EUROGRAPHICS 2025 / M. Christie, N. Pietroni, and Y.-S. Wang
(Editors)

Volume 0 (2025), Number 0

Procedural and learning-based generation of coral reef islands

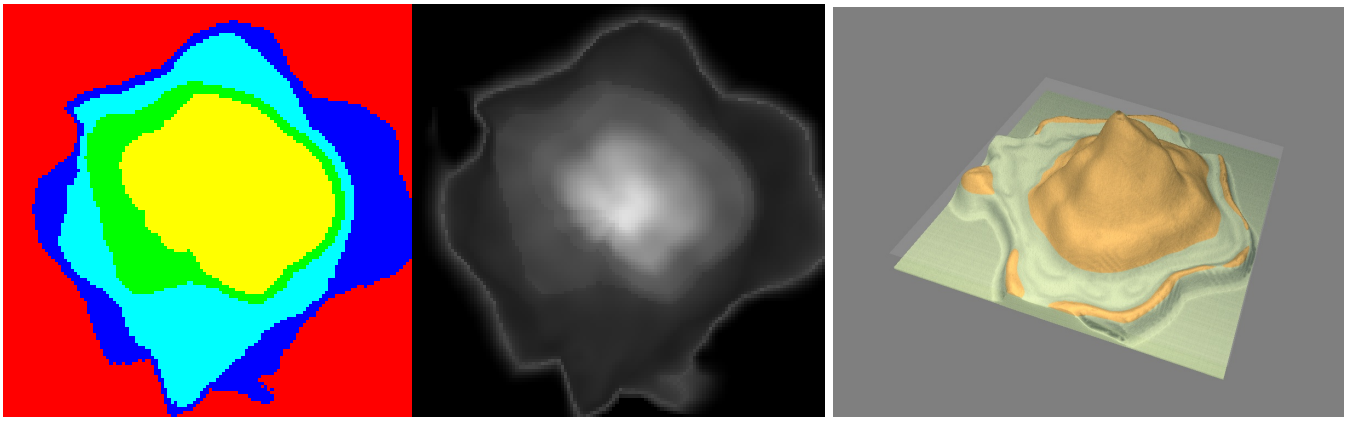

Figure 1: *From top-view and profile-view sketches, our system procedurally generates coral reef islands and trains a cGAN to produce diverse and realistic terrains.*

Abstract

We propose a procedural method for generating single circular volcanic islands with coral reefs using user sketching from two projections: a top view, which defines the island's shape, and a profile view, which outlines its elevation. These projections, commonly used in geological and remote sensing domains, are complemented by a user-defined wind field, applied as a distortion field to deform the island's shape, mimicking the effects of wind and waves on the long term. We then model the growth of coral on the island and its surronding to construct the reef following biological observations. Based on these inputs, our method generates a height field of the island. Our method is capable of creating a large variety of island models composing a dataset used for training a conditional Generative Adversarial Network (cGAN). By applying data augmentation, the cGAN allows for even greater variety in the generated islands, providing users with higher freedom and intuitive controls over the shape and structure of the final output.

Keywords: Procedural modeling, Terrain synthesis, cGAN, Coral reef, Sketch-based interface

1. Introduction

Simulating the formation of coral reef islands presents significant challenges due to the complex interplay of geological, environmental, and biological factors Hopley, 2014. One major difficulty lies in capturing the long-term subsidence of volcanic islands, which occurs over millions of years, while simultaneously modeling the upward growth of coral reefs that rely on environmental conditions such as water depth, temperature, and sunlight. This combination of slow geological processes and dynamic biological growth is difficult to replicate in a computational model.

Additionally, the biological aspects of coral growth are inherently tied to environmental factors. Coral reefs grow only within a specific range of water depth and sunlight, and their growth patterns are affected by the health of the reef ecosystem and the availability of resources. Accurately modeling these biological dependencies in a procedural system is challenging, as these factors are numerous and difficult to generalize. Moreover, the scarcity of data available obstructs the global understanding of these biomes. In a recent high-resolution mapping of shallow coral reefs Lyons et al., 2024, 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 processes. While they can generate natural-looking landscapes (such as alpine landscape, representing about a quarter of land area Körner et al., 2014), they do not account for the unique geological and biological interactions that govern coral reef island formation, thus missing coherency. Capturing these dynamics, while also providing user control during the modeling 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 can fail to produce convincing or diverse coral reef island landscapes. On the other hand, the use of deep learning methods are inoperable due to the extremly small amount of data, and the scarcity of high resolution DEM of these regions.

Despite advances in terrain generation, existing methods struggle with 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 issues, we use procedural generation as an initial step in our approach. Procedural generation employs algorithmic rules to synthesize terrain features, allowing us to encode basic patterns of coral reef island formation. 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 can produce 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.

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 like ocean, reef, beach, and mountain) generated by the procedural step and learns to produce a realistic island height field that matches this layout. By training on the many examples from the procedural generator, the cGAN captures the subtle terrain features and variations characteristic of 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, the procedural generation module is only needed to provide training data during the learning phase; it is not required for producing new islands. Instead, a user can supply a fresh semantic map through digital drawing or another simple algorithm, and the cGAN will generate a realistic island terrain accordingly. In short, our pipeline leverages procedural modeling to create a training dataset, and then relies on the learned cGAN model for the final generation of coral reef islands.

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 training of a deep learning model on synthetic data derived from procedural rules, serving 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 such as coral reef islands.

These contributions collectively show a pathway to blend user-driven design with learning-based generation in terrain modeling.

2. Overview of our method

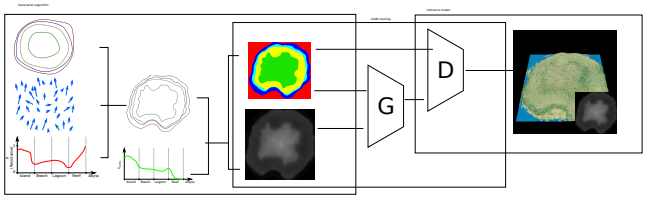

Figure 2: *Our method is split in three interleaved stages: the generation process (Section 3) which creates pairs of height fields and label maps of an island from sketches, the model training (**??**) which use a synthetic dataset from the previous stage to obtain a cGAN model that generates height fields from label maps to remove the constraints embedded in the initial generation process, and finally, the inference process (**??**) uses the trained cGAN to generate the final height fields, including the coral generation process, automatically.*

Our method for generating coral reef islands combines user-driven sketching, procedural techniques, and deep learning to create realistic and varied island terrains (Figure 2).

The pipeline consists of two distinct phases: a procedural data-generation phase and a deep-learning-driven inference phase.

2.1. Procedural generation phase

In the initial procedural phase, the user sketches key island features from two complementary viewpoints: a top view, defining the horizontal layout of island features (island boundaries, beach width, lagoon areas, coral reefs), and a profile view, specifying the vertical elevation profile from island center to ocean (Section 3.1).

Additionally, users can sketch a wind deformation map, enabling simulation of natural erosion patterns caused by wind and waves (Section 3.1.1).

From these sketches, the procedural system generates a synthetic island terrain with the keep-up stategy of coral reefs (Section 3.2) and a corresponding semantic label map, where each pixel indicates its region type (island, beach, lagoon, reef, abyss) (Section 3.2.4).

