation design.

In our approach, we translate the core principles of Darwin's theory into a procedural generation model, emulating the gradual sinking of volcanic islands while coral reefs grow to keep pace with changing sea levels. This allows us to realistically model the progressive transformation of islands from volcanic landmasses to coral-dominated atolls. By capturing this interplay, we procedurally generate a wide variety of island structures that reflect real-world geological processes.

In generating synthetic coral reef islands, we adopt a set of simplifying assumptions that are interpreted by the formation process from Darwin's theory and field observations:

Radial symmetry and radial arrangement of features: While volcanic islands do not have a perfect circular shape and reefs are not exact rings, they grow outward in roughly concentric belts (looking at the island in Figure 1.8, we see the typical layout mountain-beach-lagoon-reef-ocean). Anchoring our sketch-to-terrain engine on this radially organised structure offers two main benefits to the user: a single intuitive control over the overall island shape and while keeping distance computations simple and efficient.

Uniform profile shape: Although real islands have ridges, valleys, local bumps and many other features, we simplify the model to a single elevation curve from centre to edge, making the profile-view sketch a one-dimensional drawing, enabling users to modify elevation with a single stroke.

Figure 1.8: *Aerial view of Tuvuca Island, Lau Archipelago, Fiji.*

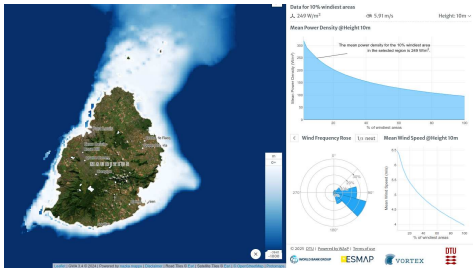


Figure 1.9: *Mauritius is deformed. We see a deep slope on the west side while the east side has a gentle slope. The wind rose, showing in which direction the mean wind velocity is pointing, is correlated with this deformation.*

Subsidence and coral "keep up" growth: Actual reefs respond to light, nutrients, and biology, but the dominant effect is that corals grow vertically to stay near the photic zone. We decouple coral growth from subsidence and simply keep reef heights in a fixed band beneath the surface, capturing Darwin's core insight using minimal parameters.

Wind and wave deformation: To represent coastal erosion resulting from sediment transport and complex hydrodynamics [TG13], we simplify the model by using a vector field painted by the user representing wind and wave directions, deforming the global island (Figure 1.9 suggest that island outlines are influenced by the average wind direction). This vector field allows radial symmetry to be broken in a controllable way, without relying on a full erosion simulator.

Independence of islands: While archipelagos share currents, sediments, and flood each other's lagoons, simulating multi-island interactions is computationally expensive and hard to control. We treat each island as an isolated entity to generate multi-island scenes by using a simple blending of individual islands, keeping our method fast and predictable.

These assumptions, grounded in theories of coral reef island formation, provide a practical foundation for procedural modelling. While they simplify the full complexity of geological and biological processes, they capture the essential dynamics needed to produce plausible island structures. Simultaneously, they ensure our system remains intuitive, controllable, and computationally efficient, enabling the generation of diverse and plausible islands suitable for interactive design.

1.2.2 Traditional terrain generation methods

Procedural generation of terrain has been a well-researched area in computer graphics and simulations, where the goal is to create large, realistic landscapes with minimal manual input. Various methods have been developed over the years to generate terrains automatically, from noise-based approaches to physically based erosion simulations, sketch-driven methods, and more recently, deep learning techniques.

However, coral reef islands present unique challenges due to the combination of long-term geological processes (mainly subsidence and coral reef growth) and environmental interactions (i.e. erosion caused by wind and waves). In this section, we review the key techniques proposed for terrain generation, highlighting their limitations for coral reef island formation, and positioning our work as an approach that addresses these challenges.

Noise-based terrain generation

Noise-based procedural generation remains one of the most widely used techniques for creating natural-looking terrains. Perlin noise [Per85], Simplex noise [Per01], and the Diamond-square algorithm [FFC82] are foundational algorithms that generate pseudo-random yet continuous variations across a grid, producing terrain features that resemble organic landscapes. These techniques have been widely adopted in computer graphics and game development due to their efficiency and visual appeal.

Beyond basic noise functions, advanced techniques such as fractal Brownian motion (fBm) and

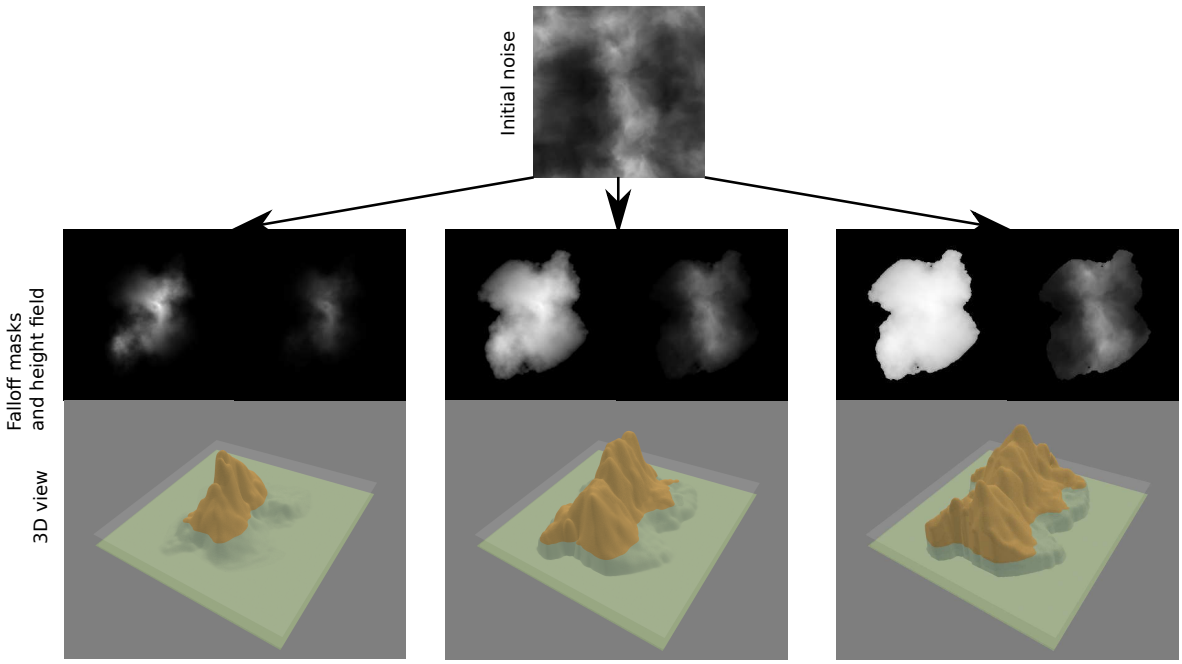


Figure 1.10: *Three different results of an island generated from noise functions. In each case, the initial height field is the same, computed through flow noise. The falloff masks are also generated with a combination of fBm noise mitigated by the Euclidean distance from the image centre, warp noise and gamma correction. The only parameter modified for each example is the gamma correction. The results are very different, and hardly controllable. It is also difficult to represent lagoons and reefs using this method.*

multifractal noise have been introduced to add finer-scale variation and detail [MKM89, EPW*03]. fBm combines multiple layers, or "octaves," of noise at different frequencies and amplitudes, producing terrains that exhibit realistic and varied features. The combination of noise with domain warping and signal processing techniques has been explored in depth in procedural modelling literature [RLL*10], enabling further control over visual complexity and terrain realism.

Noise functions are often paired with falloff maps to produce island-like terrains, where elevation gradually decreases towards the edges of the domain, mimicking coastlines and basic island shapes (see Figure 1.10). Various methods for enhancing island generation using noise and falloff blending have been proposed for applications in games and virtual worlds [Ols04]. While these techniques excel at producing large, visually diverse landscapes quickly, they suffer from several key limitations when applied to the modelling of coral reef islands.

However, limitations arise when applying these methods to coral reef islands as noise-based modelling lack grounding in geological or biological reality. They generate spatial patterns through mathematical noise, not through simulations of real-world processes such as volcanic subsidence or coral accretion. The signal processing parameters typically involved (frequency, lacunarity, gain, amplitude, ...) are tuned for visual effect rather than scientific plausibility. As highlighted in procedural modelling surveys [SKG*09, GGP*19], this disconnect results in a lack of semantic control and poor correlation with actual environmental dynamics, making it difficult to represent phenomena such as reef rings, lagoons, or atoll structures in a biologically or geologically coherent way.

Moreover, the biological aspects of coral growth are inherently tied to environmental conditions. Coral reefs form and persist only within specific ranges of water depth, sunlight, salinity, and water quality. Their growth patterns are further influenced by ecological health, nutrient availability, and symbiotic relationships. These dependencies are extremely difficult to capture in procedural noise models, which are not designed to reproduce such complex and coupled dynamics.

Our approach goes beyond the randomness of noise-based generation by incorporating real-world geological and biological processes into the terrain formation pipeline. Specifically, we model the gradual subsidence of volcanic islands and the upward growth of coral reefs, both of which are central to the long-term evolution of coral reef islands. By embedding these natural processes directly into the generation algorithm, we produce terrains that are not only more realistic but also more controllable. This integration of scientific modelling with procedural flexibility allows us to overcome the inherent limitations of traditional noise-based techniques and more accurately represent the complex formation of coral reef island systems.

Simulation-based modelling

Simulation-based terrain modelling methods aim to increase realism by replicating natural processes such as erosion, sediment transport, tectonic uplift, and vegetation growth. Unlike noise-based approaches, which rely on random functions, simulation-based approaches model causality and temporal dynamics to describe how a terrain evolves over time under physical or biological forces. These methods are often used to enhance base terrains, adding geologically plausible detail and structure [BTHB06, SKG*09].

Hydraulic and thermal erosion

Hydraulic erosion models simulate the impact of flowing water on the landscape by modelling erosion, sediment pickup, transport, and deposition. Early implementations by [MKM89] laid the groundwork for erosion in procedural generation, while more recent works have accelerated these simulations using GPU architectures [MDH07] and particle-based methods [NWD05]. These simulations either follow Eulerian fluid models or Lagrangian particle systems to capture terrain displacement.

Thermal erosion, by contrast, simulates mass movement due to gravity, redistributing material from steeper slopes to gentler gradients, akin to landslides or soil creep [BTHB06]. These erosion models generate realistic fluvial networks and landforms, but they are parameter-sensitive and computationally expensive.

Moreover, such models are generally designed for terrestrial landscapes and lack mechanisms for simulating underwater sedimentation, reef growth, or biogenic processes crucial to coral reef island formation. These models typically simulate time scales relevant to geomorphological processes (hundreds to thousands of years), which are mismatched with both the faster dynamics of biological processes (e.g., coral growth) and the slower geological evolution of reef islands.

We propose our new particle-based erosion simulation method, adapted for underwater and terrestrial landscapes, in **??**.

Tectonic uplift and geologic simulation

Geological simulation approaches such as those proposed by [CBC*16, CCB*17] and extended by [SPF*23] model terrain evolution through crustal deformation and tectonic uplift. These methods simulate isostatic adjustments, plate tectonics, or local uplift phenomena, often over geological timescales.

Although well-suited for mountain-building processes or fault line modelling, these methods are not designed to account for biogenic terrain formation, such as coral reef accretion, which is critical for simulating coral reef islands. As a result, despite being physically grounded models, they do not capture the coupled geological and biological dynamics necessary for representing the long-term evolution of reef islands.

Vegetation and ecosystem dynamics

Some simulation-based terrain models integrate ecological dynamics to reflect the feedback between

terrain and living systems. For instance, [ECC*21] and [CGG*17] simulate interactions between vegetation and terrain erosion, modelling plant colonisation, growth, and their influence on soil stability and moisture retention.

These ecosystem simulations allow more complex landscape evolution by considering biotic agents; however, they are designed primarily for terrestrial plants and temperate ecosystems. Coral colonies, in contrast, are marine organisms with strict environmental requirements (e.g., limited depth, adequate sunlight, nutrients presence, warm water temperatures, ...). Accurately simulating these dependencies would require significant computational resources. We present a deeper analysis of current ecosystem simulations in **??**'s state of the art.

Furthermore, coral growth is not a passive process such as sediment accumulation, but an active accretion system that builds calcium carbonate structures over thousands of years. These unique growth mechanisms, constrained by marine ecology, fall outside the scope of existing vegetation or soil-plant-water feedback models.

While simulation-based models represent a significant advancement over purely procedural approaches, they fall short in capturing the coupled geological and biological dynamics that shape coral reef islands. They are either computationally intensive, domain-specific, or biologically inapplicable, highlighting the need for a new class of terrain generation tools that embed long-term marine biogeomorphological processes into the procedural pipeline.

[Li21] procedurally synthesizes coral colonies structures by emulating stochastic reef growth with a Diffusion-Limited Aggregation process, producing reef patterns but does not propose control over the simulation. The integration of user input such as sketch interfaces remains essential for controlling the generated output.

1.2.3 Sketch-based terrain modelling

The term "sketching" encompasses several definitions: either refering to performing hand or body gestures, creating a rough drawing, or outlining an idea in a simplified form. Accordingly, sketch-based modelling in 3D computer graphics can be understood through three complementary perspectives, each centred around a distinct core concept: interaction, construction and interpretation.

First, "sketching" may focus on interaction, where gestures captured through hand or body motion are used to manipulate virtual objects, often in immersive environments (virtual or augmented reality), using established techniques such as sculpting and distortion [OSSJ09, CA09]. Second, "sketching" may involve construction, where simple geometric primitives (curves, parametric shapes, implicit surfaces, ...) are combined under constraints to build a more complex model. Finally, "sketching" may centre on interpretation, where the user draws strokes on a 2D canvas and the system analyses their meaning to generate a plausible 3D model.

While sketch-based modelling encompasses a wide range of techniques, including gesture-driven interaction in immersive environments, this work focuses primarily on construction and interpretation aspects. In particular, construction serves as the foundation for procedural generation techniques using geometric primitives and constraints, while interpretation becomes relevant when exploring data-driven approaches using deep learning to infer terrain structure from sketches. Interaction-based techniques, though significant in other contexts, fall outside the scope of this work.

To clarify terminology in this chapter, we refer to the constructive approach as sketch-based, and to the interpretive, learning-driven approach as learning-based. It is important to note, however, that the boundaries between these categories are inherently blurry and often overlap in practice.

In procedural terrain generation, sketch-based construction approaches enable users to shape landscapes by manipulating high-level geometric primitives through intuitive sketching interfaces. These

methods allow the definition of key terrain features (mountains, valleys, coastlines, ...) by drawing their outlines on a two-dimensional canvas, which are then procedurally transformed into 2.5D or 3D terrain representations. This approach offers a high degree of control, making it particularly effective for creative applications such as video games and robotics simulations, where modelling is primarily user-driven.

Curve-based modelling

Sketch-based terrain generation often begins with user-defined curves which act as high-level constraints to guide the shape of the terrain. These curves represent silhouettes, ridgelines, valleys, or feature outlines. Once defined, they are interpreted by the system and translated into elevation changes through various computational techniques. This approach type of approach allows for intuitive control over large-scale landforms while maintaining a procedural foundation for terrain synthesis.

One of the earliest and most influential works in this domain was introduced by [GMS09] (Figure 1.11), enabling users to sketch silhouettes, ridges, and spine curves to define complex terrain structures. The method employs multiresolution surface deformation and propagates wavelet-based noise from the sketched features to their surroundings, allowing users to generate detailed, natural-looking terrains from minimal input. This approach demonstrated the effectiveness of combining intuitive sketch input with procedural detail synthesis.