User interaction

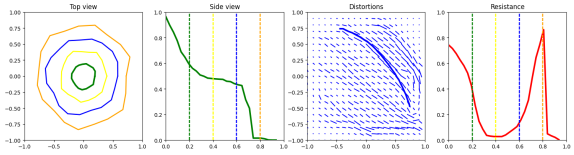

Figure 3: *The user can interact directly on the island by editing the different canvases in no specific order. This UI shows, from left to right, the top-view sketch with the different outlines of each regions, the profile-view sketch with the outlines represented in dotted lines, the wind velocity 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.*

As users draw the top-view and profile-view sketches, the system provides real-time feedback on the resulting terrain. The top-view sketch influences the horizontal layout of the island, while the profile-view sketch defines its vertical structure. These sketches can be adjusted independently, allowing the user to fine-tune both the outline and elevation of the island.

While sketching the basic shape, users can apply wind deformation strokes to modify the island's features further. These strokes represent wind and wave influences, distorting the island's shape to introduce more natural, non-radial features such as indentations along the coastline, variable lagoon shapes, or concave formations. The system automatically applies these deformations, providing real-time feedback as the user interacts with the terrain.

This interactive process, combining sketches and wind deformation, allows users to quickly iterate on their designs, refining the terrain to meet specific aesthetic or functional goals.

2.2. Learning-based generation phase

We repeat this procedural generation process many times with varied parameters (different shapes, scales, subsidence levels, and wind patterns) to create a large synthetic dataset (Section 4.1). Each dataset entry consists of a label map paired with its procedurally generated terrain height field. Data augmentation is applied to the generated pairs to reduce the impact of the constraints induced from the procedural method (Section 4.2).

We use this dataset to train a Conditional Generative Adversarial Network (cGAN), specifically the pix2pix architecture, capable of translating label semantic maps into realistic terrain height fields (**??**).

After training, the procedural step becomes unnecessary. To generate new island terrains, the user only needs to provide a label semantic map as input to the trained cGAN. The cGAN then synthesizes realistic island elevation details directly, capturing learned geological and geomorphological patterns from the synthetic training data (**??**).

User interaction

Thus, the trained cGAN provides a user-friendly interface: users draw or edit simple label maps (regions) to rapidly generate diverse, geologically plausible coral reef island terrains, incorporating realistic features such as smooth transitions between regions, detailed coral reef structures, and naturally varied shapes free from procedural constraints.

This combined procedural-and-learning approach provides a simple, flexible, and powerful tool for island terrain generation, enabling users to intuitively generate realistic and diverse coral reef islands aligned with real-world geological and biological processes such as volcanic subsidence, coral reef growth, and wind-driven erosion.

3. Procedural terrain generation

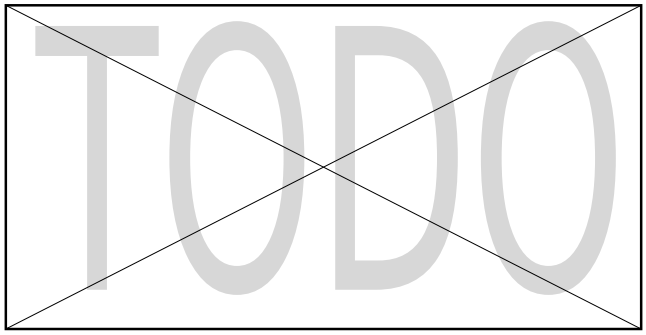

Figure 4: *The example generation fully takes its potential in the procedural techniques, using sketches from the user (top-view sketch, profile-view sketch, wind sketch, and resistance sketch) to generate a height field in accordance with a label map.*

The generation of coral reef island terrains involves a structured process that takes the user's sketches and produces a complete 3D terrain model. This process begins with the creation of the initial height field based on the user's input, followed by the application of wind deformation to introduce natural variations, and concludes with the integration of coral reef features through subsidence and coral growth modeling.

The generation of coral reef islands in this system begins with two intuitive sketch-based inputs from the user: a top-view sketch and a profile-view sketch, which define the islands horizontal layout and vertical elevation profile. In addition to these sketches, the user can further refine the terrain by applying wind deformation strokes, which simulate the effects of wind and waves on the islands shape. This combination of sketches and wind inputs gives

users precise control over both the islands structure and its natural variations, such as irregular coastlines or concave features. We will present the usefulness of these sketches in this section, and describe the technical details in the next section.

3.1. Initial height field generation

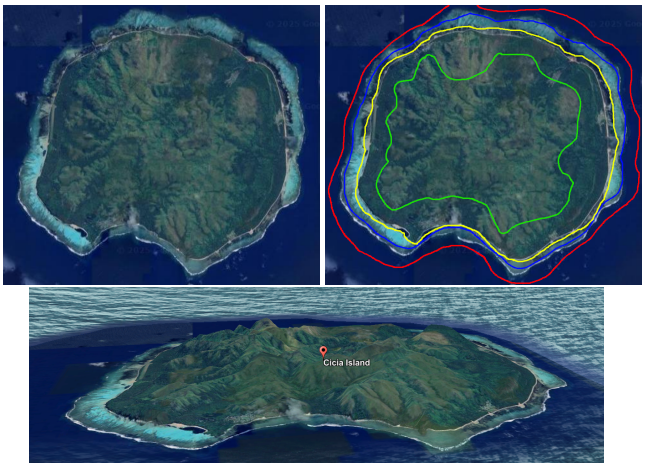

Figure 5: *(Left) A real world example of aerial image (and 3D visualization on bottom) of an island (Cicia Island) may be segmented in regions. (Right) We can represent the different regions by the boundaries they form.*

The top-view sketch defines the islands 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. The system assumes that these regions are arranged concentrically around the center of the island, with each boundary defined by a radial distance from the center.

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

In this sketch, the user defines the overall horizontal layout of the island, including the size and shape of each feature. 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.

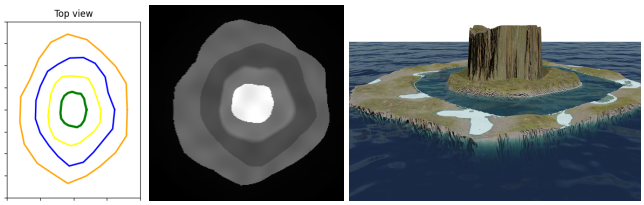

Figure 6: *Using only the outlines of the island as a input sketch, we can provide a height to each point of the field depending on the region in which it rely.*

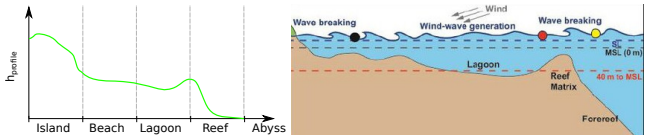

Figure 7: *(Left) A profile function $h_{profile}$ is defined as a 1D function and represents the surface from the center of the island to the abysses. (Right) The cross-section representation of an island is often represented as a 1D function defined using terrain features as landmarks.*

The profile-view sketch defines the vertical elevation profile of the island along any radial direction, offering control over the islands 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.

The regions outlines correspond to key terrain transitions: the highest point of the island (center), the island border, the beach, the lagoon, and the deep-sea abyss. The system uses these milestones to interpolate a continuous 1D height function $h_{\text{profile}}(\tilde{x})$, where $\tilde{x}$ represents a non-uniform region distance from the islands center, and $h = h_{\text{profile}}(\tilde{x})$ 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, the system can generate a full 3D terrain model that accurately reflects the users design by revolution modeling.

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, such as the island, beaches, lagoons, and abyss.

For any point $\mathbf{p}$ on the terrain, the system first computes the polar coordinates $(r_{\mathbf{p}}, \theta_{\mathbf{p}})$, where $r_{\mathbf{p}}$ is the radial distance from the island's center, and $\theta_{\mathbf{p}}$ is the angular component. The radial distance $r_{\mathbf{p}}$ is used to determine which region the point belongs to (island, beach, lagoon, reef, or abyss). The user-defined outlines in the profile sketch specify the radial limits between these regions.