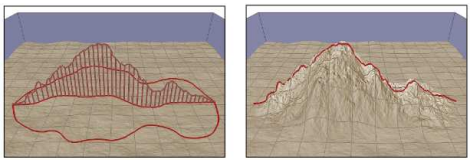


Figure 1.11: *[GMS09] propose to edit terrains through sketching by diffusing a noisy height function over a bounded region (left), effectively modelling mountain ranges (right).*

We draw direct inspiration from this work's dual-view sketching strategy, combining top-view and profile sketches, which closely aligns with our concentric curve and height-profile input approach.

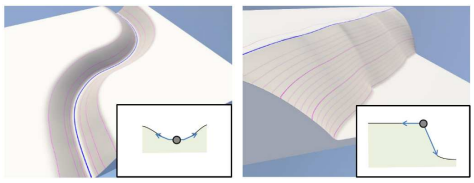


Figure 1.12: *[HGA*10] defines the surface by elevation curves with constrained slopes, forming with ease river beds and cliffs.*

Building on this idea, [HGA*10] proposed a technique based on diffusion equations. In their method, curves are annotated with geometric constraints (elevation, slope, and roughness), and a diffusion process is used to interpolate these constraints across the terrain surface. This results in smooth, continuous elevation fields that conform to user-defined features such as rivers, ridgelines, or cliffs (Figure 1.12). The use of parameterised curves as terrain anchors allows for precise control over landform shaping, while maintaining a high degree of automation.

In a different interaction paradigm, [TEC*14] introduced a first-person sketching interface, where users draw terrain silhouettes from a particular camera viewpoint (Figure 1.13). These silhouettes are then projected into 3D, and a deformation algorithm adjusts the terrain so that the drawn features are visible exactly as intended from the user's perspective. This method supports complex silhouettes with occlusions, T-junctions, and cusps, and represents an immersive and perceptually grounded approach to sketch-based terrain editing.

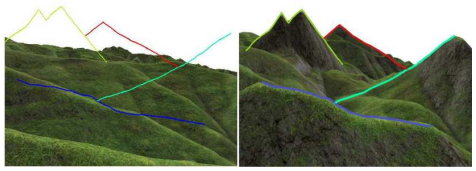


Figure 1.13: *[TEC*14] use sketching in association with camera position and direction to constrain the edition of height fields.*

These methods demonstrate the expressive power of curves as terrain-defining elements. By enabling users to sketch intuitive shapes and constraints, they bridge the gap between artistic intent and procedural complexity. Curve-driven approaches remain foundational in terrain modelling, particularly

when user control over large-scale structure is essential. While we do not use the diffusion model, the idea of sketch-defined elevation constraints along curves informs our use of user-defined shape boundaries.

Constraint-based modelling

Constraint-based and gradient-based approaches focus on exerting precise control over terrain features via formal specifications such as elevation values, slopes, or gradient fields. These methods prioritise structural accuracy and procedural consistency, making them particularly suited to applications that demand terrain realism, integration with geographic data, or fine-grained editing capabilities.

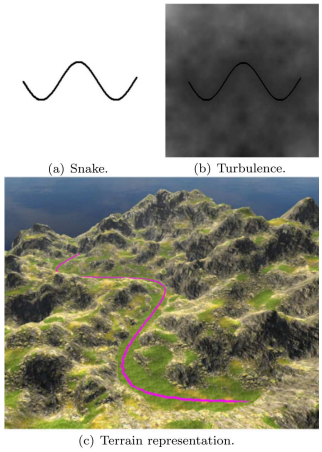


Figure 1.14: *[GCRR20] use a top-view sketch to apply constraints (a) on the noise function modelling a terrain, creating an intuitive means to create paths and roads in the resulting terrain (b and c).*

A representative example of constraint-based modelling is presented by [GCRR20], proposing a method for procedural terrain generation that respects user-defined elevation constraints (Figure 1.14). Their system allows users to fix values at specific control points (e.g., paths, landmarks) and then solve a system of equations to propagate these constraints throughout the terrain. The method integrates these constraints with a noise-based procedural function to preserve natural randomness while conforming to user intent. This approach is especially valuable when generating terrains that must align with real-world data or gameplay constraints.

We do not adopt their constraint-solving mechanism, but conceptually relate our profile sketch input to a localised height constraint.

Extending this idea, [TB18] introduce a real-time sketch-based terrain generation system based on a GPU-accelerated implementation of midpoint displacement. Users sketch curves with explicit elevation values, which act as absolute constraints, and the system extrapolates and interpolates terrain surfaces in real-time. Moreover, the user input influences the elevation constraint and interpolation method properties of the midpoint displacement algorithm using the Red, Green and Blue channels (Figure 1.15). Their model supports both global and local control over interpolation curvature and roughness, and introduces semantic labelling of sketched features (e.g., ridges vs. rivers) to influence how terrain propagates around constraints. This combination of sketch-based input, constraint propagation, and semantic control enables expressive, large-scale terrain modelling at interactive speeds. We do not reuse their fractal interpolation model, but adopt their notion of semantic labels and hierarchical constraint propagation to support real-time terrain shaping with sketch-defined features.

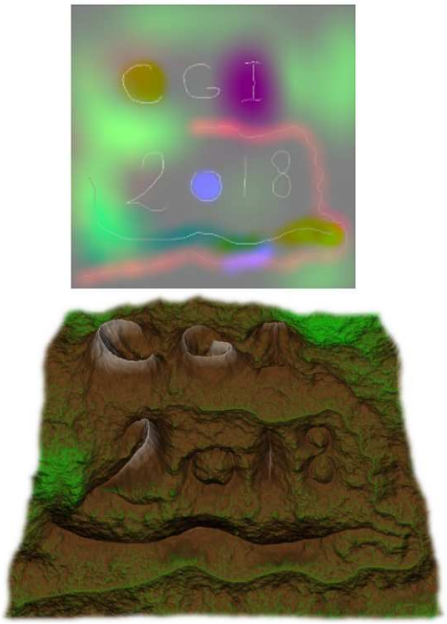


Figure 1.15: *[TB18] use top-view sketching to encode properties in the noise function used to model the terrain surface.*

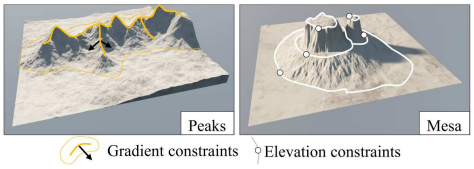


Figure 1.16: *[GPM*22] apply gradient (left) and height (right) constraints on curves and diffuse them on the terrain to create landscapes with lightweight and multi-scale representations.*

Building on the need for more intuitive editing, [GPM*22] introduce gradient domain paradigm for terrain modelling. Rather than specifying elevation values directly, users interact with slope-based representations, allowing for the manipulation of terrain inclination and the integration of local edits into global terrain structure (Figure 1.16). By controlling terrain gradients and reconstructing elevation through integration, this method enables seamless blending between regions and supports a more natural editing workflow, particularly for sculpting realistic mountain ridges, valleys, or plateaus. This gradient-domain editing approach offers interesting insights, but is not directly used, as we operate in the elevation domain with semantic control.

Both approaches offer complementary strengths: constraint-based methods ensure precise adherence to user-defined features or data sources, while gradient-based systems provide fluid, perceptual control over terrain shaping. Together, they represent a shift toward high-level modelling tools that maintain procedural expressiveness while granting users a deeper degree of terrain control.

Semantic terrain representation

Beyond geometric sketching and low-level constraints, a third class of methods explores high-level terrain construction, where users guide terrain generation using abstract or semantic inputs. These approaches aim to simplify the authoring process by allowing users to describe what a terrain should contain (e.g., a mountain or a valley) without specifying how to generate it geometrically. Such methods often rely on symbolic sketching, sparse representations, or domain-specific visual cues, offering powerful tools for conceptual design and inverse procedural modelling.

In this vein, [GGP*15] propose a method for representing terrains as sparse combinations of procedural primitives, referred to as "terrain atoms." These atoms are stored in a dictionary and are either extracted from real-world data or generated synthetically. The terrain is modelled as a linear combination of these features, forming a Sparse Construction Tree that blends primitives in a compact and expressive form (Figure 1.17). This representation facilitates terrain editing, amplification, and reconstruction from coarse user input. While this work introduces a symbolic representation of terrain via atoms, it is not reused in our method, which instead relies on semantic label maps.

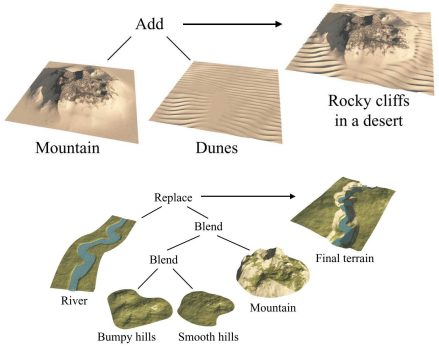


Figure 1.17: *[GGP*15] symbolically define a terrain by an aggregation of patches with real-world meaning.*

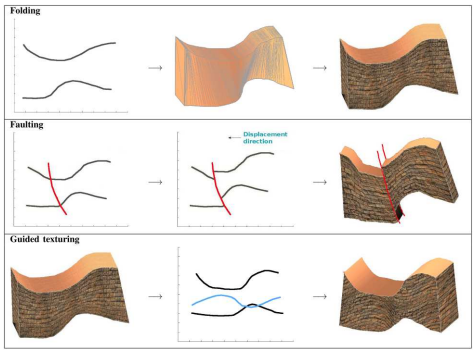


Figure 1.18: *[NVP12] use sketching to describe geological phenomena: (top) geological layer deformation, (middle) abrupt displacement from faulting, and (bottom) texture deformation.*

A more illustrative and domain-specific use case is presented by [NVP12], who introduce a system for rapid visualisation of geological concepts. Here, users sketch schematic representations of subsurface structures such as faults, folds, or strata, and the system generates plausible 3D visualisations of geological terrains (Figure 1.18). The tool is designed primarily for educational and exploratory purposes, enabling geoscientists and students to create, manipulate, and communicate complex geological scenarios through intuitive sketch input. Although it extends beyond traditional terrain elevation modelling, the work exemplifies how sketch-based systems can operate on a conceptual level and support domain-specific semantics. This work inspired our use of sketch strokes to define deformation fields, although our implementation targets structured terrain generation rather than schematic visualisation.

These high-level approaches demonstrate the potential of sketch-based modelling not just as a geometric tool, but as a semantic interface between human intention and terrain synthesis. By abstracting terrain construction into symbolic or feature-based representations, they allow users to create rich, expressive landscapes without directly engaging with low-level geometry, making them particularly valuable for tasks involving conceptual design, education, and inverse procedural modelling.

The works presented in this section illustrate the diversity of sketch-based approaches for constructive terrain modelling, from curve-driven shape control to constraint-based editing and semantic abstractions. While these methods offer valuable tools for intuitive user interaction and procedural shaping, they often lack ecological grounding, multi-view integration, or the ability to produce structured data suitable for training generative models. In our work, we reinterpret and adapt elements from these approaches (dual-view sketching, curve-based region definition, and deformation fields) to support the generation of coral reef islands through a hybrid procedural and learning-based pipeline. This constructive sketch-based foundation enables us to balance user control with scalable terrain generation in data-sparse domains.

1.2.4 Deep learning

Over the past decade, deep learning has revolutionised almost all areas of computer graphics and procedural content creation by learning complex, data-driven priors directly from examples. Unlike purely procedural or sketch-based methods, which rely on hand-tuned noise functions or geometric constraints, neural networks are trained to capture subtle patterns and high-frequency details without explicit programming of each effect. In terrain synthesis, this enables models to infer realistic elevation structures, textures, and region transitions from training data, even when that data is sparse or synthetic. In the context of coral reef islands, where high-resolution digital elevation models are rare, deep learning offers a way to abstract away low-level procedural rules and directly learn the mapping from semantic layouts (label maps) to plausible height fields. In the following sections, we first review general Generative Adversarial Networks (GANs) and then focus on their conditional variant (cGAN), which forms the backbone of our sketch-to-terrain translation pipeline.

Generative Adversarial Networks

Generative Adversarial Networks (GANs), introduced by [GPM*14], are a class of generative models in which two neural networks are trained in opposition (Figure 1.19):

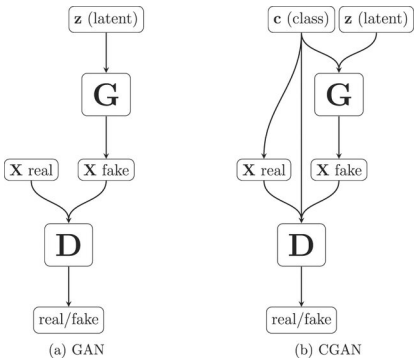


Figure 1.19: *The general structure of GAN and cGAN networks is similar: a generator network G is trained to take some noise z as input to try to create a "realistic" output X_{fake}, and a discriminator network D is trained in parallel to distinguish generated data from real data. cGAN networks introduce a class input c, used by both the generator and discriminator in their inference process. In the end, only the generator is used to create new data.*

- a generator G learns to produce synthetic "fake" data samples that resemble those from a target distribution given a random noise z,
- a discriminator D learns to distinguish "real" samples from those generated.

Through this adversarial process, the generator gradually improves its ability to mimic the underlying data distribution, enabling the creation of realistic outputs from random input.

GANs have been widely adopted for image synthesis, texture generation, and data augmentation, among other tasks. In terrain modelling, they offer the potential to generate realistic landforms by learning directly from real-world data, without requiring hand-crafted procedural rules. The following works demonstrate how different GAN variants have been applied to terrain synthesis, each with its own assumptions, design trade-offs, and limitations.

Early terrain GANs used unconditional generation, synthetising elevation maps from pure latent noise, without any spatial or semantic guidance. [WRMB18] trained a Deep Convolutional GAN (DCGAN) to generate realistic digital elevation models of mountainous landscapes, showing that terrain-like structures could emerge from purely data-driven learning. The model captures local elevation statistics and allowed for latent space interpolation, enabling smooth variations across generated terrains. However, the lack of spatial conditioning make it difficult to control or constrain features such as ridges, valleys, and coastlines, resulting in landscapes that reflected training set distributions but offered no means for intentional design.

[SW19] extended this approach with a Spatial GAN that generates height and texture maps jointly. By conditioning the generation process on spatial coordinates, their model enforced local consistency and reduced structural artefacts. This integration simplified the content pipeline by fusing geometry and appearance into a single pass.

Yet despite improved quality, the generation process remained fundamentally uncontrolled: there was no way for users to specify terrain layout, features, or semantics. As with earlier GANs, the model learned to mimic terrain distributions but could not support authoring or guided synthesis.

Later works introduced multi-stage architectures in which a second conditional GAN (cGAN) refines or interprets the output of a first-stage generator. [BP17] proposed a two-step pipeline where a DCGAN generates bare terrain heightmaps from noise, and a conditional pix2pix network adds texture based on semantic cues. This separation of geometry and appearance allows for basic stylisation and terrain remixing, but control in the initial terrain remains limited due to purely noise-driven GAN. Using a conceptually similar structure, [PC20] reversed the mapping: an unconditional GAN first synthesises aerial RGB imagery from noise, which is then passed to a cGAN trained to predict plausible DEMs. While this image-to-DEM translation enables realistic terrain reconstruction, it relies

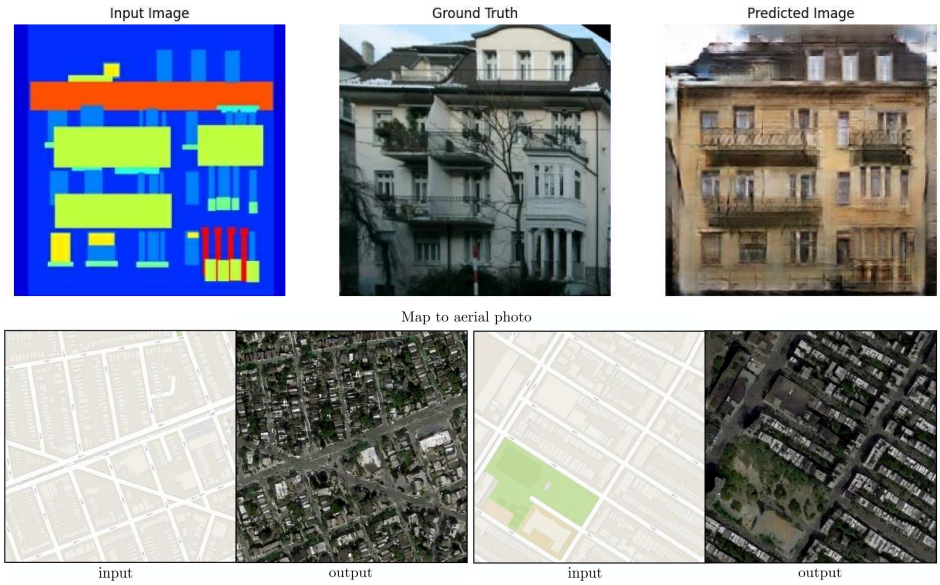


Figure 1.20: *Example of pix2pix image-to-image translation: (top) a trained model converts semantic label maps into realistic façade images, and (bottom) constructs highly plausible aerial images from navigation map sketches. This paradigm generalises to many tasks, including terrain generation.*

on large paired datasets and lacks any semantic or structural user control. In both cases, despite the introduction of a conditional refinement stage, the generation process remains fundamentally anchored in an unconstrained latent input, offering limited authoring capability and no intuitive tool to shape landform structures.

These works show a shift towards using conditional models to guide terrain generation with more structure. But because they still start from random noise, they offer little real control over the layout or meaning of the terrain. A natural next step is to guide the generation process directly from user-defined semantic inputs, such as sketches or label maps, to produce terrain that reflects both the training data and the user's intent.

Conditional GANs for terrain generation

While traditional GANs generate data from noise, conditional GANs (cGANs) extend this concept by incorporating side information, often called a class map or label map, to guide generation toward user-specified outcomes [MO14] (c (class) in Figure 1.19). This makes them particularly attractive for structured content synthesis tasks, including terrain generation, where user input often defines large-scale layout and realism must emerge from learned detail. The pix2pix framework by [IZZE17] is the canonical cGAN formulation for image-to-image translation. It uses a U-Net generator conditioned on an input image (a sketch or a label map, for example), and a PatchGAN discriminator that evaluates realism at the patch level, thus encouraging fine detail and local consistency (Figure 1.20 presents two image-to-image translations from the same pix2pix network architecture).

In terrain generation, the use of cGANs remains surprisingly rare. The most directly relevant precedent is the work of [GDG*17], who train a pix2pix-style cGAN to map sketched terrain features such as valleys, ridgelines, or peaks into full-resolution digital elevation models (DEMs). Their results demonstrate that cGANs can plausibly reconstruct complex topographic forms from sparse semantic cues, offering a promising balance between user control and learned realism. [LGP*23] updates the image-to-image process by replacing the cGAN with a Denoising Diffusion Probabilistic Model [HJA20, ND21], improving the percieved quality of the resulting terrains through an iterative noising and denoising process, trained on a large DEM dataset. Diffusion models are however known to be multiple orders of magnitude slower than GANs or cGANs [XKV22, HRJ*23], making them unusable for interactive sessions on a personal computer.

[NJSR22] inserts a Variational Autoencoder [KW22] before the cGAN phase to first compress the user input in a latent space, allowing to forward pass the cGAN with variations of the user input sketch or interpolate between different inputs in the 128 dimensions offered by the latent space.

Another approach from [VMP21] uses a GAN-based system that maps sparse "altitude dot" maps to plausible terrain imagery, acting as a minimal-interaction generative authoring tool. While not strictly a cGAN, their system reflects a similar spirit: conditioning generation on lightweight user constraints. Likewise, [PC20] and [BP17] train a cGAN to invert RGB satellite imagery into elevation data, framing terrain modelling as an appearance-to-geometry and geometry-to-appearance translation task. However, such image-to-DEM (and DEM-to-image) systems are heavily dependent on large paired datasets, limiting their applicability in settings such as coral reef islands, where training data are scarce.

Despite these advances, no standard pipeline exists for generating detailed terrains from semantic layout maps using cGANs, particularly in biologically driven environments. The potential of this approach remains underexploited, especially when it comes to coupling user control with long-term geological plausibility.

In our method, we address this gap by training a pix2pix cGAN to transform label maps (semantic region maps produced by procedural sketch-based modelling) into plausible coral reef island height fields. Each input map encodes key zones of an island (e.g., lagoon, reef crest, beach, island core) using categorical labels, serving as semantic constraints for terrain synthesis. The cGAN generator learns to condition elevation details on both the global layout and the implicit patterns encoded in the training data. By generating our own synthetic dataset with procedurally modelled coral reef islands (see Figure 1.37), we overcome the shortage of labelled elevation data and enforce geological coherence via data generation design. The choice of the pix2pix architecture is tailored by its short training and inference time and code availability by the authors, but any other conditional architecture could be substitued such as pix2pixHD or BicycleGAN [WLZ*18, ZZP*18].

This setup offers several key advantages:

- respecting user-defined structure while allowing the generator to introduce realistic variation,
- removing procedural biases (radial symmetry and fixed island typologies),
- enabling the generation of irregular, non-circular landforms while implicitly modelling geological processes such as subsidence and coral accretion through training-time priors.

In this context, conditional GANs emerge as a powerful yet underutilised tool for terrain modelling. By training on procedurally generated coral reef island data, we show that sketch-conditioned learning can effectively bridge the gap between high-level user intent and geologically plausible terrain synthesis. We note that few-shot or one-shot learning methods are emerging such as [LB25] using a Graph Neural Network, possibly solving the dataset scarcity problem, at the cost of methods falling out of the interactive-time scope with multiple seconds at inference time.

1.3 Description of our method

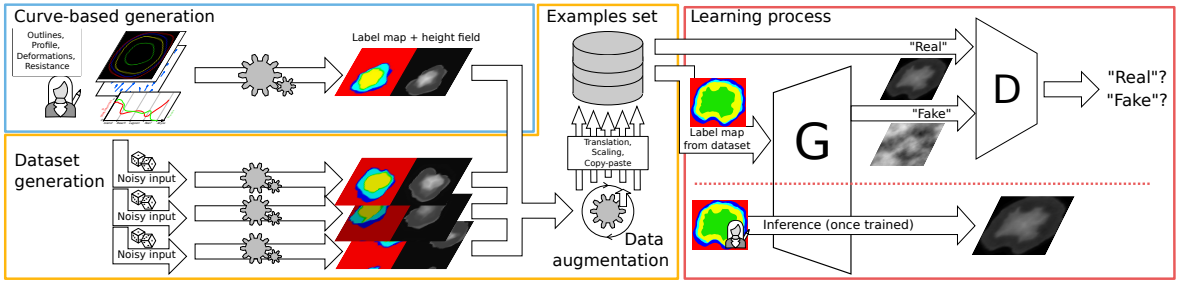


Figure 1.21: *Our pipeline first prompts the user to create single pairs of coral reef island height fields and label maps using our proposed curve-based modelling algorithm (Section 1.4). The same algorithm is applied multiple times on randomly altered versions of the user input to create a dataset, which we further enhance with data augmentation techniques (Section 1.5). Finally, we train a conditional Generative Adversarial Network, which can then be used standalone to create height fields from labelled sketches (Section 1.6).*

Our method for procedural generation of coral reef islands is composed of two independent modelling techniques shown in Figure 1.21. A first curve-based algorithm (top-left blue block, presented in Section 1.4) parametrises the surface of islands via a top-view sketch interface, describing the outlines of its constituent regions and a stroke-based wind field deforming them, as well as a profile-view sketch describing, for each region, its altitude and resistance to deformations. User inputs are given as 1D functions (altitude and resistance), a 1D polar function (island outlines), and a 2D vector field (wind field), as shown in Figure 1.22, which are convenient to randomise through the use of noise. While effective for encoding geological principles and generating structured examples, this algorithm cannot by itself provide the full freedom and irregularity required for realistic island design. This motivates the use of a learning-based component capable of generalising beyond the procedural rules.

We propose a second learning-based generation algorithm (right red blocks Figure 1.21, developed in Section 1.6), which trains a neural generator G to transform a labelled image into a height field and a discriminator D to distinguish height fields provided from the training set against height fields generated by G. This adversarial training strengthens the outputs of the generator, up to the point where we discard the discriminator and training data, only keeping the generation step (bottom-right). Users are then able to provide unseen label maps and obtain height fields as close as possible to the training dataset. Thus, while the procedural method lacks expressiveness but provides unlimited synthetic data, the cGAN offers expressiveness but requires such data. Their combination is therefore natural: the procedural model acts as a data generator, while the cGAN removes its structural biases.

We connect these two algorithms through a process of dataset generation (central orange block Figure 1.21, described in Section 1.5) in order to enforce the benefits of each while reducing their limitations. Given the initial curve-based user inputs, we create a dataset composed of similar samples processed by our first algorithm, rasterised into a 2D image of the parametrised regions, and paired with the resulting height field. We then significantly augment the dataset by employing various data augmentation techniques: translations, scaling, and copy-pasting of multiple islands in a single sample. We then obtain a dataset large enough for training our cGAN and enable users to procedurally create coral reef islands with a high level of control.

1.4 Procedural island generation

We propose a structured process for generating coral reef island terrains that takes the user's sketches and produces a complete 3D terrain model. This process begins with the creation of an initial height

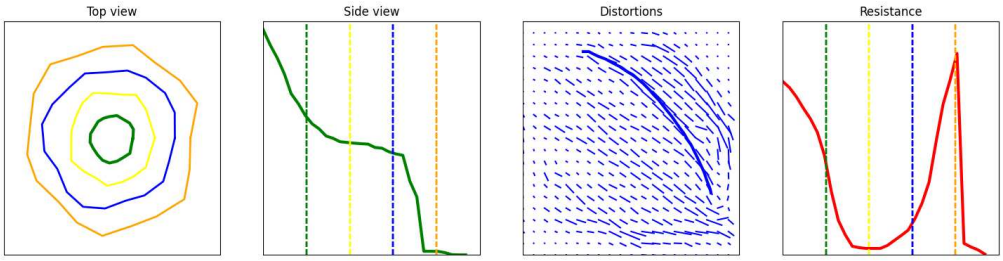


Figure 1.22: *The user interacts directly on the island by editing the different canvases in any order. This UI shows, from left to right: the top-view sketch with the different outlines of each region, the profile-view sketch with the outlines represented by dotted lines, the wind velocity sketch drawn with strokes (last stroke is visible), and the resistance function showing here a high resistance at the top of the island and on the front reef.*

field based on user inputs, and applies wind deformation to introduce natural variations, and finally integrate coral reef features via subsidence and coral growth modelling.

The generation of coral reef islands in our system begins with two intuitive sketch-based inputs from the user: a top-view sketch and a profile-view sketch, which define the island's horizontal layout and vertical elevation profile. Users further refine the terrain by applying wind deformation strokes, approximating wind and waves effects on the island's shape. This combination of sketches and wind inputs gives users precise control over both the island's structure and its natural variations, such as irregular coastlines or concave features. In this section we demonstrate the usefulness of these sketches, describe the technical details, and warp up the method in Algorithm 1.

1.4.1 Initial height field generation

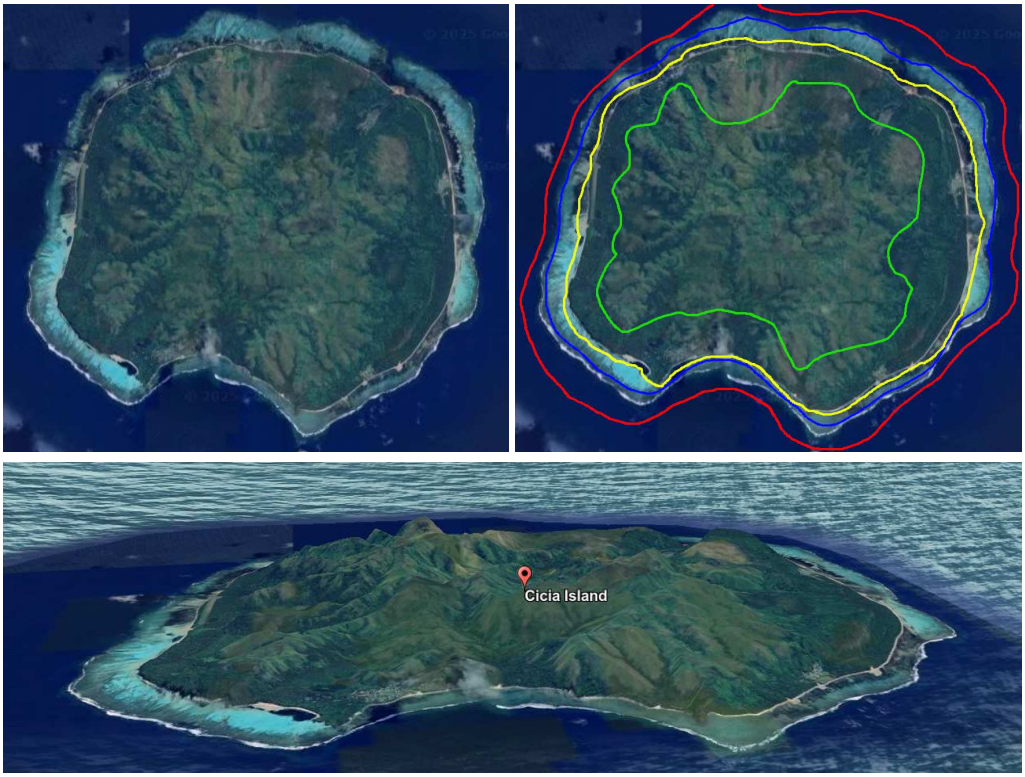


Figure 1.23: *(Left) A real-world example of aerial image (and 3D visualisation on bottom) of an island (Cicia Island) may be segmented into regions. (Right) We represent the different regions by the boundaries they form.*

The top-view sketch defines the island's outline as seen from above. Using a simple drawing interface, the user can delineate the boundaries between key regions of the island, including the island itself, the beaches, the lagoon, and the surrounding abyss (Figure 1.23). Our system assumes that these regions are arranged concentrically around the centre of the island, with each boundary defined by a radial distance from the centre (Figure 1.26).

Each region's boundary is represented in polar coordinates $(r_{\mathbf{p}}, \theta_{\mathbf{p}})$, with $r_{\mathbf{p}}$ indicating the radial distance from the island's centre and $\theta_{\mathbf{p}}$ representing the angular position. This polar representation allows us to map the user's sketch onto a circular framework, ensuring smooth transitions between regions and maintaining a coherent island layout.

In this sketch, the user defines the overall horizontal layout of the island, including the size and shape of each feature (Figure 1.24). Variations in the outline are introduced by allowing the radial distances to vary with angle, ensuring that the island is not strictly symmetrical and introducing more natural, irregular shapes.