Each point's height is determined by the profile function $h_{\text{profile}}(\tilde{x})$, where $\tilde{x}$ represents a "piecewise parametric distance" from the island's center. The piecewise parametric distance works by dividing the radial distance from the center into segments, defined by these region boundaries. Each segment corresponds to a distinct region of the terrain, and within each segment, the distance $\tilde{x}$ is interpolated between the region boundaries. For a point $\mathbf{p}$ lying between two boundaries R_i and R_{i+1}, the distance $\tilde{x}_\mathbf{p}$ is calculated as:

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

where i is the index of the nearest lower region boundary. This method allows for smooth transitions between regions, even when the spacing between boundaries varies.

For any point $\mathbf{p}$, the height is finally computed as:

$$h(\mathbf{p}) = h_{\text{profile}}(\tilde{x}_\mathbf{p}) \tag{2}$$

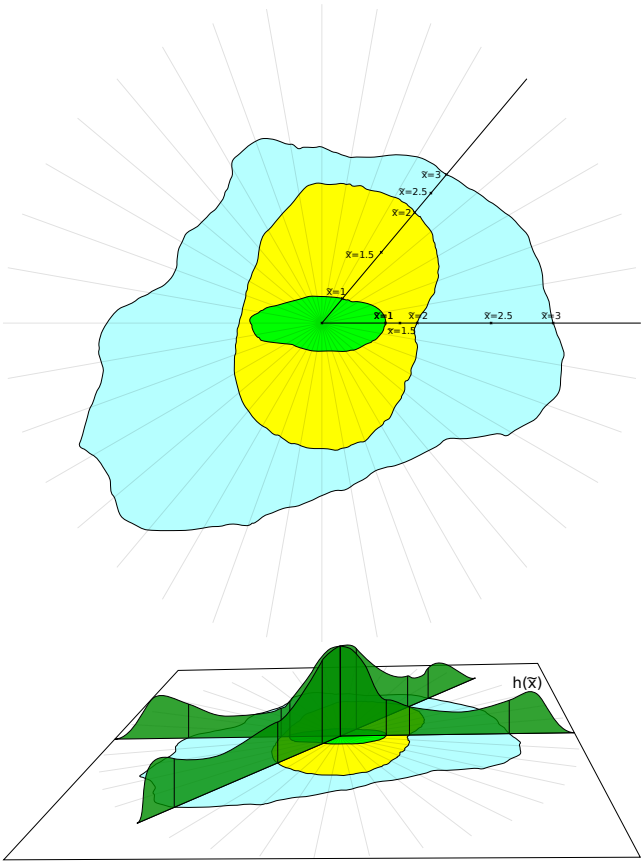

Figure 8: *The $\tilde{x}$ parameter is used to stretch the 1D height function $h_{profile}(x)$ to fit the distances from the center to the outlines of each region defined in the top-view sketch.*

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 a realistic representation of islands with smooth transitions between the key terrain features.

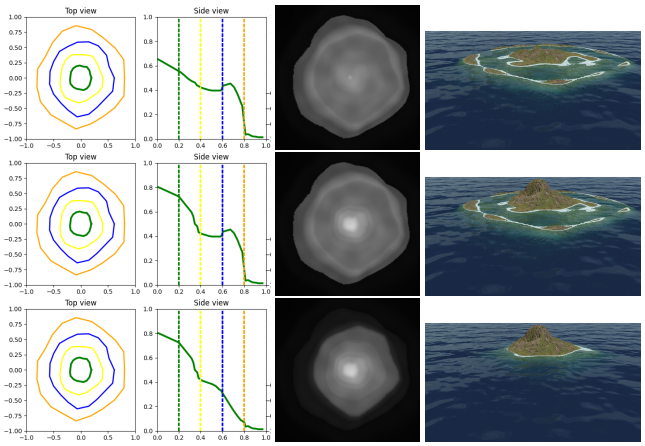

Figure 9: *Providing a smooth function between each region results in islands with plausible reliefs. We fixed the outlines while editing only the height function in order to produce, from top to bottom, a low island, a coral reef island, and finally an identical island without the reef.*

3.1.1. Wind deformation

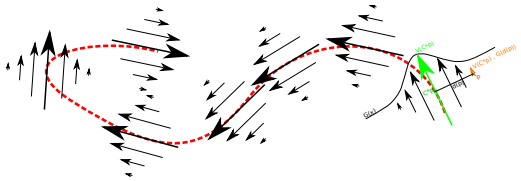

Figure 10: *From the parametric curve defined by a user (red), we define the velocity field by considering the velocity (first derivative) of the curve at the closest point $\mathbf{p}_C^*$, modulated by a gaussian distance function $G(x)$.*

In addition to the sketches, the user can influence the shape of the island by defining a wind velocity field. This field simulates the effects of wind and wave erosion on the island's surface, introducing natural deformations such as coastline indentations, and more importantly allow the user to break the radial symmetry constraint.

The wind field is represented as a series of wind 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 these curves, and the system interprets the strokes to create a velocity field that defines how the terrain should be deformed.

As the user draws a wind stroke, the system generates a set of control points 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 with distance from the wind curve using a Gaussian falloff function using the stroke width as standard deviation, ensuring that the terrain transitions smoothly from deformed regions to non-deformed areas. Once the wind strokes are applied, the system processes the wind velocity field by displacing the terrain points 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.

After generating the initial height field based on the top-view and profile-view sketches, the next step in the process introduces wind deformation. This step simulates the long-term effects of wind and wave erosion, breaking the radial symmetry of the terrain and adding natural variations such as concave coastlines and irregular island shapes.

The wind deformation can be controlled through a user-defined vector field, which represents the direction and strength of wind flows across the terrain. Users interact with the system by drawing strokes on a 2D canvas, which are then interpreted as parametric curves C representing wind patterns. Each stroke defines a wind flow in the curve's direction C', a strength S, and an effect width σ; these wind flows are used to displace the terrain, simulating the gradual reshaping of the island due to wind and wave erosion.

The strokes are represented as Catmull-Rom splines, a type of parametric curve that allows for smooth, continuous wind paths. For any point $\mathbf{p}$ on the terrain, the deformation vector $\Phi(\mathbf{p})$ is calculated based on the proximity of $\mathbf{p}$ to the nearest wind strokes. The strength of the displacement is controlled by a Gaussian scaling function, which ensures that points closer to the wind strokes experience stronger displacement, while points farther away are less affected.

The displacement function $\Phi(\mathbf{p})$ is computed as a sum of the influences from all nearby wind strokes. For each stroke, the deformation vector is scaled by a Gaussian function that smoothly decreases with the distance from $\mathbf{p}_C^*$ the closest point on the parametric curve C, as follows:

$$\Phi(\mathbf{p}) = \sum_{C \in \text{curves}} S \frac{C'(\mathbf{q})}{\|C'(\mathbf{q})\|} \cdot G_\sigma\left(\|\mathbf{p} - \mathbf{p}_C^*\|\right) \tag{3}$$

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{4}$$

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 can then compute the final height $h(\Phi \circ \mathbf{p}) = h_{\text{profile}}(t_\mathbf{p})$, or, as the implicit modeling community would write it,

$$\tilde{h} = \Phi^{-1} \circ h \tag{5}$$

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.