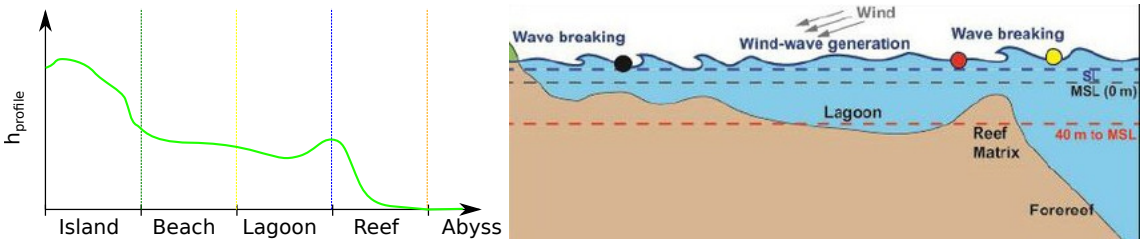
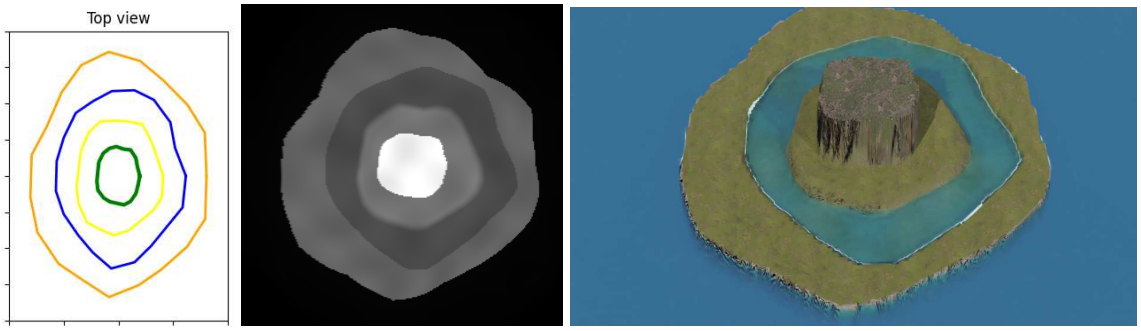


Figure 1.25: *(Left) A profile function $h_{profile}$ is defined as a 1D function and represents the surface from the centre of the island to the abysses. (Right) The cross-section representation of an island is often represented as a 1D function defined using terrain features as landmarks.*



Input sketch Output height field Result

Figure 1.24: *Using only the outlines of the island as an input sketch (left), we provide a height field depending on the region in which it relies (middle).*

The profile-view sketch defines the vertical elevation profile of the island along any radial direction, offering control over the island's height. In this view, the user specifies the elevation of different regions of the island, such as the island peak, beach, lagoon, abyss, and everything in between, by drawing the corresponding profile curve (Figure 1.25).

These regions correspond to key terrain transitions: the highest point of the island (centre), the island border, the beach, the lagoon, and the deep-sea abyss. We interpolate these milestones into a continuous 1D height function $h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$, where $\tilde{x}_{\mathbf{p}}$ represents a non-uniform region distance from the island's centre, and $h(\mathbf{p}) = h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$ gives the height at each point. This continuous profile ensures smooth elevation transitions across the island.

By combining the top-view and profile-view sketches, our method generates a coherent 3D terrain model that accurately reflects the user's design by revolution modelling.

The generation of the coral reef island terrain begins by transforming the user-defined top-view and profile-view sketches into a coherent 3D height field. This process combines the radial layout of the top-view sketch with the elevation information provided by the profile-view sketch, creating a terrain that accurately represents the desired features (i.e. the island, beaches, lagoons, and abyss).

For any point $\mathbf{p} \in \mathbb{R}^2$ on the terrain, we define its polar coordinates relative to the center of the island $\mathbf{c}$ as

$$r_{\mathbf{p}} = \|\mathbf{p} - \mathbf{c}\|, \qquad \theta_{\mathbf{p}} = \text{atan2}(\mathbf{p}.y - \mathbf{c}.y,\ \mathbf{p}.x - \mathbf{c}.x)$$

The radial distance $r_{\mathbf{p}}$ determines which region the point belongs to (island, beach, lagoon, reef, or abyss) based on the user-defined radial limits. Those outlines from the top-view sketch provide the exact boundaries between regions.

Each point's height is computed using the profile function $h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$. Instead of using the raw radial distance $r_{\mathbf{p}}$, we define a parametric region distance $\tilde{x}_{\mathbf{p}}$ that maps each point to a normalised position along the concentric regions (see Figure 1.26). The radial space is divided by user-defined boundaries

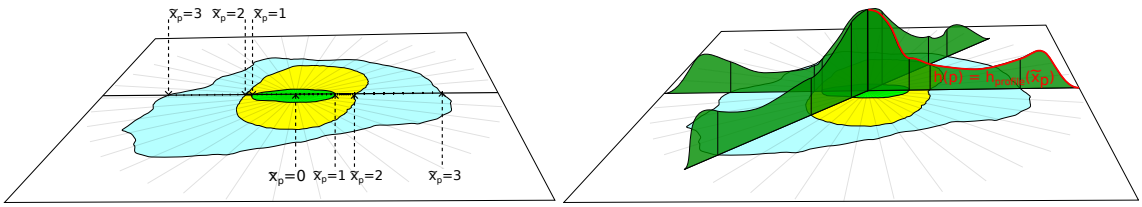


Figure 1.26: *The $\tilde{x}_{\mathbf{p}}$ parameter is used to stretch the 1D height function $h_{profile}(x)$ to fit the distances from the centre to the outlines of each region defined in the top-view sketch.*

$R_0, R_1, \ldots, R_n$, corresponding to the island centre, border, beach, lagoon, reef, and abyss.

When a point $\mathbf{p}$ lies between two boundaries R_i and R_{i+1}, its parametric distance is

$$\tilde{x}_{\mathbf{p}} = i + \frac{r_{\mathbf{p}} - R_i}{R_{i+1} - R_i} \tag{1.1}$$

where i is the index of the region containing $\mathbf{p}$ (i.e., $R_i \leq r_{\mathbf{p}} < R_{i+1}$). This linear mapping stretches each region's radial span to the interval $[i, i+1[$, ensuring smooth interpolation across region boundaries. For any point $\mathbf{p}$, the height is finally computed as

$$h(\mathbf{p}) = h_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \tag{1.2}$$

This approach ensures that the height field accurately follows the elevation profile specified by the user while maintaining smooth transitions between different regions of the island.

The result is a height field that captures both the radial structure of the island (from the top-view sketch) and the vertical elevation profile (from the profile-view sketch), producing an organic representation of islands with smooth transitions between the key terrain features. Three slightly edited height functions on an identical island layout and their associated outputs are presented in Figure 1.27. To avoid perfectly smooth surfaces, we also apply a very low-amplitude Perlin noise perturbation to the final height field, adding fine-scale irregularities.

Wind deformation

In island environments, wind and waves reshape islands through coastal erosion, which removes and redistributes material over time. Simulating this physically would require sediment transport and hydrodynamics, which is both computationally intensive and difficult to control interactively. Instead, we approximate the morphological outcome of erosion with a deformation field: user-drawn strokes define a wind velocity field that warps the island, introducing large-scale asymmetries without explicitly removing matter. To account for the fact that not all regions respond equally to erosion, we also introduce a resistance function, which modulates the deformation's strength across regions. For example, shallow coastal areas can be strongly deformed, while the deep abyss or resistant volcanic core remain largely unaffected. This combination provides an efficient and intuitive proxy for the long-term shaping role of wind and waves. Although real coastlines are shaped by both wind and wave processes, we represent them jointly through a single user-defined wind field, which serves as a controllable proxy for their combined morphological effect.

The wind field is represented as a series of strokes drawn by the user on a 2D canvas. Each stroke represents a parametric curve, where the direction and strength of the wind are encoded as a vector field. The user controls the wind's direction by drawing a set of curves, and we interpret the strokes to create a velocity field that defines how the terrain should be deformed.

As the user draws a stroke, a set of control points are generated along the curve, with the option to adjust the stroke's width. The width of each stroke determines the area of influence around the curve, where wider strokes result in broader deformations of the terrain. The deformation strength decreases

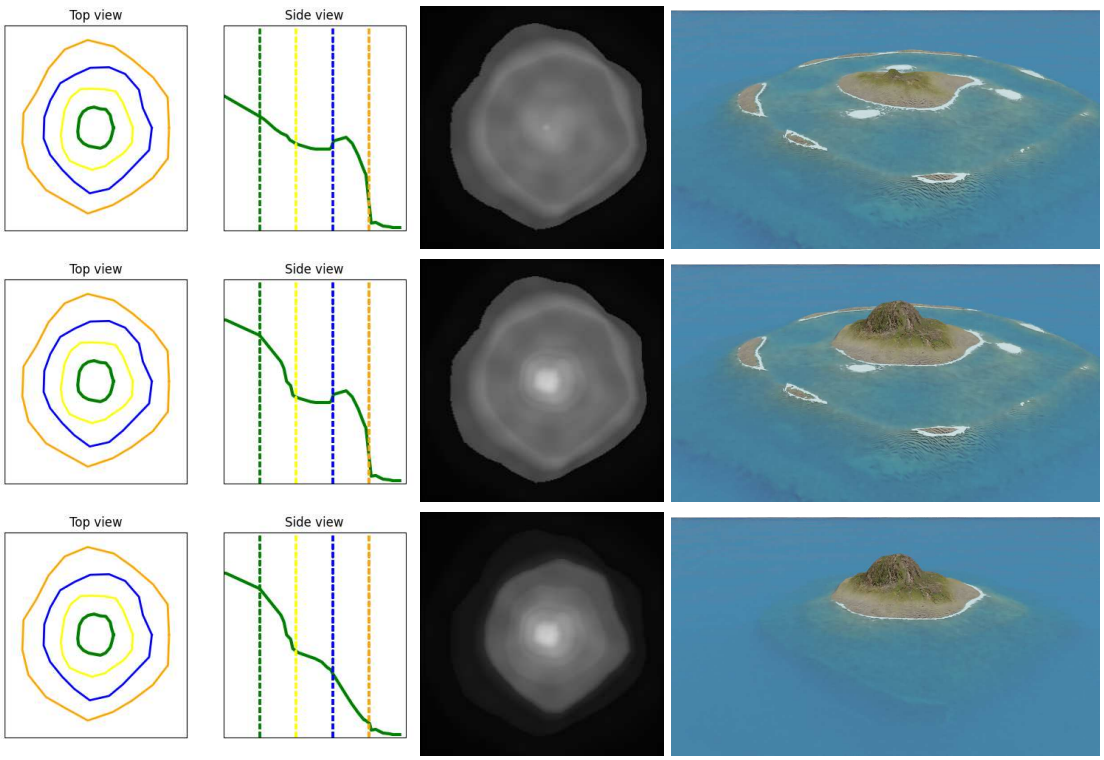


Figure 1.27: *Providing a smooth function between each region results in islands with plausible reliefs. We fixed the outlines while editing only the height function in order to produce, from top to bottom, a low island, a coral reef island, and finally an identical island without the reef.*

with distance from the curve using a Gaussian falloff function using the stroke width as standard deviation. This ensure that the terrain transitions smoothly from deformed regions to non-deformed areas.

Once the strokes are applied to the velocity field, we displace each points of the terrain accordingly. The height field, originally generated from the user's sketches, is modified by the wind field to create non-radial features, breaking the initial radial symmetry and producing a more organic island shape (Figure 1.28).

The deformation is controlled via a user-defined vector field representing the direction and strength of wind flows across the terrain. Users interact with our system by drawing strokes on a 2D canvas using Catmull-Rom splines C [CR74], a type of parametric curve that allows for smooth, continuous paths representing wind patterns. Each stroke defines a wind flow in the curve's direction C', a strength S_C, and an effect width σ (Figure 1.29); these wind flows are used to displace the terrain, approximating the gradual reshaping of the island due to wind and wave erosion. The strength of the displacement is controlled by a Gaussian scaling function $G_\sigma(x)$ that smoothly decreases with the distance from $\mathbf{p}_C^*$, the closest point on the parametric curve C, as follows:

$$\Phi_C(\mathbf{p}) = S_C \cdot \frac{C'(\mathbf{p}_C^*)}{\|C'(\mathbf{p}_C^*)\|} \cdot G_\sigma\left(\|\mathbf{p} - \mathbf{p}_C^*\|\right) \tag{1.3}$$

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{1.4}$$

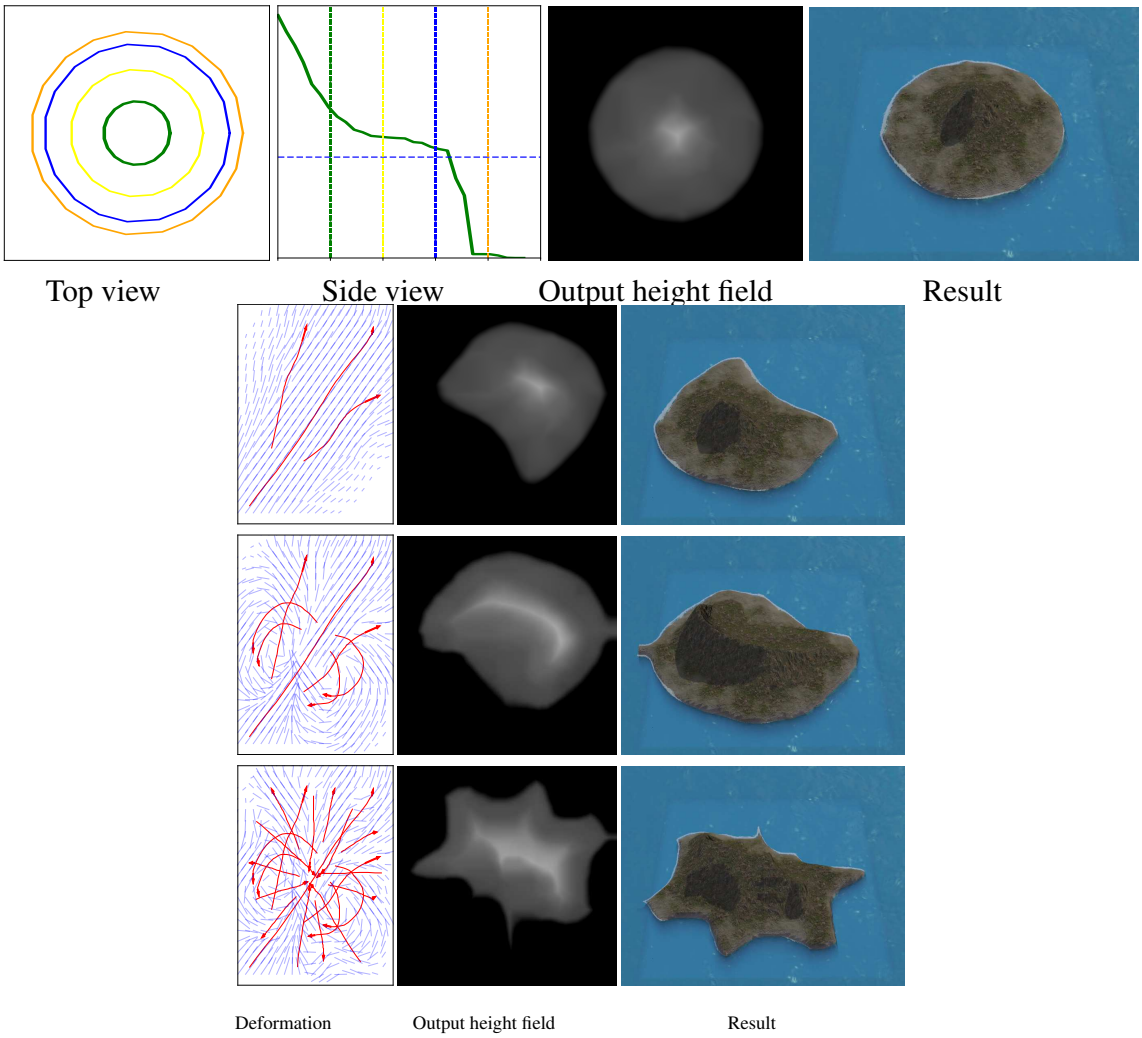


Figure 1.28: *Top row: A circular island layout and a given profile sketch creates a simple round island height field. Each following row shows a progression in an edition session, conserving the initial top-view and side-view sketch, by manipulating the deformation field (blue lines) with respectively 3, 7 and 20 user strokes (red arrows). The final result breaks the initial radial symmetry of the island layout.*

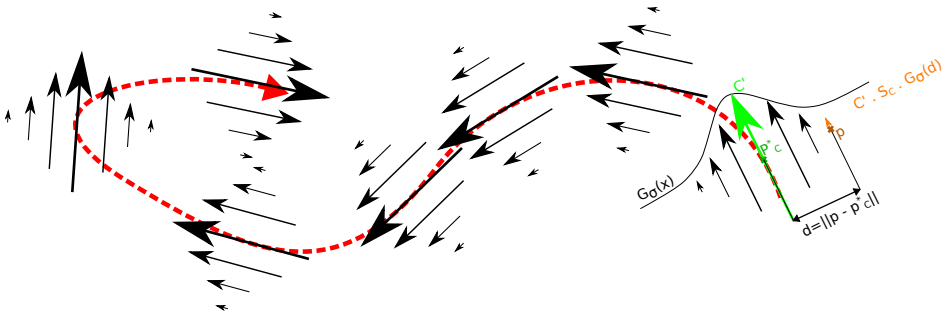


Figure 1.29: *From the parametric curve defined by a user (red), we define the velocity field by considering the directional derivative C' of the curve C at the closest point $\mathbf{p}_C^*$, modulated by a Gaussian distance function $G_\sigma(x)$ and scaled by the user-defined strength S_C.*

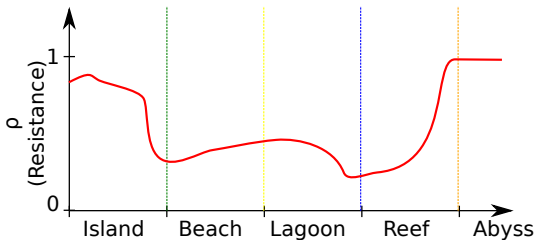


Figure 1.30: *The resistance function of the island is defined with a similar strategy as the $h_{profile}$ function. The resistance to erosion and deformation arises from multiple factors such as depth, materials, wind shadowing, biotic and abiotic factors, ... Modelling all these factors is complex. As such, using a user-defined approximation through a resistance function ρ allows for more control.*

The final deformation vector $\Phi(\mathbf{p})$ is computed as a sum of the influences from all strokes:

$$\Phi(\mathbf{p}) = \mathbf{p} + \sum_{C \in \text{curves}} \Phi_C(\mathbf{p}) \tag{1.5}$$

Once the deformation vector $\Phi(\mathbf{p})$ is computed, the terrain height at point $\mathbf{p}$ is adjusted by displacing $\mathbf{p}$ to a new point $\Phi(\mathbf{p})$. We then compute the final height $h(\Phi \circ \mathbf{p}) = h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$, or, as the implicit modelling community would write it,

$$\tilde{h} = \Phi^{-1} \circ h \tag{1.6}$$

This process introduces variations in the terrain, distorting the coastline, creating concave regions, and breaking the original radial symmetry defined by the top-view and profile-view sketches.

To ensure that certain regions of the terrain such as deep-water areas remain relatively unaffected by the wind, a resistance function $\rho(\tilde{x}_{\mathbf{p}})$ is applied. The resistance function modulates the effect of the deformation based on the previously computed piecewise parametric distance $\tilde{x}_{\mathbf{p}}$, with the same interaction means as the h_{profile} function.

We define the resistance function $\rho(\tilde{x}_{\mathbf{p}})$ similarly to the profile function, and use it to modulate the magnitude of the displacement at each point. For example, regions near the coastline (such as the beach and lagoon) have lower resistance, allowing for significant deformation (emulating coastal erosion from wave energy), while regions farther away (such as the abyss) have higher resistance, limiting the wind and coastal erosion impact.

The deformation vector previously described is then scaled by the resistance function at each point $\mathbf{p}$, such that the final deformation vector becomes

$$\tilde{\Phi}(\mathbf{p}) = \big(1 - \rho(\tilde{x}_{\mathbf{p}})\big) \cdot \Phi(\mathbf{p}) \tag{1.7}$$

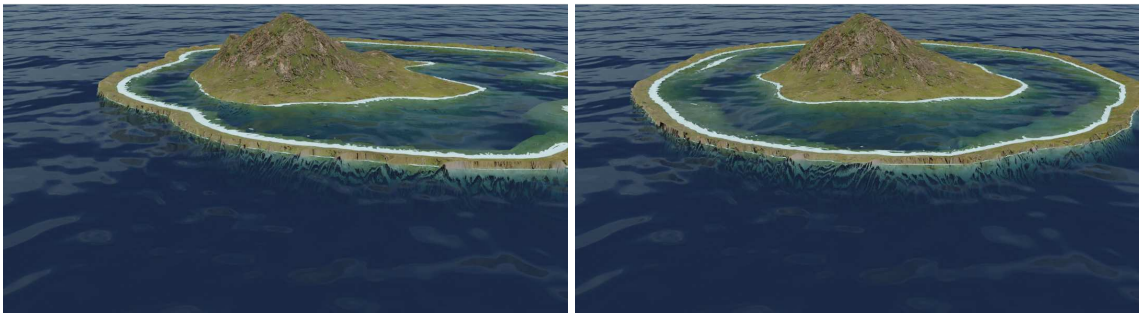


Figure 1.31: *(Left) Given a uniform velocity field and a resistance function similar to Figure 1.30, the coasts are smoothly eroded while the interior of the island is almost unaffected. (Right) Modifying the resistance function to affect a strong resistance to borders emulates the effect of coast reinforcements.*

Input: Top-view boundaries $\{R_0(\theta), \ldots, R_n(\theta)\}$, profile function h_{profile}, resistance $\rho(\tilde{x}_{\mathbf{p}})$, user strokes $\{C\}$ with width σ and strength S_C
Output: Deformed height field $\tilde{h}$ and label map $\tilde{I}$

`// (1) Initial height field and label map from sketches`
foreach *pixel* $\mathbf{p} \in \mathbb{R}^2$ **do**
  $r_{\mathbf{p}} \leftarrow \|\mathbf{p} - \mathbf{c}\|$
  $\theta_{\mathbf{p}} \leftarrow \text{atan2}(\mathbf{p}.y - \mathbf{c}.y,\ \mathbf{p}.x - \mathbf{c}.x)$
  Find i such that $R_i(\theta_{\mathbf{p}}) \le r_{\mathbf{p}} < R_{i+1}(\theta_{\mathbf{p}})$
  $\tilde{x}_{\mathbf{p}} \leftarrow i + \dfrac{r_{\mathbf{p}} - R_i(\theta_{\mathbf{p}})}{R_{i+1}(\theta_{\mathbf{p}}) - R_i(\theta_{\mathbf{p}})}$
  $h(\mathbf{p}) \leftarrow h_{\text{profile}}(\tilde{x}_{\mathbf{p}})$
  $I(\mathbf{p}) \leftarrow i$
`// (2) Wind-field deformation from strokes`
foreach *pixel* $\mathbf{p} \in \mathbb{R}^2$ **do**
  $\Phi(\mathbf{p}) \leftarrow \mathbf{0}$
  foreach *stroke* C **do**
    $\mathbf{p}_C^* \leftarrow \arg\min_{\mathbf{q} \in C} \|\mathbf{p} - \mathbf{q}\|$ `//` $\mathbf{p}_C^* \leftarrow$ `closest point of the curve` C `to the point` $\mathbf{p}$
    $\Phi_C(\mathbf{p}) \leftarrow S_C \cdot \dfrac{C'(\mathbf{p}_C^*)}{\|C'(\mathbf{p}_C^*)\|} \cdot G_\sigma\big(\|\mathbf{p} - \mathbf{p}_C^*\|\big)$ `//` $G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$
    $\Phi(\mathbf{p}) \leftarrow \Phi(\mathbf{p}) + \Phi_C(\mathbf{p})$
  $\tilde{\Phi}(\mathbf{p}) \leftarrow \big(1 - \rho(\tilde{x}_{\mathbf{p}})\big) \cdot \Phi(\mathbf{p})$ `// Modulation from resistance function`
$\tilde{h} \leftarrow \tilde{\Phi}^{-1} \circ h$ `//` $\tilde{h}(\mathbf{p}) \leftarrow h(\mathbf{p} - \tilde{\Phi})$ `for backward warping`
$\tilde{I} \leftarrow \tilde{\Phi}^{-1} \circ I$
return $\tilde{h}, \tilde{I}$

Algorithm 1: Sketches to height field with wind deformation

The deformation process results in a modified height field where the terrain has been warped according to user-defined strokes (Figure 1.31 illustrates the difference between non-resistant and resistant islands). This deformation introduces non-radial features, such as concave coastlines or irregularities along the beach and lagoon, making the island appear more natural and varied.

Both the height field and the label map (which tracks the terrain regions) are updated to reflect the deformation (Figure 1.32). This ensures that the semantic information of the terrain remains consistent even after the terrain has been warped. The label map is deformed in the same way as the height field, preserving the logical structure of the island for further post-processing, such as texturing or mesh

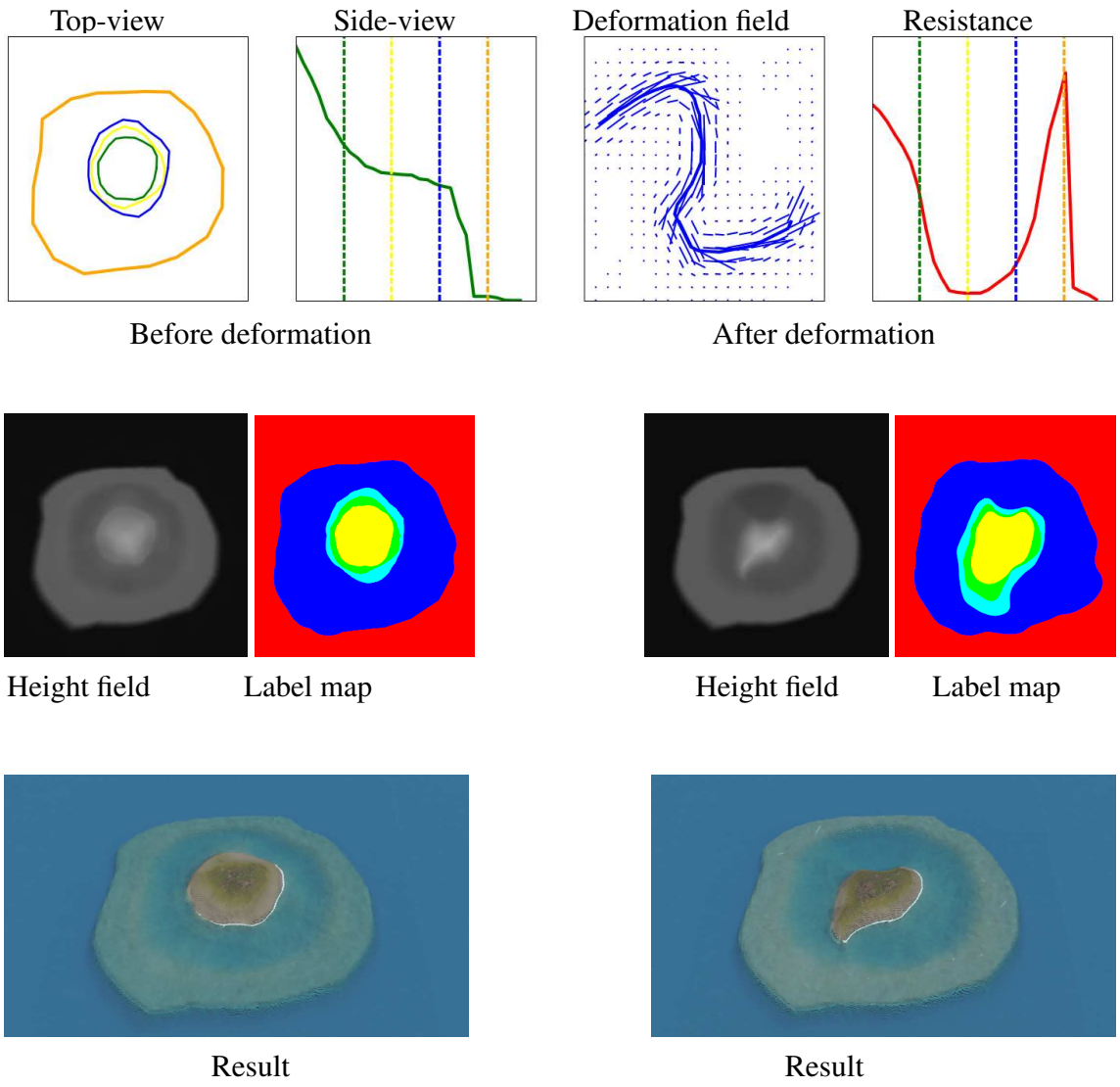


Figure 1.32: *A top-view velocity field is defined from user-provided strokes (top, blue), in association with a resistance function (top, red); a height field is deformed accordingly. Left: original height field and render; right: altered results. The beach and lagoon regions are defined with low resistance, which is visible by having only these regions deformed in bottom results.*

instancing.

In Figure 1.28, a simple circular island is generated from the initial height field. By applying strokes along one side of the island, the deformation process can create concave regions along the coastline, making the shape more irregular and mimicking the effects of real-world wind and wave erosion. The resistance function ensures that while the beach and lagoon areas are deformed, the abyss remains largely unaffected as they are far from the wind and wave effective areas, preserving the island's overall structure.

1.4.2 Coral reef modelling

Once the terrain has been generated and deformed by the wind, we emulate the subsidence of the volcanic island and, in parallel, the growth of coral reefs. These processes reflect the long-term geological evolution of coral reef islands, where the volcanic island gradually sinks (subsides) while coral reefs grow upward to "keep up" with the sinking landmass.

Subsidence

The subsidence of the island is modelled by scaling the initial height field downward, modelling procedurally the effect of the volcanic island slowly sinking into the ocean. The user provides a subsidence rate $\lambda \in [0,1]$, which represents the proportion by which the island has sunk over time. The subsidence is applied uniformly, meaning all terrain points on the island sink by the same factor.

The subsided height field $h_{\text{subsid}}(\mathbf{p})$ is computed by scaling the original height field $h(\mathbf{p})$ with the subsidence factor:

$$h_{\text{subsid}}(\mathbf{p}) = (1-\lambda) \cdot h(\mathbf{p}) \tag{1.8}$$

This scaling reduces the overall height of the island, approximating how volcanic islands sink over time due to tectonic activity and erosion. The subsidence factor λ is applied uniformly across the terrain, meaning that all points on the island experience the same degree of subsidence, regardless of their original height or location.