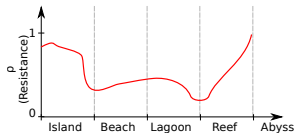

Figure 11: *The resistance function of the island is defined in the same way than the $h_{profile}$ function. The resistance to erosion and deformation arise from multiple factors such as depth, materials, wind shadowing, biotic and abiotic factors, ... Modeling all these factors is complex. As such, using a user-defined approximation through a resistance function ρ allows for more control.*

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

The resistance function $\rho(\tilde{x})$ is defined similarly to the profile function, and it controls the magnitude of the displacement at each point. For example, regions near the coastline (such as the beach and lagoon) might have lower resistance, allowing for more significant deformation (simulating 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 scaled by the resistance function at each point $\mathbf{p}$, such that the final deformation vector becomes:

$$\tilde{\Phi}(\mathbf{p}) = (1 - \rho(\tilde{x}_\mathbf{p})) \cdot \Phi(\mathbf{p}) \tag{6}$$

This ensures that the wind deformation has the greatest impact on areas like the coastline and beach, where erosion naturally plays a larger role, while deeper regions like the abyss or stronger regions like mountains remain stable and relatively unchanged.

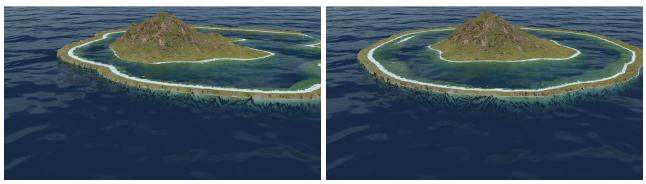

Figure 12: *(Left) Given a uniform wind velocity field and a resistance function similar as Figure 11, the coasts are smoothly eroded while the interior of the island is almost unaffected. (Right) Modifying the resistance function to affect a strong resistance to borders simulate the effect of coast reinforcements.*

The wind deformation process results in a modified height field where the terrain has been warped according to the user-defined

wind strokes. 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 wind deformation. 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.

For instance, consider a simple circular island generated from the initial height field. By applying wind 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.

3.2. Coral reef modeling

Once the terrain has been generated and deformed by the wind, the system simulates the subsidence of the volcanic island and parallely 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.

3.2.1. Subsidence

The subsidence of the island is modeled by scaling the initial height field downward, simulating the effect of the volcanic island slowly sinking into the ocean. The user provides a subsidence rate λ, which represents the proportion by which the island has sunk over time. The subsidence is applied uniformly to the terrain, meaning all 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 $h_0(\mathbf{p})$ with the subsidence factor $\lambda \in [0,1]$:

$$h_{\text{subsid}}(\mathbf{p}) = (1-\lambda)\cdot h_0(\mathbf{p}) \tag{7}$$

This scaling reduces the overall height of the island, simulating 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.

3.2.2. Coral reef growth

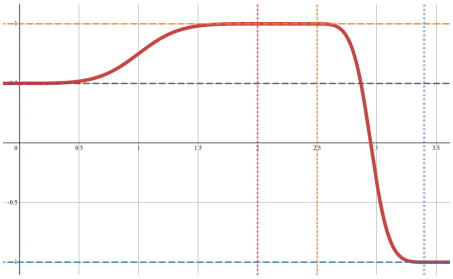

Figure 13: *The modeling of the reef growth in our model is described by a piecewise function h_{coral} which is flat in the lagoon, the crest and abyss, and follows a smoothstep function as transitions for the backreef and fore reef regions.*

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 meters below before being much scarcier.

The coral reef features (reef crest, back reef, and fore reef) are modeled separately from the subsidence process. The system generates 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.

The coral reef height field is generated using depth values specific for the various coral regions:

- The reef crest is modeled near the water surface, typically just below sea level, at $h_{\text{crest}} = -2\text{m}$,
- The back reef and lagoon are slightly deeper but remain within the range where corals can grow at $h_{\text{back}} = -20\text{m}$,
- The fore reef slopes downward into the deep ocean, transitioning into the abyss with $h_{\text{abyss}} = -100\text{m}$.

In our coral growth model, we interpolate between the different regions by applying a smoothstep operator smoothstep : $x \in \mathbb{R}$ defined as:

$$\text{smoothstep}(x) = 3x^2 - 2x^3 \tag{8}$$

For conciseness, we will note the interpolating function of a value from a to b for x clamped between x_0 and x_1 $S(a,b,x_0,x_1,x) = a + (b-a)\,\text{smoothstep}\left(\frac{x-x_0}{x_1-x_0}\right)$.

We arbitrarily define the delimitation of each of the subregions of the reef with $(x_{i|0}, x_{i|1}) \in [0,1]^2, x_{i|0} < x_{i|1}$ with $x_i = 0$ signifying the begining of the reef and $x_i = 1$ the end of the reef:

- The back reef is defined from $x_{\text{back}|0} = 0.5, x_{\text{back}|1} = 0.5$,
- The reef crest spans between $x_{\text{crest}|0} = 0.75, x_{\text{crest}|1} = 0.8$,
- And the abyss is defined at $x_{\text{abyss}|0} = 1$,

Any other subregion is defined as the transition area between two subregions.

We obtain the piecewise function as shown in Figure 13:

$$h_{\text{coral}}(x) = \sum_{r \in \text{subregions}} \begin{cases} h_r & \text{if } x_{r|0} \le x \le x_{r|1} \\ 0 & \text{otherwise} \end{cases} + \sum_{t \in \text{transitions}} \begin{cases} S(h_t, h_{t+1}, x_{t|1}, x_{t+1|0}, x) & \text{if } x_{t|1} < x < x_{t+1|0} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

3.2.3. Blending height fields

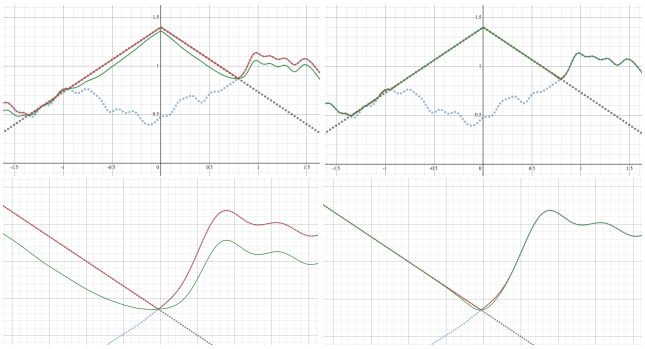

Figure 14: *Blending two functions $f : \mathbb{R} \mapsto \mathbb{R}$ (black) and $g : \mathbb{R} \mapsto \mathbb{R}$ (blue) with the* max *operator (red), causing a discontinuity, and with the* smax *operator (green), resolving the issue at the cost of slight underestimations with low values of k. Left: $k = 5$, right: $k = 50$*

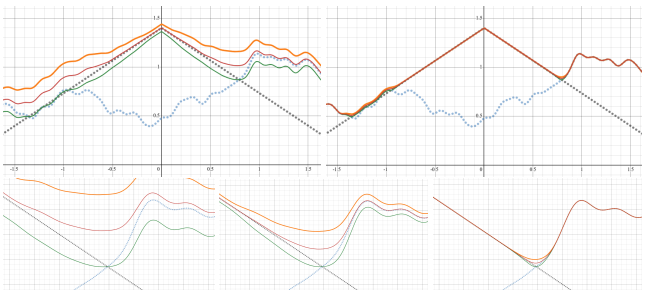

Figure 15: *The* smax^+ *operator(orange) is a function that use to overestimate the maximum value of two functions, especially when the difference between the two functions is small, while the* smax^- *function (green) tends to underestimate the* max *operator. Taking* smax *(red) as the average of* smax^- *and* smax^+ *creates a much more precise blending, even with lower values of k (Left: $k = 5$, center: $k = 10$, right: $k = 50$).*

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, the system uses a smooth max function, which smoothly blends the two height fields. The smooth max function ensures that the coral regions dominate 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 max function $\text{smax} : a, b \in \mathbb{R}^2$ as the mean of two functions, smax^- and smax^+, adapted from Ingo Quilez's smooth min function, that respectively underestimate and overestimate the function max:

$$\text{smax}^-(a,b) = a + \frac{b-a}{1+\exp(-k\cdot(b-a))} \tag{10}$$

$$\text{smax}^+(a,b) = a + \frac{b-a}{1-\exp(-k\cdot(b-a))} \tag{11}$$

$$\text{smax}(a,b) = a + \frac{\text{smax}^-(a,b) + \text{smax}^+(a,b)}{2} \tag{12}$$

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. Higher values of k brings the smax function closer to the max function (Figure 14).