Coral reef growth

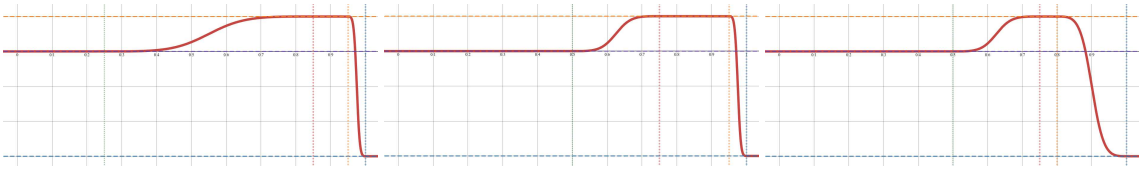


Figure 1.33: *The modelling of the reef growth in our model is described by a piecewise function h_{coral} which is flat in the lagoon, the crest and abyss, and follows a smoothstep function as transitions for the back reef and fore reef regions. The model is easily edited by tuning the height values and delimitations of the subregions. Here, three examples of reef growth function with different delimitations (vertical dotted lines), providing more or less smooth transitions between reef subregions.*

As the volcanic island subsides, coral reefs grow upward to remain close to the water surface, following the "keep-up" strategy observed in most real-world coral formations. Coral growth is restricted to regions where the depth is within the optimal range for coral development, typically from the water surface to around 30 metres below before becoming much scarcer.

The coral reef features (reef crest, back reef, and fore reef) are modelled separately from the subsidence process. We generate a coral feature height field $h_{\text{coral}}(\mathbf{p})$, which remains unaffected by the island's subsidence. This height field ensures that coral regions remain near the water surface, even as the island sinks.

In our model, coral reef growth is entirely independent of the subsided terrain. Even as the volcanic island sinks, coral growth is driven only by the proximity of terrain to the water surface, ensuring that coral features always remain near the surface, irrespective of how much the island subsides.

We define distinct reef zones anchored at specific depths:

- Reef crest near sea level: $h_{\text{crest}} = -2$,m,
- Back reef and lagoon: $h_{\text{back}} = -20$,m,
- Fore reef sloping to abyss: $h_{\text{abyss}} = -100$,m.

Each reef subregion is defined over a parametric domain $x \in [0,1]$, with $x = 0$ the beginning of the reef region and $x = 1$ its end, directly inputting the parametric distance $x = \tilde{x}_{\mathbf{p}} - i_{\text{reef}}$ with i_{reef} the

ID of the reef region, resulting in a linear interpolation between the begining and the end of the reef region in the top-view sketch. For instance:

- Back reef: $x_{\text{back,start}} = 0$, $x_{\text{back,end}} = 0.5$,
- Reef crest: $x_{\text{crest,start}} = 0.75$, $x_{\text{crest,end}} = 0.8$,
- Abyss begins at $x_{\text{abyss,start}} = 1$.

We model transition zones between these regions using a smoothstep operator [Per02]:

$$\text{smoothstep}(x) = 3x^2 - 2x^3. \tag{1.9}$$

For conciseness, denote the interpolating function as:

$$S(a, b, x_0, x_1, x) = a + (b - a)\,\text{smoothstep}\left(\frac{x - x_0}{x_1 - x_0}\right). \tag{1.10}$$

The complete coral height field, as displayed in Figure 1.33, is built as a piecewise function:

$$\begin{aligned} h_{\text{coral}}(x) = &\sum_{r \in \text{subregions}} \begin{cases} h_r & \text{if } x_{\text{r,start}} \le x \le x_{\text{r,end}} \\ 0 & \text{otherwise} \end{cases} \\ &+ \sum_{t \in \text{transitions}} \begin{cases} S(h_t, h_{t+1}, x_{\text{t,end}}, x_{\text{t+1,start}}, x) & \text{if } x_{\text{t,end}} < x < x_{\text{t+1,start}} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{1.11}$$

Blending height fields

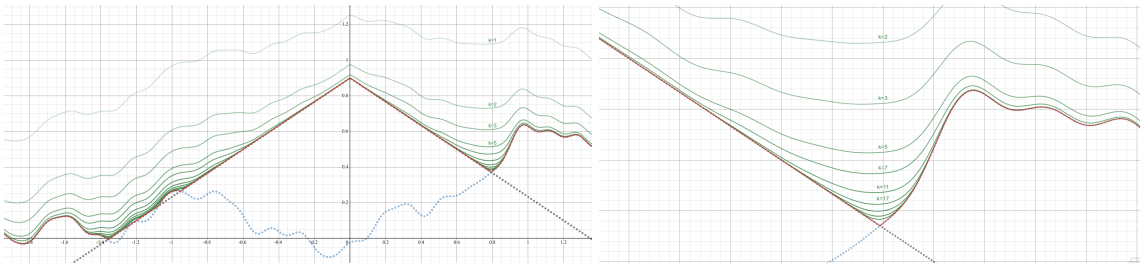
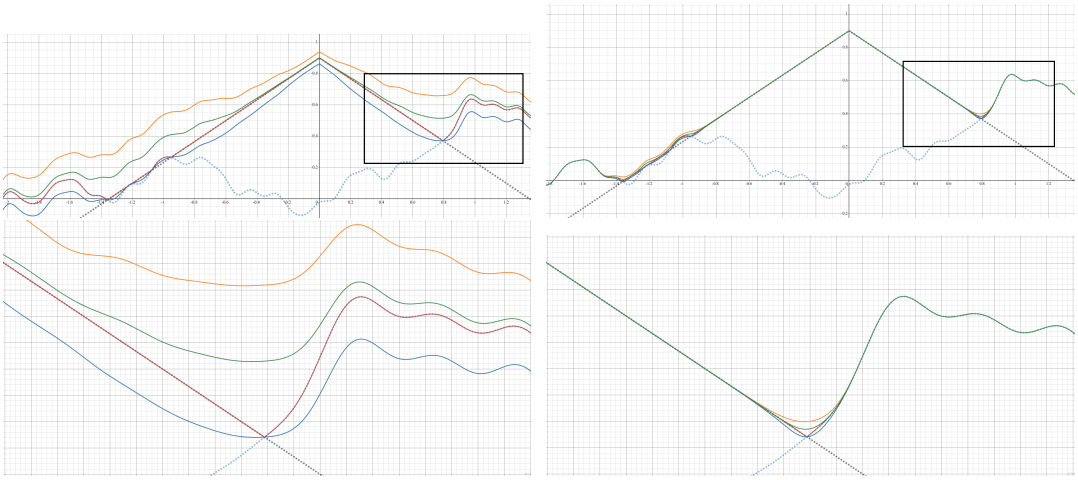


Figure 1.34: *Blending two functions $f : \mathbb{R} \mapsto \mathbb{R}$ (black, dotted) and $g : \mathbb{R} \mapsto \mathbb{R}$ (blue, dotted) with the* max *operator (red curve), causing discontinuities when $f(x) = g(x)$, and with the* smax *operator (green curves) with various sharpness values k, resolving the issue at the cost of slight overestimations with low values of k. Right presents a zoom on the domain $x \in [0.5, 1.3]$.*



$k = 5$ $k = 50$

Figure 1.35: *The* smax^+ *operator (orange curve) is a function that is used to overestimate the maximum value of two input functions (dotted curves), especially when the difference between the two functions is small, while the* smax^- *function (blue curve) tends to underestimate the* max *operator. Taking* smax *(green curve) as the average of* smax^- *and* smax^+ *creates a smooth function closely approximating the* max *operator, even with low values of sharpness* k *(Left:* $k = 5$*, right:* $k = 50$*). Bottom graphs show a zoom on the domain* $x \in [0.5, 1.3]$*.*

The final step is to blend the subsided height field $h_{\text{subsid}}(\mathbf{p})$ with the coral feature height field $h_{\text{coral}}(\mathbf{p})$ to produce the final terrain. The goal is to ensure that coral features remain near the water surface while allowing the rest of the island to subside.

To achieve this, we use a smooth maximum function, which smoothly blends the two height fields, ensuring that the coral regions dominates where coral growth is present, while the subsided island terrain dominates in other regions. This blending method ensures that the transition between the coral and subsided regions is smooth and visually consistent.

We define our smooth maximum function $\text{smax}(a,b) \in \mathbb{R}$ for $(a,b) \in \mathbb{R}^2$ as the mean of two functions, smax^- and smax^+ (shown in Figure 1.35), adapted from Inigo Quilez's smooth min function [Qui13], that respectively underestimate and overestimate the function max:

$$\text{smax}^-(a,b) = a + \frac{b-a}{1+\exp\left(-k\cdot(b-a)\right)} \tag{1.12}$$

$$\text{smax}^+(a,b) = a + \frac{b-a}{1-\exp\left(-k\cdot(b-a)\right)} \tag{1.13}$$

$$\text{smax}(a,b) = a + \frac{\text{smax}^-(a,b) + \text{smax}^+(a,b)}{2} \tag{1.14}$$

Here, $a = h_{\text{subsid}}(\mathbf{p})$ is the height from the subsided island, $b = h_{\text{coral}}(\mathbf{p})$ is the height from the coral reef feature, and k controls the smoothness of the transition (fixed arbitrarily here at $k = 5$ for heights ranging between 0 and 1). Higher values of k bring the smax function closer to the max function (Figure 1.34).

This smooth max function guarantees continuity by preventing abrupt height and slope differences between coral regions and the subsided terrain, creating a smooth, gradual transition that mimics the natural blending of coral reefs with deeper areas. The coral feature height field takes precedence where coral can grow, typically in shallow regions. In deeper regions, such as the abyss, the subsided height field naturally dominates, ensuring that the final terrain accurately reflects both subsidence and coral growth processes (Figure 1.36).

Note that the smax function is undefined for $a = b$, however, a proof of continuity for $\text{smax} \in C^\infty$ is provided in **??** (along with a deeper analysis of the function), resulting in

$$\text{smax}(a,b) = \begin{cases} a + \dfrac{1}{2k} & \text{if } a = b, \\ \dfrac{\text{smax}^-(a,b) + \text{smax}^+(a,b)}{2} & \text{otherwise} \end{cases} \tag{1.15}$$

Input: Deformed height field $\tilde{h}$, subsidence rate $\lambda \in [0,1]$, coral growth function h_{coral} (piecewise with transitions via smoothstep)
Output: Final height field $\hat{h}$

```
// (1) Subsidence
```
foreach p do
  $h_{\text{subsid}}(\mathbf{p}) \leftarrow (1-\lambda) \cdot \tilde{h}(\mathbf{p})$
```
// (2) Coral growth evaluation
```
foreach p do
  `// Use same` $\tilde{x}_{\mathbf{p}}$ `as in Algorithm 1`
  $x \leftarrow \tilde{x}_{\mathbf{p}} - i_{\text{reef}}$ `// ` $i_{\texttt{reef}}$ `= reef region ID`
  $c(\mathbf{p}) \leftarrow h_{\text{coral}}(x)$
```
// (3) Smooth maximum blend
```
foreach p do
  $\hat{h}(\mathbf{p}) \leftarrow \text{smax}\left(h_{\text{subsid}}(\mathbf{p}),\, c(\mathbf{p})\right)$
return $\hat{h}$

Algorithm 2: Subsidence, coral growth, and smooth blending

Output

The resulting terrain represents a plausible coral reef island, where the volcanic island has subsided, and coral reefs have grown upward to keep pace with the water level. The smooth blending between the subsided terrain and the coral features ensures a natural transition between regions (island, lagoon, coral reefs, and abysses).

One of the key strengths of this method is its flexibility, as the subsidence and coral reef growth processes are modelled independently, allowing for a wide range of configurations. Users can generate plausible island terrains with or without coral features, or apply the coral reef growth modelling to existing height fields from other sources. Moreover each of the pseudo-code section proposed in Algorithm 2 can be replaced with surrogate methods if desired (e.g., realistic subsidence or coral reef simulations, or the use of another blending function).

1.5 Data generation

The creation of the dataset is done through the use of the procedural algorithm for which we alter the input parameters.

For each generation, the top-view and profile-view sketches use an initial layout. Each outline of the top-view sketch is defined as a centred circle of random radius $r_{\min} \leq r^* \leq r_{\max}$. We add another deformation based on Perlin noise such that the final contour is defined as

$$r(\theta) = r^* + \eta(\theta) \tag{1.16}$$

On the other hand, we define an initial profile-view sketch by defining $h^*_{\text{profile}}(\tilde{x}_{\mathbf{p}})$, the initial height

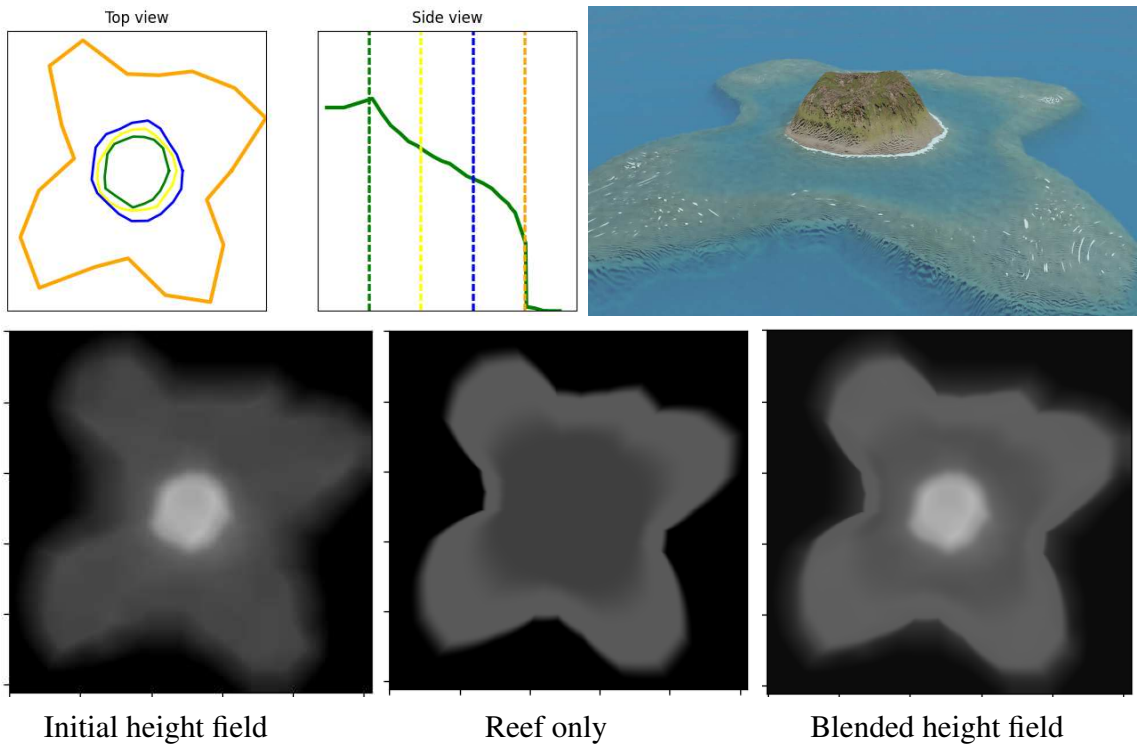


Figure 1.36: *Volcano with single vent. Bottom left: the initial height field is computed directly from the user input. Bottom centre: the reef height field is output from Equation* (1.11)*. Bottom right: blended result with Equation* (1.15)*.*