This smooth max function guarantees visual continuity by preventing abrupt height differences between the 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.

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

$$\text{smax}(a,b) = \begin{cases} a + \dfrac{1}{2k} & \text{if } a = b, \\ \dfrac{\text{smax}^-(a,b) + \text{smax}^+(a,b)}{2} & \text{otherwise} \end{cases} \tag{13}$$

3.2.4. 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 like the island, lagoon, and coral reefs.

One of the key strengths of this method is its flexibility as the subsidence and coral reef growth processes are modeled 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 simulation to existing height fields from other sources.

4. Training the cGAN

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 can create numerous island examples, 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.

4.1. Dataset generation

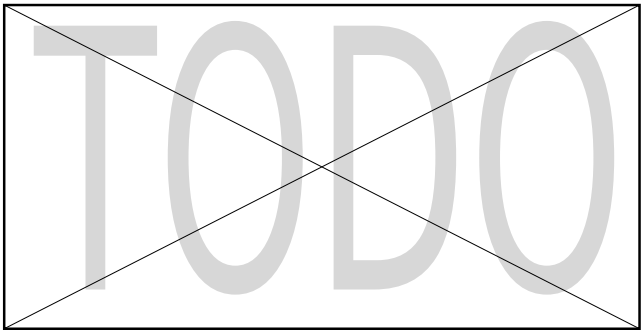

Figure 16: *Using a large set of pairs of height field-label map, the training of a deep learning model result in a user-friendly interface requiring solely a hand-drawn label map to produce a 2.5D height field of the desired island.*

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 centered 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

$$r(\theta) = r^* + \eta(\theta) \tag{14}$$

On the other hand, we define an initial profile-view sketch by defining $h^*_{\text{profile}}(\tilde{x})$ the initial height function for which fBm noise is applied to obtain

$$h_{\text{profile}}(\tilde{x}) = h^*_{\text{profile}}(\tilde{x}) \cdot \eta(\tilde{x}) \tag{15}$$

An identical process is done for the resistance function:

$$\rho(\tilde{x}) = \rho^*(\tilde{x}) \cdot \eta(\tilde{x}) \tag{16}$$

Finally, we need to generate a random wind field. The realistic nature of wind is ignored for the generation of the wind strokes in order to provide complexity and variety in the results. We generate a random number n of strokes and their path by a uniformly sampling a random number m of points. The spread and intensity of each stroke is also random.

Once all inputs are set, we generate an example for multiple level of subsidence $\lambda \in [0,1]$ to obtain a height field incorporating the coral reef modeling and the associated label map.

The Pix2pix model was originally pretrained using RGB images. In this training phase, the images were label using the HSV (Hue, Saturation, Value) color 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 retained the use of 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 utilizes 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 maximizes 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 allows the process to create quickly a large quantity of data, with multiple parameters, of a single island centered in the image.

4.2. Data augmentation

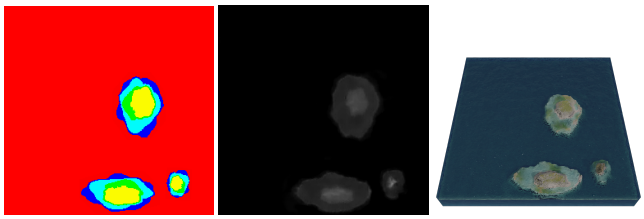

Figure 17: *By applying our three data augmentation functions, the deep learning model learns to overcome some constraints previously set by the initial algorithm: (A) the translation removes the constraint to have an island ultimately at the center of the map, (B) the directional scaling, typical from image processing, reduces the symmety constraint on the results and (C) the copy-paste unlock the possibility to obtain more than one island per map.*

To enhance the variety of the dataset and improve the model's ability to generalize, we apply several data augmentation techniques:

Translation: Since the original algorithm always centers the island, we translate the islands within the image to remove this constraint (Figure 18). This ensures that the cGAN can generate islands in any position within the frame.

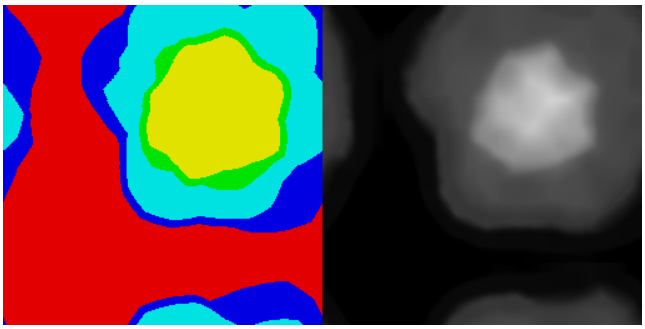

Figure 18:

Directional scaling: By scaling the terrain in one direction, we create elongated islands that resemble corridors or archipelagos, adding another layer of diversity to the dataset. Such islands are usually found on tectonic plates convergence boundaries, creating island arcs with high density of volcanic centers like the Izu-Bonin-Mariana arc system (Figure 19 shows an example of elongated island).

Copy-paste: In some cases, we combine multiple islands into a single sample, ensuring they do not overlap. The regions not covered by any island are assigned the abyss ID. Although this approach ensures non-overlapping regions, future work could explore using blending techniques to position islands more closely without the risk of overlap (Figure 20).

All augmentation techniques are applied both to the height field and the label map simultaneously to ensure consistency between the input (the label map) and the output (the height field).

5. Results and evaluation

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

5.1. Control

Using deep-learning-based models, most constraints from our initial assumptions are lifted (radial layout, isolated islands, ...). The control over the overall shapes of the islands regions are given through digital painting, here using the GIMP software. Each pixel of the image are 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 Saturation channel over the whole image (see Figure 21).

Since the model is based on statistics over the pixel values instead of hard values, users are not limited to a finite number of region identifiers, meaning that the output is more or less robust to noise (due to image compression, for example) and to the fuzzy values resulting from anti-aliasing of brushes often set by default, or resizing algorithm, by image editors, or even due to compression algorithm. The example displayed in Figure 22 presents a sketch for which the outlines of the regions are at the same time blurry and with layouts that are not expected (such as the small red regions inside the southern lagoon region or the adjacency of beach regions directly with the abyssal region) on the top figure and oversaturated on the bottom figure. The learned model does not include inconsistancies and results in plausible 3D models.

The tolerence over the input values may be used to provide even more control about the transitions between two regions. Figure 23 shows an example of input map with regions that are leaking over

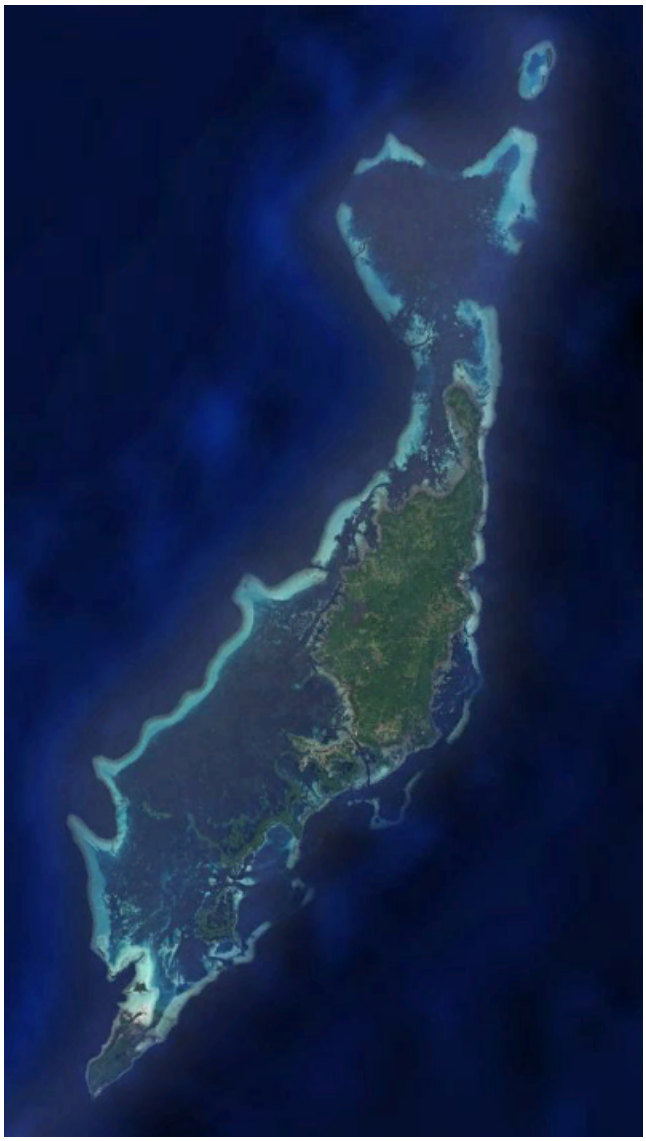

Figure 19: *Babeldaob Island, in the Caroline Islands.*

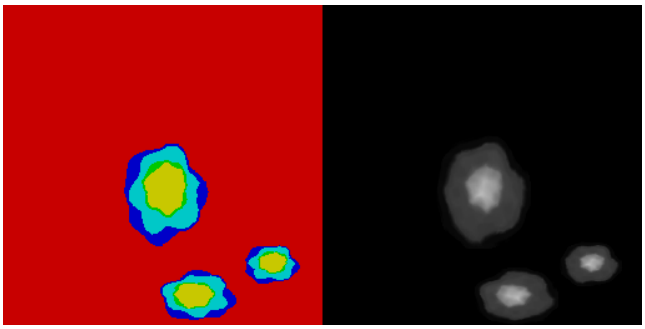

Figure 20:

neighboring regions, and the introduction of new hue values non-existant in the dataset (light green and dark green) but are the interpolated hue value of mountain regions and beach regions.

Since the procedural phase included low randomness, the output of the cGAN is limiting its inpredictibility and the results to a slight change on the input create only slight changes on the output, preventing unexpected results. Figure 21 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 (see an reproduction of Mayotte in Figure 24).

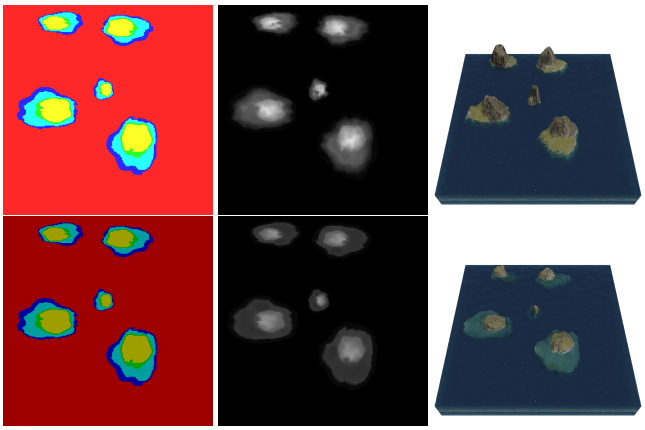

Figure 21: *An identical label map yield similar height fields over multiple inferences from the model, even after modifying the subsidence factor (visible in the luminosity of the input image).*

5.2. Performances

The Python script for the initial island dataset generation is poorly optimized and takes about 2.5s per island of size 256×256 as the parallelization does not take place here. Implementing an optimized C++ version of the initial generation process reduces this execution time to 50ms per generation.

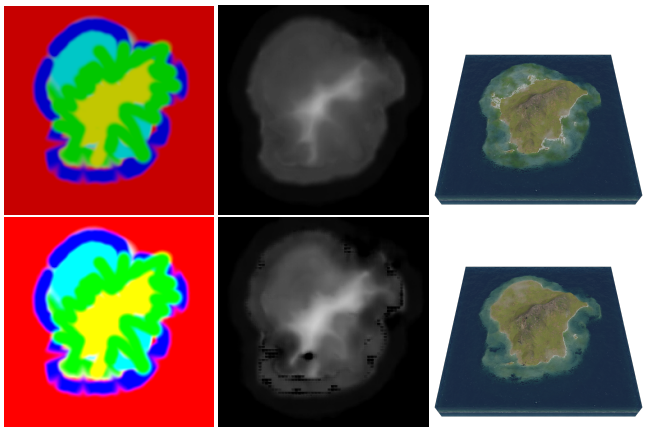

Figure 22: *Using a generative neural network allows a higher level of tolerence on the user input. Here the user used an fuzzy brush to draw the label map, resulting in some pixels that are inconsistent with the dataset and unlogical island layouts (some small "abyss" regions [red] are found between "beach" [green], "lagoon" [cyan] and "reef" [blue]). The model ignores the inconsistencies even for over-saturated pixels.*

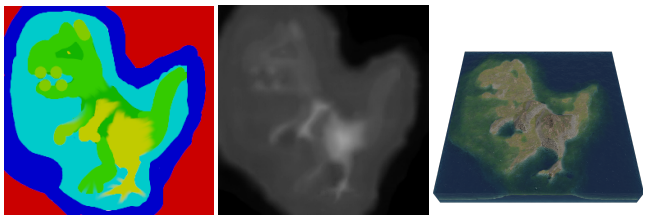

Figure 23: *Without constraints on the generation, the user may use unrealistic layout and the neural network will however output a plausible result.*

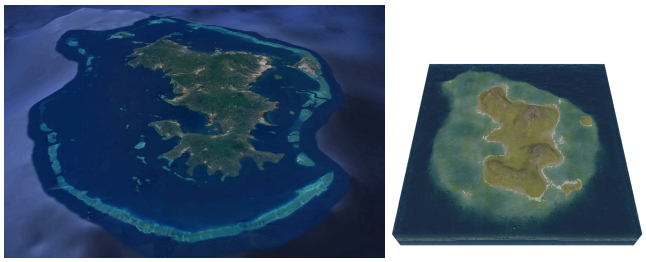

Figure 24: *Comparison between of real (left) and synthetic (right) islands of Mayotte.*

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

We not only show that using a neural network reduces the constraints on the generation process, but also that the execution time is only dependant on the network architecture, without influence from the dataset generation algorithm.

6. Advantages, limitations, and future works

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 that are not bound by the original constraints of the procedural model. By combining these two methods, we leverage the 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 about the terrain 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 (such as the 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 like vegetation. 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 in domains where real-world data is scarce, such as coral reef islands, allowing synthetic data to be effectively used for training purposes.