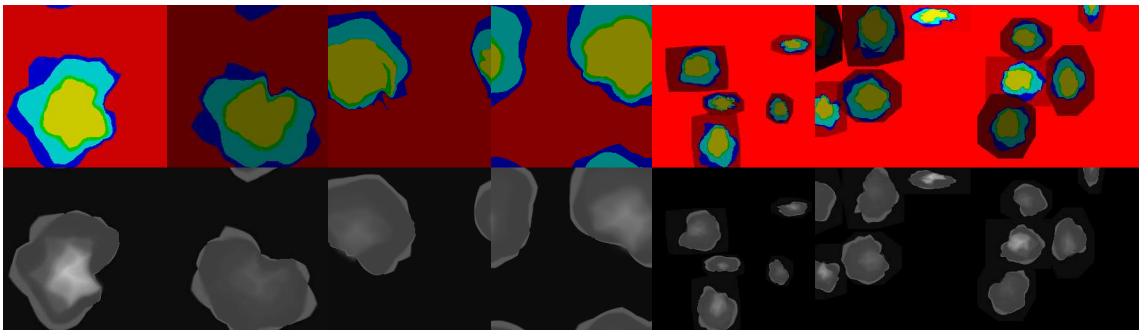


Figure 1.37: *Examples of procedural region maps used for training: left to right, canonical island, off-centre island, elongated shapes, and multi-island scenes. These maps serve as semantic inputs to the cGAN.*

function, for which fBm noise is applied to obtain

$$h_{\text{profile}}(\tilde{x}_{\mathbf{p}}) = h^{*}_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \cdot \eta(\tilde{x}_{\mathbf{p}}) \tag{1.17}$$

An identical process is done for the resistance function:

$$\rho(\tilde{x}_{\mathbf{p}}) = \rho^{*}(\tilde{x}_{\mathbf{p}}) \cdot \eta(\tilde{x}_{\mathbf{p}}) \tag{1.18}$$

Next, we generate a random deformation field. We do not target realism in stroke generation. Instead we generate a random number n of strokes and their path by uniformly sampling a random number m of points. The spread and intensity of each stroke is also random so we provide complexity and variety in the results. Finally, to further enrich the dataset with erosion-like structures, we add a set of radial strokes of small width and low strength, extending between the island centre and the exterior of

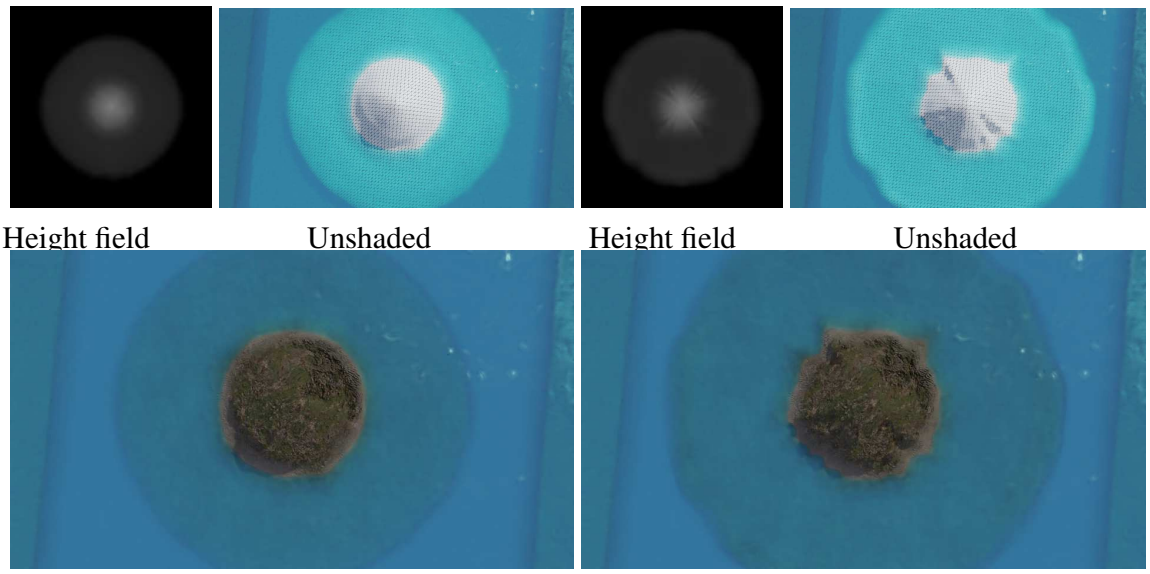


Figure 1.38: *Multiple procedural strokes are added between the island centre and the surrounding sea, alternately landward and seaward. This transforms a simple circular island (left) into a more realistic volcanic island (right) by cheaply simulating star-shaped radial drainage patterns.*

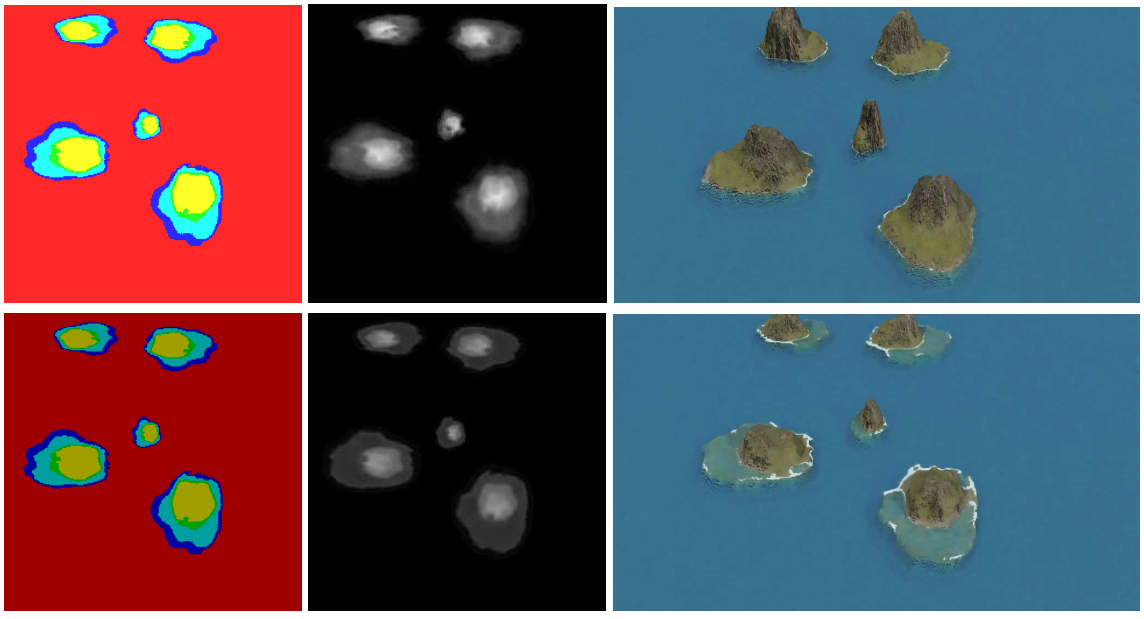


Figure 1.39: *An identical label map yields similar height fields over multiple inferences from the model, even after modifying the subsidence factor (visible in the luminosity of the input image).*

the canvas, oriented alternately landward and seaward. This produces simple radial drainage patterns illustrated in Figure 1.38.

Once all inputs are set, we generate an example for multiple levels of subsidence $\lambda \in [0, 1]$ producing height fields that incorporate coral reef modelling along with its associated label map.

The pix2pix model was originally pretrained using RGB images. In this training phase, the images were labelled using the HSV (Hue, Saturation, Value) colour space, where the Hue component specifically carried the label information. Both the Saturation and Value components were kept neutral, meaning they did not convey any significant label-related data. The target images, the ones the model

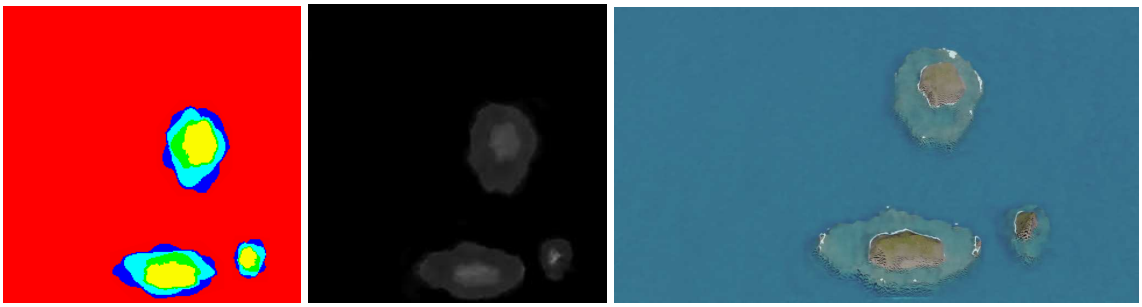
aimed to reproduce, were formatted in RGB.

For the purpose of fine-tuning the model, we continue to use the Hue component to encode the labels from the label map. We introduced a new dimension to the model's learning capabilities by incorporating the subsidence rate, denoted as λ, into the Value component. This addition not only utilises the model's existing capability to interpret the HSV format but also enriches the input data, which now carries additional, valuable environmental information.

Moreover, we purposefully left the Saturation component unchanged at this stage, reserving space for potentially including another parameter in the future, which would allow us to expand the model's utility without altering the foundational HSV encoding scheme established during its initial training. By adhering to this encoding format, we ensure continuity in data representation, which maximises the efficiency of the pretrained model. This strategic update enhances the model's adaptability and broadens its applicability to tackle new, complex challenges more effectively.

This configuration enables rapid generation of large quantity of samples, with multiple parameters, of a single island centred in the image.

1.5.1 Data augmentation



Input sketch Output height field Result

Figure 1.40: *A sample of the dataset with the input and output of the island copied and scaled at multiple positions of the image, creating three different islands in a single image.*

To enhance the variety of the dataset and improve the model's ability to generalise, we apply several data augmentation techniques in addition to the usual affine transformations (rotation, scaling, and flipping):

Translation: Since the original algorithm always centres the island, we translate the islands within the image to remove this constraint (Figure 1.40). This ensures that the cGAN can generate islands in any position within the frame. By wrapping the island on the borders (see example Figure 1.41), the model learns that data can appear on the edges of the image.

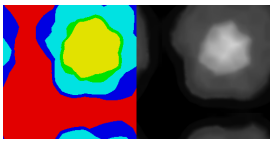


Figure 1.41: *As we translate, we wrap the height field and the label map, reducing border artefacts appearing when datasets rarely contain content on edges.*

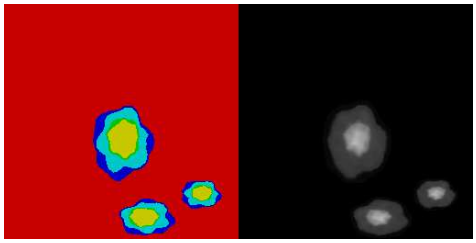


Figure 1.42: *We enforce the possibility to have multiple islands at once.*

Copy-paste: In some cases, we combine multiple islands into a single sample (Figure 1.42), with only a maximum intersection over union (IoU) of 10%, allowing the abysses to overlap but without obtaining other region types to merge. Height fields are blend using the smooth maximum function from Equation (1.15), and label blending uses the usual max operator. Regions not covered by any island are assigned the abyss ID, which correspond to the minimal height. In this case specifically, the subsidence factor (Value channel) is close to irrelevant (Figure 1.40, rightmost).

All augmentation techniques are simultaneously applied to both the height field and the label map ensuring consistency between input (the label map) and output (the height field). We summarise the complete dataset generation in Algorithm 3.

Input: Base outline radii r^*, profile h^*_{profile}, resistance ρ^*, noise η, stroke generator, subsidence grid $\{\lambda\}$
Output: Pairs $\{(\text{label map},\ \text{height field})\}$

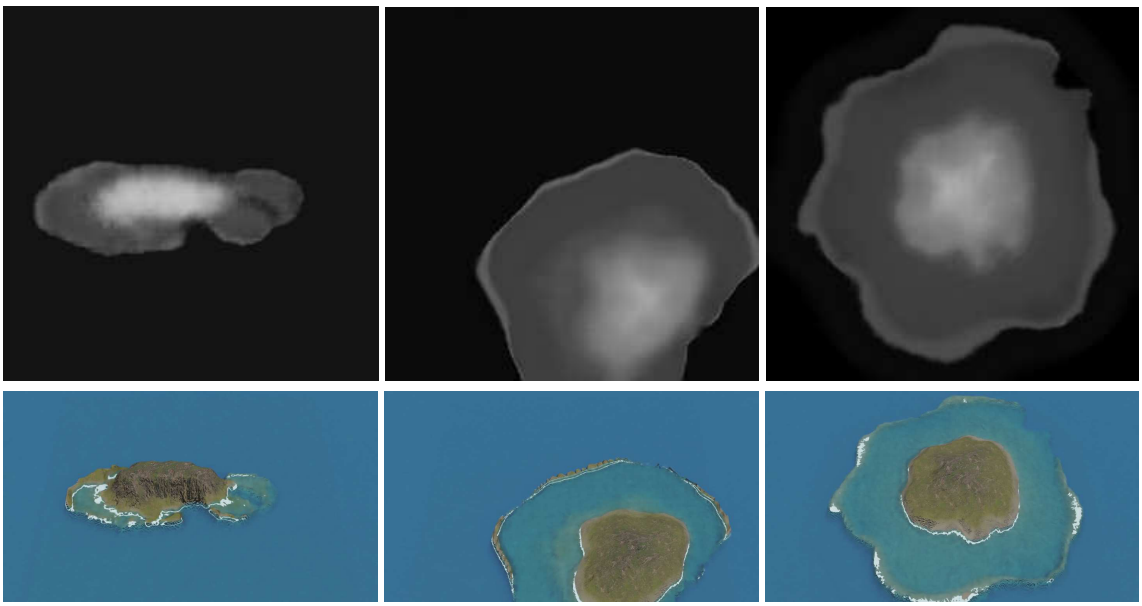
`// (1) Randomise sketches`
foreach *outline* **do**
  $r(\theta) \leftarrow r^* + \eta(\theta)$
$h_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \leftarrow h^*_{\text{profile}}(\tilde{x}_{\mathbf{p}}) \cdot \eta(\tilde{x}_{\mathbf{p}})$
$\rho(\tilde{x}_{\mathbf{p}}) \leftarrow \rho^*(\tilde{x}_{\mathbf{p}}) \cdot \eta(\tilde{x}_{\mathbf{p}})$
`// (2) Random strokes`
Generate n strokes $\{C\}$ by uniformly sampling m control points each; sample σ and S_C per stroke
`// (3) Synthesis over subsidence levels`
foreach $\lambda \in [0,1]$ **do**
  Build h and I via Algorithm 1 and apply Algorithm 2 to get $\hat{h}$
  label map $\leftarrow$ HSV image with I in the Hue channel and λ in the Value channel
  Store pair $(\text{label map},\ \hat{h})$
`// (4) Augmentation`
Apply translations with wrapping, rotations, scalings, flips to both label map and height field consistently
Copy-paste multiple islands with IoU≤ 0.1; blend heights with smax and labels with max; and assign abyss ID for uncovered regions
return dataset

Algorithm 3: Procedural dataset generation

1.6 Learning-based generation

In this section, we introduce the use of a conditional Generative Adversarial Network (cGAN), specifically the pix2pix model, to enhance the island generation process by increasing the variety and flexibility of terrains. While the initial procedural algorithm generates diverse island examples (convex and concave outlines), cGAN provides additional flexibility in generating more complex terrain without the rigid constraints of the procedural algorithm that stem from our initial assumptions based on coral reef formation theory.

Training was performed using a batch size of 1, and optimised using the Adam optimiser with a learning rate of 0.0002 and momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The loss function combined a conditional adversarial loss and an L1 loss, where the L1 loss was weighted by a factor $\lambda = 100$.