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 can break free from the radial symmetry constraint and center positioning, it still relies on the synthetic data's structure and patterns. This can limit 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 tweak parameters (e.g., island size, beach width) 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 customization available to the user.

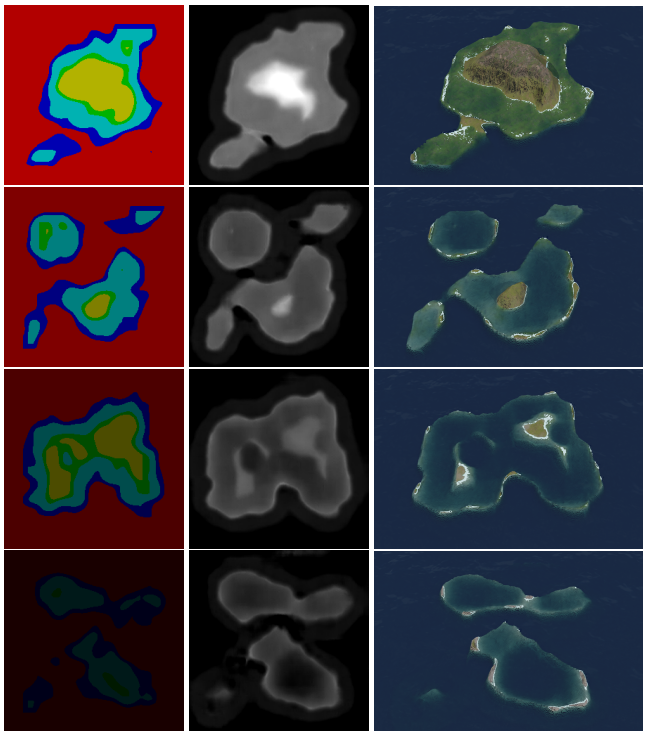

Figure 25: *Starting from random Perlin noise, transformed into a label map, we can generate a large variety of results.*

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.

While this approach brings significant advantages, there are also some limitations to consider. The reliance on a synthetic dataset means that the cGAN inherits some biases and limitations of the original procedural algorithm. This could limit the true diversity of the terrains that the model can generate, as the output is confined by the patterns present in the training data. Additionally, the cGAN model's internal logic lacks transparency, offering limited user control over the generation process once the model has been trained. This contrasts with traditional procedural methods, which typically allow for real-time tweaking of parameters.

7. Conclusion

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 simulation. By applying these methods, we were able to produce realistic terrains based on geological processes, capturing key features of coral reef islands such as 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. With data augmentation techniques, we were able to train the cGAN on a synthetic dataset, generating varied and realistic 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 tweaking parameters like wind strength or island size, could enhance the flexibility of the system.

Finally, further improvements could be made to the synthetic dataset. Incorporating more complex geological processes, such as wave erosion or tidal influences, could lead to even more realistic terrains. Additionally, refining the way islands are blended in multi-island samples, or adding more diverse input conditions (e.g., different geological settings), could help the model generalize better and produce more varied and dynamic landscapes.

One possible future improvement could involve incorporating the wind velocity field into the cGAN training process. While the label map is the only input used in the current implementation, the wind field could be added as an additional condition. This would be especially useful if the initial algorithm were augmented to include wind-driven features, such as cliffs or specific terrain deformations influenced by wind patterns. Adding the wind field as an input could help the cGAN generate more realistic terrains that better reflect the influence of wind on the landscape.

Additionally, further development could explore improving how multiple islands are combined in a single sample. For example, using blending techniques to handle overlapping regions could allow islands to be positioned closer together, enabling the generation of more complex archipelagos without sacrificing the integrity of the height field.

Many other neural networks models could be exploited to increase the possibilities, such as newer variants of cGANs Park et al., 2019, or models with style transfer functionalities Gatys et al., 2015; Zhu et al., 2020 in order to change the overall aspect of a terrain Perche et al., 2023b, 2023a, use text-to-images models Rombach et al., 2021; Radford et al., 2021 to generate height fields from a verbal prompt, or super-resolution models Dong et al., 2014 to increase the definition of details in the final output Guérin, Digne, Galin, and Peytavie, 2016.

References

Beckham, C., & Pal, C. (2017). A step towards procedural terrain generation with gans. http://arxiv.org/abs/1707.03383

Beneš, B., Těšínský, V., Hornyš, J., & Bhatia, S. K. (2006). Hydraulic erosion. *Computer Animation and Virtual Worlds*, *17*, 99–108. https://doi.org/10.1002/cav.77

Cook, M. T., & Agah, A. (2009). A survey of sketch-based 3-d modeling techniques. *Interacting with Computers*, *21*, 201–211. https://doi.org/10.1016/j.intcom.2009.05.004

Cordonnier, G., Braun, J., Cani, M.-P., Benes, B., Galin, É., Peytavie, A., & Guérin, É. (2016). Large scale terrain generation from tectonic uplift and fluvial erosion. *Computer Graphics Forum*, *35*, 165–175. https://doi.org/10.1111/cgf.12820

Cordonnier, G., Cani, M.-P., Beneš, B., Braun, J., & Galin, É. (2017). Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE Transactions on Visualization and Computer Graphics*, *24*. https://doi.org/10.1109/TVCG.2017.2689022

Cordonnier, G., Galin, É., Gain, J., Beneš, B., Guérin, É., Peytavie, A., & Cani, M.-P. (2017). Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics*, *36*. https://doi.org/10.1145/3072959.3073667

Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Image super-resolution using deep convolutional networks. http://arxiv.org/abs/1501.00092 (cit. on p. 13).

Ebert, D. S., Peachey, D., Worley, S., Hart, J. C., Musgrave, K., Perlin, K., & Mark, W. R. (2003). *Texturing and modeling, a procedural approach* (Vol. Third edition). Morgan Kaufmann.

Ecormier-Nocca, P., Cordonnier, G., Carrez, P., Moigne, A. M., Memari, P., Benes, B., & Cani, M. P. (2021). Authoring consistent landscapes with flora and fauna. *ACM Transactions on Graphics*, *40*. https://doi.org/10.1145/3450626.3459952

Fournier, A., Fussell, D., & Carpenter, L. (1982). Computer rendering of stochastic models. *Communications of the ACM*, *25*, 371–384. https://doi.org/10.1145/358523.358553

Gain, J., Marais, P., & Straßer, W. (2009). Terrain sketching. *Proceedings of I3D 2009: The 2009 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, *1*, 31–38. https://doi.org/10.1145/1507149.1507155

Galin, E., Guérin, E., Peytavie, A., Cordonnier, G., Cani, M.-P., Benes, B., & Gain, J. (2019). A review of digital terrain modeling. *Computer Graphics Forum*, *38*, 553–577. https://doi.org/10.1111/cgf.13657

Gasch, C., Chover, M., Remolar, I., & Rebollo, C. (2020). Procedural modelling of terrains with constraints. *Multimedia Tools and Applications*, *79*, 31125–31146. https://doi.org/10.1007/s11042-020-09476-3

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. http://arxiv.org/abs/1508.06576 (cit. on p. 13).

Génevaux, J.-D., Galin, É., Peytavie, A., Guérin, É., Briquet, C., Grosbellet, F., & Beneš, B. (2015). Terrain modelling

from feature primitives. https://doi.org/https://doi.org/10.1111/cgf.12530

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *Advances in Neural Information Processing Systems*, *3*, 139–144. https://doi.org/10.48550/arXiv.1406.2661

Guérin, E., Peytavie, A., Masnou, S., Digne, J., Sauvage, B., Gain, J., & Galin, E. (2022). Gradient terrain authoring. *Computer Graphics Forum*, *41*, 85–95. https://doi.org/10.1111/cgf.14460

Guérin, É., Digne, J., Galin, É., & Peytavie, A. (2016). Sparse representation of terrains for procedural modeling. *Computer Graphics Forum*, *35*, 177–187. https://doi.org/10.1111/cgf.12821 (cit. on p. 13).