Epoch 1 (~10 000 samples) | Epoch 1.5 (~15 000 samples) | Epoch 2 (~20 000 samples)

Figure 1.43: *After the first epoch (left), grid artefacts similar to a low-resolution image are visible, but are being corrected during a second epoch (middle) where erosion patterns appear in the height fields (right).*

The generator follows a U-Net architecture with encoder-decoder layers and skip connections, while the discriminator was based on a PatchGAN, classifying local image patches. These parameters are directly adopted from the original pix2pix paper [IZZE17].

During inference, our model, with a dataset of approximately 10 000 examples representing about 30 minutes of training, still outputs grid artefacts, visible in Figure 1.43. After the second epoch, visual artefacts are already rare. This training time is relatively short compared to typical deep-learning terrain generation pipelines, largely thanks to the synthetic nature and consistency of our dataset.

Once the model is trained, the user colours a 256×256 RGB image to sketch the different regions. Each region is given a specific colour based on the HSV encoding used for the dataset generation. The examples presented in this paper use red for abysses, blue for reefs, cyan for lagoons, green for beaches, and yellow for mountains. The subsidence factor presented in Section 1.4.2 is controllable through the luminosity of the input image.

The Python script for the initial island dataset generation is unoptimised and takes about 2.5 s per island of size 256×256 as we do not perform parallelisation here. Implementing an optimised C++ version of the initial generation process reduces this execution time to 50 ms per generation.

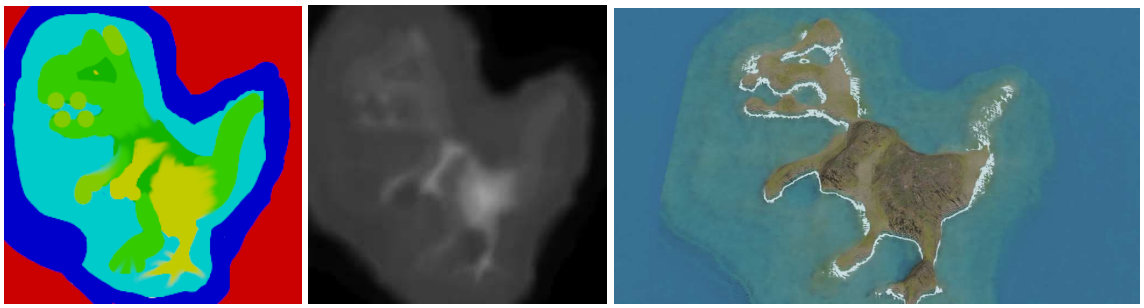
On the other hand, the inference time of our deep learning model for a single input image of dimension 256×256 remains constant regardless of scene complexity. Using the NVIDIA GeForce GTX 1650 Ti GPU with Python 3.10 and PyTorch version 2.5.1+cu121, the inference time measured is 5 ms (std 1.1 ms).

Once trained, the cGAN model generates a height field from a given label map. The output is a complete 2D height map representing the terrain's elevation at each point. Although the cGAN's internal logic is not easily interpretable, the label map remains available after inference and retains semantic terrain structure for the user.

The label map is a symbolic representation of the terrain, where each pixel of the map (each colour) represents a class. We can use this information to keep a sparse representation of the environment using medial axis transform shape descriptor, for which we will see how to treat in the next chapter

Input sketches Output height field Result

Figure 1.44: *User applied fuzzy brushes to draw the label map, resulting in some pixels that are inconsistent with the dataset and illogical island layouts: abyss regions (red) are found between beach (green), lagoon (cyan) and reef regions (blue). The neural model ignores the layout inconsistencies, and partial over-brightness as long as the pixel isn't completely white.*



Input sketches Output height field Result

Figure 1.45: *Without constraints on the generation, the user may use unrealistic layout and the neural network will however output a plausible result. Here new colours have been added (pistachio), but the network naturally considers it as a transition from mountain to beach.*

about environmental objects.

1.7 Results

The resulting model for coral reef island generation enables a high level of user control as unconstrained painting allows complex scenarios while producing height fields in real-time. In this paper we used Blender to provide renders directly from the output height fields. As our pix2pix model is trained to output 256×256 images, the resolution of the 3D models is limited by this architecture.

By using deep-learning-based models, most initial constraints are removed (radial layout, isolated islands, ...). The control over the overall shapes of the island regions is given through digital painting, here using GIMP. Each pixel of the image is encoded in HSV, with the region identifier encoded in the Hue channel. The user may increase or decrease the subsidence level of the island by modifying the Value channel over the whole image (see Figure 1.39).

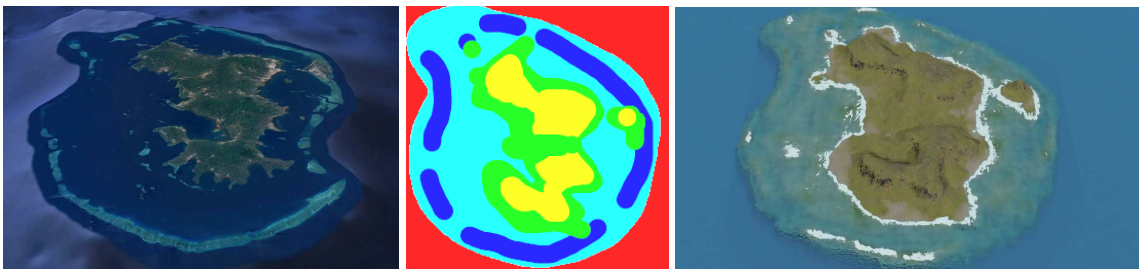


Figure 1.46: *Comparison between real (left, from Google Earth) and synthetic (right) islands of Mayotte. The label map (centre) is hand-drawn to match approximately its real-world counterpart.*

Since the model relies on pixel-level statistics rather than strict class values, it is robust to noise caused by compression, anti-aliasing from brush tools, or resizing artefacts introduced by image editors. The example displayed in Figure 1.44 (top) displays a sketch with blurry outlines and unexpected layouts (see the small red regions inside the southern lagoon region or the adjacency of beach regions directly with the abyssal region). The learned model does not include inconsistencies and results in plausible 3D models. We can note that the input label map isn't required to be hand-drawn, as a simple Perlin noise can produce it randomly, as shown in the examples of Figure 1.47. Figure 1.44 (bottom) shows highlight clipping (white pixels due to artificial oversaturation, losing the Hue value) that is not interpretable by the model and resulting in black patches in the result.

The tolerance to input values enables more control about the transitions between two regions than the procedural module. Figure 1.45 shows an example of input map with regions that are leaking over neighbouring regions, and the introduction of new hue values nonexistent in the dataset (light green and dark green) but are the interpolated hue values of mountain regions and beach regions.

Because our curve-based procedural phase included low randomness, the output of the cGAN is limiting its unpredictability, meaning that small input changes result only in minor changes on the output, preventing unexpected results. Figure 1.39 shows the result of an input map with only a variation on the subsidence level, the resulting height fields are very similar. Adding the real-time computation of outputs, it becomes possible to construct progressively a landscape and correct small mistakes to intuitively design islands inspired by real-world regions. A creation of an island similar to Mayotte, presented in Figure 1.46 was hand-made in just a few minutes.

1.7.1 Advantages of the approach

One of the main strengths of this approach is its ability to produce a wide variety of island terrains, even in the absence of real-world data. The procedural generation methods allow for high flexibility in designing both the shape and features of the island, while the use of cGAN enables further refinement and the generation of terrains not bound by the original constraints of the procedural model. By combining these two methods, we leverage the complementary advantages of both: the structured control of procedural techniques and the pattern-learning capabilities of deep learning.

A key advantage of this approach is the retention of semantic information throughout the generation process. The label map, which serves as the input to the cGAN, can also be used after terrain generation to provide a detailed representation of the different regions of the island (island, beach, lagoon, coral reef, and island body). This label map can guide post-processing operations, such as applying different textures based on terrain features or adding other environmental elements, vegetation for instance. The preservation of semantic information provides a useful connection to the next stage of terrain manipulation, making the process more versatile and adaptable to different use cases.

Furthermore, the use of an out-of-the-box cGAN model highlights the feasibility of employing existing neural network architectures with minimal modifications in the field of procedural generation. This is particularly important for domains where real-world data is scarce, such as coral reef islands, allowing

synthetic data to be effectively used for training purposes.

1.7.2 Limitations

While the cGAN model provides increased flexibility and variety in island generation, it does come with certain limitations:

Biases from the synthetic dataset: Since the cGAN model is trained entirely on a procedurally generated dataset, it inherits the biases present in the initial algorithm. For example, while the model break free from the radial symmetry constraint and centre positioning, it remains dependant on the structure and patterns of the synthetic data. This limitation reduces the true diversity of the generated terrains, as the cGAN cannot generate terrains that deviate too far from the examples in the training set.

Lack of user control: Another limitation of using cGAN in this context is the lack of real-time user control during terrain generation. While traditional procedural generation methods allow users to adjust parameters during the generation process, the cGAN model operation is abstracted from the user, providing no mechanism for direct interaction beyond the initial label map. This reduces the level of customisation available to the user.

Data-driven dependence: The quality of the generated terrain depends entirely on the quality of the training dataset. Since the dataset is synthetically generated, any limitations or biases in the initial dataset directly affect the cGAN's output. This dependence on data quality makes it crucial to design a well-augmented and varied dataset to ensure diverse and realistic outputs.

1.8 Conclusion and future works

This work has presented a novel approach to generating coral reef island terrains by combining traditional procedural methods with deep learning techniques. We first developed a procedural generation algorithm capable of creating a wide variety of island terrains through a combination of top-view and profile-view sketches, wind deformation, and subsidence and coral reef growth modelling. By applying these methods, we produced plausible terrains based on geological processes, capturing key features of coral reef islands: island, beaches, lagoons, and coral reefs.

To further enhance flexibility and realism in the generation process, we incorporated a Conditional Generative Adversarial Network (cGAN), using the pix2pix model to generate height maps from label maps of island features. The cGAN model allowed us to overcome some of the constraints inherent in the procedural algorithm, such as radial symmetry and fixed island positioning. Through data augmentation, we trained the cGAN on a synthetic dataset, enabling the generation of varied and plausible island terrains.

There are several directions for future research and improvements. One promising avenue is to incorporate the wind velocity field more directly into the cGAN training process, potentially as an additional input condition. This would allow the model to better capture wind-driven terrain features such as cliffs or other deformations influenced by wind patterns.

Another area for exploration is improving user interaction during the terrain generation process. While the current model allows for rapid terrain generation, adding more options for users to interact with the cGAN, such as adjusting parameters (wind strength and main direction, erosion, reef function parameters, ...), could enhance the flexibility of our system.

Finally, further improvements could be made to the synthetic dataset. Incorporating more complex geological processes, such as higher fidelity physical simulation of wave and rain erosion, and tidal influences, could lead to even more realistic terrains at the cost of interactivity. Additionally, refining the way islands are blended in multi-island samples, or adding more diverse input conditions (e.g.,

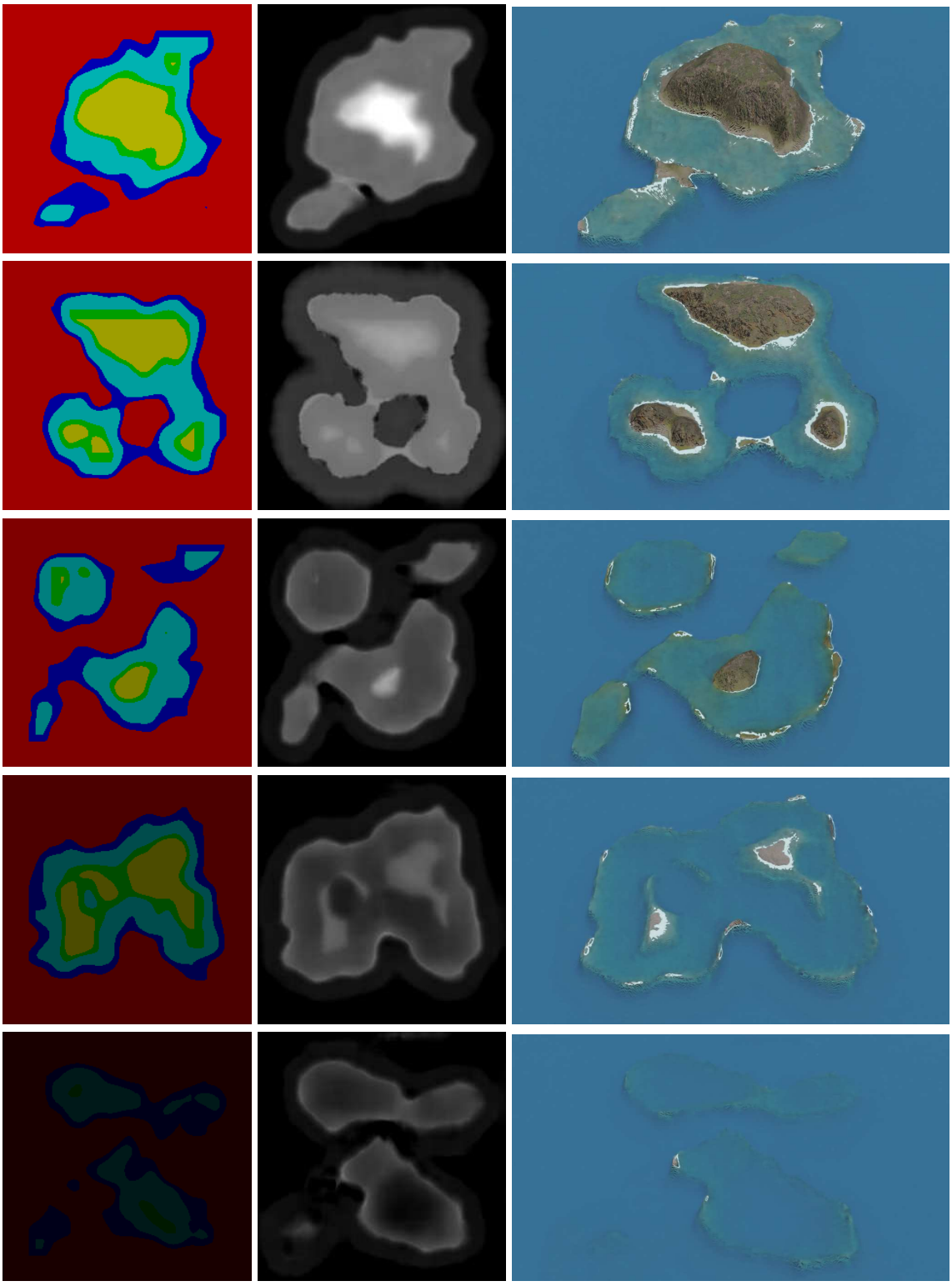


Figure 1.47: *Starting from random Perlin noise, transformed into a label map, we can generate a large variety of results.*

different geological settings), could help the model generalise better and produce more varied and dynamic landscapes.

Many other neural network models could be exploited to increase the possibilities, with newer variants of cGANs [PLWZ19], or models with style transfer functionalities [GEB15, ZCZ*20] in order to change the overall aspect of a terrain [PPB*23, PPB*23], use text-to-image models [RBL*21, RKH*21] to generate height fields from a verbal prompt, or super-resolution models [DLHT14] to increase the definition of details in the final output [GDGP16].

In summary, this chapter has demonstrated how procedural rules and deep learning can produce convincing island terrains. Yet, terrains alone are not enough: seascapes also depend on the biotic and abiotic features that inhabit them. The following chapter therefore moves beyond geometry to explore ecosystem generation.