Guérin, É., Digne, J., Galin, É., Peytavie, A., Wolf, C., Beneš, B., & Martinez, B. (2017). Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Transactions on Graphics*, *36*. https://doi.org/10.1145/3130800.3130804

Hnaidi, H., Guérin, E., Akkouche, S., Peytavie, A., & Galin, E. (2010). Feature based terrain generation using diffusion equation. *Computer Graphics Forum*, *29*, 2179–2186. https://doi.org/10.1111/j.1467-8659.2010.01806.x

Hopley, D. (2014). *Encyclopedia of modern coral reefs : Structure, form and process*. Springer, Credo Reference. (Cit. on p. 1).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967–5976. https://doi.org/10.1109/CVPR.2017.632

Körner, C., Spehn, E. M., Bugmann, H., & Zurich, E. (2014). *Mountain systems* (tech. rep.). https://www.researchgate.net/publication/238663492 (cit. on p. 2).

Lyons, M. B., Murray, N. J., Kennedy, E. V., Kovacs, E. M., Castro-Sanguino, C., Phinn, S. R., Acevedo, R. B., Alvarez, A. O., Say, C., Tudman, P., Markey, K., Roe, M., Canto, R. F., Fox, H. E., Bambic, B., Lieb, Z., Asner, G. P., Martin, P. M., Knapp, D. E., … Roelfsema, C. M. (2024). New global area estimates for coral reefs from high-resolution mapping. *Cell Reports Sustainability*, *1*, 100015. https://doi.org/10.1016/j.crsus.2024.100015 (cit. on p. 1).

Mei, X., Decaudin, P., & Hu, B. G. (2007). Fast hydraulic erosion simulation and visualization on gpu. *Proceedings - Pacific Conference on Computer Graphics and Applications*, 47–56. https://doi.org/10.1109/PG.2007.27

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. http://arxiv.org/abs/1411.1784

Musgrave, F. K., Kolb, C. E., & Mace, R. S. (1989). The synthesis and rendering of eroded fractal terrains. *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1989*, 41–50. https://doi.org/10.1145/74333.74337

Natali, M., Viola, I., & Patel, D. (2012). Rapid visualization of geological concepts. *Brazilian Symposium of Computer Graphic and Image Processing*, 150–157. https://doi.org/10.1109/SIBGRAPI.2012.29

Neidhold, B., Wacker, M., & Deussen, O. (2005). Interactive physically based fluid and erosion simulation. *Natural Phenomena*, 25–32. http://graphics.uni-konstanz.de/publikationen/Neidhold2005InteractivePhysicallyBased/Neidhold2005InteractivePhysicallyBased.pdf

Olsen, J. (2004). Realtime procedural terrain generation. *Department of Mathematics And Computer Science*, 20. https://pdfs.semanticscholar.org/5961/c577478f21707dad53905362e0ec4e6ec644.pdf%5Cnhttp://www.tsi.enst.fr/~bloch/P6/PRREC/terrain_generation.pdf%5Cnhttp://web.mit.edu/cesium/Public/terrain.pdf

Olsen, L., Samavati, F. F., Sousa, M. C., & Jorge, J. A. (2009). Sketch-based modeling: A survey. *Computers and Graphics (Pergamon)*, *33*, 85–103. https://doi.org/10.1016/j.cag.2008.09.013

Panagiotou, E., & Charou, E. (2020). Procedural 3d terrain generation using generative adversarial networks. http://arxiv.org/abs/2010.06411

Park, J., Redwine, J., Hill, T. D., & Kotun, K. (2019). Water resource and ecotone transformation in coastal ecosystems. *Ecological Modelling*, *405*, 69–85. https://doi.org/10.1016/j.ecolmodel.2019.04.015 (cit. on p. 13).

Perche, S., Peytavie, A., Benes, B., Galin, E., & Guérin, E. (2023a). Styledem: A versatile model for authoring terrains. http://arxiv.org/abs/2304.09626 (cit. on p. 13).

Perche, S., Peytavie, A., Benes, B., Galin, E., & Guérin, E. (2023b). Authoring terrains with spatialised style. *Computer Graphics Forum*, *42*. https://doi.org/10.1111/cgf.14936 (cit. on p. 13).

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH Computer Graphics*, *19*, 287–296. https://doi.org/10.1145/325165.325247

Perlin, K. (2001). Noise hardware. *Real-Time Shading SIGGRAPH Course Notes*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. http://arxiv.org/abs/2103.00020 (cit. on p. 13).

Reinhard, E., Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G., Ebert, D., Lewis, J., Perlin, K., & Zwicker, M. (2010). State of the art in procedural noise functions. In H. Hauser & E. Reinhard (Eds.), *Eurographics 2010 - state of the art reports*. The Eurographics Association. https://doi.org/https://doi.org/10.2312/egst.20101059

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. http://arxiv.org/abs/2112.10752 (cit. on p. 13).

Schott, H., Paris, A., Fournier, L., Guérin, E., & Galin, E. (2023). Large-scale terrain authoring through interactive erosion simulation. *ACM Transactions on Graphics*, *42*, 1–15. https://doi.org/10.1145/3592787

Sisodia, Y. (2022). Gan-generated terrain for game assets. *Indian Journal of Artificial Intelligence and Neural Networking*, *2*, 1–3. https://doi.org/10.54105/ijainn.F1060.102622

Smelik, R. M., Kraker, K. J. D., Groenewegen, S. A., Tutenel, T., & Bidarra, R. (2009). A survey of procedural methods for terrain modelling. *Proceedings of the CASA workshop on 3D advanced media in gaming and simulation (3AMIGAS)*.

Spick, R., & Walker, J. A. (2019). Realistic and textured terrain generation using gans. *Proceedings - CVMP 2019: 16th ACM SIGGRAPH European Conference on Visual Media Production*. https://doi.org/10.1145/3359998.3369407

Talgorn, F. X., & Belhadj, F. (2018). Real-time sketch-based terrain generation. *ACM International Conference Proceeding Series*, 13–18. https://doi.org/10.1145/3208159.3208184

Tasse, F. P., Emilien, A., Cani, M.-P., Hahmann, S., & Bernhardt, A. (2014). First person sketch-based terrain editing. In *Graphics interface*. https://hal.inria.fr/hal-00976689/file/FirstPersonSketchBasedTerrainEditing_GI2014.pdf

Voulgaris, G., Mademlis, I., & Pitas, I. (2021). Procedural terrain generation using generative adversarial networks. *European Signal Processing Conference*, *2021-August*, 686–690. https://doi.org/10.23919/EUSIPCO54536.2021.9616151

Wulff-Jensen, A., Rant, N. N., Møller, T. N., & Billeskov, J. A. (2018, January). Deep convolutional generative adversarial network for procedural 3d landscape generation based on dem. In *Interactivity, game creation, design, learning, and innovation* (pp. 85–94). Springer. https://doi.org/10.1007/978-3-319-76908-0_9

Zhu, D., Cheng, X., Zhang, F., Yao, X., Gao, Y., & Liu, Y. (2020). Spatial interpolation using conditional generative adversarial neural networks. *International Journal of Geographical Information Science*, *34*, 735–758. https://doi.org/10.1080/13658816.2019.1599122 (cit. on p. 13).